Intel Corporation
5200 N.E. Elam Young Parkway
Hillsboro, OR 97124-6497

(503) 696-8080

intel®

September 1994

Dear Paragon™ Supercomputer Customer:

This package contains your Paragon™ system DIAG1.2.2 software. With this software installed on your Paragon™ Supercomputer, you can use the Paragon™ System Diagnostics on the diagnostic station. Please read through the documentation and distribute it to those intending to use the system diagnostics.

Before using your Paragon™ System:

- **Read this letter completely.**
- **Verify the contents of this package.**
- **Read the *Paragon™ System Diagnostics DIAG1.2.2 and DIAG1.2.3 Release Notes.***

Package Contents

Your Paragon™ system diagnostic software is provided as a separate shrinkwrapped package. Please verify that it includes the items listed in Table 1 (Installation Media) and Table 2 (Documentation). If any items are missing, or if you have any questions, please contact Intel Supercomputer Systems Division as described in the "Comments and Assistance" section.

Table 1. Installation Media

Description	Order Number
Cartridge tape labeled Paragon™ Diagnostic Software Release DIAG1.2.2	313080-003
Cartridge tape labeled Paragon™ Diagnostics Mass Install Release 3.0.0	312978-001
SCO®OPEN DESKTOP® R3.0.0 for the Paragon™ Diagnostic Workstation N1 Boot Disk	312974-001
SCO®OPEN DESKTOP® R3.0.0 for the Paragon™ Diagnostic Workstation N2 File System Disk	312975-001
SCO®OPEN DESKTOP® R3.0.0 for the Paragon™ Diagnostic Workstation M01 Master Install Disk	312976-001
Paragon™ Diagnostic Workstation Tests Release 1.0 disk	312787-001

Table 2. Documentation

Description	Order Number
Paragon™ System Diagnostics DIAG1.2.2 and DIAG1.2.3 Release Notes	313059-003
Paragon™ Diagnostics Reference Manual	312702-003
Paragon™ Diagnostics Troubleshooting Guide	313001-002

What is in This Release?

This release contains Paragon™ System Diagnostics DIAG1.2.2, Release 3.0.0 of the SCO Open Desktop, the *Paragon™ Diagnostic Reference Manual*, and the *Paragon™ Diagnostic Troubleshooting Guide*.

Restrictions and Limitations of DIAG1.2.2

Every effort has been taken to ensure the quality of this release, but at the time of shipment we are aware of some limitations. Please refer to the *Paragon™ System Diagnostics DIAG1.2.2 and DIAG1.2.3 Release Notes* for known limitations and available workarounds.

Installation

NOTE

Adding or removing any boards or components from your Paragon™ system can damage the system and may invalidate your warranty. Please contact Intel Supercomputer Systems Division Customer Support for assistance in answering your questions.

For directions on how to install the Paragon™ system diagnostic software, refer to Chapter 3 in the *Paragon™ System Diagnostics DIAG1.2.2 and DIAG1.2.3 Release Notes*.

Comments and Assistance

Intel Supercomputer Systems Division is eager to hear of your experiences with our products. Please call us if you need assistance, have questions, or otherwise want to comment on your Paragon system.

U.S.A./Canada Intel Corporation
Phone: 800-421-2823
Internet: support@ssd.intel.com

Intel Corporation Italia s.p.a.
Milanofiori Palazzo
20090 Assago
Milano
Italy
1678 77203 (toll free)

France Intel Corporation
1 Rue Edison-BP303
78054 St. Quentin-en-Yvelines Cedex
France
0590 8602 (toll free)

Intel Japan K.K.
Supercomputer Systems Division
5-6 Tokodai, Tsukuba City
Ibaraki-Ken 300-26
Japan
0298-47-8904

United Kingdom Intel Corporation (UK) Ltd.
Supercomputer System Division
Pipers Way
Swindon SN3 IRJ
England
0800 212665 (toll free)
(44) 793 491056 (*answered in French*)
(44) 793 431062 (*answered in Italian*)
(44) 793 480874 (*answered in German*)
(44) 793 495108 (*answered in English*)

Germany Intel Semiconductor GmbH
Dornacher Strasse 1
85622 Feldkirchen bei Muenchen
Germany
0130 813741 (toll free)

World Headquarters
Intel Corporation
Supercomputer Systems Division
15201 N.W. Greenbrier Parkway
Beaverton, Oregon 97006
U.S.A.
(503) 629-7600 (Monday through Friday, 8 AM to 5 PM Pacific Time)
Fax: (503) 629-9147

If you have comments about our manuals, please fill out and mail the enclosed Comment Card. You can also send your comments electronically to the following address:

techpubs@ssd.intel.com (Internet)

Intel Supercomputer Users Group

The Intel Supercomputer Users Group promotes the exchange of information among users. Intel strongly supports the Users Group and encourages participation in its activities, which include: Special Interest Groups (SIGs), an annual international users conference, an electronic mail task force, and a "freeware" library of user-contributed software, available electronically to all members of the Intel Supercomputer Users' Group. For membership information contact:

JoAnne Wold (503-629-5322)
joanne@ssd.intel.com (Internet)

Sincerely,

Steve Cannon

Product Marketing Manager
Intel Supercomputer Systems Division

September 1994
Order Number: 313059-003

Paragon™ System Diagnostics DIAG1.2.2 and DIAG1.2.3 Release Notes

Intel® Corporation

WARNING

Some of the circuitry inside this system operates at hazardous energy and electric shock voltage levels. To avoid the risk of personal injury due to contact with an energy hazard, or risk of electric shock, do not enter any portion of this system unless it is intended to be accessible without the use of a tool. The areas that are considered accessible are the outer enclosure and the area just inside the front door when all of the front panels are installed, and the front of the diagnostic station. There are no user serviceable areas inside the system. Refer any need for such access only to technical personnel that have been qualified by Intel Corporation.

CAUTION

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

LIMITED RIGHTS

Preface

This document describes both the DIAG1.2.2 and DIAG1.2.3 updates to the DIAG1.2 release of the Paragon System Diagnostics. The two updates contain identical changes to the diagnostic system, and differ only in which type of configuration files they use:

DIAG1.2.2	Uses the same configuration files (*SYSCONF.BIN*, etc.) as used in DIAG1.2. Use DIAG1.2.2 if you want to use the updated diagnostics while retaining the existing configuration files.
DIAG1.2.3	Uses new configuration files that include information about NIC and MDC revisions, as well as additional entries for MDC (Memory Daughtercards), SIO (SCSI-16) and CTLR (controllers to which SCSI devices are attached).

Organization

Chapter 1	This chapter describes the features of the Paragon XP/S system diagnostics update.
Chapter 2	This chapter describes the compatibility, limitations and workarounds for the Paragon XP/S system diagnostics.
Chapter 3	This chapter describes how to install the Paragon XP/S system diagnostic software.
Chapter 4	This chapter describes how to update Paragon system firmware.
Appendix A	This appendix describes how to install the Diagnostic Station SCO ODT operating system software.
Appendix B	This appendix contains information to help you interpret MRC and NIC register information in diagnostic error messages.

Appendix C — Appendix C contains manual pages for two new or enhanced diagnostic utilities. (**showconf** only applies to DIAG1.2.3.)

Appendix D — (Applies to DIAG1.2.3 only.) This is a revised appendix to replace Appendix D in the Diagnostic Reference Manual.

Notational Conventions

This manual uses the following notational conventions:

Bold — Identifies command names and switches, system call names, reserved words, and other items that must be used exactly as shown.

Italic — Identifies variables, filenames, directories, processes, user names, and writer annotations in examples. Italic type style is also occasionally used to emphasize a word or phrase.

`Plain-Monospace` — Identifies computer output (prompts and messages), examples, and values of variables. Some examples contain annotations that describe specific parts of the example. These annotations (which are not part of the example code or session) appear in *italic* type style and flush with the right margin.

Bold-Italic-Monospace — Identifies user input (what you enter in response to some prompt).

Bold-Monospace — Identifies the names of keyboard keys (which are also enclosed in angle brackets). A dash indicates that the key preceding the dash is to be held down *while* the key following the dash is pressed. For example:

<Break> **<s>** **<Ctrl-Alt-Del>**

[ ] — (Brackets) Surround optional items.

. . . — (Ellipsis dots) Indicate that the preceding item may be repeated.

| — (Bar) Separates two or more items of which you may select only one.

{ } — (Braces) Surround two or more items of which you must select one.

Applicable Documents

For more information, refer to the *Paragon™ Diagnostic Reference Manual* and the *Paragon™ Diagnostic Troubleshooting Guide*.

Comments and Assistance

Intel Supercomputer Systems Division is eager to hear of your experiences with our products. Please call us if you need assistance, have questions, or otherwise want to comment on your Paragon system.

U.S.A./Canada Intel Corporation
Phone: 800-421-2823
Internet: support@ssd.intel.com

Intel Corporation Italia s.p.a.
Milanofiori Palazzo
20090 Assago
Milano
Italy
1678 77203 (toll free)

France Intel Corporation
1 Rue Edison-BP303
78054 St. Quentin-en-Yvelines Cedex
France
0590 8602 (toll free)

Intel Japan K.K.
Supercomputer Systems Division
5-6 Tokodai, Tsukuba City
Ibaraki-Ken 300-26
Japan
0298-47-8904

United Kingdom Intel Corporation (UK) Ltd.
Supercomputer System Division
Pipers Way
Swindon SN3 IRJ
England
0800 212665 (toll free)
(44) 793 491056 (*answered in French*)
(44) 793 431062 (*answered in Italian*)
(44) 793 480874 (*answered in German*)
(44) 793 495108 (*answered in English*)

Germany Intel Semiconductor GmbH
Dornacher Strasse 1
85622 Feldkirchen bei Muenchen
Germany
0130 813741 (toll free)

World Headquarters
Intel Corporation
Supercomputer Systems Division
15201 N.W. Greenbrier Parkway
Beaverton, Oregon 97006
U.S.A.
(503) 629-7600 (Monday through Friday, 8 AM to 5 PM Pacific Time)
Fax: (503) 629-9147

Table of Contents

Chapter 1
Product Features

Chapter 2
Limitations and Workarounds

Chapter 3
Installation Instructions

Chapter 4
Updating Paragon™ System Firmware

Appendix A
Installing the SCO Operating System

Appendix B
MRC and NIC Register Definitions

Appendix C
New Manual Pages

Appendix D
Configuration File Format

List of Illustrations

List of Tables

Product Features 1

Features of These Releases

These releases of the Paragon™ system diagnostics include the following additional features and enhancements. Except as noted, they are described in this section:

- Support for systems containing MP node boards.
- New Node Tests.
- New Floating-Point Tests.
- New RPM Tests.
- New Mesh Tests.
- New NIC type and revision information is available.
- A new "network who" (**nwho**) command is available on the diagnostic station.
- A new level of testing has been added (the **scantest** utility) to test critical JTAG scan hardware that PSD uses. An enhanced scan driver has been added to support testing scan hardware.
- A new **cnv** conversion utility is available to convert between the different node-numbering schemes used in the Paragon system. This is described in a new manual page in Appendix C.
- The **showconf** utility has been expanded, and is now described in a manual page, also in Appendix C. (DIAG1.2.3 only.)
- New versions of the **flashutil** and **romver** utilities are included. They are described in new manual pages in Appendix C.

New Node Tests

Byte Access Test

This test writes all of memory from 0xC0100000 to the top of memory (except for the area used to store the firmware code) by bytes and then reads each byte and verifies that it contains the correct data. The data written to the bytes is an incrementing pattern starting at 0x0 and going to 0xFF before starting over at 0x0 again. The test initializes all of the memory to be tested with F's (using word writes) before starting the byte writes.

Byte Access with ECC Check Test

This test is the same as the byte access test except that the DP status register corresponding to each byte written or read is checked after the write or read and the single-bit error correct, double-bit error detect, and parity error bits are checked to ensure none of these errors have occurred. If one has, the test fails. Note that during reads, the data is checked for correctness before the DP status register bits are checked.

Address Alternate Test

This test tries to change as many address and data lines as possible on each successive write by writing opposite data patterns to high and low memory addresses and interleaving the high and low memory writes. The lowest address used is 0xC0100000, and the highest address used is the largest address below the memory used to store a copy of the firmware. The number of writes done to high and low memory is 0x1000. Data is written as double words, and the low memory address is incremented by 8 on each write while the high address is decremented by 8 on each write. The first iteration of the test writes low memory addresses with all zeros and high addresses with all F's. The second iteration writes low addresses with F's and high addresses with zeros. The data is then checked by words.

Memory Unique Test

This test writes each 32-bit word in memory with a different value (a counting pattern starting at zero and incrementing by one for each word). After writing each location, all locations are read and the data read is verified for correctness.

Long LTU Line Count Test

This test does LTU transfers when the system processor has modified data in both the transmit and receive buffers and verifies that the correct data is in memory after the LTU transfer.

The test follows this sequence of steps on each test case:

1. Flushes the cache.
2. Initializes three cache lines before the transmit and receive buffer starting addresses to 0xC1C1C1C1 and 0x76767676, respectively.
3. Causes one line of data from the transmit buffer to be put into the cache.
4. Writes new data into the transmit buffer to modify the one line already in cache.
5. Causes the corresponding line of data in the receive buffer to be cached.
6. Writes data to the receive buffer to cause the cached line to become modified.
7. Starts transmit and receive LTUs for some amount of data (the line count varies with the test case) and waits for them to complete.
8. Compares the data transferred to the receive buffer with that in the transmit buffer to make sure they match.
9. Checks the equivalent of three cache lines preceding and following the transmit and receive buffers to make sure all are correct.

Any miscompares indicate possible coherency problems.

The test does LTU transfers with line counts from 0x3 to 0x200, and does six transfers for each line count, with a different cache line containing modified data on each transfer. The modified cache line varies from $m - 3$ to $m + 2$, where m is the number of lines to be transferred by the next LTU transfer. Note that when the line count is greater than m, the line modified is not transferred on that test case. This provides a check that modified data near the data to be transferred does not somehow get transferred.

The test also checks 24 words (the equivalent of three cache lines) before and after the transmit and receive buffers to make sure they did not change from the pattern they were initialized to.

The transmit buffer used in this test starts at 0xF0100000, and the receive buffer starts at 0xF0200000. These data patterns are used to initialize the areas specified:

0xC1C1C1C1	Preceding transmit buffer
0x32323232	Following transmit buffer
0x76767676	Preceding receive buffer
0x55551111	Following receive buffer

LTU All Mod Line Count Test

This test is similar to the *LTU Line Count* test except that all lines in the transmit and receive buffers are modified before starting the LTU transfers instead of just one line. Also, the line counts in this test range from 0x2 to 0x200, and only one set of LTU transfers is done for each line count instead of six.

The test also checks 24 words before and after the transmit and receive buffers to make sure they did not change from the pattern they were initialized to.

