IEEE Std 1284-2000
(Revision of
IEEE Std 1284-1994)

IEEE Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers

Sponsor

Microprocessor and Microcomputer Standards Committee
of the
IEEE Computer Society

Approved 21 September 2000

IEEE-SA Standards Board

Abstract: A signaling method for asynchronous, fully interlocked, bidirectional parallel communications between hosts and printers or other peripherals is defined. A functional subset of the signaling method may be implemented on personal computers (PCs) or equivalent parallel port hardware with new software. New electrical interfaces, cabling, and interface hardware that provides improved performance while retaining backward compatibility with this subset is detailed.
Keywords: bidirectional parallel communications, computers, interfaces, PCs, personal computers, printers

---

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Print: ISBN 0-7381-2615-2 SH94880
PDF: ISBN 0-7381-2616-0 SS94880

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

Note: Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

IEEE is the sole entity that may authorize the use of certification marks, trademarks, or other designations to indicate compliance with the materials set forth herein.

Introduction

(This introduction is not a part of IEEE Std 1284-2000, IEEE Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers.)

This standard was formally started as an IEEE effort in January 1992, but without the advance work done by a loose alliance of printer manufacturers and printer software developers called the Network Printing Alliance, this standard would not be possible.

The following lists the key contributors to the revision of this standard:

Forrest D. Wright, *Chair*
Larry Stein, *Secretary*
Lance Spaulding, *Editor*

Darrell Cox
Lee Farrell
Robert Gross
Laurie Lasslo
Hideki Morozumi
Bill Myntti
Fumio Nagasaka
David Roach
Bill Russell
Greg Shue
Jerry Thrasher
Rick Vander Wegen
Craig Whittle

The following members of the balloting committee voted on this standard:

Steven R. Bard
Lon Canaday
Keith Chow
James R. Davis
Sourav K. Dutta
Edwin Vivian El-Kareh
Lee Farrell
Thomas M. Kurihara
Joseph R. Marshall
Gene E. Milligan
Robert Mortonson
Klaus-Dieter Mueller
Atsushi Nakamura
Roman Orzol
Gary S. Robinson
David Rockwell
Jaideep Roy
Thomas J. Schaal
Akihiro Shimura
Larry Stein
Joseph Tardo
Michael G. Thompson
Forrest D. Wright
Oren Yuen

When the IEEE-SA Standards Board approved this standard on 21 September 2000, it had the following membership:

Donald N. Heirman, *Chair*
James T. Carlo, *Vice Chair*
Judith Gorman, *Secretary*

Satish K. Aggarwal
Mark D. Bowman
Gary R. Engmann
Harold E. Epstein
H. Landis Floyd
Jay Forster*
Howard M. Frazier
Ruben D. Garzon
James H. Gurney
Richard J. Holleman
Lowell G. Johnson
Robert J. Kennelly
Joseph L. Koepfinger*
Peter H. Lips
L. Bruce McClung
Daleep C. Mohla
James W. Moore
Robert F. Munzner
Ronald C. Petersen
Gerald H. Peterson
John B. Posey
Gary S. Robinson
Akio Tojo
Donald W. Zipse

*Member Emeritus

Also included is the following nonvoting IEEE-SA Standards Board liaison:

Alan Cookson, *NIST Representative*
Donald R. Volzka, *TAB Representative*

Greg Kohn
IEEE Standards Project Editor

Contents

IEEE Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers

1. Overview

1.1 Scope

This standard defines a signaling method for asynchronous, fully interlocked, bidirectional parallel communications between hosts and printers or other peripherals. A functional subset of the signaling method may be implemented on personal computers (PCs) or equivalent parallel port hardware with new software. This standard recommends new electrical interfaces, cabling, and interface hardware that provides improved performance while retaining backward compatibility with this subset.

This standard also specifies a format for a peripheral identification string and a method of returning this string to the host outside of the bidirectional data stream.

1.2 Purpose

This standard was created because there existed no defined standard for bidirectional parallel communications between personal computers and printing peripherals. Pre-existing methods utilized a wide variety of hardware and software products, each with unique and incompatible signaling schemes. This standard was developed to provide an open path for communications between computers and more intelligent printers and peripherals. The availability of a standard bidirectional protocol will encourage the development of new peripherals that return significant data, as well as basic status, to the host.

2. References

This standard shall be used in conjunction with the following publication. If the following publication is superseded by an approved revision, the revision shall apply.

IEC 60950 (1999-04), Safety of information technology equipment.[1]

3. Definitions

3.1 General terminology

The following terms are used in this standard and/or may be used in conjunction with the signaling methods defined in this standard. The definitions are not intended to be absolute, but they do reflect the use of the terms in this standard.

3.1.1 bidirectional operation: When the peripheral and host communicate using both forward and reverse data channels. As defined in this standard, Nibble and Byte Modes provide reverse channel communication and are used in conjunction with Compatibility Mode to provide bidirectional operation. Extended Capabilities Port (ECP) and Enhanced Parallel Port (EPP) Modes support bidirectional communication.

3.1.2 Centronics: The popular name for the parallel printer port used as the parallel interface for most printers and supported by most "MS-DOS compatible" PCs. The name is derived from the printer manufacturer that introduced this interface, Centronics Data Computer Corporation. Despite a basic similarity, many variations of this interface have been implemented in different peripherals and hosts. This specification describes the more prevalent variations of the "Centronics" interface and defines a family of signaling methods that are backward compatible with the typical "Centronics" interface.

3.1.3 Centronics connector: The popular name for the 36-pin ribbon contact type connector commonly used for the parallel port on printers. This connector is referred to as the "IEEE 1284-B" connector in this standard.

3.1.4 command set: A field in the Device ID message identifying the type of data expected by the peripheral. For example, a printer might use this field to report which page description language(s) it supports.

3.1.5 compatible device: A device that supports any of a specified range of popular variants of the "Centronics" interface. Compatible devices will interoperate with compliant devices in Compatibility Mode only.

3.1.6 compliant device: A device that supports either the Level 1 or Level 2 electrical interface, plus Compatible and Nibble Mode operation, as well as the negotiation phases necessary to transition between these two modes.

3.1.7 device driver: A program that runs on the host and manages the sending and receiving of information from the peripheral. The driver utilizes the link level interface defined in this standard to communicate data between the application program and the peripheral personality.

3.1.8 Device ID: A structured, variable length ASCII message identifying the manufacturer, command set, and model of the peripheral. Additional information may also be included. The message is provided by the

[1]IEC publications are available from the Sales Department of the International Electrotechnical Commission, Case Postale 131, 3, rue de Varemb , CH-1211, Gen ve 20, Switzerland/Suisse (http://www.iec.ch/). IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

peripheral in response to a request issued by the host during the negotiation phase. Provided that the peripheral supports the bidirectional mode requested by the host, this message is provided in the requested mode. The Device ID is intended to assist the host in selecting the device and/or peripheral driver appropriate to the peripheral.

3.1.9 forward channel: Data path from the host to the peripheral.

3.1.10 host: A device, typically a personal computer, that will control the communications with attached peripherals.

3.1.11 IEEE 1284-compatible device: *See:* **compatible device.**

3.1.12 IEEE 1284-compliant device: *See:* **compliant device.**

3.1.13 IEEE 1284-A connector: A plug or receptacle 25-pin subminiature D-shell connector, as specified in 8.2 and shown in Figure 24 of IEEE Std 1284-2000. This is the type of connector used on the MS-DOS compatible PC printer port adapter.

3.1.14 IEEE 1284-B connector: A plug or receptacle 36-pin ribbon type connector, as specified in 8.2 and shown in Figure 25 of IEEE Std 1284-2000. This type of connector is also known as a "Centronics Connector."

3.1.15 IEEE 1284-C connector: A miniature plug or receptacle 36-pin ribbon type connector, as specified in 8.2 and shown in Figure 26 of IEEE Std 1284-2000.

3.1.16 Level 1 device: A device that supports the Level 1 electrical interface.

3.1.17 Level 2 device: A device that supports the Level 2 electrical interface.

3.1.18 link: The physical connection and electronic hardware by which data is transferred between the host and the peripheral. This standard is concerned only with the link layer; the only information transfer defined or required as part of this standard is that necessary to control and synchronize the communication of peripheral data. The defined interface is transparent to the peripheral data communicated at data or higher layers.

3.1.19 n: When preceding a capitalized signal name, denotes a signal having negative true logic (e.g., nAck).

3.1.20 PC parallel port: The parallel printer port used as the parallel interface for most printers and supported by most (personal computers) PCs. This interface has been variously defined by different PC and peripheral manufacturers. This standard describes the more prevalent variations of the interface and defines a family of signaling methods that are backward compatible with the typical PC parallel port.

3.1.21 peripheral: A device, attached to a host via a communication link.

3.1.22 peripheral personality: The characteristics of a language processor or operating environment that a peripheral runs to interpret commands and data being sent.

3.1.23 reverse channel: Data path from the peripheral to the host.

3.1.24 solicited status: Information generated by the peripheral in response to a command from the host.

3.1.25 status: A term used generally to describe data generated by the peripheral that reflects the current operating state of the peripheral.

3.1.26 status lines: Unidirectional signals from the peripheral to the host, defined in Compatibility Mode to handshake data and to report error conditions. In other IEEE 1284 interface modes defined in this standard (IEEE Std 1284-2000), these lines are used for control, data, and/or status.

3.1.27 unidirectional operation: When the peripheral and host communicate data in one direction only. Compatibility Mode is unidirectional in the forward direction; Byte and Nibble Modes are unidirectional in the reverse direction.

3.1.28 unsolicited status: Information generated by the peripheral that has not been asked for by the host, yet is important enough that the peripheral desires to send it to the host.

3.2 Communication modes

NOTE—The signaling method described herein provides for several modes of host-peripheral communication. The interface is initially in Compatibility Mode. Other modes provide additional features and/or improved performance. Compliant devices can determine and switch to the most effective mode supported by both devices. All compliant devices are required to support Compatibility Mode, Nibble Mode, and the negotiation necessary to switch between communication modes.

3.2.1 Compatibility Mode: An asynchronous, byte-wide forward (host-to-peripheral) channel with data and status lines used according to their original definitions. Compatibility Mode is backward compatible with many existing devices, including the PC parallel port, and is the base mode common to all compliant interfaces.

3.2.2 Nibble Mode: An asynchronous, reverse (peripheral-to-host) channel, under control of the host. Data bytes are transmitted as two sequential 4-bit nibbles using four peripheral-to-host status lines. Nibble Mode is used with Compatibility Mode to implement a bidirectional channel. These two modes cannot be active simultaneously.

3.2.3 Byte Mode: An asynchronous, byte-wide reverse (peripheral-to-host) channel using the eight data lines of the interface for data and the control/status lines for handshaking. Byte Mode is used with Compatibility Mode to implement a bidirectional channel, with transfer direction controlled by the host when the host and peripheral both support bidirectional use of the data lines. The two modes cannot be active simultaneously.

3.2.4 Extended Capabilities Port (ECP) Mode: An asynchronous, byte-wide, bidirectional channel. An interlocked handshake replaces the minimum timing requirements of Compatibility Mode. A control line is provided to distinguish between command and data transfers. A command may optionally be used to indicate single-byte data compression or channel address.

3.2.5 Enhanced Parallel Port (EPP) Mode: An asynchronous, byte-wide, bidirectional channel controlled by the host device. This mode provides separate address and data cycles over the eight data lines of the interface.

3.3 Operating phases

NOTE—Interface operation is divided into a number of interface phases. Each communication mode consists of one or more phases. Additional phases are defined to cover initialization and transitions between communication modes. The names and interpretations of the interface signals may vary between phases, but are consistent within each phase.

3.3.1 Compatibility Mode phases

3.3.1.1 Compatibility Mode forward data transfer phase: Begins when the host asserts nStrobe and ends following data hold time and nStrobe de-assertion. (Note that the host is not free to send the next data byte

until the peripheral acknowledges the transfer using nAck.) The host may not initiate negotiation to a new operating mode until the interface returns to Compatibility Mode forward idle phase.

3.3.1.2 Compatibility Mode forward idle phase: When the interface is in Compatibility Mode with no data transfer in progress. The host may initiate a data transfer in Compatibility Mode or may initiate negotiation to a new operating mode.

3.3.2 Nibble and Byte Mode phases

3.3.2.1 reverse data transfer phase: When data transfers from the peripheral to the host.

3.3.2.2 reverse host busy data available phase: When the peripheral has data to transmit.

3.3.2.3 reverse host busy data not available phase: When the peripheral has no more data to transmit.

3.3.2.4 reverse idle phase: When no data transfer is in progress and the host is waiting for peripheral data. When data are available, the peripheral will cause the interface to go to the reverse interrupt phase.

3.3.2.5 reverse interrupt phase: A phase that provides the mechanism for the peripheral to alert the host that it has data to transfer. While in this phase, the host may cause the interface to go to the termination phase.

3.3.3 ECP Mode phases

3.3.3.1 ECP setup phase: A phase that sets the interface signals to the correct state for the ECP forward transfer phase. It immediately follows the negotiation phase.

3.3.3.2 ECP forward transfer phase: When the host transfers data or commands to the peripheral using HostClk. The peripheral may indicate its desire to send data to the host by asserting nPeriphRequest.

3.3.3.3 ECP forward idle phase: When no data transfer is in progress. The peripheral may indicate its desire to communicate with the host by asserting nPeriphRequest. The host may cause the interface to go to the forward to reverse phase, forward transfer phase, or the termination phase.

3.3.3.4 ECP forward to reverse phase: When the interface is changing from the forward direction to the reverse direction.

3.3.3.5 ECP reverse transfer phase: When the peripheral transfers data or commands to the host. The host may interrupt the peripheral, causing the interface to go to the reverse to forward phase.

3.3.3.6 ECP reverse idle phase: When no data transfer is in progress. The host may interrupt the peripheral, causing the interface to go to the reverse to forward phase. The peripheral may cause the interface to go to the reverse transfer phase or reverse to forward phase.

3.3.3.7 ECP reverse to forward phase: When the interface is changing from the reverse direction to the forward direction.

3.3.4 EPP mode phases

3.3.4.1 EPP address read phase: When the host is performing an address read transfer operation.

3.3.4.2 EPP address write phase: When the host is performing an address write transfer operation.

3.3.4.3 EPP data read phase: When the host is performing a data read transfer operation.

3.3.4.4 EPP data write phase: When the host is performing a data write transfer operation.

3.3.4.5 EPP initial idle phase: A transition phase from negotiation phase to EPP Mode. After a transfer operation (address or data read or write), the interface will enter the idle phase.

3.3.4.6 EPP idle phase: When no address or data transfer is in progress.

3.3.4.7 EPP termination phase: A host-initiated transition phase in which the interface is changed from EPP Mode to Compatibility Mode.

3.3.5 Additional phases

3.3.5.1 initialization phase: A phase that includes both power-on initialization and host-driven interface reset.

3.3.5.2 negotiation phase: Signal handshaking to change the signaling method from Compatibility Mode to Nibble, Byte, ECP, or EPP Mode.

3.3.5.3 power-on phase: A phase that includes power-on initialization for both devices.

3.3.5.4 termination phase: A host-initiated transition phase in which the interface is changed from Nibble, Byte, or ECP Mode to Compatibility Mode.

4. Features and compliance

4.1 Interface overview

This interface is a bidirectional extension of the existing PC parallel interface (commonly known as the Centronics interface). The bidirectional communication occurs using the signals in the existing interface. The interface supports a number of distinct communication modes, and the interpretation of interface signals depends on the current mode. Compatibility Mode provides host-to-peripheral communication in a manner compatible with the traditional unidirectional interface. Nibble and Byte Modes provide peripheral-to-host communication and may be combined with Compatibility Mode to provide bidirectional operation. ECP Mode provides symmetric bidirectional communications without the overhead of changing communication modes. EPP Mode provides asymmetric bidirectional data transfer driven by the host device. Not all devices support all communication modes. A mechanism is provided for the host and peripheral to negotiate the mode to be used for data transfers and to renegotiate as needed.

In Compatibility Mode, the host sends 8 bits to the peripheral over the interface data signals D1 through D8. In Nibble Mode, peripheral-to-host data bytes are sent as two sequential nibbles on the parallel port status lines. In Byte Mode, ECP Mode, or EPP Mode, peripheral-to-host data bytes are sent on the parallel port data signals. Each of these arrangements gives the interface at least two separate channels, one in each direction. Having two channels allows peripheral-to-host data to be transferred even when the host-to-peripheral data channel is blocked.

In the Nibble or Byte Mode, the host requests a reverse channel transfer, and the peripheral responds by indicating whether there is data to be sent. If no data is ready to be sent, the host can enter the reverse idle phase, in which the peripheral will indicate to the host when data becomes available. In the ECP or EPP Modes, the host may read or write address or data information from or to the peripheral. The host may return the link to the Compatibility Mode whenever the interface is in a “valid” state, which is uniquely specified for each mode.

4.2 Features

This interface provides several capabilities to the user. These capabilities satisfy the following list of user objectives:

a) This interface provides the capability to send data from the host computer to the peripheral at a higher speed than previously done. This is accomplished by shortening the signal timing values where possible.
b) This interface provides a path for data to be sent from the peripheral to the host using existing and future parallel port hardware.
c) The reverse channel capability reduces the amount of user interaction needed to operate the peripheral. For example, application programs can ask a printer for a list of fonts currently available. The application can then decide if new fonts must be downloaded, rather than ask the user.

4.3 Device compatibility criteria

Devices that meet the mechanical interface requirements of 8.2, the electrical requirements of 8.3.2, and the protocol compatibility requirements of 7.3, but neither the compliance requirements of that subclause nor the additional protocols of 7.4 et al., are referred to as “IEEE 1284-compatible” devices. *IEEE 1284-compatible devices are not IEEE 1284 compliant,* but they meet the minimum requirements for operation in Compatibility Mode with any IEEE 1284-compliant device.

The popular parallel port implementations vary widely in the timing of the nStrobe, nAck, and Busy signals. For instance, many existing hosts ignore the nAck signal entirely, relying on Busy to control the flow of data to the peripheral. The IEEE 1284 protocol compatibility requirements are sufficient to ensure operation of an IEEE 1284-compatible device with an IEEE 1284-compliant device, but not to ensure interoperation of two IEEE 1284-compatible devices. Conversely, the IEEE 1284 protocol compliance requirements ensure operation of IEEE 1284-compliant devices with both IEEE 1284-compliant and IEEE 1284-compatible devices.

Historically, the published specifications of the popular variants of the "Centronics-compatible" interface (see Annex C) specified a single "logic high" and a single "logic low" level for devices on both ends of the interface, leaving no operational margin for noise or losses in the connecting cable. In practical applications, the actual operating characteristics often exceeded these minimum requirements.

4.4 Device compliance criteria

This standard defines two levels of device compliance, as follows:

a) An IEEE 1284-I compliant device shall have a mechanical interface using IEEE 1284-A and IEEE 1284-B connectors, have a Level 1 electrical interface, and abide with the protocol compliance criteria, all of which are listed in 4.4.3.
b) An IEEE 1284-II compliant device shall have a mechanical interface using IEEE 1284-C connectors, have a Level 2 electrical interface, and abide with the protocol compliance criteria, all of which are listed in 4.4.3.

4.4.1 Mechanical

IEEE 1284 devices shall use IEEE 1284-A, IEEE 1284-B, or IEEE 1284-C connectors in one of the configurations depicted in Clause 8. The IEEE 1284-C connectors are recommended for all new designs.

4.4.2 Electrical

Level 1 electrical requirements provide sufficient margin to assure operation with any compatible or Level 1 compliant device. See 8.3.2 for a complete list of requirements.

Level 2 requirements ensure that all IEEE 1284 Level 2 compliant devices will interoperate with any IEEE 1284 Level 2 compliant device and in general with IEEE 1284 Level 1 compliant devices. See 8.3.3 for a complete list of requirements.

It is recommended that all new designs implement Level 2 drivers and receivers.

4.4.3 Protocol

Compliant devices shall implement Compatibility and Nibble Modes as defined within this standard, as well as the negotiation phases necessary to transition between these two modes. Implementation of Byte, ECP, and/or EPP Modes (also defined within this standard) is recommended but not required for IEEE 1284 compliance.

Peripherals may, but are not required to, implement the Device ID string as specified in 7.6. Peripherals that implement the Device ID string shall be able to return the ID string to the host in the Nibble Mode. It is recommended that peripherals be able to return the ID string to the host using any of the reverse channel modes implemented by the device with the exception of EPP.

4.5 Cable compliance criteria

Interconnection cables intended for use with IEEE 1284 devices shall meet the mechanical and electrical requirements of Clause 8. Cable assemblies that meet these requirements shall be clearly and permanently labeled “IEEE Std 1284-2000 compliant” to distinguish them from cables having the same type connectors but different electrical characteristics.

5. Interface signals

NOTE—IEEE Std 1284-2000 mode signal names are shown with their Compatibility Mode names in parentheses. Usage of mode-specific signal names imply the interface is operating in that mode.

5.1 HostClk/nWrite (nStrobe): host driven

Compatibility Mode: Set active low to transfer data into the input latch of the peripheral. Data is valid while nStrobe is low.

negotiation phase: Set active low to transfer the extensibility request value into the input latch of the peripheral. Data is valid on the leading (falling) edge of HostClk.

reverse data transfer phase: Nibble Mode: Set high during transfers to avoid latching data into the peripheral. Byte Mode: Pulsed low during transfers to acknowledge transfer of data from the peripheral. The peripheral shall ensure that this pulse does not transfer a new data byte into the input latch of the peripheral.

ECP Mode: Used in a closed-loop handshake with PeriphAck(Busy) to transfer data or address information from the host to the peripheral.

EPP Mode: Set low to denote an address or data write operation to the peripheral. Set high to denote an address or data read operation from the peripheral.

5.2 AD1...AD8 (Data1...Data8)

Driven by the host in Compatibility Mode and the negotiation phase, not used in Nibble Mode, and bidirectional in all other modes.

All modes: Data 1 is the least significant bit (bit 0).
Data 8 is the most significant bit (bit 7).

Compatibility Mode: Forward channel data.

negotiation phase: Extensibility request value.

reverse data transfer phase: Nibble Mode: Not used (host may still drive bus). Byte Mode: Reverse channel data.

ECP Mode: Host-to-peripheral or peripheral-to-host address or data.

EPP Mode: Host-to-peripheral or peripheral-to-host address or data.

5.3 PtrClk/PeriphClk/Intr (nAck): peripheral driven

Compatibility Mode: Pulsed low by the peripheral to acknowledge transfer of a data byte from the host.

negotiation phase: Set low to acknowledge IEEE 1284 support, then set high to indicate that the Xflag(Select) and data available flags may be read.

reverse data transfer phase: Nibble and Byte Modes: Used to qualify data being sent to the host.

reverse idle phase: Set low, then high by the peripheral to cause an interrupt indicating to the host that data is available.

ECP Mode: Used in a closed-loop handshake with HostAck(nAutoFd) to transfer data from the peripheral to the host.

EPP Mode: Used by the peripheral to interrupt the host. This signal is pulsed low for a minimum of 500 ns and must be raised for a minimum of 1 μs before reasserting.

5.4 PtrBusy/PeriphAck/nWait (busy): peripheral driven

Compatibility Mode: Driven high to indicate that the peripheral is not ready to receive data.

negotiation phase: Reflects the present state of the forward channel of the peripheral.

reverse data transfer phase: Nibble Mode: Data bit 3 then 7, then forward channel busy status. Byte Mode: forward channel busy status.

reverse idle phase: Forward channel busy status.

ECP Mode: The peripheral uses this signal for flow control in the forward direction. PeriphAck also provides a ninth data bit that is used to determine whether command or data information is present on the data signals in the reverse direction.

EPP Mode: This signal should be driven inactive as a positive acknowledgment from the peripheral that transfer of data or address is completed. The signal is active low. It should be driven active as an indication that the device is ready for the next address or data transfer.

5.5 AckDataReq/nAckReverse (PError): peripheral driven

Compatibility Mode: Driven high to indicate that the peripheral has encountered an error in its paper path. The meaning of this signal varies from peripheral to peripheral. Peripherals shall set nFault low whenever PError is set high.

negotiation phase: Set high to indicate IEEE 1284 support, then follows nDataAvail(nFault).

reverse data transfer phase: Nibble Mode: Data bit 2 then 6. Byte Mode: Same as nDataAvail(nFault).

reverse idle phase: Set high until the host requests a data transfer, then follows nDataAvail(nFault).

ECP Mode: The peripheral drives this signal low to acknowledge nReverseRequest. The host relies upon nAckReverse to determine when it is permitted to drive the data signals.

EPP Mode: (User Defined 1) This signal is manufacturer specific and beyond the scope of this standard.

5.6 Xflag (Select): peripheral driven

Compatibility Mode: Set high to indicate that the peripheral is online.

negotiation phase: The name Xflag refers to extensibility flag. It is used by the peripheral to reply to the requested extensibility byte sent by the host during the negotiation phase. The signal level used to indicate an affirmative response for each respective extensibility byte is shown in Table 4.

reverse data transfer phase: Nibble Mode: Data bit 1 then 5. Byte Mode: Same as negotiation phase.

reverse idle phase: Same as negotiation phase.

ECP Mode: Same as negotiation phase.

EPP Mode: (User Defined 3) This signal is manufacturer specific and beyond the scope of this standard.

5.7 HostBusy/HostAck/nDStrb (nAutoFd): host driven

Compatibility Mode: The interpretation of this signal varies from peripheral to peripheral. Set low by host to put some printers into auto-line feed mode. May also be used as a ninth data, parity, or command/data control bit.

negotiation phase: Set low in conjunction with IEEE 1284 Active(nSelectIn) being set high to request a IEEE 1284 mode. Then set high after the peripheral sets PtrClk(nAck) low.

reverse data transfer phase: Nibble Mode: Set low to indicate that the host can receive peripheral-to-host data, then set high to acknowledge receipt of that nibble. Byte Mode: Same as Nibble Mode to request and acknowledge bytes. Following a reverse channel transfer, the interface transitions to idle phase when Host-Busy(nAutoFd) is set low and the peripheral has no data available.

reverse idle phase: Set high in response to a PtrClk(nAck) low pulse to re-enter reverse data transfer phase. If set high with IEEE 1284 Active(nSelectIn) set low, the IEEE 1284 idle phase is aborted, and the interface returns to Compatibility Mode.

ECP Mode: The host drives this signal for flow control in the reverse direction. It is used in an interlocked handshake with PeriphClk(nAck). HostAck also provides a ninth data bit that is used to determine whether command or data information is present on the data signals in the forward direction.

EPP Mode: This signal is active low. It is used to denote a data cycle.

5.8 Peripheral Logic High: peripheral driven

Set high to indicate (subject to the provisions of 8.3.5) that all other signals sourced by the peripheral are in a valid state. Set low to indicate that the peripheral power is off or that peripheral-driven interface signals are otherwise in an invalid state.

Peripheral manufacturers may, but are not required to, use this signal to provide 5 V of power to an attached device. In any case, the peripheral shall limit the short-circuit current to a maximum of 1.0 A and shall provide circuitry to ensure a valid logic low level (as defined in 8.3.5) on this signal when the peripheral power is off.

5.9 nReverseRequest (nInit): host driven

Compatibility Mode: Pulsed low in conjunction with IEEE 1284 Active low to reset the interface and force a return to Compatibility Mode idle phase.

negotiation phase: Set high.

reverse data transfer phase: Set high.

ECP Mode: This signal is driven low to place the channel in the reverse direction. While in the ECP Mode, the peripheral is only allowed to drive the bidirectional data signals when nReverseRequest is low and IEEE 1284 Active is high.

EPP Mode: This signal is active low. When driven active (low), this signal initiates a termination cycle that results in the interface returning to Compatibility Mode.

5.10 nDataAvail/nPeriphRequest (nFault): peripheral driven

Compatibility Mode: Set low by the peripheral to indicate that an error has occurred. The meaning of this signal varies from peripheral to peripheral.

negotiation phase: Set high to acknowledge IEEE 1284 compatibility. In Nibble or Byte Mode, it is then set low to indicate peripheral-to-host data is available following the host setting HostBusy(nAutoFd) high.

reverse data transfer phase: Nibble Mode: Set low to indicate that the peripheral has the data ready to send to the host. Then used to send data bit 0 then 4. Byte Mode: Used to indicate that data is available.

reverse idle phase: Used to indicate that data is available.

ECP Mode: During ECP Mode, the peripheral shall drive this pin low to request communications with the host. This signal provides a mechanism for peer-to-peer communication. This signal would be typically used to generate an interrupt to the host. This signal is valid in the forward and reverse directions.

EPP Mode: (User Defined 2) This signal is manufacturer specific and beyond the scope of this standard.

5.11 1284 Active/nAStrb (nSelectIn): host driven

Compatibility Mode: Set low by host to select peripheral.

negotiation phase: Set high in conjunction with Host Busy set low to request an IEEE 1284 mode.

reverse data transfer phase: Set high to indicate that the bus direction is peripheral to host. Set low to terminate IEEE 1284 mode and to set bus direction to host to peripheral.

reverse idle phase: Same as reverse data transfer phase.

ECP Mode: Driven high by the host while in ECP Mode. Set low by the host to terminate ECP Mode and return the link to the Compatibility Mode.

EPP Mode: This signal is used to denote an address cycle. It is active low.

5.12 Host Logic High: host driven

Set high to indicate (subject to the provisions of 8.3.5) that all other signals sourced by the host are in a valid state. Set low to indicate the host power is off or host-driven interface signals are otherwise in an invalid state.

Host manufacturers may, but are not required to, use this signal to provide 5 V of power to an attached device. In any case, the host shall limit the short-circuit current to a maximum of 1.0 A and shall provide circuitry to ensure a valid logic low level (as defined in 8.3.5) on this signal when the host power is off.

6. Interface concepts

6.1 Link level and data level separation

The concepts of physical, link, and data levels originate from the open systems interconnect (OSI) model developed under the auspices of the International Organization for Standardization (ISO) and promulgated as ISO 7498:1984 [B13].[2] The IEEE 1284 interface separates the operation of the link between the host and peripheral from the data communicated over that link. This allows link layer functions, such as IEEE 1284 communication mode negotiation, to be localized in an interface driver that transports application data without having to interpret the data stream (see Figure 1).

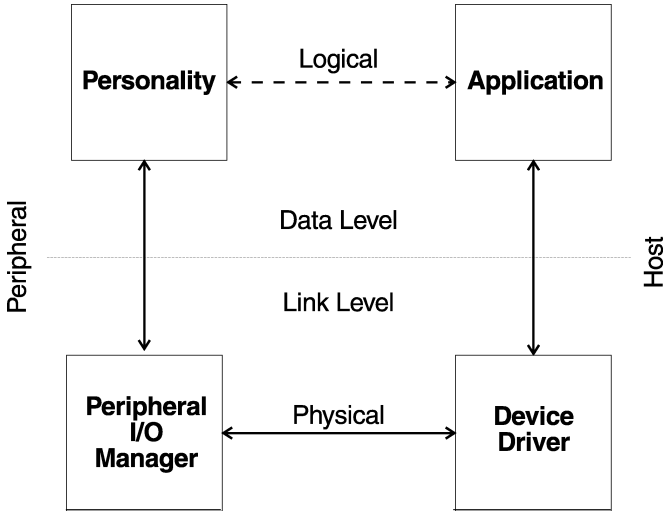


Figure 1—Separation of data level and link level

Ideally, the application and the peripheral personality should have an open dialogue that allows them to exchange information as needed. The IEEE 1284 interface provides a means by which this dialogue may be achieved. There are differences in the way the application and the personality communicate, depending on the device hardware and on the direction information is flowing. The interface drivers in the host and the peripheral deal with the details of selecting an appropriate communications mode or modes, hiding the interface dependencies from the application and peripheral personalities.

6.2 IEEE 1284 communication options

The IEEE 1284 negotiation phase allows the host and peripheral to select a mutually supported communication mode. During the negotiation phase, the host indicates which communication mode and options it would like to use via the Extensibility Request Value (see Table 4). There is one bit in the Extensibility Request Value for each IEEE 1284 mode, plus additional bits for communications options such as device identification (see 7.6) and ECP Mode Run Length Encoding (RLE).[3] The peripheral indicates that it supports all requested options by setting the Extensibility Flag low (for Nibble Mode without options) or high (for any other IEEE 1284 mode or options) as shown in Table 4. If the peripheral does not support the

[2]The numbers in brackets correspond to those of the bibliography in Annex D.

[3]The host shall only request one communication mode option at a time, i.e., Byte, ECP, etc. The Device ID may be requested using Nibble, Byte, or ECP Modes. RLE may be requested when using the ECP Mode.

requested mode or options, it sets the Extensibility Flag low,[4] and the interface returns to Compatibility Mode.

6.3 Nibble Mode/Byte Mode transfer

Many personal computers do not have the hardware necessary to receive data over the eight data lines. Nibble Mode allows peripheral-to-host data to be transferred over the parallel port status lines 4 bits at a time. Byte Mode allows the peripheral to send data to the host on the eight data lines. The host indicates its Nibble Mode/Byte Mode capability during the negotiation phase. It is required that all IEEE 1284 devices support Nibble Mode to guarantee that any two IEEE 1284 devices can communicate.

6.4 Host-initiated transfers

The host initiates all transfers in the Compatibility, Nibble, Byte, and EPP Modes and host-to-peripheral transfers in the ECP Mode except as noted in 6.5. Arbitration between ECP forward and reverse phases is controlled by the host.

6.5 Peripheral-initiated transfers

The peripheral initiates peripheral-to-host data transfers in the Nibble and Byte Modes by pulsing the PtrClk(nAck) line. The peripheral is only permitted to initiate a reverse channel transfer during the reverse idle phase in order to prevent non-IEEE 1284 hosts from interpreting the PtrClk(nAck) pulse as a request for more data from the host.

The peripheral requests use of the data bus in ECP Mode by asserting the nPeriphRequest (nFault) signal. After the host grants the request, the peripheral initiates data transfers in ECP Mode reverse phase.

The host initiates all transfers in EPP Mode.

6.6 Multiple byte transfers

The IEEE 1284 interface allows multiple bytes to be transferred from the peripheral to the host without having to renegotiate to an IEEE 1284 mode. For the Nibble and Byte Modes, the nDataAvail(nFault) signal indicates whether there is a subsequent byte ready to transfer. In the ECP Mode, the peripheral may transfer as many bytes as needed to the host, provided that the host does not issue a request to return the channel to the forward direction.

6.7 Interface errors

Errors can result from timeouts, invalid states, and invalid transitions between states. These errors can be detected by either the host computer or the peripheral. When either the host or peripheral detects an error, it shall terminate the current transfer.

When the host or peripheral detects an error, it should immediately abort (with no termination phase) and resume Compatibility Mode operation.

In particular, to protect against bus fight conditions on the bidirectional data signals, the peripheral shall stop driving the data bus within 1 μs in the event of a protocol exception. The peripheral should carefully moni-

[4]This sequence of operations was chosen so that compliant peripherals (which support Nibble Mode, but no other IEEE 1284 modes or options) can set the Extensibility Flag low without having to examine the Extensibility Request Value.

tor both IEEE 1284 Active (nSelectIn) and nReverseRequest (nInit) to detect when the host has aborted to Compatibility Mode.

NOTE—Immediate termination upon error conditions is not supported in EPP Mode.

6.8 Peripheral error resolution

Some peripherals are subject to errors or exception conditions that require operator intervention before they can accept any more data from the host (for example, a printer might run out of paper). Peripherals may also be subject to internal processing delays of indefinite length. The IEEE 1284 interface allows the host to request an IEEE 1284 mode reverse transfer in order to resolve whatever problem exists on the peripheral. If an error message has been generated by the peripheral, the host may retrieve it and report the condition to the application and/or user.

