Working Draft

T11.1/ Project 1231/ Rev 1.9

January 5, 1998

Information Technology - High-Performance Parallel Interface - 6400 Mbit/s Physical Switch Control (HIPPI-6400-SC)

Secretariat: National Committee for Information Technology Standardization (NCITS)

ABSTRACT

This document describes a protocol for controlling physical layer switches which are based on the High-Performance Parallel Interface at 6400 Mbit/s (HIPPI-6400-PH), a high-performance, point-to-point interface for transmitting digital data at peak rates of 6400 Mbit/s between data processing equipment.

Contacts:

T11.1 Vice Chairman and Technical Editor

Roger Ronald
Raytheon E-Systems
MS 35300 HD
PO Box 660023
Dallas, TX 75266-0023
Voice: 972-205-8043
FAX: 972-272-8144
E-mail: rronald@esy.com

T11.1 Chairman

Don Tolmie
Los Alamos National Laboratory
CIC-5, MS-B255
Los Alamos, NM 87545
Voice: 505-667-5502
FAX: 505-665-7793
E-mail: det@lanl.gov

Reference number
ISO/IEC 11518-xx: 199x
NCITS. ***- 199x
Printed 2/12/98

Other Points of Contact:

	T11 Chairman	T11 Vice-Chairman	NCITS Secretariat
	Roger Cummings Distributed Processing Technology 140 Candace Drive Maitland, FL 32751	Edward L. Grivna Cypress Semiconductor 2401 East 86th Street Bloomington, MN 55425	NCITS Secretariat, ITI 1250 Eye Street, NW Suite 200 Washington, DC 20005
Voice:	407-830-5522 x348	612-851-5046	202-737-8888
FAX:	407-260-5366	612-851-5087	202-638-4922
E-mail:	cummings_roger@dpt.com	elg@cypress.com	ncitssec@itic.nw.dc.us

T11.1 E-mail Reflector (for HIPPI technical discussions and notifications of web changes

Internet address for subscription to the HIPPI reflector:	Majordomo@network.com
Messages should contain a line stating...	subscribe hippi <*your e-mail address*>
Internet address for distribution via the HIPPI reflector:	hippi@network.com

T11 E-mail Reflector (forT11 meeting notices, agendas, etc.)

Internet address for subscription to the T11 reflector:	Majordomo@network.com
Messages should contain a line stating...	subscribe T11 <*your e-mail address*>
Internet address for distribution via the T11 reflector:	t11@network.com

Web Sites:

HIPPI Standards Activities	http://www.cic-5.lanl.gov/~det
T11 Activities	http://www.dpt.com/t11
NCITS	http://www.x3.org/

T11 Document Distribution

Global Engineering
15 Inverness Way East
Englewood, CO 80112-5704
Voice: 303-792-2181 or 800-854-7179
FAX: 303-792-2192

PATENT STATEMENT

CAUTION: The developers of this standard have requested that holder’s of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard.

As of the date of the publication of this standard and following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made. No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

Table Of Contents

Figures

Tables

Annex

Foreword (This Foreword is not part of American National Standard X3.xxx-199x.)

This American National Standard specifies the behavior and control for HIPPI-6400 physical layer switches. HIPPI-6400 is an efficient high-performance point-to-point interface. HIPPI-6400 physical layer switches may be used to give the equivalent of multi-drop capability, connecting together multiple data processing equipments.

This standard provides an upward growth path for legacy HIPPI-based systems.

This document includes annexes which are informative and are not considered part of the standard.

Requests for interpretation, suggestions for improvement or addenda, or defect reports are welcome. They should be sent to the National Committee for Information Technology Standards (NCITS), ITI, 1250 Eye Street, NW, Suite 200, Washington, DC 20005.

This standard was processed and approved for submittal to ANSI by Accredited Standards Committee on Information Processing Systems, X3. Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, the NCITS had the following members:

Karen Higgenbottom, Chairman (Acting)
Karen Higgenbottom, Vice-Chair
Monica Vago, Secretary

Organization Represented	Name of Representative
AMP, Inc.	John Hill Charles Brill (Alt.)
Apple Computer Inc.	David Michael Jerry Kellenbenz (Alt.)
AT&T	Thomas Frost Paul Bartoli (Alt.)
Bull HN Information Systems Inc.	Patrick L Harris
Compaq Computer Corporation	Stephen Heil Steve Park (Alt.)
Digital Equipment Corporation	Scott Jameson Richard Hovey (Alt.)
Eastman Kodak Company	Michael Nier
Hewlett-Packard Copany	Karen Higginbottom Donald Loughry (Alt.)
Hitachi American Ltd.	John Neumann Kei Yamashita (Alt.)
Hughes Aircraft Company	Harold Zebrack
IBM Corporation	Ron Silletti Joel Urman (Alt.)
Imation	Philip E. Friedlund
Institute for Certification of Computer Professionals (ICCP)	Kenneth M. Zemrowski
Lucent Technologies Inc.	Herbert Bertine Tom Rutt (Alt.)
National Communications Systems	Dennis Bodson William Olden (Alt.) Frank McClelland (Alt.)
National Institute of Standards & Technology	Michael Hogan Bruce K. Rosen (Alt.)
Panasonic Technologies Inc.	Judson Hofmann Y. Machida (Alt.)
Share Inc.	Dave Thewlis Gary Ainsworth (Alt.)
Sony Electronics Inc.	Masataka Ogawa Michael Deese (Alt.)

Storage Technology Corporation	Joseph Zajaczkowski
Sun Microsystems Inc.	Gary Robinson
Sybase Inc.	Donald Deutsch
Texas Instruments Inc.	Clyde Camp
	Fritz Whittington (Alt.)
Unisys Corporation	Arnold F. Winkler
	Stephen Oksala (Alt.)
U.S. Department of Defense/DISA	C.J. Pasquariello (Alt.)
U.S. Department of Energy	Carol Blackston
	Bruce R. White (Alt.)
Xerox Corporation	John B. Flannery
	Roy Pierce (Alt.)

Technical Committee T11 on Device Level Interfaces, which reviewed this standard, had the following members:

Roger Cummings, Chairman
Ed Grivna, Vice Chairman

A. Abe
I. Ahmed
D. Allan
D. Allen
R. Anantharaman
B. Atkinson
D. Baldwin
C. Bazaar
S. Berglund
S. Berman
C. Binford
R. Bossard
P. Boulay
J. Bowerman
C. Brill
L. Bryant
E. Cady
J. Calle
E. Chang
M. Chenery
K. Conroy
R. Cook
J. Coomes
R. Cornelius
R. Crouch
G. Crow
Z. Daggett
R. Dahlgren
S. Darnell
J. Dedek
D. Deming
K. Desai
M. Dorsett
J. Dumont
B. Edge
M. Fitzpatrick
D. Ford
D. Ford
C. Foster
M. Foster
R. Franse
K. Fredericks
E. Frymoyer
E. Gardner
A. Ghiasi
M. Griffin
D. Hagerman
B. Ham
T. Harrington
N. Harris
V. Haydu
D. Hepner
M. Hoard
D. Hollenback
T. Hudson
D. Hyer
D. Instone
B. Johnson
D. Johnson
S. Joiner
J. Jones
S. Jones
M. Karg
A. Kelley
J. Kembel
R. Kembel
B. Ketchum
R. Kleckowski
B. Kuypers
K. Kwiat
D. LaFollette
P. Lawthers
E. Lee III
P. Levin
T. Lindsay
J. Lohmeyer
A. Martin
W. Martin
J. Mathis
J. McGrath
V. Melendy
G. Milligan
C. Monia
H. Naidu
J. Neer
J. Nelson
C. Nieves
M. O'Donnell
J. Oliver
T. Palkert
J. Parker
R. Pedersen
D. Perkins
D. Peterson
W. Rickard
C. Ridgeway
R. Ronald
E. Rydell
J. Salamone
J. Schaefer
J. Scheible
P. Seto
B. Smith
R. Snively
R. Soderstrom
T. Sprenkle
J. Stai
G. Stephens
A. Stone
S. Swanson
T. Szostak
R. Taborek
J. Tatum
T. Thompson
L. Thorsbakken
D. Tolmie
H. Truestedt
S. Van Doorn
M. Wakeley
N. Wanamaker
G. Warden
C. Whitby-Strevens
B. Willard
J. Williams
S. Wilson
M. Wingard
K. Witte
R. Yonemoto
C. Zeitler

Task Group T11.1 on the High Performance Parallel Interface, which developed this standard, had the following participants:

Don Tolmie, Acting Chairman
Roger Ronald, Vice Chairman and HIPPI-6400-SC Technical Editor

M. Bancroft
A. Beckman
B. Boas
G. Boyd
H. Brandt
C. Brill
D. Brown
E. Cady
G. Chesson
J. Chung
R. Clarkson
H. Collins
R. Cummings
C. Davidson
J. Davis
R. DeFillips
M. Derstine
M. Donhowe
M. Doppke
A. G. Dornhoff
J. Ellis
R. Ellison
M. Ficarra
S. Foreman
C. Foster
M. Foster
F. Gaullier
A. Ghiasi
J. Gibbon
T. Gilbert
M. Griffin
M. Hoard
J. Hoffman
G. Huff
D. Hyer
R. Hyerle
S. Joiner
M. Karg
A. Kelley
M. Kelley
D. Knasel
M. Leib
J. Leitherer
D. Liberty
S. Locke
B. McCoy
M. McGowen
B. Newhall
R. Nikel
C. Olson
C. Pan
J. Parker
D. Parry
B. Pearson
I. Philp
C. Pick
J. Pinkerton
J. M. Pittet
S. Quan
J. Renwick
D. Sanders
C. Satterlee
D. Schwartz
W. St. John
S. Swirhun
F. Templin
C. Theorin
H. Van Deusen
S. Van Doorn
B. Weber
V. Welch
A. Widmer
B. Willard
J. Young

Introduction

This 6400 Mbits/second High-Performance Parallel Interface, Physical Switch Control (HIPPI-6400-SC) standard defines the control for HIPPI-6400 physical layer switches. HIPPI-6400 is an efficient high-performance point-to-point interface. Small fixed-size micropackets provide an efficient, low-latency, structure for small messages, and a building block for large messages. HIPPI-6400 physical layer switches may be used to give the equivalent of multi-drop capability, connecting together multiple data processing equipments.

Characteristics of this HIPPI-6400 physical switch control protocol include

- support for 48-bit Universal LAN Addresses (ULAs);
- support for restricted mode operation with a 16-bit subset of the ULA;
- procedures for use of Admin micropackets to automate ULA assignment;
- ability to span multiple physical layer switches within a fabric;
- support for physical layer switches with differing numbers of ports, all within the same fabric;
- specified reserved ULAs to aid address self-discovery, switch management, and switch control;
- support for 4 Virtual Channels; and
- broadcast capabilities with loop avoidance, using the IEEE 802.1d Spanning Tree Algorithm and Protocol, either within a switch or provided by an attached server.

American National Standard
for Information Technology –

High-Performance Parallel Interface –
6400 Mbit/s Physical Switch Control (HIPPI-6400-SC)

1 Scope

This American National Standard provides switch control for physical layer switches using the 6400 Mbits/second High-Performance Parallel Interface (HIPPI-6400), a high-performance point-to-point interface between data-processing equipment.

The purpose of this standard is to facilitate the development and use of the HIPPI-6400 in computer systems by providing common physical switch control. The standard provides switch control structures for physical layer switches interconnecting computers, high-performance display systems, and high-performance, intelligent block-transfer peripherals. This standard also applies to point-to-point HIPPI-6400 topologies.

