BOUNDARY SCAN SYSTEM TEST

Figures

Notational Conventions

This document uses the following notational conventions:

- `Courier` type indicates directory pathnames, filenames, program names commands, and screen output.
- **`Courier bold type`** indicates user input commands, options, and field inputs.
- The following conventions are used in the command descriptions:
    - *Italic* type indicates a variable or a user-supplied command.
    - Square brackets [ ] indicate an optional entry.
    - A vertical bar | indicates a choice.
    - Angle brackets < > indicate a required entry.

Boundary Scan System Test Overview

The boundary scan system test is used to test the interconnections on and between modules, to log any detected errors, and to provide a status report. The boundary scan system test includes:

- The boundary scan program (bscan)
- The boundary scan system data generator (bsb)
- The boundary scan report generator (breport)
- The runbscan script shell utility

The bscan program tests the interconnections between options on a module and connections between modules. It reports and logs any detected errors for a given configuration of a CRAY T90 series mainframe. The following two types of interconnections cannot be tested with boundary scan:

- Interconnections connected directly to the memory stacks
- I/O channels

These interconnections cannot be tested because there are no boundary scan capabilities built into the memory chips or the I/O channels.

Use bscan to verify the integrity of the mainframe after a failure occurs that shuts down the mainframe or after you complete a repair procedure. The bscan program requires control of the mainframe; therefore, you cannot run a boundary scan test and the operating system simultaneously. CRAY T932 mainframes, however, can be degraded, allowing you to run boundary scan tests on one half of the mainframe while the operating system is running on the other half of the mainframe.

The bscan program invokes bsb. The bsb program creates the system data read result vectors from a system data file (system specific file) using the module data files (supplied by Cray Research, Inc.). For more information on bsb, refer to page 21.

The breport program reads the output from bscan and presents it in an easy-to-interpret format. It also retrieves expected data for any module location and bit number (refer to Figure 9, on page 19).

The runbscan shell script utility enables the boundary scan system test to run automatically. It enables bscan to run a sequence of tests, and automatically runs breport to generate a report (refer to "runbscan," on page). The boundary scan system test takes 32 minutes to run on CRAY T94 systems, and up to 10 minutes on CRAY T932 systems.

This document includes the following information:

- Configuration files
- Data files
- The bscan program
- The breport program
- The bsb program
- The runbscan shell script utility

Configuration File

The configuration file provides the bscan program with the location of the system module you are testing. The configuration file specifies the modules you select to test.

Locations not specified in the configuration file are not tested. Locations specified in the configuration file that are not in the system header file, or port numbers that are out of range, generate an error message and terminate the program.

You may need to change the configuration file to comment out modules or to change port numbers. To change the configuration file, use the vi editor. You can use either upper- or lowercase letters; extra spaces and tabs are ignored. Comments begin with a "#" sign and continue to the next new line. The command line must contain the key words LOC (location), MOD (module type), REV (module revision number), and PORT (port number) and must be written in the same order shown in the configuration file example below:

```
# Comment line followed by a blank line

LOC=B001  MOD=CP02  REV=3100  PORT=32 #Comments
LOC=C001  MOD=CM02  REV=2004  PORT=36
```

Data Files

The boundary scan system test uses the following types of data files to use `bscan`, `breport`, `bsb`, and `runbscan`:

- Module data files
- System data files
- System data mask files

Module Data Files and System Data Files

Module data files are supplied to the site and cannot be generated locally. They are specific for module types. System data files are created using module data files and are generated on-site using the `bsb` program (refer to `bsb` on page 21).

NOTE: The `bsb` program is not included in the initial release of the Boundary Scan System Test.

System Mask Data Files

The `bscan` program generates system data mask files at run time using the module data files and system data files. The system data mask files identify the bits in the scan chain to be tested. System data mask files are generated if any of the following conditions occur:

- The system data mask file is missing
- The `-F` option is used to force a build of system data mask files
- A new configuration of modules is being tested

No changes are made to the system data mask files when you use `bscan` to test a single module.

bscan Program

The bscan program tests the interconnections on a module and between modules, and reports detected errors for a given configuration. Use bscan to verify the integrity of the mainframe after a failure occurs that shuts down the mainframe or after you complete a repair procedure.

The bscan program requires control of the mainframe; therefore, you cannot run a boundary scan test and the operating system simultaneously. CRAY T932 mainframes, however, can be partitioned, allowing you to run bscan tests on one half of the mainframe while the operating system is running on the other half of the mainframe.

The bscan program compares the actual data with the expected data and completes when a maximum pass count, or wall-clock time limit, occurs.

bscan Command

Start the bscan program from a UNIX command prompt with the following command:

```
/cri/bin/bscan [-c channel] [-d sysdata_dir]-f cfg_file
               [-h] [-i] [-m moddata_dir] [-s system]
               [-t 'test [time hh:mm:ss | maxpass n]']
               [-AFM] [module_loc...]
```

NOTE: To run a sequence of bscan tests, use the runbscan shell script utility.

bscan Command Options

Enter the bscan command options in any order. The program uses the default value if you omit an option. You may locate the options in the UNIX operating system online man pages as well as in the Appendix of this document.

bscan Tests

In order to simplify the output, the following test examples are written with the assumption that a single CP and CM module are used. The test results are sent to stdout and the error messages are sent to standard error device (stderr). The bscan tests are:

- module identification
- qport
- qchipid
- chipid
- shift
- scan

Module Identification

The bscan program returns module identification information whether a test is specified or not. Each module has ID information that bscan identifies, then bscan generates the module ID. The module identification test starts bscan on selected modules with no test specified and returns module identification information only. The following command starts the module identification test:

```
bscan -c 1 -d cptester.3100 -f cptester.3100.cfg
```

Figure 1 displays the Module Identification test output.

```
!bscan -c 1 -d cptester.3100 -f cptester.3100.cfg
!
!Location   Mod.Rev    Port  TYPE   SN    Selected
!B001      cp02.3100    32    CP    4       yes
!C001      cm02.2004    36    CM    4       yes
!D001      bs01.1001
!
```

Figure 1. Module Identification Test Output

qport Test

The qport test enables bscan to query all the BS module ports. The following command starts the qport test:

```
bscan -c 1 -t qport
```

Figure 2 displays the qport test output.

```
!bscan -c 1 -t qport
!field 1: port number
!field 3: 13 bit serial number (octal)
!
!qport started on Tue Sep 13 19:49:01 1994
00  **  *****
01  **  *****
02  **  *****
   .
   .
   .
30  **  *****
31  **  *****
32  CP      4
33  **  *****
34  **  *****
35  **  *****
36  CM      4
37  **  *****
38  **  *****
39  **  *****
40  **  *****
41  **  *****
42  **  *****
43  **  *****
44  **  *****
45  **  *****
46  **  *****
47  **  *****
!qport reached maximum pass limit with 1 passes and 0 errors
on Tue Sep 13 19:49:01 1994
```

