October 1993
Order Number: 312824-001

# Paragon™

# High Performance

# Parallel Interface Manual

Intel® Corporation

| REV. | REVISION HISTORY | DATE |
|---|---|---|
| -001 | Original Issue | 10/93 |

## WARNING

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

## CAUTION

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

## LIMITED RIGHTS

# Preface

The American National Standards Institute (ANSI) has standardized a high speed external connection for supercomputers. That product is called High Performance Parallel Interface, or HIPPI. This manual describes the Paragon™ HIPPI controller and explains how to install and configure the controller in a Paragon system.

**NOTE**

Because Paragon™ HIPPI controllers are supported by both Paragon™ XP/S and Paragon™ XP/E systems, this manual discusses the HIPPI controller in generic terms (i.e., as part of a Paragon system).

## Audience

This manual has two audiences:

- Intel Customer Support Engineers who have completed the Intel Supercomputer Systems Division Paragon Training Class will be primarily interested in those portions of the manual that discuss installation procedures (Chapters 3, 5, and 6).
- Engineers and system administrators who need to understand how the HIPPI controller works on their system will be primarily interested in the portions of the manual that discuss HIPPI packets and the raw HIPPI interface library, *libhippi.a* (Chapters 1, 2, and 4, plus Appendix A).

**NOTE**

This manual assumes that you understand the ANSI HIPPI specifications HIPPI-SC, HIPPI-LE, and HIPPI-FP.

# Organization

| | |
|---|---|
| Chapter 1 | Introduces HIPPI usage, protocol, and standards. |
| Chapter 2 | Describes the HIPPI controller, its architecture, and its operation. |
| Chapter 3 | Explains how to install a HIPPI controller. |
| Chapter 4 | Discusses software configuration issues (network configuration, packet building, and raw HIPPI). |
| Chapter 5 | Discusses HIPPI diagnostics. |
| Chapter 6 | Describes the HIPPI internal, external, and loopback (diagnostic) cables. |
| Appendix A | Contains manual pages for the raw HIPPI library (*libhippi.a)* and the HIPPI commands (**hippi_setmap**, **hippi_showmap** and **set_hippi_buffers**). |

# 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 information about limitations and workarounds, see the *Paragon™ System Software Release Notes for the Paragon™ XP/S System*. Release notes are also located in the directory */vol/share/release_notes* on your Paragon system.

For more information, refer to the following manuals:

Intel Supercomputer Systems Division manuals:

*Paragon™ User's Guide*
312489-001
Provides an overview of the Paragon OSF/1 operating system. Tells how to develop and run programs.

*Paragon™ Commands Reference Manual*
312486-001
Provides detailed information about the commands for the Paragon OSF/1 operating system.

*Paragon™ Hardware Maintenance Manual*
312822-001
Provides detailed maintenance information for the Paragon system.

*Paragon™ System Administrator's Guide*
312544-001
Provides detailed instructions for system administration for the Paragon system.

ANSI documents:

*High-Performance Parallel Interface-Physical Switch Control (HIPPI-SC)* REV 1.7
X3T9.3/91-023

*High-Performance Parallel Interface-Framing Protocol (HIPPI-FP)* REV 4.2
X3T6/91-146

*High-Performance Parallel Interface-Physical Layer Protocol (HIPPI-PH)* REV 8.1
X3T6/91-127

*High-Performance Parallel Interface-Link Encapsulation Protocol (HIPPI-LE)* REV 2.0
X3T9/90-119

# Comments and Assistance

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

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

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

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

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

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

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

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

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

**techpubs@ssd.intel.com**

# Table of Contents

## Chapter 1
## Introduction



## Chapter 2
## Theory of Operation

# Chapter 3
# Installing the HIPPI Controller



# Chapter 4
# Software Configuration

# Chapter 5
# HIPPI Diagnostics



# Chapter 6
# Cable Parts and Specifications

# Appendix A
# HIPPI Calls



# Appendix B
# HIPPI Commands

# List of Illustrations

# Introduction 1

The Paragon™ High Performance Parallel Interface controller (called the HIPPI controller in this manual) is a daughtercard that attaches to a Paragon general purpose node board (GP node board) and provides high speed communications with other, heterogeneous systems, networks, and devices.

This chapter briefly describes how the HIPPI controller is used, how data is transmitted across the HIPPI channel, and what the HIPPI standards are.

## How the HIPPI Controller is Used

As shown in Figure 1-1 on page 1-2, the HIPPI controller is typically used in one of three ways:

- As a *channel device* between a Paragon system and a disk farm or other mass storage device.
- As a *network device* that enables applications to communicate with remote nodes by using TCP/IP (Transmission Control Protocol /Internet Protocol). This form of communication relies on an external switch.
- As a *point-to-point* device that enables two Paragon systems to communicate.

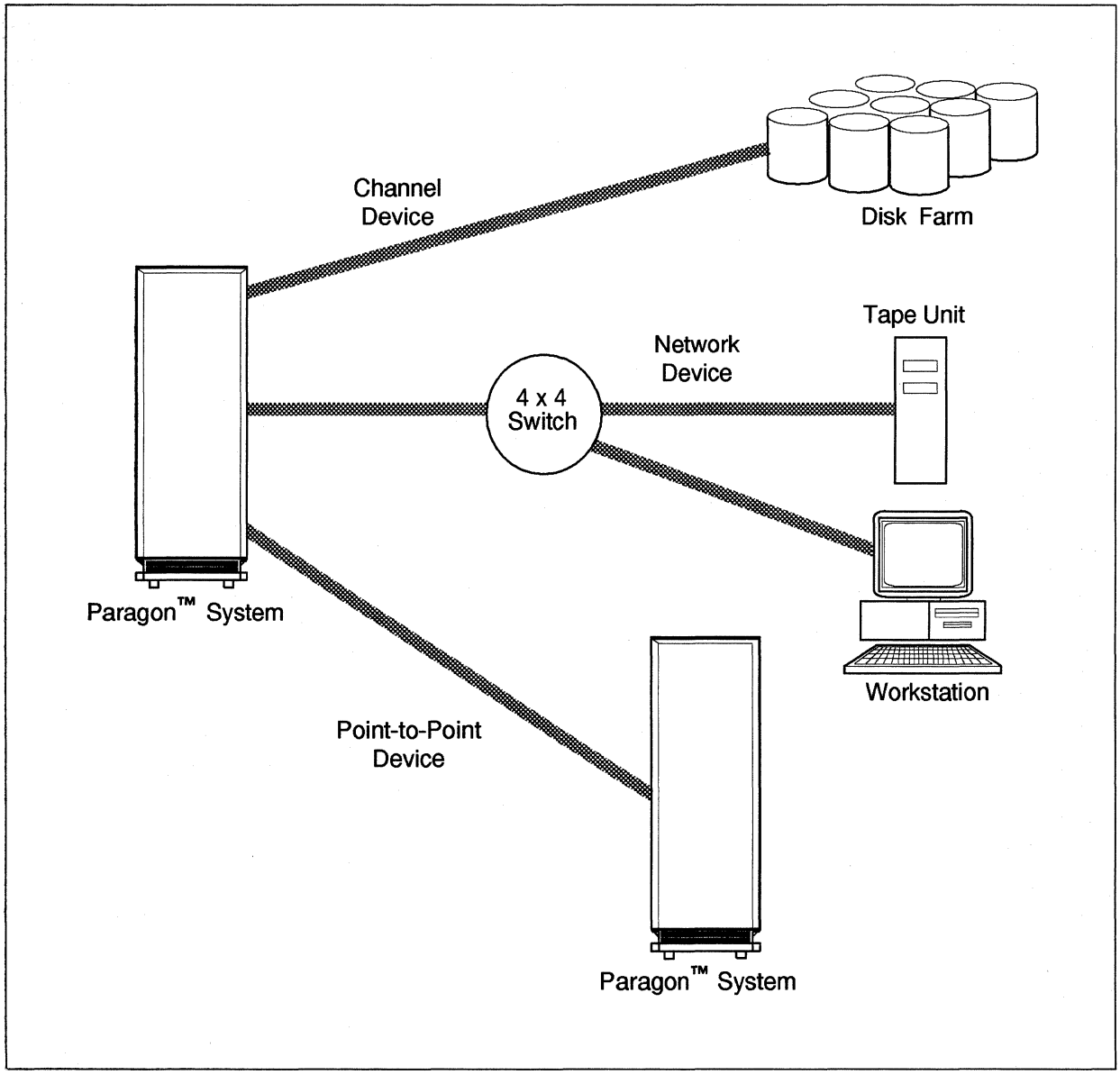


**Figure 1-1. How HIPPI is Used**

# HIPPI Protocol

The HIPPI controller transmits bursts of data using the protocol illustrated in Figure 1-2. Every time a connection is established, one or more packets are sent across the HIPPI channel. Each packet contains one or more bursts. Bursts contain the actual data and are 1 word wide (32 bits of data, 4 bits of parity) and 256 words long. To accommodate packets that are not exact multiples of 256, the ANSI standard states that either the first burst or the last may be shorter than 256 words. To implement the standard, the Paragon HIPPI controller sends the short burst last. Refer to Chapter 4 for more information about how the controller transmits data.

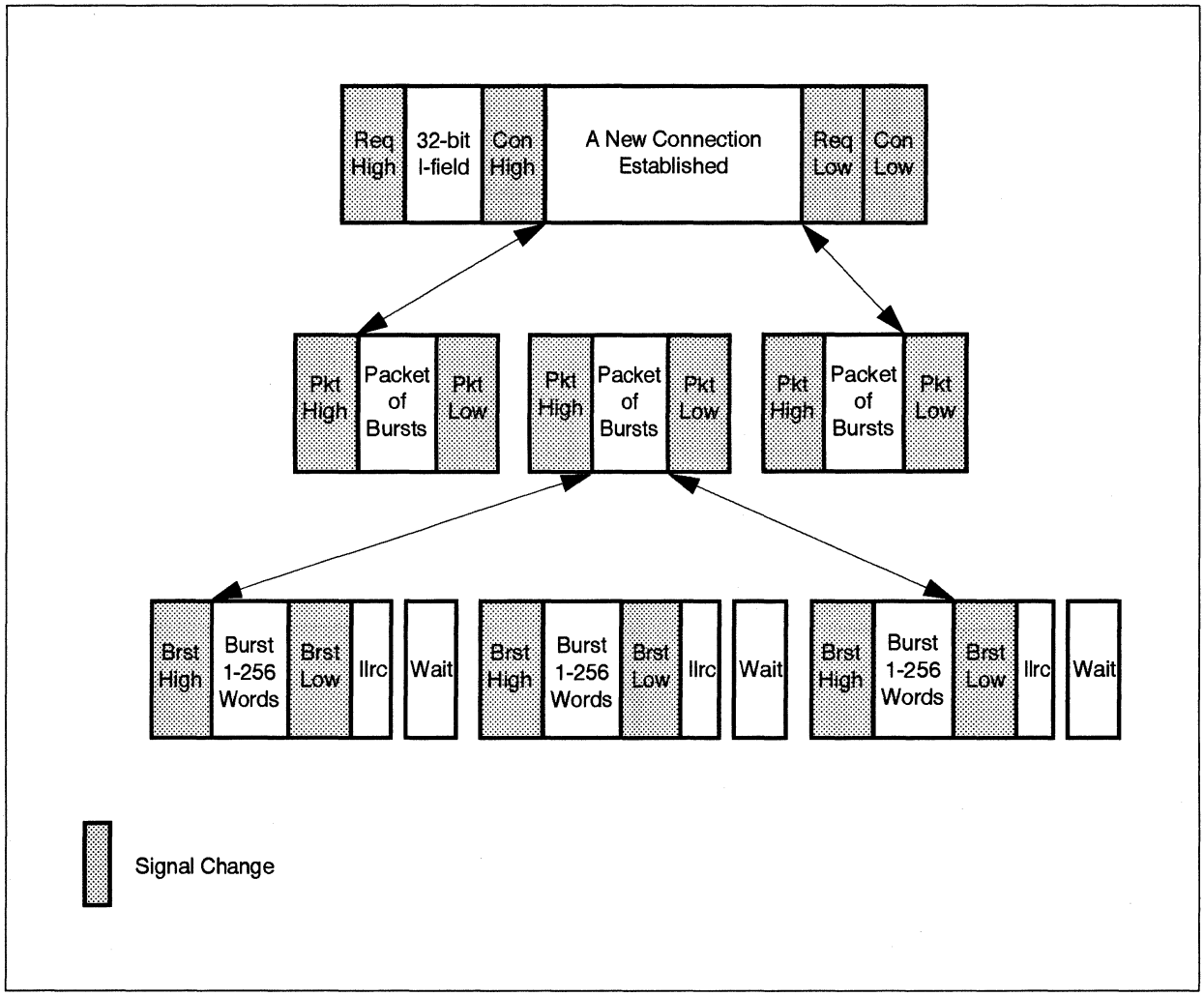


**Figure 1-2. HIPPI Protocol**

# Standards

The ANSI HIPPI standard is composed of six individual standards that address the Physical and the Data Link layer (the lowest two layers) of the International Standards Organization (ISO) reference model. The ANSI HIPI standard does not address the other elements of the ISO reference model. The six ANSI HIPPI standards are:

- The Physical standard (HIPPI-PH).
- The Switch Control Facility standard (HIPPI-SC).
- The Framing Protocol standard (HIPPI-FP).
- The Link Encapsulation standard (HIPPI-LE).
- The Intelligent Peripheral Interface standard (HIPPI-IPI-3).
- The Memory Interface (HIPPI-MI).

The following sections discuss the first four of these standards.

## The Physical Standard

The Physical Standard defines the mechanical, electrical, and signaling protocol specifications for a one-way, point-to-point interface. The Physical Layer uses a pair of twisted copper cables with a maximum length of 25 meters.

Data transfer occurs via data bursts. Each burst contains up to 256 words, each containing 32 bits of data and four bits of parity. A word is transferred during every 25 Mhz clock cycle. In general, the Physical Layer operates by host-based flow control (or READY signals) that regulate the transmission of each data burst. Look-ahead flow control (or sending multiple READY signals) allows the average data transfer rate to approach the peak transfer rate of 800M bits per second.

## The Switch Control Facility Standard

The Switch Control Facility Standard defines the control for physical layer switching in a HIPPI environment. A physical switch provides the method for interconnecting multiple HIPPI-based systems. The Switch Control Facility provides source routing and destination addressing support along with supporting different switch sizes. The Switch Control Facility does not generate the I-field, but it defines how a switch should interpret the I-field. Refer to "The I-Field" on page 2-8 for more information.

## The Framing Protocol Standard

The Framing Protocol Standard defines a common data framing protocol that supports large data transfers, a best-effort delivery mechanism with no error recovery, and an identifier field for multiplexed upper layer protocols. One HIPPI frame is equivalent to one packet. Each packet contains one or more bursts and each burst can contain anywhere from 1 to 256 words. Each word contains four bytes (32 bits of data and four parity bits). If a burst contains less than the maximum of 256 words, it is referred to as a short burst. The Framing protocol consists of three parts:

- The FP_Header_Area
- The D1_Data_Area
- The D2_Data_Area

Refer to "Server Interface and Packet Building" on page 4-5 for more information about the framing protocol.