Specifications are included for

- interleaving of Virtual Channels (VCs) within a physical channel;
- selection of Messages for transmission on physical channels;
- self discovery of configuration information; and
- broadcast capability with loop avoidance using the IEEE 802.1d Spanning Tree Algorithm and Protocol.

2 Normative references

The following standards contains provisions which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent edition of the standard listed below.

Copies of the following documents can be obtained from ANSI: Approved ANSI standards, approved and draft international and regional standards (ISO, IEC, CEN/CENELEC, ITUT) and approved and draft foreign standards (including BSI, JIS, and DIN). For further information, contact ANSI Customer Service Department at 212-642-4900 (phone), 212-302-1286 (fax) or via the World Wide Web at http://www.ansi.org. Additional availability contact information is provided below as needed.

2.1 Approved references

ANSI X3.183-1991, *High-Performance Parallel Interface – Mechanical, Electrical, and Signalling Protocol Specification (HIPPI-PH)*

ANSI X3.210-1992, *High-Performance Parallel interface, Framing Protocol (HIPPI-FP)*

ANSI X3.222-1993, *High-Performance Parallel interface, Physical Switch Control (HIPPI-SC)*

ISO/IEC 10038/ANSI/IEEE 802.1D-1990, *Media access control (MAC) bridges, (Specifies the operation of transparent bridges between IEEE 802 conferment networks)*

IEEE Std 802-1990, *IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture*

ISO/IEC 8802-2:1989 (ANSI/IEEE Std 802.2-1989), *Information Processing Systems - Local Area Networks - Part 2: Logical Link control*

2.2 References under development

At the time of publication, the following referenced standards were still under development. For information on the current status of the document, or regarding availability, contact the relevant standards body or other organization as indicated. For more information about obtaining copies of this document or for more information of the current status of the document, contact National Committee for Information Technology Standards, 1250 Eye Street, NW, Suite 200, Washington, DC 20005, 202-626-5746.

ANSI X3.xxx-199x, *High Performance Parallel Interface 6400 Mbits/s, Physical Layer (HIPPI-6400-PH)*

3 Definitions and conventions

3.1 Definitions

For the purposes of this standard, the following definitions apply.

3.1.1 Admin Element address: A 32-bit field uniquely identifying an Element.

3.1.2 Admin micropacket: A HIPPI-6400 micropacket used for configuration and management.

3.1.3 administrator: A station management entity providing external management control.

3.1.4 alternate pathing: Capability to address a Message to select from a group of ports based upon defined criteria.

3.1.5 broadcast: The capability for a Source to send one Message that arrives at multiple Destinations.

3.1.6 Destination: The receiving end of a physical link.

3.1.7 Device: Any system level component (e.g. endpoint or switch) with a HIPPI-6400 port.

3.1.8 Element: Any component of a HIPPI-6400 system that is able to receive, process, and send Admin micropackets in a manner conforming to this standard.

3.1.9 endpoint: A device that is capable of acting as a Final Destination and/or an Originating Source.

3.1.10 fabric: All of the switching equipment and resulting pathways connected together in a configuration.

3.1.11 Final Destination: The end device that receives, and operates on, the data payload portion of the micropackets. This is typically a host computer system, but may also be a translator, bridge, or router**.**

3.1.12 HIPPI-PH: High-Performance Parallel Interface - Mechanical, Electrical, and Signalling Protocol Specification (HIPPI-PH), ANSI X3.183-1991. Data is transmitted in parallel over copper twisted-pair cables at 800 or 1600 Mbits per second.

3.1.13 in-band: Switch control communications accomplished over a HIPPI-6400 link, as opposed to using an alternative (non-HIPPI-6400) communication channel.

3.1.14 link: A full-duplex connection between HIPPI-6400-PH Devices.

3.1.15 link-end: A hardware device that terminates one end of a link.

3.1.16 log: The act of making a record of an event for later use.

3.1.17 Message: An ordered sequence of one or more micropackets which have the same VC, Originating Source, and Final Destination. Messages are the basic transfer unit between an Originating Source and a Final Destination. The first micropacket of a Message is a Header micropacket. The last micropacket, which may also be the first micropacket, has the TAIL bit set.

3.1.18 micropacket: The basic transfer unit consisting of 32 data bytes and 64 bits of control information.

3.1.19 optional: Characteristics that are not required by HIPPI-6400-SC. However, if any optional characteristic is implemented, it shall be implemented as defined in HIPPI-6400-SC.

3.1.20 Originating Source: The end device that generates the data payload portion of the micropackets. This is typically a host computer system, but may also be a translator, bridge, or router.

3.1.21 Source: The sending end of a physical link.

3.1.22 switch: An equipment that provides connections between HIPPI-6400 links based on this standard.

3.1.23 Universal LAN MAC Address (ULA): A logical address stored in a Source or Destination field that uniquely identifies an Originating Source or Final Destination. The ULA conforms to the 48-bit MAC address specified by the IEEE 802 Overview Standard.

3.1.24 Virtual Channel (VC): One of four logical paths within each direction of a single link.

3.2 Editorial conventions

In this standard, certain terms that are proper names of signals or similar terms are printed in uppercase to avoid possible confusion with other uses of the same words (e.g., FRAME). Any lowercase uses of these words have the normal technical English meaning.

A number of conditions, sequence parameters, events, states, or similar terms are printed with the first letter of each word in uppercase and the rest lowercase (e.g., State, Source). Any lowercase uses of these words have the normal technical English meaning.

The word *shall* when used in this American National standard, states a mandatory rule or requirement. The word *should* when used in this standard, states a recommendation.

3.2.1 Binary notation

Binary notation is used to represent relatively short fields. For example a two-bit field containing a binary value of 10 is shown in binary format as b'10'.

3.2.2 Hexadecimal notation

Hexadecimal notation is used to represent some fields. For example a two-byte field containing a binary value of b’1100010000000011’ is shown in hexadecimal format as x'C403'.

3.2.3 Bit/Byte naming conventions

As specified in HIPPI-6400-PH:

- In a parameter that uses multiple bytes, the most-significant byte is the lowest-numbered byte;
- In a parameter that uses multiple bits, the most-significant bit is the highest-numbered bit.

3.2.4 Acronyms and other abbreviations

CRC	cyclic redundancy check
ECRC	end-to-end CRC
HIPPI	High-Performance Parallel Interface
IP	Internet Protocol
LAN	local area network
MAC	media access control
PH	Physical
SC	Switch Control
ULA	universal LAN address
VC	virtual channel

4 System overview

This paragraph provides an overview of the structure, concepts, and mechanisms used in HIPPI-6400-SC. An example system configuration is shown in figure 1.

4.1 Switch function

A HIPPI-6400 switch fabric provides a method to send Messages from an Originating Source port to a Final Destination port. Each Message travels on one of the four Virtual Channels (VCs) available in HIPPI-6400-PH (see HIPPI-6400-PH for assignments of Message type to VC). All of the micropackets of a Message are transmitted on a single VC, i.e., the VC number does not change as the micropackets travel from the Originating Source to the Final Destination over one or more links.

Different VCs are interleaved on the physical channel, allowing up to four Messages to proceed to a Destination or from a Source at any given time.

During transfer of a Message, the VC in use is busy and is unavailable for use by other Messages involving the same Source or Destination ports.

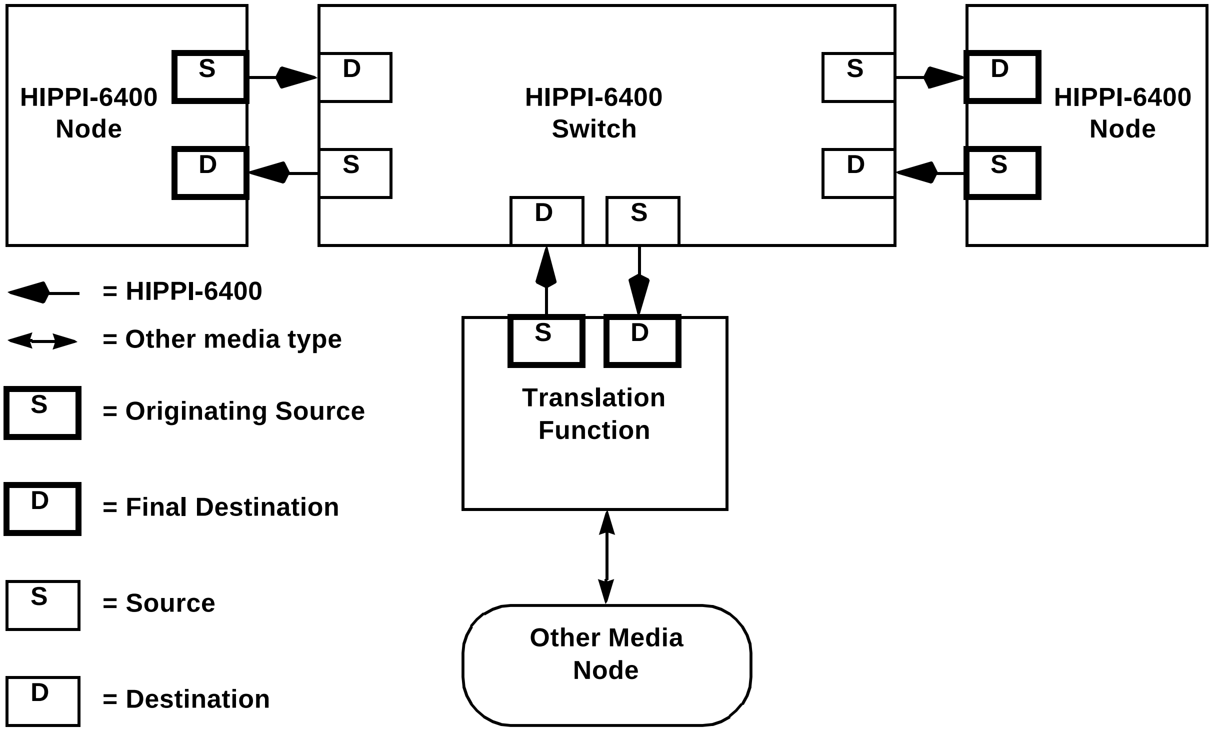


Figure 1 - System overview

4.2 Micropacket

Micropackets are the basic transfer unit for HIPPI-6400. As described in HIPPI-6400-PH, a micropacket is composed of 32 data bytes and 64 bits of control information.

The 64 bits of control information in each micropacket includes parameters for physical (PH) layer functions and for switch control (SC) functions. These functions include

- selecting a VC;
- detecting missing micropackets;
- denoting the types of information in the micropacket;
- marking the last micropacket of a Message; and
- signalling that the Message was truncated at its originator, or damaged en-route, and should be discarded.

Table 1 describes the information that the switch fabric propagates from a HIPPI-6400-PH Originating Source to a HIPPI-6400-PH Final Destination. Table 2 and table 3 describe the information that a switch fabric uses to determine micropacket routing.

Table 1 - Data carried through fabric

Description	Size
ERROR	1 Bit
TAIL	1 Bit
VC	2 Bits
TYPE	4 Bits
ECRC	16 Bits
Payload Data	32 Bytes

Table 2 - Data to route 1st micropacket in a Message

Description	Size
TAIL	1 Bit
VC	2 Bits
TYPE	4 Bits
Payload Data	32 Bytes

Table 3 - Data to Route subsequent micropackets in a Message

Description	Size
TAIL	1 Bit
VC	2 Bits
TYPE	4 Bits

Table 4 contains information that can be used to determine whether the micropacket contains errors and a means to report discovered errors.

