INTERNATIONAL STANDARD

ISO/IEC 11518-2

Second edition
2000-10

Information technology – High-Performance Parallel Interface –

Part 2: Framing Protocol (HIPPI-FP)

Reference number
ISO/IEC 11518-2:2000(E)

INTERNATIONAL STANDARD

ISO/IEC 11518-2

Second edition
2000-10

Information technology – High-Performance Parallel Interface –

Part 2: Framing Protocol (HIPPI-FP)

PRICE CODE P

For price, see current catalogue

CONTENTS

INFORMATION TECHNOLOGY – HIGH-PERFORMANCE PARALLEL INTERFACE –

Part 2: Framing protocol (HIPPI-FP)

FOREWORD

1) ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.
2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.
3) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 11518-2 was prepared by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

This edition cancels and replaces the first edition published in 1996. This second edition was updated as follows:

– the figure in the foreword was removed;
– a list of acronyms was added (see 3.3);
– the upper-layer protocol identifiers were updated (see 5.4.1).

This publication has been drafted in accordance with the ISO/IEC Directives, Part 3.

ISO/IEC 11518 consists of the following parts, under the general title *Information technology – High-Performance Parallel Interface:*

– *Part 1: Mechanical, electrical and signalling protocol specification (HIPPI-PH)*
– *Part 2: Framing Protocol (HIPPI-FP)*
– *Part 3: Encapsulation of ISO/IEC 8802-2 (IEEE Std 802.2) – Logical Link Control Protocol Data Units (HIPPI-LE)*
– *Part 4: Mapping of HIPPI to IPI device generic command sets (HIPPI-IPI)* (under consideration)
– *Part 5: Memory Interface (HIPPI-MI)* (under consideration)
– *Part 6: Physical Switch Control (HIPPI-SC)*
– *Part 8: Mapping to Asynchronous Transfer Mode (HIPPI-ATM)*
– *Part 9: Serial Specification (HIPPI-Serial)*

Annexes A, B and C are for information only.

INTRODUCTION

This standard defines the data framing for an efficient simplex high-performance point-to-point interface.

Characteristics of HIPPI-FP include:

- large block data transfers with framing to split the data into smaller bursts;
- separation of user control and data information, and early delivery of the control information;
- identifiers for multiple upper-layer protocols (ULPs);
- support for simplex topology;
- support for ULP non-word-aligned and an arbitrary number of byte transfers;
- error notifications, from the underlying physical layer, e.g. HIPPI-PH, are passed through this framing protocol to notify the upper layers of damaged data;
- provides a connection-less data service;
- best effort delivery of data, i.e. datagram;
- connection control information, which may be used for physical layer switching, is supported.

INFORMATION TECHNOLOGY – HIGH-PERFORMANCE PARALLEL INTERFACE –

Part 2: Framing protocol (HIPPI-FP)

1 Scope

This part of ISO/IEC 11518 provides data framing for a high-performance point-to-point interface between data-processing equipment. This part of ISO/IEC 11518 does not protect against certain errors that might be introduced by intermediate devices interconnecting multiple HIPPI-PHs.

The purpose of this part of ISO/IEC 11518 is to facilitate the development and use of the HIPPI in computer systems by providing common data framing. It provides an efficient framing protocol for interconnections between computers, high-performance display systems, and high-performance, intelligent block-transfer peripherals.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 11518. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 11518 are encouraged to investigate the possibility of applying the most recent edition of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 11518-1:1995, *Information technology, High-Performance Parallel Interface – Part 1: Mechanical, electrical, and signalling protocol specification (HIPPI-PH)*

3 Definitions and conventions

3.1 Definitions

For the purposes of this part of ISO/IEC 11518, the following definitions apply.

3.1.1
burst
group of words sent by the Source to the Destination

Bursts contain 1 to 256 words. Bursts that contain less than 256 words are called short bursts. On a 32-bit HIPPI-PH, bursts contain an even number of 32-bit words.

3.1.2
byte
group of eight bits

Bytes are packed four per 32-bit word, or eight per 64-bit word.

3.1.3
connection
condition of the HIPPI-PH when data transfers from Source to Destination are possible

3.1.4
connection control information (CCI)
parameter sent as part of the sequence of operations establishing a connection from a Source to a Destination

NOTE ISO/IEC 11518-6 includes examples of CCIs and topologies.

3.1.5
destination
the equipment at the end of the interface that receives the data

3.1.6
optional
features that are not required by this part of ISO/IEC 11518

NOTE However, if any optional feature defined by this part of ISO/IEC 11518 is implemented, it must be implemented according to this part of ISO/IEC 11518.

3.1.7
packet
data set sent from Source to Destination

A packet is composed of one or more bursts. The HIPPI specification does not limit the maximum packet size, but a maximum size may be imposed by a given HIPPI implementation, or by a ULP. A packet consists of a header, one or two optional ULP data sets, and optional fill.

3.1.8
service interface (SI)
connection points to the ULP

3.1.9
source
the equipment at the end of the interface that transmits the data

3.1.10
state
current condition of the interface, excluding transitions, as indicated by the control primitives

3.1.11
station management (SMT)
the supervisory entity that monitors and controls the HIPPI

3.1.12
ULP data set
data transferred between the ULP and the HIPPI-FP

3.1.13
upper-layer protocol (ULP)
protocol immediately above the HIPPI-FP service interface

3.1.14
word
a unit of information, consisting of (32 or 64) bits, matching the HIPPI-PH word size

Words contain an ordered set of four bytes or eight bytes

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., CLOCK). 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., In, Out, Enabled). Any lowercase uses of these words have the normal technical English meaning.

3.3 Acronyms

CCI connection control information

FP Framing Protocol

FPSM Framing Protocol, Station Management

IPI Intelligent Peripheral Interface

LLRC Length-longitudinal redundancy check

SMT station management

ULP upper-layer protocol

4 HIPPI structure

4.1 Structure

The HIPPI-FP has been designed in a modular fashion to support simplex or dual simplex configuration requirements.

A compliant HIPPI network shall maintain packet and burst structures from the original Source to the final Destination.

Figure 1 shows the basic organization of the information on the HIPPI.

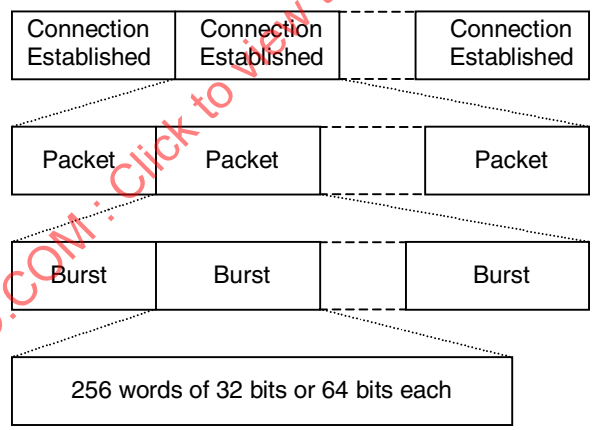

Figure 1 – Logical framing hierarchy

As specified in HIPPI-PH, once a connection is established, a packet (or multiple packets) can be sent from the Source to the Destination. Each packet contains one or more bursts. Bursts contain (1 to 256) words. Words contain four or eight bytes. Bursts that contain less than 256 words are called short bursts. A packet contains no more than one short burst. A short burst may be either the first or last burst of a multiburst packet. For error detection HIPPI-PH uses byte parity and a parity-based checksum on each burst.