6.9 ECP Mode command/data

ECP Mode supports two advanced features to improve the effectiveness of the protocol for some applications. The features are implemented by allowing the transfer of 8 bit data or 8 bit commands.

When in the forward direction, data is transferred when HostAck is high, and an 8 bit command is transferred when HostAck is low. The most significant bit of the command indicates whether it is a run-length count (for compression) or a channel address.

Table 1—Forward channel commands (when HostAck is low)

Bit 7	Bits 6..0
0	Run-length count (0–127) (mode 0011 0X00 only)
1	Channel address (0–127)

When in the reverse direction, data is transferred when PeriphAck is high, and an 8 bit command is transferred when PeriphAck is low.

Table 2—Reverse channel commands (when PeriphAck is low)

Bit 7	Bits 6..0
0	Run-length count (0–127) (mode 0011 0X00 only)
1	Channel address (0–127)

Devices indicate that they support decompression via the negotiation sequence. Devices that negotiate into ECP RLE mode (data 0011 0X00) shall support decompression of RLE data and may optionally compress it. Devices that negotiate into the non-RLE ECP Mode *shall not* transfer ECP Mode RLE compressed data.

6.9.1 ECP Mode data compression

Single-byte run-length-encoding is supported, which compresses strings of identical bytes. The compression is particularly useful on raster imaging devices. This simple compression does not preclude the use of other data compression schemes at a higher (data stream or packet) level.

When a run-length count is received, the subsequent data byte is replicated the specified number of times. A run-length count of zero specifies that only 1 byte of data is represented by the next data byte, whereas a run-length count of 127 indicates that the next byte should be expanded to 128 bytes. To prevent data expansion, however, run-length counts of zero should be avoided.

6.9.2 ECP Mode channel addressing

A channel addressing scheme is used in ECP Mode, providing 128 forward and reverse channel addresses. The channel addresses may be dynamically changed while in ECP Mode. Channel address definitions are device specific. The host and peripheral channel addresses default to zero after each negotiation from Compatibility Mode into ECP Mode. After a channel address command is issued from the host, that address becomes the current channel address for both host-to-peripheral and peripheral-to-host communications until another channel address command is issued or until termination.

The channel addressing capability is based on a master-slave relationship with the host being the master. The host can either use the current channel address or choose a new current channel address. In either case, the host sends data intended for the current forward channel address.

If the peripheral has requested attention and wishes to return data on the current channel address (current means the same as the last host-selected channel address), then the peripheral may send the data intended for the current channel address without first selecting a reverse channel address.

However, if the peripheral has requested attention and wishes to return data on an address different from the current channel address, then the peripheral shall first select the appropriate reverse channel address before sending the data. If the reverse channel address has been changed and the peripheral wishes to return data intended for the current channel address (the last host-selected address), then the peripheral shall first select the current channel address prior to sending data.

A channel address command issued from the peripheral only affects the peripheral-to-host communication channel. (The address used for host-to-peripheral communication remains unchanged.) That address remains in effect indefinitely for peripheral-to-host communications until another channel address command is issued from either the host or the peripheral or until termination.

6.10 EPP Mode addressing

EPP Mode incorporates a signaling technique that is microprocessor bus oriented. In the case of the typical bus system that has a time-multiplexed address/data bus, all cycles are initiated by the host processor. First, an address is generated on the bus and is latched by external circuitry when the processor generates an address strobe. A separate strobe is generated to perform the actual data transfer. Cycles are terminated when the addressed device signals “ready” back to the processor.

EPP Mode provides an interface that is functionally equivalent to such a system. All cycles are initiated by the host with a means of generating address and data strobes. Also analogous to a microprocessor bus, cycles are terminated and cycles can be extended by the device.

6.11 Device identification

Prior to the first peripheral-to-host transfer, the host does not know the type of device to which it is attached, or how to communicate with it. The device identification option allows the host to request ID information from the peripheral using one of the IEEE 1284 reverse data transfer modes (Nibble, Byte, or ECP). The peripheral identifies itself by sending a sequence of bytes to the host indicating its device type, device family, and language capabilities. Refer to 7.6 for details of the Device ID response format.

Nibble Mode: All IEEE 1284-compliant devices are required to support Nibble Mode, and all IEEE 1284-compliant devices that support Device ID are required to support Device ID in Nibble Mode. This ensures that host devices will be able to read the Device ID of the peripheral, if present, without extended IEEE 1284 mode negotiation.

ECP Mode: Channel addressing is not used during Device ID. RLE data compression may be used during Device ID, if requested by the host Extensibility Request Value and acknowledged by the peripheral Extensibility Flag. Forward channel data is not sent during Device ID mode. To transfer data, the host shall first terminate from the Device ID mode and renegotiate with the Device ID Request bit de-asserted in the Extensibility Request Value.

7. Interface operation

The interface is always initialized to the Compatibility Mode, a conventional, unidirectional host-to-peripheral interface. From the Compatibility Mode the host may

a) Transmit data to the peripheral using the Compatibility Mode, or
b) Direct the interface into another mode.

Refer to Annex A for signal timing definitions and to Annex B for signal event definitions.

A peripheral-to-host data transfer is initiated by the host negotiating with the peripheral for a mutually supported communication mode. At the discretion of the host, the interface can be returned to the Compatibility Mode at any time. Phase transition diagrams showing the interface as it moves between phases are shown in Figure 6, Figure 12, and Figure 16 for the Nibble and Byte, ECP, and EPP Modes, respectively.

The remainder of this clause describes the negotiation, currently defined peripheral-to-host data transfer protocols, and the return to Compatibility Mode procedures.

7.1 Power-On

Hosts and peripherals indicate their readiness to communicate by asserting Host Logic High and Peripheral Logic High respectively. All hosts with IEEE 1284-C connectors shall provide Host Logic High. All peripherals with IEEE 1284-C connectors shall provide Peripheral Logic High. The interface becomes active 500 ms after both Host Logic High and Peripheral Logic High exceed 3.0 V, at which point all control, data, and status signals are required to be valid (see 8.3.5). Since devices with IEEE 1284-A or IEEE 1284-B connectors may or may not support Host Logic High or Peripheral Logic High, there is no reliable way to initiate a transfer. If the other device is not present when a transfer begins, a timeout may occur or the interface will hang.

7.2 Initialization

Hosts may reinitialize the interface at any time by asserting nInit low in conjunction with nSelectIn low. The interface returns to the Compatibility Mode idle phase after initialization. Resetting the interface may also reset the peripheral.

7.3 Compatibility Mode

The Compatibility Mode follows pre-existing industry conventions with eight parallel bits, (Data 1–Data 8) qualified by a low-going pulse on HostClk(nStrobe). The peripheral responds by setting PtrBusy(Busy) high to hold off the next byte until the current one has been removed from the input latch. When ready for the next byte, the peripheral pulses PtrClk(nAck) low, then sets PtrBusy(Busy) low. Timing requirements on peripheral Busy ensure that compliant peripherals will interoperate with existing hosts that monitor Busy but not nAck.

Compatibility Mode timing values are specified in Figure 2 and Table 3.

After receiving Busy low from the peripheral, hosts shall not set nStrobe low until T_{ready}. The peripheral can reactivate Busy asynchronously due to some condition other than the reception of a data byte. Due to this race condition, the host may at that instant be asserting nStrobe low to send the next data byte. The peripheral shall accept at least one byte from the host after entering a Busy state in this situation.

Hosts shall provide T_{setup} from D1...D8 and nAutoFd stable to the falling edge of nStrobe.

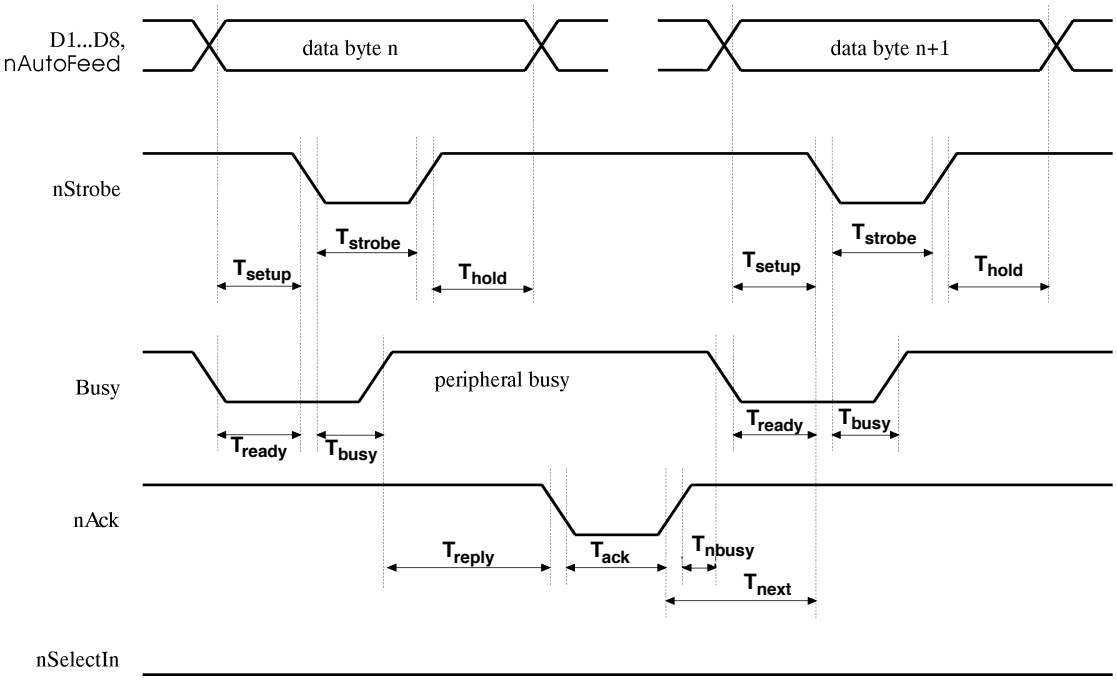


Figure 2—Compatibility Mode handshake

Hosts shall pulse nStrobe low for T_{strobe}.

Hosts shall provide T_{hold} from the rising edge of nStrobe to any change in D1...D8 or nAutoFd.

Compliant peripherals shall set Busy high before setting either nFault low, PError high, or Select low. Busy has to be held high until all these signals return to their normal state (nFault high, PError low, and Select high).

Normally, every byte sent from the host to the peripheral is acknowledged by the peripheral pulsing nAck low while the Busy line is held high. This may not be the case when a host negotiation causes the interface to change from Compatibility Mode to another IEEE 1284 interface mode after the host asserts nStrobe low but before the peripheral pulses nAck. In this event, when the interface is returned to Compatibility Mode the host shall rely on the Busy signal to determine when the peripheral is ready for the next byte.

7.4 Negotiation

The IEEE 1284-compliant host negotiates with the peripheral to determine whether or not the peripheral is IEEE 1284 compliant. (Annex B defines the signal transition events noted in the following subclauses.) The host verifies that the device is IEEE 1284 compliant when, at event 2 (see Figure 3), the peripheral, as a result of the host stimulus (event 1), sets its status lines to include PError and nFault both at logic high levels. This will not happen on a non-IEEE 1284 compliant peripheral. If the peripheral does not indicate that it is an IEEE 1284-compliant device (event 2 never happens), the host should remove the stimulus (event 1) and conclude that the device is not IEEE 1284 compliant.

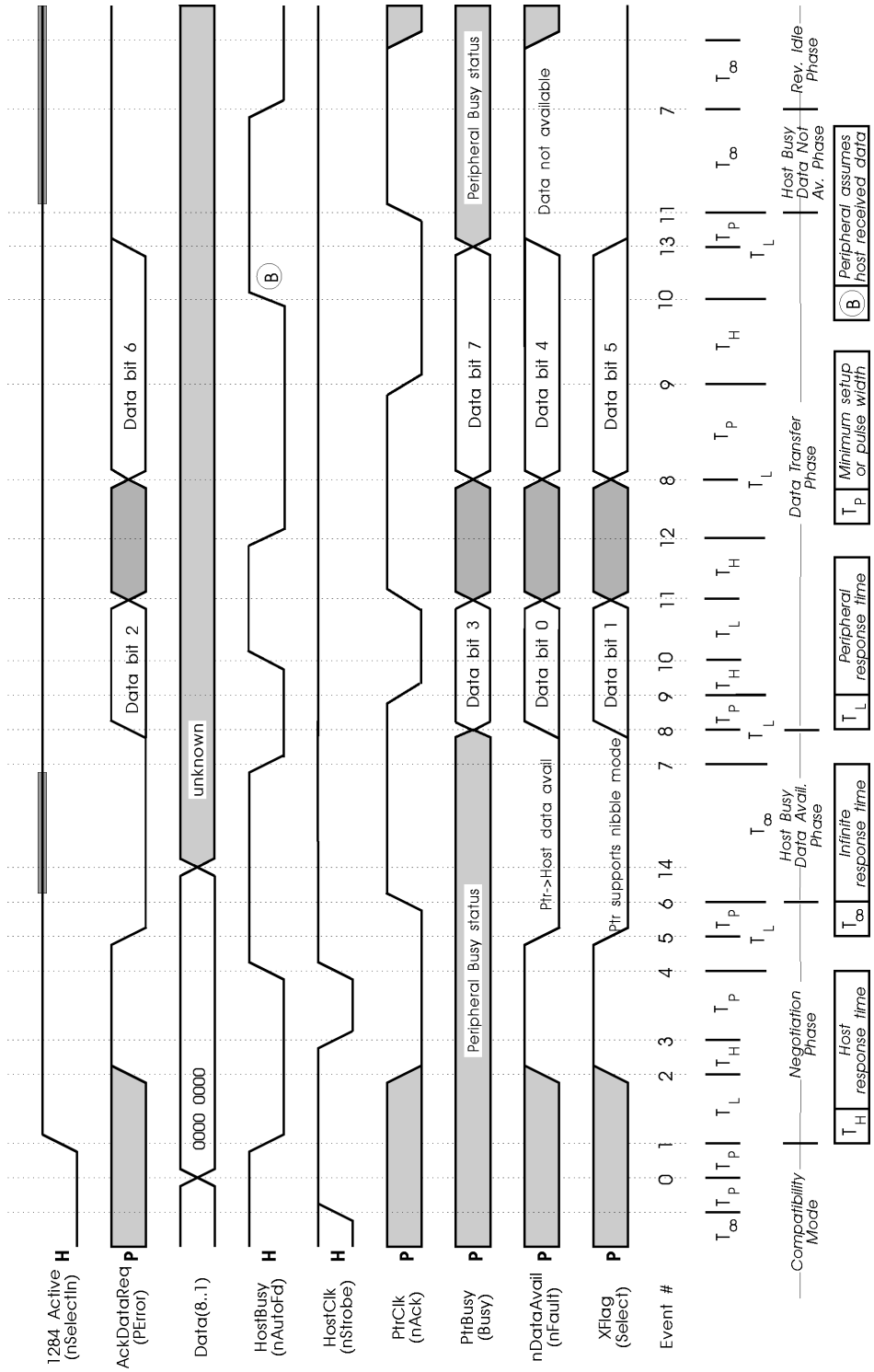


Figure 3—Nibble Mode negotiation and transfer

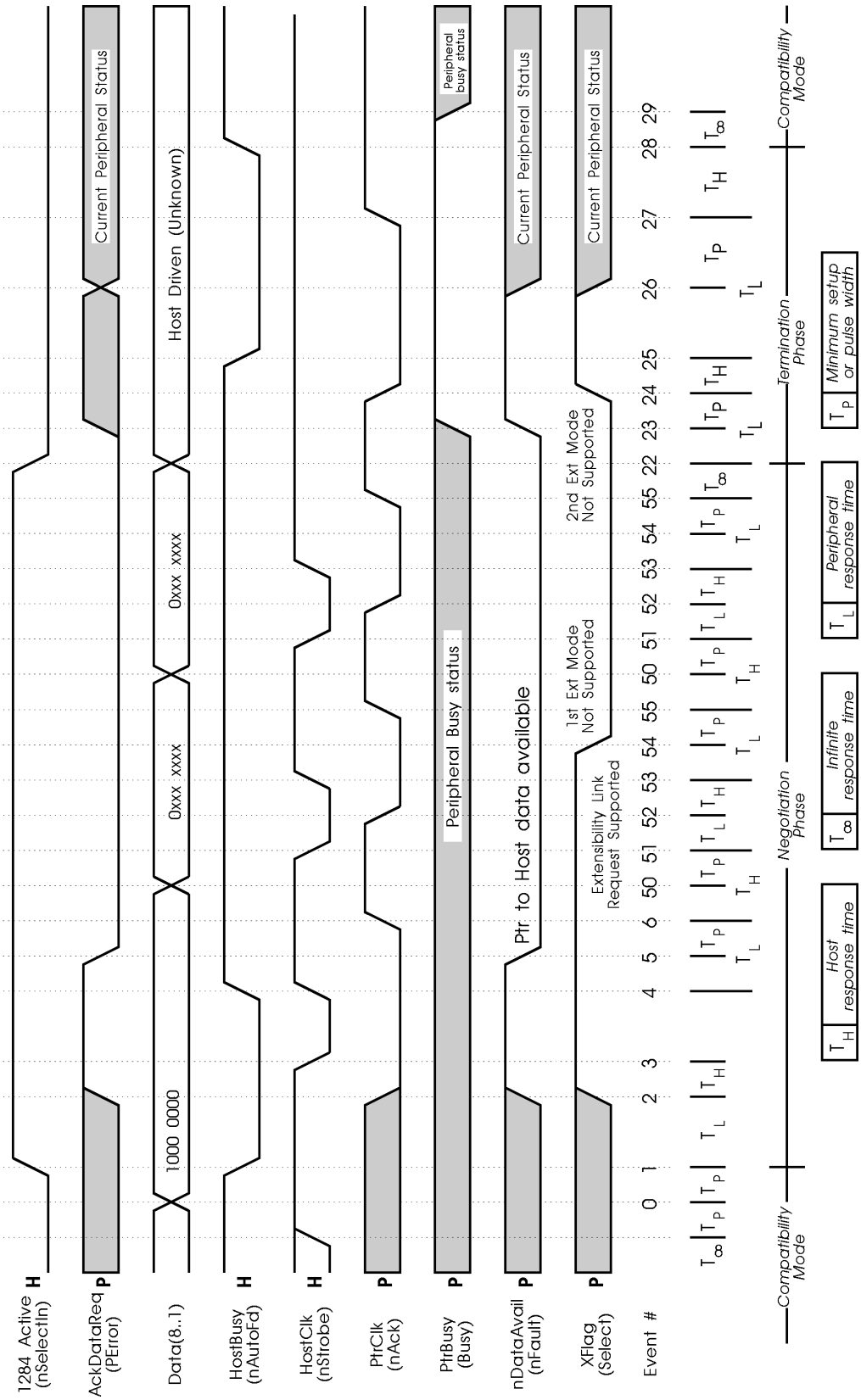


Figure 4—Negotiation with extensibility link

Table 3—Compatibility Mode handshake timing values

Parameter	Measured at	Measured from	Measured to	Value (minimum/ maximum)	Compliance
T_{ready}	Host output	Busy ≤ V_{IL}	nStrobe ≤ V_{OH}	0 min.	Compatible hosts
T_{setup} (host)	Host output	Data stable	nStrobe ≤ V_{OH}	750 ns min.	Compatible hosts
T_{setup} (peripheral)	Peripheral input	Data stable	nStrobe ≤ V_{IH}	500 ns max.[a]	Compatible peripherals
T_{strobe} (host)	Host output	nStrobe ≤ V_{OL}	nStrobe ≥ V_{OL}	750 ns min., 500 s max.	Compatible hosts
T_{strobe} (peripheral)	Peripheral input	nStrobe ≤ V_{IL}	nStrobe ≥ V_{IL}	500 ns max.	Compatible peripherals
T_{hold} (host)	Host output	nStrobe ≥ V_{OH}	Data or nAutoFd change	750 ns min.	Compatible hosts
T_{hold} (peripheral)	Peripheral input	nStrobe ≥ V_{IL}	Data or nAutoFd change	500 ns max.	Compatible peripherals
T_{busy}	Peripheral output	nStrobe ≤ V_{IL}	Busy ≥ V_{OH}	500 ns max.	Compliant peripherals
T_{reply}	Peripheral output	Busy ≥ V_{IH}	nAck ≤ V_{OH}	0 min.	Compatible peripherals
T_{ack}	Peripheral output	nAck ≤ V_{OL}	nAck ≥ V_{OL}	500 ns min., 10 μs max.	Compatible peripherals
T_{nbusy}	Peripheral output	nAck ≥ V_{OH}	Busy ≤ V_{OH}	0 min.[b]	Compliant peripherals
T_{next}	Host output	nAck ≥ V_{IL}	nStrobe ≤ V_{OH}	0 min.	Compliant hosts

NOTES

1—For more information on the history of Centronics Standard Parallel and PC-Compatible Parallel Interfaces, see Annex C and, in particular, C.6.2 for Busy-to-Ack timing variations.

2—V_{IL} is the low-level voltage input

V_{OL} is the low-level voltage output

V_{OH} is the high-level voltage output

V_{IH} is the high-level voltage input

[a]The maximum value stated for peripherals in this table are referenced to the peripheral. For example, the peripheral cannot require more than 500 ns data setup time.

[b]Recognize that complimentary signal changes may have overlapping signal transitions. The zero minimum value cannot be guaranteed.

Upon verification that the peripheral is IEEE 1284 compliant, the host will request a communication mode for the peripheral. The IEEE 1284-compliant device will acknowledge the communication mode request based on its capabilities and execute or reject the request as appropriate.

The IEEE 1284 interface protocol has been designed to allow the host to request that the peripheral use various communication modes. This is accomplished by the host placing an extensibility request value on the data bus during the negotiation phase (event 0) (see Figure 3). The value 00h requests peripheral-to-host data

exchange using Nibble Mode reverse channel transfers. The Extensibility Request Value is used to request a communication mode. Table 4 shows the current assignments of these values.

Table 4—IEEE 1284 Extensibility Request Values—Bit assignments

Definition	Extensibility values	Xflag affirmative response
Request Extensibility Link	1000 0000	High
Request EPP Mode	0100 0000	High
Request ECP Mode with RLE	0011 0000	High
Request ECP Mode	0001 0000	High
Reserved	0000 1000	High
Request Device ID; return data using Nibble Mode Reverse Channel Transfer	0000 0100	High
Byte Mode Reverse Channel Transfer	0000 0101	High
ECP Mode Transfer without RLE	0001 0100	High
ECP Mode Transfer with RLE	0011 0100	High
Reserved	0000 0010	High
Byte Mode Reverse Channel Transfer	0000 0001	High
Nibble Mode Reverse Channel Transfer	0000 0000	Low

NOTE—IEEE Std 1284.3-2000 [B8] defines other extensibility values. Other future specifications may also define new values.

Bit seven, the “extensibility link request” bit, is a special bit. It is used to add a second extensibility request byte during the negotiation phase. This allows more extensions to be added to the interface in the future. A multiple byte negotiation is shown in Figure 4.

In Table 4, the column labeled “XFlag affirmative response” denotes the level of the XFlag signal that the host expects to see, as an indication of a positive response, when it provides the specified Extensibility Request byte to the peripheral during the negotiation phase. All requests are honored by a positive response with the exception of the Nibble Mode reverse channel transfer request. This allows a “dumb peripheral” to support this specification using only the lowest peripheral-to-host transfer protocol and without having to examine the data signals on every host negotiation. For example, if the “dumb peripheral” always provides a low-level response to every host negotiation request, then when the host requests to use a Nibble Mode protocol, it will receive a positive response from the peripheral. The peripheral has already acknowledged that it is IEEE 1284 compliant by its responses to the beginning sequence of the negotiation phase (event 2).

The default IEEE 1284 communication mode for all hosts and peripherals is Compatibility Mode. This mode is maintained until the host has successfully verified that it is connected to an IEEE 1284-compliant device. To begin the negotiation phase, the host places the Extensibility Request Value on the data bus (event 0), then sets IEEE 1284 Active(nSelectIn) high and HostBusy(nAutoFd) low (event 1). (Figure 3 shows the

negotiation events along with a reverse channel data transfer for the Nibble Mode. These negotiation events through event 4 are the same regardless of the communication mode being requested.) The peripheral responds by setting PtrClk(nAck) low, nDataAvail(nFault) high, Xflag(Select) high, and AckDataReq(PError) high (event 2). The host then sets HostClk(nStrobe) low (event 3), strobing the Extensibility Value into the input data latch of the peripheral. The host then sets HostClk(nStrobe) and HostBusy(nAutoFd) high (event 4), acknowledging that it has recognized an IEEE 1284-compatible peripheral. The peripheral then sets Xflag(Select) to its appropriate value corresponding to the extensibility feature requested (event 5). It also setsAckDa-taReq(PError) and nDataAvail(nFault) low if peripheral-to-host data is available. Otherwise they are set high. The peripheral then sets Ptr-Clk(nAck) high (event 6), indicating that the other status lines may be read. If peripheral-to-host data is available, the interface enters the host busy, data available phase. If not, the interface enters the host busy, data not available phase. At event 5 and beyond, the state of the interface is based on the mode selected.

Table 5 defines the timing values at the driver (the receiver has to make allowances for any cable effects) associated with Figure 3 through Figure 23.

Table 5—IEEE 1284 mode handshake timing values

Time	Minimum	Maximum	Description
T_H	0	1.0 s	Host response time
T_∞	0	Infinite	Infinite response time
T_L	0	35 ms	Peripheral response time
T_R	0	N/A	Peripheral response time (ECP Mode only)
T_S	35 ms	N/A	Host recovery time (ECP Mode only)
T_P	0.5 μs	N/A	Minimum setup or pulse width
T_D	0		Minimum data setup time (ECP/EPP Modes only)
T_{ES}	0	125 ns	Short response time (EPP Mode only)
T_{EL}	0	10 μs	Long response time (EPP Mode only)
T_{ER}	50 μs	Infinite	Termination/nInit pulse width
T_n	100 ns	N/A	Peripheral to host data setup time
NOTE—T_R state can be ended by the host when its T_S has expired and a recovery process is started. T_S is a minimum of 35 ms and a maximum of infinity.			

These subclauses are accompanied by full-page diagrams showing the handshake sequences. Along the bottom of each diagram are numbers corresponding to signal transition events. Each of these events is listed in Annex A for reference. The event numbers are also shown in parentheses in the following subclauses to improve readability of the diagrams.

Note that these diagrams are examples of correct operation (“typical” or “nominal”) and are not necessarily requirements.

Valid termination states (see 7.7) are indicated in the diagrams by IEEE 1284 Active(nSelectIn) shown as a heavy line.

7.4.1 Successful negotiation

There are several possible outcomes of the negotiation sequence, depending on the mode requested, the response of the peripheral, and subsequent host action.

If Nibble or Byte Mode was requested and

a) The peripheral responds correctly and peripheral-to-host data is available, the host may
   1) Proceed with a reverse channel data transfer using the mode requested by the host,
   2) Remain in the host busy, data available phase, or
   3) Terminate and return to Compatibility Mode.
b) The peripheral responds correctly, but no peripheral-to-host data is available, the host may
   1) Set HostBusy(nAutoFd) low, which puts the interface into the idle phase, or
   2) Terminate and return to Compatibility Mode.

If ECP Mode was requested and the peripheral responds correctly, the link, following a setup phase, will enter the ECP forward idle phase. From this state

— The host can request to send data to the peripheral,
— The peripheral can request to send data to the host, or
— The host can terminate the ECP Mode and return to Compatibility Mode.

If EPP Mode was requested and the peripheral responds correctly, the link will enter the EPP Mode idle phase. From this phase

— The host can write data to the peripheral.
— The host can read data from the peripheral.
— The host can write an address value to the peripheral.
— The host can read an address value from the peripheral.
— The host can terminate EPP Mode and return to Compatibility Mode.

NOTE—After successfully negotiating to the EPP Mode, low-to-high transitions of Intr(nAck) shall not be treated by the host interface as an EPP interrupt request until after the first EPP cycle. In other words, Intr(nAck) is not recognized as an EPP interrupt until after the first EPP cycle is complete.

When requesting the peripheral to provide Device ID information, the extensibility byte includes a mode to be used for the peripheral-to-host transfer. Assuming that the peripheral responds correctly, the peripheral will execute the request and transfer the data. After the host has received the data for the Device ID (the number of bytes is defined by the first two bytes received from the peripheral), the host is required to return the link to the Compatibility Mode.

7.4.2 Unsuccessful negotiation

For a transfer mode request, if the peripheral does not support the requested mode it will set Xflag(Select) low if Byte, ECP, or EPP Mode was requested (event 5). [Responding with Xflag(Select) high for a request for Nibble Mode is not allowed since all IEEE 1284-compliant devices shall support Nibble Mode.] When this occurs, the host shall return the link to the Compatibility Mode and renegotiate for another transfer mode. This sequence is shown in Figure 5.

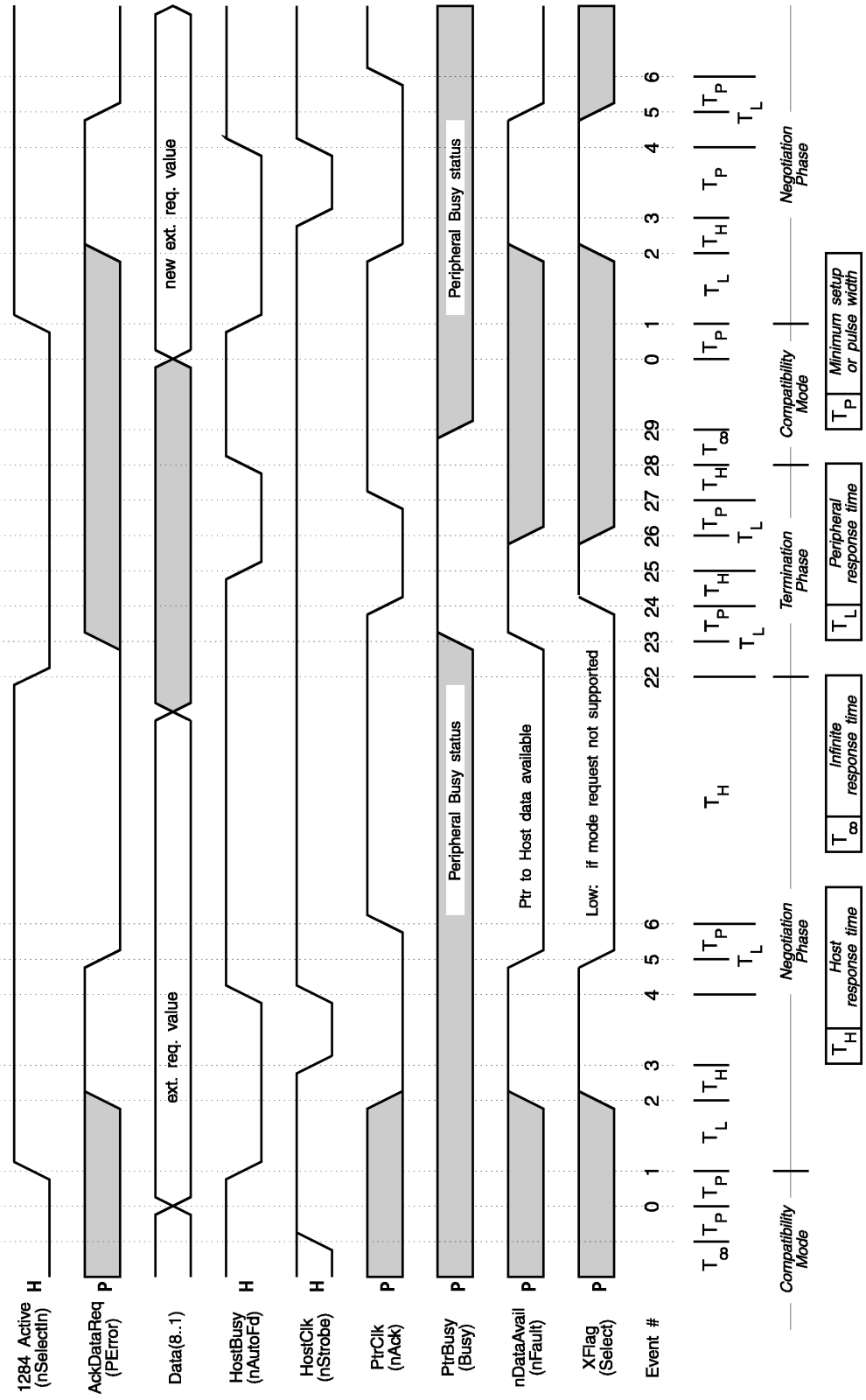


Figure 5—Failed negotiation and renegotiation

If the peripheral rejects the second byte of an extensibility link request, the host may attempt further extensibility bytes. If all attempts fail, the host will remain in Compatibility Mode.

If illegal or contradictory transfer mode requests are made (such as both ECP and Byte Mode bits are set), the peripheral shall reject the invalid combination by setting XFlag low.

7.5 Peripheral-to-host transfer modes

There are four modes defined as capable of providing peripheral-to-host data transfers. They are the Nibble Mode, Byte Mode, ECP Mode, and EPP Mode. These modes are described and illustrated in the following subclauses.

7.5.1 Nibble Mode

The signals used for the Nibble Mode transfer are described in detail in Clause 5. For convenience, they are summarized in the following table.

Nibble Mode signal name	Compatibility Mode signal name	Nibble Mode signal description
PtrClk	nAck	Reverse data transfer phase: Used to qualify data being sent to the host. Reverse idle phase: Set low then high by the peripheral to cause an interrupt indicating to the host that data is available.
PtrBusy	Busy	Reverse data transfer phase: Data bit 3 then 7, then forward channel busy status.
AckDataReq	PError	Reverse data transfer phase: Data bit 2 then 6. Reverse idle phase: Set high until the host requests a data transfer, then follows nDataAvail(nFault).
Xflag	Select	Reverse data transfer phase: Data bit 1 then 5.
HostBusy	nAutoFd	Reverse data transfer phase: Set low to indicate that host can receive peripheral-to-host data, then set high to acknowledge receipt of that nibble. Following a reverse channel transfer, the interface transitions to idle phase when HostBusy(nAutoFd) is set low and the peripheral has no data available. Reverse idle phase: Set high in response to PtrClk(nAck) low pulse to re-enter reverse data transfer phase. If set high with IEEE 1284 Active(nSelectIn) set low, the IEEE 1284 idle phase is aborted, and the interface returns to Compatibility Mode.
nDataAvail	nFault	Reverse data transfer phase: Set low to indicate that the peripheral has the data ready to send to the host. Then used to send data bit 0 (LSB) then 4. Reverse idle phase: Used to indicate that data is available.

After negotiating to the Nibble Mode, the host will set HostBusy(nAutoFd) low (event 7) to indicate that it can accept data from the peripheral (see Figure 3). The peripheral responds by placing the low-order nibble of the byte on the four reverse channel data lines (event 8), then setting PtrClk(nAck) low (event 9). The host

then latches the nibble and sets HostBusy(nAutoFd) high (event 10), signaling to the peripheral that it has received the nibble. The peripheral then sets PtrClk(nAck) high (event 11), completing the first nibble handshake. This sequence is repeated for the high-order nibble, with one exception. After the host has acknowl-edged the second nibble by setting HostBusy(nAutoFd) high (event 10), the peripheral sets the four status lines as follows (event 13): PtrBusy(Busy) to its current forward channel value; nDataAvail(nFault) low if another byte is ready to be sent; AckDataReq(PError) to the same value as nDataAvail(nFault); and Xflag(Select) to its value during the last negotiation. The peripheral then sets PtrClk(nAck) high (event 11). The host examines the status lines to determine if another byte is available and if the forward channel can receive data. Figure 3 shows negotiation and subsequent transfer of data that proceeds as follows.