Table 4 - Data used for error checking and reporting

Description	Size
ERROR	1 Bit
TYPE	4 Bits
ECRC	16 Bits
Payload Data	32 Bytes

Note that there is information used by the switch fabric that also is carried through it.

4.3 Message

As shown in figure 2, Messages are an ordered sequence of micropackets which have the same VC, Originating Source, and Final Destination. The first micropacket of a Message, i.e., the Header micropacket, contains information used to route through a HIPPI-6400 fabric (see figure 3) as well as other information as specified in HIPPI-6400-PH. The last micropacket of the Message is marked with the TAIL bit.

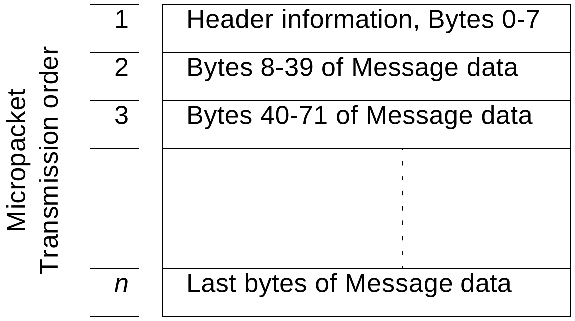

Micropacket Transmission order
1
Header information, Bytes 0-7
2
Bytes 8-39 of Message data
3
Bytes 40-71 of Message data
n
Last bytes of Message data

Figure 2 - Message format

4.4 Admin micropackets

HIPPI-6400-PH specifies a micropacket with Type = Admin. HIPPI-6400 switches use Admin micropackets for configuration discovery, address assignment, and broadcast configuration.

4.5 Broadcast

HIPPI-6400 switches provide a method for the broadcast of Messages either directly or through an external broadcast server. Broadcast Messages are propagated along a loop-free spanning tree of interconnected HIPPI-6400 switches. The spanning tree is constructed by using the IEEE 802.1d Spanning Tree Algorithm and Protocol.

5 Switch processing

5.1 Micropacket data passed through fabric

A HIPPI-6400 switch shall propagate the information shown in table 1 through the fabric. Micropacket data payload, the TAIL bit, the TYPE field, the VC field, and the ECRC shall not be modified while passing through a switch fabric. The ERROR bit shall be transferred as set if it was received as set. If the ERROR bit is received as not set, the bit may be set to indicate a switch detected error as described in 5.4.

5.2 Routing of Header micropacket

Figure 3 shows part of the Header micropacket. The complete specification is provided in HIPPI-6400-PH.

Within the Header micropacket, the Destination ULA specifies the Final Destination where a Message is to be sent.

The micropacket TYPE field (TYPE = x’9’) identifies a micropacket as a Header micropacket.

TAIL = 1 on a Header micropacket indicates that there are no other micropackets for this Message.

The micropacket VC field specifies one of four Virtual Channels that this micropacket will use to traverse the switch fabric (micropackets traverse a fabric on a single VC and never cross VCs).

Switches should support independent ULA mapping for each input port. This permits mapping the same ULA value to different output ports (but not different Final Destinations; see 7.2.3) based upon which input port received the micropacket. See Annex A for an explanation of input port specific switching functionality.

5.2.1 Switch addressing

Switches shall support a mode of operation that provides in-order delivery of all micropackets on a VC from an Originating Source to a Final Destination.

Switches may also provide optional modes of operation such as alternate pathing. These optional modes of operation are not covered by this standard and may not guarantee in-order Message delivery.

Destination ULA DB00-DB05
Source ULA DB06-DB11
Defined in HIPPI-6400-PH DB12-DB31

Figure 3 - Header micropacket addressing

5.2.2 Full Destination ULA processing

All 48 bits of the Destination ULA should be used to determine the routing of Messages. Using the entire Destination ULA provides transparent addressing interoperability with other IEEE 802 compatible networks (e.g. Ethernet). Switches that use 48-bit ULAs are also capable of interoperating in fabrics that use less than 48 bits of ULA.

5.2.3 Partial Destination ULA processing

Less than the full 48-bit Destination ULA may be used in a HIPPI-6400 switch where bridging to other IEEE 802 media is not required.

A minimum of 8 bits of the Destination ULA shall be used for determining the routing of Messages. When part of the Destination ULA rather than the entire Destination ULA is used for determining the routing of Messages, the portion used shall be the least significant bits of the Destination ULA (starting with bit 0 of DB05).

Interoperability between switches that process a subset of the Destination ULA occurs in the intersecting ULA ranges of all switches in a fabric. The smallest range of ULAs supported by any switch limits all switches in the same fabric to that limited ULA range.

Administrators of a fabric processing less than the entire 48-bit ULA shall manage the assignment of ULAs to assure that unique addressing is provided (see Clause 7). Techniques for such management (e.g. table assignment, vendor specific switch-to-switch communication) are beyond the scope of this standard.

Since other IEEE 802 media do not support switch controlled ULA assignment, the smaller ULA ranges created by using partial ULA processing effectively prevent transparent bridging.

5.3 Routing of subsequent micropackets in a Message

Subsequent micropackets in a Message (identified by TYPE = x’8’ or TYPEs x’B’ through x’E’) shall be delivered to the same Final Destination as the Header micropacket.

The VC field shall be used to distinguish which Message the micropacket belongs to (of the four VCs supported).

When a micropacket is received with the TAIL bit = 1, it indicates that the Message ends.

5.4 Error protection

If an uncorrectable error is detected in a micropacket that is forwarded, the switch shall set the ERROR bit for that micropacket.

The ERROR bit may also have been set to indicate uncorrectable errors detected prior to HIPPI-6400 origination.

Detected errors shall be logged.

5.4.1 Mandatory error checking

The switch fabric shall pass the unchanged ECRC with each micropacket as specified in HIPPI-6400-PH.

Before sending any micropacket over a HIPPI-6400 link, the switch shall validate the ECRC and set the ERROR bit if the ECRC indicates an error as specified in HIPPI-6400-PH.

5.4.2 Optional error checking

The switch fabric may verify the validity of the ECRC at any point within the fabric.

The switch may also provide additional error detection or correction for internal data errors.

5.4.3 Congestion management

Time-out mechanisms defined in HIPPI-6400-PH will act to prevent switch congestion due to lack of progress on a HIPPI-6400 link, so long as the Source end of the link is functional. However, failures in switch Source ports can prevent this mechanism from functioning.

Switches shall protect against this failure mode by checking Source output ports for continued proper function and by discarding data destined for all failed Source output ports.

5.5 Data interleaving

There are two separate requirements for switch fairness to resolve contention for shared resources. Both micropackets and Messages shall be interleaved as described. These two interleaving processes shall be considered independent and applied without regard to each other.

5.5.1 Micropacket interleaving

Micropacket interleaving between the four VCs shall be applied on a micropacket count basis.

When a switch port has more than one VC with data available for output, the switch shall ensure that micropackets from each VC are afforded an equal opportunity for progress on a physical link.

The algorithm for choosing a micropacket from the available VCs shall allow interleaving on a frequent basis. The recommended algorithm is to interleave VC streams on a single micropacket basis.

Implementations trying to keep short Messages intact (to minimize latency) may use algorithms that interleave on other than a single micropacket basis. No implementations shall permit more than 69 micropackets from a particular VC to be transferred before moving on to the next VC. This limit allows transfer of the maximum permitted VC0 Message (as specified in HIPPI-6400-PH).

Figure 4 shows a simplified switch configuration with two input ports and one output port. Assuming that traffic is available to send to port “C” on more than one VC, a compliant switch alternates between providing output across all busy VCs on link “C”, not exceeding the micropacket count limit before switching from one VC to the next VC.

5.5.2 Message/Admin micropacket interleaving

Message and Admin micropacket interleaving shall be applied whenever a current Message (or Admin micropacket) to an output port is completed.

When a switch has more than one input port with Messages or Admin micropackets ready for transfer to the same output port (on the same VC), the switch shall ensure that Messages and/or Admin micropackets from the input ports are afforded an equal opportunity for progress. All ports with pending Messages or Admin micropackets shall be serviced prior to any other port being serviced twice.

In figure 4, an example would be if both port “A” and port “B” have multiple Messages available on their VC0 links ready to send to port “C”. In this example, Messages transferred out VC0 of port “C” are required to alternate between Messages from “A” and “B”.

6 Admin micropackets

Admin micropackets are used for support and initialization of HIPPI-6400 links, Elements, and systems. Each labeled component in figure 5 could be an Element.

There are two basic types of Admin micropacket function:

- Within a HIPPI-6400 endpoint or switch, Admin micropackets can be used for internal control of components. This internal usage is done for vendor convenience and is not required to support HIPPI-6400 functionality. Many of the defined Admin micropacket commands will be useful for this control, but the commands used for ULA assignment will not be applicable;

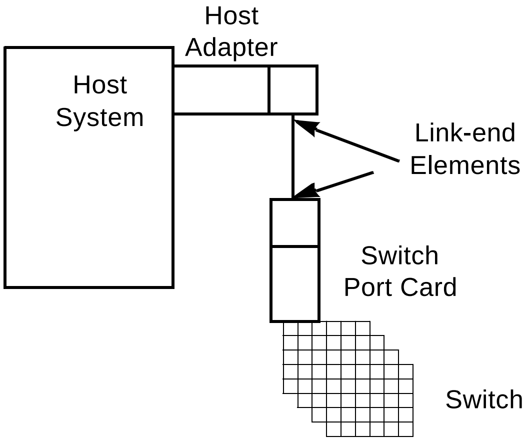


Figure 5 - Potential HIPPI-6400 Elements

- From one HIPPI-6400 Device (e.g. switch or endpoint) to another, Admin micropackets are used for topology discovery, ULA assignment, support of message broadcast, and ULA discovery. The ability to send and then receive an

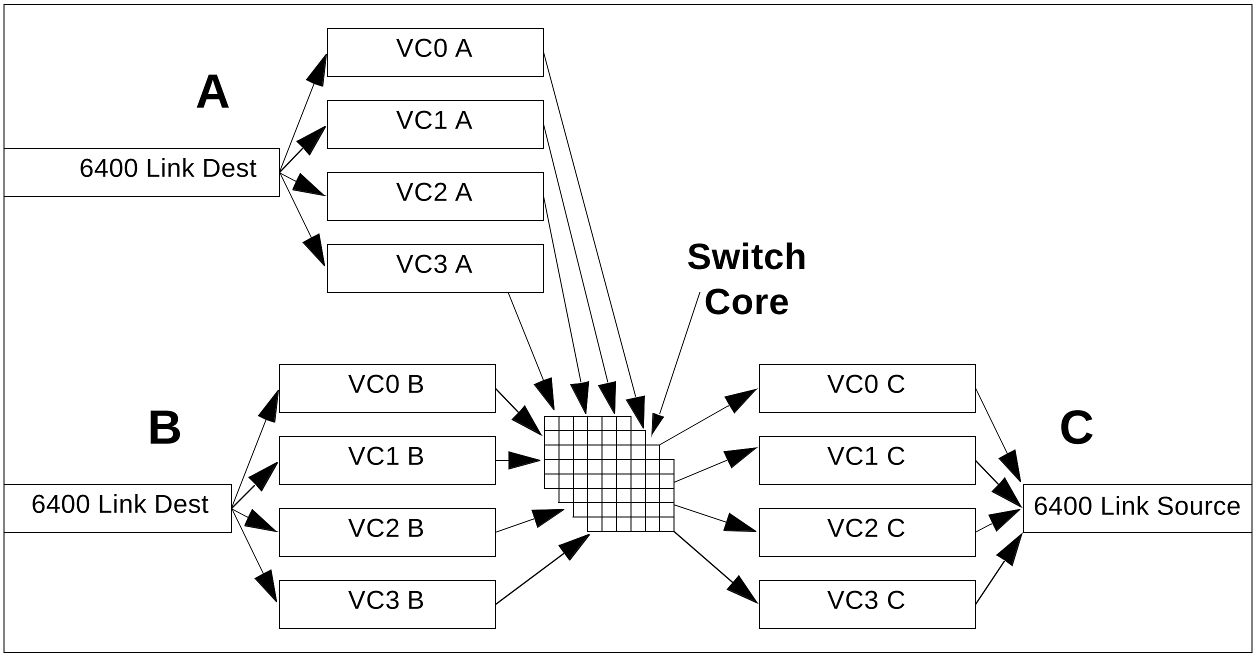


Figure 4 - HIPPI-6400 Switch

echoed micropacket may also be useful as a diagnostic feature. Most other Admin micropacket commands are not useful in this context.

