

---

# IEEE 1284 – Updating the PC Parallel Port

Heidi Frock

Most personal computers today are equipped with a parallel port, commonly used to connect the computer to a parallel printer. Because it is available on most personal computers, the parallel port is a perfect choice for connection to other peripheral devices. However, communication to peripherals across the parallel port is limited because the interface is traditionally unidirectional and there is no standard specification for the interface. Additionally, although the performance of the PC has dramatically increased, the parallel port has remained the same. This situation has led to the development of a new parallel port standard – IEEE Standard 1284-1994. This standard is based on the original Centronics Standard Parallel Port (SPP) specification, and includes the Enhanced Parallel Port (EPP) and Extended Capabilities Port (ECP).

This document describes SPP, EPP, and ECP, as defined by IEEE 1284.

## The Standard Parallel Port – the Centronics Parallel Port

The Standard Parallel Port (SPP) is also known as the Centronics parallel port. Centronics Data Computer Corporation developed the interface in the mid-1960s to be an 8-bit unidirectional parallel host-to-printer connection. The interface became widely used; however, no industry-standard specification was developed to define the interface.

The Centronics "standard" defines a 36-pin champ connector and interface signals for the printer side of the connection. The host side implementation varied widely until the introduction of the IBM PC in 1981. The host parallel port implementation used on the IBM PC, also referred to as the PC parallel interface, became the *de facto* industry PC parallel port interface. The PC parallel interface defines a 25-pin DSub connector with 8 unidirectional data lines, four control lines, and five status lines. A description of these signals is in Table 1.

Because there is no written standard, the timing relationships between the handshaking signals vary widely among printers from different manufacturers, even though they may all claim Centronics compatibility. This document will focus on the IBM PC parallel interface timing, the most common in the industry.

Table 1. The IEEE 1284 Signal Line Descriptions

SPP Signal Name	EPP Signal Name	ECP Signal Name	Source	Connector Pinout
<b>Data8-1</b> Unidirectional data lines. Data8 is the most significant.	<b>AD8-1</b> Bi-directional address and data lines. AD8 is the most significant.	<b>Data8-1</b> Bi-directional address and data lines. Data8 is the most significant.	Host/ Peripheral	1284-A: 9 - 2 1284-B: 9 - 2 1284-C: 13 - 6
<b>STROBE*</b> Data is valid during an active low pulse on this line.	<b>WRITE*</b> This signal is low during a write operation and high during a read operation.	<b>HostClk</b> This forward direction handshaking line is interlocked with PeriphAck and driven low when data is valid.	Host	1284-A: 1 1284-B: 1 1284-C: 15
<b>AUTOFD*</b> Usage of this line varies. Most printers will perform a line feed after each carriage return when this line is low, and carriage returns only when this line is high.	<b>DSTROBE*</b> This signal denotes data cycles. During a write operation, data is valid when this signal is active. During a read operation, this signal is low when the host is ready to receive data.	<b>HostAck</b> In the forward direction, this line is driven low for a command transfer, and high for a data transfer. In the reverse direction, this signal is a handshaking line interlocked with PeriphClk.	Host	1284-A: 14 1284-B: 14 1284-C: 17
<b>INIT*</b> This line is held low for a minimum of 50 $\mu$ s to reset the printer and clear the print buffer.	<b>INIT*</b> This line is driven low to terminate EPP mode and return to SPP mode.	<b>ReverseRequest*</b> This line is driven low to place the parallel port interface in the reverse direction.	Host	1284-A: 16 1284-B: 31 1284-C: 14
<b>SelectIn*</b> The host drives this line low to select the peripheral.	<b>ASTROBE*</b> This line denotes address cycles. When this signal is low, AD8-1 is an address.	<b>1284 Active</b> The host drives this line high while in ECP mode, and low to terminate ECP mode.	Host	1284-A: 17 1284-B: 36 1284-C: 16
<b>ACK*</b> The peripheral pulses this line low when it has received the previous data and is ready to receive more data. The rising edge of ACK* can be enabled to interrupt the host.	<b>INTR*</b> The peripheral can enable this signal to interrupt the host on the low to high transition.	<b>PeriphClk</b> The peripheral drives this reverse direction handshaking line low to indicate that the data is valid. PeriphClk is interlocked with HostAck.	Peripheral	1284-A: 10 1284-B: 10 1284-C: 3
<b>BUSY</b> The peripheral drives this signal high to indicate that it is not ready to receive data.	<b>WAIT*</b> The peripheral drives this signal low to acknowledge that it has successfully completed the data or address transfer initiated by the host.	<b>PeriphAck</b> This forward direction handshaking line is interlocked with HostClk and driven by the peripheral to acknowledge data received from the host. During reverse direction transfers, the peripheral drives this line high during data transfers and low during command transfers.	Peripheral	1284-A: 11 1284-B: 11 1284-C: 1
<b>PError</b> Usage of this line varies. Printers typically drive this signal high during a paper empty condition.	<b>User Defined</b>	<b>AckReverse*</b> The peripheral drives this line to follow the level of the ReverseRequest* line.	Peripheral	1284-A: 12 1284-B: 12 1284-C: 5
<b>Select</b> The peripheral drives this signal high when it is selected and ready for data transfer.	<b>User Defined</b>	<b>XFlag</b> The peripheral drives this line high to indicate that it uses ECP mode.	Peripheral	1284-A: 13 1284-B: 13 1284-C: 2
<b>FAULT*</b> Usage of this line varies. Peripherals usually drive this line low when an error condition exists.	<b>User Defined</b>	<b>PeriphRequest*</b> The peripheral drives this signal low to request a reverse transfer. This line can be used to interrupt the host.	Peripheral	1284-A: 15 1284-B: 32 1284-C: 4