On a 32-bit HIPPI-PH, bursts shall contain an even number of 32-bit words. Words shall contain an ordered set of bytes as specified in 7.1.

4.2 Error detection mechanisms

4.2.1 Byte parity

The HIPPI physical layer (HIPPI-PH) uses bit-parallel word transfers, using 32-bit words for an 800 Mbit/s data rate and 64-bit words for a 1 600 Mbit/s data rate. An odd-parity bit is also transmitted with each 8-bit byte of a word, i.e., four parity bits are transmitted with each 32-bit word. Hence an undetected error in a word would require a 2-bit error, with both bits being in the same byte.

4.2.2 LLRC

The Length-Longitudinal Redundancy Check (LLRC) implements even parity across the individual bits of multiple words in a burst. For example, bit 23 of the LLRC is the even parity of bit 23 of each word in the burst. A burst is nominally 256 words in length (1 Kbyte or 2 Kbytes), but short bursts may contain fewer words. Hence the LLRC would not detect errors where the same bit in an even number of words was incorrect.

In addition, the LLRC calculation includes the length of the burst. Hence, the LLRC would detect cases where a word was dropped or added, i.e., the length received was not the same as what was transmitted.

4.2.3 Packet length

A packet is composed of one or more bursts. In HIPPI-FP a length field specifying the number of bytes in the packet is specified. This length field provides a check for dropped or extra bursts. A special case where the packet length is not used is provided for such things as video data to a frame buffer, data collection from experimental equipment, etc.

4.3 Error detection limitations

The parity and LLRC will only fail on 4-bit errors in a rectangular pattern. That is, two bits in a byte must fail (undetected by the byte parity check) and the same two bits must fail in another word of the burst (undetected by the LLRC).

Use of the HIPPI-FP packet header length field permits the detection of lost bursts within a packet; however, no mechanism of either HIPPI-FP or HIPPI-PH allows the detection of data corruption caused by the substitution of one burst, with good parity and LLRC, for another burst of the same length.

5 HIPPI-FP service interface to upper layers

This clause describes the services provided by HIPPI-FP. The intent is to provide the formalism necessary to relate this interface to other HIPPI interfaces. How many of the services described herein are chosen for a given implementation, and whether others may be required, is up to the implementor; however, a set of HIPPI-FP 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.

In this part of ISO/IEC 11518 the ULP and station management protocol (SMT) are service users, and the HIPPI-FP is the service provider to the ULP and SMT. The interfaces consist of the ULP primitives, prefixed with FP_, and the SMT primitives, prefixed with FPSM_.

The HIPPI-FP is also the service user of the HIPPI-PH services, prefixed with PH_.

Figure 2 shows the relationship of the HIPPI-FP interfaces.

5.1 Service primitives

All of the primitives and parameters are considered as required except where explicitly stated otherwise.

HIPPI service primitives are of four types.

– *Request primitives* are issued by a service user to initiate a service from the service provider. In this part of ISO/IEC 11518, 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 service provider to acknowledge a Request.

– *Indicate primitives* are issued by the service provider 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 part of ISO/IEC 11518, 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.

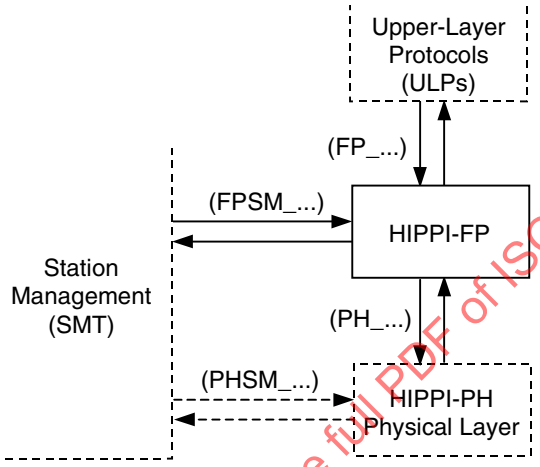

Figure 2 – HIPPI-FP service interface

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, as in figure 3, 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-FP users are depicted on either side of the vertical bars while the service provider is in the centre. A ULP or SMT implementation may present multiple requests for services, but the requests shall be serviced one at a time and in the order presented.

5.3 HIPPI-FP service primitive summary

ULP Data Transfer
FP_TRANSFER.Request (CCI, ULP-id,
D1_Size, D1_Data_Set, D2_Size,
D2_Data_Set, Keep_Connection,
Start_D2_on_Burst_Boundary)
FP_TRANSFER.Confirm
FP_TRANSFER_D1.Indicate (ULP-id, CCI,
Status, D2_Size, D2_Offset,
D1_Area_Size, D1_Data_Set)
FP_TRANSFER_D2.Indicate (ULP-id, CCI,
Status, D2_Size, D2_Offset, D2_Data_Set)
FP_TRANSFER.Response

Control Link
FPSM_CONTROL.Request (Command,
Command_Parameter)
FPSM_CONTROL.Confirm (Status)

Link Status
FPSM_STATUS.Request
FPSM_STATUS.Confirm (Status)
FPSM_STATUS.Indicate
FPSM_STATUS.Response

5.4 ULP data transfer service primitives

These primitives, as illustrated in figure 3, shall be used to transfer ULP data from the Source ULP to the Destination ULP.

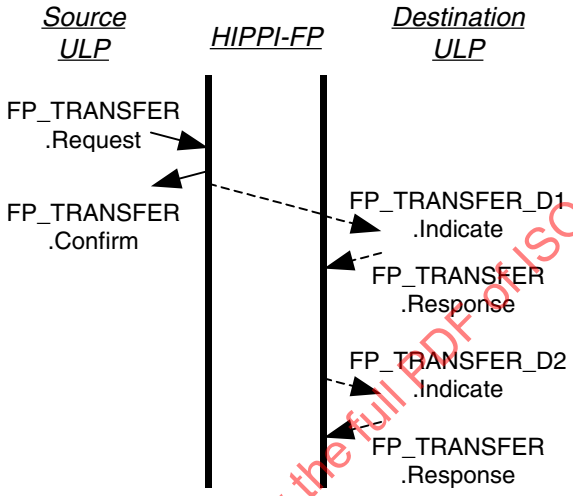

Figure 3 – Data transfer service primitives

5.4.1 ULP Identifiers

The ULP-id of the HIPPI-FP header designates the Destination ULP to which the data set is to be delivered.

NOTE 1 Identifiers registered at the time this part of ISO/IEC 11518 was approved include the following (shown in binary notation). Later registrations will be added as amendment to this part of ISO/IEC 11518.

