April 1991
Order Number: 311014-006

iPSC®/2 and iPSC®/860
SYSTEM ADMINISTRATOR'S GUIDE

intel® Corporation

REV.	REVISION HISTORY	DATE
-001	Original Issue	12/87
-002	Revision	03/88
-003	Revision	08/88
-004	Revision	10/89
-005	Revision	06/90
-006	Revision	04/91

RESTRICTED RIGHTS

PREFACE

This manual tells how to use the following Intel Supercomputer Systems Division products: iPSC®/2, iPSC®/2S, iPSC®/860, iPSC®/860S, and iPSC®/860Plus.

NOTE

This manual uses the term "iPSC system(s)" to refer to these products.

A system administrator for an iPSC system takes care of the UNIX operating system on the System Resource Manager (SRM) and the NX/2 operating system on the nodes. To perform iPSC system administrative tasks, you must have root privilege on the System Resource Manager.

This manual assumes that you already know how to administer a UNIX system and that you are familiar with the iPSC system. The manual is not an introduction to the iPSC system; it describes the tasks necessary to administer an iPSC system.

ORGANIZATION

Chapter 1	"Introduction," presents an overview of the iPSC system (hardware and software), and summarizes the tasks that a system administrator must perform to maintain this hardware and software.
Chapter 2	"Powering up, Powering down," tells how to turn the iPSC system on and off, and how to shut down the UNIX operating system gracefully.
Chapter 3	"Installing Software," directs the system administrator to the release notes for installation instructions for system software and options.
Chapter 4	"Remote Host," tells how to use the cube from a remote host workstation.

Chapter 5 "Booting the Cube," describes the cube and device configuration files and tells how to load and run the NX/2 operating system on the nodes.

Chapter 6 "System Administrator Tasks," provides step-by-step procedures for performing the most common system administrator tasks.

Chapter 7 "Running Diagnostics," provides step-by-step procedures for running and interpreting the iPSC diagnostic software (Power-On Self Test, Node Confidence Test, and Cube Diagnostic Program).

Chapter 8 "Commands and System Calls," describes (in reference manual format) the system administrator's commands and system calls.

NOTATIONAL CONVENTIONS

This manual uses the following notational conventions:

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

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

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

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

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

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

[ ] (Brackets) Surround optional items.

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

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

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

APPLICABLE DOCUMENTS

For more information, refer to these iPSC, Intel, and other manuals:

iPSC® System Manuals

iPSC®/2 and iPSC®/860 Network Queueing System Manual
Describes and tells how to use the Network Queueing System (NQS) software to queue and manage batch/device processes in the iPSC system.

iPSC®/2 and iPSC®/860 Programmer's Reference Manual
Describes iPSC system commands and system calls (both C and Fortran).

iPSC®/2 and iPSC®/860 System Acceptance Test User's Guide
Tells how to use the System Acceptance Test.

iPSC®/2 and iPSC®/860 User's Guide
Overviews the iPSC system, including hardware and software architectures. Tells how to develop and run programs.

iPSC®/2 Simulator Manual
Tells how to use the iPSC/2 Simulator for software development.

iPSC®/860 C Compiler User's Guide
Tells how to use the iPSC/860 C compiler driver.

iPSC®/860 Fortran Compiler User's Guide
Tells how to use the iPSC/860 Fortran compiler driver.

Intel® Manuals

UNIX System V Release 3.2 NFS User's/System Administrator's Guide and Reference
Describes the NFS programming environment and provides user and system administration information.

UNIX System V Release 3.2 NFS Programmer's Guide and Reference
Describes the NFS programming environment and tools.

UNIX System V Release 3.2 TCP/IP Administrator's Guide and Reference
Describes TCP/IP Network administration.

UNIX System V Release 3.2 TCP/IP Programmer's Guide and Reference
Describes the TCP/IP Network programming environment and provides information on programming tools.

UNIX System V Release 3.2 TCP/IP User's Guide and Reference
Describes the TCP/IP Network programming environment and provides user information.

i860™ 64-Bit Microprocessor Assembler and Linker Reference Manual
Tells how to use the i860 assembler and linker.

i860™ 64-Bit Microprocessor Programmer's Reference Manual
Tells how to use the i860 microprocessor.

SYP301 Installation and User's Guide
Tells how to install and start the System Resource Manager. Also provides hardware technical data.

Other Manuals

C: A Reference Manual - Harbison and Steele
Describes the C programming language.

The C Programming Language - Kernighan and Ritchie
Describes the C programming language.

The X Window System Manual Set
Describes the X Windows System application programming.

UNIX System V Manual Set
Describes UNIX System V.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

CHAPTER 2
POWERING UP, POWERING DOWN

CHAPTER 3
INSTALLING SOFTWARE

CHAPTER 4
REMOTE HOST

CHAPTER 5
BOOTING THE CUBE

CHAPTER 6
SYSTEM ADMINISTRATION TASKS

CHAPTER 7
RUNNING DIAGNOSTICS

CHAPTER 8
COMMANDS AND SYSTEM CALLS

LIST OF ILLUSTRATIONS

INTRODUCTION 1

INTRODUCTION

This chapter presents an overview of the iPSC/2 and iPSC/860 systems (hardware and software), and summarizes the tasks that a system administrator must perform to maintain hardware and software. Other than the type of compute nodes, the systems are the same. Unless specific differences are being described, the systems are referred to as iPSC systems.

HARDWARE

The major hardware components of iPSC systems are:

- System Resource Manager (SRM)
- Standard cabinet
- Optional compact cabinet

An iPSC system always has at least an SRM and computational nodes that are housed in either a compact cabinet or a standard cabinet. The cabinets house peripheral devices and nodes. A system may consist of compact cabinets only, standard cabinets only, or both compact and standard cabinets. A system with the Concurrent I/O facility (CIO) has at least one standard cabinet to contain the CIO; the compute nodes may be housed in either standard or compact cabinets.

Figure 1-1 shows the major hardware components of an iPSC system.

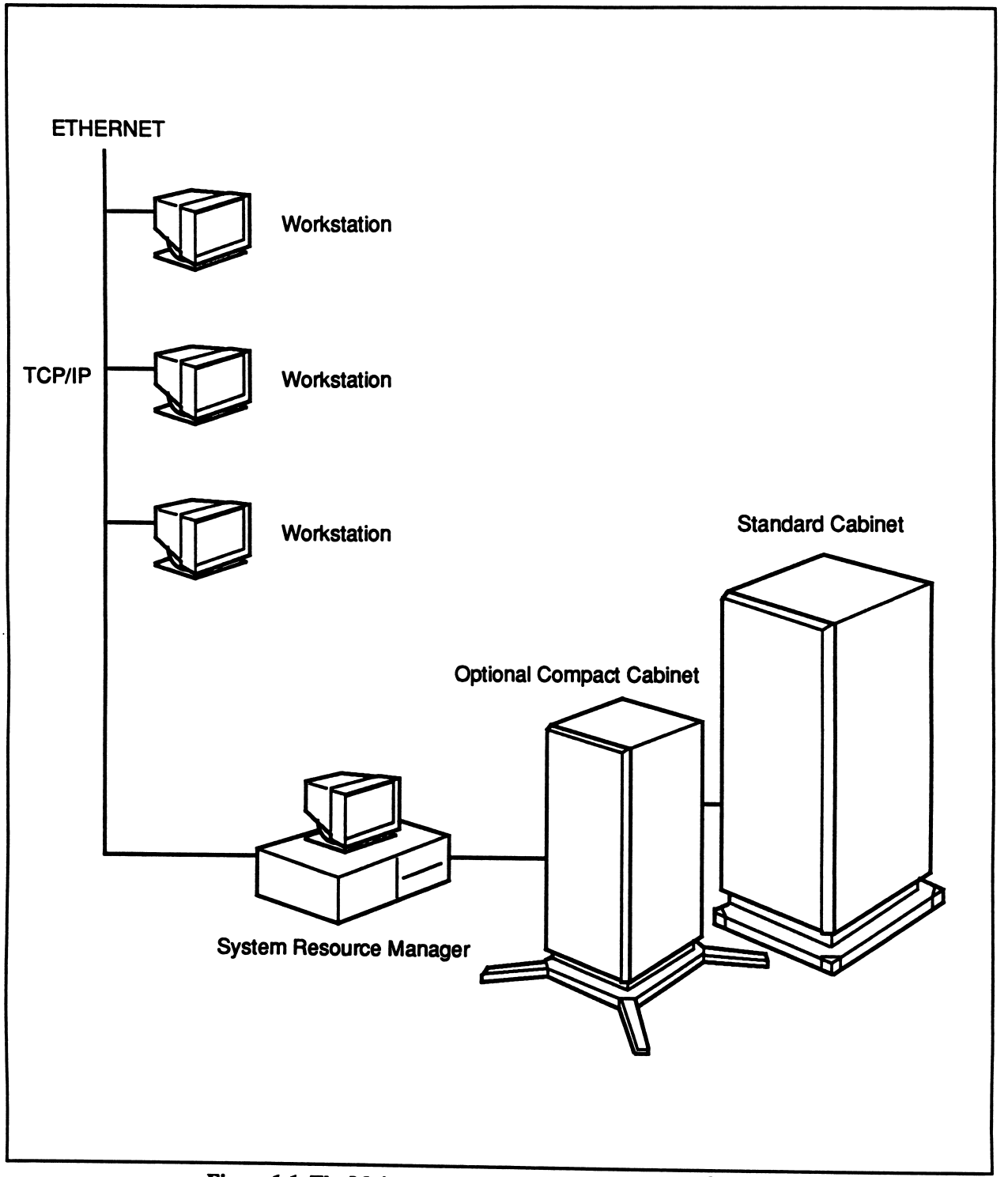

Figure 1-1. The Major Physical Components of an iPSC® System

The System Resource Manager

The SRM, a computer that is based on the Intel386™ microprocessor, provides the interface to iPSC operations. The SRM is also called the local host.

The SRM has 8.5M bytes of memory, a 380M-byte hard disk, a 1.2M-byte floppy drive, and a 60M-byte cartridge tape drive. The SRM's hard disk can be accessed by programs running on the SRM, programs running on remote hosts (with NFS), and programs running on the nodes.

The system administrator can either work directly at the SRM, or can log in to the SRM from a remote workstation. Remote workstations can be linked to the SRM over the ethernet. Remote workstations that have Remote Host Software installed are also called Remote Hosts. Supported Remote Hosts may also have i860™ Cross Compiler software loaded to allow users to compile C or Fortran node programs at the remote workstation. Refer to Chapter 1 of the *iPSC®/2 and iPSC®/860 Release 3.3 Software Product Release Notes* for a list of the Remote Hosts that can use i860™ Cross Compiler software.

Nodes

A node is a processor/memory pair. Each node's memory is distinct from the host and from other nodes. Each node runs the NX/2 operating system, communicates with the other nodes (by passing messages), and can access both the host file system and the iPSC Concurrent File System™ (CFS).

There are two types of compute nodes available: RX nodes and CX nodes. RX nodes are based on the i860™ microprocessor, and CX nodes are based on the Intel386 microprocessor. Systems consisting only of CX compute nodes are referred to as iPSC/2 systems; systems whose compute nodes are either all RX or a combination of RX and CX nodes are referred to as iPSC/860 systems.

The collection of nodes belonging to an iPSC system is referred to as a d*n* where *n* is the dimension of the largest possible cube. For example, a d7 has 2^7 or 128 nodes.

Each node is fully connected to every other node through its Direct-Connect Module™ (DCM). With the DCM, there is no "store and forward" of messages. Nodes that are nearest neighbors have nearly the same message latency as those that are not.

Each DCM provides eight channels (channel 0 through 7). Up to seven channels (channels 0 through 6) are connected to nearest neighbors. Each node in a d7 (the largest iPSC system) has seven nearest neighbors and uses all seven of the available DCM channels.

DCM channel 7 has a special use. Channel 7 on node 0 connects to the SRM. This gives the SRM the same high-speed hardware access to the DCM network as the nodes themselves. In a system that includes a Concurrent I/O (CIO) System, channel 7 on the I/O nodes is connected to CIO.

NODE OPTIONS

Nodes have the following options:

Memory An RX node can have 8M bytes, 16M bytes, 32M bytes, or 64M bytes of memory. A CX node can have 1M byte, 4M bytes, 8M bytes, or 16M bytes of memory. The memory resides on node daughter boards. If a CX node has 16M bytes, it requires two daughter boards. The second daughter board blocks the adjacent slot, preventing you from placing another node board there. Because node 0 must reside in slot 0, any 16M-byte node will block an odd slot. If even one odd slot is blocked, then all nodes above that slot must reside in even slots. Even the unblocked slots cannot be used.

Numeric Processing
The i860 microprocessor of the RX node has inherent fast numeric processing ability.

All CX nodes are equipped with a Intel387™ numeric coprocessor. You can replace the Intel387 numeric coprocessor with an SX processor, a hardware option for high-speed floating point scalar arithmetic.

Vector Processing
The i860 microprocessor of the RX node has inherent fast vector processing ability.

To enhance the vector-processing capability of CX nodes, you can also add a VX processor, a hardware option for high-speed vector arithmetic. VAST2, a preprocessor that reads standard Fortran programs and substitutes the appropriate vector routines, is offered for Intel386-based nodes.

Nodes equipped with a VX processor are called VX nodes. Each VX node is limited to 8M bytes and requires two card slots.The VX processor is a separate board and plugs into an adjacent slot. Because node 0 must reside in slot 0, the VX boards reside in odd-numbered slots. The VX board for node 0 (slot 0) resides in slot 1; that for node 1 (slot 2) resides in slot 3, etc.

If an iPSC/2 system has any VX boards, all node boards must reside in even slots only.

NODE TYPES

There are three kinds of nodes:

Compute Nodes Compute nodes (both CX and RX nodes) are suited for computational tasks.

I/O nodes I/O nodes are based on CX compute nodes. Standard I/O nodes, which are connected to disk and tape drives in the CIO system) have a SCSI (Small Computer System Interface) controller and a modified DCM (channel 7 only). Each I/O DCM connects to channel 7 of an associated compute or integrated node. The CIO Ethernet and VME Bus Interface Adapter (BIA) options use special I/O nodes with a PBX bus and no SCSI controller.

Integrated nodes Integrated nodes are based on CX nodes, and combine compute and I/O node functions.

If your system does not have CIO, the CIO Ethernet option, or the VME Bus Interface Adapter (BIA) option, your iPSC system has only compute nodes.

Compute nodes can reside either in compact or standard cabinets. I/O nodes and integrated nodes must reside in standard cabinets.

Concurrent I/O

Until the introduction of the Concurrent I/O system, there were two major bottlenecks for some applications:

- Getting information into and out of the system
- Limited amount of available mass storage

Without CIO, the only mass storage available to the nodes is the host's disk. With NFS networking software, other disks become available, but node-to-host communication still limits applications.

CIO provides each node with access to the Concurrent File System whose array of disks appears as a single virtual disk that several node processes can access simultaneously. Available file storage is in excess of 40G bytes.

All I/O hardware must be installed in a standard cabinet. This hardware includes I/O nodes, integrated nodes, peripheral devices, and VME Bus Interface Adapter (BIA) boards.

The I/O and integrated nodes provide a SCSI interface that allows up to seven SCSI drives. The standard CIO configuration has two disk drives for each SCSI controller.

The BIA board is an interface between an I/O node board (or integrated node) with a PBX interface and a VME controller board. (The VME bus is a general purpose bus used to connect computer devices.) The BIA resides in a slot adjacent to the node board, connected to the node board's PBX interface, and the VME controller plugs into the BIA. If a system consists of integrated nodes in a standard cabinet, and there are any BIA boards, then the integrated node boards must reside only in even slots.

The Standard Cabinet

There are a number of packaging options based on the standard cabinet that correspond to various combinations of cardcage modules and peripheral modules. The densest option is a standard cabinet with five modules (one cardcage module and four peripheral modules). If you choose an option with two cardcage modules, there is room left for two peripheral modules. The standard cabinet can hold up to four cardcage modules with no peripheral modules (this configuration produces the highest heat dissipation).

The cardcage in a standard cabinet is a single row with 17 slots. One slot holds the Unit Services Module (USM). The other 16 slots contain node boards, VME Bus Interface Adapter (BIA) boards, or vector boards. The peripheral module can contain up to four 760M-byte disk drives or 2G-byte tape drives.