At the end of a byte transfer (two nibbles) the peripheral reports whether it has more data for the host. If there is no more data, the host has three options

a) The host may terminate and return to Compatibility Mode,
b) The host may stay in the host busy, data not available phase, or
c) The host may set HostBusy(nAutoFd) low (event 7), putting the interface into the idle phase (this transition is shown in Figure 3 for Nibble Mode).

If there is additional data, the host has three options

a) The host may set HostBusy(nAutoFd) low, indicating that the host can accept additional data,
b) The host may stay in the host busy, data available phase, or
c) The host may terminate and return to Compatibility Mode.

Whenever the peripheral generates data for the host, it may request a reverse channel data transfer if the interface is in the reverse idle phase. Peripherals shall not indicate a new request for a reverse channel transfer unless it is in reverse idle or during negotiation. This is accomplished by setting nDataAvail(nFault) and PtrClk(nAck) low (event 18), then setting PtrClk(nAck) high (event 19). This generates an interrupt inside the host, assuming the parallel port interrupt is enabled. The host responds to the interrupt by setting HostBusy(nAutoFd) high (event 20). The peripheral then sets AckDataReq(PError) low (event 21) to acknowledge the response of the host. At this point the interface is in the host busy, data available phase and may be followed by a reverse channel transfer. This sequence is shown in Figure 8 for Nibble Mode operation.

7.5.2 Byte mode

The signals used for the Byte Mode transfer are described in detail in Clause 5. For convenience, they are summarized in the table that follows.

After negotiating to the Byte Mode, the host will place the data bus in a high impedance state (event 14), then set HostBusy(nAutoFd) low (event 7) to indicate that it can accept data from the peripheral. The peripheral responds by placing the reverse channel byte on the data bus (event 15) as shown in Figure 11. The peripheral then sets PtrClk(nAck) low (event 9). The host then sets HostBusy(nAutoFd) high (event 10), acknowledging that it has seen PtrClk(nAck) and indicating to the peripheral that it is processing the byte. The peripheral then updates the status lines (event 13) as follows: PtrBusy(Busy) to its current forward channel value; nDataAvail(nFault) low if another byte is ready to be sent; AckDataReq(PError) to the same value as nDataAvail(nFault); and Xflag(Select) to its value during the last negotiation. The peripheral then sets PtrClk(nAck) high (event 11), completing the byte handshake. At this point, the host will pulse HostClk(nStrobe) low (event 16) then high (event 17), signaling that it has received the byte. The peripheral shall not interpret this pulse as a latch signal for forward channel data. Note that events 10 and 16 may occur simultaneously, and events 7 and 17 may occur simultaneously.

Byte Mode signal name	Compatibility Mode signal name	Byte Mode signal description
HostClk	nStrobe	Will be pulsed low during Byte Mode transfers. The peripheral shall ensure that HostClk(nStrobe) does not cause data to be latched when this happens.
Data1...Data8	Data1...Data8	Reverse channel data.
PtrClk	nAck	Reverse data transfer phase: Used to qualify data being sent to the host. Reverse idle phase: Set low then high by the peripheral to cause an interrupt indicating to the host that data is available.
PtrBusy	Busy	Forward channel busy status.
AckDataReq	PError	Reverse data transfer phase: Follows nDataAvail(nFault). Reverse idle phase: Set high until the host requests a data transfer, then follows nDataAvail(nFault).
HostBusy	nAutoFd	Reverse data transfer phase: Set low to indicate that the host can receive peripheral-to-host data, then set high to acknowledge receipt of that byte. Following a reverse channel transfer, the interface transitions to idle phase when HostBusy(nAutoFd) is set low and the peripheral has no data available. Reverse idle phase: Set high in response to PtrClk(nAck) low pulse to reenter reverse data transfer phase. If set high with IEEE 1284 Active(nSelectIn) set low, the IEEE 1284 idle phase is aborted, and the interface returns to Compatibility Mode.
nDataAvail	nFault	Set low to indicate that the peripheral has data to send to the host.

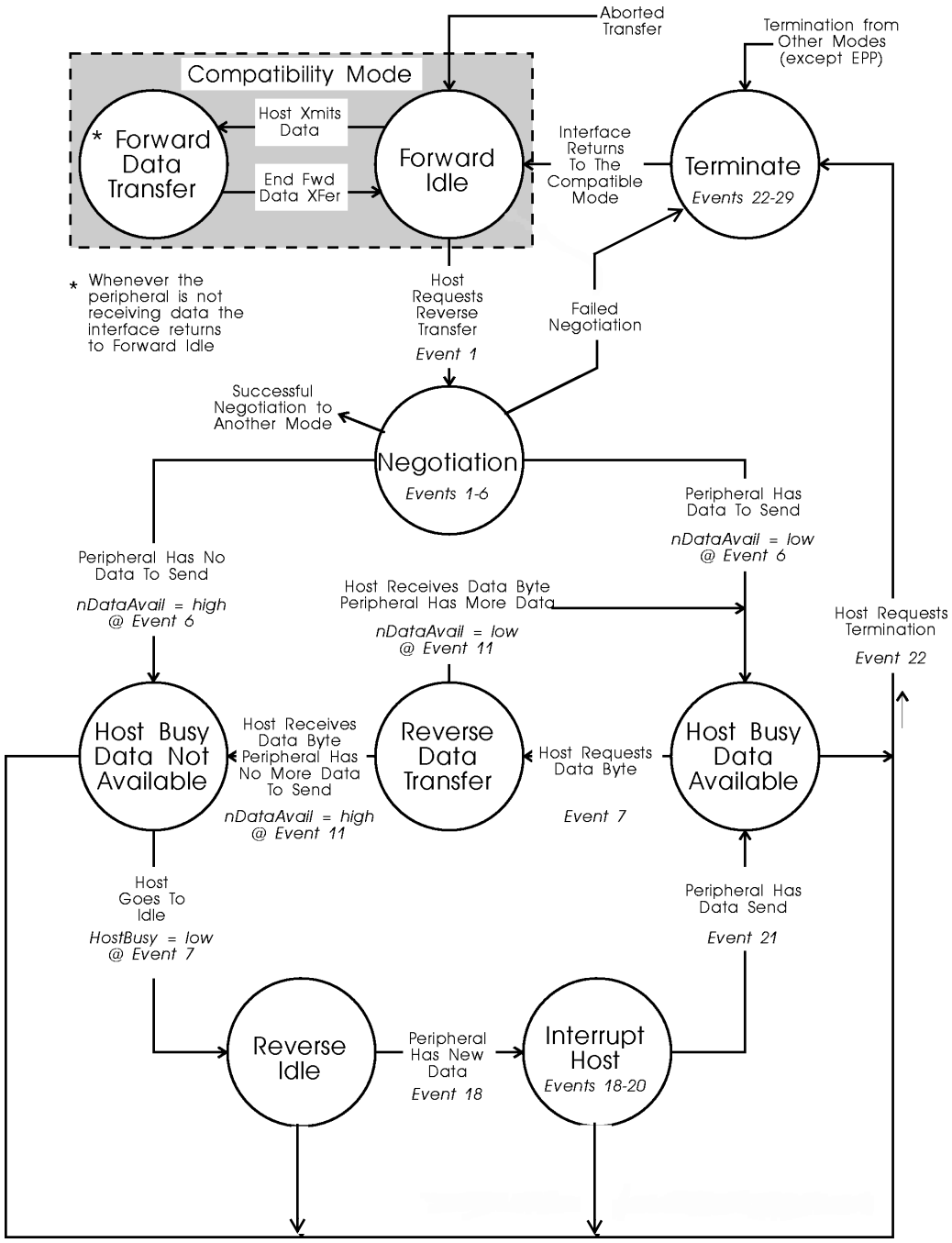
One objective in designing the Byte Mode transfer was to support the PS/2 DMA mode and still maintain efficient operation and similarity to Nibble Mode. The PS/2 generates a unique event 16 (i.e., it is not coincident with event 10), followed by event 17. A transfer is considered successful at event 17. The Byte Mode supports this as well as a more efficient style of handshaking that allows HostClk(nStrobe) to fall (event 16) at or after HostBusy(nAutoFd) is raised (event 10), and to rise again (event 17) at or before HostBusy(nAutoFd) is lowered (event 7). Software-driven hosts can thereby reduce the number of port reads and writes that shall be done during a transfer. The only restriction on events 16 and 17 is that they occur far enough apart for the peripheral to detect a low value on HostClk(nStrobe) after it raises PtrClk(nAck) (event 11). This can be guaranteed by observing a minimum pulse width or by setting HostClk(nStrobe) low while raising HostBusy(nAutoFd) (events 10 and 16, simultaneously).

At the end of a byte transfer, the peripheral reports whether it has more data for the host. If there is no more data, the host has three options

a) The host may terminate and return to Compatibility Mode.
b) The host may stay in the host busy, data not available phase.
c) The host may set HostBusy(nAutoFd) low (event 7), putting the interface into the idle phase. This transition is shown in Figure 11 for Byte Mode.

If there is additional data, the host has three options

a) The host may set HostBusy(nAutoFd) low, indicating that the host can accept additional data.
b) The host may stay in the host busy, data available phase.
c) The host may terminate and return to Compatibility Mode.



NOTES:

1—Circles represent IEEE 1284 phases.
2—This should not be considered a true state diagram.

Figure 6—Nibble and Byte Mode phase transitions

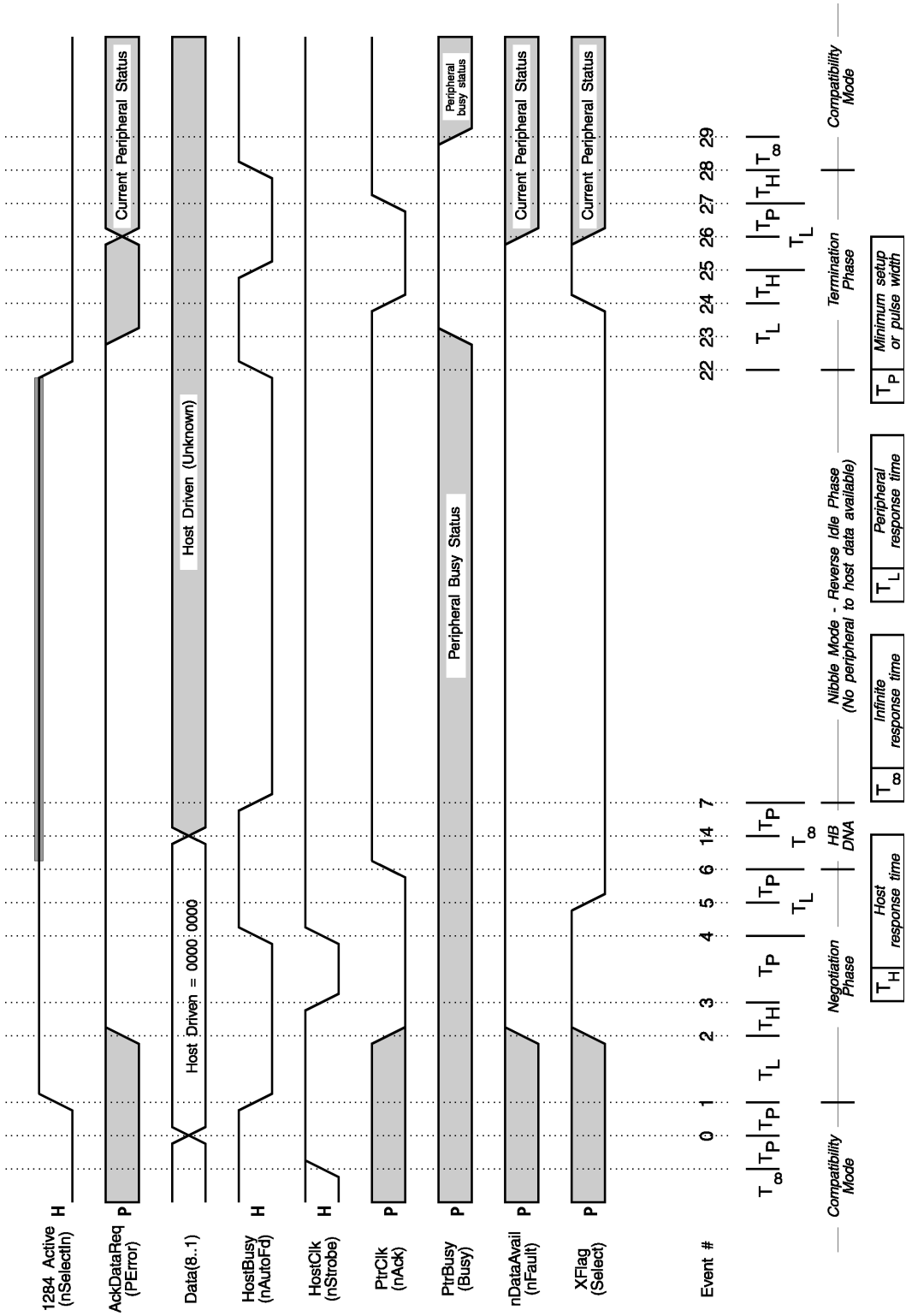


Figure 7—Nibble Mode negotiation and termination

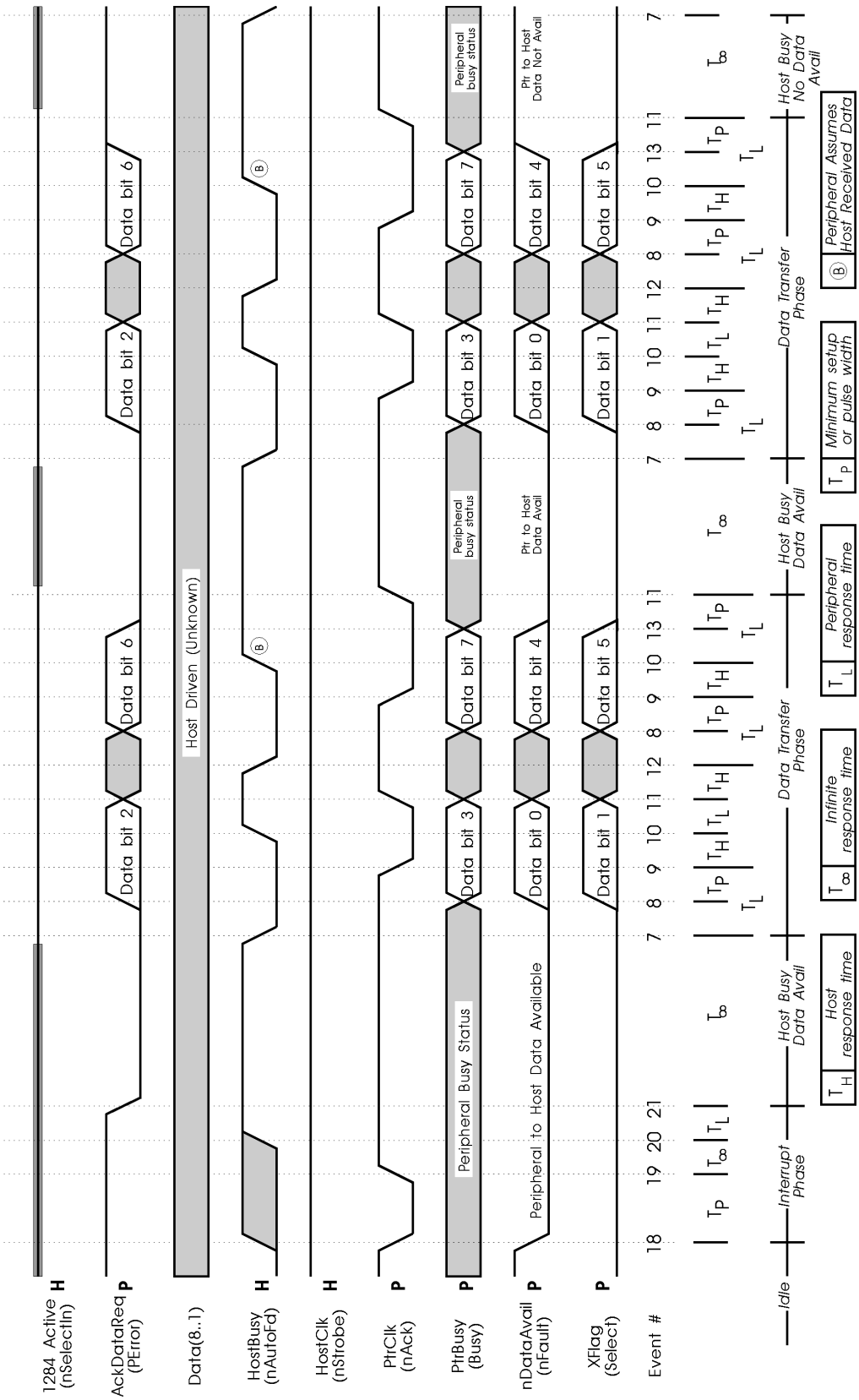


Figure 8—Nibble Mode idle phase to data transfer

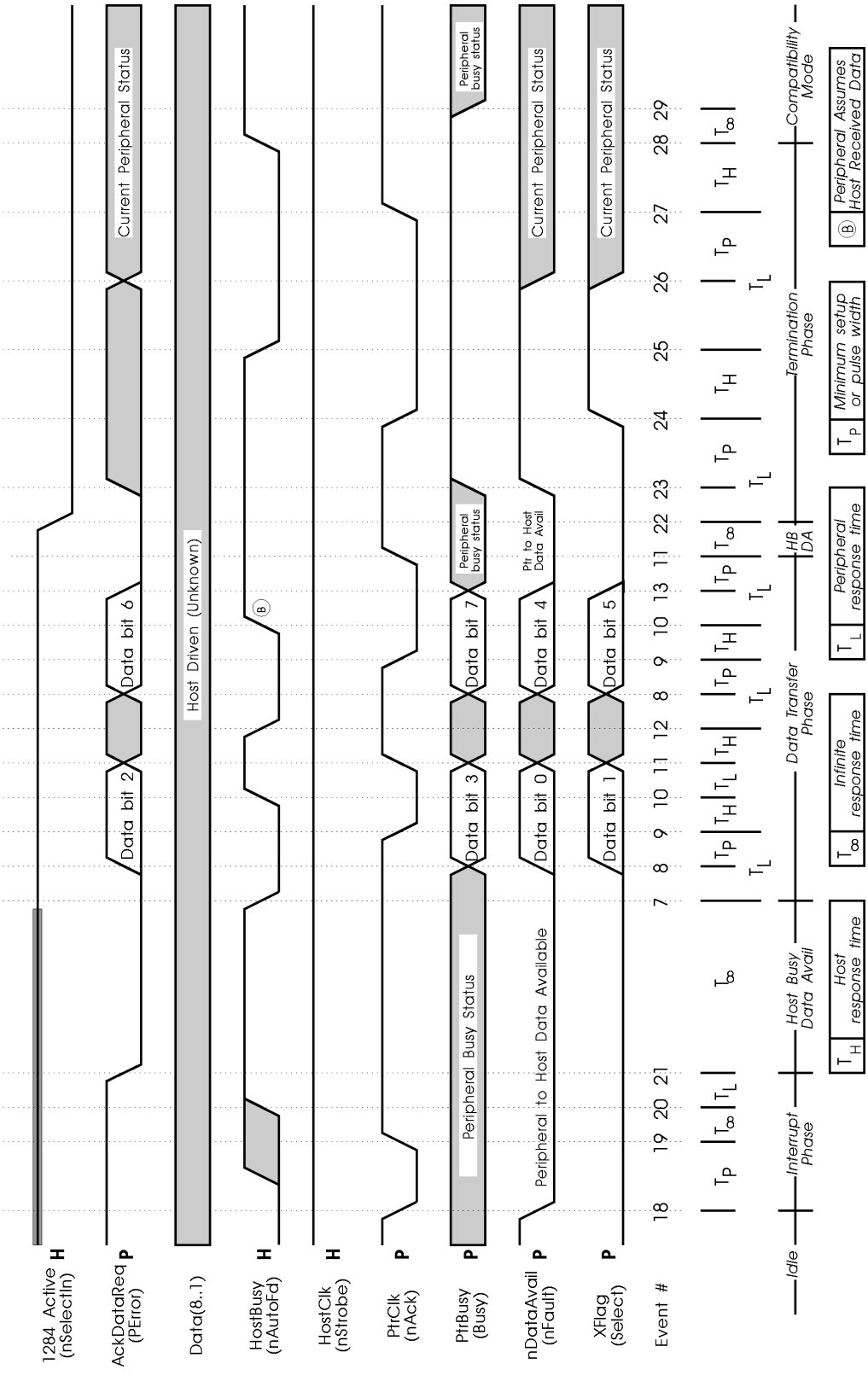


Figure 9—Nibble Mode Idle phase to data transfer to termination

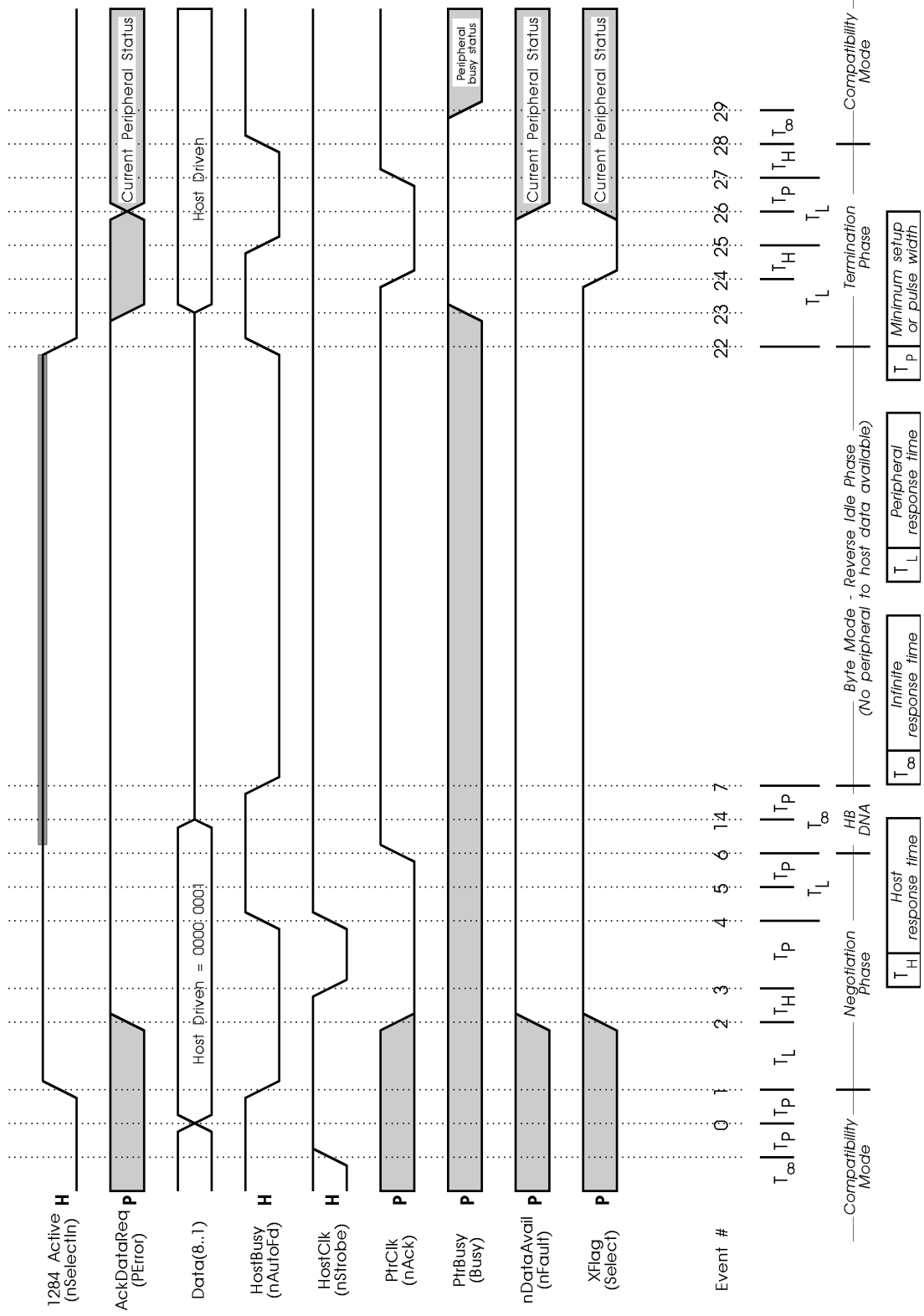


Figure 10—Byte Mode negotiation and termination

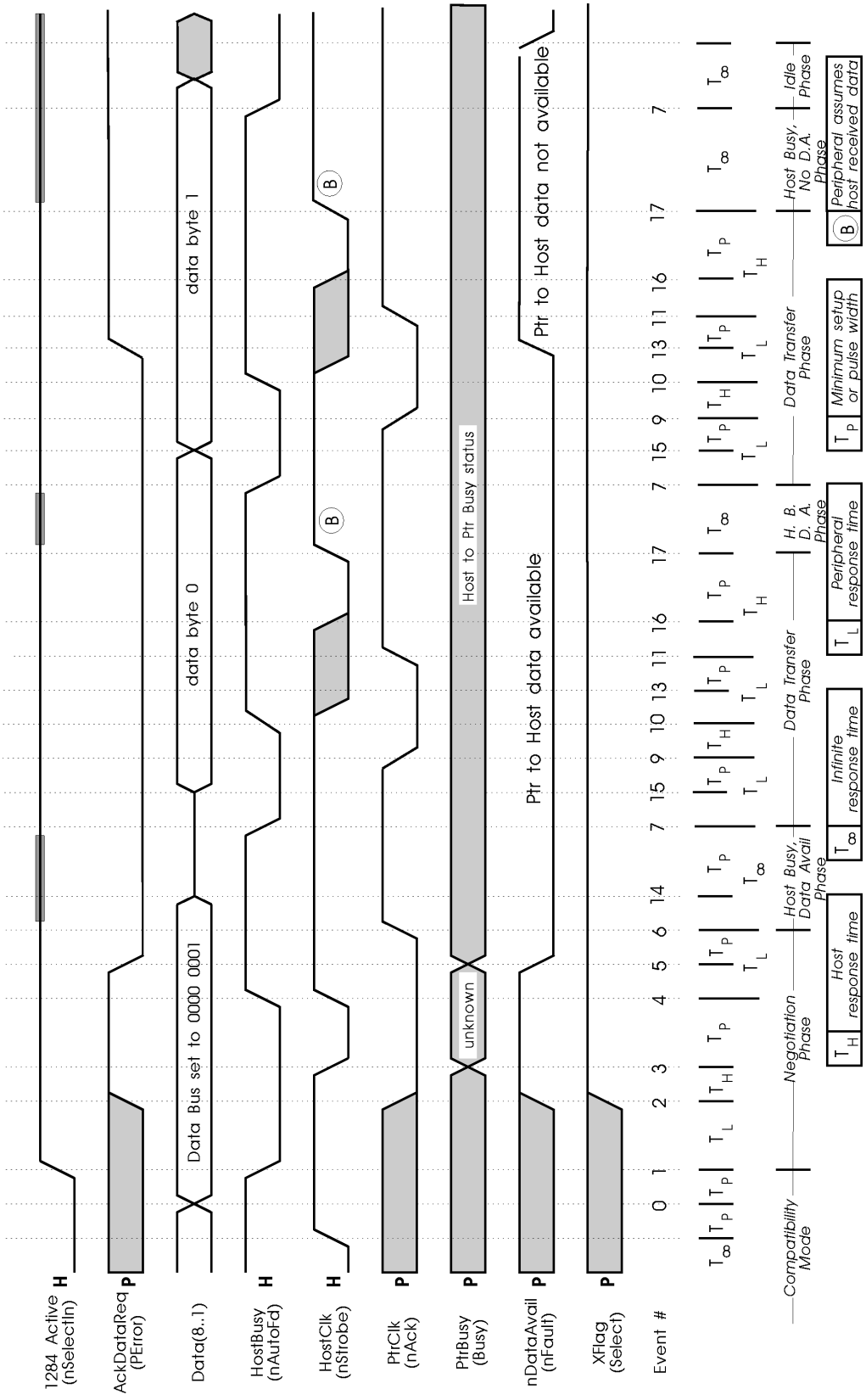
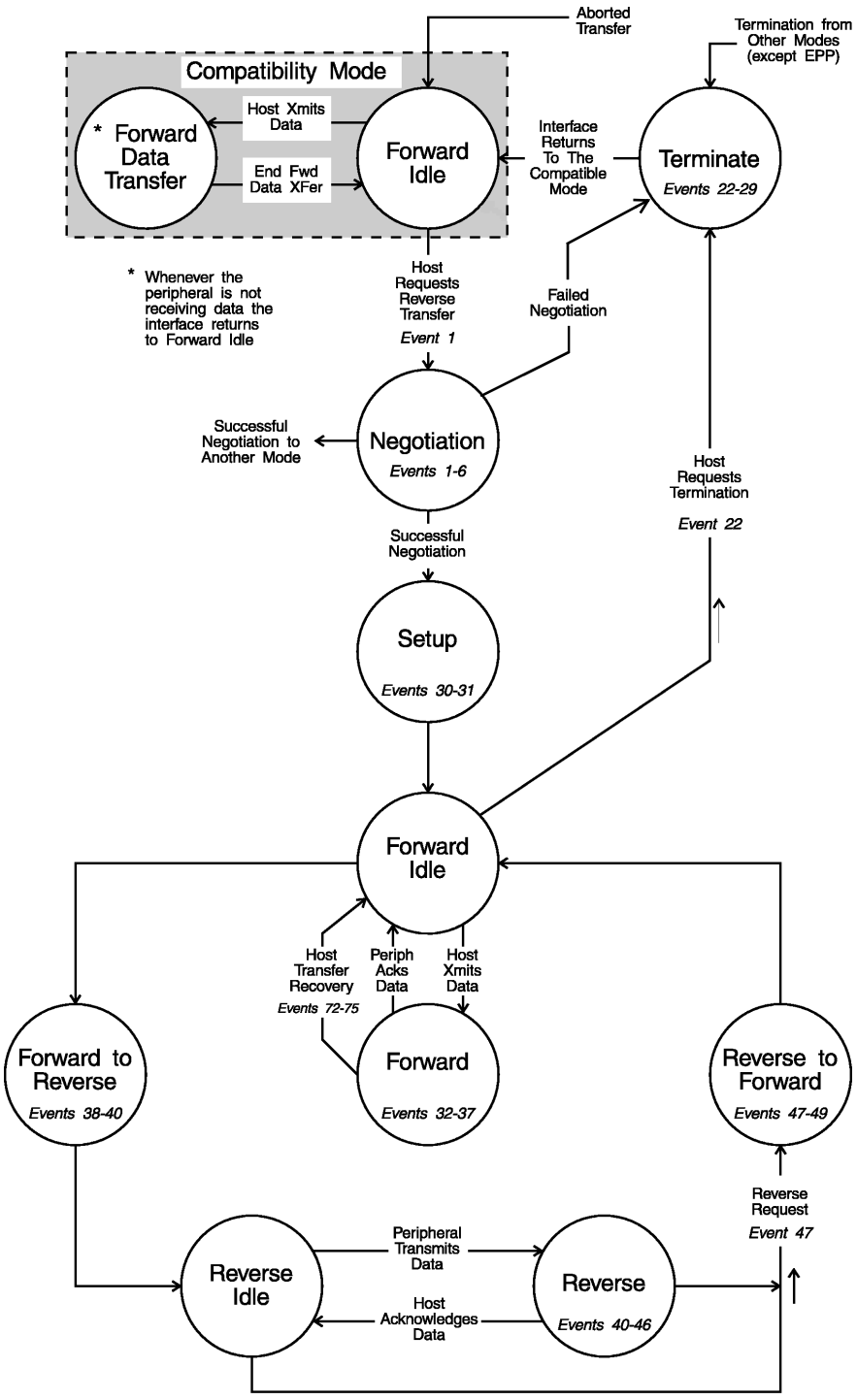


Figure 11—Byte Mode negotiation and transfer



NOTES:

1—Circles represent IEEE 1284 phases.
2—This should not be considered a true state diagram.

Figure 12—ECP Mode phase transitions

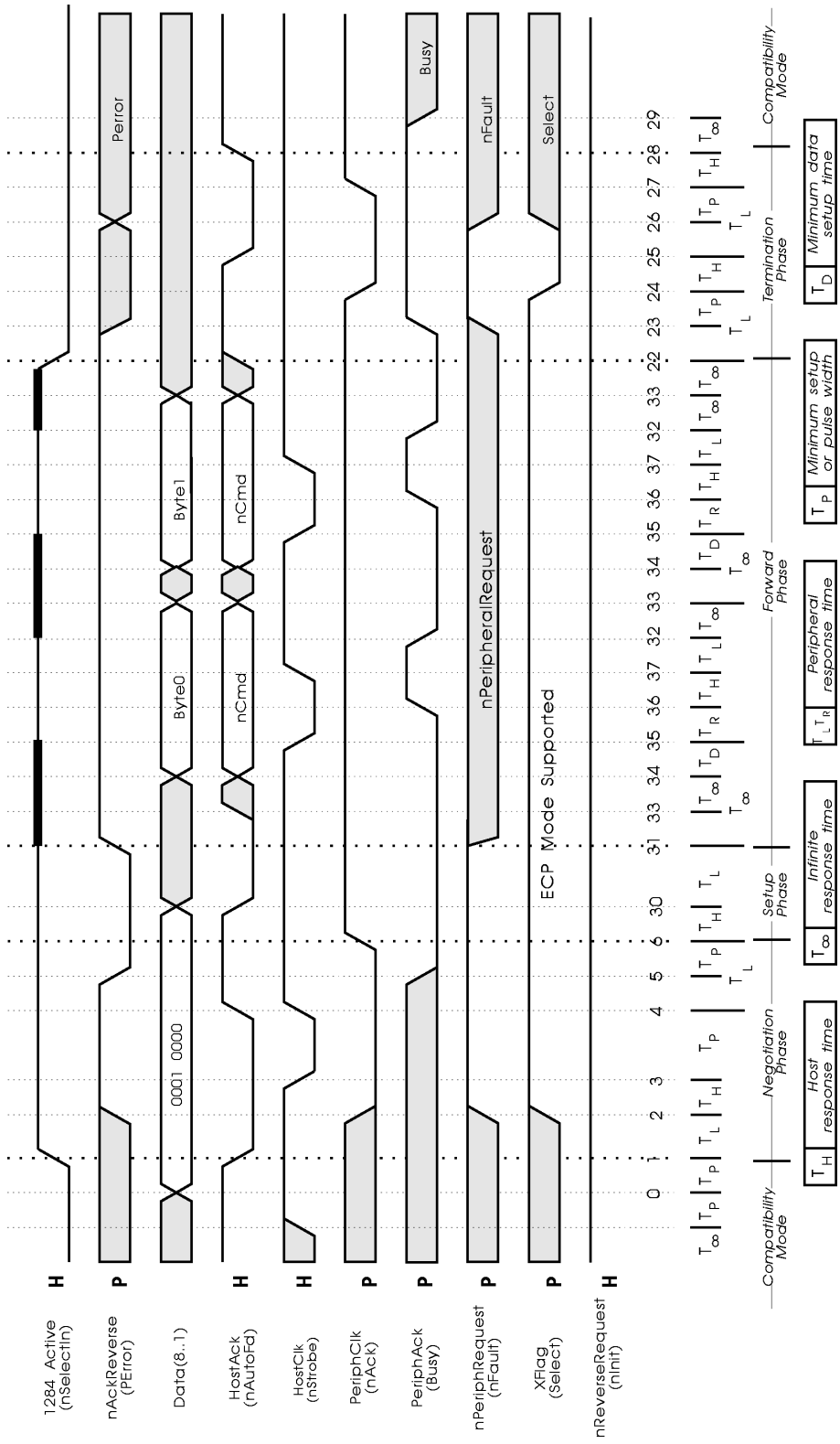


Figure 13—ECP Mode negotiation, setup, forward, and termination

Whenever the peripheral generates data for the host, it may request a reverse channel data transfer if the interface is in the idle phase. This is accomplished by setting nDataAvail(nFault) and PtrClk(nAck) low (event 18), then setting PtrClk(nAck) high (event 19). This generates an interrupt inside the host, assuming the parallel port interrupt is enabled. The host responds to the interrupt by setting HostBusy(nAutoFd) high (event 20). The peripheral then sets AckDataReq(PError) low (event 21) to acknowledge the response of the host. At this point, the interface is in the host busy, data available phase and may be followed by a reverse channel transfer. This sequence is shown in Figure 8 for Nibble Mode operation, which is the same sequence used to interrupt the host for a Byte Mode transfer. Peripherals shall not indicate a new request for a reverse channel transfer unless it is in reverse idle or during negotiation.