The transmit buffer used in this test starts at 0xF0100000, and the receive buffer starts at 0xF0200000. The data patterns below are used to initialize the areas specified:

0xC1C1C1C1	Preceding transmit buffer
0x32323232	Following transmit buffer
0x76767676	Preceding receive buffer
0x55551111	Following receive buffer

LTU Invalidate Line Count Test

This test is also very similar to the *LTU Line Count* and *LTU All Mod Line Count* tests except that it modifies all lines in the transmit and receive buffers except one before starting the LTU transfers. This test has test cases for line counts from 0x3 to 0x200, and for each line count, six subcases are performed, with a different line being left unmodified in each subcase. In the subcases, the line left unmodified varies from m *-3* to m + *2*, where m is the line count for that test case. Only lines of data transferred during the LTU are checked for correctness.

The test also checks 24 words before and after the transmit and receive buffers to ensure they did not change from the pattern they were initialized to.

The transmit buffer used in this test starts at 0xF0100000, and the receive buffer starts at 0xF0200000. The data patterns below are used to initialize the areas specified:

0xC1C1C1C1	Preceding transmit buffer
0x32323232	Following transmit buffer
0x76767676	Preceding receive buffer
0x55551111	Following receive buffer

Node Test Errors

Byte Access Test

```
fail
exp = 0x...
act = 0x...
byte address = 0x...
Byte data starting at 0x...
```

If a data miscompare is detected, the above message is displayed. The byte data displayed shows sixteen bytes before the failure and fifteen bytes after.

```
Byte memory test:  Memory Size error
Unable to determine MP Memory Size
Node Status Reg = 0x...
```

When an MP node board is being tested, the above message indicates that the memory size couldn't be determined from the *type/rev* bits on the board.

```
Byte memory test:  Memory Size error
Invalid MP Memory Size
Memory Size = ... Mbytes
```

This message, only possible when testing MP boards, indicates that the value specified by the *type/rev* bits on the board was invalid.

Byte Access with ECC Check Test

```
fail
exp = 0x...
act = 0x...
byte address = 0x...
Byte data starting at 0x...
```

If a data miscompare is detected, the above message is displayed. The byte data displayed shows sixteen bytes before the failure and fifteen bytes after.

```
ECC or parity error detected after writing address 0x...
value written = 0x...
DP_STS_LO = 0x...
DP_STS_HI = 0x...
```

This message indicates that a single-bit ECC error, a multiple-bit ECC error, or a parity error was detected while writing to memory (the DP status registers are checked after each byte write). One of the DP status registers shows which DP detected the error and what the error was.

```
ECC or parity error detected after reading address 0x...
value written = 0x...
DP_STS_LO = 0x...
DP_STS_HI = 0x...
```

This message indicates that a single-bit ECC error, a multiple-bit ECC error, or a parity error was detected while reading from memory (the DP status registers are checked after each byte read). One of the DP status registers shows which DP detected the error and what the error was.

```
Byte Memory and ECC Check test:  Memory Size error
Unable to determine MP Memory Size
Node Status Reg = 0x...
```

When an MP node board is being tested, the above message indicates that the memory size couldn't be determined from the *type/rev* bits on the board.

```
Byte Memory and ECC Check test:  Memory Size error
Invalid MP Memory Size
Memory Size = ... Mbytes
```

This message, only possible when testing MP boards, indicates that the value specified by the *type/rev* bits on the board was invalid.

Address Alternate Test

```
Data miscompare:
   exp = 0x...
   act = 0x...
   adr = 0x...
```

This message is displayed when a location in the low part of memory written during this test is determined to have an incorrect value. If an address in the high part of memory has a miscompare, the message is almost the same:

```
(High range) data miscompare:
   exp = 0x...
   act = 0x...
   adr = 0x...
```

```
Alternating Patterns test:  Memory Size error
Unable to determine MP Memory Size
Node Status Reg = 0x...
```

When an MP node board is being tested, the above message indicates that the memory size couldn't be determined from the *type/rev* bits on the board.

```
Alternating Patterns test:  Memory Size error
Invalid MP Memory Size
Memory Size = ... Mbytes
```

This message, only possible when testing MP boards, indicates that the value specified by the *type/rev* bits on the board was invalid.

Memory Unique Test

```
Addr=0x...   expect=0x...   data=0x...
```

This message is displayed when the value in a 32-bit location doesn't match the data originally written to it. The last data value shown is the actual value read from the location.

```
Address Unique test:  Memory Size error
Unable to determine MP Memory Size
Node Status Reg = 0x...
```

When an MP node board is being tested, the above message indicates that the memory size couldn't be determined from the *type/rev* bits on the board.

```
Address Unique test:  Memory Size error
Invalid MP Memory Size
Memory Size = ... Mbytes
```

This message, only possible when testing MP boards, indicates that the value specified by the *type/rev* bits on the board was invalid.

Long LTU Line Count Test

```
timeout waiting for LTUs to complete
ltu line count = 0x...
modified line = 0x...
ltu line count:  ltu's didn't complete
```

This message indicates that either the receive or transmit LTU that was started in the current test case didn't finish in the allowed time. The "modified line" value indicates which cache line was modified for the test case that didn't pass.

```
ltu line count = 0x...  failed
DP_STS_HI=0x...
DP_LTU_CNT_LO=0x...
DP_LTU_CNT_HI=0x...
ltu line count = 0x...
modified line = 0x...
exp = 0x...
act = 0x...
failing rcv buf adr = 0x...        xmit buf adr = 0x...
xmit buf starts at 0x...
rcv buf starts at 0x...
Receive Buffer Data:
   .
   .
   .
Transmit Buffer Data:
   .
   .
   .
```

This message indicates that an error was detected when comparing the data in the transmit buffer to the data in the receive buffer, meaning the data was corrupted at some point during the transfer. Eight words preceding and seven following the failing word are also displayed.

```
Error:  a location xxx the yyy buffer was modified
exp = 0x...   act = 0x...   failing address = 0x...
```

In this message, *xxx* is either "preceding" or "following", and *yyy* is either "transmit" or "receive". The test initializes locations preceding and following the transmit and receive buffers before doing any LTU transfers, and these locations are checked after each LTU transfer is done. Normally, the LTUs do not modify any of these locations.

LTU Invalidate Line Count Test

```
timeout waiting for LTUs to complete
ltu line count = 0x...
invalid line = 0x...
x mod line count:  ltu's didn't complete
```

This message indicates that either the receive or transmit LTU that was started in the current test case didn't finish in the allowed time. The "invalid line" value indicates which cache line was invalidated for the test case that didn't pass.

```
ltu line count = 0x...  failed
DP_STS_HI=0x...
DP_LTU_CNT_LO=0x...
```

```
DP_LTU_CNT_HI=0x...
ltu line count = 0x...
invalid line = 0x...
exp = 0x...
act = 0x...
failing rcv buf adr = 0x...        xmit buf adr = 0x...
xmit buf starts at 0x...
rcv buf starts at 0x...
Receive Buffer Data:
   .
   .
   .
Transmit Buffer Data:
   .
   .
   .
```

This message indicates that an error was detected when comparing the data in the transmit buffer to the data in the receive buffer, meaning the data was corrupted at some point during the transfer. Eight words preceding and seven following the failing word are also displayed.

```
Error:  a location xxx the yyy buffer was modified
exp = 0x...   act = 0x...   failing address = 0x...
```

In this message, *xxx* is either "preceding" or "following", and *yyy* is either "transmit" or "receive". The test initializes a certain number of locations preceding and following the transmit and receive buffers before doing any LTU transfers, and these locations are checked after each LTU transfer is done. Normally, the LTUs do not modify any of these locations.

All Modify Line Count Test

```
timeout waiting for LTUs to complete
ltu line count = 0x...
all mod line count:  ltu's didn't complete
```

This message indicates that either the receive or transmit LTU that was started in the current test case didn't finish in the allowed time.

```
:ltu line count = 0x...  failed
DP_STS_HI=0x...
DP_LTU_CNT_LO=0x...
DP_LTU_CNT_HI=0x...
ltu line count = 0x...
exp = 0x...
act = 0x...
failing rcv buf adr = 0x...        xmit buf adr = 0x...
```

```
xmit buf starts at 0x...
rcv buf starts at 0x...
Receive Buffer Data:
    .
    .
    .
Transmit Buffer Data:
    .
    .
    .
```

This message indicates that an error was detected when comparing the data in the transmit buffer to the data in the receive buffer, meaning the data was corrupted at some point during the transfer. Eight words preceding and seven following the failing word are also displayed.

```
Error:  a location xxx the yyy buffer was modified
exp = 0x...   act = 0x...   failing address = 0x...
```

In this message, *xxx* is either "preceding" or "following", and *yyy* is either "transmit" or "receive". The test locations preceding and following the transmit and receive buffers before doing any LTU transfers, and these locations are checked after each LTU transfer is done. Normally, the LTUs do not modify any of these locations.

New Floating-Point Tests

New Floating-Point Tests have been added to the Node Tests directory.

General Math Tests

These tests provide a basic sanity check of the i860 floating-point unit.

The tests consist of the following sub-tests:

- Summation Test
- Anti-Summation Test
- Negative Summation Test
- Anti-Negative Summation Test
- Double Test
- Divide and Subtract Test
- Factorial Test

Summation Test

This test performs a cumulative sum of two double-precision floating-point numbers.

Two double-precision floating-point numbers, a and b, are first initialized to 0.0. Then, in a loop of 5, 1000.0 is added to b and b is added to a. At the end of the loop a is checked for 15,000.0.

This test exercises the **fadd.dd** instruction (floating-point add double precision source to double-precision destination).

Anti-Summation Test

This test performs a cumulative subtraction on two double-precision floating-point numbers.

Two double-precision floating-point numbers, a and b, are used. A is first initialized to 15000.0 and b is initialized to 0.0. Then, in a loop of 5, 1000.0 is added to b and b is subtracted from a. At the end of the loop a is checked for 0.0.

This test exercises the **fsub.dd** instruction (floating-point subtract double precision source to double-precision destination).

Negative Summation Test

This test performs a cumulative summation of two negative double-precision floating-point numbers.

Two double-precision floating-point numbers, a and b, are first initialized to 0.0. Then, in a loop of 5, 1000.0 is subtracted from b then b is added to a. At the end of the loop a is checked for -15000.0.

This test exercises the **fadd.dd** and **fsub.dd** instructions.

Anti-Negative Summation Test

This test performs a cumulative sum of a double-precision negative number and a double-precision positive number.

Two double-precision floating-point numbers, a and b, are used. A is first initialized to -15000.0 and b is initialized to 0.0. Then, in a loop of 5, 1000.0 is added to b then b is added to a. At the end of the loop a is compared with 15,000.0.

This test exercises the **fadd.dd** instruction (floating-point add double-precision source to double-precision destination).

Double Test

This test performs a cumulative sum of two double-precision floating-point numbers.

Two double-precision floating-point numbers, a and b, are used. A is first initialized to 0.0 and b is initialized to 100.0. Then, in a loop of 10, b is added to a and b is assigned the new value of a. At the end of the loop a is checked for 51200.0.

This test exercises the **fadd.dd** instruction (floating-point add double-precision source to double-precision destination).

Divide and Subtract Test

This test performs a cumulative divide of two double-precision floating-point numbers.

Two double-precision floating-point numbers, a and b, are first initialized to 51200.0. Then, in a loop of 10, a is multiplied by 0.5 (divided by 2) then a is subtracted from b. At the end of the loop a is compared with b.

This test exercises the **fmul.dd** instruction (floating-point multiply double-precision source to double-precision destination).

Factorial Test

This test performs a cumulative multiply and sum of two double-precision floating-point numbers.

A is first initialized to 1.0 and b is initialized to 2.0. Then, in a loop of 9, a is multiplied by b then 1.0 is added to b. At the end of the loop a is checked for 3628800.0 (9!).

This test exercises the **fmul.dd** instruction (floating-point multiply double-precision source to double-precision destination).

Rounding Mode Tests

These tests provide a check of the i860 floating-point unit adder pipeline.

The tests consist of the following sub-tests:

- Adder Pipe Restore (+n)
- Adder Pipe Restore (-0 .ds)
- Adder Pipe Restore (-0 .sd)

- Adder Pipe Restore (-0 .dd)
- Adder Pipe Restore (-0 .ss)

Adder Pipe Restore (+n)

This test checks that the floating-point unit adder pipeline can properly restore the adder pipeline in all 4 rounding modes using all positive single-precision floating-point numbers (+n).

First the adder pipe is filled and saved. Next the pipeline is checked to verify the save worked correctly. Finally, the pipe is restored in each of the 4 rounding modes and checked.

This test exercises the floating-point unit adder pipeline in all 4 rounding modes.

Adder Pipe Restore (-0 .ds)

This test verifies the adder pipeline maintains the integrity of -0 while -0 is being shifted through the pipe and being converted from double precision to single precision.

First the adder pipeline is filled with double-precision -0 (pfamov.ds). Then the adder pipeline is emptied into the destination registers. Finally, the destination registers are compared with single-precision -0.

This test is repeated for each of the 4 rounding modes.

Adder Pipe Restore (-0 .sd)

This test verifies the adder pipeline maintains the integrity of -0 while -0 is being shifted through the pipe and being converted from single precision to double precision.

First the adder pipeline is filled with single-precision -0 (pfamov.sd). Then the adder pipeline is emptied into the destination registers. Finally, the destination registers are compared with double-precision -0.

This test is repeated for each of the 4 rounding modes.

Adder Pipe Restore (-0 .dd)

This test verifies the adder pipeline maintains the integrity of double-precision -0 while -0 is being shifted through the pipe.

First the adder pipeline is filled with double-precision -0 (pfamov.dd). Then the adder pipeline is emptied into the destination registers. Finally, the destination registers are compared with double-precision -0.

This test is repeated for each of the 4 rounding modes.

Adder Pipe Restore (-0 .ss)

This test verifies the adder pipeline maintains the integrity of single-precision -0 while -0 is being shifted through the pipe.

First the adder pipeline is filled with single-precision -0 (pfamov.ss). Then the adder pipeline is emptied into the destination registers. Finally, the destination registers are compared with single-precision -0.

This test is repeated for each of the 4 rounding modes.

New RPM Tests

Two new RPM tests have been added to the Message Network Tests directory.

RPM Global Clock Test

This test verifies the diagnostic station's global clock and the RPM's global clock counter on the nodes. The test follows these steps:

- The global clock is stopped and the nodes verify that the RPM's global counters are not incrementing.
- A local RPM global counter reset is issued and the nodes verify that the counters are cleared.
- The global clock is then started and the nodes verify that the RPM's global counters are incrementing.
- Finally, the global clock is stopped again and the host verifies that all nodes' global counters are within 1 micro second from each others.
- The bootnode is used as a reference for checking the global counters accuracy. If the bootnode is logically ignored, the next available node is selected.
- If all nodes counters are not incrementing, it is assumed that the global clock on the diagnostic station's Corelis card is not functioning.

GP nodes without RPM modules fail this test.

RPM Global Clock Test Error Messages

`Error in starting the global clock`

Internal program error.

`Error in stopping the global clock`

Internal program error.

`Since all nodes reported an error with Global Clock, The problem is most likely caused by a bad Corelis card.`

Replace Corelis card in the diagnostic station.