## The Link Encapsulation Standard

The Link Encapsulation Standard defines the methodology by which packets of data can be transported. The standard conforms to the IEEE 802.2 and ISO 8802-2 Logical Link Control standards. The Link Encapsulation Standard also provides the methodology for sending the 48-bit destination and source addresses which conform to the IEEE 802.1A standard.

# Theory of Operation 2

This chapter describes the HIPPI controller, its architecture, and its operation.

## The HIPPI Controller

As shown in Figure 2-1 on page 2-2, the HIPPI controller is a daughtercard that mounts on a Paragon general purpose (GP) node board via the node board's expansion interface. The controller measures 8.7 by 9.8 inches and has its own front panel connectors and LED cutouts. (The HIPPI controller's front panel replaces the front panel of the node board.)

The HIPPI controller contains the logic and connectors for two 32-bit simplex HIPPI channels. One channel is called the *source channel* (for outgoing data) and the other is called the *destination channel* (for incoming data). The HIPPI controller is a dual-simplex I/O channel. *Only one HIPPI node is required in order to achieve full dual-simplex operation.*

The HIPPI controller is a high-speed, direct memory access (DMA) device that transfers data, commands, and status between the GP node board memory and the HIPPI channel. The controller is designed to transfer data on both channels simultaneously. An on-board microcontroller manages channel connect and disconnect functions as well as burst-level and packet-level data transfers. The host GP node supplies power to the HIPPI controller through the expansion connector and mounting standoffs.

Packet descriptors in host node memory are constructed by the driver and include a pointer to a list of variable-sized memory blocks. The lists are implemented using a series of packet and buffer descriptors (data structures containing pointers and control and status bits that describe the buffers).

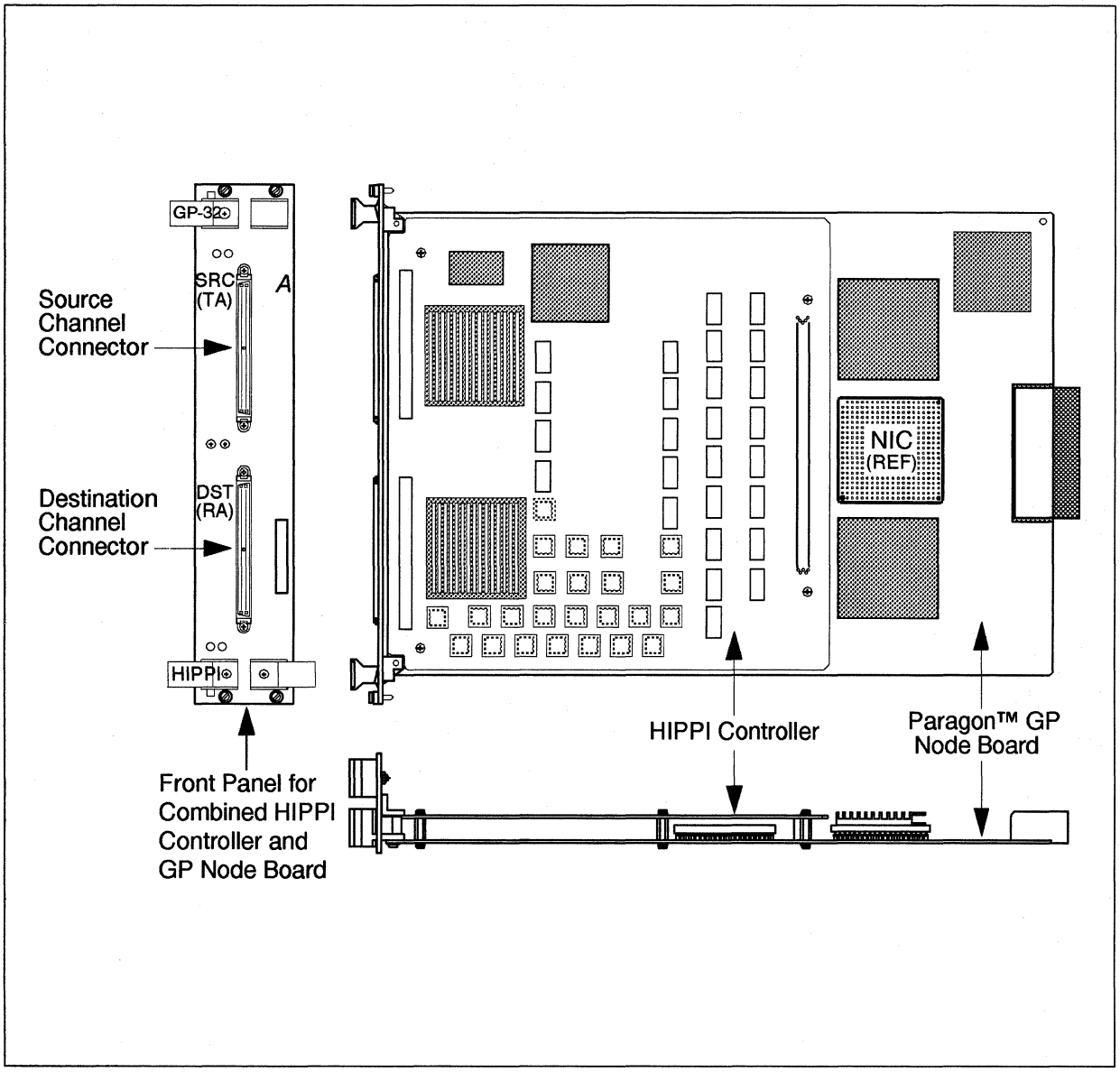


**Figure 2-1. The HIPPI Controller Mounted on a Paragon™ GP Node Board**

The HIPPI controller contains the following components:

- AMCC S2020 source interface device.
- AMCC S2021 destination interface device.
- Data first-in-first-out (FIFO) memories.
- 80960CF microcontroller with 32KB of RAM.
- Flash EPROM for 80960CF code, configuration, and diagnostics.
- Local and host expansion bus control logic.
- Control and status registers.
- Direct memory access control logic.
- HIPPI cable connectors.
- Activity light emitting diodes (LEDs).
- HIPPI controller performance monitor.

# HIPPI Controller Architecture

Architecturally, the HIPPI controller has two parts: a decision making unit and a data transfer unit. The decision making unit is a microcomputer consisting of an 80960 processor, RAM, flash RAM, and all status and control registers. The status registers provide the current status of the entire controller hardware. The control registers provide a means to exercise control over the controller hardware and to interrupt the host.

The data transfer unit consists of DMA logic, source and destination FIFOs, and source and destination HIPPI interfaces. The DMA logic transfers *blocks* of data between the node memory and the FIFO. A block consists of a maximum of 512 64-bit word transfers that are equivalent to four 32-bit word HIPPI bursts.

The host node has complete access to the controller's hardware. The controller, however, can only access the node board's memory. If the node and controller attempt to access each other's hardware simultaneously, the 80960 processor backs off until the host access cycle completes. On the controller side, the DMA and the 80960 processor should not access the node memory simultaneously. Arbitration logic is not provided, and simultaneous access will cause errors.

DMA accesses for node memory are aligned on a cache-line boundary and are not cache-coherent. However, the 80960 processor access of node memory could be from any byte and is cache-coherent. The host should not cache data that it shares with the DMA, but it can cache its shared data with the 80960 processor. Data shared with the DMA includes the HIPPI data that will be sent out or has been received. Data shared with the 80960 includes data structures.

# HIPPI Controller Operation

This section discusses the following:

- Basic HIPPI framing protocol.
- Data flow in a HIPPI controller.
- Packet transmission.
- The I-field.
- Switches in the HIPPI environment.

## Basic HIPPI Framing Protocol

The HIPPI controller transmits bursts of data using the protocol illustrated in Figure 2-2 on page 2-5. Every time a connection is established, one or more packets are sent across the HIPPI channel. Each packet contains one or more bursts. Bursts contain the actual data and are 1 word wide (32 bits of data, 4 bits of parity) and 256 words long. To accommodate packets that are not exact multiples of 256, the ANSI standard states that either the first burst or the last may be shorter than 256 words. To implement the standard, the Paragon HIPPI controller sends the short burst last. Refer to Chapter 4 for more information about how the controller transmits data.

## Data Flow

This section describes the data flow in a HIPPI controller (refer to Figure 2-3 on page 2-6).

The HIPPI controller transfers data using direct memory access (DMA). DMA transfers *from* memory are called DMA reads and DMA transfers *to* memory are called DMA writes. DMA transfers occur in a maximum of 4K-byte blocks. Transfers occur when there is room for a block in the source FIFO or when there is a block available in the destination FIFO. Block transfers alternate between channels if both have requests.

Parity is written to the source FIFO during DMA transfers. The HIPPI-PH standard specifies odd parity on outgoing data, so the node bus parity is inverted before writing into the FIFO. Parity is checked by and passed through the S2020 interface circuit.

The controller communicates using DMA with GP node memory at full speed (i.e., zero wait states for sequential accesses). This is a peak bandwidth of 400M bytes/sec and does not include the initial transfer latency, memory refresh cycles, DRAM page boundaries, or cache-coherency cycles.

There are 512 transfers made per block (4K bytes), so with overhead, the block takes about 12 micro-seconds. Since a HIPPI burst takes 10.3 ms (258 clocks at 25 MHz), each channel uses about 29% of the GP node bus bandwidth with equally fast devices at the other ends.

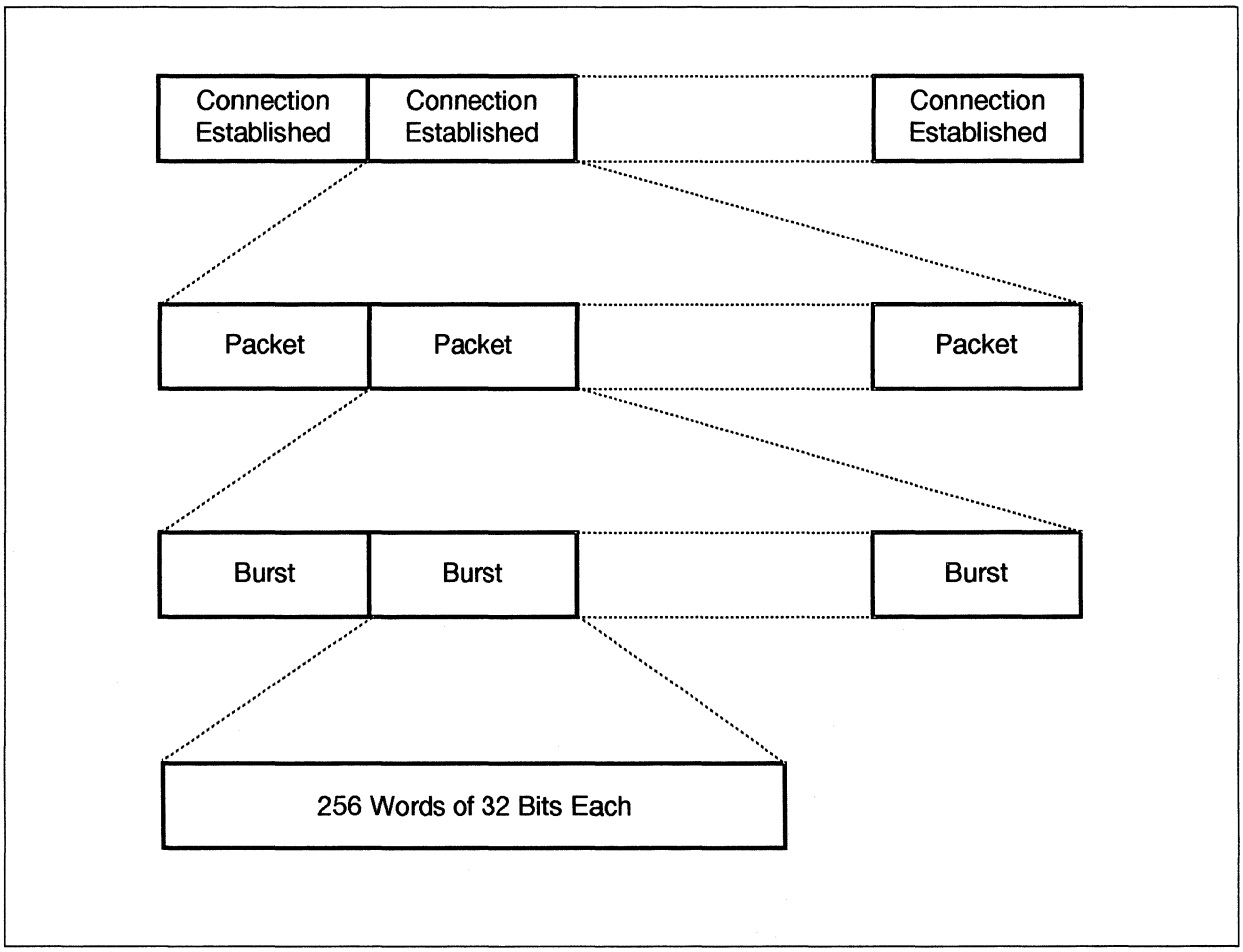


**Figure 2-2. HIPPI Framing Protocol**

The full 100 MB/s data rate is a theoretical maximum, measured to and from the GP node memory. There are several factors that decrease the data rate. First, due to inter-burst overhead, the source channel data rate is actually 98.8 MB/s, and the destination is actually 99.2 MB/s. Second, the remote device may be significantly slower than the controller, and HIPPI flow control will throttle the transfers down to the rate of the remote device. Finally, system throughput is limited by inter-node transfers, both by the mesh data rate and the GP node bus bandwidth required to move data into memory.

