The
Connection Machine
System

CM-5
Field Service Guide

Preliminary
October 9, 1992

Thinking Machines Corporation
Cambridge, Massachusetts

The
Connection Machine
System

CM-5
Field Service Guide

Preliminary
October 9, 1992

Thinking Machines Corporation
Cambridge, Massachusetts

First printing, July, 1992

Thinking Machines Corporation
245 First Street
Cambridge, Massachusetts 02142–1264
(617) 234–1000/876–1111

Contents

Chapter 1

Introduction

1.1 Purpose

The primary intent of this manual is to help you troubleshoot hardware problems in a CM-5 system. It contains a variety of information for this purpose, including: descriptions of the system hardware, a description of the error reporting system, descriptions of the diagnostic tools provided for troubleshooting hardware faults, and recommended diagnostic procedures.

How you use this manual will largely depend on how experienced you are with CM-5 systems.

- If you are new to the CM-5 and its diagnostic software environment, `cmdiag`, you should read through the entire manual at least once before you have occasion to use it. Then, when a troubleshooting situation arises, follow the basic troubleshooting procedure described in Section 2. This procedure will get you through the initial diagnostic steps and will guide you in using other sections of the manual as the particular troubleshooting session requires.

- If you have a good understanding of the system architecture and experience using `cmdiag`, you can treat this document as a reference manual, consulting it only for specific details.

NOTE: This manual assumes that you have received formal training on CM-5 system administration and maintenance issues. It does not provide comprehensive documentation of these topics.

1.2 General Troubleshooting Practices

The practices listed below have been found to promote more efficient troubleshooting in virtually all situations. They either simplify the troubleshooting task or they help avoid introducing new problems as old ones are investigated.

1. **Gather initial information.** — Before taking any active diagnostic steps, gather as much information about the failure as possible. Some questions that often uncover useful clues are listed below.

   - If the failure occurred while running a user program, ask if the program has run successfully on this CM-5 before. If so, has the CM changed in any way since then? If the answer is yes to these questions, find out what changes it has undergone since the program last ran successfully.

   - Has the user program run successfully on a different partition of this CM? If the answer is yes, focus attention on the hardware associated with the failing partition.

   - Likewise, has the program run successfully on another CM-5 system? If so, is that system different in any way — software version, hardware configuration, ECO levels, *etc*? If yes, consider the implications of those differences.

   - Have any other programs run successfully on the same CM partition? If yes, examine the differences between the successful and unsuccessful programs. For example, are the memory requirements of one program significantly different than the other?

2. **Check for simple solutions first.** — Check the obvious conditions, such as power supply or cooling fan failure. While these checks seldom lead to an immediate fix, they will avoid unnecessary troubleshooting time and effort on those few occasions when the solution is simple.

3. **Change as little as possible.** — Every modification to hardware has the inherent risk that it will introduce a new problem. When changes are unavoidable, try to adhere to the following guidelines:

   - Change one thing at a time and record all changes.

   - When you make a change that does not fix the problem, undo the change before progressing to the next step, particularly if that step involves making another change.

- If diagnostic messages point to a specific circuit board, check the board's seating before replacing it. Poor electrical contact caused by contaminated edge connectors or by inadequate seating is a common cause of faulty performance. Reseating a circuit board will help clean the metal surfaces and re-establish solid contact. After reseating a board, retest to see if the fault was corrected.

4. **Swap boards before changing cables. —** Use board swapping for fault isolation before changing cable connections. In a system that has been running successfully, cable faults are much less likely than component or board failures. In addition, disconnecting and reconnecting cables poses more risk of causing a new problem than replacing circuit boards.

5. **For intermittent problems, increase the length of test runs to stress the hardware being tested.**

6. **ALWAYS WEAR ANTI-STATIC PROTECTION WHEN HANDLING CIRCUIT BOARDS. STORE BOARDS IN ANTI-STATIC BAGS.**

1.3 Summary of Diagnostic Tools

Various diagnostic tools are provided for troubleshooting hardware failures on the CM. The following list summarizes these tools and indicates where you can find explanations of their use.

- Use **kpndbx** to investigate Processing Node failures. The procedure for using **kpndbx** is described in Appendix A. Its man page is provided in Appendix M.

- The primary diagnostic tool for investigating hardware faults in the CM-5 is `cmdiag`. Its use is described in Appendix B. The `cmdiag` man page is in Appendix M.

- A subset of `cmdiag` tests target IOBA hardware. These tests are augmented by several independent test packages, which exercise the different I/O devices and their interconnecting hardware. The various I/O-related diagnostic tools are described in Appendix F.

- A number of system verifiers are available for exercising the CM across functional boundaries. These provide comprehensive coverage of system functions by closely emulating the behavior of user applications. These verifiers are described in Appendix H.

Chapter 2

Troubleshooting Fundamentals

Often, the initial stage of a troubleshooting session — deciding what action to take first — can be the most difficult. This chapter offers a brief set of guidelines for dealing with this early phase.

The steps presented below offer a rational opening strategy for troubleshooting CM-5 hardware faults, regardless of the source of the failure. Figure 1 illustrates the key points in this procedure.

NOTE: As a matter of convenience, you can have **`cmdiag`** running on the full system at all times. This **`cmdiag`** would not be associated with any individual partition (*i.e.*, it would not be invoked with the **`-p`** option) and would therefore not interfere with timesharing daemons running on partitions. Appendix M contains the **`cmdiag`** man page.

1. If **`cmdiag`** is not already running, invoke it on the entire CM (do not use the **`-p`** option).

2. Run **`find-cm-error`**. See Appendix K for a description of **`find-cm-error`** output.

3. The next step depends on what **`find-cm-error`** reports.

   - If no errors are reported, the problem may be in a Processing Node or in an area of I/O hardware that is not accessible to the diagnostic network. In either case, PN registers may provide useful status information. Run **`kpndbx`** to read PN status. Appendix A describes the **`kpndbx`** procedure.

   - If Control Network errors are reported, use the troubleshooting procedure described in Appendix D, Tracing Control Network Errors.

- If Data Network errors are reported, use the troubleshooting procedure described in Appendix E, Tracing Data Network Errors.
- If `find-cm-error` points to I/O hardware, use the troubleshooting tools described in Appendix F.
- If all chips on a backplane or on multiple backplanes report errors, the source of the problem can be a faulty power supply, system clock, or diagnostic network. Appendix G describes the procedure for troubleshooting symptoms of this kind.

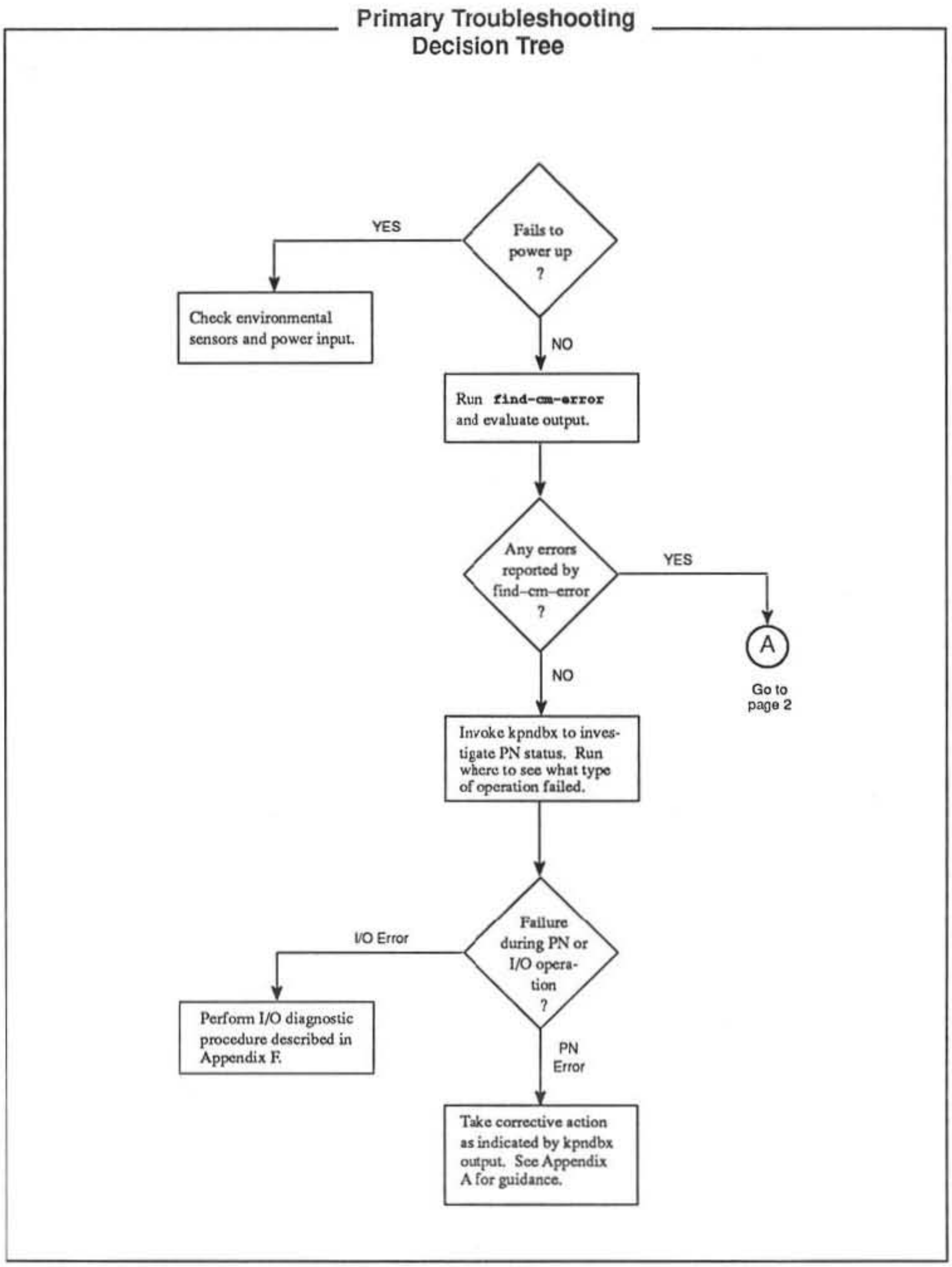


Figure 1. Strategy for initial phase of troubleshooting session.
(1 of 2)

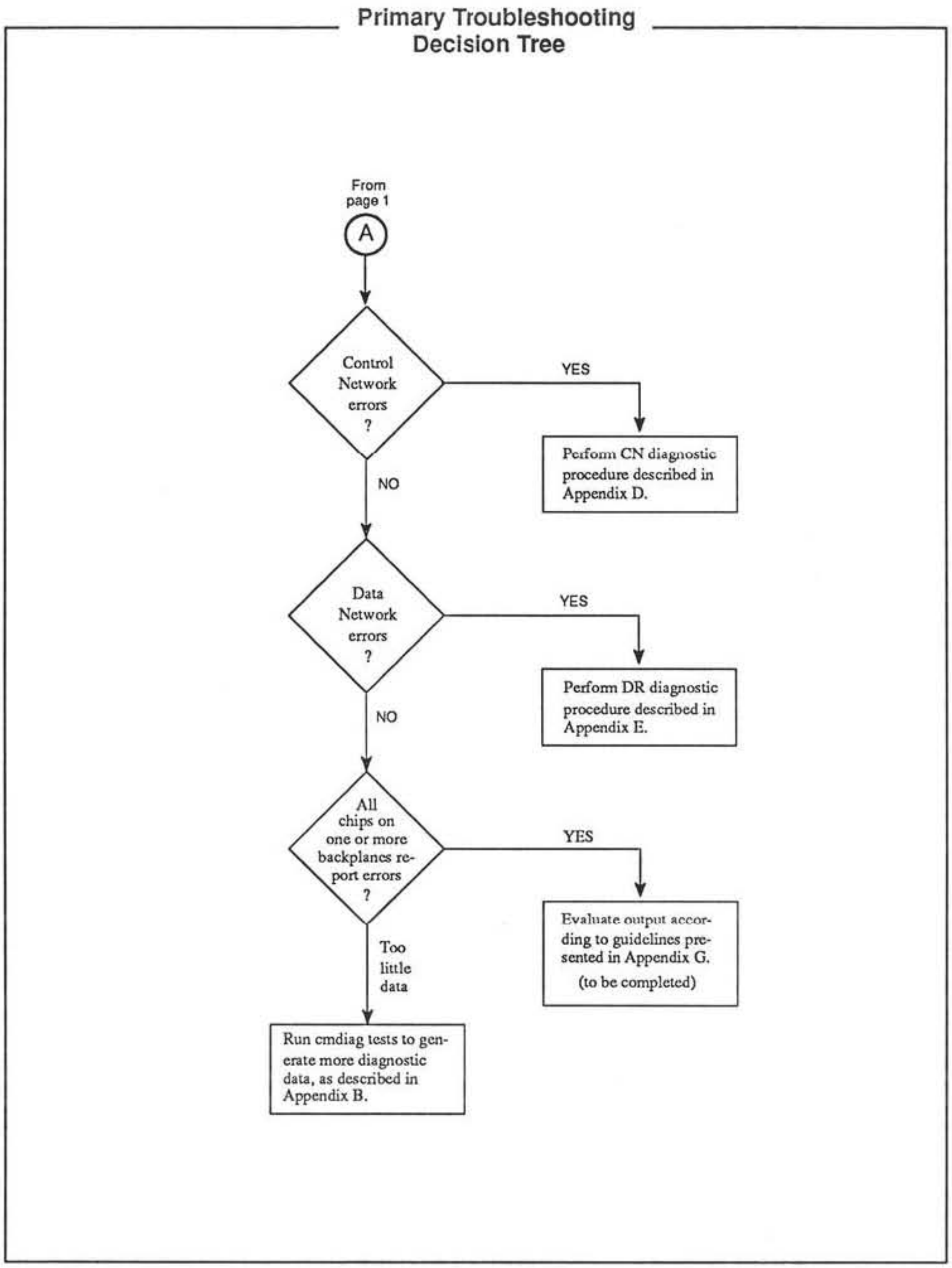


Figure 1. Strategy for initial phase of troubleshooting session.
(2 of 2)

Chapter 3

Preventive Maintenance

3.1 Summary

The CM-5 preventive maintenance program is intended to expose incipient hardware faults in a controlled setting, reducing the likelihood of failures occurring while user code is executing. There are two schedules in the CM-5 preventive maintenance program — a short, less comprehensive daily routine and a longer, more rigorous weekly procedure.

- The daily routine implements the Processing Node test group, the Data Network verifier group, and the Control Network verifier group. This procedure is illustrated in Figure 2; detailed descriptions of each step are provided in Section 3.2 and are cross-referenced in the margin of Figure 2.

- The weekly program involves running the JTAG tests in addition to the daily test groups. Figure 3 illustrates this procedure in transcript form with cross references to detailed descriptions in Section 3.2.2.

  If the system includes I/O hardware, several I/O tests provided by **cmdiag** are added to the weekly regimen. This expanded procedure is illustrated in Figure 4 with cross references to detailed descriptions in Section 3.3.3.

  NOTE: Currently, the weekly maintenance procedure is not compatible with user partitions and, so, requires exclusive use of the system.

cmdiag includes an interface to the **cmpartition** software. This interface allows you to use the high-level **cmpartition** functions to restrict the scope of **cmdiag** to a subset of the system hardware. Tests run within that partition will not interfere with user applications running in other partitions.

Daily PM Procedure

Introductory Notes

The system used to illustrate this procedure example has the following attributes.

- System is named Calliope and has 256 PNs.
- Diagnostic server is named **homer.think.com.**
- Calliope has two 128-PN partitions, which are allocated to partition managers named **virgil.think.com** and **milton.think.com.**
- Diagnostics will be run on **virgil.think.com.**

1

```
login: user_id
% su
password: root_password
SU homer.think.com /dev/console
hom# cd /usr/diag/cmdiag
```

2

```
hom# setenv CMDIAG_PATH /usr/diag/cmdiag
hom# setenv JTAG_SERVER homer.think.com
```

3

```
hom# /usr/etc/cmpartition list -l
  CM System "Calliope"
  256 Processors [ 8 Mbytes memory, SPARC IU, SPARC FPU ]
   2 Partition Managers
  virgil.think.com
  milton.think.com
Available PN Ranges:
  All PNs in use
IOP Addresses
  480

Name   Partition Manager  Size  State    Nodes     Description
V128   virgil.think.com   128   ACTIVE   0-127     virgil
M128   milton.think.com   128   ACTIVE   128-255   milton
                                         480-480
```

4

```
hom# rlogin -l root virgil.think.com
password: QuiVive
virg# setenv CMDIAG_PATH /usr/diag/cmdiag
virg# setenv JTAG_SERVER homer.think.com
virg# cmpartition stop -pm virgil.think.com
```

(continued on next page)

Figure 2. Daily preventive maintenance — 1 of 2

Daily PM Procedure

(continued from previous page)

5

```
virg# cmpartition list -l
  CM System "Calliope"
  256 Processors [ 8 Mbytes memory, SPARC IU, SPARC FPU ]
   2 Partition Managers
  virgil.think.com
  milton.think.com
Available PN Ranges:
  All PNs in use
IOP Addresses
  480

Name    Partition Manager  Size State     Nodes     Description
V128    virgil.think.com   128  ALLOCATED 0-127     virgil
M128    milton.think.com   128  ACTIVE    128-255   milton
                                          480-480
```

6

```
virg# cmdiag -C -p virgil.think.com
<CM-DIAG> rgroups m PN global broadcast combine dr partition
              .
                                    ; diagnostic test report
              .
```

NOTE

When the partition managed by **milton** becomes available for testing, repeat steps 4 through 6 on **milton**.

Figure 2. Daily preventive maintenance — 2 of 2.

Weekly PM without I/O

Introductory Notes

The system used to illustrate this procedure example has the following attributes.

- System is named Calliope and has 256 PNs.
- Diagnostic server is named **homer.think.com**.
- Calliope has two 128-PN partitions, which are allocated to partition managers named **virgil.think.com** and **milton.think.com**.
- Diagnostics will be run on **homer.think.com**.

1

```
login: user_id
% su
password: root_password
SU homer.think.com /dev/console
hom# cd /usr/diag/cmdiag
```

2

```
hom# setenv CMDIAG_PATH /usr/diag/cmdiag
hom# setenv JTAG_SERVER homer.think.com
```

3

```
hom# /usr/etc/cmpartition list -l
  CM System "Calliope"
  256 Processors [ 8 Mbytes memory, SPARC IU, SPARC FPU ]
   2 Partition Managers
  virgil.think.com
  milton.think.com
Available PN Ranges:
  All PNs in use
IOP Addresses
  480
```

```
Name   Partition Manager  Size State     Nodes     Description
V128   virgil.think.com   128  ACTIVE    0-127     virgil
M128   milton.think.com   128  ALLOCATED 128-255   milton
                                         480-480
```

4

```
hom# rlogin -l root virgil.think.com
password: QuiVive
virg# setenv CMDIAG_PATH /usr/diag/cmdiag
virg# setenv JTAG_SERVER homer.think.com
virg# cmpartition stop
virg# cmpartition delete
virg# exit
```

(continued on next page)

Figure 3. Weekly maintenance with no I/O — 1 of 2

Weekly PM without I/O

(continued from previous page)

4 (cont.)
```
hom# cmpartition delete -pm milton.think.com
```

5
```
hom# cmpartition list -l
  CM System "Calliope"
  256 Processors [ 8 Mbytes memory, SPARC IU, SPARC FPU ]
   2 Partition Managers
  virgil.think.com
  milton.think.com
Available PN Ranges:
  All PNs in use
IOP Addresses
  480
```

6, 7
```
hom# ./cmdiag -C
<CM-DIAG> rgroups m SVME
<CM-DIAG> rgroups m CLKDN
<CM-DIAG> rgroups m CLKBUF
<CM-DIAG> rgroups m SPI
<CM-DIAG> rgroups m FILLER
<CM-DIAG> rgroups m PE PEMEM
<CM-DIAG> rgroups m CN
<CM-DIAG> rgroups m DR
           .
           .                    ; diagnostic test report
           .
```

8
```
CM-DIAG> q
hom# /usr/etc/cmpartition create -pn_range 0-255
```

9
```
hom# cmreset
hom# cmreset -s
hom# ./cmdiag -C -p homer.think.com
<CM-DIAG> rgroups m PN dr combine global broadcast partition
```

Figure 3. Weekly maintenance with no I/O — 2 of 2.

Weekly PM with I/O

Introductory Notes

The system used to illustrate this procedure example has the following attributes.

- System is named Calliope and has 256 PNs.
- Diagnostic server is named **homer.think.com**.
- Calliope has two 128-PN partitions, which are allocated to partition managers named **virgil.think.com** and **milton.think.com.**
- Diagnostics will be run on **homer.think.com** partition.

*** INITIALIZE SYSTEM ***

1
```
login: user_id
% su
password: root_password
SU homer.think.com /dev/console
hom# cd /usr/diag/cmdiag
```

2
```
hom# setenv CMDIAG_PATH /usr/diag/cmdiag
hom# setenv JTAG_SERVER homer.think.com
```

3
```
hom# /usr/etc/cmpartition list -l
  CM System "Calliope"
  256 Processors [ 8 Mbytes memory, SPARC IU, SPARC FPU ]
   2 Partition Managers
  virgil.think.com
  milton.think.com
Available PN Ranges:
  All PNs in use
IOP Addresses
  480

Name   Partition Manager  Size  State      Nodes     Description
V128   virgil.think.com   128   ACTIVE     0-127     virgil
M128   milton.think.com   128   ALLOCATED  128-255   milton
                                           480-480
```

4
```
hom# rlogin -l root virgil.think.com
password: QuiVive
virg# setenv CMDIAG_PATH /usr/diag/cmdiag
virg# setenv JTAG_SERVER homer.think.com
virg# cmpartition stop
virg# cmpartition delete
virg# exit
```

(continued on next page)

Figure 4. Weekly preventive maintenance with I/O — 1 of 3

Weekly PM with I/O

(continued from previous page)

4 (cont.)

```
hom# rlogin -l milton.think.com
password: QuiVaLa
milt# setenv CMDIAG_PATH /usr/diag/cmdiag
milt# setenv JTAG_SERVER homer.think.com
milt# cmpartition delete
milt# exit
```

5

```
hom# cmpartition list -l
  CM System "Calliope"
  256 Processors [ 8 Mbytes memory, SPARC IU, SPARC FPU ]
   2 Partition Managers
  virgil.think.com
  milton.think.com
Available PN Ranges:
  All PNs in use
IOP Addresses
  480
```

6, 7

*** RUN COMPLETE JTAG TESTS ***

```
hom# ./cmdiag -C
<CM-DIAG> rgroups m
        .
        .                 ; diagnostic test report
        .
```

8, 9

*** TEST DATAVAULTS ***

```
dvlogin:user_id
password: root_password
     .
     .
dv# /usr/local/etc//diag/dvcoldboot +cn
dv# /usr/local/etc/diag/diagserver/diagserver &
```

(continued on next page)

Figure 4. Weekly preventive maintenance with I/O — 2 of 3

Weekly PM with I/O

(continued from previous page)

```
10–14 <CM-DIAG> execute-all-iopdv-tests
                 .
                 .                   ; diagnostic test report
                 .
      <CM-DIAG> execute-all-ioppe-tests
                 .
                 .                   ; diagnostic test report
                 .

            *** TEST CM–HIPPI & CM–IOPG ***

15–18 <CM-DIAG> test-cmio-device-data-xfer
                   .
                   .                 ; diagnostic report
                   .

   *** RUN PN & NETWORK TESTS ON SYSTEM-WIDE PARTITION ***

19–21 <CM-DIAG> q
      hom# /usr/etc/partition create -pn_range 0-255
      hom# cmreset
      hom# cmreset -s
      hom# ./cmdiag -C -p homer.think.com
      <CM-DIAG> rgroups m PN dr combine global broadcast partition
                   .
                   .                 ; diagnostic report
                   .

               *** RUN I/O VERIFIERS ***

22–31 <CM-DIAG> q
      hom# /usr/etc/cmpartition start -cmd ts-daemon
      hom# setenv DVWD dv_server_name
      hom# /usr/local/etc/dvtest5-vu -h -g 64, 64
                   .
                   .                 ; diagnostic report
                   .
      hom# setenv DVWD hippi_server_name
      hom# /usr/local/etc/hippi-loop5
                   .
                   .                 ; diagnostic report
                   .
```

Figure 4. Weekly preventive maintenance with I/O — 3 of 3.

3.2 Daily Preventive Maintenance

3.2.1 Initial Conditions

The diagnostic procedure described in Section 3.2.2 assumes that the partition from which you will run **cmdiag** already exists. If this is not the case and you need to create the partition, refer to the **cmpartition** man page in Appendix M for instructions.

NOTE: Currently, the only **cmdiag** test groups that may be run within a partition without affecting other partitions, are the PE and verifier test groups. Consequently, only these functions will be used during the daily preventive maintenance activities.

3.2.2 Diagnostic Procedure

Perform the procedure described below each day.

NOTE: If your CM system includes I/O facililities, these will be tested during the weekly maintenance sessions when you have full use of the system.

1. Login at the CM-5 System Administration Console as root, and change directory to **/usr/diag/cmdiag**.

   ```
   login:user_id
   password: root_password
               .
               .
   # cd /usr/diag/cmdiag
   ```

2. Set the **CMDIAG_PATH** and **JTAG_SERVER** environment variables. The default **CMDIAG_PATH** is **/usr/diag/cmdiag**. The **JTAG_SERVER** variable must specify the hostname of the diagnostic server.

   ```
   # setenv CMDIAG_PATH /usr/diag/cmdiag
   # setenv JTAG_SERVER diag_server_hostname
   ```

3. Run **cmpartition list -l** to be certain you have an accurate understanding of the current partitioning status of the CM — what partition configurations are in effect, their names, the hostnames of their partition managers, and their state of use.

   ```
   # /usr/etc/cmpartition list -l
   ```

4. If `cmpartition list -l` reports the state of the target partition as ACTIVE, it means `ts-daemon` is running on that partition. If so, `rlogin` to the appropriate partition manager and run `cmpartition stop` to halt the timesharing daemon. Then exit.

   NOTE: The following example shows `CMDIAG_PATH` and `JTAG_SERVER` being set. If these environment variables are already set on this partition manager, this step can be skipped.

   ```
   # rlogin -l root pm_name
   password: root_password
   # setenv CMDIAG_PATH /usr/diag/cmdiag
   # setenv JTAG_SERVER diag_server_hostname
   # /usr/etc/cmpartition stop
   # exit
   ```

   pm_name is the name of the targeted partition manager.

5. Run `cmpartition list -l` again. The target partition should now show an ALLOCATED status. This means the partition is defined and still associated with its partition manager, but the timesharing daemon is not running.

6. Run the daily preventive maintenance test groups.

   ```
   # cmdiag -C -p pm_name
   <CM-DIAG> rgroups m PN dr combine global broadcast
             partition
   <CM-DIAG>
   ```

   The `-p` *pm_name* option specifies the partition in which `cmdiag` will be run; *pm_name* is the hostname of the Partition Manager.

7. If any test fails, record the messages generated by the tests and notify Thinking Machines product support — (617) 234-4000.

   If no test fails, the daily preventive maintenance procedure is now complete. Return the CM-5 to regular use.

3.3 Weekly Preventive Maintenance

3.3.1 Initial Conditions

The weekly preventive maintenance procedure requires that you have exclusive use of the system for the duration of the test session.

The test sequence differs greatly depending on whether or not there is I/O hardware to be tested. Section 3.3.2 describes the procedure for systems with no I/O. Section 3.3.3 covers systems with I/O.

3.3.2 Weekly Test Procedure with No I/O

The following procedure is summarized in Figure 3 for quick reference.

1. Login at the CM-5 System Administration Console as root, and change directory to **`/usr/diag/cmdiag`**.

   ```
   login:user_id
   password: root_password
               .
               .
   # cd /usr/diag/cmdiag
   ```

2. Set the **`CMDIAG_PATH`** and **`JTAG_SERVER`** environment variables. The default **`CMDIAG_PATH`** is **`/usr/diag/cmdiag`**. The **`JTAG_SERVER`** variable must specify the hostname of the diagnostic server.

   ```
   # setenv CMDIAG_PATH /usr/diag/cmdiag
   # setenv JTAG_SERVER diag_server_hostname
   ```

3. Stop and delete all partitions. To do this, you need to know the hostname of each partition manager to which a partition is allocated. If necessary, run `cmpartition list -l` to get this information.

   ```
   # /usr/etc/cmpartition list -l
   ```

4. Then run **`cmpartition stop`** and **`cmpartition delete`** on every partition manager that has an ACTIVE partition. Run **`cmpartition delete`** on every partition manager that has an ALLOCATED partition.

   For example, if **`cmpartition list`** shows **`virgil.think.com`** as ACTIVE and **`milton.think.com`** as ALLOCATED, perform the steps shown below.

NOTE: This example is structured to demonstrate certain characteristics of the **`cmpartition stop`** and **`delete`** commands.

- Because **`cmpartition stop`** must be performed on the partition manager controlling the partition to be stopped, this example includes an **`rlogin`** to **`virgil`**, which has an ACTIVE partition.
- **`cmpartition delete`**, however, can be done remotely. Consequently, the inactive partition on **`milton`** is deleted from the diagnostic console. See step 4 (cont.) in Figure 3.

```
# rlogin -l root virgil.think.com
password: root_password
virg# /usr/etc/cmpartition stop
virg# /usr/etc/cmpartition delete
virg# exit
# /usr/etc/cmpartition delete -pm milton.think.com
#
```

NOTE: This example assumes that **`CMDIAG_PATH`** and **`JTAG_SERVER`** are already set appropriately on both partition managers. If these environment variables are not correct, log in to each partition manager and set them as follows.

```
# rlogin -l root virgil.think.com
password: root_password
virg# setenv CMDIAG_PATH /usr/diag/cmdiag
virg# setenv JTAG_SERVER diag_server_hostname
                .
                .
virg# exit
# rlogin -l root milton.think.com
password: root_password
milt# setenv CMDIAG_PATH /usr/diag/cmdiag
milt# setenv JTAG_SERVER diag_server_hostname
                .
                .
milt# exit
```

5. Run **`cmpartition list -l`** again. It should report no partitions either ACTIVE or ALLOCATED.

6. Run the manufacturing version of the JTAG test group.

```
# ./cmdiag -C
<CM-DIAG> rgroups m SVME
<CM-DIAG> rgroups m CLKDN
<CM-DIAG> rgroups m CLKBUF
```

```
<CM-DIAG> rgroups m SPI
<CM-DIAG> rgroups m FILLER
<CM-DIAG> rgroups m PE PEMEM
<CM-DIAG> rgroups m CN
<CM-DIAG> rgroups m DR
```

7. If any test fails, record the error messages generated by the tests and notify Thinking Machines product support — (617) 234-4000.

   If no test fails, go to step 8.

8. Create a partition that encompasses all PNs in the system. Enter the lowest and highest PN network addresses for *first_pn–last_pn*, respectively.

```
<CM-DIAG> q
# /usr/etc/cmpartition create -pn_range first_pn-last_pn
```

9. Execute a system reset and reset the Partition Manager's interface module. Then run the processor chip tests, followed by the Data Network and Control Network verifiers.

```
# cmreset
# cmreset -s
# ./cmdiag -C -p pm_name
<CM-DIAG> rgroups m PN dr combine global broadcast
          partition
```

   pm_name is the hostname of the Partition Manager.

10. If any test fails, record the error messages generated by the tests and notify Thinking Machines product support — (617) 234-4000.

    If no test fails, go to step 11.

11. If the system has multiple Partition Managers, repeat steps 8 and 9, using a different Partition Manager each time.

    cmreset -s must be repeated for each Partition Manager that is used to run **cmdiag**.

12. When the CM-5 passes all tests invoked in steps 6 through 9, the preventive maintenance session is complete. Return the system to regular use. This requires stopping and deleting the system-wide partition created in step 8 and recreating and starting the partitions deleted in step 4.

3.3.3 Weekly Test Procedure with I/O

The weekly preventive maintenance procedure is described below. Because it involves many steps, its description is organized into several phases to minimize confusion. The procedure is also summarized in Figure 4 for quick reference.

INITIALIZE SYSTEM

The following steps take the system from its normal operating configuration, preparing it for the first diagnostics sequence.

1. Login at the CM-5 System Administration Console as root, and change directory to **/usr/diag/cmdiag**.

```
login:user_id
password: root_password
            .
            .
# cd /usr/diag/cmdiag
```

2. Set the **CMDIAG_PATH** and **JTAG_SERVER** environment variables. The default **CMDIAG_PATH** is **/usr/diag/cmdiag**. The **JTAG_SERVER** variable must specify the hostname of the diagnostic server.

```
# setenv CMDIAG_PATH /usr/diag/cmdiag
# setenv JTAG_SERVER diag_server_hostname
```

3. Stop and delete all partitions. To do this, you need to know the hostname of each partition manager to which a partition is allocated. If necessary, run **cmpartition list -l** to get this information.

```
# /usr/etc/cmpartition list -l
```

4. Then run **cmpartition stop** and **cmpartition delete** on every partition manager that has an ACTIVE partition. Run **cmpartition delete** on every partition manager that has an ALLOCATED partition.

   For example, if **cmpartition list** shows **virgil.think.com** as ACTIVE and **milton.think.com** as ALLOCATED, do the following.

```
# rlogin -l root virgil.think.com
password: root_password
virg# /usr/etc/cmpartition stop
virg# /usr/etc/cmpartition delete
virg# exit
# rlogin -l root milton.think.com
password: root_password
milt# /usr/etc/cmpartition delete
```

```
milt# exit
#
```

5. Run **cmpartition list -l** again. It should report no partitions either ACTIVE or ALLOCATED.

RUN COMPLETE JTAG TESTS

6. Run the manufacturing version of **cmdiag rgroups**. This will perform the complete JTAG test suite, including all IOBA hardware identified in the **io.conf** configuration file.

```
# ./cmdiag -C
<CM-DIAG> rgroups m
```

7. If any test fails, record the error messages generated by the tests and notify Thinking Machines product support — (617) 234-4000.

   If no test fails, go to step 8.

TEST DATAVAULTS

8. If the system includes DataVaults, perform steps 9 through 14. If there are no DataVaults to test, skip to step 15.

9. Log on to the station manager of the first DataVault you plan to test and set the command-channel mode by running **dvcoldboot +c***n*. *n* specifies which DataVault port will be used—use either 0 or 1.

   While you are at the DataVault console, start its diagnostic server running in background. The DataVault diagnostic server will be needed in step 17.

```
login:user_id
password: root_password
            .
            .
dv# /usr/local/etc/diag/dvcoldboot +cn
dv# /usr/local/etc/diag/diagserver/diagserver &
```

10. Run the **iopdv** test from within **cmdiag**.

```
<CM-DIAG> execute-all-iopdv-tests
```

   NOTE: If the IOP and DataVault station IDs and the DataVault starting block are not already defined, you will be prompted to supply them. Specify a DataVault starting block address no higher than 960; this will ensure that test data will not exceed the 1024-block zone reserved for diagnostic use on the DataVault.

11. If any test fails, record the error messages generated by the tests and notify Thinking Machines product support — (617) 234-4000.

    If no test fails, go to step 12.

12. Run the **ioppe** tests from within **cmdiag**.

```
<CM-DIAG> execute-all-ioppe-tests
```

13. If any test fails, record the error messages generated by the tests and notify Thinking Machines product support — (617) 234-4000.

    If no test fails, go to step 14.

14. Repeat steps 9 through 11 for each DataVault in the system. Then go on to step 15.

TEST CM-HIPPI and CM-IOPG

15. If CM-HIPPI and/or VMEIO devices are also attached to the CM-5, log on to their station managers as root and start their diagnostic servers running in background. Otherwise, just proceed to step 16.

16. Verify that the file `cmio_config.`*machine_name* is present on the System Administration Console. It will be used by the end-to-end tests, which will be executed next.

17. Now, run the **cmdiag** end-to-end tests. The following command will automatically invoke the appropriate tests for all DataVaults, CM-HIPPIs, and VMEIO devices connected to the CM-5.

```
<CM-DIAG> test-cmio-device-data-xfer
```

18. If any test fails, record the error messages generated by the test and notify Thinking Machines product support — (617) 234-4000.

    If no test fails, go to step 19.