`RPM Global Counter Accuracy Error on Node <nodenum> Counter(H,L) On Node <nodenum> = <highvalue><lowvalue> On Reference Node <refnode> = <highvalue><lowvalue>`

<*nodenum*>'s global counter is not in the acceptable accuracy range compared to the reference node. If all nodes are not in the reference node range, then the reference node may be at fault.

`Node <nodenum> does not have an RPM on it`

No RPM module was detected on the specified node.

`RPM Global Clock Error on Node <nodenum> The Global Counter was incrementing After the global clock stopped Global Counter 1st read(H,L)= <highvalue><lowvalue> Global Counter 2nd read(H,L)= <highvalue><lowvalue>`

The global counter of <*nodenum*> kept counting, even though the global clock was stopped. The 56-bit counter is displayed for 2 consecutive reads.

`RPM Global Clock Error on Node <nodenum> The RPM Global Counter didn't clear after a RPM Counter Reset Global Counter (H,L)= <highvalue><lowvalue>`

A local RPM reset didn't clear the global counter of <*nodenum*>.

`RPM Global Clock Error on Node <nodenum> The Global Counter was NOT incrementing After the global clock started Global Counter 1st read(H,L)= <highvalue><lowvalue> Global Counter 2nd read(H,L)= <highvalue><lowvalue>`

The global counter of <*nodenum*> was not counting, even though the global clock was started. The 56-bit counter is displayed for 2 consecutive reads.

RPM Mesh Counters Test

This test verifies that all four ports (North, South, East and West) of the RPM mesh counters are functional. The mesh network needs to be functional before you run this test. You can verify the mesh by running other mesh tests.

The test follow these steps:

- A local reset to the four mesh counters is issued.
- The nodes verify that the counters are cleared.
- *Neighbor Concurrent Comm* test is started with 1k message length to generate some mesh traffic.
- If the mesh test passes, the nodes verify that the RPM mesh counters are incrementing.

RPM Mesh Counters Test Error Messages

```
RPM MRC North Counter of Node <nodenum> did not clear after reset MRC North
Counter = <value>
```

```
RPM MRC South Counter of Node <nodenum> did not clear after reset MRC South
Counter = <value>
```

```
RPM MRC East Counter of Node <nodenum> did not clear after reset MRC East Counter
= <value>
```

```
RPM MRC West Counter of Node <nodenum> did not clear after reset MRC West Counter
= <value>
```

A local RPM counter reset, didn't clear the corresponding RPM mesh counter.

```
Neighbor Conc Comm test failed. This test is used by RPM MESH Counters test to
generate Mesh Traffic. I am Node <nodenum>
```

The *Neighbor Concurrent Comm* test failed.

```
RPM MRC North Counter of Node <nodenum> is not incrementing MRC North Counter =
<value>
```

```
RPM MRC South Counter of Node <nodenum> is not incrementing MRC South Counter =
<value>

RPM MRC East Counter of Node <nodenum> is not incrementing MRC East Counter =
<value>

RPM MRC West Counter of Node <nodenum> is not incrementing MRC West Counter =
<value>
```

The corresponding RPM mesh counter is not incrementing even though mesh traffic was generated.

New Mesh Tests

Node Background Task

This task is executed by the USR processor(s) under the supervision of the SYS processor. All background tests operate on base board memory beginning on an address just above that used by the "Message Passing Tests" and ending 1Mbyte below the top of memory. The type of background task can be selected by the operator by setting the environment variable USR_TASK to one of the recognized values.

D	Disable background task (default)
0	Run all tests (currently 1 - 4)
1	Dword Alternate Ones
2	Byte Alternate Ones
3	Dword Sliding Ones
4	Byte Sliding Ones
5	CPU Spin Loop

Background Task Error Messages

The failure messages from the following background memory tests have a common format. The first field is a tag that identifies the area of the test software the message originated from, as follows:

AD	Dword Alternate Ones,
AB	Byte Alternate Ones,

SD	Dword Sliding Ones,
SB	Dword Sliding Ones,
ERR1/ERR2	1st compare/2nd compare.

The second field identifies the node that reported the error. Next is the user CPU that detected the memory fault. The last three fields report the address, expected data, and received data that caused the comparison to fail.

Dword Alternate Ones

Initializes memory with a pattern of 0x55555555. Then reads the current location, performs a bit-wise inversion of the 32-bit data and writes this pattern to the current location. The current location is tested for the new pattern, if the compare fails and error is printed and the test stops. The test makes five passes over the memory range.

```
AD ERR1: 01A15(79), CPU0, Adr=F0602CC0, Exp=AAAAAAAA, Rcv=AAABAAAA
```

Byte Alternate Ones

Initializes memory with a pattern of 0x55. Then reads the current location, performs a bit-wise inversion of the 8-bit data and writes this pattern to the current location. The current location is tested for the new pattern, if the compare fails, an error is printed and the test stops. The test makes five passes over the memory range.

```
AB ERR1: 01A15(79), CPU0, Adr=F0602CC2, Exp=AA, Rcv=AB
```

Dword Sliding Ones

Initializes memory with the pattern 0x00000001, then tests the current location for the current 32-bit pattern. If the compare fails, an error is printed and the test stops. Otherwise, the test continues with the next pattern in the sequence being written at the current location. If that comparison passes, the test continues to the next location. The test makes 63 passes over the memory range. The pattern used follows this progression:

```
0x00000001
0x00000003
.   .   .
0x7FFFFFFF
0xFFFFFFFF
0xFFFFFFFE
.   .   .
0x80000000
```

```
0x00000000
```

From the first compare:

```
SD ERR1: 01A15(79), CPU0, Adr=F0602CC0, Exp=000007FF, Rcv=03FF07FF
```

From the second compare:

```
SD ERR2: 01A15(79), CPU1, Adr=F0603044, Exp=FFF80000, Rcv=FFF90000
```

Byte Sliding Ones

Initializes memory with the pattern 0x01, then tests the current location for the current 8-bit pattern. If the compare fails, an error is printed and the test stops. Otherwise, the next pattern in sequence is written at the current location and the test continues. If that compare passes, the test continues to the next location. The test makes 15 passes over the memory range. The pattern used follows this progression:

```
0x01
0x03
.  .  .
0x7F
0xFF
0xFE
.  .  .
0x80
0x00
```

From the first compare:

```
SB ERR1: 01A15(79), CPU0, Adr=F0602CC2, Exp=7F, Rcv=3F
```

From the second compare:

```
SB ERR2: 01A15(79), CPU1, Adr=F0603041, Exp=F8, Rcv=F9
```

Background Task Error messages (General)

```
ERROR: 00B13(29) USR0: No Response.
```

This message indicates that the node's SYS processor has been unable to start the background task on the USR processor indicated.

```
ERROR: 00B13(29) USR0: Synchronization Failure.
```

During the initialization of the background the USR processor(s) are asked to wait at this point for a signal from the SYS processor. The error indicates that the identified USR processor could not complete the handshake with the SYS processor as required.

```
ERROR: 00B13(29) USR0: No activity detected
```

After the background task has been started on the USR processor(s) the SYS processor tests an activity counter that the USR processor is required to update. This error means that there has been no change in the counter for the last three checks.

Message Network Test Error Messages

General Message Network Test Errors

```
RECV TIMEOUT: 00B15(31), Xcnt=4, Rcnt=278, NIC:<HI 32bits> <LO 32bits>

RMIP: 00C01(33)->00B15(31), LTU0: 15  XMIP: 00B15(31)->01A14(78), LTU1: 451
```

The last message the destination tried to receive could not complete before the time-out for a single message expired. The default time-out value is 10 seconds. The message identifies the node reporting the error, the number of messages left to send and/or receive during this test and the NIC status of this node.

Following the error message the node reports if there is a `Receive Message In Progress (RMIP)` and/or a `Xmit Messages In Progress (XMIP)`. These entries identify the source, destination and LTU line count associated with the message. This line does not appear if no messages are in progress.

```
ERROR: 00B15(31), Xcnt=4, Rcnt=278, NIC:<HI 32bits> <LO 32bits>

RMIP: 00C01(33)->00B15(31), LTU0: 15  XMIP: 00B15(31)->01A14(78), LTU1: 451
```

During all testing the node keeps track of the messages left to send and/or receive. This message appears if the time limit for the test is reached before all messages have been processed. The message identifies the node reporting the error, the number of messages left to send and/or receive during this test and the NIC status of this node.

Following the error message, the node reports if there is a `Receive Message In Progress (RMIP)` and/or a `Xmit Messages In Progress (XMIP)`. These entries identify the source, destination and LTU line count associated with the message. This line does not appear if no messages are in progress.

```
DP ERROR: 00B15(31), DP_STATUS(h,l) = 0x<HI 32bits>, 0x<LO 32bits>
```

This message is printed if any one of three hardware error bits in the DP status register are set.

Bit 8	Single-bit ECC error corrected.
Bit 9	ECC Error detected (multiple bits).
Bit 10	Bus parity error detected.

Of these three bits, a non-corrected ECC error (bit 9) is reported as an error by the software—the other two are treated as information only. They are printed to the screen and logged in *psd.def*, but not in *psd.log*.

```
Cold NIC Timeout: 00B15(31) :Msg 00C01(33)->00B15(31)
  NIC status: <HI 32bits> <LO 32bits>,  LTU Lines left: 15

LTU Re-Start Failed: Msg 00C01(33)->00B15(31)
  NIC status: <HI 32bits> <LO 32bits>: Line Count = 15
```

Both these messages indicate that the process of restarting a stalled incoming message has been unsuccessful. In both cases the message identifies the source and destination of the receive message in progress. The NIC status of the destination node appears along with the number of lines remaining in the current LTU operation.

```
LTU work-a-round FAILED: Line_cnt <-1>:
  NIC: <HI 32bits> <LO 32bits>
```

If, in the process of restarting the LTU, we decrement the line count past zero, this message is printed.

```
INVALID DEST: 00B15(31) : Msg 00C01(33)->00B14(30)
  NIC status: <HI 32bits> <LO 32bits>

MSG SIZE ERROR: 00B15(31) : Msg 00C01(33)->00B15(31)
  NIC status: <HI 32bits> <LO 32bits>
```

Both these messages indicate that information received in the header of a message is not consistent with the node ID or current message size in affect at the time. In both cases the message identifies the source and destination of the receive message in progress. The NIC status of the destination node is printed on the second line.

Test Initialization Errors

```
Message destination 00B15(31), conflicts node value 95
```

The node value sent to the node while *Initializing Node IDs* conflicts with the node identified as the message destination.

Test Command Errors

```
Mesh command error: recv_bumper(): NIC status: <HI 32bits> <LO 32bits>"

Mesh command error: recv_header(): NIC status: <HI 32bits> <LO 32bits>"

Mesh command error: TxNotEmpty: NIC status: <HI 32bits> <LO 32bits>"
```

Any one of these messages indicate that the node could not process the last test commands broadcast to it over the mesh by the *BOOT_NODE*.

New NIC Type and Revision Information

hwcfg now puts information about the type and revision level of the NIC (Network Interface Chip) on each node into *SYSCONFIG.TXT*.

gencfg processes NIC type and revision information.

cfgpar puts the same information into *SYSCONFIG.BIN*.

showconf displays the new NIC information.

New Network Who (nwho) Command

The SCO operating system now includes **nwho** utility, which displays which machine users are logged in from. Refer to the online manual page for more information.

New JTAG Scan Tests

A new **scantest** utility has been added to verify hardware that PSD uses. The **scantest** utility provides the ability to verify the entire Paragon system's JTAG scan bus for functional and timing related failures. This test first evaluates the entire connection between the diagnostic station and the first connection within the first cabinet (backplane and power controller boards). Then the utility evaluates the scan bus row by row starting with "A". In the event of a failure, appropriate failure information is displayed to provide the means for quickly isolating and replacing bad FRU(s). It is recommended that this utility be run at system installation prior to **hwcfg**, which relies on the scan bus to collect configuration information.

Refer to the manual page for **scantest** in Appendix C for details.

The **scantest** utility provides the mechanism to accurately verify the JTAG scan bus throughout the entire Paragon system. Verification starts with the diagnostic station's diaboard and its cable interconnection to the four subsequent backplanes and power controller. Upon successful execution of the first part, each row's entire scan chain is verified, starting with row A. There are two levels of messages; the default is *error only,* while the **-v** option selects *verbose messages* (action and error messages). A summary of row scan failure information, along with a suggested FRU(s), is displayed.

Usage errors initiate an illustration of the correct syntax. The **-c *<cabinet count>*** option is required. The others depend on the system and the operator's choice.

```
Warning: Scan Test expecting full system - Cable x not present
```

Refer to Scan Library Error Message `xx: scan cables A-D are not connected to backplanes` in the *Diagnostic Troubleshooting Guide*. Cable E is checked for connection to the Power Controller board. If the system is not fully configured with 5 rows (A through E), then use option **-i**.

```
Unable to Select Corelis Cable Row xx
```

Refer to Scan Library Error Message `xx: Error selecting backplane bb in cabinet cc`. This also applies to the Power Controller on Row E.

```
Error: Corelis Cable to Primary Linker in Row xx Failed.
```

Check the diaboard cables for seating. The possible bad FRU could be the individual backplane or power controller boards identified by the row designator.

```
Failure in Backplane Row xx SSP xx of Cabinet xx

Failure in Backplane Row xx Node xx of Cabinet xx

Failure in Power Controller SSP xx of Cabinet xx
```

Refer to the scan failure summary that summarizes these failures along with possible bad FRU identification starting points for debug.Content-Type: text

New Processor Tests

Register / MP3 UART Test

This test checks the UART functionality on the MP3 nodes. The UART is an Intel 82510 serial controller. The test puts the controller into internal loopback mode, which causes the contents of the transmit FIFO of the chip to be received immediately by the receive FIFO. The data received is compared with the data sent.

Error Messages

```
Serial Interface RESET Failed
Line Status Register following reset = xxxxxxxx
```

The software reset of the serial controller through the *Internal Command Register* has failed. The *Line Status Register* value is displayed. The definition of the Line Status Register bits are as follows:

Bit 7 Reserved

Bit 6 *TxST* - Transmit Machine Status Bit

If high, it indicates that the *Transmit Machine* is in Idle state. Idle may indicate that the TxM is either empty or disabled.

Bit 5 *TFSt* - Transmit FIFO Status

This message indicates that the *Transmit FIFO* level is equal to or below the *Transmit FIFO* Threshold.

Bit 4 *BkD* - Break Detected

This bit indicates that a *Break Condition* has been detected—*RxD* input was held low for one character frame plus a stop bit.

Bit 3 *FE* - Framing Error Detected

This bit indicates that a received character did not have a valid stop bit.

Bit 2 *PE* - Parity Error Detected

This bit indicates that a received character had a parity error.

Bit 1 *OE* - Overrun Error

This bit indicates that a received character was lost because the *Rx FIFO* was full.