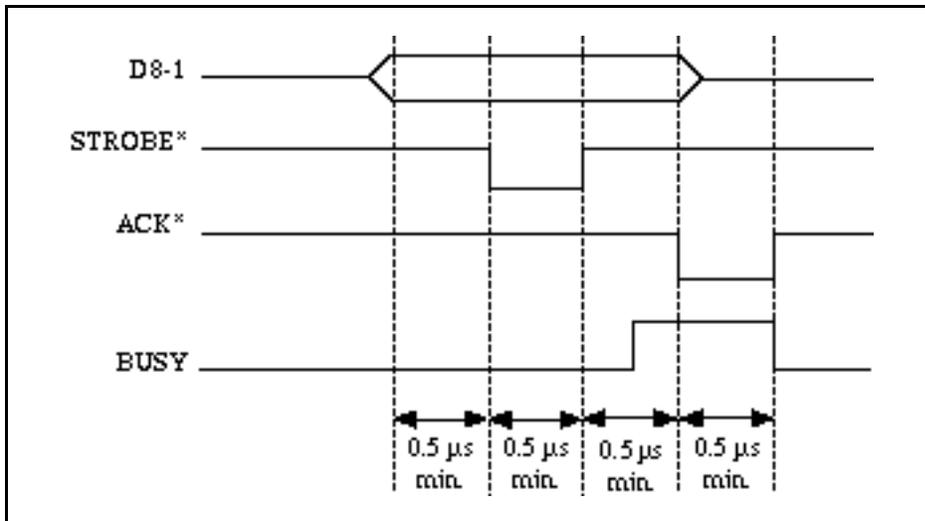


Figure 1. SPP Data Transfer Timing

The basic SPP data transfer is shown in Figure 1. When the printer is ready to receive data, it drives BUSY low. The host drives valid data on the data lines, waits a minimum of 500 ns, then pulses STROBE\* for a minimum of 500 ns. Valid data must remain on the data lines for a minimum of 500 ns after the rising edge of STROBE\*. The printer will receive the data and drive BUSY active to indicate that it is processing the data. When the printer has completed the data transfer, it will pulse the ACK\* line active for a minimum of 500 ns and de-assert BUSY, indicating it is ready for the next data byte.

The SPP defines three registers to manipulate the parallel port data and control lines and read the parallel port status lines. These registers and the corresponding offset to the parallel port starting address are shown in Table 2.

Table 2. SPP Registers

Register	Offset	7	6	5	4	3	2	1	0
Data Register	0	D7	D6	D5	D4	D3	D2	D1	D0
Status Register	1	BUSY*	ACK*	PError	Select	FAULT*	IRQ*	Reserved	Reserved
Control Register	2	Reserved	Reserved	Reserved	IRQEN	SelectIn	INIT*	AUTOFD	STROBE

The host computer software must execute four steps to perform one data byte transfer across the parallel port:

1. Write valid data to the data register
2. Poll the BUSY line - wait for it to be inactive
3. Write to the control register to drive STROBE active
4. Write to the control register to de-assert the STROBE signal

The minimum setup, hold and pulse width times required by SPP data transfers greatly limits performance. Taking into account software latency times, the maximum possible transfer rates are 150 kbytes/s. Typical transfer rates are around 10 kbytes/s.

Many file transfer programs overcome the unidirectional limitation of the PC parallel port by using four of the status lines (SLCT, BUSY, PE, ERROR) to send data to the host four bits at a time. The ACK line can be used to interrupt the host to indicate that data is ready to be read.