The Optional Compact Cabinet

Each compact cabinet contains one 34-slot cardcage that can hold up to 32 compute nodes. There can be up to four compact cabinets in an iPSC system.

The cardcage has an upper row and a lower row:

- The upper row contains nodes 0 through 15 (first cabinet), 32 through 47 (second cabinet), 64 through 79 (third cabinet), and 96 through 111 (fourth cabinet). The rightmost slot in the upper row of each cabinet contains the USM, which is used for booting and diagnostic purposes.
- The lower row contains nodes 16 through 31 (first cabinet), 48 through 63 (second cabinet), 80 through 95 (third cabinet), and 112 through 127, (fourth cabinet). The rightmost slot in the lower row of each cabinet is an empty slot that is reserved for future use. In the latest backplane versions, the empty slot has power and must not be used to store spare nodes.

SOFTWARE

The iPSC software consists of:

- Operating systems (for SRM and nodes)
- Networking software
- iPSC extensions (commands, libraries, and background processes)
- Diagnostic programs

Operating Systems

The SRM runs the UNIX System V operating system with iPSC extensions that interface to the nodes. These extensions let you allocate and deallocate cubes, load programs on the nodes, and obtain node status information.

For information on how to administer the UNIX operating system, refer to the *UNIX System V/386 Release 3.2 System Administrator's Guide* and the *UNIX System V/386 Release 3.2 System Administrator's Reference Manual.*

Each node runs the NX/2 operating system, which provides message-passing capability, memory management, and process management. The NX/2 operating system also manages the numeric coprocessor and optional vector processor.

Networking Software

The SRM and remote workstations also run the TCP/IP networking software. This software lets you remotely log into other systems on the network and transfer files between systems. For example, a typical system administrative task is to boot the NX/2 operating system on the nodes. This must be done on the SRM. TCP/IP lets you remotely log into the SRM (from a remote workstation) to issue the **bootcube** command.

Optionally, the SRM and remote workstations can run the NFS networking software. NFS allows files to be shared transparently over a local network. NFS lets you access files on other workstations as though they were resident on your own workstation, thus eliminating the need to log into another system and copy the files.

Another networking option is the CIO Ethernet option. This consists of TCP/IP running on the RX nodes of the iPSC system, and X Window client libraries that allow you to run and develop X Window applications.

iPSC® Extensions

The iPSC extensions contain application libraries, shell commands, and several background processes. The libraries provide iPSC system calls that are available to both host and node programs. Several background processes enable message passing between the nodes and the host. A typical background process is a file server (called *fserver*) that runs on both the SRM and remote hosts. The *fserver* process allows the node to run standard I/O functions on the host.

An iPSC application can have a host program that runs on either the SRM or a remote host, and a node program that runs on a group of allocated nodes called a cube. iPSC commands let you control the allocation and operation of a cube.

This manual assumes that you are familiar with the iPSC commands available to regular users. As system administrator, you have access to the following additional commands:

bootcube Loads the NX/2 operating system on the nodes (when issued on the SRM). Starts up the communications daemon (when issued on a remote host). Discussed in Chapters 5 and 8.

plogon and **plogoff**
These commands turn logging of process-related information on and off. Discussed in Chapters 6 and 8.

mkcfs Makes (or remakes) a concurrent file system. Discussed in Chapters 6 and 8.

mkdev Allocates a file which represents a tape drive managed by the concurrent file system. Discussed in Chapters 6 and 8.

cbackup and **crestore**
These commands run only under the node shell (nsh), and are used to back up and restore a concurrent file system. Discussed in Chapters 6 and 8.

These commands require root privilege.

As system administrator, you also have access to additional switches on **relcube** that let you release cubes belonging to other users. These additional switches are described in the *iPSC®/2 and iPSC®/860 Programmer's Reference Manual.*

Diagnostic Programs

Two diagnostic programs are automatically run on power-up. The Power-On Self Test (POST) runs automatically when the SRM is powered up. The Node Confidence Test (NCT) runs automatically when each node is powered up.

For additional testing, run the Cube Diagnostic Program (CDP). This program checks all the communication links (node-to-node and node-to-SRM). CDP also tests VX boards, the SX processors, Intel387 numeric processor, SCSI modules, and hard disks.

The system administrator is often the person who calls the Intel Service Representative when a failure occurs. At that time, you may be asked to run CDP to obtain further information about the failure.

SYSTEM ADMINISTRATION TASKS

The system administrator is not necessarily the most knowledgeable person about the iPSC system. However, the system administrator is the person with the most responsibility.

The system administrator is the "keeper of the root password." Because root actions can affect the entire system, you should reveal the root password to others only when necessary. And when those others no longer need root access, you should change the root password. If several people are doing root-level activity and not communicating with each other, a system's configuration can easily enter into an unknown state. The system administrator should know the system's configuration at all times.

You should also treat root activity with care. Use root privilege only when it is really necessary to do so. If you can perform a task without root privilege, you should do so. This minimizes the consequences of any errors you may make.

iPSC-specific system administrative tasks are as follows:

- Powering up and powering down (Chapter 2)
- Reinstalling and updating iPSC software. General information is provided in Chapter 3, but specific installation instructions are in the product release notes.
- Remote host information (Chapter 4)
- Booting the cube (Chapter 5)
- Establishing network addresses for TCP/IP nodes (Chapter 6)
- Releasing cubes belonging to other users (Chapter 6)
- Removing the root password (Chapter 6)

- Making a new Concurrent File System (Chapter 6)
- Backing up and restoring CFS files (Chapter 6)
- Preventive maintenance (Chapter 6)
- Running diagnostic programs and interpreting the results (Chapter 7)

POWERING UP, POWERING DOWN 2

INTRODUCTION

This chapter describes how to power up an installed iPSC system, shut down the system gracefully, and turn off the power.

POWERING UP

When powering up the system, first power up the System Resource Manager, and then the nodes.

Powering up the System Resource Manager

1. Turn on the monitor (the switch is located on front of the monitor).

2. Turn on the SRM computer (the switch is located on the right side near the back of the unit).

   If the monitor is not illuminated, check the monitor's brightness adjustment, verify that the monitor's AC power cord and signal cable are plugged into the back of the SRM, and that the SRM's AC power cord is connected to line power.

3. When power is applied, the Power-On Self Test (POST) runs automatically. It checks the 80386 processor, keyboard, display, system memory, and most peripheral devices. (Refer to the *SYP301 Installation and User's Guide* for more information about this test.) The following message appears:

   ```
   Booting the UNIX System ...
   ```

   If you receive any error messages, refer to Appendix B in the *SYP301 Installation and User's Guide*.

4. The system is booted when the following prompt appears:

   ```
   Console Login:
   ```

Powering up the Nodes

1. Move the circuit breakers located at the bottom rear of each cabinet to the ON position (to the left as viewed from the rear). This supplies the system with AC power. You must turn this switch to the ON position for every cabinet.

2. Verify that the AC power indicator light (located at the top front of each cabinet) are on. Standard cabinets have one small green indicator light for each card cage (up to four). Compact cabinets have a single amber light indicating AC power.

3. Turn the key switch located at top front of the compact cabinet to the vertical (ON position). This supplies the system with DC power. Standard cabinets do not have a key switch.

NOTE

In systems with multiple compact cabinets, only one key switch needs to be turned to the ON position to supply power to the rest of system. If you turn more than one key switch to ON, you must turn the key switches on every compact cabinet to the OFF position to turn system power off.

4. Verify that the green (DC) indicator light (located at the top front of each unit) is on.

5. When power is applied, the Node Confidence Test (NCT) runs automatically. Verify that the red indicator light on the node boards turns on as the NCT is run in parallel on all nodes. The green light turns on, and the red light turns off when the node passes the test successfully (except for the Unit Services Module).

   If the nodes do not pass their NCTs, refer to the section, "Node Confidence Test (NCT)" (page 7-4) in Chapter 7, "Running Diagnostics."

POWERING DOWN

Although the goal of a system administrator is to keep the system up and running, there are times when you need to power it down. For example, you should power a cabinet down when you swap a node board. Also, you should power your systems down when you receive notice that building power is about to be shut off.

Stopping the UNIX Operating System

1. Login as ***root.***

2. Use the following command to ensure that you are in the root directory:

```
# cd /
```

3. Be sure everyone is logged off. You can check this with the **who** command.

CAUTION

If the system is stopped while users are logged on, all running user processes will be killed.

4. Invoke the **shutdown** command. To use the default 60-second grace period and have the confirmation questions asked, enter the following:

```
# shutdown
```

Refer to the *UNIX V/386 System Administrator's Reference Manual* for more information about **shutdown**.

Removing Power

When the message `Reset the CPU to reboot` appears:

1. Turn off the system monitor (the switch is located on the front).

2. Turn off the SRM computer (the switch is located on the right side near the back of the unit).

3. Turn the key switch located at the top front of each compact cabinet clockwise to the horizontal (OFF) position.

4. Move the circuit breakers located at the bottom rear of each unit to the OFF position (to the right as viewed from the rear). You must turn this switch to the OFF position for every cabinet.

INSTALLING SOFTWARE 3

INTRODUCTION

System software is installed originally by Intel Supercomputer Systems Division Customer Engineers. You need to re-install your software **only** if the software on your system becomes damaged. If that happens, please call SSD Customer Support before performing any installations:

1-800-421-2823 (Customer Support Hotline)
(44) 793 641 469 (in England)
Your Local Intel Sales Office (in Europe)
support@ssd.intel.com (Internet address)

If it should become necessary to re-install the software, the re-installation instructions are in the appropriate Product Release Notes. The current iPSC System Software Product Release Notes contain descriptions of how to install the following software:

- UNIX System V/386 software
- TCP/IP and Ethernet Drivers for the SRM
- iPSC Software
- Remote Host Software on a Remote Host

CAUTION

When iPSC software is installed or re-installed, the Cube Diagnostic Program (CDP) and System Acceptance Test (SAT) should be run to verify proper installation. Make sure that all of the CDP tests pass before running the SAT. Both the CDP and SAT test must pass before the installation can be considered successful.

Descriptions of how to re-install options such as the Network File System (NFS), Lisp, Ada, etc., are contained in the release notes (or customer letter in products that do not require a full set of release notes) for the current version of that product.

If you perform the installation procedures exactly as described in the current release notes, you will create a software system that is identical to the one originally installed by SSD.

REMOTE HOST 4

RUNNING REMOTE HOST SOFTWARE

To use the cube from a remote host workstation, you must start a remote *commser* on the workstation:

1. Login as ***root.***

2. To start a *commser*, execute **bootcube** or **rebootcube** on the remote host.

Once the remote *commser* is running, you can get cubes and run applications.

COMPILING REMOTE HOST APPLICATIONS

NOTE

The Sun workstation and the iPSC system have different internal representations of integer and floating-point numbers. See the descriptions of the **createstruc** and **CTOH** utilities in the *iPSC®/2 and iPSC®/860 Programmer's Reference Manual* for information on converting messages to and from cube-byte order.

To compile existing iPSC SRM applications, the makefiles must be modified as follows:

1. If LIBDIR in the Remote Host software source makefile is defined as either */lib* or */usr/lib*, then replace the **-host** switch with **-lhost**. Otherwise, the entire path of the host library must be specified.

   For example, if LIBDIR is defined to be */usr/myusername/lib*, you must link with */usr/myusername/lib/libhost.a*.

2. Change the node compilation rule for **cc** and **f77** to **rcc** and **rf77**, respectively.

3. Change **ld**, **ar**, and **as** to **rld**, **rar**, and **ras**, respectively. For details, refer to the *iPSC®/2 and iPSC®/860 Programmer's Reference Manual.*

REMOTE HOST SECURITY

User access to remote host facilities is defined in the file */usr/ipsc/lib/ruser*. This file contains lists of workstations and users that are allowed remote host access for the iPSC system.

Each line of the */usr/ipsc/lib/ruser* file starts with the *hostname* of a remote host workstation. The remaining entries on the line (each separated by a space) are the user names or group names of users that are allowed remote host access from the *hostname* identified at the start of the line. The following example shows three remote hosts (*hostname*) and the users that are authorized to gain cube access from that remote host.

```
enterprise johnd ev_engr_grp
intrepid janed ev_engr_grp
midway georgeg alm fredf
```

Note that the first two remote hosts each have separate single users while also sharing the same group. The final remote host has only three separate users authorized to gain cube access from that remote host.

If any of the users named has an entry in the */etc/passwd* file, commands such as **rcc**, **rf77**, and **rld** (when executed by **ripscd**) will execute with a *uid* equal to the *uid* found with their entry in */etc/passwd*. If the named user has no entry in */etc/passwd*, the commands executed by **ripscd** will execute with a *uid* equal to the *uid* found in */etc/passwd* for **ripscd**.

BOOTING THE CUBE 5

INTRODUCTION

Booting the cube means loading the NX/2 operating system on the nodes and starting the communications daemons on the SRM. Use the **bootcube** command to boot the cube.

This command has two versions—one that runs on the SRM, and one that runs on a remote host:

- When you run **bootcube** on the SRM, it loads the NX/2 operating system on the nodes and starts up the SRM communications daemons.
- When you run **bootcube** on a remote host, it starts up the remote host's communications daemon.

Only the superuser can use **bootcube**. There is a user version of the **bootcube** command called **rebootcube**. This command operates like **bootcube**, but it has no options.

Options for **bootcube** include such capabilities as loading the NX/2 operating system onto a cube of smaller than maximum dimension, forcing a boot even if some cubes are still allocated, and executing only a portion of the boot sequence.

This chapter describes the full capability of the **bootcube** command.

Chapter 7 contains a formal description (in reference manual format) of the **bootcube** command.

THE CUBE CONFIGURATION FILE

The cube configuration file is called *cubeconf* and it is located in */usr/ipsc/conf* on the SRM. It is a text file that describes the hardware configuration of the iPSC system. The **bootcube** and the cube diagnostic programs read *cubeconf*. Refer to the remainder of this section for more information about *cubeconf*.

Before your iPSC system was shipped, *cubeconf* was set with values describing your configuration. Unless you change the hardware configuration of your system, you should not need to change this file.

Figure 5-1 shows a typical *cubeconf* file. It describes a d6 (64-node) iPSC/860 system with four I/O nodes. There are two standard cabinets in this system. One cabinet has four cardcages with a total of 64 compute nodes, and the other cabinet contains the I/O nodes and peripherals. The header information in the file describes the meaning of the fields.

```
/*                                  Cube
 *                             Configuration File
 *
 * @(#)Release 3.2
 *
 * cubeconf 7.15 90/05/03 09:24:16
 *
 * Copyright (c) 1987,1988,1989,1990 Intel Corporation
 */

/* KEYWORD                       DESCRIPTION
 *
 * cardcages              Total number of cardcages, 0-32
 *
 * slots                  Number of compute slots, 0-128
 *
 * baud                   Diagnostic link bit rate, 19200 or 38400
 *
 * backplane_rev          Backplane revision letter, A or B, for old USM's
 *
 *
 * USM Configuration:              (USMn, n is USM number)
 *
 * USMn xyyb.s            where:
 *                          xyy is USM revision/USM firmware version
 *                          b is backplane revision
 *                          s is the number of slots in the cardcage
 *
 *
 * Cardcage Configuration:         (CCn, n is USM number)
 *
 * CCn  id0,id1,id2,id3,id4,id5,id6,id7,id8,id9,id10,id11,id12,id13,id14,id15
 *
 *                        where each id is the network address of the node
 *                        in the corresponding slot, or blank if empty.
 *                        CC lines are needed only for I/O nodes.
 *
```

Figure 5-1. Cube Configuration File for a d6 iPSC®/860 System (1 of 4)

```
 *
 * Slot Configuration:          (Sn, n is slot number)
 *
 * Sn    pppNODEmdf       node board, where:
 *                          ppp is processor type, 386 or 860
 *                          m is memory size in megabytes (1, 4, 8, or 16)
 *                          d is DCM version (A-H)
 *                          f is floating point or I/O options:
 *                                7               - 387
 *                                SX or SXA       - Weitek 1167 or Weitek 1167A
 *                                SCSI            - SCSI interface
 *
 *       FAILEDmdf        inoperative node board.
 *
 *       VXnnnn           vector board, nnnn is:
 *                        board revision/program memory/ram option/alu option
 *
 *       MX               4 megabyte memory board.
 *
 *       EMPTY            slot is ignored.
 *
 *       VMEv.xxxx        Bus Interface Adapter to VME board xxxx
 *
 */

baud    38400
```

Figure 5-1. Cube Configuration File for a d5 iPSC®/860 System (2 of 4)

In the system described in this cubeconf file, all of the nodes are RX node boards with 8M bytes of memory, and DCM revision D. Three of the slots in the cardcage in the I/O cabinet have I/O nodes. The device configuration file (described later) shows that each I/O node is connected to two SCSI (Small Computer System Interface) disk drives.