CREATE SYSTEM-WIDE PARTITION and RUN PROCESSOR TESTS and NETWORK VERIFIERS

19. Create a partition that encompasses all PNs in the system. Enter the lowest and highest PN network addresses for *first_pn–last_pn*.

```
<CM-DIAG> q
# /usr/etc/cmpartition create -pn_range first_pn-last_pn
```

20. Execute a system reset and reset the Partition Manager's interface module. Then run the processor chip tests, followed by the Data Network and Control Network verifiers.

```
# cmreset
# cmreset -s
# cmdiag -C -p pm_name
<CM-DIAG> rgroups m PE dr combine global broadcast
          partition
<CM-DIAG>
```

    pm_name is the hostname of the Partition Manager and specifies the partition in which **cmdiag** will be run.

21. If any test fails, record the error messages generated by the tests and notify Thinking Machines product support — (617) 234-4000.

    If no test fails, go to step 22.

RUN I/O VERIFIERS

22. When the CM-5 passes all tests invoked up through step 19, it is time to run the system verifiers that include full-speed I/O. This procedure begins at step 23.

23. Ensure that **fsserver** is running on all DataVaults, CM-HIPPIs, and VMEIO devices connected to the CM-5.

24. Start the timesharing daemon on the partition created in step 19.

```
<CM-DIAG> q
# /usr/etc/cmpartition start -cmd ts-daemon
```

25. Next, choose one DataVault or VMEIO device and set the DVWD environment variable to specify that device. *server_name* is the hostname of the file server running on the DataVault or VMEIO.

```
# setenv DVWD server_name:
```

26. Run the hardware portion of **dvtest5**. Use the **-g** argument to specify a geometry that will produce a data block size appropriate for the I/O device. For example, the recommended geometry values for a DataVault are:

```
# /usr/diag/tsd/dvtest5 -h -g 64,64
```

This will produce 16-Kbyte blocks, which matches the DataVault block size. Smaller block sizes are typically used for VMEIO devices, the exact size depending on the storage characteristics of the device.

27. If **dvtest5** fails, record the error messages generated by the tests and notify Thinking Machines product support — (617) 234-4000.

If it does not fail, go to step 28.

28. Repeat steps 24 through 27 for every DataVault and VMEIO device connected to the CM-5.

29. When **dvtest5** has been run on all DataVaults and VMEIO devices, run the **hippi-loop** verifier for each CM-HIPPI connected to the CM-5. Change the DVWD environment variable to specify the CM-HIPPI.

```
# setenv DVWD server_name:
# /usr/diag/tsd/hippi-loop
```

30. If **hippi-loop** fails, record the error messages generated by the tests and notify Thinking Machines product support — (617) 234-4000.

If it does not fail, go to step 31.

31. Repeat steps 29 and 30 for each CM-HIPPI device.

32. When all DataVault, VMEIO, and CM-HIPPI devices have passed **`dvtest5`** and **`hippi-loop`**, the weekly preventive maintenance session is complete.

    Return the CM-5 and its I/O devices to regular use. To do this, stop and delete the system-wide partition created in step 19 and recreate and restart the partitions deleted in step 4.

Chapter 4

System Startup and Shutdown

This chapter describes the procedure for bringing a CM-5 from a powered down condition to the state where it is ready to run user programs. It shows how to create partitions and start the timesharing daemon running on them. It also explains how to stop the timesharing daemon, delete partitions, and shut down the CM-5.

These procedures are presented in several levels of detail, from a high-level view of the general tasks to detailed descriptions of each step.

- Figure 5 and Figure 6 identify the major tasks involved in powering a CM-5 system up and down, including partition creation and control.
- Figure 7 and Figure 8 present the individual steps involved in each power-up and power-down task in a quick-reference format.
- Sections 4.1 and 4.2 provide detailed descriptions of these procedures.

The power-up procedure assumes that the CM-5 is completely installed (hardware and software), including all cabling.

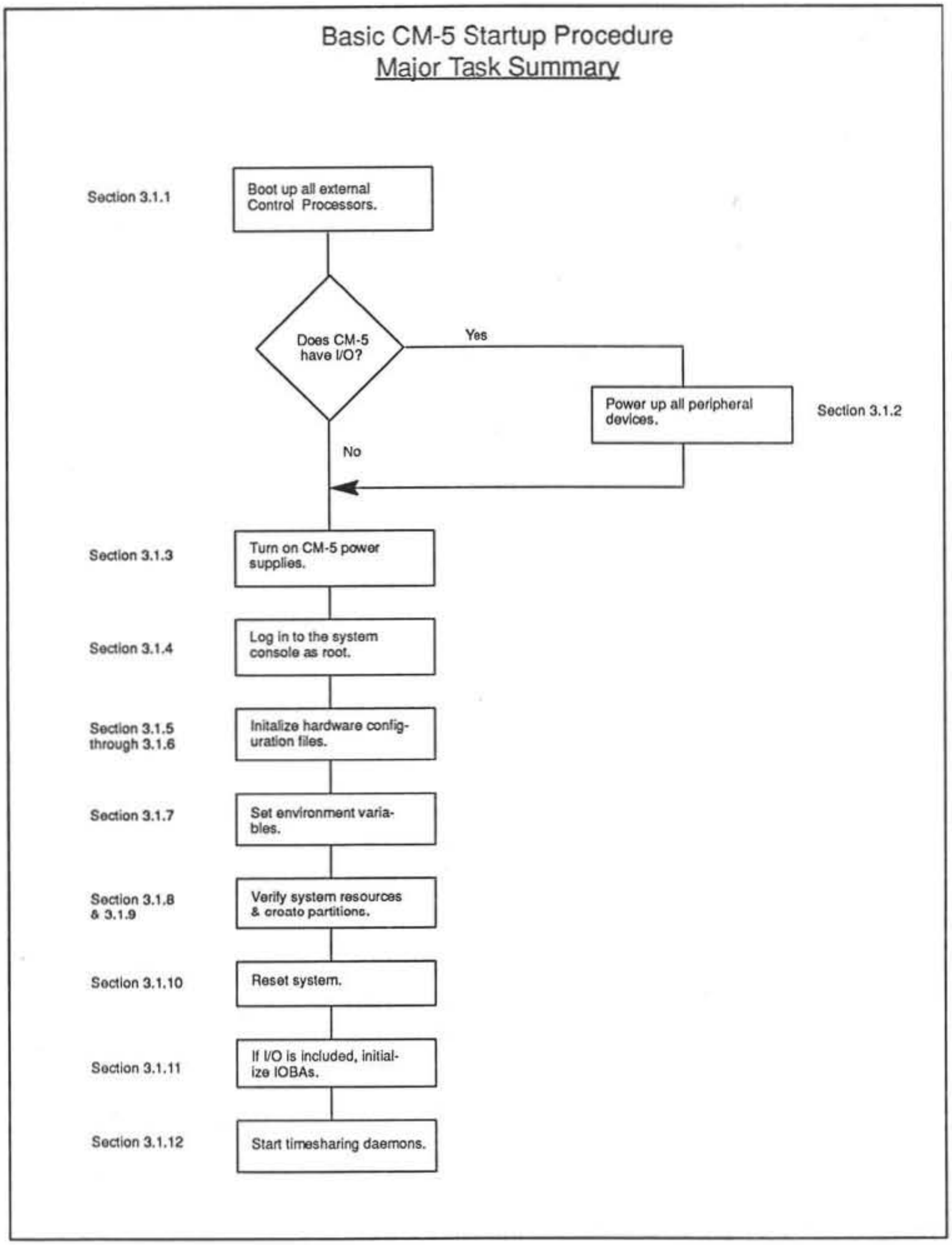


Figure 5. CM-5 startup procedure.

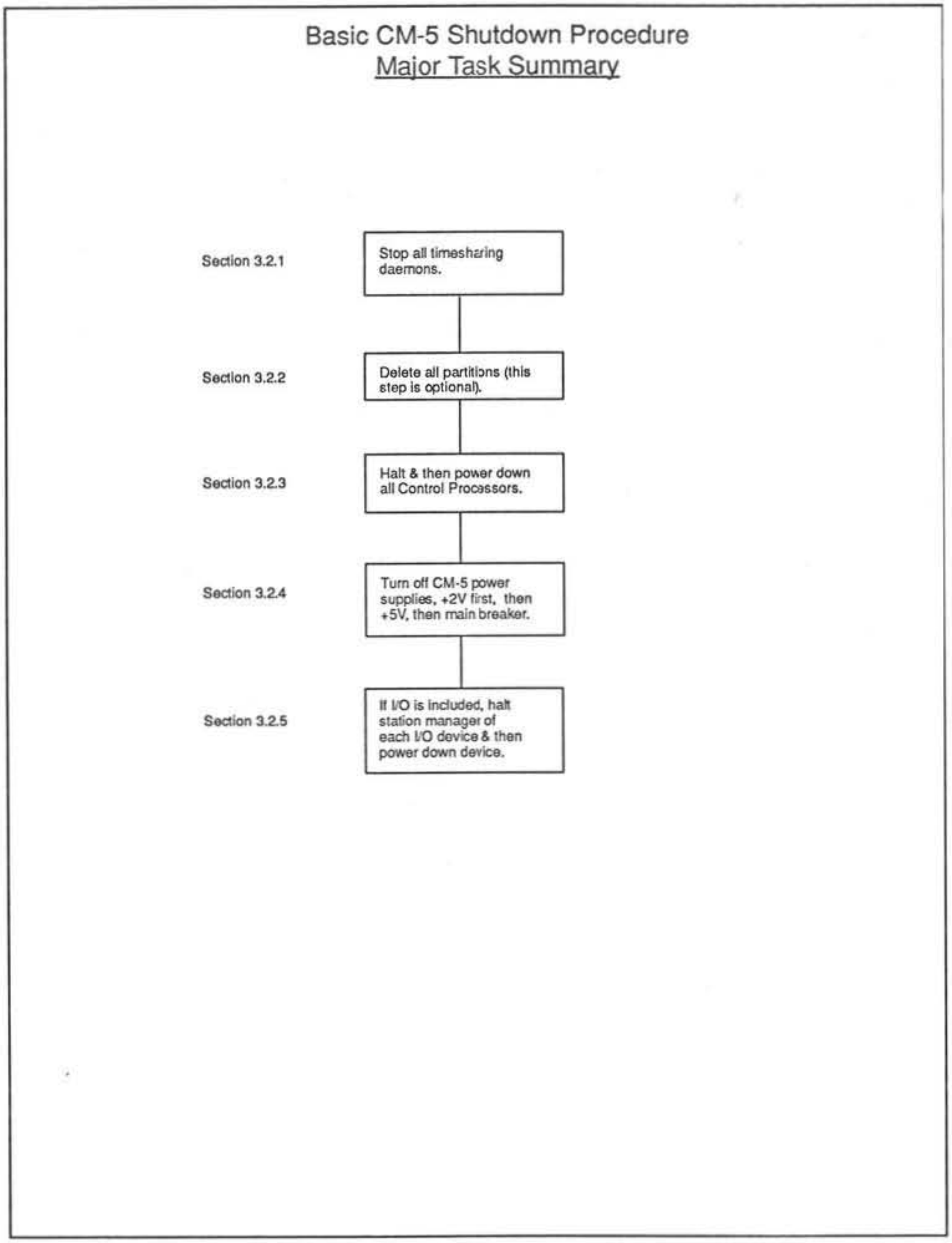


Figure 6. CM-5 shutdown procedure.

CM-5 Startup Procedure — Quick Reference

1. Boot up each external Control Processor.
2. If the system includes I/O, power up all I/O devices.
3. Power up the CM-5 cabinet(s). In multiple-cabinet systems, power up the network cabinet first.
4. Log in to the system administration console as root.
5. If the system's hardware configuration has been changed since the last boot session, update **/etc/cm/configuration/hardware.install** to reflect the changes.
6. If the system's I/O configuration has been been modified since the system was last booted, update **/etc/io.conf** to reflect the changes.
7. Set **CMDIAG_PATH** to specify the diagnostic library pathname and **JTAG_SERVER** to point to the diagnostic server. Set these environment variables on all Control Processors. The default **CMDIAG_PATH** is **/usr/diag/cmdiag**.

   ```
   # setenv DIAG_PATH /usr/diag/cmdiag
   # setenv JTAG_SERVER diag_server_hostname
   ```

8. Create the desired partitions. This and subsequent steps may be implemented by a script. If not, run **cmpartition create** for each partition. For example:

   ```
   # /usr/etc/cmpartition create -pm homer -pn_range 0-63
   # /usr/etc/cmpartition create -pm milton -pn_range 64-127
   ```

9. Run **cmreset** to reset the system hardware and **cmreset -s** on each partition manager to reset the partition manager's interface module.
10. If the CM includes I/O, initialize each IOBA by running **io_cold_boot**.

    ```
    # /usr/etc/io_cold_boot
    ```

11. Start each partition by running a separate **cmpartition start** on the associated partition managers.

    ```
    # /usr/etc/cmpartition start -cmd ts-daemon
    # /usr/etc/cmpartition start -cmd ts-daemon
    ```

Figure 7. CM-5 startup procedure.

CM-5 Shutdown Procedure — Quick Reference

1. Stop timesharing daemons on all partitions. Log in as root to each Partition Manager and run **cmpartition stop**.

   ```
   # /usr/etc/cmpartition stop
   ```

2. Delete all partitions. This can be done from the system administration console.

   ```
   # /usr/etc/cmpartition delete -pm homer
   # /usr/etc/cmpartition delete -pm milton
   ```

3. Halt and then power down all Control Processors.
4. Turn off CM-5 power supplies.
5. If I/O is included, halt the station manager of each I/O device and power down the device.

Figure 8. CM-5 shutdown procedure.

4.1 System Startup

The startup procedure is organized into 11 steps. These steps are summarized in Figure 7 for quick reference. Background details for the various steps are presented in the balance of this section.

4.1.1 Boot External Control Processors

Power up any external Control Processors and verify that their boot sequence is successful. The location of the power switch will depend on which Sun model is used to implement the Control Processor. If you have any questions about this step, refer to the applicable Sun documentation.

4.1.2 Power Up All Peripherals

If the CM-5 system includes peripheral devices, such as DataVault, CM-HIPPI, CM-IOPG, and/or other VMEIO devices, apply power to their power supplies and boot up their station managers.

4.1.3 Power Up the CM

Each cabinet in the CM-5 system is equipped with its own set of power switches.

On device cabinets, these switches are located behind the louvered corner panel that covers the cabinet's power supplies. See Figure 9. The panel is held closed by magnetic latches along the main face of the cabinet and is hinged on the cabinet's end wall. To open the panel, briefly press against the latched face and then release; the panel should swing out away from the cabinet, exposing the power supply bay. Again see Figure 9.

Network cabinets have their circuit breakers on the opposite side of the cabinet, as shown in Figure 10. To reach these switches, slide the covering panel to the right.

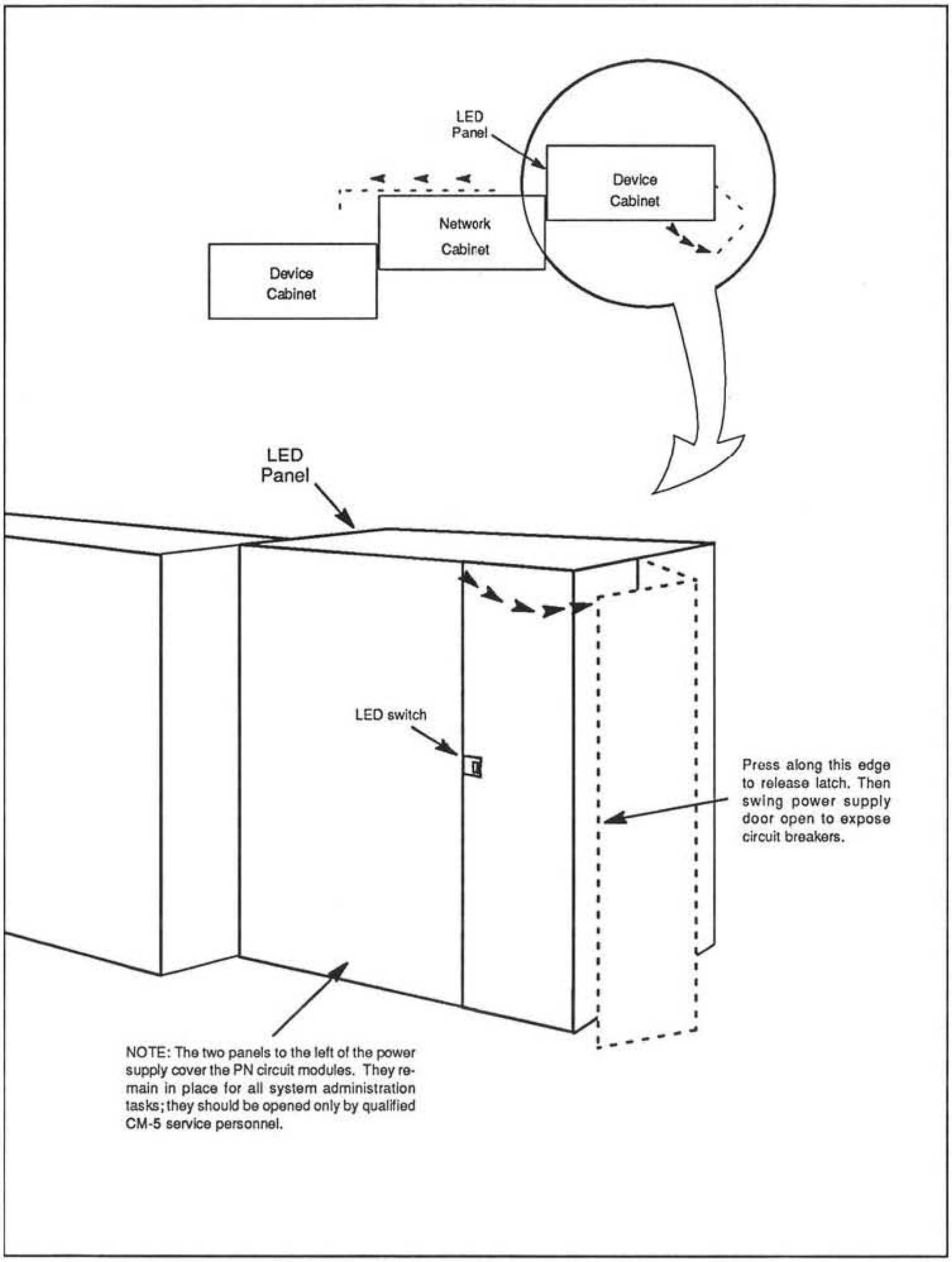


Figure 9. Access to device cabinet circuit breakers.

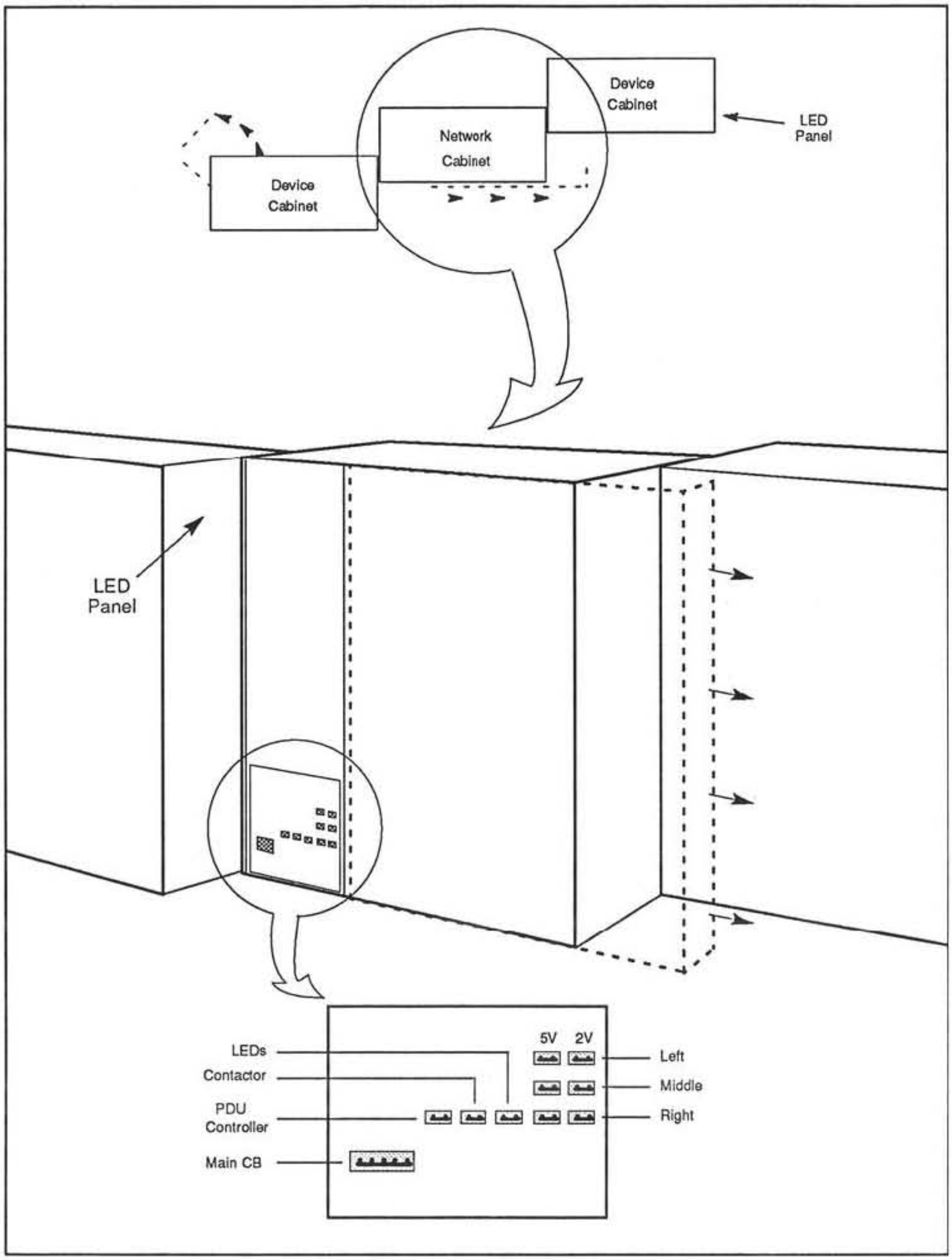


Figure 10. Access to network cabinet circuit breakers.

NOTE: In systems with multiple cabinets, the network cabinet contains a master clock, to which all other clocks are referenced. In such systems, power up the network cabinet first, to permit the master clock time to stabilize before the other cabinets come on line.

Turn on network cabinet power in the following sequence.

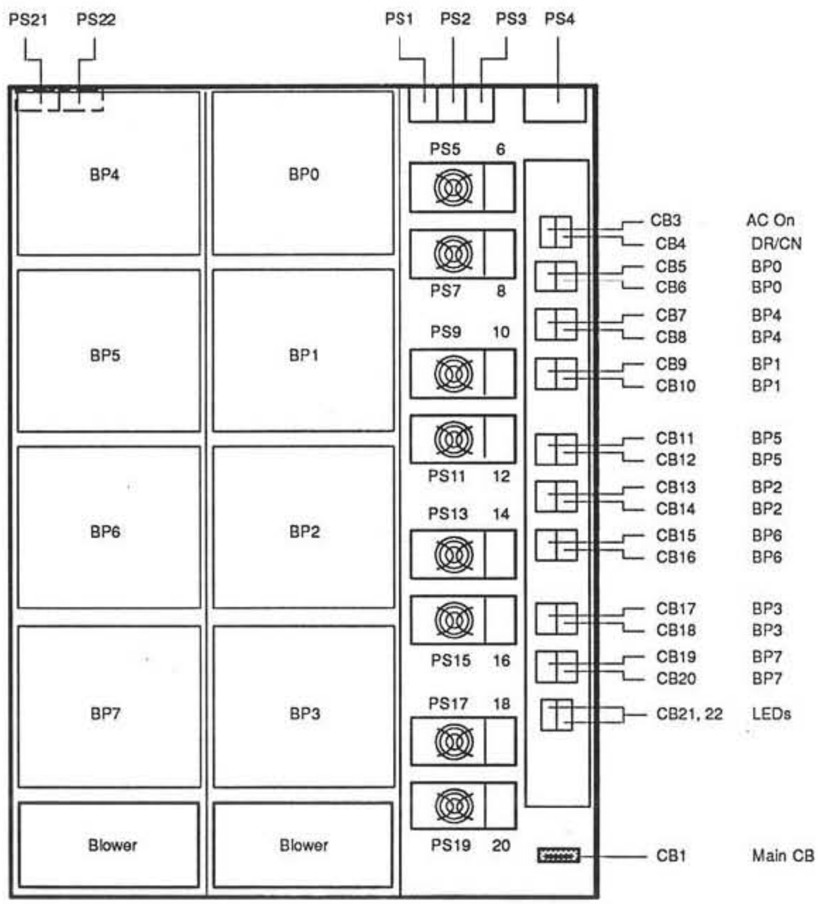
- Main circuit breaker
- PDU (Power Distribution Unit) controller
- Contactor
- LEDs
- +5 V supplies (any order)
- +2 V supplies (any order)

Apply power to the device cabinet power supplies in the following sequence. Figure 11 shows the locations of the referenced circuit breakers. Repeat this sequence for all device cabinets in the system.

- Main circuit breaker — CB1
- AC and DR/CN — CB3 and CB4
- +5 V supplies — CB5, CB7, CB9, CB11, CB13, CB15, CB17, and CB19
- +2 V supplies — CB6, CB8, CB10, CB12, CB14, CB16, CB18, and CB20
- LEDs — CB21 and CB22

When the processing nodes power up, their boot-mode sequence is indicated on the LED panel by a left-to-right "chase pattern." After applying power to the CM cabinets, check their LED panels to verify that they have booted successfully. If this pattern is not displayed, check to see that the cabinet's LED switch is in the "fast copy" mode. Table 1 summarizes the various LED mode settings for this switch. See Figure 9 for the location of the LED switch.

View from Louvered Panel Side



NOTE: BP0—BP7 are shown to clarify the physical relationship of the power supplies to the circuit modules they service. There is no need to remove the panel that covers these backplanes for any system administration tasks.

BP = Backplane
CB = Circuit Breaker
PS = Power Supply

Figure 11. Device cabinet circuit breaker locations.

Table 1. LED Mode Switch Settings.

HEX VALUE	MODE	DEFINITION
0*	Freeze LEDs	Maintains current state of LEDs.
1	Fast Copy (default)	Copies PN values to LEDs; special boot-mode sequence generates left-to-right chase pattern.
3	Interleaved Copy	Displays PN values on LEDs; groups of four even-numbered rows shift their display to the left and odd-numbered groups of four rows shift to the right.
5	Random	Generates random patterns.
7	Interleaved Random	Same shifting behavior as mode 3 but with values supplied by random number generator.
9	LEDs On	Turns all LEDs on.
A	LEDs Off	Turns all LEDs off.
B	Blink	Alternately turns all LEDs on and off; on for 0.5 second and off for 0.5 second.
C	Test	Continuously runs powerup self tests; these tests are described in the CM-5 Field Service Guide.
D	Display PM Loop	Used in hardware diagnostic sessions to trace connectivity problems.

* Switch settings 0, 2, 4, 6, 8, E, and F all specify Freeze mode.

4.1.4 Log in to the System Administration Console

Log in to the System Administration Console as root.

4.1.5 Update hardware.install if Hardware Has Changed

If any CM-5 hardware components have been installed, removed, or repositioned since the last time the system was powered up, update `/etc/cm/configuration/hardware.install` to reflect the changes. One-to-one hardware replacements do not require editing the file since the net change is zero.

`hardware.install` is created at the factory to define the specific hardware configuration of a given system. So long as the system's hardware configuration remains unchanged, this file will require no attention.

Appendix A describes the `hardware.install` contents. If you are still uncertain about how to edit this file, please contact your Thinking Machines Corporation representative for guidance.

4.1.6 Update io.conf if I/O Bus Configuration Has Changed

If the system's I/O bus configuration has changed since the system was last powered up, edit `/etc/io.conf` to incorporate those changes. If the change also involves adding, removing, or relocating any IOBA hardware, you will need to edit `hardware.install` as well.

`io.conf` defines the bus attributes, such as station ID and bus arbiter status, of the I/O devices connected to the E-CMIO bus.

Appendix B describes `io.conf`. If you are uncertain about how to edit this file, please contact your Thinking Machines Corporation representative for guidance.

4.1.7 Set Environment Variables

Two environment variables, `CMDIAG_PATH` and `JTAG_SERVER`, must be set correctly to ensure reliable partitioning behavior.

`CMDIAG_PATH` *`pathname`* This variable identifies the home directory of the JTAG library, which contains information needed by the patitioning software. The factory-set default is **`/usr/diag/cmdiag`**.

`JTAG_SERVER` *`hostname`* This variable specifies the Control Processor on which the JTAG server is running. *`hostname`* must be the name of the System Administration Console.

4.1.8 Check the Current Partitioning State

If you are at all uncertain about the current availability of Processing Nodes and Control Processors, use the **`cmpartition list -l`** command to display this information. This step is entirely optional.

4.1.9 Bringing up a System — Example

Figure 12 provides a sample listing showing the steps involved in creating and activating partitions in a newly powered up system. The example shown in Figure 12 represents a system named calliope with 128 Processing Nodes. The system contains two Control Processors named homer and milton; homer is the System Administration Console.

The first page of Figure 12 contains a summary of the various steps in the sequence organized into seven sections.

Listing Section	Summary Description
1	The environment variables `CMDIAG_PATH` and `JTAG_SERVER` must be set to the appropriate values. The `setenv` commands are being run from homer, the System Administration Console.
2	`cmpartition list -l` shows the current state of partitioning in the CM. In this example, the system calliope has 128 PNs and two partition managers, homer and milton. There are no partitions in the current configuration, leaving all PNs available for new partitions.
3	Next, two partitions are created, each containing 64 PNs.
4	`cmpartition list -l` is run again to verify that the desired partitions were successfully created.
5	Before the timesharing daemon can be started, the system hardware must be reset. In addition, the interface board in the partition manager must be reset; this is done by the `-s` flag.
6	The timesharing daemon is started on the partition managed by homer. The `-K` argument to `ts-daemon` tells which file contains the OS kernel that is to be downloaded. In this case, the default file is `kernel.hw`.
7	Again, `cmpartition list -l` is run to verify that the timesharing daemon is running. The response shows the partition managed by homer is ACTIVE while the other partition is still only ALLOCATED.

Figure 12. Bringing up a system — listing example (page 1 of 3).

(login portion of listing has been omitted)

```
1   homer# setenv CMDIAG_PATH /usr/diag/cmdiag
    homer# setenv JTAG_SERVER homer

2   homer# cmpartition list -l
    CM System "calliope"
         128 Processors [ 16 Mbytes memory, SPARC IU, SPARC FPU ]
         2 Partition Managers
            homer.think.com
            milton.think.com
    Available PN Ranges:
     0-127

    IOP Addresses
     131
     195

3   homer# cmpartition create -pm homer -pn_range 0-63 -iop 131
    homer# cmpartition create -pm milton -pn_range 64-127 -iop 195

4   homer# cmpartition list -l
    CM System "calliope"
         128 Processors [ 16 Mbytes memory, SPARC IU, SPARC FPU ]
         2 Partition Managers
            homer.think.com
            milton.think.com
    Available PN Ranges:
     All PNs in use

    IOP Addresses
     131
     195

    Name    Partition Manager   Size   State       Nodes     Description
    homer   homer.think.com     64     ALLOCATED   0-63
    milton  milton.think.com    64     ALLOCATED   64-127
                                                   131-131
                                                   195-195

5   homer# cmreset
    homer# cmreset -s

6   homer# cmpartition start -cmd ts-daemon -K kernel.hw
```

(continued on next page)

Figure 12. Bringing up a system — listing example (page 2 of 3).

(continued from previous page)

7

```
homer#
homer# cmpartition list -l
CM System "calliope"
     128 Processors [ 16 Mbytes memory, SPARC IU, SPARC FPU ]
     2 Partition Managers
        homer.think.com
        milton.think.com
Available PN Ranges:
 All PNs in use

IOP Addresses
 No IOP on This Machine

Name    Partition Manager   Size   State       Nodes     Description
homer   homer.think.com     64     ACTIVE      0-63
milton  milton.think.com    64     ALLOCATED   64-127
```

8

```
homer# rlogin -l root milton
Password: root_password
milt# setenv CMDIAG_PATH /usr/diag/cmdiag
milt# setenv JTAG_SERVER homer
milt# cmreset -s
milt# /usr/etc/cmpartition start -cmd ts-daemon
milt# exit
homer#
```

Figure 12. Bringing up a system — listing example (page 3 of 3).

4.1.10 Create User Partitions

Create partitions with the **cmpartition create** command. Run this command on the System Administration Console as shown in Figure 12.

To associate an IOBA with a partition, include the **-iop** option in the **create** argument list. The following example creates two 64-PN partitions, **homer** and **milton**, and associates an IOBA with each. One IOBA is at network address 131 and the other is at address 195. In this example, **homer** serves as both system administration console and partition manager.

```
homer# cmpartition create -pm homer -pn_range 0-63 -iop 131
homer# cmpartition create -pm milton -pn_range 64-127 -iop 195
```

4.1.11 Reset the CM and Individual PM Network Interfaces

Run **cmreset** on the system administration console to execute a system-wide hardware reset. Then reset each partition manager's network interface by running **cmreset -s** on each partition manager. The following example shows two partition managers, **homer** and **milton**; **homer** also serves as the system administration console.

```
homer# /usr/diag/cmreset
homer# /usr/diag/cmreset -s
homer# rlogin -l root milton
Password: root_password
milton# /usr/diag/cmreset -s
```

cmreset is required after each power up cycle to synchronize system clocks and initialize all registers and switches to a known state. **cmreset -s** resets the network interface of the partition manager on which it is executed. This reset must be performed separately on each partition manager. For example, if your system has a partition manager,

4.1.12 Initialize the I/O System

If the CM includes I/O devices, coldboot the CM-5 IOBA (Input/Output Bus Adapter) hardware. An IOBA is a set of circuit modules within the CM-5 that together form the interface to a CMIO bus. If the CM-5 has multiple CMIO buses, each is connected to a separate IOBA.

The command for coldbooting IOBAs is **io_cold_boot**, which downloads the I/O kernel to a processor in the IOBA. This must be done any time **cmreset** is executed.

NOTE: **io_cold_boot** depends on configuration information that is generated by **ts-daemon**. Consequently, the Control Processor on which **io_cold_boot** is executed must have run the timesharing daemon at least once before the cold-booting operation can be performed. **ts-daemon** need not be running, however, at the time **io_cold_boot** is invoked.

The I/O configuration file,**io.conf**, provides **io_cold_boot** with the address information needed to download the kernel to each IOBA in the system.

The syntax for using **io_cold_boot** is as follows.

```
/usr/etc/io_cold_boot
```

NOTE: **io_cold_boot** expects three files to reside in the following locations.

```
/etc/io.conf
/usr/etc/io_kernel.hw
/usr/etc/io_download
```

If these files are stored elsewhere, the **-I**, **-K**, and **-B** switches (respectively) must be given to **io_cold_boot** to point the program to the correct files.

4.1.13 Activate Partitions

Once a partition has been created and the resets described in Section 4.1.11 have been performed, the partition is ready to be activated. To activate a partition, run the **cmpartition start** command on the Partition Manager associated with that partition.

NOTE

cmpartition start and **cmpartition stop** must be executed on the Control Processor that is assigned to manage the partition being started or stopped.

When `cmpartition start` has completed, the partition is ready for use.

4.2 System Shutdown

The shutdown procedure is organized into 5 steps. These steps are summarized in Figure 8 for quick reference. Background details for the various steps are presented in the balance of this section.

4.2.1 Stop All Timesharing Daemons

Execute **cmpartition stop** on each partition to halt the timesharing daemon and put the partition into inactive status. The **cmpartition stop** subcommand must be executed separately on each partition manager for every partition you wish to deactivate. The **stop** subcommand syntax is:

```
cmpartition stop
```

4.2.2 Delete All Partitions

This step is optional. After stopping a partition, you may deallocate that partition with the command **cmpartition delete**. If you do not do this, the partition is automatically reallocated upon restart of the system. The **cmpartition delete** subcommand syntax is:

cmpartition delete {[**-pm** *hostname*] | [**-name** *partitionname*]}

-pm *hostname* defaults to the hostname of the partition manager on which the **delete** command is executed.

-name *partitionname* associates an optional name of your choice with the partition. This argument can be used instead of **-pm** *hostname* to specify the partition. It has no default value.

4.2.3 Shut Down External Control Processors

Halt and then power down any external Control Processors. The location of the power switch will depend on which Sun model is used to implement the Control Processor. If you have any questions about this step, refer to the applicable Sun documentation.

4.2.4 Power Down the CM-5

Turn off all network cabinet circuit breakers in the following sequence.