## The Bidirectional Port

The IBM PS/2 computer enhanced the standard PC parallel interface by adding bidirectional drivers to the eight data lines. The I/O connector and signal assignments remained the same. A parallel port with bidirectional drivers is often referred to as an extended mode parallel port. IBM refers to a bidirectional port as a Type 1 parallel port. IBM also defines Type 2 and Type 3 parallel ports which use a DMA channel to write/read blocks of data to/from the parallel port. The parallel port on most computers is configured at the factory to operate as an unidirectional parallel port. A setup utility specifically for the system must be used to select bidirectional operation.

The Register map for an IBM PS/2 parallel port is shown in Table 3. An extended mode (Type 1) parallel port has only the first three registers. These registers are identical to the SPP register set with an additional Direction bit in the parallel port control register. The last three registers in Table 3 are only present in Type 2 and 3 parallel ports.

Type 2 and 3 DMA transfers follow the SPP timing as described earlier. During Type 2 or 3 DMA writes, the DMA controller writes data to the data register and a STROBE pulse is automatically sent. When the ACK is received from the peripheral, a DMA request is sent and the next byte is then transferred. The peripheral can drive BUSY to hold off the transfer. During Type 2 or 3 DMA reads, a pulse on the ACK line generates a DMA request and initiates the transfer to system memory. The DMA controller reads the data register and a STROBE pulse is automatically generated.

Although IBM defined Type 2 and 3 parallel ports to increase parallel port performance, only IBM computers implement the ports. Thus, there is a lack of software that takes advantage of the DMA feature. By comparison, EPP and ECP are industry standards supported by a wide variety of hardware and software manufacturers.

Table 3. Bidirectional Port Registers

Register	Offset	7	6	5	4	3	2	1	0
Data Register	0	D7	D6	D5	D4	D3	D2	D1	D0
Status Register	1	BUSY*	ACK*	PError	Select	FAULT*	IRQ*	Reserved	Reserved
Control Register	2	Auto Strobe	Reserved	Direction	IRQEN	SelectIn	INIT*	AUTOFD	STROBE
Interface Control Register	3	Start DMA	Reset EOD	TC/ACK IRQEN	Select IRQEN	FAULT IRQEN	PError IRQEN	Set EOD	DMAEN
Interface Status Register	4	Reserved	EOD	TC/ACK INT	Select INT	FAULT INT	PError INT	Reserved	Reserved
Reserved Register	5	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

## Enhanced Parallel Port – EPP

EPP was developed to provide for high-speed, bidirectional data transfers that are compatible with the register map of the existing standard parallel port. The EPP specification assigns traditional microprocessor bus signaling functions to the standard parallel port lines (i.e. address strobe, data strobe) to access adapter hardware directly. See Table 1 for a description of these parallel port signals.

A data transfer on a standard parallel port requires several software steps. EPP adds additional hardware and registers to automatically generate control strobes and data transfer handshaking with a single I/O instruction. With

an ISA machine, maximum possible transfer rates are 2 Mbytes/s. Transfer rates up to 10 Mbytes/s can be achieved on other platforms.

EPP operations are typically two-phase bus cycles initiated by the host. The host first selects a register within the peripheral and performs an address cycle. Then the host performs a series of read and/or writes to that selected register. EPP defines a single interrupt request signal, INTR, enabling the peripheral with a means to signal the host. EPP has four basic operations – address write, address read, data write, and data read.

An **EPP Address Write** cycle is shown in Figure 2. The host first asserts **WRITE\***, places the address byte on the **AD8-1** lines, and asserts **ASTROBE\***. The peripheral de-asserts **WAIT\*** indicating that it is ready for the address byte. The host then de-asserts **ASTROBE\*** and the peripheral latches the address lines on the rising edge of **ASTROBE\***. The peripheral then indicates that it is ready for the next cycle by asserting **WAIT\***.