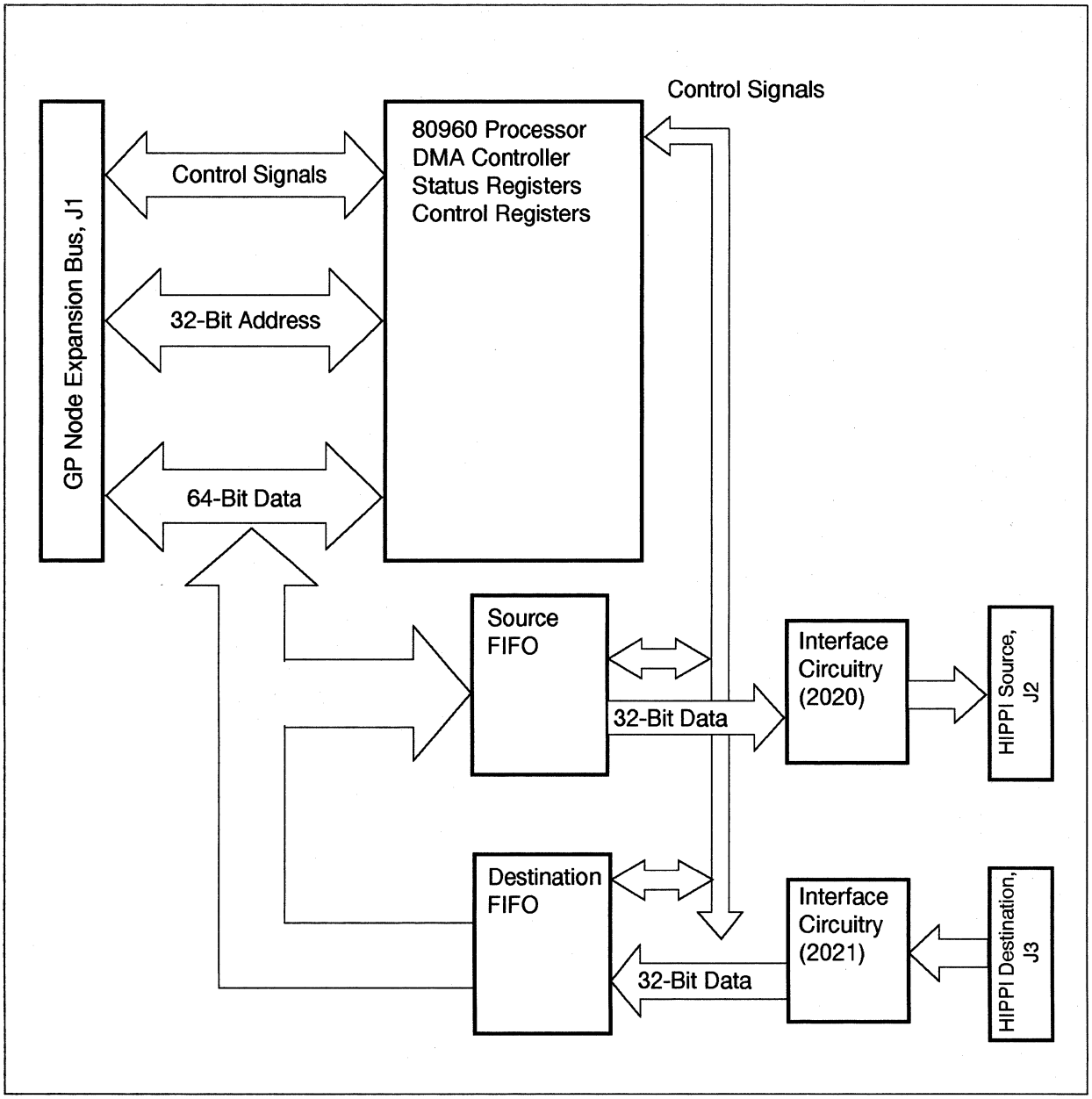


**Figure 2-3. Data Flow in a HIPPI Controller**

## Sending a Packet to Its Destination

This section describes how data is transmitted across a simple HIPPI channel. Figure 2-4 shows which signals are active during a simple HIPPI transmission.

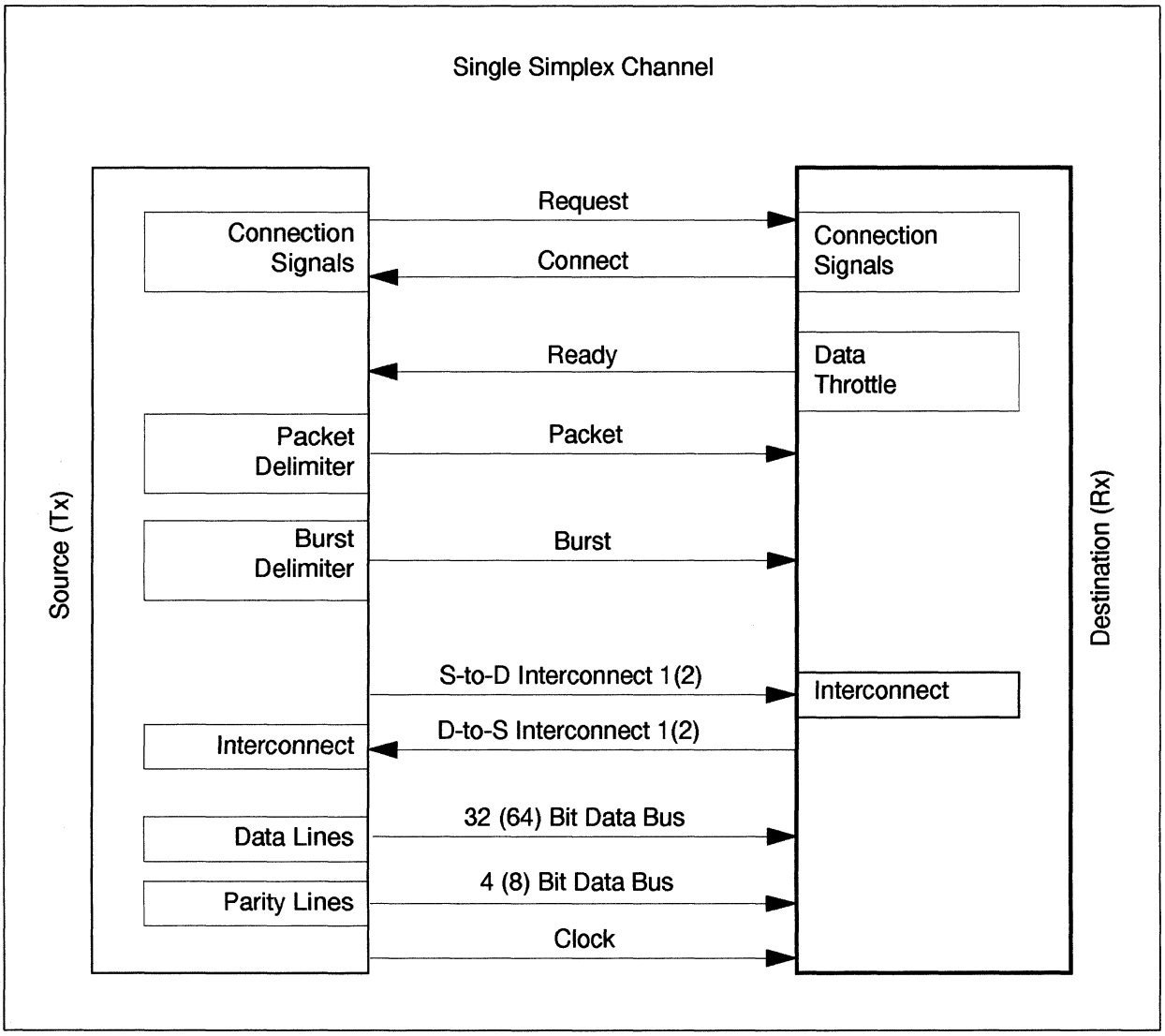


**Figure 2-4. The HIPPI Signal Diagram**

An important part of HIPPI transmission is the I-field. The I-field consists of a 32-bit field that contains connection routing information. The I-field precedes each HIPPI connection. Routing tables convert the network addresses to I-fields. This conversion process allows the HIPPI protocol to send and receive data between a wide variety of systems and peripherals.

The I-field is bounded by two events: the REQUEST signal being set to true and the CONNECT signal being set to true. At the highest level, a connection is bounded by both the REQUEST and the CONNECT signals being set to true. At the next highest level, the packet (which contains bursts of data) is bounded by the packet signal being set to true. This means that all bursts of data inside a packet occur between the time when the packet signal is set to true and the time when it is set to false. At the lowest level, each burst of data is bounded by the burst signal being set to true.

The I-field is described in more detail in the following section.

## The I-Field

The I-field is a 32-bit field sent as part of the sequence of the physical layer operations when establishing a connection between a HIPPI source or destination. The I-field contains connection routing information. Routing tables convert network addresses to I-fields, thus allowing the HIPPI protocol to communicate with many different systems. (Refer to "Routing Tables" on page 4-3 for more information.)

Figure 2-5 on page 2-9 shows the I-field format and is followed by a description of the I-field bits. Refer to the HIPPI-SC specification for more information about the I-field.

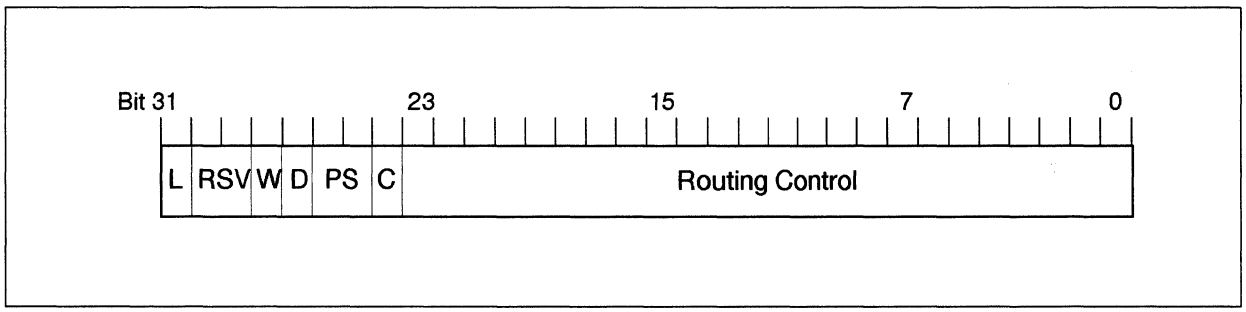


**Figure 2-5. I-field Format**

L — Locally administered (bit 31). If L=0, the I-field is defined by the HIPPI-SC standard. If L=1, the entire I-field will fall under the user's local administration, and the HIPPI-SC standard will not apply.

RSV — Reserved (bits 30, 29). The reserved bits are transmitted as zeros.

W — Double-wide (bit 28). If W=0, the Source is using the 800M bits/sec date rate option. The W bit is used in conjunction with the INTERCONNECT signals on Cable A and Cable B. The INTERCONNECT signals tell a switch or end point that the cable is physically attached to an active HIPPI port. The W bit is used to tell the switch or Destination end point whether or not Cable B is being used in a particular connection.

D — Direction (bit 27). If D=0, the right-hand end of the routing control field is the current sub-field. The right-hand end contains the least significant bits. See Figure 2-5. If D=1, the left-hand end of the routing control field is the current sub-field. The left-hand end contains the most significant bits. When a reverse path exists, a destination end point may return a reply to a received packet by using the same I-field with the D bit complemented.

PS — Path Selection (bits 26, 25). These bits are used as follows:

- 00 = Source routing: A specific route through the switches, with output port numbers specified for each switch.
- 01 = Destination address: Switches select the first route from a list of possible routes.
- 11 = Destination address: Switches select a route.
- 10 = Reserved

C — Camp-on (bit 24): If C=0, the switch replies with a connection reject sequence if the switch is unable to complete the connection. If C=1, a switch will attempt to establish a connection until either the connection is completed or the source aborts the connection request.

## I-Field Source Addressing

When the I-field PS bits are set to 00, the routing control field is split into multiple sub-fields. The size of the sub-fields depends on the size of the switch using it. The number of bits in the sub-field equals $\log_2 N$, where $N$ is the switch size. A 16 x 16 HIPPI switch would use a 4-bit sub-field (Figure 2-6).

When the I-field D bit is set to 0, the switch uses the current sub-field (right-most bits of the routing control field) to select the switch output port. The switch right-shifts (end off) the routing control field by the number of bits in the sub-field and inserts the switch port number in the left-most bits of the routing control field (Figure 2-6).

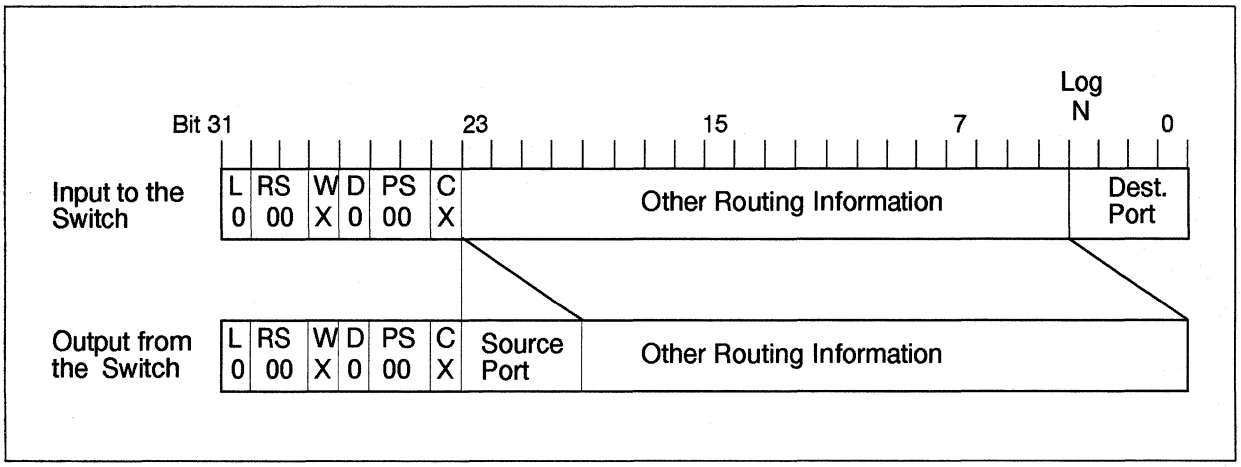


**Figure 2-6. I-field with Source Routing, D = 0**

When the I-filed D bit is set to 1, the current sub-fields are at the left-end of the routing control field. The routing control field is shifted left, and the port number is inserted at the right end of the routing control field (Figure 2-7 on page 2-11).

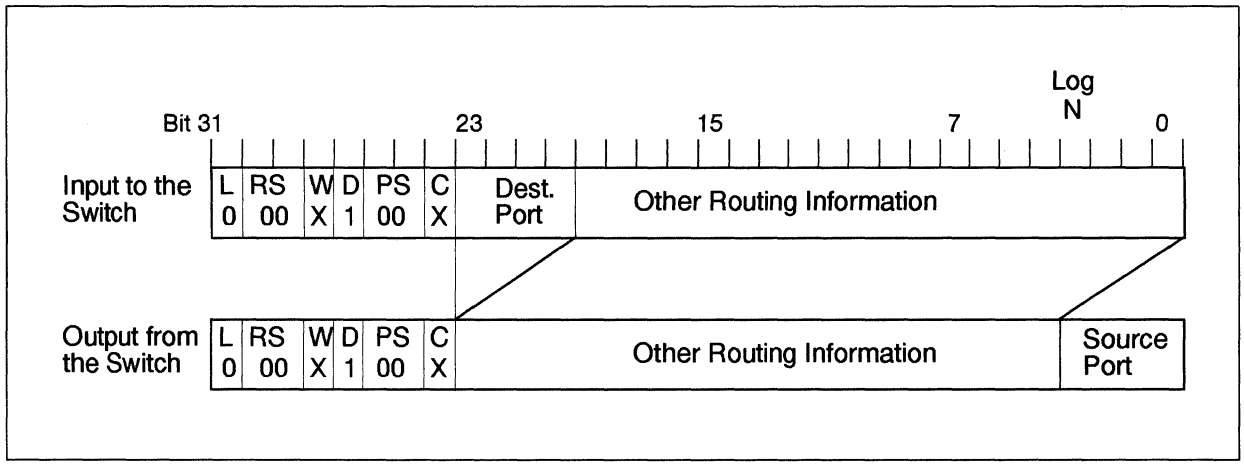


**Figure 2-7. I-Field with Source Routing, D = 1**

## I-Field Destination Addressing

When the I-field PS bits are set to 01 or 11, the routing control field is split into two 12-bit fields: one 12-bit field specifies the address of the destination end point and the other specifies the address of the source end point. When the D bit (direction) is set to 0, the right-hand 12 bits specify the destination address, and the left-hand 12 bits specify the source address (Figure 2-8). When the D bit is set to 1, the right hand 12 bits specify the source address, and the left-hand 12 bits specify the destination address (Figure 2-9).

