Boundary Scan System Test

(CRAY T90™ Series)

HDM-117-B

Cray Research/Silicon Graphics Proprietary

Record of Revision

REVISION	DESCRIPTION
	August 1995. Original printing. This printing corresponds to the MT-T2.1.0 diagnostic release.
A	March 1996. This revision corresponds to the MT-T2.2.0 offline diagnostic release.
B	October 1997. This revision corresponds to the MT-T2.3.0 offline diagnostic release. This revision reorganizes the document to present information from the perspective of the user.

Any shipment to a country outside of the United States requires a letter of assurance from Cray Research, Inc.

Requests for copies of Cray Research, Inc. publications should be directed to:

CRAY RESEARCH, INC.
Customer Service Logistics
1100 Lowater Road
P.O. Box 4000
Chippewa Falls, WI 54729-0078
USA

Comments about this publication should be directed to:

CRAY RESEARCH, INC.
Service Publications and Training
890 Industrial Blvd.
P.O. Box 4000
Chippewa Falls, WI 54729-0078
USA

BOUNDARY SCAN SYSTEM TEST

Figures

Figures (continued)

Tables

Notational Conventions

This document uses the following notational conventions:

- `Courier` type indicates directory pathnames, filenames, program names, commands, and screen output. This notational convention is not used in headings or the table of contents.

- **`Courier bold`** type indicates commands that you should enter. This notational convention is not used in headings or the table of contents.

- *Italic* type indicates a variable. This notational convention is not used in headings or the table of contents.

- The following conventions are used in the command descriptions:

  - Square brackets [ ] indicate an optional entry.
  - A vertical bar | indicates a choice.

- All directories in this document that do not start with slash (`/`) are relative to `/cri/cme/t32/` for a maintenance workstation (MWS) and `/opt/CYRIdiag/t32/` for a system workstation (SWS).

  For example, `rel/bscan/misc/equiv/` is `/cri/cme/t32/rel/bscan/misc/equiv/` on an MWS and `/opt/CYRIdiag/t32/rel/misc/equiv/` on an SWS.

Boundary Scan System Test Overview

The boundary scan system test checks the interconnections on and between modules, logs any detected errors, and provides a status report for CRAY T90™ series mainframes. The boundary scan system test cannot test the following types of interconnections:

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

For CRAY T94™ and CRAY T916™ mainframes, the boundary scan system test requires complete control of the mainframe; therefore, you cannot run a test and the operating system (OS) simultaneously. For CRAY T932™ mainframes, you can configure the mainframe to run tests in one half of the mainframe while the OS runs in the other half of the mainframe.

Boundary Scan System Test Software Components

Table 1 describes the software components of the boundary scan system test.

Table 1. Boundary Scan System Test Software Components

Component	Description
`runbscan` shell script	Automates boundary scan system testing: This shell script automatically runs a sequence of the `bscan` tests and then runs the `breport` program to generate formatted output. The `runbscan` shell script is the recommended way to use the boundary scan system test. Refer to “runbscan Utility” on page 18 for more information.
`bsb` program	Creates data files that list the system interconnections between modules and their expected scan values for a given CRAY T90 series system configuration: This program provides the necessary data that the `bscan` program needs to test the mainframe. Refer to “bsb Program” on page 54 for more information.
`bscan` program	Tests the interconnections on a module and between modules and reports detected errors for a given CRAY T90 series system configuration: This program enables you to verify the integrity of the mainframe after a failure causes the OS to go down or after you complete a repair procedure. Refer to “bscan Program” on page 54 for more information.

Table 1. Boundary Scan System Test Software Components (continued)

Component	Description
`breport` program	Presents the output from `bscan` in an easy-to-read format: This program takes a file that contains the output from the `bscan shift` or `scan` tests and sorts the data and then removes duplicate lines to create easy-to-read output. Refer to "breport Program" on page 54 for more information.
`bsplot` program	Uses the output from the `breport` program to create a graphical map of any errors found by the `scan` test. Refer to "bsplot Program" on page 55 for more information.
`railplot` program	Uses the output from the `scan` test to create a graphical representation of faulty electronic zero insertion force (EZIF) connector contacts. Refer to "railplot Program" on page 59 for more information.

Because the `runbscan` script automates testing, you should normally use the `runbscan` script to perform boundary scan system testing (instead of manually running the individual programs).

Viewing Man Pages Online

To view the online manual (man) pages for `bsb`, `bscan`, `breport`, or `runbscan`, enter the `man` command followed by the program name at the UNIX® command prompt. For example, to view the `bscan` program man page, enter:

```
man bscan
```

Boundary Scan System Test Directories and Files

Figure 1 shows the directory structure that the boundary scan system test uses. The remainder of this section describes the directories and files.

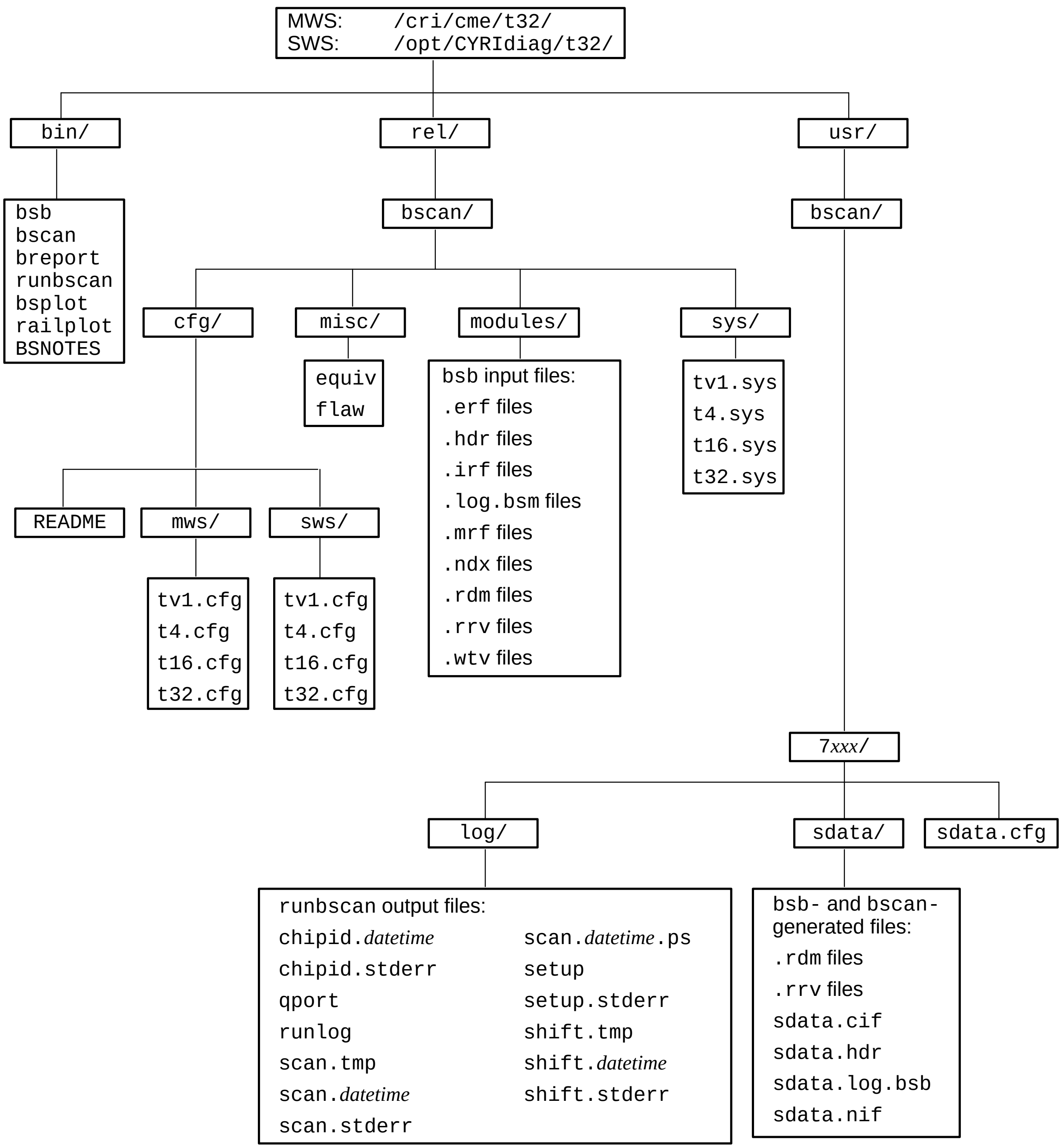

Figure 1. Boundary Scan System Test Directory Structure

The base directory on an MWS is `/cri/cme/t32/`. The base directory on an SWS is `/opt/CYRIdiag/t32/`. The boundary scan system uses the following main directory structures under the base directories:

- `bin/`
- `rel/bscan/`
- `usr/bscan/`

bin/ Directory

The `bin/` directory contains the executable files for the boundary scan system test.

Table 2. `bin/` Directory Contents

File	Description
`breport`	Executable file for the `breport` program
`bsb`	Executable file for the `bsb` program
`bscan`	Executable file for the `bscan` program
`BSNOTES`	Text file that contains the boundary scan system test release notes for the current diagnostic release
`bsplot`	Executable file for the `bsplot` program
`railplot`	Executable file for the `railplot` program
`runbscan`	Executable shell script for the `runbscan` utility

rel/bscan/ Directory Structure

The `rel/bscan/` directory structure contains information that the boundary scan system test needs to run.

The diagnostic release installation script creates all information in this directory structure; you cannot modify any files in the `rel/bscan/` directory structure.

cfg/ Directory

The `cfg/` directory in the `rel/bscan/` directory includes sample configuration files for the different types of mainframes.

README File

The `README` file in the `cfg/` directory contains the following text:
`# Sample default configuration files for a tv1, t4, t16 and t32 system`.

mws/ Directory

The `mws/` directory in the `cfg/` directory contains sample configuration files for systems that use an MWS. There are sample configuration files for test vehicles (`tv1.cfg`), CRAY T94 mainframes (`t4.cfg`), CRAY T916 mainframes (`t16.cfg`), and CRAY T932 mainframes (`t32.cfg`).

The `runbscan -setup` command uses these files to create the initial `sdata.cfg` file in the `usr/bscan/7`*xxx*`/` directory.

You can also manually use one of these samples for your system by copying the file to the `usr/bscan/7`*xxx*`/` directory and renaming it as `sdata.cfg`.

sws/ Directory

The `sws/` directory in the `cfg/` directory contains sample configuration files for systems that use an SWS. There are sample configuration files for test vehicles (`tv1.cfg`), CRAY T94 mainframes (`t4.cfg`), CRAY T916 mainframes (`t16.cfg`), and CRAY T932 mainframes (`t32.cfg`).

The `runbscan -setup` command uses these files to create the initial `sdata.cfg` file in the `usr/bscan/7`*xxx*`/` directory.

You can also manually use one of these samples for your system by copying the file to the `usr/bscan/7`*xxx*`/` directory and renaming it as `sdata.cfg`.

misc/ Directory

The `misc/` directory in the `rel/bscan/` directory contains the `equiv` and `flaw` files.

The `equiv` file contains a list of equivalent chip IDs for specific locations on modules.

The `flaw` file contains a list of `bscan` errors that can be flawed out because they do not affect operation of the mainframe.

modules/ Directory

The `modules/` directory in the `rel/bscan/` directory contains the data files for all types of modules. The `bsb` application uses the module data files to generate the system data files that the `bscan` program needs.

Module data files contain specific information for each type of module. These data files are supplied to the site in the diagnostic release: You cannot generate these files on-site.

Table 3 describes the contents of the `module/` directory.

Table 3. `module/` Directory Contents

File Extension	Description
`.erf`	Module error reference file
`.hdr`	Module header information file
`.irf`	Module interconnect reference file
`.log.bsm`	Runtime log file (generated by the `bsm` program)
`.mrf`	Miscellaneous reference file that contains interconnect references that do not fit in the `.erf` and `.irf` files
`.ndx`	Module error reference index file
`.rdm`	Module read data mask file

Table 3. `module/` Directory Contents (continued)

File Extension	Description
`.rrv`	Module read result vector file
`.wtv`	Module write test vectors

sys/ Directory

The `sys/` directory in the `rel/bscan/` directory contains system description files that describe the types of systems that are available.

