Scan Tools
(SGI™ 3000 Family)

SGI Confidential & Proprietary Information - For Internal Recipients Only

Document Number 108-0263-002

CONTRIBUTORS
Written by Darrin Goss
Edited by Allison Gosbin
Production by Rhonda Kunsman
Engineering contributions by Karen Beighley, Kurt Kermes, and Jim Young

Record of Revision

Version	Description
001	August 2000 Original printing.
002	April 2001 This revision corresponds to the Internal Support Tools CD 2.4. It describes the new `brick_scan` tool and provides additional information about the scan hardware and the `scantool` application.

Contents

Figures

Tables

Chapter 1

Introduction

The Institute of Electrical and Electronics Engineers (IEEE) standard 1149.1, *Standard Test Access Port and Boundary-Scan Architecture*, defines a set of test logic and related features that enable connections within and between ASICs to be tested. These features are collectively called boundary scan, scan, or JTAG.

The test logic includes a test access port (TAP), a TAP controller, an instruction register, and a group of test data registers. (Refer to Figure 1-1.)

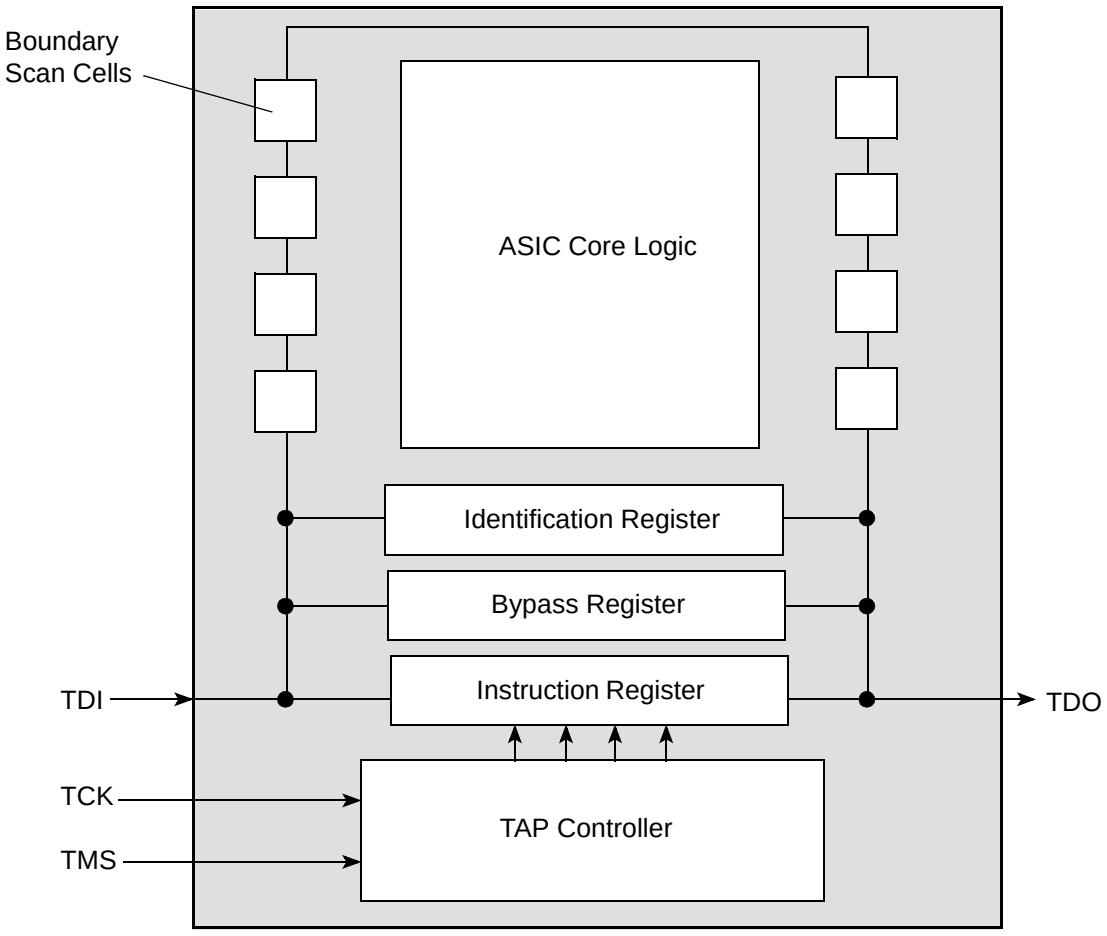

Figure 1-1 Boundary Scan Test Logic

Table 1-1 describes the boundary scan components.

Table 1-1 Boundary Scan Components

Component	Description
Test access port (TAP)	Provides access to the test logic and consist of four signal pins: test data input (TDI), test data output (TDO), test clock (TCK), and test mode select (TMS)
TAP controller	Controls the instruction register, bypass register, identification register, and boundary scan register
TMS signal	Controls test operations
TCK signal	Inputs a signal that clocks the TAP controller
TDI signal	Inputs data serially
TDO signal	Outputs data serially
Instruction register	Receives, holds, and decodes boundary scan instructions
Bypass register	Connects the TDI and TDO signals during a BYPASS instruction, which causes the data to go through the bypass register instead of the boundary scan cells (This register is used to verify integrity of the scan chain and to perform board-level testing.)
Identification (ID) register	Holds 32 bits of information that specify the manufacturer of the chip (or ASIC), the part number of the chip, and the revision of the chip ID registers are optional components; not all chips contain ID registers
Boundary scan cells	Components of the boundary scan register that are connected in a shift-register path around the boundary of the chip; these components are located between the I/O pins and the core logic, which enables the boundary scan operation to control and monitor the I/O pins
ASIC core logic	Contains the functional components of the ASIC

The following components contain ASICs that include the scan/JTAG test logic as defined by the IEEE 1149.1 standard:

- C bricks
- I bricks
- P bricks
- R bricks
- X bricks

The SGI 3000 family diagnostic suite includes two scan tests that use the scan/JTAG hardware to test connections within boards and between boards.

1.1 Scan Hardware

The following components contain hardware that is used to perform scan testing: the L1, L2, and L3 system controllers and the C, R, I, P, and X bricks.

1.1.1 System without an R brick or L2 System Controller

In a system without an R brick or L2 system controller (refer to Figure 1-2):

- The L3 system uses a serial or USB connection to communicate with the L1 system controller in a C brick.
- The L1 system controller in the C brick uses an RS-422 connection (through a portion of an Xtown2 or NUMAlink 3 cable) to communicate with the L1 system controllers in other I, P, X, and C bricks.
- The L1 system controller in each C, R, I, P, and X brick acts as a JTAG bus master and passes JTAG commands / data to the scan interface chip (SIC) in each brick.
- The SIC in each brick passes the JTAG commands / data to the individual scan chains (up to four) in the brick. (Table 1-2 on page 1-7 shows the components on each scan chain.)
- The scan chains pass the commands / data to the individual ASICs in the bricks.

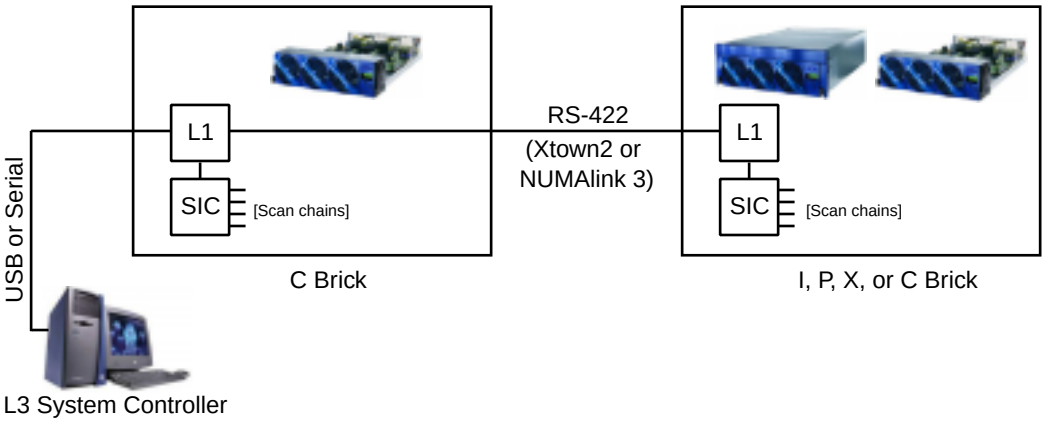

Figure 1-2 Scan Hardware Components (System without an R Brick or L2 System Controller)

1.1.2 System with an R Brick but without an L2 System Controller

In a system that has an R brick but does not have an L2 system controller (refer to Figure 1-3.):

- The L3 system controller uses a serial or USB connection to communicate with the L1 system controller in a C brick or a USB connection to communicate with the L1 system controller in an R brick.
  - If the L3 system controller is connected to a C brick, the L1 system controller in the C brick uses an RS-422 connection (through a portion of an Xtown2 or NUMAlink 3 cable) to communicate with the L1 system controllers in other I, P, X, and C bricks.
  - If the L3 system controller is connected to an R brick, the L1 system controller in the R brick uses an RS-422 connection (through a portion of a NUMAlink 3 cable) to communicate with the L1 system controller in another R brick or a C brick.
- The L1 system controller in each C, R, I, P, and X brick acts as a JTAG bus master and passes JTAG commands / data to the SIC in each brick.
- The SIC in each brick passes the JTAG commands / data to the individual scan chains (up to four) in the brick. (Table 1-2 on page 1-7 shows the components on each scan chain.)
- The scan chains pass the commands / data to the individual ASICs in the bricks.

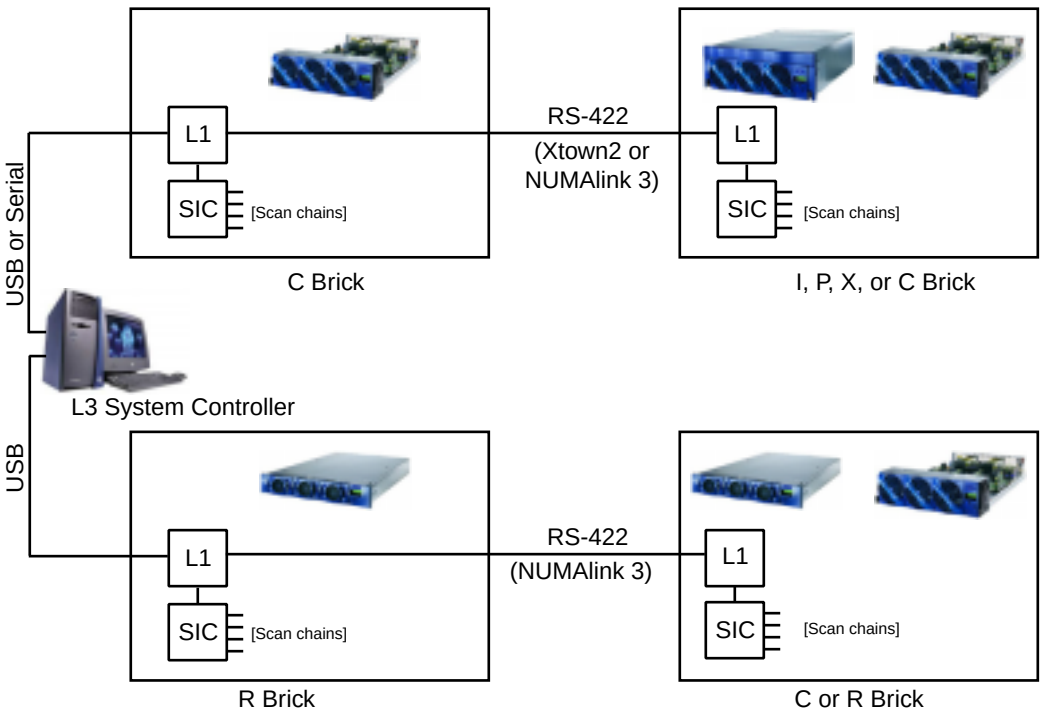

Figure 1-3 Scan Hardware Components (System without an L2 System Controller)

1.1.3 System with an L2 System Controller

In a system with an L2 system controller (refer to Figure 1-4):

- The L3 system controller uses an Ethernet connection to communicate with the L2 system controller.

  Note: If necessary, the L3 system controller can bypass the L2 system controller and use a serial or USB connection to communicate directly with the C and R bricks (as in a system without an L2 system controller).

- The L2 system controller uses USB connections to communicate with the L1 system controllers in the C bricks and R bricks.
  - If the L2 system controller is connected to a C brick, the L1 system controller in the C brick uses an RS-422 connection (through a portion of an Xtown2 or NUMAlink 3 cable) to communicate with the L1 system controllers in other I, P, X, and C bricks.
  - If the L2 system controller is connected to an R brick, the L1 system controller in the R brick uses an RS-422 connection (through a portion of a NUMAlink 3 cable) to communicate with the L1 system controller in another R brick or a C brick.
- The L1 system controller in each C, R, I, P, and X brick acts as a JTAG bus master and passes JTAG commands / data to the SIC in each brick.
- The SIC in each brick passes the JTAG commands / data to the individual scan chains (up to four) in the brick. (Table 1-2 on page 1-7 shows the components on each scan chain.)
- The scan chains pass the commands / data to the individual ASICs in the bricks.