If you take a board out and do not insert another board, enter *EMPTY* for the board's slot value. This allows the system software to partition cubes around an empty slot. The cube partitioning works best if nodes fill the lowest numbered slots. Hence, if you must remove a node, replace it with the highest numbered node and mark that highest numbered slot **EMPTY**.

Cardcages and the Number of Slots

The *cubeconf* file indicates the number of cardcages in the iPSC system. The identifier is *cardcages*. This number begins at 1 and is equal to 1 more than the highest numbered USM in your system.In the example in Figure 5-1, the system has four cardcages in one cabinet and one cardcage in the other cabinet. It has 5 USMs, one for each cardcage. The USM number increments for every 16 slots.The identifier *cardcages* is equal to 5.

The file also indicates the total number of compute slots in an iPSC system. The cardcage in a compact cabinet has 34 slots, but only 32 can hold node boards — the other two (one for the USM and the other reserved) are unnumbered. In a standard cabinet, each cardcage has 17 slots, and 16 can hold node boards, with the remaining slot (for the USM) unnumbered. Note that the *slots* indicator represents the total number of compute slots, whether or not the slot actually contains a compute node. The example in Figure 5-1 has a total of five cardcages, of which only four are compute cardcages, with a total of 64 compute slots.

The USM and CC Entries

The USM*n* entry identifies a cardcage. Each cardcage has its own USM. *n* is the USM ID and is jumpered on the USM itself. It starts at 0 and increments by one for every 16 slots in the system.

If the standard cardcage has I/O nodes, it must also have a CC*n* entry. The *n* in CC*n* corresponds to the *n* in the USM*n* entry for the cardcage containing the I/O nodes. The CC*n* line has a field for each slot in the standard cardcage. Commas separate the fields. If the slot contains an I/O node, its CC*n* field contains the I/O node's DCM address. The DCM address is:

128 +	the physical node number of the compute node whose channel 7 is connected to the I/O node

In general, if the slot contains a compute node, an integrated node, or a BIA board, the CC field is blank.

In the example in Figure 5-1, cardcage 4 (the cardcage in the I/O cabinet) contains only I/O nodes. The DCM address of the first I/O node is 138. This I/O node is connected via channel 7 to the node in slot 10 (138 – 128). The second slot in the standard cabinet contains the BIA board. The third and fourth slots contain I/O nodes connected via channel 7 to the nodes in slot 26 (154 – 128) and slot 42 (170 – 128) respectively.

```
cardcages       5

slots   64

USM0    A21H.16
USM1    A21H.16
USM2    A21H.16
USM3    A21H.16
USM4    A21H.16

CC4     138,,154,170,,,,,,,,,,,,
```

Figure 5-1. Cube Configuration File for a d5 iPSC®/860 System (3 of 4)

Slot Assignments

Slot assignments are identified as S*n* where *n* is the slot number. The slot numbers begin with 0 and increment sequentially. For example, compact cabinets contains slots 0 through 31. If the iPSC system has another compact cabinet, its first slot is slot 32. Standard cabinets can contain up to four cardcage modules, each with 16 slots. Only 64 of those slots per cabinet may contain compute nodes.

The slot assignment field for a compute node identifies the node type, the memory size, and the DCM version of the node. For CX nodes, it also indicates whether the node has an Intel387™ numeric coprocessor or an SX scalar processor.

The example following shows. Slots 0 through 63 have 860NODE8F as their slot assignments. 860NODE8F means that a node has an i860 microprocessor, 8M bytes of memory, and DCM revision F. The slot assignment designations are described in the file header. As another example, 386NODE4D7 is the designation of a node with a Intel386™ microprocessor, 4M bytes of memory, DCM revision D, and an Intel387 numeric coprocessor. If the node had an SX module instead of an Intel387 numeric coprocessor, the assignment value would be 386NODE8DSX.

```
S0      860NODE8F
S1      860NODE8F
S2      860NODE8F
S3      860NODE8F
S4      860NODE8F
S5      860NODE8F
S6      860NODE8F
S7      860NODE8F
S8      860NODE8F
    .
    .
    .
S55     860NODE8F
S56     860NODE8F
S57     860NODE8F
S58     860NODE8F
S59     860NODE8F
S60     860NODE8F
S61     860NODE8F
S62     860NODE8F
S63     860NODE8F
S64     386NODE4E7SCSI
S65     386NODE8E7SCSI
S66     386NODE4E7SCSI
S67     EMPTY
S68     EMPTY
S69     EMPTY
S70     EMPTY
S71     EMPTY
S72     EMPTY
S73     EMPTY
S74     EMPTY
S75     EMPTY
S76     EMPTY
S77     EMPTY
S78     EMPTY
S79     EMPTY
```

Figure 5-1. Cube Configuration File for a d5 iPSC®/860 System (4 of 4)

The slot assignment field for an I/O or integrated node indicates SCSI in the last field of the designation.

Instead of a node board, a slot may contain a VX board or a BIA board. Either of these must reside in an odd-numbered slot and is associated with the node board in the lower even slot. For example, a VX board in slot 1 must be associated with the node board in slot 0, and a BIA board in slot 67 would be associated with the I/O node in slot 66.

The BIA board is in an odd-numbered slot. If the standard cabinet containing the CIO contained compute or integrated nodes in addition to the BIA board, they would have to be in even-numbered slots. I/O nodes, however, can reside in any slot. In this example, however, the cardcage contains I/O nodes only, so they are packed tightly and are not restricted to even-numbered slots.

bootcube Will Create Slot Assignment Values

If the slot assignments are not listed in the file, **bootcube** creates another version of *cubeconf* in */tmp* and uses that version. As it does this, **bootcube** displays the following message for each empty slot it encounters:

```
Node xx: Does not respond
```

where *xx* is the slot number.

Consequently, if you are unsure of your slot configuration, you can save the current version of *cubeconf*, make a new version minus the slot assignments, and then issue **bootcube**. The **bootcube** command will create a new cube configuration file in */tmp/cubeconf*. Note that changing the slot

value is not enough. If you want a new *cubeconf*, you should remove all the slot values. For example, if you mislabel a slot as **EMPTY**, **bootcube** will not correct the entry. Rather, it will treat the slot as really empty.

If the diagnostic program detects that your hardware configuration does not match what is documented in the file, it reports an error.

THE DEVICE CONFIGURATION FILE

Each I/O node with a SCSI interface can have up to seven peripheral devices associated with it (numbered 0 through 6). The peripherals are identified in the device configuration file. This is called **devconf** and is located in the directory */usr/ipsc/conf* on the SRM. The **bootcube** command does not use this file. The diagnostic program uses the device configuration file.

Unless you modify your hardware configuration, you should not have to modify the device configuration file. It is set at the factory with the values appropriate for your configuration. As the system administrator, however, you should understand the format of the file.

Figure 5-2 shows an example of *devconf*. It describes a system in which the 16 slots in the CIO standard cabinet's cardcage are slots 32 through 47 in the iPSC system. It contains I/O nodes in the even-numbered slots. The system has 16 disk drives. Each I/O node has two disk drives with unit numbers 0 and 1.

```
/*                              Device
 *                           Configuration File
 *
 * Copyright (c) 1989 Intel Corporation
 *
 *
 * Device Configuration:
 *
 * type model unit: slot, slot, ... ;
 *
 *  where:
 *   type  is device type (DISK or TAPE)
 *   model is device model string (MAXTOR.XT-4380S or MAXTOR.XT-8760S)
 *   unit  is device unit number (0, 1, 2, 3, 4, 5, or 6)
 *   slot  is physical node slot id to which the device is attached
 *
 */

DISK    MAXTOR.XT-8760S
                        0:      32,34,36,38,40,42,44,46;
                        1:      32,34,36,38,40,42,44,46;
```

Figure 5-2. Device Configuration File for a d5 iPSC®/2 System

The diagnostic program CDP requires the device configuration file. It needs the information in this file to test the disk drives. CDP will not test the disk drives if it cannot read the device configuration file.

If the diagnostic program detects that your hardware configuration does not match what is documented in the file, it reports an error.

EXECUTING bootcube ON THE SRM

1. Make sure that no cubes are allocated. (The UNIX **who** command tells who is currently logged in, and the iPSC **cubeinfo** command tells what cubes are currently allocated.)

2. Become the superuser.

3. Issue the **bootcube** command.

Most of the time you will run **bootcube** without any options. **bootcube** runs as a series of steps. You can display the steps without executing them with the **-L** switch. For example:

```
# bootcube -L
Order of operation:
1.  Reset driver
2.  Scan cardcages
3.  Reset nodes
4.  Scan nodes
5.  Download 386 boot loader: /usr/ipsc/lib/bootld
6.  Download 860 boot loader: /usr/i860/ipsc/lib/bootld
7.  Start boot loader
8.  Load SRM Direct Connect Module
9.  Query nodes
10.  Load 386 node Direct Connect Modules
11.  Load 860 node Direct Connect Modules
12.  Initialize node Direct Connect Modules
13.  Test nodes
14.  Reset Direct Connect Modules
15.  Check configuration file
16.  Execute startup run file: /usr/ipsc/lib/rc1 -Q /usr/ipsc/conf/cubeconf
17.  Send load command to nodes
18.  Get boot partition
19.  Download /usr/ipsc/lib/nx.b
20.  Download /usr/i860/ipsc/lib/nx.b
21.  Release boot partition
22.  Execute startup run file: /usr/ipsc/lib/rc2 -Q /usr/ipsc/conf/cubeconf
23.  Execute startup run file: /usr/ipsc/lib/rc3 -Q /usr/ipsc/conf/cubeconf
24.  Start TCP services
#
```

During a **bootcube**, all steps that are performed are displayed, while those not performed are not listed.

The **bootcube** command runs predefined shell scripts:

rc1	Starts up the iPSC daemons and stores their process ID numbers in the file */tmp/procno*.
rc2	Performs further initialization and allocates a zero-node cube called *iocube*. The Concurrent I/O system needs *iocube*.
rc3	Initializes the Concurrent I/O system.

NOTE

Because **bootcube** options let you specify alternate shell files, it is important for the security of the system never to turn on the setuid root bit in the permissions of **bootcube**.

You can choose to run specific **bootcube** steps. For example:

```
# bootcube -D2
```

runs only the first two steps of **bootcube**. This stops all file server and communications activity on the host and is something you should do before running the CDP diagnostic program.

If there are allocated cubes when you issue **bootcube**, it does not run and you get the following message:

```
You must release all cubes before running bootcube
```

The **-f** switch forces a boot even if cubes are allocated.

To boot your cube without starting CFS, use the command **bootcube -o**.

EXECUTING bootcube ON THE REMOTE HOST

There is a separate version of **bootcube** specifically for starting the appropriate daemons on the remote host machine.

The only flags that can be used with this version are **-L**, **-D**, **-f**, **-R**, and **-S**. These switches have the same meaning on the remote host as they do on the SRM. However, the shutdown file *rc0* and the startup file *rc1* kill and start up processes on the remote host, not the SRM.

SYSTEM ADMINISTRATION TASKS 6

INTRODUCTION

This chapter describes some typical system administrative tasks. Even in systems where several users know the root password, only the system administrator should perform administrative tasks.

Typical system administrative tasks include:

- Installing software and booting the cube (described in Chapters 3 and 5, respectively)
- Recording the hardware configuration
- Establishing network addresses for TCP/IP nodes
- Deallocating cubes that belong to other users
- Removing the root password
- Performing routine preventive maintenance
- Swapping node boards and returning defective boards for repair
- Starting and stopping process logging
- Establishing links to user accounting software
- Making a Concurrent File System
- Backing up and restoring the Concurrent File System

RECORDING THE HARDWARE CONFIGURATION

As a system administrator, you should record the hardware configuration of your iPSC system (SRM, cabling, and node configurations).

System Resource Manager Configuration

The Intel386™ mother board in the system resource manager provides eight slots as follows:

Slot 1	16-bit	Empty
Slot 2	16-bit	Empty
Slot 3	16-bit	TCP/IP network board
Slot 4	8-bit	Vega Video7 monitor board
Slot 5	8-bit	Tape controller board for the cartridge tape drive
Slot 6	32-bit	8M-byte memory board
Slot 7	32-bit	iPSC interface board for communicating with the cube
Slot 8	16-bit	Disk controller board for the Winchester disk and the high-density floppy disk

Note that the two empty slots are 16-bit slots. 8-bit boards can be inserted into these slots as long as the board's software can handle 16-bit interrupts.

Cabling Configuration

When the system was originally installed, your SSD Service Representative left a copy of the iPSC cabling diagram. Keep this in a safe place. Intel Supercomputer Systems Division also retains a copy and will provide additional copies on request. The diagram left by your SSD Service Representative illustrates the cable connections for the iPSC system.

Node Configuration

Keep a record of your iPSC system node configuration. This means identifying the options for each node. Figures 6-1 and 6-2 illustrate the possibilities for each node. One way to keep this record is to make copies of these figures and check off the appropriate options.

The hardware configuration is recorded in the file *cubeconf* in the */usr/ipsc/conf* directory.

Also, be sure to record any special characteristics of your SRM. The only hardware option currently available for the SRM is a larger disk drive. The standard SRM has a 380M-byte disk drive.

If you have a Concurrent I/O system, record the location and size of your disk drives as well as the options for the I/O nodes.