Unlisted ULP-id values are reserved. Processing received packets with unlisted ULP-id values is undefined.

00000100 = ISO 8802.2 Link Encapsulation
00000110 = IPI-3 Slave, i.e., IPI-3 Master to Slave
00000111 = IPI-3 Master, i.e., IPI-3 Slave to Master
00001000 = IPI-3 Peer
00001010 = HIPPI-FC map to Fibre Channel ULPs
00001100 = Scheduled Transfer
00001101 = HIPPI-6400 Encapsulation
1xxxxxxx = Locally assigned

5.4.2 FP_TRANSFER.Request

Issued by the Source ULP to request a data transfer. If a connection to the Destination specified by the CCI does not currently exist, then a connection will be established. At the completion of the transfer the connection may be broken unless Keep_Connection was specified. The packet format is defined in 7.2, and shown in figure 8.

Semantics – FP_TRANSFER.Request (
CCI,
ULP-id,
D1_Size,
D1_Data_Set,
D2_Size,
D2_Data_Set,
Keep_Connection,
Start_D2_on_Burst_Boundary)

The CCI is passed directly to the underlying HIPPI-PH.

The ULP-id identifies the Destination ULP. See 5.4.1.

D1_Size is the length, in bytes, of the ULP D1_Data_Set to be placed in the first burst of the packet. The maximum D1_Size shall be 1 016 bytes. A value of D1_Size equal to zero indicates a null D1_Data_Set and shall cause the FP header D1_Data_Set_Present bit to be set to 0. See 7.2.2.

D1_Data_Set is the ULP data set to be placed in the first burst, and delivered separately from the D2_Data_Set. The D1_Data_Set is intended for control information.

D2_Size is the length, in bytes, of the ULP data set to be placed in the remainder of the packet. The maximum determinate D2_Size is 4 294 967 294 bytes ($2^{32} - 2$). A D2_Size of hexadecimal FFFFFFFF shall mean that the length is indeterminate at the start of the transfer. Indeterminate length packets may be longer or shorter than the maximum determinate size. See B.1 for suggestions on transferring larger size, or indeterminate size, packets. The D2_Size shall be set to zero to indicate the absence of the D2_Data_Set.

D2_Data_Set is the ULP data set to be sent. Placement of the D2_Data_Set shall be governed by the Start_D2_on_Burst_Boundary parameter. The D2_Data_Set is intended for user data, or the whole data set if separate control information is not used. A ULP may deliver the D2_Data_Set to HIPPI-FP in multiple segments. To decrease latency and conserve buffers, implementations may start transmission before receiving all of these segments.

Keep_Connection true says that another ULP data set with the same routing information is coming, and the physical HIPPI-PH connection should be maintained if possible. When Keep_Connection is false, the connection may be broken after this packet. Servicing FP_TRANSFER.Requests from other ULPs may also cause the connection to be broken, e.g., requests to different Destinations, or requests with Keep_Connection false. Keep_Connection is a local control parameter and is not passed to the Destination.

Start_D2_on_Burst_Boundary controls the starting location for the D2_Area. If true, then the D2_Area shall start at the beginning of the second HIPPI-PH burst. If false, then the D2_Area may start in the first burst.

Issued – The Source ULP issues this primitive to the Source HIPPI-FP to request the transfer of the ULP data set to the Destination.

Effect – The Source HIPPI-FP shall accept the ULP data set for transmission. The HIPPI-FP shall build an HIPPI-FP header, as specified in 7.2, and send the packet as a series of bursts to the Destination. If (1) the D1_Data_Set does not completely fill the first burst, and (2) Start_D2_on_Burst_Boundary = true, and (3) the underlying HIPPI-PH supports short first bursts, then this HIPPI-FP shall use a short first burst whose length is sufficient to completely contain the D1_Data_Set. If any of the above conditions are not met, then a 256-word first burst shall be used.

5.4.3 FP_TRANSFER.Confirm

This primitive acknowledges the FP_TRANSFER.Request from the Source ULP.

Semantics – FP_TRANSFER.Confirm (Status)

Status shall be:

– Accept – the HIPPI-PH has completed the connection and accepted the packet for transmission.
– Reject – the Destination has rejected the connection request, no bursts were transmitted.
– Timeout – the Destination did not respond to the connection request within the timeout period. No bursts were transmitted. See A.4.7

Issued – The HIPPI-FP shall issue this primitive to the Source ULP to acknowledge the FP_TRANSFER.Request.

Effect – Unspecified

5.4.4 FP_TRANSFER.Indicate

These primitives indicate to the Destination ULP that the D1_Data_Set, or D2_Data_Set, of a packet, addressed to this particular ULP has been received from the Source.

Semantics –

```
FP_TRANSFER_D1.Indicate (
    ULP-id,
    CCI,
    Status,
    D2_Size,
    D2_Offset,
    D1_Area_Size,
    D1_Area)

FP_TRANSFER_D2.Indicate (
    ULP-id,
    CCI,
    Status,
    D2_Size,
    D2_Offset,
    D2_Data_Set)
```

ULP-id is the ULP to receive the data. See 5.4.1.

CCI is the CCI for the current connection, i.e., received with the PH_RING.Indicate connection request.

Status denotes whether the data set being delivered was received with errors. Status includes, but is not limited to, errors in the packet.

D2_Size is the length of the D2_Data_Set, in bytes, as received in the FP_Header. If D2_Size equals hexadecimal FFFFFFFF, then it is up to the ULP to determine the validity and actual length of the D2_Data_Set.

D2_Offset is the number of unused bytes from the start of the D2_Area to the first byte of the D2_Data_Set. The D2_Offset is used by the Source and Destination to keep proper word alignment on the D2_Data_Set so as to avoid shifting and copying the data to achieve alignment at the Destination. The D2_Offset allows the Source memory image of the D2_Data_Set, even if it does not start on a 64-bit word boundary, to be reproduced at the Destination.

D1_Area_Size is the size of the D1_Area being passed to the ULP. The actual size of the D1_Data_Set is self-defining within the D1_Area.

D1_Area contains the D1_Data_Set. It is up to the Destination ULP to determine the size of the D1_Data_Set and extract it from the D1_Area. See 7.2.2.

The D2_Data_Set is the D2 ULP data being delivered to the ULP.

Issued – The Destination HIPPI-FP shall issue this primitive to the Destination ULP when a ULP data set has been received. A packet containing both the D1_Data_Set and the D2_Data_Set shall generate primitives for both the D1_Data_Set and the D2_Data_Set.

Effect – Unspecified

5.4.5 FP_TRANSFER.Response

This primitive acknowledges a FP_TRANSFER.Indicate for either the D1_Data_Set or the D2_Data_Set.

Semantics – FP_TRANSFER.Response

Issued – The Destination ULP issues this primitive to acknowledge receipt of the FP_TRANSFER.Indicate.

Effect – The Destination HIPPI-FP is enabled to issue another FP_TRANSFER.Indicate.

5.5 Control service primitives

These primitives, as illustrated in figure 4, shall be used to set parameters and control the interface. Note that a Control primitive can be initiated from either the Source or Destination.