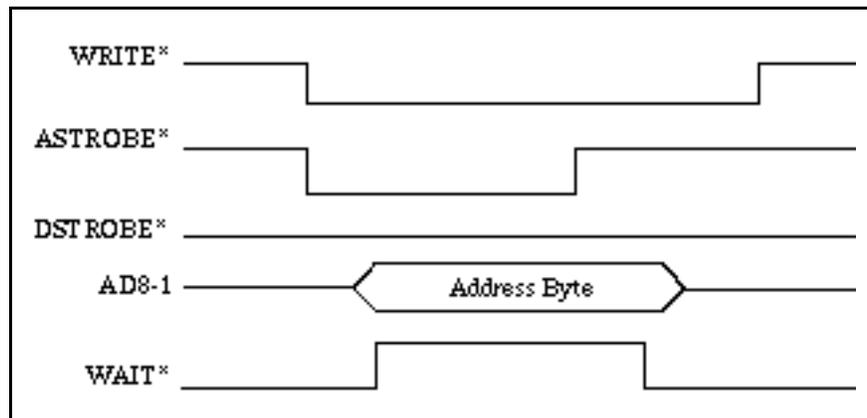


Figure 2. EPP Address Write Cycle

An **EPP Address Read** cycle is shown in Figure 3. The host de-asserts **WRITE\***, places the **AD7-0** lines in a high impedance state, and asserts the **ASTROBE\*** signal. The peripheral then drives the address byte on the **AD7-0** lines and re-asserts **WAIT\*** to indicate that the address byte is valid. The host will read the address lines when it sees the **WAIT\*** unasserted, and then de-asserts **ASTROBE\***. The peripheral then places the **AD8-1** lines in an high impedance state and asserts **WAIT\*** to indicate it is ready for the next cycle.

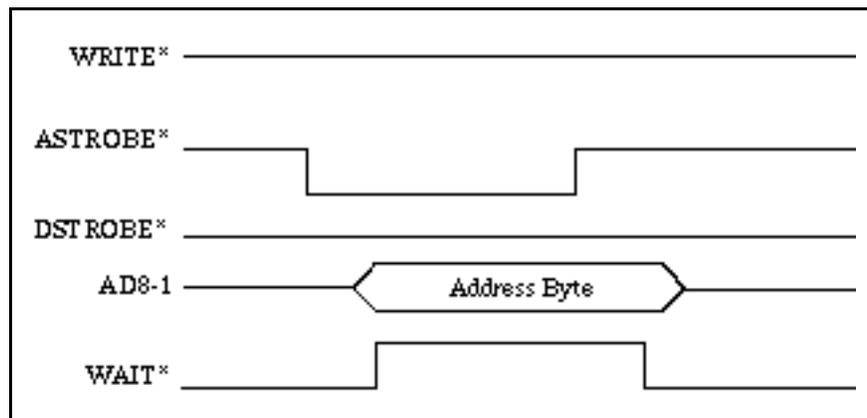


Figure 3. EPP Address Read Cycle

An **EPP Data Write** cycle is shown in Figure 4. The host first asserts **WRITE\***, places the data byte on the **AD8-1** lines, and asserts **DSTROBE\***. The peripheral de-asserts **WAIT\*** indicating that it is ready for the data byte. The

host then de-asserts DSTROBE\* and the peripheral latches the data lines on the rising edge of DSTROBE\*. The peripheral then indicates that it is ready for the next cycle by asserting WAIT\*.

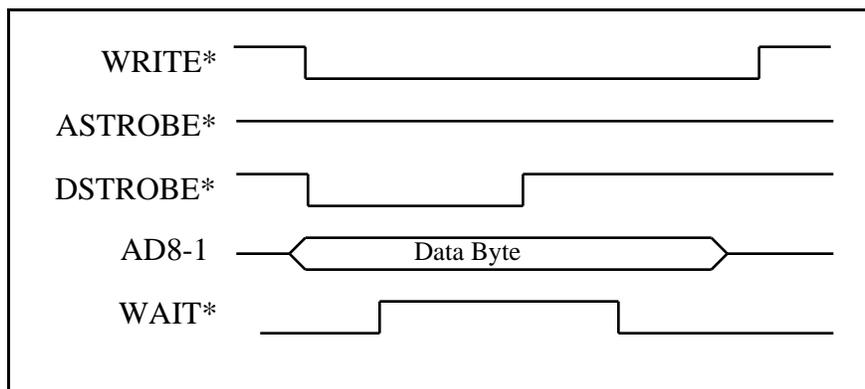


Figure 4. EPP Data Write Cycle

An **EPP Data Read** cycle is shown in Figure 5. The host de-asserts WRITE\*, places the AD8-1 lines in a high impedance state, and asserts the DSTROBE\* signal. The peripheral then drives the data byte on the AD8-1 lines and re-asserts WAIT\* to indicate that the data byte is valid. The host will read the data lines when it sees the WAIT\* unasserted, and then de-asserts DSTROBE\*. The peripheral then places the AD8-1 lines in an high impedance state and asserts WAIT\* to indicate it is ready for the next cycle.