6.1 Elements

An Element is any component of a HIPPI-6400 system that is able to receive, process, and send Admin micropackets in a manner conforming to this standard.

Each end of a HIPPI-6400 link shall operate as an Element. Other components of switches or adapters may optionally conform to the Element definition. These could include adapter cards, integrated circuits, or software entities.

At a minimum, Elements shall support commands and responses for the discovery of Element function, ULA assignment, and ULA discovery. Implementation of other functions called for by Admin micropacket commands are optional.

All Elements shall respond to each Admin micropacket command (except RESET, for which no response is ever made) with the specified response Admin micropacket (see 6.9) or with an INVALID_COMMAND Admin micropacket. If an Element does not implement an Admin command, it shall return status to that effect in the response micropacket.

6.2 Admin micropacket functions

A small set of commands allow for

- diagnostic “pings” between HIPPI-6400 Elements, either locally or across a link;
- initial Element address assignment;
- discovery of the function of an Element (e.g. switch or non-switch);
- HIPPI-6400 Source ULA assignment;
- discovery of Destination ULAs attached to a local switch;
- registration for broadcast;
- selection/configuration of a broadcast server; and
- vendor defined register access/functionality.

6.3 Admin micropacket format

Table 5 and figure 6 both show the format of an Admin micropacket. Admin micropackets shall contain the following fields:

- *Key:* The Key field is used in certain operations to validate that the originator is authorized to perform the requested operation. Because the key is only 8 bits in length and is returned in response to the SET_ELEMENT_ADDRESS, the protection provided by the key is minimal and only protective against accidental changes. Vendors may also choose to protect their system configuration in other unspecified ways. For example, a vendor may only allow commands that cause configuration changes to occur through a specific port;
- *Hop Count:* If the incoming hop count is zero, the micropacket shall be processed or discarded without a response. If the destination Admin Element address is x’FFFFFFFF’, a hop count of zero shall indicate that the Admin micropacket is valid for local processing. All other hop count values in conjunction with a destination Admin Element address of x’FFFFFFFF’ indicate that the micropacket shall continue to be forwarded. The value contained in the Hop Count field shall be decremented by one each time an Admin micropacket exits an Element. If an Admin micropacket is received without a valid Element address match and it cannot be forwarded, it shall be discarded without a response. See figure 7 for a diagram showing how Element addresses are processed;
- *Destination Admin Register:* The Destination Admin Register field specifies a register within a HIPPI-6400 Element. There are no specific registers required in any Element by this standard and use of any register(s) is optional;
- *Destination Admin Element Address:* The Destination Admin Element Address field shall be used to specify a particular Element of a HIPPI-6400 system that is the destination of an Admin micropacket command;
- *Admin Command:* The Admin Command field shall contain a value to specify the meaning and interpretation of the Admin micropacket. Table 7 contains all of the defined values, along with a description of the functions and parameters associated with each command;

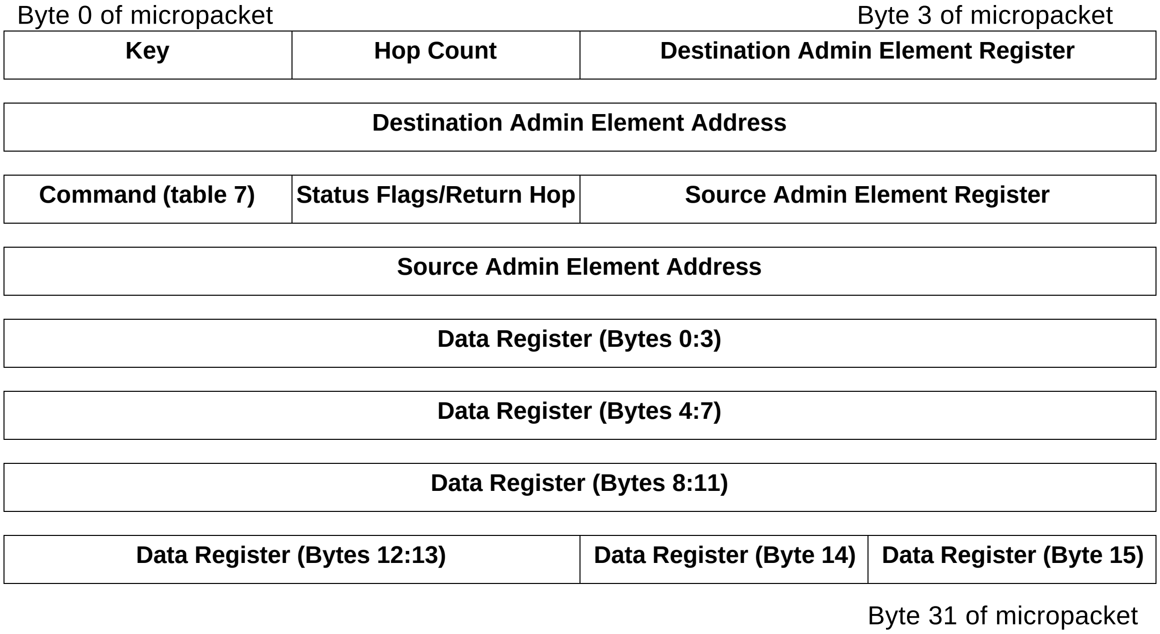
Table 5 - Admin micropacket Format

Byte	Function
0	Key
1	Hop Count
2:3	Destination Admin Register (designates a local register within an Element)
4:7	Destination Admin Element Address (Destination Element address in a HIPPI-6400 domain)
8	Admin Command (see table 7)
9	Status flags (see table 6) / Return Hop Count
10:12	Source Admin Register (designates a local register within an Element)
12:15	Source Admin Element Address (Source Element address in a HIPPI-6400 domain)
16:31	Data Register

- *Status Flags / Return Hop Count:* When the Admin micropacket is a command, the Return Hop Count field shall be used to communicate the proper hop count value for returning status. The Return Hop Count field may be set to x’FF’ when using Element addressing in lieu of a specific return distance. When the Admin micropacket is a response, the Status Flags field shall be used to return operation results. Table 6 pro-

Table 6 - Status flags

Bit	Meaning
0	Undefined Operation
1	Invalid Key
2	Parameter Out of Range
3	reserved
4	Invalid Register Address
5	Data Register not Valid
6	Unimplemented Command
7	Command Failed



Key	**Hop Count**	**Destination Admin Element Register**	
Destination Admin Element Address			
Command (table 7)	**Status Flags/Return Hop**	**Source Admin Element Register**	
Source Admin Element Address			
Data Register (Bytes 0:3)			
Data Register (Bytes 4:7)			
Data Register (Bytes 8:11)			
Data Register (Bytes 12:13)		**Data Register (Byte 14)**	**Data Register (Byte 15)**

Figure 6 - Admin micropacket byte format

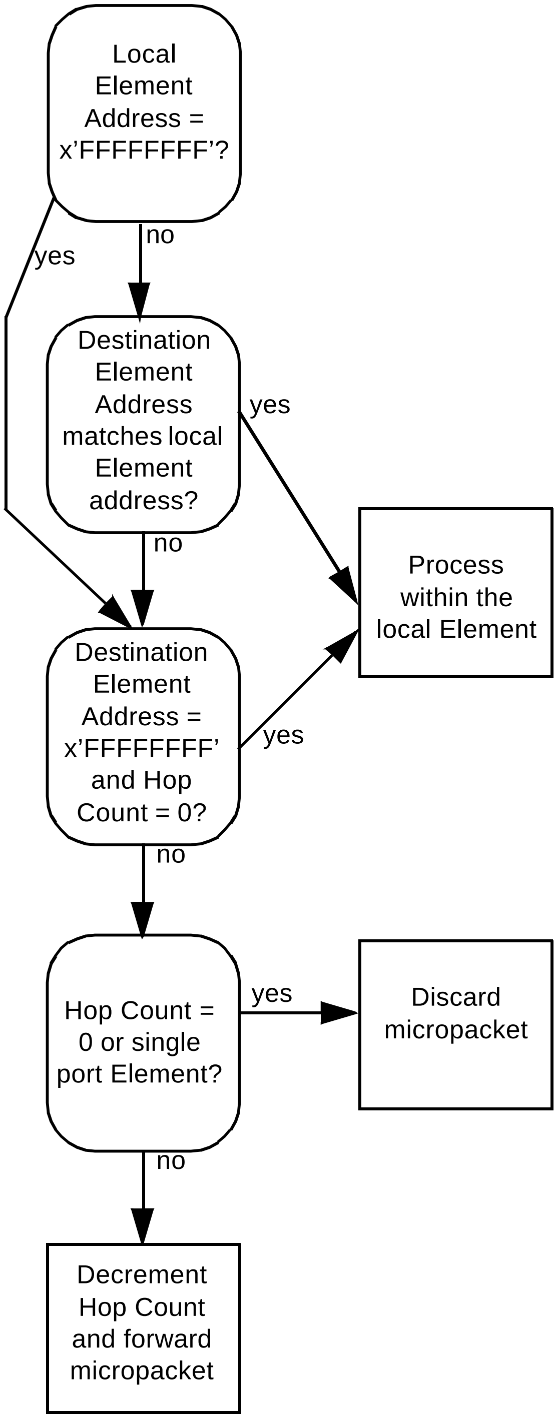

Local Element Address = x'FFFFFFFF'?
yes
no
Destination Element Address matches local Element address?
yes
no
Process within the local Element
Destination Element Address = x'FFFFFFFF' and Hop Count = 0?
yes
no
Hop Count = 0 or single port Element?
yes
Discard micropacket
no
Decrement Hop Count and forward micropacket

Figure 7 - Admin micropacket addressing

vides definitions for each bit. In each case, flag bit = 1 indicates that the listed exception has occurred;

- *Source Admin Register:* The Source Admin Register field may be used to specify a register within a HIPPI-6400 Element that can be used as a "reply-to" Element address for certain operations. There are no specific registers required in any Element by this standard;
- *Source Admin Element Address:* The Source Admin Element Address field is used to specify the particular Element of a HIPPI-6400 system that initiated a sequence of Admin packets. The source Admin Element address shall be used as a "reply-to" Element address;
- *Data Register:* The Data Register is a 16-byte field that shall be used to carry data for any Admin operation.

Unused fields shall be sent as zeros.

6.4 Admin micropacket commands and responses

Descriptions are provided for each of the Admin commands and responses. Some commands are described completely in the following paragraphs. Other commands are building blocks for functions that will be described in later clauses, such as ULA configuration and the broadcasting of Messages.

Commands may be inappropriate when issued in some circumstances. For example, requesting that a host change its switch addressing using the PORT_REMAP command is inappropriate because hosts are not switches. Elements shall respond to inappropriate commands with either the paired response for the command or the INVALID_COMMAND response, setting the Unimplemented Command flag or the Undefined Operation flag in the flag byte.