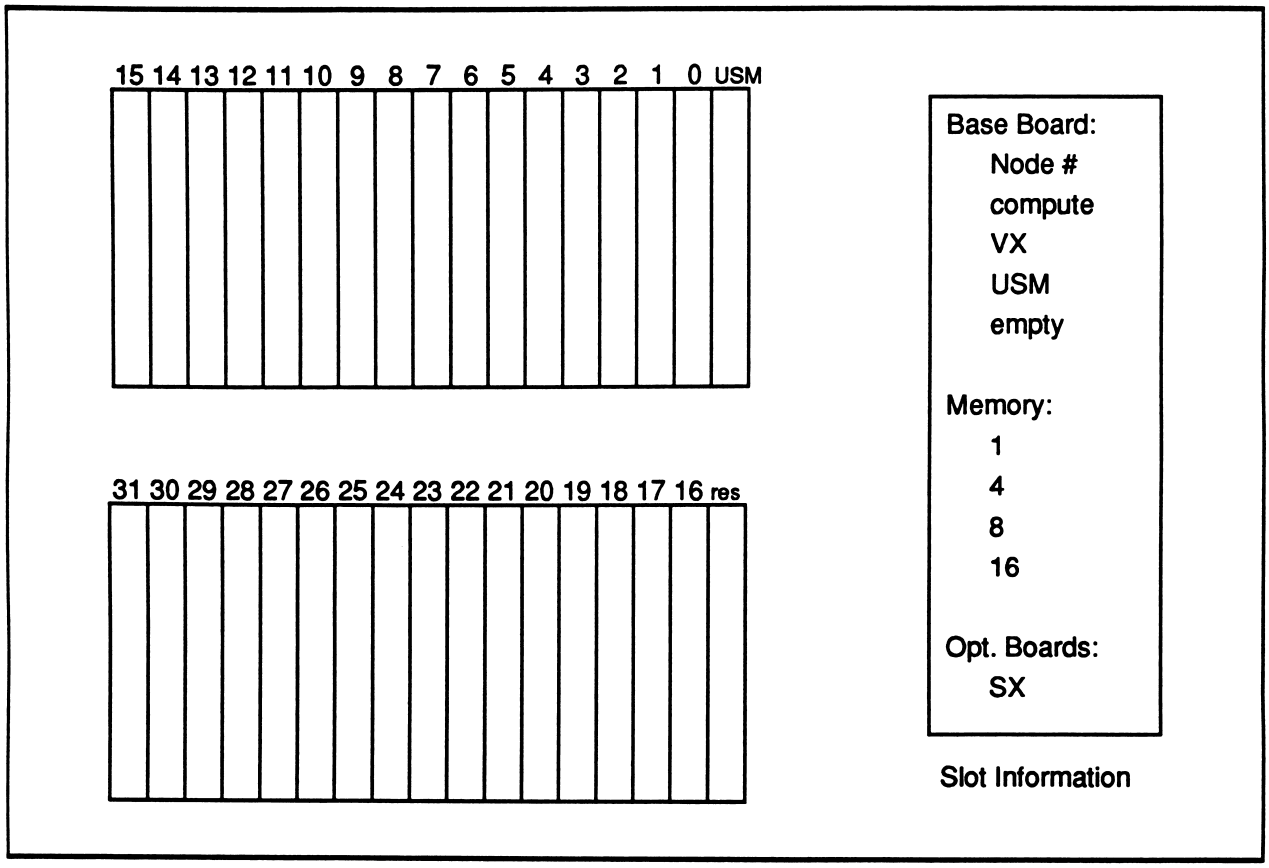

Figure 6-1. The Slot Arrangement for a Compact Cabinet

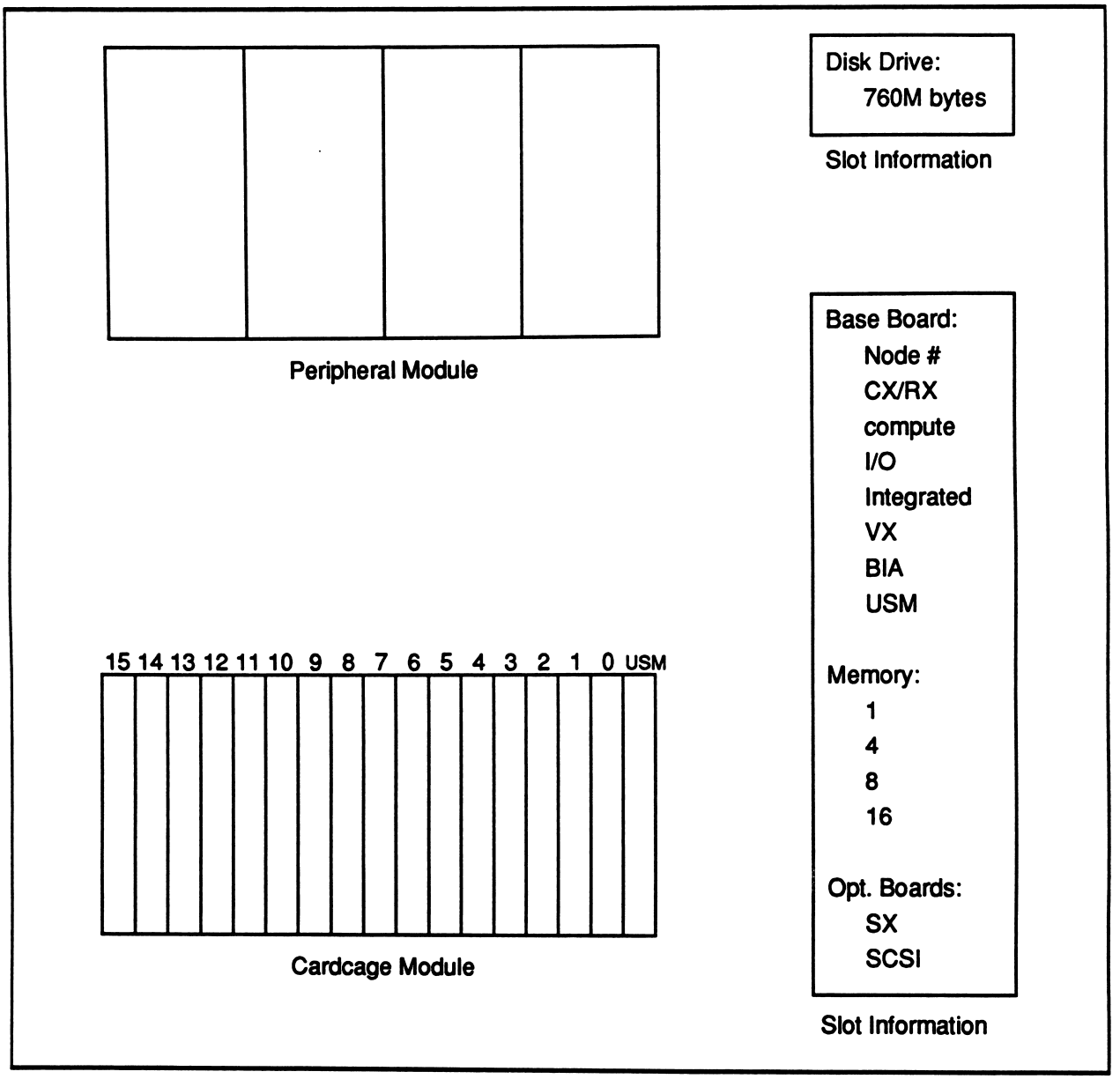

Figure 6-2. The Slot Arrangement for a Standard Cabinet

ESTABLISHING NETWORK ADDRESSES FOR TCP/IP NODES

If your system has the CIO Ethernet option, you need to assign names and IP addresses to the TCP/IP nodes in the iPSC system and make sure that they are made known to your network. The program **ipconf** is supplied to help you add or change the names and or IP addresses of the nodes in the */etc/hosts* file on the SRM. This program is interactive, and should be self-explanatory. When you execute **ipconf**, it first looks in the *cubeconf* file to see if any of the I/O nodes are designated as Ethernet nodes.

If you just installed the TCP/IP node hardware in your system, make a copy of the file as follows:

```
# cp  /usr/ipsc/conf/cubeconf  /usr/ipsc/conf/cubeconf.sav
```

Next edit the */usr/ipsc/conf/cubeconf* file, deleting everything after the "baud 38400" line. Now you can execute a **bootcube** to build a new *cubeconf* file that includes the new node hardware. The **bootcube** fails in step 25, but the *cubeconf* file you need is built in the */tmp* directory. You can then move the */tmp/cubeconf* into the */usr/ipsc/conf/cubeconf* file that **ipconf** uses. The following example shows you how to create the *cubeconf* file for **ipconf**.

```
# bootcube -f
 1.  Reset driver
 2.  Scan cardcages
 3.  Reset nodes
 4.  Scan nodes
 5.  Download 386 boot loader: /usr/ipsc/lib/bootld
 6.  Download 860 boot loader: /usr/i860/ipsc/lib/bootld
 7.  Start boot loader
 8.  Load SRM Direct Connect Module
Load SRM DCM with /usr/ipsc/lib/hbits.g
 9.  Query nodes
10.  Load 386 node Direct Connect Modules
Load 1 node DCM's with /usr/ipsc/lib/386nbits.d          Varies with hardware
Load 1 node DCM's with /usr/ipsc/lib/386nbits.g          Varies with hardware
11.  Load 860 node Direct Connect Modules
Load 2 node DCM's with /usr/ipsc/lib/860nbits.d          Varies with hardware
Load 4 node DCM's with /usr/ipsc/lib/860nbits.f          Varies with hardware
12.  Initialize node Direct Connect Modules
13.  Test nodes
14.  Reset Direct Connect Modules
15.  Check configuration file
16.  Execute startup run file: /usr/ipsc/lib/rc1  -Q /tmp/cubeconf
17.  Send load command to nodes
18.  Get boot partition
19.  Download /usr/ipsc/lib/nx.b
20.  Download /usr/i860/ipsc/lib/nx.b
21.  Release boot partition
22.  Execute startup run file: /usr/ipsc/lib/rc2   -Q /tmp/cubeconf
```

```
No disks available
Non-CFS initialization complete
24.  Start TCP services
(host) startsp: adminproc: Can't assign requested address
(host) Starting Socket Process: Can't assign requested address

Warning: Using /tmp/cubeconf as configuration file.
Please check your configuration file.

# cp /tmp/cubeconf /usr/ipsc/conf/cubeconf
#
```

The following example shows the **ipconf** options available to you. Further interactive questions follow a response in which you indicate a desire to change the existing configuration.

```
# ipconf
Enter slot number of the Service node for board #1 [0]: 34
Enter official network name of board #1(in slot 34): bear
WARNING: node does not have corresponding ip address in /etc/hosts
Enter internet address of board #1(slot 34): 999.99.99.9

Board #1:
        Internet address: 999.99.99.9
        Official network name: bear
        Slot number of the Service node: 34
        Slot number of the TCP/IP board 35
        ISC Alias: pacificr_IPSC_34

Would you like to:
   Quit without installing (q) or
   Keep (k), Delete (d), this entry or
   Change:
     All (a), Name (n), Inet address (i), Slot number (s) of this entry: [k] k


Would you like to:
   Add another board (a)
   Modify a board configuration (m)
   List the board configuration (l)
   Quit without installing (q)
   Install the new configuration and quit (i)? [a] i
The original hosts file /etc/hosts is saved as /etc/hosts.sav
/etc/hosts updated for bear
Creating enpar file </usr/ipsc/lib/enpar35.desc> for board #1 bear
Compiling enpar file to </usr/ipsc/lib/enpar35.S> for board #1 bear

#
```

Any changes to names or addresses that you make by using **ipconf** are automatically added to the */etc/hosts* file on the SRM.

CAUTION

Ipconf writes the updated */etc/hosts* file using the current *umask*. If you have changed the default *umask*, verify that the */etc/hosts* file is both *world* and *group* readable (**chmod 644**) before you reboot the iPSC system. If */etc/hosts* is not world readable, **bootcube** fails in step 24.

After executing **ipconf**, reboot the iPSC system, and then use the UNIX **ping** command on the SRM to ensure that the board is responding, as in this example:

```
# ping bear
PING bear: 56 data bytes
64 bytes from 999.99.99.9: icmp_seq=0. time=0. ms
64 bytes from 999.99.99.9: icmp_seq=1. time=0. ms
64 bytes from 999.99.99.9: icmp_seq=2. time=0. ms
64 bytes from 999.99.99.9: icmp_seq=3. time=0. ms
64 bytes from 999.99.99.9: icmp_seq=4. time=0. ms
64 bytes from 999.99.99.9: icmp_seq=5. time=0. ms
64 bytes from 999.99.99.9: icmp_seq=6. time=0. ms
<Ctrl-c>
----bear PING Statistics----
7 packets transmitted, 7 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 0/0/0
```

Follow your standard procedures for adding nodes to your network to ensure that the workstations can communicate with the new nodes.

DEALLOCATING A USER'S CUBE

The **relcube** command lets you deallocate cubes that you own. **relcube** without arguments deallocates the currently attached cube. The **-c** switch lets you specify a particular cube, and the **-a** switch deallocates all the cubes that you own.

When run by root, **relcube** has two additional switches: The **-F** switch releases all allocated cubes, and the **-f** switch releases a specified cube. The **-f** switch takes the cube ID as an argument.

The **cubeinfo** command with the **-s** switch lists the cube IDs of all the allocated cubes.

For example, to deallocate a cube with ID 2, enter the following command:

```
% relcube -f 2
```

There is no cube with ID 0, but if you supply a 0, you deallocate all cubes. The switch **-F** also performs this function. For example:

```
% relcube -f 0
```

is the same as the following:

```
% relcube -F
```

The cube *iocube* (which has 0 nodes) always appears on the **cubeinfo** display. Even root cannot deallocate this cube.

REMOVING ROOT'S PASSWORD

As the system administrator, you can remove the root password. Use this procedure if you forget the root password or if it is changed without your knowledge:

1. If you are able to log in:

   A. Log in as any user.

   B. Enter the following:

      ```
      # sync; sync
      ```

   C. Make sure that nobody is logged in. (The **who** command tells who is currently logged in.)

   D. Log out.

2. If you are unable to log in, let the system sit idle for at least 30 seconds in case the automatic **sync** is working.

3. Turn off the machine.

4. Wait 10 seconds.

5. Turn the machine on, and quickly insert the diskette labeled "UNIX Boot Floppy" into the diskette drive.

6. When the following message appears:

   ```
   Strike enter to install the UNIX System on your hard disk
   ```

   Press **<Ctrl-\>** to abort the installation procedure and get the command prompt.

7. Enter the following to clean the file system:

   ```
   # fsck -y /dev/dsk/0s1
   ```

   If the system reports that modifications were made, run **fsck** again:

   ```
   # fsck -y /dev/dsk/0s1
   ```

   Repeat until no changes are reported.

8. Enter the following:

   ```
   # mount /dev/dsk/0s1 /mnt
   ```

9. Save a copy of the existing password and shadow files. This is a precaution against an editing mistake. Issue the following commands:

   ```
   # cp /mnt/etc/passwd /mnt/etc/passwd.sav
   # cp /mnt/etc/shadow /mnt/etc/shadow.sav
   ```

10. Remove the root password with the following sequence of commands (commands are in bold italic type; comments in italics to the right of the commands describe the effect of each command):

```
# ed /mnt/etc/shadow          Edits the shadow file.
                              The number of bytes in the file
                              appears on the screen
1p                            Prints the first line of the file, which should
                              be the root entry. The command
                              is the number 1 followed by a p.
s/root:[^:]*/root:            Removes the root password.
w                             Writes the file and shows the
                              number of lines.
q                             Quits the editor.
ed /mnt/etc/passwd            Edits the passwd file.
                              The number of bytes in the file
                              appears on the screen.
1p                            Prints the first line of the file,
                              which should be the root entry.
s/:x:/::/                     Removes the root password.
w                             Writes the file and shows the number of lines.
q                             Quits the editor.
```

11. Enter the following:

```
# sync; sync
```

12. Enter the following:

```
# umount /mnt
```

13. Enter the following:

```
# sync; sync; uadmin 2 0
```

    to shut down the system.

14. When the light on the diskette drive turns off, remove the boot diskette from the drive.

15. Press **<Ctrl-Alt-Del>** to reboot the UNIX operating system.

16. When the system boots, login as root and set a new password for root.

PERFORMING PREVENTIVE MAINTENANCE

Routine preventive maintenance consists of:

- Running **fsck** when necessary
- Cleaning the air filter in a compact cabinet
- Cleaning tape drive heads

Running fsck

The **fsck** program is a file system check-and-repair program that makes several consistency checks on a specified file system. The comparable utility for the Concurrent File System runs each time you boot the cube. (If you think the CFS files have been corrupted, run **bootcube**.) Refer to the *UNIX System V/386 Release 3.2 System Administrator's Guide* and the *UNIX System V/386 Release 3.2 System Administrator's Reference Manual* for information about **fsck**.

Inspecting and Cleaning Air Filters in Compact Cabinets

If you have compact cabinets, inspect their air filters every three months and clean them at least every six months. Standard cabinets do not have air filters.

If your location is exceptionally dusty, inspect and clean the filter every month. Cleaning the filter will help the unit run cooler, thus reducing the chances for hardware malfunction.

Follow this procedure for removing and cleaning the filter for each computational unit:

1. Notify all users that you will be powering the system down. Then, power down the system as described in Chapter 2.
2. Open the front door by moving the lever located at the bottom right counterclockwise 90 degrees.
3. Turn off all the units. Turn the key switch on each unit clockwise to the horizontal (OFF) position.
4. Turn off all circuit breakers. Move the white circuit breakers (located at the bottom back of all units) to the right (OFF) position.
5. Unplug the units from the AC power receptacle.
6. Locate the grill (approximate size is 3-1/2 by 14 inches) just above the lower edge of the door.
7. Using a #1 small-tip Phillips screwdriver, remove the five screws holding the grill in place.

8. Slide out the air filter (identified by the half-inch silver band that encircles it).

9. Wash the filter in warm, soapy water.

10. Shake off any excess water and allow it to air dry for several hours.

WARNING

Be sure the air filter is dry before replacing it inside the unit. Because the filter is directly over the AC power distribution in the unit's base, water coming into contact with this area can cause a short in the system.

11. Replace a cleaned and thoroughly dried air filter into its position in the base of the unit.

12. Replace the grill.

13. Use a #1 small-tip Phillips screwdriver to screw in the five screws that hold the grill in place.

14. Plug the AC power cord into the wall receptacle.

15. Apply power.

16. To close the front door, move its lever clockwise 90 degrees toward the unit.

Cleaning Tape Drive Heads

The 8mm tape drive heads should be cleaned after 30G bytes of data transfer or once a month under normal conditions. A cleaning kit, which includes instructions and is good for three cleanings, is provided with systems that include 8mm tape drives. This kit removes particle buildup on the read/write heads and the tape path, which can cause media errors. Additional cleaning kits can be ordered from EXABYTE (part number 180123) or Intel (part number 314313-001).