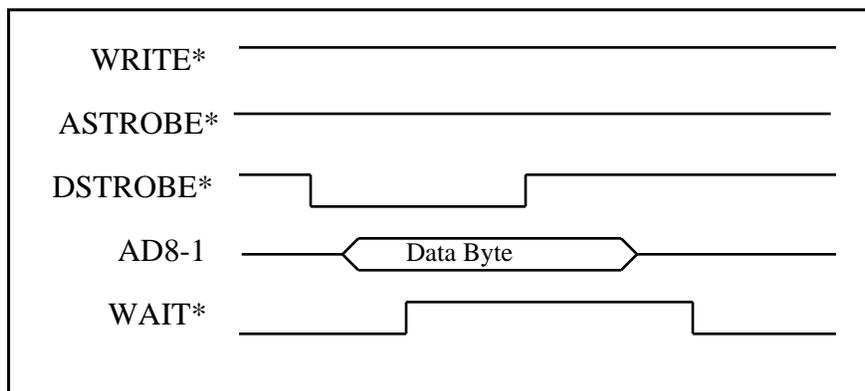


Figure 5. EPP Data Read Cycle

EPP defines five parallel port registers in addition to the three registers of the standard parallel port. These registers are used to automatically place the address or data information on the parallel port data lines and then generate the address strobe and data strobe signals automatically. The parallel port register set for EPP is shown in Table 4. The parallel port data register, status register, and control register have the same bit assignments as described for standard parallel ports.

Data written to the Auto Address Strobe register is placed on the parallel port data lines followed by an automatic active low pulse on the ASTROBE\* line. Data written to any of the Auto Data Strobe registers is placed on the parallel port data lines followed by an automatic active low pulse on the DSTROBE\* line. When one of the Auto Data Strobe registers is read, the DSTROBE\* line is pulsed and the value on the parallel port data lines is returned.

The microprocessor bus architecture of the EPP standard makes it ideal for communicating directly to peripheral hardware; it acts like a mini-expansion port aimed at accessing intelligent peripherals which are controlled with register accesses.

Table 4. EPP Mode Parallel Port Register Map

Register	Read or Write	Register Offset
Parallel Port Data Register	Write	0
Parallel Port Status Register	Read	1
Parallel Port Control Register	Read/Write	2
Auto Address Strobe Register	Read/Write	3
Auto Data Strobe Register	Read/Write	4
Auto Data Strobe Register	Read/Write	5
Auto Data Strobe Register	Read/Write	6
Auto Data Strobe Register	Read/Write	7

## Extended Capabilities Port – ECP

ECP is an extension to the standard parallel port developed by Microsoft and Hewlett Packard. The specification defines automatic hardware handshaking, command and data cycles, and DMA transfers to a FIFO location. The handshaking signals for data transfers have the same timing relationships as defined for standard parallel ports. Table 1 shows the parallel port signal line descriptions for an ECP parallel port. ECP claims maximum transfer rates of 2.4 Mbytes/s when performing DMA transfers on an ISA computer.

ECP defines data transfers from the host computer to the peripheral as forward transfers. Figure 6 shows an ECP forward transfer. In the forward direction, the host will drive HostClk low when data is available on the parallel port lines, and the peripheral will drive PeriphAck high after it accepts the data.

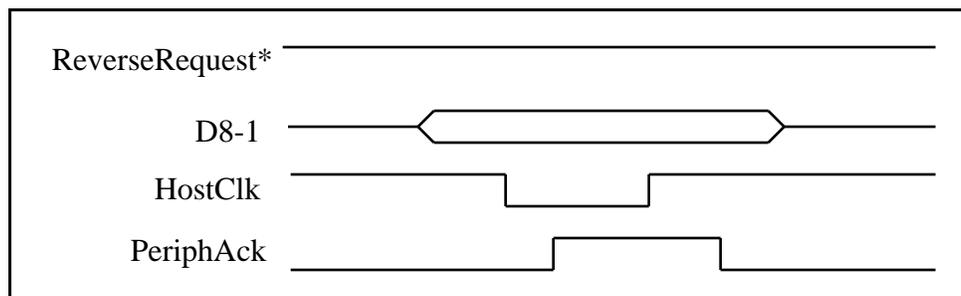


Figure 6. ECP Forward Transfer

Data transfers from the peripheral to the host computer are called reverse transfers. Figure 7 shows an ECP reverse direction transfer. In the reverse direction, the peripheral will drive `PeriphClk` low when data is available on the parallel port lines, and the host will drive `HostAck` low after it has accepted the data.