7.5.3 ECP Mode

The signals used for the ECP Mode transfer are described in detail in Clause 5. For convenience they are summarized in the table below.

ECP Mode signal name	Compatibility Mode signal name	ECP Mode signal description
HostClk	nStrobe	Used in a closed-loop handshake with PeriphAck(Busy) to transfer data or address information from the host to the peripheral.
Data1...Data8	Data1...Data8	Host-to-peripheral or peripheral-to-host address or data.
PeriphClk	nAck	Used in a closed-loop handshake with HostAck(nAutoFd) to transfer data from the peripheral to the host.
PeriphAck	Busy	The peripheral uses this signal for flow control in the forward direction. PeriphAck also provides a ninth data bit used to determine whether command or data information is present on the data signals in the reverse direction. See 6.9 and Figure 14, events 42–46.
HostAck	nAutoFd	The host drives this signal for flow control in the reverse direction. It is used in an interlocked handshake with PeriphClk(nAck). HostAck also provides a ninth data bit that is used to determine whether command or data information is present on the data signals in the forward direction. See 6.9 and Figure 13, events 34–32.
NReverseRequest	nInit	This signal is driven low to place the channel in the reverse direction. While in the ECP Mode, the peripheral is only allowed to drive the bidirectional data signals when nReverseRequest is low and IEEE 1284 Active is high.
NPeriphRequest	nFault	During ECP Mode, the peripheral shall drive this pin low to request communications with the host. This signal provides a mechanism for peer-to-peer communication. This signal would be typically used to generate an interrupt to the host. This signal is valid in the forward and reverse directions.
NAckReverse	PError	The peripheral drives this signal low to acknowledge nReverseRequest. The host relies upon nAckReverse to determine when it is permitted to drive the data signals.

To begin the negotiation phase, the host places the ECP extensibility request value on the data bus (event 0), then sets IEEE 1284 Active (nSelectIn) high and HostAck (nAutoFd) low (event 1). The peripheral responds by setting PeriphClk (nAck) low, nPeriphRequest (nFault) high, Xflag(Select) high, and nAckReverse (PError) high (event 2). The host then sets HostClk (nStrobe) low (event 3). The host then sets HostClk (nStrobe) and HostAck (nAutoFd) high (event 4), acknowledging that it has recognized an IEEE 1284-compatible peripheral. The peripheral then sets Xflag (Select) high if it supports ECP Mode (event 5). It also sets nAckReverse (PError) and PeriphAck (Busy) low. The peripheral then sets PeriphClk (nAck) high (event 6), indicating that the other status lines may be read. The interface now enters the setup phase. Figure 13 demonstrates a successful negotiation.

If the peripheral does not support ECP Mode, it will set Xflag(Select) low during negotiation. When this occurs, the host shall terminate the session and renegotiate for another transfer mode (see 7.4.2).

The setup phase is entered immediately following a successful negotiation. After seeing PeriphClk (nAck) go high, the host sets HostAck (nAutoFd) low (event 30). The peripheral responds by setting nAckReverse (PError) high (event 31). The interface now enters the forward idle phase. The setup phase transitions are shown in Figure 13.

While in the forward idle or forward data transfer phases, the peripheral may asynchronously assert the nPeriphRequest (nFault) to request that the channel be reversed. When the peripheral is not busy, it sets PeriphAck (Busy) low (event 32).

The host prepares for data transfer by placing data on the bus (event 34). The host sets HostClk (nStrobe) low to indicate valid data on the bus (event 35). The peripheral then sets PeriphAck (Busy) high to acknowledge the hand-shake (event 36). The host then sets HostClk (nStrobe) high (event 37). The peripheral then latches the data and sets PeriphAck (Busy) low, completing the transfer. This sequence is shown in Figure 13.

There is a possibility of the forward channel becoming stalled. The stall condition will exist if the peripheral is unable to accept the data byte being transferred by the host at event 35. In this condition, the peripheral will not acknowledge the handshake (event 36). A handshake, Host Transfer Recovery, has been defined to recover from this condition. Host devices that are not concerned with utilizing the ECP Mode Host Transfer Recovery handshake to break out of an interface stalled condition need not implement this handshake. However, any peripheral that can cause an ECP stall condition by stopping the interface at event 35 (an infinite timeout), shall support the Host Transfer Recovery handshake in the ECP Mode.

The Host Transfer Recovery handshake is defined as follows:

> If the host, following event 35, determines that a stall condition exists, the host may abort the transfer of the current byte by setting nReverseRequest (nInit) low (event 72). The peripheral, regardless of whether it has accepted the byte from the host (event 36 happened), shall discard the byte (if applicable) and acknowledge the host by setting nAckReverse (PError) low. The host then returns nReverseRequest (nInit) high (event 74), and the peripheral follows by returning nAckReverse (PError) high (event 75). This sequence, shown in Figure 15, will return the interface to the state that existed prior to host event 35.

The data transfer timing is designed to provide three one-way cable delays for data setup if data is driven simultaneously with HostClk (nStrobe).

The forward to reverse phase is entered from the forward idle phase. The host places the data bus in a high-impedance state and sets HostAck (nAutoFd) low (event 38). After waiting for the minimum setup time, the host then sets nReverseRequest (nInit) low (event 39). The peripheral then acknowledges the reversal by setting nAckReverse (PError) low (event 40). The peripheral is now permitted to drive the data bus. The interface now enters the reverse phase. This sequence is shown in Figure 14.

The reverse idle or reverse data transfer phases may be entered only from the forward to reverse phase. When the host is not busy, it sets HostAck (nAutoFd) low (event 46).

The peripheral prepares for data transfer by placing data on the bus (event 42). The peripheral then sets PeriphClk (nAck) low to send the data (event 43). The host then sets HostAck (nAutoFd) high to acknowledge the handshake (event 44). The peripheral then sets PeriphClk (nAck) high (event 45). The host is expected to accept the data and sets HostAck (nAutoFd) low (event 46), completing the transfer. This sequence is shown in Figure 14.

The reverse to forward phase is entered from the reverse idle phase. HostAck (nAutoFd) may be high or low when the reverse to forward phase is entered. The host sets nReverseRequest (nInit) high (event 47). The peripheral then places the data bus in a high impedance state, sets PeriphAck (Busy) to indicate the proper forward channel status, and sets PeriphClk (nAck) high (event 48). If the peripheral was in the middle of a data transfer (PeriphClk low), it assumes that the data byte will be discarded by the host and suspends the transfer. After waiting the minimum setup time, the peripheral then sets nAckReverse (PError) high to acknowledge the change of direction (event 49). The host is now permitted to drive the data bus. The interface now enters the forward phase. This sequence is shown in Figure 14.

To terminate from ECP mode, the host sets IEEE 1284 Active (nSelectIn) low and HostAck (nAutoFd) high (event 22), which will initiate one of two types of termination. The first type is a handshake, which allows the peripheral to tell the host when it has returned to compatible mode. The second is an immediate abort, with no guarantee of interface integrity. If the interface was in a “valid” state, which is any state where a reverse data transfer is not in progress, the peripheral will perform the handshake. If the interface was not in a “valid” state, the peripheral will abort immediately. Valid states are indicated in the data transfer diagrams by IEEE 1284 Active (nSelectIn) shown as a heavy line.

To terminate from a valid state, the peripheral will respond to IEEE 1284 Active (nSelectIn) set low by setting PeriphAck (Busy) and nPeriphRequest (nFault) high (event 23). The peripheral will then set Xflag (Select) to its opposite sense and PeriphClk (nAck) low (event 24). The host then sets HostAck (nAutoFd) low (event 25). The peripheral then sets the compatible mode peripheral status on nPeriphRequest (nFault), Xflag (Select), and nAckReverse (PError) (event 26). The peripheral then sets PeriphClk (nAck) high (event 27). The host ends the termination handshake by setting HostAck (nAutoFd) high (event 28), which returns the interface to the Compatibility Mode idle phase. The peripheral may then change PeriphAck (Busy) (event 29) to accept host-to-peripheral data. This sequence is shown following a data transfer in Figure 13.

When IEEE 1284 Active (nSelectIn) is set low in an invalid state, the peripheral aborts immediately. This is to protect both the peripheral and the host. The unexpected transition of IEEE 1284 Active (nSelectIn) and possibly other signals could be caused by a user switching a switch box at the wrong time or by a cable that has worked loose. If a reverse channel data transfer is aborted, the current byte in transit is lost, but the peripheral will hold that byte in its output register. The next time the host performs a reverse channel transfer, that byte will be the first one sent.

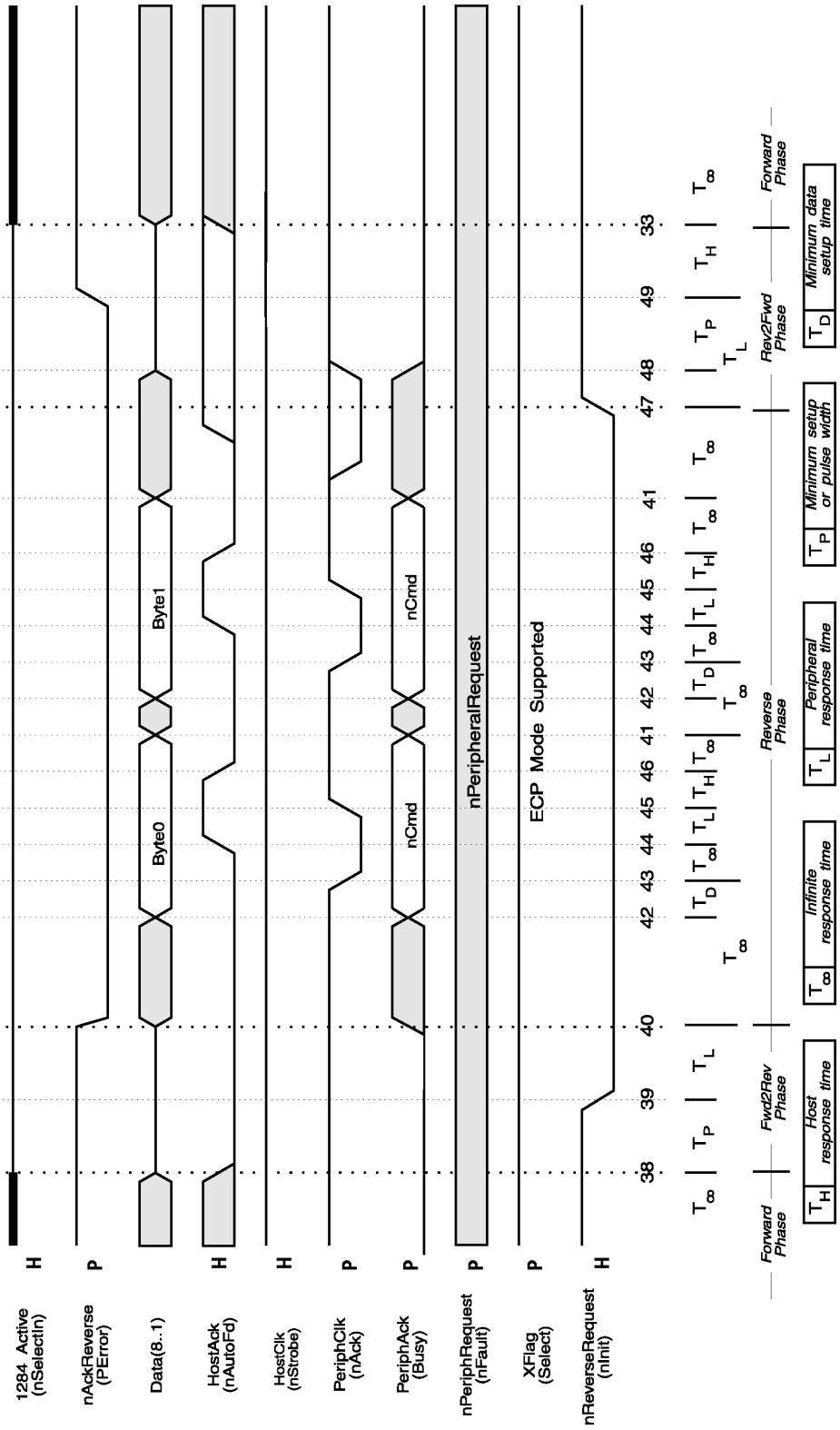


Figure 14—ECP Mode forward to reverse, reverse, and reverse to forward

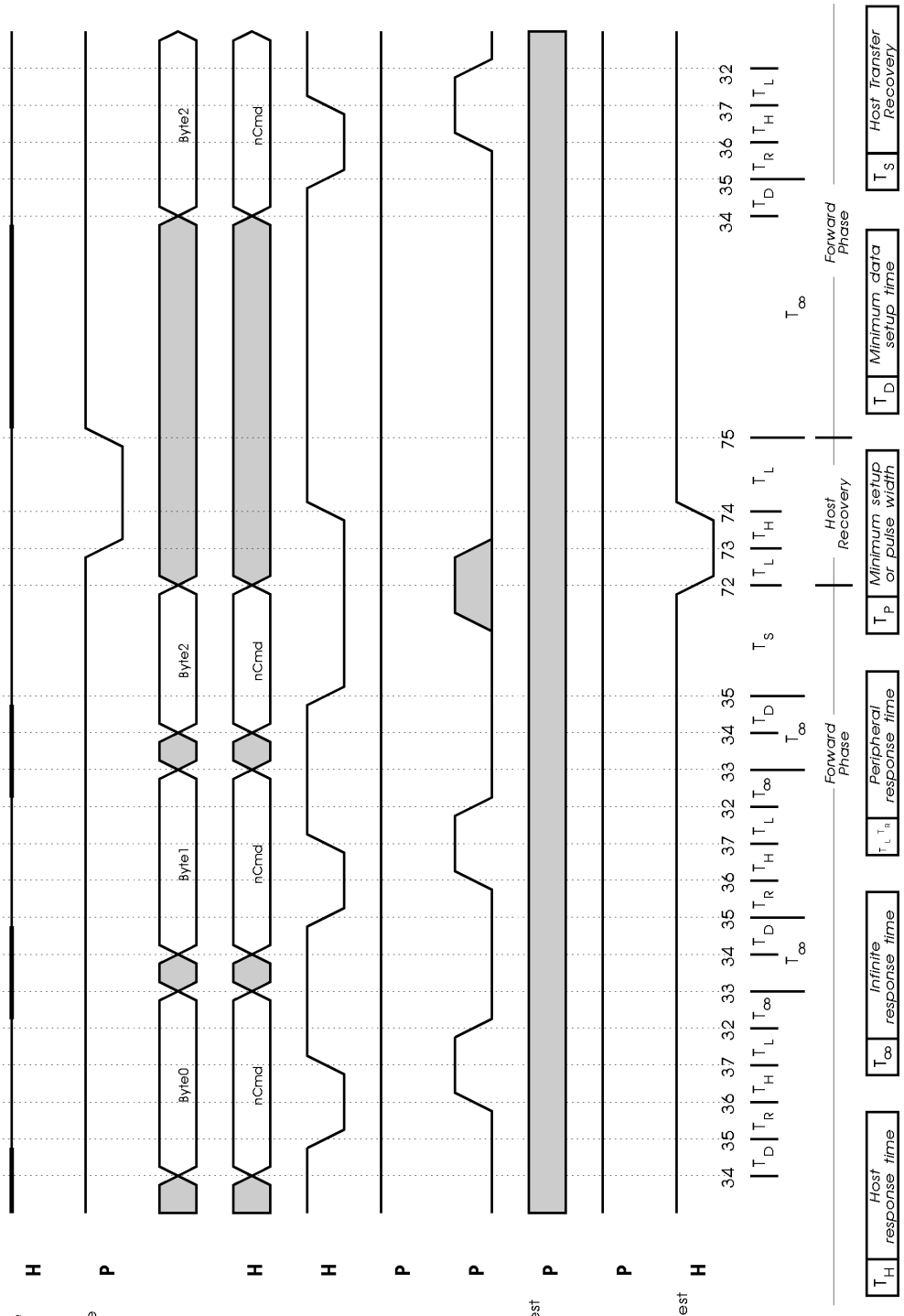


Figure 15—ECP Mode host transfer recovery

7.5.4 EPP Mode

The signals used for the EPP Mode transfer are described in detail in Clause 5. For convenience, they are summarized in the table below.

EPP Mode signal name	Compatibility Mode signal name	EPP Mode signal description
nWrite	nStrobe	Set low to denote an address or data write operation to the peripheral. Set high to denote an address or data read operation from the peripheral.
AD1...AD8	Data1...Data8	Host-to-peripheral or peripheral-to-host address or data.
Intr	nAck	Used by the peripheral to interrupt the host. This signal is pulsed low for a minimum of 500 ns and must be raised high for a minimum of 1 μs before reasserting.
nWait	Busy	This signal should be driven inactive as a positive acknowledgment from the peripheral that transfer of data or address is completed. The signal is active low. It should be driven active as an indication that the device is ready for the next address or data transfer.
nDStrb	nAutoFd	This signal is active low. It is used to denote a data cycle.
nInit	nInit	This signal is active low. When driven active (low), this signal initiates a termination cycle that results in the interface returning to Compatibility Mode.
nAStrb	nSelectIn	This signal is used to denote an address cycle. It is active low.
User defined 1	PError	This signal is manufacturer specific and beyond the scope of this standard.
User defined 2	nFault	This signal is manufacturer specific and beyond the scope of this standard. It is commonly used to indicate data is available as in ECP mode.
User defined 3	Select	This signal is manufacturer specific and beyond the scope of this standard.

The fundamental EPP cycles are described in the following paragraphs. They are accompanied by figures showing the handshaking sequences. Along the bottom of each figure are numbers corresponding to signal transition events. Each of these events is listed in Annex B along with information regarding timing constraints relative to other transition events. The event numbers are also designated by parentheses in the following paragraphs.

Typically, EPP operates on a two-phase bus cycle. First, the host selects the register within a device for subsequent operations. Second, the host performs a series of read and/or write byte operations to the selected register. All operations on EPP devices are performed asynchronously.

EPP provides a single interrupt request signal for the device. An EPP device can generate an interrupt request to the host by asserting Intr. The EPP specification does not include an interrupt acknowledge cycle or any means of masking the interrupt. These functions may be implemented at a higher layer in a device-specific manner. The interrupt operation is independent of other cycles. Though the interrupt signal may be asserted at any time, interrupts will not interfere with EPP cycles.

Upon negotiaion into EPP mode, the peripheral sets Xflag(Select) high if it supports EPP mode (event 5). It also sets nWait(Busy) low. After negotiating into EPP Mode, the XFlag, AckDataReq, and nDataAvail signals may be used for user-defined peripheral status bits.

Four operations are supported on EPP

a) Address write
b) Data write
c) Address read
d) Data read

These operations are defined as follows.

To begin an address write cycle (Figure 17), the host asserts nWrite, places the register address on the data signals, and asserts nAStrb (event 56). The peripheral responds by de-asserting nWait (event 58) to indicate that it is ready to receive the address byte. When the host recognizes nWait as inactive, it will de-assert nAStrb (event 59) to latch the address byte into the device. The peripheral acknowledges the end of the cycle and indicates that it is ready for the next cycle to begin by asserting nWait (event 60). The host can then modify the address/data on the data signals and change the state of the nWrite signal.

To begin a data write cycle (Figure 18), the host asserts nWrite, drives the data signals, and asserts nDStrb (event 62). The peripheral responds by de-asserting nWait (event 58) to indicate that it is ready to receive the data byte. When the host recognizes nWait as inactive, it will de-assert nDStrb (event 63) to latch the data byte into the device. The peripheral acknowledges the end of the cycle and indicates that it is ready for the next cycle to begin by asserting nWait (event 60). The host can then modify the address/data on the data signals and change the state of the nWrite signal (event 61).

To begin an address read cycle (Figure 19), the host de-asserts nWrite and places the data signals in a high impedance state, then asserts nAStrb (event 64). The peripheral responds by driving the data signals with the address byte (event 65) and then de-asserting nWait (event 58) to indicate that the address is valid. When the host recognizes nWait as inactive, it will read the address from the data signals and de-assert nAStrb (event 59). The peripheral places the data signals in a high impedance state (event 66), then acknowledges the end of the cycle, and indicates that it is ready for the next cycle to begin by asserting nWait (event 60). The host can then drive the data signals and change the state of the nWrite signal.

To begin a data read cycle (Figure 20), the host de-asserts nWrite and places the data signals in a high impedance state, then asserts nDStrb (event 67). The peripheral responds by driving the data signals (event 65) and de-asserting nWait (event 58) to indicate that the data is valid. When the host recognizes nWait as inactive, it will read the data from the data signals and de-assert nDStrb (event 63). The peripheral places the data signals in a high impedance state (event 66), acknowledges the end of the cycle, and indicates that it is ready for the next cycle to begin by asserting nWait (event 60). The host can then drive the data signals and change the state of the nWrite signal.