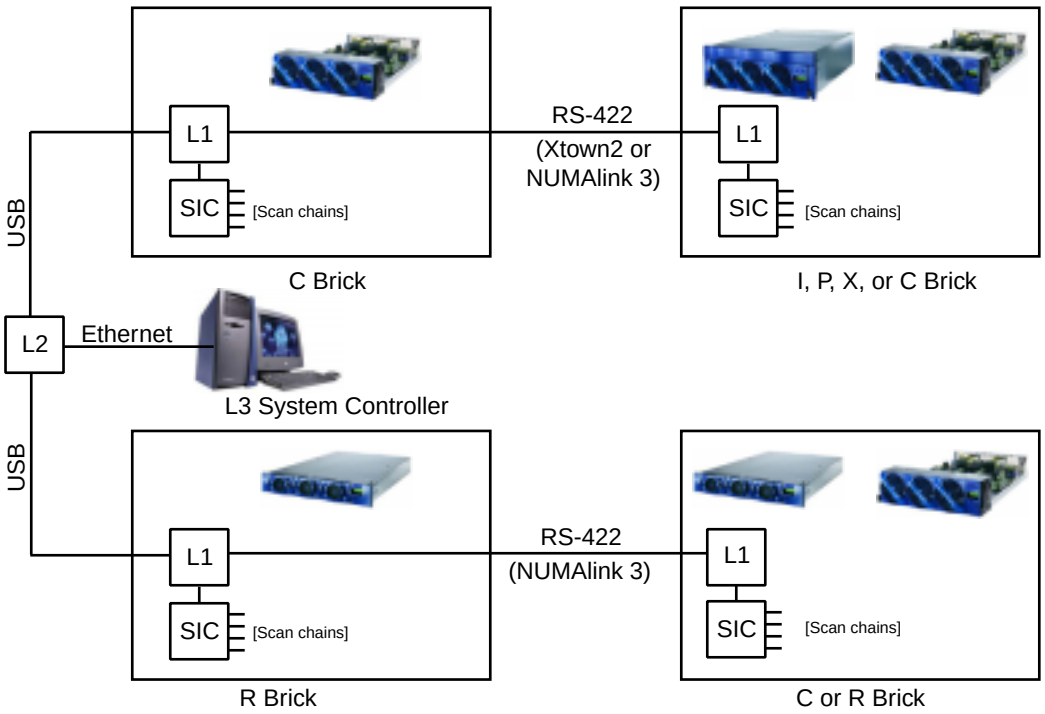

Figure 1-4 Scan Hardware Components (System with an L2 System Controller)

1.1.4 Scan Chains

Table 1-2 lists the components in each scan chain for the various bricks.

Table 1-2 Scan Chain Components

Brick	Chain 0	Chain 1	Chain 2	Chain 3
C (IP35)	PIMM0	PIMM1	Bedrock	DIMM0 DIMM1 DIMM2 DIMM3 DIMM4 DIMM5 DIMM6 DIMM7
C (IP37, SIC 0)	Synergy 0	Synergy 1	Bedrock	DIMM0 DIMM1 DIMM2 DIMM3 DIMM4 DIMM5 DIMM6 DIMM7
C (IP37, SIC 1)	CPU 0	CPU 1	CPU 2	CPU 3
I	XBG0	IOC3	PCI11 PCI12 PCI13 PCI21 PCI22	Not used
P	Not used	PCI41 PCI42 PCI51 PCI52 PCI61 PCI62	PCI11 PCI12 PCI21 PCI22 PCI31 PCI32	XBG2 XBG0 XBG1
R	RTR0	Not used	Not used	Not used
X	XBG0	PORT8 (XIO slot) PORT9 (XIO slot)	PORTC (XIO slot) PORTD (XIO slot)	Not used

1.2 Scan Software

The diagnostic suite includes two scan tests:

- Boundary scan interconnect test

  The boundary scan interconnect test manipulates the boundary scan registers in various chips throughout the system to verify system interconnections. It applies a set of test vectors to the system to detect physical connection defects (for example, stuck-at faults and shorts).

- Scan-based link-level protocol interconnect test

  The scan-based link-level protocol interconnect test uses additional scan-accessible registers to exercise brick-to-brick connections at-speed. It tests the connectors on C, R, I, P, and X bricks and the cables between these bricks.

  Note: The scan-based link-level protocol test runs only in components that use LLP (router, Bedrock, and Xbridge).

The diagnostic suite also includes a scripting tool, `scantool`. `scantool` enables you to control scan operations from Tcl/Tk scripts.

The scan software resides on the L3 system controller. To use the scan software, you must have an L3 system controller, a laptop loaded with the L3 software, or a remote support connection to a system.

1.3 Standard Test Process

Use the following process to test a system with the scan tools:

1. Install the cables (normal or loopback) for the configuration that you want to test.
2. Start the L2 system controller software, and power up the hardware.
3. Log into the L3 system controller as `sgidiag`.
4. Run the boundary scan interconnect test to verify the scan hardware and verify connections within bricks.
5. Run the scan-based link-level protocol interconnect test to verify connections between bricks.

Chapter 2

Boundary Scan Interconnect Test

This chapter describes the boundary scan interconnect test.

2.1 About the Boundary Scan Interconnect Test

The boundary scan interconnect test verifies connections within C, R, I, X, and P bricks. Use it to verify proper operation of any brick that you add to a system.

The boundary scan interconnect test uses two applications:

- A test application (`scantest`) that manipulates the boundary scan registers in various chips of a brick to verify connections within the brick. `scantest` applies a set of test vectors to a brick to detect physical connection defects (stuck-at faults and shorts).
- A graphical user interface application (`brick_scan`) that enables you to set test parameters and view test output.

When the boundary scan interconnect test detects an error, it identifies the chips and pins on the failing net and also displays the actual and expected data.

2.2 Test Algorithm

The boundary scan interconnect test uses the following test algorithm:

1. Load the chain configuration and reference data.
2. Load the reference data for each board definition.
3. Initialize JTAG communications to the selected scan controller(s).
4. Initialize the chain TAPs and configure the SIC(s).

5. Perform comprehensive scan hardware integrity tests to verify that the scan hardware is functional enough to perform boundary scan interconnect testing:
   - Perform the instruction register length test.
   - Perform the instruction register path test.
   - Perform the bypass register length test.
   - Perform the bypass register path test.
   - Perform the chain-level boundary register length test.
   - Perform the chain-level boundary register path test.
6. Perform the board-level boundary scan register interconnect test.
7. Print a program termination status message.

Warning: **Each step must pass before the next step can be run. If the integrity tests in Step 5 do not pass, the interconnect test does not run because the results would be invalid. Do not manually run the interconnect test until the integrity tests pass.**

Figure 2-1 shows the output from each step of this algorithm.

```
   Boundary-Scan Test Software (scantest V3.11) Mon Apr  2 11:37:41 2001

1  Loading chain configuration and reference data
      File: /stand/sysco/cfg/scan/c001dimm04.cfg

2  Loading reference data for each board definition
      Path: /stand/sysco/data/scan/cmp

3  Initializing JTAG communications to scan controller(s)
      No hardware; running in software simulation/emulation mode

4  Initializing chain TAPs and configuring SICs:
   001c01 with SIC at address 0x00 (CER=0x0C)

   Testing Integrity of Boundary Scan Instruction Registers
   Performing IR length test on active UUTs:
   001c01 SIC 0x0 ... passed (chain length=149)

   Performing IR path test on active UUTs:
   001c01 SIC 0x0 ... passed

   Testing Integrity of Boundary Scan Bypass Registers
   Performing BYPASS length test on selected UUTs:
5  001c01 SIC 0x0 ... passed (chain length=19)

   Performing BYPASS path test on selected UUTs:
   001c01 SIC 0x0 ... passed

   Testing Integrity of Boundary Scan Data Registers
   Performing chain-level BOUNDARY length test on selected UUTs:
   001c01 SIC 0x0 ... passed (chain length=1706)

   Performing chain-level BOUNDARY path test on selected UUTs:
   001c01 SIC 0x0 ... passed

6  Performing system-level interconnect test on given configuration:
   (22 of 22 patterns will be applied)
   ++++++++++++++++++++++
   Errors were detected during interconnect test.

   Failure information for BSR interconnect test:

   Module and raw position : 000-000022
   Expected data           : 10101010100101010101
   Actual data             : 00000000000000000000
   Difference data         : ^ ^ ^ ^ ^  ^ ^ ^ ^ ^ ^
   Driving node index      : 00000000000000001100
   Logical description     : [001c01:NODE:BR:MB_DATA_43_BUS]
   001c01:DIMM0:U2:A4
   Physical description    : [001c01:NODE:J0J2:AD35] 001c01:DIMM0:A9B3:1

   Number of failing signals displayed from test: 1

7  Normal Program Termination
```

Figure 2-1 Boundary Scan Interconnect Test Algorithm Components

2.3 Graphical User Interface

The `brick_scan` graphical user interface enables you to set test parameters and view test output. (Refer to Figure 2-2.)

The `Brick Selection` parameters specify the brick location (rack and slot) and type of brick to test.

The following bricks are currently supported: IP35 C bricks, IP37 C bricks, I bricks, R bricks, P bricks, and X bricks.

If you select an IP35 C brick, you can also select which PIMMs to test (PIMM0 and/or PIMM1).

If you select an IP37 C brick, you can also select the CPU type (2 MB or 4 MB) and the CPUs to test (CPU 0, 1, 2, and/or 3).

The `Test Selection` parameters specify the tests to run.

The `Loop` parameter specifies the number of times to run the tests. (Default = 1 time)

The `Scan Communications Method` parameters specify how the L3 is connected to the system.

Use the `L2 Emulator` option if the L3 is connected directly to a brick through a serial or USB connection.

Use the `Ethernet L2` option if the L3 is connected to an L2 through an Ethernet connection. Specify the IP address of the L2 in the `L2 Address` field.

The command buttons display help information, run the selected tests, and exit the graphical user interface.

The test output area provides detailed information about the activities that the boundary scan interconnect test performed and error messages for any failures that it detected.

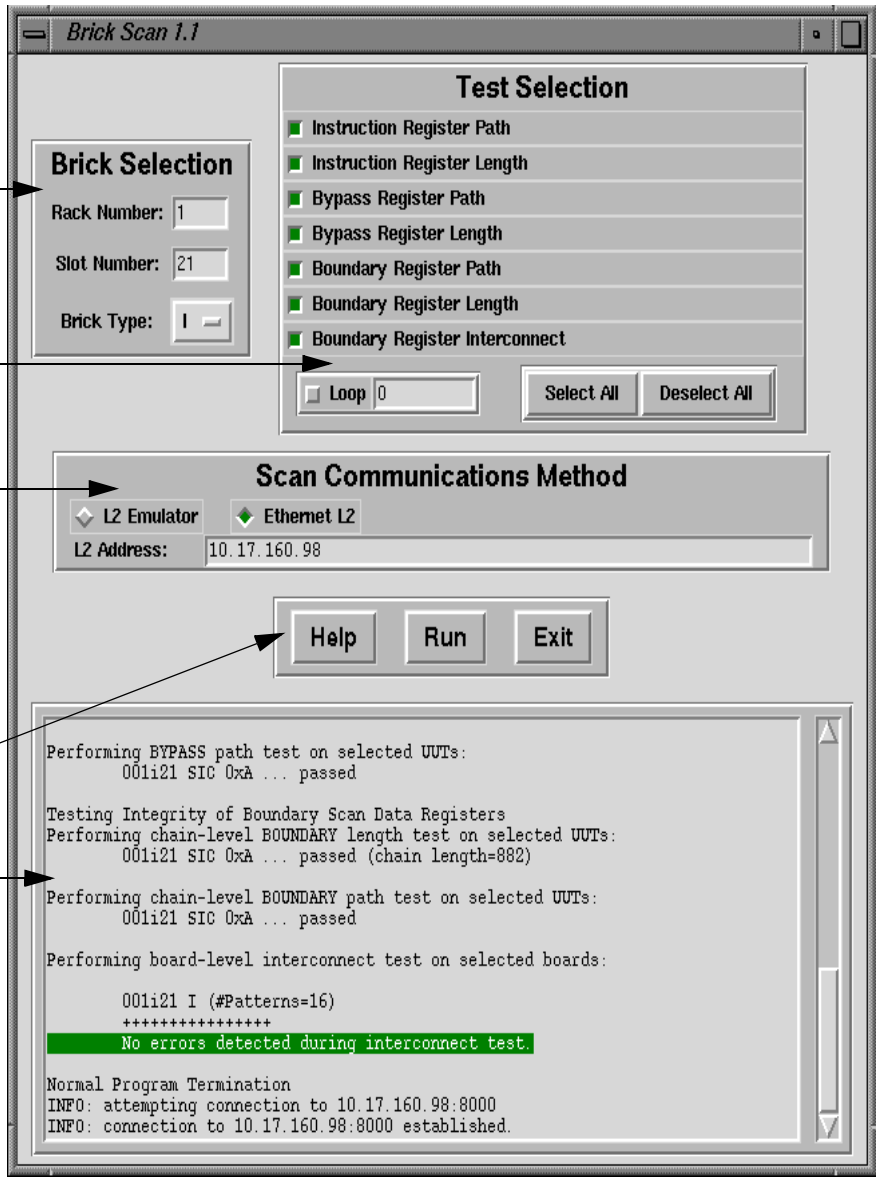

Figure 2-2 Boundary Scan Test Graphical User Interface

2.4 Running the Boundary Scan Interconnect Test from brick_scan

Warning: **Ensure that the system is offline before you run the boundary scan interconnect test. If the operating system is running when you run this test, the operating system will crash and customer data will be lost.**

Perform the following procedure to run the boundary scan interconnect test on a single brick from the `brick_scan` graphical user interface:

1. Start the L2 system controller software and power up the system.
2. Log into the L3 system controller as `sgidiag` (default password: `sgi!diag`).
3. Enter **`brick_scan`** to start the `brick_scan` application.