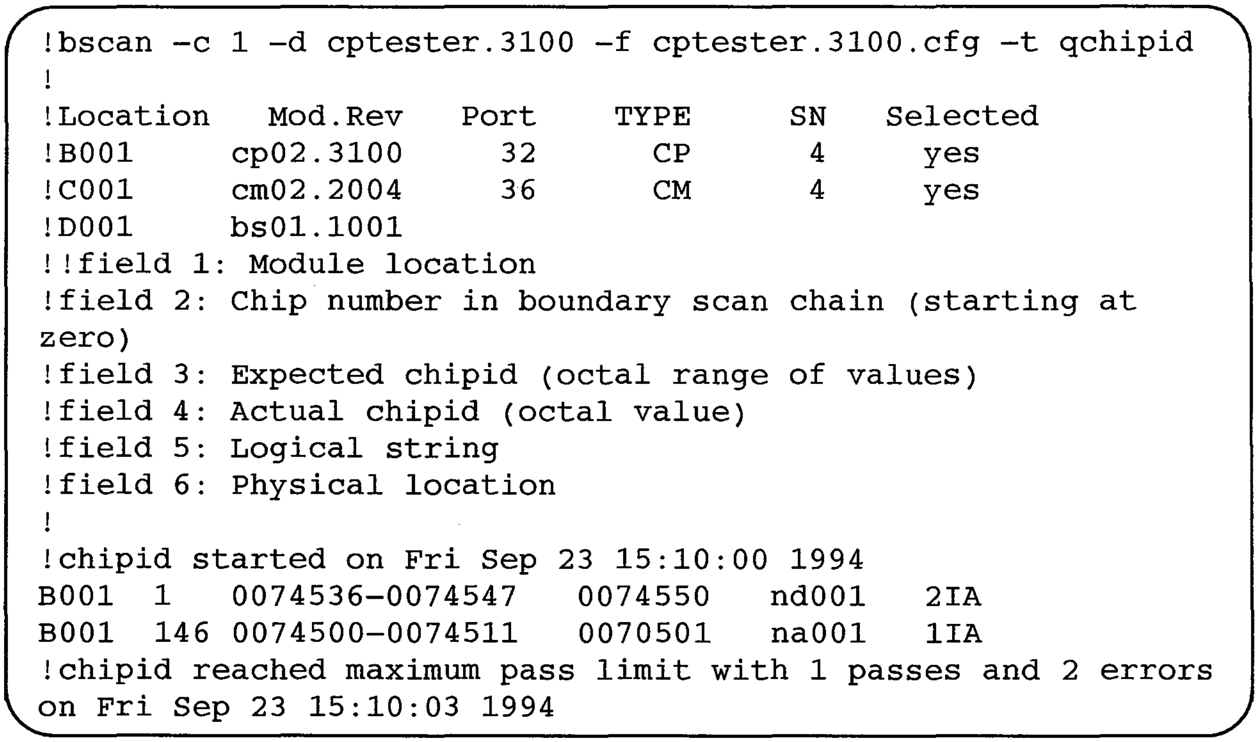
Figure 2. The qport Test Output

chipid Test

The chipid test checks the option identifications (chip IDs) on all module locations. A chip ID is a 16-bit number that identifies the option type. The chipid test detects an error when there is a difference between the expected chip ID range and the actual chip ID. The chipid test checks each module until all modules selected are tested. The following command starts the chipid test:

```
bscan -c 1 -d cptester.3100 -f cptester.301.cfg
-t chipid
```

Figure 3 displays the chipid test output.



```
!bscan -c 1 -d cptester.3100 -f cptester.3100.cfg -t qchipid
!
!Location   Mod.Rev   Port    TYPE     SN   Selected
!B001     cp02.3100    32      CP      4      yes
!C001     cm02.2004    36      CM      4      yes
!D001     bs01.1001
!!field 1: Module location
!field 2: Chip number in boundary scan chain (starting at
zero)
!field 3: Expected chipid (octal range of values)
!field 4: Actual chipid (octal value)
!field 5: Logical string
!field 6: Physical location
!
!chipid started on Fri Sep 23 15:10:00 1994
B001  1   0074536-0074547   0074550   nd001   2IA
B001  146 0074500-0074511   0070501   na001   1IA
!chipid reached maximum pass limit with 1 passes and 2 errors
on Fri Sep 23 15:10:03 1994
```

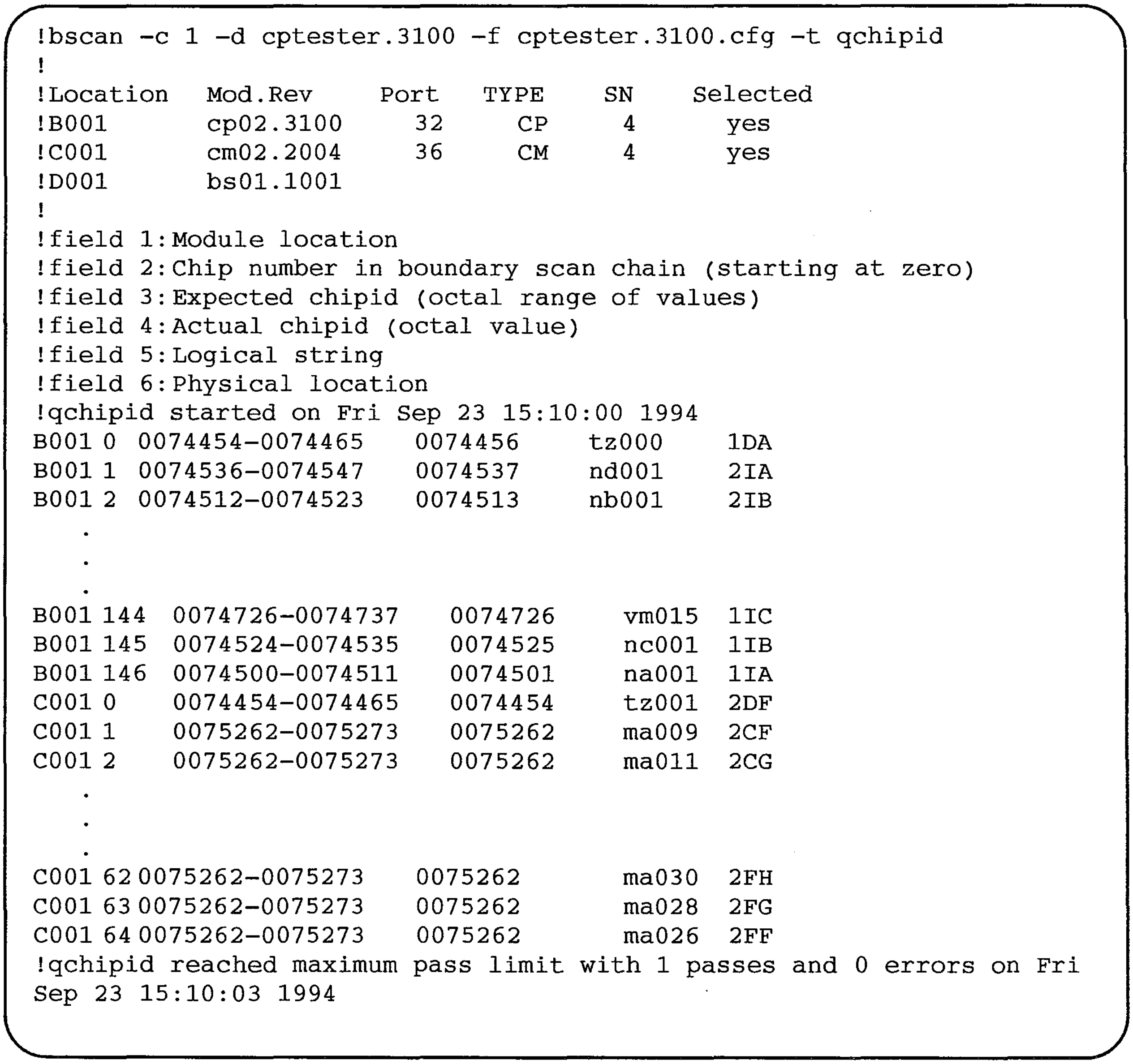
Figure 3. The chipid Test Output

`qchipid` Test

The `qchipid` queries the chip IDs on all selected modules and reports the chip ID locations. The `qchipid` test does not test for differences between the actual and expected chip IDs. The following command starts the `qchipid` test:

```
bscan -c 1 -d cptester.3100 -f cptester.3100.cfg
-t qchipid
```