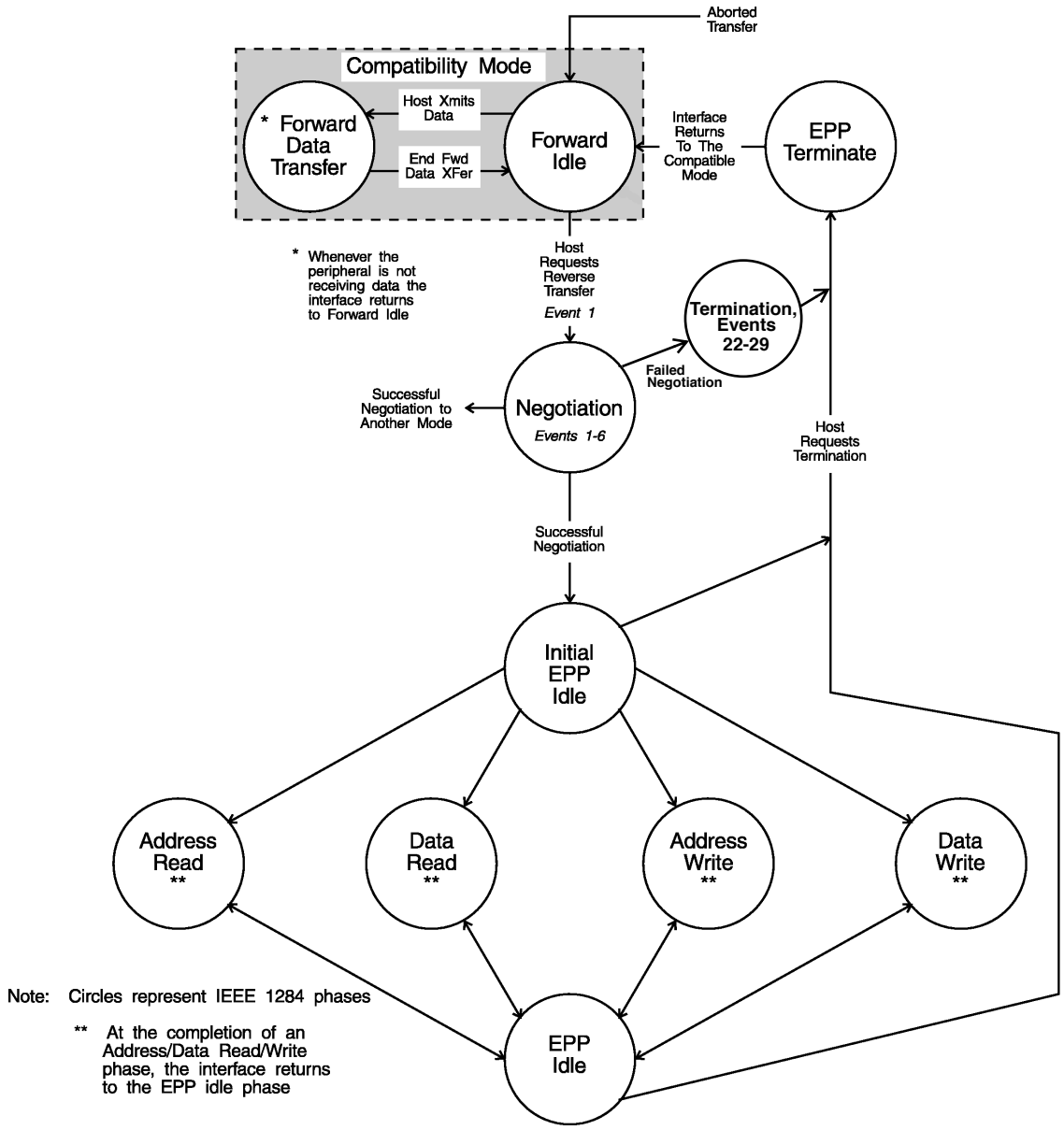


Figure 16—EPP Mode phase transitions

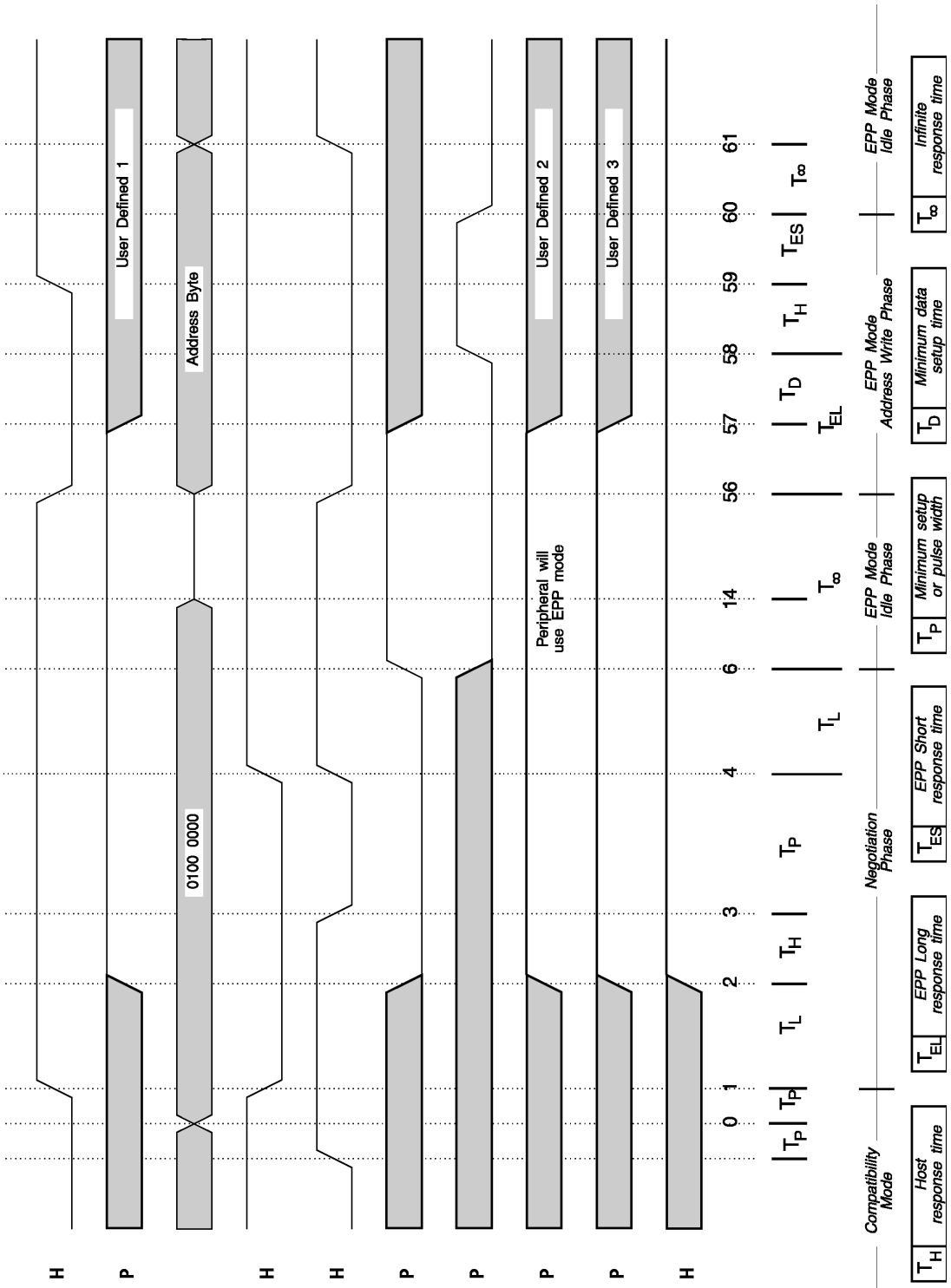


Figure 17—EPP Mode negotiation, address write, and idle

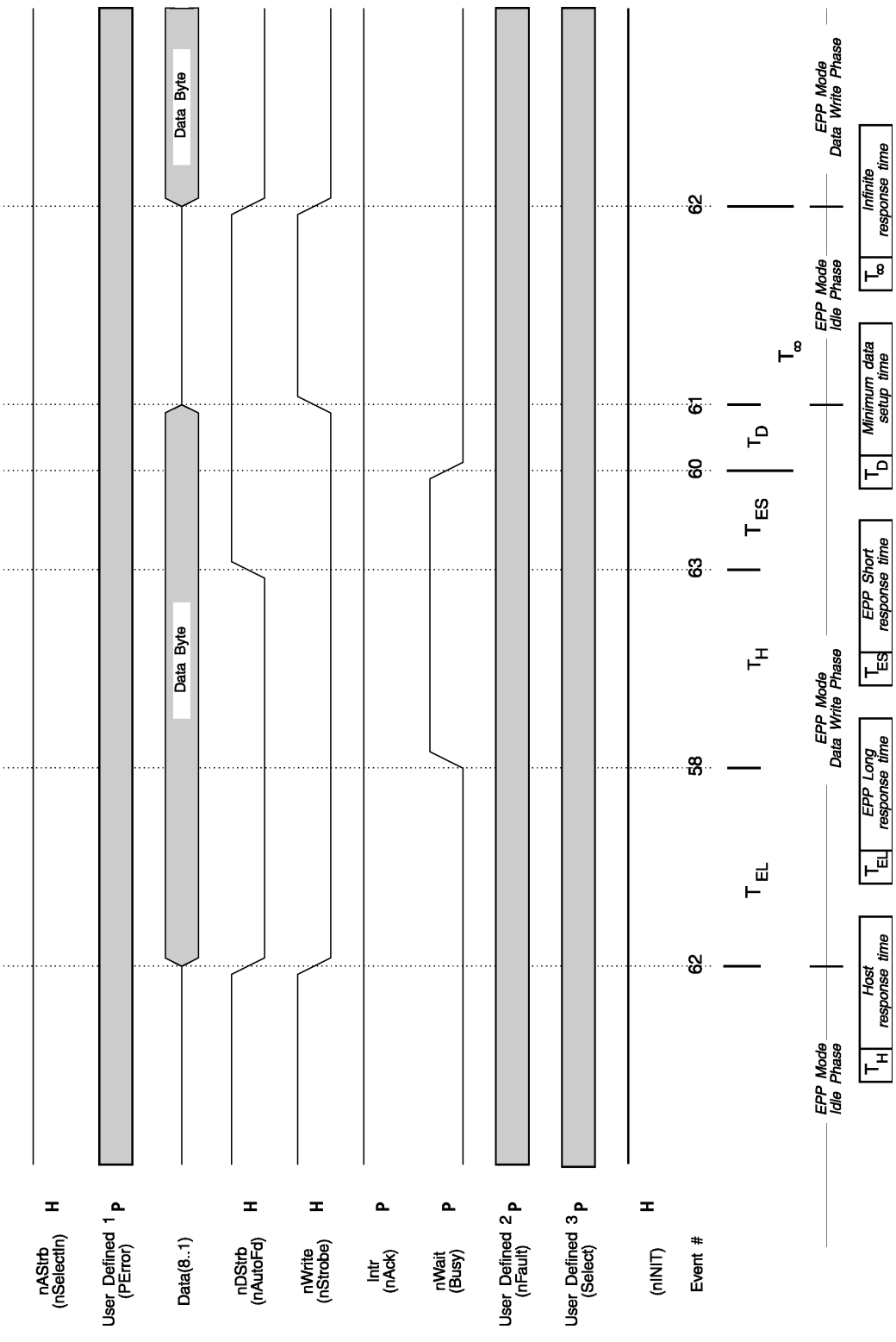


Figure 18—EPP Mode data write

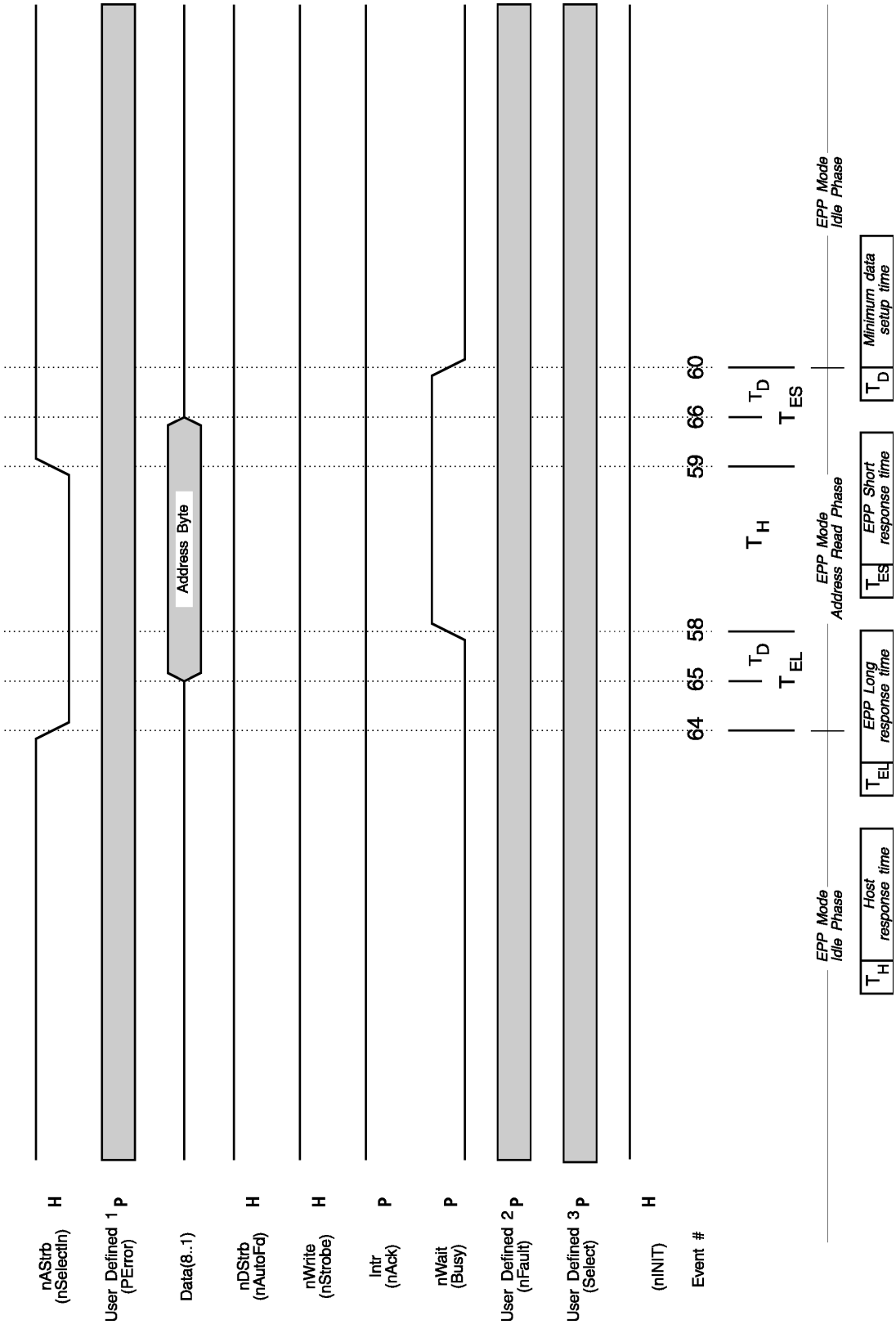


Figure 19—EPP Mode address read

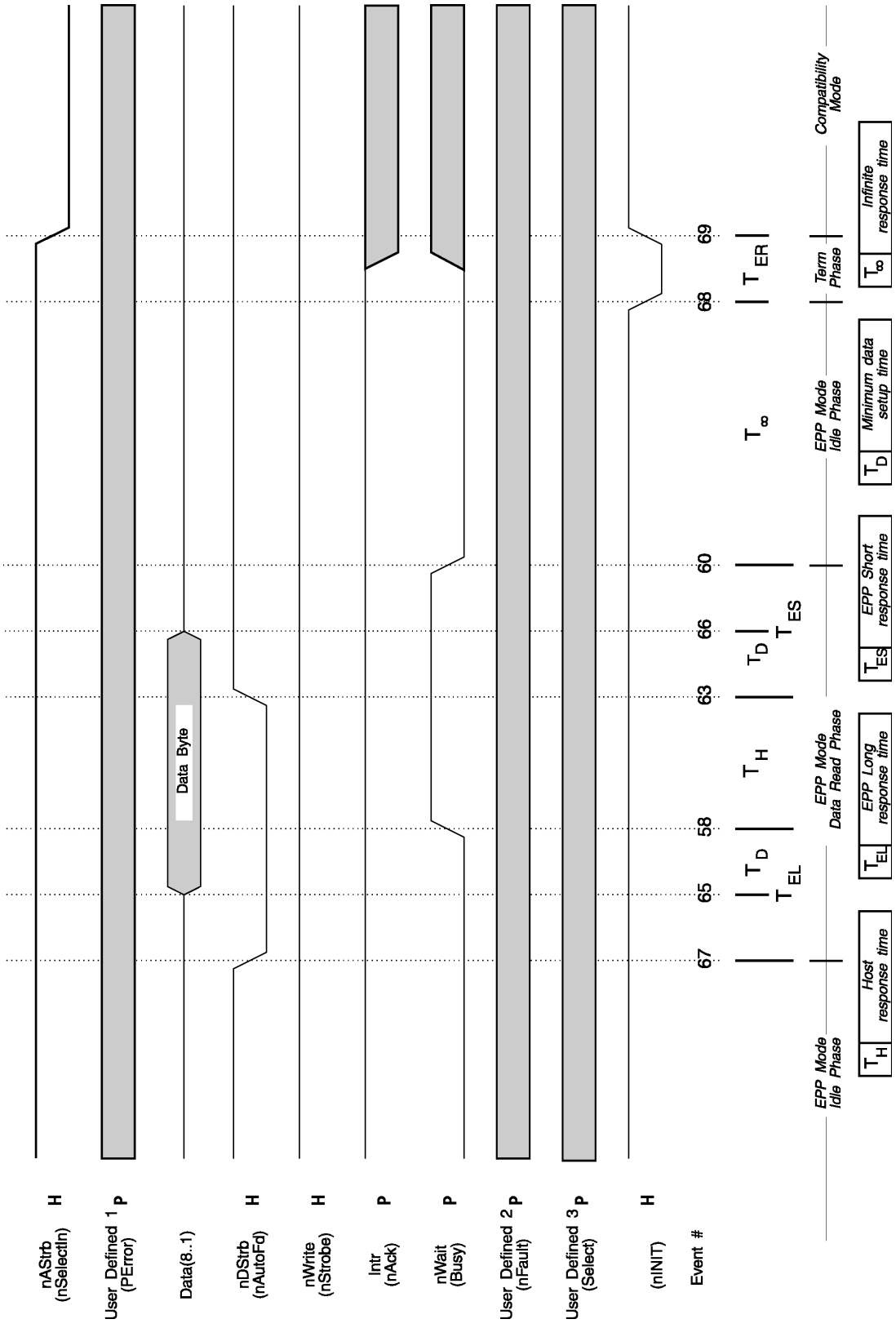


Figure 20—EPP Mode data read and termination

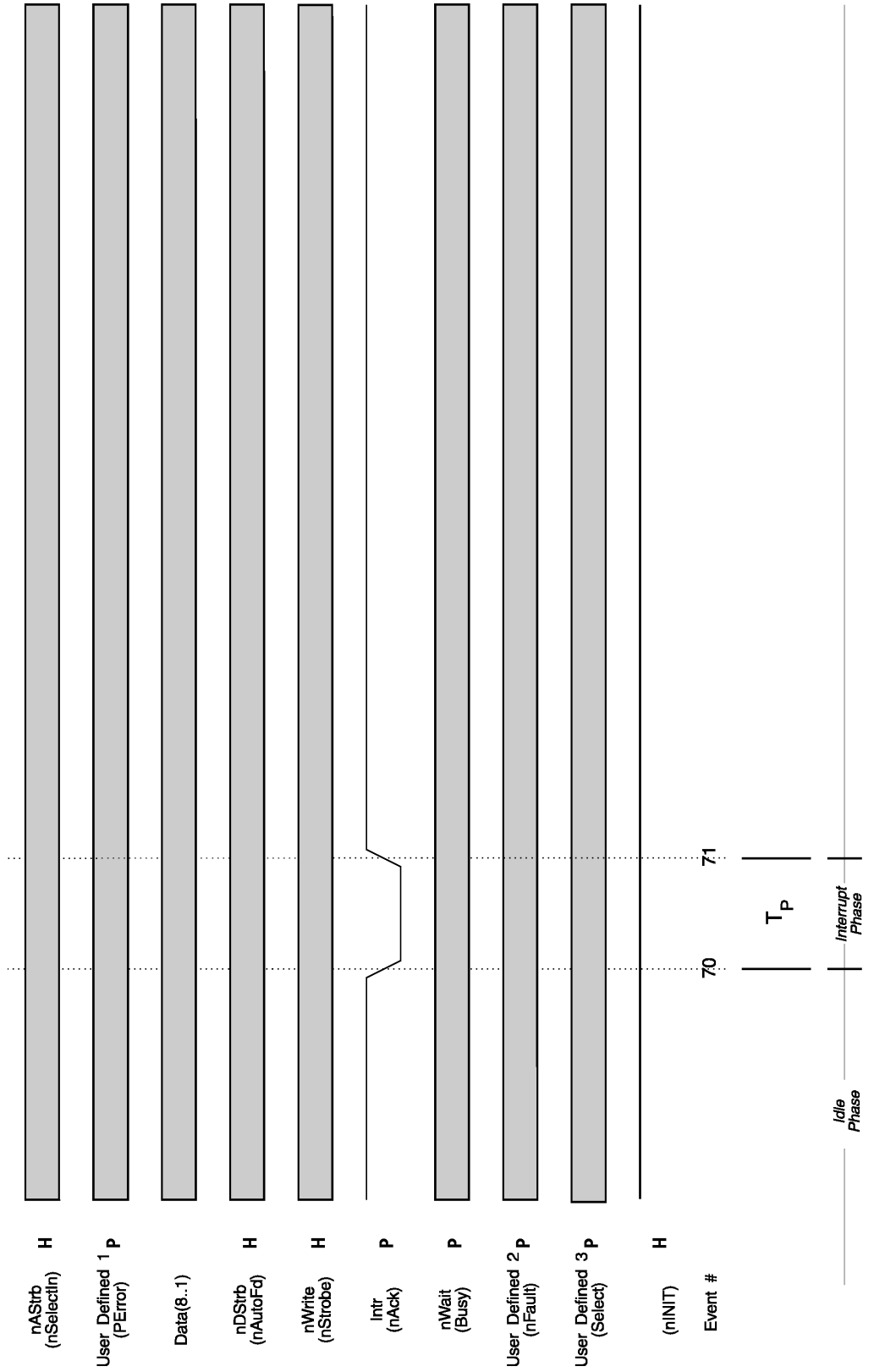


Figure 21—EPP Mode peripheral interrupt

7.6 Device ID

The Device ID is a length field followed by a case-sensitive string of ASCII characters defining peripheral characteristics and/or capabilities. To instruct the peripheral to send this string to the host, the host (from the Compatibility Mode) shall request a peripheral-to-host data transfer using the appropriate combination of bits in the extensibility byte.

The Device ID will be sent from the peripheral to the host using the mode requested with the extensibility byte. The host is required to interpret the first two bytes of the peripheral-to-host message as a message length field.

If the peripheral accepts the Device ID negotiation but indicates that data is not available at either the start of the Device ID reverse transfer or at any time before the full count has been sent to the host, the host shall either enter the reverse idle phase or enter the termination phase. If the host terminates the Device ID transfer before all bytes have been transferred, the peripheral will discard the remainder of the Device ID string. Therefore, each time the host requests Device ID, the complete string will be sent.

The Device ID is a sequence of bytes. The first two bytes are the length of the sequence, including the two length bytes. The first byte is the most significant byte. (Length values of x'0000', x'0001', and x'0002' are reserved.) *After receiving the sequence, the host is required to return the link to Compatibility Mode regardless of whether the peripheral has additional data to send.*

Following the two length bytes, the sequence is composed of a series of keys and values of the form:

```
key: value {,value};
```

repeated for each key. As indicated, each key will have at least one value, and may have more than one value. The minimum necessary keys (case-sensitive) are `MANUFACTURER`, `COMMAND SET`, and `MODEL`. (These keys may be abbreviated as `MFG`, `CMD`, and `MDL`, respectively.) Each implementation will supply these three keys and possibly additional ones as well. Each key (and each value) is a string of characters. Any characters except colon (:), comma (,), and semicolon (;) may be included as part of the key (or value) string. Any leading or trailing white space characters (SPACE[x'20'], TAB[x'09'], VTAB[x'0B'], CR[x'0D'], NL[x'0A'], or FF[x'0C']) in the string is ignored by the parsing program (but is still counted as part of the overall length of the sequence).

An example ID String, in a form similar to an assembler DB directive, showing optional `COMMENT` and `ACTIVE COMMAND SET` keys and their associated values (the text is actually all on one line)

```
x'00',x'79',
MANUFACTURER:ACME Manufacturing;
COMMAND SET:PCL,MPL;
MODEL:LaserBeam ?;
COMMENT:Anything you like;
ACTIVE COMMAND SET:PCL;
```

NOTE—x'00' denotes a hexadecimal value of zero; the x and single quotes are not part of the string.

While more expensive to store and transfer than a 1–4 bytes ID code, an ID string in this format does not require administration by a central authority to assign codes for devices, manufacturers, etc.

7.7 Termination

To terminate either the Nibble, Byte, or ECP IEEE 1284 Modes, the host sets IEEE 1284 Active(nSelectIn) low and HostBusy (nAutoFd) high if not set already (event 22), which will initiate one of two types of termination. The first type is a handshake, which allows the peripheral to tell the host when it has returned to Compatibility Mode. The second is an immediate abort, with no guarantee of interface integrity. If the interface was in a "valid" state, which is any state where a reverse data transfer is not in progress, the peripheral will perform the handshake. If the interface was not in a "valid" state, the peripheral will abort immediately. Valid states are indicated in the data transfer diagrams by IEEE 1284 Active(nSelectIn) shown as a heavy line.

To terminate the EPP Mode of a IEEE 1284 device (Figure 20), the host asserts the nInit signal (event 68). The peripheral shall be reset to its initial Compatibility Mode (event 69).

7.7.1 Valid state termination

To terminate from a valid state, the peripheral will respond to IEEE 1284 Active(nSelectIn) set low by setting PtrBusy(Busy) and nDataAvail(nFault) high (event 23). The peripheral will then set Xflag(Select) to its opposite sense and PtrClk(nAck) low (event 24). The host then sets HostBusy(nAutoFd) low (event 25). The peripheral then sets the Compatibility Mode peripheral status on nDataAvail(nFault), Xflag(Select), and AckDataReq(PError) (event 26). The peripheral then sets PtrClk(nAck) high (event 27). The hosts ends the termination handshake by setting HostBusy(nAutoFd) high (event 28), which returns the interface to the Compatibility Mode idle phase. The peripheral may then change PtrBusy(Busy) (event 29) to accept host-to-peripheral data. This sequence is shown following a data transfer in Figure 9, and from the reverse idle phase in Figure 22.

7.7.2 Immediate termination

When IEEE 1284 Active(nSelectIn) is set low in an invalid state, the peripheral aborts immediately. This is to protect both the peripheral and the host. The unexpected transition of IEEE 1284 Active(nSelectIn) and possibly other signals could be caused by a user switching a switch box at the wrong time or by a cable that has worked loose. If a reverse channel data transfer is aborted, the current byte in transit is lost, but the peripheral will hold that byte in its output register. The next time the host performs a reverse channel transfer, that byte will be the first one sent.

NOTE—This immediate termination is not supported in EPP Mode, which has no provision for operation with conventional switch boxes.

7.8 Collisions

When in the idle phase, the peripheral can report that peripheral-to-host data is available. It is possible that this may occur at the same time that the host attempts to terminate from idle phase and return to Compatibility Mode.

The peripheral will begin the interrupt phase (events 18 and 19). If IEEE 1284 Active (nSelectIn) goes low prior to HostBusy (nAutoFd) changing from a high to a low state (event 7; not shown), the peripheral should recognize the host has entered the termination phase and should complete the normal termination handshake (events 23 through 29). This handshake sequence is shown in Figure 23.

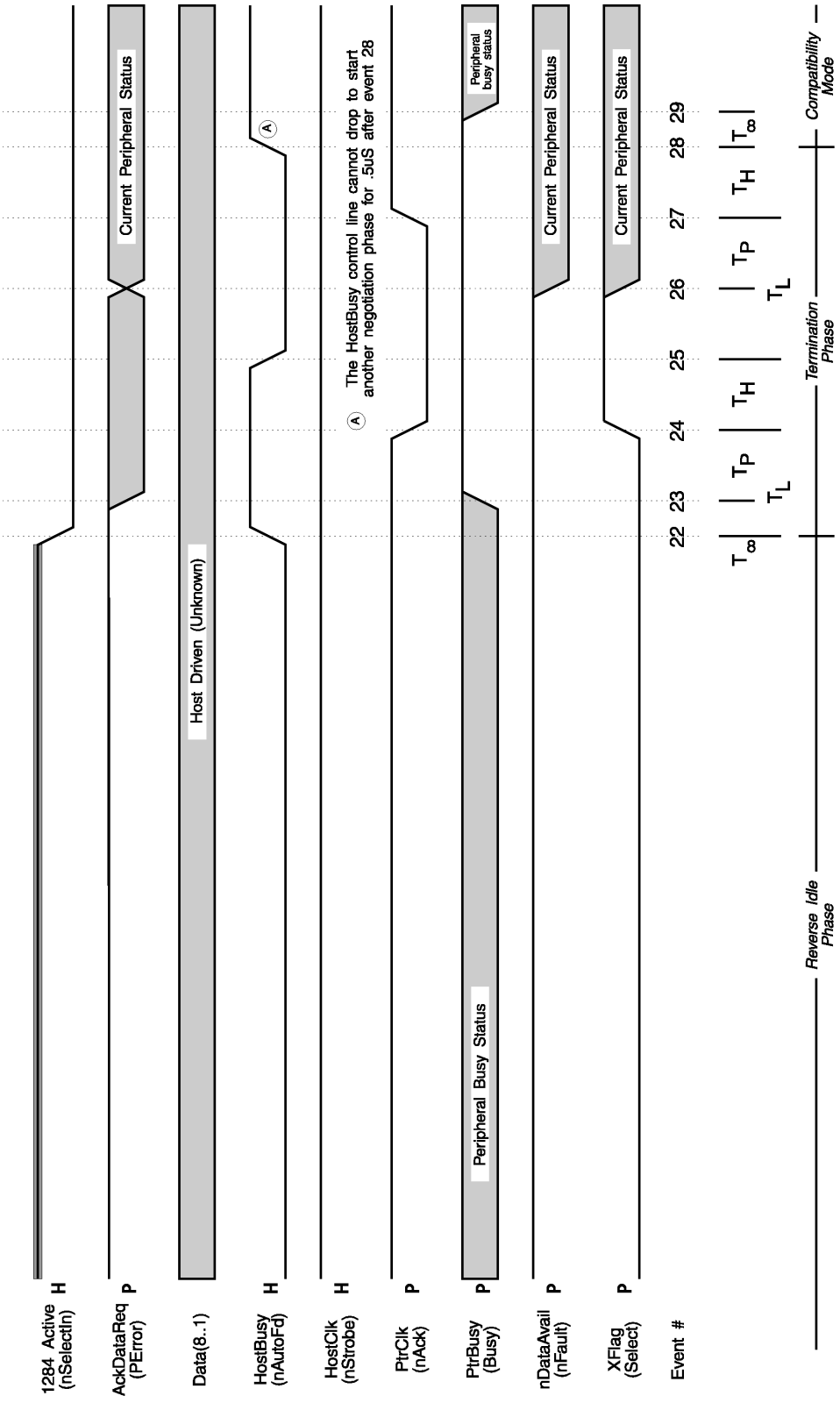


Figure 22—Nibble Mode idle phase to termination

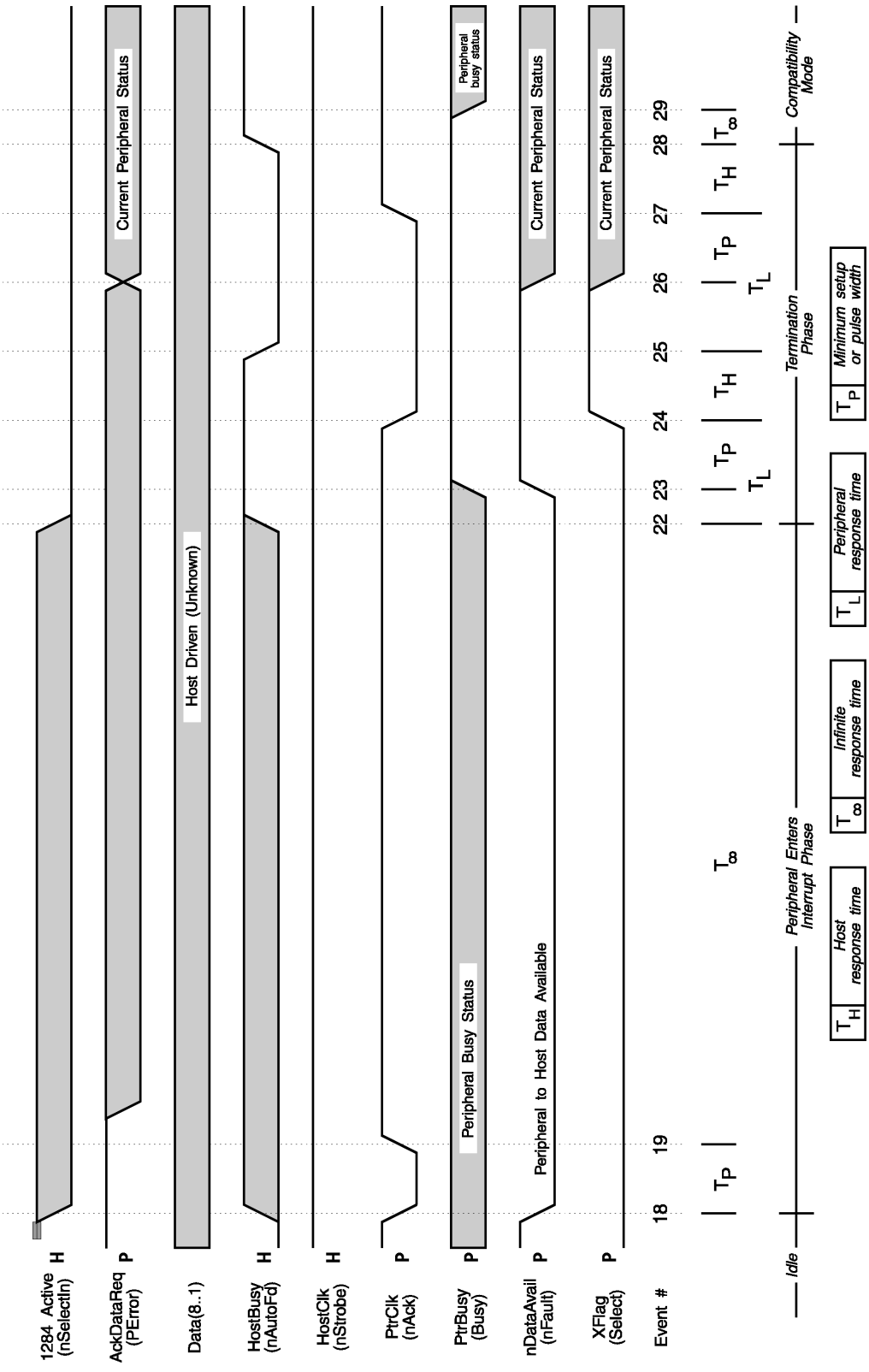


Figure 23—Nibble Mode termination collision

8. Mechanical and electrical interface

8.1 General considerations

These physical interface specifications are provided to define the interconnection of parallel port devices. Input and output interface signals are assumed to be industry standard 5 V dc transistor-transistor logic (TTL)-level compatible. Devices may be connected by a cable or directly connected (connector-to-connector).

The broad objectives and assumptions underlying the recommendation contained throughout this clause are

a) To provide a low-cost means for communication between a host computer and peripheral.
b) To ensure compatibility of independently developed mechanical and electrical interfaces.
c) To provide ease of installation and service.
d) To make use of existing cables and connectors.
e) To provide compatibility with well-behaved and well-designed host and peripherals already in service.

8.2 Mechanical characteristics

Three interface connectors are defined. These three connectors are denoted IEEE 1284-A, IEEE 1284-B, and IEEE 1284-C. The IEEE 1284-A and IEEE 1284-B connectors are equivalent to existing connectors on compatible devices. The IEEE 1284-C connector is a new, higher density connector and is recommended for new designs.

Intermateability is not supported directly between the existing connectors (IEEE 1284-A and IEEE 1284-B) and the new connector (IEEE 1284-C). If required, interposer blocks or cables shall be constructed using the connections shown in Figure 27 through Figure 31.

All specified connectors are available in a variety of orientations. No specific orientation is recommended. Sufficient space shall be provided around the device connector to allow for the mating connector, connector shell, and latch assembly.

8.2.1 Connectors

NOTE—The notes to Figure 24 through Figure 26 include several example part numbers from various manufacturers. These lists are not all-inclusive nor are they meant to imply exclusive or recommended parts. Part numbers are current as of the approval date of this standard.

The IEEE 1284-A connector (see Figure 24) is the existing DB25 connector series commonly used for the host side. This connector is available from a variety of manufacturers. The cable shall be retained using 4-40 jack screws and stand-offs as shown in Figure 24.

The IEEE 1284-B connector (see Figure 25) is the existing 36-pin 0.085 centerline connector series commonly used on the peripheral side. This connector is available from a variety of manufacturers. The cable shall be retained using wire bails and latches as shown in Figure 25.

The IEEE 1284-C connector (see Figure 26) is a new 36-pin 0.050 centerline connector recommended by this standard. This connector is used for both host and peripheral sides of the interface. This connector is available from a variety of manufacturers. The cable shall be retained using clip latches as shown in Figure 26.

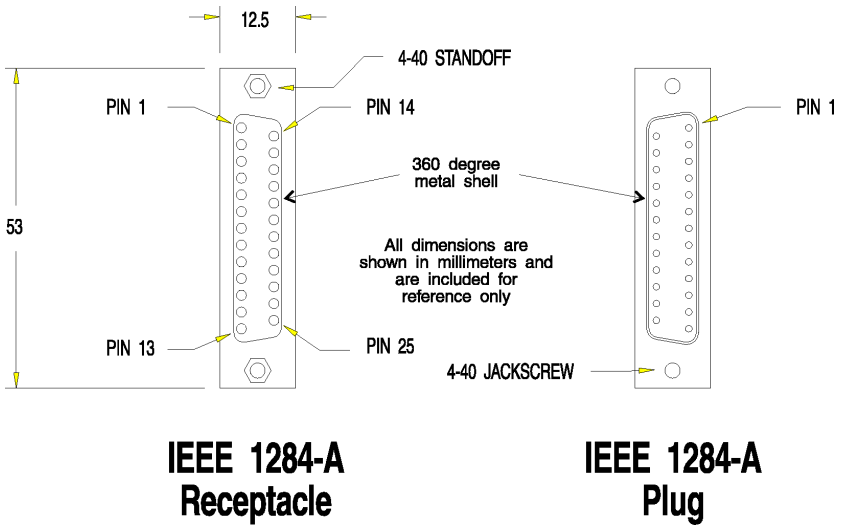


Figure 24—Physical drawing of the IEEE 1284-A connector

NOTE—The following example part numbers are provided for the convenience of the reader. They are neither exclusive nor recommended. IEEE 1284-A: PC Mount Receptacle—AMP 747846-4, 3M 8325-60XX & 89925-X00X, Molex 82009 Series; Cable Plug—AMP 747948-1, 3M 8225-X0XX, Molex 71527 Series; Shielded Cover Kit with Latches—AMP included with 747948-1, 3M 3357-9225 & 3357-6525, Molex 82002 Series.

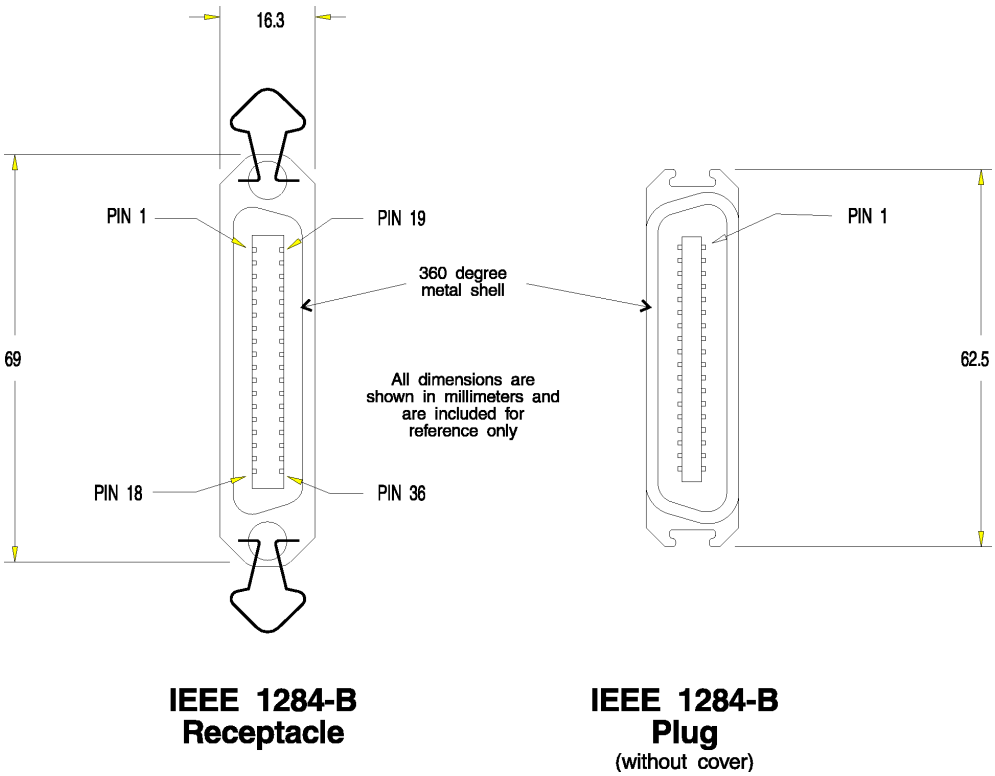


Figure 25—Physical drawing of the IEEE 1284-B connector

NOTE—The following example part numbers are provided for the convenience of the reader. They are neither exclusive nor recommended. IEEE 1284-B: PC Mount Receptacle—AMP 555119-1, 3M 3367-300X & 3448-62, Molex 71519 Series, Amphenol Series 57-40360; Cable Plug—AMP 554950-1, Molex 71522 Series; Shielded Cover Kit with Latches—AMP 554945-1, Molex 71523 Series.

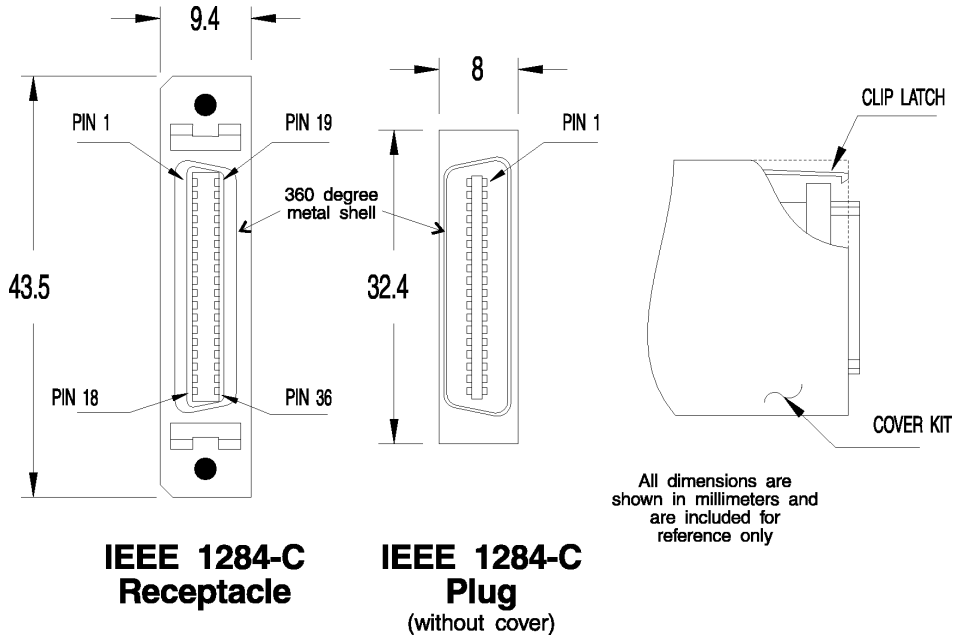


Figure 26—Physical drawing of the IEEE 1284-C connector

NOTE—The following example part numbers are provided for the convenience of the reader. They are neither exclusive nor recommended. IEEE 1284-C: PC Mount Receptacle —AMP 2-175925-5, 2-176971-5 & 2-178238-5, 3M 10236-52A2VC, 10236-5202VC, N10236-52A2VC, N10236-5202VC, 10236-6202VC & 10236-0200EC, Molex 52311-3611, 52311-3621, 52515-3611, 52679-3611, 52696-3611, 52698-3611, 52699-3621 & 52700-3611, Harting 60-11-036-5132 & 60-11-036-5140; Cable Plug—AMP 2-175677-5, 3M 10136-6000EC, Molex 52316-3611, Harting 60-13-036-5200; Shielded Cover Kit with Latches—AMP 176793-5, 3M 10336-A200-00, 10336-3210-00X & 10336-A500-00, Molex 52370-3600 & 52370-3610, Harting 60-13-036-0153.

Devices intended to be interconnected by cables shall be constructed with the connector receptacles attached to the devices. Interconnection cables for such devices shall be constructed with the connector plugs at each end of the cable. Devices intended for direct attachment shall be constructed with the connector plug attached to the device. Interconnection cables for such devices shall be constructed with a connector receptacle at one end and a connector plug at the other end of the cable. In either case, the device connector pin assignments shall conform to 8.2.2 and the cable wiring shall conform to 8.2.3.

Device connectors shall provide a 360˚ metal shell (see Figure 24 through Figure 26) to provide cable shield continuity into the device on both ends.

8.2.2 Connector pin assignments

Pinout description of the IEEE 1284 connectors (IEEE 1284-A, IEEE 1284-B, and IEEE 1284-C) are shown in Table 6, Table 7, and Table 8. These tables show the signal pin number, source, and signal name for each signal. The sources are: H to designate sourced by the host, P to designate sourced by the peripheral, and Bi-Di to designate a bidirectional signal (one that can be sourced by either the host or the peripheral). Signal names are listed for each mode supported.

8.2.2.1 IEEE 1284-A

Table 6 shows the pin numbers and their assigned signal names for the IEEE 1284-A connector. The 1284-A connector is a 25 pin subminiature D-shell connector. A physical description of the IEEE 1284-A connector can be found in Figure 24.

Table 6—IEEE 1284-A connector pin assignments

Pin #	Source	Compatible	Nibble	Byte	ECP	EPP
1	H	nStrobe	HostClk	HostClk	HostClk	nWrite
2	Bi-Di[a]	Data 1 (Least Significant Bit)				AD1
3	Bi-Di[a]	Data 2				AD2
4	Bi-Di[a]	Data 3				AD3
5	Bi-Di[a]	Data 4				AD4
6	Bi-Di[a]	Data 5				AD5
7	Bi-Di[a]	Data 6				AD6
8	Bi-Di[a]	Data 7				AD7
9	Bi-Di[a]	Data 8 (Most Significant Bit)				AD8
10	P	nAck	PtrClk	PtrClk	PeriphClk	Intr
11	P	Busy	PtrBusy	PtrBusy	PeriphAck	nWait
12	P	PError	AckDataReq	AckDataReq	nAckReverse	User defined 1
13	P	Select	Xflag	Xflag	Xflag	User defined 3
14	H	nAutoFd	HostBusy	HostBusy	HostAck	nDStrb
15	P	nFault	nDataAvail	nDataAvail	nPeriphRequest	User defined 2
16	H	nInit	nInit	nInit	nReverseRequest	nInit
17	H	nSelectIn	IEEE 1284 Active	IEEE 1284 Active	IEEE 1284 Active	nAStrb
18	Signal Ground (nStrobe)					
19	Signal Ground (Data 1 and Data 2)					
20	Signal Ground (Data 3 and Data 4)					
21	Signal Ground (Data 5 and Data 6)					
22	Signal Ground (Data 7 and Data 8)					
23	Signal Ground (Busy and nFault)					
24	Signal Ground (PError, Select, and nAck)					
25	Signal Ground (nAutoFd, nSelectIn, and nInit)					

[a]Data signals can be read by some but not all host devices.

