HIGH-PERFORMANCE PARALLEL INTERFACE - 6400 Mbit/s Physical Layer

(HIPPI-6400-PH)

April 5, 1996

Secretariat:

Information Technology Industry Council (ITI)

ABSTRACT: This standard specifies a physical-level, point-to-point, full-duplex, link interface for reliable, flow-controlled, transmission of digital data at 6400 Mbit/s, per direction, over parallel copper cables across distances of TBD m, or over parallel fiber-optic cables across distances of TBD m. Small fixed-size micro-packets provide an efficient, low-latency, structure for small messages, and a building block for large messages. Services are provided for transporting data associated with other HIPPI protocols.

NOTE:

This is an internal working document of X3T11, a Technical Committee of Accredited Standards Committee X3. As such, this is not a completed standard. The contents are actively being modified by X3T11. This document is made available for review and comment only. For current information on the status of this document contact the individuals shown below:

POINTS OF CONTACT:

Roger Cummings (X3T11 Chairman)
Distributed Processing Technology
140 Candace Drive
Maitland, FL 32751
(407) 830-5522 x348, Fax: (407) 260-5366
E-mail: cummings_roger@dpt.com

Carl Zeitler (X3T11 Vice-Chairman)
IBM Corporation, MS 9440
11400 Burnet Road
Austin, TX 78758
(512) 838-1797, Fax: (512) 838-3822
E-mail: zeitler@ausvm6.vnet.ibm.com

Don Tolmie (HIPPI-6400-PH Technical Editor)
Los Alamos National Laboratory
CIC-5, MS-B255
Los Alamos, NM 87545
(505) 667-5502, Fax: (505) 665-7793
E-mail: det@lanl.gov

Comments on Rev 0.2, April 5, 1996

This is a preliminary document undergoing lots of changes. Many of the additions are just place holders, or are put there to stimulate discussion. Hence, do not assume that the items herein are correct, or final – everything is subject to change. This page tries to outline where we are; what has been discussed and semi-approved, and what has been added or changed recently and deserves your special attention. This summary relates the changes since the previous revision, and is done in lieu of margin bars, which would be too numerous in this preliminary state of the document. Also, previous open issues are outlined with a single box, new ones are marked with a double bar on the left edge of the box.

Please help us in this development process by sending comments, corrections, and suggestions to the Technical Editor, Don Tolmie, of the Los Alamos National Laboratory, at det@lanl.gov. If you would like to address the whole group working on this document, send the message to hippi-ext@think.com.

1. Figure 2 showing the micro-packet format was changed was changed for clarity and the bits and bytes renumbered.
2. In 6.1, the placement of the ECRC and LCRC fields was swapped.
3. In 6.3, the micro-packet TYPEs were expanded and enumerated. Table 2 was updated, and an Admin type was added in anticipation of its being needed.
4. The text in 6.6.1, 6.6.2, and 6.6.3 is largely new and unchecked.
5. All of 6.7 about the Header micro-packet contents is new and unchecked.
6. All of clause 7 about transfers over VC3 is new. It is based on the HIPPI-800 mapping proposal by Don Tolmie that was presented at the March meeting.
7. 8.4 about which fields remain constant for re-transmitted micro-packets is new and unchecked.
8. The pseudo-code in 9.1 for error checking is new and unchecked.
9. The bit numbers in tables 7 and 8 was changed to reflect the nomenclature in figure 2.
10. The nomenclature for the 5-bit code was changed to wxTyz for clarity, and had not been checked.
11. The micro-packet waveforms in figures 6 and 7 are new and have not been checked.
12. The training sequences in figures 8 and 9 are based on those presented by Hansel Collins at the March meeting. They have not been checked.
13. The dynamic skew compensation text in 11 is new and had not been checked.
14. A new clause 14 was added for Timing. The stuff in 14.1 is a place holder to see how we want to handle the text in a general manner.
15. A place holder Annex A is attached. It will eventually describe the HIPPI-800 to HIPPI-6400 mapping.
16. There are lots of little nits that have been fixed, or added, or changed.

Contents

Tables

Figures

Annexes

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

This American National Standard specifies a physical-level, point-to-point, full-duplex, link interface for reliable, flow-controlled, transmission of digital data at 6400 Mbit/s, per direction, over parallel copper cables across distances of TBD m, or over parallel fiber-optic cables across distances of TBD m. Small fixed-size micro-packets provide an efficient, low-latency, structure for small messages, and a building block for large messages. Services are provided for transporting data associated with other HIPPI protocols.

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 X3 Secretariat, Information Technology Industry Council, 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 X3 Committee had the following members:

> (List of X3 Committee members to be included in the published standard by the ANSI Editor.)

Subcommittee X3T11 on Device Level Interfaces, which developed this standard, had the following participants:

> (List of X3T11 Committee members, and other active participants, at the time the document is forwarded for public review, will be included by the Technical Editor.)

Introduction

This American National Standard specifies a physical-level, point-to-point, full-duplex, link interface for reliable, flow-controlled, transmission of digital data at 6400 Mbit/s, per direction, over parallel copper cables across distances of TBD m, or over parallel fiber-optic cables across distances of TBD m. Small fixed-size micro-packets provide an efficient, low-latency, structure for small messages, and a building block for large messages. Services are provided for transporting data associated with other HIPPI protocols.

Open Issue - Are characteristics correct ? Complete ?

Characteristics of a HIPPI-6400-PH physical-layer interface include:

– A data transfer bandwidth of 6400 Mbit/s (800 MByte/s).

– A full-duplex link capable of independent full-bandwidth transfers in both directions simultaneously.

– Four virtual circuits providing a limited multiplexing capability.

– A fixed size transfer unit, i.e., a 32-byte micro-packet, for hardware efficiency.

– A small transfer unit resulting in low latency for short messages, and a building block for large messages.

– Credit-based flow control that prevents buffer overflow.

– End-to-end, as well as link-to-link, check sums.

– Automatic retransmission of errored data providing guaranteed, in-order, error free, data delivery.

– An ac coupled parallel electrical interface for limited distance applications, and a parallel fiber-optic interface for longer distances.

– Support for carrying legacy HIPPI-800 and HIPPI-1600 traffic.

American National Standard
for Information Technology –

High-Performance Parallel Interface – 6400 Mbit/s Physical Layer (HIPPI-6400-PH)

1 Scope

This American National Standard specifies a physical-level, point-to-point, full-duplex, link interface for reliable, flow-controlled, transmission of digital data at 6400 Mbit/s, per direction, over parallel copper cables across distances of TBD m, or over parallel fiber-optic cables across distances of TBD m. Small fixed-size micro-packets provide an efficient, low-latency, structure for small messages, and a building block for large messages. Services are provided for transporting data associated with other HIPPI protocols.

Specifications are included for:

– automatic retransmission of errored data;

– the format of a small data transfer unit called a micro-packet;

– a message structure that includes routing information for network applications;

– end-to-end, as well as link-to-link, check sums;

– the timing requirements of the parallel signals;

– a parallel interface using copper coaxial cable;

– a parallel interface using ribbon fiber-optic cable.

2 Normative references

The following American National Standard contains provisions which, through reference in this text, constitute provisions of this American National Standard. At the time of publication, the edition indicated was 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.

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.xxx-199x, *High-Performance Parallel Interface, 6400 Mbit/s Switch Control (HIPPI-6400-SC).*

3 Definitions and conventions

3.1 Definitions

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

Open Issue - Agreement on definitions to date

3.1.1 bit error rate (BER): The statistical probability of a transmitted bit being erroneously received in a communication system. The BER is measured by counting the number of erroneous bits at the output of the receiver and dividing by the total number of bits.

3.1.2 credit: A credit corresponds to one micro-packet's worth of buffer space available in the Destination's VC buffer.

3.1.3 Destination: The equipment that receives the data.

3.1.4 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.

3.1.5 HIPPI port: A HIPPI-6400-PH, or HIPPI-PH, Source or Destination.

3.1.6 link: A full-duplex connection between HIPPI-6400-PH devices.

3.1.7 micro-packet: The basic transfer unit consisting of 32 data bytes and 64 bits of control information.

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

3.1.9 Source: The equipment that transmits the data.

3.1.10 virtual channel (VC): One of four paths within a link.

3.1.11 virtual connection: A logical connection over VC3 between a Source and Destination.

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 the 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 11000100 00000011 is shown in hexadecimal format as x'C403'.

3.3 Acronyms and other abbreviations

ACK	acknowledge
CR	credit amount parameter
CRC	cyclic redundancy check
ECRC	end-to-end CRC
HIPPI	High-Performance Parallel Interface
LCRC	link CRC
ns	nanoseconds
RSEQ	receive sequence number
TSEQ	transmit sequence number
VC	virtual channel
VCR	virtual channel CRedit selector
μs	microseconds
Ω	ohms

4 System overview

This clause provides an overview of the structure, concepts, and mechanisms used in HIPPI-6400-PH.