   The `brick_scan` application displays a warning message. (Refer to Figure 2-3.) Review this information and ensure that the system is ready for testing; then, click `Continue`.

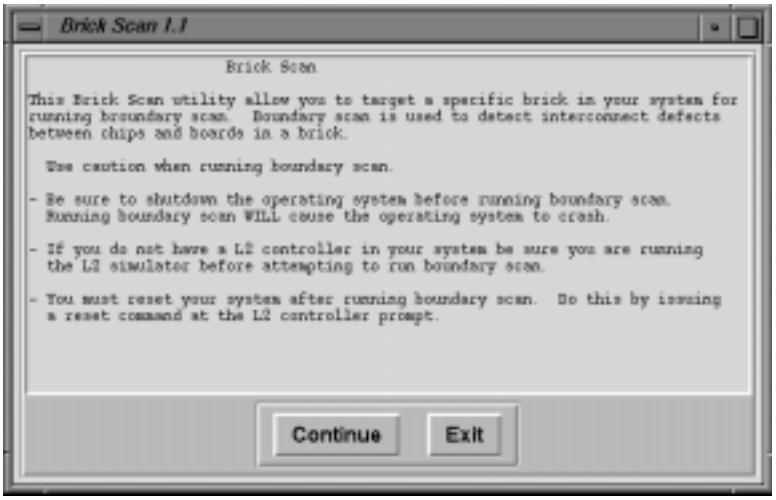

Figure 2-3 Boundary Scan Interconnect Test Warning Message

4. Select the brick that you want to test (refer to Figure 2-4):
   - Enter the rack number in the `Rack Number` field.
   - Enter the slot number in the `Slot Number` field.

     If you do not know the rack and slot locations of the brick that you want to test, use the L2 `cfg` command to determine the bricks that are available.
   - Select the brick type from the `Brick Type` menu.

     If you are testing an IP35 C brick, you can select the PIMMs to test (none, PIMM0, and / or PIMM1).

     If you are testing an IP37 C brick, you can select the CPU type (2 MB or 4 MB) and the CPUs to test (CPU 0, 1, 2, and / or 3).

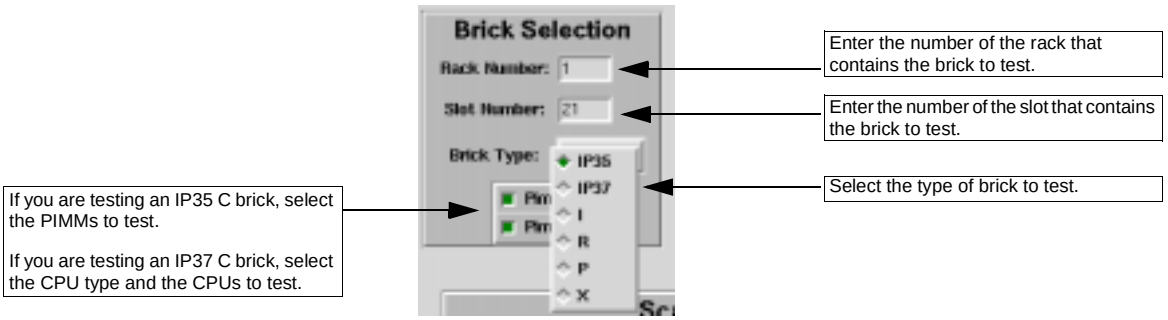

Figure 2-4 Setting the Brick Selection Parameters

5. Select the test sections that you want to run. (Refer to Figure 2-5.) SGI recommends that you run all test sections.

   Warning: **Each test section must pass before the next section can be run. If the integrity tests do not pass, the interconnect test does not run because the results would be invalid. Do not manually run the interconnect test until the integrity tests pass.**

   If you wish to run the test multiple times, change the `Loop` parameter, and click on the box next to the `Loop` parameter. The test defaults to one pass.

   Note: If you set the `Loop` parameter but do not click the box next to it, the test performs only one pass. If you want to run multiple passes, you must set the `Loop` parameter and click on the box next to it.

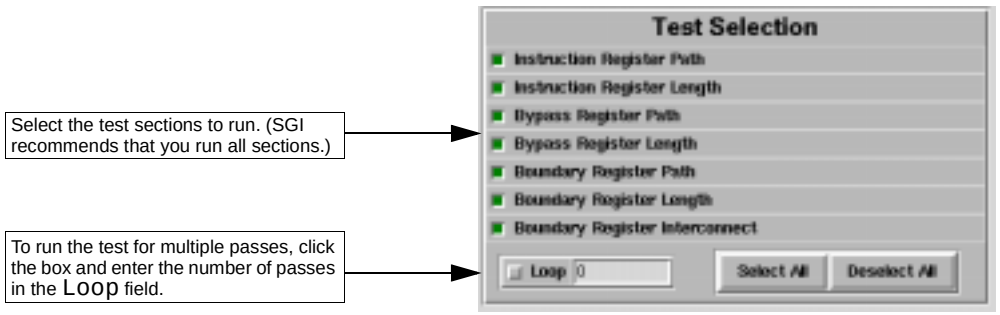

Figure 2-5 Setting the Test Selection Parameters

6. Select the `Scan Communications Method`. (Refer to Figure 2-6.)

   - If the L3 system controller is connected a brick through a serial or USB connection, select the `L2 emulator` option.

     If you select the `L2 emulator` option, the L2 emulator software must be running before you run the boundary scan interconnect test.

   - If the L3 system controller is connected to an L2 system controller via an Ethernet connection, select the `Ethernet L2` option.

     The `brick_scan` application defaults to the first L2 system controller that it finds. If the `L2 address` field does not contain the IP address of the L2 system controller that you want to use, use the L2 `l2find` command to determine the correct IP address and enter the address in the field.

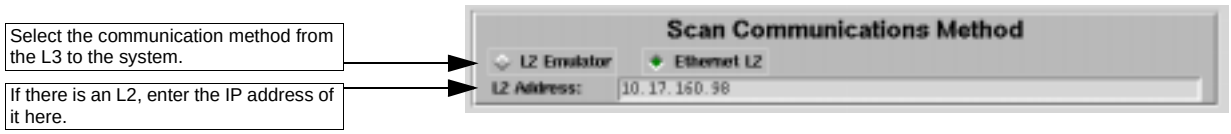

Figure 2-6 Setting the Scan Communications Method Parameters

7. Click on the Run button. (Refer to Figure 2-7.)

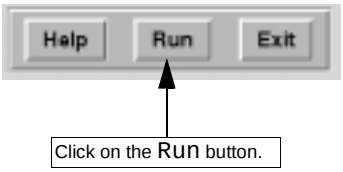

Figure 2-7 Run Button

8. Interpret the output. (Refer to Section 2.6, “Test Output.”)

   Note: The boundary scan interconnect test saves the test output in the `scantest.dmp` file in the current directory; however, it overwrites this file each time you run the test. If you want to save the output so you can analyze it later, copy the contents of the file to a new file before you run the boundary scan interconnect test again.

9. When you are done testing, enter the L2 `reset` command to reset the system.

2.5 Running the Boundary Scan Interconnect Test from scantest

Running the boundary scan interconnect test with the `scantest` command enables you to manually control the settings that the test uses. It also enables you to run the boundary scan interconnect test on multiple bricks.

Warning: **Ensure that the system is offline before you run the boundary scan interconnect test. If the operating system is running when you run this test, the operating system will crash and customer data will be lost.**

Perform the following procedure to run the boundary scan interconnect test on one or more bricks with the `scantest` command:

1. Start the L2 system controller software and power up the system.
2. Log into the L3 system controller as `sgidiag` (default password: `sgi!diag`).
3. Enter **`scantest -f /stand/sysco/cfg/scan/<file>`** to run the boundary scan interconnect test. (Table 2-1 describes the command line options available for `scantest`.)

   Note: The boundary scan interconnect test saves the test output in the `scantest.dmp` file in the current directory; however, it overwrites this file each time you run the test. If you want to save the output so you can analyze it later, copy the contents of the file to a new file before you run the boundary scan interconnect test again.
4. Interpret the output. (Refer to Section 2.6, “Test Output.”)
5. When you are done testing, enter the L2 `reset` command to reset the system.

Table 2-1 scantest Command Line Options

Option	Description
`-h`	Displays the available command line options and then exits.

`-f <file>`	Specifies a configuration file to use. The configuration file defines the hardware to test. (Default: `slit.cfg`)
	Use the following configuration files to test all bricks in a system:
	`sys8p.cfg` = system with 8 processors
	`sys16p.cfg` = system with 16 processors
	`sys32p.cfg` = system with 32 processors
	`sys64p.cfg` = system with 64 processors
	`sys128p.cfg` = system with 128 processors
	For example, to test all bricks in a 16-processor system, enter the following command:
	`scantest -f /stand/sysco/cfg/scan/sys16p.cfg`
	Use the following configuration files to test single bricks in a system:
	`c001.cfg` = IP35 C brick
	`i002.cfg` = I brick
	`p001.cfg` = P brick
	`r001.cfg` = R brick
	`x001.cfg` = X brick
	`ip37_002cF.cfg` = IP37 C brick with 2-MB CPUs (CPU 0, 1, 2, and 3)
	`ip37_002CF.cfg` = IP37 C brick with 4-MB CPUs (CPU 0, 1, 2, and 3)
	For example, to test the R brick that is connected to the L2 system controller:
	`scantest -f /stand/sysco/cfg/scan/r001.cfg`
	Other configuration files are available in the `/stand/sysco/cfg/scan/` directory. Refer to the `/stand/sysco/cfg/scan/README.cfg` file for more information.
	The `ID` field in each configuration file describes the brick to test. By default, the `ID` field is set to zero (0), which selects the local brick. Use the following formula to select a different brick:
	ID = (rack * 64) + slot
	For example, to test an I/O brick at location 002i10 (rack =2, slot = 10), set the `ID` field to (2 * 64) + 10, which equals 138.
	If you want to modify a configuration file, copy it to a directory for which you have write permission. If you modify the ID field, you must set the first number of the chain definition to the same value (for example, `START CHAIN 138`).

`-ID`	Overrides the `ID` field listed in a configuration file. This enables you select a different brick for testing without modifying the configuration file. Use the following formula to select a different brick: ID = (rack * 64) + slot For example, to test a P brick in slot 7 of rack 2, enter the following command: `scantest -f /stand/sysco/cfg/scan/p001.cfg -ID 135`
`-d <file>`	Specifies the output file that receives output from the test. (Default: `scantest.dmp`)

Table 2-1 (continued) scantest Command Line Options

Option	Description
`-e <num>`	Defines the maximum number of errors that the test displays. (Valid values are between 0 [display only pass/fail information] and 10,000; default: 200)
`-p <num>`	Specifies the number of test passes that `scantest` should perform. (Valid values are between 1 and 10,000; default: 1)
`-t <test>`	Specifies the tests to run. Valid tests: irp = instruction register path test irl = instruction register length test ir = instruction register path test and instruction register length test prp = bypass register path test prl = bypass register length test pr = bypass register path test and bypass length test brp = boundary register path test brl = boundary register length test br = boundary register path test and boundary register length test int = boundary register interconnect test id = read and display device ID registers all = all tests Use + to select multiple tests (for example: ir+br+int). The order that the tests are specified in is irrelevant. Certain tests depend on other tests completing successfully, so selecting one test may cause other tests to be automatically selected. (Default: ir+pr+br+int)
`-m <mode>`	Specifies the mode for register tests. Valid modes: normal = normal lengths and minimal pattern(s) elen = extended lengths epat = extended pattern set If normal is specified, then no other option is valid. (Default: normal)
`-l <level>`	Specifies the level of the register tests. Valid modes: chn = chain level register test brd = board level register test chip = chip level register test (Default: chn)

Table 2-1 (continued) scantest Command Line Options

Option	Description
`-C`	Selects an individual Scan Interface Chip (SIC) to run the test. (The test will not run subordinate chains / hardware.) When you use this option: You can specify the irl, irp, ir, prl, prp, or pr tests with the `-t` option. The `-l` option is ignored. The `-m` option is used for lengths and patterns.
`-s <mode>`	Selects a stepping mode. Valid modes: no = no test stepping; run continuous all = step through each pattern; prompt user err = step through failing pattern; prompt user (Default: no)
`-v`	Selects verbose mode, which writes additional messages to the output.