8.2.2.2 IEEE 1284-B

Table 7 shows the pin numbers and their assigned signal names for the IEEE 1284-B connector. The 1284-B connector is a 36-pin ribbon type connector. A physical description of the IEEE 1284-B connector can be found in Figure 25.

Table 7—IEEE 1284-B connector pin assignments

Pin #	Source	Compatible	Nibble	Byte	ECP	EPP
1	H	nStrobe	HostClk	HostClk	HostClk	nWrite
2	Bi-Di[a]	Data 1 (Least Significant Bit)				AD1
3	Bi-Di[a]	Data 2				AD2
4	Bi-Di[a]	Data 3				AD3
5	Bi-Di[a]	Data 4				AD4
6	Bi-Di[a]	Data 5				AD5
7	Bi-Di[a]	Data 6				AD6
8	Bi-Di[a]	Data 7				AD7
9	Bi-Di[a]	Data 8 (Most Significant Bit)				AD8
10	P	nAck	PtrClk	PtrClk	PeriphClk	Intr
11	P	Busy	PtrBusy	PtrBusy	PeriphAck	nWait
12	P	PError	AckDataReq	AckDataReq	nAckReverse	User defined 1
13	P	Select	Xflag	Xflag	Xflag	User defined 3
14	H	nAutoFd	HostBusy	HostBusy	HostAck	nDStrb
15		Not defined				
16		Logic Gnd				
17		Chassis Gnd				
18	P	Peripheral Logic High				
19		Signal Ground (nStrobe)				
20		Signal Ground (Data 1)				
21		Signal Ground (Data 2)				
22		Signal Ground (Data 3)				
23		Signal Ground (Data 4)				

Table 7—IEEE 1284-B connector pin assignments (continued)

Pin #	Source	Compatible	Nibble	Byte	ECP	EPP
24		Signal Ground (Data 5)				
25		Signal Ground (Data 6)				
26		Signal Ground (Data 7)				
27		Signal Ground (Data 8)				
28		Signal Ground (PError, Select, nAck)				
29		Signal Ground (Busy, nFault)				
30		Signal Ground (nAutoFd, nSelectIn, nInit				
31	H	nInit	nInit	nInit	nReverseRequest	nInit
32	P	nFault	nDataAvail	nDataAvail	nPeriphRequest	User Defined 2
33		Not defined				
34		Not defined				
35		Not defined				
36	H	nSelectIn	IEEE 1284 Active	IEEE 1284 Active	IEEE 1284 Active	nAStrb
NOTE—Pins not defined by this standard are used by manufacturers at their own risk.						

[a]Data signals will be driven by some but not all peripheral devices.

8.2.2.3 IEEE 1284-C

Table 8 shows the pin numbers and their assigned signal names for the IEEE 1284-C connector. The IEEE 1284-C connector is a miniature 36-pin ribbon type connector. A physical description of the IEEE 1284-C connector can be found in Figure 26.

8.2.3 Cable interconnections

Interposer blocks and interconnecting cables shall be wired as shown in Figure 27 through Figure 31. Electrical characteristics of interconnecting cables shall meet the requirements of 8.3.1.

8.3 Electrical characteristics

The interface operates at 5 V logic levels. Interface voltage levels range between 0 V and 5 V (nominal), not to exceed a peak positive voltage of 5.5 V, nor to exceed a peak negative voltage of –0.5 V.

Each connected device shall have electrical characteristics as specified in the following subclauses. All characteristics specified apply to the total segment, i.e., host circuit card traces, host connector, cable, peripheral connector, and peripheral circuit card traces, unless otherwise specified.

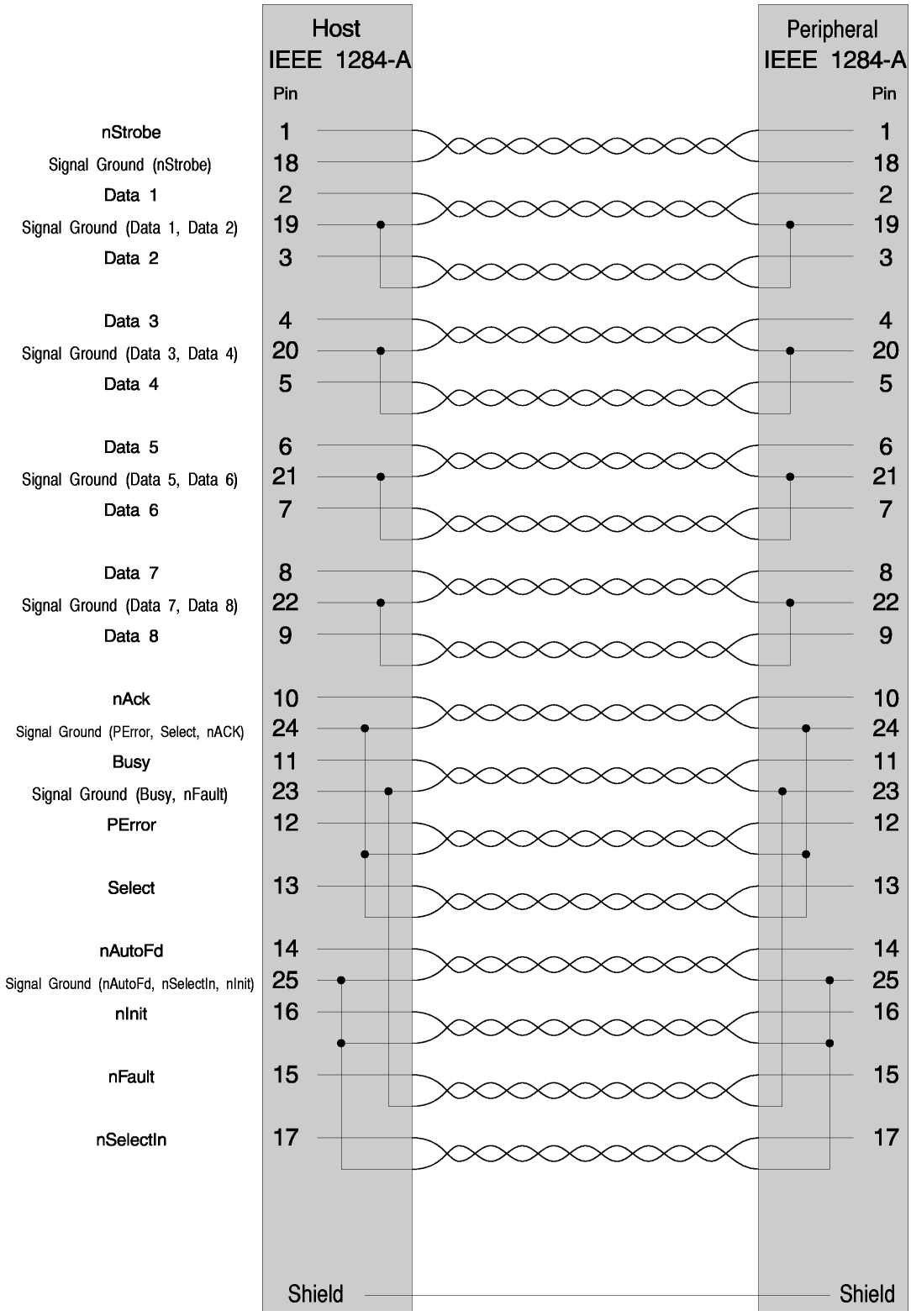


Figure 27—IEEE 1284-A (Host) to IEEE 1284-A (peripheral) wiring diagram

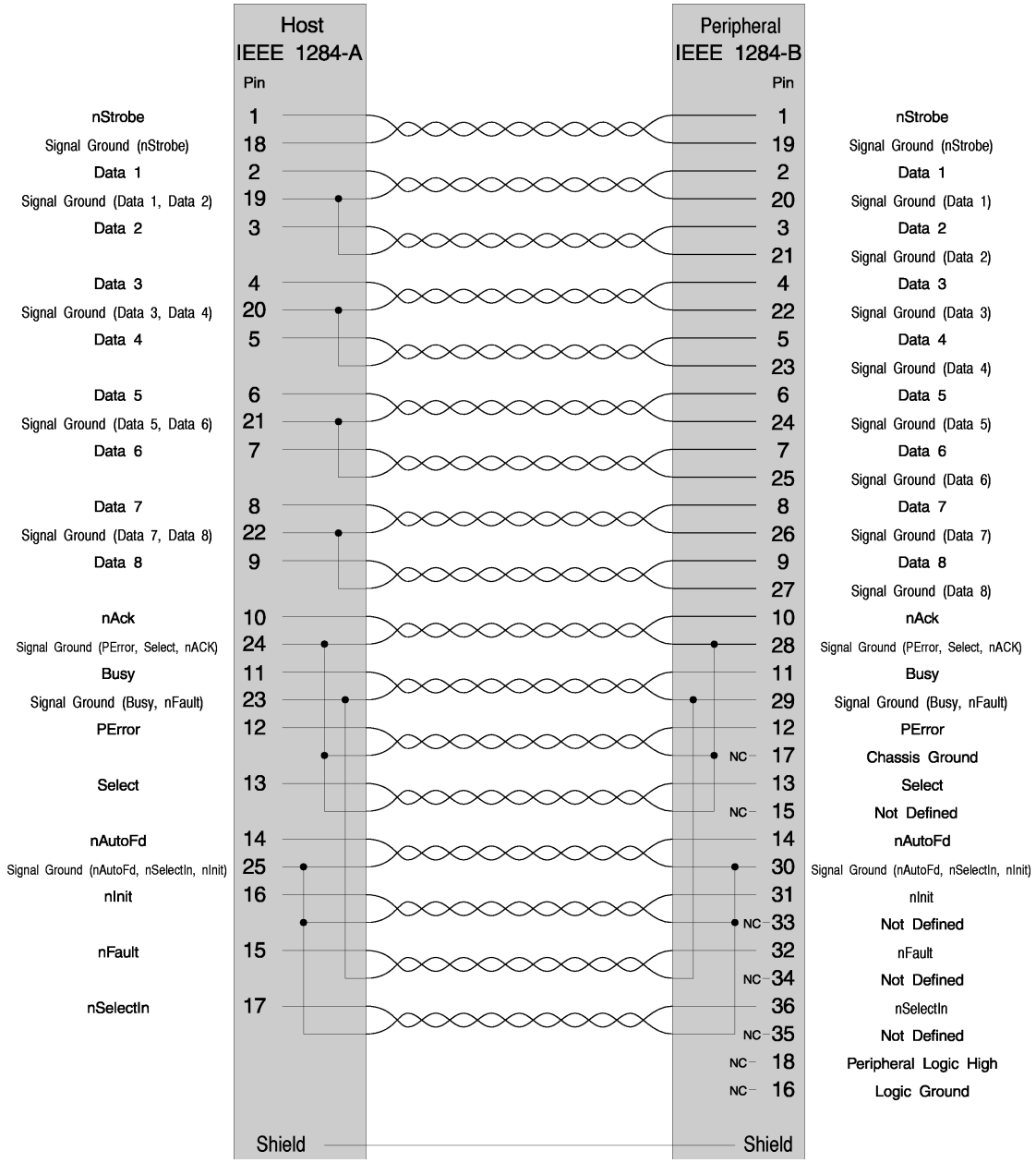


Figure 28—IEEE 1284-A (host) to IEEE 1284-B (peripheral) wiring diagram

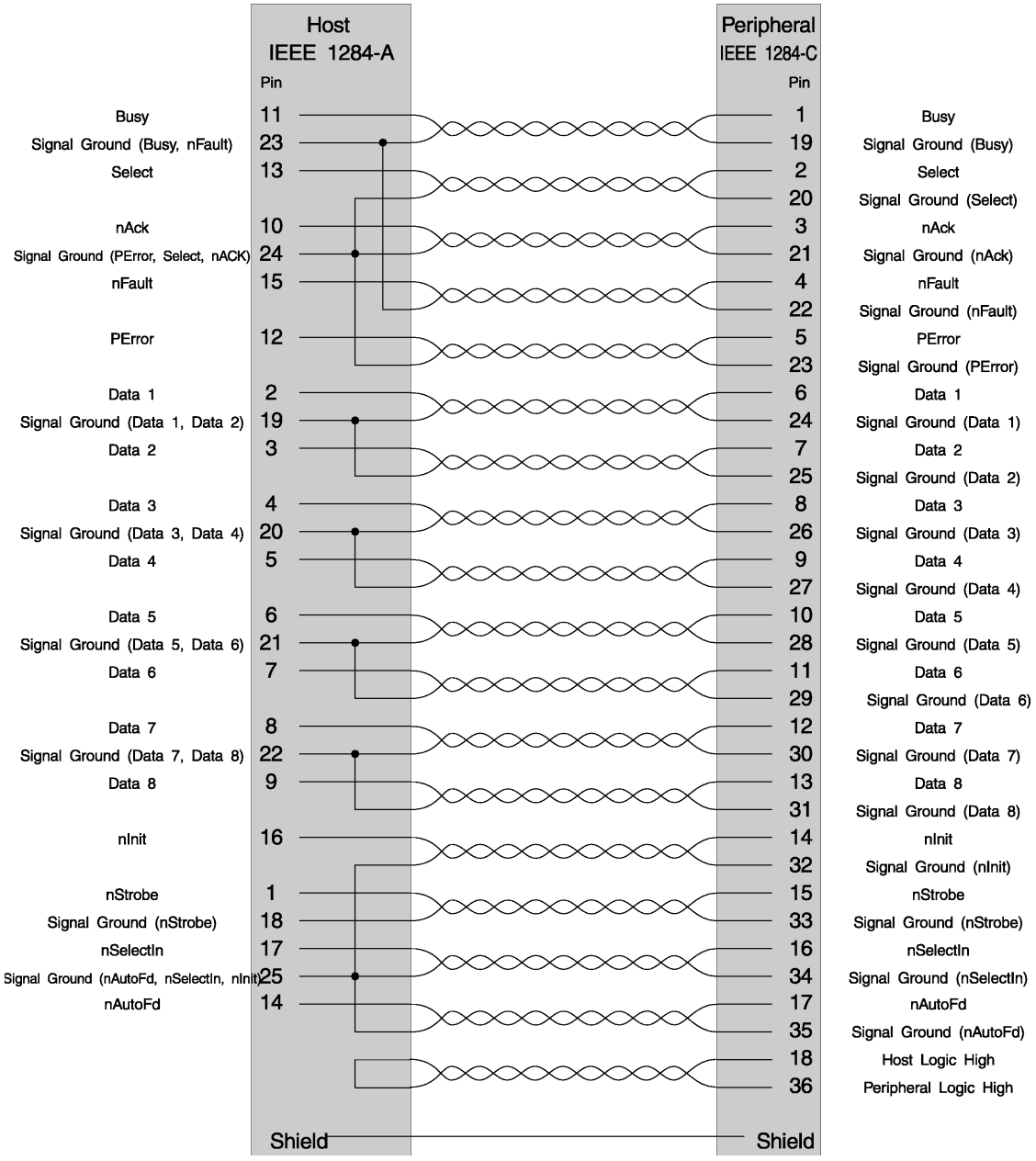


Figure 29—IEEE 1284-A (host) to IEEE 1284-C (peripheral) wiring diagram

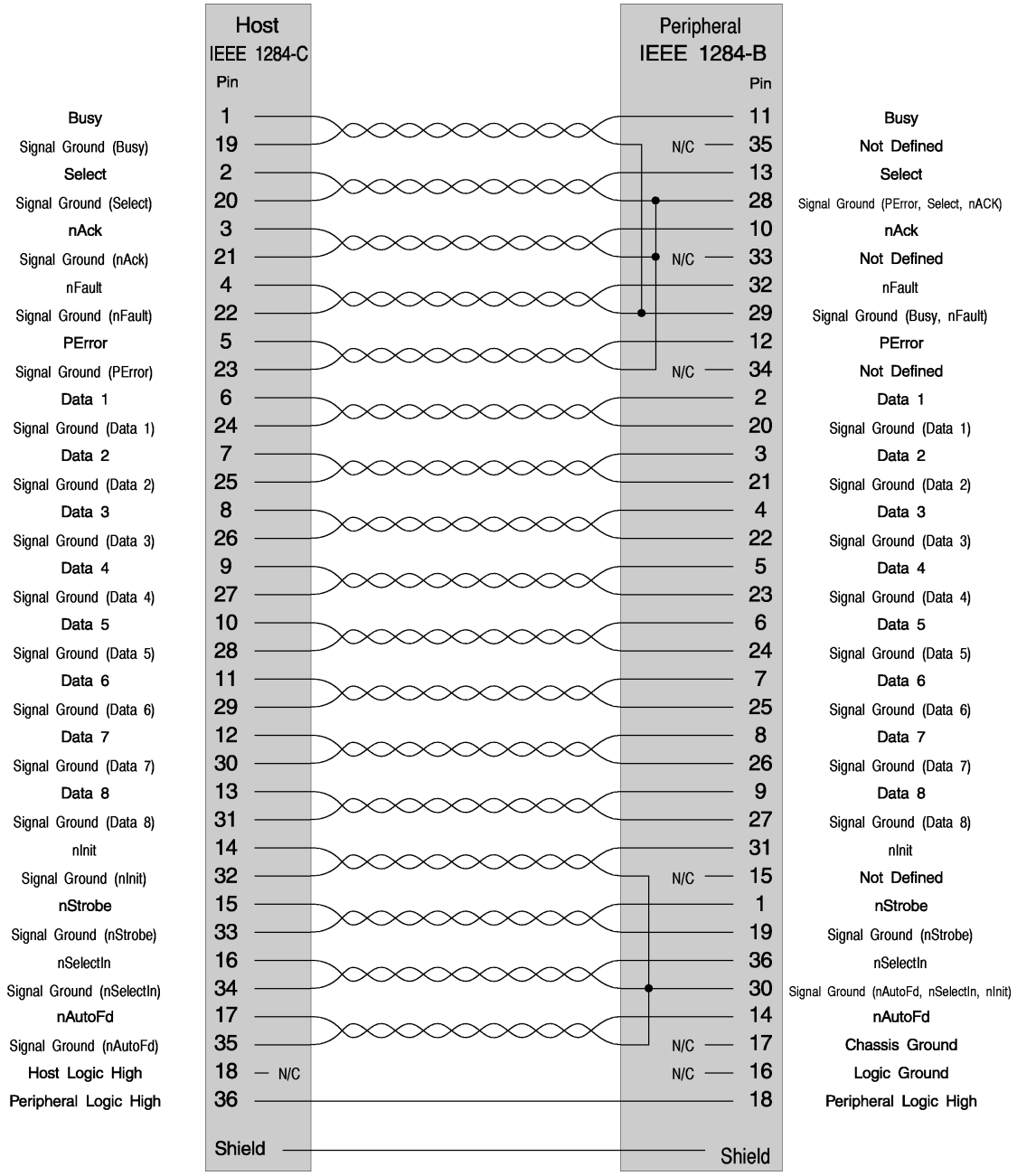


Figure 30—IEEE 1284-C (host) to IEEE 1284-B (peripheral) wiring diagram

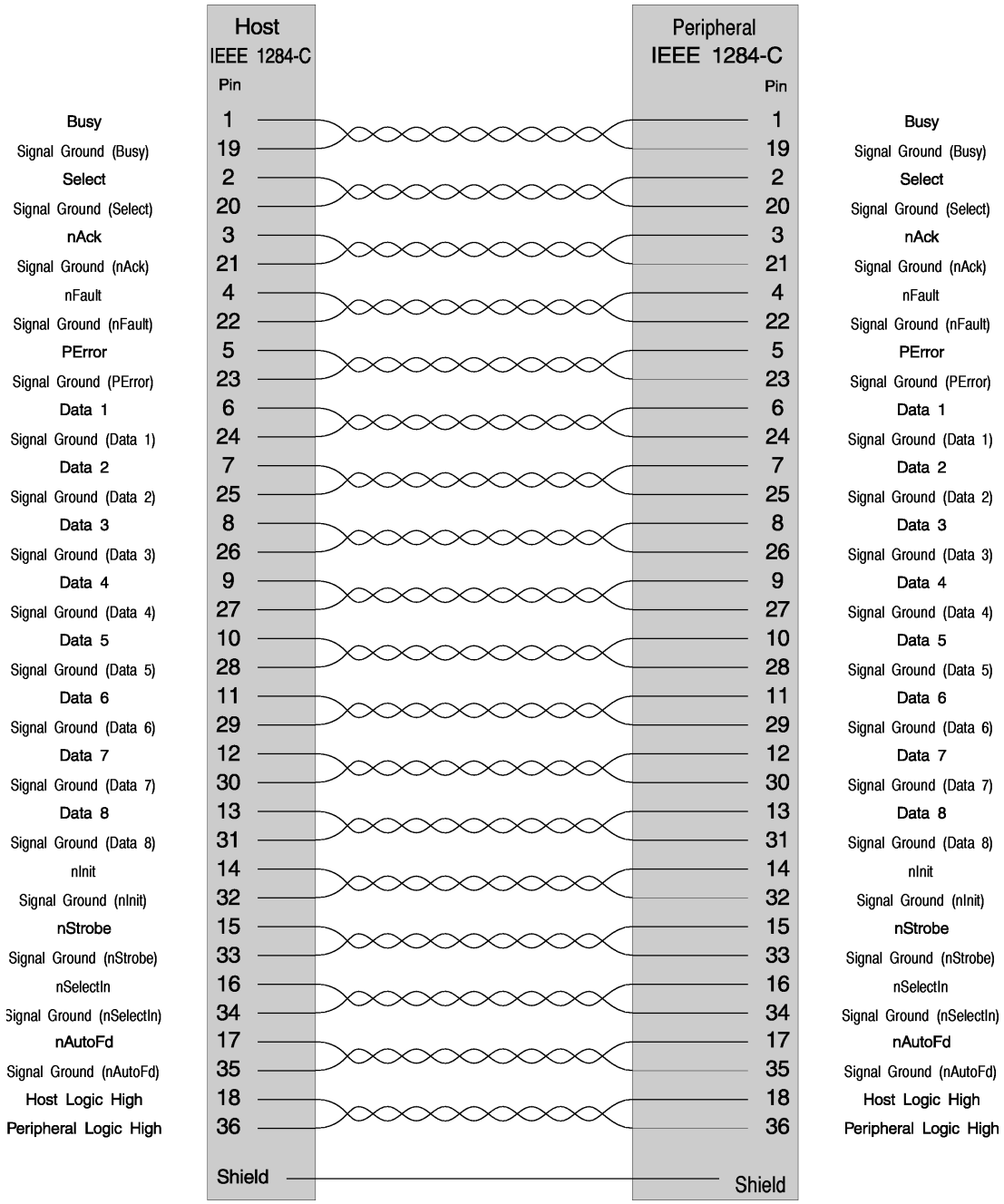


Figure 31—IEEE 1284-C (host) to IEEE 1284-C (peripheral) wiring diagram

Table 8—IEEE 1284-C connector pin assignments

Pin #	Source	Compatible	Nibble	Byte	ECP	EPP
1	P	Busy	PtrBusy	PtrBusy	PeriphAck	nWait
2	P	Select	Xflag	Xflag	Xflag	User defined 3
3	P	nAck	PtrClk	PtrClk	PeriphClk	Intr
4	P	nFault	NDataAvail	nDataAvail	nPeriphRequest	User defined 2
5	P	PError	AckDataReq	AckDataReq	nAckReverse	User defined 1
6	Bi-Di	Data 1 (Least Significant Bit)				AD1
7	Bi-Di	Data 2				AD2
8	Bi-Di	Data 3				AD3
9	Bi-Di	Data 4				AD4
10	Bi-Di	Data 5				AD5
11	Bi-Di	Data 6				AD6
12	Bi-Di	Data 7				AD7
13	Bi-Di	Data 8 (Most Significant Bit)				AD8
14	H	nInit	NInit	nInit	nReverseRequest	NInit
15	H	nStrobe	HostClk	HostClk	HostClk	NWrite
16	H	nSelectIn	IEEE 1284 Active	IEEE 1284 Active	IEEE 1284 Active	NAStrb
17	H	nAutoFd	HostBusy	HostBusy	HostAck	NDStrb
18	H	Host Logic High				
19		Signal Ground (Busy)				
20		Signal Ground (Select)				
21		Signal Ground(nAck)				
22		Signal Ground (nFault)				
23		Signal Ground (PError)				
24		Signal Ground (Data 1)				
25		Signal Ground (Data 2)				
26		Signal Ground (Data 3)				

Table 8—IEEE 1284-C connector pin assignments (continued)

Pin #	Source	Compatible	Nibble	Byte	ECP	EPP
27		Signal Ground (Data 4)				
28		Signal Ground (Data 5)				
29		Signal Ground (Data 6)				
30		Signal Ground (Data 7)				
31		Signal Ground (Data 8)				
32		Signal Ground (nInit)				
33		Signal Ground (nStrobe)				
34		Signal Ground (nSelectIn)				
35		Signal Ground (nAutoFd)				
36	P	Peripheral Logic High				

8.3.1 IEEE Std 1284-2000 compliant cable

The connecting cable shall consist of 18 pairs of signal wires and shall be double-shielded (both braid and foil).

The arrangement of a typical cable is shown in Figure 32.

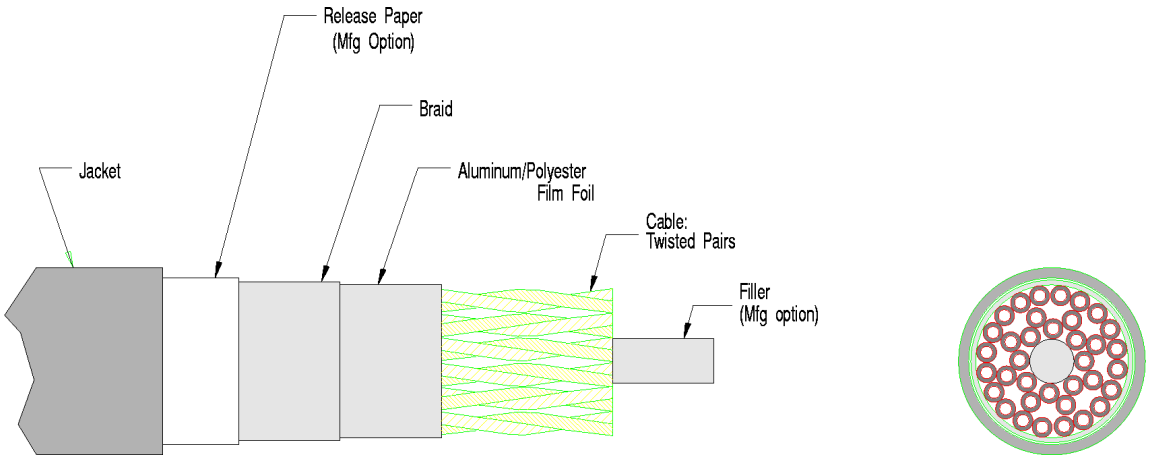


Figure 32—Physical cable

Cable assemblies shall meet the following characteristics:

a) The minimum conductor size shall be #28 AWG (≈0.080 mm^2)
b) The characteristic unbalanced impedance of each signal and ground pair shall be 62 Ω ± 6 Ω, over the frequency band of 4–16 MHz.
c) The unbalanced capacitance of each cable pair shall not exceed 107 pF/m at 1 MHz.
d) The maximum dc resistance of each cable wire shall be 0.232 Ω/m at 20 °C.
e) Maximum end-to-end attenuation shall not exceed 1.5 dB at 5 MHz.
f) The maximum propagation delay of the cable shall be 58 ns.
g) The maximum propagation delay difference between any two signal pairs shall be 2.5 ns.
h) The maximum zero-to-peak crosstalk noise (both near end and far end) shall be 10% as measured with a 5.0 ns rise/fall 2 MHz square wave, with each pair (source and victim) terminated in its characteristic impedance at both ends (see Figure 33).
i) The cable shall have a minimum of 85% optical braid coverage over the foil.
j) The cable shield shall be connected to the connector backshell using a 360° concentric method low impedence connection. A pigtail type connection is not acceptable.
k) Each signal pair shall be twisted at least 36 turns per meter.

Cable assemblies that meet these requirements shall be clearly and permanently labeled “IEEE Std 1284-2000 compliant” to distinguish them from cables having the same type of connectors but different electrical characteristics.

8.3.2 Level 1 device

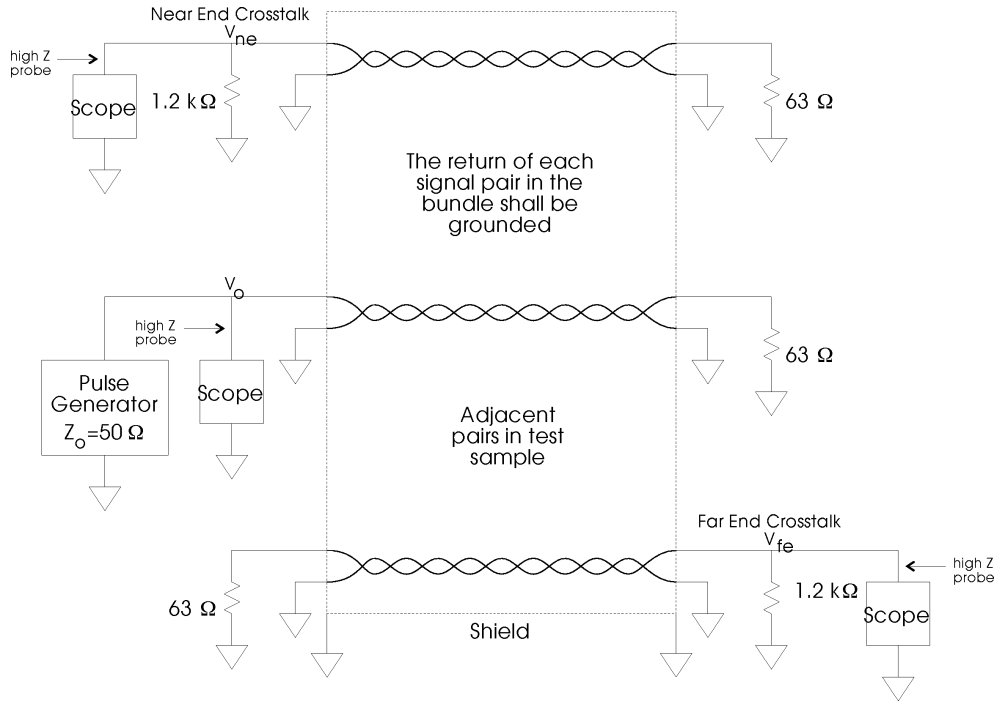
Level 1 devices drive the interface with asymmetric 5 V TTL circuits. Level 1 device requirements are designed to remain consistent with pre-existing installed devices. These devices are characterized by steady-state electrical specifications.

To assure operation with other Level 1 (or compatible) devices, the manufacturer shall provide sufficient voltage and timing margins to assure operation with at least 2 m of IEEE Std 1284-2000-compliant cable (unless the device is designed and intended to attach directly) when connected to a device that meets minimum Level 1 requirements.

Level 1 compliant devices shall meet all the electrical and timing requirements specified in the following subclauses.

8.3.2.1 Driver requirements

a) Drivers shall operate at nominal 5 V TTL levels.
b) The open circuit high-level output voltage shall not exceed 5.5 V.
c) The open circuit low-level output voltage shall be no less than –0.5 V.
d) The high-level output voltage shall be at least 2.4 V at a source current of 0.32 mA.
e) The low-level output voltage shall not exceed 0.4 V at a sink current of 14 mA.
f) Pull-up resistors, if present, shall be not less than 1.8 kΩ ± 5% for control and status signals, and not less than 1.0 kΩ ± 5% for data signals.
g) Circuit and stray capacitance shall not exceed 50 pF of uncompensated capacitance. Additional capacitance may be compensated for by providing additional source and sink currents within the driver circuitry.



Signal Generator Duty Cycle

V_O	is the output voltage (0 V to 5 V into cable)
T_R	is the rise time of the signal (5 ns)
T_F	is the fall time of the signal (5 ns)
V_{ne}	is the near end voltage, maximum voltage (zero to peak)
V_{fe}	is the far end voltage, maximum voltage (zero to peak)
Crosstalk NE	is the near end crosstalk $\lvert V_{NE}(MAX) / V_O \rvert$
CrosstalkFE	is the far end crosstalk, $\lvert V_{FE}(MAX) / V_O \rvert$
z_o	is the pulse of the output generator (50 Ω)

The frequency of the output pulse is 2 Mhz.

Figure 33—Crosstalk measurement technique

8.3.2.2 Receiver requirements

a) Receivers shall operate at nominal 5 V TTL levels.

b) The receiver high-level input threshold shall not exceed 2.0 V.

c) The receiver low-level input threshold shall be at least 0.8 V.

d) The receiver high-level input sink current shall not exceed 0.32 mA at 2.0 V.

e) The receiver low-level input source current shall not exceed 12 mA at 0.8 V.

f) Pull-up resistors, if present, shall be not less than 470 Ω ± 5% for control and status signals, and not less than 1000 Ω ± 5% for data signals. It is recommended that these minimum values be used. Current supplied by pull-up resistors, if present, shall be included in the input current limits, list items d) and e).

g) Circuit and stray capacitance shall not exceed 50 pF of uncompensated capacitance. Compensation for larger capacitance may be supplied by a pull-up resistor, provided that it meets the requirements of list item f).

h) The maximum rise or fall time is as expected for a 2 m cable, 120 ns as measured between the 0.8 V and 2.0 V levels.

i) The receiver should be able to withstand peak input voltage transients between –2.0 V and 7.0 V without damage or improper operation.

8.3.3 Level 2 device

Level 2 devices drive the interface with 5 V circuits with ac symmetric impedances. However, receiver inputs operate with TTL-compatible logic level thresholds.

Level 2 devices capitalize on the transmission line characteristics of the connecting cable. Driver output impedance is matched to the characteristic impedance of the cable, while receiver inputs terminate the cable with a deliberate high-impedance mismatch. When the driver changes state, this produces an incident voltage wave in the cable, with a magnitude of half the applied voltage. When this voltage wave reaches the high-impedance receiver, the mismatch produces a reflection (which essentially doubles the voltage) so that the initial applied voltage appears across the receiver.

The reflected wave from the impedance mismatch then propagates back along the cable. When this voltage wave reaches the driver, its output voltage completes the transition to the final steady-state voltage level, where it remains until the next transition.

All input signals provide pullup resistors to 5 V to ensure operation with Level 1 and compatible devices. The specified value (1.2 kΩ ± 5%) is a compromise. This value provides sufficient current to meet Level 1 timing, while providing 90% reflection of the incident voltage at the receiver.

NOTE—IEEE-1284.3-2000 [B8] requires the use of 10 kΩ pullup resistors on the data lines (Data1–Data8). Some legacy host parallel port implementations which have open collector/drain output may experience data corruption problems due to effective skew between nStrobe and the data lines caused by differences in these pullup values.

Level 2 compliant devices shall meet all the electrical and timing requirements specified in the following subclauses.

8.3.3.1 Driver requirements

a) The open circuit high-level output voltage shall not exceed 5.5 V.

b) The open circuit low-level output voltage shall be no less than –0.5 V.

c) The dc steady-state, high-level output voltage shall be at least 2.4 V at a source current of 14 mA.

d) The dc steady-state, low-level output voltage shall not exceed 0.4 V at a sink current of 14 mA.

e) The driver output impedance (R_O in figure 34) shall be 45–55 Ω, at one-half the actual driver V_{OH} minus V_{OL} voltage, (V_{OH} and V_{OL} are the actual measured voltages of the output device). This causes the driver to generate an incident wave slightly in excess of one-half V_{OH} minus V_{OL} amplitude into an infinitely long 62 Ω transmission line, as measured at the driver/cable interface, for transitions in either direction.

f) The driver slew rate shall be 0.05–0.40 V/ns.