Figure 4 displays the `qchipid` test output.



```
!bscan -c 1 -d cptester.3100 -f cptester.3100.cfg -t qchipid
!
!Location   Mod.Rev     Port   TYPE    SN    Selected
!B001       cp02.3100    32     CP     4       yes
!C001       cm02.2004    36     CM     4       yes
!D001       bs01.1001
!
!field 1:Module location
!field 2:Chip number in boundary scan chain (starting at zero)
!field 3:Expected chipid (octal range of values)
!field 4:Actual chipid (octal value)
!field 5:Logical string
!field 6:Physical location
!qchipid started on Fri Sep 23 15:10:00 1994
B001 0  0074454-0074465   0074456    tz000    1DA
B001 1  0074536-0074547   0074537    nd001    2IA
B001 2  0074512-0074523   0074513    nb001    2IB
    .
    .
    .
B001 144  0074726-0074737   0074726    vm015  1IC
B001 145  0074524-0074535   0074525    nc001  1IB
B001 146  0074500-0074511   0074501    na001  1IA
C001 0    0074454-0074465   0074454    tz001  2DF
C001 1    0075262-0075273   0075262    ma009  2CF
C001 2    0075262-0075273   0075262    ma011  2CG
    .
    .
    .
C001 62 0075262-0075273   0075262      ma030  2FH
C001 63 0075262-0075273   0075262      ma028  2FG
C001 64 0075262-0075273   0075262      ma026  2FF
!qchipid reached maximum pass limit with 1 passes and 0 errors on Fri
Sep 23 15:10:03 1994
```

Figure 4. The `qchipid` Test Output

shift Test

The shift test verifies the integrity of the boundary scan chain on all selected modules. The shift test detects an error when there is a difference between the data pattern written to a module and the data pattern read from the module. The shift test checks each module until all modules selected have been tested. The following command starts the shift test:

```
bscan -c 1 -d cptester.3100 -f cptester.3100.cfg
-t shift
```

Figure 5 displays the shift test output.

```
!bscan -c 1 -d cptester.3100 -f cptester.3100.cfg -t shift
!
!Location   Mod.Rev   Port    TYPE     SN   Selected
!B001     cp02.3100    32      CP      4      yes
!C001     cm02.2004    36      CM      4      yes
!D001     bs01.1001
!
!field 1: Module location
!field 2: Bit number in boundary scan chain (starting at
zero)
!field 3: Pattern number (starting at zero)
!         Shift patterns: 0000,0377,0125,0252,0000
!field 4: Expected data value (0 or 1)
!
!shift started on Fri Sep 23 17:02:09 1994
C001      0  01  1
C001      1  01  1
C001      2  01  1
!shift reached maximum pass limit with 1 passes and 3 errors
on Fri Sep 23 17:02:11 1994
```

Figure 5. The shift Test Output

scan Test

The scan test verifies the integrity of connections within and between all selected modules. The scan test detects an error when there is a difference between the expected data pattern for a module and the data pattern read from the module. You may run the scan test on any grouping of modules in the mainframe. The default is to test all modules connected in the mainframe. The following command starts the scan test:

```
bscan -c 1 -d cptester.3100 -f cptester.3100.cfg
-t "scan maxpass 2"
```

Figure 6 displays the scan test output.

```
!bscan -c 1 -d cptester.3100 -f cptester.3100.cfg -t scan
maxpass 2             !
!Location   Mod.Rev    Port  TYPE     SN   Selected
!B001     cp02.3100     32    CP      4      yes
!C001     cm02.2004     36    CM      4      yes
!D001     bs01.1001
!
!
!field 1: Module location
!field 2: Bit number in boundary scan chain (starting at
zero)
!field 3: Pattern number (0 - 29)
!field 4: Expected data value (0 or 1)
!
!scan started on Fri Sep 30 05:02:22 1994
B001  19219  02  1
B001  19219  03  0
B001  19219  02  1
B001  19219  03  0
!scan reached maximum pass limit with 2 passes and 4 errors
on Fri Sep 30 05:02:33 1994
```

Figure 6. The scan Test Output

Error Messages

The following list describes the error messages that the bscan program writes to stderr. *Italic* type indicates a variable.

Error Message	Description
bscan: Illegal option *x*	Option *x* is invalid. Correct the option and restart the test with a valid option
bscan: Illegal argument *x*	Argument *x* is invalid. Correct the argument and restart the test with a valid argument.
bscan: IO Channel Open failed on channel *x*	The channel selected (from the command line or the default) cannot be accessed. Verify that you used the correct channel number. If the problem persists, contact your system support staff.
bscan: *directory* : directory can not be found or read.	The directory cannot be found or read. Verify the path to the *directory* and verify that the read, write, and execution permissions are enabled. Correct and rerun the test.
bscan: Need to specify a system name.	A system name must be specified when the default sysdata_dir is used or when the basename of the sysdata_dir is different from the system header file name. Correct and rerun the test.
bscan: Module function failed	A module request to the boundary scan module has failed. Contact your system support staff.
bscan: Channel function failed	A channel function request to the BS module has failed. Contact your system support staff.
bscan: Location or port number invalid on line *n* in file *filename*	Invalid location or port number on line *n* in the configuration file *filename*.

Error Message	Description
`bscan: filename: error in configuration file`	Encountered a bad line in the configuration file. Refer to “Configuration File” on page 4 of this document. Correct the file and rerun the test.
`bscan:` *x*`: location doesn't have port defined.`	Module location *x* specified on the command line does not have a port defined in the configuration file. Correct the file and rerun the test.
`bscan:` *x*`: unable to find location in system.`	Module location *x* specified in the command line is not defined in the system header file. Correct and rerun the test.

Test Completion

A `bscan` test stops under the following conditions:

- A test successfully completes the maximum number of passes (`maxpass` *n*).
- A specified wall-clock time for the test (time *hh*:*mm*:*ss*) elapses.
- You enter the `-h` option on the command line: the `bscan` program writes help information to `stdout` and then terminates.
- The `bscan` program detects an error in the command line entry and writes a message to `stderr`. Only the first error detected is reported.

Exit Codes

The following list describes the exit codes:

Exit Code	Description
0	Successful completion of a `bscan` test.
1	A fatal error occurred (file missing, bad option, etc.).
2	Data errors are detected.

breport Program

Boundary scan report (breport) is a report generator for the boundary scan program (bscan). The breport program takes the error output from the bscan tests and presents it in an easy-to-interpret format. The bscan program logs errors by module location, bit number, pattern number, and expected value. The breport program first sorts the failures and then compares adjacent lines. The breport program condenses the final report by removing the second and succeeding copies of repeated lines for a module location. Pattern numbers are represented in a bit mask in which an uppercase letter P represents a picked bit, an uppercase letter D represents a dropped bit, a lowercase letter p represents an intermittent picking bit, and a lowercase letter d represents an intermittent dropping bit.

breport Command

The breport options can be entered in any order. If an option is omitted, the program uses the default value.

```
breport [-hx] [-d sysdata_dir] [-e err_file] [-m
moddata_dir] [-t test]
```

To start the report generator, enter the following options on the command line:

* **breport**

* Enter the specific command options including the file location.