4.1 Links

HIPPI-6400-PH defines a point-to-point physical link for transferring micro-packets. The physical links, as shown in figure 1, are bi-directional. The logical links are simplex, i.e., the data inbound and outbound are completely separate. Some control information, e.g., credit, flows in the reverse direction, and it is included in the micro-packets flowing in the reverse direction. This is why the physical links must be bi-directional with information flowing in both directions simultaneously.

A link is composed of two Sources that transmit information, and two Destinations that receive information. Each end of a link has a Source and a Destination.

The data path is 16 bits wide for a copper implementation, and is 16 (or fewer) bits wide for a fiber implementation. The control path is one-fourth the width of the data path, e.g., the control path for a copper implementation would be 4 bits wide. CLOCK and FRAME are individual signals carried on separate conductors.

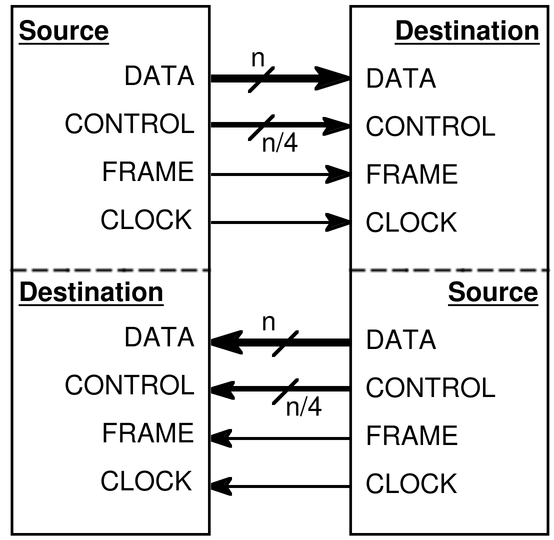


Figure 1 – HIPPI-6400-PH link showing signal lines

Open Issue – Should figure 1 show n and n/4 lines as the general case, or just use 16/8 and 4/2 since those are the widths we are specifying in detail?

4.2 Virtual channels

Four virtual channels, VC0,VC1,VC2, and VC3 are available in each direction on each link. The VCs are assigned to specific message sizes and connection methods. All of the micro-packets of a message are transmitted on a single VC, i.e., the VC number does not change as the micro-packets travel from the original Source to the final Destination over one or more links. The VCs provide a multiplexing mechanism which can be used to prevent a large message from blocking a small high-priority message until the large message has completed.

4.3 Micro-packet

Micro-packets are the basic transfer unit. As shown in figure 2 a micro-packet is composed of 32 data bytes and 64 bits of control information. At 6400 Mbit/s a micro-packet is transmitted every 40 ns. Credit and retransmit operations are done on micro-packets.

The 64 bits of control information in each micro-packet includes parameters for:

– selecting a VC;

– detecting missing micro-packets;

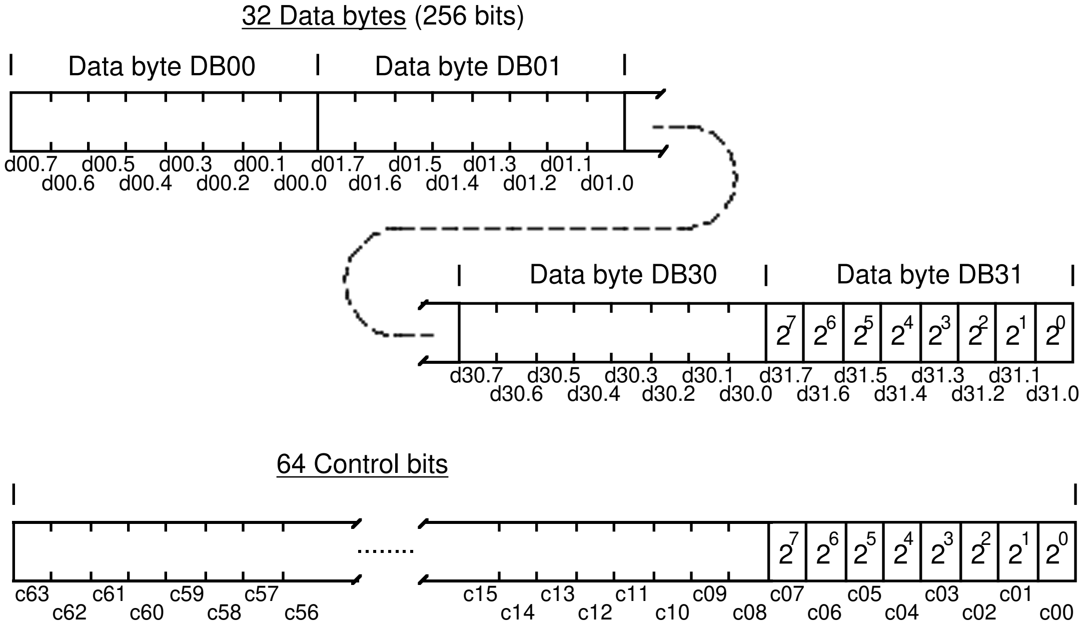
– denoting the types of information in the micro-packet;

– marking the last micro-packet of a message;

– signalling that the message was truncated at its originator, or damaged en-route, and should be discarded;

– passing credit information from the Destination to the Source;

– Link-level and end-to-end checksums.



Naming conventions:

Data bytes are labeled capital DB and a two-digit number, e.g., DB00
Data bits are labeled lower case d, a two-digit byte number, and a one-digit bit number, e.g., d31.7
Control bits are labeled lower case c and a two-digit number, e.g., c00

Figure 2 – Logical micro-packet format and naming conventions

4.4 Message

As shown in figure 3, messages are contiguous groups of micro-packets which have the same VC. The first micro-packet of a message, i.e., the Header micro-packet, contains information used to route through a HIPPI-6400 fabric. The last micro-packet of the message is marked with the TAIL bit.

Micro-packet Transmission order		
1	Routing information	c63–c00
2	1st 32 bytes of mesage data	c63–c00
3	2nd 32 bytes of message data	c63–c00
⋮	⋮	
n	Last bytes of message data	c63–c00

Figure 3 – Message format

4.5 FRAME and CLOCK signals

The FRAME signal, carried on a separate signal line, marks the beginning of micro-packets. Both edges of the CLOCK signal, also carried on a separate signal line, are used for strobing the data. The data, control, and FRAME signals from a Source are synchronous with that Source's CLOCK signal. The CLOCK rate is dependent on the width of the data bus, e.g., a 16-bit data bus utilizing 4b/5b encoding requires the CLOCK line to run at 250 MHz and each data and control line may transition every 2 ns.

4.6 Flow control

Credit-based flow control is used. As shown in figure 4 the credits are assigned on a VC basis, i.e., VC0's credits are separate from VC1's credits. The Destination end of a link grants credits to match the number of free receive buffers for a particular VC. The Source consumes credits as it moves micro-packets from the VC Buffers to the Output Buffer. Note that flow control is on a link basis, i.e., hop-by-hop.

4.7 Retransmission

Go-back-N retransmission is used. The link-level CRC of each micro-packet is checked at the Destination side of a link; at the Input Buffer in figure 4. Correct micro-packets are acknowledged, incorrect micro-packets are discarded. Retransmission of incorrect micro-packets is automatic. Note that retransmission is independent of the VC used, and also independent of the credit information, i.e., retransmission occurs between the Output and Input Buffers in figure 4 while VC and credit information pertains only to the VC Buffers. Retransmission is on a link basis, i.e., hop-by-hop.

Open Issue – Should both the LCRC and ECRC be checked at each node?

4.8 Check functions

As shown in table 1, two 16-bit cyclic redundancy checks (CRCs) are used, and they use different polynomials. The end-to-end CRC (ECRC) covers just the data portion of the micro-packets, and is unchanged from the original Source to the final Destination. The ECRC is accumulated over an entire message, i.e., it is not re-initialized for Data micro-packets.

Open Issue – Do the LCRC and ECRC use different polynomials?

The link CRC (LCRC) covers all of the data and control bits of a micro-packet, with the exception of itself and the ECRC. The LCRC is initialized for each micro-packet, and must be calculated fresh for each link since other control fields change.

4.9 Copper physical layer (optional)

Open Issue - Length of coax interface.

Open Issue - Overall shield on coax cable?

Open Issue - AC coupling to coax, e.g., transformer, capacitor

Open Issue – Is micro-coax used, or micro-twinax?

Open Issue – Is it a single cable for full-duplex, or two cables?

The optional HIPPI-6400-PH copper variant uses a cable with 44 micro-coax conductors, 22 in each direction, and an overall shield. The length is limited to TBD. The signals are transformer coupled to the micro-coax to accomodate some difference in the ground potential between the equipment.

4.10 Fiber physical layer (optional)

Open Issue - Width of fiber interface, e.g., 16+4+1+1, or 8+2+1+1?

Open Issue - Length of fiber interface.

The optional HIPPI-6400-PH fiber variant uses a fiber-ribbon cable with 12 multi-mode fibers. The length is limited to TBD.