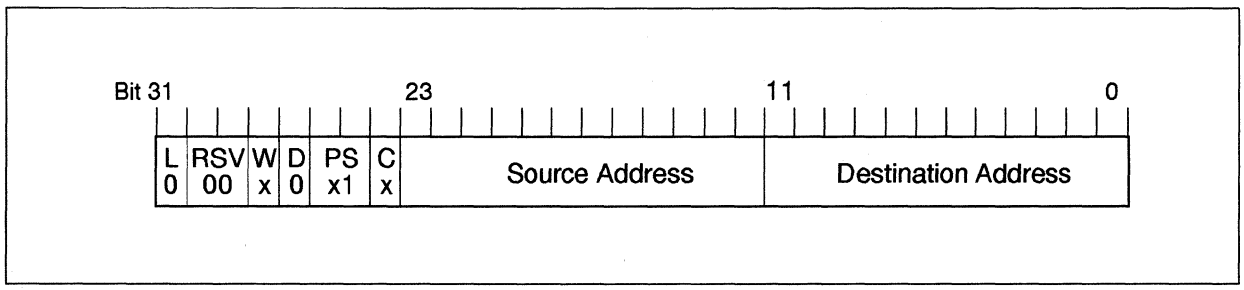


**Figure 2-8. I-Field with Destination Address, D = 0**

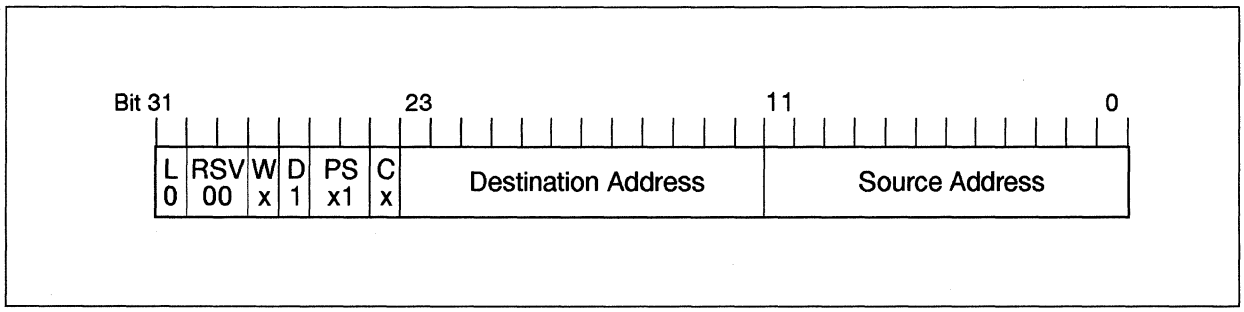


**Figure 2-9. I-Field with Destination Address, D = 1**

## The Use of Switches In the HIPPI Environment

A HIPPI switch is a physical device that accepts input from one port and, after interrogating the I-field, passes all control and data to the receiving port. A HIPPI switch may be compared to an electrical switch in that, once the connection is made between an input port and an output port, it is as if a solid wire were connecting the two ports. The Camp-on bit in the I-field provides control of the switch. If the Camp-on bit is set, the switches are instructed to attempt a connection until the connection is completed or the source cancels the connection request.

A HIPPI switch is typically identified by the number of input and output ports in the form "A x B." The "A" refers to the number of input ports and "B" to the number of output ports. Thus, an 8 x 8 switch consists of eight input ports and eight output ports

In a practical HIPPI application, a switch may provide switching capability for numerous ports. Figure 2-10 shows an 8 x 8 HIPPI switch. In Figure 2-10, these input and output ports are labelled A1 through A8 and B1 through B8. The switch in Figure 2-10 could be used to control the connection between any A port to any B port, thus allowing numerous connections.

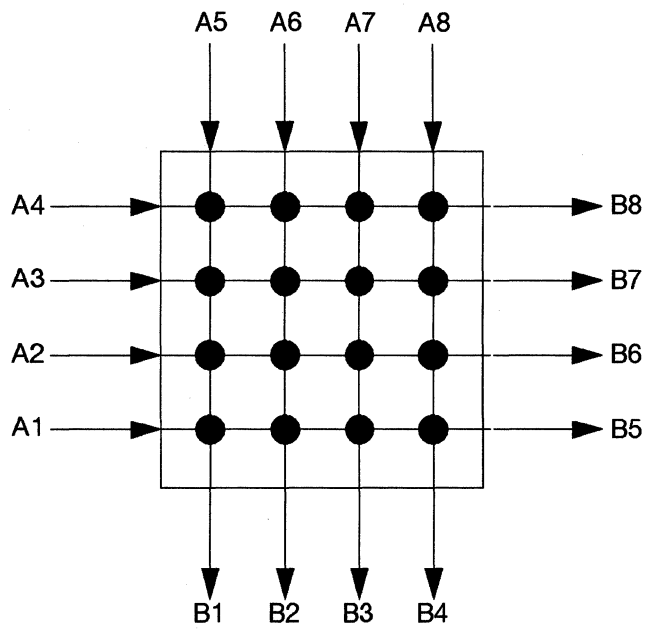


**Figure 2-10. An 8 x 8 HIPPI Switch**

# Installing the HIPPI Controller 3

This chapter explains how to install the HIPPI controller in a Paragon system, how to cable the controller to the system, and how to run cables out to an external system or peripheral device.

## Tools Needed

You need the following tools:

- Small Phillips screwdriver.
- Small flat-bladed screwdriver.
- Antistatic wrist strap.
- Antistatic surface on which to lay the boards while working on them (the antistatic bags used to ship the boards are satisfactory).

## The HIPPI Controller and the GP Node Board

As shown in Figure 2-1 on page 2-2, the system is shipped from the factory with the HIPPI controller already mounted on a GP node board. The controller is mounted on the expansion interface connector of the node board. When combined, the board and controller (called the HIPPI board) take up two slots in the cardcage.

The primary side of the HIPPI board (the side with most of the active components) faces away from the GP node board. The HIPPI controller mounts to six metal standoffs on the GP node with 2-52 screws, #2 washers, and lockwashers. The standoffs provide proper clearance for the expansion connector (J1).

The channel connectors on the HIPPI board are accessible through the HIPPI board's front panel. On the front panel, the channel connectors are labelled "SRC" for the source connector (J2) and "DST" for the destination connector (J3). The board contains a pair of LED's for each channel, one red and one green. Each pair is mounted at the upper end of its associated connector.

# Installing the HIPPI Controller

## CAUTION

You must shut the system down before installing the controller. Before shutting the system down, make sure that no applications are running on the system. For more information about how to shut the system down, refer to the *Paragon™ System Administrator's Guide.*

Follow these steps to install the controller:

1. Turn off the system's power.

2. Once power is off, open the cabinet door (Figure 3-1 on page 3-3). The door latch is located at the bottom right of the front door. Unlock the door by moving the latch 90° counterclockwise.

3. Attach the antistatic strap to your wrist and connect the other end of the strap to a solid ground.

## NOTE

The HIPPI controller/GP node assembly requires two cardcage slots. The following steps tell how to remove GP node boards from the cardcage. If you already have two adjacent empty slots in which to install the HIPPI controller, skip steps 4 through 8.

4. Loosen the capture screws at the top and bottom of the GP node's front panel (Figure 3-2 on page 3-4).

5. Remove the node board by grasping the nylon clips on the top and bottom of the board, and pull the clips toward you. This should dislodge the board.

6. Carefully slide the board out of the cardcage. Take care to handle the board only by its edges.

7. Put the GP node board in an antistatic bag and lay it aside.

8. Repeat steps 4 through 7 for the second GP node.

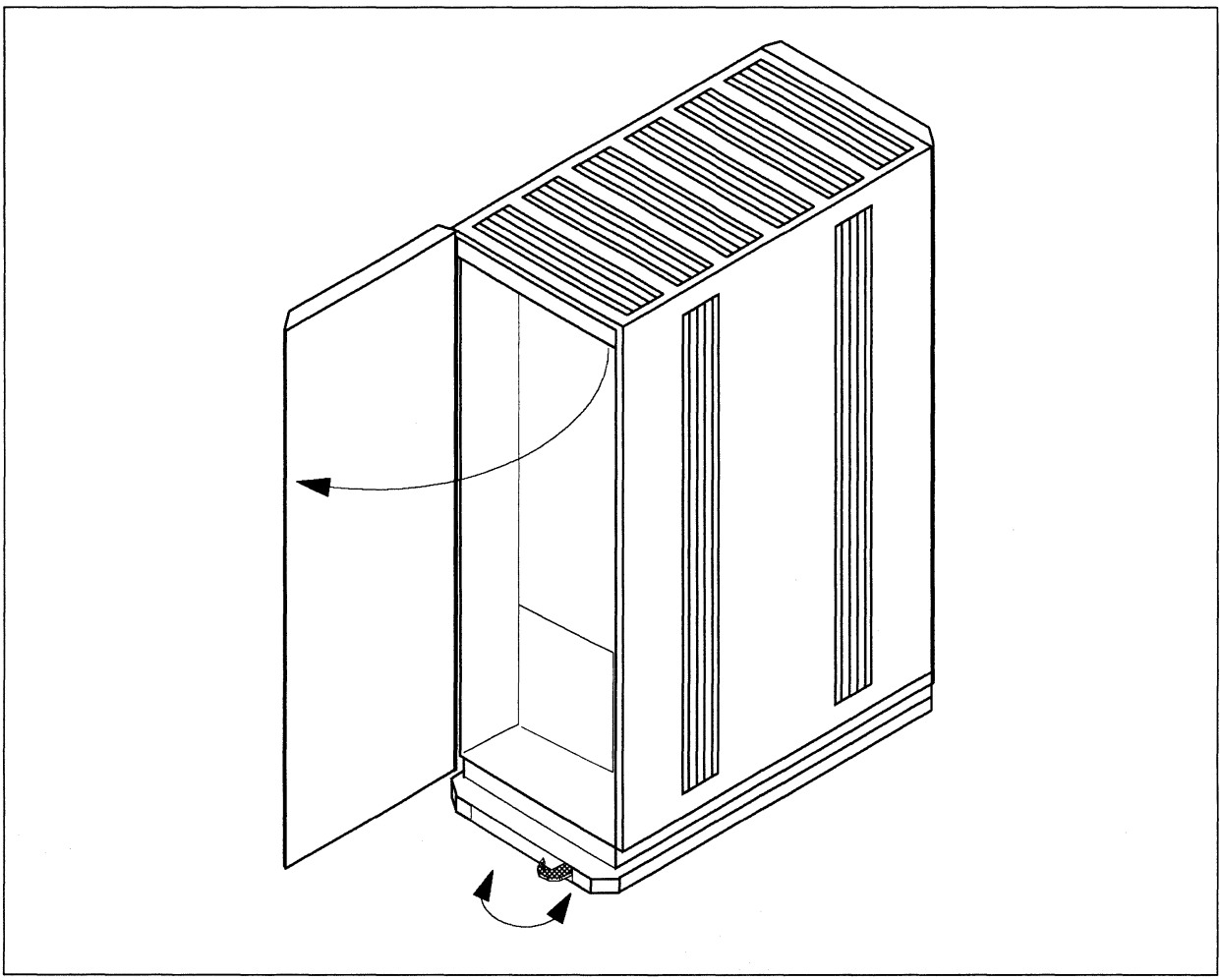

**Figure 3-1. Opening the Paragon™ Cabinet Door**

9. Remove the combination HIPPI controller/GP node from its antistatic bag. Handle the board only by its edges.

10. Align the board edges with the cardguide rails. Slide the board into the cardcage until the board connectors meet the backplane (Figure 3-3 on page 3-5).

11. Align the node board connector with the backplane and press firmly on the board until the board connector mates with the backplane.

12. When the board is properly seated, the nylon clips straighten out and the board front panel is flush with the frame of the cabinet.

13. Tighten the capture screws at the top and bottom of the board front panel.

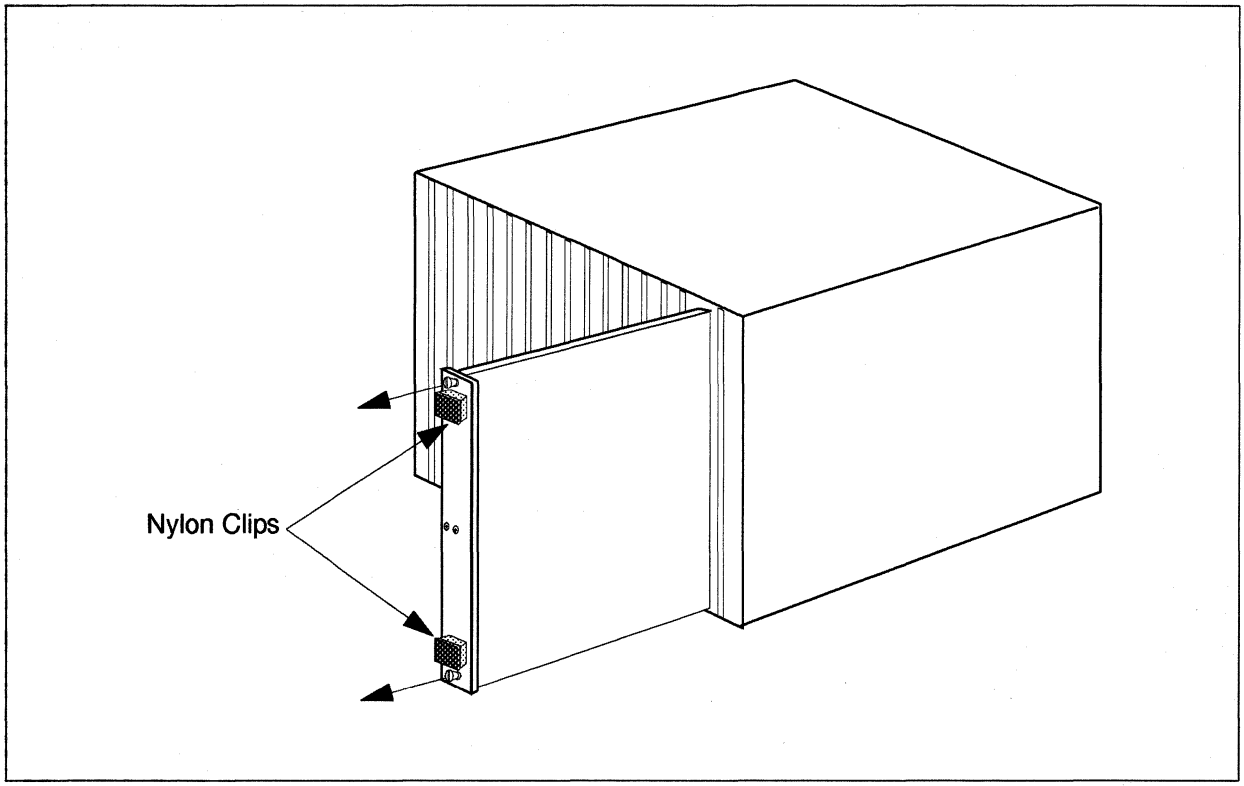


**Figure 3-2. Removing a GP Node Board from the Cardcage**

# Installing the Cables

*For each HIPPI controller* in the system, there are two identical internal cables that are used to connect the controller to the system I/O panel and two identical external cables that are used to connect the I/O panel to an external device. *For each system*, there is a single loopback cable that is used to perform loopback diagnostic tests. This cable connects to the ends of the source and destination channel connectors on the HIPPI controller to allow the user to verify loopback transfers.