The `bsb` program uses the system description files to generate the system data files that the `bscan` program needs. The `tv1.sys`, `t4.sys`, `t16.sys`, and `t32.sys` files describe the module placement and connector-to-connector connections in each type of system.

usr/bscan/ Directory Structure

The `usr/bscan/` directory structure contains information that is generated by the boundary scan system tests and information that you can modify.

7xxx/ (Serial Number) Directory

The `7`*xxx*`/` directory in the `usr/bscan/` directory contains the files for a specific mainframe. The actual directory name is the serial number of the mainframe; for example, the directory is `7001/` for the mainframe with serial number 7001.

log/ Directory

The `log/` directory in the `7`*xxx*`/` directory contains files that are generated by the `bscan` program, `breport` program, `runbscan` shell script, and `bsplot` program. Table 4 describes the contents of the `log/` directory.

Table 4. `log/` Directory Contents

File	Description
`chipid.`*datetime*	`stdout` from running `bscan` with the `chipid` test selected
`chipid.stderr`	`stderr` from running `bscan` with the `chipid` test selected
`qport`	`stdout` from running `bscan` with the `qport` test selected
`runlog`	Log file that contains information about when `runbscan` was executed and where any detected errors were located
`scan.tmp`	`stdout` from running `bscan` with the `scan` test selected
`scan.`*datetime*	`stdout` from running `breport` with a `scan.tmp` file as input
`scan.stderr`	`stderr` from running `bscan` with the `scan` test selected
`scan.`*datetime*`.ps`	PostScript output file from the `bsplot` program with a `scan.`*datetime* file as input
`setup`	`stdout` from running `bscan` with no test selected
`setup.stderr`	`stderror` from running `bscan` with no test selected

Table 4. `log/` Directory Contents (continued)

File	Description
`shift.tmp`	`stdout` from running `bscan` with the `shift` test selected
`shift.`*datetime*	`stdout` from running `breport` with a `shift.tmp` file as input
`shift.stderr`	`stderr` from running `bscan` with the `shift` test selected

sdata/ Directory

The `sdata/` directory in the `7`*xxx*`/` directory contains various system data files.

The `bsb` program generates the system data files from the `sdata.cfg` configuration file, the module data files, and the system description file.

The `bscan` program generates system data mask files (`.rdm` files) at run time from the module data files and system data files. The system data mask files contain data masks that specify which bits in the boundary scan chain (BSC) `bscan` should test. The modules that are specified in the configuration file determine the contents of the system data mask files. The `bscan` program generates system data mask files if either of the following conditions occurs:

- A system data mask file is missing
- A new configuration of modules is being tested

Table 5 describes the `bsb`-generated and `bscan`-generated files in the `sdata/` directory.

Table 5. `sdata/` Directory Contents

File	Description
`sdata.err.bsb`	System error report file
`.rdm`	System-specific module read data mask file
`.rrv`	System-specific module read result vector file
`sdata.cif`	System connection information file
`sdata.hdr`	System header information file
`sdata.log.bsb`	System log report file
`sdata.nif`	System numerical interconnect file

sdata.cfg (Configuration File)

The `sdata.cfg` file in the `7`*xxx*`/` directory is the configuration file, which provides the boundary scan system test with the information necessary to test modules in a system. The configuration file specifies the modules for the specific system that you will test.

Locations that are not specified in the configuration file are not tested. Locations specified in the configuration file that are not in the *system*.`hdr` file, or port numbers that are out of range, generate an error message and terminate the boundary scan system test.

Figure 2 shows an example configuration file. You can use the `vi` editor to change the configuration file.

```
mws7001$ cd usr/bscan/7001
mws7001$ cat sdata.cfg
# USMID @(#)mtt220/cme/t32/src/bscan/cfg/mws/t4.cfg     10.1    01/30/97
14:31:31
#    t4.cfg
#
# Set TEST equal to 0 to ignore module
# Using the pound sign (#) to ignore the module entry
# will cause the configuration to rebuild unnecessarily.
# Note: Do NOT delete or add lines to this file.

CHASSIS=t4

LOC=A001 MODREV=io01.1013 PORT=05 TEST=1

LOC=B001 MODREV=cpe1.1005 PORT=36 TEST=1
LOC=B002 MODREV=cpe1.1005 PORT=37 TEST=1
LOC=B003 MODREV=cpe1.1005 PORT=38 TEST=1
LOC=B004 MODREV=cpe1.1005 PORT=39 TEST=1

LOC=C001 MODREV=sr01.1007 PORT=04 TEST=1

LOC=C002 MODREV=cm02.2100 PORT=23 TEST=1
LOC=C003 MODREV=cm02.2100 PORT=22 TEST=1
LOC=C004 MODREV=cm02.2100 PORT=21 TEST=1
LOC=C005 MODREV=cm02.2100 PORT=20 TEST=1
```

Figure 2. Sample Configuration File

The following conditions apply to configuration files:

- Either uppercase or lowercase letters are acceptable.
- Extra spaces and tabs are ignored.
- Lines that begin with a # sign are ignored by the `bscan` program.
- The first uncommented line must include the key word `CHASSIS`, which must be set to the chassis type of the hardware (`tv1`, `t4`, `t16`, or `t32`).
- Each line must contain the following key words set to the values for the system that you are testing: `LOC` (location), `MODREV` (module type and revision number), `PORT` (port number), and `TEST` (0 = do not test the module, 1 = test the module).

  Ensure that you enter the keywords in the sequence shown in Figure 2. If any of these parameters change, you must update them in the configuration file.

- You can obtain a list of valid `MODREV` values with the command sequence shown in the following example (from an MWS):

```
mws7001$ cd /cri/cme/t32/rel/bscan/modules
mws7001$ ls *.hdr | sed ’s/.hdr$//’
cm02.2100
cm03.3007
cm04.2001
cp02.3100
cpe1.1005
io01.1013
io02a.1005
io02b.1005
io02c.1005
nw01.1005
si01.1006
sj01.1002
sr01.1007
```

Most examples in this document use system data and configuration files for a CRAY T94 system (t4 chassis).

NOTE: IO02 modules have three boundary scan chains: io02a (BS section), io02b (I/O section), and io02c (GigaRing™ I/O section). Figure 3 shows a configuration file with io02b and io2c included; you cannot test the io02a chain in the field.

```
sws7033$ cd usr/bscan/7033
sws7033$ cat sdata.cfg
#    t4.cfg
#
# Set TEST equal to 0 to ignore module
# Using the pound sign (#) to ignore the module entry
# will cause the configuration to rebuild unnecessarily.
# Note: Do NOT delete or add lines to this file.

CHASSIS=t4
LOC=A001 MODREV=io02b.1005 PORT=36 TEST=1
LOC=A001 MODREV=io02c.1005 PORT=37 TEST=1

LOC=B001 MODREV=cpe1.1005 PORT=00 TEST=1
LOC=B002 MODREV=cpe1.1005 PORT=04 TEST=0
LOC=B003 MODREV=cpe1.1005 PORT=08 TEST=0
LOC=B004 MODREV=cpe1.1005 PORT=12 TEST=0

LOC=C001 MODREV=sr01.1007 PORT=16 TEST=1

LOC=C002 MODREV=cm03.3007 PORT=20 TEST=1
LOC=C003 MODREV=cm03.3007 PORT=24 TEST=0
LOC=C004 MODREV=cm03.3007 PORT=28 TEST=1
LOC=C005 MODREV=cm03.3007 PORT=32 TEST=0
```

Figure 3. Sample Configuration File (with IO02 Module)

runbscan Utility

The `runbscan` shell script utility automatically runs the `bscan` program and the `breport` program. The `runbscan` utility is the recommended way to run the boundary scan system test. The `runbscan` shell script automatically executes `bscan`. If the `scan` test detects errors, `runbscan` automatically runs `breport` to generate a report and `bsplot` to provide a graphical representation of the errors.

Setting Up the runbscan Utility

Before you use `runbscan` to perform any testing, you should set up the utility by using the `runbscan -setup` command and then the `runbscan -t qport` command.

Using the runbscan -setup Command

The first command that you should enter is `runbscan -setup`. This command performs the following actions:

- It creates a `$HOME/.bscan` file that contains the serial number of the system you want to test. (This action is performed only for an SWS.)
- It creates the `usr/bscan/7`*`xxx`*`/` directory if it does not already exist.
- It copies the configuration file from `rel/bscan/cfg/` (if requested) into the `usr/bscan/7`*`xxx`*`/` directory.

  `runbscan` prompts you to select the CPU type (CP02 or CPE01) and memory type (CM02, CM03, or CM04) that the system uses. (If the system contains a mixture of CPU or memory types, you should select the primary module type that the system uses. Then, you should manually modify the configuration file to include the secondary CPU or memory modules that the system uses.)

Figure 4 shows an example of using the `runbscan -setup` command on an MWS. Figure 5 shows an example of using the `runbscan -setup` command on an SWS.

```
mws7001$ runbscan -setup
*******************************************************************
** This routine uses the mainframe serial number specified on
** the command line or if not specified uses the serial number
** (MF_SERIAL_NUM) in the /cri/mws/cfg2.0 file.  To help automate
** running this script the /cri/mws/cfg2.0 file should be updated
** with the mainframe serial number using xcfg.
** The serial number defines where the cfg, sdata and log
** files reside (Ex: /cri/cme/t32/usr/bscan/7001).
*******************************************************************

From the /cri/mws/cfg2.0 file:
  Serial Number: <7001>
4 Digit Serial Number (default: 7001)? 7001
    a) tv1
    b) t4
    c) t16
    d) t32
Select the chassis type (default: t4)? b

*******************************************************************
** This section generates a configuration file in the
** /cri/cme/t32/usr/bscan/7001
** directory.  The default configuration
** file name is sdata.cfg.  If sdata.cfg already exists
** the file will be renamed with a unique number appended
** to the original filename.  The new cfg file will assume
** all modules in the system are to be tested.  To disable
** a module from being tested set TEST=0 on the corresponding
** line in the cfg file.  It is also recommended to verify
** the port number specified at each module location.
*******************************************************************


*******************************************************************
** You will be queried for what type of modules are in the
** system.  When only one choice is possible that module type
** will be put into the configuration file.  On mixed cpu or
** memory systems always choose the module type with the greatest
** number of modules in the system.  Final updating of the
** configuration file may need to be done using vi
*******************************************************************

    1) cp02.3100
    2) cpe1.1005
Select a number? 1
    1) cm02.2100
    2) cm03.3007
Select a number? 1
file: /cri/cme/t32/usr/bscan/7001/sdata.cfg
View the configuration file using vi (<s> to skip, return to view)? s
runbscan setup complete
Exiting...
```

Figure 4. Example of `runbscan -setup` Command (MWS)

```
sws7033$ runbscan -setup
*******************************************************************
** This routine uses the mainframe serial number specified on
** the command line or if not specified uses the serial number
** (BSCAN_SN) in the /opt/home/craydiag/.bscan file.  To help automate
** running this script the /opt/home/craydiag/.bscan file will be created.
** The serial number defines where the cfg, sdata and log
** files reside (Ex: /opt/CYRIdiag/t32/usr/bscan/7001)
*******************************************************************

4 Digit Serial Number (default: none)? 7033
    a) tv1
    b) t4
    c) t16
    d) t32
Select the chassis type (default: t4)? b

*******************************************************************
** This section generates a configuration file in the
** /opt/CYRIdiag/t32/bdata/usr/bscan/7033
** directory.  The default configuration
** file name is sdata.cfg.  If sdata.cfg already exists
** the file will be renamed with a unique number appended
** to the original filename.  The new cfg file will assume
** all modules in the system are to be tested.  To disable
** a module from being tested set TEST=0 on the corresponding
** line in the cfg file.  It is also recommended to verify
** the port number specified at each module location.
*******************************************************************


*******************************************************************
** You will be queried for what type of modules are in the
** system.  When only one choice is possible that module type
** will be put into the configuration file.  On mixed cpu or
** memory systems always choose the module type with the greatest
** number of modules in the system.  Final updating of the
** configuration file may need to be done using vi
*******************************************************************

    1) cp02.3100
    2) cpe1.1005
Select a number? 2
    1) cm02.2100
    2) cm03.3007
Select a number? 1
file: /opt/CYRIdiag/t32/usr/bscan/7033/sdata.cfg
View the configuration file using vi (<s> to skip, return to view)? s
runbscan setup complete
Exiting...
```

Figure 5. Example of `runbscan -setup` Command (SWS)

Using the runbscan -t qport Command

The next command that you should enter is `runbscan -t qport`. This command runs the `qport` utility, which returns each BS port number, followed by the module ID for the module that is connected to the port. For example, the following text from Figure 20 on page 42 shows that BS port 20 is connected to a CM02 module with serial number 30:

BS Port Number	Module Type	Serial Number
20	CM02	30

Use the output from the `qport` utility to verify that the information in the configuration file is correct for each BS port number. For example, the configuration file shown in Figure 2 on page 15 has the following entry, which corresponds to BS port 20:

```
LOC=C005 MODREV=cm02.2100 PORT=20 TEST=1
```

This output shows that port 20 is connected to a CM02 module. You may need to check the System Test and Checkout (STCO) swap log to verify that the serial number for the module is 30.