Each response shall set the appropriate status flags as specified in table 6.

6.4.1 PING

PING may be used to request a response micropacket for diagnostic validation. The Data Register field may be used to send data that will be echoed in the PING_RESPONSE.

The receiving Element shall return a PING_RESPONSE.

Table 7 - Admin commands and responses

Cmnd Value	Function	VC	Key Req'd?	Action	Implementation Required?	Typically Sent By
x'0'	PING	1	No	Asks for a PING_RESPONSE	No	An Element that collects status
x'1'	PING_RESPONSE	2	No	Acknowledges the PING command	Yes	Any Element
x'2'	SET_ELEMENT_ ADDRESS	1	Yes, except first time after reset	Set Admin Element address	No	An Element that configures and/ or controls other Elements
x'3'	SET_ELEMENT_ ADDRESS_ RESPONSE	2	Yes	Acknowledges the SET_ELEMENT_ ADDRESS command	No	An Element that is configured by an external Element
x'4'	RESET	1	Yes	Commands Element to initialize itself	No	An Element that configures and/ or controls other Elements
x'5'	EXCHANGE_ ELEMENT_ FUNCTION	1	No	Provides and requests Element Function	Yes for endpoints and switch Elements	Switches and endpoints
x'6'	ELEMENT_ FUNCTION_ RESPONSE	2	No	Response to a EXCHANGE_ ELEMENT_FUNCTION command	Yes	All Elements
x'7'	ULA_REQUEST	1	No	Requests a Source ULA	Yes for endpoints and switch Elements, not required for links	Switches and endpoints
x'8'	ULA_RESPONSE	2	No	Provides a Source ULA	Yes for switch Elements, not required for endpoints or links	Switches
x'9'	READ_REGISTER	1	Optional	The sender requests a register value	No	An Element that configures and/ or controls other Elements
x'A'	READ_REGISTER_ RESPONSE	2	No	Returns data from the requested register	No	An Element that is configured and/or controlled by other Elements
x'B'	WRITE_ REGISTER	1	Optional	Requests that a register value be updated	No	An Element that configures and/ or controls other Elements
x'C'	WRITE_ REGISTER_ RESPONSE	2	No	Status for a WRITE_REGISTER	No	An Element that is configured and/or controlled by other Element

Table 7 - Admin commands and responses

Cmnd Value	Function	VC	Key Req'd?	Action	Implementation Required?	Typically Sent By
x'D'	INVALID_ COMMAND	2	No	Response to an unrecognized or unimplemented command	Yes	Any Element
x'E'	ULA_LIST_ REQUEST	1	No	Asks for a list of connected ULAs	No	Broadcast server
x'F"	ULA_LIST_ RESPONSE	2	No	Provides a list of connected ULAs	Yes for switches	Switches
x'10'	PORT_REMAP	1	Yes	Changes the ULA to port routing for one input port	No	Broadcast server
x'11'	REMAP_RESPONSE	2	No	Status for a PORT_REMAP	Yes for switches	Switches
x'12'	PORT_MAP_ REQUEST	1	Optional	Gets the physical switch port used to send to a given ULA	No	Broadcast server
x'13'	PORT_MAP_ RESPONSE	2	No	Returns the physical switch port used to send to a given ULA	Yes for switches	Switches
x'14' - x'7F'	Reserved	N/A	N/A	Not defined	No	N/A
x'80'- x'FF'	Vendor defined	1/2	Optional	Optional action defined uniquely by vendor (commands must be sent on VC1, responses must be sent on VC2)	No	Vendor unique

Uniqueness of a particular PING_RESPONSE after a PING time-out can be established by sending different values in the Data Register.

6.4.2 PING_RESPONSE

PING_RESPONSE is the response to the PING command.

The receiving Element may use this response to validate that the PING'ed Element is operational. The Data Register field shall contain a copy of the data originally sent in the PING command.

6.4.3 SET_ELEMENT_ADDRESS

SET_ELEMENT_ADDRESS may be used to configure an Element with a specific Element address.

The use of Admin micropacket commands for Element address assignment is optional. No Element is required to assign Element addresses.

If this is the first SET_ELEMENT_ADDRESS command received after a reset, the value in the Key field shall be ignored. Later uses of the SET_ELEMENT_ADDRESS command shall validate that the Key field value matches the current key.

If the above criteria for key value are met, the receiving Element shall set its Admin Element address to be equal to the value set in the lower 4 bytes (12:15) of the Data Register field and shall set its key value to the new key provided in byte 8 of the Data Register. The provided key shall be retained for subsequent command validity checking. Once the Admin Element address is set, it shall not be changed without validating the key value or until the Element is reset.

The receiving Element shall respond with a SET_ELEMENT_ADDRESS_RESPONSE or an INVALID_COMMAND.

The action of this command will override previous usage of the command. Therefore, the SET_ELEMENT_ADDRESS may be safely reissued if a previous invocation timed-out without a response.

6.4.4 SET_ELEMENT_ADDRESS_RESPONSE

SET_ELEMENT_ADDRESS_RESPONSE is a response to the SET_ELEMENT_ADDRESS command.

If this response is sent by an Element capable of setting its Element address, the current valid key shall be returned in byte 8 of the Data Register field and the current Element address of this Element shall be returned in the lower 4 bytes (12:15) of the Data Register field. The current Element address and proper key value shall be returned regardless of the success or failure of the SET_ELEMENT_ADDRESS operation.

The use of Admin micropacket commands for Element address assignment is optional. An Element incapable of setting its Element address shall set the Unimplemented Command flag or the Undefined Operation flag in the flag byte of its response.

6.4.5 RESET

RESET shall cause an Element to initialize itself. This includes clearing the current Element address and key. It may also include other vendor unique functions and may not be the same as the actions caused by a HIPPI-6400 link reset or initialize.

RESET may be propagated further depending upon vendor specific implementation and configuration.

There is no response for a RESET. An INVALID_COMMAND shall not be sent when receiving a RESET, even if the command is unimplemented.

6.4.6 EXCHANGE_ELEMENT_FUNCTION

The sender provides its Element function value and Element broadcast configuration in byte (15) of the Data Register. Figure 8 shows the byte format.

The receiver shall respond with an ELEMENT_FUNCTION_RESPONSE.

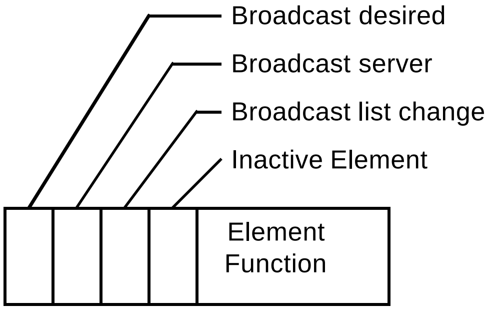


Figure 8 - Element function byte

Element function shall be one of the following in the lower four bits of byte (15):

- *Switch Element (b’0000’):* Used for switches;
- *Link-end Element (b’0001’):* Used when the Element is a link-end;
- *Non-switch Element (b’0010’):* Used for an endpoint Element; and
- *Unknown Element (b’0011’):* Any element not included in the other categories.

Unused Element function values are reserved.

The fourth most significant bit in the Element function byte (labeled in figure 8 as “inactive Element”) shall be used to indicate an inactive Element.

The upper three bits in the Element function byte shall be used to communicate broadcast parameters (see 8 and 9 for a description of broadcast functions). An inactive Element shall set the upper three bits in the Element function byte to zero. Active Elements shall set the broadcast configuration as follows:

- The most significant bit, if set to b’1’, shall indicate that this Element desires to receive broadcast Messages;
- The second most significant bit, if set to b’1’, shall indicate that this Element is able and willing to act as a broadcast server for this switch;
- The third most significant bit shall be set to b’0’ in the EXCHANGE_ELEMENT_FUNCTION (this bit is used in the ELEMENT_FUNCTION_RESPONSE).

Elements able and willing to act as a broadcast server for this switch shall issue this operation at least once per second and no more than twice per second.

The receiver of an EXCHANGE_ELEMENT_FUNCTION with the inactive Element bit shall

- invalidate any Source ULAs previously negotiated with the inactive Element;
- remove the inactive Element from eligibility for broadcast server functions; and
- end transmission of all HIPPI-6400 messages to the inactive Element.

Other bytes in the Data Register are defined as Vendor Unique and may be used in any way desired by the equipment provider.

This command may be safely reissued if it has timed-out without a response. Any response from the same Destination should provide identical results.

6.4.7 ELEMENT_FUNCTION_RESPONSE

ELEMENT_FUNCTION_RESPONSE is the response to the EXCHANGE_ELEMENT_FUNCTION command.

The sender shall return its Element function value and Element broadcast configuration in byte (15) of the Data Register. Figure 8 shows the byte format.

Element functions are specified in the EXCHANGE_ELEMENT_FUNCTION command.

If the Element function byte of the EXCHANGE_ELEMENT_FUNCTION specifies an inactive Element (indicated by setting the fourth most significant bit as shown in figure 8) the upper three bits in the Element function byte of the ELEMENT_FUNCTION_RESPONSE shall be set to zero.

If the Element function byte of the EXCHANGE_ELEMENT_FUNCTION does not specify an inactive Element, bits shall be returned in the ELEMENT_FUNCTION_RESPONSE as follows:

- The most significant bit of the Element function byte shall be echoed as received;
- The second most significant bit of the Element function byte shall be set to b’1’ if the switch has selected the receiver of this response to be the broadcast server for this switch. The bit will be set to b’0’ if the receiver of this response has not been selected to be the broadcast server for this switch;
- All Elements other than switches shall set the third most significant bit to b’0’. A switch shall set the third most significant bit to b’0’ to indicate that there is no change in the list of ports registered to receive broadcasts. If the list of ports registered to receive broadcasts has changed since the first port in the switch (port one) has been read by the current active broadcast server (see ULA_LIST_REQUEST/ ULA_LIST_RESPONSE), the switch shall set the third most significant bit to b’1’ and maintain this setting until a ULA_LIST_REQUEST from the current active broadcast server has been received for the first switch port.

The receiver of an ELEMENT_FUNCTION_RESPONSE with the inactive Element bit set shall

- invalidate any Source ULAs previously negotiated with the inactive Element;
- remove the inactive Element from eligibility for broadcast server functions; and
- end transmission of all HIPPI-6400 messages to the inactive Element.

Other bytes in the Data Register are specified as Vendor Unique and may be used in any way desired by the equipment provider.

6.4.8 ULA_REQUEST

The sender requests that the receiver return a HIPPI-6400 ULA.

The receiving Element shall respond with a ULA_RESPONSE or an INVALID_COMMAND.

The ULA_REQUEST command shall only be sent to switch Elements. If the request is made by an endpoint, the sender is requesting a Source ULA from the receiving switch. If the sender is a switch, the sender is requesting the ULA of the receiving switch.

Endpoint senders shall provide an offered base ULA in bytes (10:15) of the Data Register.

The most significant bit of byte 6 of the Data Register shall indicate that this is an additional request for an address and that previous addresses assigned to this port shall be retained as valid in addition to the address(es) assigned by this instance of the command. The balance of bytes (6:7) shall contain a count of desired addresses.

There are no parameters when this command is issued by a switch.

As specified in 7.2, this command is normally issued by endpoints or switches.

As long as the most significant bit of byte 6 of the Data Register is not set, the action of this command will override previous uses of the command. In this case, this command may be safely reissued after a time-out.