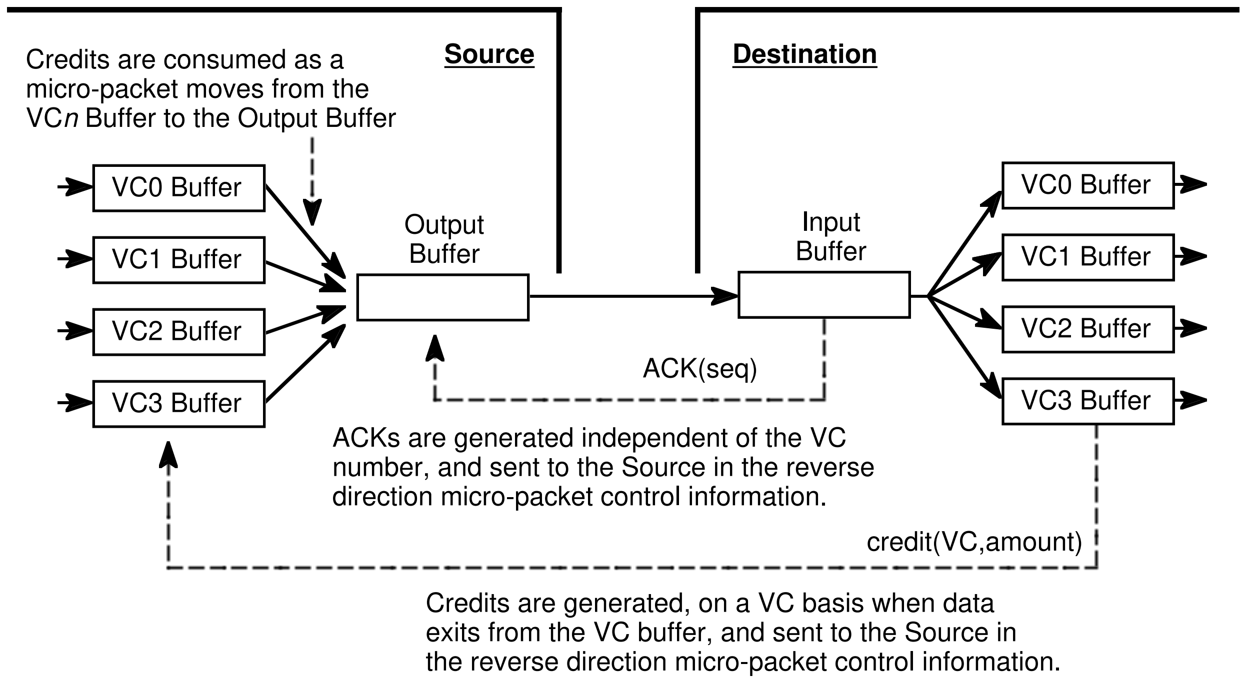


Figure 4 – Reverse direction control information

Table 1 – CRC coverage's in a 128-byte message

Micro-packet number	Data Bytes DB00 – DB31 contents	ECRC coverage	LCRC coverage
1	Header	Header	Header, c63 – c32
2	Bytes 1 – 32	Bytes 1 – 32	Bytes 1 – 32, c63 – c32
3	Bytes 33 – 64	Bytes 1 – 64	Bytes 33 – 64, c63 – c32
4	Bytes 65 – 96	Bytes 1 – 96	Bytes 65 – 96, c63 – c32
5	Bytes 97 – 128	Bytes 1 – 128	Bytes 97 – 128, c63 – c32

5 Service interface

Open Issue - Is there any reason not to include a service interface

Open Issue - Should the service interface be in the body of the document, in clause 4, in another clause, or in a normative annex?

This clause specifies the services provided by HIPPI-6400-PH. The intent is to allow ULPs to operate correctly with this HIPPI-6400-PH. How many of the services described herein are chosen for a given implementation is up to that implementor; however, a set of HIPPI-6400-PH services must be supplied sufficient to satisfy the ULP(s) being used. The services as defined herein do not imply any particular implementation, or any interface.

Figure 5 shows the relationship of the HIPPI-6400-PH interfaces.

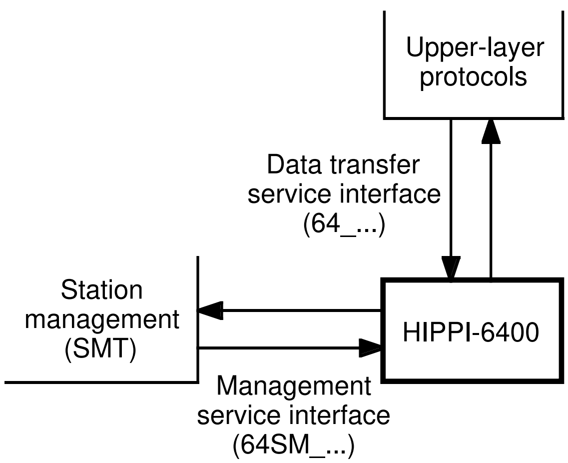


Figure 5 – HIPPI-6400-PH service interface

5.1 Service primitives

The primitives, in the context of the state transitions in clause 5, are declared required or optional. Additionally, parameters are either required, conditional, or optional. All of the primitives and parameters are considered as required except where explicitly stated otherwise.

HIPPI-6400-PH service primitives are of four types.

– *Request primitives* are issued by a service user to initiate a service provided by the HIPPI-6400-PH. In this standard, a second Request primitive of the same name shall not be issued until the Confirm for the first request is received.

–*Confirm primitives* are issued by the HIPPI-6400-PH to acknowledge a Request.

– *Indicate primitives* are issued by the HIPPI-6400-PH to notify the service user of a local event. This primitive is similar in nature to an unsolicited interrupt. Note that the local event may have been caused by a service Request. In this standard, a second Indicate primitive of the same name shall not be issued until the Response for the first Indicate is received.

– *Response primitives* are issued by a service user to acknowledge an Indicate.

5.2 Sequences of primitives

The order of execution of service primitives is not arbitrary. Logical and time sequence relationships exist for all described service primitives. Time sequence diagrams are used to illustrate a valid sequence. Other valid sequences may exist. The sequence of events between peer users across the user/provider interface is illustrated. In the time sequence diagrams the HIPPI-6400-PH users are depicted on either side of the vertical bars while the HIPPI-6400-PH acts as the service provider.

NOTE - The intent is to flesh out the service primitives similar to what is in HIPPI-PH today.

Open Issue - Do we need primitives to inform the SMT of the skew alignment status?

6 Micro-packet contents

6.1 Bit and byte assignments

As shown in figure 2, each micro-packet shall consist of 32 data bytes and 64 bits of control information. The data bytes shall be numbered DB00 - DB31. DB31 shall be transmitted first. The data bits in the micro-packet shall be numbered d*xx.y* where *xx* is the byte number and *y* is the bit number in the byte.

The 64 bits of control information shall be numbered as bits c63 – c00. Control bit c63 shall be transmitted first. As shown in figure 2, a field with a numerical value shall have its high-order bit in the highest numbered bit position.

The control information shall contain the following parameters located in the bits specified. Unless otherwise noted, the parameters are supplied by the Source side of a link.

VC (2 bits, c63-c62) – The virtual channel selector. (See 6.2.)

TYPE (4 bits, c61-c58) – Identifies the type of information within the micro-packet. (See 6.3.)

TAIL (1 bit, c57) – TAIL = 1 identifies the last micro-packet of a message. TAIL = 0 means that more micro-packets for this message follow.

ABORT (1 bit, c56) – ABORT = 1 means that an unrecoverable error has been detected in the message, do not check the ECRC. ABORT = 0 means that the message is OK so far.

Open Issue – The requirement to discard a message that has ABORT = 1 has been deleted. Is this OK?

Open Issue – Should the ABORT bit be renamed "ERROR"?

TSEQ (8 bits, c55-c48) – Sequence number of transmitted micro-packet. (See 6.4.)

RSEQ (8 bits, c47-c40) – Sequence number associated with micro-packet ACK. (See 6.4.)

VCR (2 bits, c39-c38) – Virtual channel number associated with credit addition. (See 6.5.)

CR (6 bits, c37-c32) – Amount of credit to add to the virtual channel specified in VCR. (See 6.5.)

NOTE - The RSEQ, VCR, and CR parameters are supplied by the Destination-side of the link for delivery to the Source-side at the far end of the link.

LCRC (16 bits, c31-c16) – Link level checksum covering the 32 data bytes, and the c63 through c32 control bits, in this micro-packet. (See 6.6.)

ECRC (16 bits, c15-c00) – End-to-end checksum covering only the data bytes in this message. (See 6.6.) In all other micro-packet ECRC is unused and may be any value.

Open Issue – The LCRC was moved ahead of the ECRC; does anyone have any problem with this?

6.2 Virtual channel (VC) selector

Four virtual channels shall be available in each direction on a link. Messages on the virtual channels shall be assigned as follows:

– VC0 = Messages with a maximum size of 64 data micro-packets (2048 bytes) plus one Header micro-packet (32 bytes).