- +2 V supplies
- +5 V supplies
- LEDs
- Contactor
- PDU controller
- Main circuit breaker

Turn off all device cabinet power supplies in the following sequence.

- +2 V supplies — CB6, CB8, CB10, CB12, CB14, CB16, CB18, and CB20
- AC and DR/CN — CB3 and CB4
- +5 V supplies — CB5, CB7, CB9, CB11, CB13, CB15, CB17, and CB19
- LEDs — CB21 and CB22
- Main circuit breaker — CB1

4.2.5 Shut Down All Peripherals

If the system includes I/O, halt the station manager of each I/O device and then power down the device.

Chapter 5

CM Error Logging System

Whenever a hardware fault causes the timesharing daemon to exit, the CM error logging system records the event in **/var/log/cm-errors.log**. On these occasions, **ts-daemon** passes information about the fault to the SunOS **syslog** system, which performs the actual logging. **syslog**'s local7 error facility is reserved for these messages.

5.1 Implementing CM Error Logging

Ordinarily, CM-5 systems are shipped from the factory with error logging implemented. If, for any reason, error logging must be enabled in the field, this is done by adding the following lines to the end of **/etc/syslog.conf**

```
# Connection Machine Logging Facility
local7.debug   /var/log/cm-errors.log
```

NOTE: These lines must appear just as shown here, including the comment on the first line.

The first field identifies the error facility and the level of filtering to be applied to the error reporting. In this case, the error facility is called **local7** and the error severity level is **debug**. Selecting **debug** means all errors will be reported.

The second field specifies the file in which **local7** errors will be logged.

5.2 Error Message Description

Error messages consist of a single line containing 9 fields separated by vertical line characters, |. The 9 fields comprising each message are:

```
timestamp host | seconds | general error category |
program name | subprogram name | userid | groupid |
processid | error message
```

timestamp host	Contains a timestamp for the message and the hostname of the CP on which the timesharing daemon is running. This field is always terminated with the term, `syslog:`
seconds	Gives the time in seconds since January 1, 1970.
general error message	Presents a high-level description of the type of error being reported.
program name	Identifies the timesharing daemon as the source of the message.
subprogram name	Identifies the user program that was running when the error was detected.
userid	Identifies the owner of the program that was running when the error was detected.
groupid	Identifies the group associated with the program that was running when the error was detected. This field is not currently implemented. Its place in the message contains the default **`-1`**.
processid	Gives the **`processid`** of the program that was running when the error was detected.
error message	This is a text string that describes the error.

The following sample message illustrates the kind of information to be found in **`cm-errors.log`**.

```
Jul 5 11:39:39 yeats syslog: | 678728379 | Hardware in
error state | Timesharing daemon | /user/prod/filter.a
| 1556(kjr) | -1(<no group>) | 11837 | FatalInterrupt:
time sharing detected error on NI.
```

Appendix A

Investigating ts–daemon Failures with kpndbx

When the timesharing daemon fails, it can be productive to examine the contents of the PN registers, which will often identify which PN(s) caused `ts-daemon` to fail.

This appendix explains how to extract this information with `kpndbx`, an extension of the UNIX debugging facility, `dbx`. The procedure for using `kpndbx` follows.

1. `kpndbx` needs certain hardware configuration information to function, namely the total number of PNs in the system as well as the range of PNs you want it to examine. If you are not certain of these details, use `cmpartition list -l` to display the needed information.

   ```
   % /usr/etc/cmpartition list -l
   ```

   Appendix M contains the `cmpartition` man page.

2. Set the environment variable `PN_KERNEL` to point to the operating system kernel. This usually resides in `/usr/etc/kernel.hw`.

   ```
   % setenv PN_KERNEL /usr/etc/kernel.hw
   ```

3. Set the environment variable `CMDIAG_PATH` to point to the `cmdiag` directory. This resides in `/usr/diag/cmdiag`.

   ```
   % setenv CMDIAG_PATH /usr/diag/cmdiag
   ```

4. Invoke **kpndbx**. When **kpndbx** responds by asking for the partition size, enter the total number of PNs in the system. In the following example, the system has a total of 256 PNs.

```
% /usr/bin/kpndbx
partition size? 256
%
```

5. Use the **set $pnlist** [*m*:*n*] command to tell **kpndbx** which PNs to examine. [*m*:*n*] specifies the physical addresses of the first and last PNs in the range, respectively. In the following example, the range includes 64 PNs, in which the first PN is 128 and the last PN is 191.

```
% set $pnlist [128:191]
```

6. Use the following commands to tell **kpndbx** to display a summary of the PN status.

```
% set $page_size = 0
% pnsummary all
```

7. Examine the resulting summary. An example of this output is shown below with an explanation of its contents following.

```
pn number 24:
  running, pc = 81cc, psr = 110010a2, tbr = f8000180
  CMNA_interrupt_cause = 0
```

The contents of this example are explained below.

- `pn number` is the relative address of the PN within the partition. The physical address of the PN in this example is 152 (128 + 24).
- The first entry of the second line indicates the general state of the PN at the time of failure. It will show either `running` or `error`.
- The rest of the second line displays the contents, in hexadecimal, of the three key registers: pc, psr, and tbr.
- The third line identifies the level of the interrupt that caused the PN to terminate operation. In this case it was interrupt level 0.

8. Keep the following in mind as you evaluate the set of register states that **kpndbx** displays across the range of PNs.

- Ordinarily, the PN that caused **`ts-daemon`** to fail will be in the error state (`error` will be displayed instead of `running`).
- If no PN shows the error state, check the `tbr` values of all PNs. If all PNs except one have the same `tbr` value, the exception is probably the failing PN.
- If all PN have the same `tbr` value, check the `pc` and `psr` values. Usually, the `pc` values will be different, but close, across the range of PNs. The `psr` values will mostly be the same across the PN range, with two or three different from the rest. Look for a PN whose `pc` and `psr` values are significantly different; it is likely to be the cause of the failure.
- If **`kpndbx`** does not yield any of these clues, generate more error information with **`cmdiag`**. Appendix B explains how to use **`cmdiag`** for this purpose.

Appendix B

Generating Error Information

This appendix presents the most general procedure for troubleshooting CM-5 hardware problems. Follow this procedure when you have too little information to focus attention on a particular area of the hardware. It is likely to generate useful error messages no matter which part in the CM-5 is failing.

The procedure is presented in two versions. One version is designed for troubleshooting hardware within a partition without interfering with user applications running on other partitions. This procedure is described in B.1. If the tests prescribed in B.1 are not sufficiently exhaustive, follow the full-system procedure contained in B.2. This procedure requires access to the complete system; no timesharing daemons can be running.

Figure 13 provides a summary of the partition-contained procedure in quick-reference format. Figure 14 provides the same quick-reference information for the full-system procedure.

Partition-Contained Diagnostics

Introductory Notes

The system used to illustrate this procedure example has the following attributes.

- System is named Calliope and has 256 PNs.
- Diagnostic console is named **homer.think.com.**
- Calliope has two 128-PN partitions, which are allocated to partition managers named **virgil.think.com** and **milton.think.com.**
- Diagnostics will be run on **virgil.think.com** partition manager.

1

```
login: user_id
% su
password: root_password
SU homer.think.com /dev/console
hom# cd /usr/diag/cmdiag
```

2

```
hom# setenv CMDIAG_PATH /usr/diag/cmdiag
hom# setenv JTAG_SERVER homer.think.com
```

3

```
hom# /usr/etc/cmpartition list -l
  CM System "Calliope"
  256 Processors [ 8 Mbytes memory, SPARC IU, SPARC FPU ]
   2 Partition Managers
  virgil.think.com
  milton.think.com
Available PN Ranges:
  All PNs in use
IOP Addresses
  480

Name  Partition Manager  Size  State    Nodes     Description
V128  virgil.think.com   128   ACTIVE   0-127     virgil
M128  milton.think.com   128   ACTIVE   128-255   milton
                                        480-480
```

4

```
hom# rlogin -l root virgil.think.com
password: QuiVive
virg# cmpartition stop
```

(continued on next page)

Figure 13. Generating diagnostic information on a partition — 1 of 2

Partition-Contained Diagnostics

(continued from previous page)

5
```
virg# cmpartition list -l
  CM System "Calliope"
  256 Processors [ 8 Mbytes memory, SPARC IU, SPARC FPU ]
   2 Partition Managers
  virgil.think.com
  milton.think.com
Available PN Ranges:
  All PNs in use
IOP Addresses
  480

Name   Partition Manager  Size  State      Nodes     Description
V128   virgil.think.com   128   ALLOCATED  0-127     virgil
M128   milton.think.com   128   ACTIVE     128-255   milton
                                           480-480
```

6
```
virg# cmdiag -p virgil
<CM-DIAG> rgroups m PE global broadcast combine dr partition
             .
             .                 ; diagnostic test report
             .
```

7, 8
```
<CM-DIAG> find-cm-error
             .
             .                 ; error system report
             .
```

Figure 13. Generating diagnostic information on a partition — 2 of 2.

System-Wide Diagnostics

Introductory Notes

The system used to illustrate this procedure example has the following attributes.

- System is named Calliope and has 256 PNs.
- Diagnostic console is named **homer.think.com**
- Calliope has two 128-PN partitions, which are allocated to partition managers named **virgil.think.com** and **milton.think.com.**
- Diagnostics will be run on **homer.think.com** partition.

1
```
login: user_id
% su
password: root_password
SU homer.think.com /dev/console
hom# cd /usr/diag/cmdiag
```

2
```
hom# setenv CMDIAG_PATH /usr/diag/cmdiag
hom# setenv JTAG_SERVER homer.think.com
```

3
```
hom# /usr/etc/cmpartition list -l
  CM System "Calliope"
  256 Processors [ 8 Mbytes memory, SPARC IU, SPARC FPU ]
   2 Partition Managers
  virgil.think.com
  milton.think.com
Available PN Ranges:
  All PNs in use
IOP Addresses
  480

Name   Partition Manager  Size  State      Nodes     Description
V128   virgil.think.com   128   ACTIVE     0-127     virgil
M128   milton.think.com   128   ALLOCATED  128-255   milton
                                           480-480
```

4
```
hom# rlogin -l root virgil.think.com
password: QuiVive
virg# cmpartition stop
virg# cmpartition delete
virg# exit
hom# rlogin -l milton.think.com
password: QuiVaLa
milt# cmpartition delete
milt# exit
```

(continued on next page)

Figure 14. Generating diagnostic information on the full system — 1 of 2

System-Wide Diagnostics

(continued from previous page)

5

```
hom# cmpartition list -l
  CM System "Calliope"
  256 Processors [ 8 Mbytes memory, SPARC IU, SPARC FPU ]
   2 Partition Managers
  virgil.think.com
  milton.think.com
Available PN Ranges:
  All PNs in use
IOP Addresses
  480

Name    Partition Manager  Size  State       Nodes       Description
V128    virgil.think.com   128     0-127       virgil
M128    milton.think.com   128     128-255     milton
                                            480-480
```

6, 7

```
hom# ./cmdiag -C
<CM-DIAG> rgroups m SVME
<CM-DIAG> rgroups m CLKDN
<CM-DIAG> rgroups m CLKBUF
<CM-DIAG> rgroups m SPI
<CM-DIAG> rgroups m FILLER
<CM-DIAG> rgroups m PE PEMEM
<CM-DIAG> rgroups m CN
<CM-DIAG> rgroups m DR
          .
          .                   ; diagnostic test report
          .
```

8

```
<CM-DIAG> q
hom# /usr/etc/cmpartition create -pn_range 0-255
```

9

```
hom# cmreset
hom# cmreset -s
hom# ./cmdiag -C -p homer.think.com
<CM-DIAG> rgroups m PN dr combine global broadcast partition
```

10

```
<CM-DIAG> find-cm-error
```

Figure 14. Generating diagnostic information on the full system — 2 of 2.

B.1 Running `cmdiag` within a Partition

1. Log in to the diagnostic console as root and change directory to **/usr/diag/cmdiag**.

```
login: user_id
% su
password: root_password
SU console_name /dev/console
# cd /usr/diag/cmdiag
```

2. Set the **CMDIAG_PATH** and **JTAG_SERVER** environment variables. The default **CMDIAG_PATH** is **/usr/diag/cmdiag**. Set the **JTAG_SERVER** variable to the diagnostic server hostname.

```
# setenv CMDIAG_PATH /usr/diag/cmdiag
# setenv JTAG_SERVER diag_server_hostname
```

3. Run **cmpartition list -l** to be certain you have an accurate understanding of the current partitioning status of the CM — what partition configurations are in effect, their names, the hostnames of their partition managers, and their state of use.

```
# /usr/etc/cmpartition list -l
```

4. If **cmpartition list -l** reports the state of the target partition as ACTIVE, it means **ts-daemon** is running on that partition. If so, **rlogin** to the appropriate partition manager and run **cmpartition stop** to halt the timesharing daemon. Then exit.

```
# rlogin -l root pm_name
password: root_password
# /usr/etc/cmpartition stop
# exit
```

5. Run **cmpartition list -l** again. The target partition should now show an ALLOCATED status. This means the partition is defined and still associated with its partition manager, but the timesharing daemon is not running.

6. Run the manufacturing version of the processor chip tests, followed by the Data Network and Contol Network verifiers. Use the **-p** *pm_name* option to restrict these tests to the desired partition. Appendix M explains **cmdiag** options in full.

```
# ./cmdiag -p pm_name
<CM-DIAG> rgroups m PE global broadcast combine dr
        partition
```

NOTE: When running **`cmdiag`** as root, you must explicitly specify the local cmdiag search path (precede **`cmdiag`** command with **`./`**).

7. Analyze any error messages and, if the failure source is obvious, take appropriate corrective action. If additional diagnostic information is needed, go to step 8.

8. Run **`find-cm-error`**; the error system utility will report on any hardware failures it finds.

```
<CM-DIAG> find-cm-error
```

9. Analyze the **`find-cm-error`** output. The next step will depend on the nature of the error messages reported in steps 7 and 8. In most cases, you will proceed along one of the following lines.

   - If a single component is identified as failing (most likely at the leaf node level), simply replace the field-replaceable unit on which it resides. Appendix C discusses this path in more detail.

   - If a Control Network failure is reported, the source of the failure may be ambiguous. Appendix D explains how to parse Control Network error reports to isolate the fault to a single component or interconnect path.

   - Data Network error reports are less ambiguous than Control Network error messages, but do need special analysis. Appendix E explains how to troubleshoot Data Network failures.

   - If the failure symptoms indicate an I/O-related failure, use the **`cmdiag`** I/O tests that can be run within a partition to evaluate the I/O tests hardware more closely. Appendix F identifies which I/O tests are partition-contained. Appendix H describes the procedure for using these tests.

B.2 Running `cmdiag` on the Full System

1. Log in to the diagnostic console as root and change directory to **`/usr/diag/cmdiag`**.

```
login: user_id
% su
```

```
password: root_password
# console_name /dev/console
# cd /usr/diag/cmdiag
```

2. Set the **CMDIAG_PATH** and **JTAG_SERVER** environment variables. The default **CMDIAG_PATH** is **/usr/diag/cmdiag**. Set the **JTAG_SERVER** variable to the diagnostic server hostname.

```
# setenv CMDIAG_PATH /usr/diag/cmdiag
# setenv JTAG_SERVER diag_server_hostname
```

NOTE: No user activity will be possible beginning with the next step.

3. Stop and delete all partitions. To do this, you need to know the hostname of each partition manager to which a partition is allocated. If necessary, run **cmpartition list -l** to get this information.

```
# /usr/etc/cmpartition list -l
```

4. Then run **cmpartition stop** and **cmpartition delete** on every partition manager that has an ACTIVE partition. Run **cmpartition delete** on every partition manager that has an ALLOCATED partition.

   For example, if **cmpartition list** shows **virgil.think.com** as ACTIVE and **milton.think.com** as ALLOCATED, do the following.

```
# rlogin -l root virgil.think.com
password: root_password
virg# /usr/etc/cmpartition stop
virg# /usr/etc/cmpartition delete
virg# exit
# rlogin -l root milton.think.com
password: root_password
milt# /usr/etc/cmpartition delete
milt# exit
#
```

5. Run **cmpartition list -l** again. It should report no partitions either ACTIVE or ALLOCATED.

6. Run the manufacturing level of the JTAG test group.

```
# ./cmdiag -C
<CM-DIAG> rgroups m SNI
<CM-DIAG> rgroups m CLKDN
<CM-DIAG> rgroups m CLKBUF
<CM-DIAG> rgroups m SPI
<CM-DIAG> rgroups m FILLER
<CM-DIAG> rgroups m PE PEMEM
```

```
<CM-DIAG> rgroups m CN
<CM-DIAG> rgroups m DR
```

NOTE: When running **cmdiag** as root, you must explicitly specify the local **cmdiag** search path (precede **cmdiag** command with **./**).

7. Analyze any error messages and, if the failure source is obvious, take appropriate corrective action. See step 11 for guidance.

   If this step does not provide sufficient information, go to step 8.

8. Create a partition that encompasses all PNs in the system. Enter the lowest and highest PN addresses for *first_pn* and *last_pn*, respectively.

```
<CM-DIAG> q
# /usr/etc/cmpartition create -pn_range first_pn-last_pn
```

9. Execute a system reset and reset the partition manager's interface module. Then run the processor chip tests, followed by the Data Network and Control Network verifiers.

```
# cmreset
# cmreset -s
# ./cmdiag -C -p pm_name
<CM-DIAG> rgroups m PN dr combine global broadcast
          partition
```

10. Analyze any error messages and, if the failure source is obvious, take appropriate corrective action. If additional diagnostic information is needed, run **find-cm-error** again and go on to step 11.

11. Analyze the **find-cm-error** output. The next step will depend on the nature of the error messages reported in step 7. In most cases, you will proceed along one of the following lines.

    - If a single component is identified as failing (most likely at the leaf node level), simply replace the field-replaceable unit on which it resides. Appendix C discusses this path in more detail.

    - If a Control Network failure is reported, the source of the failure may be ambiguous. Appendix D explains how to parse Control Network error reports to isolate the fault to a single component or interconnect path.

- Data Network error reports are less ambiguous than Control Network error messages, but do need special analysis. Appendix E explains how to troubleshoot Data Network failures.
- If the failure symptoms indicate an I/O-related failure, run the I/O tests described in Appendix F to evaluate the I/O hardware more closely. Appendix H describes the procedure for using these tests.

Appendix C

Failures at the Leaf Node Level

C.1 Overview

When error messages from `kpndbx` or `find-cm-error` point to a specific PN, corrective action is straightforward. The procedure in such cases is summarized below.

1. Identify the PN module on which the faulty component resides and determine its location in the system (cabinet, backplane, slot).
2. Replace the unit (see Section C.2 for board replacement procedure).
3. Run the complete diagnostics test suite to verify that the system is now fully functional. Remember that the timesharing daemon must not be running in the partition.

   `# ./cmdiag -m -p` *pm_name*

4. If any errors are reported, go to Appendix B for further guidance. If no errors are reported, the system can be returned to regular service.

C.2 PN Board Replacement Procedure

(to be supplied)

Appendix D

Tracing Control Network Errors

D.1 Introduction

When a component or interconnect path in the Control Network fails, the failure usually propagates through the network. As a result, `cmdiag` reports that many CN nodes have failed—every node containing an erroneous checksum value.

This appendix explains how to analyze **`find-cm-error`** reports to quickly narrow the search to no more than two nodes and their interconnecting signal path. Then, corrective action simply becomes a process of elimination among those three candidates.

Isolating faults in the Control Network depends on having a clear understanding of how errors propagate through the network. The following concepts are key to this understanding.

- **Similar to message broadcasting**—Error status messages propagate through the Control Network in a manner analogous to standard message broadcasting. An important exception to this analogy is, however, that error messages begin at the point where the error is detected. Unlike ordinary broadcasts, which always begin at a leaf node, error messages can originate at any level in the network.

- **Up errors and down errors**—When an error is detected before it reaches the partition's root node, it is flagged as an up error. The CN node that detects the error generates an error status and forwards it upward to the root. The root node then broadcasts it downward to all nodes in the partition. In such cases, one or more nodes will report up errors and every node in the partition will report a down error. Figure 15 shows an example of this.

- **Down errors only**—When an error is first detected in a downward path (no up error is detected), the error status is broadcast to all nodes below the node that first detected the error. In this case, no up errors are reported, and the root never sees the error. See Figure 16 for an example.

Keep these concepts in mind as you go through the procedure presented in Section D.2.

D.2 Fault Isolation Procedure

NOTE: For this procedure, you will need readable copies of the CN cable assembly drawings for all levels of the network in your system. You will also need a large surface (eg, table) on which to spread these drawings open.

1. Lay out the CN cable assembly drawings in a location where you can also read the error system report.

2. Examine the `find-cm-error` output, looking for up errors. If there are any up errors, perform the steps labeled **UP ERRORS**, beginning with step 3. If there are no up errors, go to step 13.

3. **UP ERRORS** — On the cable assembly drawings, locate all CN components that show up-error status in the `find-cm-error` report. Ignore components with only down errors.

4. **UP ERRORS** — In the set of components reporting up errors, find the component that is at the lowest point in the network tree. When you identify this component, you will have narrowed the search to the following elements. Figure 15 illustrates this.

   - this component
   - the set of components that are its immediate children
   - the paths that connect the children to this parent

   NOTE: The component's children are implicated because a faulty component will not necessarily generate its own error status. Therefore, the first component to report an error may actually be reflecting an error that originated in one of its children.

   The next step is to narrow the search further.

5. **UP ERRORS** — Analyze the error message for the component selected in step 4. This message will ordinarily tell you which of the internal nodes (nodes 0, 1, and 2) detected the failure. It may also associate the failure with a particular child path connected to the node.

   This time you want to find the lowest level node that is reporting an up error status and, if possible, which child path (left or right) is associated with that error. When you have determined this, you will have narrowed your search to the following elements. Figure 17 illustrates this.

   - this node
   - the child node connected to this node by the path implicated in the report
   - or the interconnecting path

   NOTE: If a particular path is not specified in the error report, you may need to involve both children (left and right) in the process of elimination. This is shown in Figure 18.

6. **UP ERRORS** — Identify the circuit board(s) that contain the parent and child nodes identified in step 5. If these nodes are on different boards, identify the interconnecting cable as well.

   NOTE: If the parent node identified in step 5 is node 1, all suspect elements reside within the same component: the primary node (node 1), both children (nodes 0 and 2), and the interconnecting paths. In this case, the next steps will involve only one circuit board.

7. **UP ERRORS** — Replace the circuit board containing the parent node and rerun the `cmdiag` test that exposed the error..

8. **UP ERRORS** — If the test reports no errors, the board replacement may have corrected the problem. Verify by running `cmdiag -f`.

   If the system passes the complete manufacturing version of `cmdiag`, return the system to regular operation. If errors are reported, go to step 9.

9. **UP ERRORS** — Restore the original board containing the parent node to its slot. Then replace the circuit board containing the implicated child node and rerun the test that reported the error.

   NOTE: If `find-cm-error` does not implicate either child path (see Figure 5), choose one child node to replace first. If the test continues to fail, restore the board just removed, replace the other child node board, and run the test again.

10. **UP ERRORS** — If the test reports no errors, replacing the child node board may have corrected the problem. Verify by running `cmdiag -m`.

    If the system passes the complete manufacturing version of `cmdiag`, return the system to regular operation. If errors are reported, go to step 11.

11. **UP ERRORS** — Examine the connectors on the cable identified in step 5. If you find bent or damaged pins, repair if possible. If repair is not possible, replace the cable. In either case, run the `cmdiag` test that reported the error when you finish working on the cable.

    NOTE: If `find-cm-error` does not implicate either child path (Figure 18), examine both. Then repair/replace them one at a time, running the failing test in between.

12. **UP ERRORS** —If no errors are reported, repairing/replacing the cable may have corrected the problem. Verify by running `cmdiag -f`.

    If the system passes the complete manufacturing version of `cmdiag`, return the system to regular operation. If errors are reported, call Cambridge for assistance.

13. **DOWN ERRORS ONLY** — On the CN cable assembly drawings, locate the components that show down error status in the `find-cm-error` report.

14. **DOWN ERRORS ONLY** — In the set of components identified in step 13, find the component that is at the highest point in the network tree. When you identify this component, you will have narrowed the search to the following elements. Figure 16 illustrates this.

    - this component
    - the set of components that are its immediate parents
    - the paths that connect the parents to this child

    NOTE: The component's parents are implicated because a faulty component will not necessarily generate its own error status. Therefore, the first component to report an error may actually be reflecting an error that originated in one of its parents.

    The next step is to narrow the search further.

15. **DOWN ERRORS ONLY** — Analyze the error message for the component selected in step 14. This message will ordinarily tell you which of the internal nodes (nodes 0, 1, and 2) detected the failure. It may also associate the failure with a particular parent path connected to this node.

This time you want to find the highest level node that is reporting a down error status and, if possible, which parent path (left or right) is associated with that error. When you have determined this, you will have narrowed your search to the following elements. Figure 19 illustrates this.

- this node
- the parent node connected to this node by the path implicated in the report
- or the interconnecting path itself

NOTE: If a particular path is not specified in the error report, you may need to involve both parents (left and right) in the process of elimination. This is shown in Figure 20.

16. **DOWN ERRORS ONLY** — Identify the circuit board(s) that contain the parent and child nodes identified in step 15. If these nodes are on different boards, identify the interconnecting cable as well.

    NOTE: If the child node identified in step 15 is either node 0 or node 2, all suspect elements reside within the same component: the primary node (node 0 or 2), and its parent (node 1), and the interconnecting path. In this case, the next step will involve only one circuit board.

17. **DOWN ERRORS ONLY** — Replace the circuit board containing the child node and rerun the test that detected the error.

18. **DOWN ERRORS ONLY** — If the test reports no errors, replacing the board may have corrected the problem. Verify by running `cmdiag -f`.

    If the system passes the complete manufacturing version of `cmdiag`, return the system to regular operation. If errors are reported, go to step 19.

19. **DOWN ERRORS** — Restore the original board containing the child node to its slot. Then replace the circuit board containing the implicated parent node and rerun the test that reported the error.

    NOTE: If `find-cm-error` does not implicate either parent path (see Figure 20), choose one parent node to replace first. If the test continues to fail, restore the board just removed, replace the other parent node board, and run the test again.

20. **DOWN ERRORS ONLY** — If the test reports no errors, replacing the board may have corrected the problem. Verify by running `cmdiag -f`.

If the system passes the complete manufacturing version of `cmdiag`, return the system to regular operation. If errors are reported, go to step 21.

21. **DOWN ERRORS ONLY** — Examine the connectors on the cable identified in step 14. If you find bent or damaged pins, repair if possible. If repair is not possible, replace the cable. In either case, run the `cmdiag` test that reported the error when you finish working on the cable.

    NOTE: If `find-cm-error` does not implicate either parent path (see Figure 20), examine both. Then repair/replace them one at a time, running the failing test in between.

22. **DOWN ERRORS ONLY** — If no errors are reported, the cable may have been the source of the problem. Verify by running `cmdiag -f`.

    If the system passes the complete manufacturing version of `cmdiag`, return the system to regular operation. If errors are reported, call Cambridge for assistance.

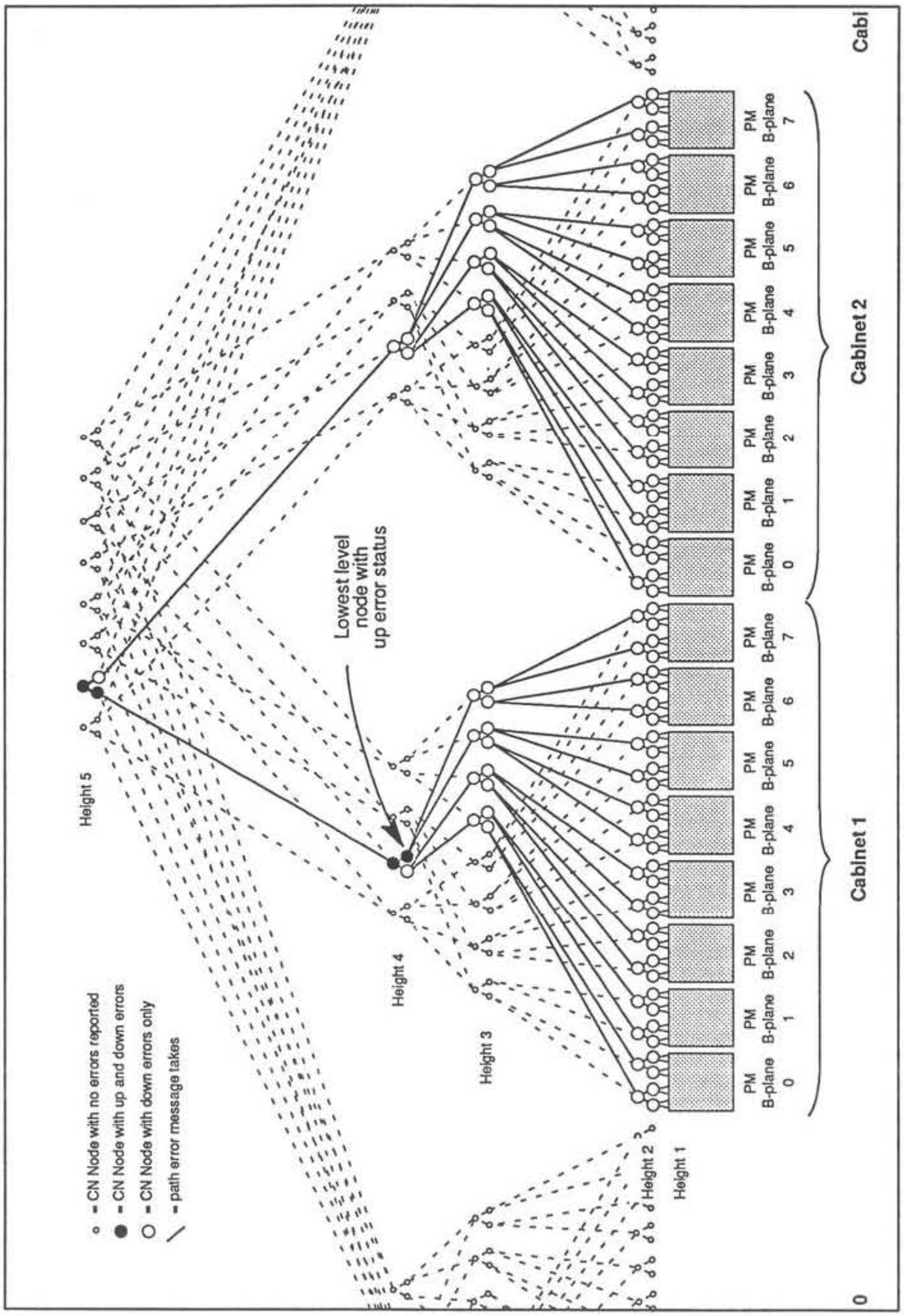


Figure 15. Component-level analysis — up and down errors

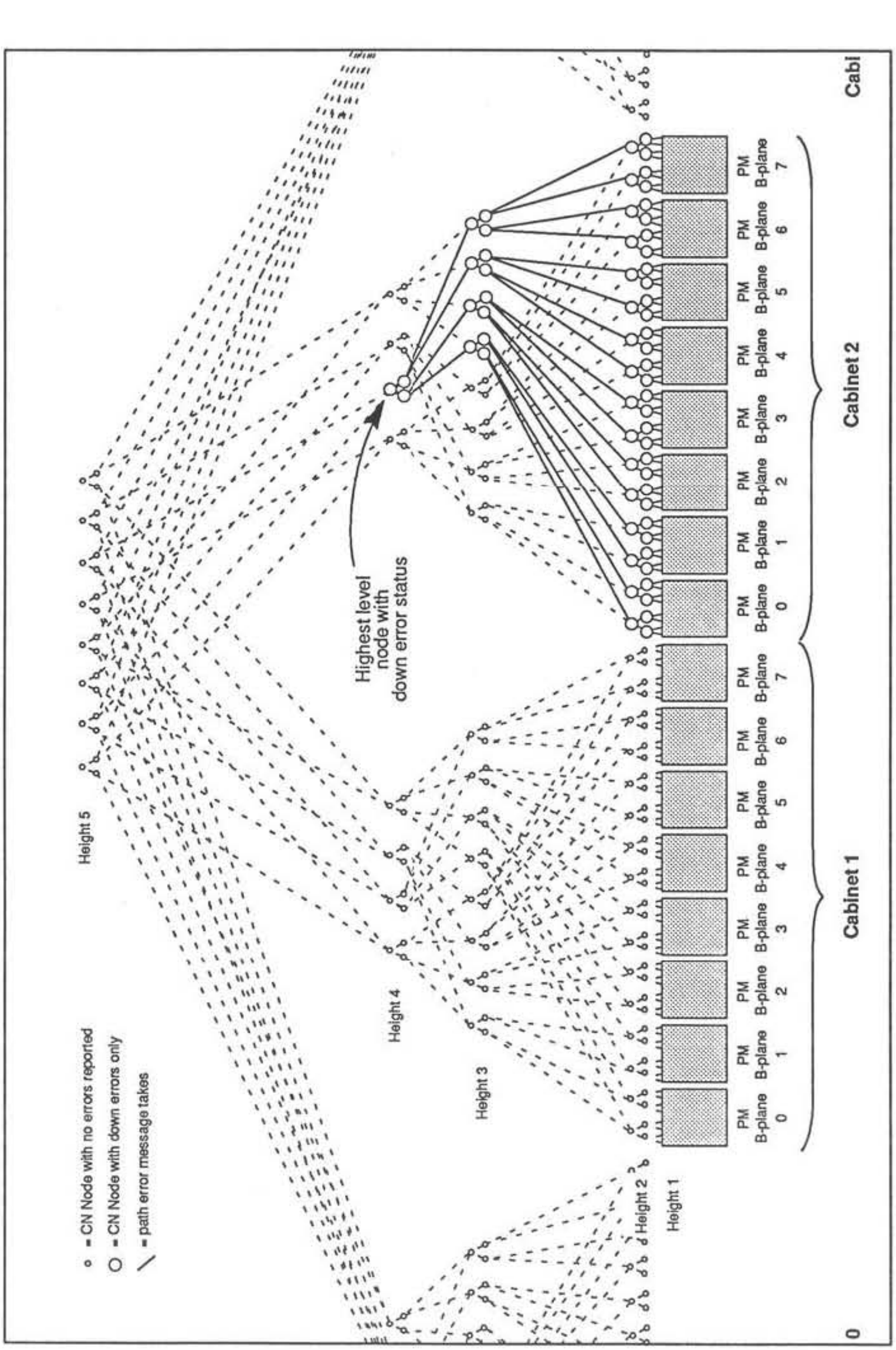


Figure 16. Component-level analysis — down errors only

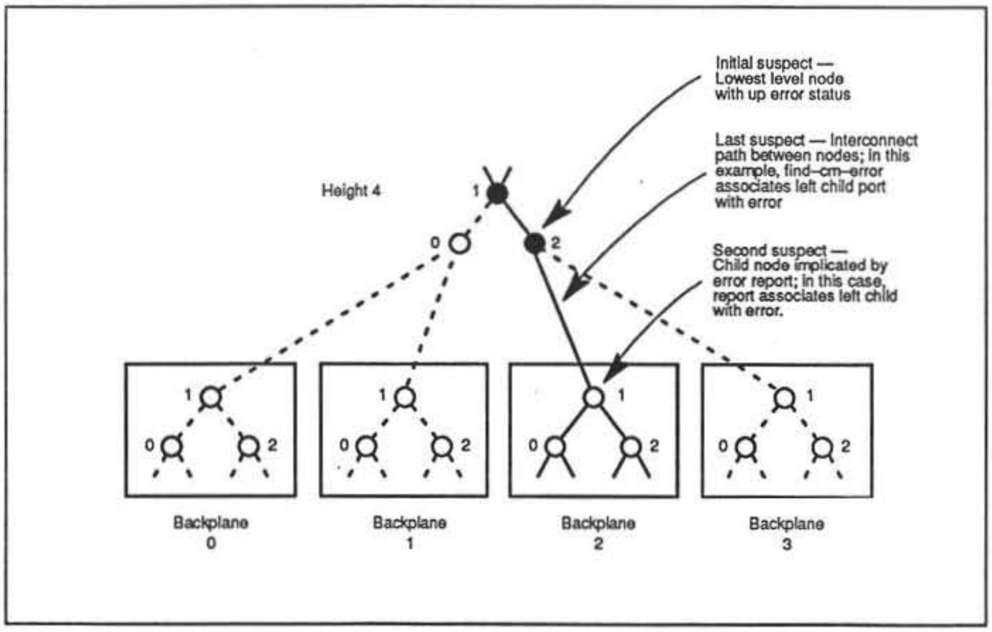


Figure 17. Node-level analysis — up and down errors
(with child path specified)