Bit 0 *RFIR* - Receive FIFO Interrupt Request

This bit indicates that the *Rx FIFO* level is above the *Rx FIFO Threshold.*

```
Serial Interface Internal Loopback Test Failed
Sent: y & Got: z
```

The internal loopback test failed. The data that was sent on the transmitter side (*TXD*) didn't match the data received on the receiver side (*RXD*). The mismatched values are displayed.

Multiple Processor / System Processor and LTU Receive Test

This test checks that if data in the system processor's cache is modified and an LTU transfer is done from different memory locations than the data in cache, the data from the cache is NOT the data transferred during the LTU.

The test follows these steps:

1. Two areas in memory are initialized with known data patterns (area1 with pattern1, area2 with pattern2).
2. Area2 is read to fill the cache with data.
3. The data in the cache is modified with pattern3.
4. An LTU transfer is performed from area1 to area2.
5. The memory in area2 is checked to make sure that it contains pattern1.

Error Messages

```
LTU transmit or receive operation did not complete in allowed time
either XMIT_LTU_DONE or RCV_LTU_DONE bit not set in DP_STS_HI
DP_STS_HI = xx

Data in transmit buffer incorrectly modified during LTU
Expected = xx
Actual   = yy
Transmit buffer address = zz

Data transferred during LTU did not match expected value
Expected = xx
Actual   = yy
Receive buffer address = zz

A single or multiple bit error or a parity error was detected
DP_STS_LO = xx
DP_STS_HI = xx
DP LO internal register 4 = xx
DP HI internal register 4 = xx
Lower 12 bits of register corresponding to DP that detected error
are address bits indicating where error occurred

Data preceding LTU transmit buffer was modified
Expected = xx
Actual   = yy
Failing address = zz
```

```
xmt buffer address = zz

Data preceding LTU receive buffer was modified
Expected = xx
Actual   = yy
Failing address = zz
rcv buffer address = zz

Data following LTU transmit buffer was modified
Expected = xx
Actual   = yy
Failing address = zz
xmt buffer address = zz

Data following LTU receive buffer was modified
Expected = xx
Actual   = yy
Failing address = zz
rcv buffer address = zz
```

Multiple Processor / User Processor and LTU Receive Test

This test checks that if data in the user processors' cache is modified and an LTU transfer is done from different memory locations than the data in cache, the data from the cache is NOT the data transferred during the LTU.

The test follows these steps:

1. Two areas in memory are initialized with known data patterns (area1 with pattern1, area2 with pattern2).
2. Area2 is read to fill the cache with data.
3. The data in the cache with pattern3.
4. An LTU transfer is performed from area1 to area2.
5. The memory in area2 is checked to verify that it contains pattern1.

Error Messages

```
User(x) processor did not modify data in its cache in allowed time
Value of memory flag location = xx

LTU transmit or receive operation did not complete in allowed time
either XMIT_LTU_DONE or RCV_LTU_DONE bit not set in DP_STS_HI
```

```
DP_STS_HI = xx
Testing USR(x) processor

Data in transmit buffer incorrectly modified during LTU
Expected = xx
Actual   = yy
Transmit buffer address = zz
Testing USR(x) processor

Data transferred during LTU did not match expected value
Expected = xx
Actual   = yy
Receive buffer address = zz
Testing USR(x) processor

A single or multiple bit error or a parity error was detected
DP_STS_LO = xx
DP_STS_HI = xx
DP LO internal register 4 = xx
DP HI internal register 4 = xx
Lower 12 bits of register corresponding to DP that detected error
are address bits indicating where error occurred
Testing USR(x) processor

Data preceding LTU transmit buffer was modified
Expected = xx
Actual   = yy
Failing address = zz
xmt buffer address = zz
Testing USR(x) processor

Data preceding LTU receive buffer was modified
Expected = xx
Actual   = yy
Failing address = zz
rcv buffer address = zz
Testing USR(x) processor

Data following LTU transmit buffer was modified
Expected = xx
Actual   = yy
Failing address = zz
xmt buffer address = zz
Testing USR(x) processor

Data following LTU receive buffer was modified
Expected = xx
Actual   = yy
```

```
Failing address = zz
rcv buffer address = zz
Testing USR(x) processor
```

Multiple Processor / User Processors Coherency Test

This test checks coherency between the 2 MP3 user processors. When one user processor has modified data in its cache and the other user processor writes that location, the data in the first user processor's cache is verified to be invalidated.

Error Messages

```
User0 processor started but did not complete its code
Flag0 value = xx

User1 processor started but did not complete its code
Flag1 value = xx

User0 and User1 processors did not start running
Flag0 value = xx
Flag1 value = xx

User0 processor TIMED OUT waiting for a handshake signal from USR1
Status value = xx

User1 processor TIMED OUT waiting for a handshake signal from USR0
Status value = xx

Usr0 proc cache not invalidated when usr1 proc wrote to a cached
location modified by the usr0 proc
Value in usr0 proc cache = xx
Value in memory = xx
Memory address used = xx

Modified data in usr0 proc cache not written to memory when usr1
proc did a read
Data read by usr1 proc = xx
Data in usr0 proc cache = xx
Memory address used = xx

Usr1 proc cache not invalidated when usr0 proc wrote to a cached
location modified by the usr1 proc
Value in usr1 proc cache = xx
Value in memory = xx
Memory address used = xx
```

```
Contents of usr1 proc cache not written to memory when usr0 proc
did a read of a cached address modified by the usr1 proc
Data read by usr0 proc = xx
Data in usr1 proc cache = xx
Memory address used = xx
```

Limitations and Workarounds 2

This chapter contains known limitations and workarounds in this release of the Paragon™ system diagnostics (PSD). Please read this chapter before you use the diagnostic software.

Note

The Paragon system diagnostics should not be running when the Paragon system operating system is to be booted.

Hard Reset Error Recovery

If you use the reset button on an XP/E system diagnostic station to do a hard reset, or cycle the power on the diagnostic station of any system, you will make an "ungraceful" exit from Paragon System Diagnostics.

When **psd** begins its initialization, it saves a copy of the *SYSCONFIG.BIN* file into *SYSBIN.ORIG*. If the diagnostic station reports:

```
Cannot save the binary configuration file: /u/paragon/diag/SYSBIN.ORIG already
exists
```

Remove this file to run **psd** without error.

Hardware Revision Levels

The minimum hardware revision level supported by this release of PSD is listed in Table 2-1. Refer to the Comments and Assistance section in the Preface for instructions on contacting Intel SSD Customer Service for this information.

Table 2-1. Compatible Hardware Revision Levels for DIAG1.2.2 and DIAG1.2.3

Field Replaceable Unit (FRU)	Component	Revision	Comments
GP Node	Node Board	Fab7-011 and up	
	Flash EPROM	V3.3	Contains the correct address to check for the existence of an MDC.
		V3.2	
		V3.1	
	NIC ASIC	A step	
		B step	
MP Node	Node Board	Fab 2.1	
	Flash EPROM	V2.0	
	BIC ASIC	B step	
MDC	MDC Board	Fab 3	16- and 32-Mbyte versions are available as Fab 3
	Node Board	as per GP	Only used with GP Nodes
	Flash EPROM	V1.4	Needs GP 3.3 firmware
MIO	Node Board(s)	as per GP and MP	See above entry
	Daughtercard	Fab2	
		Fab3	
	Flash EPROM	tftp - 1.13 MIO - 1.0	
		tftp - 1.13 MIO - 1.1	Adds Ethernet tests and fixes SCSI and asynchronous bugs
		tftp - 1.13 MIO - 1.2	Adds Ethernet tests and fixes SCSI and asynchronous bugs
		tftp - 1.13 MIO - 1.3	Fixes Ethernet tests

Table 2-1. Compatible Hardware Revision Levels for DIAG1.2.2 and DIAG1.2.3

Field Replaceable Unit (FRU)	Component	Revision	Comments
HIPPI Board	Node Board(s)	GP Node - Fab8 and up	
		MP Node - Fab 2.1	
	Daughtercard	Fab3	
	Flash EPROM	V1.2	
RAID Controller	Controller Board	92/01	PSD 1.2 provides RAID OS 3.06
Disk Drives	Maxtor	MXT-1240	Intel P/N 317961-001
	Seagate	ST31200N	Intel P/N 340573-001
Tape Drive	HP	3470	Intel P/N 316897-001
	HP	1533	Intel P/N 340744-001

If you make any system changes, first consult the *Paragon™ XP/S Diagnostics Reference Manual* and the *Paragon™ OSF/1 User's Guide.*

Compatible Software

The results of booting the Paragon system OS with different combinations of scan driver, Paragon system OS, and diagnostics software are shown in Table 2-2. A successful boot or test is indicated with a 'Y' (minimal testing was done) and an unsuccessful boot or test is indicated with a 'N'.

Table 2-2. Paragon™ System Software Compatibility (1 of 3)

OS	Diagnostics	Scan Driver	OS Boot Method	OS Boot Results	PSD Test Results
R1.1	R1.1	0.6	async	Y	Y
			fscan	Y	
			scanio	Y	
		0.8	async	N	Y
			fscan	N	
			scanio	N	
R1.1	DIAG1.2*	0.6	async	Y	Y
			fscan	Y	
			scanio	Y	
		0.8	async	N	Y
			fscan	N	
			scanio	N	
R1.1.3	R1.1	0.6	async	Y	Y
			fscan	Y	
			scanio	Y	
		0.8	async	Y	Y
			fscan	Y	
			scanio	Y	

Table 2-2. Paragon™ System Software Compatibility (2 of 3)

OS	Diagnostics	Scan Driver	OS Boot Method	OS Boot Results	PSD Test Results
R1.1.3	DIAG1.2*	0.6	async	Y	Y
			fscan	Y	
			scanio	Y	
		0.8	async	Y	Y
			fscan	Y	
			scanio	Y	
R1.1.4	R1.1	0.6	async	Y	Y
			fscan	Y	
			scanio	Y	
		0.8	async	Y	Y
			fscan	Y	
			scanio	Y	
R1.1.4	DIAG1.2*	0.6	async	Y	Y
			fscan	Y	
			scanio	Y	
		0.8	async	Y	Y
			fscan	Y	
			scanio	Y	
R1.2	R1.1	0.6	async	Y	Y
			fscan	N	
			scanio	Y	
		0.8	async	Y	Y
			fscan	Y	
			scanio	Y	

Table 2-2. Paragon™ System Software Compatibility (3 of 3)

OS	Diagnostics	Scan Driver	OS Boot Method	OS Boot Results	PSD Test Results
R1.2	DIAG1.2*	0.6	async	Y	Y
			fscan	N	
			scanio	Y	
		0.8	async	Y	Y
			fscan	Y	
			scanio	Y	
R1.2	DIAG1.2.2	1.0	async	Y	Y
			fscan	N	
			scanio	Y	

- "DIAG1.2" includes the subsequent updates DIAG1.2.1.
- "R1.2" includes R1.2 of the operating system, and all subsequent updates.
- The 0.6 scan driver was released with the R1.1 Diagnostics.
- The 0.8 scan driver was released with DIAG1.2.
- The combination of R1.1 Paragon System OS and the 0.8 version of the scan driver should not be used. This is the reason why patch R1.1.3 had a modified reset script.
- All test results are for V3.x of GP node firmware.
- **fscan** and the scan driver should be compatible. For example, R1.1 **fscan** is built with the 0.6 scan driver, and R1.2 **fscan** is built with the 0.8 scan driver, which has large-system improvements in it.

FRU Identification

GP Node Identification

The codes in Table 2-3 identify the FRU (Field Replaceable Unit) numbers for the different GP Node boards that might be in a system.

Table 2-3. GP Node FRU Identification

FRU Number	Description
AIxx	All Pre-1.2-compatible GP Nodes (except 32 MB Fab 8 boards)—MCP *OFF*
AJxx	Pre-1.2-compatible 32 MB Fab 8 GP Nodes—MCP *OFF*
AKxx	1.2-compatible Fab 7 GP Nodes—MCP *ON*
ALxx	Not used
AMxx	1.2-compatible Fab 8 (16 MB) GP Nodes—MCP *ON*
ANxx	1.2-compatible Fab 8 (32 MB) GP Nodes—MCP *ON*

The codes are shown in the *SYSCONFIG.TXT* file, as in the following example line. The "AK" entry in this example identifies a 1.2-compatible Fab 7 unit with the Message Coprocessor (MCP) turned on:

```
S 0 GPNODE AK00 16 MIO B02
```

Refer to Appendix D of the *Diagnostics Reference Manual* for more information.

MIO Daughtercard Identification

The FRU identification for MIO boards in *SYSCONFIG.TXT* is a placeholder and does not contain type or revision information.

HIPPI Daughtercard Identification

The FRU identification for HIPPI boards in *SYSCONFIG.TXT* is a placeholder and does not contain type or revision information.

Power Controller Identification

The following versions of Power Controllers are used—all of which are compatible with the current release of Diagnostics:

```
PC AU00
PC AU01
PC AU02
```

LED Controller Identification

The only version of the LED Controller is identified as follows:

```
LED AM00
```

Backplane Identification

A variety of backplane versions are used—all of which are compatible. The following is an example:

```
BP A AC00
```

Installation Instructions 3

This chapter describes the steps necessary to install the Paragon™ System Diagnostic Software.

NOTE

To install the Paragon System Diagnostic Software, you must have completed the installation of the SCO OPEN DESKTOP Release 3.0.0. (This is the same release used with the previous version of Diagnostic Software.) If the operating system is not in place, follow the procedure shown in Appendix A to install it before installing the diagnostic software.

To check the version of the operating system on the diagnostic station, type the following command at the OS prompt:

```
uname -X
```

If it does not report "Release = 3.2v4.2", you must install a new operating system.

The procedures in this chapter use the conventions described in the Preface. You should also be aware of the following conventions:

- The instruction "Enter ***character(s)***" means type the indicated character(s), and then press the **<Enter>** key. For example, "Enter *y*" means type the letter "y", and then press the **<Enter>** key.
- In prompts, square brackets surround a default value. Pressing **<Enter>** selects the indicated default value.
- Some steps in these procedures cause a great deal of information to be displayed. However, the step as described here may show only the last message displayed. Also, do not be concerned if the indicated message does not appear immediately. Some steps take several minutes to complete.

Installing the Paragon™ Diagnostic Software

Installation Time: Approximately 10 minutes.

Installation Media: One cartridge tape labeled "Paragon™ Diagnostic Software Release 1.2.2" (313080-003).

OR, the file named *diag.tar* from the DIAG1.2.3 release.

Information you need:
root password.
IP address of the Paragon System boot node.
IP address of the diagnostic station.
The total number of cabinets in the Paragon system.

Requirements for Installation

You will need certain data on hand for use during the installation. Use this form to gather and record the required data.