– VC1 = Messages with a maximum size of 4096 data micro-packets (128 KBytes) plus one Header micro-packet (32 bytes).

– VC2 = Messages with a maximum size of 4096 data micro-packets (128 KBytes) plus one Header micro-packet (32 bytes).

– VC3 = Scheduled messages of unlimited size. (See clause 7.) Each message shall contain a Header micro-packet as the first micro-packet of the message.

6.3 Micro-packet TYPEs

The 4-bit TYPE parameter shall indicate the contents of the micro-packet. Unspecified TYPE values are reserved for future use.

6.3.1 TYPE = link control micro-packets

Control micro-packets operate at the link level, do not carry any user data, acknowledgments, or credit update information. Control micro-packets include:

– Reset (TYPE = x'2') – Sent by the Source to initiate a Reset operation in the Destination. (See xx.x)

– Reset_ACK (TYPE = x'3') – The Destination has seen, and completed, the Reset operation.

– Initialize (TYPE = x'4') – Sent by the Source to initiate an Initialization operation in the Destination. (See xx.x)

– Initialize_ACK (TYPE = x'5') – The Destination has seen, and completed, the Initialization operation.

Open Issue – How do these four micro-packet types relate to Cold Start (12.1) , Warm Start (12.2), Link Reset (12.3), and Training Sequences (11)?

6.3.2 TYPE = Null micro-packets

Null micro-packets (TYPE = x'7') are gap-fillers, sent when there are no Data micro-packets, or credit updates, to send. Even though they carry ACKs, Null micro-packets may be discarded by the Destination to compensate for drift differences between the received CLOCK signal and the Destination's internal clock. A Null micro-packet shall be sent at least every ?? µs to provide a chance to compensate for CLOCK rate differences between the Source and Destination.

Open Issue - Null micro-packets may be needed to account for clock drift and clock rate differences. How large a difference do we need to accomodate?

Open Issue - What is the maximum time between Null micro-packets?

6.3.3 TYPE = Data micro-packets

Data micro-packets (TYPE = x'8') carry user data, credit updates, and ACKs.

6.3.4 TYPE = Header micro-packets

Header micro-packets (TYPE = x'9') carry routing and control information.

Table 2 – Micro-packet contents summary

	Reset/Init	Null	Credit-only	Header	Data	Admin
Data Bytes contents	n.u.	n.u.	n.u.	Routing information	32 bytes of user data	Administrative information
VC	n.u.	n.u.	n.u.	any	any	any
TYPE (hex)	2,3,4,5	7	A	9	8	F
TAIL	n.u.	n.u.	n.u.	0	= 1 on last micro-packet of message	n.u.
ABORT	n.u.	n.u.	n.u.	0	= 1 if error	n.u.
TSEQ	n.u.	n.u.	increments	increments	increments	increments
RSEQ	n.u.	ACK	ACK	ACK	ACK	ACK
VCR	n.u.	n.u.	any	any	any	any
CR	n.u.	n.u.	any	any	any	any
LCRC	single	single	single	single	single	single
ECRC	single	single	single	single	accumulating	single

n.u. = not used, may contain any value
single = the CRC is calculated and checked for this single micro-packet
accumulating = ECRC continues from the previous micro-packet, it is not initialized

Open Issue – The Admin micro-packet type was added; does this cause anyone any problems? Are its functions defined in this document? In the -SC document? In another document?

6.3.5 TYPE = Credit-only micro-packets

Credit-only micro-packets (TYPE = x'A') carry credit update information, and acknowledgments, when there are no Data micro-packets to send.

6.3.6 TYPE = Admin micro-packets

Admin micro-packets (TYPE = x'F') carry operations and maintenance information between the end nodes of a link. Admin micro-packets are fully specified in ????

Open Issue – Where are Admin micro-packets specified? In this document? In an Annex? In another document? Which one?

6.4 Sequence number parameters

The transmit sequence number (TSEQ) shall increment by one for each Data, Header, and Idle micro-packet transmitted, and shall wrap from x'FE' to x'00'. The receive sequence number (RSEQ) shall be used to acknowledge (ACK) these micro-packets. RSEQ shall equal the TSEQ of the micro-packet being acknowledged. RSEQ = x'FF' indicates that no ACK is being transmitted.

NOTES

1 The TSEQ and RSEQ parameters are independent of the virtual channel used to transmit the micro-packet.

2 The TSEQ and RSEQ parameters are local to a specific link. For example, a micro-packet that transverses more than one link will most likely have different TSEQ numbers on the different links.

The wrap at x'FE' shall be taken into account when processing ACKs. For example, if the pervious ACK had RSEQ = x'F7', and an ACK with RSEQ = x'03' is received, then the micro-packets whose TSEQ value = x'F8' through x'FE', and from x'00' through x'03', are acknowledged and their memory may be reused by the Source.

6.5 Credit update parameters

The Destination shall insert the VCR and CR parameters in micro-packets to inform the Source that CR number of micro-packet buffers have been freed up for the VC indicated by VCR. This gives the Source permission to transmit CR number of micro-packets on this VC. The Source shall increase its Credit Counter for this virtual channel by the value in CR. The Source Credit Counter shall be able to record up to 255 credits. A credit update that would exceed 255 shall result in an error indication. (See xx.x.)

NOTES

1 The CR value is an incremental update value, not the number of buffers currently available in the Destination.

2 At 40 ns per micro-packet, and 5 ns per meter of cable, each credit is equivalent to about 8 meters of cable. Hence, a Credit Count, and Destination buffer capacity per virtual channel, of 255 will support full bandwidth on a 1 km link when round trip time is taken into account and Destination latency is low.

6.6 Check functions

Open Issue – Just about all of 6.6 is new text that needs to be checked for correctness and clarity.

6.6.1 Intended use of CRCs

Two 16-bit cyclic redundancy checks (CRCs) shall be used. The link CRC (LCRC) is calculated fresh for every micro-packet, and checked at each receiver. The end-to-end CRC (ECRC) operates in exactly the same fashion, with one exception. Namely, the ECRC is not initialized for each micro-packet of a message, but is accumulated across all of the micro-packets of the message.

The combination of two 16-bit CRCs provides the equivalent of a 32-bit CRC for link-level checking of individual micro-packets. In addition, the 16-bit ECRC provides checking over a whole message. While this standard only covers the link level, carrying the ECRC across intermediate devices, for example, across switches, is strongly encouraged.

6.6.2 Link-lever CRC (LCRC)

The link CRC (LCRC) shall cover all of the data bytes, and the control bits except for the ECRC and LCRC. The LCRC generator and checker shall be initialized to all ones for each micro-packet. The order that bits are included in the LCRC calculation shall be as specified in table 3.

The LCRC polynomial shall be:

NOTE - The intent is to use the CRC specification in the FDDI document as a template for specifying the polynomial.

Table 3 – Bit ordering for LCRC calculation

Open Issue - What is the order that bits are fed to the ECRC calculation?

6.6.3 End-to-end CRC (ECRC)

The end-to-end CRC (ECRC) shall include only the micro-packet's data bytes, not the control bits, in its calculation. The order that bits are included in the ECRC calculation shall be as specified in table 4.

The ECRC checker and generator for a VC shall be initialized to all ones after all micro-packets except Data micro-packets whose TAIL bit = 0.

NOTES

1 Since a message travels down a single VC, and micro-packets from different VCs can be interleaved on a link, the ECRC seed must be specific to a VC.

2 The ECRC is not re-initialized for Data micro-packets, i.e., in the middle of a message, so that it is cumulative across all of the micro-packets of a message.

3 By initializing the ECRC checker and generator after all but Data micro-packets whose TAIL bit = 0, the Header micro-packet is excluded from the cumulative ECRC, and the intermediate Data micro-packets are included. (See table 1.)

Open Issue - This text assumes that the Header micro-packet is not included in the ECRC calculation. Is this correct?

The ECRC polynomial shall be:

NOTE - The intent is to use the CRC specification in the FDDI document as a template for specifying the polynomial.

The ECRC may optionally be checked prior to transmission, and ABORT = 1 set in the micro-packet if the ECRC is in error. See HIPPI-6400-SC for suggested use of this option.

Open Issue – Should this be an option, or mandatory. If an option, should it be in this document? In HIPPI-6400-SC?

Table 4 – Bit ordering for ECRC calculation

Open Issue - What is the order that bits are fed to the ECRC calculation?

6.7 Header micro-packet contents

Open Issue – The Header micro-packet contents are new, and need to be reviewed.

Header micro-packets shall contain:

Destination_Address (16 bits, DB00–DB01) – The logical address of the Destination node. (See ANSI X3.xxx-199x, HIPPI-6400-SC for details.)

Source_Address (16 bits, DB02–DB03) – The logical address of the Source node. (See ANSI X3.xxx-199x, HIPPI-6400-SC for details.)

Message_Length (32 bits, DB04–DB07) – The length, in bytes, of the message data following this Header micro-packet.

EtherTYPE (16 bits, DB08–DB09) – A field identifying the upper-layer protocol associated with this message.

Not used (8 bits, DB10) – These bits are not used, and may contain any value.