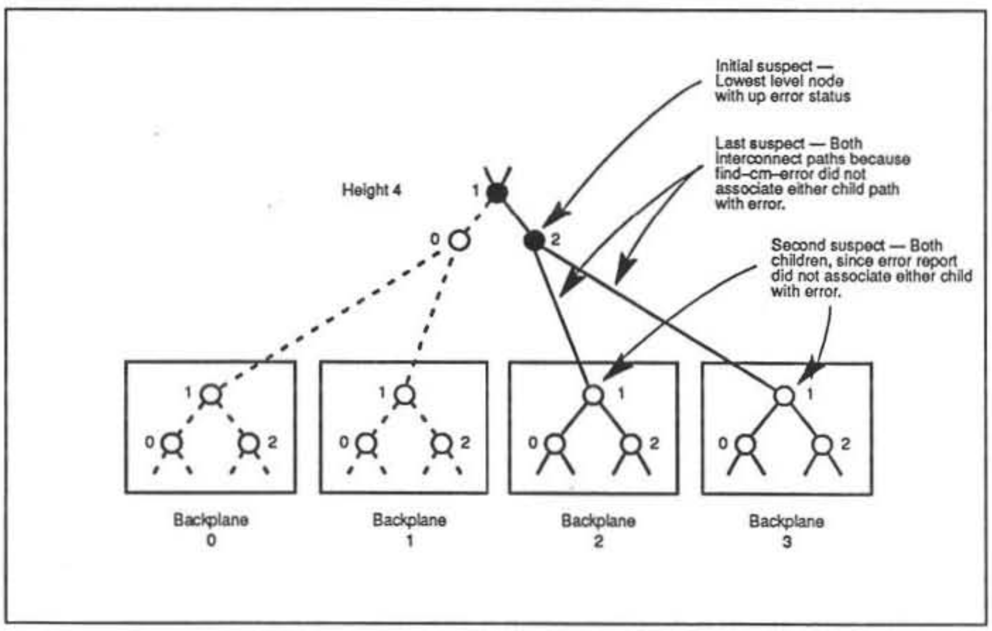


Figure 18. Node-level analysis — up and down errors
(with child path specified)

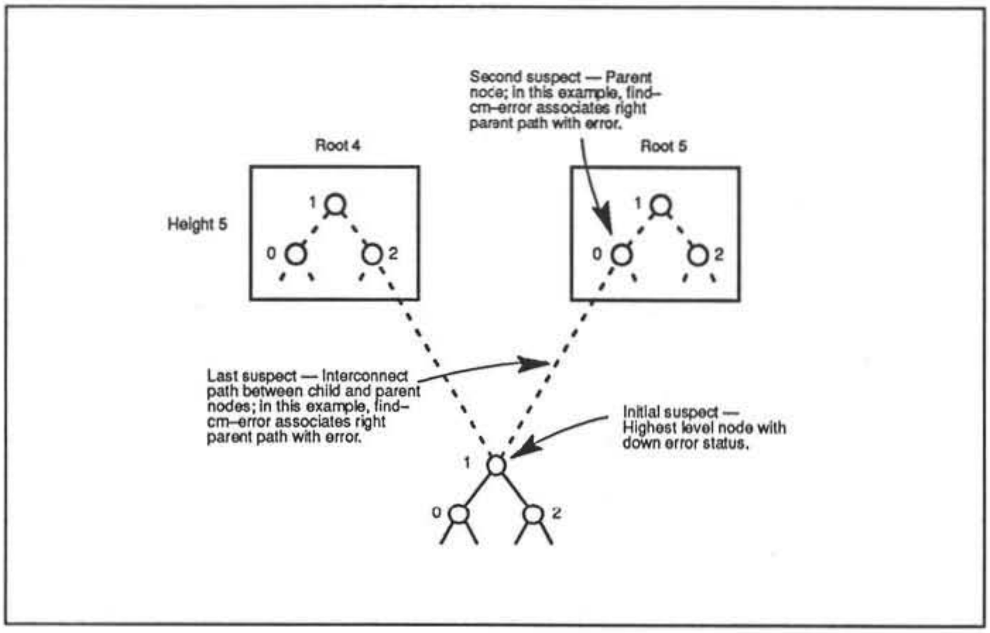


Figure 19. Node-level analysis — down errors only
(with parent path specified)

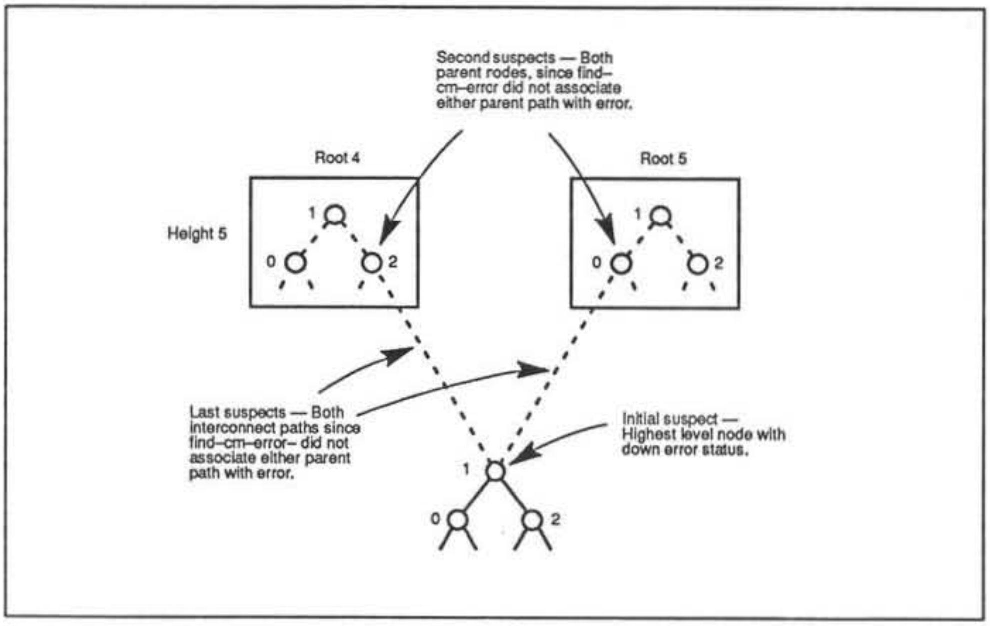


Figure 20. Node-level analysis — down errors only
(with no parent path specified)

Appendix E

Tracing Data Network Errors

When a component or interconnect path in the Data Network fails, the failure will propagate through the network along a single path. The result is a set of Data Network nodes all reporting error status.

This appendix explains how to use `find-cm-error` to isolate Data Network faults to no more than two components and their interconnecting signal path. At that point, you can identify and replace the faulty part through a process of elimination.

The Data Network troubleshooting procedure is described below.

NOTE: For this procedure, you will need legible copies of the Data Network cable assembly drawings for all levels of the network in your system. You will also need a large surface (eg, table) on which to spread these drawings open.

NOTE: This procedure is based on the assumption that you have already run `cmdiag` tests and have executed `find-cm-error`.

1. Lay out the network cable assembly drawings in a location where you can also read the error system report.

2. Examine the `find-cm-error` output, looking for the network node reporting a FATAL error status.

   NOTE: The first DR component to detect the error will record a unique error status, called a FATAL error. All subsequent DR components in the error message path will report SOFT errors.

3. Locate the fatal-error component on the Data Network cable assembly drawings. Ignore all soft-error components. Note which parent or child port is implicated in the fatal error. Figure 21 shows an example of this.

At this point, you will have narrowed the search to the following elements. Figure 22 illustrates the node-level view of these elements.

- first suspect — the fatal-error component
- second suspect — the component at the other end of the parent or child path associated with the fatal error
- third suspect — the implicated path itself

NOTE: The component at the other end of the implicated path is suspect because a failing component will not necessarily generate its own fatal error status. Therefore, the fatal-error component may actually reflect an error originating in a parent or child connected to it.

4. Identify the circuit board(s) that contain the components identified in step 3. If these components are on different backplanes, identify the interconnecting cable that was implicated in step 3 as well.

5. Replace the circuit board containing the fatal-error component and rerun the **`cmdiag`** test that exposed the error.

6. If **`cmdiag`** reports no errors, the board replacement may have corrected the problem. Verify by running **`cmdiag -f`**.

   If the system passes the complete manufacturing version of **`cmdiag`**, return the system to regular operation. If errors are reported, go to step 7.

7. Restore the original board containing the fatal-error component to its slot. Then replace the board containing the second suspect (the parent or child component) and rerun the test that reported the error.

8. If **`cmdiag`** reports no errors, the board replacement may have corrected the problem. Verify by running **`cmdiag -f`**.

   If the system passes `cmdiag -f`, return the system to regular operation. If errors are reported, go to step 9.

9. Examine the connectors on the cable identified in step 3. If you find bent or damaged pins, repair if possible. If repair is not possible, replace the cable. In either case, rerun the test that reported the error when you finish working on the cable.

   If no errors are reported, return the system to regular operation. If errors are reported, call Cambridge for assistance.

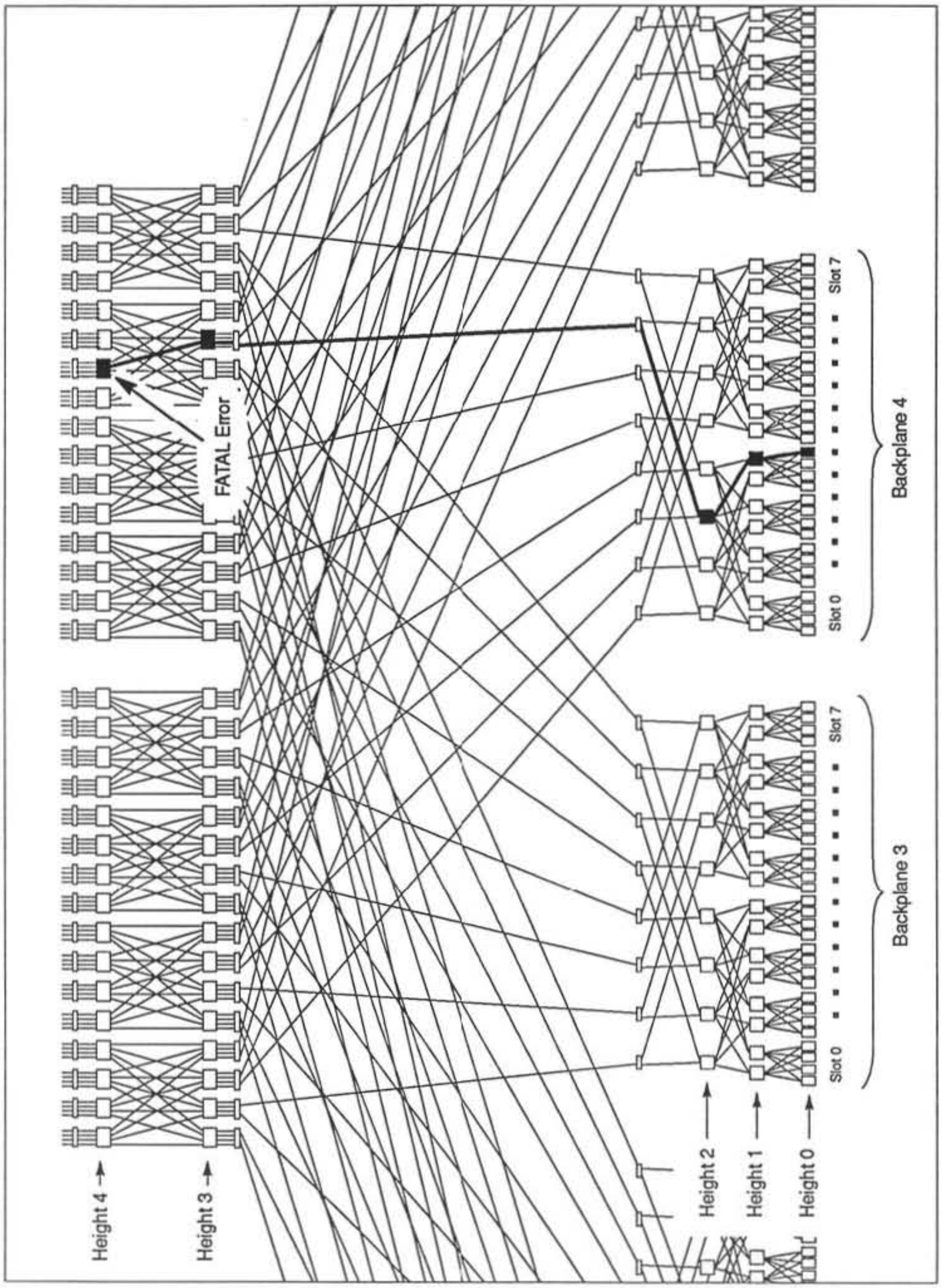


Figure 21. DR error analysis

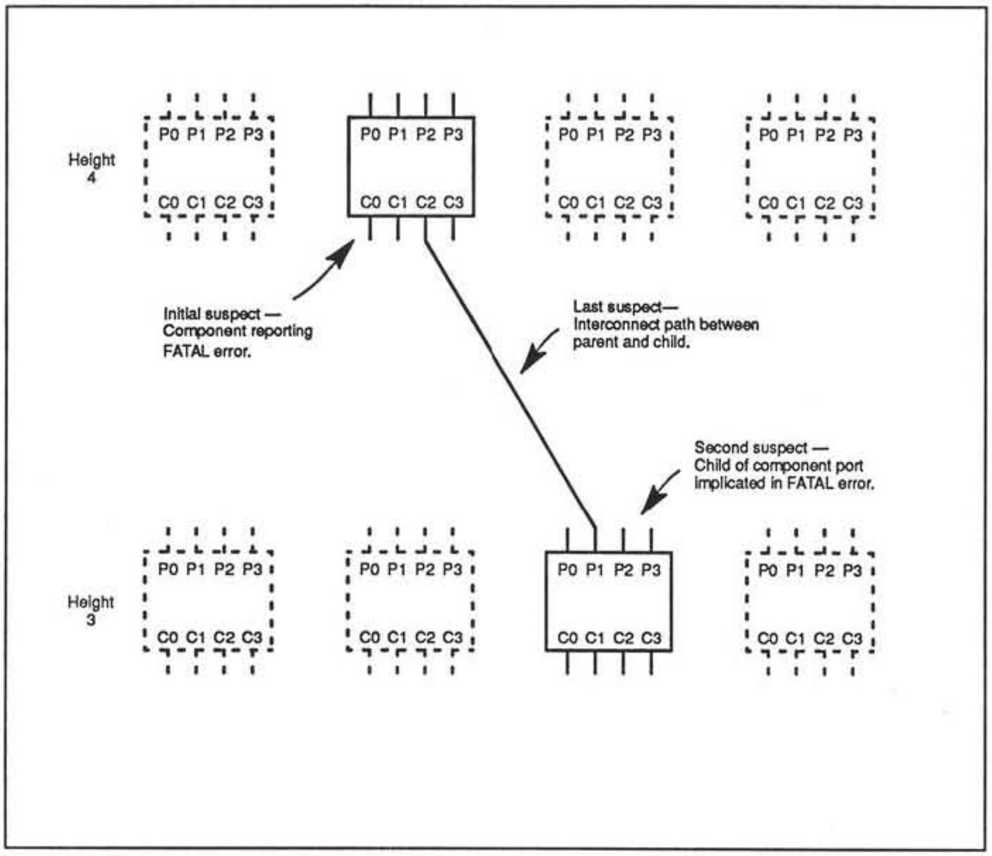


Figure 22. Data Network node-level analysis

Appendix F

I/O Diagnostic Tools

F.1 Introduction

This Appendix describes the diagnostic tools that the CM provides for troubleshooting I/O problems. Table 2 identifies the I/O test programs and summarizes the diagnostic coverage they provide. Figure 23 shows where the various diagnostic programs execute.

These I/O test programs are described more fully in the following sections.

- **Section F.2, IOBA Internal Tests**
- **Section F.3, CM-Based Verifiers**
- **Section F.4, DataVault Internal Tests**
- **Section F.6, CM-IOPG Internal Tests and Verifiers**
- **Section F.7, CM-HIPPI Internal Tests and Verifiers**

Appendix H explains how to use these diagnostic tools in typical I/O troubleshooting activities.

NOTE: SDA diagnostics are described in the SDA Field Service Guide.

Table 2. CM-5 I/O Diagnostic Tools.

Test Name (by category)	Execution Environment	Summary
IOBA internal diagnostics		
IOCLK, IODR, IOCNTRL IOBUF, IOCHNL	/usr/diag/cmdiag Execute on CM-5 diagnostic server.	JTAG scan tests that check signal paths on the IOBA boards; each test name indicates the board it covers.
IOP-CNTRL, IOP-CHNL, IOP-BUF, IOP-SYS	/usr/diag/cmdiag Execute on CM-5 diagnostic server.	Set of functional tests that exercise specific sections of IOBA hardware.
CM-based Verifiers		
execute-all-iopdv-tests	/usr/diag/cmdiag Execute on CM-5 diagnostic server.	IOBA writes test data to the DataVault and reads it back; this program verifies cabling between the CM-5 and the DataVault.
execute-all-ioppe-tests	/usr/diag/cmdiag Execute on CM-5 diagnostic server.	PNs write test data to IOBA buffers and read it back; this program verifies data and control paths between the IOBA and the DataVault.
test-cmio-device-data-xfer	/usr/diag/cmdiag Execute on CM-5 diagnostic server.	PNs write test data to all I/O devices listed in the io.conf file and read it back; this program verifies all relevant control and data paths in the I/O subsystem; you can select individual tests to focus attention on specific sections of hardware.
dvtest5-vu dvtest5-sparc	/usr/local/etc Execute on CM-5 diagnostic server.	Emulates I/O-intensive user applications; functions include opening files and directories, selecting an SDA or IOBA I/O Processor, writing test data from PNs to target DataVaults or CM-IOPGs and reading it back. dvtest5-vu requires PNs with vector units. dvtest5-sparc can be run on systems with or without vector units.
hippi-loop5	/usr/local/etc Execute on CM-5 diagnostic server.	Resembles dvtest5-sparc, except it reads and writes a CM-HIPPI.

Table 3. Table 2. CM-5 I/O Diagnostic Tools
(continued)

Test Name (by category)	Execution Environment	Summary
DataVault internal diagnostics		
`dvdiag`	`/usr/local/etc/diag` Execute on DataVault diagnostic server.	A diagnostic package that tests the functionality of all internal DataVault hardware. It includes a loop-back test that checks the DataVault's CMIO bus interface
CM-IOPG internal diagnostics		
`viodiag`	`/usr/local/etc` Execute on CM-IOPG diagnostic server.	A diagnostic program that tests the CM-IOPG's internal hardware.
CM-IOPG verifiers		
`serial`	`/usr/local/etc` Execute on CM-IOPG diagnostic server.	Writes test data from the CM-IOPG to a DataVault and reads it back; this program verifies the data and control paths (including CMIO bus) between the CM-IOPG and the DataVault.
`TapeDVxfervfr`	`/usr/local/etc` Execute on CM-IOPG diagnostic server.	Writes test data from the CM-TUD to a DataVault and reads it back; this program verifies the data and control paths (including CMIO bus) between the tape system and the DataVault.
CM-HIPPI internal diagnostics		
`iopdiag, dstdiag, srcdiag, sysdiag,`	`/usr/local/etc/diag` Execute on CM-HIPPI diagnostic server.	Set of functional tests that diagnose specific sections of CM-HIPPI hardware.

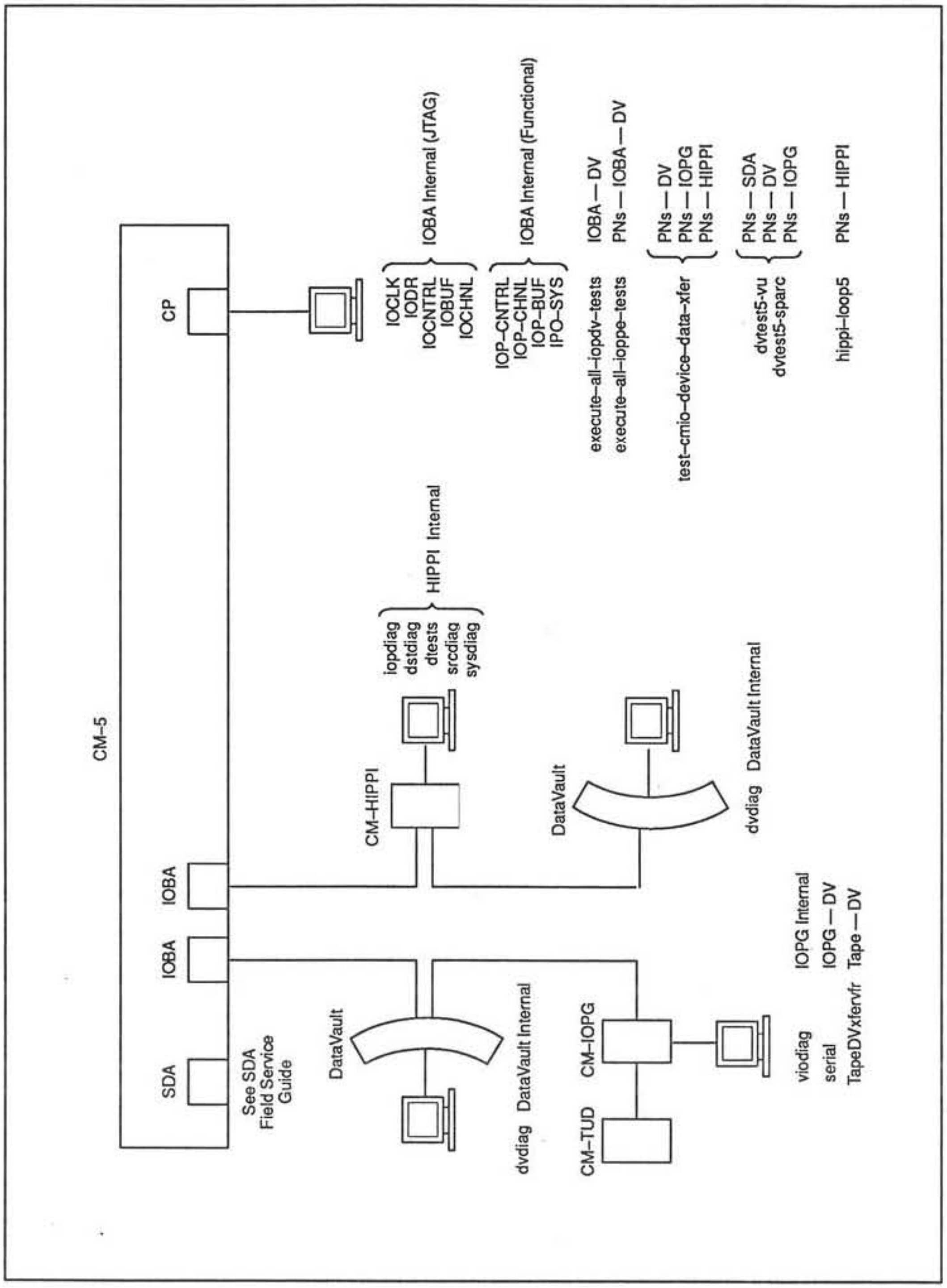


Figure 23. CM-5 I/O test programs — platform and coverage summary.

F.2 IOBA Internal Diagnostics

cmdiag contains nine test groups that specifically target IOBA functionality. These test groups are listed below and described in Sections F.2.1 through F.2.9. The first five are JTAG tests; the other four are functional tests that exercise specific sections of IOBA hardware.

IOCLK	JTAG	Section F.2.1
IODR	JTAG	Section F.2.2
IOCNTRL	JTAG	Section F.2.3
IOBUF	JTAG	Section F.2.4
IOCHNL	JTAG	Section F.2.5
IOP-CNTRL	Functional	Section F.2.6
IOP-BUF	Functional	Section F.2.7
IOP-CHNL	Functional	Section F.2.8
IOP-SYS	Functional	Section F.2.9

NOTE: In addition to these nine I/O-specific test groups, the Data Network verifier, **dr**, provides some coverage of I/O functionality as well. It tests the IOBA's IOCNTRL board as if it were a Processing Node attached to the network.

Test groups focus on particular boards or subsystems in the IOBA; the test names indicate their respective coverage. Figure 24 illustrates the IOBA board configuration in block diagram form.

Invoke these tests in the **cmdiag** shell using the **rgroups** command with **m** (manufacturing version) flag. For example:

```
<CM-DIAG> rgroups m IOP-SYS
```

runs verifier tests on the IOSYS board.

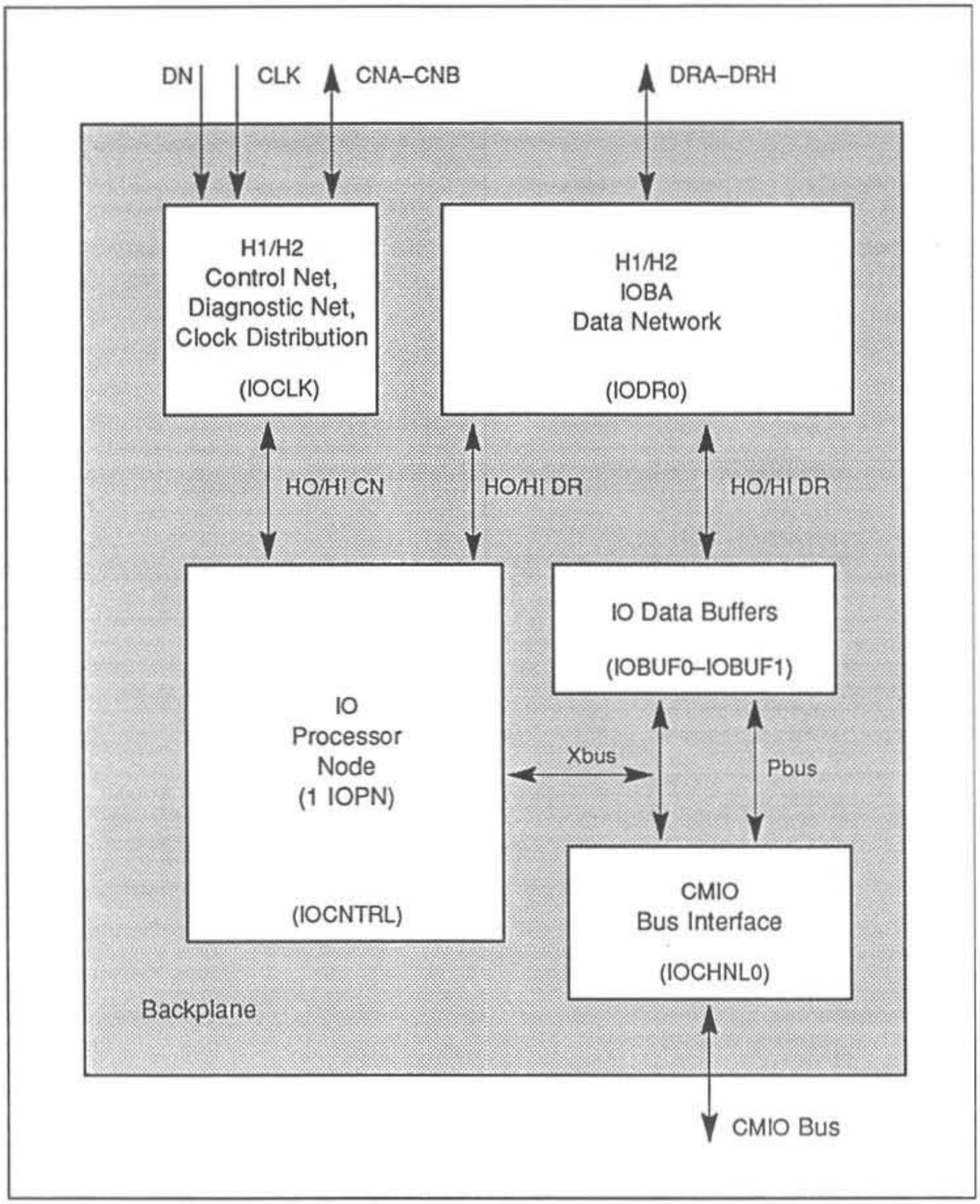


Figure 24. IOBA basic block diagram.

F.2.1 IOCLK (JTAG)

This group runs boundary scan tests on all JTAG-accessible IOCLK board components.

F.2.2 IODR (JTAG)

This group runs boundary scan tests on all JTAG-accessible IODR board components. If the IOBA contains multiple IODR boards, all are tested in parallel.

F.2.3 IOCNTRL (JTAG)

This group runs boundary scan tests on all JTAG-accessible IOCNTRL board components.

F.2.4 IOBUF (JTAG)

This group runs boundary scan tests on all JTAG-accessible IOBUF board components. If an IOBA contains multiple IOBUF boards, all are tested in parallel.

F.2.5 IOCHNL (JTAG)

This group runs boundary scan tests on all JTAG-accessible IOCHNL board components. If an IOBA contains multiple IOBUF boards, all are tested in parallel.

F.2.6 IOP-CNTRL (Functional)

This group runs a set of functional tests on the IOBA Control board (IOCNTRL). Its primary diagnostic focus is XBUS operations and related functionality.

NOTE: Run the `cmdiag PE` test group first to verify Processing Node functionality before running `IOP-CNTRL`.

F.2.7 IOP-BUF (Functional)

This group tests the fuctionality of the IOBA Buffer board (IOBUF). The tests will begin with the first IOBUF board they encounter and then step through all subsequent IOBUF boards in the IOBA. The tests will automatically repeat until all IOBUF boards in the IOBA have been tested.

In a system with multiple IOBAs, these tests will be run on all IOBAs in parallel. If the IOBAs have different numbers of IOBUF boards, the tests will repeat until all IOBUF boards in the largest set have been tested. Consequently, some IOBUF boards in the smaller sets will experience redundant testing as **IOP-BUF** wraps around to the first IOBUF board in the set.

F.2.8 IOP-CHNL (Functional)

This group tests the functionality of the IOBA Channel board.

These tests require the environment variable IOP_STATION_ID to be set before they are run. This variable takes the form IOP*XXX*_STATION_ID, where *XXX* is the physical network address of the IOP (in decimal).

If this variable is not already set at the time you invoke **IOP-CHNL**, you will be asked to supply it. If there are multiple IOPs, you will be prompted for each undefined IOP station ID. The following example illustrates the dialog for a system with two IOPs at address locations 480 and 490.

```
<CM-DIAG> rgroups m IOP-CHNL
IOPXXX_STATION_ID? 480
IOPXXX_STATION_ID? 490
        .
        .                        ; tests execute
        .
<CM-DIAG>
```

NOTE: If you do not know the IOP address(es), run `cmpartition list -l`.

F.2.9 IOP-SYS (Functional)

This test group focuses primarily on PBUS operations. Related functionality on other IOBA boards is also tested as a by-product of the PBUS exercises.

These tests require the environment variable `IOPXXX_STATION_ID` to be set before they are run. See section F.2.8, IOP-CHNL for details.

F.3 CM-Based Verifiers

This section describes several system-level verifiers that are executed on the CM's diagnostic server or from a partition running a timesharing daemon.

The main service provided by each of these the verifiers is to transfer test data to and from a peripheral device, thereby exercising most or all of the functionality that is needed by user I/O. Data patterns are varied to locate bit-sensitive faults more readily.

The choice of which verifiers to use can be influenced by a number of considerations, including: the type of periperal involved, the tradeoff between test rigor and time available to test, and personal preference. A brief summary of each verifier is provided below. Additional detail is presented in Sections F.3.1 through F.3.3.

- Focused CM-to-DataVault Verifiers: This consists of two complementary verifiers, each of which exercises a separate segment of the CM-to-DataVault path. One transfers test data between the IOBA and the DataVault; the other transfers test data between the PNs and the IOBA. The second verifier also writes test data from the PNs out to the DataVault and reads it back to check the I/O data and control paths across their full length. These verifiers are described in Section F.3.1.

- End-to-End Tests: This refers to a package of three independent verifiers, each tailored for a different type of peripheral: for a DataVault, CM-IOPG, or CM-HIPPI. The user interface provides considerable control over details of the test data transfers. All transfers are between the PNs and the appropriate peripheral device. These are described in Section F.3.2.

- `dvtest5-vu, dvtest5-sparc`: This program performs a comprehensive emulation of an I/O-intensive user application, including verification of the file system software. It can target either a DataVault or CM-IOPG as its peripheral device. **dvtest5-vu** and **vtest5-sparc** are described in Section F.3.3.

- **hippi-loop5**: This verifier is equivalent to **dvtest5-sparc**, except it targets CM-HIPPIs. It is described in Section F.3.4.

F.3.1 Focused CM-to-DataVault Verifiers

cmdiag includes two complementary test programs that transfer blocks of test data between the CM and a DataVault. Each program focuses on a different section of the PN-to-DataVault I/O path.

- **execute-all-iopdv-tests** — This program writes various data patterns from the IOBA to the DataVault, reads the data back, and compares the read data with the data that was sent.

- **execute-all-ioppe-tests** — This program has two phases. First, it writes test data from the PNs to the IOBA buffers and reads it back, verifying that segment of the I/O path. Then, the PNs write test data all the way out to the DataVault and read it back, verifying the I/O path's full length.

  If this test is run on a partition, it uses all PNs in its partition. If it is run on the entire system, it uses all PNs found by autosizing

  NOTE: The PNs used by **execute-all-ioppe-tests** must be logically contiguous (within a continuous address range), and the number of PNs must be a power of two.

Three environment variables must be set before either test can be run. These are described below.

- **IOP*XXX*_STATION_ID** specifies the station ID of the IOP to be used in the test. *XXX* is the IOP's physical network address (in decimal). If this variable is not already set, you will be prompted for it. If you are uncertain about the system's IOP addresses, run **cmpartition list -l**.

- **IOP*XXX*_DV_START_BLOCK** specifies the starting block address to which the IOBA will write test data patterns. Choose any number from 1 to 960. If this value is not already set, you will be prompted to supply it.

  NOTE: The maximum start address (960) is determined by the size of the region on the DataVault that is reserved for diagnostic use and by the largest number of blocks written by these tests. The DataVault's diagnostic zone measures 1024 blocks. and the largest number of blocks written by these tests is 64 — consequently, 1023 – 63 = 960.

- **IOP*XXX*_DV_STATION_ID** must match the station ID of the DataVault port that will be used by the test. If you do not know the DataVault's station ID, open **/etc/io.conf**, which describes the CM's I/O configuration. Appendix J explains how to interpret the contents of **/etc/io.conf**.

Both tests require the DataVault port that will be used in the test to be configured for command channel operation. To invoke command channel mode, run **dvcoldboot +c*n*** on the DataVault you plan to test. Enter **+c0** or **+c1** to specify the appropriate DataVault port.

When you are done with these tests, run **dvcoldboot -c*n*** (with $n = 0$ or 1) to restore the port to the data channel mode.

NOTE: Although the DataVault firmware controlling the port ordinarily switches automatically to data channel mode as needed, it is advisable to explicitly turn command channel mode off before using the DataVault port for any other operations.

F.3.2 End-to-End Tests

cmdiag includes three groups of I/O verifiers, each of which targets a different type of peripheral device.

The tests within each group differ from one another in the data patterns they use and/or in the specific hardware modules they target. Currently the three groups of end-to-end tests support DataVault, CM-HIPPI, and VMEIO peripherals (*e.g.*, CM-IOPG). Figure 26 through Figure 27 describe the various tests contained in each device-specific group.

NOTE: A diagnostic server must be running on the I/O device's host computer. The procedure for starting an I/O device's diagnostic server can be found in the I/O device's installation and service manual.

These tests report errors to the screen and to **diag-error-log.*hostname*** in the local directory.

F.3.2.1 Alternative Approaches to Using End-to-End Tests

Invoke end-to-end tests within the **cmdiag** shell in either of two ways:

- Invoke a single, device-specific test group to verify the functionality of a particular peripheral device and its I/O path. This is the approach you are most likely to use when troubleshooting suspected I/O hardware faults. Figure 25 through Figure 27 provide detailed descriptions of these device-specific test groups. Section F.3.2.2 explains how to use them.

- Alternatively, you can use a single, high-level command, **test-cmio-device-data-xfer**, to automatically execute all test groups that are appropriate for your I/O configuration. This can be a convenient method for verifying all devices on a multidrop bus or a complete I/O system during a major scheduled maintenance session. See Section F.3.2.3 for details.