If the most significant bit of byte 6 of the Data Register is set and the operation times out, re-issuance of a timed out command could result in duplicate ULA registrations. When multiple ULAs are being requested with more than a single operation and a time-out occurs, the process shall be re-initiated and start with an operation that has the most significant bit of byte 6 of the Data Register not set.

6.4.9 ULA_RESPONSE

ULA_RESPONSE is a response to the ULA_REQUEST command.

If the original ULA_REQUEST was received from a switch, Bytes (10:15) of the Data Register shall contain the ULA of the switch sending the ULA_RESPONSE.

If the ULA_REQUEST was received from an endpoint, Bytes (10:15) of the Data Register shall contain a ULA for the receiver to use as a Source ULA. This may or may not be the offered Source ULA passed in the ULA_REQUEST command. For a receiver needing to add a single Source ULA, this value shall be directly utilized.

If the most significant bit of byte 6 is set, it indicates that the Source ULA(s) assigned shall be considered as additional to those assigned in a previous ULA_REQUEST/ULA_RESPONSE operation. For a receiver needing multiple Source ULAs, the balance of bytes (6:7) shall be used as a count of sequential ULAs that start at the base value contained in bytes (10:15) of the Data Register.

The first ULA registered on a port using the ULA_REQUEST/ULA_RESPONSE shall be the only Source ULA used for sending broadcast Messages from this port.

As specified in 7.2, this response is issued by switch Elements.

6.4.10 READ_REGISTER

The sender requests a value from the register specified in the Destination Admin Element Register.

The receiving Element shall respond with a READ_REGISTER_RESPONSE or an INVALID_COMMAND.

The use of Admin micropackets for register access is optional. If register access commands are supported, there are no requirements for particular functions or modes specified by this standard.

Contents of registers and their meaning are not specified in this standard.

Register reads should not be destructive (the registers should retain their values even after being read). This practice will allow re-reading of registers in the event of a READ_REGISTER time-out.

6.4.11 READ_REGISTER_RESPONSE

READ_REGISTER_RESPONSE is a response to the READ_REGISTER command.

The sender shall return the data from the requested register in the Data Register field.

- Single bytes are sent in byte (15);
- Two-byte words are sent in bytes (14:15);
- Four-byte words are sent in bytes (12:15);
- Eight-byte words are sent in bytes (8:15); and
- Sixteen-byte words sent in bytes (0:15).

The use of Admin micropackets for register access is optional. If register access commands are supported, there are no requirements for particular functions or modes specified by this standard. Elements incapable of supporting this operation shall set the Unimplemented Command flag or the Undefined Operation flag in the flag byte of their response.

Contents of registers and their meaning are not specified in this standard

6.4.12 WRITE_REGISTER

The sender requests that a register value be updated with the value contained in the Data Register.

- Single bytes are sent in byte (15);
- Two-byte words are sent in bytes (14:15);
- Four-byte words are sent in bytes (12:15);
- Eight-byte words are sent in bytes (8:15); and
- Sixteen-byte words are sent in bytes (0:15).

The receiving Element shall respond with a WRITE_REGISTER_RESPONSE or an INVALID_COMMAND.

The use of Admin micropackets for register access is optional. If register access commands are supported, there are no requirements for particular functions or modes specified by this standard. No Element is required to issue this command.

Contents of registers and their meaning are not specified in this standard.

Registers should be designed so that the WRITE_REGISTER command can be verified if the response times out. This can be accomplished by making writable registers readable with the READ_REGISTER command.

6.4.13 WRITE_REGISTER_RESPONSE

WRITE_REGISTER_RESPONSE is a response to the WRITE_REGISTER command.

The sender shall echo the value written to the specified Data Register. The contents of the Data Register shall be sent as zeros if the update was not successful.

The use of Admin micropackets for register access is optional. If register access commands are supported, there are no requirements for particular functions or modes specified by this standard. Elements incapable of supporting this operation shall set the Unimplemented Command flag or the Undefined Operation flag in the flag byte of their response.

Contents of registers and their meaning are not specified in this standard.

6.4.14 INVALID_COMMAND

INVALID_COMMAND is a response when an unrecognized or unimplemented command other than RESET is received on VC1.

INVALID_COMMAND is a possible response to

- SET_ELEMENT_ADDRESS;
- ULA_REQUEST;
- READ_REGISTER;
- WRITE_REGISTER;
- ULA_LIST_REQUEST;
- PORT_REMAP; and
- PORT_MAP_REQUEST.

Elements sending an INVALID_COMMAND shall set the Unimplemented Command flag or the Undefined Operation flag in the flag byte of their response.

Byte (15) of the Data Register shall contain the Admin command value from the micropacket that was the cause of this response.

6.4.15 ULA_LIST_REQUEST

ULA_LIST_REQUEST may be sent to Switch Elements to request information on whether attached HIPPI-6400 endpoints are registered to receive broadcast Messages and to learn the Source ULA that will be used for all broadcast Messages originated from the endpoint. The list also includes ULAs for communicating with attached switches and a ULA for the responding switch.

One request may be made to learn the status for each physical port of the switch or to learn the ULA of the responding switch.

The receiving Element shall respond with a ULA_LIST_RESPONSE or an INVALID_COMMAND.

The Data Register (byte 2:3) shall contain a number (0 for the switch or 1 thru *n*, corresponding to the physical port numbering) to identify which position in the list is being requested).

This command may be safely reissued if it has timed-out without a response. Any response from the same Destination should provide current results.

6.4.16 ULA_LIST_RESPONSE

ULA_LIST_RESPONSE is a response to a ULA_LIST_REQUEST.

Switches shall use this response to provide visibility into a sequentially organized list.

The list shall contain one entry for the responding switch (numbered as 0) and one entry for each physical port of this switch (numbered 1 thru *n*).

- For the responding switch, the ULA shall be the unique address assigned for use by the IEEE 802.1d Spanning Tree Algorithm and Protocol.
- For attached switches, the port ULA shall be the unique address assigned to the attached switch for use by the IEEE 802.1d Spanning Tree Algorithm and Protocol (learned by the responding switch through the switch-to-switch ULA_REQUEST/ULA_RESPONSE).
- For attached endpoints, the port ULA shall be the first ULA registered using the ULA_REQUEST/ULA_RESPONSE.

Byte (0) of the ULA_LIST_RESPONSE shall include the Element Function byte for the port, including the upper two bits in the Element Function used to communicate broadcast configuration parameters. The bit that signifies the broadcast server shall only be set for the port of the broadcast server and shall be zeroed for all other ports. Ports in the switch that are not configured for operation (but are within the 1 thru n range) as well as ports that have not performed both the EXCHANGE_ELEMENT_FUNCTION/ ELEMENT_FUNCTION_RESPONSE and ULA_REQUEST/ULA_RESPONSE shall return Byte (0) = b’00000011’.

Bytes (2:3) of the ULA_LIST_RESPONSE shall contain the list number copied from the ULA_LIST_REQUEST.

Bytes (10:15) of the Data Register shall contain the port ULA.

When access is attempted to list values not included in the list (past the end of the list), the Parameter Out of Range bit shall be set in the response. The Operation Failed bit shall not be set in this case.

Elements incapable of supporting this operation shall set the Unimplemented Command flag or the Undefined Operation flag in the flag byte of their response.

6.4.17 PORT_REMAP

PORT_REMAP may be sent to request a modification in the port mapping table used for selection of an output switch port (from the Destination ULA contained in a micropacket). The operation requests a new port mapping for a single ULA on one input port. Ports are numbered from 1 thru *n*, corresponding to the physical port numbering.

The receiving Element shall respond with a REMAP_RESPONSE or an INVALID_COMMAND.

Bytes (10:15) of the Data Register shall contain the Destination ULA being remapped.

Bytes (2:3) shall identify the input port being remapped.

Bytes (6:7) shall indicate the switch output port to be used when the specified Destination ULA is received on the specified switch input port. A zero value in bytes (6:7) shall signify that there is no valid port mapping for this ULA on the specified input port (i.e., Messages sent to this ULA on the specified input port shall be discarded).

The PORT_REMAP operation shall only be allowed for the port that has been selected as the broadcast server.

The action of this command will override previous usage of the command. Thus, this command may be safely reissued after a time-out.

6.4.18 REMAP_RESPONSE

REMAP_RESPONSE is a response to the PORT_REMAP command.

REMAP_RESPONSE shall be sent after the PORT_REMAP operation has been completed. This sequence (completing the action prior to sending the status) allows the PORT_REMAP initiator to determine when he may send a HIPPI-6400 Message and have it transmitted through the switch to the new desired output port.

All bytes of the Data Register shall echo the data sent in the PORT_REMAP operation.

If a request is made to remap a port that is physically not present in the switch, the Parameter Out of Range bit shall be set in the response and the command shall not be performed.

PORT_REMAP operations shall be rejected if initiated over any port other than the one that has been selected to be the broadcast server.

The Operation Failed bit shall be set in any case where the remapping operation fails or if the operation is rejected.

Elements incapable of supporting this operation shall set the Unimplemented Command flag or the Undefined Operation flag in the flag byte of their response.

6.4.19 PORT_MAP_REQUEST

PORT_MAP_REQUEST may be used to request the return of an entry in the port mapping table used to map a ULA to a physical port. The operation requests a port mapping for a single ULA on one input port.

The receiving Element shall respond with a PORT_MAP_RESPONSE or an INVALID_COMMAND.

Bytes (10:15) of the Data Register shall contain the requested Destination ULA mapping.

Bytes (2:3) shall identify the input port.

This command may be safely reissued if it has timed-out without a response. Any response from the same Destination should provide equivalent results.

6.4.20 PORT_MAP_RESPONSE

PORT_MAP_RESPONSE is a response to the PORT_MAP_REQUEST.

Switches shall use this response to return the physical port mapping that will be used for sending to a specified Destination ULA on an input port. Ports are numbered from 1 thru *n*, corresponding to the physical port numbering.

Bytes (2:3) of the Data Register shall echo the input port sent in the PORT_MAP_REQUEST operation.

Bytes (6:7) of the Data Register shall indicate the switch output port to be used when the Destination ULA is received on the specified switch input port. A zero value shall signify that there is no valid port mapping for this ULA on this input port (i.e., Messages sent to this ULA will be discarded).

Bytes (10:15) of the Data Register shall echo the Destination ULA sent in the PORT_MAP_REQUEST operation.

If a request is made for a ULA that is not mapped for the specified input port, the Parameter Out of Range bit shall be set in the response. The Operation Failed bit shall not be set in this case.

Elements incapable of supporting this operation shall set the Unimplemented Command flag or the Undefined Operation flag in the flag byte of their response.

6.4.21 Reserved Admin micropacket functions

Reserved Admin micropacket functions shall not be sent.

Receivers shall perform normal Element address processing and forwarding of Admin micropackets, regardless of the Function code.

Micropackets received for local processing with Reserved Function codes shall be responded to by an INVALID_COMMAND with the status flag set for an Unimplemented Command or an Undefined Operation.

6.5 Sending Admin micropackets

Admin micropackets shall be sent on the following VC specified for each command and response:

- Admin micropacket commands shall be sent on VC1;
- Admin micropacket responses shall be sent on VC2;

- Admin micropackets shall not be sent on VC0 or VC3.

6.6 Addressing of Admin micropackets

The Admin micropacket format contains a 32-bit source and destination Admin Element address. This space is adequate to uniquely identify Elements in configurations of up to 2^{32} Elements.