Extension (8 bits, DB11) :

Extension = x'00' means that bytes DB12–DB29 of this Header micro-packet are unused, and may contain any value.

Extension = x'01' means that this Header micro-packet contains a Schedule_Header in bytes DB12–DB19. Bytes DB20–DB29 are not used and may contain any value.

Extension = x'03' means that this Header micro-packet contains a Schedule_Header in bytes DB12–DB19, and a H800_Header in bytes DB20–DB29.

Not used (16 bits, DB30–DB31) – These bits are not used, and may contain any value.

Schedule_Header (32 bits, DB12–DB15) – This optional header (see 7.2) is used when scheduling, or executing, transfers over VC3.

H800_Header (80 bits, DB16–DB25) – This optional header (see annex A) is used to map HIPPI-800 transfers.

Table 5 – Header micro-packet summary

	Data byte number	Function
In all Header micro-packets	00 01	Destination_ Address
	02 03	Source_Address
	04 05 06 07	Message_Length
	08 09	EtherTYPE
	10	Not used
	11	Extension
Schedule _Header (Optional)	12	Command
	13	Buffers_Available
	14 15	Connection_id
H800_Header (Optional)	16 17 18 19	I-Field
	20 21 22 23	Alternate_I-Field
	24 25	First_Burst_Size
In all Header micro-packets	26 27 28 29 30 31	Not used

7 VC3 scheduled transfers

Open Issue – All of the text in clause 7 is new and needs to be reviewed for correctness and clarity.

Unlike transfers on VC0, VC1 or VC2, transfers on VC3 shall be started only when the Destination is ready to accept the message, i.e., buffer space is available and no other VC3 transfers are underway to this Destination. Scheduling messages over VC0 shall be used to control the VC3 transfer. The VC3 path between the Source and Destination is called a virtual connection.

Since the Destination has indicated its ability to accept the message, VC3 will not become congested. In essence, the Destination smoothly controls the message flow. For comparison, on VC0, VC1, and VC2, the Source blindly sends messages into the fabric where they may have to wait behind other VC3 messages to the same Destination, or wait for Destination buffers to free up. These scheduled transfers add additional messages on VC0 and some round-trip overhead.

7.1 Scheduling messages

Scheduling messages, consisting of a Header micro-packet with no attached Data micro-packets, shall be transmitted on VC0. Information to control VC3 transfers shall be contained in the Schedule_Header portion of the Header micro-packet.

A typical sequence would consist of:

a The Source requests a VC3 virtual connection to the Destination.

b The Destination reserves its VC3 for this virtual connection, and informs the Source that the connection request is accepted, and the it has Buffers_Available buffer space currently available for this virtual connection.

c The Source sends one or more messages to the Destination.

d The Destination may update the Buffers_Available as Destination buffer space becomes available.

e The Source, or Destination, breaks the VC3 virtual connection.

7.2 Schedule_Header

A Schedule_Header shall be used in scheduling messages on VC0, and in the data messages on VC3. When the 4-byte Schedule_Header is used, it shall be in Data bytes DB12 through DB15 of a Header micro-packet. The Schedule_Header shall contain:

Command (8 bits, DB12) = The command to be executed. (See 7.3.1.)

Buffers_Available (8 bits, DB13) = An incremental update of the Destination's free buffer space for a particular virtual connection. Granularity is 4 KBytes. (See 7.3.2.)

Connection_id (16 bits, DB14–DB15) – Connection identifier. The tuple Source_Address, Destination_Address, and Connection_id, shall be used to uniquely identify the virtual connection.

7.2.1 Command parameter

The command parameter is encoded so that the low-order bit signifies whether the command originated from the Source or Destination side of the virtual connection, allowing commands to be easily routed to the recipient, (i.e., Sources talk to Destinations, and Destinations talk to Sources). Table 6 is a summary of the commands, and specifies the VC over which the command will be sent. Unused command codes are reserved for future use.

Request_Connection (x'00') – Request a virtual connection to the Destination specified in the Destination_Address field (Data bytes DB00–DB01). If the Destination cannot immediately make the connection, the Destination shall issue a Reject_Connection.

Request_Camp-on (x'10') – Request a virtual connection to the Destination specified in the Destination_Address field (Data bytes DB00–DB01). If not able to immediately make the connection, the Destination shall queue the request until it can respond with an Accept_Connection. Requests may be queued from multiple Sources; the algorithm to service them is outside the scope of this standard.

Accept_Connection (x'01') – The Destination has VC3 available for this transfer. This scheduling message shall also contain the Buffers_Available value for this virtual connection.

Table 6 – Scheduled transfers command summary

Command	From to *	Code	VC used	Number of micro-packets
Request_Connection	S –> D	x'00'	VC0	1
Request_Camp-on	S –> D	x'10'	VC0	1
Accept_Connection	D –> S	x'01'	VC0	1
Reject_Connection	D –> S	x'03'	VC0	1
Data_Present	S –> D	x'02'	VC3	2 to (2^{27} – 1)
Last_Data_Present	S –> D	x'06'	VC3	2 to (2^{27} – 1)
Update_Buffers_Available	D –> S	x'05'	VC0	1
Source_Breaks_Connection	S –> D	x'08'	VC0	1
Destination_Breaks_Connection	D –> S	x'09'	VC0	1
* S = Source, D = Destination				

Open Issue – Is the above table useful? Are there items that should be added? Deleted?

Reject_Connection (x'03') – The Destination is unable to complete the virtual connection at this time.

Data_Present (x'02') – This message is the first, or intermediate, data message for this virtual connection.

Last_Data_Present (x'06') – This message is the last data message for this virtual connection.

Update_Buffers_Available (x'05') – Additional buffering is now available in the Destination.

Source_Breaks_Connection (x'08') – The Source breaks the virtual connection. VC3 is now available for other uses.

Destination_Breaks_Connection (x'09) – The Destination breaks the virtual connection. VC3 is now available for other uses.

Open Issue – Are the command names acceptable? Can you suggest better names?

7.2.2 Buffers_Available parameter

The Buffers_Available parameter is sent by the Destination to inform the Source about additional free buffer space for this virtual connection. The Buffers_Available parameter shall be sent in either Accept_Connection or Update_Buffers_Available scheduling messages. The granularity is 4 KByte. For example, Buffers_Available = x'08' means that an additional 32 KBytes is available in the Destination's buffer for this virtual connection.

The Source shall keep a running total of the buffer space in the Destination, adding in Buffers_Available parameters from the Destination, and subtracting for data sent to the Destination. The Source shall initialize its view of the Destination's buffer space to zero when it sends a Request_Connection command.

8 Source specific operations

Open Issue – The "Message specific" and "Message independent" control parameter sections were removed as being mostly duplicates of text in 6.1. Are any problems seen?

8.1 Credit update indications on Source side

Credit update indications from the far end are received on the local Destination side, and passed to the local Source side, as shown in figure 4. A credit update shall increase the available credit, by the amount in the CR parameter, on the virtual channel whose number is the value in the VCR parameter.

8.2 ACK indications on Source side

ACK indications from the far end are received on the local Destination side, and passed to the local Source side, as shown in figure 4. An ACK indication acknowledges all of the transmitted micro-packets whose TSEQ ≤ RSEQ, i.e., the memory allocated to these micro-packets may be re-used. RSEQ = x'FF' shall be ignored.

Two consecutive ACKs with the same RSEQ value, and RSEQ ≠ x'FF', shall trigger a retransmit operation of all of the micro-packets in the Output Buffer (see figure 4) whose TSEQ is greater than the value in the duplicated RSEQ.

Open Issue – How many consecutive times should RSEQ = last good micro-packet be sent? Twice? Three times? Should RSEQ = x'FF' then be sent?

Open Issue – Is a time-out used to trigger a retransmit rather than a duplicate RSEQ? If so, how long is the timer?

Open Issue – Is there a maximum number of times a retransmit will be tried?

8.3 ACKs and credit updates to far end

The local Destination side sends ACKs and credit update information to the far end by first queuing them to the local Source side, as shown in figure 4. The Source side shall transmit this information in micro-packets using the appropriate control bits.

Since the ACKs and credit update information do not share their fields with any other parameters they can be sent with every micro-packet.

The local Destination may queue multiple ACK RSEQ parameters before one is transmitted by the local Source end. The RSEQ parameter should be over-written so that the ACK message transmitted uses the latest value of RSEQ.

Open Issue – Is the above statement true? Is it necessary?

8.4 Retransmissions

The Destination may request retransmission of unacknowledged micro-packets. (See 8.2.) The following parameters, from the original micro-packet, shall have the same value in the retransmitted micro-packet.

– VC

– TYPE

– TAIL

– ABORT

– TSEQ

– VCR

– CR

– ECRC

The following parameters may change as a micro-packet is retransmitted.

– RSEQ

– LCRC

Open Issue – Are these unchanged/changed parameters the right ones?