In either case, the end-to-end tests must be run on a partition that has an IOBA associated with it — that is, the partition must have been created with the **-iop** *address* option.

DataVault Test Group

cm5-write-datavault

Writes data from CM PNs to DataVault.

Syntax: `cm5-write-datavault pattern block_size speed`

For example,

```
cm5-write-datavault ffffffff 4 2
```

writes 4 blocks of data in pattern 0xffffffff in speed 2 to a DataVault.

---

cm5-read-datavault

Reads data from DataVault.

Syntax: `cm5-read-datavault pattern block_size speed`

For example,

```
cm5-read-datavault 2 4 2
```

reads 4 blocks of data in speed 2 from a DataVault.

---

cmio-dv-iope-xfer

Transfers data from CM PNs to a DataVault.

Syntax: `cmio-dv-iope-xfer pattern block_size speed`

For example,

```
cmio-dv-iope-xfer ffffffff 4 2
```

transfers 4 blocks of data in pattern 0xffffffff in speed 2 to a DataVault.

---

cmio-dv-iope-all-pattern-xfer

Transfers all the data patterns 0x00000000, 0xffffffff, 0xaaaaaaaa, 0x55555555, and 0x37c837c8 from CM PNs to a DataVault.

Syntax: `cmio-dv-iope-all-pattern-xfer block_size speed`

For example,

```
cmio-dv-iope-all-pattern-xfer 4 2
```

transfers 4 blocks of data in patterns 0x00000000, 0xffffffff, 0xaaaaaaaa, 0x55555555, and 0x37c837c8 in speed 2 to a DataVault.

Figure 25. DataVault test group — page 1 of 1

HIPPI Test Group

ABBREVIATIONS USED:

HIPPI_IOP = CM-HIPPI interface to the CMIO bus.

SM = CM-HIPPI's Station Manager.

SRC = CM-HIPPI's Source module.

DEST = CM-HIPPI's Destination module.

---

cm5-write-hippi

Writes data from PNs to CM-HIPPI.

Syntax: **cm5-write-hippi test pattern block_size speed**

test = **0, 1, 2, 3,** or **4** indicate different CM-HIPPI paths:

0	CM-5 → HIPPI_IOP → SM
1	CM-5 → HIPPI_IOP → SRC → SM
3	CM-5 → HIPPI_IOP → SRC → DEST → SM

For example,

```
cm5-write-hippi 0 ffffffff 4 2
```

writes 4 blocks of data in pattern 0xffffffff at speed 2 to a CM-HIPPI. The data path is
CM-5 → HIPPI_IOP → HIPPI_Station_Manager

---

cm5-read-hippi

Reads data from CM-HIPPI.

Syntax: **cm5-read-hippi test block_size speed**

test = **0, 1, 2, 3,** or **4** indicate different CM-HIPPI paths:

0	SM → HIPPI_IOP → CM-5
2	SM → DEST → HIPPI_IOP → CM-5

For example,

```
cm5-read-hippi 2 4 2
```

reads 4 blocks of data in speed 2 from a CM-HIPPI. The data path is
HIPPI_Station_manager → HIPPI_destination → HIPPI_IOP → CM-5

Figure 26. HIPPI test group — page 1 of 3

HIPPI Test Group
(continued)

---

cmiohippi-data-pattern-xfer

Transfers data from PNs to CM-HIPPI.

Syntax: **cmiohippi-data-pattern-xfer test pattern block_size speed**

test = **0**, **1**, **2**, **3**, or **4** indicate different CM-HIPPI paths:

0 CM-5 → HIPPI_IOP → SM → HIPPI_IOP → CM-5
1 CM-5 → HIPPI_IOP → SRC → SM → HIPPI_IOP → CM-5
2 CM-5 → HIPPI_IOP → SRC → SM → DEST → HIPPI_IOP → CM-5
3 CM-5 → HIPPI_IOP → SRC → DEST → SM → HIPPI_IOP → CM-5
4 CM-5 → HIPPI_IOP → SM → DEST → HIPPI_IOP → CM-5

For example,

```
cmiohippi-data-pattern-xfer 2 ffffffff 4 2
```

transfers 4 blocks of data in pattern 0xffffffff at speed 2 to CM-HIPPI. The data path is:
CM-5 → HIPPI_IOP → HIPPI_Source → HIPPI_Station_Manager → HIPPI_Destination → HIPPI_IOP → CM-5

---

cmiohippi-cmio-data-xfer

Transfers all the data patterns 0x00000000, 0xffffffff, 0xaaaaaaaa, 0x555555555, and –x37c837c8 from CM PNs to CM-HIPPI.

Syntax: **cmiohippi-cmio-data-xfer block_size speed**

For example,

```
cmiohippi-cmio-data-xfer 0 4 2
```

transfers 4 blocks of data in patterns 0x00000000, 0xffffffff, 0xaaaaaaaa, 0x55555555, and 0x37c837c8 in speed 2 to a CM-HIPPI.

Figure 26. HIPPI test group — page 2 of 3

HIPPI Test Group
(continued)

`cmiohippi-cmio-data-all-path-xfer`

Transfers all the data patterns 0x00000000, 0xffffffff, 0xaaaaaaaa, 0x55555555, and 0x37c837c8 from CM PNs to CM-HIPPI. Tests all different CM-HIPPI paths.

Syntax: **`cmiohippi-cmio-data-all-path-xfer block_size speed`**

For example,

```
cmiohippi-cmio-data-xfer 0 4 2
```

transfers 4 blocks of data in patterns 0x00000000, 0xffffffff, 0xaaaaaaaa, 0x55555555, and 0x37c837c8 in speed 2 to a CM-HIPPI. The test paths are:

(0) CM-5 → HIPPI_IOP → SM → HIPPI_IOP → CM-5
(1) CM-5 → HIPPI_IOP → SRC → SM → HIPPI_IOP → CM-5
(2) CM-5 → HIPPI_IOP → SRC → SM → DEST → HIPPI_IOP → CM-5
(3) CM-5 → HIPPI_IOP → SRC → DEST → SM → HIPPI_IOP → CM-5
(4) CM-5 → HIPPI_IOP → SM → DEST → HIPPI_IOP → CM-5

Figure 26. HIPPI test group — page 3 of 3

VMEIO Test Group

---

cm5-write-vmeio

Writes data from CM PNs to VMEIO.

Syntax: **cm5-write-vmeio pattern mode ram-mode block_size speed**

mode	**m** (master) or **s** (slave).
ram-mode	**m** (memory), **f** (fifo) or **b** (bypass)

For example,

cm5-write-vmeioi ffffffff m f 4 2

writes 4 blocks of data in pattern 0xffffffff in speed 2 to a VMEIO in master mode.

---

cm5-read-vmeio

Reads data from CM PNs to VMEIO.

Syntax: **cm5-read-vmeio mode ram-mode block_size speed**

mode	**m** (master) or **s** (slave).
ram-mode	**m** (memory), **f** (fifo) or **b** (bypass)

For example,

cm5-read-vmeio m f 4 2

reads 4 blocks of data in speed 2 to a VMEIO in master mode.

Figure 27. VMEIO test group — page 1 of 2

VMEIO Test Group
(continued)

---

cm5-vmeio-iope-xfer

Transfers data from CM PNs to VMEIO.

Syntax: **`cm5-vmeio-iope-xfer pattern mode ram-mode block_size speed`**

`mode`	**`m`** (master) or **`s`** (slave).
`ram-mode`	**`m`** (memory), **`f`** (fifo) or **`b`** (bypass)

For example,

```
cmio-vmeio-iope-xfer ffffffff m f 4 2
```

transfers 4 blocks of data in pattern 0xffffffff in speed 2 to a VMEIO in master mode.

---

cmio-vmeio-iope-all-pattern-xfer

Transfers all thne data patterns 0x00000000, 0xffffffff, 0xaaaaaaaa, 0x55555555, and 0x37c837c8 from CM PNs to VMEIO.

Syntax: **`cmio-vmeio-ope-all-pattern-0xfer access-mode ram-mode block-count speed`**

For example,

```
cmio-vmeio-iope-all-pattern-xfer 4 m f 2
```

transfers 4 blocks of data in patterns 0x00000000, 0xffffffff, 0xaaaaaaaa, 0x55555555, and 0x37c837c8 in speed 2 in master mode to a VMEIO.

Figure 27. VMEIO test group — page 2 of 2

F.3.2.2 Executing Individual Tests

Perform the following steps to execute individual tests. Figure 28 provides an example of this procedure.

NOTE: The I/O device must have its diagnostic server running in background.

1. Log in as root on the CM-5 diagnostic server and change directory to **/usr/diag/cmdiag**.

   ```
   login: user_id
   % su
   password: root_password
   SU hostname /dev/console
   # cd /usr/diag/cmdiag
   ```

2. Set the **CMDIAG_PATH** and **JTAG_SERVER** environment variables. The default **CMDIAG_PATH** is **/usr/diag/cmdiag**. The **JTAG_SERVER** variable must specify the hostname of the diagnostic server.

   ```
   # setenv CMDIAG_PATH /usr/diag/cmdiag
   # setenv JTAG_SERVER diag_server_hostname
   ```

3. Create a partition using the **-iop** *address* option to associate a particular IOBA with the partition.

   ```
   # cmpartition create -pm pm_name -iop address
   ```

   NOTE: If the desired partition already exists, halt its timesharing daemon by executing **cmpartition stop** on the partition manager.

   ```
   # rlogin -l root pm_name
   password: root_password
   # cmpartition stop
   # exit
   ```

4. Invoke the **cmdiag** environment and specify which peripheral device the test program will write to and read from. Do this by entering the following at the <CM-DIAG> prompt.

   ```
   # ./cmdiag -p pm_name
   <CM-DIAG> select-cmio-server hostname
   ```

   This command establishes a link to the desired I/O diagnostic server; *hostname* is the hostname of the I/O device on which that server is running.

5. Next, identify the type of peripheral device that will be involved in the test.

```
<CM-DIAG> init-cmio-diag-environment device_type
```

   device_type identifies the type of I/O device; legal strings are: **"hippi"**, **"dv"**, or **"vmeio"**. This command also resets the partition.

6. Individual tests can now be invoked at the <CM-DIAG> prompt. Three examples from the CM-HIPPI test group are shown below. The procedure example shown in Figure 28 uses tests from the DataVault group.

```
<CM-DIAG> cm5-write-hippi 0 ffffffff 4 2
<CM-DIAG> cm5-read-hippi 0 4 2
<CM-DIAG> cmiohippi-cmio-data-xfer
```

Invoking Individual End-to-End Tests
(see Section B.3.2.2)

Introductory Notes

The system used to illustrate this procedure example has the following features.

- Diagnostic server is named **homer.think.com**. Prompt = hom#.
- Tests will be run on partition named **virgil.think.com** and a DataVault, which is connected to an IOBA at address 480. Prompt for partition manager is virg#.

1
```
login: user_id
% su
password: root_password
SU homer.think.com /dev/console
hom# cd /usr/diag/cmdiag
```

2
```
hom# setenv CMDIAG_PATH /usr/diag/cmdiag
hom# setenv JTAG_SERVER homer.think.com
```

3
```
hom# cmpartition create -pm virgil.think.com -iop 480
```

4
```
hom# ./cmdiag -p virgil.think.com
<CM-DIAG> select-cmio-server dv1-server
```

5
```
<CM-DIAG> init-cmio-diag-environment "dv"
```

6
```
<CM-DIAG> cm5-write-datavault
            .
            .                    ; diagnostic test report
            .

<CM-DIAG> cm5-read-datavault
            .
            .                    ; diagnostic test report
            .

<CM-DIAG> quit
```

Figure 6. Example for invoking individual end-to-end tests.

B.3.2.3 Executing Test Groups Automatically

test-cmio-device-data-xfer automatically executes the appropriate set of test groups for the I/O configuration in which it is invoked. It decides which test groups to run based on the following.

- **test-cmio-device-data-xfer** must be run on a partition that has an IOBA associated with it (*i.e.*, the partition was created with the **-iop** *address* option).
- The list of I/O devices connected to that IOBA is provided to **cmdiag** when **cmdiag** is invoked with the **-p** *pm_name* option.
- **test-cmio-device-data-xfer** surveys the list of I/O devices. It then selects the first device listed and runs the test group that applies to its device type.
- If multiple I/O devices are connected to that IOBA, **test-cmio-device-data-xfer** will select the next listed device and run the appropriate test group for it. It repeats this process until all I/O devices connected to the target IOBA have been tested.

Perform the following steps to run end-to-end tests automatically. Figure 7 provides an example of this procedure.

NOTE: All I/O devices connected to the target IOBA must have its diagnostic server running in background.

1. Log in as root on the CM-5 diagnostic server and change directory to **/usr/diag/cmdiag**.

```
login: user_id
% su
password: root_password
SU hostname /dev/console
# cd /usr/diag/cmdiag
```

2. Set the **CMDIAG_PATH** and **JTAG_SERVER** environment variables. The default **CMDIAG_PATH** is **/usr/diag/cmdiag**. The **JTAG_SERVER** variable must specify the hostname of the diagnostic server.

```
# setenv CMDIAG_PATH /usr/diag/cmdiag
# setenv JTAG_SERVER diag_server_hostname
```

3. Create a partition using the **-iop** *address* option to associate a particular IOBA with the partition.

   ```
   # cmpartition create -pm pm_name -iop address
   ```

   NOTE: If the desired partition already exists, halt its timesharing daemon by executing **cmpartition stop** on the partition manager.

   ```
   # rlogin -l root pm_name
   password: root_password
   # cmpartition stop
   # exit
   ```

4. Invoke the **cmdiag** environment and enter **test-cmio-device-data-xfer** at the <CM-DIAG> prompt.

   ```
   # ./cmdiag -p pm_name
   <CM-DIAG> test-cmio-device-data-xfer
   ```

Invoking End-to-End Tests Automatically

(see Section B.3.2.3)

Introductory Notes

The system used to illustrate this procedure example has the following features.

- Diagnostic server is named **homer.think.com**. Prompt = hom#.
- Tests will be run on partition named **virgil.think.com** and a DataVault, which is connected to an IOBA at address 480. Prompt for partition manager is virg#.

1
```
login: user_id
% su
password: root_password
SU homer.think.com /dev/console
hom# cd /usr/diag/cmdiag
```

2
```
hom# setenv CMDIAG_PATH /usr/diag/cmdiag
hom# setenv JTAG_SERVER homer.think.com
```

3
```
hom# cmpartition create -pm virgil.think.com -iop 480
```

4
```
hom# ./cmdiag -p virgil.think.com
<CM-DIAG> test-cmio-device-data-xfer
          .
          .                    ; diagnostic test report
          .

<CM-DIAG> quit
```

Figure 7. Example for invoking individual end-to-end tests.

B.3.3 dvtest5–vu, dvtest5–sparc

dvtest5-vu and **dvtest5-sparc** are two versions of the same system verifier program — **dvtest5-vu** is used on systems with vector units installed and **dvtest5-sparc** is used on systems without vector units. Each creates and writes test files to the DataVault or to a VMEIO-based peripheral, such as a CM-IOPG and then reads back each file, comparing it with the expected data pattern. Sincle both versions are functionally identical, they are referred to here as **dvtest5-***ext*.

dvtest5-*ext* is functionally equivalent to a user application that has file system calls. Consequently, it requires the following conditions.

- The partition in which it is executed must have **ts-daemon** running.
- The IOBA that will be used must be defined in **/etc/io.conf**.
- **fsserver** must be running in background on the I/O device.
- The DVWD environment variable must specify the file server host of the target I/O device.

The syntax for **dvtest5-***ext* is as follows. **NOTE:** Use either **dvtest5-vu** or **dvtest5-sparc** in place of **dvtest5-***ext*.

dvtest5-vu | **dvtest5-sparc -x -t -1 -a**[*n*] **-s -h**
-g *int$_1$,...int$_n$* **-d** *directory-name* **-1** *testname...testname*

-x	Exit on error.
-t	Report tersely.
-1	Run the applicable test(s) once and then exit.
-a[1]	Run all tests automatically (no menu). If **-a1** is specified, the tests run once; otherwise they loop forever. Stop execution by entering Ctrl-C.
-s	Run software test subset automatically (no menu).
-h	Run hardware test subset automatically (no menu).
-g *int$_1$,...int$_n$*	Specify a geometry to be applied to the data being transferred. A string of two or more integers separated by commas specify the geometry. Each integer represents a dimension measured in 32-bit words.

`-l` *testname*(s)   Run the test(s) specified by the *testname* argument(s).

Run **dvtest5-*ext* -a** to exercise the CM-5 system most thoroughly. This option includes tests of the file system software and would ordinarily be done following installation of CM-5 system software and/or installation of a new I/O device. **dvtest5-*ext*** cycles through the tests repeatedly until you exit with `Ctrl-C`.

Use the **-g** switch with one or more integer arguments to specify a geometry for the test data. Each integer argument specifies a dimension in units of four bytes (32-bit words). For example, the recommended geometry for data sent to a DataVault is 64 by 64, which yields a 16,384-byte data block, the DataVault's default block size.

Use the **-h** option to limit the tests to hardware functions; this will save time during routine (i.e., preventive) maintenance sessions when the validity of the system software is not in doubt.

Likewise, the **-s** option allows you to focus diagnostic attention on software functions.

The `-l` option lets you specify individual tests by name. It is useful when troubleshooting specific hardware or software functions.

If you do not specify any tests via `-a`, `-s`, `-h`, or `-l` *testname*, `dvtest5-`*ext* will present you with a test menu, allowing you to specify individual tests by number. This menu is shown in Figure 8.

```
1.  basics               test file creation/deletion (-s)
2.  data                 test simple file read/write (-s, -h)
3.  write                test writing files (-s, -h)
4.  link                 test link/unlink (-s)
5.  abs_seek             test absolute seek (random) (-s, -h)
6.  rel_seek             test relative seek (deterministic)
                         (-s, -h)
7.  dir_basics           test mkdir/rmdir/chdir (-s)
8.  many_dirs            test creating many directories (-s)
9.  many_files           test creating/deleting many
                         files (-s)
10. serial_io            test serial I/O transfers (-s)
11. mixed_io             test mixed serial and parallel I/O
                         (-s, -h)
12. parallel_partial     test parallel I/O w/ partial blocks
                         (-s, -h)
13. transpose            test transposing serial data (-s)
14. transfer_timing      test transfer speed (-s, -h)
15. raw_transfer_timing test raw transfer speed (-s, -h)
16. overhead_timing      test overhead speed (-s, -h)
17. max_transfer_timing test max transfer speed
18. reliability          test reliability (-h)
```

Figure 8. The `dvtest5`-*ext* menu.

If **`dvtest5`** fails, read the DCP error log on the DataVault. To read this log, invoke **`dvdiag`** on the DataVault console and run **`rdlog`**.

```
% /usr/diag/tsd/dvdiag
<DV-DIAG> rdlog byte_count byte_offset
```

byte_count specifies the number of bytes of log contents you want to display. *byte_offset* specifies the starting byte to be displayed. For example, to display the most recent 300 bytes of log contents, enter:

```
<DV-DIAG> rdlog 300 0
```

B.3.4 hippi–loop5

`hippi-loop5` performs a role similar to **`dvtest5`** for CM-HIPPI systems. It runs application code with file system calls to verify that data transfers between the CM–5 to CM-HIPPI are fully functional. Test data completes a circuit by looping from the destination board to the source board within the CM-HIPPI.

The command syntax for **`hippi-loop5`** is as follows.

`hippi-loop5 -i`*ifield* **`-r -w -p`***pattern* **`-s`***size* **`-D -U`**
`-n`*N* **`-v -g -h -R`**

`-i`*ifield*	Use *ifield* as the I-field when establishing the loopback connection
`-r`	Test only CM-5 reads from the CM-HIPPI channel.
`-w`	Test only CM-5 writes to the CM-HIPPI channel
`-p`*pattern*	Specify the data pattern to be used. Valid pattern inputs are: data–equal–address, random, or a hex value for a constant pattern. Default is data–equal–address.
`-s`*size*	Specify the size of the transfer. Valid size inputs are: 16k, 32k, 64k, 128k, 1M, 2M, and 4M.
`-D`	Drop the connection between tests.
`-U`	Use the existing connection if possible.
`-n`*Nrep*	Repeat each pattern *Nrep* times.
`-v`	Report verbosely when establishing and breaking connections.
`-g`	Print interface status.
`-h`	Use standard CM-HIPPI ports instead of loopback ports.

`hippi-loop5` has the same prerequisites as **`dvtest5`**, namely:

- The partition in which it is executed must have **`ts-daemon`** running.
- The target CM-HIPPI must be defined in **`/etc/io.conf`**.
- **`fsserver`** must be running in background on the CM-HIPPI.

- **DVWD** must specify the CM-HIPPI's file server host.

If **hippi-loop5** fails, read **dmesg** on the CM-HIPPI to determine which byte was in error. On the CM-HIPPI console, enter:

```
% dmesg
```

B.4 DataVault Internal Diagnostics

dvdiag is a DataVault file server shell command that invokes a special diagnostic environment for testing DataVault hardware functions. This environment is controlled entirely by the file server, enabling you to test nearly all DataVault functions without the aid of the CM or other external computer. **dvdiag** resides on the DataVault's station manager in **/usr/local/etc/diag**.

The overall command syntax is:

dvdiag -m -f -g*groupname* **-C +Ebdfilt -Ebdfilt**

The first three arguments relate to how **dvdiag** tests are accessed The other three control the behavior of the **dvdiag** command interface (**-C**) and the test environment (**+E** and **-E**).

dvdiag provides three test access modes. Depending on how you use the first three arguments, you can invoke:

- a complete, predefined test suite
- functional test groups
- indidvidual tests

Each of these modes in described separately in Sections B.4.1 through B.4.3. The arguments **+E** and **-E** are also explained in Section B.4.1. **-C** is explained in Section B.4.3.

B.4.1 Complete Test Suite

To invoke a comprehensive test suite that will exercise nearly all of the Data-Vault's internal hardware functions, use the following command syntax.

```
dvdiag -m +Ebdfilt -Ebdfilt
```

This executes the most rigorous level of testing. While its name implies use in a manufacturing setting, this level is appropriate for field use as well. The `-f` (field) argument invokes a somewhat abbreviated level of testing and may be used when test time must be kept to a minimum. The full manufacturing test suite is fast enough, however, to be suitable for nearly all diagnostic situations.

`+E` enables the environment condition represented by its accompanying flag. `-E` is used to disable the same set of environment variables. These variables are explained below.

`b`	Break on error (default)
`d#`	Display number (#) of errors. Default is 16. If Log–errors is also set, this flag controls how many errors are logged.
`f`	loop Forever
`i`	Ignore errors
`l`	Log errors
`t`	Trace option—this is intended for debugging activities in manufacturing; it is not ordinarily used in the field.

Repeat the `+E` or `-E` switch for each variable, separating them with spaces. The following example illustrates a command line that enables command completion, specifies 20 as the number of errors to display, and enables error logging. The same command disables the break-on-error condition.

```
dvdiag -C +Ed20 +El -Eb
```

B.4.2 Functional Test Groups

You can invoke a specific subgroup of tests within the `-f` suite by specifying the `-g`*groupname* argument, where *groupname* is the name of a predefined group of tests. The test groups currently available in `dvdiag` are summarized below.

- **DCP** — This group exercises hardware functions on the DCP board.
- **DVI** — This group exercises hardware functions on the DVI boards.
- **SCSI** — This group runs a set of tests specific to the SCSI boards.
- **DV** — This group tests connectivity between the DCP, DVI, DP, and SCSI boards. It also runs drive sparing and ECC logic tests for the DP board.
- **DVX** — This test reads and writes 1K buffers of data at full speed.

For example, to run the field program's DVI test group, type:

```
% dvdiag -f -gDVI
```

B.4.3 Invoking Individual Tests

If you enter **dvdiag** without the **-f** (or **-m**) argument, you invoke the **dvdiag** command interpreter, which is represented by the <DV-DIAG> prompt. At this prompt, you can explicitly invoke any individual tests or test groups that are contained in **dvdiag**. The syntax for operating in this mode is:

```
dvdiag -C +Ebdfilt -Ebdfilt
```

The **-C** argument is the command completion option. It allows you to enter abbreviated commands at the <DV-DIAG> prompt. Instead of typing the full command, just enter enough characters to distinguish the command from all others and then press **Escape**. When the command completion facility finishes the full name, press **Return** to enter it.

The **+E** and **-E** arguments function the same as described in Section B.4.1.

Enter the names of the tests you want to run at the <DV-DIAG> prompt. When the test completes, **dvdiag** returns you to the <DV-DIAG> prompt. For example, to test the DataVault's parity generator, enter:

```
<DV-DIAG> test-dcp-parity-generation
<DV-DIAG>
```

In addition to the individual test commands, the **dvdiag** command interpreter recognizes a number of other commands, which invoke various auxilliary functions and utilities. They are described in Appendix B of the DataVault Installation and Service Manual.

B.5 CM-IOPG Internal Diagnostics

viodiag is a package of functional tests that exercise the CM-IOPG internal hardware. It resides in **/usr/local/etc** on the CM-IOPG station manager.

viodiag's user interface closely resembles that of **dvdiag**. That is, you can invoke a complete test suite, functional test groups, or individual tests. The command syntax for using **viodiag** is as follows.

viodiag -m -f -g*groupname* **-i -C -Efihbld +Efihbld**
-s*filename*

-m	Run Manufacturing diagnostic tests for VMEIO device.
-f	Run Field diagnostic tests for VMEIO device.
-g*groupname*	Run tests for group specified by *groupname.*
-i	Include Interactive tests.
-C	Allow command Completion within **viodiag**.
+E	Set diagnostic Environment (activate options): **f** = Loop forever **i** = Ignore errors **h** = Halt on error **b** = Break on error (default) **l** = Log errors (default) d# = Display error count (default=16)
-E	Set diagnostic Environment (deactivate options).
-s*filename*	Execute a **viodiag** shell file given by *filename*

As with **dvdiag**, **viodiag** offers a comprehensive test suite that is invoked by:

```
% viodiag -f
```

The environment variables governing how the test suite behaves can be specified on the same command line. For example,

```
% viodiag -f +Ed20 +El -Eb
```

specifies 20 as the number of errors to display and enables error logging. The same command disables the break-on-error condition.

viodiag also provides a set of predefined functional test groups, which are invoked using the -g*groupname* argument. For example, the following tests the RAM FIFO flags.

```
% viodiag -f -gRAM-FIFO-FLAGS
```

The **viodiag** test groups are listed below.

```
CMIO-BUS-TEST
CMIO-INTERRUPTS
CMIO-FIFO-FLAGS
CMIO-PORT-LOOPBACK
CMIO-PARITY
INTERACTIVE
MASTER-STATUS
RAM-FIFO-FLAGS
RAM-PARITY
REGISTERS
SLAVE-FIFO-RAM
SLAVE-MAPPED-RAM
SLAVE-TIMEOUT
VME
VME-ADDRESS-GEN-TEST-MODE
VME-ADDRESS-GENERATOR
VME-INTERRUPT
VME-MASTER
VME-MASTER-TIMEOUT
VMEIO-CM-TRANSFERS
```

NOTE: In order to catch any errors reported by `viodiag`, you must have a window open on `viodiag` before you start the test. Then, if any test fails, go to the window and type **`show-board-status`**.

B.6 CM-IOPG Verifiers

There are two programs that can be used to verify the ability to move files between a CM-IOPG system and a DataVault on the same CMIO bus. These programs, `serial` and `TapeDVxfervfr`, reside in directory `/usr/local/etc` on the CM-IOPG. They are described in Sections B.6.1 and B.6.2.

B.6.1 serial

`serial` transfers test data between the CM-IOPG and the DataVault, comparing the data it reads with the data that it sent. It verifies the CMIO bus as well as major portions of the CM-IOPG and DataVault internal hardware.

The procedure for using `serial` is shown below.

1. Log in to the CM-IOPG station manager and set the DVWD environment variable to specify the DataVault.

   ```
   login: user_id
   Password: user_password
   % setenv DVWD datavault_hostname
   ```

   Use the hostname of the DataVault file server for *datavault_hostname*.

2. To invoke `serial`, enter:

   ```
   % /usr/local/etc/serial
   ```

B.6.2 TapeDVxfervfr

`TapeDVxfervfr` transfers test files between the CM-IOPG and the DataVault and between the CM-IOPG and the CM-TUD. It alternates between the DataVault and the CM-IOPG using various block sizes. While executing, the test will use system memory and VMEIO memory. It also uses both variable and fixed block mode access to the tape. The complete program provides a means for verifying the ability to transfer files between a CM-TUD and a DataVault.

Before initiating `TapeDVxfervfr`, verify that an appropriate tape cartridge is installed in the selected tape drive and that the DavaVault file name can be re-written if it already exists. If the file name does not exist, a new file will be created.

There are two user commands associated with this program, `TapeDVxfervfr` and `test_DV_to_Tape`. The first command invokes the program executive and the second starts the verifier program itself.

`test_DV_to_Tape` will prompt you for the following argument input.

Parameter	Expected Format	Description
tape drive	/dev/<drive-name>	Enter the hostname of the tape drive.
DV file	=dvault:/<path>/<file-name>	Enter the complete path to the DataVault file.
user_blksize	=Tape Block size	Enter the block size of the transfer.
vmeio_unit	=VMEIO unit number	Enter the unit number of the VMEIO module.

The procedure for using `TapeDVxfervfr` is shown below.

1. To initiate the program, enter:

```
% /usr/local/etc/TapeDVxfervfr
DIAGNOSTIC EXECUTIVE FOR DV/Tape Tests
State: RELEASE-6-1 Date: 91/08/13 11:57:43 State: RELEASE-6-1

<test_DV_to_Tape-DIAG>
```

2. At the prompt, enter:

```
<test_DV_to_Tape-DIAG> test_DV_to_Tape4
tape drive /dev/dv_hostname
DV file =dvault:/pathname/filename
```

```
user_blksize tape_block_size
vmeio_unit vmeio_unit_number
```

B.7 DM-HIPPI Internal Diagnostics

The CM-HIPPI provides a set of tests that allow you to diagnose its internal hardware functions. These tests are organized into four programs.

- **srcdiag** Tests source board functionality; see Section B.7.1.
- **destdiag** Tests destination board functionality; see Section B.7.2.
- **iopdiag** Tests iop board functionality; see Section B.7.3.
- **sysdiag** Tests the ability of the source board to send data and for the destination board to receive data (uses loopback cable). See Section B.7.4.

These tests are executed on the CM-HIPPI station manager. Before starting the tests, log in to the station manager as root and change directory to **/usr/local/etc/diag**.

```
login: user_id
% su
password: root_password
SU hostname /dev/console
# cd /usr/local/etc/diag
#
```

B.7.1 Source Board Functional Test

The following procedure tests Source board functionality.

1. First, ensure that the Source board is <u>not</u> cabled to any other device, such as a remote HIPPI device or to the Destination Board via a loopback cable.

2. Invoke the Source board command interpreter with the command completion flag.

```
# srcdiag -C
<srcdiag>
```

3. When you see the Source board diagnostic prompt, run the following tests.

```
<srcdiag> ktest 1   /* tests Source's AMD 29K internals*/
<srcdiag> score k   /* displays results */
<srcdiag> stest 1   /* tests Source's internals and*/
                    /* station manager interface*/
<srcdiag> score s   /* displays results*/
<srcdiag> quit
#
```

B.7.2 Destination Board Functional Test

The following procedure tests Destination board functionality.

1. First, ensure that the Destination board is <u>not</u> cabled to any other device, such as a remote HIPPI device, or to the Source board via a loopback cable.

2. Invoke the Destination board command interpreter.

```
# destdiag -C
<destdiag> ktest 1 /* tests Destination's AMD 29K*/
<destdiag> score k
<destdiag> dtest 1 /* tests Destination's internals and*/
                   /* station manager internals*/
<destdiag> score d
<destdiag> quit
#
```

B.7.3 IOP Board Functional Test

1. Ensure that no IOP boards are attached to a CMIO bus.

2. Invoke the IOP diagnostic interpreter with the -C flag and run the following tests.

```
# iopdiag -C
<iopdiag> select iop0    /* select first IOP */
```

```
<iopdiag> itest 1          /* test first IOP */
<iopdiag> score i          /* print results for first IOP */
<iopdiag> select iop1      /* repeat for each IOP */
<iopdiag> itest 1                       :
<iopdiag> score i          /* repeat for each IOP */
<iopdiag> quit
#
```

3. Repeat the **select iop**, **itest**, and **score** commands for each IOP. For each new **select iop**, simply change the number of the board to be selected. Remember, the boards are numbered 0–7 from right to left, as viewed from the front of the system.

B.7.4 System (Loopback) Test

The next step is to test the ability of the Source board to send data and the Destination board to receive data. This is done by connecting their OUT and IN ports on the CM-HIPPI bulkhead via a loopback cable.

1. Install the loopback cable between the OUT and IN ports on the CM-HIPPI bulkhead. **NOTE:** Do not install any CMIO bus cables yet.

2. From directory **/usr/local/etc/diag**, invoke the system command interpreter.

```
# sysdiag -C
<sysdiag>
```

3. When you are asked if you are attached to an I/O bus, answer no.

4. At the `<sysdiag>` prompt run the following test.

```
<sysdiag> etest 1
<sysdiag> score e
```

Appendix G

Troubleshooting Power Supply, Clock, and Diagnostic Network Faults

G.1 Introduction

(to be supplied)

Appendix H

Tracing I/O Errors

H.1 Introduction

This appendix describes three sets of diagnostic procedures for I/O-related hardware problems.

- Section H.2 presents the basic procedure for troubleshooting CM-5 I/O failures.
- Section H.3 describes a supplementary procedure for exercising an IOBA-to-DataVault connection. This procedure can be useful as a cross-check of other diagnostic results, particularly for elusive hardware problems that exhibit unusual failure modes.
- Section H.4 explains the procedure for using the system exercisers, **dvtest5** and **hippi-loop**. These programs are recommended for verifying overall system functionality after a hardware failure has been corrected and before the system is returned to regular service.

H.2 Basic I/O Troubleshooting Procedure

(to be supplied)

H.3 Verifying IOBA-to-DataVault Path

(to be supplied)

H.4 System Verifiers for DataVault and HIPPI Paths

(to be supplied)

Appendix I

hardware.install file

I.1 Introduction

/etc/cm/configuration/hardware.install describes the hardware configuration of the CM-5 system on which it resides. This file is created at the factory to reflect the state of the particular CM-5 as it will be installed.

When the system's hardware configuration is changed, you will need to edit the file to reflect the change; **hardware.install** must always match the configuration of the system.

NOTE: For one-to-one replacement of hardware modules, you do not need to update **hardware.install** because there is no net change to the hardware configuration.

Figure 31 illustrates a sample **hardware.install** file. Its contents are explained below.

NOTE: The numbers to the left of the shaded areas, and the shading itself, are not part of **hardware.install**. They have been added to aid in description of the file.

1
```
#define STANDARD_PN_BACKPLANE
	{
		PN			0-7
		PNMEM			0-7
		CLKDN			0
		CN			0
	}
#define STANDARD_DR_BACKPLANE
	{
		DR			0-15
		CLKBUF			0
	}
#define STANDARD_CN_BACKPLANE
	{
		CN			0-5
		CLKBUF			0
		CLKDN			0
	}
```

2
```
CM_System
{
```

3
```
	Name =		"Calliope"
```

4
```
	DR_Height =	5
```

5
```
	PN_Type
	{
		PN_Memory = 8Mb
		PN_IU =		"SPARC"
		PN_FPU =	"SPARC"
	}
```

6
```
	Partition_Manager	0
	{
		Hostname = "homer.think.com"
		Console
		Diagnostic_Processor
	}
	Partition_Manager	1
	{
		Hostname = "milton.think.com"
	}
```

(continued on next page)

Figure 31. **hardware.install** example — (page 1 of 3)

```
7       PN_Cabinet     0
        {
             PN_Backplane        0-7    STANDARD_PN_BACKPLANE
             DR_Backplane        8-9    STANDARD_DR_BACKPLANE
             CN_Backplane        10     STANDARD_CN_BACKPLANE
        }

8       PN_Cabinet     1
        {
             PN_Backplane    3
             {
                  SPI        0
                  SVME       0    { Partition_Manager = 0 }
                  SPI        4
                  SVME       4    { Partition_Manager = 1 }
                  FILLER     1-3
                  FILLER     5-7
                  CN         0
                  CLKDN 0
             }
             IO_Backplane    7
             {
                  IOBUF      1
                  IOBUF      2
                  IOCNTRL    0
                  IOCHNL     0
                  IODR       0
                  IOCLK      0
             }
             DR_Backplane    8-9
             {
             DR              0-15
             CLKBUF          0
             }
             CN_Backplane    10
             {
                  CN         0-5
                  CLKBUF     0
                  CLKDN      0
             }
        }
```

(continued on next page)

Figure 31. **hardware.install** example — (page 2 of 3)

```
9           DR_Cabinet        4096
            {
                  DR_Backplane      0
                  {
                        DR5         0-15
                        CLKBUF      0
                  }
                  DR_Backplane      1
                  {
                        DR5         0-15
                        CLKBUF      0
                  }
                  CN_Backplane      4
                  {
                        CN          0-1
                        CN          4-5
                        CLKBUF      0
                        CLKDN       0
                        CLKDN       1
                  }
            }
      }
```

Figure 31. **hardware.install** example — (page 3 of 3)

I.2 **File Header** (shaded area 1)

The first area in Figure 31 contains file header information only. You needn't ever edit this content.