Data Needed	Enter data in this column
Total number of Paragon system cabinets	
The *root* password for the diagnostic station	*Protect system passwords in a secure place.*
The IP Address of the Paragon System Boot Node	
The IP Address of the diagnostic station	

Installing the Diagnostic Software

1. Shut down the operating system on the Paragon system with the following steps:

A. On the Paragon System, change to the root directory:

```
cd /
```

B. Sync the memory:

```
sync;sync
```

C. Close down the operating system:

```
shutdown now
```

D. Unmount all file systems:

```
umount -A
```

E. Stop the processor:

```
halt
```

F. Return to the diagnostic station prompt:

```
~~.
```

2. Verify that the correct version of the SCO Open Desktop® operating system is installed on the diagnostic station:

   A. Login as *root* on the diagnostic station.

   B. Issue the following command to find out what version of the operating system is installed.

   ```
   DS#uname -X
   ```

   Eleven lines of information will be printed on the display. The *Release...* line should read:

   ```
   Release = 3.2v4.2
   ```

   If it does not, you must install a new version of the operating system onto the diagnostic station, using the procedure in Appendix A, before continuing with this procedure.

3. Change to the root directory:

```
DS#cd /
```

4. Change the **umask** for directory creation:

```
DS#umask 022
```

5. If a diagnostic daemon is running, stop it with the following command:

```
DS#dsdc stop
```

NOTE

Ignore any of the following error messages: `dsdc: Command not found` or `DSD shutdown: DSD is not running` or `DSD shutdown: [DSD shutdown complete]` and continue with the installation.

The daemon will either be restarted automatically when the diagnostic station is rebooted, or restarted manually at the end of this procedure.

6. Insert the Paragon System Diagnostic Software Release 1.2.2 tape in the tape drive or place the DIAG1.2.3 *diag.tar* file into the */u/tmp* directory.

NOTE

You may need to create the */u/tmp* directory.

7. For DIAG1.2.2, extract the files from the tape:

   (This step takes a few minutes.)

```
DS#tar xvpf /dev/rct0
```

   For DIAG1.2.3, extract the files from **tar** file:

```
DS#tar xvpf /u/tmp/diag.tar
```

8. If this is the first installation of DIAG1.2.2 or DIAG1.2.3, go to step 9. If you are unsure, check to see whether the Diaboard driver is version 1.0, with the following command:

   ```
   DS#strings /unix | grep Dia
   ```

   If the version is 1.0, go to Step 14. Otherwise, continue to Step 9.

9. The scan utilities directory has now been created. Change to that directory:

   ```
   DS#cd /etc/conf/pack.d/scan
   ```

10. Install the Driver:

    ```
    DS#./buildscan
    ```

    If the OS has previously been installed, you may be prompted about whether you want to rebuild the kernel. Answer *yes* (*y*).

    The system now builds */unix*.

    (This step takes a few minutes.)

Note

The following messages are normal; ignore them:

```
device driver for scan does not exist configuring
scan driver into kernel

/dev/scan does not exist, building into kernel
```

11. When asked if you want this kernel to boot by default, enter ***y*** (for yes).

12. When asked if you want the kernel environment to be rebuilt, enter ***y*** (for yes).

13. Shutdown the diagnostics station:

    ```
    DS#shutdown -y -g0
    ```

14. When prompted to reboot, press **<Enter>**.

15. Login as *root* on the diagnostics station.

16. Do one of the following:

    - Check that *DIAG_ALIAS* and *PARA_ALIAS* are defined in the */etc/hosts/* file. The alias variables should be included on the lines that contain the Paragon System and Diagnostic Station IP numbers. (This is the recommended way to define system IP addresses.)

      ```
      xxx.xx.xx.xx DS_name DIAG_ALIAS DS_name.def.com
      xxx.xx.xx.xx Paragon_name PARA_ALIAS
      ```

    - Modify the */u/paragon/diag/psdenv* file to include the IP definition lines as follows. (This is the old way of defining system IP addresses for PSD.)

      ```
      OUR_IP_ADDR=Paragon Boot Node IP Address
      DS_IP_ADDR=Diagnostic Station IP Address
      ```

17. Change directory to */usr/paragon/boot*:

    ```
    DS#cd /usr/paragon/boot
    ```

    Find out if *DEVCONF.TXT* and *MAGIC.MASTER* files exist. If they are not found in */usr/paragon/boot*, then do the next step. If the files are present, skip the next step.

18. Do one of the following:

    - Restore the *DEVCONF.TXT* and *MAGIC.MASTER* files now if you saved them prior to installation of SCO ODT 3.0.0.

    - Create *DEVCONF.TXT* and *MAGIC.MASTER* files. You can alter the samples found in */u/paragon/diag/sample*. Refer to the *Paragon System Diagnostics Reference Manual* for a detailed description of these files.

19. Change directory to */u/paragon/diag*:

    ```
    DS#cd /u/paragon/diag
    ```

20. Run the **hwcfg** utility to generate an intermediate hardware configuration file (see manual page for **hwcfg**). It will generate intermediate file */usr/paragon/boot/HWCONFIG.TXT*.

    ```
    DS#hwcfg
    ```

    If PSD was installed before, it will prompt you to ask whether you want to overwrite *HWCONFIG.TXT*. Answer *yes* (*y*).

Note

The message `Check cable: Warning Cable E (power control) not present` is normal; ignore it.

If the message `hwcfg: The number of cabinets must be specified` is reported, use the **-c** switch with **hwcfg** to specify the number of cabinets in your system.

21. Check that the JTAG scan bus is functional before determining the system configuration:

```
DS#scantest -cnumber_of_cabinets -i
```

22. Run the configuration merge utility, **mergecfg**, to generate *SYSCONFIG.TXT* (see manual page for **mergecfg**). It will generate */usr/paragon/boot/SYSCONFIG.TXT* file.

```
DS#mergecfg
```

If *SYSCONFIG.TXT* already exists, it will prompt you to ask if you want to overwrite the file. Answer *yes* (**y**).

23. Run the configuration parser, **cfgpar**, to generate *SYSCONFIG.BIN* (see manual page for **cfgpar**). It will generate the binary file */u/paragon/diag/SYSCONFIG.BIN*.

```
DS#cfgpar
```

24. Use **flashutil** to update the Paragon System Flash EPROM contents in your system. See Chapter 4 of these release notes on how to update the Flash EPROMs.

25. If you did not do Steps 11 through 15 to build a new scan driver and did not reboot the diagnostic station, restart the diagnostic daemon manually:

```
DS#dsdc start
```

NOTE

The message `DSD started` is normal.

26. To enter the diagnostic menu, enter:

```
DS#psd
```

Updating Paragon™ System Firmware 4

Installation Time:	Approximately 1 minute.
Installation Media:	The update is part of the diagnostic software.
Information you need:	*root* password.

The chapter describes how to use **flashutil** to update the firmware in a Paragon system.

Caution

This procedure updates all nodes at the same time. There is a very small risk in this method: if a power glitch occurs during the approximately 25 seconds required for updating, it is possible that the contents of every EPROM could be corrupted.

The alternative is to update one node at a time, or a small range of nodes. A power glitch would then disturb the EPROM contents in only a single node or a small set of nodes. However, a 512-node machine, for example, would require several hours to update that way.

If a power glitch occurs while updating the specified node, you may not be able to recover this node. Recovering from a power glitch may require an external EPROM programmer to reprogram a flash EPROM.

Note

You must install the Paragon XP/S system diagnostic software before you update any firmware.

If your Paragon System has GP node firmware below version V3.1, you need to update those nodes to V3.1 prior to updating to V3.3. Refer to the *Release Notes* for DIAG1.2 for instructions.

If you receive `Response timeout: node...` errors, when using **flashutil**, check that the small power connecters (1" x 1", with three wires) in the lower-right corner of the backplanes are seated properly.

1. There are three methods for updating the Paragon System firmware. Choose one of the following methods:

   - Update one node at a time:

     ```
     DS#flashutil -s node
     ```

     This is the safest method for protecting against power glitches.

   - Update a range of nodes:

     ```
     DS#flashutil -s first_node..last_node
     ```

     You may use the node-range option to do a section of your system at a time. This method localizes the risk to a group of nodes. Updating a cabinet of nodes is possible with this method.

   - Update your entire system:

     ```
     DS#flashutil
     ```

     This choice carries the greatest risk, but provides the quickest update. All nodes are updated in parallel.

2. Choose the target Flash from the menu that **flashutil** displays:

```
Please select the Flash memory for the update

     1 ---> Program the GP          Flash memory
     2 ---> Program the MIO         Flash memory
     3 ---> Program the HIPPI       Flash memory
     4 ---> Program the MDC         Flash memory
     8 ---> Program the MP          Flash memory
     9 ---> Program the MP Flex     Flash memory
    28 ---> ROM version report
    30 ---> Exit flashutil no Flash programming
```

To update the GP (for example), enter ***1***

NOTE

The HIPPI selection works on 256 Kbyte firmware. It will not program older 128 Kbyte HIPPI devices (Fab 2).

3. The **flashutil** program returns a message asking if you want to reset the Paragon system.

```
This program will reset the Paragon system. Do you wish to
continue? (y/n)
```

To cancel at this point, enter either a carriage return or ***n*** (for no).

To update, enter ***y*** (for yes).

4. The program initializes the system, loads the nodes with the code to reprogram the EPROMs, along with the *fw_all.bin* file, which contains the new firmware for all flash EPROMs, then displays a warning message. You now have one last chance to abandon the update:

```
Warning! current flash EPROM contents will be erased and
replaced.
Proceed? (yes/no)
```

Enter "no" to abandon the update, or enter "yes" to update.

Any response other than ***yes*** (fully spelled out) cancels the update.

flashutil then sends a command to each node in sequence, causing the node to program the flash EPROM image that now resides in RAM into the selected flash EPROM. **flashutil** displays a "+" for each node on which the target EPROM is programmed, and a "-" for each node on which the target EPROM is not programmed correctly. For example, if there are five nodes in a system, with the third one including an MIO daughtercard, **flashutil** displays the following series as it goes through the nodes to reprogram MIO flash EPROMs:

```
--+--
```

If no error message follows the "+" sign, the node programmed correctly. A "-" sign indicates that the selected target was not found on that node—it does not indicate an error or an empty slot.

NOTE

A system that contains GP nodes with a mix of old and new firmware (for example when a board is placed in a system that has previously been updated) will need to be operated the same as if all nodes in the system contain the old firmware.

5. If you do enter ***yes***, the update proceeds. Each node returns a status message to **flashutil** (via the scan bus) when it completes the update.

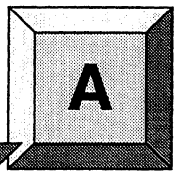
6. Confirm that all target EPROMs now contain the correct updated firmware. Use the **flashutil** utility with the **-r** and **-t** switches to display the version number that it finds on the node boards:

```
DS#flashutil -r -t gp
```

flashutil will display a report showing the version numbers of the GP node flash EPROMs in your system:

```
GP.FLASH -              (expected count=4, actual count=4)
  Version V3.3 found on the following nodes:
   00A00 00A01 00A02 00A03
```

Installing the SCO Operating System



This appendix describes the steps necessary to install SCO Open Desktop Release 3.0.0.

The procedures in this appendix use the conventions described in the Preface. You should also be aware of the following conventions:

- The instruction "Enter ***character(s)***" means type the indicated character(s), and then press the **<Enter>** key. For example, "Enter *y*" means type the letter "y", and then press the **<Enter>** key.
- In prompts, square brackets surround a default value. Pressing **<Enter>** selects the indicated default value.
- Some steps in these procedures cause a great deal of information to be displayed. However, the step as described here may show only the last message displayed. Also, do not be concerned if the indicated message does not appear immediately. Some steps take several minutes to complete.

Installing SCO OPEN DESKTOP Release 3.0.0

Installation Time: Approximately 45 minutes.

Installation Media: One cartridge tape labeled "SCO OPEN DESKTOP R3.0.0 for the Paragon™ Diagnostic Workstation SCO Mass Install Tape Vol 1 of 1" (312978-001).

One disk labeled "SCO OPEN DESKTOP R3.0.0 for the Paragon™ Diagnostic Workstation N1 Boot Disk" (312974-001).

One disk labeled "SCO OPEN DESKTOP R3.0.0 for the Paragon™ Diagnostic Workstation N2 File System Disk" (312975-001).

One disk labeled "SCO OPEN DESKTOP R3.0.0 for the Paragon™ Diagnostic Workstation M01 Master Install Disk" (312976-001).

Requirements for Installation

You will need certain data on hand for use during the installation. Use this form to gather and record the required data.

Data Needed	Enter data in this column
The SCO Serial Number (located in the SCO OPEN DESKTOP box)	
The SCO Activation Key (located in the SCO OPEN DESKTOP box)	
The system name of the diagnostic station	
The root password of the diagnostic station	*Protect system passwords in a secure place.*
The IP address of the diagnostic station	

Data Needed	Enter data in this column
The domain name of the diagnostic station (use the **hostname** command to find it)	
The Netmask of the diagnostic station	
The Broadcast IP address of the diagnostic station	
The IP address of the Paragon System Boot Node	
The total number of Paragon system cabinets	

It is essential to make backup copies of:

- Diagnostic station-specific files */etc/hosts* and */etc/resolv.conf* (if they exist)
- Paragon System diagnostic configuration files */usr/paragon/boot/DEVCONF.TXT*, */usr/paragon/boot/MAGIC.MASTER*, and */usr/paragon/BOOTMAGIC.md* files (if they exist)
- Paragon OSF/1 files which reside on the diagnostic station in the directory trees */usr/local/bin* and */usr/paragon/boo*

If you haven't done so already, shut down the operating system on the Paragon System with the following steps:

1. On the Paragon System, change to the root directory:

```
cd /
```

2. Sync the memory:

```
sync;sync
```

3. Close down the operating system:

```
shutdown now
```

4. Unmount all file systems:

```
umount -A
```

5. Stop the processor:

```
halt
```

6. Return to the diagnostic station prompt:

```
~~.
```

Reinstalling SCO OPEN DESKTOP

If you are reinstalling SCO OPEN DESKTOP over an existing system, use a utility, such as **fdisk**, to delete the active UNIX partition on the diagnostic station.

1. To find the active partition (see the manual page for fdisk to interpret the returned information), enter:

```
fdisk -p
```

2. Delete the active partition. For example, if partition 1 is active, enter:

```
fdisk -d 1
```

Install SCO OPEN DESKTOP Procedure

WARNING

These procedures overwrite the Paragon System diagnostic station disk drive. Make a backup of any user file(s) you want to retain.

1. Insert the SCO N1 Boot disk into the disk drive.
2. Boot the diagnostic station by turning the power on.
3. At the boot prompt, press **<Enter>.**
4. When prompted, insert the SCO N2 File System disk and press **<Enter>**.

Note

Ignore the normal message `warning: /dev/ropipe was not in mount table.`

5. When prompted to select the type of tape drive, enter the following:

   scsi

Note

The prompt in the next step refers to the MIT System Image Vol. I tape. Our corresponding product is called the *"SCO Mass Installation Toolkit Tape Vol. I"* and is used in place of the MIT tape.

6. When prompted:

   A. Verify that the SCO M01 Master Install diskette is in the floppy drive.

   B. Verify that the SCO Mass Installation Toolkit Tape Vol. 1 is in the tape drive.

   C. Press **<Enter>**.

   (This step takes about 30 minutes.)

Note

Ignore the message `errno 26, Text file busy....`

7. When prompted to set system time, enter ***y*** (for yes).

   If you are not in North America, enter ***n*** (for no) in response to step 8 and go to step 11.