Intel SSD supplies all these cables for your system. Refer to Chapter 6, "Cable Parts and Specifications," for more information about the cables.

The following sections tell how to install the internal and external cables. Refer to Chapter 5, "HIPPI Diagnostics" for more information about using the loopback diagnostic cable.

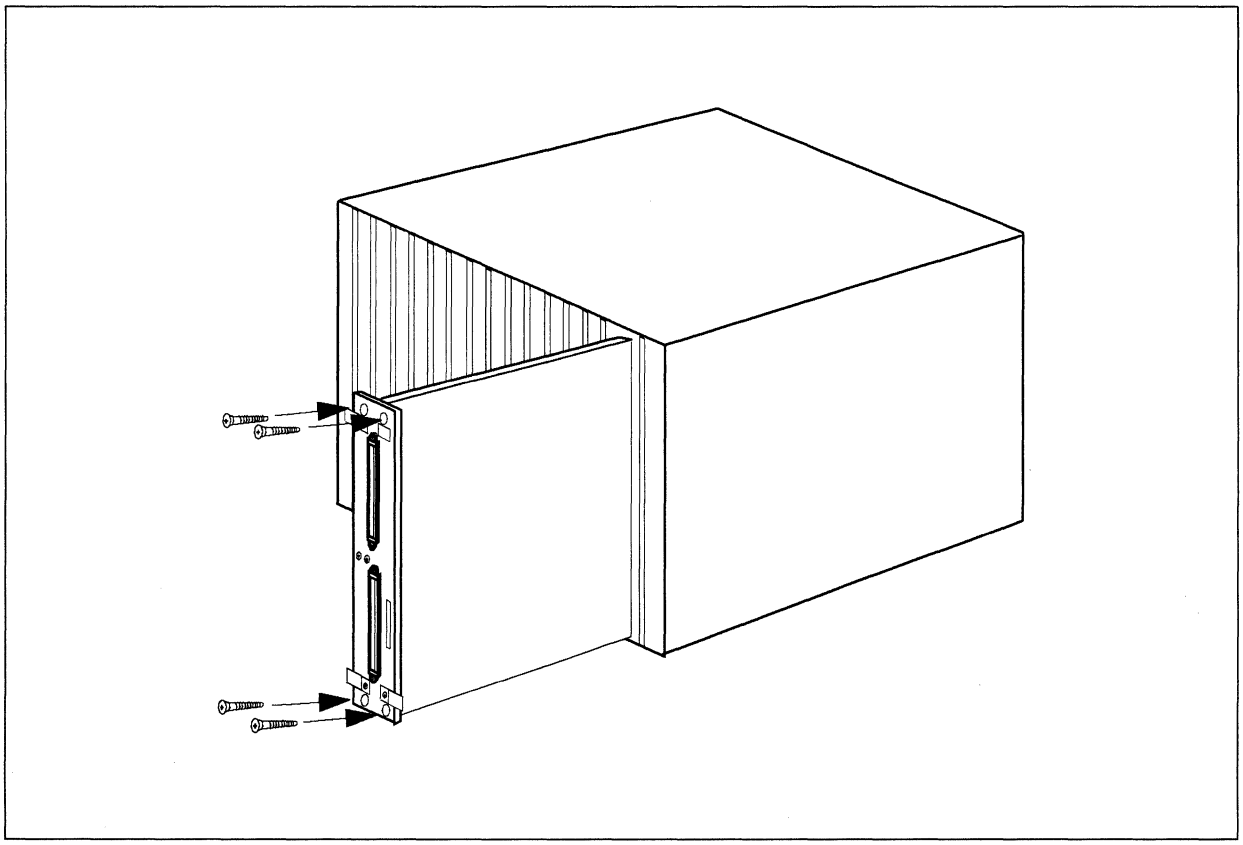

**Figure 3-3. Installing the Combined HIPPI/Node Board in the Cardcage**

## Installing Internal Cables

Each HIPPI controller comes with two 6.5-foot, 100-pin "internal" cables that are used to connect the controller to the I/O panel. One cable attaches to the Source Channel Connector and the other attaches to the Destination Channel Connector. See Figure 2-1 on page 2-2 and Figure 3-4 on page 3-7. To connect the cables:

1. Connect one end of each cable to the HIPPI controller GP node by aligning the pins to the connectors. Push the cables on to the connectors (which are pressure sensitive).

2. Allowing the cables to hang loosely inside the cabinet, connect the other end of each cable to the top of the I/O bulkhead. You can use any of the following pairs of slots in the bulkhead: (1A, 2A), (3A, 4A), (5A, 6A), or (7A, 8A).

## Installing External Cables

Each HIPPI controller comes with two 25-meter "external" cables that are used to connect the I/O panel to an external system, switch or peripheral device. See Figure 3-4 on page 3-7. To connect the cables:

1. Connect one end of each cable to the connectors on the bottom of the bulkhead.

2. Connect the other end of each cable to the external system, switch, or peripheral device.

## Closing the Cabinet Door

Close and latch the front door to the Paragon cabinet. Lock the door by moving the latch 90° clockwise. See Figure 3-1 on page 3-3.

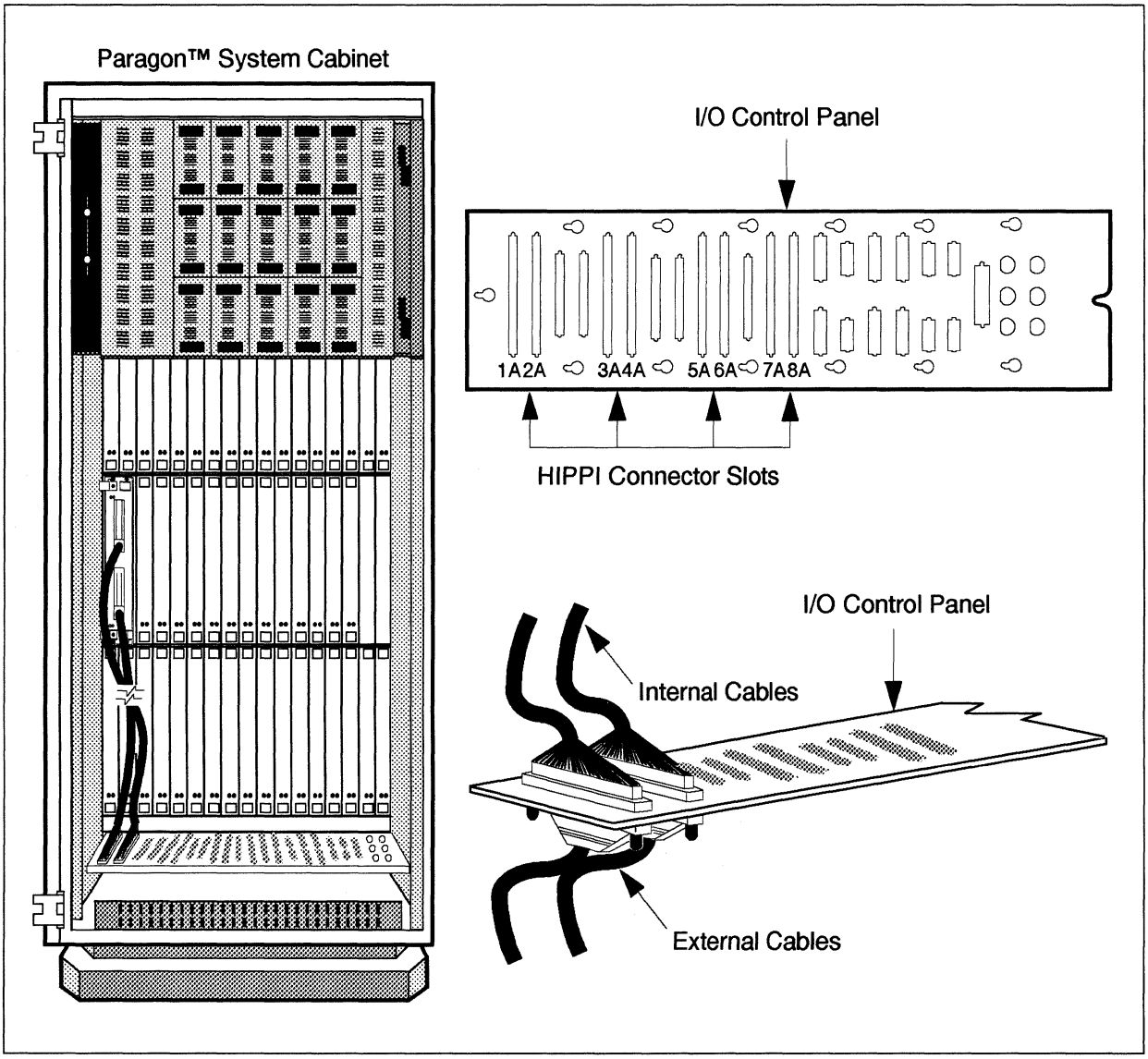


**Figure 3-4. Cabling the HIPPI Controller**

# Software Configuration 4

This chapter discusses the following software configuration issues:

- Configuring a network interface.
- Activating the interface.
- Defining and installing routing tables.
- Building data packets.
- Raw HIPPI.

## Network Configuration

The device driver for the HIPPI controller serves as a bridge between the Mach microkernel and the HIPPI controller card. The device driver is integral to the microkernel, but you must configure the network interface for your own system.

## Configuring the Network Interface

To configure the network interface, you must know the following values:

| | |
|---|---|
| I-field settings | The I-field is a 32-bit control field defined by the HIPPI-SC specification. For more information, refer to "The I-Field" on page 2-8. |
| IP address | Each HIPPI board in a system must be assigned its own unique IP address. Your system administrator can provide the IP address. |
| ULA (IEEE) address | The ULA (Upper Layer Address) for each HIPPI board is unique and is written into the controller's flash memory when the controller is manufactured. Your system administrator can provide the ULA. The ULA has the following form: `1:0c:34:65:0:26` |

The following sections explain how you use these values.

## Activating the Network Interface

You must do two things to activate the network interface: update the *DEVCONF.TXT* file and issue an **ifconfig** command.

The *DEVCONF.TXT* file tells the system where the HIPPI board is located (on reset, the system uses the information in *DEVCONF.TXT* to update file *SYSCONFIG.TXT*). The entry in *DEVCONF.TXT* should look like this:

```
S9 GPNODE N04 16 MRC 04 HIPPI 00A09 H04
```

Refer to the *Paragon™ System Administrator's Guide* for more information about files *DEVCONF.TXT* and *SYSCONFIG.TXT*.

The **ifconfig** command assigns an Internet address and activates the interface. For example:

```
# /sbin/ifconfig \<hippinode, servernode\> ifhip0 192.9.2.5 up
```

In this example:

- *hippinode* is the number of the GP node on which your HIPPI controller is installed (using the root node numbering scheme).
- *servernode* is the node number of the network server (using the root-node numbering scheme). The network server is the node where the TCP protocol processing will be performed.
- *192.9.2* is the network number and *5* is the host number. Note that an entry for this IP address and a unique hostname must be in the */etc/hosts* file.

# Routing Tables

HIPPI routing tables provide the server with a mapping from the Internet address and the ULA address to the I-fields. Each table includes the IP address, the ULA, and the I-field. Each routing table is maintained in a file that this manual refers to as a *hippi.map*.

## Routing Tables for Simple Networks

The following example shows a *hippi.map* file for the simple, one-switch network shown in Figure 4-1 on page 4-4.

```
#HIPPI Network Routing Table
#
#IP Address       ULA                I_field
#-----------      --------------     ----------
 192.9.3.1        1:0c:34:65:0:26    0x1000000   #Jaguar
 192.9.3.2        1:0c:34:65:0:27    0x1000002   #Panther
 192.9.3.3        1:0c:34:65:0:28    0x1000001   #Cheetah
 192.9.3.4        1:0c:34:65:0:29    0x1000003   #Cougar
```

## Routing Tables for Complex Networks

In the simple network shown in Figure 4-1, a single routing table supports all hosts on the network, and all hosts go through the same switch to reach their destination. In a more complex network with multiple switches, there may be multiple routing tables, and maintaining them becomes a more complex job. Each host's routing table must define the interconnecting paths through the switching fabric. If there are failures with the switches, you must modify the routing tables to accommodate a new path.