Refer to “runbscan -t qport” on page 41 for more information.

Remember the following concepts when you work with the configuration file:

- The initial configuration file is created with default physical location-to-BS port connections. (For example, for an MWS system, B001 is connected to BS port 36.)
- Use the output from the `qport` utility to verify the boundary scan configuration by verifying all BS ports that are used and their physical locations.
- Swapping ports between module locations without updating the BS configuration file will cause boundary scan failures.

`runbscan` generates the boundary scan system data the first time that it executes a test other than `qport`. (Figure 6 shows an example of `runbscan` generating the data.) `runbscan` regenerates data only when neccessary.

```
mws7001$ runbscan -t chipid
Serial Number 7001 specified in /cri/mws/cfg2.0
******** Running gen_data test
bin/bscan   -rd usr/bscan/7001

@(#)mst310/t32/src/bscan/genv/bsb.c     10.4    05/12/97 - BOUNDARY_SCAN
TEST_OPERAND GENERATOR

Loading system configuration file.
Loading the module mask files.
Loading the module reference files.
Building the inter-module connections.
Generating the system result operands.
Saving the inter-module connections.
Saving the module connection data.
Saving the system mask operands.
Saving the system header file.

The total number of modules is: 10.
The total number of interconnects is: 7544.

bsb finished with 0 errors.
!bin/bscan -rd usr/bscan/7001
System boundary-scan data built successfully
******** gen_data test ran successfully
******** Running chipid test
bin/bscan   -d usr/bscan/7001 -t'chipid maxpass 1' -E rel/bscan/misc/equiv

******** chipid test ran successfully
```

Figure 6. Example of `runbscan` Generating the Boundary Scan Data

Starting the runbscan Utility

Once you have set up the `runbscan` utility, you can start it from the Workspace menu or from a UNIX command prompt.

Starting the runbscan Utility from the Workspace Menu

The script that installs the boundary scan system test also modifies the Workspace menu on MWS and SWS systems: it adds options that enable you to start `runbscan` directly from the Workspace menu.

Figure 7 shows how to access the BOUNDARY SCAN portion of the Workspace menu on an MWS. Figure 8 shows how to access it on an SWS. To start `runbscan` on specific channel(s), choose the appropriate menu options from the BOUNDARY SCAN submenus.

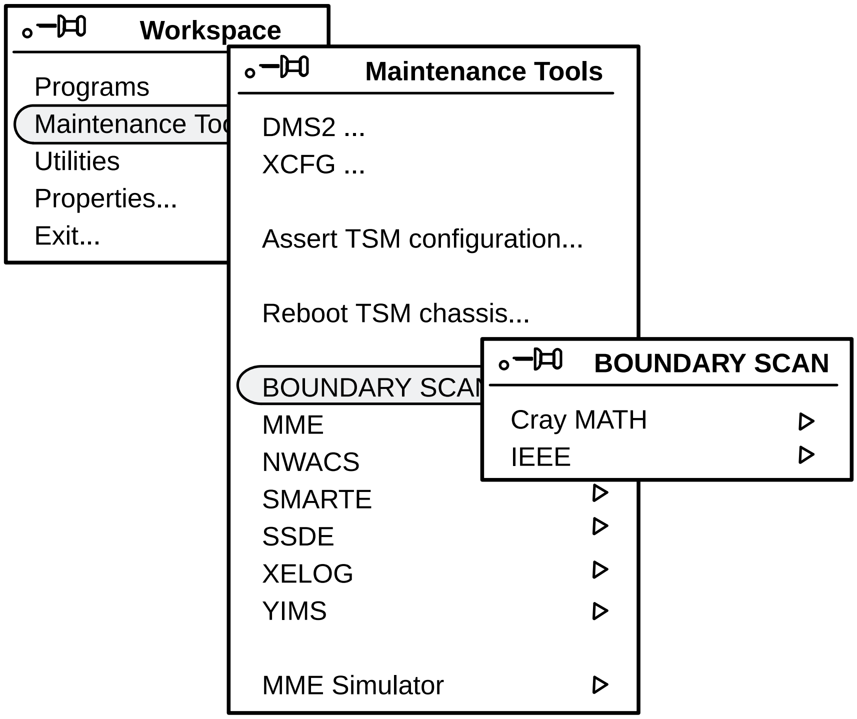

Figure 7. Accessing the BOUNDARY SCAN Submenu on an MWS

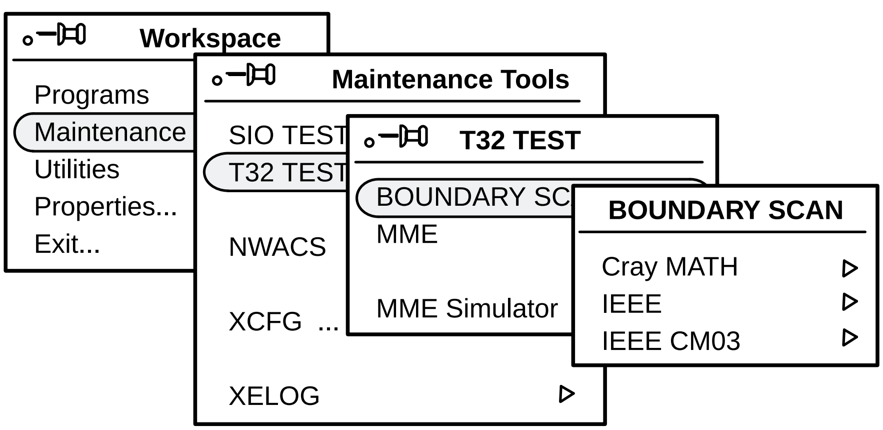

Figure 8. Accessing the BOUNDARY SCAN Submenu on an SWS

Starting the runbscan Utility from a UNIX Command Prompt

You can start the `runbscan` shell script from a UNIX command prompt.

Use the following command on an MWS:

`runbscan [-c` *ch_bs0*`[,`*ch_bs1*`]] [-e` *errcnt*`] [-f` *cfg_file*`]`
`[-L` *log_dir*`] [-m` *moddata_dir*`] [-P` *args*`] [-p` *passcnt*`] [-setup]`
`[-t` *test*`] [-HlnS]` *serial_number*

Use the following command on an SWS:

`runbscan [-e` *errcnt*`] [-f` *cfg_file*`]`
`[-h sn`*serial_number*`-mpn0,[sn`*serial_number*`-mpn1]] [-L` *log_dir*`]`
`[-m` *moddata_dir*`] [-P` *args*`] [-p` *passcnt*`] [-setup] [-t` *test*`]`
`[-abHlnS]` *serial_number*

Table 6 describes the command line options that `runbscan` uses. (An *X* in the “MWS” column indicates that the option is valid on an MWS; an *X* in the “SWS” column indicates that the option is valid on an SWS.)

Enter the `runbscan` command line options in any order. The program uses a default value if you omit an option.

Table 6. `runbscan` Command Line Options

Option	MWS	SWS	Description
`-a`		X	Runs the boundary scan system test with only the modules attached to BS 0. (This option is valid only for CRAY T932 mainframes.)
`-b`		X	Runs the boundary scan system test with only the modules attached to BS 1. (This option is valid only for CRAY T932 mainframes.)
`-c` *ch_bs0*`[,`*ch_bs1*`]`	X		Selects the boundary scan modules and front-end interface (FEI) channels that `bscan` should use. Use the *ch_bs0* parameter for CRAY T94 and CRAY T916 mainframes and for the boundary scan module located in side A of CRAY T932 mainframes (BS 0). Use the *ch_bs1* parameter for the boundary scan module located in side B of CRAY T932 mainframes (BS 1).

Table 6. `runbscan` Command Line Options (continued)

Option	MWS	SWS	Description
`-c` *ch_bs0*`[,`*ch_bs1*`]` (continued)	X		Replace the *ch_bs0* and *ch_bs1* parameters with the number of the FEI channel that is cabled to the boundary scan module that you are using. You can enter a dash (`-`) to skip the *ch_bs0* option. For example, you could enter `-c -,13` to use `bscan` with a CRAY T932 mainframe. The following examples show how to use this option: `-c 3` Use FEI channel 3 with BS 0 in a CRAY T94, CRAY T916, or CRAY T932 mainframe `-c 3,13` Use FEI channel 3 with BS 0 and FEI channel 13 with BS 1 in a CRAY T932 mainframe `-c -,13` Use FEI channel 13 with BS 1 in a CRAY T932 mainframe The default for *ch_bs0* is 3 on CRAY T94, CRAY T916, and CRAY T932 mainframes. The default for *ch_bs1* is 13 on CRAY T932 mainframes.
`-e` *errcnt*	X	X	Sets the maximum number of test execution errors that can occur before the test exits. (The default is 100,000 errors.)
`-f` *cfg_file*	X	X	Specifies the configuration file *cfg_file* that contains the module location, module type, module revision, and port number for each module in the system. The default is `usr/bscan/7`*xxx*`/sdata.cfg`.
`-H`	X	X	Displays help information. `runbscan` immediately exits after it displays the help information.
`-h` `sn`*serial_number*`-mpn0,` `[sn`*serial_number*`-mpn1]`		X	Selects the multi-purpose node (MPN) to use. `sn`*serial_number*`-mpn0` connects to BS 0 and `sn`*serial_number*`-mpn1` connects to BS 1 (in a CRAY T932 mainframe). You can enter a dash (`-`) to skip the `sn`*serial_number*`-mpn0` option. For example, you could enter `-h -,sn7033-mpn1` to use `bscan` with a CRAY T916 mainframe. (Refer also to the `-a` and `-b` command line options.)

Table 6. `runbscan` Command Line Options (continued)

Option	MWS	SWS	Description

`-h` sn*serial_number*`-mpn0,` `[sn`*serial_number*`-mpn1]` (continued)		X	The following examples show how to use this option:
			
			`-h sn7033-mpn0`
			
			Use the MPN attached to BS 0 in a CRAY T94 mainframe.
			
			`-h sn7233-mpn0,sn7233-mpn1`
			
			Use the MPN attached to BS 0 and the MPN attached to BS 1 in a CRAY T932 mainframe.
			
			`-h -,sn7233-mpn1`
			
			Use the MPN attached to BS 1 in a CRAY T932 mainframe.
			
			The default is `sn7`*xxx*`-mpn0, sn7`*xxx*`-mpn1` on systems with serial numbers between 7200 and 7299. The default is `sn7`*xxx*`-mpn0` for all other systems.

`-L` *log_dir*	X	X	Specifies the directory in which `runbscan` should place output files. The default is `usr/bscan/7`*xxx*`/log`.
`-l`	X	X	Specifies the pass count for the scan test. A popup window prompts you for the maximum pass count.
`-m` *moddata_dir*	X	X	Specifies the directory in which the module data files are located. The module data files are supplied to the site. The default directory is `rel/usr/modules/`.
`-n`	X	X	Specifies that `runbscan` should not run the `bsplot` program.
`-P` *args*	X	X	Passes the quoted string in *args* to the `bscan` program.
`-p` *passcnt*	X	X	Sets the maximum number of passes to complete before exiting. (The default is 2 passes. This option applies only to the `scan` test.)
`-S`	X	X	Suppresses inclusion of the timestamp in the filenames for `runbscan` output files. (The timestamp is .*mmddHHMM*, where *mm* is the month, *dd* is the day, *HH* is the hour, and *MM* is the minute.)