8.4 Virtual channel priorities

Micro-packets shall be chosen from the VCs with credit available in a round-robin fashion, e.g., from VC0, then VC1, VC2, VC3, VC0, etc., switching on each micro-packet. Optionally, messages are kept flowing once started, i.e., if the VC continues to have micro-packets to send, and has credit available, then that VC has priority over the other VCs. If no micro-packets are available to send, or have credit, then an Credit-only or Null micro-packets shall be sent.

Open Issue – This seems to capture the essence of the discussions through March. This text is only located in this document until another place can be found for it, e.g., HIPPI-6400-SC.

9 Destination specific operations

9.1 Checking for errors

Upon receipt of a micro-packet, the Destination shall check the LCRC and ECRC for errors.

```
IF LCRC = error
   THEN request retransmission
         log error
         EXIT
IF ECRC = error
AND IF ABORT = 0
   THEN request retransmission
         log error
         EXIT
ELSE accept micro-packet
```

If accepted, the micro-packet shall be delivered to the virtual channel buffer designated by the VC parameter.

Open Issue - Is the ECRC checked at each receiver, including intermediate nodes?

Skipping a TSEQ number within a message will result in an ECRC error. Losing a TAIL bit will result in concatenating two messages and an ECRC error.

Open Issue - Are these error conditions correct?

Open Issue - Are there other errors that are checked for?

9.2 Generating ACKs

If the LCRC is correct, the Destination shall queue, to the local Source side, an ACK with an RSEQ whose value equals the value of the TSEQ parameter in the received micro-packet. This acknowledges this micro-packet.

If the LCRC is incorrect, the Destination shall queue, to the local Source side, an ACK with RSEQ equal the TSEQ parameter of the last correctly received micro-packet. This requests retransmission of the flawed micro-packet, and all other micro-packets transmitted since this flawed micro-packet. The Destination shall discard all micro-packets until the originally flawed micro-packet is received correctly.

10 Signal line encoding

10.1 Signal line bit assignments

The data bytes and control bits shall be transmitted on the signal lines specified in table 7 for a 16-bit wide interface, and as specified in table 8 for an 8-bit wide interface. Nomenclature for the data and control bits is detailed in figure 2. Data signal lines are labeled capital D and a two-digit number, e.g., D00. Control signal lines are labeled capital C and a one-digit number, e.g., C0.

Table 7 – Signal line bit assignments in a 16-bit system

	Signal lines																			
bit	**C3**	**C2**	**C1**	**C0**	**D 15**	**D 14**	**D 13**	**D 12**	**D 11**	**D 10**	**D 09**	**D 08**	**D 07**	**D 06**	**D 05**	**D 04**	**D 03**	**D 02**	**D 01**	**D 00**
a	12	08	04	00	24.4	24.0	25.4	25.0	26.4	26.0	27.4	27.0	28.4	28.0	29.4	29.0	30.4	30.0	31.4	31.0
b	13	09	05	01	24.5	24.1	25.5	25.1	26.5	26.1	27.5	27.1	28.5	28.1	29.5	29.1	30.5	30.1	31.5	31.1
c	14	10	06	02	24.6	24.2	25.6	25.2	26.6	26.2	27.6	27.2	28.6	28.2	29.6	29.2	30.6	30.2	31.6	31.2
d	15	11	07	03	24.7	24.3	25.7	25.3	26.7	26.3	27.7	27.3	28.7	28.3	29.7	29.3	30.7	30.3	31.7	31.3
a	28	24	20	16	16.4	16.0	17.4	17.0	18.4	18.0	19.4	19.0	20.4	20.0	21.4	21.0	22.4	22.0	23.4	23.0
b	29	25	21	17	16.5	16.1	17.5	17.1	18.5	18.1	19.5	19.1	20.5	20.1	21.5	21.1	22.5	22.1	23.5	23.1
c	30	26	22	18	16.6	16.2	17.6	17.2	18.6	18.2	19.6	19.2	20.6	20.2	21.6	21.2	22.6	22.2	23.6	23.2
d	31	27	23	19	16.7	16.3	17.7	17.3	18.7	18.3	19.7	19.3	20.7	20.3	21.7	21.3	22.7	22.3	23.7	23.3
a	44	40	36	32	08.4	08.0	09.4	09.0	10.4	10.0	11.4	11.0	12.4	12.0	13.4	13.0	14.4	14.0	15.4	15.0
b	45	41	37	33	08.5	08.1	09.5	09.1	10.5	10.1	11.5	11.1	12.5	12.1	13.5	13.1	14.5	14.1	15.5	15.1
c	46	42	38	34	08.6	08.2	09.6	09.2	10.6	10.2	11.6	11.2	12.6	12.2	13.6	13.2	14.6	14.2	15.6	15.2
d	47	43	39	35	08.7	08.3	09.7	09.3	10.7	10.3	11.7	11.3	12.7	12.3	13.7	13.3	14.7	14.3	15.7	15.3
a	60	56	52	48	00.4	00.0	01.4	01.0	02.4	02.0	03.4	03.0	04.4	04.0	05.4	05.0	06.4	06.0	07.4	07.0
b	61	57	53	49	00.5	00.1	01.5	01.1	02.5	02.1	03.5	03.1	04.5	04.1	05.5	05.1	06.5	06.1	07.5	07.1
c	62	58	54	50	00.6	00.2	01.6	01.2	02.6	02.2	03.6	03.2	04.6	04.2	05.6	05.2	06.6	06.2	07.6	07.2
d	63	59	55	51	00.7	00.3	01.7	01.3	02.7	02.3	03.7	03.3	04.7	04.3	05.7	05.3	06.7	06.3	07.7	07.3

NOTES

1 The two-digit numbers in the C*n* columns are the control bits, c*nn.*

2 The three-digit numbers in the D*nn* columns are the data bits, d*xx.y,* where *xx* is the byte number and *y* is the bit number in the byte.

3 The bits in a column are transmitted on the associated signal line, top entry first, bottom entry last.

4 The four-bit groupings in a column denote 4-bit code groups (dcba) for encoding/decoding to/from the 5-bit transmission codes (zyTxw) specified in table 9.

Open Issue – Tables 7 and 8 need to be reviewed for correctness and clarity.

Table 8 – Signal line bit assignments in an 8-bit system

	Signal lines									
bit	**C1**	**C0**	**D 07**	**D 06**	**D 05**	**D 04**	**D 03**	**D 02**	**D 01**	**D 00**
a	08	00	24.0	25.0	26.0	27.0	28.0	29.0	30.0	31.0
b	09	01	24.1	25.1	26.1	27.1	28.1	29.1	30.1	31.1
c	10	02	24.2	25.2	26.2	27.2	28.2	29.2	30.2	31.2
d	11	03	24.3	25.3	26.3	27.3	28.3	29.3	30.3	31.3
a	12	04	24.4	25.4	26.4	27.4	28.4	29.4	30.4	31.4
b	13	05	24.5	25.5	26.5	27.5	28.5	29.5	30.5	31.5
c	14	06	24.6	25.6	26.6	27.6	28.6	29.6	30.6	31.6
d	15	07	24.7	25.7	26.7	27.7	28.7	29.7	30.7	31.7
a	24	16	16.0	17.0	18.0	19.0	20.0	21.0	22.0	23.0
b	25	17	16.1	17.1	18.1	19.1	20.1	21.1	22.1	23.1
c	26	18	16.2	17.2	18.2	19.2	20.2	21.2	22.2	23.2
d	27	19	16.3	17.3	18.3	19.3	20.3	21.3	22.3	23.3
a	28	20	16.4	17.4	18.4	19.4	20.4	21.4	22.4	23.4
b	29	21	16.5	17.5	18.5	19.5	20.5	21.5	22.5	23.5
c	30	22	16.6	17.6	18.6	19.6	20.6	21.6	22.6	23.6
d	31	23	16.7	17.7	18.7	19.7	20.7	21.7	22.7	23.7
a	40	32	08.0	09.0	10.0	11.0	12.0	13.0	14.0	15.0
b	41	33	08.1	09.1	10.1	11.1	12.1	13.1	14.1	15.1
c	42	34	08.2	09.2	10.2	11.2	12.2	13.2	14.2	15.2
d	43	35	08.3	09.3	10.3	11.3	12.3	13.3	14.3	15.3
a	44	36	08.4	09.4	10.4	11.4	12.4	13.4	14.4	15.4
b	45	37	08.5	09.5	10.5	11.5	12.5	13.5	14.5	15.5
c	46	38	08.6	09.6	10.6	11.6	12.6	13.6	14.6	15.6
d	47	39	08.7	09.7	10.7	11.7	12.7	13.7	14.7	15.7
a	56	48	00.0	01.0	02.0	03.0	04.0	05.0	06.0	07.0
b	57	49	00.1	01.1	02.1	03.1	04.1	05.1	06.1	07.1
c	58	50	00.2	01.2	02.2	03.2	04.2	05.2	06.2	07.2
d	59	51	00.3	01.3	02.3	03.3	04.3	05.3	06.3	07.3
a	60	52	00.4	01.4	02.4	03.4	04.4	05.4	06.4	07.4
b	61	53	00.5	01.5	02.5	03.5	04.5	05.5	06.5	07.5
c	62	54	00.6	01.6	02.6	03.6	04.6	05.6	06.6	07.6
d	63	55	00.7	01.7	02.7	03.7	04.7	05.7	06.7	07.7

NOTES:

1 The two-digit numbers in the C*n* columns are the control bits, c*nn*.

2 The three-digit numbers in the D*nn* columns are the data bits, d*xx.y*, where *xx* is the byte number and *y* is the bit number in the byte.