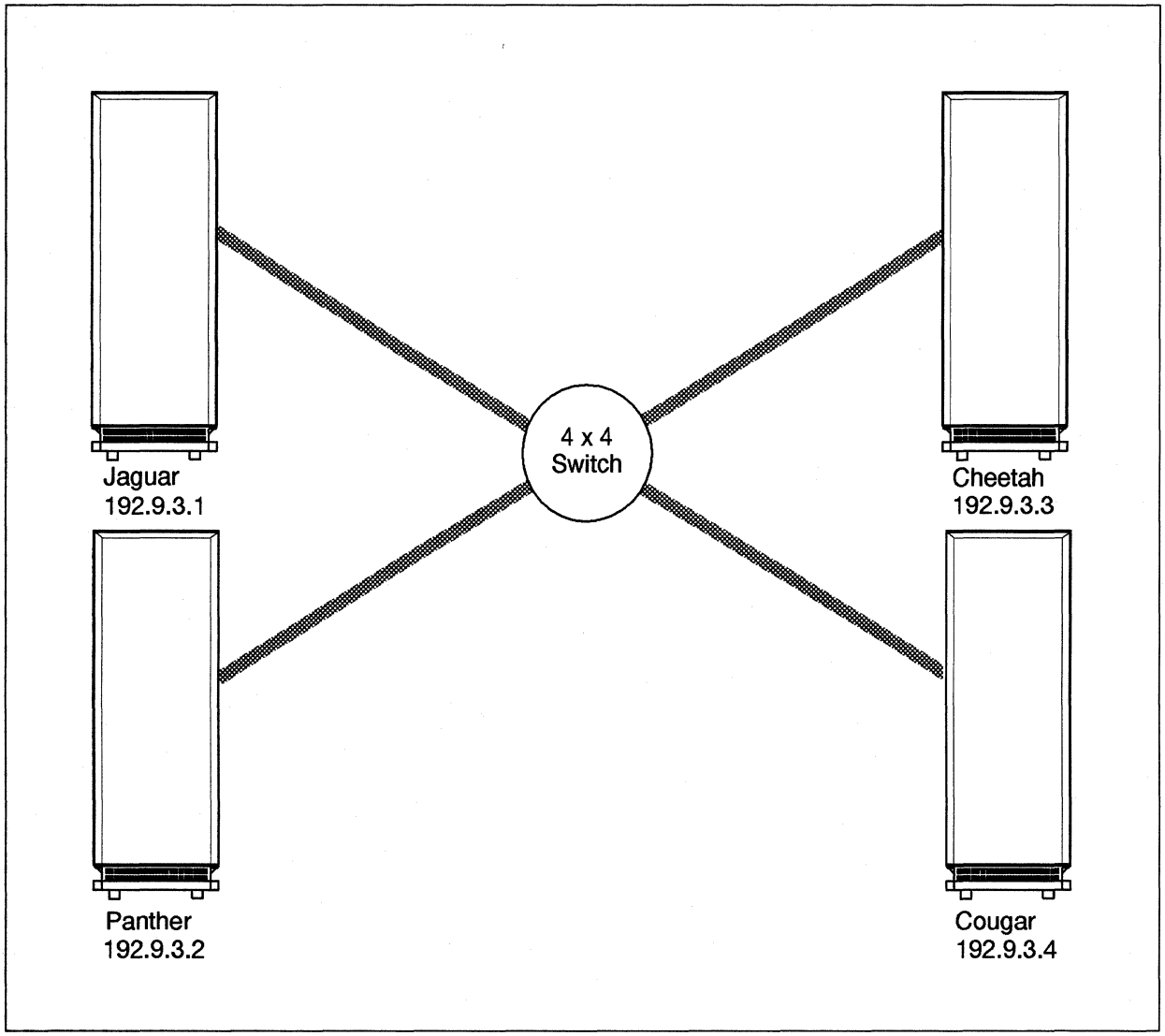


**Figure 4-1. Sample Network with One HIPPI Switch**

## Routing Table Commands

There are two routing table commands: **hippi_showmap** and **hippi_setmap**.

Use the **hippi_showmap** command to display the current routing table. For example:

```
# hippi_showmap
```

Use the **hippi_setmap** command to load a routing table (formatted like *hippi.map* above) into the device driver. For example:

```
# hippi_setmap ifhip0 hippi.map
```

### NOTE

Every time the **hippi_setmap** command is executed, it replaces the entire table.

Refer to Appendix A for more information about the **hippi_setmap** and **hippi_showmap** commands.

# Server Interface and Packet Building

The OSF server supports TCP/IP traffic over HIPPI as well as raw HIPPI frames via the raw HIPPI library, *libhippi.a*. Use of the HIPPI interface for TCP/IP is transparent to the user. As with other interfaces the TCP/IP protocol engine routes data over the HIPPI interface when the network address of the destination dictates.

The server formats each packet of information as specified in Figure 4-2 on page 4-6. Each packet consists of three areas: the FP_Header_Area, the D1_Area, and the D2_Area. The FP_Header_Area contains the Upper Level Protocol (ULP) identification which designates the destination ULP. This identifies where the packet of information is to be delivered. Both D1_Area and D2_Area are data areas. These packet areas are described in the following sections.

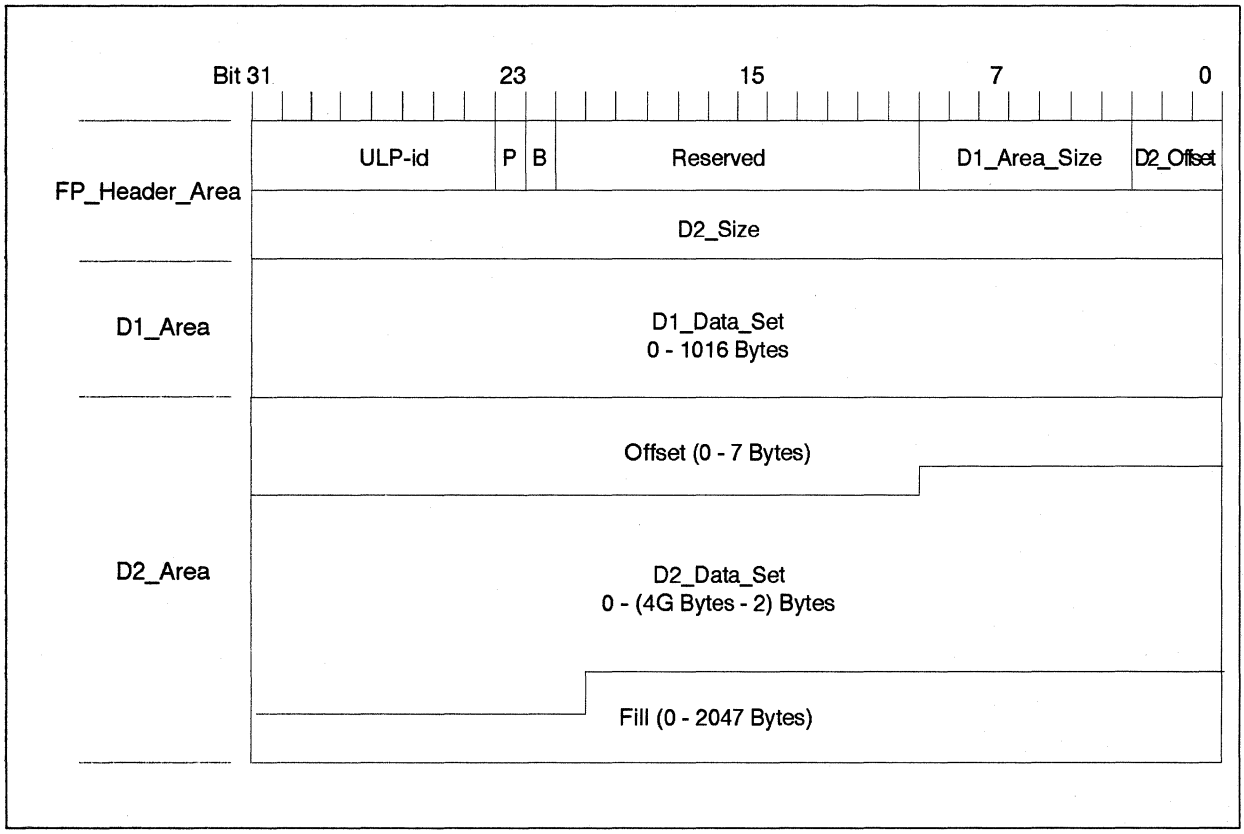


**Figure 4-2. HIPPI Packet Format**

## The FP_Header_Area

The FP_Header_Area comprises the ULP-id, which is eight bits in length. A value of one in the P bit indicates that a D1_Data_Set is present in this packet. A value of zero in the P bit indicates there is no D1_Data_Set present in this packet. A value of zero in the B bit indicates the D2_Area starts at or before the beginning of the second burst of the data packet. A value of one in the B bit indicates the D2_Area starts at the beginning of the second HIPPI-PH burst of the data packet. The D1_Area_Size designates the number of 64-bit words between the end of the 64-bit Header Area and the start of the D2_Area. The D2_Offset field contains the number of bytes in the Offset field in the D2_Area (i.e., the number of "junk" bytes at the beginning of the D2_Area). The D2_Size field contains the number of bytes in the D2_Area (including the Offset field); this number is aligned on an 8-byte boundary.

# The D1_Area

The D1_Area follows the FP_Header_Area. If the P bit of the Header Area is equal to zero, then the D1_Data_Set is not present and the contents of the D1_Area are ignored. The D1_Data_Set is the first information in the D1_Area. This area contains control information that may be delivered to the Destination Upper Level Protocol on receipt, without waiting for the arrival of other bursts of the packet. The maximum size of the D1_Data_Set is 127 64-bit words.

## The LE_Header

The Link Encapsulation (LE) protocol specification is the currently supported protocol for detailing the header information passed to the D1_Area. The LE protocol envelopes a 802.2 LLC packet for transmission over HIPPI Framing Protocol. The Destination Hub Address, Destination Port, Source Hub Address and Source Port fields in the LE_Header identify the physical address of the hub which this packet has started from or is destined to. Figure 4-3 shows the LE Packet Format.

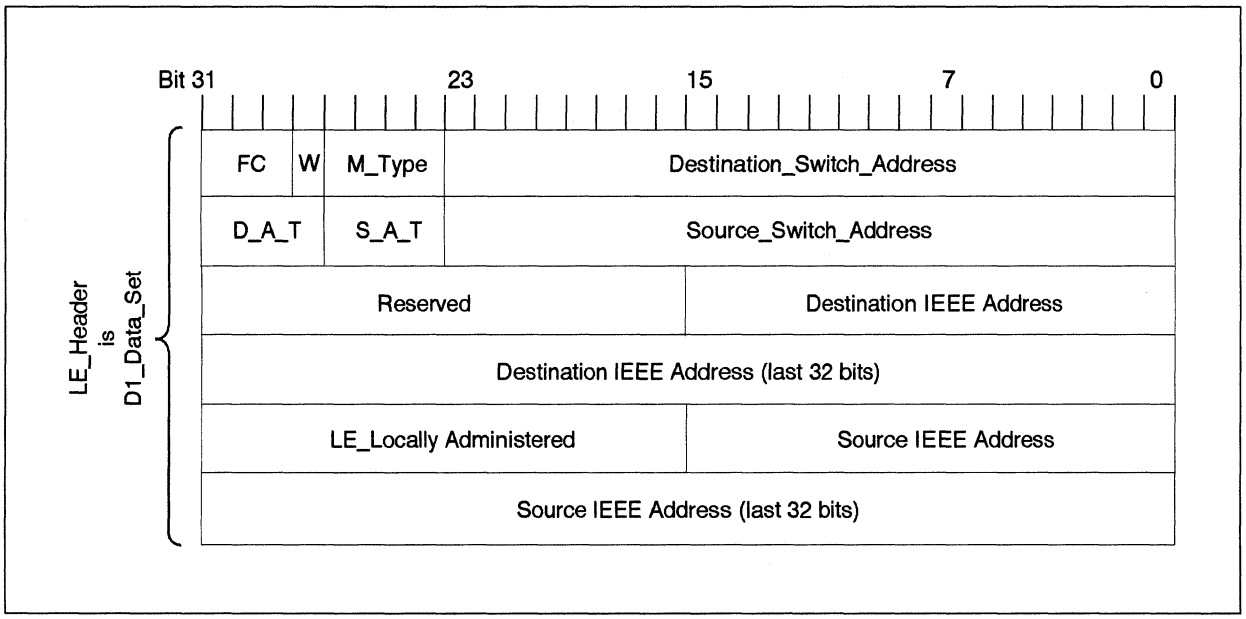


**Figure 4-3. Link Encapsulation Packet Format Header**

Figure 4-4 on page 4-8 depicts a HIPPI packet. The figure shows the Framing Protocol and the Link Encapsulation Packet Format together. The LE_Header is placed in the D1_Area of the HIPPI FP Packet.

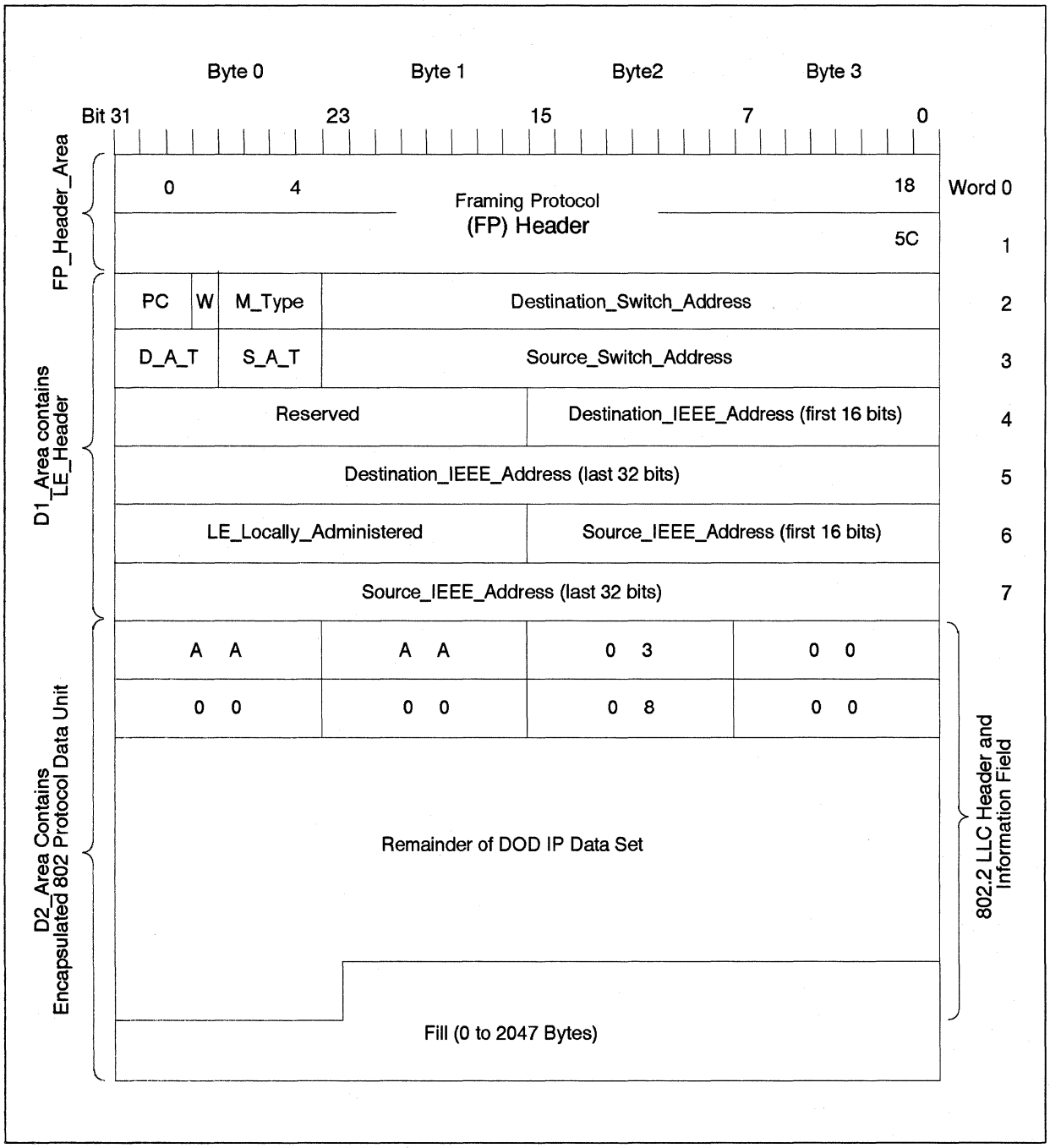


**Figure 4-4. A HIPPI Framing Protocol Packet with an LE_Header**

## The D2_Area

If the D2_Size is not zero in the Header Area, D2_Area will then immediately follow the D1_Area and will start on a 64-bit boundary and will contain the D2_Data_Set. If the B bit in the Header Area equal one, then the D2_Area will start at the beginning of the second HIPPI burst. The Offset is the unused bytes from the start of the D2_Area to the first byte of the D2-Data_Set.