Table 6. `runbscan` Command Line Options (continued)

Option	MWS	SWS	Description
`-setup`	X	X	Creates the output directory, configuration file, and (on SWS systems only) the `$HOME/.bscan` file. The `-setup` option determines the mainframe serial number, chassis type, and module types and creates a configuration file for the system. (You may need to edit the configuration file if the system includes nonstandard port numbers or if the system is not fully populated.)
serial_number	X	X	Specifies the four-digit serial number of the system. `runbscan` uses the serial number to name the directories that it uses. For SWS systems, the default value is specified in the `$HOME/.bscan` file (if the file exists). For MWS systems, the default value is specified after the keyword `MF_SERIAL_NUM` in the `$HOME/cfg2.0` file.
`-t` *test*	X	X	Specifies the test or utility to use (which can be only one of the following tests or utilities). By default, `bscan` uses the module identification (ID) utility. Test — Description `qport` — Queries each port for module identification information. If the port does not respond, asterisks (*) indicate that no module type and module serial number are available. `qport` ignores all other command line options except `-c`, `-h`, and `-v`. `qchipid` — Queries the chip IDs and displays the chip IDs for all options on a module. `chipid` — Tests chip ID; reports an error only if a chip ID is outside the logical equivalence range. The chip ID logical equivalence is the range of decimal values of the chip ID from X . . . X0 to X . . . X9.

Table 6. `runbscan` Command Line Options (continued)

Option	MWS	SWS	Description	
`-t` *test* (continued)	X	X	<u>Test</u>	<u>Description</u>
			`setup`	Queries only the BS ports specified in the configuration file. If any of the BS ports fail to respond or the module type that a module returns does not match the module type in the configuration file, the setup test fails; when this happens, runbscan will not continue until corrective action is taken.
			`scan`	Tests the interconnections on a module and between modules.
			`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.
			`qidcode`	(This test is available only on an SWS.) Queries the identification code (idcode) registers on the GigaRing options.

Using runbscan to Run All Boundary Scan Tests

If you enter the `runbscan` command without the `-t` command line option or start the `runbscan` utility from the Workspace menu, the `runbscan` utility runs all applicable boundary scan system tests.

Example Output

Figure 9 shows an example (from an MWS) of the output that `runbscan` generates when a test detects errors.

```
mws7001$ runbscan 7001
******** Running gen_data test
bin/bscan -rd usr/bscan/7001
!bin/bscan -rd usr/bscan/7001
******** gen_data test ran successfully

******** Running setup test
bin/bscan -d usr/bscan/7001
******** setup test ran successfully

******** Running chipid test
bin/bscan -d usr/bscan/7001  -t'chipid maxpass 1'
******** chipid test ran successfully

******** Running shift test
bin/bscan -d usr/bscan/7001  -t'shift maxpass 1'
******** shift test ran successfully

******** Running scan test, SCAN will make 2 passes.
bin/bscan -d usr/bscan/7001  -t'scan maxpass 2'
runbscan:  bin/bscan found data errors
  bin/breport file:  /cri/cme/t32/usr/bscan/7001/log/scan.05210915
******** scan test found data errors
```

Location of the File That Contains the `breport` Error Information

Figure 9. Output from the `runbscan` Command (MWS)

Figure 10 shows an example (from an SWS) of the output that `runbscan` generates when all tests complete without detecting errors.

```
sws7033$ runbscan
Serial Number 7033 specified in /opt/home/craydiag/.bscan

******** Running gen_data test
bin/bscan -h sn7033-mpn0  -rd usr/bscan/7033
!bin/bscan -h sn7033-mpn0 -rd usr/bscan/7033
******** gen_data test ran successfully

******** Running setup test
bin/bscan -h sn7033-mpn0  -d usr/bscan/7033
******** setup test ran successfully

******** Running chipid test
bin/bscan -h sn7033-mpn0  -d usr/bscan/7033 -t’chipid maxpass 1’
-E rel/bscan/misc/equiv
******** chipid test ran successfully

******** Running shift test
bin/bscan -h sn7033-mpn0  -d usr/bscan/7033 -t’shift maxpass 1 ’
******** shift test ran successfully

******** Running scan test, SCAN will make 2 passes.
bin/bscan -h sn7033-mpn0  -d usr/bscan/7033 -t’scan maxpass 2 ’
-F rel/bscan/misc/flaw
******** scan test ran successfully
```

Figure 10. Output from the `runbscan` Command (SWS)

Using runbscan to Run Individual Boundary Scan Tests

You can use the `-t` command line option to specify that `runbscan` should run only one test. Table 7 provides quick-reference descriptions of the `runbscan` tests and utilities.

Table 7. `runbscan` Tests and Utilities

Test/Utility	Description
`chipid` test	Verifies that the chip option identifications (chip IDs) for the chips on the selected modules are in the appropriate ranges.
`qchipid` utility	Returns the chip option identification (chip ID) for each chip on the selected modules.
`qport` utility	Returns the module identification information for each module connected to a boundary scan module port.
`scan` test	Tests the interconnections on a module and between modules.
`scx_qidcode` utility	Queries the identification code (icode) registers on the GigaRing options. (This utility is available only on an SWS.)
`shift` test	Tests the scan chain with different patterns.

The remainder of this section describes the tests and utilities that the following commands execute:

- `runbscan -t chipid`
- `runbscan -t qchipid`
- `runbscan -t qidcode`
- `runbscan -t qport`
- `runbscan -t scan`
- `runbscan -t shift`

runbscan -t chipid

The `runbscan -t chipid` command runs the `chipid` test.

The `chipid` test reads the entire BSC for each selected module and uses the module read data mask (*module*`.rdm`) files and system read result vector (*system*`.rrv`) files to verify the chip IDs. (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 selected modules are tested.

Figure 11 shows an example of the `chipid` test output from an MWS.

NOTE: The output from this test is nearly identical on both the MWS and SWS.

```
mws7001$ runbscan -t chipid
Serial Number 7001 specified in /cri/mws/cfg2.0

******** Running gen_data test
bin/bscan -rd usr/bscan/7001
!bin/bscan -rd usr/bscan/7001
******** gen_data test ran successfully********

Running chipid test
bin/bscan -d usr/bscan/7001  -t’chipid maxpass 1’
-E rel/bscan/misc/equiv
runbscan:  bin/bscan chipid found data errors
  bin/bscan file:  /cri/cme/t32/usr/bscan/7001/log/chipid.08261443
******** chipid test found data errors
```

Figure 11. Example `chipid` Test Output (MWS)

In this example, the `chipid` test detected an error. The log file for the test contains more information about the error. Figure 12 and Figure 13 show the log file for this example.

```
mws7001$ cat /cri/cme/t32/usr/bscan/7001/log/chipid.08261443
!bin/bscan -d usr/bscan/7001 -t chipid maxpass 1 -E rel/bscan/misc/equiv
!
!Location     Mod.Rev      BS     Port    TYPE      SN      Selected
!A001-00      io01.1013     0       5     IO01       1      yes
!B001-01      cp02.3100     0      36     CP02       6      yes
!B002-02      cp02.3100     0      37     CP02      16      yes
!B003-03      cp02.3100     0      38     CP02      15      yes
!B004-04      cp02.3100     0      39     CP02       4      yes
!C001-05      sr01.1007     0       4     SR01       1      yes
!C002-06      cm02.2100     0      23     CM02      10      yes
!C003-07      cm02.2100     0      22     CM02      34      yes
!C004-08      cm02.2100     0      21     CM02      33      yes
!C005-09      cm02.2100     0      20     CM02      30      yes
!
!BEGIN EQUIV
!        mod_rev    sn  equiv chipids
!      cp02.3100     6  10:076734,22:076746,58:076734,61:076734,
!                       86:076734,90:076734,125:076746,137:076734
!      cp02.3100    16  10:076734,22:076746,58:076734,61:076734,
!                       86:076734,90:076734,125:076746,137:076734
!      cp02.3100    15  10:076734,22:076746,58:076734,61:076734,
!                       86:076734,90:076734,125:076746,137:076734
!      cp02.3100     4  10:076734,22:076746,58:076734,61:076734,
!                       86:076734,90:076734,125:076746,137:076734
!      cm02.2100    10  11:076652,12:076652,13:076652,14:076652,
!                       19:076652,20:076652,21:076652,22:076652,
!                       25:076664,26:076664,27:076664,28:076664,
!                       29:076664,30:076664,31:076664,32:076664,
!                       33:076664,34:076664,35:076664,36:076664,
!                       37:076664,38:076664,39:076664,40:076664,
!                       43:076652,44:076652,45:076652,46:076652,
!                       51:076652,52:076652,53:076652,54:076652
!      cm02.2100    34  11:076652,12:076652,13:076652,14:076652,
!                       19:076652,20:076652,21:076652,22:076652,
!                       25:076664,26:076664,27:076664,28:076664,
!                       29:076664,30:076664,31:076664,32:076664,
!                       33:076664,34:076664,35:076664,36:076664,
!                       37:076664,38:076664,39:076664,40:076664,
!                       43:076652,44:076652,45:076652,46:076652,
!                       51:076652,52:076652,53:076652,54:076652
!      cm02.2100    33  11:076652,12:076652,13:076652,14:076652,
!                       19:076652,20:076652,21:076652,22:076652,
!                       25:076664,26:076664,27:076664,28:076664,
!                       29:076664,30:076664,31:076664,32:076664,
!                       33:076664,34:076664,35:076664,36:076664,
!                       37:076664,38:076664,39:076664,40:076664,
!                       43:076652,44:076652,45:076652,46:076652,
!                       51:076652,52:076652,53:076652,54:076652
!      cm02.2100    30  11:076652,12:076652,13:076652,14:076652,
!                       19:076652,20:076652,21:076652,22:076652,
!                       25:076664,26:076664,27:076664,28:076664,
!                       29:076664,30:076664,31:076664,32:076664,
!                       33:076664,34:076664,35:076664,36:076664,
!                       37:076664,38:076664,39:076664,40:076664,
!                       43:076652,44:076652,45:076652,46:076652,
!                       51:076652,52:076652,53:076652,54:076652
!END EQUIV
```

Figure 12. Example `chipid` Log File (Part 1 of 2)

```
!
!field 1: Module location
!field 2: Chip position 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 Tue Aug 26 14:43:32 1997
A001-00     8   0114236-0114247   0114267  dr001 1DB
!chipid reached maximum pass limit with 1 passes and 1 errors
on Tue Aug 26 14:43:32 1997
```

Field Descriptions (the `!field 1` through `!field 6` lines)

Module Error Information (`chipid` Error Report) (the `A001-00` line)

Information Line States That 1 Error Was Detected (the `!chipid reached maximum pass limit` line)

Figure 13. Example `chipid` Log File (Part 2 of 2)

In Figure 13, the lines `!field 1: Module location` through `!field 6: Physical location` describe the information in the `chipid` error report that follows. The line `A001-00 8 0114236-0114247 0114267 dr001 1DB` is the `chipid` error report.

Use the field descriptions to interpret the error report as follows:

`!field 1` =	`A001-00`	The location of the module in error
`!field 2` =	`8`	The chip number
`!field 3` =	`0114236-0114247`	The expected range of the chip ID
`!field 4` =	`0114267`	The actual chip ID
`!field 5` =	`dr001`	The logical string, option type, and number
`!field 6` =	`1DB`	The physical location of the option (In this example, board `1` location `DB`)

NOTE: In tests without errors, the field descriptions are displayed on the screen, but no error report appears.

Field engineers will find the `chipid` test useful to determine the integrity of the BSC. A single error may indicate that a boundary scan cell has failed on an ASIC; this is a nonfatal error. A contiguous series of `chipid` failures on a module may indicate a broken BSC, which invalidates all other boundary scan system test results.

Using the equiv File

When more than one chip ID is possible for a specific location on a module, the chip IDs are specified in the `equiv` file. The `equiv` file is updated when a new ASIC is designed to replace an existing ASIC.

`runbscan` uses the read-only `equiv` file that is located at `rel/bscan/equiv`. When `equiv` files become outdated, updated `equiv` files will be announced through field notices. The field notices will include instructions about how to obtain and install the updated `equiv` files.