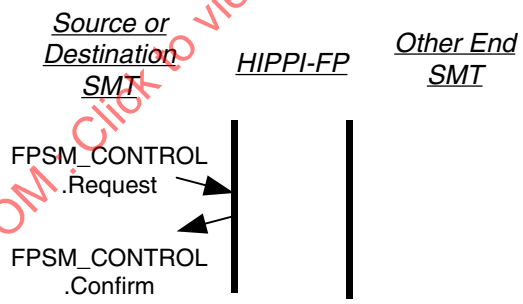

Figure 4 – Control service primitives

5.5.1 FPSM_CONTROL.Request

Issued by either the Source SMT or Destination SMT to set parameters, or otherwise control the local HIPPI-FP. Several functions are specified and others are left to specific implementations.

Semantics – FPSM_CONTROL.Request (Command, Command_Parameter)

The Command specifies the function to be performed. The parameters are specific to each function.

The Commands and Command_Parameters for the Source side include but are not limited to

Reset
Break Connection
Indicate Enable/Disable

Reset resets the HIPPI-FP, breaks any existing connections, and cancels any pending .Request primitives.

Break Connection breaks any existing connections.

Indicate Enable/Disable allows/disallows issuance of FPSM_STATUS.Indicate primitives.

The Commands and Command_Parameters for the Destination side include but are not limited to

Reset
Break Connection
Allow/Disallow/Reject Connection
Indicate Enable/Disable

Reset resets the HIPPI-FP, breaks any existing connections, and cancels any pending. Request primitives.

Break Connection breaks any existing connections.

Allow/Disallow/Reject Connection. Sets Connection_Enable. Allow enables the Destination to make a connection. Disallow instructs the Destination to ignore connection requests. Reject instructs the Destination to respond to connection requests with rejected connection sequences.

Indicate Enable/Disable allows/disallows the HIPPI-FP to issue FPSM_STATUS.Indicate primitives.

Issued – The Source or Destination SMT issues this primitive to perform some control function over the interface as a whole.

Effect – The HIPPI-FP shall perform the function specified.

5.5.2 FPSM_CONTROL.Confirm

This primitive acknowledges the FPSM_CONTROL.Request to the issuing SMT.

Semantics – FPSM_CONTROL.Confirm (Status)

Status reports the success or failure of the FPSM_CONTROL.Request commands.

Issued – The HIPPI-FP shall issue this primitive to the SMT when the command specified in the FPSM_CONTROL.Request has been accepted.

Effect – Unspecified

5.6 Status service primitives

These primitives, as illustrated in figure 5, shall be used to obtain status information from the local HIPPI-FP. Note that a Status primitive can be initiated from either the Source or Destination, and shall only affect the local end of the interface.

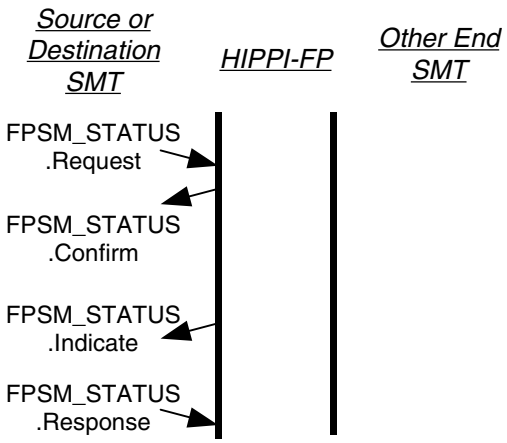

Figure 5 – Status service primitives

5.6.1 FPSM_STATUS.Request

Issued by either the Source SMT or Destination SMT to request a status report.

Semantics – FPSM_STATUS.Request

Issued – The SMT issues this primitive to obtain the status of the HIPPI-FP.

Effect – The HIPPI-FP shall respond with a FPSM_STATUS.Confirm.

5.6.2 FPSM_STATUS.Confirm

This primitive replies to the previous FPSM_STATUS.Request with status information.

Semantics – FPSM_STATUS.Confirm (Status)

The Source side Status shall contain, but is not limited to

Errors
Current state of HIPPI connection
FPSM_STATUS.Indicates Enabled/Disabled

The Destination side Status shall contain, but is not limited to

Errors
Current state of HIPPI connection
Last CCI Received
Connections Allowed/Disallowed/Rejected
FPSM_STATUS.Indicates Enabled/Disabled

Issued – The HIPPI-FP shall issue this primitive to the SMT in response to a FPSM_STATUS.Request.

Effect – Unspecified

5.6.3 FPSM_STATUS.Indicate

This primitive informs the SMT entity that a major event has occurred that affects the operation of the HIPPI-FP.

Semantics – FPSM_STATUS.Indicate

Issued – The HIPPI-FP, when enabled, shall issue this primitive to the SMT whenever a major event is detected. Major events include but are not limited to

Detection of an illegal state transition

NOTE 2 If a FPSM_CONTROL.Request is accepted successfully but not completed, then an FPSM_STATUS.Indicate can be used to indicate completion.

Effect – Unspecified

NOTE 3 Upon receipt of this primitive the local SMT entity should issue a FPSM_STATUS.Request to read status and determine which event occurred.

5.6.4 FPSM_STATUS.Response

This primitive acknowledges the FPSM_STATUS.Indicate.

Semantics – FPSM_STATUS.Response

Issued – The SMT issues this primitive to acknowledge receipt of the FPSM_STATUS.Indicate.

Effect – The HIPPI-FP, if enabled, is allowed to issue another FPSM_STATUS.Indicate.

6 HIPPI-PH to HIPPI-FP services

A summary of the primitives used to connect the HIPPI-FP to the HIPPI-PH is included here. The complete specification of the primitives is contained in the HIPPI Mechanical, Electrical, and Signalling Protocol Specification (HIPPI-PH) document (ISO/IEC 11518-1).

Initiate a Connection

PH_RING.Request (CCI)
PH_RING.Confirm
PH_RING.Indicate (CCI)
PH_RING.Response

Complete the Connection

PH_ANSWER.Request (Accept/Reject)
PH_ANSWER.Confirm
PH_ANSWER.Indicate (Accept/Reject)
PH_ANSWER.Response

Packet Control

PH_PACKET.Request (Begin/End)
PH_PACKET.Confirm (Accept/Reject)
PH_PACKET.Indicate (Begin/End,Status)
PH_PACKET.Response

Burst Transfer

PH_TRANSFER.Request (Length,Burst)
PH_TRANSFER.Confirm (Accept/Reject)
PH_TRANSFER.Indicate (Status,Length,Burst)
PH_TRANSFER.Response

Terminate the Connection

PH_HANGUP.Request
PH_HANGUP.Confirm
PH_HANGUP.Indicate
PH_HANGUP.Response

7 HIPPI data formats

7.1 Word and byte formats

The data transferred between the HIPPI-PH and the HIPPI-FP shall be 32-bit or 64-bit words. The words shall consist of 32 signals or 64 signals labelled D00 through D31 or D00 through D63. The size of the words shall match the HIPPI-PH.