8. When asked if you are in North America, enter ***y*** (for yes) or enter ***n*** (for no).

9. When asked for your time zone, enter your time zone number and press **<Enter>**.

10. When asked if daylight savings applies to your time zone, enter either ***y*** (for yes) or ***n*** (for no).

11. Enter the correct date and time using the format of year, month, day, hour and minute. This example is for a date and time of March 9, 1994 at 6:22 p.m.:

    9403091822

12. When asked if you want to set the system name, enter ***y***.

13. Enter your diagnostic station name and press **<Enter>**.

14. When asked if the mail system should be a different name, enter ***n***.
15. When prompted, press **<Enter>** to continue.
16. When prompted to serialize the system, respond with ***y***.

Note

If you respond "Yes" to the question in step 17, you will be forced to start this procedure over at step 1.

17. When asked if you want to execute floppy-based serialization, respond with ***n***.
18. Enter Serial Number and Activation Key codes at the prompts.

    (This step takes about 20 seconds.)
19. When asked if you want to change your answer to any of these questions, respond with ***q***.

    The system now builds */unix*. (This step takes a few minutes.)
20. When prompted to reboot the system, remove any remaining floppy disk(s) and/or tape(s) and press **<Enter>** to reboot.

Note

In the next step you have only 5 seconds to press **<Enter>** after the boot prompt appears.

21. When the boot prompt appears, enter single-user mode by pressing **<Enter>** within 5 seconds.
22. Wait for the single-user mode login prompt, then enter the password:

    paragon3
23. Run the password utility:

    passwd
24. When prompted to choose your own password, respond with **1.**
25. When prompted, enter your new password.
26. When reprompted, reenter your new password.

NOTE

Do not restore the password file from a backup. Doing so will compromise the system security and may cause boot problems on the diagnostic station. Use the **passwd** or **sysadmsh** utilities to change the diagnostic station password.

27. Edit the file */etc/default/tcp* by changing the lines in the *tcp* file as shown in Table A-1.

Table A-1. Edit Values in the */etc/default/tcp* File

Current	Change To:
DOMAIN = default.com	DOMAIN = *DS system's Domain name*
IPADDR = nnn.nnn.nnn.nnn	IPADDR = *DS system's IP address*
NETMASK = nnn.nnn.nnn.nnn	NETMASK = *netmask*
BROADCAST = nnn.nnn.nnn.nnn	BROADCAST = *broadcast IP address*

28. Restore your */etc/hosts* file from your backup copy, if one was created, or modify the existing */etc/hosts* file.

Note

When you restore the */etc/hosts* file, you must also alias the DS domain name to the DS IP number. Use the **hostname** command to find the domain name.

29. Reboot the diagnostic station:

```
reboot
```

This completes the installation of the basic SCO OPEN DESKTOP Release 3.0.0 software on the diagnostic station.

MRC and NIC Register Definitions B

This appendix provides detailed information about the contents of the registers in both MRC and NIC devices. The information may be used to elaborate on diagnostic error messages that contain raw register information.

MRC Register Definitions and Organization

The MRC status and control register is a 64-bit read-only register that is used to read the internal state and configuration of the MRC, and to control MRC operation (Figure B-1). A description of the function of each bit is shown below. When a register bit is set to one, the condition is true. In the following description,

- Bit 0 is the first bit read.
- The first 18 bits are the status bits.
- The next 26 locations are reserved. These can not be written to and are always "0".
- The final 20 locations are the control bits.

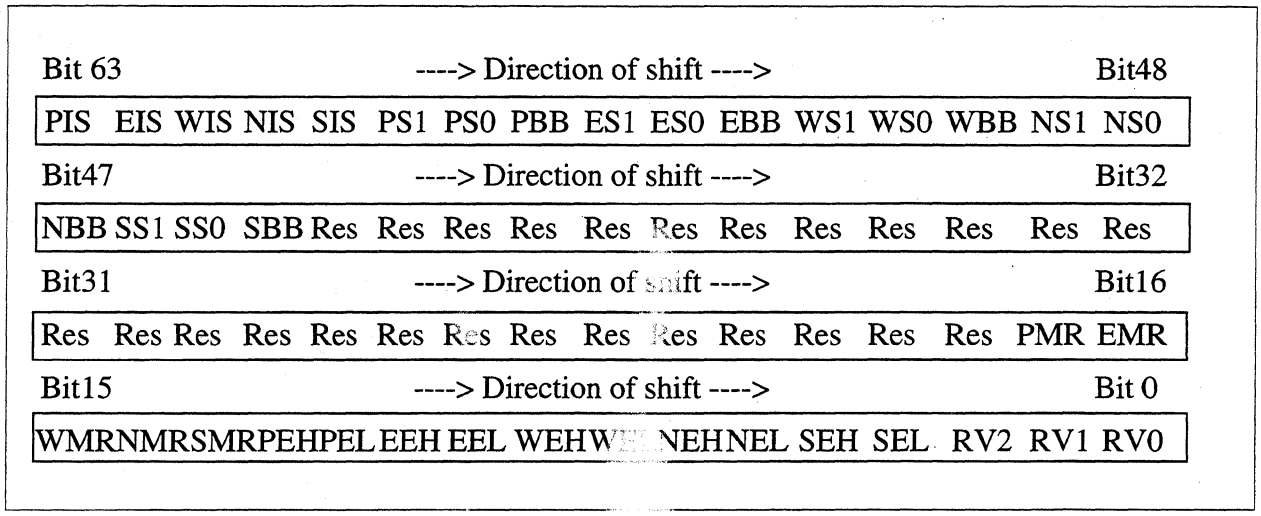


Figure 2-1. MRC Register Bits

Control registers

PIS	Input Streaming Processor port
EIS	Input Streaming East port
WIS	Input Streaming West port
NIS	Input Streaming North port
SIS	Input Streaming South port
PS1	Upper bit Streaming Processor port
PS0	Lower bit Streaming Processor port
PBB	Misroute Enable Processor port
ES1	Upper bit Streaming East port
ES0	Lower bit Streaming East port
EBB	Misroute Enable East port
WS1	Upper bit Streaming West port
WS0	Lower bit Streaming West port
WBB	Misroute Enable West port
NS1	Upper bit Streaming North port
NS0	Lower bit Streaming North port
NBB	Misroute Enable North port
SS1	Upper bit Streaming South port
SS0	Lower bit Streaming South port
SBB	Misroute Enable South port

Status Registers

PMR	Misrouted message detected on processor port
EMR	Misrouted message detected on east port
WMR	Misrouted message detected on west port
NMR	Misrouted message detected on north port
SMR	Misrouted message detected on south port
PEH	Upper byte plus tail parity error on processor port
PEL	Lower byte parity error on processor port
EEH	Upper byte plus tail parity error on east port
EEL	Lower byte parity error on east port
WEH	Upper byte plus tail parity error on west port
WEL	Lower byte parity error on west port
NEH	Upper byte plus tail parity error on north port
NEL	Lower byte parity error on north port
SEH	Upper byte plus tail parity error on south port
SEL	Lower byte parity error on south port
RV2	Revision level most significant bit
RV1	Revision level bit 1
RV0	Revision level least significant bit

NIC Status Register Definitions

The NIC status register (status_reg) is a 64-bit read-only register that is used to read the internal state and configuration of the NIC. A description of the function of each bit is shown below. When a status register bit is set to one, the condition is true except where notes (bits 0, 1, 6, 9). Bits 43 to 63 always read zero.

FIFO Flag Status

Table B-1. FIFO Flag Status

Status Register Bit	Function
status_reg(0)	*transmit FIFO not full* flag - When asserted, (i.e. "zero") indicates the transmit FIFO is full.
status_reg(1)	*transmit FIFO not almost full* flag - Asserted (i.e. "zero") if the transmit FIFO is at or within 32 64-bit words from being full.
status_reg(2)	*transmit FIFO almost empty* flag - When asserted, indicates the transmit FIFO is almost empty.
status_reg(3)	*transmit FIFO empty* flag - When asserted, indicates the transmit FIFO is empty.
status_reg(4)	*receive FIFO full* flag - When asserted, indicates the receive FIFO is full.
status_reg(5)	*receive FIFO almost full* flag - Asserted if the receive FIFO is almost full.
status_reg(6)	*receive FIFO not almost empty* flag - Asserted (i.e. "zero") if the receive FIFO is at or within 32 64-bit words from being empty.
status_reg(7)	*can read two words* - When asserted, indicates the receive channel contains at least two 64-bit words.
status_reg(8)	*can read one word* - When asserted, indicates the receive channel contains as least one 64-bit word.
status-reg(9)	*receive FIFO not empty* - When asserted, (i.e. "zero") indicates the receive FIFO is empty. When deasserted (i.e. "one") indicates receive FIFO is holding data.

Processor Port Status

Table B-2. Processor Port Status

Status Register Bit	Function
status_reg(10)	*eod in NIC* - When asserted, indicates that the NIC receive FIFO contains at least one complete packet.
status_reg(11)	*receive channel ready* - When asserted, indicates that the NIC has data that can be read from the processor port.
status_reg(12)	*transmit channel ready* - When asserted, indicates that the NIC is able to accept input data from the processor port.
status_reg(13)	*eod last word* - This bit is set when the most recent word transferred to the processor interface carried with it the *end of data* bit (eod). Software uses this bit to make sure the "eod marked" word was the last word in the packet. This bit stays set until the next non-eod word is delivered to the processor interface.

Error Status Bits

Table B-3. Error Status Bits

Status Register Bit	Function
status_reg(14)	*network crc0 error* - When asserted, indicates that a CRC error has occurred in the lower 32 bits of an incoming packet.
status_reg (15)	*network crc1 error* - When asserted, indicates that a CRC error has occurred in the upper 32 bits of an incoming packet.
status_reg (16)	*processor port parity0 error* - When asserted, indicates that a parity error has occurred in the lowest byte of an incoming word from the processor port.
status_reg(17)	*processor port parity1 error* - See description above.
status_reg(18)	*processor port parity2 error* - See description above.
status_reg (19)	*processor port parity3 error* - See description above.
status_reg(20)	*processor port parity4 error* - See description above.
status_reg(21)	*processor port parity5 error* - See description above.
status_reg(22)	*processor port parity6 error* - See description above.
status_reg(23)	*processor port parity7 error* - See description above.
status_reg(24)	*network parity0 error* - When asserted, indicates that a parity error has occurred in the lower byte of an incoming packet from the network.
status_reg(25)	*network parity1 error* - When asserted, indicates that a parity error has occurred in the upper byte of an incoming packet from the network. This parity bit also covers the tail bit with the upper byte.
status_reg(26)	*transmit FIFO overrun* - When asserted, indicates that an attempt to write the transmit FIFO was made when it was full.
status_reg(27)	*receive FIFO overrun* - When asserted, indicates that an attempt was made to write the receive FIFO when it was full.
status _reg(28)	*receive underrun* - When asserted indicates that an attempt was made to read the receive channel when it was empty.

Miscellaneous Status Bits

Table B-4. Miscellaneous Status Bits

Status Register Bit	Function
status_reg(29)	*xreq - state of transmit request* pin (xreq)
status_reg(30)	*xack - state of transmit acknowledge* pin (xack)
status_reg(31)	*rreq - state of receive request* pin (rreq)
status_reg(32)	*rack - state of receive acknowledge* pin (rack)
status_reg(33)	*xmip* - If set, indicates a transmit message is in progress. In other words, a header of a message has gone out, but the tail has not. This bit will stay set as long as the "tail" of the message is inside the NIC.
status_reg(34)	*rmip* - If set, indicates a receive message is in progress. Function is similar to xmip. This bit will stay set from the arrival of the first 16-bit portion of a message header until the tail of the same message arrives.

NIC Identification Bits

These bits identify the revision level and type of the NIC.

Table B-5. NIC Identification Bits

Status Register Bit	Function
status_reg(37: 35)	NIC revision code. Reserved for internal use.
status_reg.(42: 38)	NIC type code. Reserved for internal use.

New Manual Pages C

This appendix contains new manual pages for Diagnostic utilities:

cnv	The numbering scheme conversion utility.
flashutil	A new manual page for an enhanced version of the Flash EPROM updating utility.
gencfg	A new manual page for the modified configuration generating utility.
romver	A new manual page for the modified ROM-version utility.
scantest	The new scan hardware test utility.
showconf	The newly enhanced utility to display the system configuration. (DIAG1.2.3 only)

CNV CNV

Paragon™ System slot numbering conversion utility.

Syntax

cnv [**-n** *cabinets*] {**-c** *CBS_number* | **-r** *OS_number* | **-d** *DIAG_number* | **-m** *mesh_number*} [**-v**] [**-F** *binfile*]

Arguments

-n *cabinets* Specifies the total number of cabinets installed in the system. If the binary configuration file *SYSCONFIG.BIN* exists, the default is the number of cabinets specified in that file. If the configuration file does not exist, the default is 1.

-c *CBS_number* The Cabinet-Backplane-Slot number of a node. For example, ***0C7*** specifies slot 7 in backplane C of cabinet 0.

-r *OS_number* The operating system node number (also known as the "logical number" or "root partition node number") of a node.

-d *DIAG_number*
The Diagnostic number of a node.

-m *mesh_number*
The mesh number of a node's MRC in the system. For example, ***3,12*** specifies node 12 in row 3.

-v Displays the position of the specified node on a graphical map of one cabinet in the system.

[**-F** *binfile*] Specifies the binary configuration file if it is in a location other than the default */u/paragon/diag/SYSCONFIG.BIN*. If the **-n** switch is used, the **-F** switch is ignored.

Description

This utility converts a node number in one numbering scheme to the other numbering schemes. With the **-v** command line switch, it also displays a map showing the location of the specified node

CNV *(cont.)*

CNV *(cont.)*

Example

To convert operating system (OS) node number 12 to the other numbering schemes in a one-cabinet system, type:

```
DS# cnv -n 1 -r 12
CBS  OS DIAG MESH
0D15 12  63  3,15
```

To show the physical location of the node with Diagnostic number 24 in a two-cabinet system, type:

```
DS# cnv -c 2 -d 24 -v
CBS  OS DIAG MESH
0B8  93  24  1,8

Cabinet 0 of 2
----------
.  .  .  .
.  .  .  .
.  .  .  .
.  .  .  .
----------
.  .  .  .
.  .  .  .
.  .  .  .
.  .  .  .
----------
.  .  .  .
.  .  .  .
.  .  .  .
.  *  .  .
----------
.  .  .  .
.  .  .  .
.  .  .  .
.  .  .  .
----------
```

FLASHUTIL FLASHUTIL

Allows reprogramming of the flash EPROMs on Paragon™ system node boards and daughtercards.

Syntax

flashutil [**-a**] [**-b** *binFile*] [**-c** *cfgFile*] [**-defhino**] [**-p** [*target*]] [**-q** *version*] [**-r**] [**-s** *x*[*..y*]] [**-t** *select*] [**-v**] [**-x**]

Arguments

-a Reprogram nodes one at a time (serially). The default is to program all nodes at the same time.