For information on preventive maintenance or service on nine-track tape drives, contact SSD Customer Support.

SWAPPING NODE BOARDS

NOTE

Contact SSD Customer Support before swapping a node board. You should only swap node boards under the direction of SSD Customer Support. Board swapping should *never* occur without the knowledge and consent of the system administrator.

You might need to swap a node board in the following situations:

- If you feel that node 0 may be bad, you might try swapping it with the highest numbered node board. For an iPSC system to come up, node 0 must be working. If replacing node 0 with the highest numbered node board causes the iPSC system to come up, its resulting dimension will be smaller. An iPSC system must have a number of nodes equal to a power of two. Additional nodes that don't increase the total number to the next power of two are ignored.
- If the diagnostic program CDP identifies a node as bad, move the bad node to a different location and determine if the error follows the node.
- If you are certain that a node board is bad and you have a spare, you may want to install the spare and send the defective node board to SSD Customer Support for repair.

To swap a node board, perform the following steps:

CAUTION

Guard against static when replacing or swapping nodes. Use a proper grounding strap.

1. To open the front door of a unit, move the lever (located at the bottom) counterclockwise 90 degrees.
2. Turn off power to the cabinet. On a compact cabinet, turn the key switches on all compact cabinets clockwise to the horizontal (OFF) position. Be sure that the green (DC) lights turn off. It is OK if the amber light (indicating AC power) stays on. On a standard cabinet, turn off the power switch in the back.

3. With a small screwdriver loosen the screws at both ends of the malfunctioning board's front panel (see Figure 6-3). Do not remove the screws from the front panel.

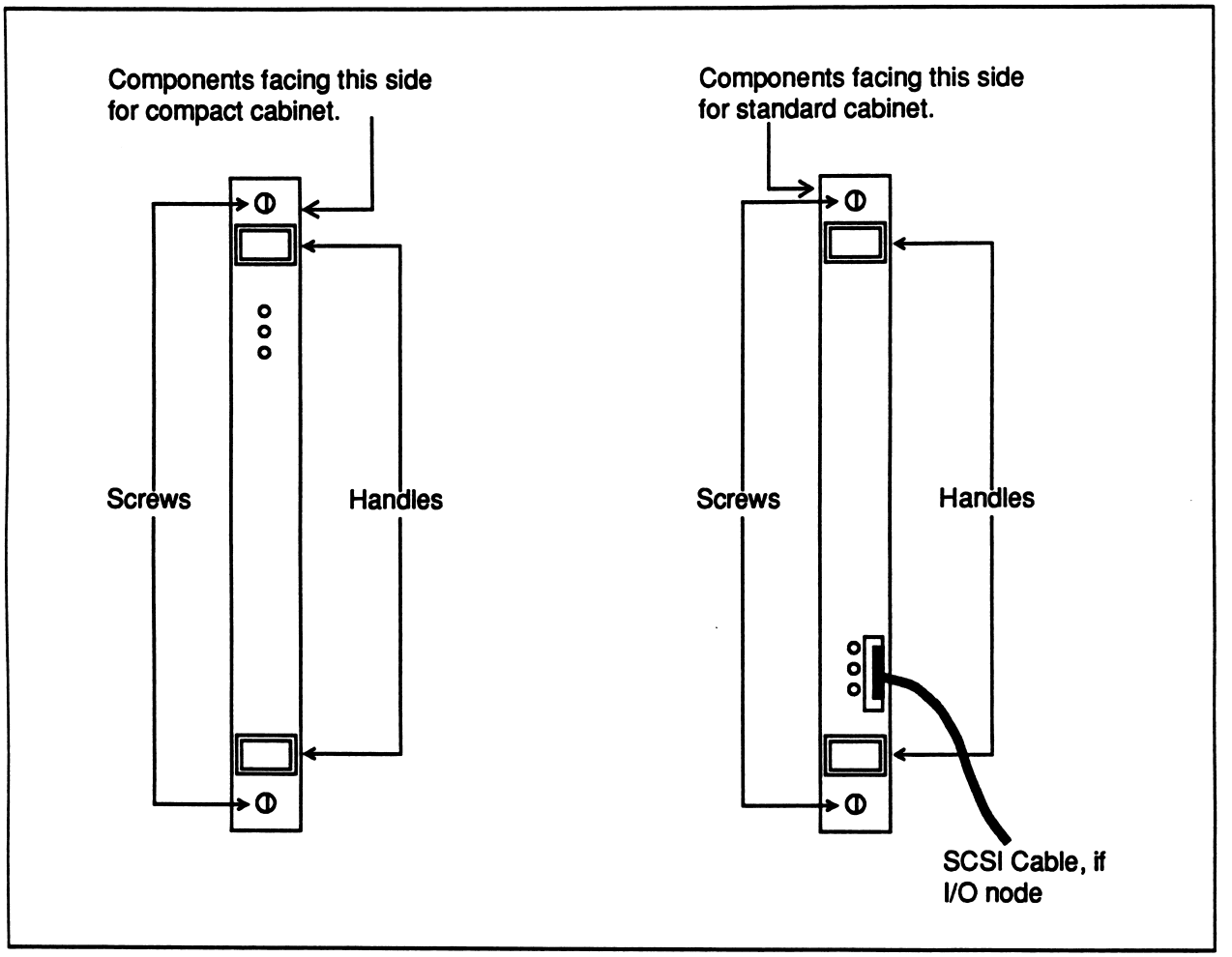

Figure 6-3. Node Board's Front Panel

4. Grasp both handles and push them apart. The board pops free of the backplane.

5. Set the defective board aside on an antistatic surface to be packaged and returned to SSD Customer Support for repair.

6. Insert a spare board into the tracks of the empty slot. Push the ejector handles to the center. Firmly seat the board with the palm of your hand.

7. With a small screwdriver tighten the screws in place. **DO NOT OVERTIGHTEN.**

8. If you cannot tighten the screws, the board is not properly seated. Continue pushing the board back into slot. Use a firm up and down rocking motion.

9. Apply power.

10. Refer to the next section for information about returning a malfunctioning board to SSD Customer Support for repair.

RETURNING BOARDS FOR REPAIR

When calling SSD Customer Service, have the following information available:

- Serial number of the computational unit (located in back at the base of the unit).
- Serial number of the board (usually on the back of the board).
- Your shipping and billing addresses.
- Purchase order number (for billing purposes) if your warranty has expired.

Always contact SSD Customer Support *before* returning a board for replacement. You will be given a Return Materials Authorization (RMA) number, shipping instructions, and other important information. Contact SSD Customer Support at:

1-800-421-2823 (Customer Support Hotline)
(44) 793 641 469 (in England)
Your Local Intel Sales Office (in Europe)
support@ssd.intel.com (Internet address)

Before shipping a board, make sure it is adequately protected as follows:

- The board should be placed in an antistatic bag within a padded shipping bag.
- Allow sufficient room in the shipping carton for protective padding such as flow pack, foam, etc.
- Write, "Attn: Return Material Authorization (RMA) *number*" on the *outside* of the shipping carton. *number* is the RMA given to you by SSD Customer Support.
- Label the shipping carton FRAGILE.

Address and ship the carton to the address given to you by your SSD Customer Service Representative.

LOGGING INFORMATION

The system administrator controls the process-logging function. This function lets you keep track of the execution of **bootcube**, allocation and deallocation of cubes, and the loading and completion of node processes. The logging function gives you another tool to monitor your system:

- As a system administrator for an iPSC system with a number of users, you may find it necessary to monitor who is using the nodes and for how long.
- Users may request that you set up process-logging to aid them in debugging their applications.
- If your iPSC system is having a problem, SSD Customer Support may request that you create a log file.

You turn on logging with the **plogon** command, and turn it off with the **plogoff** command:

- If you specify the **-c** switch, **plogon** logs only the allocation and release of cubes, not the loading and completion of node processes.
- If you specify the **-a** switch, **plogon** appends the logging information to an existing logfile. Otherwise, **plogon** overwrites an existing logfile.
- If you specify the **-i** switch, **plogon** reports the current output log file name without changing the logging state.
- If you specify a filename, **plogon** uses that file as the log file instead of the default file */usr/ipsc/log/process.log*.

For example:

- The following command turns on logging of cube allocation and deallocation, and writes the output to the file */usr/you/logfile*:

```
# plogon -c /usr/you/logfile
```

- The following command turns off logging:

```
# plogoff
```

There are also system call versions of **plogon()** and **plogoff()** available to C and Fortran programs. C has access to underscore versions of each. When used with Fortran, invoke them as subroutines. Only a host program with root privileges can run these system calls:

- **plogon()** takes four arguments:
  - The first argument is a short integer. If it is not 0, the logging information is appended to the specified log file (if it exists). Setting the integer to 1 in the system call is the same as specifying the **-a** switch in the command.
  - The second argument is a short integer. If it is not 0, the logging function logs only cube allocations and deallocations. Setting the integer to 1 in the system call is the same as specifying the **-c** switch in the command.
  - The third argument is a short integer. If it is not 0, the logging function prints a string that identifies the current log file to *stdout*. The **plogon()** system call returns a string setting the integer to 1. This is the same as specifying the **-i** switch in the **plogon** command.
  - The fourth argument is a string that identifies the log file. A null string chooses the default log file, */usr/ipsc/log/process.log*.
- **plogoff()** has no arguments.

Chapter 8 contains formal descriptions (in reference manual format) of the **plogon** and **plogoff** commands and system calls.

NOTE

The log file can grow very large, using a lot of SRM disk space. Therefore, be judicious in using **plogon** and **plogoff**.

ACCOUNTING SOFTWARE SUPPORT

The base iPSC system software and the NX node software have features that allow you to link to your own accounting software and keep track of certain key events in the iPSC system. Your accounting software may include procedures to keep track of the iPSC system usage time for individuals or groups, or provide a method of access control to the system. The accounting hooks in the iPSC system provide the tools you need for accounting software support.

The communications server (*commser*) process on the SRM or on a remote host calls a procedure named **account_hook()** at start-up time, with each **getcube**, with each **relcube**, and at shutdown. The source code for **account_hook()** has been provided to allow you to make any changes that are necessary to link your own accounting software to the iPSC software.

One possible additional use of **account_hook()** is to verify the privilege level of the user requesting node allocation. Because **getcube()** must get a 0 return value from **account_hook()** in order to be successful, the **account_hook()** source could be modified to check a list of authorized users and return a non-zero value if the requestor is not on the list. If you make no change to the **account_hook()** source code, no accounting will be done.

The source code for **account_hook()** is provided in */usr/ipsc/lib/commser.bin/hooks.c*. This is the version of the **account_hook()** procedure that is compiled and linked with NX software. In order to use the values returned by **account_hook()**, your accounting software must be able to accept two parameters; an *event* and a *cube-information* parameter. Perform the following steps in order to modify the **account_hook()** source code:

1. Edit the routine **account_hook()** to do what you want the iPSC accounting hook software to do.

2. Ensure that there are no users on the iPSC system.

3. Login as "root" and shut down the cube (kill the *commser* process). The following example accomplishes this step:

```
# bootcube -D1
```

4. Save the old *commser* executable code under a different name. For example:

```
# cp /usr/ipsc/lib/commser /usr/ipsc/lib/commser.org
```

NOTE

The default version of **account_hook()** just prints a log of user activity in the default log file. The default *commser* log file is at */usr/ipsc/lib/commser.log*. The **account_hook()** source code is at */usr/ipsc/lib/commser.bin/hooks.c*.

5. Enter the directory */usr/ipsc/lib/commser.bin and issue the command:*

```
# make
```

This makes a new *commser* executable.

6. Now issue the following command:

```
# make install
```

NOTE

The **make install** command will try to remove *commser*. Be sure you have saved the original version of *commser* before executing this command.

7. Reboot the iPSC system by issuing the following command:

```
# bootcube
```

Your modified version of **account_hook()** should now be running as part of the *commser* process.

MAKING A CONCURRENT FILE SYSTEM™

To make a Concurrent File System (CFS), use the **mkcfs** command. Chapter 8 contains a formal description (in reference manual format) of the **mkcfs** command.

When you make a Concurrent File System, you destroy all existing concurrent files and return the CFS to its initial state. You also have the option of renaming the CFS.

When you have a brand new system, you must first make a Concurrent File System before doing a **bootcube**. If during the course of operation your CFS becomes corrupted, you must remake it. Before remaking a corrupted system, you may try to back up your files and then later restore as many files as possible.

Use the following procedure to make a Concurrent File System:

1. Login as *root*.

2. Issue the **bootcube** command, specifying that only the first 23 steps be run. This performs a boot, but does not fully initialize the Concurrent File System:

```
# bootcube -D22
```

3. Issue the **mkcfs** command. This command gives you the option of changing the file system's name:

```
# mkcfs
2 drives ready on 1 node.
File system "original" already exists, continue? y
Do you wish to change the file system name? y
Enter volume name: new                          This is an example name
Are you sure you want to destroy all files? y
```

4. Issue **bootcube** again, and when it finishes, exit superuser:

```
# bootcube
    .
    .
    .
# exit                                          Exit from superuser
```

When **bootcube** is run, the CFS directories and files are checked for consistency before it is started up. The CFS file fixer tries to fix any problems that are detected.

CONNECTING THE TAPE DRIVES INTO CFS

If you have tape drives (either 8mm or nine-track tape drives) for your Concurrent File System™, you need to connect it to the file system after making a CFS with **mkcfs**. Use the following procedure.

1. Login as *root*.

2. Get a small (no more than four nodes) cube and start up the node shell (**nsh**).

3. Use **showvol** to find out the volume of the tape drive.

4. Use **mkdev** to connect each tape drive to a unique name in CFS.

The following example connects an EXABYTE 8mm tape drive at volume 4 to the file name */cfs/tape*.

```
% su
# getcube -t4
getcube successful: cube type 4m8rxn0 allocated
# nsh
# showvol
   0    642    0 FD    670,171,136      662,794,240  MAXTOR  XT-8760S      B5A
   1    654    1 FD    670,171,136      663,031,808  MAXTOR  XT-8760S      B5A
   2    654    2 FD    670,171,136      660,959,232  MAXTOR  XT-8760S      B5A
   3    654    3 FD    670,171,136      662,724,608  MAXTOR  XT-8760S      B5A
   4    642    3 RT          1,024      spd0 den0 B  EXABYTE EXB-8200      425A

 # mkdev 4 /cfs/tape
Installing [4, 0] in CFS at tape
# ls -l /cfs
total 0
brw-rw-rw-   0 root     root       4,  0  Dec 31  1969 tape
# exit
# relcube
```

Now the tape can be accessed as though it were a UNIX device. The calls **open()**, **close()**, **read()**, **write()**, and **ioctl()** can talk to the drive, as well as the node commands **tar**, **mt**, and **dd**.

BACKING UP THE CONCURRENT FILE SYSTEM™

To back up the Concurrent File System, execute the **cbackup** command under the node shell. Chapter 8 contains a formal description (in reference manual format) of the **cbackup** command.

You can back up all your volumes or select particular volumes. Use the **showvol** command to get the volume numbers. To backup specific files, use **tar** to put them on tape, or **cp** to copy the files into your host file system. When you archive specific files to tape, the **star** command is usually faster than **tar**. It speeds tape access by using **stream** to gather the **tar** output into large buffers before writing to the tape. When you extract the files, **star** reads large buffers before unarchiving, again to speed up tape access.

If your backup media is not large enough to hold the entire backup, be sure to use the **-s** switch. This switch specifies the size of the backup media. This example first gets you into the node shell (a requirement for **cbackup**) and then uses the -s switch.

```
# nsh
# cbackup -s 45M  /dev/tape
```

The **cbackup** command knows that it must request another tape after writing 45M bytes of data. It is an error to leave out the **-s** switch if your media cannot hold your entire backup.

Finally, the -a switch specifies that all the blocks on a disk, whether or not they actually contain data, will be backed up.

When you run **cbackup** and **crestore**, the CFS must be fully started, but other users must not be accessing the CFS. If before issuing **cbackup** and **crestore**, you release all cubes and then get a cube of the maximum size, you will block out most users. This does not prevent a user from allocating a cube with 0 nodes, but such a user is unlikely to disturb the file system.