The D2_Data_Set can range in size from zero to an indeterminate number of bytes. The Fill part of the D2_Area is the unused bytes between the end of the D2_Data_Set and the end of the D2_Area, for example, the end of the packet. If a D2_Size of all binary ones is used, then there is no Fill in the D2_Area.

## Inbound Packets

When an incoming HIPPI packet is received, the device driver sends it to the destination channel using standard microkernel network code. The HIPPI packets are filtered using the raw HIPPI Upper Layer Protocol (ULP) and (optionally) a port number that must appear in the first word of D1. If the filter reads an LE_Header in the packet, then that packet is sent to the TCP-IP server. See Figure 4-5.

# Raw HIPPI

The raw HIPPI interface is implemented as a user library (*libhippi.a*) that sits on top of the Mach device interface. The library consists of routines that allow you to define HIPPI packet formats and to open, close, write, and read HIPPI channels. To use *libhippi.a*, your code must contain the include file *raw_hippi.h*. For detailed information about the *libhippi.a* functions, refer to the manual pages in Appendix A.

## Raw HIPPI Usage Models

The raw HIPPI interface supports two usage models, called HIPPI_RAW and HIPPI_DATA. You select a model when you call the **hippi_open** function (see Appendix A). Whichever model you use, the library maintains state information to ensure that each open HIPPI connection functions correctly.

The HIPPI_RAW model assumes you are a sophisticated user of HIPPI and therefore *does not* provide HIPPI frame formatting. Instead, you responsible for formatting the HIPPI frame, allocating and deallocating memory using the functions in *libhippi.a*, and rounding word sizes (per the HIPPI-FP specification).

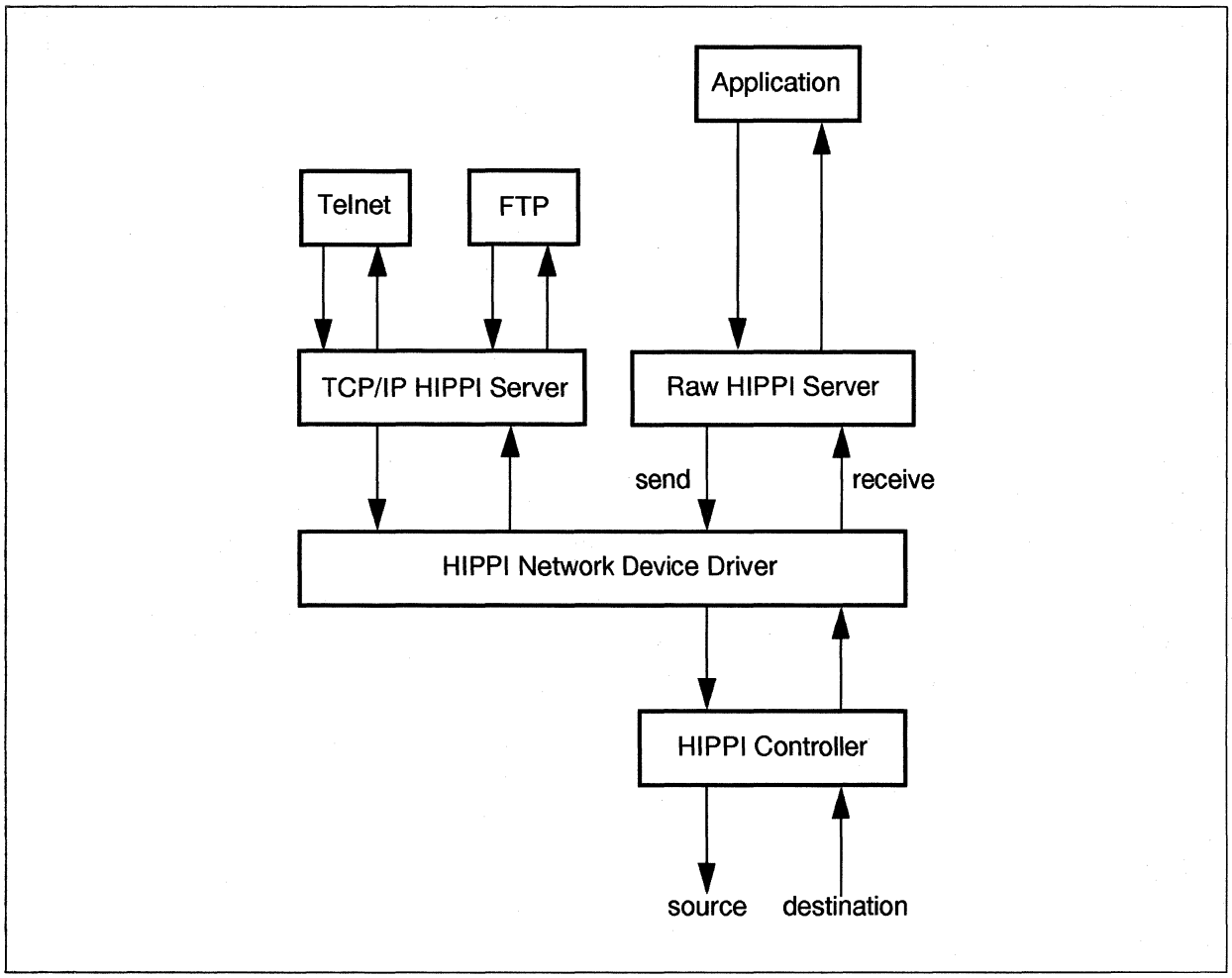


**Figure 4-5. HIPPI Packet Incoming and Outgoing Flow**

The HIPPI_DATA model assumes you are not a sophisticated user of HIPPI and therefore *does* provide HIPPI frame formatting. You are also responsible in this model (as in HIPPI_RAW), for allocating and deallocating memory using the *libhippi.a* functions.When you use these memory functions in the HIPPI_DATA model, memory is allocated in such a way that the library can add HIPPI headers to frames without copying data and the driver is able to process frames efficiently.

## Using Raw HIPPI

Before you can use a raw HIPPI channel, several events must occur:

1. Your system administrator must use the **rmknod** command (the remote version of **mknod**) to make a special file for each HIPPI channel on your system. Normally this is done just after system installation and need not be done again (unless the HIPPI board is changed). As an example, the following command creates a HIPPI channel (called hippi.one) having major device number 24, minor device number 0, and node number 5:

   ```
   # rmknod /dev/hippi.one c 24 0 5
   ```

2. Your system administrator must reboot the system. Booting the system automatically invokes the **set_hippi_buffers** command, which configures the buffers for each HIPPI channel. The system administrator can also issue the **set_hippi_buffers** command at other times. As an example, the following command configures buffers for HIPPI channels according to the information in file *myfile*:

   ```
   # set_hippi_buffers myfile
   ```

   Each entry in file *myfile* lists a channel name, the size of each buffer, and the number of buffers. For example:

   ```
   /dev/hippi.one 1048576 6
   /dev/hippi.two 2097152 3
   ```

3. You must open the HIPPI channel using the **hippi_open** command. For example, the following command opens HIPPI channel hippi.one in read-write mode:

   ```
   # hippi_open (/dev/hippi.one, HIPPI_RAW, O_RDWR)
   ```

# HIPPI Diagnostics 5

This chapter introduces the HIPPI controller diagnostic tests and explains how to install the diagnostic loopback cable.

## Diagnostics

The HIPPI controller diagnostic tests consist of a power-up test and several loopback tests. The loopback tests are HIPPI diagnostic tests that only function if a loopback cable is installed on the HIPPI controller.

### Power Up Test

The power up test is an extension of the Paragon System Node Confidence Test (NCT). If the power-up test passes, the system assumes that the HIPPI controller functions correctly. This test is not intended to provide detailed fault isolation beyond the Field Replaceable Unit (FRU) level. The HIPPI power up tests returns a pass/fail indication to the main node confidence test.

For more detailed information about HIPPI power-up diagnostics, see the *Paragon™ Diagnostic Reference Manual.*

### Loopback Tests

Some of the HIPPI diagnostic tests require a loopback cable. If the loopback isn't installed, these tests return PASS. If you install a loopback cable, the tests transfer data through the cable. The following tests require a loopback cable:

- testHippiReadyCntr
- testHippiConnect

- testHippiReject
- testHippiAutoCon
- testHippiBROUT
- testHippiParity
- testHippiLoopback

To test the HIPPI switch and the routing table using the Loopback Test, make sure that the host's address is contained in the routing table. The HIPPI packets will go out the HIPPI switch and back.

The diagnostic cable connects the HIPPI controller's source and destination channel connectors together. The diagnostic cable mates with the external cable connector (Figure 5-1).

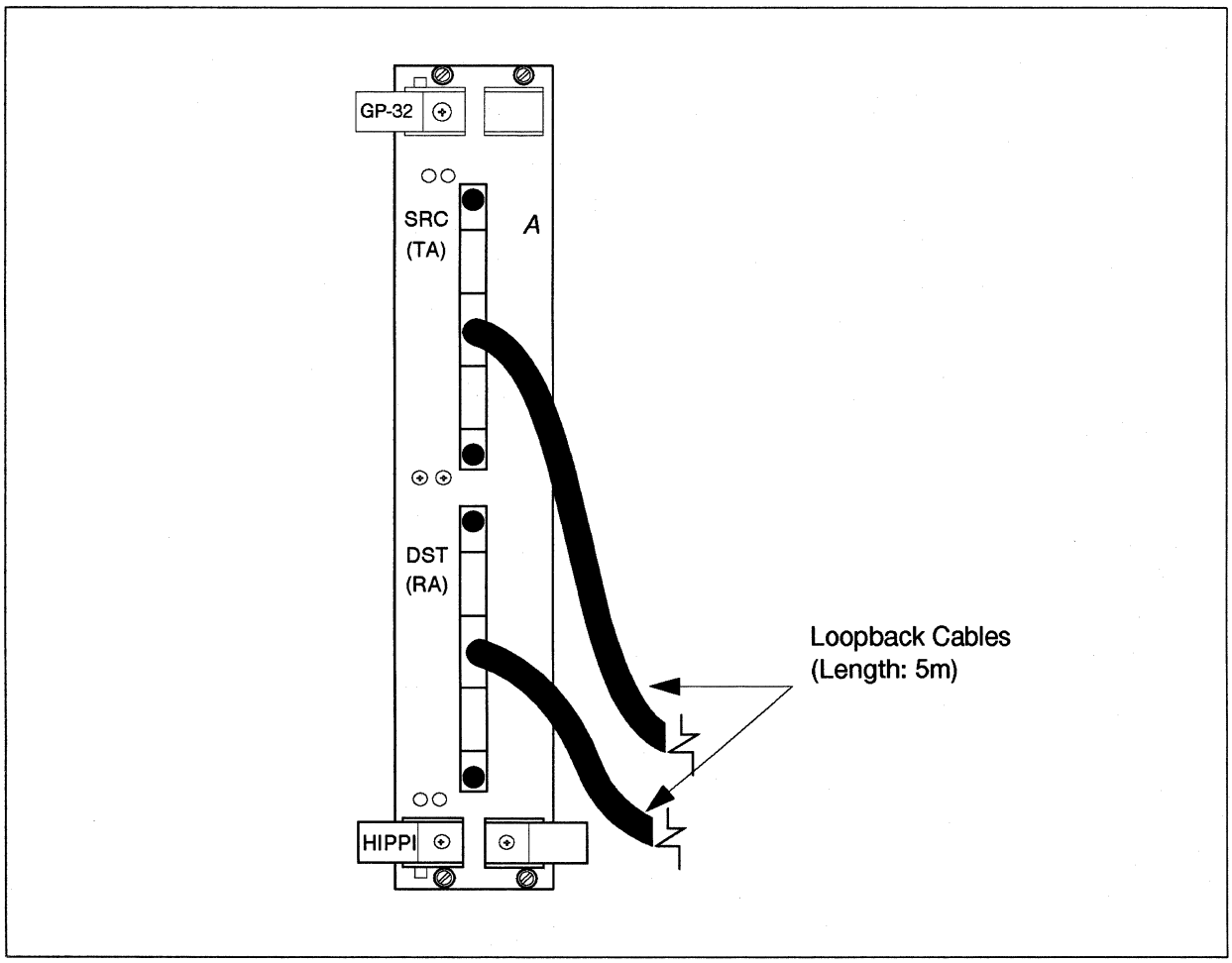


**Figure 5-1. Connecting the Loopback Cables**

# Cable Parts and Specifications 6

This chapter contains the specifications for the internal, external, and loopback cables that are used to connect the HIPPI controller to external devices and to test the controller's operation.

## Internal Cables

The two internal cables for each HIPPI controller run from the front panel of the controller to the system I/O panel. One cable is for the source channel, one for the destination channel. See Figure 3-4 on page 3-7.

### Internal Cable Characteristics

The ends of each internal cable are labelled P1 and P2. The P1 end is a right-angle, 100-pin, plug connector that has a metal housing and latching retainer hardware; it mates with J2 and J3 on the HIPPI controller board.

The P2 end of the internal cable is a standard HIPPI panel mount connector; it receives the standard *external* HIPPI cable connector (including retaining screws). The standard HIPPI female screw locks are also located on the P2 end of the cable.

Each internal cable is 6.5 feet long.

### Internal Cable Implementation

The internal cables are Madison Cable DOSDK00010 or equivalent. The connectors used on this cable must be the functional equivalent of:

- P1 AMP 749611-8 or 749621-9, with latching backshell AMP 74919-1 or 749206-1.
- P2 AMP 749877-9 with female screw locks AMP 749087-1.

# External Cables

The two external cables for each HIPPI controller run from the I/O panel to an external system, switch, or device. One cable is for the source channel, one for the destination channel. See Figure 3-4 on page 3-7.