-b *binFile* Specify the binary firmware image pathname. Defaults to */u/paragon/diag/fw_all.bin*.

-c *cfgFile* Specifies the pathname of the binary system configuration file. Defaults to */u/paragon/diag/SYSCONFIG.BIN*.

-d Debug mode. This switch causes the program to print extra trace information.

-e Excludes the version selected by the **-q** switch from the **-r** flash version report.

-f When this program runs normally, warnings and confirmations are issued to prevent unintentionally resetting the Paragon System and to confirm the program operation. This switch causes the program to bypass the confirmation requests.

-h Displays usage information.

-i Specifies that the nodes are not to be initialized. Assumes the nodes are already in the state to accept commands to update the firmware.

-n Specifies that a response won't be expected from the NIC boot loader.

-o Specifies the update is to be performed on a system containing one or more GP nodes with "old" firmware (317053-004).

-p [*target*] Programs the target flash EPROMs on the specified nodes. The *target* may be any of the following: *gp*, *mio*, *hippi*, *mdc*, *mp*, or *mpflex*. If you do not specify a target, the **flashutil** prompts for one. If the specified target does not exist on a selected node, the request is ignored.

-**q** *version* A query filter for the **-r** flash version report. The **-q** switch only searches for the version specified and only for the target type of firmware specified by the **-t** switch. This switch must be used with the **-r** switch.

FLASHUTIL *(cont.)*

FLASHUTIL *(cont.)*

-r	The flash version report switch specifies that no flashes are to be programmed. The optional target type, as specified by the **-t** switch, indicates which type of flash EPROM to report version information on. If no **-t** switch is used, version information is reported for all types of flash memories in the system. Version reports can be filtered using the **-e** and **-q** switches.
-s *x*[..*y*]	Specifies a single node or range of nodes on which to perform the firmware update. The default is all nodes in the configuration file (*/u/paragon/diag/SYSCONFIG.BIN,* if not specified).
-t *select*	Selects which target types to gather version information on. The *select* parameter can be *gp, mio, hippi, mdc, mp*, or *mpflex*. One or more target types may be specified with commas as delimiters (no spaces) between the target types. If **-t** is not specified, version information on all types of flashes in the system is reported. This switch must be used with the **-r** switch.
-v	Specifies using the verbose mode.
-x	Echoes the checksum of the specified flash and the checksum of the firmware image to be programmed.

Description

flashutil is a stand-alone program that reprograms or reports firmware version information about any flash memory on a node board or daughtercard in a Paragon system.

The system is initialized using **initutil**. Then **tftp** loads the boot node's RAM with the binary image of the firmware to be programmed, along with the executable code to actually perform the programming operation. The boot node broadcasts the codes to all other nodes over the mesh routing backplane and starts them executing. The program sends a command sequentially to each node causing it to erase and reprogram a flash EPROM or to return firmware version information. If **-a** is specified, then the update is performed serially; otherwise, all nodes are programmed in parallel.

When flashing the EPROM, the program sends a target-selection command sequentially to each selected node. An update command is then sent to each selected node in sequence. The program on each node checks for the specified target on that node, and on any expansion boards connected to it. If the targeted flash EPROM exists, the program erases and reprograms it. If the **-a** switch is used along with **-p**, the reprogramming is performed serially, one node at a time. Otherwise, the reprogramming is done in parallel on all selected nodes.

FLASHUTIL *(cont.)* FLASHUTIL *(cont.)*

Usage

flashutil begins with a menu to select which firmware to update:

```
Please Select the Flash option below

 1 ---> Program the GP Flash memory
 2 ---> Program the MIO Flash memory
 3 ---> Program the HIPPI Flash memory
 4 ---> Program the MDC Flash memory
 8 ---> Program the MP Flash memory
 9 ---> Program the MPFLEX Flash memory
28 ---> Flash version report
30 ---> Exit flashutil no Flash programming
```

If a program update is selected, **flashutil** reads and displays the binary image to determine the version the nodes are to be updated to. **flashutil** then reads the binary configuration.

If the **-f** or **-r** switches are not used, **flashutil** issues a confirmation request to prevent unintentionally resetting the attached Paragon system.

```
"This program will reset the attached Paragon system"
 "Please confirm with y/n (n): "
```

To cancel an update, enter either ***return*** or ***n***.

flashutil initializes the nodes and MRC's in the Paragon system based on the configuration found in the binary configuration file (the default is */u/paragon/diag/SYSCONFIG.BIN*). The initialization is done using ***initutil -l -p***, or ***initutil -w -o -p*** if the **-o** flag was used. If the **-o** flag is not used, a Level 1 mesh test is performed prior to loading the non-boot nodes. The Level 1 mesh test sequentially tests the mesh connections between the current node and each of its installed neighbors.

Rev. -004 GP node firmware (prior to GP node fab 7-011) requires the non-boot nodes to auto-boot using the NIC bootloader. The **-o** flag causes **initutil** to wait for the non-boot nodes to enter the NIC bootloader. This mode then relies on the successful broadcast of the update firmware over the mesh.

flashutil checks for the IP addresses of the boot node and the diagnostic station from the */etc/hosts* file on the diagnostics station. The DIAG_ALIAS and PARA_ALIAS tags need to exist before **flashutil** proceeds.

Once all nodes have completed their self tests (and optionally the Level 1 mesh test), the boot node loads three files from the diagnostic Station via ethernet (**tftp**):

FLASHUTIL *(cont.)*

FLASHUTIL *(cont.)*

loader Mesh loader.

fw_all.bin New firmware binary image.

flash.node Node-executable code for programming the EPROMs.

The mesh loader program broadcasts the binary image and the node executable files to the other nodes in the system. **flashutil** then causes the mesh loader to start the node executable on the other nodes and itself. **flashutil** then presents a confirmation request (if **-f** is not used):

```
Warning! current Flash EPROM contents will be erased and
replaced.
Proceed? (yes/no)
```

Enter ***yes*** (fully spelled out) to proceed. Anything else aborts the update.

flashutil then sends a command to each node in sequence, causing the node to program the flash EPROM image that now resides in RAM into the target flash EPROM. **flashutil** displays a "+" for each node that is programmed, and a "-" for each node that isn't programmed. For example, if there are five nodes in a system, with the third one including an MIO daughtercard, **flashutil** displays the following series as it goes through the nodes to reprogram MIO Flash EPROMs:

```
--+--
```

If no error message follows the "+" sign, the target flash was programmed correctly. A "-" sign indicates that the selected target was not found on that node—it does not indicate an error.

A system that contains GP nodes with a mix of old and new firmware (for example, when a board is placed in a system that has previously been updated) needs to be handled as if all nodes in the system contain the old firmware.

Use ***flashutil -r [-q version] [-t select] [-e]*** or ***romver*** to verify that all target flashes were updated. Doing this causes each node to return a checksum of the contents of the flash EPROM specified, and flashutil compares those checksums to ones kept in a database. All nodes that match firmware versions with checksums in the database are displayed under that version heading, while nodes that don't match any checksum are displayed under a "??" version heading. An example of this output is shown below:

FLASHUTIL *(cont.)*

FLASHUTIL *(cont.)*

```
   GP FLASH                                    (expected count = #, actual
count = #)
          Version # found on the following nodes:
          cbs-node-#   cbs-node-#   cbs-node-# ...
          cbs-node-#   cbs-node-#   cbs-node-# ...

  MIO FLASH                                   (expected count = #, actual
count = #)
          Version # found on the following nodes:
          cbs-node-#   cbs-node-#   cbs-node-# ...
          cbs-node-#   cbs-node-#   cbs-node-# ...
```

Note that the expected count is determined from the configuration file (*/u/paragon/diag/SYSCONFIG.BIN*) and the actual count is from the results sent to **flashutil** from the nodes. If these counts differ, *SYSCONFIG.BIN* or */usr/paragon/boot/DEVCONF.TXT* may not reflect the actual system configuration.

Examples

To update a single GP node:

```
flashutil -s node# -p gp
```

To update any target at one particular node:

```
flashutil -s node#
```

A menu prompts for a target flash to program.

To quickly update all MP node firmware:

```
flashutil -p mp
```

FLASHUTIL *(cont.)*

FLASHUTIL *(cont.)*

To report version information for all flash EPROMs in a system:

```
flashutil -r
```

To check for all MP nodes with V1.2 FLEX bits:

```
flashutil -r -t mpflex -q V1.2
```

Files

/u/paragon/diag/fw_all.bin	Binary EPROM image
/u/paragon/diag/loader	Mesh bootloader
/u/paragon/diag/flash.node	Node executable

See Also

cfgpar, **initutil**, **mrcutil**, **psd**, **romver**, **rstutil**

GENCFG GENCFG

gencfg scans the Paragon™ system hardware connected to the diagnostic station and produces hardware configuration information in human-readable format.

Syntax

gencfg -c *Cabinets* [**-f** *devName*]

Arguments

-f *devName* Specifies the name of the device to open for communication with the system via the diagnostic station. The default *devName* is */dev/scan*.

-c *Cabinets* Specifies the total number of cabinets installed in the system. This is a mandatory argument.

Description

gencfg scans the Paragon system hardware connected to the Diagnostic Station and produces hardware configuration information in human-readable format. It queries a specified number of Paragon system cabinets and provides information about Power Control, LED, Backplanes, Nodes and MRCs.

NOTE

Information about MIOs, TAPEs, DISKs, MDCs, and RAIDs are not generated automatically. These are specified in the hand-created DEVCONF.TXT file.

The output from **gencfg** can later on be used with the utility **mergecfg** to merge the device configuration file *DEVCONF.TXT* in order to produce a consolidated *SYSCONFIG.TXT* describing the Paragon system hardware in its entirety.

Example

To generate the configuration for a 5-cabinet system,

```
gencfg -c 5 > /usr/paragon/boot/hwcfg.txt
```

GENCFG *(cont.)*

GENCFG *(cont.)*

See Also

hwcfg, mergecfg (file formats of *SYSCONFIG.TXT* and *DEVCONF.TXT*.)

scantest (useful in scan-bus-related failures.)

ROMVER ROMVER

Display version information about firmware contained in flash EPROMs on Paragon™ system node boards and daughtercards.

Syntax

romver [**-dfhiv**] [**-n** *node*]

Arguments

-d	Debug mode. In debug mode, **romver** displays information about the sequence of utilities and internal routines used to perform the version check.
-f	Causes **romver** to "force" the system reset without asking for confirmation first. The is useful if the output from **romver** is directed to a file—otherwise the program will print the message asking for the confirmation to the file and you will not see it or respond to it.
-h	Displays this help information.
-i	Specifies that **romver** will not initialize the nodes.
-v	Puts **romver** into verbose mode, to provide more detailed information about the firmware in individual nodes.
-n	Indicates that **romver** should check the version only on the specified node.
-x *value*	Specifies the expected checksum value (hex). If this argument is not used, the expected value is taken from the first node.

Description

romver displays the firmware versions of GP, MIO, HIPPI, MDC, MP, and MPFLEX flash memories in a Paragon system. **romver** is actually a script that calls **flashutil**, so the actual work of requesting and checking flash information from nodes is done by **flashutil**. **flashutil** resets the system and then loads a program (*flash.node*; see the **flashutil** manual page) onto each node. It then sends a command to each node requesting checksums of all firmware on the node and attached daughtercards. When nodes send the information back, **flashutil** compares the checksums to checksums stored in a database it keeps track of. It then provides a report detailing what versions of firmware were reported. Checksums that don't match any checksum in the database are displayed along with a "*??*" version number.

ROMVER *(cont.)*

ROMVER *(cont.)*

NOTE

You must be a superuser to use **romver**, and **romver** resets the Paragon system.

Example

```
DS# romver

Paragon Flash EPROM Utility
Copyright (c) 1994 Intel Corporation
DIAG_REL_1.2.2 Fri Aug 19 16:31:41 PDT 1994

Flash version report targets [ GP MP MPFLEX MIO MDC HIPPI ]

loNode: 00A03(3),  hiNode: 00A14(14)

This program will reset the attached Paragon system
Please confirm with y/n (n): y

Initializing nodes ...
Loading /u/paragon/diag/flash.node

GP FLASH -                        ( expected count = 4,  actual
count = 4)
Version V3.3 found on the following nodes:
00A03 00A11 00A12 00A14
MP FLASH -                        ( expected count = 2,  actual
count = 2)
Version V2.0 found on the following nodes:
00A04 00A08
MPFLEX FLASH -                    ( expected count = 2,  actual
count = 2)
Version V1.0 found on the following nodes:

00A04 00A08
MIO FLASH -                       ( expected count = 1,  actual
count = 1)
Version V1.3 found on the following nodes:
00A03
MDC FLASH -             ( expected count = None,  actual count = 0)
HIPPI FLASH -                   ( expected count = None,  actual
count = 0)
Test Clean-Up.

DS#
```

ROMVER *(cont.)*

ROMVER (cont.)

See Also

flashutil

SCANTEST SCANTEST

scantest verifies the Paragon™ system JTAG scan hardware connected to the diagnostic station.

Syntax

scantest -c *Cabinets* **[-i] [-v]**

Arguments

-c *Cabinets*	Specifies the total number of cabinets installed in the system. This is a mandatory argument.
-i	Ignores testing a specific backplane or power controller row.
-v	Specifies using verbose messages.

Description

scantest verifies the Paragon system JTAG scan hardware connected to the diagnostic station. This test first evaluates the entire connection between the diagnostic station and the first connection within the first cabinet (backplane and power controller boards). Then the utility evaluates the scan bus row by row starting with "A". Every node that is present, through the backplane's NodePresent bits, has its scan bus tested. Error messages point to an area to begin debugging. This utility is most useful in debugging scan bus failures—especially for **hwcfg** scan failures. It is recommended that this utility be run at system installation prior to **hwcfg**, which relies on the scan bus to collect configuration information.

Example

To test five cabinets, fully configured with four backplanes and one power controller per cabinet, and display only standard error messages (non-verbose), enter:

```
DS# scantest -c 5
```

To test three cabinets, partially configured with two backplanes and no power controller, and display only standard error messages (non-verbose), type:

```
DS# scantest -c 3 -i
```

To test eight cabinets, fully configured, and display all messages (verbose), type the following command:

SCANTEST *(cont.)*

SCANTEST *(cont.)*

```
DS# scantest -c 8 -v
```

See Also

hwcfg, gencfg

SHOWCONF SHOWCONF

showconf displays the Paragon™ system binary configuration in a human-readable form.

Syntax

showconf [**-f** *binfile*] [[**-a**] | [**-c** <*Cabinet*> [**-b** <*backplane*> [**-s** <*slot*>]]]]

Arguments

-f *binfile*	Specifies the name of the binary configuration file. The default is */u/paragon/diag/SYSCONFIG.BIN*.
-a	Specifies that the entire configuration is to be displayed.
-c *cabinet*	Display the configuration of a specific cabinet. The valid range of cabinet numbers is what is permitted for the binary configuration file.
-b *backplane*	Display the configuration of a specific backplane. The **-c** option should also be specified. The valid range of backplanes is what is permitted for the binary configuration file.
-s *slot*	Display the configuration of a specific slot. The **-c** and **-b** options should also be specified. The valid range of slot numbers is *0 .. 15*.