NOTE: The runbscan shell script executes breport automatically when the program detects a data failure.

breport Command Options

Enter the breport command options in any order. The program uses the default value if you omit an option. You may locate the options in the UNIX online man pages as well as in the man page section of this document.

Examples of `breport`

The `bscan` test results are sent to `stdout`. When `breport` executes, it takes the resulting test data and presents it in an easy-to-interpret format. This example uses the following command sequence:

```
bscan -d cptester.3100 -f cptester.3100.cfg -t
scan maxpass 2
```

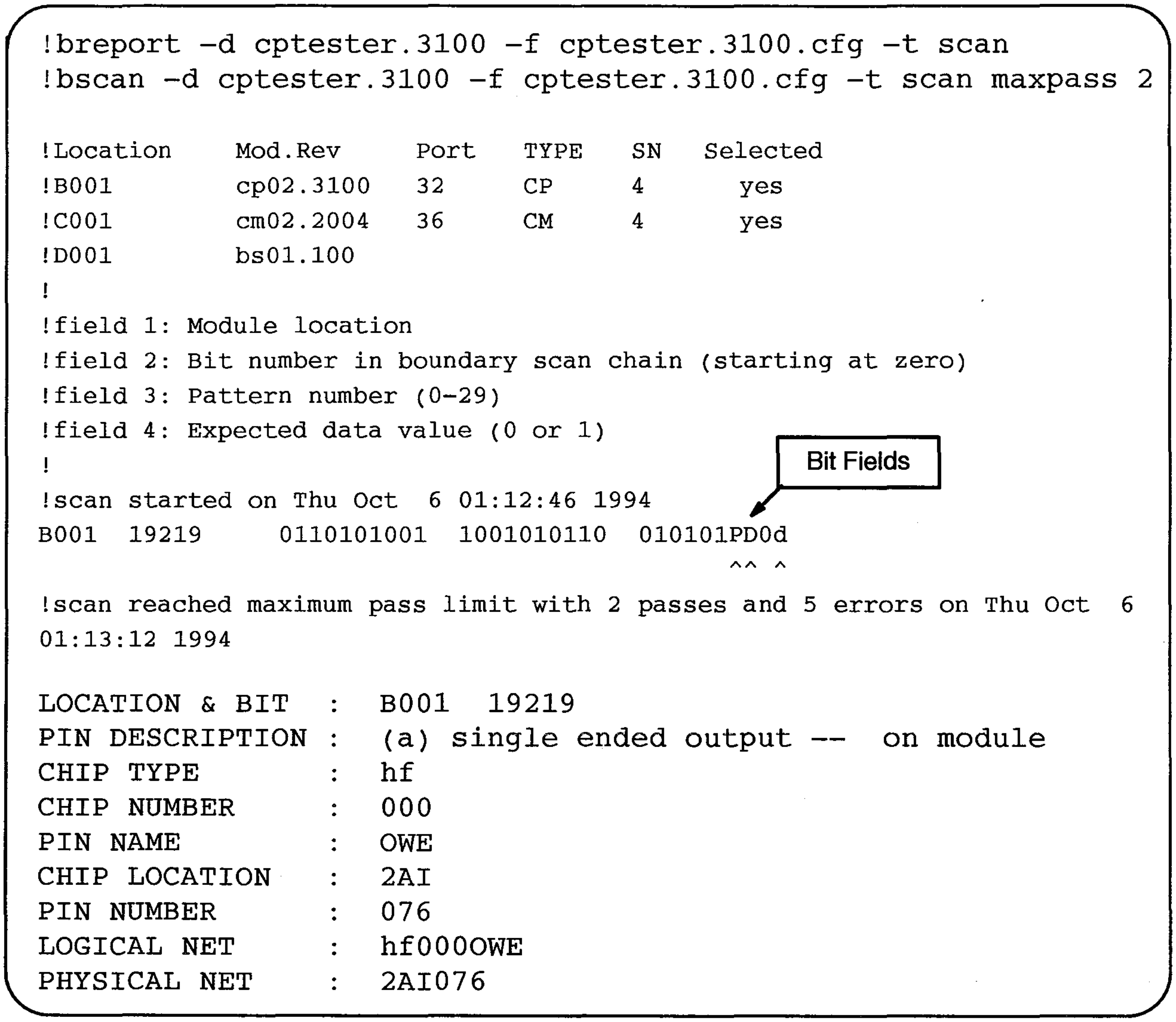
Figure 7 displays the contents of the `scan` test file after completing the `scan` test of the `bscan` program.

```
!Location   Mod.Rev    Port  TYPE  SN  Selected
!B001       cp02.3100  32    CP    4   yes
!C001       cm02.2004  36    CM    4   yes
!D001       bs01.100
!
!field 1: Module location
!field 2: Bit number in boundary scan chain (starting at zero)
!field 3: Pattern number (0-29)
!field 4: Expected data value (0 or 1)
!
!scan started on Thu Oct  6 01:12:46 1994
B001  19219  02  1
B001  19219  03  0
B001  19219  00  1
B001  19219  02  1
B001  19219  03  0
!scan reached maximum pass limit with 2 passes and 5 errors on
Thu Oct  6 01:13:12 1994
```

Figure 7. Example of `scan` Test File Contents

To create the report after the `bscan` program terminates, type in the `breport` command with the required options. For the example created in Figure 7, the following command line was used, which created the file displayed in Figure 8.

```
breport -d cptester.3100 -e cptester.3100.scan
```



```
!breport -d cptester.3100 -f cptester.3100.cfg -t scan
!bscan -d cptester.3100 -f cptester.3100.cfg -t scan maxpass 2

!Location    Mod.Rev     Port   TYPE   SN   Selected
!B001        cp02.3100   32     CP     4      yes
!C001        cm02.2004   36     CM     4      yes
!D001        bs01.100
!
!field 1: Module location
!field 2: Bit number in boundary scan chain (starting at zero)
!field 3: Pattern number (0-29)
!field 4: Expected data value (0 or 1)
!
!scan started on Thu Oct  6 01:12:46 1994
B001  19219     0110101001  1001010110  010101PD0d
                                              ^^ ^
!scan reached maximum pass limit with 2 passes and 5 errors on Thu Oct  6
01:13:12 1994

LOCATION & BIT   :  B001  19219
PIN DESCRIPTION  :  (a) single ended output --  on module
CHIP TYPE        :  hf
CHIP NUMBER      :  000
PIN NAME         :  OWE
CHIP LOCATION    :  2AI
PIN NUMBER       :  076
LOGICAL NET      :  hf000OWE
PHYSICAL NET     :  2AI076
```