The following example shows how to back up a small concurrent file system (it has only two files):

1. Use the node shell to list the files in the CFS:

```
% getcube                                    You have to get a cube to run the node shell
getcube successful: cube type 32m4n0 allocated
% nsh
% cd /cfs
% ls -l
total 2
-rw-r--r--   1 you      other        144 Apr 28 11:52 file1
-rw-r--r--   1 you      other        288 Apr 28 11:52 file2
% exit                                                          Exit from node shell
% relcube
relcube released 1 cube
```

2. Become the superuser, do a **getcube** to lock out other users, do a **showvol** to see the contents of the CFS, and enter the node shell:

```
% su
Password:                                                       Enter root password
# getcube                                                     Lock out other cube users
getcube successful: cube type 32m4n0 allocated
# showvol

Volume Node Drive    Size       Free       Vendor  Model Revision

   0   642    0   649,359,360  648,544,256  MAXTOR  XT-8760S  B2
   1   642    1   670,171,136  669,335,552  MAXTOR  XT-8760S  B3C
# nsh
```

3. Insert a tape into the SRM's tape drive.

4. Do the backup. Note that because the file system is small and fits on one tape, there is no need to use the -s switch:

```
# cbackup /dev/tape
backup "new" on Fri Apr 28 11:58:33 1989
# exit                                                    Exit node shell
# relcube
relcube released 1 cube
# exit                                                    Exit superuser
%
```

RESTORING THE CONCURRENT FILE SYSTEM™

To restore a Concurrent File System, run the **crestore** command under the node shell. Chapter 8 contains a formal description (in reference manual format) of the **crestore** command.

You can restore all your volumes or select particular volumes. Use the **showvol** command to get the volume numbers if you need them. You can only restore from tapes that have been written with the **cbackup** command.

The following example restores the file system that was saved in the last section. First, insert the backup tape into the SRM's tape drive. To better illustrate the file system restoration, the two files belonging to the existing file system are deleted.

```
% su                                                      Enter the root
Password:                                                   password
# getcube
getcube successful: cube type 32m4n0 allocated
# nsh
# cd /cfs
# rm file?
# ls -l
total 0
# crestore /dev/tape
Restore "new" created Fri Apr 28 11:58:33 1989
File system "new" exists, overwrite? y
# cd /cfs
# ls -l
total 2
-rw-r--r--   1 you     other        144 Apr 28 11:52 file1
-rw-r--r--   1 you     other        288 Apr 28 11:52 file2
# exit                                                  Exit from the node shell
# relcube
relcube released 1 cube
# exit                                                    Exit superuser
%
```

RUNNING DIAGNOSTICS 7

INTRODUCTION

This chapter describes the diagnostics available for the iPSC system. The diagnostic programs consist of the Power-On Self Test (POST), the Node Confidence Test (NCT), and the Cube Diagnostic Program (CDP).

The first two programs (POST and NCT) are automatically run when you power up the SRM and the nodes. You must specifically invoke the last program (CDP) to run it. Normally, an SSD Customer Support Representative runs the cube diagnostic program. However, when characterizing a problem in preparation for a site visit, an SSD Customer Support Representative may request that the system administrator run diagnostics.

This chapter shows how to run the diagnostic tests and gives a recommended strategy for identifying problems.

DIAGNOSTIC DESCRIPTION

This section briefly describes all the major iPSC diagnostics.

POST	The SRM runs the Power-On Self Test automatically when powered up. POST checks the Intel386™ microprocessor, the keyboard, the display, system memory and most peripheral devices connected to the SRM. Refer to the *SYP301 Installation and User's Guide* for more information about POST.
NCT	Each node runs the Node Confidence Test automatically when powered up. It checks the CPU, RAM, cache RAM, and peripheral components. Control is then transferred to the EPROM-resident Node Boot Monitor.

Up to this point, tests verify hardware integrity on the System Resource Manager and the node boards in a standalone environment. Use CDP to check the communication links between the cube and the System Resource Manager as well as between all the nodes and options such as the VX board and the Concurrent I/O facility.

You should be directly logged into the System Resource Manager to run CDP. CDP is not designed to operate on a remote workstation, even if you remotely log into the SRM.

The Cube Diagnostic Program runs under the UNIX operating system. It includes the following interface tests:

DLT	The Diagnostic Link Tests check the diagnostic link including the System Resource Manager's interface board, the USM, and the RS422 connection.
NST	The Node Stand-alone Tests check the Node Confidence Test (NCT) results, the node Direct-Connect Modules™ (DCM), and extended RAM on the node boards.
HLT	The Host Link Tests check the DCM link including the System Resource Manager's DCM, part of the cube DCM buffer board, node 0's DCM, and the ability of these boards to communicate with the host.
IOLT	The I/O Link Tests check the DCM link including the I/O node DCMs, part of the cube DCM buffer board, some compute node DCMs, and the ability of these boards to communicate.
NLT	The Node Link Tests check the internode communication links and node DCMs.
CLT	The Cube Link Tests check the remaining parts of the cube DCM buffer board and DCMs that HLT and IOLT did not check.
OHT	The Optional Hardware Tests check optional boards such as the VX vector board, the SX scalar processor, the Intel387™ numeric coprocessor, the Concurrent I/O facility, and the BIA board.

NOTE

CDP also tests a hybrid cube which has a mixture of node boards with varying amounts of memory and memory boards. It tests nodes configured with a vector board or an SX scalar processor. CDP automatically reads the cube and device configuration files (*/usr/ipsc/conf/cubeconf* and */usr/ipsc/conf/devconf*). Refer to Chapter 5 for more information about the cube and device configuration files.

DIAGNOSTIC PROCEDURE

The procedures for running each of the diagnostic tests are summarized below.

Power-On Self Test (POST)

The Power-On Self Test is automatically run when the System Resource Manager is powered up. It checks the Intel386 microprocessor, the keyboard, the display, system memory, and most peripheral devices connected to the SRM each time you turn on the system.

1. To turn on the monitor, push the ON/OFF switch located on the front of the video module. Turn off the power to the SRM. The ON/OFF switch is located on the right side of the SRM.

2. Turn on power to the monitor and the SRM. POST begins turning the keyboard status lights on and off.

3. A display similar to the following will then appear:

```
Phoenix 80386 ROM BIOS Version xx.xx  xx.xx
Copyright (c) 1985 - 1988 Phoenix Technologies Ltd
All Rights Reserved
```

4. The POST memory size test displays the amount of memory it has tested. It displays the values in the upper-left corner of the screen as the test progresses. POST takes from 3 to 15 seconds to complete, depending on the amount of resident memory. Standard systems are shipped with 8M bytes of RAM.

5. Adjust the monitor if the memory test is not displayed clearly.

6. When POST completes, the SRM will beep once if there were no failures.

NOTE

Two beeps or combinations of three sets of beeps indicate a system board failure. Refer to Appendix B of the *SYP301 Installation and User's Guide* to determine corrective action.

Node Confidence Test (NCT)

The Node Confidence Test is the node's equivalent to the SRM's Power-On Self Test. Each node runs its EPROM-resident copy of NCT when the iPSC cabinet is powered up or the nodes are reset using **bootcube** or **rebootcube**. You can also run NCT as part of CDP. To run NCT on the nodes of a particular cabinet, perform the following steps:

1. If the iPSC cabinet is powered up, turn the power off to the nodes. On the compact cabinet, use the key switch at the top front of the cabinet. The standard cabinet does not have a key switch. Instead, use the main power switch at the bottom back of the cabinet.

   Wait about 10 seconds. Turn the power back on.

2. Verify that both the AC and DC power indicators are lit. If they are not lit, perform the following steps:

   A. Make sure that the cabinet is plugged into a power outlet.

   B. Check the cabinet circuit breaker or breaker box if the AC indicator is not lit.

   C. If you feel confident that the cabinet is receiving power but no power indicators are lit, call SSD Customer Support.

3. After NCT has completed (within 7 seconds), verify that the green LED on each node board is lit (except for USM board(s)):

   A. If the red indicator light is lit or blinking on only one node board, perform the following steps:

      1) Turn the power to the cabinet off. Wait about 10 seconds. Turn the power back on again.

      2) If the red LED is still lit, turn the power OFF.

      3) Replace the faulty node board with a spare board. Be sure to call SSD Customer Support before replacing a node board.

      4) Turn the power on.

      5) If this does not correct the problem, call SSD Customer Support.

   B. If more than one node board has its red indicator lit, perform the following steps:

      1) Turn the power off. Wait about 10 seconds. Turn the power back on.

      2) If the red LEDs are still lit, run CDP.

C. If neither the green nor the red indicator is lit, perform the following steps:

1) Turn the power off.

2) Reseat the board and then turn the power on.

3) If reseating does not correct the problem, turn the power off and replace the node board. Be sure to call SSD Customer Support before replacing a board.

D. If the red indicator light on any node board is blinking, call SSD Customer Support.

Cube Diagnostic Program (CDP)

The Cube Diagnostic Program (CDP) checks the communication links between the cube and the System Resource Manager as well as between all the nodes. The CDP also tests the optional capability of the iPSC system. CDP tests the SX Scalar Processor (Weitek 1167), the VX vector board, the Intel387 numeric coprocessor, the Concurrent I/O facility, and the BIA board.

CAUTION

When iPSC software is installed or re-installed, the Cube Diagnostic Program (CDP) and System Acceptance Test (SAT) should be run to verify proper installation. Make sure that all of the CDP tests pass before running the SAT. Both the CDP and SAT test must pass before the installation can be considered successful.

1. To invoke CDP you must be root and be in the */usr/ipsc/diag* directory or have that path defined in your shell path variable. To make */usr/ipsc/diag* your current directory, enter the following:

```
# cd /usr/ipsc/diag
```

2. Ensure that no other users are on the cube and that all file server and communications server activity is stopped. You can stop all file and communications server activity with the following command:

```
# bootcube -D1
1.  Reset driver
```

3. To invoke CDP, enter the following:

```
# ./cdp
```

4. The Main Menu is displayed and you are prompted to enter the test you want to run. If you select, "Run Standard Tests," the DLT, NST, HLT, IOLT, NLT, CLT, and OHT tests will be run in that order using default values. To select "Run Standard Tests," enter the number next to "Run Standard Tests".

   If you peruse the CDP menus, you'll notice that some tests are listed as "extended." The extended tests are reserved for use by SSD Customer Support. They are not run as part of the Standard Tests.

   An extended test requires operator intervention. Some also require additional hardware, such as a loopback connector. The Generic Link Tests in the Main Menu are extended tests and hence not part of the Standard Tests.

5. Run **bootcube** after exiting CDP. This returns the cube to its normal state:

   ```
   # bootcube
   ```

DIAGNOSTIC STRATEGY

This section presents a diagnostic strategy for isolating problems with the iPSC system. It also recommends a course of action when intermittent problems are encountered.

Isolating Problems

Figure 7-1 shows the recommended strategy for isolating problems with the iPSC system. By following the flow diagram, incremental levels of hardware are tested.

First, power up the SRM and the monitor and run the Power-On Self Test. Next, boot the UNIX operating system. If no abnormal messages are encountered, power up the cube and run the EPROM-based test, NCT. If no failures are encountered, run CDP.

CDP consists of several tests which, if used as shown in the flow diagram, test the communication paths between the SRM and the nodes and then test the communication paths among the nodes themselves. If multiple problems exist, it is sometimes helpful to specify a smaller cube dimension to isolate a specific group of nodes for testing.

Intermittent Problems

Intermittent problems may be difficult to detect unless you can record the system's activities for an extended period of time. One way to do this is to run CDP's standard tests in continuous mode for some extended period of time. CDP maintains an error log file which the user can access through the Options Menu. To access the error log file, select "Options Menu" from CDP's Main Menu and then select "Error Log". Then, select "Display Log".

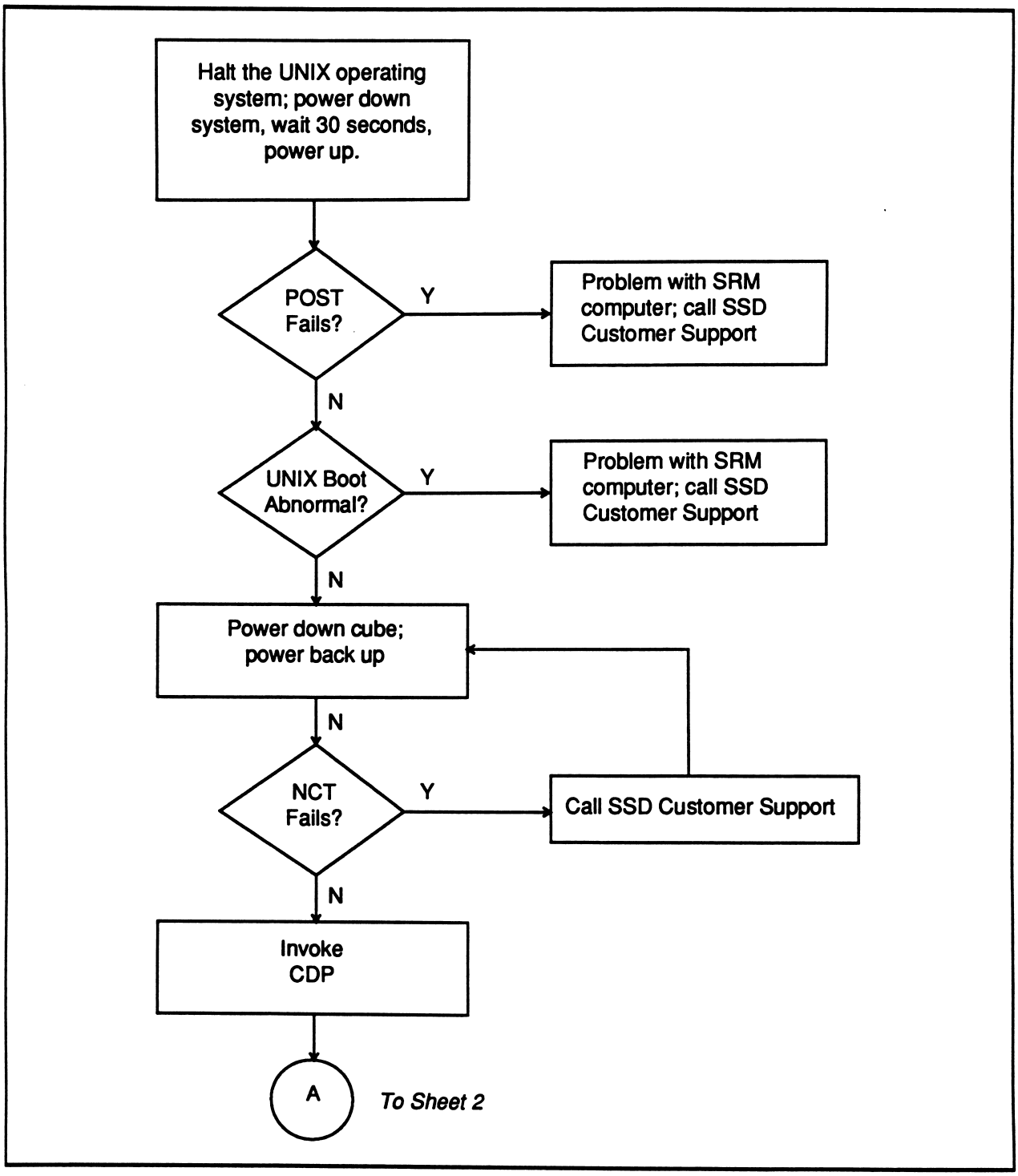

Figure 7-1. Diagnostic Strategy (1 of 2)