There are two possible destination Admin Element addresses that can result in delivery of an Admin micropacket to an Element for local processing:

- If the destination Admin Element address = x'FFFFFFFF' and hop count = 0. This technique allows access to neighbors (who may possibly have unknown Element addresses) by setting the hop count to control how far distant an Element is in hop count. For example, a hop count of three would pass through three neighboring Elements before being decremented to zero and being processed by the fourth Element;
- When the assigned Element address is not equal to x'FFFFFFFF and the assigned Element address matches the destination Admin Element address. This technique allows use of a flat logical address space for access to each Element when all of the Element addresses are known.

Response Admin micropackets shall use the source Admin Element address and return hop count provided in the original Admin micropacket command as the destination Admin Element address and hop count.

6.7 Processing Admin micropackets

With Elements that have two ports, a received Admin micropacket shall either be

- processed locally by the Element;
- discarded; or
- forwarded out the second port.

If a received Admin micropacket contains a valid Element address pointing to the current local Element, it shall be processed locally.

If a received Admin micropacket does not contain an Element address pointing to the current local Element, there are two possible results. If the hop count value is zero, the packet shall be discarded. Otherwise the hop count shall be decremented by one and the packet shall be forwarded to the Element's other port, i.e., the port that did not deliver this micropacket to this Element.

Response Admin micropackets shall be sent on the port that received the original Admin micropacket command.

Elements that have a single port shall discard Admin micropackets that are not addressed to be processed locally. Receivers of Admin micropackets shall only process and/or respond to Admin micropackets received on the specified proper VC.

- Admin micropackets received on VC0 or VC3 shall be logged as an error and discarded without a response;
- Admin micropackets received on VC1 shall be processed as a received command, discarded (due to an expired hop count), or forwarded (if the Element address does not match);
- Admin micropackets received on VC2 shall be processed as a received response, discarded (due to an expired hop count), or forwarded (if the Element address does not match). Responses that are received unexpectedly shall be logged as an error and discarded without a response. A response Admin micropacket shall never be sent in reply to an Admin micropacket received on VC2.

Admin micropackets that arrive with either ERROR = 1 or TAIL = 0 shall be logged as an error and discarded without a response.

Selection of the proper port for packet forwarding, from a set of ports in a multi-port Element, is not covered by this standard. Multi-port Element support is optional and may be added in a vendor unique manner.

6.8 Admin Element address assignment

Each Element in a HIPPI-6400 connected collection of Elements may be provided an Element address for operation and control. Element addresses may be assigned through any suitable means, including use of the commands, SET_ELEMENT_ADDRESS and SET_ELEMENT_ADDRESS_RESPONSE. These commands allow an intelligent system Element to assign Element addresses to other Elements within the configuration. Element addresses shall be assigned so that Element address duplication in the connected Element address environment does not occur.

Regardless of whether an Element address is assigned, each Element shall always respond to an Element address of x'FFFFFFFF' when hop count = 0.

This standard does not specify how the intelligent system Element chooses Element addresses for assignment. The discovery of topologies beyond two ports and the mechanisms for multi-port Element address assignment are not covered by this standard. Multi-port Element support is optional and may be added in a vendor unique manner.

6.9 Admin micropacket flow control

Admin micropacket operations (with the exception of RESET) consist of a command and a paired response operation. To avoid overrun of receivers, no more than one operation shall be initiated to a single destination Element from a single source Element. Therefore, Elements shall send only a single command:

- PING;
- SET_ELEMENT_ADDRESS;
- EXCHANGE_ELEMENT_FUNCTION;
- ULA_REQUEST;
- READ_REGISTER;
- WRITE_REGISTER;
- ULA_LIST_REQUEST;
- PORT_REMAP; or
- PORT_MAP_REQUEST;

before receiving the paired response micropacket:

- PING_RESPONSE;
- SET_ELEMENT_ADDRESS_RESPONSE;
- ELEMENT_FUNCTION_RESPONSE;
- ULA_RESPONSE;
- READ_REGISTER_RESPONSE;
- WRITE_REGISTER_RESPONSE;
- ULA_LIST_RESPONSE;
- REMAP_RESPONSE;
- PORT_MAP_RESPONSE;

or receiving an:

- INVALID_COMMAND response from the targeted Element;

or until:

- a time-out period of at least one second has elapsed.

Since RESET has no response, Elements that have sent a RESET shall wait at least one second before attempting any other operation to the Element that has been reset.

7 ULA configuration

In addition to switching HIPPI-6400 Messages between ports, HIPPI-6400 ports shall support in-band communications for switch management functions.

To support topology discovery and ULA configuration, HIPPI-6400 Destination ports shall be capable of receiving and processing micropackets with TYPE = Admin over any connected HIPPI-6400 link.

To support topology discovery and ULA configuration, HIPPI-6400 Source ports shall be capable of sending micropackets of TYPE = Admin over any connected HIPPI-6400 link.

7.1 Determination of connectivity

As a step in the procedure to establish a ULA for self identification (used as the Source ULA field), endpoints and switches shall identify if they are connected to another endpoint or to a switch.

Intervening link support hardware and interface Elements may be present on either side of a HIPPI-6400 link. These intermediate Elements will typically not contain information useful for ULA assignment. The endpoint discovering information shall identify these intermediate points to discover the location of an Element capable of exchanging information about ULA configuration.

Information about the function of connected Elements is collected by sending an EXCHANGE_ELEMENT_FUNCTION Admin micropacket. The endpoint may directly select a destination if the appropriate Admin Element address information is already known, or it may use hop-count Element addressing to discover what is connected and how far away (in hops) the Element of interest is located.

If an Element responds that it is a link-end Element or an unknown Element, the probing system shall continue to the next Element. Once a connected Element is identified as an endpoint or switch, topology determination is complete.

In figure 9, an example of an endpoint to endpoint link is shown. In this example, System A needs to determine the Element function of System B, for ULA configuration. System B also needs to determine the Element function of System A, for the same reason. The following example traces the operation of System A.

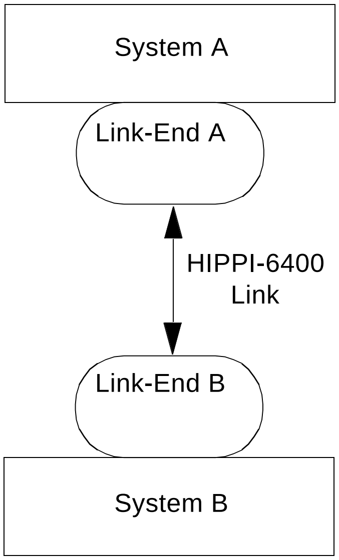


Figure 9 - Endpoint to endpoint connect

System A begins by probing each Element that supports Admin micropackets until it reaches the endpoint of System B:

a System A sends an EXCHANGE_ELEMENT_FUNCTION Admin micropacket to the closest point with an Element address of x'FFFFFFFF' and a hop-count of 0. This will be received and processed by Link-End A. Link-End A will respond in the ELEMENT_FUNCTION_RESPONSE Admin micropacket that it is a link-end Element. System A must therefore go further to reach another endpoint or switch;

b System A sends an EXCHANGE_ELEMENT_FUNCTION Admin micropacket to the next closest point with an Element address of x'FFFFFFFF' and a hop-count of 1. This will be received and processed by Link-End B. Link-End B will respond in the ELEMENT_FUNCTION_RESPONSE Admin micropacket that it is a link-end Element. System A must therefore go further to reach another endpoint or switch;

c System A sends an EXCHANGE_ELEMENT_FUNCTION Admin micropacket to the 3rd closest point with an Element address of x'FFFFFFFF' and a hop-count of 2. This will be received and processed by System B. System B will respond in the ELEMENT_FUNCTION_RESPONSE Admin micropacket that it is an endpoint. System A now knows where to exchange information regarding ULAs.

In the above example, System A presumably would have been aware that the most directly attached component (Link-End A) is part of its own configuration and that it need not communicate with that component. It therefore would not have needed to start with a hop-count of 0 (but was not detrimentally effected by doing so).

System B could determine the type of Element for System A in two ways:

- System B could duplicate the above steps in reverse; or
- System B could use the information provided in the EXCHANGE_ELEMENT_FUNCTION command that System A sent to System B when the third step in the above exchange took place. Endpoints should not wait for the other end to perform an exchange, but if the exchange occurs at an appropriate time, they may take advantage of the occurrence.

7.2 ULA exchange

Once the other end of the link has been identified as to type (switch or non-switch endpoint), ULAs are configured.

7.2.1 Endpoints on both ends

If both ends of the link are endpoints, each side shall use the Source ULA assigned to it using the procedures specified in the IEEE 802 Overview Standard.

7.2.2 Switches on both ends

Switch to switch ULA configuration shall occur to exchange ULAs for the broadcast function. A ULA_REQUEST shall be sent by each switch to all directly connected switches. Upon receipt of a ULA_REQUEST Admin micropacket, the receiving switch shall respond with a ULA_RESPONSE Admin micropacket. The ULA_RESPONSE shall contain a unique ULA assigned to the responding switch for use by the 802.1d Spanning Tree Algorithm and Protocol.

Switch to switch ULA discovery to learn the full set of connected ULAs on distant switches is handled outside of this standard but shall ensure the uniqueness of ULAs within the fabric. Methods of switch configuration could include static manual table entry or automated ULA learning algorithms.

7.2.3 Endpoint to switch

If endpoints discover that they are connected to switches, they shall advertise their Source ULA (assigned to the endpoint using the procedures specified in the IEEE 802 Overview Standard). The ULA offer shall be made by sending a ULA_REQUEST Admin micropacket.

Upon receipt of a ULA_REQUEST Admin micropacket, the receiver shall respond with a ULA_RESPONSE Admin micropacket. The ULA_RESPONSE shall contain a Source ULA valid for the ULA_RESPONSE recipient. This ULA may be the same as advertised in the original ULA_REQUEST offer or it may be different.

This returned Source ULA shall be accepted and subsequently used in all HIPPI-6400 Messages by the receiver of the ULA_RESPONSE Admin micropacket.

Regardless of whether the returned Source ULA is the same as the Source ULA originally offered by the endpoint, the switch is the final selector of the Source ULA that will be used by the endpoint.

A single Source ULA shall not be assigned more than once in the same fabric.

Switches shall wait for connected endpoints to initiate ULA exchange.

8 Broadcast/multicast

All switches shall either directly support the broadcast/multicast of Messages or shall provide support for a HIPPI-6400 broadcast server that is connected to the switch by a HIPPI-6400 link.

8.1 Broadcast/multicast operation

Messages sent to the broadcast address (see table 8) shall be delivered to all endpoints within a HIPPI-6400 fabric that have registered their desire to receive broadcasts and multicasts. Unless support is provided for multicast groups, messages sent to multicast addresses listed in table 8 shall be delivered to all endpoints within a HIPPI-6400 fabric that have registered their desire to receive broadcasts and multicasts.

This standard does not provide support for multicast groups. Note that unless multicast group support is otherwise provided, a HIPPI-6400 switch fabric treats broadcast and multicast in an identical manner. Thus, multicast Messages are delivered to all participating systems connected to the HIPPI-6400 fabric.

8.2 Supported broadcast and multicast ULAs

Table 8 shows the minimum set of broadcast and multicast ULAs that shall be supported.

These addresses shall be supported in all switches, regardless of their use of full or partial ULA processing.

Other addresses may be supported for broadcast and multicast.