Figure 14 displays the contents of an `equiv` file. `runbscan` ignores blank lines and lines that begin with the # symbol in `equiv` file.

```
sws7033$ cat equiv
# USMID @(#)mst310/t32/src/bscan/scripts/equiv  10.2    04/03/97 16:08:38
#    bscan.equiv
#
#  - This file can be used to add equivalent chipids when running the
#    -t chipid option under the bscan program.  Equivalent chipids are
#    needed when more than one equivalent chipid can be specified for
#    a location on a module.  To view the chipids on a module
#    use the -t qchipid command line option with the bscan program.
#
#  - Equivalent chipids are specified as pairs of numbers separated
#    by a colon.  Each pair of equivalent chipids is separated by a comma.
#    The first number in the pair is a decimal index number as output
#    in field 2 by the chipid or qchipid test.  The second number is the
#    equivalent chipid.  A leading ZERO indicates that the equivalent
#    chipid will be read in as an octal value.  Otherwise, a decimal value
#    is assumed.
#
#  - A SN=0 will apply to all MODREV of this type.
#    More than one equivalent chipid can be assigned to an index.
#    For example: 22:076746,22:076734  will allow three values to be
#    valid for index 22; the default, 076746 and 076734.
#
#  - The following is an example for the module cpe1, revision 1005,
#    without regard for serial number.
#
# The above example is represented as follows (without the # sign)
# MODREV=cpe1.1005 SN=0 CHIPIDS=58:076734,61:076734,86:076734
# and so on...
MODREV=cpe1.1005 SN=0
CHIPIDS=10:076734,22:076746,57:076734,60:076734,85:076734,89:076734,124:076746,136:
076734
MODREV=cp02.3100 SN=0
CHIPIDS=10:076734,22:076746,58:076734,61:076734,86:076734,90:076734,125:076746,137:
076734
MODREV=cm02.2100 SN=0
CHIPIDS=11:076652,12:076652,13:076652,14:076652,19:076652,20:076652,21:076652,22:07
6652,25:076664,26:076664,27:076664,28:076664,29:076664,30:076664,31:076664,32:07666
4,33:076664,34:076664,35:076664,36:076664,37:076664,38:076664,39:076664,40:076664,4
3:076652,44:076652,45:076652,46:076652,51:076652,52:076652,53:076652,54:076652
MODREV=cm03.3007 SN=0
CHIPIDS=11:076652,12:076652,13:076652,14:076652,19:076652,20:076652,21:076652,22:07
6652,25:076664,26:076664,27:076664,28:076664,29:076664,30:076664,31:076664,32:07666
4,33:076664,34:076664,35:076664,36:076664,37:076664,38:076664,39:076664,40:076664,4
3:076652,44:076652,45:076652,46:076652,51:076652,52:076652,53:076652,54:076652
MODREV=nw01.1005 SN=0
CHIPIDS=13:076676,14:076676,15:076676,16:076676,17:076676,25:076676,26:076676,35:07
6676,36:076676,37:076676,38:076676,39:076676,52:076676,53:076676,54:076676,55:07667
6,56:076676,64:076676,65:076676,74:076676,75:076676,76:076676,77:076676,78:076676
MODREV=io02b.1005 SN=0
CHIPIDS=7:0114402,11:0114402,15:0114402,19:0114402,54:0114402,58:0114402,62:0114402
,66:0114402
```

Figure 14. Example `equiv` File

runbscan -t qchipid

The `runbscan -t qchipid` command runs the `qchipid` utility.

The `qchipid` utility queries and reports the chip IDs on all selected modules. The `qchipid` utility does not test for errors; it only displays the chip IDs on each selected module.

Figure 15 shows an example of the `qchipid` utility output from an MWS.

NOTE: The output from this utility is nearly identical on both the MWS and SWS.

```
mws7001$ runbscan -t qchipid
Serial Number 7001 specified in /cri/mws/cfg2.0

******** Running gen_data test
bin/bscan -rd usr/bscan/7001
!bin/bscan -rd usr/bscan/7001
******** gen_data test ran successfully

******** Running qchipid test
bin/bscan -d usr/bscan/7001 -t'qchipid maxpass 1' -E rel/bscan/misc/equiv
******** qchipid test ran successfully
View the qchipid file using vi (<s> to skip, return to view)? s
```

Figure 15. Example `qchipid` Utility Output (MWS)

Figure 16 shows the contents of the log file for this example.

```
mws7001$ cat /cri/cme/t32/usr/bscan/7033/log/qchipid
!bscan -d usr/bscan/7001 -t qchipid maxpass 1 -E rel/bscan/misc/equiv
!
!Location   Mod.Rev     BS    Port   TYPE     SN      Selected
!A001-00    io01.1013    0      5    IO01      1      yes
!B001-01    cp02.3100    0     36    CP02      6      yes
!B002-02    cp02.3100    0     37    CP02     16      yes
!B003-03    cp02.3100    0     38    CP02     15      yes
!B004-04    cp02.3100    0     39    CP02      4      yes
!C001-05    sr01.1007    0      4    SR01      1      yes
!C002-06    cm02.2100    0     23    CM02     10      yes
!C003-07    cm02.2100    0     22    CM02     34      yes
!C004-08    cm02.2100    0     21    CM02     33      yes
!C005-09    cm02.2100    0     20    CM02     30      yes
!
!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 Tue Aug 26 06:13:32 1997
A001-00  0   0074454-0074465   0074456     tz000     1CH
A001-00  1   0075476-0075507   0075500     de001     1CG
A001-00  2   0075440-0075451   0075440     dc001     1AD
•••
!qchipid reached maximum pass limit with 1 passes and 0 errors on
Tue Aug 26 06:13:32 1997
```

Figure 16. Example `qchipid` Log File (MWS)

The `qchipid` output format is similar to the format of the `chipid` output.

NOTE: Figure 16 does not contain the complete information that the `qchipid` test produces (••• represents the missing information). The `qchipid` output displays the chip IDs from all the chips on all scanned modules.

runbscan -t qidcode (SWS Only)

The `runbscan -t qidcode` command runs the `scx_qidcode` utility.

The `scx_qidcode` utility queries the identification code (idcode) registers on the GigaRing options. This utility runs only on an SWS.

Figure 17 shows an example of the `scx_qidcode` utility output from an SWS.

```
sws7033$ runbscan -t qidcode
Serial Number 7033 specified in /opt/home/craydiag/.bscan

******** Running gen_data test
bin/bscan -h sn7033-mpn0  -rd usr/bscan/7033
!bin/bscan -h sn7033-mpn0 -rd usr/bscan/7033
******** gen_data test ran successfully

******** Running scx_qidcode test
bin/bscan -h sn7033-mpn0 -d usr/bscan/7033 -t'scx_qidcode maxpass 1'
******** scx_qidcode test ran successfully
View the scx_qidcode file using vi (<s> to skip, return to view)? s
```

Figure 17. Example `scx_qidcode` Utility Output (SWS Only)

Figure 18 shows the contents of the log file for this example.

```
sws7033$ cat /opt/CYRIdiag/t32/usr/bscan/7033/log/scx_qidcode
!bin/bscan -h sn7033-mpn0 -d usr/bscan/7033 -tscx_qidcode maxpass 1
!
!Location  Mod.Rev    BS  Port      TYPE    SN        Selected
!A001-00  io02b.1005   0    40      IO02    16        yes
!A001-01  io02c.1005   0    37      JTAG     0        yes
!B001-02   cpe1.1005   0     0      CPE1    29        yes
!B002-03   cpe1.1005   0     4
!B003-04   cpe1.1005   0    24
!B004-05   cpe1.1005   0    12
!C001-06   sr01.1007   0    32      SR01    44        yes
!C002-07   cm02.2100   0    16      CM02   496        yes
!C003-08   cm02.2100   0     8
!C004-09   cm02.2100   0    24      CM02   499        yes
!C005-10   cm02.2100   0    32
!
!field 1: Module location
!field 2: Chip number in boundary scan chain (starting at zero)
!field 3: 4 bit revision number (hex)
!field 4: 16 bit part number (hex)
!field 5: 11 bit manufacturing number (hex)
!field 6: 1 bit mandatory signifing valid idcode
!field 7: IDCODE unformatted
!
!scx_qidcode started on Wed Jun  4 10:45:09 1997
A001-01      0  0x0  0x01a4  0x036  1   1a406d
A001-01      1  0x0  0x01a4  0x036  1   1a406d
A001-01      2  0x0  0x01a4  0x036  1   1a406d
A001-01      3  0x0  0x01a4  0x036  1   1a406d
A001-01      4  0x0  0x01a4  0x036  1   1a406d
A001-01      5  0x0  0x01a4  0x036  1   1a406d
A001-01      6  0x0  0x01a4  0x036  1   1a406d
A001-01      7  0x0  0x01a4  0x036  1   1a406d
!scx_qidcode reached maximum pass limit with 1 passes and 0 errors on Wed
Jun  4 10:45:09 1997
```

Figure 18. Example `scx_qidcode` Log File (SWS Only)

runbscan -t qport

The `runbscan -t qport` command runs the `qport` utility.

The `qport` utility reads all ports on the boundary scan module. The ports that successfully complete the read will display the module type and the module serial number. For example, in Figure 20, the entry `04 SR01 1` shows that port 4 connects to a shared module with a serial number of 1. This entry is the module identification value from the BSC on the port.

Figure 19 shows an example of the `qport` utility output from an MWS.

```
mws7001$ runbscan -t qport
Serial Number 7001 specified in /cri/mws/cfg2.0

******** Running qport test
bin/bscan -t qport
******** qport test ran successfully
View the qport file using vi (<s> to skip, return to view)? s
```

Figure 19. Example `qport` Utility Output (MWS)

Figure 20 shows the contents of the `qport` log file for this example.

```
mws7001$ cat /cri/cme/t32/usr/bscan/7001/log/qport
!bin/bscan -t qport
!
!field 1: Channel number (octal)
!field 2: Port number
!field 3: Module type id
!field 4: 11 bit serial number
!
!qport started on Fri Mar 21 09:27:50 1997
03          00  ****  *****
03          01  ****  *****
03          02  ****  *****
03          03  ****  *****
03          04  SR01      1   <-- Shared Module Information from
03          05  IO001     1       the qport Output.
03          06  ****  *****
03          07  ****  *****
03          08  ****  *****
03          09  ****  *****
03          10  ****  *****
03          11  ****  *****
03          12  ****  *****
03          13  ****  *****
03          14  ****  *****
03          15  ****  *****
03          16  ****  *****
03          17  ****  *****
03          18  ****  *****
03          19  ****  *****
03          20  CM02     30
03          21  CM02     33
03          22  CM02     34
03          23  CM02     10
03          24  ****  *****
03          25  ****  *****
03          26  ****  *****
03          27  ****  *****
03          28  ****  *****
03          29  ****  *****
03          30  ****  *****
03          31  ****  *****
03          32  ****  *****
03          33  ****  *****
03          34  ****  *****
03          35  ****  *****
03          36  CP02      6
03          37  CP02     16
03          38  CP02     15
03          39  CP02      4
03          40  ****  *****
03          41  ****  *****
03          42  ****  *****
03          43  ****  *****
03          44  ****  *****
03          45  ****  *****
03          46  ****  *****
03          47  ****  *****
!qport reached maximum pass limit with 1 passes and 0 errors on Fri
Mar 21 09:27:50 1997
```

Figure 20. Example `qport` Utility Log File (MWS)

Ports that fail to be read when `qport` queries them display asterisks (`*`) for the module type and serial number. For example, in Figure 20, the entry `06 **** ****` has no module information.