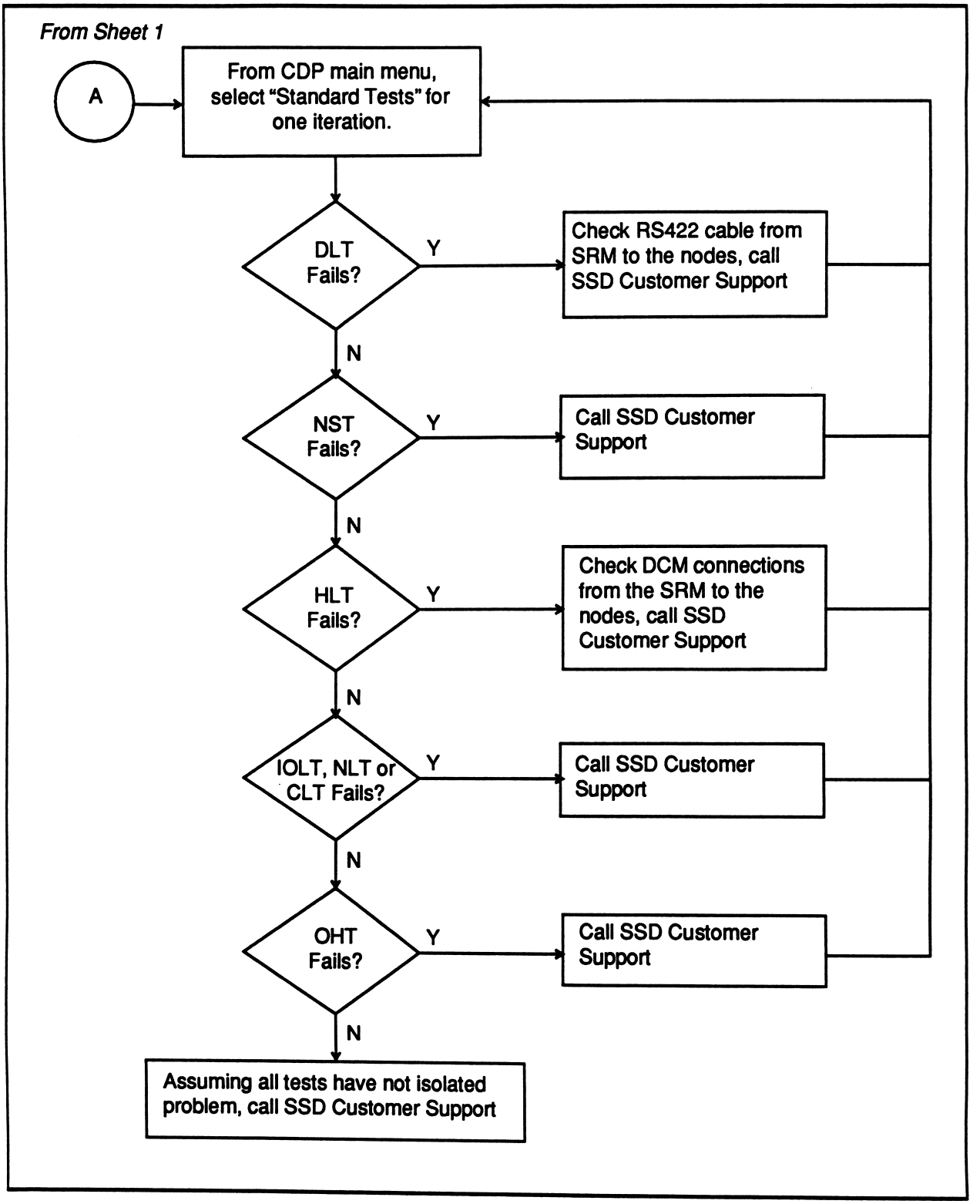

Figure 7-1. Diagnostic Strategy (2 of 2)

Corrective Action

If you receive an error message as a result of running CDP, perform the following steps:

1. Check the cable from the SRM to the nodes. The connectors should be securely in place on the SRM's back panel and on the plate at the bottom back of cabinet 0.

2. Swap the node board (or its neighbor) that you suspect has a problem with another node board. Be sure to call SSD Customer Support before swapping a board. (Refer to Chapter 6 for information about swapping boards.)

3. If you have a vector system, swap the vector processor board you suspect has a problem with a spare vector processor board.

If the above course of action does not correct the problem, it should be referred to Intel Supercomputer Systems Division. When you call for service, be prepared to supply the following information:

- Serial Number of failing unit
- Failure symptom(s)
- Description of any remedial action attempted
- CDP software version level
- A list of any CDP error messages

There are two ways to determine your software version level. One way is to invoke CDP; this displays the version level. The other way, without running CDP, is to use the UNIX **what** command. First, you must be in the directory */usr/ipsc/diag*. Then, enter the following:

```
# what cdp          shows CDP's version level
# what VX.exe       shows CDP's vector processor test version level
# what SX.exe       shows CDP's Weitek test version level
# what SCSI.exe     shows CDP's SCSI test version level
# what 387.exe      shows CDP's 387 numeric coprocessor's test version level
# what VME.exe      shows CDP's VME Bus Interface Adapter test version level
```

Direct all service inquiries to SSD Customer Support at:

1-800-421-2823 (Customer Support Hotline)
(44) 793 641 469 (in England)
Your Local Intel Sales Office (in Europe)
support@ssd.intel.com (Internet address)

COMMANDS AND SYSTEM CALLS 8

INTRODUCTION

This chapter describes (in reference manual format) the system administrator's commands and system calls:

- You enter commands at the console.
- You use system calls in programs.

SYSTEM ADMINISTRATOR COMMANDS

These are the system administrator commands:

bootcube Loads the node operating system onto the nodes of the cube. Discussed in Chapter 4.

cbackup and **crestore**
Backs up and restores a concurrent file system. Discussed in Chapter 5.

mkcfs Makes (or remakes) a concurrent file system. Discussed in Chapter 5.

mkdev Allocates a file which represents a tape drive managed by the concurrent file system. Discussed in Chapter 6.

plogon and **plogoff**
Turns logging of process-related information on and off. Discussed in Chapter 5.

The commands are described in alphabetical order.

BOOTCUBE BOOTCUBE

Load the node operating system onto the nodes of the cube. This command is for use by the system administrator only.

Syntax

bootcube [**-B** *file*] [**-b** *file*] [**-C**] [**-D***n* | **-D***m-n*] [**-d** *dim* | **-n** *nodes*] [**-f**] [**-K** *file*][**-k** *file*] [**-L**] [**-l**] [**-o**] [**-Q** *file*] [**-R** *file*] [**-r** *file*] [**-S** *file*] [**-s** *file*] [**-T** *file*] [**-U** *file*]

Arguments

-B *file* — Reads an alternative hardware definition file; the default is */usr/ipsc/lib/hbits.x*, where **x** is an extension consisting of a single alphabetic character, indicating a revision level. This is a binary file.

CAUTION

Loading bad bits can destroy the DCM hardware.

-b *file* — Loads an alternative CX node bootloader; the default is */usr/ipsc/lib/bootld*.

-C — Causes configuration information to be displayed during the bootcube process.

-D*n*; **-D***m-n* — Runs the specified steps. **-D***n* stops execution after step *n*. **-D***m-n* runs steps *m* to *n* (for example, **-D3-4** skips steps 1 and 2 and runs steps 3 and 4). Use only one **-D** on the invocation line. To list the steps without actually executing them, use **-L**. This switch is valid on the remote host.

CAUTION

Running **bootcube** without beginning at step 1 (for example, ***bootcube -D3-24***) will generally not produce a fully operational system.

BOOTCUBE *(cont.)*

BOOTCUBE *(cont.)*

-d *dim*; **-n** *nodes* Resets and reloads the NX/2 operating system into a cube the size of **-d** *dim*ension or **-n** *nodes*. If the size is not specified, the entire cube is reloaded. If you enter a size that is larger than the one specified in the cube configuration file, an error is returned. If the number of nodes you specify is not a power of two, the next higher power of two is used. All other nodes will be nonfunctional.

-f Forces **bootcube** to proceed even though all cubes have not been released. Use of this option may leave file server processes running which must be killed manually. Use of this option may also leave open **bootcube** lock files that must be deleted. This switch is valid on the remote host.

-K *file* Loads an alternative CX node operating system; the default is */usr/ipsc/lib/nx.b*.

-k *file* Loads an alternative RX node operating system; the default is */usr/i860/ipsc/lib/nx.b*.

-L Lists the **bootcube** steps without executing them. This switch is valid on the remote host.

-l Disables use of the cube from remote development machines.

-o Boots all except the disk and tape I/O subsystem, and any ENET or BIA nodes.

-Q *file* Uses an alternative cube configuration file; the default is */usr/ipsc/conf/cubeconf*.

-R *file* Uses an alternative shutdown shell file; the default is */usr/ipsc/lib/rc0*. This switch is valid only on the remote host.

-r *file* Loads an alternative RX node bootloader; the default is */usr/i860/ipsc/lib/bootld*.

-S *file* Uses an alternative start-up shell file; the default is */usr/ipsc/lib/rc1*. This switch is valid on the remote host.

-s *file* Loads alternative socket driver; the default is */usr/ipsc/lib/sockproc*.

-T *file* Uses an alternative start-up shell file; the default is */usr/ipsc/lib/rc2*.

-U *file* Uses an alternative start-up shell file; the default is */usr/ipsc/lib/rc3*.

BOOTCUBE *(cont.)*

BOOTCUBE *(cont.)*

Description

The **bootcube** command loads the NX/2 operating system into the cube and starts up system daemons. Only root can run **bootcube**. This command returns an error unless the entire cube is available (no users can have cubes allocated) unless you use the **-f** option. The **rebootcube** command is available to users who are not root, but need to reload the NX/2 operating system onto the nodes.There is a separate version of **bootcube** specifically for starting the appropriate daemons on the remote host machine. You must be root on the remote workstation to run the command. The only flags that can be used with this version are **-L**, **-D**, **-f**, **-R**, and **-S**.

The **bootcube** command affects all of the nodes in an iPSC system. Therefore, if more than one user has cubes on the system, and one user requires that the cube be rebooted, all cubes are affected.

The **bootcube** disables the **getcube** command while it is executing. If any of the cube utility commands or library calls (for example, **getcube**, **getcube()**, **showvol**, etc.) is attempted while a **bootcube** is in progress, a message to that effect is printed and the cube utility command is aborted. In addition, host programs that call **setpid()** while a **bootcube** is in progress return with an error message similar to that for cube utility commands and library calls.

Examples

1. Load the NX/2 operating system into entire cube:

```
# bootcube
```

2. Reset and load NX/2 into a 8-node cube:

```
# bootcube -n8
```

3. Load an alternative operating system:

```
# bootcube -K /usr/joelc/my_os
```

BOOTCUBE *(cont.)*

BOOTCUBE *(cont.)*

Errors

`bootcube must be run by superuser`

Only the superuser can run **bootcube**.

`You must release all cubes before running bootcube`

Unless you use the **-f** switch, **bootcube** will not run if there are allocated cubes.

`Steps specified for -D out of order`

If you specify **-D***m*-*n*, *m* must be less than *n*.

`xxx` `cannot open`

bootcube cannot open the file *xxx*.

`File` *`xxx`* `cannot be found`

bootcube cannot find the file *xxx*.

`Bootcube aborted. Another bootcube in progress`

The presence of lock file */usr/spool/locks/LCK..bootcube* was detected. This lock file will still exist after interrupting and terminating a **bootcube** in progress. You can use the **-f** switch to force **bootcube** to ignore the existence of a lock file and proceed with the **bootcube**.

`Invalid number of slots:` *`xxx`* `maximum:` *`xxx`*

The **bootcube -d** switch specifies more slots than the configuration file.

`Failure trying to download operating system`

The system hung while **bootcube** was downloading the NX/2 operating system. Retry at least once before calling SSD Customer Support.

`Node 0 failed: Cannot load operating system`

Node 0 must be working if **bootcube** is to successfully load the NX/2 operating system on the nodes.

BOOTCUBE *(cont.)*

BOOTCUBE *(cont.)*

`Error occurred while reading a port on the SRM direct connect channel`

The SRM Direct-Connect Module has a problem. Call SSD Customer Support.

`Error occurred while performing DCM reset` *x* `on the SRM direct connect channel`

The SRM Direct-Connect Module has a problem. Call SSD Customer Support.

`Error occurred while initializing hardware on the SRM direct connect channel`

The SRM Direct-Connect Module has a problem. Call SSD Customer Support.

`diagnostic link:` *xxxx*

Internal **bootcube** error. Call SSD Customer Support.

`diagnostic link: Timeout`

Nodes are unresponsive. Check cable connections.

`loadboot` *xxxx*

Internal **bootcube** error. Call SSD Customer Support.

`querycube` *xxxx*

Internal **bootcube** error. Call SSD Customer Support.

`resetcube Invalid diagnostic channel mode`

Internal **bootcube** error. Call SSD Customer Support.

`Reset timed out`

Nodes are unresponsive. Check cable connections.

`starting socket process: No such device`

The ENET address or the ENET board has a problem.

BOOTCUBE *(cont.)*

BOOTCUBE *(cont.)*

`setavail` *xxxx*

Internal **bootcube** error. Call SSD Customer Support.

`testcube` *xxxx*

Internal **bootcube** error. Call SSD Customer Support.

See Also

rebootcube

CBACKUP CBACKUP

Back up one or more CFS disk drives. This command is for use by the system administrator only.

NSH Syntax

cbackup [**-s** *mediasize*] [**-a**] [*volume...*] [*filename*]

Arguments

-s *mediasize*	Allows you to specify the size in bytes of the media used for backup.
-a	Backs up all of the blocks in each volume, whether the blocks contain data or not. Without this parameter, only those blocks containing data are backed up.
volume...	A space-separated list of the numbers of the disk drives (volumes) that you want to back up. If no volume numbers are listed, the command will back up all volumes.
filename	The name of the file or device to which the backup is to be written. If no file or device is named, the backup is written to *stdout*, allowing you to pipe the output to other commands, such as **tar**.

Description

NOTE

This command only operates within the **nsh** command. You must be logged in as "root" before this command will be successful.

The **cbackup** command lets you back up the volumes (disk drives) of the concurrent file system. The *filename* parameter is the name of a file or device (such as */cfs/tape*) to which you are backing up the data. Without the *filename* parameter, the backup goes to *stdout*. Use the **crestore** command to restore the backup. It is good practice to issue a **getcube** to reserve an entire cube before performing backup procedures.

By default, the **cbackup** command backs up only those blocks that contain data. However, if you use the **-a** option, the entire disk image will be backed up, regardless of whether blocks contain data. Data passed to the backup device are buffered up to 256K bytes.

CBACKUP *(cont.)* CBACKUP *(cont.)*

The *volume* parameter lets you specify the volume number(s) that you want to back up. With no volume numbers specified, all disk drives are backed up.

The -s parameter lets you specify the size (in bytes) of the media to which you are backing up the information. If you have more data to back up than the tape can hold, specify the size to avoid losing data. A "K" after the number specifies kilobytes (1024 bytes). For example, 600K bytes is interpreted as 600 kilobytes. Similarly, an "M" after the number specifies megabytes (1048576 bytes). A "G" after the number specifies gigabytes. The number before the magnitude specifier must be a whole number. The standard cartridge tape size is 45M bytes. The 2-hour, 8mm tape is 2.2G bytes.

Example

To back up the data on volumes 1 and 2 to a tape, designated as the device */cfs/tape*, specifying a maximum backup space of 500M bytes, enter the following:

```
# nsh
# cbackup -s 500M 1 2 /cfs/tape
```

Errors

`cbackup: Media size too small`

Ensure that you specify the media size correctly or use larger backup media.

`cbackup: Bad disk number`

Use **showvol** to find valid disk numbers.

`cbackup: Cannot create` *nnn*

File name is invalid. Use the UNIX filename conventions.

`cbackup: Disk Process not initialized`

Reboot the cube and initialize the disk processes.

`cbackup: Not enough memory for bitmap`

Not enough room for the buffer on the SRM disk. Obtain more space on SRM disk.

CBACKUP *(cont.)*

CBACKUP *(cont.)*

`cbackup: Bitmap error`

Error in the disk process. Try rebooting with a **bootcube** or **rebootcube** command.

`cbackup: Write error`

Problem with the backup media. Check to see that the end of the tape is not overwritten.

See Also

crestore

CRESTORE CRESTORE

Restore a saved CFS image to the volume(s) it came from. This command is for use by the system administrator only.

NSH Syntax

crestore [*volume...*] [*filename*]

Arguments

volume...	A space-separated list of the volume numbers of the disk drives to which the backup data are to be restored. Only volumes that have been backed up may be included in the list. If you do not list any *volume* numbers, all volumes backed up to that file will be restored.
filename	The name of the file or device that contains the backup. If no file or device is named, *stdin* is read.