Description

showconf displays the Paragon system binary configuration file in a human-readable format. The output format closely resembles *SYSCONFIG.TXT*. Without any options specified, by default, it displays the header and quick flags set up in */u/paragon/diag/SYSCONFIG.BIN*. The header of the binary configuration file consists of:

- A magic number indicating any major revisions in the config structure.
- *file size*, *first_cabinet*, *last_cabinet*, *first_backplane*, *last_backplane* and *bootnode*.
- *first_node* and *last_node*, which are used for dynamic scaling when using diagnostics.
- Information regarding the presence or absence of scan cables.

SHOWCONF *(cont.)* SHOWCONF *(cont.)*

When **showconf** is used with **-a** option, it displays the entire system configuration, providing information about all cabinets, backplanes, slots, controllers and the devices attached to the controllers. This switch is a useful option to compare the binary file with the contents of *SYSCONFIG.TXT*. Differences in the output format are:

- Devices are displayed in groups of controllers.
- OS-specific support fields such as *NOPAGER* and *PAGE_TO* are not displayed. *RAID3* or *RAID5* and *DAT* are displayed generically as *RAID* and *TAPE*.
- State information of various system units (cabinet, backplane, slot, power controller, LED controller) are displayed. For example, if a slot is set to be ignored dynamically during diagnostics, it is displayed *IGNORED*. Units that fail are reported with *FAILED* state.

The **-c**, **-b**, **-s** options are useful for displaying the configuration of a specific cabinet, backplane or slot.

Example

To display the entire configuration of the system, type:

```
DS# showconf -a
```

To display the configuration of cabinet 5, backplane 0, slot 5, type:

```
DS# showconf -c 5 -b 0 -s 5
```

To display the configuration of cabinet 5, all backplanes and slots, type:

```
DS# showconf -c 5
```

See Also

gencfg, **hwcfg**, **mergecfg**, **cfgpar**, **cfgmod** (file formats of *SYSCONFIG.TXT* and *DEVCONF.TXT*.)

Configuration File Format D

Understanding the Configuration Files

This appendix describes the new format of the *SYSCONFIG.TXT*, *DEVCONF.TXT*, and *MAGIC.MASTER* files that is being introduced with the DIAG1.2.3 update. These files reside in the directory */usr/paragon/boot*.

NOTE

The changes in this Appendix only apply to the DIAG1.2.3 update.

The *SYSCONFIG.TXT* and *DEVCONF.TXT* files are primarily hardware configuration files and *MAGIC.MASTER* is primarily a software configuration file.

To understand the format of the configuration files, you need to be familiar with Paragon™ system conventions for node numbering and backplane identification. Node numbering in *SYSCONFIG.TXT* and *DEVCONF.TXT* files follows the CBS numbering scheme, while node numbering in *MAGIC.MASTER* follows the OS numbering scheme. Both schemes are described in Appendix C of the *Paragon System Diagnostic Reference Manual.*

The *SYSCONFIG.TXT* File

The following example shows a line from a typical *SYSCONFIG.TXT* file. Each alphanumeric value represents the Field Replaceable Unit (FRU) and revision number of a component. The FRU type is represented by two alpha characters, number 00 as AA, 01 as AB, and so on. The revision is represented by a two-digit decimal number. (Refer to the Chapter 2 for a description of the FRU

types.) The following line indicates that physical slot number *0* contains a GP node with FRU type *10*, at revision level *10*, with 16 megabytes of memory, with an MIO type 1 and revision 02, and an MRC at revision 05. The MIO is connected to a RAID5, a DAT controller and an Ethernet controller.

```
S 0 GPNODE AK10 16 MRC 05 MIO B02 RAID5 DAT ENET
```

The following example shows a typical *SYSCONFIG.TXT* file. Note that backplanes are identified with letters as A, B, C, and D, with A closest to the floor.

```
CABINET 0                                    identifies the cabinet number

PC AU02                          identifies the power controller FRU type/revision level
LED AM00                          identifies the LED controller FRU type/revision level

BP A AC04                              identifies the backplane FRU type/revision level
S  0 GPNODE AK10 16 MRC 03 NIC A01
S  1 GPNODE AK10 16 MRC 03 NIC A01
S  2 GPNODE AK10 16 MRC 03 NIC A01
S  3 GPNODE AK10 16 MRC 03 NIC A01
S  4 GPNODE AK10 16 MRC 03 NIC A01
S  5 GPNODE AK10 16 MRC 03 NIC A01
S  6 GPNODE AK10 16 MRC 03 NIC A01
S  7 GPNODE AK10 16 MRC 03 NIC A01
S  8 GPNODE AK10 16 MRC 03 NIC A01
S  9 GPNODE AK10 16 MRC 03 NIC A01
S 10 GPNODE AK10 16 MRC 03 NIC A01
S 11 GPNODE AK10 16 MRC 03 NIC A01
S 12 GPNODE AK10 16 MRC 03 NIC A01
S 13 GPNODE AK10 16 MRC 03 NIC A01
S 14 GPNODE AK10 16 MRC 03 NIC A01
S 15 GPNODE AK10 16 MRC 03 NIC A01
   .
   .
   .
BP D AC04
S  0 EMPTY
S  1 EMPTY
S  2 GPNODE AK10 16 MRC 03 HIPPI 01C07 H04
S  3 GPNODE AK10 16 MIO B02 MRC 03 NIC A01 CTLR0 RAID5 DAT
S  4 EMPTY
S  5 EMPTY
S  6 EMPTY
S  7 EMPTY
S  8 EMPTY
S  9 EMPTY
S 10 EMPTY
S 11 EMPTY
```

```
S 12 EMPTY
S 13 EMPTY
S 14 EMPTY
S 15 EMPTY
```

The *DEVCONF.TXT* File

You must create your own *DEVCONF.TXT* file. The device information is enclosed between the keywords DEVICES and END_DEVICES. There is a one-line description for each device connected to each slot in the system. This document describes the general format of the device configuration specification and device specifics. In the description of a device, keywords are entered in upper-case and a placeholder for each field is described in angled brackets. For example, ID is a keyword and <IdNumber> is a placeholder. Each device that is described here is followed by an example entry. The general format of a device description is as follows:

<GenericName> <SlotInCBS> <DeviceSpecific>

- <GenericName> is one of {ENET, MIO, SIO, MDC, DISK, RAID, TAPE, HIPPI}
- <SlotInCBS> is of the form *nnxss* where

  nn Specifies the cabinet number in the range 00 through 99.

  x Specifies the backplane from one of the alpha characters {A,B,C,D}.

  ss Specifies the slot number in the range 00 through 15.

- <DeviceSpecific> fields are defined below for each device.

ENET

This specifies that an Ethernet is attached to the MIO board in the specified slot. It has no <DeviceSpecific> field.

To specify that (Cabinet 0, Backplane B, slot 5) has an ethernet, enter:

```
ENET 00B05
```

MIO

This entry describes the MIO board attached to a slot. The <DeviceSpecific> field for an MIO is of the form *frr*. The <DeviceSpecific> field *frr* is currently a placeholder where *f* is an alpha character in the range {A .. Z} and *rr* is a numeral in the range {00 .. 99}. A slot that has ENET or SCSI devices such as RAID, DISK, TAPE should either have an MIO or an SIO attached to it.

To specify that (Cabinet 1, Backplane A, Slot 12) has an MIO having a <DeviceSpecific> field with a value of A04, enter:

```
MIO 01A12 A04
```

MDC

This entry describes the memory daughter card attached to a slot. The <DeviceSpecific> field for an MDC is of the form *frr* <memsize>. The <DeviceSpecific> field frr is currently a placeholder where *f* is an alpha character in the range {A .. Z} and *rr* is a numeral in the range {00 .. 99} and <memsize> is one of {16, 32, 64, 128} in Mbytes

To specify that (Cabinet 1, Backplane A, Slot 12) has an MDC having a <DeviceSpecific> field with a value of A04 and 64 Mbytes, enter:

```
MDC 01A12 A04 64
```

DISK

This entry describes the Disk storage device attached to a slot. The <DeviceSpecific> fields are:

ID <IdNumber> [BC <DevCap>] <SymName> [PagingInfo] <Controller>

- <IdNumber> specifies the SCSI ID of the device from one of {0 .. 6}
- <DevCap> specified with BC is an optional field and is approximated to the nearest Gbyte of the device capacity. This field is used for diagnostic purpose.If this field is omitted, it defaults to 1.
- <SymName> is a symbolic name used by the Paragon System OS and is one of {MAXTOR, MAXTOR1240, PANTHER}
- [PagingInfo] is an optional field used for Paragon System OSF/1 debugging. The permitted arguments are NO_PAGER or PAGE_TO. The PAGE_TO argument is followed by a logical device name.
- <Controller> is described later in this section.

To specify a MAXTOR disk in (Cabinet 2, Backplanes C, Slot 11), using the logical device *rz0a* for paging, enter:

```
DISK 02C11 ID 5 MAXTOR PAGE_TO rz0a
```

RAID

This entry describes the RAID devices. The <DeviceSpecific> fields are:

ID <IdNum> SW <RevNum> LV <LevelNum> DC <devCnt> SID <ScsiIdNum> [BC <DevCap>] <SymName> [PagingInfo] <Controller>

- <IdNum> is the SCSI identification number of the RAID. Legal values are {0 .. 6}
- <RevNum> is the RAID software revision number. The current supported value is 3.02
- <LevelNum> is the RAID level number. Legal values are: {0, 1, 3, 5}
- <devCnt> is the number of devices attached to RAID controller.
- <ScsiIdNum> is the SCSI identification number of the devices attached to the RAID controller. Legal values are: {0 .. 6}
- <DevCap> specified with BC is an optional field and is approximated to the nearest Gbyte of the device capacity. This field is used for diagnostic purpose.If this field is omitted, it defaults to 4.
- <SymName> symbolic name used by the Paragon System OS. Legal values are {RAID3, RAID5}
- [PagingInfo] is an optional field used for Paragon System OSF/1 debugging. The permitted arguments are NO_PAGER or PAGE_TO. The PAGE_TO argument is followed by a logical device name.
- <Controller> is described later in this section.

Example:

```
RAID 00D03 ID 5 SW 3.02 LV 3 DC 5 SID 7 RAID3
RAID 00D02 ID 5 SW 3.02 LV 3 DC 5 SID 7 BC 2 RAID3 NOPAGER
RAID 00D04 ID 5 SW 3.02 LV 3 DC 5 SID 7 BC 2 RAID3 PAGE_TO rz0a
```

TAPE

This entry describes the tape device attached to a slot. The <DevSpecific> fields are:

ID <IdNum> <SymName> <Controller>

- <IdNum> is the SCSI identification number of the tape. Legal values are {0 .. 6}.
- <SymName> is the symbolic name used by the Paragon System OS. The only supported value is DAT

Example:

```
TAPE 050B08 ID 3 DAT
```

The <Controller> Field

This field is applicable to the <Controller> field of SCSI device {RAID, TAPE , DISK} entries. It specifies the controller to which a SCSI device is attached. Valid controller entries are {CTLR0, CTLR1}. By default the SCSI devices are assumed to be on CTLR0. A slot that has an MIO can only have its SCSI devices on CTLR0. However, a slot that has SCSI-16 (SIO) can have the SCSI devices either on CTLR0 and/or CTLR1.

```
RAID 00D03 ID 5 SW 3.02 LV 3 DC 5 SID 7 RAID3 CTLR1
SIO  00D03
```

The following defaults to CTLR0:

```
TAPE 00D04 ID 4
MIO  00D03 H04
```

HIPPI

This entry describes a HIPPI device attached to a slot. The <DevSpecific> field is:

frr which shows the FRU type and revision, where *f* is an alphabetical character in the range {A .. Z} and *rr* is a numeral in the range {00 .. 99}.

To specify a HIPPI device in location 00A09 (Cabinet 0, Backplane A, slot 9) of type H04, enter:

```
HIPPI 00A09 H04
```

Device combinations

In the following table, columns excluding the Description column are mutually exclusive entries for a given slot. The row describes a device or controller or none. An **Y** in the table cell indicates a valid device combination. An **N** indicates an invalid device combination.The item NONE in the description column together with MIO, for instance, indicates that a slot may have an MIO with no device attached to it.

Table D-1 Compatible Modules

Description	MIO	SIO	MDC	HIPPI
ENET	Y	Y	N	N
SCSI Device	Y	Y	N	N
CTLR0	Y	Y	N	N
CTLR1	N	Y	N	N
NONE	Y	Y	Y	Y

From the table, the following Combinations are valid for a given slot:

- A slot having just an MIO or SIO or MDC or HIPPI
- A slot having {MIO, ENET} or {MIO, SCSI device} or {MIO, SCSI Device, CTLR0} or {MIO, ENET, SCSI device, CTLR0}
- A slot having {SIO, ENET} or {SIO, SCSI device} or {SIO, ENET, {SCSI Device, CTLR0},{ SCSI Device, CTLR1}}

The following are some invalid combinations:

- A slot having {MIO, SIO} or {MIO, SCSI Device, CTLR1}
- A slot having {MDC, MIO} or {MDC, HIPPI}
- A slot having {MDC, SCSI Device, CTLR0}

Sample *DEVCONF.TXT* File

```
DEVICES

MIO 00A02 A04
MIO 02D03 A04
MIO 03D12 A04
MIO 03D15 A04
MIO 00B07 A04

ENET 00B07
RAID 02D03 ID 1 SW 3.02 LV 5 DC 5 SID 6 RAID3
DISK 03D12 ID 2 MAXTOR
```

```
TAPE 03D15 ID 3 DAT
HIPPI 00A09 H04

SIO 03D03 H04
RAID 03D03 ID 1 SW 3.02 LV 5 DC 5 SID 6 RAID3 CTLR1
RAID 03D03 ID 1 SW 3.02 LV 5 DC 5 SID 6 BC 2 RAID3 CTLR

MDC 04D04 A04 64

END_DEVICES
```

The *MAGIC.MASTER* File

The *MAGIC.MASTER* file must consist of at least the following lines:

```
BOOT_FIRST_NODE=7
BOOT_CONSOLE=cm
BOOT_GREEN_LED=Dci
BOOT_RED_LED=Dcgl
```

The node numbering here is OS numbering and identifies node 7 as the boot node in this example. Make sure that the *MAGIC.MASTER* file correctly identifies the boot node.

For systems that have two or more RAID subsystems, make sure that *MAGIC.MASTER* has a *DEV_TAB* bootmagic entry for all RAID subsystems that are additional to the one on the boot node. The format is as follows:

DEV_TAB=***<nbr1>nbr1:<nbr2>nbr2***

The *<nbr1>nbr1* is the root partition node number of the node connected to one RAID. Both *<nbr1>* and *nbr1* are the same. *<nbr2>* and *nbr2* identify the second RAID subsystem.

In general, you must add a *<nbr>nbr* entry for all devices attached to MIO nodes, except for the boot node.