Figure 21 shows an example of the `qport` utility output from an SWS.

```
sws7033$ runbscan -t qport
Serial number 7033 specified in /opt/home/craydiag/.bscan

******** Running qport test
bin/bscan -h sn7033-mpn0  -d usr/bscan/7033 -t qport
******** qport test ran successfully
View the qport file using vi (<s> to skip, return to view)? s
```

Figure 21. Example `qport` Utility Output (SWS)

Figure 22 shows the contents of the `qport` log file for this example.

```
sws7033$ cat /opt/CYRIdiag/t32/usr/bscan/7033/log/qport
!bscan -h sn7033-mpn0 -t qport
!
!field 1: Hostname string
!field 2: Port number
!field 3: Module type id
!field 4: 11 bit serial number
!
!qport started on Wed May 14 14:24:29 1997
sn7033-mpn0  00  CPE1     29
sn7033-mpn0  01  ****  *****
sn7033-mpn0  15  ****  *****
.
.example lines removed
.
sn7033-mpn0  33  !!01      0
sn7033-mpn0  34  ****  *****
sn7033-mpn0  35  ****  *****
sn7033-mpn0  36 *IO02     16
sn7033-mpn0  37  JTAG  -----
sn7033-mpn0  38  ****  *****
sn7033-mpn0  39  ****  *****
sn7033-mpn0  40  ****  *****
sn7033-mpn0  47  ****  *****
!qport reached maximum pass limit with 1 passes and 0 errors on Wed
May 14 14:24:40 1997
```

Figure 22. Example `qport` Utility Log File (SWS)

On an SWS, unused boundary scan ports return all ones; the `qport` utility represents unused ports with all asterisks (`*`). (Refer to the output for ports 34, 35, 38, 39, 40, and 47 in Figure 22.) Ports that have tdi/tdo grounded return `!!00 0`. (Port 33 is normally grounded on SWS systems. Refer to the output for port 33 in Figure 22.)

An asterisk (`*`) in front of the four-character module ID indicates that the module was found in a different location in the BSC than the default. (At the time of this printing, only the module IDs for IO02 modules are found in a BSC location other than the default. Refer to the output for port 36 in Figure 22.)

runbscan -t scan

The `runbscan -t scan` command runs the `scan` test.

The `scan` test verifies the integrity of connections within and between all selected modules. The `scan` test detects an error when the data pattern that is read from the module is different from the data pattern in the read result vector file: `usr/bscan7`*xxx*`/sdata/`*location*`.`*module*`.`*revision*`.rrv` (for example, `usr/bscan/7001/sdata/C002.cm02.2100.rrv`). All modules specified in the configuration file are tested.

Figure 23 shows an example of the `scan` test output from an MWS.

NOTE: The output from this test is nearly identical on both the MWS and SWS.

```
mws7001$ runbscan -t scan
Serial Number 7001 specified in /cri/mws/cfg2.0
******** Running gen_data test
bin/bscan   -rd usr/bscan/7001
!bin/bscan -rd usr/bscan/7001
******** gen_data test ran successfully
******** Running scan test
bin/bscan   -d usr/bscan/7001 -t'scan maxpass 1 '
  runbscan: bscan found data errors
  bin/breport file: /cri/cme/t32/usr/bscan/7001/log/scan.08271307
******** scan test found data errors
```

Figure 23. Example `scan` Test Output (MWS)

When the `scan` test detects errors, `runbscan` runs the `breport` program to generate an error report. Figure 24 shows the `breport` error report (from an MWS) for the example that Figure 23 shows.

```
mws7001$ cat /cri/cme/t32/usr/bscan/7001/log/scan.08271453
!bin/breport -d usr/bscan/7001 -e usr/bscan/7001/log/scan.tmp
!bin/bscan -d usr/bscan/7001 -t scan maxpass 1
!
!Location  Mod.Rev    BS  Port      TYPE    SN        Selected
!A001-00   io01.1013   0     5      IO01     1        yes
!B001-01   cp02.3100   0    36      CP02     6        yes
!B002-02   cp02.3100   0    37      CP02    16        yes
!B003-03   cp02.3100   0    38      CP02    15        yes
!B004-04   cp02.3100   0    39      CP02     4        yes
!C001-05   sr01.1007   0     4      SR01     1        yes
!C002-06   cm02.2100   0    23      CM02    10        yes
!C003-07   cm02.2100   0    22      CM02    34        yes
!C004-08   cm02.2100   0    21      CM02    33        yes
!C005-09   cm02.2100   0    20      CM02    30        yes
!
!BEGIN FLAW
!        mod_rev    sn  flawed bits
!      cp02.3100    36  11325,35973
!      cp02.3100    37  11325,35973
!      cp02.3100    38  11325,35973
!      cp02.3100    39  11325,35973
!END FLAW
!
!field 1: Module location-index
!field 2: Bit number in boundary scan chain (starting at zero)
!field 3: Pattern number (starting at zero)
!field 4: Expected data value (0 or 1)
!
!scan started on Wed Aug 27 13:07:36 1997
B001-01  45694     01  P1P1010110  1P10010101  0110100101
                       ^ ^          ^
C002-06   8960     01  P1D0P1D00D  P11PP1D0D0  D01PD0D0P1
                       ^ ^ ^ ^  ^  ^  ^^ ^ ^   ^  ^^ ^ ^
!scan reached maximum error limit with 0 passes and 18 errors on Wed Aug
27 13:07:40 1997
!
!<src loc> <bitno>     <pin type> <logical> <physical> <log/phy offset>
!    <sib loc>         <pin type> <logical> <physical> <log/phy offset>
!    <dst loc> <bitno> <pin type> <logical> <physical> <log/phy offset> .
!-------------------------------------------------------------------------
!BEGIN ERF
B001-01  45694      j vf003OIDvm015INA 2IG2671IC155 1 .
B001-01  13642      A ci001OBIzb001IBI 2CI3202YB103 0
    C002-06   8960  J za001OBImf002IAI 2YA1032BA251 1 .
!END ERF
```

Representation of the Actual Bit Patterns 0 through 31 (Refer to Figure 25)

Figure 24. Example `breport` Output (MWS)

The top portion of Figure 24 shows the command line, the module identification information, and the field descriptions. The middle portion shows the representation of the actual and expected data and a brief explanation of the `scan` error information (the last two lines). The bottom portion shows the physical and logical net information. This information is from the *module*`.erf` file that corresponds to the module and boundary scan bit number.

Table 8 describes the symbols that `breport` uses to indicate the results of comparing the expected and actual data patterns.

Table 8. `breport` Bit Representation Symbols

Symbol	Description
`0`	The actual data was 0 and matched the expected data.
`1`	The actual data was 1 and matched the expected data.
`P`	Indicates a picked bit: The test expected the bit to be a 0, but the actual value was a 1 for every pass of the test.
`D`	Indicates a dropped bit: The test expected the bit to be a 1, but the actual value was a 0 for every pass of the test.
`p`	Indicates an intermittent picking bit: The test expected the bit to be a 0, but the actual value was a 1 for some, but not all, passes.
`d`	Indicates an intermittent dropping bit: The test expected the bit to be a 1, but the actual value was a 0 for some, but not all, passes.

The bit fields to the right of the module location and bit number represent the expected results for all patterns (refer to Figure 25). The information displayed in Figure 25 relates to the information in Figure 24. In Figure 24, the first error is at module location `C002-06,` at bit number `8960,` which expected a data value of 0. Figure 25 also shows the same information with a `P` (a picked bit) at bit 1, which indicates that a data value of 0 was expected but that a data value of 1 was received.

```
!field 1: Module location-index
!field 2: Bit number in boundary scan chain (starting at zero)
!field 3: Pattern number (starting at zero)
!field 4: Expected data value (0 or 1)
!
!scan started on Wed Aug 27 13:07:36 1997
B001-01  45694     01  P1P1010110  1P10010101  0110100101
                       ^ ^          ^
C002-06   8960     01  P1D0P1D00D  P11PP1D0D0  D01PD0D0P1
                       ^ ^ ^ ^  ^  ^  ^^ ^ ^   ^  ^^ ^ ^
```

Bits 31 - 30   Bits 29 - 20   Bits 19 - 10   Bits 9 - 0

Figure 25. Representation of the Failing Data Pattern Numbers for a BSC Bit

In Figure 25, the error at location `B001-02`, BSC bit number `45694`, is an intraconnect failure (failure on the CP module). The error at location `C002`, BSC bit number `8960`, is an interconnect failure (failure between the CP and CM module). The interconnect failure may be on the module at the location `B001`, or on the module at location `C002`, or on the electronic zero insertion force (EZIF) connector.

Using the flaw File

`runbscan` uses a read-only `flaw` file that is located at `rel/bscan/flaw` to flaw out any boundary scan errors that do not affect mainframe operation. (`runbscan` does not test the BSC bits that are specified in the `flaw` file.)

Figure 26 shows the contents of an example `flaw` file. The information below the # signs are the modules, revision numbers, serial numbers, and bits that are skipped (not tested).

```
sws7033$ cat flaw
# USMID @(#)mst310/t32/src/bscan/scripts/flaw   10.1    08/14/97 15:28:37
#    bscan.flaw
#
#  - This file can be used to ignore failing bits in the
#    boundary scan chain of a module.  The following is an example
#    for the module io01, revision 1013, and serial number 12.
#    Bits 12, 15 and 17 are masked out and no errors will
#    be reported for these bits in the boundary scan chain.
#
# The above example is represented as follows (without the # sign)
# MODREV=io01.1013 SN=12 BITS=12,15,17
# MODREV=cp02.3100 SN=15 BITS=12233
# and so on...
MODREV=cpe1.1005 SN=0 BITS=11009,35657
MODREV=cp02.3100 SN=0 BITS=11325,35973
```

Figure 26. Example `flaw` File

NOTE: You can also create your own flaw file and use it with the `bscan` program. Refer to the `bscan` man page for more information.

When `flaw` files become outdated, updated `flaw` files will be announced through field notices. The field notices will include instructions about how to obtain and install the updated `flaw` files.

runbscan -t shift

The `runbscan -t shift` command runs the `shift` test.

The `shift` test verifies the integrity of the BSC on all selected modules by writing to and reading from the BSC. The `shift` test serially shifts test patterns and responses through the BSC. The test uses byte patterns that are all 0's, all 1's, alternating 0's to 1's, and alternating 1's to 0's. The final pattern is all 0's, which forces the preceding pattern out of the BSC.

The `shift` test detects an error when the data pattern that is written to a module is different from the data pattern that is read from the module. The `shift` test checks each module until all selected modules are tested.

Figure 27 shows an example of the `shift` test output from an MWS.

NOTE: The output from this test is nearly identical on both the MWS and SWS.

```
mws7001$ runbscan -t shift
Serial Number 7001 specified in /cri/mws/cfg2.0

******** Running gen_data test
bin/bscan   -rd usr/bscan/7001
!bin/bscan -rd usr/bscan/7001
******** gen_data test ran successfully

******** Running shift test
bin/bscan   -d usr/bscan/7001 -t'shift maxpass 1 '
  runbscan: bscan found data errors
  bin/breport file: /cri/cme/t32/usr/bscan/7001/log/shift.08271453
******** shift test found data errors
```

Figure 27. Example `runbscan -t shift` Output (MWS)

In this example, the `shift` test detected an error. The log file for the test contains more information about the error. Figure 28 shows the log file for this example.

```
mws7001$ cat /cri/cme/t32/usr/bscan/7001/log/shift.08271453
!bin/breport -d usr/bscan/7001 -e usr/bscan/7001/log/shift.tmp
!bin/bscan -d usr/bscan/7001 -t shift maxpass 1
!
!Location  Mod.Rev    BS  Port      TYPE    SN        Selected
!A001-00   io01.1013   0     5      IO01     1        yes
!B001-01   cp02.3100   0    36      CP02     6        yes
!B002-02   cp02.3100   0    37      CP02    16        yes
!B003-03   cp02.3100   0    38      CP02    15        yes
!B004-04   cp02.3100   0    39      CP02     4        yes
!C001-05   sr01.1007   0     4      SR01     1        yes
!C002-06   cm02.2100   0    23      CM02    10        yes
!C003-07   cm02.2100   0    22      CM02    34        yes
!C004-08   cm02.2100   0    21      CM02    33        yes
!C005-09   cm02.2100   0    20      CM02    30        yes
!
!field 1: Module location-index
!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 Wed Aug 27 14:52:36 1997
A001-00  16693     01D0    ─┐
                     ^      │
A001-00  16694     10D0     ├─ BSC Bits That Failed
                     ^      │
A001-00  16695     01D0    ─┘
                     ^
!shift reached maximum error limit with 0 passes and 3 errors on Wed Aug
27 14:53:20 1997
!
!<src loc> <bitno>     <pin type> <logical> <physical> <log/phy offset>
!    <sib loc>         <pin type> <logical> <physical> <log/phy offset>
!    <dst loc> <bitno> <pin type> <logical> <physical> <log/phy offset> .
!------------------------------------------------------------------------
!BEGIN ERF
A001-00  16693      j db003OCKdr030IFC 2AG0402DO072 1 .
A001-00  16694      j db003OCJdr030IFB 2AG0392DO071 1 .
A001-00  16695      j db003OCIdr030IFA 2AG3472DO070 1 .
!END ERF
```

Figure 28. Example `shift` Test Log File (MWS)

Figure 28 shows that at module location `A001-00`, the bit numbers in the BSC that failed were `16693`, `16694`, and `16695`. The expected data value for each of these bits is 1. The actual data value for each of these bits is 0.

runbscan Exit Codes

`runbscan` will return one of the following exit codes when it completes executing:

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

runbscan Error Messages

`runbscan` returns error messages from the `bscan` and `breport` programs.