Figure 8. Running breport Using the Output from bscan

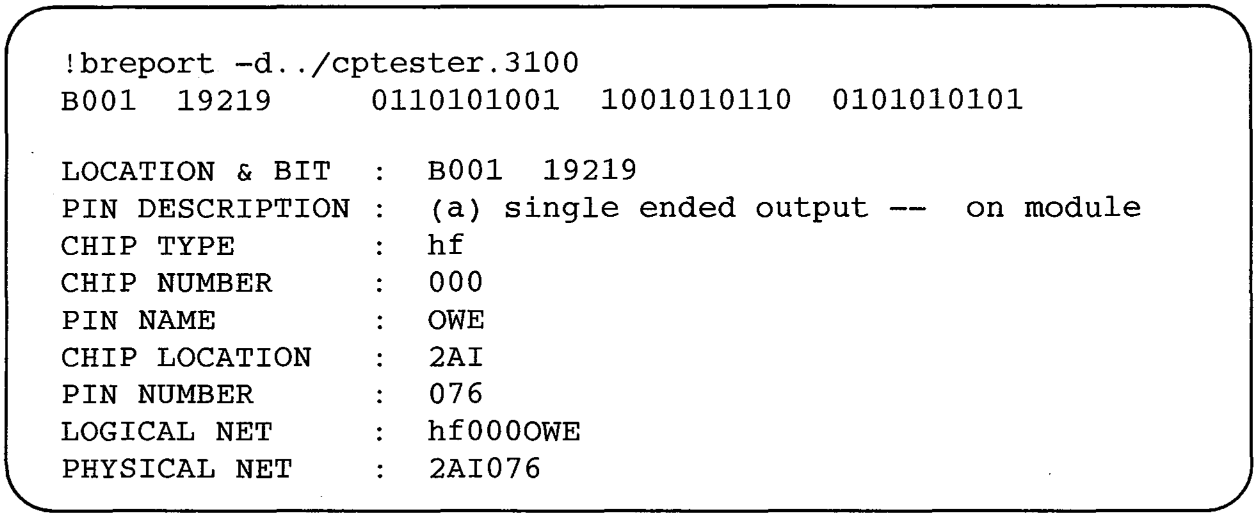
The bit fields to the right of the module location and bit number represent the expected results for all patterns. An uppercase P and D represents a failure on both passes and a lowercase p and d represents a failure on one pass. Pattern 0 is represented by bit = 0, pattern 1 by bit = 1 and so on. In the above example, pattern 0 expected a 1 and received a 0 during one of two passes of bscan. Patterns 2 and 3 dropped and picked during both passes.

The additional information below the bscan termination message is the physical and logical net information from the module.erf file that corresponds to the module type and boundary scan bit number.

The breport program can also be used to retrieve expected data for any module location and bit number. The example in Figure 9 uses the following command sequence:

```
$ echo "B001 19219" | breport -d../cptester.3100
```

Figure 9 displays the result of the above command.



```
!breport -d../cptester.3100
B001  19219     0110101001  1001010110  0101010101

LOCATION & BIT   :  B001  19219
PIN DESCRIPTION  :  (a) single ended output --  on module
CHIP TYPE        :  hf
CHIP NUMBER      :  000
PIN NAME         :  OWE
CHIP LOCATION    :  2AI
PIN NUMBER       :  076
LOGICAL NET      :  hf000OWE
PHYSICAL NET     :  2AI076
```

Figure 9. breport - Retrieve Expected Data

Error Messages

Errors messages are sent to stderr. The following list describes the error messages:

Error Message	Description
`breport: illegal option` *x*	Option *x* is invalid. Correct the option and restart the test.
`breport: Illegal argument` *x*	Argument *x* is invalid. Correct the argument and restart the test.
`breport: directory: directory cannot be found or read.`	Verify that the path to the *directory* is correct and that the read, write, and execute permissions are set properly. Correct and rerun.

Error Message	Description
`breport: Could not find location` *x* `in file` *y*	The module location specified in the input file to `breport` cannot be found in the boundary scan data directory. The `breport` program is most likely being executed with a different data directory than the directory `bscan` was executed on. Correct and rerun.
`breport: Need to specify a system name.`	A system name must be specified when the default `sysdata_dir` is used or when the basename of the `sysdata_dir` is different from the `system header` file name. Correct and rerun.
`breport: bad input line` *n*`, requires location and bit number`	Correct line *n* of input and rerun.

Exit Codes

The following list describes the exit codes:

Exit Code	Description
0	Successful completion of a `bscan` test.
1	A fatal error occurred (file missing, bad option, etc.).
2	Data errors are detected.

bsb Program

NOTE: The bsb will not be included in the initial offline diagnostic release of boundary scan system test. The bsb program does not build the system input module read result vector (.rrv) files.

The bsb program builds the boundary scan data for a defined system configuration. For a given system configuration, bsb generates system specific files such as:

- System numerical interconnect file (.nif)
- Updated read result vector files (.rrv)
- Read data mask file (.rdm)

bsb Command

The bsb program options can be entered in any order. If an option is omitted, the program uses the default value.

```
bsb [-h] [-v] [-p] [-t] [-i dir] [-o dir] [-s dir]
system
```

bsb Command Options

Enter the bsb command options in any order. The program uses the default value if you omit an option. You may locate the command options in the man pages for bsb.

NOTE: The bsb man pages will not be included in the initial release of the *Boundary Scan System Test* documentation.

bsb Files

The bsb program creates the following system data files:

Files	Descriptions
system.sys	Input system configuration file
module.hdr	Input module header information file
module.erf	Input module error reference file

Files	Descriptions
`module.rrv`	Input module read result vector file
`module.rdm`	Input module read data mask file

The `bsb` program generates the following system-specific files:

Files	Descriptions
`system.hdr`	Output system header information file
`system.nif`	Output system numerical interconnect file

`slot.module.rev.xxxx .rrv`	Output system-specific module read result vector where:
	slot = The physical slot (for example, B001)
	module = The two-channel module designated (for example CP)
	rev = The module revision number
	xxxx = A four-digit number

runbscan

The runbscan shell script utility automatically runs the breport program and the bscan program. It enables bscan to run a sequence of tests and provides the text shown in Figure 10. This text tells you if any errors were detected and where you can view the file containing the breport data. The contents of the breport data file are similar to the information shown in Figure 8 on page 18.

```
******** Running setup test
bscan  -d SN7002.1002 -f SN7002.1002.cfg
******** setup test ran successfully

******** Running chipid test
bscan  -d SN7002.1002 -f SN7002.1002.cfg  -t'chipid maxpass 1'
  bscan found data errors, see file /cri/mws/bscan/log/SN7002.1002.chipid
******** chipid test found data errors

******** Running shift test
bscan  -d SN7002.1002 -f SN7002.1002.cfg  -t'shift maxpass 1'
  bscan found data errors, see file /cri/mws/bscan/log/SN7002.1002.shift
  breport data, see file /cri/mws/bscan/log/SN7002.1002.shift.rpt
******** shift test found data errors

******** Running scan test
bscan  -d SN7002.1002 -f SN7002.1002.cfg  -t'scan maxpass 2'
  bscan found data errors, see file /cri/mws/bscan/log/SN7002.1002.scan
  breport data, see file /cri/mws/bscan/log/SN7002.1002.scan.rpt
******** scan test found data errors
```

Figure 10. Output from the runbscan Command

Normally the bscan program terminates when an error is detected, and you must run the breport program to receive an error report. The runbscan shell script automatically executes bscan, and if an error is detected, runbscan uses breport to automatically generate a report.

APPENDIX: BOUNDARY SCAN SYSTEM MAN PAGES

NAME

bscan – Boundary scan system test

SYNOPSIS

/cri/bin/bscan [-c *channel*] [-d *sysdata_dir*] -f *cfg_file* [-h] [-i] [-m *moddata_dir*] [-s *system*] [-t "*test* [time *hh:mm:ss* | maxpass *n*]"] [-AFM] [-v] [*module_loc*...]

IMPLEMENTATION

T90 series

DESCRIPTION

The boundary scan program (bscan) tests the interconnections on a module and between modules, and reports detected errors for a given configuration on a CRAY T90 system. bscan runs until the maximum pass count or wall clock time limit is reached.