Table 8 - Supported broadcast and multicast ULAs

ULA Hex Format *IEEE Canonical Format*	**Function**
x'FFFFFFFFFFFF' *FF-FF-FF-FF-FF-FF*	General broadcast address
x'0180C2000000' *80-01-43-00-00-00*	802.1d bridge group address
x'0180C2000001' thru x'0180C200000F' *80-01-43-00-00-80 thru 80-01-43-00-00-F0*	Reserved for future 802.1d standardization
x'0180C2000010' *80-01-43-00-00-08*	All LANs bridge management group address

8.3 Registration for broadcast and multicast

Attached endpoints and switches may register to receive broadcasts and multicasts. This shall be done by setting the most significant bit of the Element function byte to b'1' in the EXCHANGE_ELEMENT_FUNCTION operation. Attached endpoints and switches may choose not to receive broadcast/multicast Messages. This shall be done by setting the most significant bit of the Element function byte to b'0' in the EXCHANGE_ELEMENT_FUNCTION operation.

Switches shall maintain a list of ports. This list shall include one entry with a ULA and an element function byte for

- each endpoint directly connected to this switch that has made at least one ULA_REQUEST; and
- each switch directly connected to this switch that has provided its unique ULA (via the ULA_REQUEST/ULA_RESPONSE process) for the IEEE 802.1d Spanning Tree Algorithm and Protocol.

Endpoints registered to receive broadcasts and multicasts shall be sent any broadcast/multicast Message regardless of its point of origin. Directly connected switches shall be sent broadcast/multicast Messages in accordance with the 802.1d Spanning Tree Algorithm and Protocol.

8.4 Spanning tree operation

Switches shall participate in the IEEE 802.1d Spanning Tree Algorithm and Protocol, either directly or through an external broadcast server. This algorithm constructs a loop-free topology (called the spanning tree) by placing selected links in the network in the forwarding state and non-selected links in the blocking state for the purposes of broadcast/multicast. To avoid broadcast loops, switches shall propagate broadcast/multicast Messages only along those links which are placed in the forwarding state by the Spanning Tree Algorithm and Protocol.

To construct the spanning tree, switches exchange IEEE 802.1d configuration Bridge Protocol Data Units (BPDUs) which contain parameters (e.g. root ID, path cost, port identifier) for use by the spanning tree algorithm. The periodic exchange of BPDUs both configures the initial spanning tree and reconstructs the spanning tree in the event of switch failure(s) and/or the addition of new equipment to the network.

9 Broadcast emulation

Switches not capable of directly supporting broadcast shall route broadcast and multicast Messages to a broadcast server. The endpoint selected as a broadcast server shall forward received broadcast/multicast Messages to each attached endpoint port that has registered to receive them. Additionally, endpoints selected as broadcast servers shall implement the IEEE 802.1d Spanning Tree Algorithm and Protocol and shall forward broadcast/multicast Messages to those directly connected switches whose links have been placed in the forwarding state.

When sending Messages to implement the IEEE 802.1d Spanning Tree Algorithm and Protocol, the broadcast server shall use the Source ULA of the attached switch (the switch that the endpoint is supporting as a broadcast server). This ULA shall be learned by performing the ULA_LIST_REQUEST with a port number of zero.

9.1 Selection of broadcast server

Non-broadcast capable switches shall select a broadcast server from attached hosts who have indicated their ability and willingness to perform the broadcast server function.

The indication of qualified broadcast servers is provided by an EXCHANGE_ELEMENT_FUNCTION operation with the second most significant bit of the Element function byte set to b’1’.

One server shall be selected per non-broadcast capable switch. The server shall be notified by returning the second most significant bit of the Element function byte set to b’1’ in the ELEMENT_FUNCTION_RESPONSE.

The EXCHANGE_ELEMENT_FUNCTION and ELEMENT_FUNCTION_RESPONSE shall be exchanged at a rate of one to two per second. This continued exchange allows selection of a broadcast server as needed to deal with equipment failures and to accommodate added or removed systems. If a broadcast server fails to provide a EXCHANGE_ELEMENT_FUNCTION within 5 seconds, the switch shall select a new broadcast server.

9.2 Broadcast server configuration

Switches shall make available a list of broadcast information through the ULA_LIST_REQUEST and ULA_LIST_RESPONSE operations. Hosts selected to be broadcast servers shall request this list and use it to determine which ports should receive broadcast Messages and to learn the required Source ULA for IEEE 802.1d Spanning Tree Algorithm and Protocol Messages.

The ELEMENT_FUNCTION_RESPONSE received from the switch indicates when broadcast information available via the ULA_LIST_REQUEST message has changed. The ELEMENT_FUNCTION_RESPONSE notification of a list change is cleared after a ULA_LIST_REQUEST (see 6.4.15) has been made for the first port of the switch. Each time the ELEMENT_FUNCTION_RESPONSE indicates the list has changed, the broadcast server shall update its list of broadcast/multicast ports by requesting the entire list in sequential order starting with port one.

The broadcast server shall configure the switch ULA mapping using PORT_REMAP Admin micropackets. All Messages with the broadcast and multicast ULAs, unless being sent by the broadcast server, shall be delivered to the broadcast server. This requires mapping each broadcast address, on each input port, to point to the broadcast server’s own physical port. The broadcast server shall also configure the switch mapping to deliver Messages with the Destination ULA of the switch to itself in support of the IEEE 802.1d Spanning Tree Algorithm and Protocol.

Broadcast server configuration shall be performed once each time a broadcast server is selected and as needed whenever the broadcast configuration information is changed.

9.3 Sending broadcast Messages

The broadcast server shall send broadcast Messages sequentially. For each destination, the broadcast server shall configure its own port mapping so that the particular broadcast address ULA points to the desired physical port. The process is:

a send a PORT_REMAP admin micropacket;

b wait for a PORT_REMAP_RESPONSE micropacket;

c send the broadcast Message.

The process shall be repeated for each endpoint physical port registered to receive broadcasts or for each directly connected switch needing to receive the Message as required by the 802.1d Spanning Tree Algorithm and Protocol.

10 Configuration sequence

Configuration shall be performed prior to sending any HIPPI-6400 messages when local configuration information is unknown and/or the link has transitioned from a non-operating state.

Configuration information should be invalidated when a link has failed.

There are several initialization and/or configuration processes specified in this standard. These include

- Element configuration;

- Source ULA assignment;
- broadcast registration; and
- broadcast server selection and configuration.

Typically, Element configuration is performed first in the configuration sequence. However, Element configuration is a vendor unique function and could occur at a different point or be omitted from the configuration sequence.

Once any needed Element configuration is complete, Source ULAs are established, broadcast registration is performed, and, if needed, broadcast server selection is performed. The first step in the Source ULA process (exchanging the EXCHANGE_ELEMENT_TYPE and the ELEMENT_TYPE_RESPONSE operations) is also the mechanism for broadcast registration and broadcast server selection. Thus, broadcast registration and broadcast server selection can be completed concurrently with establishing a Source ULA.

Broadcast registration and broadcast server selection can be modified by subsequent EXCHANGE_ELEMENT_TYPE/ ELEMENT_TYPE_RESPONSE sequences. This ability to modify earlier operations allows separating ULA establishment from broadcast registration and broadcast server selection if it is desired.

Source ULA configuration may also be modified. Additional Source ULAs may be requested, either adding to or replacing the current Source ULA(s).

Further, all configuration information between two Elements shall be undone by a new EXCHANGE_ELEMENT_TYPE/ ELEMENT_TYPE_RESPONSE sequence with the inactive Element bit set (see figure 8). This action shall invalidate any active Source ULAs, reset broadcast registration, and remove broadcast server registration configured on the link. One example of when the reset of configuration information may be desired is when host shuts down due to operational needs or because of system failure.

Annex A
(informative)

Switching

A.1 General

HIPPI-6400 switching of Messages is accomplished by processing the Destination ULA field of the HIPPI-6400-PH MAC header. This may be done based on the complete contents of the Destination ULA (48 bits) or on a subset of the field.

If a subset of the Destination ULA is used for switching, switches must ensure that Source ULAs are unique in the portion of the ULA operated on by the switch. Clause 7 describes the process of ULA configuration that gives switches final authority in configuration of Source ULAs.

When connections are made to other networks, the address range of the two (or more) networks is limited by the smaller of the connected address ranges.

For example, HIPPI-PH can be switched to communicate with HIPPI-6400 so long as all of the communicating systems restrict their addresses to 12 bits. The total number of addresses is therefore limited to 4096 (minus reserved addresses).

The Destination ULA field in the Header micro-packet is used to control HIPPI-6400 physical layer switches, supporting the interconnection of many Devices. Figure 10 shows an example configuration that will be used to describe how HIPPI-6400 switches function. Three hosts and two switches are shown, actual configurations may be smaller or larger.

Although there is only a single mode of operation (ULA addressing) specified for HIPPI-6400, users can achieve a form of source routing (as described in HIPPI-SC) by their selection of port configuration.

A.2 Logical addressing

With logical addressing, ULAs specify where a Message is to be delivered, not the path to take to get there. Originating Sources use the same ULA to reach a Final Destination, no matter where the Originating Source is located.

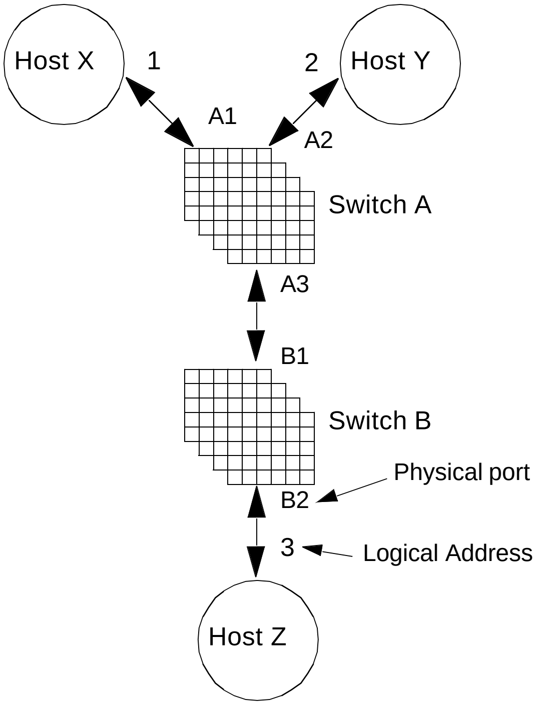


Figure 10 - Hosts and switch configuration

In figure 10, Host X, Host Y, and even Host Z can use ULA “3” to specify that a Message should be sent to Host Z.

With ULAs, the intermediate switches are responsible for selecting an appropriate path.

It is envisioned that switches can be built to use look-up tables at each input port to map ULAs to Destinations. A look-up table can be indexed using the Destination ULA field. The look-up table would be used to hold a possible path(s) for a Destination.

A major advantage of using ULAs is that only the switches need to know the fabric interconnection topology and the hosts only need to know the ULAs. Hence if a link or port fails, switches can address around it without the hosts having to know about it or do anything special.

A.3 Input specific logical addressing

Because each input port is specified to contain a unique ULA look-up capability, it is possible to use logical switch addressing for limited source routing. Note that only the input portion of a port is involved in addressing. When a Message exits on a particular output port, it crosses that link without further addressing until received at the next input.

This capability means that it is possible to create addressing that could result in infinite looping of a micropacket. This will rarely be desirable and should be avoided.

One possible use of input port specific routing is to provide a test capability for monitoring the performance of specific links. In figure 10, if Host Y wants to monitor the state of the link between switch A and switch B, he can send a Message to switch A and then to switch B. Port B1's ULA table (at switch B) can direct the Message back to B1, then switch A, and back to Host Y. To do this, the same ULA must be handled differently by individual ports. Table 9 shows a simplified look-up table that would work in this example.

Table 9 - Port look-up table

ULA	Port Number	Destination
2	A2	A3
2	B1	B1
2	A3	A2

Because there are many available ULAs, normal flat addressing can be used for host communications with other ULAs used to support input specific logical routing for test and monitoring purposes.