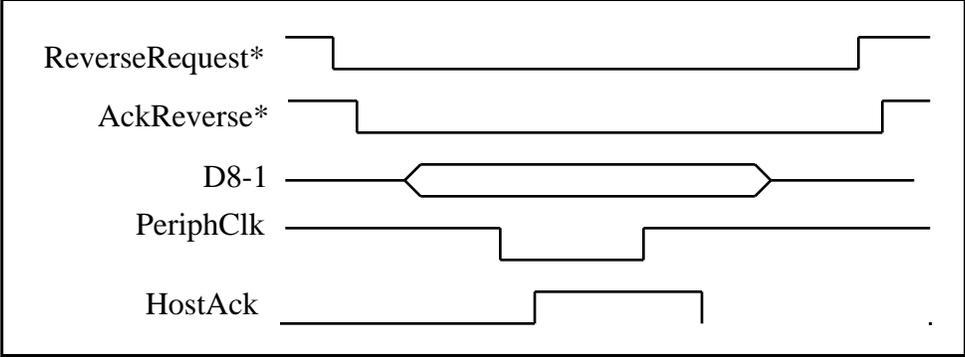


Figure 7. ECP Reverse Transfer

ECP defines two types of address cycles. The first is channel addressing which is used to address registers in a peripheral. The second is part of a simple data compression scheme. A run length count is specified during the address cycle to indicate the number of times the next data byte received is to occur in the data buffer.

Table 5. ECP Mode Parallel Port Register Map

Register	Offset	7	6	5	4	3	2	1	0
Data Register	0	D7	D6	D5	D4	D3	D2	D1	D0
Status Register	1	BUSY*	ACK*	PError	Select	FAULT*	IRQ*	Reserved	Reserved
Control Register	2	Auto Strobe	Reserved	Direction	IRQEN	SelectIn	INIT*	AUTOFD	STROBE
ECP Address FIFO Register	0	dType	A6	A5	A4	A3	A2	A1	A0
ECP Data FIFO Register	400	D7	D6	D5	D4	D3	D2	D1	D0
ECP Data FIFO Upper Register (optional)	401	D15	D14	D13	D12	D11	D10	D9	D8
Test FIFO Register (ECP FIFO Mode)	400	D7	D6	D5	D4	D3	D2	D1	D0
Test FIFO Upper Register (ECP FIFO Mode)	401	D15	D14	D13	D12	D11	D10	D9	D8
Configuration Register A (ECP Config. Mode)	400	impID3	impID2	impID1	impID0	Reserved	Reserved	Reserved	Reserved
Configuration Register B (ECP Config. Mode)	401	compress	intrValue	intrLine2	intrLine1	intrLine0	dmaCh2	dmaCh1	dmaCh0
Extended Control Register	402	mode2	mode1	mode0	ErrIntr En*	dmaEn	service Intr	full	empty

ECP defines six registers in addition to the three standard parallel port registers. These new registers automatically place the address or data information on the parallel port data lines and then generate the parallel port handshaking signals. The parallel port register set for ECP is shown in Table 5. The ECP Address and Data FIFOs contain at least 16 bytes and are used in both the forward and reverse directions for smooth data flow and improved data transfer rates. Data written to the ECP Address FIFO register is automatically transmitted on the parallel port. The ECP Data FIFO Register is used to transfer data between the host and peripheral.

The goal of ECP is to improve the parallel port for plug-and-play and Windows environments by making the interface bidirectional and increasing performance.

# The IEEE 1284 Standard

The IEEE 1284 standard defines an interface compatible with several distinct operating modes including EPP, ECP, a Compatibility mode for unidirectional communication with existing SPP ports, and Nibble and Byte modes for bidirectional communication with existing unidirectional and bidirectional ports.

## Negotiation

All devices in an IEEE 1284 system power up in Compatibility mode; a device may be designed to use one or several of the IEEE 1284 operating modes. Using a negotiation sequence, the host obtains a Device/ID code from the peripheral and then selects a compatible operating mode. The host performs the following steps to execute an IEEE 1284 negotiation sequence. See Figure 8.

- Place the IEEE 1284 8-bit extensibility code on the data lines, see Table 6.
- Assert the SelectIn\* line high and the AUTOFD\* line low.
- The peripheral will then drive PError high, ACK\* low, FAULT\* high and Select high if it is IEEE 1284 compliant. If the peripheral does not drive these lines, the host must assume that it is not 1284 compliant and treat it as such.
- Drive STROBE\* low.
- Drive STROBE\* high, AUTOFD\* high.
- The peripheral will then drive PError low, FAULT\* low, Select high if the extensibility code matched a mode that it offers.
- The peripheral will drive ACK\* high indicating the status lines are valid.