The bscan command accepts the following options:

-c *channel*
Selects the channel to use. The default is channel 0.

-d *sysdata_dir*
Specifies the directory in which the system boundary scan data files are located. The system boundary scan data files are generated by the bsb(8) program. The default is the current (.) directory.

-f *cfg_file*
Specifies the configuration file, *cfg_file*, that contains the module location, module type, module revision, and port number for each module in the system. You must specify the configuration file except when executing the -t qport test option.

-h
Generates an online help display containing a synopsis and a brief description of the command options and arguments. The bscan program immediately exits after displaying help information.

-i
Deselects modules in the *module_loc* list and uses all other modules specified in *cfg_file*.

-m *moddata_dir*
Specifies the directory in which the module boundary scan data files are located. The module boundary scan data files are supplied to the site. The default directory is sysdata_dir.

-s *system*
Specifies the system name. The system name is determined by the boundary scan system data generator (bsb(8)) program. The default system name is defined as the base name of the sysdata_dir directory name.

-t *test* Specifies one (only) of the following tests.

qport
Queries each port for module identification information. If the port does not respond, asterisks (*) are used to designate that no module type and module serial number is available. qport ignores all other command line options except -c, -h, and -v.

qchipid
Queries the chip option identification (chip ID). The chip IDs for all chip options on a module are displayed.

`chipid` Tests and displays the chip option identification (chip ID). An error is reported only if the chip ID returned is outside the logical equivalence range. The chip ID logical equivalence is defined as the range of decimal values of the chip ID from X...X0 to X...X9.

`shift` Tests the scan chain with different patterns. Patterns are generated from a byte value and duplicated for the scan chain length. Patterns are written in and read out of each selected module in the system and then are compared.

`scan` Tests the interconnections on a module and between modules.

`time` *hh:mm:ss*

Sets the test execution time in elapsed time (wall-clock). The time is specified in hours (*hh*), minutes (*mm*), and seconds (*ss*); minutes and seconds; or just seconds. Use colons as delimiters, for example: `22:02:44`. This option is used only if the `-t` option is specified. The *test* name and this option must be enclosed in quotation marks.

`maxpass` *n*

Sets the test execution maximum number of passes. The default for *n* is 1. If `time` is set to a value other than 0, the specified option overrides `maxpass`. This option is used only if the `-t` option is specified. The *test* name and this option must be enclosed in quotation marks.

`-A` Writes the system interconnects to `stdout` and exits `bscan`.

`-F` Forces a build of the system `.rdm` files.

`-M` Forces module type and module serial number query off. The default is to return module identification at the start of the test execution.

`-v` Verbose mode. Displays pass and error count after each pass of a test.

module_loc

Specifies a list of module locations to test. Separate location entries with a space, for example, `boo1 boo2`. The default is all locations specified in *cfg_file*.

CAUTIONS

When using the `time` option, the test execution time may exceed the time you specified. Test execution time is checked after each pass of `bscan`. For example, if a test takes 1 minute to complete a pass and `time` is set to 1:01, the test makes 2 passes and completes after 2 minutes.

EXIT STATUS

Exit status is 0 if the test completes successfully with no errors detected. Exit status is 1 if a fatal error occurs (file missing, bad command-line option, and so on), or 2 if data errors are detected.

EXAMPLES

Example 1: The following command prints to `stdout` the module identification information for all boundary scan module ports:

```
bscan -t qport
```

Example 2: The following command prints to `stdout` any errors detected while executing the scan test. The scan test detects an error when a difference occurs between the expected data pattern for a module and the data pattern read from the module. System and module data files are in directory `t4.3100`.

```
bscan -d t4.3100 -f tv.3100.cfg -t "scan maxpass 2"
```

SEE ALSO

`breport`(8) for information on invoking the boundary scan report generator
`bsb`(8) for information on building system data files
`runbscan`(8) for information on invoking the boundary scan shell script

NAME

breport – Report generator for boundary scan system test (bscan)

SYNOPSIS

/cri/bin/breport [-d *sysdata_dir*] [-e *err_file*] [-h] [-m *moddata_dir*] [-s *system*] [-t shift|scan]

IMPLEMENTATION

T90 series

DESCRIPTION

The breport program takes the output from bscan(8) and presents it in an easy-to-interpret format. bscan(8) reports errors by module location, scan chain bit number, pattern number, and expected data value. breport first sorts the failures and then compares adjacent lines. The second and succeeding copies of repeated lines for a module location and bit number are removed. Pattern numbers are represented in a bit mask in which the least significant bit represents pattern 0, and so on. The following conventions are used:

0	Actual or expected data
1	Actual or expected data
P	Picked bit
D	Dropped bit
p	Intermittent picked bit
d	Intermittent dropped bit

The breport command accepts the following options:

-d *sysdata_dir*
Specifies the directory in which the system boundary scan data files are located. The system boundary scan data files are generated by the bsb(8) program. The default is the current (.) directory.

-e *err_file*
Specifies the output file from the bscan(8) program. The output file can be either shift or scan errors.

-h
Generates an online help display containing a synopsis and a brief description of the command options and arguments. The breport program immediately exits after displaying help information.

-m *moddata_dir*
Specifies the directory in which the module boundary scan data files are located. The module boundary scan data files are supplied to the site. The default directory is sysdata_dir.

-s *system*
Specifies the system name. The system name is determined by the boundary scan system data generator (bsb(8)) program. The default system name is defined as the base name of the sysdata_dir directory name.

`-t shift|scan`

Reports failures for scan (the default) or shift data. If the option is not specified, `breport` searches for the input data string `"!scan started"` or `"!shift started"` preceding the failing data. If the string is found `breport` reports failures for the test type specified by the string.

EXIT STATUS

Exit status is 0 for successful completion, 1 for a fatal error (file missing, bad command-line option, and so on).

EXAMPLES

Test results are sent to `stdout`. Error messages are sent to `stderr`.

Example 1: This example uses the following output from `bscan`:

```
!bscan -d cptester.3100 -f cptester.3100.cfg -t scan maxpass 2
!
!Location   Mod.Rev    Port      TYPE    SN        Selected
!B001       cp02.3100   32       CP       4        yes
!C001       cm02.2004   36       CM       4        yes
!D001       bs01.100
!
!field 1: Module location
!field 2: Bit number in boundary scan chain (starting at zero)
!field 3: Pattern number (0-29)
!field 4: Expected data value (0 or 1)
!
!scan started on Thu Oct  6 01:12:46 1994
B001  19219  02  1
B001  19219  03  0
B001  19219  00  1
B001  19219  02  1
B001  19219  03  0
!scan reached maximum pass limit with 2 passes and 5 errors
on Thu Oct  6 01:13:12 1994
```

`breport` is run using the above output from bscan:

```
$ breport -d cptester.3100 -e cptester.3100.scan

!breport -d cptester.3100 -e cptester.3100.scan
!bscan -d cptester.3100 -f cptester.3100.cfg -t scan maxpass 2
!
!Location  Mod.Rev   Port      TYPE    SN        Selected
!B001      cp02.3100   32      CP       4        yes
!C001      cm02.2004   36      CM       4        yes
!D001      bs01.100
!
!field 1: Module location
!field 2: Bit number in boundary scan chain (starting at zero)
!field 3: Pattern number (0-29)
!field 4: Expected data value (0 or 1)
!
!scan started on Thu Oct  6 01:12:46 1994
B001  19219     0110101001  1001010110  010101PD0d
                                              ^^ ^
!scan reached maximum pass limit with 2 passes and 5 errors
on Thu Oct  6 01:13:12 1994

LOCATION & BIT   :  B001  19219
PIN DESCRIPTION  :  (a) single ended output --  on module
CHIP TYPE        :  hf
CHIP NUMBER      :  000
PIN NAME         :  OWE
CHIP LOCATION    :  2AI
PIN NUMBER       :  076
LOGICAL NET      :  hf000OWE
PHYSICAL NET     :  2AI076
```

The bit fields to the right of the module location and bit number represent the expected and actual data for all scan data patterns. Pattern 0 is represented by location 2^0, pattern 1 by 2^1, and so on. From the above example pattern 0 expected a 1 and received a 0 during one of two passes of bscan. Patterns 2 and 3 dropped and picked bits during both passes.