NOTE 4 The HIPPI-PH defined in the HIPPI Mechanical, Electrical and Signalling Protocol Specification (HIPPI-PH) (ISO/IEC 11518-1) uses 32-bit words for the 800 Mbit/s option and 64-bit words for the 1 600 Mbit/s option.

The data transferred between the HIPPI-FP and the ULP shall be an ordered byte stream. The byte positions within the HIPPI words, for both the 32-bit and 64-bit HIPPI-PHs, shall be as shown in table 1. Byte 0 is the first byte in the ordered byte stream, byte 1 is the second byte, etc.

Table 1 – Byte assignments

Byte No.	Data signals on 32-bit HIPPI	Data signals on 64-bit HIPPI
0	D31-D24	D31-D24
1	D23-D16	D23-D16
2	D15-D08	D15-D08
3	D07-D00	D07-D00
4	D31-D24	D63-D56
5	D23-D16	D55-D48
6	D15-D08	D47-D40
7	D07-D00	D39-D32

Figure 6 shows the complete mapping of a 16-byte ordered byte stream on a 32-bit HIPPI-PH, and the same ordered byte stream on a 64-bit HIPPI-PH. Byte 0 is the first byte of the byte stream.

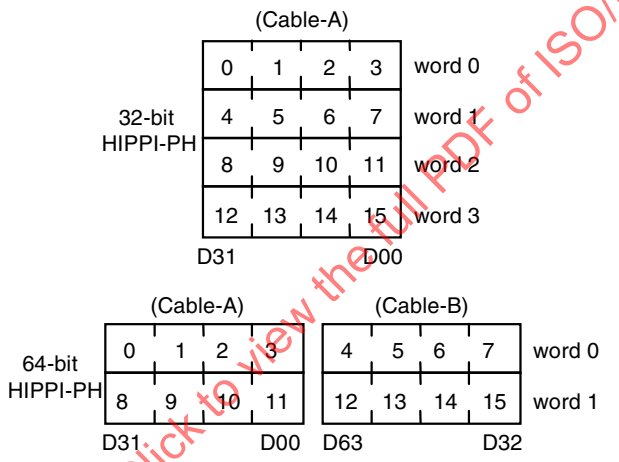

Figure 6 – Ordered byte stream to HIPPI-PH

Within each byte of the interface, the highest numbered signal shall be the most significant bit of the byte. An example byte is shown in figure 7.

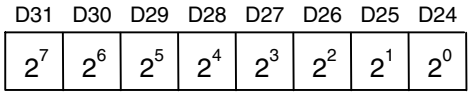

Figure 7 – Bit significance within a byte

7.2 HIPPI-FP packet format

The packet data presented by the Source ULP with a FP_TRANSFER.Request primitive shall be transferred to the Destination with an HIPPI-FP header as shown in figure 8. The image is shown as it would appear on Cable-A of a 32-bit HIPPI-PH. The most-significant bit of the individual fields shown in figure 8 is at the left end of the field.

The HIPPI-FP packets shall be composed of three areas, (1) Header_Area, (2) D1_Area, and (3) D2_Area, each starting and ending on a 64-bit boundary. If the D1_Data_Set is used as control information, the D2_Data_Set is intended as the data associated with that control information. See B.2 for packet examples.

NOTE 5 Several implementations have failed due to ignoring the 64-bit boundaries specified above. Implementors should take care to conform to the 64-bit boundary specification, as well as the other specifications in this standard.

7.2.1 Header_Area

The Header_Area shall be the first 64 bits of the packet, and shall be completely contained in the first burst of the packet.

ULP-id (8 bits) designates the Destination ULP to which the packet is to be delivered. See 5.4.1.

P = D1_Data_Set_Present (1 bit) = 1 designates that a D1_Data_Set is present in this packet.

B = Start_D2_on_Burst_Boundary (1 bit) = 0 designates that the D2_Area starts at or before the beginning of the second burst of the packet. B = 1 designates that the D2_Area starts at the beginning of the second HIPPI-PH burst of the packet.

D1_Area_Size (8 bits) designates the size of the D1_Area, i.e., the number of 64-bit words between the end of the 64-bit Header_Area and the start of the D2_Area.

D2_Offset (3 bits) designates the number of Offset bytes from the start of the D2_Area to the first byte of the D2_Data_Set.

NOTE 6 The concatenation of the D1_Area_Size and D2_Offset, referenced to the end of the header area, points to the first byte of the D2_Data_Set.

Reserved (11 bits). All of the reserved bits shall be transmitted as zeros.

D2_Size (32 bits) is the length, in bytes, of the D2_Data_Set portion of the packet. The D2_Size does not include the bytes contained in the D2_Offset, or in the Fill following the D2_Data_Set. A D2_Size of hexadecimal FFFFFFFF specifies that the D2_Data_Set size is unknown at the start of the packet transfer. This means that packets with an unknown D2_Data_Set size cannot be terminated at arbitrary byte boundaries, only at 64-bit HIPPI-PH burst boundaries. A D2_Size of zero (0) specifies that the D2_Area and the D2_Data_Set do not exist.

7.2.2 D1_Area

The D1_Area shall immediately follow the Header_Area, shall be completely contained in the first burst, shall contain an integral number of 64-bit words, and shall contain the D1_Data_Set (if present).

If present, the D1_Data_Set shall be the first information in the D1_Area. If the P bit of the Header_Area = 0, then the D1_Data_Set is not present, and the contents of the D1_Area may be ignored. The D1_Data_Set is intended for control information that may be delivered to the Destination ULP on receipt, without waiting for the arrival of other bursts of the packet.

The size of the D1_Data_Set shall be self-defining. For example, the HIPPI-IPI, identified by ULP-id = 00000111, uses a variable length D1_Data_Set byte string with the length imbedded in the byte string. The maximum size of the D1_Data_Set shall be 1 016 bytes, i.e., the maximum size that fits in the first burst of a 32-bit HIPPI-PH.

NOTE 7 The Source and Destination are not symmetrical. At the Source, the ULP determines the D1_Data_Set size, but the HIPPI-FP determines the size of the D1_Area. At the Destination, the whole D1_Area is passed to the ULP, and the ULP must extract the D1_Data_Set from the D1_Area. The D1_Data_Set, if present, is completely contained within the D1_Area. However, the D1_Area may be larger than the D1_Data_Set, for example, to complete the first burst of a class 3a packet (see B.2.3). A D1_Area with no D1_Data_Set, i.e., P = 0, is also permitted.

7.2.3 D2_Area

The D2_Area, if D2_Size is not zero, shall immediately follow the D1_Area, shall start and end on a 64–bit boundary, and shall contain the D2_Data_Set. If the B bit of the Header_Area = 1, then the D2_Area shall start at the beginning of the second HIPPI-PH burst.

The Offset is the unused bytes from the start of the D2_Area to the first byte of the D2_Data_Set.

The D2_Data_Set may range in size from zero to an indeterminate number of bytes (see D2_Size in 7.2.1).