Table 9 describes the error messages that `runbscan` returns to `stderr` for `bscan`.

Table 9. `bscan` Error Messages

Error Message	Description
`bscan: Illegal option` *x*	An option *x* is invalid. Correct the option and restart the test with a valid option.
`bscan: Illegal argument` *x*	An 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 that the path to the directory is correct and verify that the read, write, and execute permissions are enabled. Correct and rerun the test.
`bscan: Module function failed`	A module function request to the boundary scan module failed. Contact your system support staff. (MWS only)
`bscan: Channel function failed`	A channel function request to the BS02 failed. Contact your system support staff. (MWS only)
`bscan: Location or port number invalid on line` *n* `in file` *filename*	An invalid location or port number exists on line *n* in the configuration file *filename*. Rerun the test with the correct location or port number.

Table 9. `bscan` Error Messages (continued)

Error Message	Description
`bscan:` *filename*`: error in configuration file`	The test encountered a bad line in the configuration file. Correct the file and rerun the test. Refer to "sdata.cfg (Configuration File)" on page 14 of this document.
`bscan:` *x*`: location doesn't have port defined.`	Module location *x* in the command line does not have a port defined in the configuration file. Correct the configuration file and rerun the test.
`bscan:` *x*`: unable to find location in system.`	Module location *x* in the command line is not defined in the *system*`.hdr` file. Correct the configuration file and rerun the test.

Table 10 describes the error messages that `runbscan` returns to `stderr` for `breport`.

Table 10. `breport` 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.`	The directory is missing or invalid. Verify that the path to the directory is correct and that the read, write, and execute permissions are set properly. Enter the correct path and rerun.
`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 probably using a different data directory than the directory `bscan` used. Supply the correct module location and rerun.
`breport: Need to specify a system name.`	You must specify a system name when you use the default *sysdata_dir* or when the base name of the *sysdata_dir* is different from the *system*`.hdr` filename. Specify the correct system name and rerun.
`breport: bad input line` *n*`, requires location and bit number`	You entered the input line *n* incorrectly; correct the input line with the location and bit number and rerun.

bsb Program

The boundary scan system build (`bsb`) program generates data files that contain the system interconnections between modules and their expected scan data values for a given configuration of a CRAY T90 series system.

Normally, you should not need to manually run the `bsb` program on-site. The `runbscan` shell script automatically runs the `bsb` program to generate the data files for your system. If you need to manually run the `bsb` program, refer to the `bsb` man page for more information.

bscan Program

The `bscan` program tests the interconnections on modules and between modules for a given configuration and reports any errors that it detects. Use `bscan` to verify the integrity of the mainframe after a failure occurs that causes the OS to go down or after you complete a repair procedure.

The `bscan` program compares the actual data with the expected data and completes when it exceeds a maximum pass count or a specified wall-clock time limit.

For CRAY T94 and CRAY T916 mainframes, the `bscan` program requires complete control of the mainframe; therefore, you cannot run a `bscan` test and the OS simultaneously. For CRAY T932 mainframes, you can configure the mainframe to run `bscan` tests in one half of the mainframe while the OS runs in the other half of the mainframe.

Normally, you should not need to manually run the `bscan` program on-site. The `runbscan` shell script automatically runs the `bscan` program to test your system. If you need to manually run the `bscan` program, refer to the `bscan` man page for more information.

breport Program

The boundary scan report generator (`breport`) processes the information from the boundary scan program (`bscan`). The `breport` program takes the error output from the `scan` and `shift` tests and presents it in a condensed, organized 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.

Normally, you should not need to manually run the `breport` program on-site. The `runbscan` shell script automatically runs the `breport` program to process the information from the `bscan` program. If you need to manually run the `breport` program, refer to the `bscan` man page for more information.

bsplot Program

The `bsplot` program uses the output from the `breport` program to create a graphical map of any errors detected by the `scan` test. This enables you to easily detect the failing components.

`bsplot` reads the information between the `!BEGIN ERF` and `!END ERF` lines in the `breport` output and uses this information to create a graphical map of the errors.

Normally, you should not need to manually run the `bsplot` program on-site. The `runbscan` shell script automatically runs the `bsplot` program to create a PostScript™ file that contains the graphical map of any errors that the `scan` test detects. If you need to manually run the `bsplot` program, refer to the `bsplot` man page for more information.

bsplot Output

The `bsplot` program produces a PostScript file. Figure 29 shows an example of the output from the `bsplot` program.

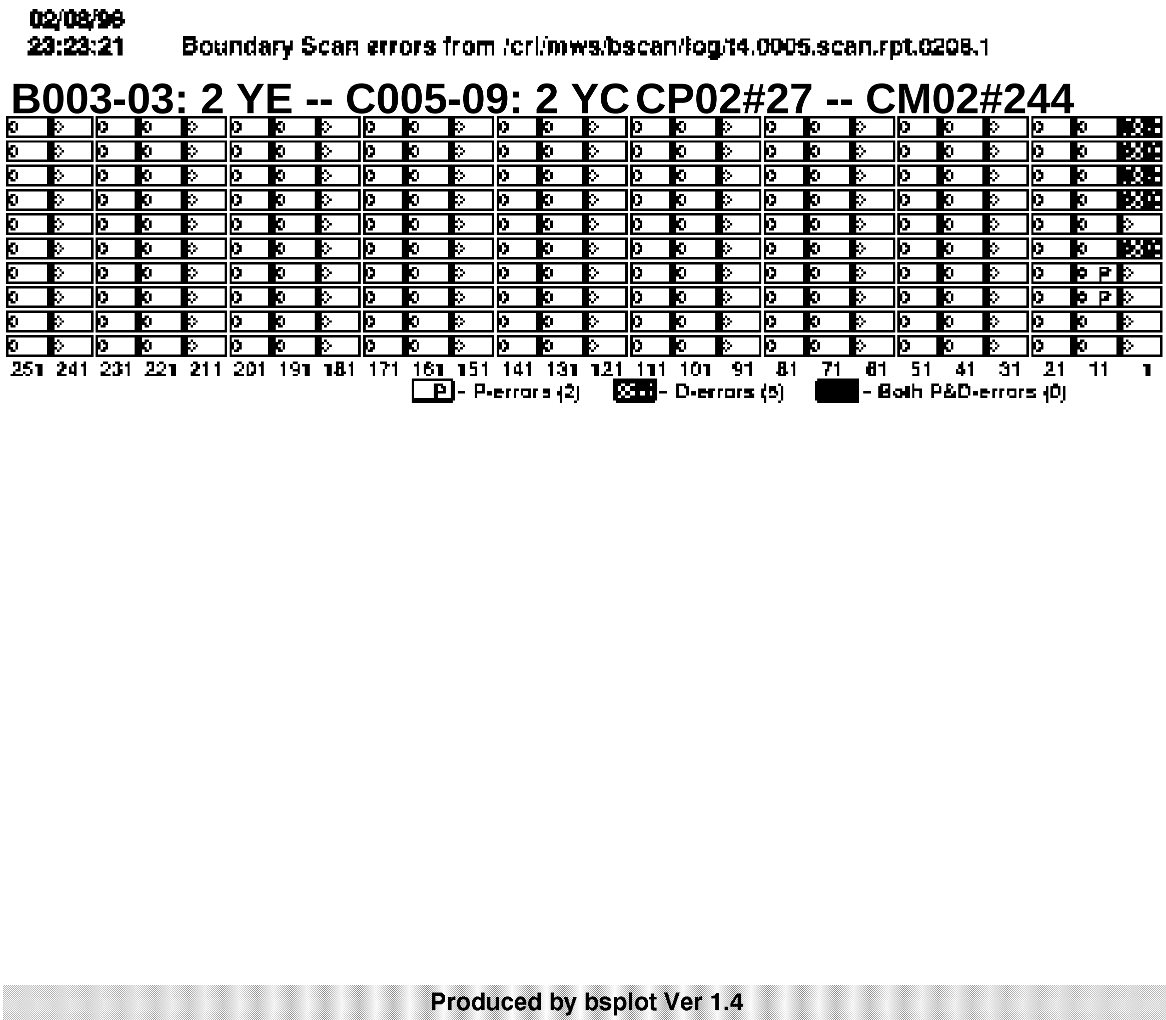

Figure 29. `bsplot` Output

Viewing and Printing the bsplot Output File

The `bsplot` program uses the following convention to name the output file: `usr/bscan/`*serial_number*`/log/scan.`*datetime*`.ps`. The `runbscan` output indicates the location of the `bsplot` output. You can view the file with the `pageview` command. You can print the file with the `lpr` command.

Interpreting the bsplot Output

Figure 30 describes the information that the `bsplot` output contains. The subsections that follow the figure describe the failure information that the `bsplot` output contains (the locations of the connector pads with `scan` errors, the type and serial number of each module, and the failing pad information).

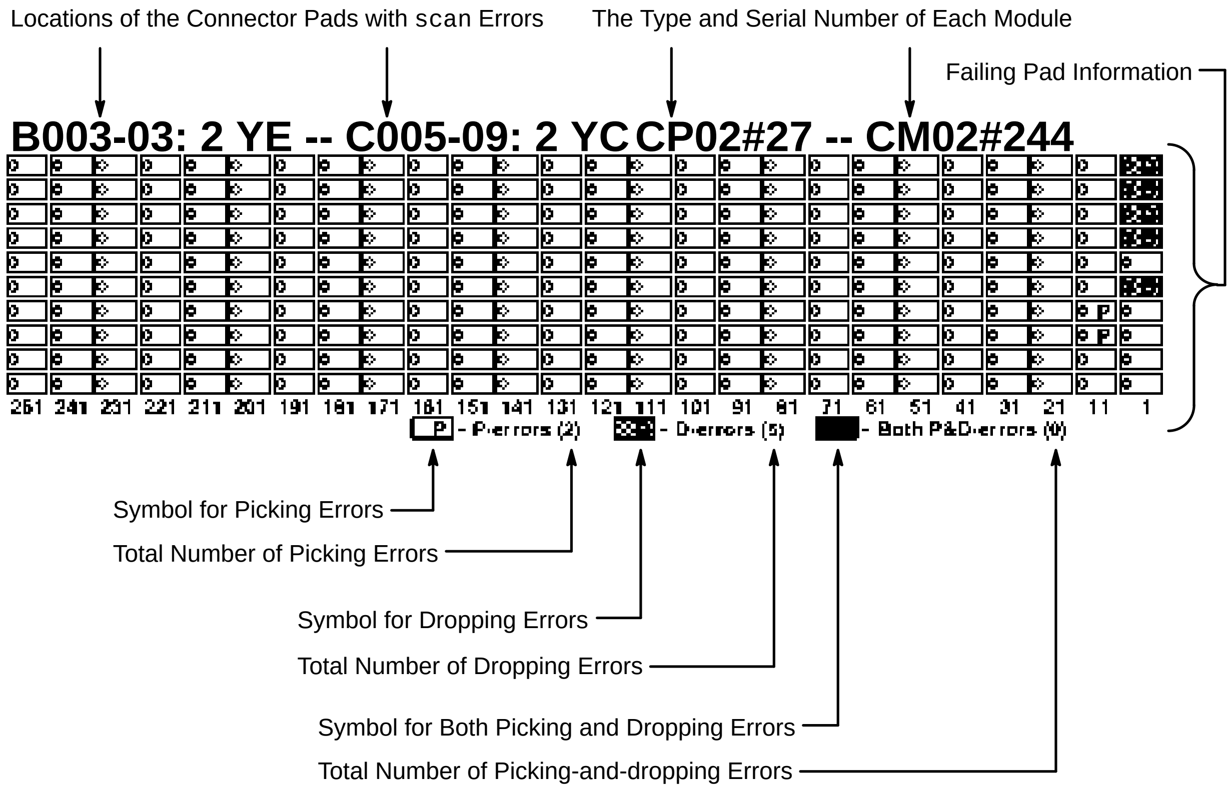

Figure 30. Information Shown in the `bsplot` Output

Locations of the Connector Pads with scan Errors

This information provides the location of the connector pads that have errors. The `bsplot` output contains the following location information for the modules:

- The physical location of the module in the mainframe. (For example, in Figure 30, the modules are located in slots B003-03 and C005-09).