The additional information below the bscan termination message is the physical and logical net information from the module .erf file corresponding to the module type and boundary scan bit number.

Example 2: breport can also be used to retrieve expected data for any module location and bit number.

```
$ echo "B001 19219" | breport -d../cptester.3100
!breport -d../cptester.3100
B001  19219     0110101001  1001010110  0101010101


LOCATION & BIT   :  B001  19219
PIN DESCRIPTION  :  (a) single ended output --  on module
CHIP TYPE        :  hf
CHIP NUMBER      :  000
PIN NAME         :  OWE
CHIP LOCATION    :  2AI
PIN NUMBER       :  076
LOGICAL NET      :  hf000OWE
PHYSICAL NET     :  2AI076
```

DATA FILES

breport uses two types of data files: module and system. Module data files are supplied to the site and cannot be generated locally. System data files are generated from module data files and can be generated on site using the bsb(8) program.

SEE ALSO

bscan(8) for information on invoking the boundary scan program
bsb(8) for information on building system data files
runbscan(8) for information on invoking the boundary scan shell script

The boundary scan system data generator (bsb) man pages are not available.

Reader Comment Form

Title: Boundary Scan System Test
Preliminary Information

Number: HDM-xxx-0
December 1994

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)

______Too much information ____

______Too little information ____

______Just the right amount of information

Your comments help Hardware Publications and Training improve the quality and usefulness of your publications. Please use the space provided below to share your comments with us. When possible, please give specific page and paragraph references. We will respond to your comments in writing within 48 hours.

NAME ____
JOB TITLE ____________________________________
FIRM ____
ADDRESS ______________________________________
CITY ____________________ STATE _________ ZIP __________
DATE ____

[or attach your business card]

CRAY
RESEARCH, INC.

CUT ALONG THIS LINE

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO 6184 ST. PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY
RESEARCH, INC.

Attn: Hardware Publications and Training
890 Industrial Boulevard
Chippewa Falls, WI 54729