Description

NOTE

This command only operates within the **nsh** command. You must be logged in as "root" before this command will be successful.

The **crestore** command lets you restore previously backed-up disk images to the disks from which they were backed up. The *filename* parameter is the name of a file or device (such as */cfs/tape*) from which the backed-up data are to be read. If you do not use the *filename* parameter, *stdin* is read.

If you do not list any *volume* numbers, **crestore** will automatically restore all volumes backed up to the specified file or device. You can only restore data to a volume from which the data have been previously backed up with the **cbackup** command. Your volume list may be a subset of the volumes previously backed up; if so, only the listed volumes will be restored. Use the **showvol** command to get a list of valid *volume* numbers.

It is good practice to issue a **getcube** to reserve an entire cube before performing backup procedures.

CRESTORE *(cont.)* CRESTORE *(cont.)*

Example

1. To restore from */cfs/tape* the backup previously done of drives 1 and 2, enter the command (notice that since all backed-up data on */cfs/tape* are to be restored in this example, it is not necessary to specify the volume numbers):

```
# crestore /cfs/tape
```

2. To restore backed-up data to volumes 0, 1, and 3 only from a file named *bak.mar1*:

```
# crestore 0 1 3 bak.mar1
```

Errors

`crestore: Bad disk number`

Use **showvol** to find valid volume numbers.

`crestore: Cannot open` *nnn*

File name is invalid. Make sure file name is correct.

`crestore: Disk process not initialized`

Reboot the cube and initialize the disk processes.

`crestore: Cannot open tty`

To avoid conflicting with *stdio*, **crestore** opens another device during the transfer. Make sure the tty device is in */dev*.

`crestore: Invalid backup format`

Error in backup media.

`crestore: Incorrect media label number`

Error in media. Make sure you are restoring to the same drives backed up.

CRESTORE *(cont.)*

CRESTORE *(cont.)*

```
crestore: Invalid block number
```

Error in backup media.

```
crestore: Read error
```

Problem with backup media. Check to see that the end of the tape is not overwritten.

See Also

cbackup, showvol

MKCFS MKCFS

Construct a file system. Intended for use by system administrator only.

CAUTION

This command is to be used by your system administrator only. If your file system has existing files on it, they will be destroyed in creating a new concurrent file system (CFS).

Syntax

mkcfs

Arguments

None

Description

The **mkcfs** command constructs a new concurrent file system, or allows you to add new drives to the current system. Before running **mkcfs**, execute the commands:

```
# bootcube -D22
```

The **bootcube** command with the -D22 option is required to initialize the disk software without initializing the file system. The size of the cube you get with **getcube** is not important because **mkcfs** requires only the cube software, not the nodes themselves.

The **mkcfs** command has no options; instead, it is interactive, giving you several opportunities to discontinue the process so you do not accidently replace an existing CFS with a new one. After running **mkcfs**, perform a **bootcube** command. The command creates a volume label for each drive in the system. You define the file system name when you execute **mkcfs**. The volume label identifier and version are predefined.

If there is no existing file system, it prompts you for a name for the file system, and then builds the CFS. To see a list of available drives, execute the **showvol** command. If there is an existing file system, entering the command causes this response:

```
n drives ready on m nodes
File system name already exists. Continue?
```

MKCFS *(cont.)*

MKCFS *(cont.)*

You can respond either with a *y* or an *n*. If your response is negative (*n*), the CFS remains unchanged. If your response is *y*, it will ask you if you want to change the file system name. If you do, it will prompt you for a new name and create a new CFS, assigning volume numbers to all drives. File system names may have spaces within the name, and a maximum of 30 characters.

If you do not want to change the name, but only want to add new drives to the file system, it will detect the new drives, and ask you if you need labels for the new drives. If you respond with *y* (yes), it will supply all new drives with labels, either all at once, or asking you for each new drive individually.

Examples

Add 8 new drives to an existing CFS with 32 existing drives.

```
# mkcfs
32 drives ready on 4 nodes
File system "old cfs" already exists. Continue? y
Do you wish to change the file system name? y
Enter new file system name: new cfs
Are you sure you want to destroy all files? y
#
```

Errors

`mkcfs: Disk process not initialized`

Reboot the cube and initialize the disk processes.

`mkcfs: Invalid node` *nnn* `for volume` *nnn*

Reboot the cube and reinitialize the disk processes.

`mkcfs: Bootcube in progress`

A **bootcube** is in progress. Try again when the **bootcube** is completed.

See Also

bootcube, getcube, showvol

MKDEV MKDEV

Allocates a file that represents a tape managed by CFS.

NSH Syntax

mkdev *volume devicename*

Arguments

volume	the volume number for the device. It is the same volume number returned by the **showvol** command.
devicename	the name for the file representing the device. It must be a complete path and filename.

Description

NOTE

This command only operates within the **nsh** command. You must be logged in as "root" before this command will be successful.

The **mkdev** command allocates a file which represents a tape drive managed by CFS. Referencing this file will access the device. This file resides only in CFS. Use the **showvol** command to find the volume number of your tape devices.

MKDEV *(cont.)* MKDEV *(cont.)*

Examples

The following example creates a CFS file which represents the device whose volume number is 3. This file would have to be created in order to access a tape device.

```
mkdev 3 /cfs/mt0
```

Errors

`Usage: mkdev volume device`

Either the *volume* or *devicename* arguments are missing.

`mkdev: volume number: Invalid argument`

The *volume* argument is not a valid volume number.

`mkdev: Pathname must be within CFS`

The *devicename* argument must be within the CFS.

See Also

nsh, tapemode

PLOGON, PLOGOFF PLOGON, PLOGOFF

Log process related information such as execution of **bootcube**, cube allocation, cube deallocation, process initiation and process completion. This command is for use by the system administrator only.

Syntax

plogon [**-a**] [**-c**] [**-i**] [*pathname*]
plogoff

Arguments

-a	Append logging information to *pathname* (if it exists).
-c	Turn off process related information.
-i	Query for current output log file name. No change of logging state.
pathname	The output log file. If no file is specified, the default is */usr/ipsc/log/process.log*.

Description

The **plogon** command opens a log file and logs process related information. You can specify a log file on the command line or use the default */usr/ipsc/log/process.log* log file. The **plogon** command logs seven types of actions: execution of **bootcube**, allocating a cube, loading a process onto nodes, completing a process, executing **plogon** or **plogoff**, and releasing a cube. With the **-c** option, **plogon** logs only the allocation and release of cubes. Upon completion, **plogon** always prints the name of the output log file on *stdout*.

PLOGON, PLOGOFF *(cont.)*

PLOGON, PLOGOFF *(cont.)*

Starting Process Logging

When executing **plogon**, one of three possible entries is created in the log file. For example:

```
Cube 2          Wed Apr 27 13:59:41 1989  Plogon: Start logging.
```

is the entry created when a log file is first opened by **plogon.**

```
Cube2           Wed Apr 27 13:58:51 1989
   Plogon: Already logging to this file (/usr/ipsc/log/process.log)
```

is the entry created when **plogon** is run twice specifying the same output log file with no intervening **plogoff** execution. Running **plogon** twice in a row may be desirable in order to change the state of the (**-c**) cubeonly switch. Logging will continue to the same output file.

```
Cube 2        Wed Apr 27 13:49:41 1989
  Plogon:  Switching to log file /tmp/process.log.
```

is the entry created when **plogon** is run twice with no intervening **plogoff** with the second **plogon** specifying a new output log file. The new logfile will have a **plogon** start logging entry.

Booting the Cube

When the first steps of **bootcube** are executed an entry will be made in the log file if process logging is still running from a previous invocation of **plogon.** The form of the log entry is:

```
Cube 2:     Wed Apr 27 13:49:41 1989
    Commserver terminated.
```

PLOGON, PLOGOFF *(cont.)*

PLOGON, PLOGOFF *(cont.)*

Allocating a Cube

When a cube is allocated, the following information is logged: cube ID, time stamp, the action (a **getcube**), the cube owner, the cube name, the host network name, the process ID of the process that ran the **getcube**, and the cube type. For example:

```
Cube 2          Wed Apr 27 13:59:41 1989
    Getcube: owner joelc name cube1 host acda pid 4470 type 1m8n4
```

This cube has a cube ID of 2, its index in the cubeinfo table. It was allocated on Wednesday, April 27, 1989 at 13:56:41. The user executing the command was *joelc*. The name of the cube was *cube1*, and the network name of its host was *acda*. The process that ran the **getcube** was 4470. If a program issued **getcube**, this *pid* is the UNIX *pid* of that program. If a user issued **getcube** at the command line, this *pid* is the UNIX *pid* of the process that was created to run the **getcube**. This *pid* is also displayed in the corresponding **relcube** log entry. The allocated cube has one node with 8M bytes of memory in slot 4.

Loading a Process Onto Nodes

When a process is loaded onto a cube, the following information is logged: the cube ID, timestamp, the action (a new process), the new process *pid*, the set of physical slot IDs, and the filename of the loaded process. The set of physical slot IDs is represented as a binary number displayed in hex. A set bit indicates that the node in that slot was loaded. For example, 0x00000000F identifies slots 0, 1, 2, and 3; 0x000000001 identifies slot 0. Here is a typical entry:

```
Cube 4          Wed Apr 27 13:59:41 1989
New process: pid 0 nodeset 0x000000010 /usr/joelc/tests/hello
```

This cube has a cube ID of 4. It was loaded on Wednesday, April 27, 1989 at 13:63:41. The node process loaded had an NX/2 *pid* of 0. It was loaded onto the node in slot 4. The filename of the process was */usr/joelc/tests/hello*.

PLOGON, PLOGOFF *(cont.)*

PLOGON, PLOGOFF *(cont.)*

Completing a Process

When a node process completes, the following information is logged: the cube ID, time stamp, the action (process done), the NX/2 *pid* of the completed process, the node slot on which the process ran, and the process exit code. For example:

```
Cube 4          Wed Apr 27 13:61:41 1989
Process done: pid 0 nodeset 0x10 e-code 0
```

This cube has a cube ID of 4. A process running on this cube completed on Wednesday, April 27, 1989 at 13:57:41. The loaded process had an NX/2 *pid* of 0. It was loaded onto the node in slot 4. Its exit code was 0.

Releasing a Cube

When a cube is deallocated, the following information is logged: cube ID, time stamp, the action (a **relcube**), the cube owner, the cube name, the host network name, the process ID of the process that ran the corresponding **getcube**, and the cube type. For example:

```
Cube 2          Wed Apr 27 13:63:41 1989
    Relcube: owner joelc name cube1 host acda pid 4470 type 1m8n4
```

This cube has a cube ID of 2, its index in the cubeinfo table. It was released on Wednesday, April 27, 1989 at 13:57:41. The user executing the command was *joelc*. The name of the cube was *cube1*, and the network name of its host was *acda*. The *pid* of the process that ran the corresponding **getcube** was 4470. The released cube consisted of one node with 8M bytes of memory in slot 4.

PLOGON, PLOGOFF *(cont.)*

PLOGON, PLOGOFF *(cont.)*

Example

To turn on logging and store the log in */usr/joelc/mylog*, enter the following:

```
# plogon /usr/joelc/mylog
```

To turn off logging, enter the following:

```
# plogoff
```

Error

`Permission denied`

Only the superuser can run a **plogon** or **plogoff**.

`No active log exists`

The **plogoff** command cannot find a log function to terminate.

C SYSTEM CALLS

There are two C system calls:

plogon	Turns on logging of process-related information. (See Chapter 6.)
plogoff	Turns off logging of process-related information. (See Chapter 6.)

PLOGON(), PLOGOFF() PLOGON(),PLOGOFF()

Log process related information such as execution of **bootcube, plogon, or plogoff,** cube allocation, cube deallocation, process initialization, and process completion. Requires root privilege to run.

Synopsis

plogon(*append, noprocs, query, logfile*)
plogoff()

Parameter Declarations

plogon

short *append, noprocs, query*;
char **logfile*;

plogoff

None

Return Value

None

PLOGON(), PLOGOFF() *(cont.)*

PLOGON(), PLOGOFF() *(cont.)*

Example

This example turns on logging to the file */usr/joelc/mylog*. Execution of **bootcube, plogon,** or **plogoff,** cube allocation, cube deallocation, process initialization, and process completion are logged:

```
static char logfile[] = "/usr/joelc/mylog";
      .
      .
plogon(0,0,0,logfile);                    /* logging turned on */
      .
      .
plogoff();                                /* logging turned off */
      .
      .
```

Environment

Host, Node

Languages

C, Fortran

Description of Parameters

append is a short integer. When it is not 0, the logging information is appended to *logfile* (if *logfile* exists). When it is 0, the logging information overwrites an existing *logfile*.

noprocs is a short integer. When it is not 0, no process loading or termination information is logged. When it is 0, process initialization, and process completion are logged as well as execution of **bootcube, plogon, plogon(), plogoff, plogoff(), getcube, getcube(), relcube,** and **relcube()**.

query is a short integer that when not 0 causes **plogon()** to return the name of the current logfile in the parameter *logfile*. No other action is performed.

logfile is a string that identifies the pathname of the log file. If it is null, the default */usr/ipsc/log/process.log* is used.

PLOGON(), PLOGOFF() *(cont.)*

PLOGON(), PLOGOFF() *(cont.)*

Discussion

plogon() turns on the logging function, and **plogoff()** turns it off. If already logging to a file, **plogon()** continues logging to the same file or a new one if a different file was specified.

The process executing **plogon()** or **plogoff()** must be running at root privilege. If the process does not have root privilege, it aborts when it encounters the call.

Errors

`plogon: Permission denied`

Only the superuser can run **plogon()**.

`plogoff: Permission denied`

Only the superuser can run **plogoff()**.

`plogoff: No active log file exists`

plogoff() cannot find a log function to terminate.

FORTRAN SYSTEM CALLS

There are two Fortran system calls:

PLOGON() Turns on logging of process-related information. (See Chapter 6.)

PLOGOFF() Turns off logging of process-related information. (See Chapter 6.)

PLOGON(), PLOGOFF()

PLOGON(),PLOGOFF()

Log process related information such as execution of **bootcube, plogon,** or **plogoff,** cube allocation, cube deallocation, process initialization, and process completion. Requires root privilege to run.

Synopsis

SUBROUTINE **PLOGON**(*append, noprocs, query, logfile*)
SUBROUTINE **PLOGOFF**()

Parameter Declarations

PLOGON

INTEGER*2 *append, noprocs, query*
CHARACTER *logfile*(*)

PLOGOFF

None

Return Value

None

PLOGON(), PLOGOFF() *(cont.)*

PLOGON(), PLOGOFF() *(cont.)*

Example

This example turns on logging to the file */usr/joelc/mylog*. Execution of **bootcube, plogon,** or **plogoff,** cube allocation, cube deallocation, process initialization, and process completion are logged:

```
      CHARACTER*16 FNAME
      FNAME = '/usr/joelc/mylog'
          .
          .
C LOGGING TURNED ON
      CALL PLOGON(0,0,0,fname)
          .
          .
C LOGGING TURNED OFF
      CALL PLOGOFF()
          .
          .
```

Environment

Host, Node

Languages

C, Fortran

PLOGON(), PLOGOFF() *(cont.)*

PLOGON(), PLOGOFF() *(cont.)*

Description of Parameters

append is an integer. When it is not 0, the logging information is appended to *logfile* (if *logfile* exists). When it is 0, the logging information overwrites an existing *logfile*.

noprocs is an integer. When it is not 0, no process loading or termination information is logged. When it is 0, process initialization, and process completion are logged as well as execution of **bootcube, plogon, plogon(), plogoff, plogoff(), getcube, getcube(), relcube,** and **relcube()**.

query Will only print the current log file name on *stdout*. Does not change the **plogon** state.

logfile is a string that identifies the pathname of the log file. If it is null, the default */usr/ipsc/log/process.log* is used.

Discussion

plogon() turns on the logging function, and **plogoff()** turns it off. If already logging to a file, **plogon()** continues logging to the same file or a new one if a different file was specified.

The process executing **plogon()** or **plogoff()** must be running at root privilege. If the process does not have root privilege, it aborts when it encounters the call.

Errors

`plogon: Permission denied`

Only the superuser can run **plogon()**.

`plogoff: Permission denied`

Only the superuser can run **plogoff()**.

`plogoff: No active log file exists`

plogoff() cannot find a log function to terminate.