2.6 Test Output

The boundary scan interconnect test returns output in the following ways:

- `scantest` returns output to the shell where you execute it.
- `brick_scan` displays test output in the lower portion of the interface. (Refer to Figure 2-8.)

  The test output area provides detailed information about the activities that the test performed and errors messages for any failures that it detected. Failure and error messages are highlighted in red, and pass messages are highlighted in green.

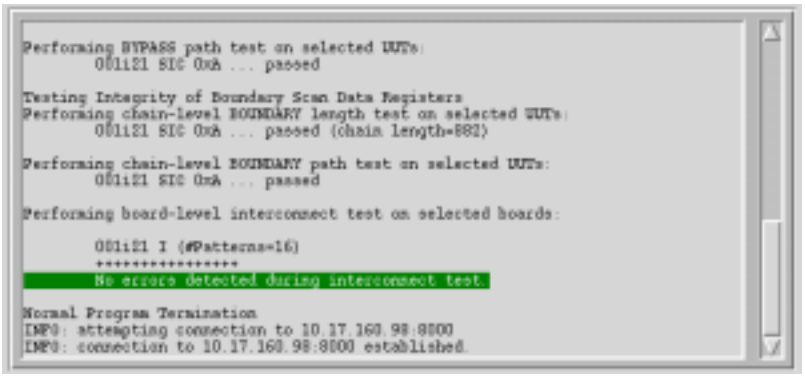

Figure 2-8 Boundary Scan Interconnect Test Output (in brick_scan Interface)

The boundary scan interconnect test returns the following error messages:

```
ERROR - Unable to open output file for writing.
      File: scantest.dmp
      Writing output data to screen instead of file
```

This message appears when you do not have permission to create the `scantest.dmp` output file. The boundary scan interconnect test continues to run, but it does not send output to a file. To save test output to a file, run the test from a directory where you have write permission.

`Unable to run interconnect test because of register test error(s).`

This message appears when one of the boundary scan validation tests fails. When this happens, the interconnect test does not run. Review the output that precedes this message to determine the failing register:

- If the failing register is an instruction register, verify that the brick is powered up. If the brick is powered up, there is a failure in the boundary scan chain. If the brick is not powered up, power it up and rerun the test.
- If the failing register is not an instruction register, there is a failure in the boundary scan chain.

```
ERROR - scantest V3.11 (snl3sc.c @ 280, code=0)
  Unable to read scan response packet from controller.
  Unknown Module 0, message = -1
  Confirm configuration of brick specified with ID=483 (rack=7, slot=35)
```

This message typically appears when you specify incorrect rack and slot numbers for the brick to test. Verify the rack and slot number of the brick you want to test and rerun the test.

This message also appears if the brick is completely powered off (the L1 system controller is not running). Power on the brick and rerun the test.

`Errors were detected during interconnect test.`

This message appears when the boundary scan interconnect test detects a failing component in the brick. Review the output following this message to determine the failing component.

After displaying this message, the test continues; when the testing is completed, the test generates a list of all chain positions that had failing data and identifies the failing chips and pins by physical and logical locations. The detailed failure information uses the following format:

```
BOUNDARY REGISTER INTERCONNECT FAILURE
--------------------------------------

Chain and raw position  : 000-000601
Expected data           : 0110011010010101
Actual data             : 0000000000000100
Difference data         :  ^^  ^^ ^  ^   ^
Driving node index      : 1111111111110011
Logical description     : [101i01:I:XBG0:PF_PCI_SERR_N]
101i01:I:IOC3:PCI_SERR_L
Physical description    : [101i01:I:H7F9:AD21] 101i01:I:J5D4:N2
```

The failure information contains the following data:

- `Chain and raw position`: indicate the location where the test detected a miscompare. The format of this value is <chain number>-<bit position>. (For example, `000-000601` indicates that the test detected the miscompare on chain number 0 at bit position 601.)
- `Expected data`: indicates the signature that the test expects to receive for the data patterns that it is using.
- `Actual data`: indicates the actual signature that was sensed for the data patterns.
- `Difference data`: contains a `^` symbol for each pattern in the signature where the actual data did not match the expected data.

Note: The number of data patterns that the test uses depends on the number of nets being tested. Two types of patterns exist: patterns to detect *stuck-at* failures and patterns to detect *short* failures. For the `Expected data`, `Actual data`, and `Difference data` values, each 0/1 value represents a pattern, with the first pattern (pattern 0) starting at the right and the pattern numbers incrementing by 1 as they go to the left (refer to Figure 2-9).

```
Expected data          : 0110011010010101
Actual data            : 0000000000000100
Difference data        :  ^^  ^^ ^  ^   ^
                                    | | |
                                    | | pattern 0
                                    | pattern 2
                                    pattern 4
```

Figure 2-9 Data Pattern Indicator

- `Driving node index`: indicates which node on the net is *driving* the transfer for each test pattern. (The driving node sends the data pattern to the receiving node.)

  The node index value is a single digit that specifies which node is the driving node for the data pattern. The digit starts at 0 for the first node on the left side of the `Logical description` and `Physical description` lists and then increments by 1 for each successive node. The node index value displays an x if a net is not valid for a pattern (for example, no driver is available).

  A net may have more than one driver if bidirectional signals are available on the net. If possible, *stuck-at* patterns are transferred from all potential drivers; *short* patterns are applied using a single driver.

- `Logical description`: provides a logical description of the nodes on the failing net. The format of each node is <chain and brick>:<board>:<chip>:<pin>. (Refer to `Figure 2-10`.) The boundary scan interconnect test displays the node that detected the miscompare in square brackets `[ ]`.

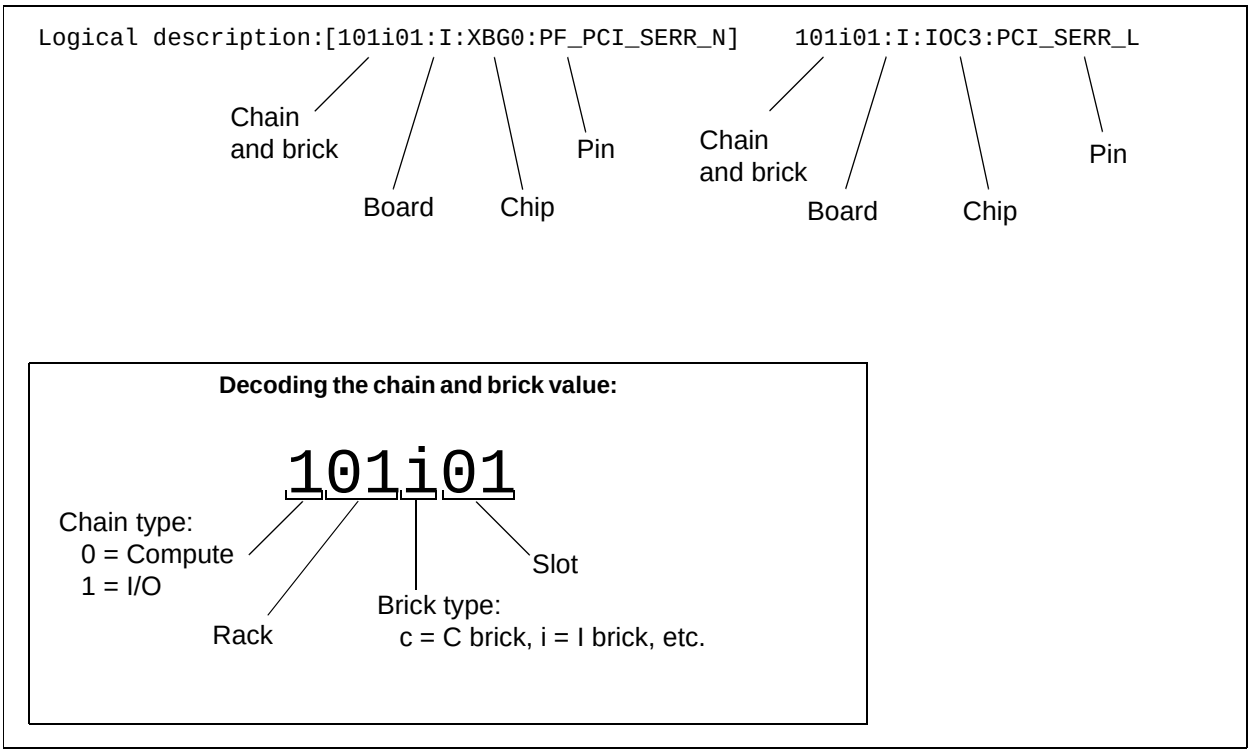

Figure 2-10 Logical Description Format

- `Physical description`: provides a physical description of the nodes on the failing net. The format of each node is <chain and brick>:<board>:<chip>:<pin>. `scantest` displays the node that detected the miscompare in square brackets `[ ]`.

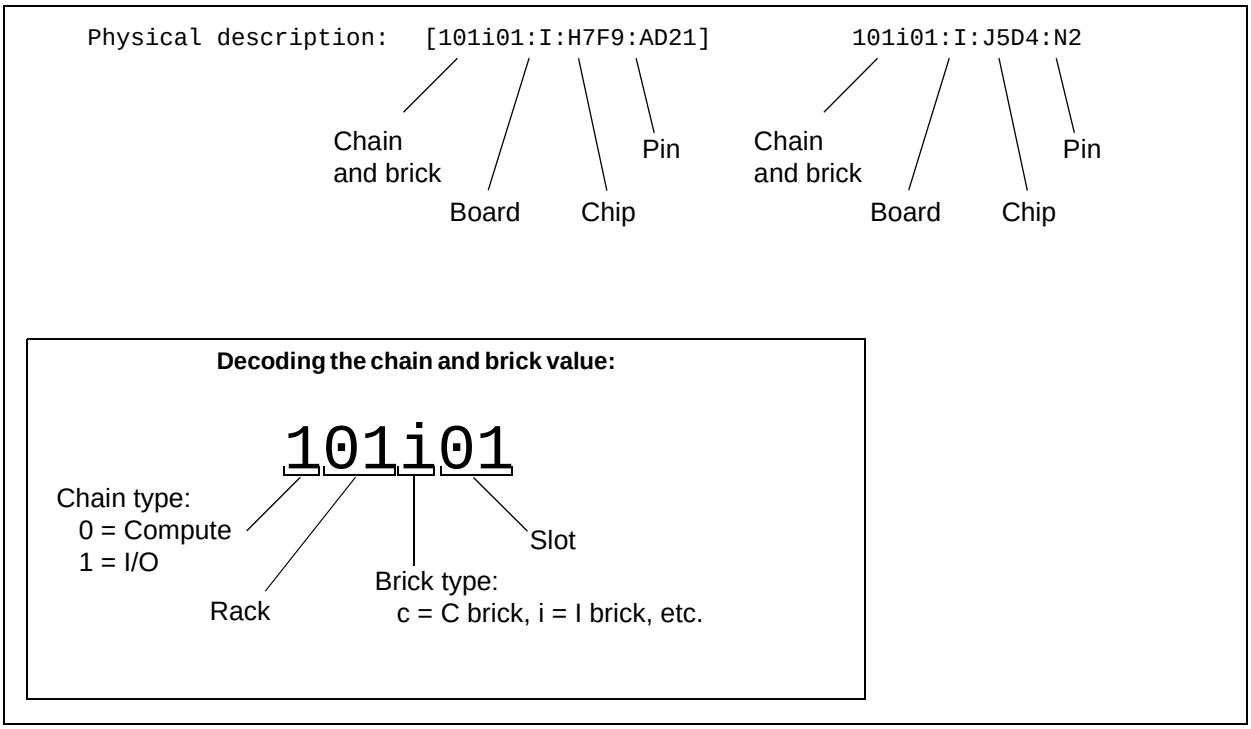

Figure 2-11 Physical Description Format

2.7 Troubleshooting Tips

- If the boundary scan interconnect test detects an error during the SIC selection process, then a failure occurred in the scan hardware and/or the path to the SIC. Check the scan signals (TCK, TMS, TDI, TDO, and TRST) that go to the SIC and the SIC address lines.
- If the boundary scan interconnect test detects an error during the register level tests, then a failure occurred in the scan hardware and/or path after the SIC. Check the scan signals (TCK, TMS, TDI, TDO, and TRST) at each component in the chain.
- If the boundary scan interconnect test detects a failure during the interconnect test, identify and correct it before you run the scan-based link-level protocol interconnect test.