Fill is the unused bytes between the end of the D2_Data_Set and the end of the D2_Area, i.e., the end of the packet. If a D2_Size of all binary ones is used, then there is no Fill.

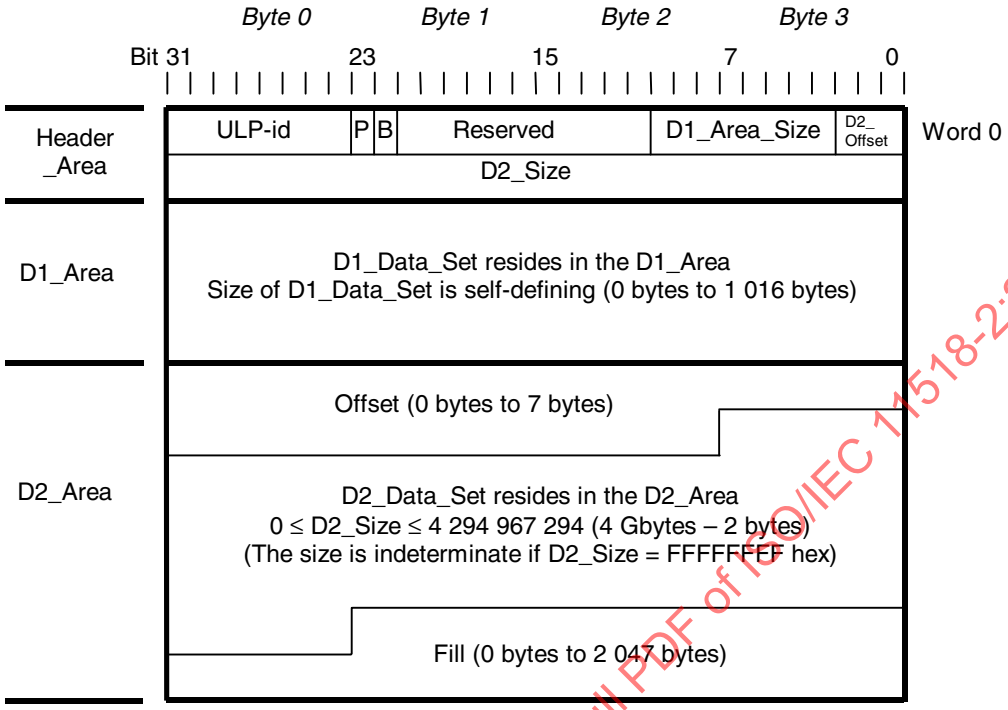

Figure 8 – HIPPI-FP packet format

Annex A
(informative)

State transitions and pseudo-code

A.1 General

The framing protocol service interface and the HIPPI-PH service interface are tied together by the state transition pseudo-code. The flow diagrams are included as a convenience for the user, the pseudo-code is more complete.

The state transitions and flow diagrams do not describe the means or specific implementation by which the functions are provided. However, other implementations with fewer or additional states should behave in a manner which is compatible with peer protocols implemented identical to the model.

Source states start with the letter S, Destination states with the letter D. Source states within this standard are numbered 2Jx0 where J = 0 - 4. Destination states within this document are numbered 2Kx0 where K = 5 - 9. This numbering scheme is used to avoid confusion between the Source and Destination states, and between states in this document and states in the HIPPI-PH physical layer document.

A.2 State exit

Within a state that is testing for some condition, the pseudo-code is assumed to loop indefinitely within the state until some exit condition is met.

In the event that control sequence errors are detected, the state machine breaks any existing connection and returns to the disabled state (S2000 or D2500). For the purposes of reporting errors, control sequence errors take precedence over data errors.

A.3 Interlocks

The implementor is cautioned that interlock flags or queues may be necessary to enforce the requirement of 5.1 that a second .Indicate or .Request primitive may not be issued by the HIPPI-FP until a .Response or .Confirm primitive of the same name has been received. These interlocks are not illustrated in the pseudo-code.

A.4 Source pseudo-code

The Source flow diagram in figure A.1 gives an overview of the Source pseudo-code.

A.4.1 S2000

Disabled state; initialize the HIPPI-FP. Enter on Power-up, system master reset, or illegal state transition.

Initialize
Issue PH_HANGUP.Request
Set Connection_Present = false
Goto S2010

A.4.2 S2010

Wait for any connections to be broken. Time T1, specified in the HIPPI-PH document, is the round-trip propagation delay plus action time.

IF time T1 timeout

THEN Goto S2020

A.4.3 S2020

Idle; wait for a transfer request from the ULP.

IF PH_HANGUP.Indicate
THEN Issue PH_HANGUP.Response
Set Connection_Present = false
IF FP_TRANSFER.Request
THEN Goto S2030

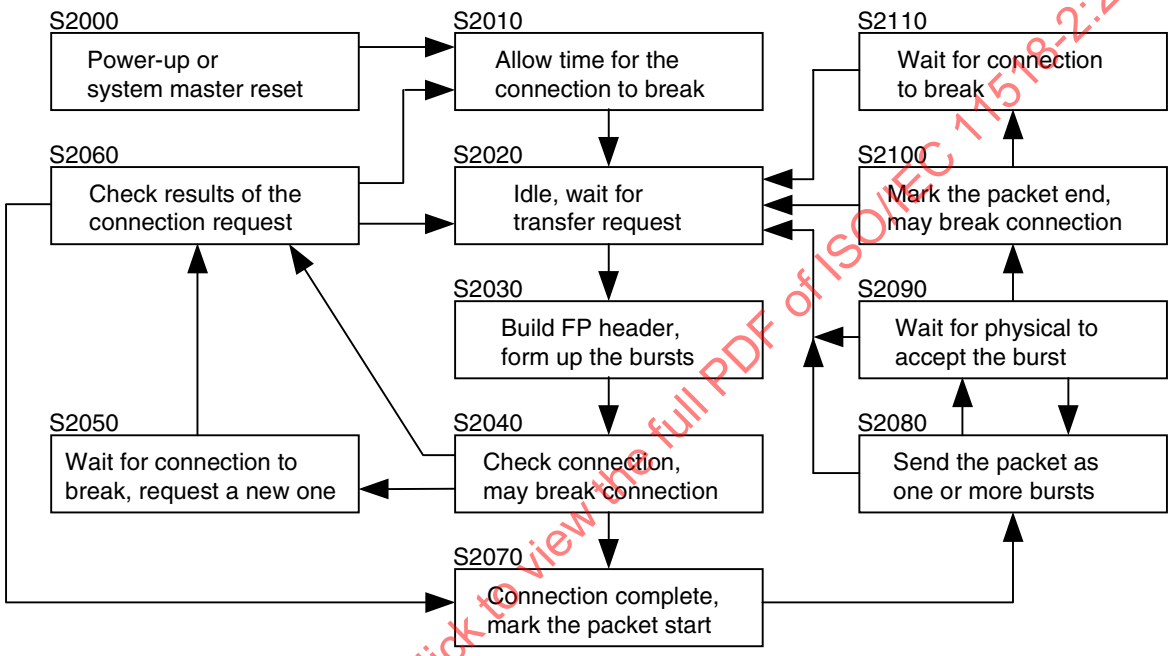

Figure A.1 – Source flow diagram

A.4.4 S2030

Build the FP header based on the parameters in the FP_TRANSFER.Request primitive, and form up bursts for transmission.