- The board of the module on which the connector pad is located. (For example, in Figure 30, both connector pads are located on board 2 of the modules that contain the connector pads.)

- The location of the connector pad on which the failure is located. (For example, in Figure 30, the failure occurs between the connector pad at location YE and the connector pad at location YC.)

You can use this information to determine where the errors occurred. For example, in Figure 30, the errors occurred between the connector pad at location YE on board 2 of the module in slot B003-03 and the connector pad at location YC on board 2 of the module in slot C005-09.

The Type and Serial Number of Each Module

This information indicates the module type and serial number of each module. For example, in Figure 30, the two modules are a CP02 module (serial number 27) and a CM02 module (serial number 244).

Failing Pad Information

The failing pad information indicates the pads for which the `scan` test detected errors. Table 11 shows the symbols that `bsplot` uses to indicate errors.

Table 11. `bsplot` Error Symbols

Symbol	Description
	Indicates a pad with a picking error
	Indicates a pad with a dropping error
	Indicates a pad with both picking and dropping errors

The pad numbering starts with 1 in the lower-right corner, continues from the bottom to the top of each column, and ends with 260 in the upper-left corner. There are 10 pads in each column.

For example, in Figure 30, pad 5 has a dropping error.

railplot Program

The `railplot` program uses the output from the `scan` test to create a graphical representation of faulty EZIF connector contacts.

NOTE: The `runbscan` shell script does not automatically run the `railplot` program. You must manually run the `railplot` program.

Starting the railplot Program

You can start the `railplot` program from a UNIX command prompt with the following command on an MWS or SWS:

```
railplot
```

`railplot` prompts you for the information it uses.

Figure 31 shows an example of running `railplot` with a CRAY T94 mainframe.

```
sws7033$ railplot

@(#)railplot.c     1.4     16 Apr 1996

Enter the name of the boundary scan error file:  scan.04301912

Enter the name of the output plot file:  scan.04301912.rp


*************************************************************************
* This program requires the path of the connector needing repair.  The  *
* path is in the form <system location> <board> <connector location>.   *
* An example would be:  F006-21 1 YB                                    *
*************************************************************************

Enter the connector path system location:  B001-02

Enter the connector path board number:  1

Enter the connector path connector location:  YB

CP or SI side:  CP

RAILPLOT completed successfully.
```

Figure 31. Example of Using `railplot`

Figure 32 shows the `railplot` output file from this example.

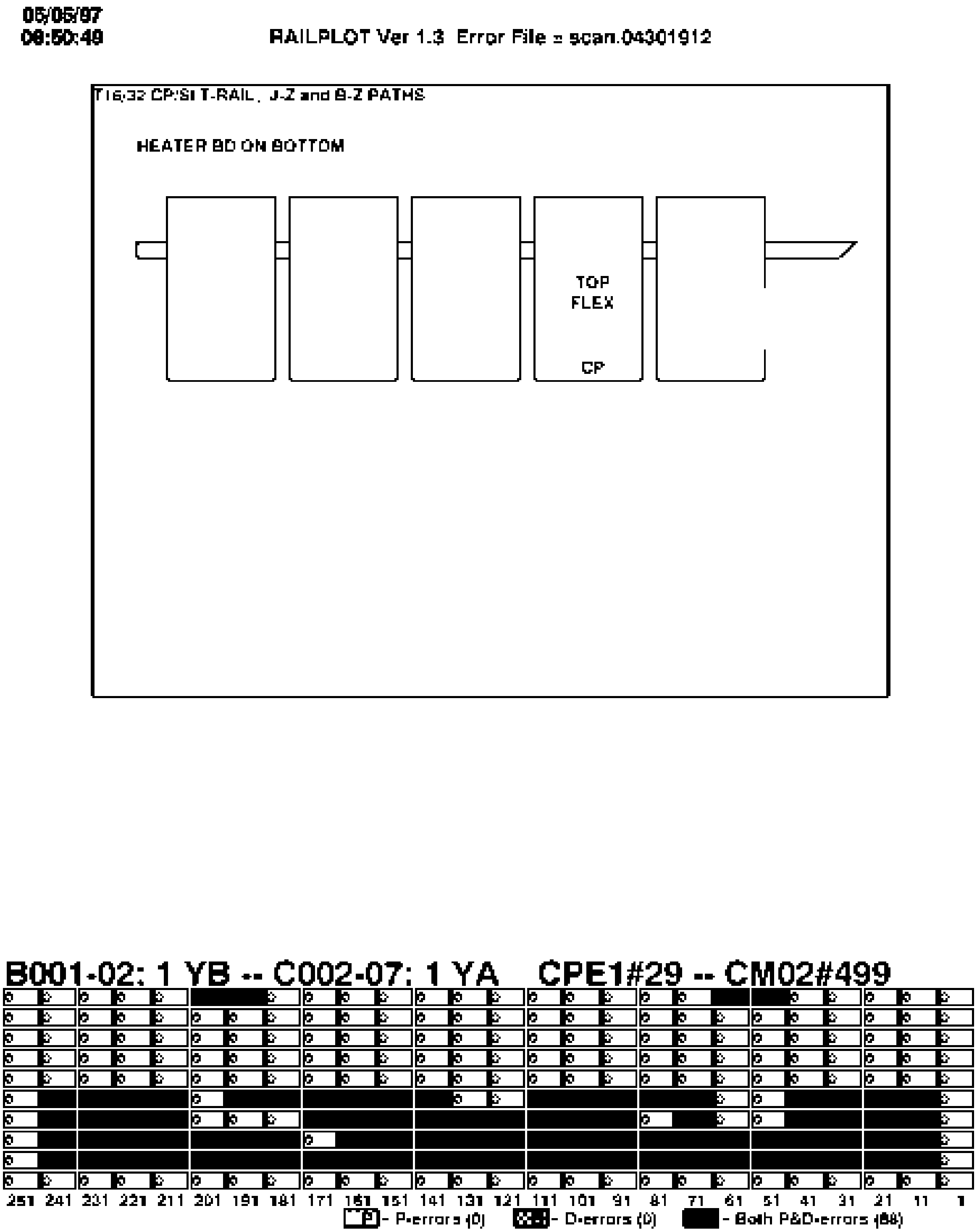

Figure 32. Example `railplot` Output File

Viewing and Printing the railplot Output File

The `railplot` program saves the PostScript with the filename that you specify at the prompt `Enter the name of the output plot file`. You can view the file with the `pageview` command. You can print the file with the `lpr` command.

Interpreting the railplot Output

The top portion of the output provides an illustration that shows the T-rail group and failing flex location for the defective T-rail assembly.

The bottom portion of the output shows the failing EZIF connector contacts on the T-rail assembly.

Figure 33 describes the information that the bottom portion of the `railplot` output contains. The subsections that follow the figure describe the failure information that the `railplot` output contains (the locations of the connector pads with `scan` errors, the type and serial number of each module, and the failing pad information).

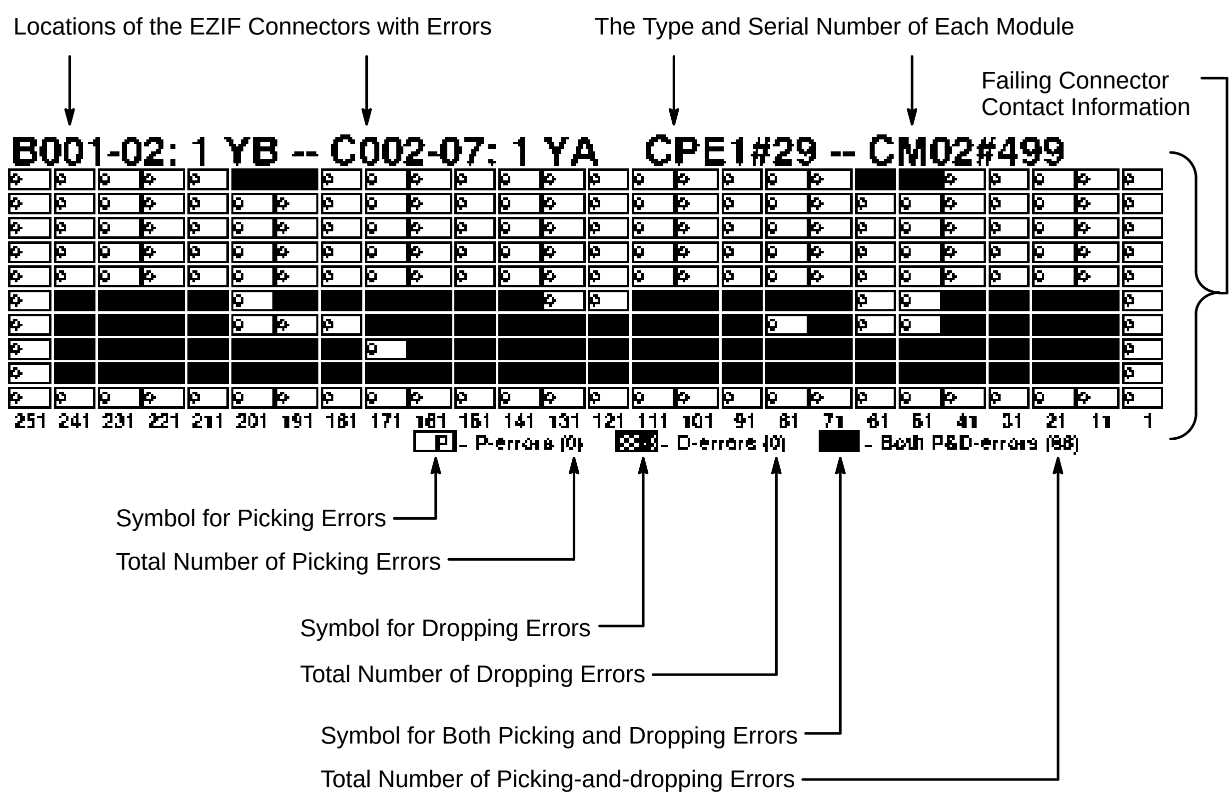

Figure 33. Information Shown in the Bottom Portion of the `railplot` Output

Location of the EZIF Connector with Errors

This information provides the location of the EZIF connectors that have errors. The `railplot` output contains the following location information for the modules:

- The physical location of the module in the mainframe. (For example, in Figure 33, the modules are located in slots B001-02 and C002-07). The failing EZIF connectors will be in the T-rail assembly that connects to one of these modules.
- The board of the module to which the connector is attached. (For example, in Figure 33, both EZIF connectors are attached to board 1 of the modules.)
- The location of the connector pad on the module that is connected to the failing EZIF connector. (For example, in Figure 33, the failure occurs in the EZIF connectors that are attached between the connector pad at location YB and the connector pad at location YA.)

You can use this information to determine where the errors occurred. For example, in Figure 33, the errors occurred in the EZIF connector that is attached between the connector at location YB on board 1 of the module in slot B001-02 and the connector at location YA on board 1 of the module in slot C002-07.

The Type and Serial Number of Each Module

This information indicates the module type and serial number of each module. For example, in Figure 33, the two modules are a CPE1 module (serial number 29) and a CM02 module (serial number 499).

Failing Connector Contact Information

The failing connector contact information indicates the EZIF connector contacts for which the `scan` test detected errors. Table 12 shows the symbols that `railplot` uses to indicate errors.

Table 12. `railplot` Error Symbols

Symbol	Description
	Indicates a pad with a picking error
	Indicates a pad with a dropping error
	Indicates a pad with both picking and dropping errors

The EZIF connector contact numbering starts with 1 in the lower-right corner, continues from the bottom to the top of each column, and ends with 260 in the upper-left corner. There are 10 connector contacts in each column.

For example, in Figure 33, connector contact 12 has picking and dropping errors.