2.8 Example 1: Possible PIMM Failure

The following output shows a possible PIMM failure:

```
Boundary-Scan Test Software (scantest V3.11) Mon Apr  2 11:31:20 2001

Loading chain configuration and reference data
   File: /stand/sysco/cfg/scan/c00155.cfg

Loading reference data for each board definition
   Path: /stand/sysco/data/scan/cmp

Initializing JTAG communications to scan controller(s)
   No hardware; running in software simulation/emulation mode

Initializing chain TAPs and configuring SICs:
001c01 with SIC at address 0x00 (CER=0x07)

Testing Integrity of Boundary Scan Instruction Registers
Performing IR length test on active UUTs:
001c01 SIC 0x0 ... passed (chain length=129)

Performing IR path test on active UUTs:
001c01 SIC 0x0 ... passed

Testing Integrity of Boundary Scan Bypass Registers
Performing BYPASS length test on selected UUTs:
001c01 SIC 0x0 ... passed (chain length=41)

Performing BYPASS path test on selected UUTs:
001c01 SIC 0x0 ... passed

Testing Integrity of Boundary Scan Data Registers
Performing chain-level BOUNDARY length test on selected UUTs:
001c01 SIC 0x0 ... passed (chain length=4226)

Performing chain-level BOUNDARY path test on selected UUTs:
001c01 SIC 0x0 ... passed

Performing system-level interconnect test on given configuration:
(22 of 22 patterns will be applied)
++++++++++++++++++++++
Errors were detected during interconnect test.

Failure information for BSR interconnect test:

Module and raw position : 000-001280
Expected data           : 10101001010110100101 01
Actual data             : 00000000000000000000
Difference data         : ^ ^ ^  ^ ^ ^^ ^  ^ ^ ^
Driving node index      : 11111111111111111002211
Logical description     : 001c01:NODE:BR:SYSAD_P0_34_BUS
[001c01:PIMM0:CPUA:SYSAD_34_BUS] 001c01:PIMM0:CPUB:SYSAD_34_BUS
Physical description    : 001c01:NODE:J0J2:AL14 [001c01:PIMM0:F9C3:C12]
001c01:PIMM0:C0C3:C12
```

```
Module and raw position : 000-001283
Expected data           : 1010011010100101010101
Actual data             : 11111111111111111111
Difference data         :  ^ ^^  ^ ^ ^^ ^ ^ ^ ^
Driving node index      : 11111111111111112211
Logical description     : 001c01:NODE:BR:SYSAD_P0_40_BUS
[001c01:PIMM0:CPUA:SYSAD_40_BUS] 001c01:PIMM0:CPUB:SYSAD_40_BUS
Physical description    : 001c01:NODE:J0J2:AR13 [001c01:PIMM0:F9C3:C16]
001c01:PIMM0:C0C3:C16

Number of failing signals displayed from test: 2

Normal Program Termination
```

In this example, the boundary scan interconnect test detected two failing signals.

The first signal is a connection between the Node (Bedrock ASIC) and PIMM0.

```
Logical description     : 001c01:NODE:BR:SYSAD_P0_34_BUS
[001c01:PIMM0:CPUA:SYSAD_34_BUS] 001c01:PIMM0:CPUB:SYSAD_34_BUS
Physical description    : 001c01:NODE:J0J2:AL14 [001c01:PIMM0:F9C3:C12]
001c01:PIMM0:C0C3:C12
```

The second signal is also connection between the Node (Bedrock ASIC) and PIMM0.

```
Logical description     : 001c01:NODE:BR:SYSAD_P0_40_BUS
[001c01:PIMM0:CPUA:SYSAD_40_BUS] 001c01:PIMM0:CPUB:SYSAD_40_BUS
Physical description    : 001c01:NODE:J0J2:AR13 [001c01:PIMM0:F9C3:C16]
001c01:PIMM0:C0C3:C16
```

The failing hardware could be PIMM0, the Bedrock ASIC, the board that contains them, or a connector between them. For this example, you would replace PIMM0 and run the test again.

2.9 Example 2: Possible DIMM Failure

The following output shows a possible DIMM failure:

```
Boundary-Scan Test Software (scantest V3.11) Mon Apr  2 11:37:41 2001

Loading chain configuration and reference data
   File: /stand/sysco/cfg/scan/c001dimm04.cfg

Loading reference data for each board definition
   Path: /stand/sysco/data/scan/cmp

Initializing JTAG communications to scan controller(s)
   No hardware; running in software simulation/emulation mode

Initializing chain TAPs and configuring SICs:
001c01 with SIC at address 0x00 (CER=0x0C)

Testing Integrity of Boundary Scan Instruction Registers
Performing IR length test on active UUTs:
001c01 SIC 0x0 ... passed (chain length=149)

Performing IR path test on active UUTs:
001c01 SIC 0x0 ... passed

Testing Integrity of Boundary Scan Bypass Registers
Performing BYPASS length test on selected UUTs:
001c01 SIC 0x0 ... passed (chain length=19)

Performing BYPASS path test on selected UUTs:
001c01 SIC 0x0 ... passed

Testing Integrity of Boundary Scan Data Registers
Performing chain-level BOUNDARY length test on selected UUTs:
001c01 SIC 0x0 ... passed (chain length=1706)

Performing chain-level BOUNDARY path test on selected UUTs:
001c01 SIC 0x0 ... passed

Performing system-level interconnect test on given configuration:
(22 of 22 patterns will be applied)
++++++++++++++++++++++
Errors were detected during interconnect test.

Failure information for BSR interconnect test:

Module and raw position : 000-000022
Expected data           : 10101010100101010101
Actual data             : 00000000000000000000
Difference data         : ^ ^ ^ ^ ^  ^ ^ ^ ^ ^ ^
Driving node index      : 00000000000000001100
Logical description     : [001c01:NODE:BR:MB_DATA_43_BUS] 001c01:DIMM0:U2:A4
Physical description    : [001c01:NODE:J0J2:AD35] 001c01:DIMM0:A9B3:1
```

```
Module and raw position : 000-000643
Expected data           : 1001010110100101010101
Actual data             : 111111111111111111111
Difference data         :  ^^ ^ ^  ^ ^^ ^ ^ ^ ^
Driving node index      : 0000000000000000001100
Logical description     : [001c01:NODE:BR:MB_DATA_90_BUS] 001c01:DIMM4:U2:A3
Physical description    : [001c01:NODE:J0J2:AG05] 001c01:DIMM4:A9B3:64

Number of failing signals displayed from test: 2

Normal Program Termination
```

In this example, the boundary scan interconnect test detected two failing signals.

The first signal is a connection between the Node (Bedrock ASIC) and DIMM0:

```
Logical description     : [001c01:NODE:BR:MB_DATA_43_BUS] 001c01:DIMM0:U2:A4
Physical description    : [001c01:NODE:J0J2:AD35] 001c01:DIMM0:A9B3:1
```

The second signal is a connection between the Node (Bedrock ASIC) and DIMM4:

```
Logical description     : [001c01:NODE:BR:MB_DATA_90_BUS] 001c01:DIMM4:U2:A3
Physical description    : [001c01:NODE:J0J2:AG05] 001c01:DIMM4:A9B3:64
```

The failing hardware could be the DIMMs, the Bedrock ASIC, the board that contains them, or a connector between them. For this example, you would swap DIMM0 and DIMM4 with other DIMMs that do not have failures to determine whether the failures move with the DIMM(s). If the failures move with the DIMM(s), you would replace the DIMM(s).

Chapter 3

Scan-based Link-level Protocol Interconnect Test

This chapter describes the scan-based link-level protocol test.

3.1 About the Scan-based Link-level Protocol Interconnect Test

Several ASICs contain an additional boundary scan register (LLP register) that can be used to send a single micropacket (160 bits in eight 20-bit, time-MUXed transfers) from one LLP port to another port. The scan-based link-level protocol (LLP) interconnect test includes an application (`slit`) that uses the LLP registers to exercise router links at-speed. It tests one or two ports per router at a time.

This test uses a minimal number of patterns to determine whether a port is functional. If it detects an error, it attempts to isolate the failure to a single net. If detects a single failing net, it identifies all chips and pins on the net and displays the actual/expected data.

The following ASICs support the at-speed LLP test feature:

- Router (ports A, B, C, D, E, F, G, and H)
- Bedrock (ports NI and II)
- Xbridge (ports 8, 9, A, B, C, and D)

3.2 Test Algorithm

This test uses the following algorithm:

1. Load the chain configuration and reference data.
2. Load the LLP link configuration and selection data.
3. Initialize JTAG communication to the scan controller(s).
4. Initialize the chain TAPs and configure the SIC(s).
5. Perform the scan hardware integrity tests:
   - Perform the IR length test.
   - Perform the chain-level LLP REG0 length test.
   - Perform the chain-level LLP REG1 length test.
6. Perform the scan-based LLP interconnect test.
7. Print a program termination status message.

Warning: **Each test step must pass before the next step can be run. If the integrity tests in Step 5 do not pass, the interconnect test does not run because the results would be invalid. Do not manually run the interconnect test until the integrity tests pass.**

Figure 3-1 shows the output from each step of this algorithm.

```
pyro% slit -f /stand/sysco/cfg/scan/p001.cfg -i
/stand/sysco/cfg/scan/p001.links

Scan-based LLP Interconnect Test (slit 2.6) Tue Jun 27 11:00:28 2000

1  Loading chain configuration and reference data
      File: /stand/sysco/cfg/scan/p001.cfg

2  Loading LLP link configuration and selection data
      File: /stand/sysco/cfg/scan/p000.links

3  Initializing JTAG communications to scan controller(s)

4  Initializing chain TAPs and configuring SICs:
           101p01 with SIC at address 0x0A (CER=0x08)

   Performing IR length test on active UUTs:
           101p01 ... passed (chain length=9)

5  Performing chain-level LLP REG0 length test on selected UUTs:
           101p01 ... passed (chain length=1011)

   Performing chain-level LLP REG1 length test on selected UUTs:
           101p01 ... passed (chain length=3)

6  Performing scan-based LLP interconnect test on selected link:

           Driving port   (SSD) = P:U0:A
           Receiving port (SSR) = P:U0:A
           (number of patterns=4; flit mask per pattern=0xFF)
           ++*+*++*+++**+++**+*+*+**+++*+++*+*++*+**
           Errors were detected during interconnect test.

   Failure information for scan-based LLP interconnect test:

   Pattern (8-bits each) : ---01--- ---02--- ---03--- ---04---
   Actual data           : 01101010 0000.... 0000.... 10000000
   Difference data       :   ^^     ^^ ^     ^^ ^      ^ ^  ^
   Logical description   : 101p01:U0:PORT_A_OUT_DATA_6
   [101p01:U0:PORT_A_IN_DATA_6]

7  Normal Program Termination
```

Figure 3-1 slit Test Algorithm Components

3.3 Running the Scan-based Link-level Protocol Interconnect Test

Perform the following procedure to run the scan-based link-level protocol interconnect test:

Warning: **Ensure that the system is offline before you run this test. If the operating system is running when you use this test, the operating system will crash and customer data will be lost.**

1. Start the L2 software and power up the system.
2. Log on to the L3 system controller as `sgidiag` (default password: `sgi!diag`).
3. Enter **`slit -f /stand/sysco/cfg/scan/<file> -i /stand/sysco/cfg/scan/<file>`** to run the test.

   Example: `slit -f /stand/sysco/cfg/scan/sys8p.cfg -i /stand/sysco/cfg/scan/sys8p.links`

   Table 3-1 describes the command line options that are available.
4. Interpret the output. (Refer to "Test Output" on page 3-7 for more information.)

Table 3-1 slit Command Line Options

Option	Description
`-h`	Displays the available command line options and then exits.
`-f <file>`	Specifies a configuration file to use. The configuration file defines the hardware to test. (Default: `slit.cfg`) Use the following configuration files to test entire systems: `sys8p.cfg` = system with 8 processors `sys16p.cfg` = system with 16 processors `sys32p.cfg` = system with 32 processors `sys64p.cfg` = system with 64 processors `sys128p.cfg` = system with 128 processors Use the following configuration files to target specific bricks: `c001.cfg` = IP35 C brick `i002.cfg` = I brick `p001.cfg` = P brick `r001.cfg` = R brick `x001.cfg` = X brick `ip37_002cF.cfg` = IP37 C brick with 2 MB CPUs (CPU 0, 1, 2, and 3) `ip37_002CF.cfg` = IP37 C brick with 4 MB CPUs (CPU 0, 1, 2, and 3) Other configuration files are available in the `/stand/sysco/cfg/scan/` directory. Refer to the `/stand/sysco/cfg/scan/README.cfg` file for more information.