The values for items e) and f) are measured at the connector pin under load.

8.3.3.2 Receiver requirements

a) The receiver shall withstand peak input voltage transients between –2.0 V and 7.0 V without damage or improper operation.

b) The receiver high-level input threshold shall not exceed 2.0 V.

c) The receiver low-level input threshold shall be at least 0.8 V.

d) The receiver shall provide at least 0.2 V input hysteresis. Increasing this hysteresis (up to a maximum of 1.2 V) will improve noise immunity.

e) The receiver high-level input sink current shall not exceed 20 μA at 2.0 V.

f) The receiver low-level input source current shall not exceed 20 μA at 0.8 V.

g) Circuit and stray capacitance shall not exceed 50 pF.

8.3.4 Level 2 example

Simplified examples of Level 2 drivers, receivers, and transceivers are shown in Figure 34 and Figure 36.

The intent of the termination is to match the characteristic impedance at the source and have a high impedance at the load. This will cause half the source voltage to be injected into the cable and a doubling of voltage at the load. Because of component tolerance, the matching source impedance is slightly lower than the nominal characteristic impedance of the cable to guarantee proper voltage levels at the load.

Referring to Figure 34, R_D is output impedance of the circuitry of the active driver. R_S is a series resistor used for impedance matching. R_O represents the combined output impedance of the driver and impedance matching resistor (the sum of R_D and R_S). R_O is less than the cable impedance (Z_O) as specified by item e) of 8.3.3.1.

Figure 35 combines the driver and receiver circuitry to form a transceiver. The addition of the 1.2 kΩ resistor does not significantly affect the output driver impedance (R_O). R_I is the input driver impedance.

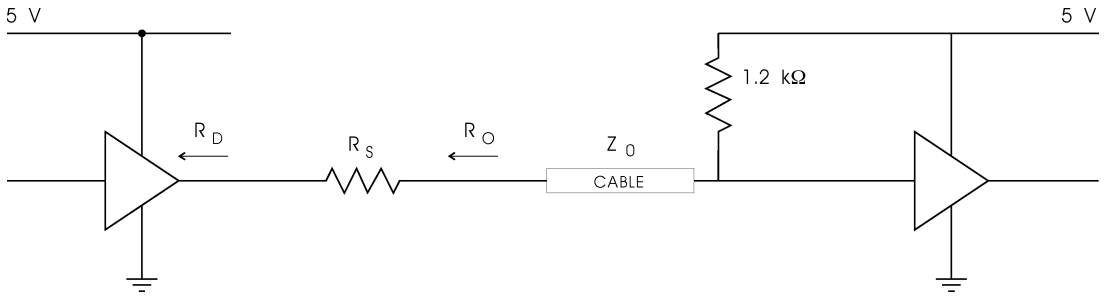


Figure 34—Level 2 driver/receiver example

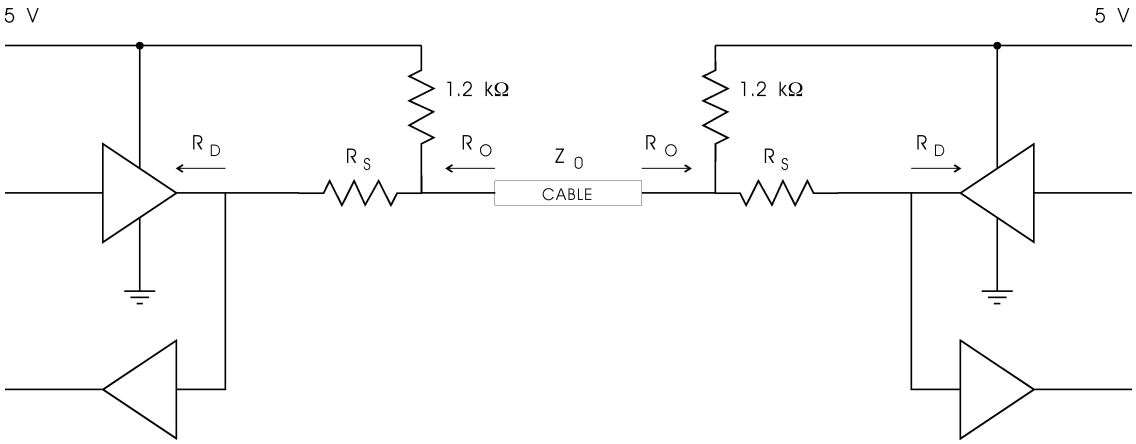


Figure 35—Level 2 transceiver example

8.3.5 Power-off state of the interface

The Host Logic High and Peripheral Logic High signals may be used by the peripheral and host respectively to detect a powered-off or cable-disconnected state. To ensure that this can be detected, both the IEEE 1284 host and IEEE 1284 peripheral shall provide circuitry to guarantee a low logic level (2.0 V or less) on these signals when they are powered off. The receiver shall provide an impedance equivalent to 7.5 kΩ to ground.

On power-up sequences, an IEEE 1284 device shall guarantee that the signals that it sources be in a valid state within 500 ms of its respective logic high signal (Host Logic High or Peripheral Logic High) equal or exceeding 3.0 V. See Figure 36 for a typical power-off detection circuit.

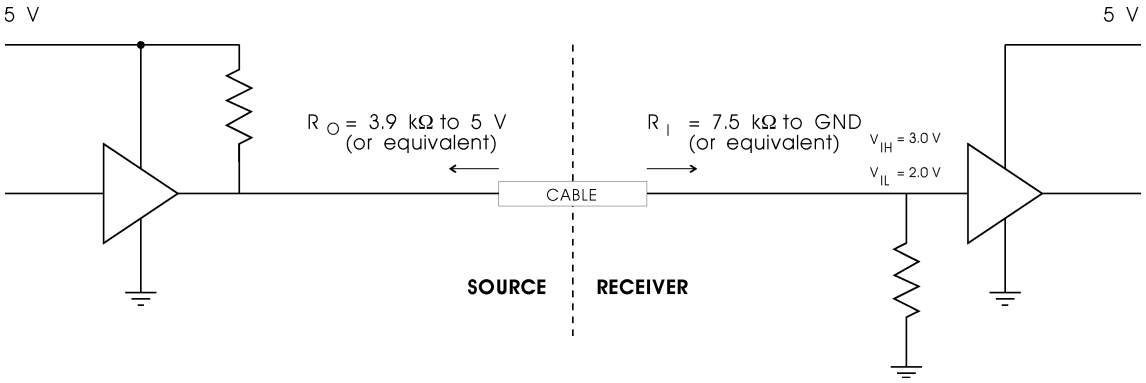


Figure 36—Typical power-off detection circuit

8.3.6 Environmental specifications

8.3.6.1 General safety

All equipment meeting this standard shall conform to IEC 60950 (1999-04).[5]

8.3.6.2 Cabling safety

NOTE—This subclause sets forth a number of recommendations and guidelines related to safety concerns; the list is neither complete nor does it address all possible safety issues. The designer is urged to consult the relevant local, national, and international safety regulations to ensure compliance with the appropriate requirements.

Cable systems described in this subclause are subject to at least four direct electrical safety hazards during their installation and use. These hazards are as follows:

a) Direct contact between cabling components and power, lighting, or communication circuits.
b) Static charge buildup on cables, components, and users.
c) High-energy transients coupled onto the cable system.
d) Voltage potential differences between grounds to which various components are connected.

Such electrical safety hazards shall be avoided or appropriately protected against for proper cable installation and performance. In addition to provisions for proper handling of these conditions in an operational system, special measures shall be taken to ensure that the intended safety features are not negated during

[5]Information on references can be found in Clause 2.

installation of the new cable system or during modification or maintenance of the existing cable system. Isolation requirements are not a part of this standard.

8.3.6.3 Grounding

The cable shield should be grounded to the chassis of both the attached peripheral and the host computer.

8.3.6.4 Electromagnetic emission and electromagnetic susceptibility (EMC)

Specific requirements for applicable local and national codes are considered beyond the scope of this standard.

8.3.6.5 Temperature and humidity

The interface is expected to operate over a reasonable range of environmental conditions related to the temperature, humidity, and physical handling (such as shock and vibration). Specific requirements and values for these parameters are considered beyond the scope of this standard. It is recommended that manufacturers indicate in product specifications of the devices the distance and operating environmental conditions over which the aforementioned specifications will be met.

9. Software support

9.1 General considerations

There are several conceptual layers involved in communication with a peripheral. The software layers affected by IEEE 1284 are

a) The software that manipulates the interface hardware to transfer information.
b) The software that prepares the information to send, or interprets the information received.

Depending on the host operating environment of interest, the name “device driver” may encompass either or both of the these components. For purposes of this discussion, “driver” will refer to item a) (the software that manipulates the hardware) and “application” will refer to item b) (the software that prepares or interprets information).

Peripheral data is available only when the host places the interface in a mode capable of peripheral-to-host data transfer. This requires the host periodically to place the interface in one of these modes to retrieve any data that might be available. This polling might be under the applications control, or the operating system could start a timer task that would periodically check the peripheral for available data. When data is available, the system could either receive the message and hold it for an application or notify an application to retrieve the data.

Polling can be reduced or eliminated by leaving the interface in an IEEE 1284 idle phase (reverse idle phase for the Nibble and Byte Modes) whenever possible. In one of the IEEE 1284 idle phases, the driver may eliminate polling by enabling an interrupt (if supported by the host hardware) to occur when the peripheral has data to report. If an interrupt cannot be used, then the complexity of polling can be reduced from an IEEE 1284 negotiation to a simple port read, if the interface is in an IEEE 1284 idle phase.

9.2 Application level compatibility

The IEEE 1284 interface requires the operating system vendor to provide a new driver that can access the status read-back features of the new interface. These new features make more interactive information available to applications. However, application software support of the advanced functionality will not be avail-able until new versions of the applications are released. Therefore, to ensure that current applications operate correctly with the IEEE 1284 interface, it is important that the IEEE 1284 driver be backward compatible with drivers available at the current time.

9.3 MS-DOS IEEE 1284 driver

The new driver can be designed to replace or supplement the current BIOS driver or can use the MS-DOS driver interface. Since all BIOS accesses to the parallel port are typically byte oriented, and since MS-DOS typically uses higher efficiency block transfers, the MS-DOS interface is preferred. Both interfaces should be implemented to remain consistent with current BIOS and MS-DOS device usage. Using the typical BIOS API, data is sent to a peripheral a byte at a time, and status will be read back a byte at a time. Using the MS-DOS API, data could be sent or read in blocks of several bytes (or even several kilobytes) for greater throughput. BIOS level functions could also be implemented to transfer blocks of data.

9.4 Windows 1284 driver

In the Windows environment, the COMM Driver can be replaced by a new driver that allows reading of LPT ports in addition to the current functionality of writing to LPT ports, and reading and writing COM ports. As a multitasking, message-based environment, Windows stands to benefit more than other environments (such as MS-DOS) by using the IEEE 1284 idle phase. In addition, there is no “byte” oriented transfer to support, hence the block features of IEEE Std 1284-2000 can be fully utilized to achieve the greatest throughput.

9.5 Reverse channel data

One of the objectives of the IEEE 1284 interface is to provide a separation between the physical link that is sending data and the data itself. The definition of the information that will be transferred to the host will be peripheral specific; therefore, the driver shall transfer the data in both directions without attempting to interpret the information. The driver shall assume that binary data is being transferred and shall transfer each byte to or from the application without modification.

9.6 Link performance

The “burst” rate of transferring data through the link will depend on the speed of the host computer, the driver implementation, and the peripheral being used. Overall software performance and link throughput will depend upon the application-to-driver interface and the polling rate, as well as the burst rate of transferring a status message once the host begins the transfer

Annex A

(normative)

Timing specifications

Output signal timing is specified at the V_{OH} and V_{OL} levels, as measured at the source device connector. Input signal timing is specified at the V_{IH} and V_{IL} levels, as measured at the receiving device connector.

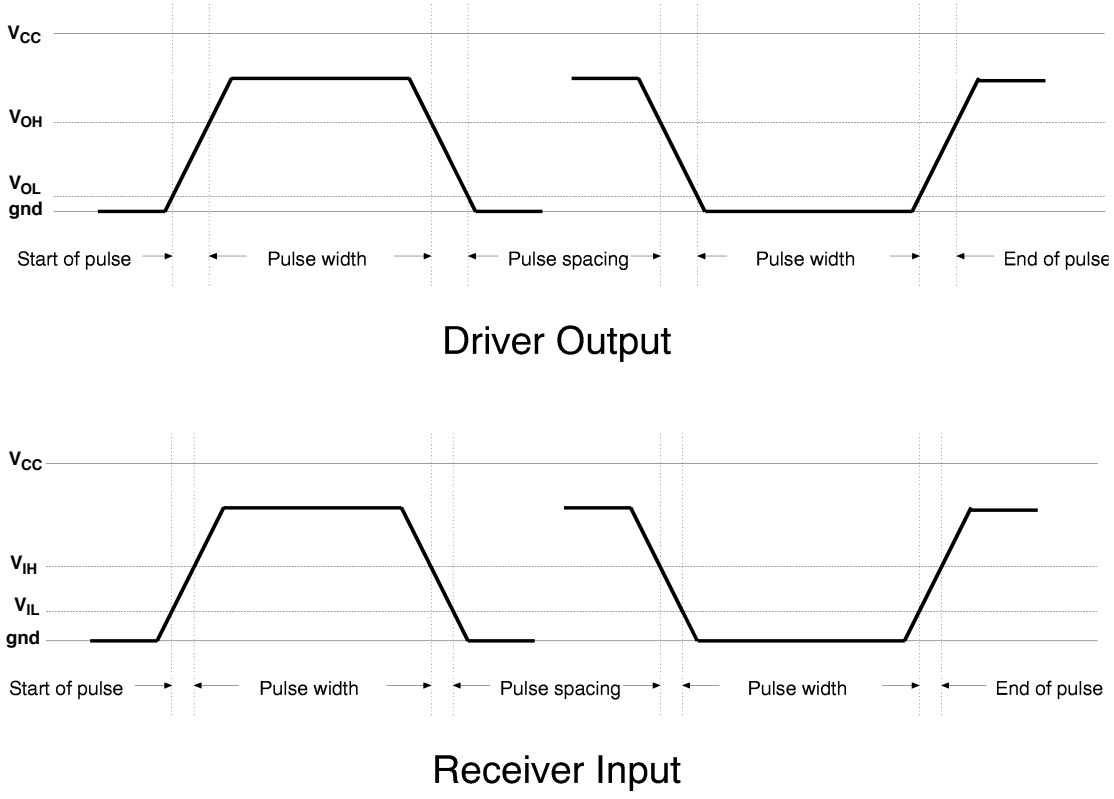


Figure A.1—Typical timing waveforms

Timing measurements are made with all device outputs terminated as shown in Figure A.2 and all device inputs terminated as shown in Figure A.3.

Each timing specification includes a compliance requirement, which indicates whether this specification applies to the host or the peripheral and whether the specification applies to all compatible devices, or just to IEEE Std 1284-2000 compliant devices.

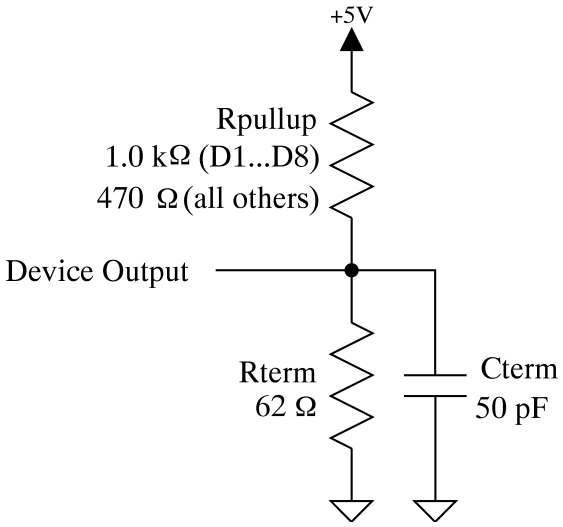


Figure A.2—Driver test termination ac equivalent circuit

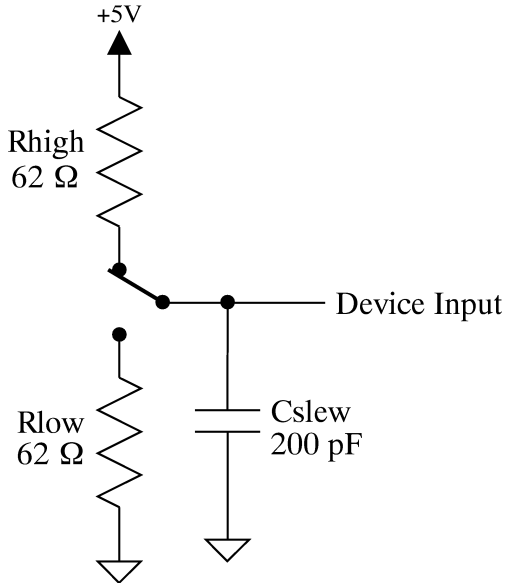


Figure A.3—Receiver test termination ac equivalent circuit

Annex B

(normative)

Signal transition events

Each signal transition diagram has numbers corresponding to the events that cause the transitions. The following is a list of those events. More details on these events can be found in Clause 7.

0) Host sets extensibility request value on data bus: 00h for Nibble Mode, 01h for Byte Mode, etc.
1) Host requests an IEEE 1284 mode transfer by setting IEEE 1284 Active(nSelectIn) high and HostBusy(nAutoFd) low.
2) Peripheral indicates IEEE 1284 support by setting AckDataReq(PError), Xflag(Select), and nDataAvail(nFault) high, and PtrClk(nAck) low.
3) Host sets HostClk(nStrobe) low to latch extensibility request value into peripheral.
4) After waiting the minimum HostClk(nStrobe) pulse width, host sets HostClk(nStrobe) and HostBusy(nAutoFd) high to acknowledge the support by the peripheral of IEEE 1284 protocol.
5) Xflag(Select) is set to reflect the support by the peripheral of the requested extension. Busy is set to indicate whether the peripheral can accept data from the host in the Compatibility Mode. Detailed descriptions of other signals are described in their respective clauses.
6) Peripheral sets PtrClk(nAck) high, informing the host that the four interface status signals are valid.
7) Host sets HostBusy(nAutoFd) low to indicate that it can accept data.
8) Peripheral places a nibble on the four status lines (low order nibble for first handshake, high order nibble for second handshake).
9) Peripheral sets PtrClk(nAck) low to indicate that data is valid.
10) Host sets HostBusy(nAutoFd) high to handshake with the peripheral and (in Nibble Mode) to indicate that it has received data and cannot accept more at the moment.
11) Peripheral acknowledges receipt of data by the host by setting PtrClk(nAck) high.
12) Host sets HostBusy(nAutoFd) low, indicating that it can accept the second nibble.
13) Peripheral sets status lines to indicate if a subsequent peripheral to host byte is ready to transfer, and sets PtrBusy to indicate its current forward channel status.
14) In Byte Mode, host places data bus in a high impedance state in anticipation of peripheral-to-host transfer. In the Nibble Mode, the data bus is undefined.
15) In Byte Mode, peripheral places bytes on data bus.
16) Host sets HostClk(nStrobe) low. This event may happen after the falling or rising edge of PtrClk(nAck), i.e., events 9 and 11. It is defined in the diagrams to occur coincident with event 10. The peripheral shall **not** interpret this event as a latch signal for forward channel data.
17) Host sets HostClk(nStrobe) high. It shall occur before or coincident with setting HostBusy(nAutoFd) low (event 7).
18) Peripheral sets nDataAvail(nFault) low to indicate data is available, and sets PtrClk(nAck) low.
19) After waiting the minimum pulse width, the peripheral sets PtrClk(nAck) high to generate an interrupt to the host.
20) Host sets HostBusy(nAutoFd) high to acknowledge the request of the peripheral.
21) Peripheral sets AckDataReq(PError) low to acknowledge the host.

22) Host sets IEEE 1284 Active(nSelectIn) low and HostBusy(nAutoFd) high (if not already) to request termination of IEEE 1284 mode.

23) Peripheral places data bus in a high impedance state (if in Byte Mode) and sets Busy and nFault high.

24) Peripheral acknowledges the request of the host by setting PtrClk(nAck) low and Xflag(Select) to its opposite sense.

25) Host handshakes with the peripheral by setting HostBusy(nAutoFd) low.

26) Peripheral sets AckDataReq(PError), nDataAvail(nFault), Xflag(Select) and nReverseRequest(nInit) in ECP Mode to their current Compatibility Mode values. Busy is still high to block incoming data.

27) Peripheral completes handshake by setting PtrClk(nAck) high.

28) Host completes handshake by setting HostBusy(nAutoFd) high.

29) Peripheral updates Busy to current Compatibility Mode status.

30) Host sets HostAck (nAutoFd) low to acknowledge successful negotiation.

31) Peripheral acknowledges that it is now operating in ECP Mode by raising nAckReverse (PError). PeriphAck (Busy) and nPeriphRequest (nFault) are now active.

32) Peripheral drives PeriphAck (Busy) low, indicating that it is ready to accept data.

33) Host is idle.

34) Host places Data on the bus. The command bit (nCmd/HostAck) is driven to the appropriate level.

35) Host sets HostClk (nStrobe) low to indicate valid data is on the bus.

36) Peripheral handshakes, setting PeriphAck (Busy) high.

37) Host raises nStrobe to continue the handshake. The peripheral is expected to use this edge of nStrobe to latch the data.

38) Host places the data bus in a high impedance state and sets HostAck (nAutoFd) low to prepare for a bus reversal.

39) Host sets nReverseRequest (nInit) low to initiate a bus reversal.

40) Peripheral sets nAckReverse (PError) low to acknowledge the bus reversal. HostAck (nAutoFd) is now active.

41) Peripheral is idle.

42) Peripheral drives Data and nCmd (Busy) onto the bus.

43) Peripheral sets PeriphClk (nAck) low to indicate that data is available on the bus.

44) Host acknowledges the assertion of PeriphClk (nAck) by setting HostAck (nAutoFd) high.

45) Peripheral continues the handshake by setting PeriphClk (nAck) high. The host is expected to use this edge of PeriphClk to latch the data.

46) Host completes the transfer, accepting the byte by setting HostAck (nAutoFd) low.

47) Host sets nReverseRequest (nInit) high to initiate a bus reversal (back to the forward direction). The host may continue to handshake, receiving don't care data.

48) Peripheral terminates any ongoing transfer, places the data bus in a high impedance state, sets PeriphClk (nAck) high, and places valid status on the PeriphAck (Busy) and nPeriphRequest (nFault) lines.

49) Peripheral acknowledges that the bus has been relinquished by setting nAckReverse (PError) high.

50) Host places the additional extensibility byte on the data bus.

51) Host sets HostClk(nStrobe) low after waiting the data setup time to tell the peripheral that the additional extensibility byte should be read.

52) Peripheral sets PtrClk(nAck) low to tell the host that it has captured the byte.

53) Host sets HostClk(nStrobe) high.

54) Peripheral sets Xflag(Select) to reflect support of the requested extension by the peripheral. PtrBusy(Busy) is updated to indicate whether or not the peripheral can accept data from the host in Compatibility Mode.

55) Peripheral, after a data setup time, sets PtrClk(nAck) high, informing the host that the four interface signals are valid.

56) Host sets nAStrb(nSelectIn) low to indicate that an address cycle is taking place. During write cycles, the host will also assert nWrite(nStrobe) low. At this time, the host also places an address byte on the parallel port data lines.

57) Peripheral holds the state of the AckDataReq(PError), Intr(nAck), nDataAvail(nFault), and XFlag(Select) until the first EPP cycle is initiated by either the nAStrb(nSelectIn) or nDStrb(nAutoFd) going low. At this time, these signals are peripheral-to-host status bits, except for Intr(nAck), which is used to generate interrupts to the host.

58) Peripheral sets nWait(Busy) high to indicate that it has accepted the data sent during a write cycle and to indicate that it has placed a valid byte of data on the data lines during a read cycle.

59) Host sets nAStrb(nSelectIn) high to indicate that the cycle is complete.

60) Peripheral sets nWait(Busy) low to indicate that it is ready for the next EPP transfer.

61) In response to the peripheral de-asserting the nWait(Busy) signal, the host sets nWrite(nStrobe) high and places the data lines in a high impedance state.

62) Host sets nDStrb(nAutoFd) low to indicate that a data cycle is taking place. During write cycles, the host will also assert nWrite(nStrobe) low. At this time, the host also places a byte of data on the parallel port data lines.

63) Host sets nDStrb(nAutoFd) high to indicate that the cycle is complete.

64) Host sets nAStrb(nSelectIn) low to indicate that an address cycle is taking place. In addition, the host does not assert nWrite(nStrobe), which implies a read operation.

65) Data valid from peripheral.

66) Data lines from peripheral are placed in a high impedance state.

67) Host sets nDStrb(nAutoFd) low to indicate that a data cycle is taking place. In addition, the host does not assert nWrite, which implies a read operation.

68) Host sets nInit low in order to terminate EPP Mode and return to Compatibility Mode.

69) Host sets nInit high.

70) Peripheral asserts the Intr(nAck), indicating an interrupt from the peripheral to the host.

71) Peripheral de-asserts the Intr(nAck), completing the interrupt phase.

72) After waiting for the minimum required time (TS), the host may abort the host-to-peripheral data transfer in progress by setting nReverseRequest (nInit) low.

73) Peripheral handshakes, setting nAckReverse (PError) low and (if not already set) PeriphAck (Busy) low, indicating that the peripheral-to-host data transfer in progress has been aborted and the data byte has been discarded.

74) Host raises nReverseRequest (nInit) and HostClk (nStrobe) to continue the handshake.

75) Peripheral completes the handshake by raising nAckReverse (PError) high, returning the link to a host idle condition.

Annex C

(informative)

Centronics and PC-compatible parallel interfaces

C.1 Overview

This annex provides a description of the existing Centronics-compatible and PC-compatible parallel printer interfaces in widespread use throughout the industry. This annex is not a part of the IEEE Std 1284-2000, but is included for information only. It serves to provide a formal definition of Compatibility Mode devices, as well as providing a historical perspective of the development of this de facto standard interface. This annex should not be used for new designs. Designing to IEEE Std 1284-2000 will provide compatibility with these older implementations.

This attachment provides a baseline specification of interface characteristics and operations for both hosts and printers. The IEEE 1284 interface is designed to be interoperable with many devices that meet these specifications.

This annex is divided into the following clauses:

— Centronics interface background information
— Classic Centronics Standard Parallel Interface
— PC parallel and PC-compatible printer interfaces
— Enhanced and bidirectional parallel printer interfaces
— Printer interface variations

C.2 Centronics interface background information

The “Centronics-compatible” printer interface is a unidirectional 8 bit parallel host-to-printer connection. This interface was introduced by Centronics Data Computer Corporation in the mid-1960s for a series of small serial-impact printers. This series of printers was very successful in the market, and the “Centronics-compatible” interface has become the primary interface used between most small computers and associated printers.

Although widely used, there is no existing industry standard specification for this interface. When a device claims it supports a “Centronics-compatible” interface, this simply means the interface is backward-compatible with Centronics printers introduced roughly 30 years ago. The core set of signals[6] in a Centronics-compatible interface typically include

— Eight data lines (host to printer)
— Two control lines (host to printer)
— Five status lines (printer to host)

[6]The original Centronics interface often contained several additional signals, usually differing by printer model. Also, some manufacturers have added other signals to the interface. However, this list includes all the basic signals needed to realize the interface.

PC-compatible computers provided a basic Centronics-compatible printer interface, with a few additional control signals. Some major PC manufacturers added bidirectional driver circuits to the hardware. However, this capability has remained unused due to the absence of standard data link and data transfer protocols.

Data transfer rates have also remained limited. With absolute minimum Centronics setup, hold, and pulse width times, the theoretical transfer rate limit of the printer is 100 000 bytes/s. However, when connected to host systems, response time latency and software overhead often greatly reduce this figure. For example, the typical PC transfer rate is only 10 000 bytes/s.

Printer control and status provided by the classic Centronics interface is primitive at best. While this was adequate for simple, “dumb” text printers of previous decades, it is increasingly inadequate for sophisticated modern printers and multi-user networked environments. Today, printed pages consist of mixed text and graphics, or even pure graphics, which may require over a million bytes of data per page. Limited data transfer rates are often a bottleneck, and users complain bitterly when their pages-per-minute printer prints at minutes-per-page speed. These shortfalls are directly addressed by this standard.

C.3 Classic Centronics Standard Parallel Interface

This clause provides core specifications for the classic Centronics Standard Parallel Interface.[7] This includes interface connector and pin assignments, signal definitions and characteristics, and electrical specifications. Paralleling the historical development of the interface, this description is focused at the printer end.

The Centronics standard defined the interface signals only at the printer. Host implementations varied widely (often uniquely) from host system to host system in both supported signals and interface connectors, although 37-pin D-shell (DB-37) connectors were often used on host computers. A predominant host implementation did not emerge until after 1981, with the introduction of the DOS-compatible PC. (The PC implementation is discussed in C.4.)

C.3.1 Interface connector and pin assignments

The Centronics standard external interface connector is a 36-pin Amphenol No. 57-40360 or equivalent connector. This connector is, by definition, the same as the IEEE 1284-B connector.

Signal names, external interface connector pin assignments, and signal sources are listed in Table C.1. The interface was designed to be used with twisted pair cabling. Separate return leads (signal ground) need to be twisted within the cable to protect critical signals. For these signals, a second pin number is listed in the table. While all these pins are connected to signal ground within the printer, the cable return lead twisted with the specified signal are connected to this pin within the connector.

[7]These specifications are taken from Centronics Engineering Standard Number 9 [B2], with supplemental information from Centronics Field Bulletins, with the permission of Genicom Corporation.

Table C.1—Centronics connector pin assignments

Signal name	Pin	Source	Signal name	Pin	Source
/Strobe	1, 19	Host	Select (On-line)	13	Printer
Data 1	2, 20	Host	Signal Ground*	14	
Data 2	3, 21	Host	OscExt	15	Printer
Data 3	4, 22	Host	Signal Ground	16	
Data 4	5, 23	Host	Chassis Ground	17	
Data 5	6, 24	Host	5 V	18	Printer
Data 6	7, 26	Host	/Input Prime	31, 30	Host
Data 7	8, 25	Host	/Fault	32	Printer
Data 8	9, 27	Host	Light Detect*	33	Printer
/Ack	10, 28	Printer	Line Count	34	Printer
Busy	11, 29	Printer	Line Count Return	35	Printer
Paper Empty	12	Printer	Reserved*	36	Printer

NOTES
1—Second pin number is twisted pair return (signal ground).
2—A leading slash (e.g., "/Name") denotes an active low signal.
3—Signals marked with "*" (pins 14, 33, and 36) were redefined as host-source signals in the PC-compatible interface and are required for IEEE Std 1284-2000 bidirectional operation.

Strobe (Pins 1/19)	This negative going pulse is used to transfer incoming data from the data lines into the printer electronic circuitry. The pulse duration has to be a minimum of 1.0 μs, not to exceed 500 μs. This signal drives TTL logic terminated by a 470 Ω resistor to 5 V. See Figure C.1 for the relationship of leading and trailing edges of the data strobe with data on data lines.
Data Lines (Pins 2/20, 3/21, 4/22, 5/23, 6/24, 7/25, 8/26, 9/27)	The eight (8) data lines drive TTL logic terminated by a 1 kΩ resistor to 5 V. A high logic level represents a binary one, and a low logic level represents a binary zero. The logic level of each data line has to be settled at least 1 μs before the leading edge of the strobe pulse and remain at its logic level at least 1 μs after the trailing edge of the strobe pulse.[a]
Input Prime (Pins 31/30)	This line, when low, causes the input buffer to be cleared, the printer logic to be reset, and the print head to be returned to the left margin. This signal is terminated by a 470 Ω resistor to 5 V.

[a]Typically, the Data 1 signal is used for the least significant bit, and Data 8 for the most significant bit, of the data byte being transferred. However, this is strictly a convention, as data interpretation is the responsibility of the processes that are communicating via this interface. Interface operation in no way depends on any particular bit order or meaning.

C.3.2 Signal definitions

Definitions of host- and printer-generated signals are given in the following subclauses.

C.3.2.1 Host-generated signals

Host-generated signals consist of the eight data lines plus two control signals. The characteristics of each of these, and the associated connector pins, are listed here.

C.3.2.2 Printer-generated signals

There are ten printer-generated signals, plus one pin reserved for printer-specific functions. The characteristics of each of these, and the associated connector pins, are listed here.

Five of these signals, by common usage, belong to the "core set" needed to realize the interface. These five "core" signals are Acknowledge, Busy, Paper Empty, Select, and Fault.

Acknowledge (Pins 10/28)	This negative going signal is used to verify the completed transfer of incoming data or to signify the completion of a functional operation (such as carriage return, line feed, vertical tab, form feed, delete, or bell). Once a code is sent to the printer, an acknowledgment pulse has to be received before a new code can be sent (unless otherwise specified). The acknowledgment pulse is sourced from a TTL logic circuit for at least 1 ms. All codes are acknowledged even if it is a nonfunctional code for the product. Transmission of data prior to receipt of acknowledge for the previous data may result in loss of data.
Busy (Pins 11/29)	This high going signal provides a positive dc level indication during any interval when the printer cannot receive data. It is also positive (active) when the paper-empty and/or fault status line is true or an input prime is present. It is sourced from the output of a TTL logic circuit.
Paper Empty (Pin 12)	This high going signal provides a positive dc level indication during a paper- empty condition. A paper-empty condition also causes the fault and busy lines to be active. It is sourced from the output of a TTL logic circuit.
Select (Pin 13)	This high going signal provides a positive dc level indication that the printer is selected and is available for data transfer. The nonselect condition (de-select) causes the busy and fault lines to be active. It is sourced from the output of a TTL logic circuit.
External Oscillator (Pin 15)	This external oscillator signal is provided for external interface timing (typically 100–200 kHz, when present). This signal is sourced from a TTL logic circuit.
5 V (Pin 18)	This high going signal provides a positive dc level indication that the printer is connected and powered on.
Fault (Pin 32)	This low going signal provides a positive dc level indication that a fault condition exists in the printer. This signal is sourced from a TTL logic circuit.
Light Detect (Pin 33)	This high going signal provides a positive dc level indication that indicates the video lamp is inoperative. It is sourced from the output of a TTL logic circuit.
Line Count (Pin 34)	Both sides of the line count switch appear at the interface connector. This switch is opened and closed during each line feed operation. A voltage level delivered to the switch would be pulsed off and on each time a line feed operation is performed.
Line Count Return (Pin 35)	Return side of line count switch. Refer to Line Count signal.
Reserved (Pin 36)	Not used. This pin is reserved for printer-specific functions.[a]

[a]Pin 36 was redefined in the PC-compatible interface to be an input signal. Please refer to Clause 5 for additional information.

C.3.3 Electrical specifications

C.3.3.1 Voltage levels

The interface circuitry is 5 V (nominal) TTL logic (SN7400 Series). Voltage levels range between 0 V and 5 V (nominal).

C.3.3.2 Logic levels, pulses, and time

A logic One (or high) signal is defined as a voltage in the range of 2.4 V to 5 V, not to exceed a peak positive voltage of 5.5 V.