I.3 **CM System** (shaded area 2)

This line introduces the balance of the file, which describes the physical composition of the system. The general organization of the system description consists of :

- General system attributes — system name, DR height, and processing node type (Sections I.4 through I.6).
- Individual descriptions of partition managers (Sections I.7 and I.8).
- Individual descriptions of device cabinets and network cabinet (Sections I.9 and I.10).

I.4 **System Name** (shaded area 3)

This line specifies an arbitrary name for the system. In this example, the system name is "Calliope."

I.5 **DR Height** (shaded area 4)

This line indicates the highest — or root — level of the data network. In this example, the highest level is 5.

I.6 **PN Type** (shaded area 5)

These lines describe the system's processing nodes in terms of the following attributes.

- PN Memory — Specifies the memory capacity per PN. Currently, this attribute can be either 8 Mbytes or 32 Mbytes. In this case, it is 8 Mbytes.
- PN IU — Specifies the type of integer unit in use. Currently, SPARC is the only valid entry.
- PN FPU — Specifies the type of floating point unit in use. Currently, SPARC is the only valid entry.

I.7 **Partition Managers** (shaded area 6)

This section describes each partition manager (PM) in the system. There are two PMs in this example.

Each PM is identified by an integer, which is arbitrarily chosen to distinguish that PM from all others in the system. In this example, the PMs are designated 0 and 1. This designation is followed by a list of attributes. The PM 0 attributes are defined below.

- Hostname — This is a quoted string that gives the PM a name that may be easier to remember than its integer designation. In this example, PM 0 is named `"homer.think.com."`
- Console — This entry indicates that PM 0 serves as the system administration console.
- Diagnostic Processor — This entry indicates that the diagnostic server, **`jtagserver`**, runs on this PM.

PM 1 serves no other role than partition manager. Consequently, its only attribute entry is its hostname, which in this example is `"milton.think.com."`

I.8 **PN Cabinet 0** (shaded area 7)

This section describes the composition of a single device cabinet. The first item of description is an integer that uniquely identifies this cabinet in a multi-cabinet system. By convention, this integer indicates the cabinet's physical position in relation to other device cabinets in the system. Figure 32 shows the cabinet numbering scheme used for CM-5 systems of up to 2 K network addresses.

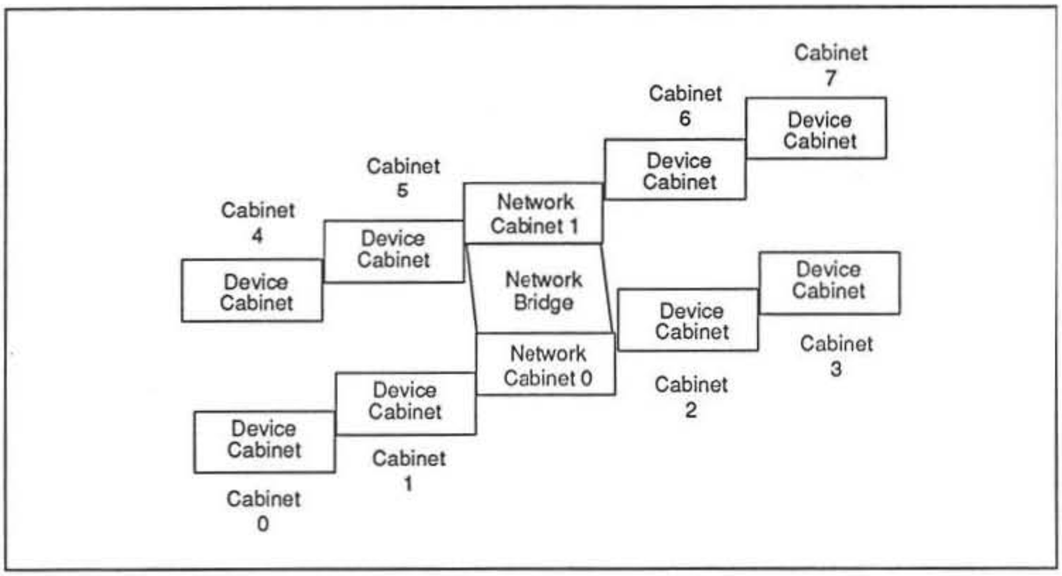


Figure 32. CM-5 device cabinet numbering system.

NOTE: Where `hardware.install` refers to PN cabinets, understand that it means device cabinets. The term PN (processing node) cabinets is a historical artifact. Likewise, you should translate references to DR cabinets to network cabinets.

The cabinet contents are then listed by backplane. Figure 33 shows how backplanes are numbered in a device cabinet, and Figure 34 through Figure 36 show the slot configurations of the standard PN, DR, and CN backplanes.

In this example, `hardware.install` shows cabinet 1 to have the following backplane configuration.

- PN Backplanes 0–7 — All eight PN backplanes contain the standard PN configuration of circuit modules. Consequently, a detailed breakdown of the backplane contents is not given. This standard configuration includes eight PN circuit modules, plus a CN module and a CLKDN module.

- DR Backplanes 8–9 — These backplanes contain circuit modules that form the uppermost levels of the device cabinet's data network. In systems with multiple cabinets, these backplanes are connected by cable to the network cabinet.
- CN Backplane 10 — The control network backplane contains circuit modules that form the uppermost levels of the device cabinet's control network. In systems with multiple cabinets, this backplane is connected by cable to the network cabinet.

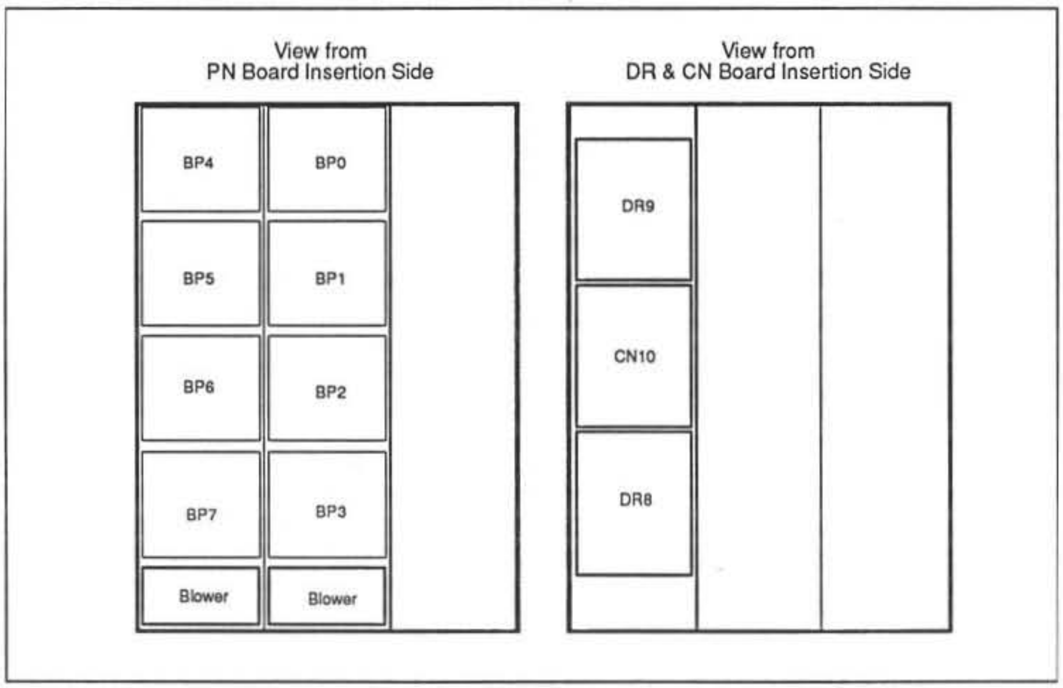


Figure 33. CM-5 device cabinet backplane numbering

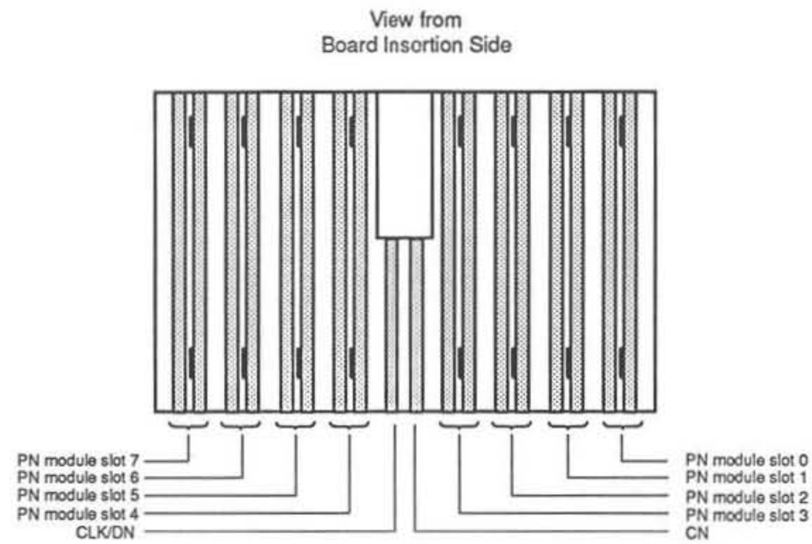


Figure 34. Standard PN backplane slot assignments

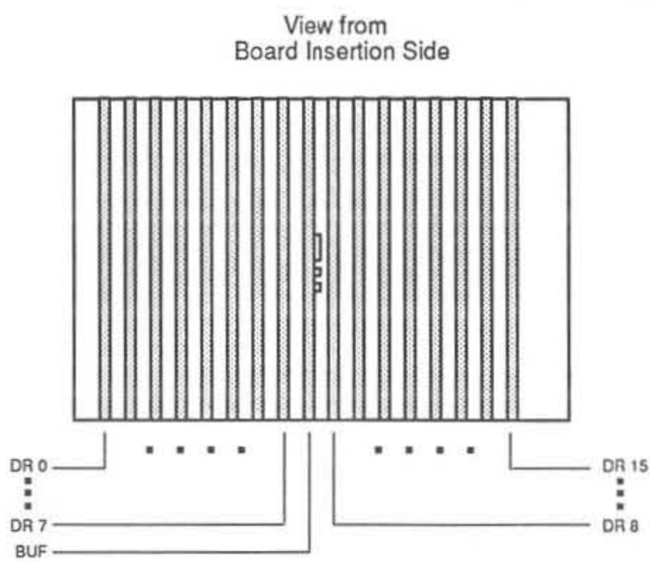


Figure 35. Device cabinet DR backplane slot assignments
(backplanes 8 and 9)

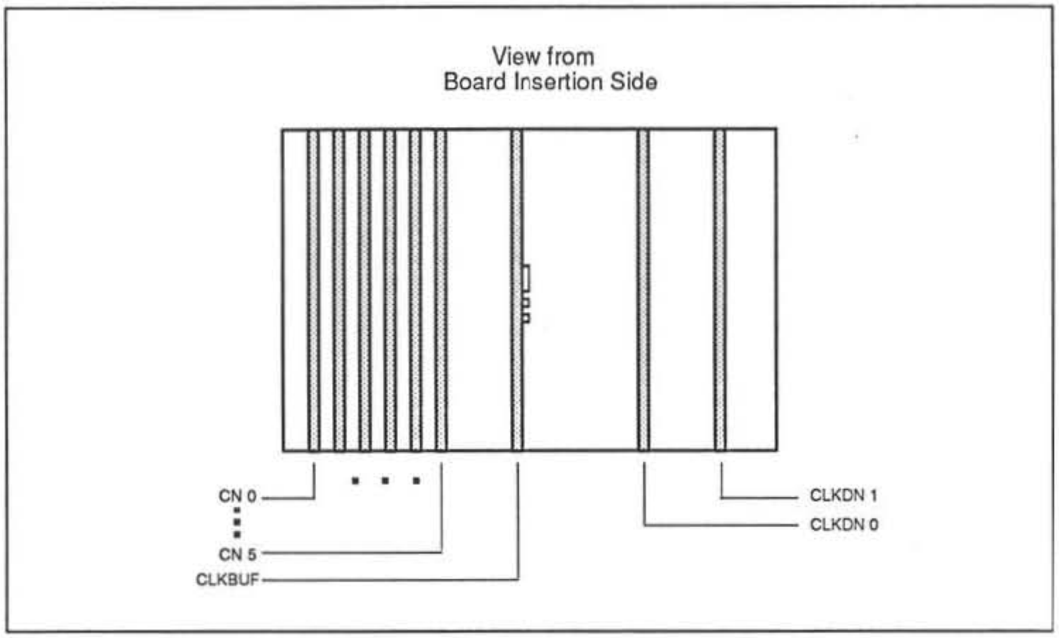


Figure 36. Device cabinet CN backplane slot assignments (backplane 10)

I.9 PN Cabinet 1 (shaded area 8)

In this sample file, a second device cabinet, cabinet 2, contains interfaces to the partition managers as well as an I/O interface. The various backplane configurations in cabinet 2 are described separately in Sections I.9.1 through I.9.4.

I.9.1 PN Backplane 3

A single standard PN backplane, backplane 3, contains both PM interfaces as well as auxiliary circuit modules. No other standard PN backplanes are used in this cabinet.

The circuit modules that fill the backplane slots are summarized below, with the slot location(s) indicated to the right of each circuit module type.

- SPI 0 The SPI in slot 0 is the interface to partition manager 0.

▪ SVME	0	This entry associates an SVME module with the SPI in slot 0.
▪ SPI	4	The SPI in slot 4 is the interface to partition manager 1.
▪ SVME	4	This entry associates an SVME module with the SPI in slot 4.
▪ FILLER	1–3	These three slots contain circuit modules that fill the gap in the network that would otherwise occur when a backplane is not fully populated with functional network devices, such as PNs.
▪ FILLER	5–7	Same as FILLER 1–3.
▪ CN	0	This slot contains a portion of the control network.
▪ CLKDN	0	This slot contains the backplane's interface to the clock and diagnostic networks.

I.9.2 I/O Backplane 7

This backplane contains a set of circuit modules that together form the interface to a CMIO bus and one or more peripherals attached to the bus. These peripherals can include DataVaults, CM-HIPPIs, and/or CM-IOPGs.

The I/O backplane and its constituent circuit modules are referred to as an IOBA, (Input Output Bus Adapter). Figure 37 illustrates the slot organization of an IOBA chassis. A standard IOBA configuration contains six circuit modules; their `hardware.install` entries are summarized below.

▪ IOBUF	1	This entry indicates that slot 1 contains one IOBUF module.
▪ IOBUF	2	Slot 2 contains the second IOBUF module.
▪ IOCNTRL	0	Each IOBA has one I/O control module; it is always identified by the label 0.

▪ IOCHNL	0	This line indicates that an I/O channel is provided in slot 0.
▪ IODR	0	A standard IOBA has one data network interface module; it is always identified by the label 0.
▪ IOCLK	0	Each IOBA has one clock interface module; it is always identified by the label 0.

Another file, `io.conf`, contains additional I/O configuration information. It defines various attributes concerning the components connected to the CMIO bus, including this IOBA, that are of interest to the fileserver. If the IOBA or CMIO bus are modified in any way that affects these attributes, `io.conf` must be updated as well. Appendix J describes `io.conf` in detail.

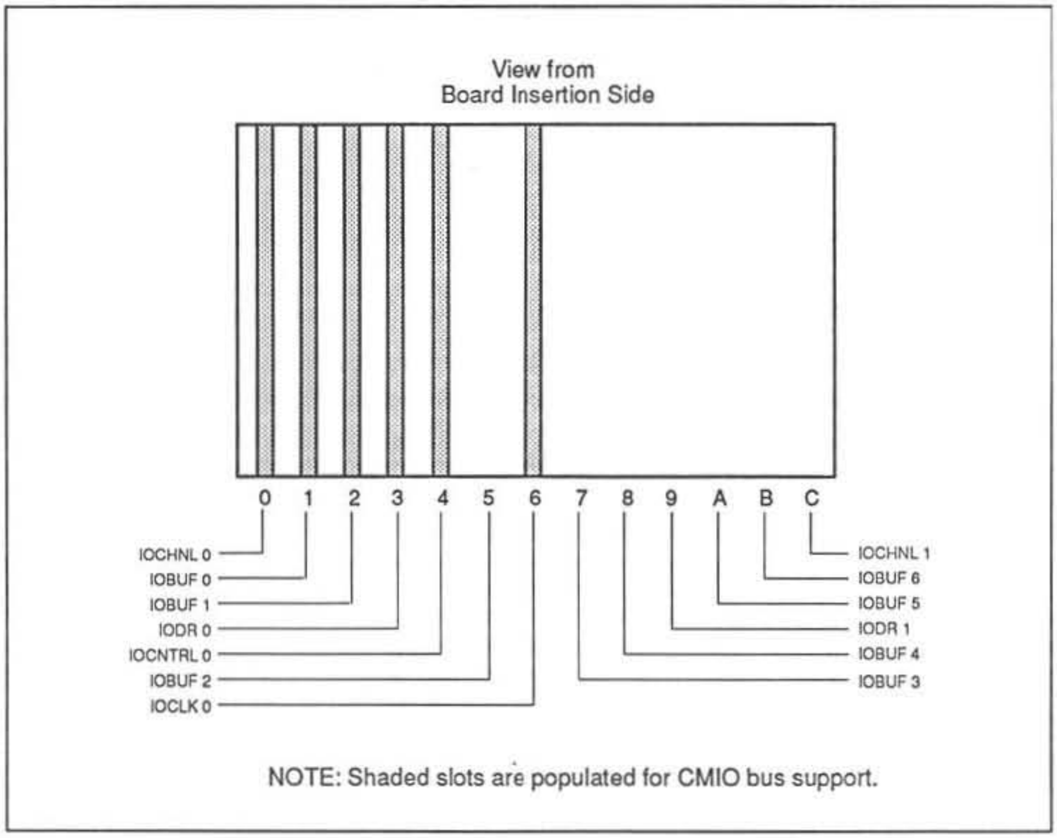


Figure 37. IOBA backplane slot assignments with standard board configuration.

I.9.3 DR Backplane 8–9

The data network backplanes 8 and 9 contain:

▪ DR	0–15	These backplanes contain 16 DR circuit modules, which provide the link among all data network components residing in this cabinet. In multi-cabinet systems, they also form the interface to the higher levels of the data network in the network cabinet.
▪ CLKBUF	0	This module buffers and distributes system clocks to the Data Network boards.

I.9.4 CN Backplane 10

The control network backplane, backplane 10, contains: 6 CN circuit modules and a CLKDN circuit module.

▪ CN	0–5	This backplane contains 6 CN circuit modules, which provide the link among all control network components residing in this cabinet. In multi-cabinet systems, they also form the interface to the higher levels of the control network in the network cabinet.
▪ CLKBUF	0	This module receives the system clock signal from the CLKDN board and drives it out to the Control Network boards.
▪ CLKDN	0	In systems with two or more device cabinets (greater than 256 network addresses), this module is the interface to the clock and diagnostic networks residing in the network cabinet. In single-device-cabinet systems, this module serves as the system clock and diagnostic network root.

I.10 DR Cabinet (shaded area 9)

The network cabinet contains the data and control network modules that form the uppermost levels of their respective trees. The first entry in this section is an identifier for this cabinet—a large integer that will distinguish this network cabinet from all other cabinets in the system. By convention, 4096 is used as the identifier for the first network cabinet in the system.

NOTE: Except for the requirement that this number be large enough to exceed the highest possible device cabinet number, its value has no specific meaning.

In the network cabinet, the DR and CN backplanes are in the center section of the cabinet — the space occupied by backplanes 0–3 in a device cabinet. Figure 38 shows the location of these backplanes. Figure 39 and Figure 40 show the DR and CN slot assignments in each.

The `hardware.install` entries representing these backplanes are summarized below.

- DR 0 — This backplane contains 16 DR circuit modules and a CLKBUF circuit module.
- DR 1 — This backplane contains 16 DR circuit modules and a CLKBUF circuit module.
- CN 4 — This backplane contains 6 CN circuit modules, one CLKBUF circuit module, and two CLKDN circuit modules.

NOTE: Although the sample `hardware.install` file represents only a single network cabinet with only two DR backplanes populated, Figure 39 and Figure 40 illustrate the backplane slot assignments for systems with two network cabinets and height 6 DR and CN modules.

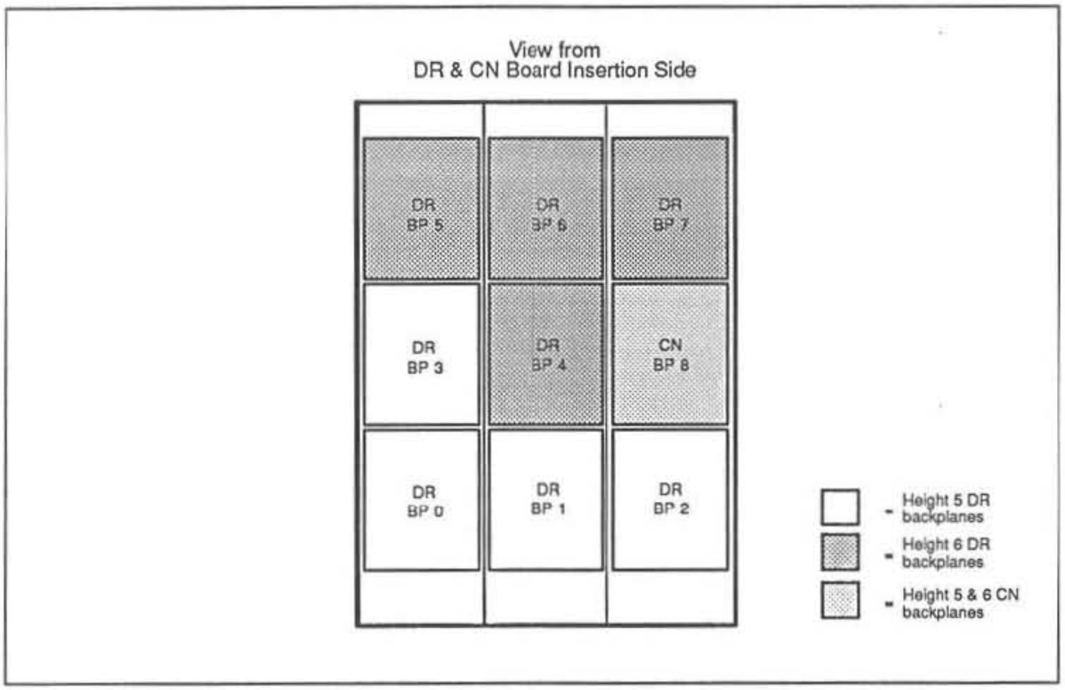


Figure 38. CM-5 network cabinets 0 and 1 backplane numbering

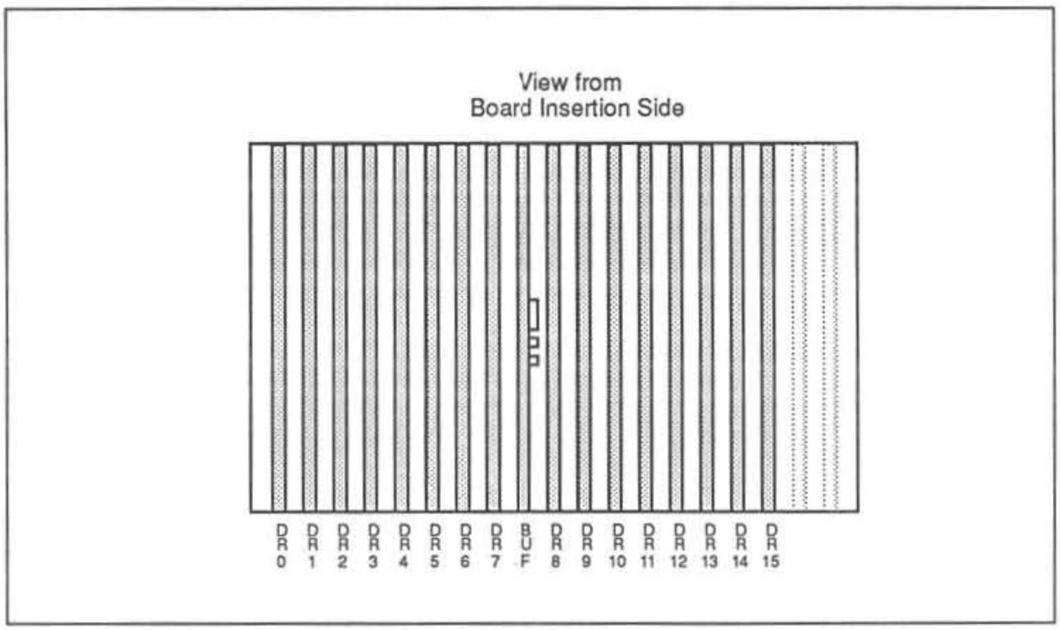


Figure 39. DR backplane slot assignments for network cabinets 0 and 1

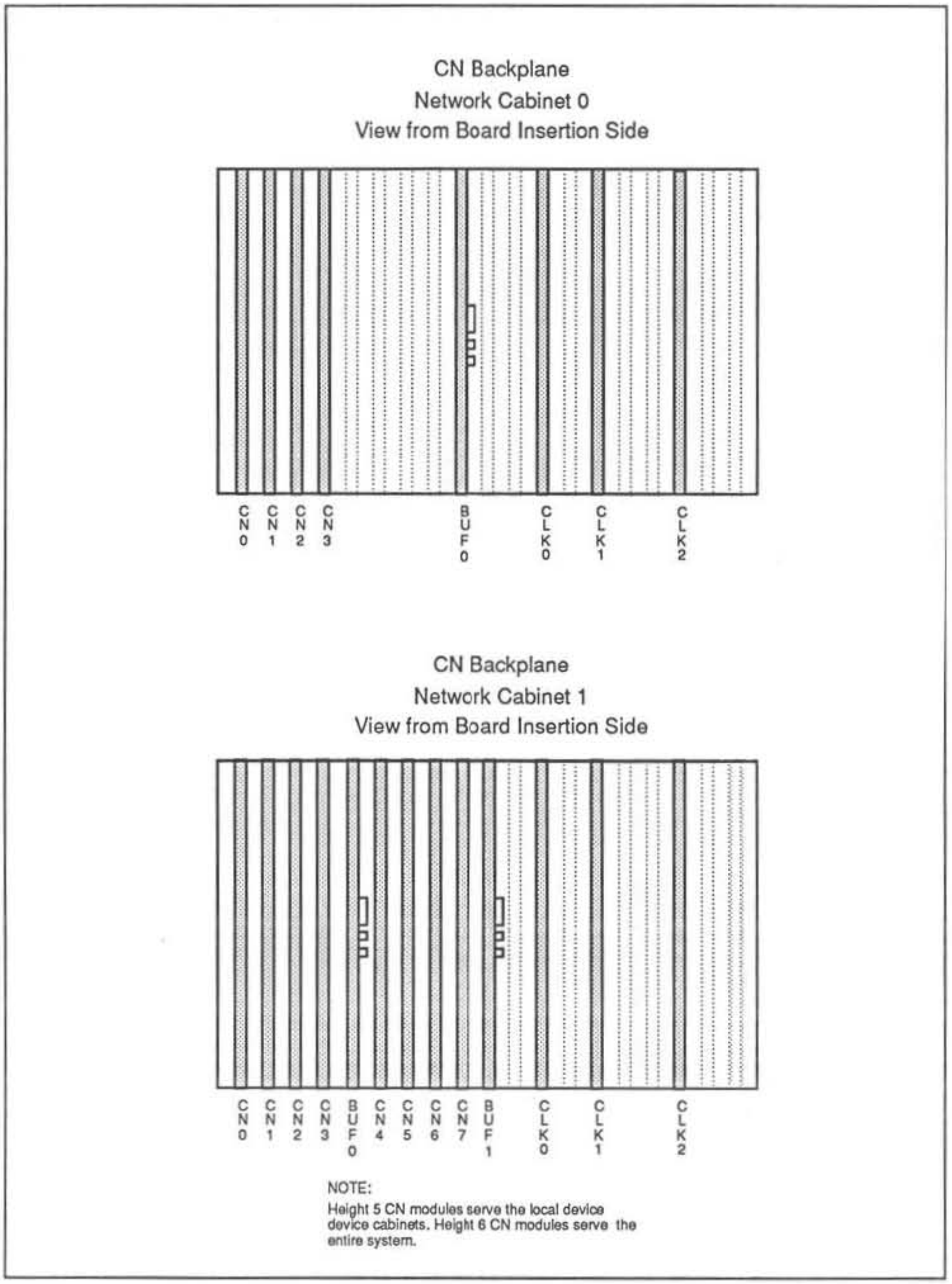


Figure 40. CN backplane slot assignments for network cabinets 0 and 1

Appendix J

io.conf file

The file `/etc/io.conf` is the CM-5 I/O system's configuration file. It must always accurately reflect the state of the I/O system. `io.conf` is created by a Thinking Machines Customer Support representative when the I/O system is installed. Thereafter, `io.conf` must be updated whenever the I/O system is reconfigured. This section explains the components of `io.conf` so that you can edit them if the system's configuration changes.

`io.conf` is an ASCII file. As such, these general-format rules apply:

- Numeric arguments can be specified in hex (leading `0x` or `0X`), octal (leading `0`), or decimal.
- Characters following a semicolon on the same line are ignored.
- As long as all entries are left-justified, `io.conf` can contain any amount of white space.
- The parser is case-sensitive; all alpha-text must be typed into `io.conf` exactly as shown in this section.

Some of the entries in `io.conf` require you to count hardware entities. Count the first entity as number 1, not number 0.

Figure 41 illustrates an I/O system with two IOBAs, two DataVaults, a CM-HIPPI, a CM-IOPG, and a CM-TUD. Figure 42 represents the `io.conf` file for the configuration shown in Figure 41.

The rest of `io.conf` consists of two main modules:

- The `Channel_Board_Configuration` module contains information about the IOBAs. Section J.1 describes this module in detail.
- The `IO_device_configuration` module describes the I/O devices configured into the system. Section J.2 describes this module in detail.

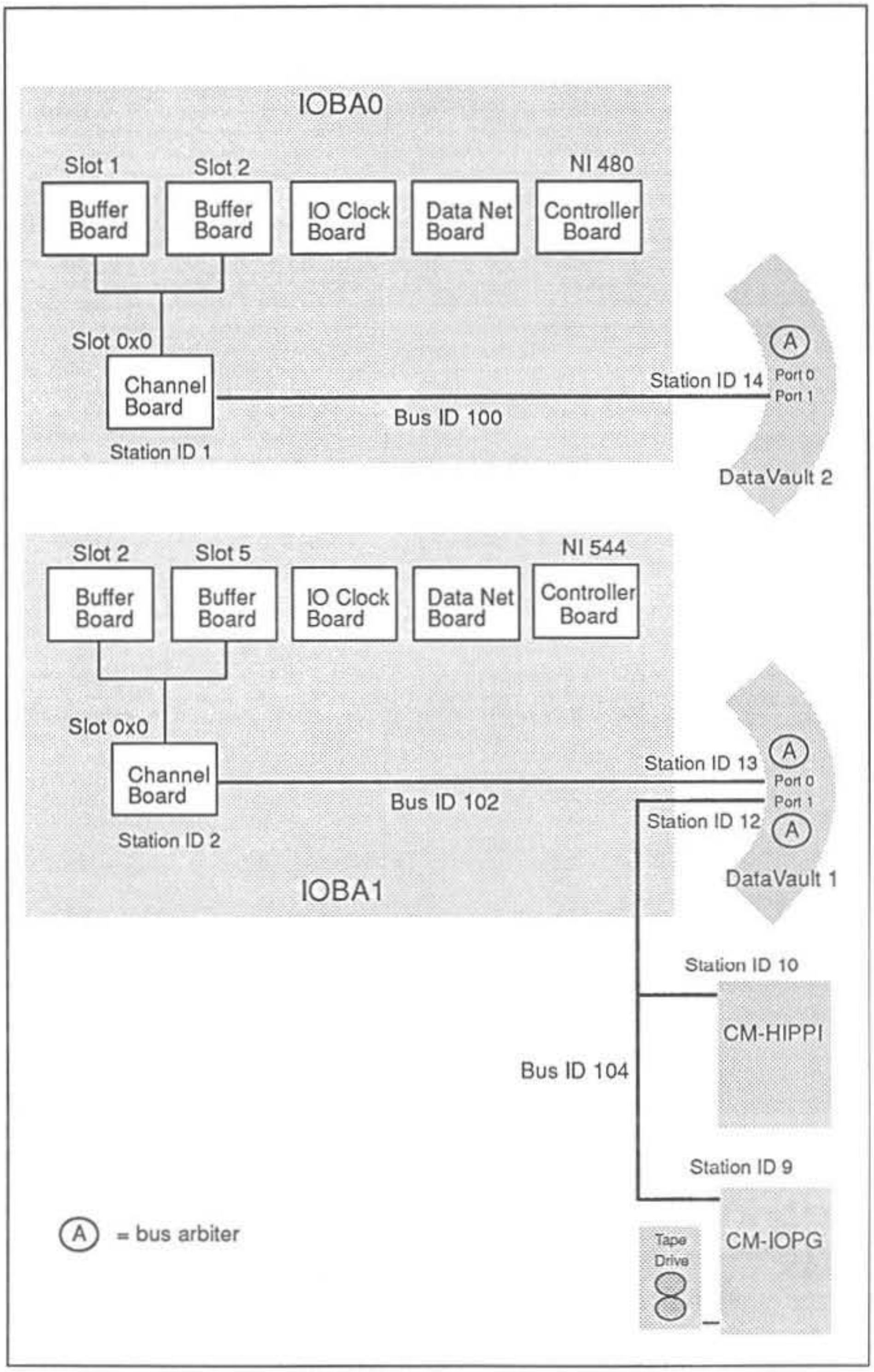


Figure 41. Sample CM-5 I/O system configuration.

```
IO_configuration_file_v1.0 ;Identification string

Channel_Board_Configuration
2               ; Total number of channel boards in
                ; all IOBAs combined.

                ; Channel board in IOBA0
480             ; IOPN NI physical address
0x0             ; Channel board slot number
100             ; CMIO bus id
1               ; Station id
0               ; CMIO arbiter status flag
2               ; CMIO bus speed
0x1             ; Buffer board slot number
                ;     (Must be one of 1,2,5,7,8,a,b)
0x2             ; Buffer board slot number
                ;     (Must be one of 1,2,5,7,8,a,b)

                ; Channel board in IOBA1
544             ; IOPN NI physical address
0x0             ; Channel board slot number
102             ; CMIO bus id
2               ; Station id
0               ; CMIO arbiter status flag
2               ; CMIO bus speed
0x2             ; Buffer board slot number
                ;     (Must be one of 1,2,5,7,8,a,b)
0x5             ; Buffer board slot number
                ;     (Must be one of 1,2,5,7,8,a,b)
```

— continued —

Figure 42. Sample **io.conf** for the I/O system diagrammed in Figure 41.

```
IO_device_configuration
5                  ; Number of IO devices in system

dv2                ; hostname of IO device 1 (Also
                   ; default host name)
DV                 ; type of IO device 1
14                 ; station id of IO device 1
100                ; bus id of IO device 1

dv1                ; hostname of IO device 2
DV                 ; type of IO device 2
13                 ; station id of IO device 2
102                ; bus id of IO device 2

dv1                ; hostname of IO device 3
DV                 ; type of IO device 3
12                 ; station id of IO device 3
104                ; bus id of IO device 3

hiocl              ; hostname of IO device 4
HIPPI              ; type of IO device 4
10                 ; station id of IO device 4
104                ; bus id of IO device 4

iopgl              ; hostname of IO device 5
VME                ; type of IO device 5
9                  ; station id of IO device 5
104                ; bus id of IO device 5
```

Figure 42, continued. Sample **io.conf** for the I/O system diagrammed in Figure 41.