Table 3-1 (continued) slit Command Line Options

Option	Description
`-i <file>`	Specifies a file that contains input for the test. The input file describes the port-to-port connections that will be tested. (Default: `slit.in`) Use the following files to test entire systems: `sys8p.links` = system with 8 processors `sys16p.links` = system with 16 processors `sys32p.links` = system with 32 processors `sys64p.links` = system with 64 processors `sys128p.links` = system with 128 processors Use the following files to target specific bricks: `c001.links` = IP35 C brick `i002.links` = I brick `p001.links` = P brick `r001.links` = R brick `x001.links` = X brick `ip37_002.links` = IP37 C brick
`-d <file>`	Specifies a file to receive output from the test. (Default: `slit.dmp`)
`-p <num>`	Specifies the number of test passes that `slit` should perform. (Valid values are between 1 and 10,000; default: 1)
`-e <num>`	Specifies the maximum number of errors to display from the interconnect test. (Range: 0 to 10,000; default: 200) If you specify 0 errors, no detailed failure information is reported, which may be useful if the test is run from an automated application.
`-t <test>`	Specifies the tests to run. Valid tests: irp = instruction register path test irl = instruction register length test ir = instruction register path test and instruction register length test prp = bypass register path test prl = bypass register length test pr = bypass register path test and bypass register length test lrp = LLP register path test lrl = LLP register length test lr = LLP path test and boundary length test llp = LLP at-speed interconnect test id = read and display device ID registers all = all tests Default test selection: irl+lrl+llp
`-m <mode>`	Specifies the mode for the register tests. Valid modes: normal = normal lengths and minimal pattern(s) elen = extended lengths epat = extended pattern set If normal is specified, then no other option is valid.(Default: normal)

Table 3-1 (continued) slit Command Line Options

Option	Description
`-l <level>`	Specifies the level of the register tests. Valid modes: chn = chain level register test brd = board level register test chip = chip level register test (Default: chn)
`-C`	Selects an individual Scan Interface Chip (SIC) to run the test. (The test will not run on any other SICs.) When you use this option: You can specify one of the following tests with `-t`: irl, irp, ir, prl, prp, pr The `-l` option is ignored. The `-m` option is used for lengths and patterns.
`-s <mode>`	Selects a stepping mode. Valid modes: no = no test stepping; run continuous all = step through each pattern; prompt user err = step through failing pattern; prompt user (Default: no)
`-L <pats>`	Selects the LLP test pattern that the test uses. Valid options: detect = patterns to detect interconnect faults isolate = patterns to isolate short/bridging faults stress = patterns to stress the interconnects custom = user-defined patterns (Default: detect)
`-F <mask>`	Selects the LLP flits that the test should capture and compare. (Each test packet contains eight separate flits.) The mask parameter is a bitmap of the chosen flits: If bit 0 is set, then flit 0 is captured/compared. If bit 1 is set, then flit 1 is captured/compared. (This parameter can be a hexadecimal number between 0x01 and 0xFF; default: 0xFF)
`-R`	Disables use of resets.
`-v`	Selects verbose mode, which writes additional messages to the output.

5. When you are done testing, enter the L2 `reset` command to reset the system.

3.4 Test Output

When this test detects an error, it continues testing the entire suite of test vectors before it generates error output. Once it has completed the entire suite of test vectors, it generates output that shows the actual and expected values for a miscompare. It also displays the specific chip and pin information for each net that had a miscompare. (The failing chip and pin are identified by logical location.)

The error output uses the following format:

```
Errors were detected during interconnect test.

Failure information for scan-based LLP interconnect test:

Pattern (8-bits each) : ---01--- ---02--- ---03--- ---04---
Actual data           : 01101010 0000.... 0000.... 10000000
Difference data       :   ^^     ^^ ^     ^^ ^      ^ ^  ^
Logical description   : 101p01:U0:PORT_A_OUT_DATA_6
[101p01:U0:PORT_A_IN_DATA_6]
```

The error information contains the following information:

- `Pattern`: indicates the data pattern that `slit` was using when the miscompare occurred. (Each data signal transfers 8 bits of data per test pattern. The number of patterns applied is determined by the pattern type and LLP version. `slit` supports three predefined pattern sets [detect, isolate, and stress] and user-defined custom patterns.)
- `Actual data`: shows the actual data that was read from the system. (If valid data was not captured for a specific pattern [due to test configuration or a problem] a period (.) is shown.)
- `Difference data`: contains a `^` symbol for each pattern in the signature where the actual data did not match the expected data.
- `Logical description`: provides a logical description of the ports on the failing connection.

  The format of each node is `<board>:<chip>:<pin>`. The driver is listed first, followed by the receiver; the receiver is the node that detected the miscompare.

  The format of the <pin> is `PORT_<port>_<dir>_DATA_<num>`, in which:

  `<port>` = LLP port on ASIC (for Xbridge: 8, 9, A, B, C, D)

  `<dir>` = `OUT` for SSD output; `IN` for SSR input

  `<num>` = Data line number (0 - 19)

  Note: The test can only identify a differential pair; it cannot isolate which signal of the pair is failing.

3.5 Troubleshooting Tips

- If the fault is detected between the A and B ports, use a *single-wrap* cable or loopback cable on each port to isolate the errors to a specific port. Replace the HIC on a failing port to isolate the error to the motherboard or HIC.
- If multiple failures are detected on the A or B ports, double-check the seating of the HIC and cable.
- If an entire port fails (20 data lines, 2 clock lines and data ready) and all of the inputs are sampling zero (0), check the Widget Present signal. (The receivers are disabled if the Widget Present is not correct.)
- If the `slit` LLP register test fails but the boundary scan interconnect test passes, check the system clock (required for LLP register to shift).
- If the LLP-register interconnect test does not capture valid data for any pattern, use the L1 system controller to verify that both ports are configured with the same speed. Then, verify that the clocks are running.

3.6 Example

For this example, assume that `slit` generates the following error information:

```
SCAN-BASED LLP INTERCONNECT FAILURE
-----------------------------------
es
Pattern (8-bits each) : ---01--- ---02--- ---03--- ---04---
Actual data           : 00000000 00000000 00000000 00000000
Difference data       :  ^ ^^ ^  ^^ ^  ^  ^^ ^  ^  ^^ ^  ^
Logical description   : 101p01:P:XBG0:PORT_9_OUT_DATA_6
[101p01:P:XBG1:PORT_8_IN_DATA_6]
```

From the output in this example, you can determine the following information:

Node Number	Chain/ Brick	Board Name	Logical Chip	Logical Pin	LLP Port	Signal Type	Data Line
0	101p01	P	XBG0	PORT_9_OUT_DATA_6	9 SSD	OUT	6
1	101p01	P	XBG1	PORT_8_IN_DATA_6	8 SSR	IN	6

The data is driven from the SSD of port 9 on 101p01:P:XBG0 and received in the SSR of port 8 on 101p01:P:XBG1. The data line is valid for all bits of each pattern, and captured all zeroes (0's).

Chapter 4

scantool Application

This chapter describes the `scantool` application. `scantool` enables you to control scan operations from Tcl/Tk scripts. `scantool` includes graphical user interface and command line modes.

4.1 Available Scripts

Table 4-1 through Table 4-3 describe useful scripts that are available in the current diagnostic release.

Table 4-1 IP35 C-brick Scripts (Located in /stand/sysco/data/scan/diags/ip35)

Script	Description
`ip35_debug.tcl`	Configures the debug port MUX to steer a selected set of 32 signals to the debug port of the Bedrock ASIC where the signals can be viewed by a logic analyzer (This script is not used in the field.)
`ip35_dump_br_latches.tk`	Dumps the Bedrock ASIC latches on an IP35 (This script could be used with the FRU analyzer to dump the state of the Bedrock ASIC when a brick does not respond to an NMI.)
`ip35_llp.tcl`	Configures and runs a simple LLP test packet by using the at-speed LLP test feature (This script is not used in the field.)
`ip35_membist.tcl`	Runs the on-board memory built-in self test (BIST) feature on the Bedrock ASIC (This script is not used in the field.)

Table 4-2 IP37 C-brick Scripts (Located in /stand/sysco/data/scan/diags/ip37)

Script	Description
`ip37_di_bist.tk`	Runs the IP37 L4 cache memory BIST
`ip37_dump_br_latches.tk`	Dumps the Bedrock ASIC latches on an IP37 (This script could be used with the FRU analyzer to dump the state of the Bedrock ASIC when a brick does not respond to an NMI.)
`ip37_membist.tcl`	Runs the on-board memory built-in self test (BIST) feature on the Bedrock ASIC (This script is not used in the field.)
`ip37_trigger.tk`	Runs the Synergy ASIC with various clocking waveforms and duty cycles (This script is not used in the field.)
`ip37_syn_mmr_dump.tk`	Captures the value of all Synergy ASIC memory mapped registers (MMRs) This script provides useful data when the system does not respond to an NMI and also provides a noninvasive and concurrent snapshot of the MMR values.
`ip37_trigger.tk`	Sets the clock counter and debug port parameters, which enable a user to configure the Synergy ASIC core clock to stop after a specific number of clock cycles in three modes: 1. The clock counter starts decrementing as soon as it is activated. 2. The clock counter decrements each time a value/mask combination of the debug port is matched. 3. The clock counter resets each time a value/mask combination of the debug port is matched (watchdog mode). (This script is not used in the field.)

Table 4-3 R-brick Scripts (Located in /stand/sysco/data/scan/diags/rbrick)

Script	Description
`r_debug.tcl`	Configures the debug port MUX to steer a selected set of 32 signals to the debug port where the signals can be viewed by a logic analyzer (This script is not used in the field.)
`r_stopclk.tcl`	Stops the router clock

4.2 Running Scripts from a Graphical User Interface

Perform the following procedure to run `scantool` scripts from the graphical user interface on a single brick:

1. Start the L2 system controller and power up the system.
2. Log on to the L3 system controller as `sgidiag`. (Default password: `sgi!diag`)
3. Enter **`scantool`** at the L3 system controller prompt:

   ```
   scantool [-f <script_name>] [-n | -s |-0 | -1]
   [--ls <host> | -- interface <iface> | --ssn <serial>[:<rack>] |
   --sysname <system_name>[:<rack>] | --scdev <device> | --serial
   <host>:<port> | --dev <device name>]
   ```

   Table 4-4 describes the command line options that you can use to control `scantool`.

Table 4-4 scantool Command Line Options (Graphical User Interface Mode)

Option	Description
`-c <config_file>`	Selects the configuration file to use
`-f <script_name>`	Selects the script to run You can also use the graphical user interface to select which script to run.
`-h`	Displays help information
`-n`	Sets the communication method to “null” mode: `scantool` runs the script but does not send the commands to the scan hardware. No scan data is received from the hardware.
`-s`	Sets the communication method to “simulation” mode: `scantool` connects to a Verilog simulator through a socket connection. (This configuration is not used in the field.)
`-0`	Sets the communication method to “SN0” mode: `scantool` connects to an external system through the parallel port of a Silicon Graphics Indy visual workstation. (This configuration is not used in the field.)
`-1`	Sets the communication method to “SN1” mode: `scantool` connects to an SGI 3000 series system through an L3 system controller. The L3 system controller must be connected to an L2 system controller or it must be running the L2 emulator. This is the default communication method.
`--l2 <host>`	Connects to the L2 system controller with the specified host name or IP address
`--interface <iface>`	Connects to the L2 system controller with the specified network interface (for example, eth0)

Table 4-4 (continued) scantool Command Line Options (Graphical User Interface Mode)

Option	Description
`--ssn <serial>[:<rack>]`	Connects to the system with the specified serial number (for example, L01234567)
`--sysname <system_name>[:<rack>]`	Connects to the system specified by `<system_name>`
`--scdev <device>`	Connects to the specified system controller
`--serial <host>:<port>`	Connects to the specified Ethernet host name and port
`--dev <device name>`	Connects to the specified local serial port device

Figure 4-1 shows the `scantool` graphical user interface (also called the graphical console).

Figure 4-1 scantool Graphical User Interface (Initial Window)

4. Choose **`File -> Set SINGLE brick Config`**. (The interface displays the `Configure One Brick` window; refer to Figure 4-2.)
5. Select the brick that you want to test (refer to Figure 4-2):
   - Enter the rack number in the `Rack Number` field.
   - Enter the slot number in the `Slot Number` field.

     If you do not know the rack and slot locations of the brick that you want to test, use the L2 `cfg` command to determine the bricks that are available.
   - Select the brick type from the `Brick Type` menu.

     If you are testing an IP35 C brick, you can select the PIMMs to test (none, PIMM0, and / or PIMM1).

     If you are testing an IP37 C brick, you can select the CPU type (2 MB or 4 MB), CPU build (P0 or P1), and the CPUs to test (CPU 0, 1, 2, and / or 3).