A logic Zero (or low) signal is defined as a voltage in the range of 0.0 V to 0.4 V, not to exceed a peak negative voltage of –0.5 V.

A signal whose pulse width is not critical is defined as a level (e.g., the data inputs). A signal whose pulse width is critical is defined as a pulse (e.g., Strobe), and the pulse width is specified. Pulse width is measured at 2.4 V for a logic One and 0.4 V for a logic Zero.

Delay time is defined as the interval between the specified signal and the reference signal at the receiving end of the cable. It is measured at the 2.4 V point for a logic One and 0.4 V for a logic Zero.

Switching time is defined as the rise or fall time of a signal, whichever is greater. It is specified between 0.4 V and 2.4 V points. Maximum switching time for signals is 0.2 μs (not including setup and hold times).

C.3.3.3 Current requirements

The printer interface can source up to 0.320 mA at 2.4 V for a high signal output and sink up to 14 mA for a low voltage. Similarly, the sending device interface has to be able to source 0.320 mA at 2.4 V for a high voltage and sink up to 14 mA for a low output.

C.3.3.4 Line termination

The printer interface terminates input data lines Data1–Data8 with 1 kΩ to 5 V and control lines Strobe and Input Prime with 470 Ω to 5 V.

C.3.4 Interface timing

In general, the data transfer sequence consists of the host device placing the appropriate character code on the data lines to the printer and then generating the Strobe pulse. The printer, after a slight delay, responds by activating the Busy signal, then deactivates Busy and generates an Acknowledge pulse. This sequence is illustrated in Figure C.1.

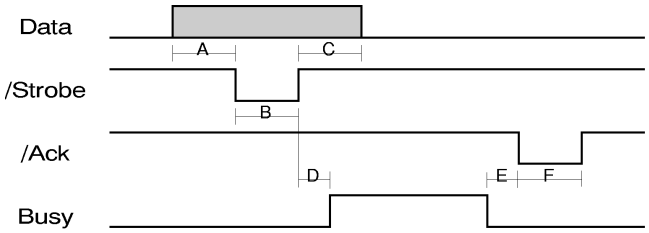
There are four timing features of particular interest in the Centronics Standard Parallel Interface.

a) A Strobe pulse is not recognized until the previous character data has been acknowledged.

b) The Strobe pulse is used as a gated signal, and the data value sampled is the state of the data lines at the trailing edge of Strobe. (In contrast, both the IEEE 1284 interface and some recent “Centronics-compatible” printers latch the data lines at the leading edge of Strobe.)

c) The Ack pulse follows the trailing edge of Busy. (A significant later change was to put the Ack pulse inside of Busy.)

d) The busy period (after receipt of Strobe) may be prolonged indefinitely. That is, there is no specified maximum time for which Busy may be asserted.



Legend	Time Interval	Min	Max
A	Data setup time	1.0	
B	Strobe pulse width	1.0	500
C	Data hold time	1.0	
D	Busy delay	0	1.5
E	Ack delay	2.5	10
F	Ack pulse width	2.5	6

All times in μs

Figure C.1—Centronics Standard Parallel Interface Timing

C.4 PC Parallel and PC-Compatible Printer Interfaces

The PC Parallel Interface was introduced by International Business Machines Corporation in 1981 for a series of small personal computers. Originally provided as a second interface on the Monochrome Adapter card, this interface has been widely adopted throughout the industry, becoming the first dominant Centronics-compatible host-side implementation. At the same time, a series of PC-compatible printers were introduced. These printers provided a parallel interface that was, in most respects, Centronics-compatible, although it contained several differences. This printer variant of the parallel interface has become known as the PC-Compatible Printer Interface.

This subclause provides core specifications for the PC-Compatible Printer Interface and the PC Parallel Interface.[8] This includes interface connector and pin assignments, and electrical specifications. Signal descriptions are limited to differences from the Centronics Standard Parallel Interface, described in the preceding subclause.

The PC Parallel Interface, although Centronics-compatible, introduced three significant differences

a) The host interface used a 25-pin D-shell (DB-25) connector, reducing the pin count by terminating ten of the cable return leads into five ground pins inside the cable connector (i.e.,connector pins 18–22 are each attached to two cable wires).[9]

b) Two additional host control signals were defined, AutoFdXT and Select In. On the printer end, three signals were redefined to accommodate the added control signals and return leads.

c) The printer data transfer timing was modified to move the Acknowledge pulse inside the Busy signal, such that Acknowledge ended at the same time that Busy ended. However, the PC host interface remained Centronics-compatible, as it ignored the Acknowledge/Busy timing relationship. The PC interface relied on the Acknowledge pulse for data transfer handshaking.

In addition to the Acknowledge/Busy timing change, the PC-Compatible Printer Interface also increased signal line termination values.

[8]These specifications are taken from the IBM Personal Computer Technical Reference Options and Adapters Manual [B7], with supplemental information from the IBM Personal Computer Hardware Reference Library [B4] and the Epson Dot Matrix Printers Reference Manual [B3].

[9]This smaller connector allowed both the Monochrome Display and the Parallel Interface circuitry to be placed on a single card. A 37-pin connector, while often used on earlier host systems, required almost all of the limited card-edge connector space. Using a 25-pin connector for the parallel interface allowed two connectors to be placed in this very valuable space, as the monochrome display connector was very small.

C.4.1 Interface connectors and pin assignments

The PC Parallel Interface connector is a female 25-pin D-shell (DB-25) connector. This connector is, by definition, the same as the IEEE 1284-A connector. The PC-Compatible Printer Interface connector is the same 36-pin D-shell as the classic Centronics interface (IEEE 1284-B connector). Table C.2 specifies signal names, connector pin assignments and corresponding printer connector pins, and signal sources.

Table C.2—PC Parallel Interface and PC-Compatible Printer Interface connector pin assignments

Signal name	PC connector pins		Printer connector pins		Source
	Signal	Return	Signal	Return	
/Strobe	1	18	1	19	Host
Data 1	2	18	2	20	Host
Data 2	3	19	3	21	Host
Data 3	4	19	4	22	Host
Data 4	5	20	5	23	Host
Data 5	6	20	6	24	Host
Data 6	7	21	7	25	Host
Data 7	8	21	8	26	Host
Data 8	9	22	9	27	Host
/Ack	10	22	10	28	Printer
Busy	11	24	11	29	Printer
Paper Empty	12		12		Printer
Select (On-line)	13		13		Printer
/AutoFdXT*	14		14		Host
/Fault	15		32		Printer
/Init (/Reset)	16	25	31	30	Host
/Select In*	17	23	36	33*	Host
Chassis Ground			17		
Not Used*	N/C		15		

Table C.2—PC Parallel Interface and PC-Compatible Printer Interface connector pin assignments (continued)

Signal name	PC connector pins		Printer connector pins		Source
	Signal	Return	Signal	Return	
Signal Ground	N/C		16		
Not Used*	N/C		18		
Not Used*	N/C		34		
5 V*	N/C		35		Printer

NOTES
1—A leading slash (e.g., "/Name") denotes an active low signal.
2—Chassis ground connection is made to the host-connector shroud via a cable shield.
3—Signals denoted with "*" are redefined from the classic Centronics interface.
4—Pin numbers denoted "N/C" indicate no connection to the host connector.
5—Signal names in parentheses [e.g., Select (On-line)] are commonly used synonyms for these signal names.

C.4.2 Signal definitions

This subclause provides definitions for signals added or redefined by the PC Parallel and PC-Compatible Printer Interfaces. Except as specified here, all other signals remain the same as the Centronics Standard Parallel Interface as defined in C.3.2.

Definitions of host- and printer-generated signals are given in Table C.2.

C.4.2.1 Host-generated signals

This subclause lists changes and differences for host-generated signals. When both host and printer connectors use the same pin number, only one pin number is given.

The PC Parallel and PC-Compatible Printer Interfaces introduced two additional host-generated control signals, AutoFdXT and Select In. These signals, although seldom used, are described in Table C.2.

AutoFdXT (Pin 14)	This line, when low, causes the paper to be automatically fed one line upon receipt and execution of the Carriage Return control code.
Init (Host pins 16/25, Printer pins 31/30)	This line performs the same "hard reset" function as the classic Centronics signal Input Prime. However, the IBM/Epson specification added a minimum pulse width requirement of 50 μs.
Select In (Host pins 17/23, Printer pins 36/33)	This line, when low, enables data input into the printer. When supported by the printer, details vary with the implementation. In some cases, this line acts as a simple active low enable line. In others, this signal enables ASCII DC1/DC3 control codes as printer Select/De-select commands to enable or disable data input. Note that printer pin 33 has been redefined as a signal ground return lead for this signal.

C.4.2.2 Printer-generated signals

This subclause lists changes and differences for printer-generated signals, even though many of these signals were, technically, listed as “No Connection” for PC hosts by many manufacturers. Accordingly, pin numbers given are for the printer connector only.

Not Used (Pin 15)	Not used by the PC-Compatible Printer Interface. Not connected to the PC Parallel Interface. (The Centronics Standard Parallel Interface defined this lead as the External Oscillator signal.)
Not Used (Pin 18)	Not used by the PC-Compatible Printer Interface. Not connected to the PC Parallel Interface. (The Centronics Standard Parallel Interface defined this lead as the 5 V signal.)
Signal Ground (Pin 33)	Signal ground connection, added for the return lead of the Select In signal. (The Centronics Standard Parallel Interface defined this lead as the Light Detect signal.)
Not Used (Pin 34)	Not used by the PC-Compatible Printer Interface. Not connected to the PC Parallel Interface. (The Centronics Standard Parallel Interface defined this lead as the Line Count signal.)
5 V (Pin 35)	This high going signal provides a positive dc level indication that the printer is connected and powered on. It is a new signal definition; however, it is not connected to the PC Parallel Interface. (The Centronics Standard Parallel Interface defined this lead as the Line Count Return signal.)

C.4.3 Electrical specifications

C.4.3.1 Voltage levels

The interface circuitry for both the PC Parallel and PC-Compatible Printer Interfaces typically used 74LSxx Series logic (5 V TTL) for both input and output signals, although 7400 Series logic was occasionally used to drive some control signals.

C.4.3.2 Logic levels

The PC Parallel and PC-Compatible Printer Interfaces used the same logic levels (TTL 5 V nominal) and signal definitions as the Centronics Standard Parallel Interface.

C.4.3.3 Current requirements

Input current requirements have not been published for either the PC Parallel or the PC-Compatible Printer Interfaces. However, as these interfaces typically used 74LSxx Series logic and larger pull-up resistor values, required input currents can be expected to somewhat less than that needed for the Centronics Standard Parallel Interface. The PC-Compatible Printer Interface input requirements can also be estimated from the published PC interface output current specifications.

The output current provided by the original (Monochrome Adapter) PC interface did not, strictly speaking, meet Centronics interface requirements. In particular, the open collector drivers provided for the control signals were limited to sinking only 7 mA at 0.8 V.[10] However, later parallel interface adapters (such as the Serial/Parallel Adapter provided for the PC/AT computer), provided additional sink current that corrected this problem.

Table C.3 lists the output current provided by both the original Monochrome Adapter and some later parallel interfaces. This table also includes values for local pull-up resistors provided to source the current for the open collector circuits.

Table C.3—PC Parallel Interface output current

Signal	Circuit	Pull-up	Source	PC Sink	PC/AT Sink
Control	Open Collector	4.7 kΩ	—	7 mA at 0.8 V	16 mA at 0.4 V
Data1–Data8	74LS24x	—	3 mA at 2.4 V	24 mA at 0.5 V	24 mA at 0.5 V
NOTES 1—Control signals include Strobe, Init, AutoFdXT, and Select In. 2—The 4.7 kW pull-up resistor is sufficient to source 0.55 mA at 2.4 V. 3—74LS24x indicates a member of the SN74LS240 Series of line driver circuits. 4—These values are typical for the 74LS series.					

C.4.3.4 Line termination

The PC-Compatible Printer Interface typically terminated both input data lines (Data1–Data8) and control signals (Strobe, Init, AutoFdXT, and Select In) with 1 kΩ to 5 V. However, some implementations have terminated data lines with values as high as 11 kΩ, and control signals with values as high as 3.3 kΩ.

The PC Parallel Interface provided no explicit line termination for input status signals. That is, these signals (Ack, Busy, Paper Empty, Select, and Fault) are connected directly to the inputs of 74LSxx Series logic, with no external termination network.

C.4.4 Interface timing

In general, the data transfer sequence is the same as the Centronics Standard Parallel Interface. However, there were three significant changes in the PC-Compatible Printer Interface.

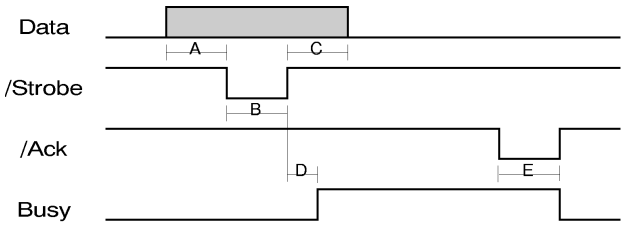
a) The Acknowledge/Busy timing relationship was modified such that the Acknowledge pulse was generated within the Busy active period. That is, the Acknowledge pulse ended at the same time that Busy ended. This sequence is illustrated in Figure C.2.
b) Minimum strobe pulse width, and data setup and hold times, were reduced to 0.5 μs.
c) Minimum Init pulse width to reset the printer was defined to be 50 μs.

C.4.4.1 PC-Compatible Printer Interface timing

Data transfer timing is shown in Figure C.2; applicable pulse widths, setup, and hold times are listed in this figure.

.

[10]This initial unfortunate oversight is largely responsible for the 2 m cable length restriction often associated with the PC parallel interface.



Legend	Time Interval	Min	Max
A	Data setup time	0.5	
B	Strobe pulse width	0.5	500
C	Data hold time	0.5	
D	Busy delay	0	
E	Ack pulse width	2.5	

All times in μs

Figure C.2—PC-Compatible Printer Interface timing

C.4.4.2 PC Parallel Interface timing

The PC Parallel Interface specified a minimum input Acknowledge pulse width of 0.5 μs at the PC interface connector.

Output data setup and hold times, as well as Strobe pulse width, are not guaranteed. This is primarily due to the lack of hardware support for these signals. Instead, each signal is under direct software control, and software instructions are responsible for writing the output data, waiting to assure setup, explicitly setting Strobe low, waiting for the pulse width, and then explicitly setting Strobe high. While ROM BIOS support is provided for these actions, nothing prevents application software from taking direct control of these signals. However, due to both software latency and programming allowances, output timing is typically provided with generous margins.

C.5 Enhanced and bidirectional parallel printer interface

Later PCs, including most microchannel machines introduced in 1987 and later, provided additional enhancements to the PC Parallel Interface.[11] This interface, while completely backward-compatible with the PC Parallel Interface, added bidirectional drivers to the eight data lines, as well as provisions for reversing the directions of nominal output (printer control) signals. This new interface, however, used the same interface connector, pinout, and signals as the earlier PC interface.

These new computers provided three variations of the enhanced parallel interface, typically denoted PS/2 Type 1, Type 2, and Type 3. Although these three types differed primarily by internal system-related enhancements, the Type 2 and Type 3 interfaces provided hardware support for setup time and Strobe pulse generation.

C.5.1 Interface connectors and pin assignments

The Enhanced (Types 1, 2, and 3) interface connectors and pin assignments are identical to the PC Parallel Interface.

C.5.2 Signal definitions

The Enhanced interface signals are identical to the PC Parallel Interface, when configured for printer output. While the interface signals can be redefined for bidirectional data transfer, these redefinitions are unique to proprietary software and are beyond the scope of this annex.

[11]These specifications are taken from the IBM Personal System/2 Hardware Interface Technical Reference [B5].

C.5.3 Electrical specifications

C.5.3.1 Voltage and logic levels

The Enhanced interfaces support the same voltage and logic levels as the PC Parallel Interface. These circuits are 5 V TTL, typically 74LSxx Series or equivalent logic.

C.5.3.2 Line termination and current requirements

Line terminations for the Enhanced interfaces have varied with the exact implementation. While output current values are relatively consistent, input current requirements vary, typically with variations in the input termination. Generally, these values remain compatible with both the Centronics Standard Parallel Interface and the PC-Compatible Printer Interface.

Figure C.3 shows the general arrangement of drivers, receivers, and line termination components for this interface. Signals that are input only do not include the driver circuit. Also, some signals do not provide all termination components, while others are present only in some implementations. For example, the series resistor (R_2) is present only for Data and Strobe signals.

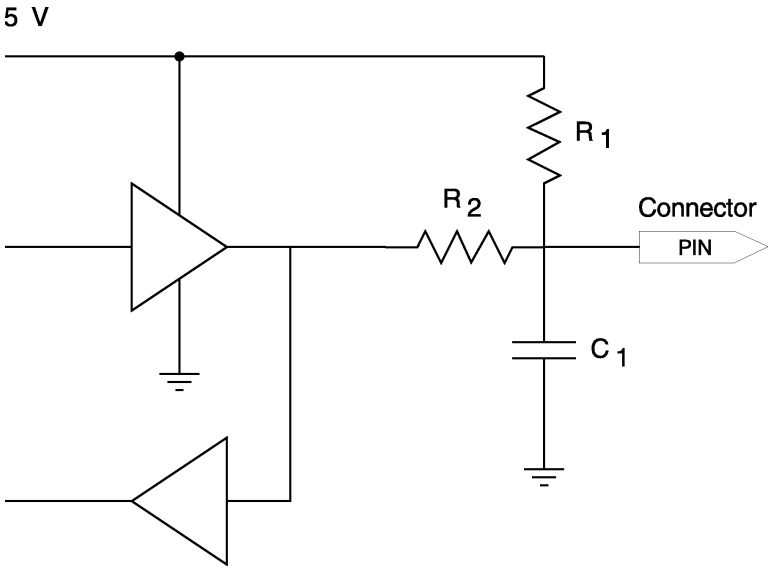


Figure C.3—Enhanced (Type 1, 2, and 3) interface circuits

Table C.4 identifies termination components for each signal, as well as driver/receiver circuitry and output and input characteristics. Where components have varied with the implementation, a range of values is given. Termination components that are not included in all implementations are also identified

C.5.4 Interface timing

Enhanced interface timing is essentially the same as the PC interface. In particular, since direct software control of the output signals was retained, output timing cannot be guaranteed for the reasons outlined in C.4. However, the Type 2 and Type 3 interfaces also provide the ability to enable hardware data setup and strobe pulse generation.

When enabled, the Type 2 and Type 3 interfaces generate data transfer timings automatically. These interfaces provide data setup times of 1.0 μs ± 0.25 μs, and a Strobe pulse width of 1.0 μs ± 0.25 μs.

C.6 Printer interface variations

Printer variations of the basic Centronics interface flourished in the decade following the introduction of the first MS-DOS-compatible PC. Almost all of these printers claimed Centronics compatibility, but in many cases this was not strictly true. Typically, the "core" interface signals and basic handshaking remained the same, but signal timing and edge relations often differed. Most commonly, printer variations involved Busy signal timing, both at the leading edge (with respect to Strobe) and the trailing edge (with respect to Ack). [12]

These variations resulted from the lack of an industry standard specification for the interface. Without an authoritative specification, many printer vendors were forced to rely on the meager and inadequate documentation published in user manuals. As the target market was often the rapidly expanding PC base, many printers were designed by studying how PC software used the interface, rather than by matching Centronics printer operation. Accordingly, while timing details varied widely, these printers typically worked well with most PCs. However, printers from different manufacturers were seldom "plug-compatible," and often were not "plug-compatible" with Centronics, even when Centronics interface compatibility was claimed.

The most common variations involved Busy signal timing. These included both when the Busy signal was asserted and when it was released. During data transfer, printers assert Busy upon receipt of a Strobe pulse and release Busy when an Ack pulse is issued. The relationship of Busy to Strobe generally occurred in two ways:

— Busy-while-Strobe
— Busy-after-Strobe

However, the relationship of Busy to Ack generally occurred in three ways:

— Ack-in-Busy
— Ack-after-Busy
— Ack-while-Busy

Of these Busy timing variants, Centronics printers used only Busy-after-Strobe and Ack-after-Busy. Peripherals supporting IEEE 1284 Compatibility Mode use the Busy-while-Strobe and Ack-in-Busy conventions.

Each of these sets of Busy timing relationships are discussed in the following subclauses.

[12]The edge relationships of the other printer status signals (Online, Paper Empty, and Fault) have also varied widely, both among these signals and with respect to the Busy signal. These status signals, however, do not directly participate in the data transfer process, and it was generally well known to host software that they could change state asynchronously. Accordingly, although almost every possible combination of edge relations among these signals have been used, these typically had no effect on interface operation except in the rare cases where the edge-to-edge differences have become large).

Table C.4—Enhnaced (Type 1, 2, and 3) interface output current

Signal name	Dir (I/O)	Termination			Output characteristics					Input characteristics			
		R_1	R_2	C_1	Driver	I_{OH}	V_{OH}	I_{OL}	V_{OL}	Receiver	I_H	V_{IH}	V_{IL}
Data1 Data8	I/O	2000*	33	2200*	74LS24x	2.6	2.4	24	0.5	74LS24x	0.04	2	0.8
Strobe	I/O	2000k–4700	33	2200*	O.C.			20	0.5	74LSxx	0.04	2	0.8
Init	I/O	2000k–4700		2200*	O.C.			20	0.5	74LSxx	0.04	2	0.8
Auto-FeedXT	I/O	2000k–4700		2200*	O.C.			20	0.5	74LSxx	0.04	2	0.8
Select In	I/O	2000k–4700		2200*	O.C.			20	0.5	74LSxx	0.04	2	0.8
Ack	I	1000k–10 000*		68–2200*						74LSxx	0.04	2	0.8
Busy	I	1000k–10 000*		68–2200*						74LSxx	0.04	2	0.8
Paper Empty	I	1000k–10 000*		68–2200*						74LSxx	0.04	2	0.8
Select	I	1000k–10 000*		68–2200*						74LSxx	0.04	2	0.8
Fault	I	1000k–10 000*		68–2200*						74LSxx	0.04	2	0.8

NOTES:
1—Units: resistances are in ohms; capacitances are in picofarads; currents are in milliamperes; voltages are in volts.
2—Drivers labeled “O.C.” are open collector. High output voltage and current depend on the pull-up resistor.
3—Termination values marked with asterisk (*) indicate that these components are not present in some implementations. This is equivalent to “or none.”
4—Termination values or driver types left blank indicate that these components are not present in any implementation.

C.6.1 Strobe-to-Busy timing variations

There are two Strobe-to-Busy timing variations in general usage. These two variations typically differ in which edge of the Strobe pulse is considered to be the "active" edge. The common names for these timing variations arise from the appearance of the wave forms. For Busy-while-Strobe timing, the falling (or leading) edge of the Strobe pulse is used to activate Busy. Thus, Busy is usually asserted while Strobe is asserted. For Busy-after-Strobe timing, Busy is activated by the rising (or trailing) edge of Strobe. Here, Busy is asserted only after the Strobe pulse is complete.

These two timing variations are contrasted in the following figures. Busy-while-Strobe is shown in Figure C.4, and Busy-after-Strobe is shown in Figure C.5. In both cases, the delay from the active edge of Strobe to the assertion of Busy is indicated by the time "t." Typical values for Strobe edge to Busy active delays are listed in Table C.5.

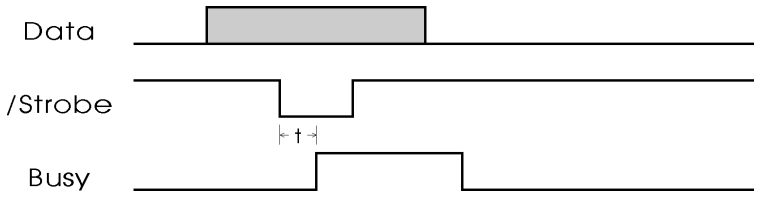


Figure C.4—Busy-while-Strobe timing

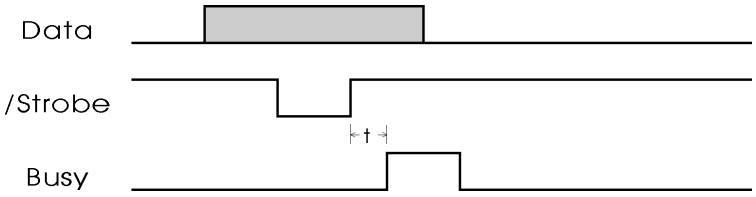


Figure C.5—Busy-after-Strobe timing

Table C.5—Typical Strobe-to-Busy delay timings

Timing variant	Minimum	Maximum
Busy-while-Strobe	0 s	1.0 s
Busy-after-Strobe	0 s	1.5 s

These timing values illustrate the dangers of the occasional but somewhat hazardous use of Busy for data flow control.[13] While Busy is asserted upon receipt of a Strobe pulse, this may not happen immediately. (In some cases, assertion of Busy may be deferred indefinitely, such as the second Busy-after-Strobe variant listed in Table C.5.) Host systems that rely solely on the state of Busy to determine when the next byte can

[13]The original Centronics specification cautioned that only Ack should be used to verify the receipt of the data (see C.3.2.2).

be sent risk losing data. This can occur when the host writes a byte to the interface, looks at Busy and finds it not set, then writes the next byte. If, under these conditions, the printer has not yet set Busy for the first byte, the second data byte will be lost.

The Busy-while-Strobe variant attempts to address this problem by asserting Busy upon receipt of the leading edge of Strobe. While this reduces the probability of encountering problems, it does not eliminate the race condition hazard introduced by hosts relying solely on the state of Busy for flow control. The original Centronics interface was designed around the concept of Strobe/Ack handshaking, and this remains the best method to ensure data transfer integrity.

C.6.2 Busy-to-Ack Timing Variations

There are three Busy-to-Ack timing variations in general usage. These variations differ in whether the Ack pulse is issued before or after the Busy signal is de-asserted, or if Busy is de-asserted while the Ack pulse is active. As in the previous clause, the common names for these timing variations arise from the appearance of the wave forms. The common names of the three variants are

— Ack-in-Busy: The Ack pulse is sent before Busy is de-asserted (the Ack occurs inside the Busy signal).

— Ack-after-Busy: The Busy signal is de-asserted before the Ack pulse is sent (the Ack occurs after the Busy signal).

— Ack-while-Busy: The Ack pulse begins while Busy is active, but ends after Busy has been de-asserted (the Busy signal changes while Ack is active).

These timing variations are compared in Figure C.6. Typical timing values for the associated Ack and Busy edges are listed in Table C.6.

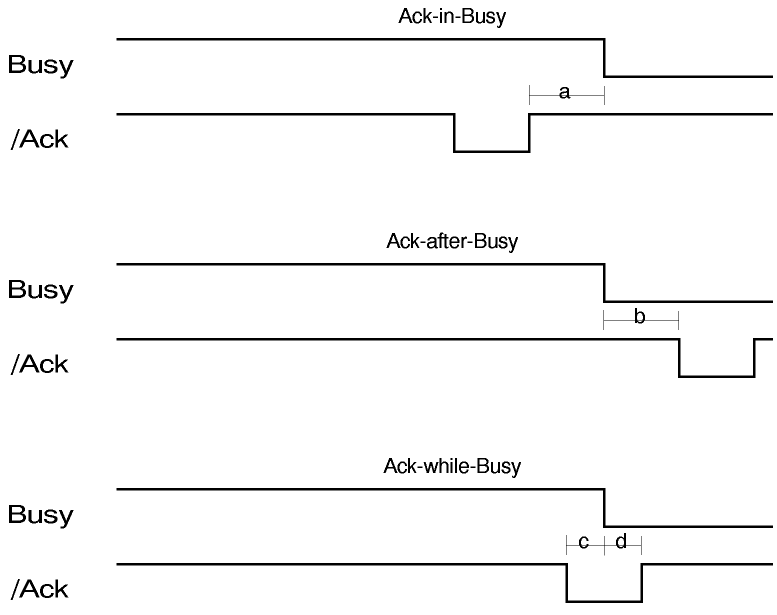


Figure C.6—Busy/Ack timing relations

Table C.6—Typical Busy/Ack timings

	Legend	Minimum	Typical	Maximum
Ack-in-Busy				
Ack high to Busy low	a	0 μs	2.5 μs	—
Ack-after-Busy				
Busy low to Ack low	b	2.5 μs	—	10 μs
Ack-while-Busy				
Ack low to Busy low	c	—	5 μs	—
Busy low to Ack high	d	—	7 μs	—

These timing variations generally have little effect upon the integrity of the data transfer, as both signals are controlled by the printer. Typically, regardless of the method used to terminate the transfer cycle, the printer ensures that internal data transfer has been completed before initiating the end-of-cycle process. However, host systems should exercise some caution before beginning the next data transfer cycle, due to these timing variations.

In particular, it is important to recall that the Busy signal can be asserted under a variety of conditions, not just during the data transfer cycle. Host systems that rely on Ack to determine the end of the data transfer cycle should ensure that Busy has been cleared before strobing the next byte. This is particularly important when connected to printers that use the Ack-in-Busy and Ack-while-Busy timing variants. As shown in Table C.6, these variants allow the Busy signal to be prolonged indefinitely, even after the Ack pulse has been initiated.

Annex D

(informative)

Bibliography

[B1] ANSI/IEEE Std 260.1-1993, American National Standard Letter Symbols for Units of Measurement (SI Units, Customary Inch-Pound Units, and Certain Other Units).

[B2] Centronics Engineering Standard Number 9, Revision B, Apr. 9, 1980.

[B3] Epson America Inc., Epson Dot Matrix Printers Reference Manual, 1988.

[B4] IBM Personal Computer Hardware Reference Library, Volume V (Number 6025005).

[B5] IBM Personal System/2 Hardware Interface Technical Reference—Common Interfaces, Oct. 1990 (S84F-9809); Update, Oct. 1991 (S04G-3281).

[B6] IBM Technical Reference, Personal Computer AT, Mar. 1984 (1502243).

[B7] IBM Technical Reference, Personal Computer Options and Adapters, Volume 1, Apr. 1984 (6322509); Volume 2, Apr. 1984 (6322509); and Volume 3, Aug. 1984 (6322522).

[B8] IEEE Std-1284.3-2000, IEEE Standard for Interface and Protocol Extensions to IEEE 1284-Compliant Peripherals and Host Adapters.

[B9] IEEE Std-1284.4-2000, IEEE Standard for Data Delivery and Logical Channels for IEEE 1284 Interfaces.

[B10] Intel Corp., Intel 386 SL Microprocessor SuperSet Data Book, Sept. 1991 (240814-003).

[B11] Intel Corp., Intel 386 SL Microprocessor SuperSet Programmer's Reference Manual, 1991 (240815-002.)

[B12] Intel Corp, Intel 386 SL Microprocessor SuperSet System Design Guide, 1991 (240816-002).

[B13] ISO 7498:1984, Information processing systems—Open systems interconnection—Basic reference model.

[B14] Microsoft Corp., Extended Capabilities Port Protocol and ISA Interface Standard, Revision 1.14, July 14, 1993.

NOTE—This document is available at http://www.fapo.com/ieee1284.htm.

[B15] SMSC FDC37C665/666GT Super I/O Controller, SMSC Data Sheet.

NOTE—This document is available at http://www.smsc.com.

[B16] The IEEE Standard Dictionary of Electrical and Electronic Terms, Sixth Edition.

Annex E

(informative)

Reducing data loss in ECP reverse termination

IEEE 1284 modes ECP and EPP have been incorporated in nearly all new PC platforms shipped since 1997. The implementation of these modes is performed using a combination of software-required tasks and hardware state machines to implement the actual data transfer phases. The two best sources for information detailing this host parallel port register model are

1) Microsoft Corp.: Extended Capabilities Port Protocol and ISA Interface Standard [B14]
2) SMSC: SMSC FDC37C665/666GT Super I/O Controller Data Sheet [B15]

The Microsoft document describes how the ECP, Byte, and Compatible modes are to be implemented on an ISA interface. The SMSC document is a superset of the Microsoft model and adds standardized support for EPP mode. Many, but not all of the various Super I/O controllers on the market follow these register models. Support for the SMSC register model is virtually mandatory for driver developers due to its wide spread usage.

This annex deals with an issue that is a result of particular host controller chip implementations, and not with a fault of IEEE Std 1284-1994. While these implementations are outside the scope of this standard it is necessary to clarify this particular problem and present a workaround.

When in the ECP Reverse phase, data transfer is controlled by the peripheral, not the host controller (refer to Figure 14). The host requests the reverse phase when it sets nInit low (event 39). Once in reverse, it is the peripheral that clocks the data into the host PC. The host signals that it is ready by asserting HostAck low (event 46). When the peripheral has data or command to send, it places this information on the databus and asserts PeriphClk low (event 43). When the host can accept the byte, it will acknowledge the PeriphClk by driving HostAck high (event 44). The peripheral will then drive PeriphClk high (event 45). **It is this edge/ event that the data is considered transferred.** The host will then assert HostAck low (event 46) indicating that it has accepted the data and is ready for the next cycle.

The problem being presented in this annex is a result of some Super I/O controllers accepting the data at event 44 rather than event 45. When in the ECP Reverse phase, the host PC may request a termination to ECP forward at any time. This termination may occur when HostAck is either low or high. In other words, it may occur anytime from event 46 up to event 45. An example of this is shown as event 47 in Figure 14. When this occurs, both the host and the peripheral are supposed to discard the current data byte being transferred and then continue with the termination back into ECP forward.

The problem is that if the I/O controller has internally accepted the data at event 44, and a reverse termination is occurring between events 44 and 45, then the host has accepted the byte and the peripheral has not counted it as transferred. This causes a miscount in the data transfer. This byte may be accepted by the host twice; once before the termination, and once again upon the next reverse phase transfer. Since the register model does not require that there be any synchronization between the termination request and the current data transfer phase, the host driver software cannot “know” exactly what phase the transfer is in. While this problem does not occur on all I/O controllers, and may even be rare on those controllers that do exhibit this problem, there are a couple of actions that the driver developer can perform to limit susceptibility to this type of data transfer error.

FIFO Full: All of the integrated Super I/O controllers use a FIFO (firs in first out) to receive data during the ECP reverse data transfer phase. This FIFO is usually 8 bytes or 16 bytes in depth, although other depths are

possible. The best way to ensure that the data transfer count will be correct and that there will be a clean termination is to only terminate when the receive FIFO is full. At this point the interface may be anywhere between phases 46 and 44 and a termination will not cause any mismatch on the data transfer count. Note that the peripheral may still have more data to send (nFault asserted). The device driver may need to return to ECP reverse later to complete receive the rest of the data.

Timeout: The FIFO method is fine as long as there is enough data to fill the FIFO. Small packets (less than a FIFO size) or the runt packets that may trail a large reverse data buffer will not fill the FIFO completely. In this case a data transfer timer works the best. When the device driver wants to terminate the reverse transfer it does two things:

1) Enable the FIFO Full interrupt
2) Start a data transfer timeout timer

If the driver receives the FIFO Full interrupt then the task is explained as above. If the timer interrupt occurs then the driver will just proceed with the termination. The assumption is that the timeout interval is much longer than the time required to fill the FIFO. If this has not happened then the reverse data has been in idle for some time and the likelihood of generating an invalid termination at this point is almost nil. When the timer interrupt happens the driver should check the state of the nFault signal. If this is high then the reverse data buffer has been received completely and there is no more data for the peripheral to send. Termination at this point is not a problem. If the nFault line is still asserted then it is still all right to terminate but the host may want to return to reverse later to receive any additional data.

Theses techniques have been shown to be quite useful in preventing ECP reverse data errors.