Fold

STAPLE

```
#! /bin/sh

# script to run boundary scan tests (chipid, shift, scan)
# if data errors are found the report generator will be run
# Don Heian  Sept 17, 1994
#

NAME=$0
PROG='bscan'
REPORT='breport'
RM=/bin/rm
RET_STATUS=0

# The cfg file and data directory must reside in BSCAN_DIR!
BSCAN_DIR="$HOME/bscan"       # directory where cfg file and data dir resides
ERR_DIR="$BSCAN_DIR/bloginfo" # directory where error files reside
BIN_DIR=$HOME/bin             # directory $PROG and $REPORT binaries reside
RUNLOG=$ERR_DIR/runlog        # file containing a log of what happened

# Add the boundary scan programs to the search path
PATH=$BIN_DIR:$PATH
export PATH

# To create long version of this shell script, simply rename this
# version with a ".long" extension and link.
# For example: $ ln program program.long
if [ -z "`echo $NAME | sed -n 's?.*\(\.long\)$?\1?p'`" ]; then
    PASSCNT=2
else
    PASSCNT=100
fi
echo
echo
echo SCAN WILL MAKE    $PASSCNT    passes.


CHAN=

#
# function dotest() - this routine executes the $PROG with the desired
# $TESTNAME (chipid, shift or scan).  Error files are placed in
# $ERR_DIR.  Empty (0 size) stderr files are removed.  If data errors
# are found the $REPORT program is run.
#
dotest()
{
eval echo \""$HEADER\""
echo "$CMDLN"

eval $CMDLN $* >$ERR_DIR/${SYSTEM}.${TESTNAME} 2>$ERR_DIR/${SYSTEM}.${TESTNAME}.stderr
EXIT=$?

# Exit code of
#   0 - pass
#   1 - fatal error running program
#   2 - data error(s)
#   * - undefined (report to author of program!)
case "$EXIT" in
0) eval echo \""$TAIL\"";;
1) echo "  Fatal error executing $PROG, see file $ERR_DIR/${SYSTEM}.${TESTNAME}.stderr:" |
 tee -a $RUNLOG
   cat $ERR_DIR/${SYSTEM}.${TESTNAME}.stderr | tee -a $RUNLOG
   RET_STATUS=1
   eval echo \""$EXEC\""
   exit 1;;
1) echo "  $PROG found data errors, see file $ERR_DIR/${SYSTEM}.${TESTNAME}" | tee -a $RUN
LOG
   if [ "$TESTNAME" != "chipid" ]; then
       eval echo \""  $REPORT data, see file $ERR_DIR/${SYSTEM}.${TESTNAME}.rpt\"" | tee -
a $RUNLOG
       eval $REPORT -d $DATA_DIR -e $ERR_DIR/${SYSTEM}.${TESTNAME} \
       >$ERR_DIR/${SYSTEM}.${TESTNAME}.rpt \
       2>$ERR_DIR/${SYSTEM}.${TESTNAME}.rpt.stderr
   fi
   RET_STATUS=1
   eval echo \""$ERROR\"";;
*) echo "Illegal exit code <$EXIT>"
   eval echo \""$ERROR\""
   exit 1;;
esac

# clean up stderr files that are zero bytes
if [ ! -s "$ERR_DIR/${SYSTEM}.${TESTNAME}.stderr" ]; then
    eval $RM -f "$ERR_DIR/${SYSTEM}.${TESTNAME}.stderr"
fi
if [ ! -s "$ERR_DIR/${SYSTEM}.${TESTNAME}.rpt.stderr" ]; then
    eval $RM -f "$ERR_DIR/${SYSTEM}.${TESTNAME}.rpt.stderr"
fi
}

HEADER='
******** Running ${TESTNAME} test'

TAIL='\
******** ${TESTNAME} test ran successfully'

ERROR='\
******** ${TESTNAME} test found data errors'

EXEC='\
******** ${TESTNAME} test found execution error'

USAGE="Usage: $NAME [-hi] [-c#] system [module_loc...]"
HELP="$NAME - Run boundary scan tests

$USAGE

    where
        -c#        : select channel number (default: 0)
        -h         : display this help information
        -i         : invert module_loc list (don't run this list of modules)
        system     : boundary scan system name
        module_loc: list of module locations to test (defaults to all) "

while getopts c:hi CHAR; do
    case $CHAR in
    c) CHAN="-c $OPTARG";;
    h) echo "$HELP"; exit 1;;
    i) IFLAG="-i";;
    \?) echo $USAGE; exit 1;;
    esac
done
shift `expr $OPTIND - 1`

#check for system name specified on command line
if [ $# -lt 1 ]; then
   echo "$NAME: No system name specified."
   echo $USAGE
   exit 1
fi

SYSTEM=$1
DATA_DIR=$SYSTEM      # system and data directory name are the same
shift

# check if the BSCAN_DIR exists
cd $BSCAN_DIR 2>/dev/null 1>&2
if [ "$BSCAN_DIR" != "`pwd`" ]; then
    echo "Cannot cd (change directory) to $BSCAN_DIR"
    exit 1
fi

# check if the data directory exists
if [ ! -d $DATA_DIR ]; then
    echo "$NAME: $DATA_DIR: Doesn't exist or isn't not a directory."
    echo $USAGE
    exit 1
fi

# check if cfg file exists
if [ ! -f ${SYSTEM}.cfg ]; then
    echo "$NAME: ${SYSTEM}.cfg: Filename does not exist."
    echo "${SYSTEM}.cfg must reside in the $BSCAN_DIR directory"
    echo $USAGE
    exit 1
fi

# check if $PROG file exists
type "$PROG" 2>/dev/null 1>&2
if [ 0 -ne "`echo $?`" ]; then
    echo "$NAME: ${PROG}: Filename does not exist, check PATH variable"
    echo $USAGE
    exit 1
fi

# check if $REPORT file exists
type "$REPORT" 2>/dev/null 1>&2
if [ 0 -ne "`echo $?`" ]; then
    echo "$NAME: ${REPORT}: Filename does not exist, check PATH variable"
    echo $USAGE
    exit 1
fi

# Truncate log file to last 400 lines
ed - $RUNLOG <<\End 2>/dev/null 1>&2
1,$-400d
w
q
End

CMDLN="$PROG $CHAN -d $DATA_DIR -f ${SYSTEM}.cfg $IFLAG maxpass $PASSCNT $*"
echo "`date`: $CMDLN" >> $RUNLOG  # record run in logfile

# define CMDLN here as positonal parameters are lost in function dotest

# check for correct setup (port number, io, etc before running tests)
TESTNAME="setup"
CMDLN="$PROG $CHAN -d $DATA_DIR -f ${SYSTEM}.cfg $IFLAG $*"
dotest

TESTNAME="chipid"
```

```
CMDLN="$PROG $CHAN -d $DATA_DIR -f ${SYSTEM}.cfg $IFLAG \
-t'${TESTNAME} maxpass 1' $*"
dotest

TESTNAME="shift"
CMDLN="$PROG $CHAN -d $DATA_DIR -f ${SYSTEM}.cfg $IFLAG \
-t'${TESTNAME} maxpass 1' $*"
dotest

TESTNAME="scan"
CMDLN="$PROG $CHAN -d $DATA_DIR -f ${SYSTEM}.cfg $IFLAG \
-t'${TESTNAME} maxpass $PASSCNT' $*"
dotest

exit $RET_STATUS
```

```
!breport -d tester003.3100 -e /home/ted2/bstest/bscan/bloginfo/tester003.3100.sc
!bscan -c 1 -d tester003.3100 -f tester003.3100.cfg -tscan maxpass 2
!
!Location  Mod.Rev    Port      TYPE    SN        Selected
!B001      cp02.3100    24      CP       06       yes
!C001      cm02.2004    36      CM      017       yes
!D001      bs01.1001
!
!field 1: Module location
!field 2: Bit number in boundary scan chain (starting at zero)
!field 3: Pattern number (0-29)
!field 4: Expected data value (0 or 1)
!
!scan started on Sun Oct 23 10:59:24 1994
B001  19219     0110101001  1001010110  010101PD01
                                              ^^

B001  23345     1001010110  1010P11001  0101100101
                                ^

B001  23382     P1P11PP11P  1PP11PP1P1  1PP11PP1P1
                ^ ^  ^^  ^   ^^  ^^ ^    ^^  ^^ ^

B001  23383     P1P11PP11P  1PP11PP1P1  1PP1P11PP1
                ^ ^  ^^  ^   ^^  ^^ ^    ^^ ^  ^^

B001  24656     1001010110  1010011001  010101P101
                                              ^

B001  24657     1001010110  1010011001  0101011P01
                                               ^

B001  32618     1001100110  1010010110  010101P101
                                              ^

C001   9774     P1P11P1PP1  P1P11PP11P  1P1P1P1PP1
                ^ ^  ^ ^^   ^ ^  ^^  ^   ^ ^ ^ ^^

!scan reached maximum pass limit with 2 passes and 100 errors on Sun Oct 23 10:5

LOCATION & BIT  :  B001  19219
PIN DESCRIPTION :  (a) single ended output --  on module
CHIP TYPE       :  hf
CHIP NUMBER     :  000
PIN NAME        :  OWE
CHIP LOCATION   :  2AI
PIN NUMBER      :  076
LOGICAL NET     :  hf000OWE
PHYSICAL NET    :  2AI076

LOCATION & BIT  :  B001  23345
PIN DESCRIPTION :  (j) single ended input -- on module
CHIP TYPE       :  ha
CHIP NUMBER     :  003
PIN NAME        :  IEE
CHIP LOCATION   :  2AH
PIN NUMBER      :  051
LOGICAL NET     :  ha003OMDha003IEE
PHYSICAL NET    :  2AH1202AH051

LOCATION & BIT  :  B001  23382
PIN DESCRIPTION :  (j) single ended input -- on module
CHIP TYPE       :  ha
CHIP NUMBER     :  003
PIN NAME        :  IAB
CHIP LOCATION   :  2AH
PIN NUMBER      :  003
LOGICAL NET     :  ch001OLBha003IAB
PHYSICAL NET    :  2GG2172AH003

LOCATION & BIT  :  B001  23383
PIN DESCRIPTION :  (j) single ended input -- on module
CHIP TYPE       :  ha
```

P or D is bad
↓ pick ↓ drop

capital letters mea
problem on both pa

small letter means
problem on 1 pass

```
CHIP NUMBER      :  003
PIN NAME         :  IAA
CHIP LOCATION    :  2AH
PIN NUMBER       :  002
LOGICAL NET      :  ch001OLAha003IAA
PHYSICAL NET     :  2GG2182AH002

LOCATION & BIT   :  B001  24656
PIN DESCRIPTION  :  (j) single ended input -- on module
CHIP TYPE        :  hb
CHIP NUMBER      :  000
PIN NAME         :  IGJ
CHIP LOCATION    :  1AA
PIN NUMBER       :  375
LOGICAL NET      :  ha003OMFhb000IGJ
PHYSICAL NET     :  2AH1221AA375

LOCATION & BIT   :  B001  24657
PIN DESCRIPTION  :  (j) single ended input -- on module
CHIP TYPE        :  hb
CHIP NUMBER      :  000
PIN NAME         :  IGI
CHIP LOCATION    :  1AA
PIN NUMBER       :  374
LOGICAL NET      :  ha003OMEhb000IGI
PHYSICAL NET     :  2AH1211AA374

LOCATION & BIT   :  B001  32618
PIN DESCRIPTION  :  (j) single ended input -- on module
CHIP TYPE        :  ch
CHIP NUMBER      :  002
PIN NAME         :  IGF
CHIP LOCATION    :  1DG
PIN NUMBER       :  301
LOGICAL NET      :  cj000OCKch002IGF
PHYSICAL NET     :  1CI0661DG301

LOCATION & BIT   :  C001  9774
PIN DESCRIPTION  :  (J) single ended input -- off module
CHIP TYPE        :  mf
CHIP NUMBER      :  012
PIN NAME         :  IAM
CHIP LOCATION    :  1CA
PIN NUMBER       :  027
LOGICAL NET      :  za006OBMmf012IAM
PHYSICAL NET     :  1YB0131CA027
```