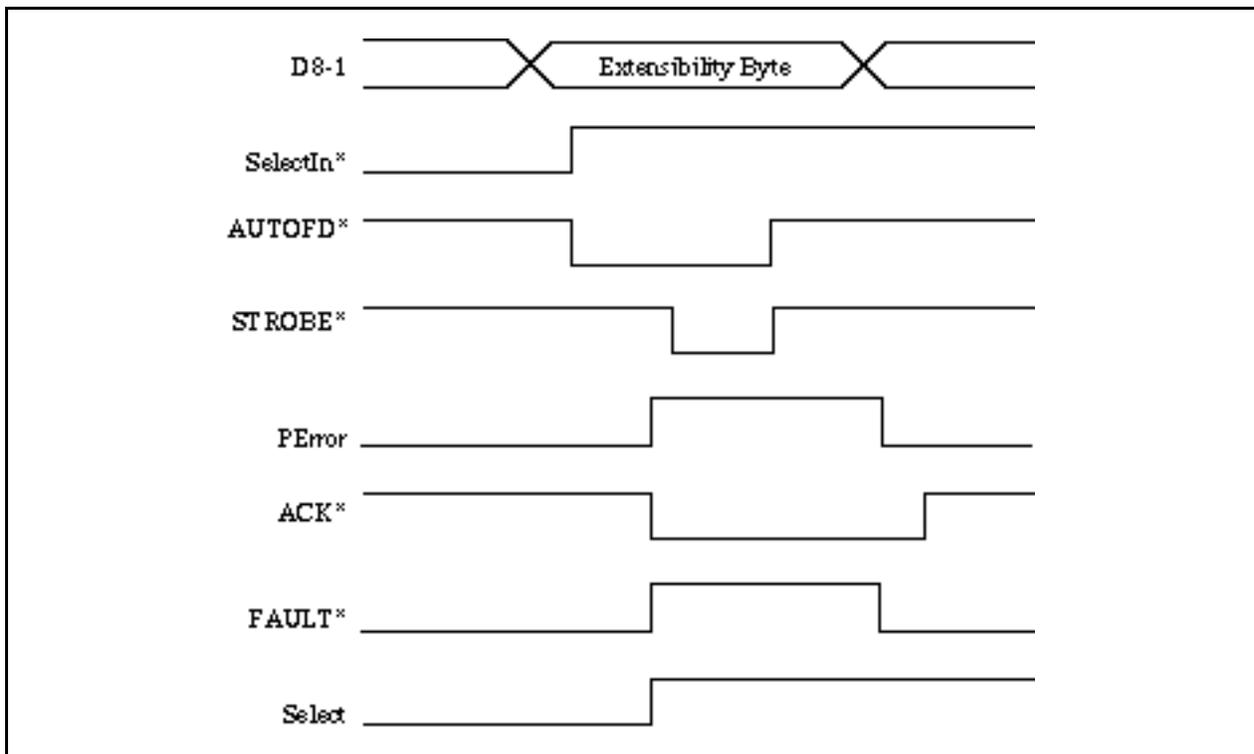


Figure 8. IEEE 1284 Negotiation

Table 6. 1284 Extensibility Request Bytes

Extensibility Byte	Definition	Description
1000 0000	Request Extensibility Link	This byte is used to add a second extensibility request byte to the negotiation phase. This allows for mode modes in the future.
0100 0000	Request EPP Mode	
0010 0000	Request ECP Mode with RLE	ECP mode with run-length encoding (RLE) data decompression
0001 0000	Request ECP Mode	ECP mode without data decompression
0000 1000	Reserved	Reserved for future use
0000 0100	Request Device ID using Nibble Mode	Receive the Device ID a nibble at a time across the status lines
0000 0101	Request Device ID using Byte Mode	Receive the Device ID a byte at a time across the data lines.
0001 0100	Request Device ID using ECP Mode without RLE	Receive the Device ID without ECP data compression
0011 0100	Request Device ID using ECP Mode with RLE	Receive the Device ID with ECP data compression
0000 0010	Reserved	Reserved for future use
0000 0001	Byte Mode Reverse Channel Transfer	Use the data lines bidirectionally to send data from the peripheral to the host.
0000 0000	Nibble Mode Reverse Channel Transfer	Use the parallel port status lines to send data one nibble at a time from the peripheral to the host.

To exit the Nibble, Byte, or one of the ECP modes, the host sets SelectIn\* low. To exit the EPP mode, the host asserts the INIT\* line. In both cases, the peripheral device will reset into Compatibility Mode (unidirectional SPP operation).