Set D2_Size and D2_Offset as appropriate
Maintain_Connection = Keep_Connection
IF D1_Size not = 0
THEN D1_Data_Set_Present = true
Put D1_Data_Set in first burst
ELSE D1_Data_Set_Present = false
Set D1_Area_Size = *n*
(where D1_Size ≤ *n* ≤ Max_Burst_Size - 8)
IF Start_D2_on_Burst_Boundary = true
THEN the D2_Data_Set starts in second burst
ELSE the D2_Data_Set can start in first burst
Form up bursts for transmission
Goto S2040

A.4.5 S2040

Check the connection. If no connection exists, then initiate a connection sequence. If a connection already exists to this Destination, then omit the connection sequence and go on to the data transfer. If a connection to a different Destination exists, then break that connection and initiate a connection sequence to the new Destination.

IF Connection_Present = false
    THEN Issue PH_RING.Request (CCI)
        Goto S2060
IF PH_HANGUP.Indicate
    THEN Issue PH_HANGUP.Response
        Set Connection_Present = false
        Issue PH_RING.Request (CCI)
        Goto S2060
IF CCI = Old_CCI
    THEN Goto S2070
Issue PH_HANGUP.Request
Goto S2050

NOTE 8 Breaking the connection and state S2050 are optional when using the HIPPI with non-switched dedicated interfaces.

A.4.6 S2050

Wait for the original connection to be broken before requesting a new connection to this Destination.

IF PH_HANGUP.Indicate
    THEN Issue PH_HANGUP.Response
        Set Connection_Present = false
        Issue PH_RING.Request (CCI)
        Goto S2060

A.4.7 S2060

Wait for the Destination to respond to the connection request. Time T2 is the maximum time to wait for a connection to be completed and is used to avoid hanging the HIPPI-FP. No assumptions are made as to where or how timer T2 is implemented. It could be done in the Source HIPPI-FP or the ULP, with hardware, software or a mix of the two. The suggested default value of T2 is 10 s.

IF PH_ANSWER.Indicate (Accept)
    THEN Set Connection_Present = true
        Set Old_CCI = CCI
        Goto S2070
IF PH_ANSWER.Indicate (Reject)
    THEN Issue FP_TRANSFER.Confirm (Reject)
        Goto S2020
IF time T2 timeout
    THEN Issue FP_TRANSFER.Confirm (Timeout)
        Issue PH_HANGUP.Request
        Goto S2010

A.4.8 S2070

A connection now exists; send the packet. Mark the beginning of the packet and inform the ULP that the transfer request has been accepted.

Issue FP_TRANSFER.Confirm (Accept)
Issue PH_PACKET.Request (Begin)
Goto S2080

NOTE 9 The FP_TRANSFER.Confirm is shown being issued before the packet is sent to the Destination. An implementation may optionally move this primitive so that the FP_TRANSFER.Confirm occurs after the packet has been sent.

A.4.9 S2080

Transfer a burst of the packet. If the ULP is delivering data to the HIPPI-FP in segments, make sure that a complete burst of data is available before issuing the PH_TRANSFER.Request primitive.

IF PH_HANGUP.Indicate
    THEN Issue PH_HANGUP.Response

```
        Set Connection_Present = false
        Goto S2020
    ELSE Issue PH_TRANSFER.Request
            (Length,Burst)
        Decrement burst count
        Goto 2090
```

A.4.10 S2090

Wait for the HIPPI-PH to accept the burst, or for the Destination to break the connection. If all of the bursts have not been transmitted, then go back to transmit another burst.

```
IF PH_HANGUP.Indicate
    THEN Issue PH_HANGUP.Response
        Set Connection_Present = false
        Goto S2020
IF PH_TRANSFER.Confirm
    THEN IF all bursts have been transmitted
            THEN Goto S2100
            ELSE Goto S2080
```

A.4.11 S2100

All of the bursts have been transmitted, mark the end of the packet. The connection is broken at the completion of the packet unless Keep_Connection was specified in the FP_TRANSFER.Request.

```
Issue PH_PACKET.Request (End)
IF Maintain_Connection = true
    THEN Goto S2020
Issue PH_HANGUP.Request
Goto S2110
```

A.4.12 S2110

Wait for the connection to be broken.

```
IF PH_HANGUP.Indicate
    THEN Issue PH_HANGUP.Response
        Set Connection_Present = false
        Goto S2020
```

A.5 Destination pseudo-code

The Destination flow diagram in figure A.2 gives an overview of the Destination pseudo-code.

A.5.1 D2500

Disabled state; initialize the HIPPI-FP and break any connections that may exist. Enter on Power-up, system master reset, or illegal state transition.

```
Initialize
Issue PH_HANGUP.Request
Goto D2510
```

A.5.2 D2510

Wait for the hangup to complete or time T1 to expire. Time T1, specified in the HIPPI-PH document, is the round-trip propagation delay plus action time.

```
IF PH_HANGUP.Indicate
    THEN Issue PH_HANGUP.Response
        Goto D2520
```

IF time T1 timeout
    THEN Goto D2520

A.5.3 D2520

Idle; wait for a connection request from the Source HIPPI. Save the CCI for later FP_TRANSFER.Indicate primitives.

IF PH_RING.Indicate (CCI)
    THEN Issue PH_RING.Response
        CCI_Received = CCI
        Goto D2530

A.5.4 D2530

Accept or reject the connection request from the Source.

IF Connection_Enable = Disallow
    THEN Goto D2540
IF Connection_Enable = Reject
    THEN Goto D2550
IF CCI_Received = Unknown or illegal
    THEN Goto D2550
Issue PH_ANSWER.Request (Accept)
Goto D2560

A.5.5 D2540

Connections are disallowed, wait for the Source to abort this connection request.

IF PH_HANGUP.Indicate
    THEN Issue PH_HANGUP.Response
        Goto D2520

A.5.6 D2550

The connection is being rejected.

Issue PH_ANSWER.Request (Reject)
Goto D2520

A.5.7 D2560

A connection is established. Wait for the start of a packet or a disconnect.

Set Packet_Error = false
Set First_Burst_Error = false
IF PH_PACKET.Indicate (Begin)
    THEN Issue PH_PACKET.Response
        Goto D2570
IF PH_HANGUP.Indicate
    THEN Issue PH_HANGUP.Response
        Goto D2520

A.5.8 D2570

A packet has been started. Wait for the first burst.

IF PH_TRANSFER.Indicate
    THEN Goto D2580
IF PH_HANGUP.Indicate

THEN Issue PH_HANGUP.Response
Goto D2520

A.5.9 D2580

The first burst has just been received. If there are errors in the first burst, then record it. Extract the FP header parameters, and if a D1_Data_Set is present then pass the D1_Area to the ULP. The actual size of the D1_Data_Set is defined or implied within the D1_Area.