J.1 The Channel_Board_Configuration Module

io.conf must contain exactly one **Channel_Board_Configuration** module, which describes the IOBAs. The **Channel_Board_Configuration** module is comprised of submodules that describe the IOBAs' channel boards.

The first line of the **Channel_Board_Configuration** module specifies the total number of channel boards in the system. Following this line are one or more submodules: one submodule for each channel board. Each submodule must contain eight lines, in this order:

- The first line specifies the physical address of the NI on the controller board in the same IOBA as the channel board.
- The second line specifies the slot number of the channel board.
- The third line specifies the bus ID of the CMIO bus to which the channel board is connected.
- The fourth line specifies the station ID of the channel board.
- The fifth line specifies whether the channel board is the bus arbiter (`1`) or not (`0`).
- The sixth line specifies the CMIO bus's speed. This value is always `2`.
- The seventh and eighth line each specify the slot number of one of the two buffer boards associated with the channel board. It does not matter which board's slot number is listed on the seventh line and which is listed on the eighth.

The ordering of the submodules is arbitrary. That is, if there is more than one IOBA in the system, you need not place the submodule for IOBA0's channel board before the submodule for IOBA1's channel board, although it is conventional to do so.

J.2 The IO_device_configuration Module

`io.conf` must contain exactly one `IO_device_configuration` module. Its submodules describe each I/O device — DataVault *port*, CM-IOPG, and CM-HIPPI — configured into the system.

The first line of the `I/O_device_configuration` module specifies how many I/O devices are in the system. Be sure to count each configured DataVault *port* as a separate device.

Each device must be described by exactly one submodule. Each submodule must contain four lines, in the order listed:

- The first line specifies the hostname of the device.

- The second line specifies a code that indicates the type of the device:
    - `DV` indicates a DataVault port.
    - `VME` indicates a CM-IOPG.
    - `HIPPI` indicates a CM-HIPPI.
- The third line specifies the device's station ID.
- The fourth line specifies the bus ID of the CMIO bus on which the device resides.

The ordering of the submodules is arbitrary except as it is used by the CMFS file system and I/O diagnostics to determine the default I/O device. The system determines the default device by searching **`io.conf`** for the first channel board that has at least one I/O device on its bus, and then, if there is more than one device, choosing the one that appears first in the **`IO_device_configuration`** module.

Appendix K

Error Reporting System

K.1 Overview

The CM-5 error reporting system provides useful information about failures disclosed by **`cmdiag`** tests. When a diagnostic routine finds a hardware fault, the error system parses the error status of all visible components in the partition under test and, upon request, reports its findings.

This report provides a summary description of each test failure and identifies which module (circuit board) and individual components are implicated in the failure.

Error messages are logged in **`diag-error-log.`*`hostname`*** in the local directory on the associated Partition Manager. ***`hostname`*** is the name given to the Partition Manager.

NOTE: This discussion assumes that diagnostics are being run on a partition rather than the entire CM. The description applies equally to a partition that encompasses all of the Processing Nodes in the system.

The command to read the error system report on line is **`find-cm-error`**.

```
<CM-DIAG> find-cm-error
```

The error reporting system responds to this command by displaying the contents of **`diag-error-log.`*`hostname`***. Alternatively, you can read the error log directly in a gmacs buffer or output it to a printer.

K.2 Interpreting Error System Reports

Figure 43 shows examples of the types of error messages to be found in `diag-error-log.`*`hostname`*. The basic format is the same for all messages, regardless of the type of error being reported or the type of hardware associated with the error.

Section K.2.1 discusses the contents of the first error message example shown in Figure 43, a Control Network error. It also explains the various features of the message format that are common to all message types. Sections K.2.2 through J.2.?? discuss other message types, with particular emphasis on their special characteristics.

```
Global  Address {Cabinet 0 Backplane 0 Slot 0} Type CN
Network Address {CN_NODE Height 2 Leaf 1 Root 0}
  ID_Prom [TYPE 03 REV 00 ID# 00ac] Pod_Type CN
Chip_Name CN-1 Chip_Type FEDEX IR_Scan 1000010101000000001
  Register NODE-ESTAT-0 1011111101111111111
    Bit  1. NODE-0-UP-TYPE
    Bit  8. NODE-0-UP-HARD-ERROR
  Register NODE-ESTAT-1 1011111101111111111
    Bit  1. NODE-1-UP-TYPE
    Bit  8. NODE-1-UP-HARD-ERROR
  Register NODE-ESTAT-2 1011111101111111111
    Bit  1. NODE-2-UP-TYPE
    Bit  8. NODE-2-UP-HARD-ERROR
```

Figure 43. `diag-error-log` example.

K.2.1 Control Network Error Example

The first four lines of all error messages are nearly identical. Their contents are summarized below, using Figure 43 for reference.

- The first line specifies the physical address of the hardware reporting the error. For example, the first error shown in Figure 43 was detected in slot 0 of backplane 0 in cabinet 0. The module occupying that location is a CN board.

```
Global Address {Cabinet 0 Backplane 0 Slot 0} Type CN
```

- The second line gives the network address of the error. In the first example, the error was reported from the Control Network node at height 2, leaf 1 attached to root 0.

  ```
  Network Address {CN_Node Height 2 Leaf 1 Root 0}
  ```

  With this information, you can find this node in the CN topology chart and understand its place in the failed operation's CN communication path.

- The third line identifies the ID prom of the module that reported the error. In the first error example, the ID prom is of type 03, revision level 00 and has the hexadecimal tag 00ac. It resides on a CN module (this last piece of information is redundant with the first line).

  ```
  ID_Prom [TYPE 03 REV 00 ID# 00ac] Pod_Type CN
  ```

  Note, this information is intended primarily for long-term tracking of hardware failure patterns. It has no relevance to troubleshooting.

- The fourth line identifies the individual chip that is most closely associated with the error report. In the first example, CN chip 1 of chip type FEDEX is called out and its instruction register contents are scanned out.

  ```
  Chip_Name CN_1 Chip_Type FEDEX IR_Scan 1000010101000000001
  ```

  IR bit 0 is leftmost.

- The remaining lines describe the error itself, displaying the contents of each relevant error status register and explaining the meaning of each relevant status bit. These lines will vary most from message to message, depending on the type of error and the type of hardware reporting it. These lines are discussed more fully below for the CN error example. Sections J.2.3 through J.2.?? explain these lines for other error types.

The error example in Figure 43 shows a CN node reporting an error. This node has three status registers, 0, 1, and 2. NO TAG shows where these registers reside in the node and how they relate to adjacent nodes in the network.

```
Register NODE-ESTAT-0 10111111011111111
   Bit 1. NODE-0-UP-TYPE
   Bit 8. NODE-0-UP-HARD-ERROR
Register NODE-ESTAT-1 10111111011111111
   Bit 1. NODE-0-UP-TYPE
   Bit 8. NODE-0-UP-HARD-ERROR
Register NODE-ESTAT-2 10111111011111111
```

```
Bit 1. NODE-0-UP-TYPE
Bit 8. NODE-0-UP-HARD-ERROR
```

Bit 1 of each register indicates that it received a faulty message (CRC value was invalid) from a node lower in the tree (from a child). Bit 8 indicates that this is the first node in the message path to detect the error. As this faulty message is propagated further through the CN, nodes subsequent to this one will set their soft error bits.

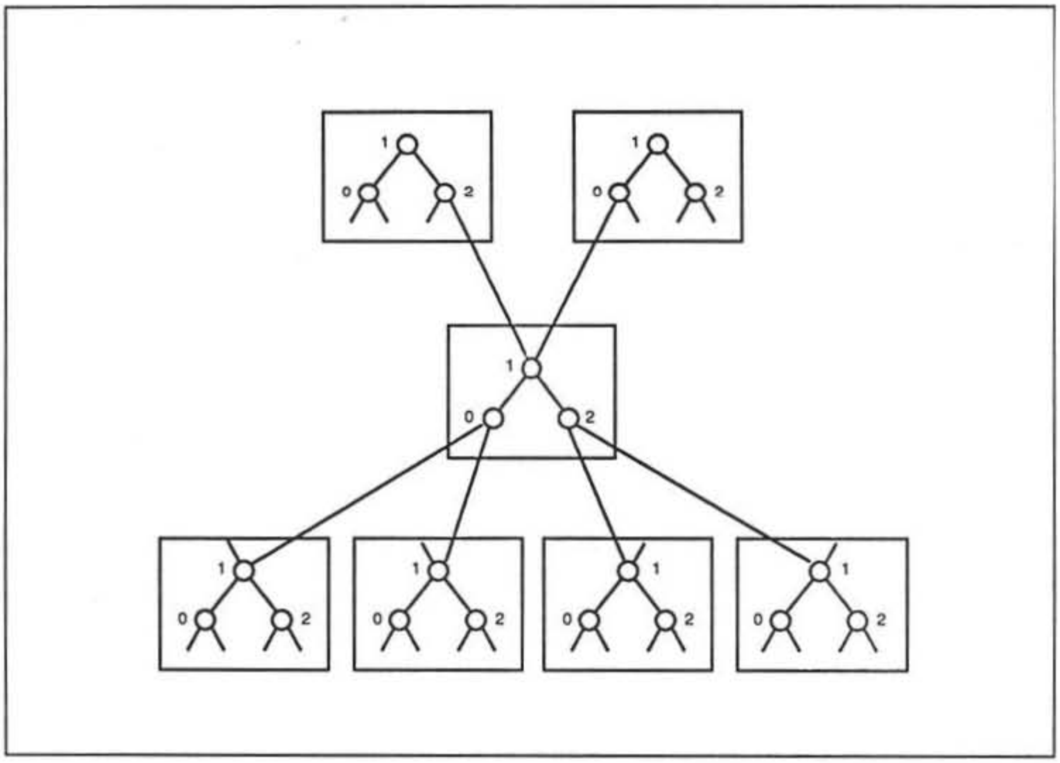


Figure 44. CN node configuration.

K.2.2 Data Network Error Example

(to be supplied)

Appendix E

dvtest5 Description

dvtest5

dvtest5, dvtest5-sparc, dvtest5-vu — User–level verifier for SDA File Systems (SFS), IOBA File Systems (CMFS), and supporting hardware.

Syntax

dvtest5-sparc | dvtest5-vu [**-x**] [**-t**] [**-1**] [**-g***int1*, ... *intn*] [**-d** *directory-name*] {[**-a**[**1**] | **-s** | **-h** | [**-l** *testname* [*testname*] ...]]}

-x	Exit on error.
-t	Report tersely.
-1	Run selected test(s) one time only, rather than looping forever.
-g *int1, ... intn*	Specify a geometry to be applied to the data being transferred, using a string of one or more integers separated by commas.
-d *directory_name*	Causes **dvtest5** | **dvtest5-vu** to change directory to directory_name before starting.
-a[**1**]	Run all tests automatically (no menu). This is the most thorough exerciser. If **-a1** is specified, the tests run once. Otherwise htey run forever; stop by executing Ctrl-C.

`-s`	Run software test subset automatically (no menu). `-s` is generally used only when new software has been installed.
`-h`	Run hardware test subset automatically (no menu). `-h` is generally used during preventive maintenance.
`-l` *testname(s)*	Run the tests specified by testname. See the menu illustration below.

1.basics	test file creation/deletion (–s)
2.data	test simple file read/write (–s, –h)
3.write	test writing files (–s, –h)
4.link	test link/unlink (–s)
5.abs_seek	test absolute seek (random) (–s, –h)
6.rel_seek	test relative seek (deterministic) (–s, –h)
7.dir_basics	test mkdir/rmdir/chdir (–s)
8.many_dirs	test creating many directories (–s)
9.many_files	test creating/deleting many files (–s)
10.serial_io	test serial I/O transfers (–s)
11.mixed_io	test mixed serial and parallel I/O (–s, –h)
12.parallel_partial	test parallel I/O with partial blocks (–s, –h)
13.transpose	test transposing serial data (–s)
14.transfer_timing	test transfer speed (–s, –h)
15.raw_transfer_timing	test raw transfer speed (–s, –h)
16.overhead_timing	test overhead speed (–s, –h)
17.max_transfer_timing	test max transfer speed
18.reliability	test reliability (–h)

Description

NOTE: **`dvtest5`** has been replaced by **`dvtest5-sparc`** (for non–vector–unit CM–5 systems) and **`dvtest5-vu`** (for CM–5 systems that have vector units).

The **`dvtest5-sparc | dvtest5-vu`** program is an acceptance test that uses either an SDA system or an IOBA (CMIO bus adapter) and a CMFS device to perform all I/O functions available to user applications. Among other things, these programs test every I/O data and control path, check Ethernet connections, and open files and directories (in directory **`dvtest`**). **`dvtest5-sparc | dvtest5-vu`** determine which device to use according to the setting of CMFS_PATHTYPE.

- If CMFS_PATHTYPE is set to unix, **dvtest5-sparc | dvtest5-vu** uses the local UNIX or UNIX–compatible file system — the SDA, if the CM–5 system contains one.

- If CMFS_PATHTYPE is set to cmfs, **dvtest5-sparc | dvtest5-vu** uses a CMFS file system — e.g., a DataVault, CM–HIPPI, CM5–HIPPI, VMEIO host computer, or CM–IOPG. The program consults DVWD and, if necessary, DVHOSTNAME to determine which CMFS device to use. If DVHOSTNAME and DVWD do not define the default hostname, the program uses the default CMFS device for the first IOBA listed with the kernel. If there are no IOBAs listed with the kernel,the program consults the configuration file **/usr/local/etc/dv_hostname**. If that file is missing, the program uses the CMFS file system device on the local host.

- If CMFS_PATHTYPE is set to mixed, **dvtest5-sparc | dvtest5-vu** checks the directory name specified via the **-d** flag: If the directory name is specified by a pathname that does not contain a colon (:), the program uses the SDA. If the pathname does contain a colon, the program checks for a CMFS–hostname component (i.e., the string before the colon) of the pathname. If the pathname does contain a CMFS–hostname component, the program uses that device. If the pathname does not contain a CMFS–hostname component, the program follows the heuristic for CMFS_PATHTYPE = cmfs, as described above.

- If CMFS_PATHTYPE is not set, **dvtest5-sparc | dvtest5-vu** asks the kernel what I/O hardware the system contains. If there is only an SDA, the program uses it. If there is at least one CMFS device but no SDA, the program follows the heuristic for CMFS_PATHTYPE = cmfs. If there is both an SDA and at least one CMFS device, the program follows the heuristic for CMFS_PATHTYPE = mixed. If the kernel sees no I/O hardware, the program follows the heuristic for CMFS_PATHTYPE = cmfs.

Requirements

dvtest5-sparc | dvtest5-vu must be run from a PM that controls a partition in which **ts-daemon** is running. For CM–5 systems that contain CMFS devices, also make sure that the **io.conf** file is correct and that the **fsserver** is running in the background on all CMFS data storage devices with which **dvtest5-sparc | dvtest5-vu** will communicate. For CM–5 systems that contain an SDA, make certain that the SDA's SFS file system is mounted.

```
1.  basics               test file creation/deletion (-s)
2.  data                 test simple file read/write (-s, -h)
3.  write                test writing files (-s, -h)
4.  link                 test link/unlink (-s)
5.  abs_seek             test absolute seek (random) (-s, -h)
6.  rel_seek             test relative seek (deterministic)
                         (-s, -h)
7.  dir_basics           test mkdir/rmdir/chdir (-s)
8.  many_dirs            test creating many directories (-s)
9.  many_files           test creating/deleting many
                         files (-s)
10. serial_io            test serial I/O transfers (-s)
11. mixed_io             test mixed serial and parallel I/O
                         (-s, -h)
12. parallel_partial     test parallel I/O w/ partial blocks
                         (-s, -h)
13. transpose            test transposing serial data (-s)
14. transfer_timing      test transfer speed (-s, -h)
15. raw_transfer_timing test raw transfer speed (-s, -h)
16. overhead_timing      test overhead speed (-s, -h)
17. max_transfer_timing test max transfer speed
18. reliability          test reliability (-h)
```

Figure 17. The **dvtest5** menu.

dvtest5 Defaulting Rules

In systems with multiple IOBAs, **dvtest5** applies the rules outlined below to determine which IOBA and data-storage device to use. **NOTE:** Other programs that use default-device methods to select an I/O device follow these rules as well.

- If the environment variables **DVWD** and/or **DVHOSTNAME** specify a hostname, **dvtest5** uses those values to determine which data-storage device it will use. It then determines the IOBA it will use by examining the **Channel_Board_Configuration** module in **/etc/io.conf** and uses the first channel board listed that is on the same bus as the data storage device. If there is no IOBA on the same bus, **dvtest5** instead uses the rule explained in the next bullet. See Appendix B for a description of **io.conf**.

- If neither `DVWD` nor `DVHOSTNAME` are set indicate a target hostname, `dvtest5` examines the `Channel_Board_Configuration` module in `io.conf` and uses the first channel board listed that has a data-storage device on the same bus. If the the bus has more than one data-storage device, the program uses the device listed first in `io.conf`'s `IO_device_configuration` module.

In a standard I/O configuration, these defaulting rules allow all devices to be tested via `DVWD` and/or `DVHOSTNAME` manipulation.

Appendix F

Man Pages

NAME

/usr/diag/cmdiag – Run CM-5 hardware diagnostics.

SYNOPSIS

cmdiag [**-p** *partition-name*] [[**-g***groupname*] [**-C**] [**-Ebdfilt**][**+Ebdfilt**]

DESCRIPTION

cmdiag is the principle tool for diagnosing hardware problems in the CM-5. **cmdiag** provides four major categories of functional tests:

JTAG scan tests provide scan access to all internal components of Thinking Machine's proprietary chips and boundary scan testing of all chip inputs and outputs.

Connectivity tests support connectivity checks between components in the scan chains, including connectivity across the control and data networks.

Processing node tests evaluate the functionality of the PN circuits, including: the instruction processor (SPARC chip), vector execution unit, memory controller, and network interface.

I/O Processor (IOP) tests exercise the various functions that comprise a CM I/O partition, including: the I/O clock, I/O control, I/O interface to the data network, the I/O buffer, and the I/O channel.

Verifier tests simulate the kind of activity a user application would impose on the CM.

All functions and test routines are accessible via a single user interface. The user invokes the diagnostics from a shell on a CP. Whenever possible, **cmdiag** should be executed from the CP that is the master diagnostic processor (the CP connected by cable to the root node of the diagnostic network). Error messages regarding hardware failures are sent to **diag-error-log.***hostname*. The section RUNNING CMDIAG ON A PARTITION gives a step-by-step explanation about how to run **cmdiag**.

cmdiag takes several optional switches. (See the section CMDIAG COMMAND-LINE SWITCHES, below.) There are no required switches, although we recommend running diagnostics on a specific partition by using the **-p** switch. Run without the **-p** switch, **cmdiag** runs on the entire machine.

When **cmdiag** is executed routinely after bringing up a partition, running the groups **PE**, **global**, **combine**, and **dr** should be sufficient. Once a week or so, we recommend running the complete test suite by creating a partition encompassing the entire machine and running **cmdiag -p -m**. Currently this takes approximately two hours.

Executed without the **-p-**, **-m**, **-f**, or **-g** switches, cmdiag immediately provides a diagnostic environment, which is represented by the prompt <**cmdiag**>. This diagnostic environment supports a set of diagnostic-related utilities and commands as well as the individual tests that comprise the predefined diagnostic test groups. (The utilities, commands, tests, and test groups are listed in the section CMDIAG TESTS AND COMMANDS.) To exit the diagnostic environment, type **exit** at the <**cmdiag**> prompt.

CMDIAG COMMAND-LINE SWITCHES

The switches are as follows:

-p runs diagnostics on the specified partition.
-m executes diagnostic manufacturing tests of CM-5 system components.
-f executes diagnostic field tests of CM-5 system components. Field tests are a subset of the

manufacturing tests.

-g executes tests for *groupname* only. For a list of groupnames, see the section CMDIAG TESTS.

-C enables command completion within diagnostics environment.

+E activates diagnostic environment options:

- **b** = Break on error (default)
- **d#** = Display error count (default = 16)
- **f** = Loop forever
- **i** = Ignore errors
- **l** = Log errors (default)
- **t** = Display trace

-E Deactivate diagnostic environment options (see **+E**).

RUNNING CMDIAG ON A PARTITION

1. Execute **cmpartition stop** to halt the timesharing daemon running on the partition.

2. Reset the partition's registers and switches by executing **/usr/diag/cmreset**.

3. Reset the interface to the partition manager by executing **/usr/diag/cmreset -s**.

4. Check that the pertinent environment variables are set correctly (see the section ENVIRONMENT VARIABLES). In particular, if you must run the diagnostics from a CP that is not the system console/master diagnostic processor, be sure the **JTAG_SERVER** environment variable is set appropriately.

5. (This step is necessary only if the hardware has changed, requiring an edit of **etc/cm/configuration/hardware.install**.) Check the directory defined by the CMDIAG_PATH environment variable to see if there are any files that must be deleted. Delete all files whose names contain the hostname of the CP from which you are executing **cmdiag**.

6. Execute **cmdiag**. Usually, running a few test groups via the following syntax is sufficient:
   syscon% **/usr/diag/cmdiag -p** *partition-name* **-f -gPE -gglobal -gcombine -gdr**

Analyze any failure reports; descriptive error messages are sent to **diag-error-log.***hostname* in your current directory. Rerun any appropriate tests.

7. Delete **diag-error-log.***hostname* when its contents are no longer needed.

CMDIAG TESTS AND COMMANDS

This section lists the names of the tests and commands that **cmdiag** can run. The first subsection categorizes the tests by groupname (see the -g switch, above). The second subsection lists all tests and commands categorized by which part of the machine they serve to diagnose.

Test Groups

Group : SVME (Tests SVME hardware.)

1. test-svme-serial-data-path
2. test-svme-id-prom
3. test-svme-ni-latch-drive
4. test-svme-ni-latch-sample
5. test-svme-ni-chip

Group : SNI
1. test-sni-serial-data-path
2. test-sni-id-prom
3. test-sni-led-reg
4. test-sni-ni-chip

Group : CLKDN
1. test-clkdn-serial-data-path
2. test-clkdn-analog-env-data
3. test-clkdn-analog-env-control
4. test-clkdn-digital-env-data
5. test-clkdn-csr
6. test-clkdn-id-prom
7. test-clkdn-pll-control
8. test-clkdn-pod-status

Group : CLKBUF
1. test-clkbuf-serial-data-path
2. test-clkbuf-analog-env-data
3. test-clkbuf-analog-env-control
4. test-clkbuf-digital-env-data
5. test-clkbuf-csr
6. test-clkbuf-pod-status
7. test-clkbuf-id-prom

Group : IOCLK
1. test-ioclk-serial-data-path
2. test-ioclk-analog-env-data
3. test-ioclk-analog-env-control
4. test-ioclk-digital-env-data
5. test-ioclk-id-prom
6. test-ioclk-cn-switch
7. test-ioclk-cn-chip

Group : SASYS
1. test-sasys-serial-data-path
2. test-sasys-analog-env-data
3. test-sasys-analog-env-control
4. test-sasys-digital-env-data
5. test-sasys-csr
6. test-sasys-pod-status
7. test-sasys-pll-control
8. test-sasys-cn-chip
9. test-sasys-dr-chip
10. test-sasys-drive-sync-control
11. test-sasys-drive-sync-data
12. test-sasys-id-prom

Group : SPI
1. test-spi-serial-data-path
2. test-spi-id-prom
3. test-spi-dr-chip
4. test-spi-cn-chip

Group : DR
1. test-dr-serial-data-path
2. test-dr-id-prom
3. test-dr-dr-chip
4. test-dr5-serial-data-path
5. test-dr5-id-prom
6. test-dr5-dr-chip
7. test-dr-dr-bsr
8. test-afd-bsr

Group : CN
1. test-cn-serial-data-path
2. test-cn-switch
3. test-cn-id-prom
4. test-cn-chip
5. test-cn-cn-bsr

Group : FILLER
1. test-filler-serial-data-path
2. test-filler-dr-chip
3. test-filler-id-prom

Group : CMPE (Tests ability of PNs to interact with NI, DR, and CN hardware.)
1. test-pe-serial-data-path
2. test-pe-ni-chip
3. test-pe-dr-chip
4. test-pe-cn-chip
5. test-pe-id-prom

Group : PEMEM
1. test-pemem-serial-data-path
2. test-pemem-mc-chip
3. test-pemem-id-prom

Group : IOCNTRL
1. test-iocntrl-serial-data-path
2. test-iocntrl-id-prom
3. test-iocntrl-mc-chip 4. test-iocntrl-ni-chip

Group : IODR
1. test-iodr-serial-data-path
2. test-iodr-dr-chip
3. test-iodr-id-prom

Group : IOBUF
1. test-iobuf-id-prom
2. test-iobuf-ni-chip
3. test-iobuf-pbus-buffer
4. test-iobuf-serial-data-path
5. test-iobuf-xbus-buffer
6. test-iobuf-xbus-data-in
7. test-iobuf-xbus-data-out

Group : IOCHNL
1. test-iochnl-cmio-cntrl-out
2. test-iochnl-cmio-cntrl-in
3. test-iochnl-id-prom
4. test-iochnl-pbus-buffer
5. test-iochnl-response-data
6. test-iochnl-serial-data-path
7. test-iochnl-xbus-buffer
8. test-iochnl-xbus-data-in
9. test-iochnl-xbus-data-out

Group : SAC
1. test-sac-serial-data-path
2. test-sac-mc-chip
3. test-sac-ni-chip
4. test-sac-id-prom

Group : SADR
1. test-sadr-serial-data-path
2. test-sadr-dr-chip
3. test-sac-id-prom

Group : IOP-CNTRL
1. reset-system
2. initialize-pe-memory
3. load-secondary-boot
4. clear-pe-memory
5. load-iopcntrl-tests
6. execute-all-iopcntrl-tests

Group : IOP-BUF
1. reset-system
2. initialize-pe-memory
3. load-secondary-boot
4. clear-pe-memory
5. load-iopbuf-tests
6. execute-all-iopbuf-tests

Group : IOP-CHNL
1. reset-system
2. initialize-pe-memory
3. load-secondary-boot
4. clear-pe-memory
5. load-iopchnl-tests
6. execute-all-iopchnl-tests

Group : IOP-SYS
1. reset-system
2. initialize-pe-memory
3. load-secondary-boot
4. clear-pe-memory
5. load-iopsys-tests
6. execute-all-iaopsys-tests

Group : PE

1. reset-system
2. test-jtag-backdoor-connection
3. test-jtag-backdoor-interrupt-clear
4. test-jtag-backdoor-request-clear
5. test-jtag-backdoor-command-channel
6. test-mc-register-read
7. test-cmu-boot-mode
8. test-broadcast-interrupt-receive
9. test-mc-reduce
10. initialize-pe-memory
11. load-secondary-boot
12. test-cmu-run-mode
13. clear-pe-memory
14. test-cmu-run-mode
15. router-init
16. test-pe-memory
17. load-file petests
18. test-cmu-run-mode
19. test-mc-reduce
20. execute-all-pe-tests
21. test-cmu-boot-mode

Group : global (Verifies CM-5's ability to perform global communication operations.)

1. reset-and-load-for-test-group
2. test-cn-async-global-supervisor
3. test-cn-async-global-user
4. test-cn-sync-global
5. test-cn-sync-global-roll-call

Group : broadcast

1. reset-and-load-for-test-group
2. test-broadcast-scalar-send-enable
3. test-broadcast-scalar-supervisor
4. test-broadcast-scalar-user
5. test-broadcast-pn-supervisor
6. test-broadcast-pn-user
7. test-broadcast-interrupt-scalar-send
8. test-broadcast-interrupt-pn-send

Group : combine

1. reset-and-load-for-test-group
2. test-combine-pn-data-is-one
3. test-combine-pn-data-is-zero
4. test-combine-pn-multiword-carry
5. test-combine-pn-multiple-stacked-scan
6. test-combine-pn-overflow-detection
7. test-combine-pn-segmented-scan
8. test-combine-reduce-to-scalar
9. test-combine-int-on-rec-ok
10. test-combine-flush

Group : dr (Verifies CM-5's ability to transfer messages across the data network.)

1. reset-and-load-for-test-group
2. test-dr-scalar-to-pe
3. test-dr-tag-scalar-send
4. test-dr-pn-to-scalar
5. test-dr-tag-pn-send
6. test-dr-length-scalar-send
7. test-dr-length-pn-send
8. test-dr-pn-static-send
9. test-dr-pn-dynamic-send
10. test-dr-flow-control-pn-to-pn
11. test-dr-rec-stop
12. test-dr-afd-router-empty-pn
13. test-dr-afd-router-empty-scalar
14. test-dr-afd-router-full
15. test-dr-int-rec-ok-scalar
16. test-dr-int-rec-ok-pn

Group : partition (Verifies CM-5's ability to perform global, broadcast, combine, and DN operations within a partition.)

1. reset-and-load-for-test-group
2. test-partition-global-scalar-static
3. test-partition-global-pe-static
4. test-partition-global-dynamic
5. test-partition-combine-scalar-static
6. test-partition-combine-pe-static
7. test-partition-combine-dynamic
8. test-partition-broadcast-scalar-static
9. test-partition-broadcast-pe-static
10. test-partition-broadcast-dynamic
11. test-partition-dr-scalar-afd
12. test-partition-dr-pe-afd

Comprehensive List of Commands and Tests

--General Utilities--

alias	continue-from-abort	getenv
help	list-commands	list-groups
run-groups	script	silent-script
setenv	set-diag-environment	show-all-errors
shell	show-diag-environment	whatis

--JTAG Utilities--

find-cm-error	find-cm-pn-error	find-cm-dr-error
find-cm-cn-error		

--JTAG Status Commands--

show-all-pe-status	show-pe-status	show-spi-status

show-cn-status	show-dr-status	show-svme-status

--JTAG Config Commands--

configure-all-pes	configure-pe	configure-spi
configure-all-cns	configure-cn	configure-all-drs
configure-dr	configure-filler	configure-svme
enable-auto-reset	reset-quad	

--CMIO HIPPI Commands--

close-cmio-diag-connection	cmiohippi-cmio-data-all-path-xfer	cmiohippi-cmio-data-xfer
cmiohippi-data-pattern-xfer	cmio-vmeio-memory-iope-all-mode-xfer	cmiohippi-coldboot
cm5-write-datavault	cmio-dv-iope-all-pattern-xfer	cmio-dv-iope-xfer
cm5-read-vmeio	cmiohippi-sm-iop-src-dst-diffiop-sm-xfer	cm5-write-vmeio
cmio-vmeio-iope-all-pattern-xfer	cmio-vmeio-iope-xfer	cm5-read-datavault
cmio-vmeio-memory-iope-all-pattern-xfer	cmio-vmeio-memory-iope-xfer	cm5-read-hippi
cmio-vmeio-iope-all-mode-xfer	cmiohippi-select-ports	cmiohippi-sm-dst-iop-sm-xf
cmiohippi-sm-src-dst-sm-ifield-xfer	cmiohippi-sm-src-dst-sm-xfer	cm5-write-hippi
cmiohippi-sm-iop-src-dst-sameiop-sm-xfer	cmiohippi-sm-iop-src-sm-xfer	cmiohippi-standlone-tests
creat-cmiohippi-comparsion-data	data-xfer-on-all-iobuf-chnl	establish-cmio-diag-connecti
get-cmioc-bus-id	get-iope-config	get-station-ids-on-bus
select-next-cmiohippi-iop-on-cmio-bus	reset-cmiohippi	init-cmio-diag-environment
select-cmio-server	set-cmiohippi-arbiter	set-cmiohippi-check-data
set-cmiohippi-check-parity	set-iop-buffer-and-chnl	show-cmiohippi-diag-enviro
show-corrupted-data-on-hippi	test-cmio-device-data-xfer	

--JTAG Scan Commands--

add-multi-chip-sample	add-multi-chip-scan	instantiate-multi-chip-sampl
instantiate-multi-chip-scan	multi-chip-sample	multi-chip-scan
nb-scan-in-pod-udr	read-id-prom-by-pod-name	sample-pod-register
sample-pod-udr	scan-in-pod-register	scan-in-pod-udr scan-dr

--JTAG DN Commands--

autosize	dn-reset	dn-reset-to-clear
jtag-reset	jtag-run-test-idle	phoenix-ir-test
server-connect	server-disconnect	test-dn-channel-reg

--JTAG Equip. Set Commands--

build-autosizing-file	build-cbs-diag-partition	build-equipment-set
check-count-and-first	check-equipment-set	clear-diag-partition
compare-autosizing-file	generat-na-partition	load-autosizing-file
load-diag-partition	print-all-pod-id-prom	read-all-pod-id-prom
read-all-pod-id-prom-in-partition	show-all-pod-na-list	show-autosizing
show-diag-partition	show-hlr-partition	walk-sdn-tree

--JTAG Filler Commands--

select-filler
test-filler-dr-chip
test-filler-id-prom
test-filler-serial-data-path
select-all-fillers

--JTAG SASYS Commands--

reset-sasys
sample-sasys-csr
sample-sasys-pod-status
select-sasys
test-sasys-analog-env-control
test-sasys-analog-env-data
test-sasys-cn-chip
test-sasys-csr
test-sasys-digital-env-data
test-sasys-dr-chip
test-sasys-drive-sync-control
test-sasys-drive-sync-data
test-sasys-id-prom
test-sasys-pll-control
test-sasys-pod-status
test-sasys-serial-data-path
write-sasys-pod-status

--JTAG SADR Commands--

select-sadr
test-sadr-id-prom
test-sadr-dr-chip
test-sadr-serial-data-path

--JTAG SAC Commands--

select-sac
test-sac-id-prom
test-sac-mc-chip
test-sac-ni-chip
test-sac-serial-data-path

--JTAG IOCLK Commands--

reset-ioclk
sample-ioclk-csr
sample-ioclk-pod-status
select-ioclk
test-ioclk-analog-env-control
test-ioclk-analog-env-data
test-ioclk-cn-chip
test-ioclk-cn-switch
test-ioclk-csr
test-ioclk-digital-env-data
test-ioclk-id-prom
test-ioclk-pod-status
test-ioclk-serial-data-path
write-ioclk-pod-status

--JTAG IODR Commands--

select-iodr
test-iodr-id-prom
test-iodr-dr-chip
test-iodr-serial-data-path

--JTAG IOCHNL Commands--

select-iochnl
test-iochnl-cmio-cntrl-out
test-iochnl-cmio-cntrl-in
test-iochnl-id-prom
test-iochnl-pbus-buffer
test-iochnl-response-data
test-iochnl-serial-data-path
test-iochnl-xbus-buffer
test-iochnl-xbus-data-in
test-iochnl-xbus-data-out

--JTAG IOBUF Commands--

select-iobuf	test-iobuf-id-prom	test-iobuf-ni-chip
test-iobuf-pbus-buffer	test-iobuf-serial-data-path	test-iobuf-xbus-buffer
test-iobuf-xbus-data-in	test-iobuf-xbus-data-out	

--JTAG IOCNTRL Commands--

select-iocntrl	test-iocntrl-id-prom	test-iocntrl-mc-chip
test-iocntrl-ni-chip	test-iocntrl-serial-data-path	

--JTAG CLKBUF Commands--

reset-clkbuf	sample-clkbuf-csr	sample-clkbuf-pod-status
select-clkbuf	test-clkbuf-analog-env-control	test-clkbuf-analog-env-data
test-clkbuf-csr	test-clkbuf-digital-env-data	test-clkbuf-id-prom
test-clkbuf-pod-status	test-clkbuf-serial-data-path	write-clkbuf-pod-status

--JTAG DR5 Commands--

select-dr5	test-dr5-dr-chip	test-dr5-id-prom
test-dr5-serial-data-path		

--JTAG DR Commands--

select-dr	select-all-drs	test-dr-dr-chip
test-dr-id-prom	test-dr-serial-data-path	test-dr-dr-bsr
test-afd-bsr		

--JTAG CN Commands--

select-cn	select-all-cns	test-cn-cn-bsr
test-cn-chip	test-cn-id-prom	test-cn-serial-data-path
test-cn-switch		

--JTAG PEMEM Commands--

select-all-pemems	select-pemem	test-pe-pemem-bsr
test-pemem-id-prom	test-pemem-mc-chip	test-pemem-serial-data-path

--JTAG PE Commands--

select-all-pes	select-pe	test-pe-cn-chip
test-pe-dr-chip	test-pe-id-prom	test-pe-ni-chip
test-pe-serial-data-path		