3 The bits in a column are transmitted on the associated signal line, top entry first, bottom entry last.

4 The four-bit groupings in a column denote 4-bit code groups (dcba) for encoding/decoding to/from the 5-bit transmission codes (zyTxw) specified in table 9.

10.2 Source-side encoding for dc balance

Open Issue – This text was revised based on the March meeting. For example, all of the bit numbers changed. It needs to be reviewed for correctness and clarity.

The transmitted signals shall be encoded to achieve dc balance on each signal line. Table 9 specifies the 5-bit signal line codes (zyTxw) corresponding to the 4-bit input codes (dcba) from tables 7 and 8. For example, on signal line D00, the first dcba 4-bit code consists of bits d31.0, d31.1, d31.2, and d31.3.

A running count, called the Disparity Count, shall be kept of all the ones and zeros transmitted since the link was reset. The Disparity Count shall be incremented for each one transmitted, and decremented for each zero transmitted.

Open Issue – Should the Disparity Count be reset on Skew Adjustments?

The appropriate 5-bit code value from table 9, based on the current value of the Disparity Count, shall be transmitted in the sequence, w,x,T,y,z. (T = true/complement bit) For example in the right column of tables 7 and 8, if:

a = d31.0 = 1 (least-significant bit)
b = d31.1 = 0
c = d31.2 = 0
d = d31.3 = 0
and Disparity Count = +1 before encoding,

then, based on the third column second row in table 9, transmit on D00:

w = 1 (transmitted first)
x = 0
T = 1
y = 0
z = 0

Disparity Count = 0 after encoding.

NOTES

1 The range for the Disparity Count is from +4 to -5.

2 The Disparity Count may also be updated by adding or subtracting the value of Delta Disparity shown in table 9. Add Delta Disparity if Disparity Count < 0; subtract if ≥ 0.

3 The 5-bit code is derived by inserting a 1 in the middle of the 4-bit code, and then transmitting either the true or complement value of the resultant 5-bit quantity.

4. The maximum run length, i.e., the longest string of continuous 1s or 0s, is 11. The string of 4-bit code points creating the maximum run length is x'efc'. Start with Disparity Count = +3 or +4 for a string of 11 zeros. Start with Disparity Count = -4 or -5 for a string of 11 ones.

The data and control signal lines shall be synchronized with the CLOCK and FRAME signals as shown in figures 6 and 7. Figures 6 through 9 are read left to right, i.e., events on the left occur before those on the right.

Table 9 – Line coding

4-bit code dcba	5-bit code when Disparity < 0 zyTxw	5-bit code when Disparity ≥ 0 zyTxw	Delta Disparity
0000	11011	00100	3
0001	11010	00101	1
0010	11001	00110	1
0011	00111	11000	1
0100	10011	01100	1
0101	01101	10010	1
0110	01110	10001	1
0111	01111	10000	3
1000	01011	10100	1
1001	10101	01010	1
1010	10110	01001	1
1011	10111	01000	3
1100	11100	00011	1
1101	11101	00010	3
1110	11110	00001	3
1111	11111	00000	5

10.3 Destination-side decoding

The received signals shall each be decoded in groups of five bits according to table 9.

NOTES

1 The FRAME signal marks the beginning of a 5-bit code.

2 Decoding can be implemented by examining the middle bit of the 5-bit code; if 1 then use the outer bits uncomplemented, if 0 then complement before use.

3 There are no illegal 5-bit codes.

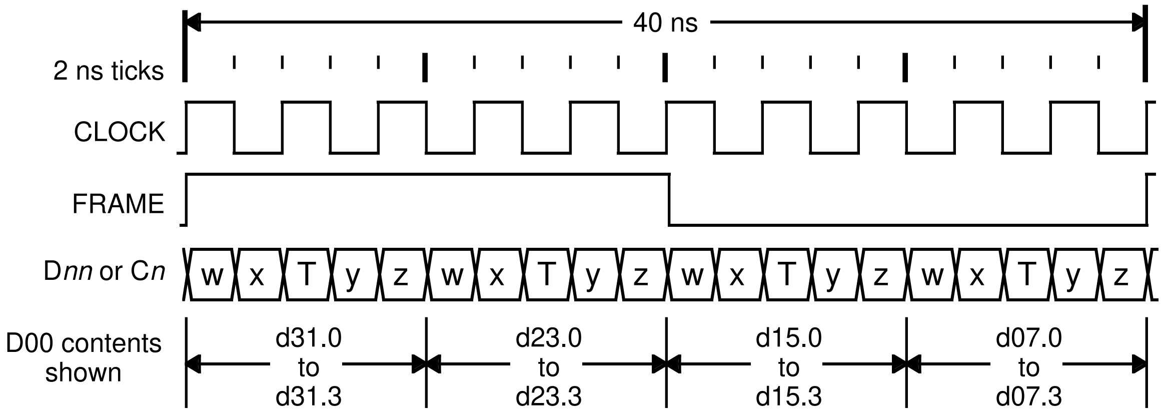


Figure 6 – 16-bit system micro-packet

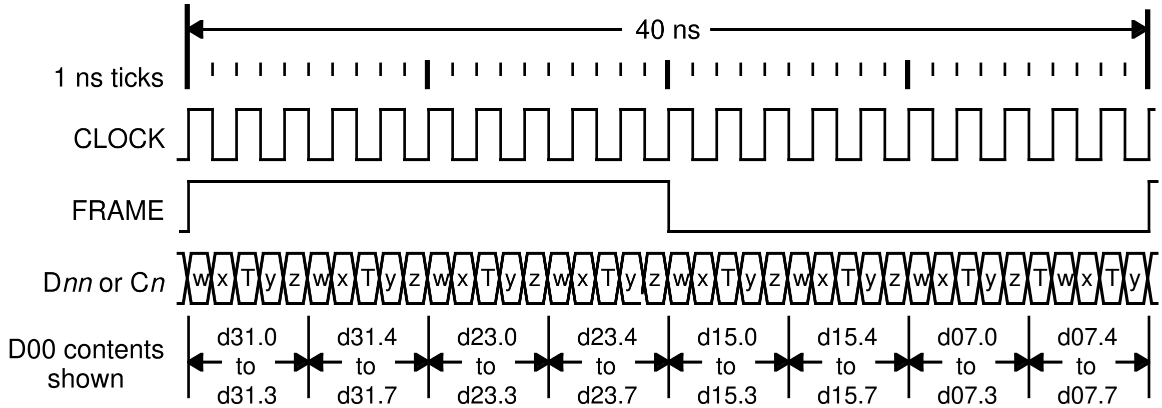


Figure 7 – 8-bit system micro-packet

Open Issue – These figures are new and need to be reviewed for correctness and clarity.

10.4 Logic levels

A logical 1 shall be transmitted on an electrical interface as the more positive electrical signal. A logical 1 shall be transmitted on an optical interface as greater light output power than for a logical 0.

Open Issue - Are these proposed logic levels OK?

10.5 FRAME signal

The FRAME signal transitions shall be as shown in figures 6 through 9.

11 Dynamic skew compensation

The Destination shall compensate for up to 10 ns of skew between the signals. Skew is defined as the time between when the fastest and slowest signals reach the Destination. Training sequences (see figures 6 and 7) shall be used to measure the skew, and perform dynamic skew adjustments. A 101010 pattern on the FRAME signal shall be used to identify a training sequence.

Open Issue – 10 ns of skew tolerance was picked out of the air, what is the correct value?