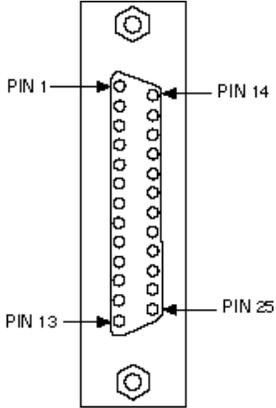
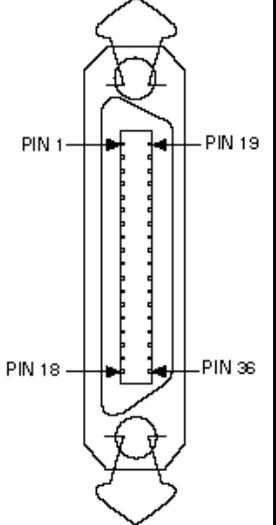
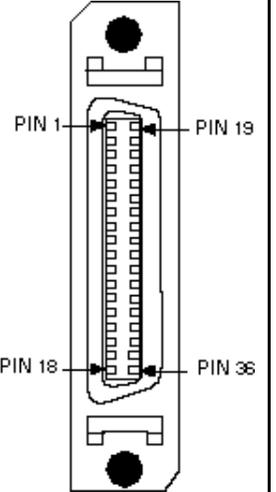
## Device ID

The Device ID is a length field followed by a string of ASCII characters defining the peripheral's characteristics and/or capabilities. This method was chosen because it does not require a central authority to assign device and manufacturer codes. In a Plug and Play system, the host must be able to determine that a device has been added, identify it, and automatically install the necessary device drivers. The host will use the Device ID sequence to recognize parallel port devices.

## Connectors and Cables

IEEE 1284 defines three interface connectors: 1284-A, 1284-B and 1284-C. The 1284-A connector is equivalent to the existing 25-pin DSub connector commonly used on the host side of the connection. The 1284-B connector is equivalent to the 36-pin Champ connector commonly used on the peripheral side of the connection. The 1284-C connection is a new 36-pin 0.050 centerline connector. The 1284 specification recommends this connector for both the host and peripheral sides of the connection. See Table 7 for the SPP pin assignments for each specified IEEE 1284 connector.

Table 7. SPP Signal Assignments for Defined IEEE 1284 Connectors

Pin Number	1284-A 25-pin Dsub	1284-B 36-pin Champ	1284-C 36-pin high density
			
1	STROBE*	STROBE*	BUSY
2	Data1	Data1	Select
3	Data2	Data2	ACK*
4	Data3	Data3	FAULT*
5	Data4	Data4	PError
6	Data5	Data5	Data1
7	Data6	Data6	Data2
8	Data7	Data7	Data3
9	Data8	Data8	Data4
10	ACK*	ACK*	Data5
11	BUSY	BUSY	Data6
12	PError	PError	Data7
13	Select	Select	Data8
14	AUTOFD*	AUTOFD*	INIT*
15	FAULT*	Not Defined	STROBE*
16	INIT*	Logic Ground	SelectIn*
17	SelectIn*	Chassis Ground	AUTOFD*
18	Ground	Peripheral Logic High	Host Logic High
19	Ground	Ground	Ground
20	Ground	Ground	Ground
21	Ground	Ground	Ground
22	Ground	Ground	Ground
23	Ground	Ground	Ground
24	Ground	Ground	Ground
25	Ground	Ground	Ground

<b>Pin Number</b>	<b>1284-A</b> 25-pin Dsub	<b>1284-B</b> 36-pin Champ	<b>1284-C</b> 36-pin high density
26		Ground	Ground
28		Ground	Ground
29		Ground	Ground
30		Ground	Ground
31		INIT*	Ground
32		FAULT*	Ground
33		Not Defined	Ground
34		Not Defined	Ground
35		Not Defined	Ground
36		SelectIn*	Peripheral Logic High

The 1284 standard also specifies a new "IEEE 1284 compliant" cable assembly, which shall meet specific electrical characteristics. These cables are designed for maximum performance and must be clearly labeled as IEEE 1284 compliant.

## Daisy Chaining

Under the IEEE 1284 Daisy Chain Specification, up to eight devices can be connected to a single parallel port. Each daisy chain device has two parallel port connectors – a host and a pass through connector. The host is connected to the host connector on the first device. The pass through connector of the first device is connected to the host connector of the next device, and so on. A device that does not support daisy chaining can be connected to the pass through connector of the last daisy chain device.

## Conclusion

The IEEE 1284 standard brings definition and higher performance to the PC parallel port. Parallel port devices will become easier to configure because new operating systems will bring Plug and Play to the parallel port with the Device/ID identification sequence. With these enhancements, the parallel port will be an even better low-cost, readily available I/O port on the PC.

## References

*IEEE Standard 1284-1994*

*IBM PS/2 Technical Reference, IBM*

*ECP Specification, Microsoft*

*1284 Daisy Chain Specification, DISCTEC*

*Plug and Play Parallel Port Devices Specification, Microsoft*