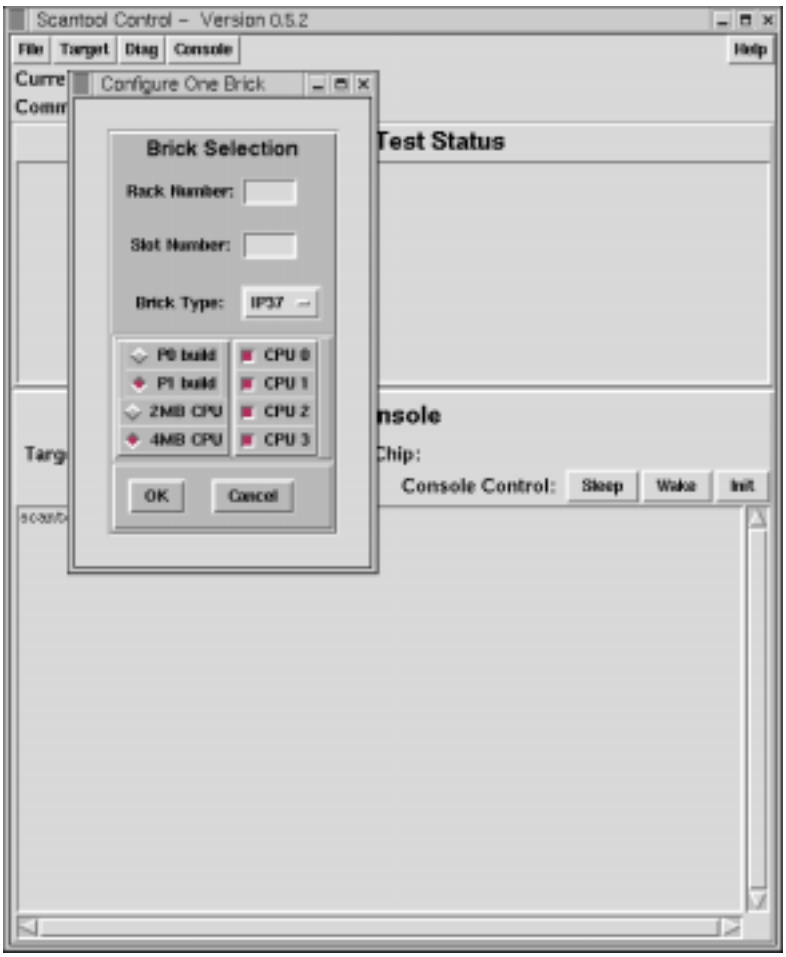

Figure 4-2 Selecting the Brick to Test

Figure 4-3 shows example settings used to select CPUs 0, 1, 2, and 3 in an IP37 C brick located in slot 8 of rack 5. (In this example, the CPUs are 4-MB P1 CPUs.)

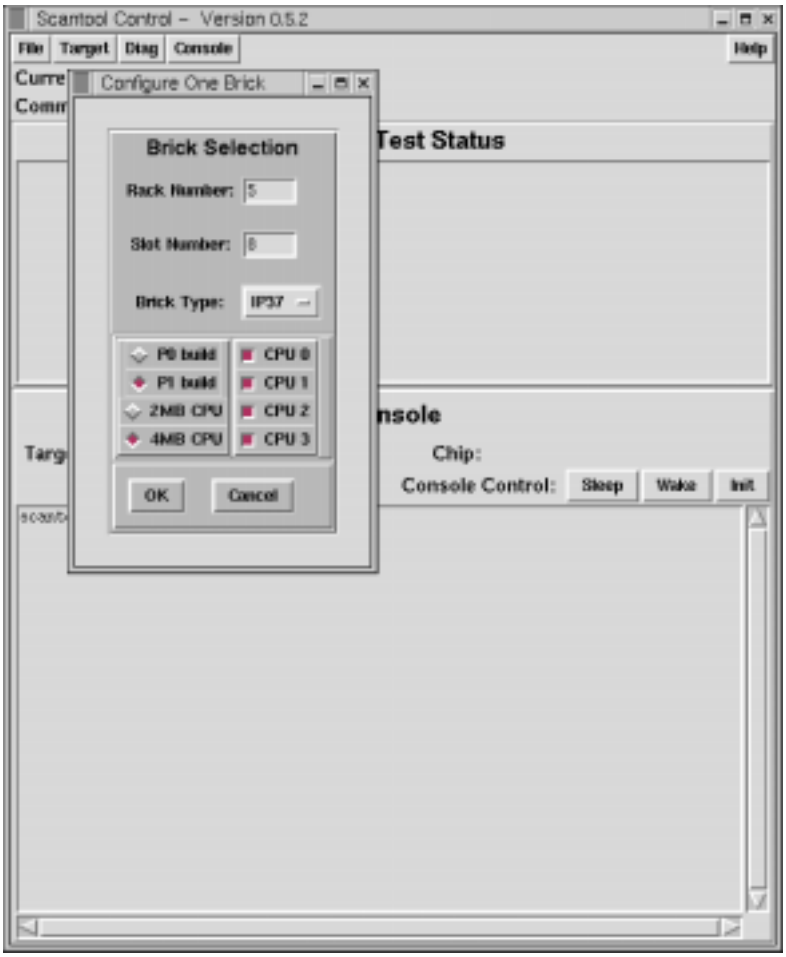

Figure 4-3 Selecting the Brick (Example Settings)

6. Click on the `OK` button.

   Figure 4-4 shows the interface with the brick selected. Notice that the `Targeted Module` field shows the brick from this example (005c08 = C brick in slot 8 of rack 5).

Figure 4-4 Interface with Brick Selected

7. Choose **`Diag -> Start Console Diag`** to load a script.

   Note: If you do not want to run a script, you can enter individual `scantool`, Tcl, and Tk commands at the `scantool>` prompt.

   The interface displays the `Open` window. (Refer to Figure 4-5.) Use this window to select the script to run.

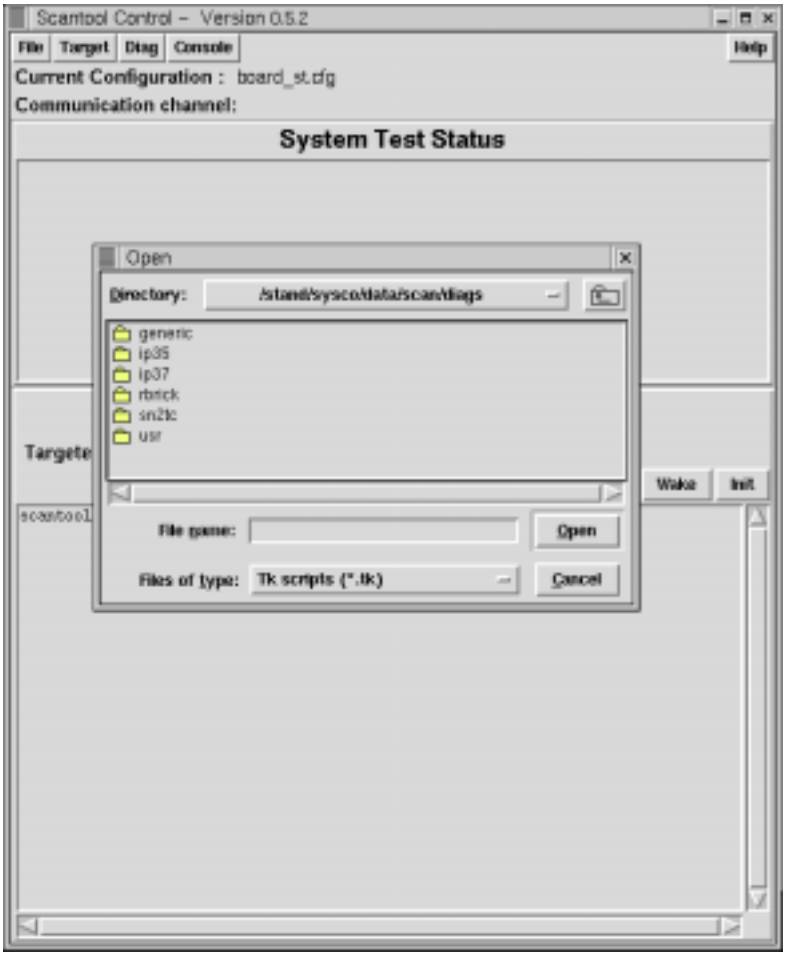

Figure 4-5 Selecting the Script to Run

8. Select the script that you want to run.

   - To select a Tcl script, set the `Files of type` field to `Tcl scripts`. Then, select the script from the appropriate directory. (Refer Figure 4-6.)
   - To select a Tk script, set the `Files of type` field to `Tk scripts`. Then, select the script from the appropriate directory. (Refer to Figure 4-7.)

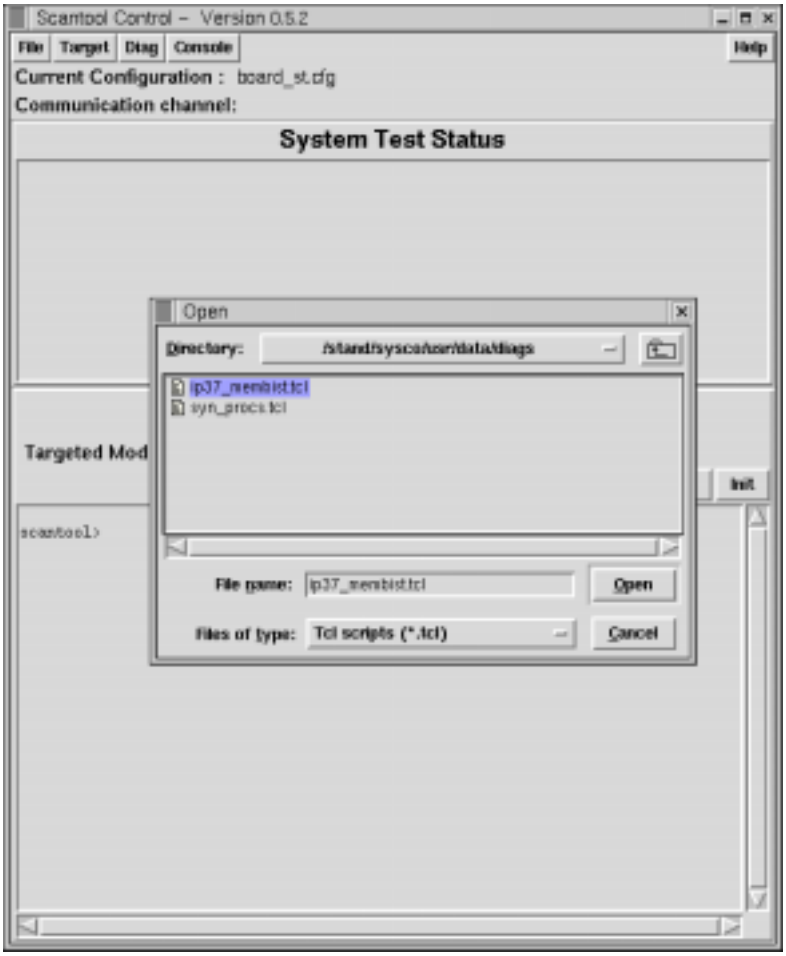

Figure 4-6 Selecting a Tcl Script

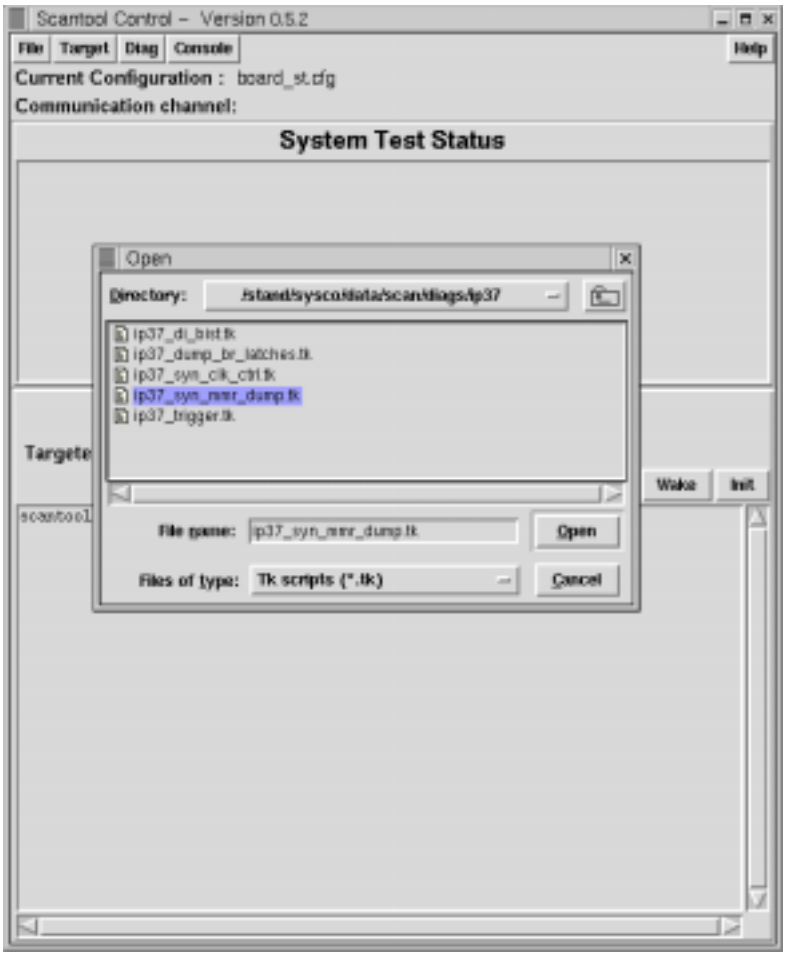

Figure 4-7 Selecting a Tk Script

If you select a Tcl script, the output from the script appears in the `Console` portion of the window. (Refer to Figure 4-8.)