Issue PH_TRANSFER.Response
IF PH_TRANSFER.Indicate (Status) = Error
THEN Set First_Burst_Error = true
Copy values from FP header
ULP-id = header.ULP-id
D2_Size = header.D2_Size
D2_Offset = header.D2_Offset
IF Header.D1_Data_Set_Present = true (i.e., P bit)
THEN Status = PH_TRANSFER.Indicate (Status)
Issue FP_TRANSFER_D1.Indicate
(ULP-id, CCI_Received, Status,
D2_Size, D2_Offset, D1_Area_Size,
D1_Area)
Goto D2590
ELSE Assemble burst into buffer
Goto D2590

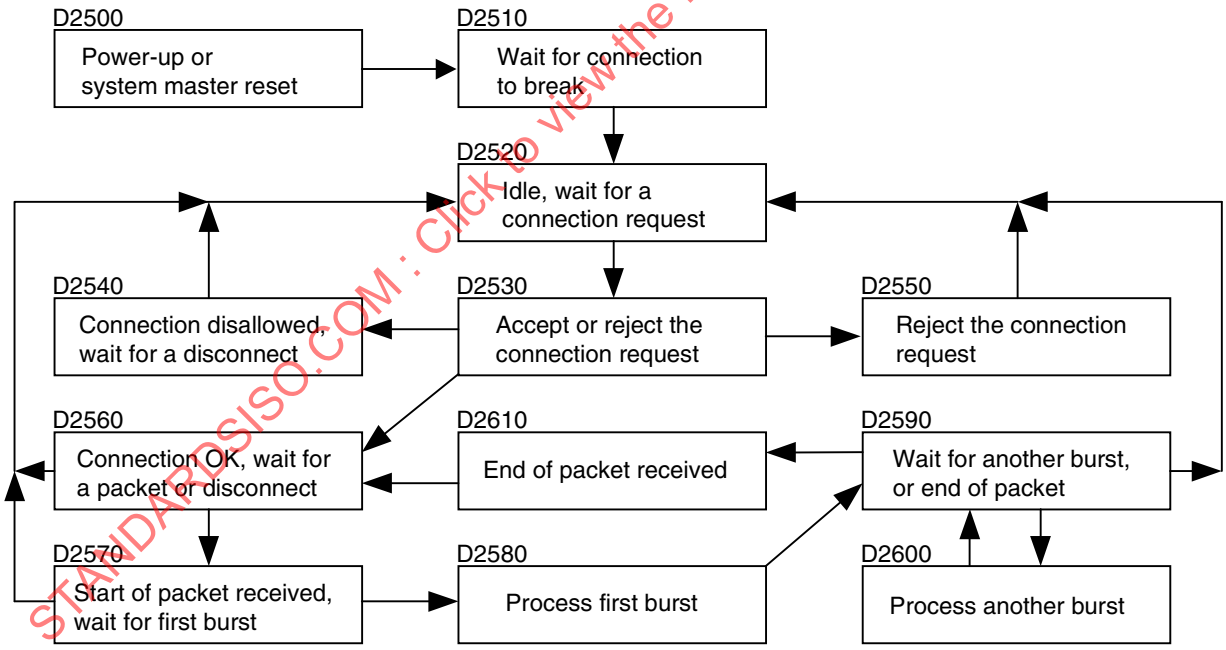

Figure A.2 – Destination flow diagram

NOTE 10 – There are cases where delivery of the packet, even with errors, is highly desirable; but the ULP should treat first burst errors with extreme caution. A first burst error could be in the FP header, e.g., any of the ULP-id, D1_Data_Set_Present, D1_Area_Size, D2_Offset, D2_Size, or Start_D2_on_Burst_Boundary, fields may be incorrect.

A.5.10 D2590

Wait for more bursts for this packet, or an end of packet indication.

IF PH_TRANSFER.Indicate
THEN Goto D2600
IF PH_PACKET.Indicate (End)

THEN Goto D2610
IF PH_HANGUP.Indicate
THEN Issue PH_HANGUP.Response
Goto D2520

A.5.11 D2600

Another burst has been received. Packet_Error indicates an error in other than the first burst of a packet.

Issue PH_TRANSFER.Response
IF PH_TRANSFER.Indicate (Status) = Error
THEN Set Packet_Error = true
Assemble burst into buffer
Goto D2590

A.5.12 D2610

The end of packet indication has been received. If the D2_Data_Set is present, then pass the D2_Data_Set to the ULP.

IF D2_Size not equal 0
THEN Status = First_Burst_Error, Packet_Error
Issue FP_TRANSFER_D2.Indicate (ULP-id,
CCI_Received, Status, D2_Size,
D2_Offset, D2_Data_Set)
Issue PH_PACKET.Response
Goto D2560

Annex B
(informative)

Implementation observations

B.1 Data transfer service primitive

The data transfer service primitive as defined in 5.4 assumes that the Source ULP knows the packet size. This may not always be the case. The primitives are an abstraction intended to clarify the operation of HIPPI-FP, they do not represent an implementation.

For example, the HIPPI-FP header D2_Size field has the option of denoting a packet of indefinite size by using the hexadecimal value of FFFFFFFF. This may be useful for transferring such things as pictures, where the interface is started and left running with one packet supplying many frames of picture data. Another example is transferring data from a magnetic tape of unknown format.

In these cases, the Source ULP may not know the complete size when it starts, and the Destination ULP does not care about the size. The Source would need some way to keep feeding data to the interface, and then signalling the end when complete. The Destination ULP would also probably accept the data in smaller portions, and not worry about receiving the end of packet before processing the data.

B.2 Classes of packets

Packets can be organized in a variety of ways to best take advantage of the hardware, firmware, and software implementations used, and the applications being supported. The HIPPI-PH define the different classes. This HIPPI-FP document supports all of the classes. Particular implementations may operate more efficiently with some classes than with others.

The abbreviations used in the figures are:

H = FP header (8 bytes)

D1_A = D1_Area (contains D1_Data_Set, if present)

(O) = 0 - 7 optional Offset bytes

D2_D = D2_Data_Set

(F) = Optional Fill

The D1_Area contains the D1_Data_Set, if present, and may contain additional pad bytes to fill out to the end of the D1_Area. It is assumed that the D1_Data_Set contains control information that is associated with the information in the D2_Data_Set.

The optional Offset bytes are used to start the D2_Data_Set at other than a 64-bit boundary. It is assumed that the D2_Data_Set contains user data.

The optional Fill bytes are used to pad out the D2_Data_Set to the end of a physical burst.

A full burst is either 1 Kbytes with the 32-bit HIPPI-PH, or 2 Kbytes with the 64-bit HIPPI-PH option.

B.2.1 Class 1, Single short burst

Figure B.1 illustrates class 1 transfers, in which a packet is composed of a single short burst, minimizing the number of CLOCK periods required to transfer data.