--JTAG CLKDN Commands--

reset-net-clock-switch reset-clkdn sample-clkdn-csr
sample-clkdn-pll-control sample-clkdn-pod-status select-clkdn
reset-all-clkdns set-clkdn-pll set-net-clock-switch
test-clkdn-analog-env-control test-clkdn-analog-env-data test-clkdn-csr
test-clkdn-digital-env-data test-clkdn-id-prom test-clkdn-pll-control
test-clkdn-pod-status test-clkdn-serial-data-path write-clkdn-pod-status

--JTAG SPI Commands--

select-spi test-spi-cn-chip test-spi-dr-chip
test-spi-id-prom test-spi-serial-data-path

--JTAG SNI Commands--

select-sni test-sni-id-prom test-sni-led-reg
test-sni-ni-chip test-sni-serial-data-path

--JTAG SVME Commands--

connect-svme read-byte read-long
read-word select-svme test-svme-id-prom
test-svme-ni-chip test-svme-ni-latch-sample test-svme-ni-latch-drive
test-svme-serial-data-path write-long write-word
wr-rd-ver-dn-channel-reg

--PM Diag Utilities--

verbose safety enable-pm-board
disable-pm-board enable-pe disable-pe
map-in-pe map-out-pe map-in-pe-board
map-out-pe-board map-in-pe-backplane map-out-pe-backplane
read-ni-register write-ni-register request-backdoor-dump
request-combine-dump request-left-router-dump request-right-router-dump
reset-system reset-svme quit

--PM Diag Router Utilities--

router-init set-self-address check-self-address
dump-chunk-table read-memory-using-ldr read-memory-using-rdr
send-left-router-message send-right-router-message

--PM Diag Tests--

initialize-pe-memory clear-pe-memory load-secondary-boot

test-pe-memory
load-pe-tests
load-iop-tests
load-iopbuf-tests
load-iopchnl-tests
load-iopcntrl-tests
load-iopsys-tests
load-iopdv-tests
load-ioppe-tests
execute-single-pe-tests
execute-pe-tests
execute-all-iopcntrl-tests
execute-all-iopbuf-tests
execute-all-iopchnl-tests
execute-all-iopsys-tests
execute-all-iopdv-tests
execute-all-ioppe-tests
execute-all-pe-tests
load-file
load-iope-file

--PM Diag Config Tests--

test-jtag-backdoor-connection
test-jtag-backdoor-interrupt-clear
test-jtag-backdoor-request-cl
test-jtag-backdoor-command-channel
test-mc-register-read
test-broadcast-interrupt-recei
test-mc-reduce
test-cmu-boot-mode
test-cmu-run-mode

--PM Diag Debug Commands--

reduce-memory
set-memory
read-memory
write-memory
dump-memory
load-memory
reduce-mc-register
set-mc-register
read-mc-register
write-mc-register
dump-mc-register
reduce-mc-prom
read-mc-prom
dump-mc-prom
diff-pe-prom
dump-pe-prom
reduce-cmu-register
set-cmu-register
read-cmu-register
write-cmu-register
dump-cmu-register
reduce-io
set-io
read-io
write-io
dump-io
reduce
set
write
read
write-double
read-double
dump
select-buffer-board
select-channel-board
set-start-block
set-station-id
select-iop-ver-data
loop-write
loop-read
loop-write-double
loop-read-double
loop-write-read
loop-write-read-double
write-pes-backdoor
read-pes-backdoor
extract-message-buffer
extract-message-buffer-ldr
extract-message-buffer-rdr
dump-cmu-reset-state
cmp-use-control-net
call-single-pe-function
lookup-pe-symbol
load-pe-cmp-map
call-diag-function
lookup-diag-symbol
load-symbol-table

--Verifier Support Functions--

broadcast-user-data
broadcast-supervisor-data
monitor-bc-receive
monitor-sbc-receive
monitor-dr-receive
pe-extract-message
pe-read-memory
pe-read-ni-register
pe-write-memory
pe-write-ni-register
pe-write-ni-register-fast
query-all-pe-error-status
send-data-to-other-node
set-user-symbols
setup-pe-address
show-all-accessible-reg-names
show-all-user-symbols
show-pe-memory
show-chunk-table-data
show-scalar-ni-register
tell-pe-to-broadcast
tell-pe-to-combine
tell-pe-to-dr-loop-drain
tell-pe-to-dr-loop-send
tell-pe-to-drain-dr
tell-pe-to-fill-dr
tell-pe-to-check-flow
tell-pe-to-read-broadcast
tell-pe-to-read-combine
tell-pe-to-read-dr
tell-pe-to-send-dr
vfr-diagnose-dr-pe-to-pe
vfr-make-pe-send-dr
vfr-setup-address-tables
vme-hardware-debugger
write-scalar-ni-register

--Verifier Init Function--

just-load-no-reset	load-chunk-table-data	load-ndiag-pemon
reload-ndiag-pemon	reset-and-load-for-test-group	restart-ndiag-pemon
run-all-vfr-tests		

--Verifier Broadcast Tests--

run-all-broadcast-tests	test-broadcast-interrupt-pn-send	test-broadcast-interrupt-scal
test-broadcast-scalar-send-enable	test-broadcast-scalar-supervisor	test-broadcast-scalar-user
test-broadcast-pn-supervisor	test-broadcast-pn-use	

--Verifier Global Tests--

run-all-global-tests	vfr-diagnose-async-global	vfr-diagnose-sync-global
test-cn-async-global-user	test-cn-sync-global	test-cn-sync-global-roll-call

--Verifier Combiner Tests--

	vfr-diagnose-combine-reduce-to-scalar	vfr-diagnose-combine
test-combine-int-on-rec-ok	test-combine-pn-data-is-one	test-combine-pn-data-is-zero
test-combine-pn-multiple-stacked-scan	test-combine-flush	test-combine-pn-overflow-de
test-combine-pn-segmented-scan	test-combine-reduce-to-scalar	test-combine-pn-multiword-

--Verifier Data Router Tests--

run-all-dr-tests	test-dr-afd-router-full	test-dr-afd-router-empty-pn
test-dr-afd-router-empty-scalar	test-dr-flow-control-pn-to-pn	
test-dr-int-rec-ok-scalar	test-dr-int-rec-ok-pn	test-dr-length-scalar-send
test-dr-length-pn-send	test-dr-pn-dynamic-send	test-dr-pn-static-send
test-dr-pn-to-scalar	test-dr-rec-stop	test-dr-scalar-to-pe
test-dr-tag-scalar-send		

--Verifier Partitioning Tests--

test-partition-broadcast-dynamic	test-partition-broadcast-pe-static	test-partition-broadcast-scala
test-partition-combine-scalar-static	test-partition-combine-pe-static	test-partition-dr-scalar-afd
test-partition-dr-pe-afd	test-partition-global-dynamic	test-partition-global-pe-stati
test-partition-global-scalar-static		

--Verifier SVME Board Tests--

ni-access-test-interrupt-reg	ni-access-test-readable-writable-reg	ni-access-test-rec-fifo
ni-access-test-reg-after-reset	ni-access-test-all	ni-access-test-send-fifo
ni-access-test-writable-fields	ni-broadcast-full-test	ni-broadcast-test-rec-abstain
ni-broadcast-test-single-word	ni-broadcast-test-write-rfifo	ni-combine-test-legal-patter

ni-combine-test-illegal-patterns	reset-vme-int-enable_bit	set-test-attribute
set-vme-int-enable_bit	show-reg-test-result	test-all-registers
test-all-ni-registers	test-all-vme-interface-registers	test-command-reg-access
test-data-reg-access	test-dn-parent-reg-access	test-dn-child-reg-access
test-int-force-on-off	test-ni-presence	test-ni-reset-condition
test-ni-word0-latch-access	test-ni-word1-latch-access	test-reset-reg
test-register		

--SA Library Interface Tests--

sa-auto-reset-partition	sa-disable-control-net	sa-disable-control-net
sa-force-sync-global-complete	sa-set-all-com-flush-send	sa-set-all-com-control
sa-test-open	sa-test-close	sa-test-reset-partition
sa-test-isolate-dr	sa-test-connect-dr	sa-test-connect-cn
sa-test-disconnect-cn	sa-test-get-components	

ENVIRONMENT VARIABLES

Set the environment variables below for the current system configuration. In most cases the default values will be correct.

SVMEDEV *number*
: This variable tells the device driver which SVME to talk to. A DIP switch setting on the SVME board determines the value of *number*, which can be 0, 1, 2, or 3. For example, if the DIP is set to 00001000 (where 1 is up) then the board is SVME0. The default is 0.

CMDIAG_PATH *pathname*
: This variable tells **cmdiag** where to find various descriptors and the files it uses. The default is ./.

PM_OBJECT_PATH *pathname*
: This variable tells **cmdiag** where to look for the files that will be downloaded into each processing node. The default is **./object/**.

SVME_RESET
: This variable defines whether the SVME board is reset upon execution of **cmdiag**. The default is no reset.

JTAG_SERVER *hostname*
: This variable tells **cmdiag** where the **jtagserver** is running--usually on the system console/master diagnostic processor. Setting this variable is not required if you are running **cmdaig** from the master diagnostic processor.

JTAG_RESET_FILE *filename*
: *filename* specifies the reset script used to do a reset. The *filename* that is in effect at system installation should not be changed.

RESTRICTIONS

cmdiag and the timesharing daemon cannot run on the same partition.

NAME

/usr/etc/cmpartition -- partition Connection Machine (CM-5, CM-5-LD) hardware resources

SYNOPSIS

cmpartition list	[-l]
cmpartition create	[-pm *hostname*] [-name *partition_name*] {-size *n* | -pn_range *range* [-pn_range *range*]} [-description *partition_description*] [-iop *integer_address*]
cmpartition start	[-pm *hostname*] [-name *partition_name*] [-n *integer*] [-reva] -cmd *command_name command_arg1..command_argn*
cmpartition stop	[-pm *hostname*] [-name *partition_name*]
cmpartition delete	[-pm *hostname*] [-name *partition_name*]

DESCRIPTION

cmpartition is the principal system administration interface for configuring the CM-5 and CM-5-LD processor and network hardware into usable resources known as partitions.

Partitions are mutually disjoint subsets of the Connection Machine hardware that execute independent copies of CMOST, the Connection Machine operating system. CMOST in turn schedules and manages all user processes within the partition.

A partition is minimally defined by a single control processor designated as the Partition Manager (PM) and a set of parallel processing nodes (PN's), plus the nodes of the control network (CN) that link all of the processors into a common communications domain. Once this set of connections is created among the specified processors, it persists until it is explicitly deleted or the hardware is reset. Typically, partitions are torn down and recreated on the order of a few times a day.

In order for the PM to make use of the processor nodes assigned to its partition, it must notify its copy of the CMOST kernel of their number and locations. It must then download the kernel image to be run on the parallel processors in the partition. Lastly, it must start up the timesharing access mode for user programs on the PM. [Note: Currently the starting and stopping of a particular partition requires direct access to the CMOST kernel on the partition manager; thus these operations must be performed on the partition manager itself using the **rsh** command from the system console.]

cmpartition is comprised of a set of five commands that control the various aspects of partition management. Only one of the commands -- **cmpartition list** -- can be executed without root priveleges. The **cmpartition** commands are:

cmpartition list

This command prints out on the standard output a short description of the Connection Machine hardware, followed by a short list of all currently configured partitions and their attributes. This is the default subcommand; that is, **cmpartition** is equivalent to **cmpartition list**.

-l

Prints an expanded list of partition attributes.

cmpartition create

This command allocates and reserves Connection Machine resources for the new partition by editing the file **/etc/cm/configuration/partitions.current**. To bring up a partition, it must be both created and started. The **cmpartition create** command must be called from the system console.

-pm *hostname*

The hostname of the unique control processor associated with the partition. This control processor will be the partition manager for this partition. If this switch is not included on the command line, by default the partition's PM is the control processor on which **cmpartition create** is executed.

-name *partition_name*
A unique name for the partition. There is no default value.

-description *partition_description*
A string that tells users about the partition. The description is included in the output of **cmpartition list -l**.

-size *n*
The number of PN's to be configured in the new partition. The value *n* must be less than or equal to the total number of available PNs. Currently this switch assigns a range that begins at PN address 0. This switch cannot be specified on the same command line as the **-pn_range** switch.

-pn_range *range*
A range of PN addresses of the form *x-y*, specifying the first and last address in this range. The **pn_rangeR** *range* switch can be specified more than once, although ranges cannot overlap. Use **pn_range** to configure partitions more precisely than the **-size** switch allows. This switch cannot be specified on the same command line as the **-size** switch.

-iop *integer_address*
Supported for the CM-5 only, this switch specifies an I/O processor to be associated with the partition being created. This argument is needed to support **cmdiag** tests that involve I/O. *integer_address* is the I/O processor's network address.

cmpartition start
This command initializes the partition configured for the specified partition manager and starts up the timesharing access mode on that partition manager. After the **cmpartition start** command is executed, users can run programs on that partition. **cmpartition start** must be called from the PM of the partition you wish to start (usually via **rsh** from the system console).

-pm *hostname*
The hostname of the unique partition manager of the partition to be started. If neither this switch nor **-name** is included on the command line, the partition started is the one managed by the PM on which **cmpartition start** is executed.

-name *partition_name*
The partition's unique name, given by **cmpartition create**. There is no default value.

-n *int*
The number of times the timesharing daemon (**ts-daemon**) should automatically be restarted upon failure. Default is 10.

-reva
Notifies the operating system that some or all of the processing nodes' NI chips are revision A chips, which require special handling. The **-reva** switch is required if there are any revision A chips in the system; if there are no revision A chips (the usual case), do not specify **-reva**.

To determine the revision status of your CM-5 system's NI chips, examine the output of the command **dcmni** (executed on a CP): chips marked **Phoenix** are revision A, while chips marked **Phoenix B** are revision B.

-cmd *command_name command_arg1...command_argn*
A command followed by its arguments. No other switches can follow the **-cmd** switch since they would be interpreted as one of the command's arguments. Currently there is no need to specify any command but **ts-daemon**, which starts the

timesharing daemon running on the partition.

cmpartition stop

This command terminates the timesharing access mode on the partition manager from which this command is executed. **cmpartition stop** must be called from the PM of the partition you wish to stop (usually via **rsh** from the system console). After the **cmpartition stop** command is executed, users can no longer run programs on that partition. Unless the partition has also been deleted from the **partitions.current** file, it can be restarted simply by executing the **cmpartition start** command (that is, the **cmpartition create** command is not necessary).

-pm *hostname*
The hostname of the unique partition manager of the partition to be stopped. If neither this switch nor **-name** is included on the command line, the partition stopped is the one managed by the PM on which **cmpartition stop** is executed.

-name *partition_name*
The partition's unique name, as given by **cmpartition create**. There is no default value.

cmpartition delete

This command deallocates the resources of the partition and removes its definition from the file **/etc/cm/configuration/partitions.current**. The **cmpartition delete** command must be called from the system console.

-pm *hostname*
The hostname of the unique partition manager associated with the partition that you wish to delete.If neither this switch nor **-name** is included on the command line, the partition deleted is the one managed by the PM on which **cmpartition delete** is executed.

-name *partition_name*
The partition's unique name, as given by **cmpartition create**. There is no default value.

CONFIGURATION GUIDELINES

Partitions must be configured carefully so as not to strand PNs or cause unnecessary competition on shared resources such as the data network. This section contains a brief discussion of the rules governing the size and distribution of partitions under Version 7.1 of CMOST. As the operating system matures, these rules are expected to become considerably more liberal. The purpose of the current restrictions is to ensure maximum protection for user applications, as they run in one partition, from being corrupted by processes running in other partitions.

Following the rules listed below will ensure reliable partition isolation. It is sometimes possible to create viable partitions that deviate from these rules, but we do not recommend doing so. (Note that **cmpartition create** will try to accomodate any creation request; it is up to the user to be knowledgeable of the configuration if the rules are not followed.)

1. The number of PNs in a partition must be a power of 2. This rule is further defined according to Connection Machine model:

   CM-5: A partition must contain at least 32 PNs. (The only exception to this rule is a CM-5 that has a total of 16 PNs; all 16 PNs must be configured as one partition.)

   CM-5-LD: A partition must contain either 16 or 32 PNs; that is, the CM-5-LD can support one partition of 32 PNs or two partitions of 16 PNs each.

2. The PNs within a partition must be contiguous in the address space, except when there is only

one partition.

3. The first PN of a contiguous set must start on an address where:
   address MOD *partition_size* = 0

4. There must be a partition manager for each partition. Each PM can manage only one partition.

5. (CM-5 only) The maximum number of partitions that a CM-5 can accommodate is a function of the total number of PNs in the Connection Machine:

CM-5 SIZE (IN # PNs)	MAXIMUM NUMBER OF PARTITIONS
16	1
32, 64	2
128, 256	4
512, 1K	8
2K, 4K	16
8K	32

For example, these are some possible partitionings of a CM-5 with 256 PNs and 4 PMs:
1 partition of 256 PNs.
2 partitions, each of 128 PNs.
3 partitions, two of 64 PNs and one of 128 PNs.
4 partitions, two of 32 PNs, one of 64 PNs, and one of 128 PNs.
4 partitions, each of 64 PNs.

The above partitionings use all the system's PNs; of course, you can set up a partition configuration that does not include all available PNs.

EXAMPLES

```
% cmpartition list -l
CM System "Sand"
   256 Processors [ 8 Mbytes memory, SPARC IU, SPARC FPU ]
   2 Partition_Managers
      beethoven.think.com
      haydn.think.com
Available PN Ranges:
 All PNs in use

Name  Partition_Manager    Size  State      Nodes    Description
      beethoven.think.com  128   ALLOCATED  0-127    beethoven.think.com
      haydn.think.com      128   ALLOCATED  128-255  haydn.think.com

% cmpartition delete -pm beethoven.think.com
% cmpartition delete -pm haydn.think.com

% cmpartition create -pm beethoven.think.com -pn_range 0-63
% cmpartition start -pm beethoven.think.com -cmd ts-daemon
```

% **cmpartition stop -pm beethoven.think.com**

FILES

/etc/cm/hardware.install	A description of the Connection Machine hardware as installed.
/etc/cm/partitions.current	A description of all currently configured partitions.

SEE ALSO

ts-daemon(8), **hardware.install**(8), **cmbes**(8)

BUGS

It is recommended that all **cmpartition** commands be initiated from the system console. Use the remote shell (**rsh**) command to run **cmpartition start** and **cmpartition stop** on the PM that manage the pertinent partition. This is presently necessary to preserve resource allocation consistency.

NAME

dvcoldboot - Powers up, spares, and heals the DataVault; initializes DataVault configuration variables.

SYNTAX

dvcoldboot [+sD,S | -sD,S | -pP,I | +aP
| -aP | -bP,N | -n | -i | +C | -C | -help]

ARGUMENTS

+sD,S	Replace (faulty) drive D with spare S. D must be less than 39, and S must be 0, 1, or 2.
-sD,S	Replace spare S with (repaired) drive D. D must be less than 39, and S must be 0, 1, or 2.
-pP,I	Set station ID on port P of the DataVault to the value of I. P must be 0 or 1, and I must be less than 16.
+aP	Set port P to be the arbiter on the bus. P must be 0 or 1.
-aP	Set port P to not be the arbiter on the bus. P must be 0 or 1.
-bP,N	Set bus ID for port P to the value of N. P must be 0 or 1, and N must be less than 256.
-n	Use no spares.
-i	Initialize the configuration file and spare settings.
+cC	After power-up only, turns on command-channel mode and selects port *C* to be a command channel. *C* must be 0 or 1. This flag is valid on CM-5 systems only.
-cC	Turns off command-channel mode on port *C*. This flag is valid for CM-5 systems only.
-help	Print on the screen information about dvcoldboot.

WHERE EXECUTED

DataVault file server computer.

DESCRIPTION

The command dvcoldboot is used when powering up the DataVault, when sparing and healing the DataVault, or when setting a bus ID, station ID, or bus arbiter. If no argument is specified, dvcoldboot initializes the configuration variables, using the values stored in the DataVault's configuration file, /usr/local/etc/diag/dv_coldboot.config. Whenever dvcoldboot executes, it automatically stores any new configuration settings in this file.

Powering Up the DataVault

dvcoldboot must be executed when the DataVault is initially powered up or restarted and after DataVault diagnostics are executed. If the DataVault computer crashes, dvcoldboot automatically executes when the file server is rebooted.

dvcoldboot downloads the DataVault's microcode and allocates the disk drives according to the configuration file; it also sets the bus ID, station ID, and arbitration status for both DataVault ports according to the configuration file.

Configuring the DataVault

When dvcoldboot is executed with configuration arguments (-p, +a, -a, -b), the utility updates the DataVault's configuration file, /usr/local/etc/diag/dv_coldboot.config, which resides on the MicroVax.

If the configuration file is missing (for example, because it has been accidentally deleted), dvcoldboot issues a warning. Execute dvcoldboot with the -i option to recreate the fields in the file; then execute dvcoldboot with configuration arguments to update the configuration settings.

Ignore-errors (i)

prevents any errors from being reported.

Loop-forever (f)

causes a test to loop forever through all subtests (tests that it calls) when it encounters an error. This environment variable is often useful in the field and is ordinarily enabled during troubleshooting diagnostics. Ctrl-C aborts this option.

Display-error-count (d#)

allows you to control how many errors the diagnostic program will display or log for each test. The default error count to be idsplayed is 16. You can change this variable by entering a decimal value as an integer argument.

Log-errors (l) (default option)

causes the error handler to write all error messages to a log file rather tan display them. In the current implementation, this file is named diag-error-log and is located in /usr/local/etc/diag.

Display-trace (t)

allows you to display or inhibit messages that are built into tests with the TRACE ((msg")) macro. It is intended for use in a manufacturing environment and is ordinarily disabled in the field.

Executed with the -m or -f arguments, hippidiag runs the requested predefined diagnostic test suite. To run a subset of the tests, specify the -g option with one or more groupnames. The groupnames are listed below.

Executed without the -m, -f, or -g options, hippidiag immediately provides a command-line interpreter, represented by the prompt <hippi-DIAG>, which supports the four sub-diagnostic packages as well as the individual tests that comprise the predefined diagnostic programs. To run a sub-diagnostic from the <hippi-DIAG> prompt, simply type the name of the sub-diagnostic and press the Return key. The sub-diagnostic prompt will then appear. For example,

<hippi-DIAG> srcdiag
<SRC-DIAG>

If you append -C to the command line above, the sub-diagnostic prompt will appear followed by a list of the tests that you can run at the sub-diagnosticprompt.

Generally, it is best to run all test groups within a sub-diagnostic, check the results, and then rerun any failed tests individually. Before rerunning the failed tests, either exit and re-enter the sub-diagnostic, or reset the 29K board by typing

<SRC-DIAG> reset29k

Any error messages generated by the tests are sent to standard error and standard output. The error messages are also logged in /usr/local/etc/diag/diag-error-log on the CM-HIPPI.

The srcdiag Sub-Diagnostic

srcdiag is a standalone diagnostic package for the CM-HIPPI's source board. It consists of three groups of tests:

- The all-ktest group, listed below, contains 29 tests (ktest0ktest28). These tests diagnose and verify the functionality of the source board's 29Kside registers. See the Restrictions section of this man page.

  ktest0-vme-command-reg-read　ktest1-vme-command-reg-write
  ktest2-check-reset-reg　ktest3-access-err-force-parity
  ktest4-check-IFIFO-status　ktest5-EPROM-checksum
  ktest6-IRAM-address-lines　ktest7-IRAM-memory-check
  ktest8-DRAM-address-lines　ktest9-DRAM-memory-check
  ktest10-DRAM-byte-access　ktest11-SM-FIFO-echo
  ktest12-VME-side-IFIFO-status　ktest13-HPPI-side-IFIFO-status
  ktest14-ofifo-status　ktest15-LED-marching-pattern
  ktest16-RS232-config-DIP-switch　ktest17-VME-INT-parity-error
  ktest18-VME-INT-bus-error　ktest19-VME-INT-SMI-FIFO-empty
  ktest20-VME-INT-SMO-FIFO-ready　ktest21-VME-INT-HPPI-request
  ktest22-software-trap-register　ktest23-HPPI-INT-SMDIF-ready
  ktest24-HPPI-INT-SM-IACK　ktest25-HPPI-INT-SMDOF-empty
  ktest26-HPPI-INT-SMDIF-parity　ktest27-HPPI-fifo-reset-bits
  ktest28-read-dip-switch

- The all-stest group contains 17 tests (stest0stest16). These tests verify and diagnose the functionality of the source board's VMEside (Sun-side) registers.

  stest0-hppi-reset　stest1-hppi-data-fifo-in-status
  stest2-hppi-data-fifo-out-status stest3-hppi-data-fifo-read-write
  stest4-event-fifo-write-status　stest5-event-fifo-read-status
  stest6-event-fifo-read-write　stest7-send-packet-in-standalone
  stest8-force-parity-error-send-burst　stest9-read-write-iop-target-ram
  stest10-total-counter-read-write stest11-iop-counter-read-write
  stest12-force-SMDIF-parity-error stest13-force-DRAM-ODD-parity-error
  stest14-force-DRAM-EVEN-parity-error stest15-force-IRAM-ODD-parity-error
  stest16-force-IRAM-EVEN-parity-error stest17-reset-hppis-from-vme-side

- The src-board-test consists of all the tests in the all-ktest group and all the tests in the all-stest group.

You can run these test groups either via hippidiag -g, or by typing run-groups test-group-name at the <SRC-DIAG> prompt. For example,

```
<SRC-DIAG> run-groups all-ktest
```

The tests that make up the test groups can also be run individually at the <SRC-DIAG> prompt. Completion" mode is available: type the first letter of the test and press the Esc key to step through all tests beginning with that letter. If you want to run the test, press the Return key; otherwise, press the Esc key to continue stepping through tests.

Check the test results of tests run individually by typing score, a space, and the name of the test (completion" mode is available). For example,

```
<SRC-DIAG> score stest3-hippi-data-fifo-read-write
```

NAME

hippi_loop5 Exercises the CM-HIPPI system.

Syntax hippi-loop [-rwDUvghR | -iifield | -ppattern | -ssize | -nNreps]

-r Only test reading from the HIPPI channel to the CM.

-w Only test writing from the CM to the HIPPI channel.

-D Drop the connection between tests.

-U Use the existing connection if possible.

-v Be verbose when displaying information about connection setup and termination.

-g Display the status of the CM-HIPPI source and/or destination board.

-h Use the HIPPI ports rather than the loopback ports.

-R Test a connection to a remote system; a matching process is running at the remote end of the HIPPI channel.

-iifield Use ifield as the I-field when establishing the loopback connection.

-ppattern Use pattern to create the data being transferred. pattern may be data-equal-address , random, or a hexadecimal number (for example, 0x0) for a constant pattern.

-ssize Transfer size bytes of data. For size , you may specify 16k, 32k, 64k, 256k , 1m, 2m, or 4m. (You may use uppercase or lowercase letters for k and m.)

-nNreps Repeat each pattern Nreps times.

Description

The hippi_loop system exerciser runs code with CMFS library calls to verify that data transfers between the CM and the CM-HIPPI occur successfully. Test data completes a circuit by looping from the destination board to the source board on the CM-HIPPI.

If you do not supply the -h option, hippi-loop attempts to use the loopback ports on the source and destination boards; be sure these ports are connected with the loopback cable supplied with the system. If you want hippi-loop to use the HIPPI ports rather than the loopback ports, attach a cable (at most 25 meters long) to the IN and OUT ports on the CM-HIPPI bulkhead and issue hipp i-loop with the -h option.

If you do not supply a pattern, hippi-loop uses a default set of patterns and tests each pattern once.

If you do not supply the -r or -w option, hippi-l oop tests both reading from the CM-HIPPI and writing to the CM-HIPPI.

The hippi-loop command does not provide diagnostic information; it simply reports whether or not the data transfer tests were successful. True diagnostic tests must be run to isolate failed components.

SEE ALSO

dvtest5
hippidiag

NAME

/usr/etc/io_cold_boot – Boot the IOBAs in a system

SYNOPSIS

io_cold_boot [**-B** *Download executable file*] [**-D** *logical channel index*] [**-i** *IOBA NI address*] [**-I** *IO configuration file*]
[**-K** *Kernel executable file*] [**-L** *Log file*] [**-m** *memory size*] [**-T** *timeout in seconds*]
[**-x** *MMU control register value*]

DESCRIPTION

io_cold_boot downloads and boots the IOPN kernel in the IOBA subsystem. The IOPN is the PN that resides on the IOBA's controller board. **io_cold_boot** must be run from the PM of an active partition; before running **io_cold_boot**, you must:

1. Set the environment variables **CMDIAG_PATH** and **JTAG_SERVER**. (If these are not set, **io_cold_boot** prints an error message and exits.)
2. Execute **cmreset** on the System Administration Console.
3. Execute **cmreset -s** on the PM that will manage the partition created and started in Step 4. (Even if this PM is the same control processor as the System Administration Console, **cmreset -s** must be executed subsequent to executing **cmreset** with no switches.) 4. Execute **cmpartition create** and **cmpartition start**. (This obviates use of the **-S** flag, formerly used to specify the physical NI address of the CP. If the **-S** flag is specified, it is now ignored.) You need not start the **ts-daemon** when executing **cmpartition start**, but you can--**io_cold_boot** can run regardless of whether the timesharing daemon is running.

io_cold_boot takes several switches. In a standard configuration, no switches are required. However, the environment variables **CMDIAG_PATH** and **JTAG_SERVER** must be set. If they are not, **io_cold_boot** will print an error message and exit.

ARGUMENTS

-B	specifies the executable to download the IO kernel (default = **/usr/etc/io_download**).
-D	specifies the logical index of an IOBA channel that should be marked as offline or "down".
-i	specifies that only one IOBA (at the given NI address) is to be booted. The default is that all IOBAs listed in **io.conf** are booted.
-I	specifies the IO configuration file (default = **/etc/io.conf**).
-K	specifies the IOPN kernel executable file (default = /usr/etc/**io_kernel.hw**).
-l	sets the log level. This is a bit mask which specifies which modules should send messages to the log file. By default, only messages from error handlers are logged. To turn on more verbose logging, set this value to 0x108.
-L	sets the log file (default = **/dev/tty**).
-m	sets the memory size of the IOPN in megabytes (default = 8).
-T	sets the timeout to download one IOBA in seconds. This operation usually takes 45 seconds. The default timeout is set to 180 seconds.
-x	sets the value downloaded to the MMU control register on the IOPN. The default enables the cache in write-thru mode. To disable the cache, set this value to 0x1; to enable the cache in copy-back mode, set this value to 0x501.

SEE ALSO

ts-daemon(8), **cmpartition**(8)

NAME

viodiag - Executes diagnostic tests for a CMIOP or VMEIO host computer.

SYNTAX

viodiag [-m|-f|-ggroupname|-i|-c|-Efibldt|+Efibldt|-sfilename]

ARGUMENTS

-m	Execute manufacturing diagnostic tests for the CMI OP or VMEIO host computer.
-f	Execute field service diagnostic tests for the CMI OP or VMEIO host computer.
-g	Execute tests for groupname only.
-i	Include interactive tests.
-C	Enable command completion. +E Set diagnostic environment (activate options).
-E	Set diagnostic environment (deactivate options). f = loop forever i = ignore errors b = break on error (default) l = log errors (default) d# = display error count (default = 16) t = display trace messages
-s	Execute a viodiag shell file given by filename.

WHERE EXECUTED

CMIOP
VMEIO host computer

DESCRIPTION

viodiag is the diagnostic program for CM-IOPs and VMEIO host computers.

Command completion mode (-C option) lets you type the first first few letters of any viodiag command and use the ESC key to complete the command.

The -g option of viodiag allows you to run selected groups of tests. The test groups and their titles are listed below. To run a test group, enter its title at the command line. For example, to run all tests in the VME group in field mode, enter at the command line

viodiag -f -gVME

viodiag Test

Groups

CMIO-BUS-TEST CMIO-INTERRUPTS
test-cmio-slave-busy-nak test-cmio-no-arbitor-interrupt
test-self-target-select test-cmio-master-done-interrupt
test-sender-id-check test-cmio-slave-done-interrupt
test-cmio-overflow-timeout-interrupt
test-cmio-target-select-timeout
test-cmio-port-parity-interrupt
test-vmeio-generate-exception-interrupt

CMIO-FIFO-FLAGS CMIO-PORT-LOOPBACK
test-cmio-input-empty-flag test-cmio-data-bus
test-cmio-input-half-full-flag test-cmio-port-data-loopback
test-cmio-status
test-cmio-status-latches-control-bus-on-exp

CMIO-PARITY INTERACTIVE
test-cmio-port-parity-gen test-leds

MASTER-STATUS RAM-FIFO-FLAGS
test-vmeio-master-status-read-mode test-ram-fifo-empty-flag
test-vemio-master-status-write-mode test-ram-fifo-full-flag

RAM-PARITY REGISTERS
test-ram-parity-gen test-cm-data-bus-low
test-ram-parity-rams test-cm-data-bus-high
test-status-reg-aft er-reset
test-command-reg
test-setup-reg
test-vme-address-reg
test-vme-count-reg
test-read-pointer-reg
test-write-pointer-reg
test-word-count-reg
test-data-reg
test-ram-port-regis ter

SLAVE-FIFO-RAM SLAVE-MAPPED-RAM
test-fifo-ram-slave-wr-slave-rd test-unaligned-ram-transfer
test-mapped-ram

SLAVE-TIMEOUT VME
test-data-underflow-vme-timeout test-vme-data-bus
test-data-overflow-vme-timeout

VME-ADDRESS-GEN-TEST-MODE VME-ADDRESS-GENERATOR
test-vme-address-generator test-vme-address-gen-master-rd
test-write-pointer-increment test-vme-address-gen-master-wr
test-read-pointer-increment test-master-wr-shutoff-on-fifo-empty

VME-INTERRUPT VME-MASTER
test-vme-master-done-interrupt test-ram-loop-master-read
test-ram-parity-interrupt test-ram-loop-master-write
test-ram-loop-master-transfer

VME-MASTER-TIMEOUT VMEIO-CM-TRANSFERS
test-vio-master-read-vme-timeout test-data-transfer-vmeio-to-from-
test-vio-master-write-vme-timeout cmioc
test-vmeio-exception-generation
test-vmeio-slaveship
test-vmeio-exception-reception

Use viodiag -s to run diagnostic tests in a sequence and frequency that you select.

Construct a diagnostics shell file by creating a file, filename, of viodiag test commands. When viodiag -sfilename runs, the commands within filename execute.

For example, below is a diagnostics shell file, filename. When viodiag -sfilename executes, the two tests run.

```
test-vme-address-gen-master-rd 1024 m
test-vme-address-gen-master-wr 2048 m
quit
```

Executed with or without arguments, viodiag starts the diagnostic environment running, which is represented by the prompt, <vio-diag>

The tests listed within groups can be run individually at the <vio-diag> prompt. In addition, the following tests can be run individually:

write-vmem-byte write-pmem-long
wr-rd-ver-cmio-setup-req write-register
wr-rd-ver-data-req wr-rd-ver-command-req
wr-rd-ver-read-pointer-req wr-rd-ver-ram-port-req
wr-rd-ver-vme-address-req wr-rd-ver-setup-req
wr-rd-ver-word-count-req wr-rd-ver-vme-count-req
wr-rd-ver-write-pointer-req

In addition to tests, at the prompt you can also run the following commands:

alias command_name Display the alias for command_name.
continue-from-abort Continue to run the test sequence.
help Display a help menu.list-commands Display all viodiag commands and tests.
list-groups Display the viodiag test groups.
reset Do a hard reset of the CMIOP or VMEIO host computer.
set-diag-environment Set the diagnostic environment variables.

show-diag-environment Display the setting of the diagnostic environment variables.
show-all-errors Display all the errors generated by this execution of viodiag.
whatis command_name Display a brief man page for command_name.

The following troubleshooting utilities can also be run at the <vio-diag> prompt:

display-board-status read-pmem-block
display-dram-status read-register
display-fifo-status read-vmem-byte
display-ram-fifo-contents display-cmio-status
cmio-read-register search-dram-contents
display-interrupt-status display-ram-status
read-pmem-long read-vmem-block

RESTRICTIONS

Do not run viodiag manufacturing tests if the CMIOP or VMEIO host computer is connected to any CMIO bus -- the entire CMIO bus system will be unpredictably affected. (These tests may change the CMIOP's or VMEIO host computer's status, that is, its station ID and arbiter.)

Be sure there is an arbiter on the bus before running viodiag in field mode.

SEE ALSO

viodiag
dvcoldboot
fsserver