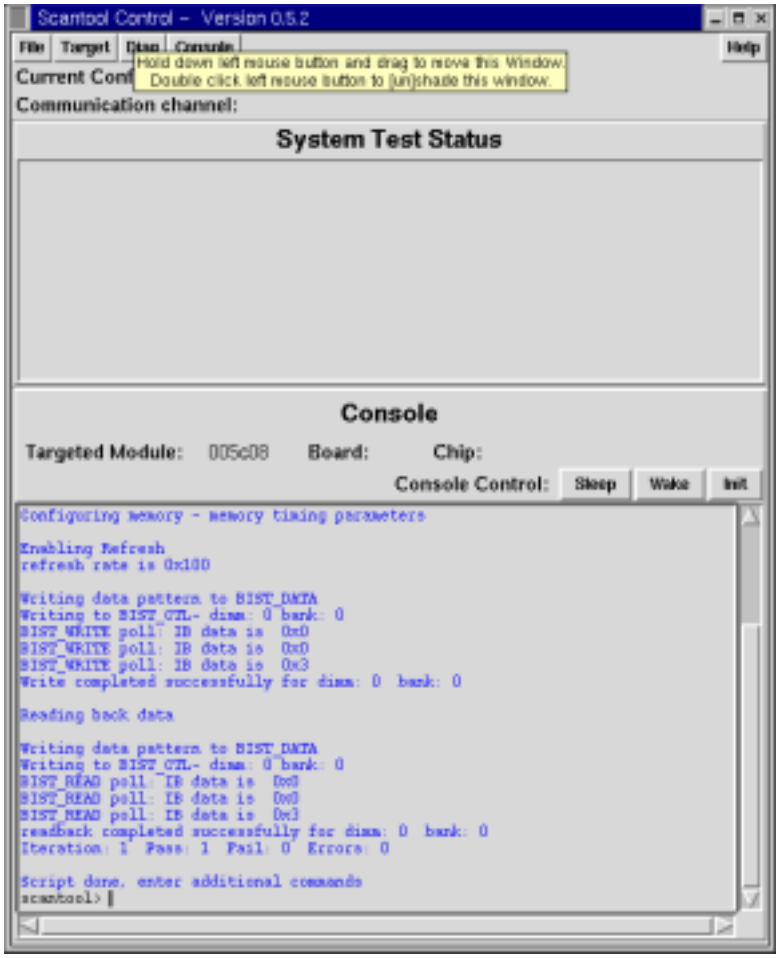

Figure 4-8 Running a Tcl Script

If you select a Tk script, the script displays a new window that you can use to set parameters. (Refer to Figure 4-9.) Set the parameters, and click `GO` to run the script. Output from the script appears in the window. (Refer to Figure 4-10.)

Figure 4-9 Running a Tk Script (Example Window)

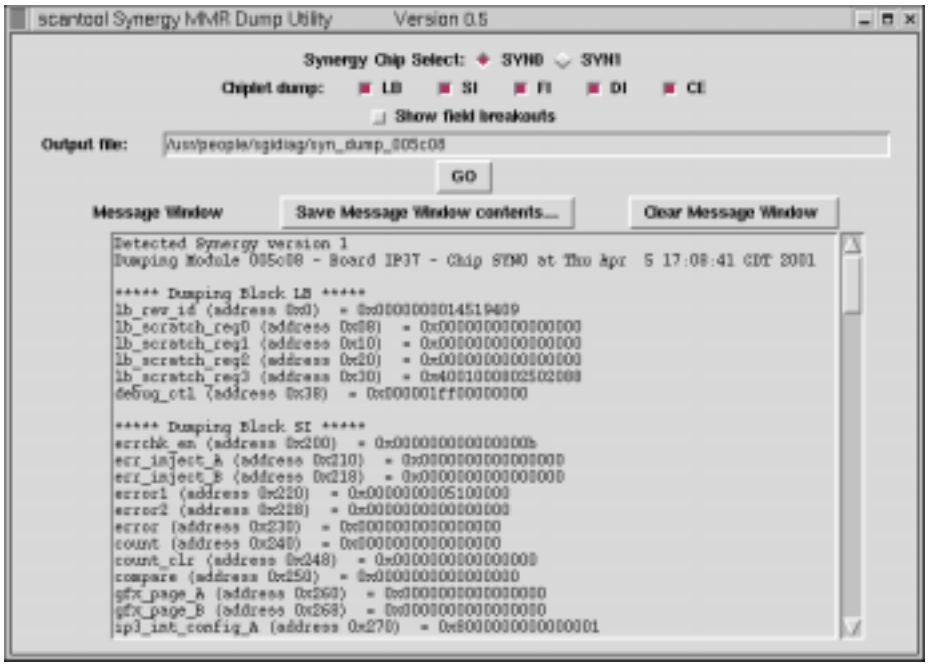

Figure 4-10 Running a Tk Script (Example Output)

9. When you finish running scripts, close any windows that `scantool` opened and then choose **`File -> Exit`**.

   Note: Normally, you do not need to reset the system after using `scantool`. However, if you dump the internal latches (typically after a system hang), you should enter the L2 `reset` command to reset the system.

4.3 Running Scripts from the Command Line

Perform the following procedure to run `scantool` scripts from the command line:

1. Start the L2 system controller and power up the system.
2. Log on to the L3 system controller as `sgidiag`. (Default password: `sgi!diag`)
3. Ensure that the required environment variables are properly set. (Refer to Table 4-5.)

Table 4-5 scantool Environment Variables

Variable	Setting
SCAN_PATH_DATA	/stand/sysco/data/scan
SCAN_PATH_CONFIG	/stand/sysco/cfg
SCAN_PATH_BIN	/stand/sysco/bin
LD_LIBRARY_PATH	/stand/sysco/lib
TCL_LIBRARY	/stand/sysco/lib

4. Select the configuration file to use. (Refer to Table 4-6.)

Table 4-6 scantool Configuration Files

File	Description
001c10_st.cfg	Targets the C brick in slot 10 of rack 1
001c13_st.cfg	Targets the C brick in slot 13 of rack 1
001c16_st.cfg	Targets the C brick in slot 16 of rack 1
001c21_st.cfg	Targets the C brick in slot 21 of rack 1
001c24_st.cfg	Targets the C brick in slot 24 of rack 1
001c29_st.cfg	Targets the C brick in slot 29 of rack 1
001c32_st.cfg	Targets the C brick in slot 32 of rack 1
001c35_st.cfg	Targets the C brick in slot 35 of rack 1
001r19_st.cfg	Targets the R brick in slot 19 of rack 1
001r27_st.cfg	Targets the R brick in slot 27 of rack 1
sys_8p_st.cfg	Targets bricks in default configuration for an 8-processor system (C bricks and one I brick)

Table 4-6 (continued) scantool Configuration Files

File	Description
sys_16p_st.cfg	Targets bricks in default configuration for a 16-processor system (C bricks, R bricks, and one I brick)
sys_32p_st.cfg	Targets bricks in default configuration for a 32-processor system (C bricks, R bricks, and one I brick)
sys_64p_st.cfg	Targets bricks in default configuration for a 64-processor system (C bricks, R bricks, and one I brick)
sys_128p_st.cfg	Targets bricks in default configuration for a 128-processor system (C bricks, R bricks, and one I brick)

Note: If you need to modify a configuration file, copy the file to $SCAN_USER_CONFIG/user_st.cfg and modify it. Then, use the -c command line option to load the modified file.

5. Select (or create) the script that you want to run. (Refer again to Table 4-1 through Table 4-3 for the scripts that are available in the diagnostic release.)

6. Enter the **scantool -l** command at the L3 system controller prompt:

```
scantool -l -c <config_file> [-f <script_name>] [-g] [-n | -s |-0 | -1]
[--ls <host> | -- interface <iface> | --ssn <serial>[:<rack>] |
--sysname <system_name>[:<rack>] | --scdev <device> | --serial
<host>:<port> | --dev <device name>]
```

Table 4-7 describes the options that you can use to control scantool in command line mode.

Table 4-7 scantool Command Line Options (Command Line Mode)

Option	Description
-l	Starts scantool in command line mode (a prompt appears at which you can enter scantool commands)
-c <config_file>	Specifies the configuration file to use
-f <script_name>	Specifies the script to run If you do not specify a script to run, scantool displays a % prompt at which you can enter scantool commands.
-h	Displays help information
-g	Enables scantest to run graphical Tk scripts in command line mode
-n	Sets the communication method to “null” mode: scantool runs the script but does not send the commands to the scan hardware. No scan data is received from the hardware.

Table 4-7 (continued) scantool Command Line Options (Command Line Mode)

Option	Description
`-s`	Sets the communication method to "simulation" mode: `scantool` connects to a Verilog simulator through a socket connection. (This configuration is not used in the field.)
`-0`	Sets the communication method to "SN0" mode: `scantool` connects to a system through the parallel port of a Silicon Graphics Indy visual workstation. (This configuration is not used in the field.)
`-1`	Sets the communication method to "SN1" mode: `scantool` connects to an SGI 3000 series system through an L3 system controller. The L3 system controller must be connected to an L2 system controller or it must be running the L2 emulator.
`--l2 <host>`	Connects to the L2 system controller with the specified host name or IP address
`--interface <iface>`	Connects to the L2 system controller with the specified network interface (for example, eth0)
`--ssn <serial>[:<rack>]`	Connects to the system with the specified serial number (for example, L01234567)
`--sysname <system_name>[:<rack>]`	Connects to the system specified by `<system_name>`
`--scdev <device>`	Connects to the specified system controller
`--serial <host>:<port>`	Connects to the specified Ethernet host name and port
`--dev <device name>`	Connects to the specified local serial port device

For example, the following command line starts `scantool` and runs the `ip35_membist.tcl` script on the IP35 C brick in slot 16 of rack 1 of an SGI 3000 system:

```
pyro> scantool -l -c $SCAN_PATH_CONFIG/scan/001c16_st.cfg -f
$SCAN_PATH_DATA/diags/ip35/ip35_membist.tcl
```

7. Interpret the output.

Note: Normally, you do not need to reset the system after using `scantool`. However, if you dump the internal latches (typically after a system hang), you should enter the L2 `reset` command to reset the system.

4.4 scantool Script Commands

You can copy the standard `scantool` scripts and customize them for your site. You can also create new `scantool` scripts. (Table 4-8 shows the script commands that are available.)

Note: To modify a script, you must copy it to a user-writable directory (for example, `/stand/sysco/usr` or `/usr/people/sgidiag`).

Table 4-8 scantool Script Commands

Command	Description
SEL	Selects an object
DSEL	Deselects an object
INST	Sets the current instruction
GOINST	Scans the current instruction into a scan chain
DATA	Sets the current data pattern or reads data
GODATA	Scans the current data pattern into a scan chain
REG	Specifies a user-defined data pattern
REGDATA	Sets the user-defined data
RESET	Resets the TAP controller(s) on the current scan chain
SIC	Causes a “family” to access and manipulate the scan interface chip (SIC)
GORUN	Moves the TAP controller to the run-test-idle state and toggles the TCK signal
GOIDLE	Moves the TAP controller to the run-test-idle state
INFO	Causes a “family” to extract information from an entity
SIMACC	Causes a “family” to control a Verilog simulation
EXIT	Exits the script

Refer to the following URL for more information about these commands:

`http://wwwcf.americas.sgi.com/~kermes/scantool/scantool_toc.html`

Reader Comment Form

Title: Scan Tools (SGI™ 3000 Family) **Number: 108-0263-002**

Your feedback on this publication will help us provide better documentation in the future. Please take a moment to answer the few questions below.

For what purpose did you primarily use this document?

_____Troubleshooting _____Tutorial or introduction
_____Reference information _____Classroom use
_____Other - please explain

_____

Using a scale from 1 (poor) to 10 (excellent), please rate this document on the following criteria and explain your ratings:

_____Accuracy _____
_____Organization _____
_____Readability _____
_____Physical qualities (binding, printing, page layout) _____
_____Amount of diagrams and photos _____
_____Quality of diagrams and photos _____

Completeness (Check one and explain your answer)

_____Too much information _____ Too little information _____Correct amount

_____
_____
_____

You may write additional comments in the space below. Mail your comments to the address below, fax them to us at +1 715 726 4353, or e-mail them to us at *spt@sgi.com.* When possible, please give specific page and paragraph references. We will respond to your comments in writing within 48 hours.

_____
_____
_____

NAME _____
JOB TITLE _____
E-MAIL ADDRESS _____
SITE/LOCATION _____
TELEPHONE _____
DATE _____
[or attach your business card]

Attn: Service Publications and Training
890 Industrial Boulevard
P.O. Box 4000
Chippewa Falls, WI 54729-0078
USA