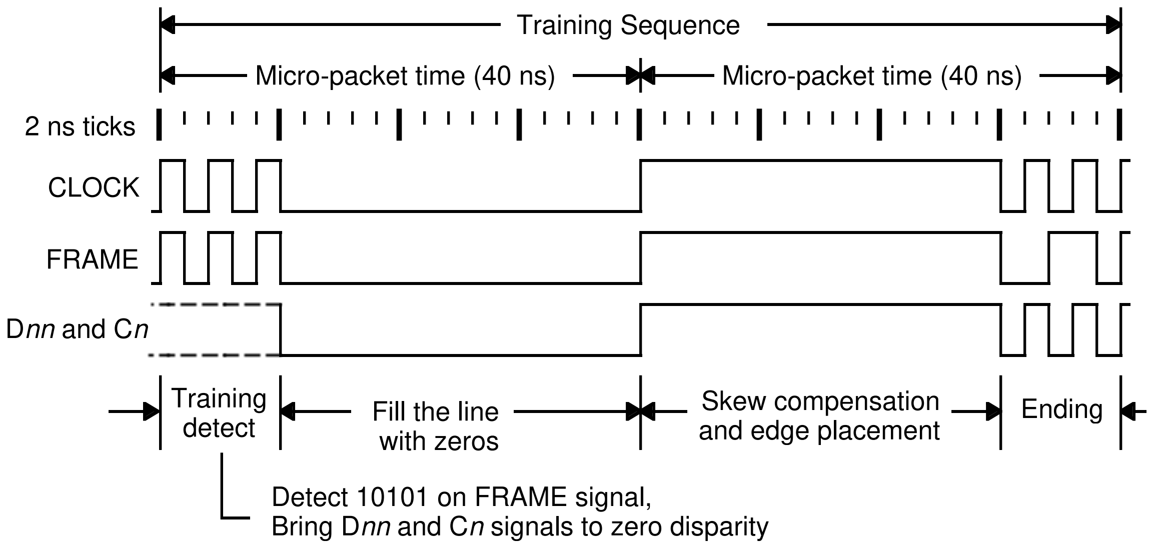


Figure 8 – 16-bit system training sequence

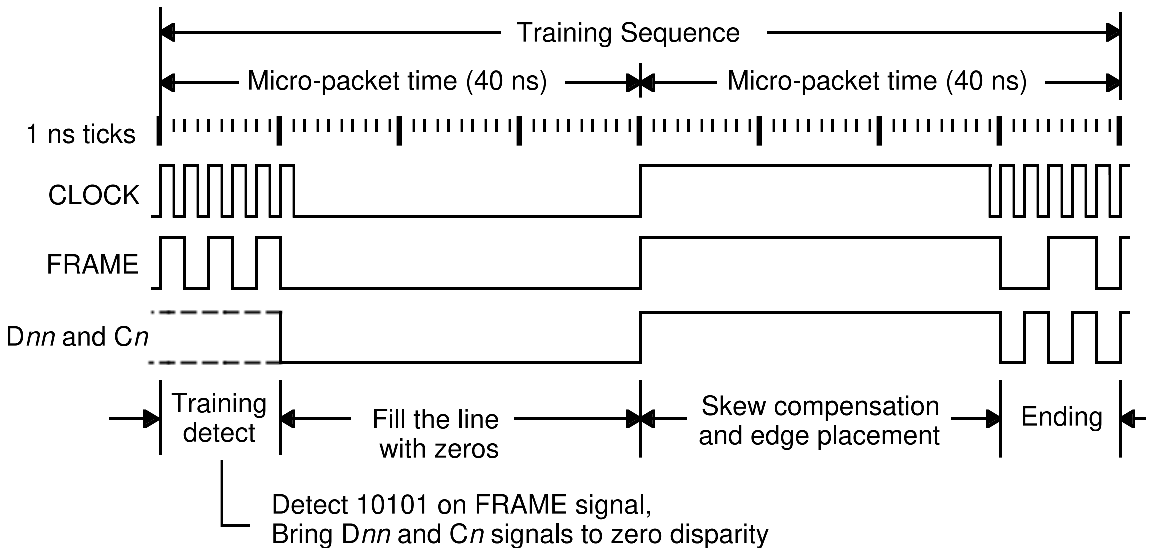


Figure 9 – 8-bit system training sequence

Open Issue – These figures are new and need to be reviewed for correctness and clarity.

A single training sequence shall be inserted at least every ?? ns to adjust the dynamic skew. During the first 5 CLOCK ticks of a training sequence the Source shall insert appropriate Data and Control bits to drive the Disparity Count (see 10.2) on each signal line to zero.

Open Issue – How often must training sequences be issued?

Open Issue - Is dynamic skew compensation mandatory or optional?

12 Initialization

Three levels of initialization, or reset, are specified, power-on or cold start, warm start, and link reset. They differ in the amount of hardware reset, amount of time until the link is available, and the amount of data lost.

Open Issue - What is the difference between cold and warm start?

12.1 Cold start

Cold start is triggered by Source power-on. A cold start shall ???

Open Issue - What are the results of a cold start?

A cold start operation shall be signaled to the far end by the local Source side toggling the FRAME signal at the same rate as the CLOCK signal, and in-phase with the CLOCK signal for one second. During this time the Destination side shall use the dynamic skew compensation to align the FRAME signal to the CLOCK signal.

Open Issue – What is the maximum time between Skew Adjustment micro-packets?

Open Issue – Is there a maximum rate or, or minimum time between, Skew Adjustment micro-packets?

Open Issue - Are there actions or resets that occur during this second?

Open Issue – Is one second the correct amount of time for the reset?

At the end of the second, if the local Destination side has received a cold start indication, i.e., the FRAME signal toggling at the same rate as the CLOCK signal, then the local Source side shall decrease the toggling rate of the FRAME signal to one-half the rate of the CLOCK signal.

Open Issue - What happens if the local Destination end has not received a cold start indication, i.e., FRAME toggling at CLOCK rate?

Open Issue - Are there any special actions that occur at this time?

Open Issue - How long do you stay in this state, and how do you get out?

Open Issue - When do you do the dynamic skew alignment on the data and control bits?

12.2 Warm start

Warm start is triggered by

Open Issue - What triggers a warm start?

A warm start shall

Open Issue - What are the results of a warm start?

12.3 Link Reset

Link Reset shall be triggered by ??.

Open Issue - What triggers a Link Reset?

A Link Reset shall

Open Issue - What are the results of a Link Reset?

13 Maintenance and control features

Open Issue - What maintenance or control messages are possible?

Open Issue - Do we want to standardize all possible maintenance and control messages?

Open Issue - What is the relative priority of maintenance and control messages?

Open Issue - Should we have a MIB for a HIPPI-6400-PH link? For other HIPPI-6400-PH?

14 Timing

Open Issue – The HIPPI-PH specification will be used as a starting point for these specifications. Does HIPPI-PH include specifications that are not pertinent to HIPPI-6400?

14.1 Source CLOCK signal

The transmitted CLOCK signal at the Source bulkhead connector shall have a nominal period of 4 ns ± 0.4 ps (± 0.01%) for a 16-bit system, and a period of 2 ns ± 0.2 ps (±0.01%) for an 8-bit system.

CLOCK symmetry, measured as the percentage of time in the high state compared to the total CLOCK period, shall be 50 (±5) %. Peak jitter shall be less than 0.05 ns.

Open Issue – The tolerances were taken from HIPPI-PH, are they correct here?

14.2 Destination CLOCK signal

14.3 FRAME, Data, and Control signals

15 Copper interface (optional)

15.1 General

Open Issue - Width of interface – 16+4+1+1

Open Issue - Baud rate, tolerance

Open Issue - What is the shape of the FRAME signal?

15.2 Electrical output interface

Open Issue - Voltage, current, risetime specifications.

Open Issue - What is the output skew specification?

Open Issue - What coupling is used to cable – capacitor? Transformer? What are the parameters?

15.3 Electrical input interface

Open Issue - Input voltage levels

Open Issue - What is the input skew specification?

Open Issue - What coupling is used to cable – capacitor? Transformer? What are the parameters?

15.4 Electrical connector

Open Issue - Connector pin assignments

Open Issue - Connector specifications?

Open Issue - Different connectors at Source and Destination? Same with keying?

15.5 Cable specifications

Open Issue - Cable type, number of conductors

Open Issue - Cable length – max., min.

Open Issue - Cable electrical specs. – impedance, loss, relative skew, ...

Open Issue - Cable mechanical – shielding, size, material

15.6 Grounding

Open Issue - Ground shield at Source? Destination? Both?

Open Issue - What to do with unused conductors? Unused pins?

15.7 Cable termination

Open Issue - Termination value

Open Issue - Are terminators used on all cables? A function of length?

Open Issue - Termination location – at receiver? in connector?

16 Optical interface (optional)

16.1 General

Open Issue - Width of interface – 8+2+1+1?

Open Issue - Signalling frequency, tolerance

Open Issue - What is the shape of the FRAME signal?

16.2 Optical output interface

Open Issue - Optical parameters – wavelength, power, …

Open Issue - What is the output skew specification?

16.3 Optical input interface

Open Issue - Optical parameters – power...

Open Issue - What is the input skew specification?

16.4 Optical connector

Open Issue - Connector pin assignments

Open Issue - Connector specifications?

Open Issue - Different connectors at Source and Destination? Same with keying?

Open Issue - Field termination of connectors

16.5 Optical cable specifications

Open Issue - Cable type, number of fibers

Open Issue - Cable length

Open Issue - Cable optical specs. – loss, crosstalk, relative skew, ...

Open Issue - Cable mechanical – fiber pitch, size, jacket material

xx Other Open Issues

Open Issue - Should we have an annex describing the use of HIPPI-6400-PH links in networks? It could show a group of links crossing a switch and illustrate how the VC is unchanged through the fabric, and most of the other parameters are local to the links.

Annex A
(normative)

Mapping HIPPI-800 over HIPPI-6400

Will be based on proposal presented by Don Tolmie at the March HIPPI-6400 meeting in Mt. View. Omitted now due to lack of time.