## External Cable Characteristics

The external cables must meet the requirements specified in the ANSI X3T9.3/88-023 specification for connectors, pin assignments, shielding, wire color code, and electrical behavior.

Each external cable is 25 meters long.

## External Cable Implementation

The external cables must be the functional equivalent of:

- 1 meter cable - AMP 749755-13.
- 5 meter cable - AMP 749755-2.
- 15 meter cable - AMP 749755-3.
- 25 meter cable - AMP 749755-4.

# Loopback Cable

The loopback cable connects the HIPPI controller's source and destination channel connectors together. See Figure 5-1 on page 5-2. The cable allows you to run loopback transfers on the HIPPI controller. Some of the controller's diagnostics require the use of a loopback cable.

## Loopback Cable Implementation

The loopback cable is a Madison Cable DOSDO7BTIA or equivalent. The connectors used on this cable must be the functional equivalent of AMP 749070-9 with female screw locks and housing.

# HIPPI Calls A

This appendix contains manual pages for the *libhippi.a* library of routines.

See the *Paragon™ C System Calls Reference Manual* for manual pages for system calls unique to Paragon OSF/1.

The manual pages in this appendix are also available online, using the **man** command.

# HIPPI_BIND() HIPPI_BIND()

Selects the incoming data that the user wants to receive.

## Synopsis

```
#include <raw_hippi.h>

int hippi_bind(
        int ihandle,
        u_char ulp,
        u_long port );
```

## Parameters

*ihandle* Specifies the HIPPI connection (ihandle) to be bound.

*ulp* A value greater than 0x80 that specifies the selected packets. (Values below 0x80 are reserved for use by ANSI.) The maximum value for *ulp* is 0xff.

*port* A positive value that specifies the port to bind to. A -1 value for *port* indicates that the *ulp* alone should be used to select the incoming packets.

## Description

You must call the **hippi_bind** function in order to read data from an open HIPPI channel. The call allows your application to establish a peer-to-peer relationship with another application. Incoming packets for your application are then correctly de-multiplexed based on the given ULP or ULP and port. (For the port to be evaluated, it must be in the first word of the D1_Area of the incoming packet.)

This routine will fail if another process is currently receiving data with the same ULP or ULP and port value.

## Return Value

On successful completion, **hippi_bind** returns 0. On failure, it returns -1 and sets the global variable *errno* to the appropriate value.

## HIPPI_BIND() *(cont.)*

HIPPI_BIND() *(cont.)*

### Errors

| | |
|---|---|
| EADDRINUSE | Another process is currently receiving data with the same ULP or ULP and port. |
| EADDRNOTAVAIL | The given ULP was not greater than 0x80. |

# HIPPI_CLOSE() HIPPI_CLOSE()

Closes a HIPPI connection and cleans up the state information maintained for the connection.

## Synopsis

```
#include <raw_hippi.h>

int hippi_close(
        int ihandle );
```

## Parameters

*ihandle* Specifies the HIPPI connection (ihandle) to be closed.

## Description

The **hippi_close** routine closes the HIPPI connection referred to by *ihandle* and cleans up the library state for this connection.

## Return Value

On successful completion, **hippi_close** returns 0; on failure, it returns -1.

## Errors

EBADF The *ihandle* referenced an invalid HIPPI connection.

# HIPPI_CONFIG() HIPPI_CONFIG()

Specifies packet framing semantics for HIPPI connections that are opened as HIPPI_DATA mode.

## Synopsis

```
#include <raw_hippi.h>

int hippi_config(
        int ihandle,
        u_long ifield,
        u_long ulp,
        u_long b,
        char *d1_data,
        u_short d1_len );
```

## Parameters

*ihandle* Specifies an open HIPPI connection.

*ifield* The address to which the packet is to be sent. The *ifield* must be compliant with the HIPPI-SC specification.

*ulp* The upper layer protocol to send the packet to.

*b* Indicates where the D2_Area starts in the packet:

| | |
|---|---|
| 0 | Indicates that D2_Area is not aligned |
| 1 | Indicates that D2_Area is burst-boundary aligned |

*d1_data* A pointer to the D1_data area.

*d1_len* Size of D1 data (the number of 64-bit words in *d1_data*).

## Description

The **hippi_config** function should only be used for connections opened in HIPPI_DATA mode; it should not be used for connections opened in HIPPI_RAW mode.

## HIPPI_CONFIG() *(cont.)*

HIPPI_CONFIG() *(cont.)*

The **hippi_config** function sets the format characteristics of the first burst of outbound packets. Parameter *b* should be TRUE if D2 must start at a burst boundary. If *d1_data* isn't NULL, *d1_len* bytes are copied into the *d1_data* area of the HIPPI frame. This combination of options result in a small data copy, but enables users to fully control the contents of the first burst.

### Return Value

On successful completion, **hippi_config** returns 0; on failure, it returns -1. (Failure indicates an invalid ihandle; the validity of the other parameters can only be determined at run time.)

### Errors

| | |
|---|---|
| EBADF | The connection referenced by ihandle is not a HIPPI_DATA mode connection. |

# HIPPI_MEMFREE() HIPPI_MEMFREE()

Releases the memory acquired by either **hippi_memget** or **hippi_read**.

## Synopsis

```
#include <raw_hippi.h>

int hippi_memfree(
        int ihandle,
        char *ptr,
        u_long size,
        int how );
```

## Parameters

*ihandle* Specifies the HIPPI connection (ihandle) associated with the memory to free.

*ptr* Pointer to the memory to free.

*size* Size, in bytes, of the block of memory to free.

*how* This parameter should be:

| | |
|---|---|
| 0 | If the memory was allocated by **hippi_memget**. |
| 1 | If the memory was acquired by **hippi_read**. |

## Description

This routine releases memory pointed to by *ptr*. This routine must be called for memory allocated by **hippi_memget**() and for memory acquired by **hippi_read**().

## Return Value

On successful completion, **hippi_memfree** returns 0; on failure, it returns -1.

# HIPPI_MEMGET() HIPPI_MEMGET()

Allocates memory for a HIPPI connection.

## Synopsis

```
#include <raw_hippi.h>

char * hippi_memget(
        int ihandle,
        u_long size );
```

## Parameters

*ihandle* Specifies the HIPPI connection.

*size* Specifies the number of bytes to allocate.

## Description

The **hippi_memget** routine allocates at least *size* bytes. The value of *size* should be large enough to hold the largest packet the user will send, including the I-field, headers, and data. The **hippi_memget** routine must be used with HIPPI_DATA connections and can be used with HIPPI_RAW. You should call **hippi_config** routine before calling **hippi_memget**.

## Return Value

On successful completion, **hippi_memget** returns a pointer to the allocated memory. For HIPPI_RAW mode, the pointer points to the start of the FP header area. For HIPPI_DATA mode, the pointer points to the place in the packet where HIPPI expects the user to put data (based on the parameters supplied in the call to **hippi_config**).

On failure, **hippi_memget** returns a null pointer.

## See Also

**hippi_config**

# HIPPI_OPEN() HIPPI_OPEN()

Establishes an open HIPPI connection.

## Synopsis

```
#include <fcntl.h>
#include <raw_hippi.h>
int hippi_open(

        char *dev_name,
        u_long hippi_mode,
        u_long mode );
```

## Parameters

*dev_name* HIPPI device through which connection is to be established. (Use the **rmknod** command to create the device.)

*hippi_mode* HIPPI mode: must be HIPPI_RAW or HIPPI_DATA.

*mode* UNIX I/O mode: should be O_RDONLY, O_WRONLY, or O_RDWR. (Other UNIX I/O modes, such as 0_CREAT, O_TRUNC, and O_EXCL, make no sense.) The mode must match the permissions set on the devices by the system administrator (using the **chmod** command).

## Description

The **hippi_open** routine opens a HIPPI connection and selects the connection's HIPPI and I/O modes.

## Return Value

On successful completion, **hippi_open** returns a positive integer. This integer, called the ihandle, is used in subsequent HIPPI operations for the connection. On failure, **hippi_open** returns -1.

## Errors

| | |
|---|---|
| EMFILE | There are too many open files. |

# HIPPI_READ() HIPPI_READ()

Reads from an open HIPPI connection.

## Synopsis

```
#include <raw_hippi.h>

char *hippi_read(
        int ihandle,
        u_long *len );
```

## Parameters

| | |
|---|---|
| *ihandle* | Specifies HIPPI connection to read from. |
| *len* | Pointer to the number of bytes that were read. |

## Description

The **hippi_read** function reads from the HIPPI connection specified by *ihandle*, stores the number of bytes that were read in the word pointed to by *len*, and returns a pointer to the bytes that were read.

You must call **hippi_bind** before calling **hippi_read** (otherwise **hippi_read** will fail).

## Return Value

On successful completion, **hippi_read** returns a pointer to the bytes that were read. The pointer points to the start of the FP header area.

On failure, **hippi_read** returns a null pointer.

## Errors

| | |
|---|---|
| EBADF | The *ihandle* referenced an invalid HIPPI connection. |

## See Also

**hippi_bind()**

# HIPPI_WRITE() HIPPI_WRITE()

Writes to an open HIPPI connection.

## Synopsis

```
#include <raw_hippi.h>

int hippi_write(
        int ihandle,
        char *ptr,
        u_long len );
```

## Parameters

| | |
|---|---|
| *ihandle* | HIPPI connection to which packet is written. |
| *ptr* | Pointer to write buffer. |
| *len* | Number of bytes to write. |

## Description

The **hippi_write()** function writes a packet to the open HIPPI connection referenced by *ihandle*. If HIPPI_RAW mode was selected in the call to **hippi_open()**, *ptr* points to the fully formatted frame (including the I-field), and **hippi_write()** writes the packet starting at *ptr*. If HIPPI_DATA mode was selected in the call to **hippi_open()**, **hippi_write()** completes the packet, adjusting *ptr* for copying the D1_data and FP header as specified in the call to **hippi_config()**, and then writes the packet.

## Return Value

On successful completion, **hippi_write()** returns 0; on failure it returns -1.

## Errors

| | |
|---|---|
| EBADF | The *ihandle* referenced an invalid HIPPI connection. |

## HIPPI_WRITE() *(cont.)*

HIPPI_WRITE() *(cont.)*

| | |
|---|---|
| EIO | An I/O error occurred. |
| EMSGSIZE | The packet to be written was larger than the HIPPI board could write. |

## See Also

**hippi_config(), hippi_open()**

# HIPPI Commands B

This appendix contains manual pages for the **hippi_setmap**, **hippi_showmap**, and **set_hippi_buffers** commands.

See the *Paragon™ Commands Reference Manual* for manual pages for parallel commands unique to Paragon OSF/1.

The manual pages in this appendix are also available online, using the **man** command.

# HIPPI_SETMAP HIPPI_SETMAP

Loads a HIPPI network routing table into the HIPPI driver.

## Syntax

**hippi_setmap [-d]** [*ifhip0*] [*mapfile*]

## Arguments

| | |
|---|---|
| **-d** | Deletes all entries in the routing table. |
| *ifhip0* | Specifies the interface number of the board whose routing table is being loaded. You must use **ifconfig** to define *ifhip0* before using *ifhip0* here. |
| *mapfile* | Specifies the file that contains the HIPPI network routing table. The **hippi_setmap** command loads this routing table into the HIPPI driver. |

## Description

The **hippi_setmap** command loads the HIPPI network routing table into the HIPPI server (this is accomplished by broadcasting, so you do not need to specify the receiving HIPPI interface.) The routing table lists IP addresses, ULAs, and I-fields and enables **hippi_setmap** to map IP addresses to I-fields.

## Examples

The **hippi_setmap** command takes as input a file formatted like the following:

```
#HIPPI Network Routing Table
#
#IP Address        ULA                 I-field
#-----------       --------------      ----------
 192.9.3.1         1:0c:34:65:0:26     0x1000000   #Zombie
 192.9.3.2         1:0c:34:65:0:27     0x1000002   #Jaguar
 192.9.3.3         1:0c:34:65:0:28     0x1000001   #Balu
 192.9.3.4         1:0c:34:65:0:29     0x1000003   #Carlsbad
```

In the above table, the camp-on bit in the I-field is set. This instructs the HIPPI switches to attempt a connection until the connection is completed or the source cancels the connection request.

To load the file *hippi.map* into the server, type:

```
hippi_setmap  ifhip0 hippi.map
```

## HIPPI_SETMAP *(cont.)*

HIPPI_SETMAP *(cont.)*

### See Also

**hippi_showmap, ifconfig(8)**

# HIPPI_SHOWMAP HIPPI_SHOWMAP

Displays the current HIPPI network routing table.

## Syntax

**hippi_showmap [-n]**

## Arguments

**n** Causes the command to search for and display the ASCII names of the hosts in the routing table. These names correspond to IP addresses.

## Description

The **hippi_showmap** command displays the current HIPPI network routing table (loaded by the most recent **hippi_setmap** command).

Without the -n argument, **hippi_showmap** displays IP addresses in the first column:

```
#IP Address        ULA                 I-field
#-----------       --------------      ----------
192.9.3.1
192.9.3.2
```

With the -n argument, **hippi_showmap** displays hostnames in the first column:

```
#Hostname          ULA                 I-field
#-----------       --------------      ----------
Tornado
Hurricane
```

## Example

To display the current routing table, type the following:

```
hippi_showmap
```

## See Also

**hippi_setmap**

# SET_HIPPI_BUFFERS SET_HIPPI_BUFFERS

Sets the number and size of HIPPI receive buffers.

## Syntax

**set_hippi_buffers** [-v] [*filename*]

## Arguments

| | |
|---|---|
| **-v** | Invokes verbose mode, in which both diagnostic and error output are saved in file */usr/tmp/raw_hippi.log*. |
| *filename* | File containing configuration data. Default is */etc/hippi.conf*. |

## Description

The **set_hippi_buffers** command sets the number and size of the HIPPI receive buffers based on the information in file */etc/hippi.conf* (default) or file *filename*.

The **set_hippi_buffers** command is executed automatically (using the default file) when the system is booted. Although it is not recommended, the command can also be executed (using file *filename*) by the system administrator when the system is running. In this case, the number and size of the receive buffers is application-dependent, but for performance reasons the cumulative size of all the buffers should not exceed 16M bytes (one-half of the memory available on a 32M byte GP node board). The file should have the same format as the default file:

```
# Device name   Receive Buffer size   Number of buffers
# -----------   -------------------   -----------------
/dev/hippi.x    1048576               6
/dev/hippi.y    2097152               3
............    .......               .
```

Whether the command is invoked at boot time or run time, the receive buffer size specified for each device also determines the send buffer size for that device. This size is called the maximum transfer unit.

## Examples

To configure the buffers automatically at boot time, modify file */etc/hippi.conf* and boot the system. To configure the buffers from a file called *myfile* while the system is running, enter:

```
set_hippi_buffers myfile
```

## SET_HIPPI_BUFFERS *(cont.)*

SET_HIPPI_BUFFERS *(cont.)*

### See Also

**rmknod**