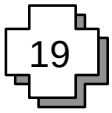

Enhanced Parallel Port

19.1 Introduction

The Centronics parallel port only allows data to be sent from the host to a peripheral. To overcome this the IEEE have published the 1284 standard which is entitled 'Standard Signaling Method for a Bi-directional Parallel Peripheral Interface for Personal Computers'. It allows for bi-directional communication and high communication speeds, while it is backwardly compatible with existing parallel ports.

The IEEE 1284 standard defines the following modes:

- Compatibility mode (forward direction only). This mode defines the transfer of data between the PC and the printer (Centronics mode, as covered in the two previous two chapters).
- Nibble mode (reverse direction). This mode defines how 4 bits are transferred, at a time, using status lines for the input data (sometimes known as Hewlett Packard Bitronics). The Nibble mode can thus be used for bi-directional communication, with the data lines being used as outputs. To input a byte thus requires two nibble cycles.
- Byte mode (reverse direction). This mode defines how 8 bits are transferred at a time.
- Enhanced Parallel Port (EPP). This mode defines standard bi-directional communications and is used by many peripherals, such as CD-ROMs, tape drives, external hard disks, and so on.

19.2 IEEE 1284 Data Transfer Modes

In the IEEE 1284 standard, the control and status signals for nibble, byte and EPP modes have been renamed. It also classifies the modes as forward (data goes from the PC), reverse (data is sent to the PC) and bi-directional. Both the compatibility and nibble modes can be implemented with all parallel ports (as the nibble mode uses the status lines and the compatibility mode only outputs data). Some parallel ports support input and output on the data lines and thus support the byte mode. This is usually implemented by the addition of a direction bit on the control register.

19.3 Compatibility mode

The compatibility mode was discussed in Chapters 17 and 18. In this mode the program sends data to the data lines and then sets the $\overline{\text{STROBE}}$ low and then high (see Figure 19.1). This then latches the data to the printer. The operations that the program does are:

1. Data is written to the data register.
2. The program reads from the status register to test to see if the BUSY signal is low (that is, the printer is not busy).
3. If the printer is not busy then the program sets the $\overline{\text{STROBE}}$ line active low.
4. The program then makes the $\overline{\text{STROBE}}$ line high by de-asserting it.

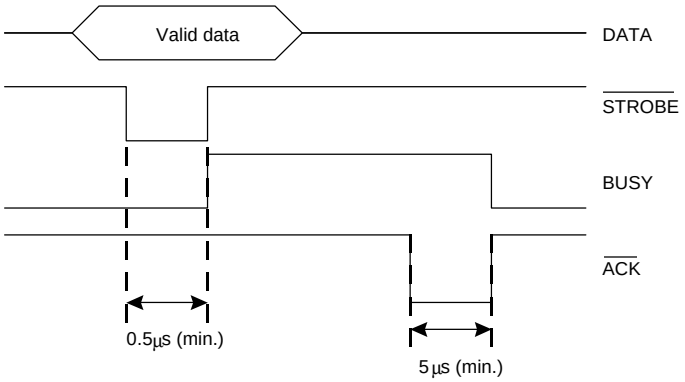

Figure 19.1 Compatibility mode transfer.

19.4 Nibble mode

This mode defines how 4 bits are transferred, at a time, using status lines for the input data (sometimes known as Hewlett Packard Bi-tronics). The Nibble mode can thus be used for bi-directional communication, with the data lines being used as outputs. To input a byte thus requires two nibble cycles.

As seen in Chapter 3 there are five inputs from the parallel port (BUSY, $\overline{\text{ACK}}$, PE, SELECT and $\overline{\text{ERROR}}$). The status of these lines can be found by simply reading the upper 5 bits of the status register. The BUSY, PE, SELECT and $\overline{\text{ERROR}}$ are normally used with $\overline{\text{ACK}}$ to interrupt the processor.

Table 19.1 defines the names of the signal in the nibble mode and Figure 19.2 shows the handshaking for this mode.

The nibble mode has the following sequence:

1. Host (PC) indicates that it is ready to receive data by setting HostBusy low.
2. The peripheral then places the first nibble on the status lines.
3. The peripheral indicates that the data is valid on the status line by setting PtrClk low.
4. The host then reads from the status lines and sets HostBusy high to indicate that it has received the nibble, but it is not yet ready for another nibble.
5. The peripheral sets PtrClk high as an acknowledgement to the host.
6. Repeat steps 1–5 for second nibble.

Table 19.1 Nibble mode signals.

Compatibility signal name	*Nibble mode name*	*In/out*	*Description*
$\overline{\text{STROBE}}$	$\overline{\text{STROBE}}$	O	Not used.
$\overline{\text{AUTO FEED}}$	HostBusy	O	Host nibble mode handshake signal. It is set low to indicate that the host is ready for nibble and set high when the nibble has been received.
$\overline{\text{SELECT INPUT}}$	1284Active	O	Set high when the host is transferring data.
$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	O	Not used.
$\overline{\text{ACK}}$	PtrClk	I	Indicates valid data on the status lines. It is set low to indicate that there is valid data on the control lines and then set high when the HostBusy going high.
BUSY	PtrBusy	I	Data bit 3 for one cycle then data bit 7.
PE	AckDataReq	I	Data bit 2 for one cycle then data bit 6.
SELECT	Xflag	I	Data bit 1 for one cycle then data bit 5.
$\overline{\text{ERROR}}$	$\overline{\text{DataAvail}}$	I	Data bit 0 for one cycle then data bit 4.
D0–D7	D0–D7		Not used.

These operations are software intensive as the driver requires to set and read the handshaking lines. This limits transfer to about 50 KBytes/s. Its main advantage is that it works with all printer ports because it uses the standard Centronics setup and is normally used in low-speed bi-directional operations, such as ADC adapters, reading data from switches, and so on.

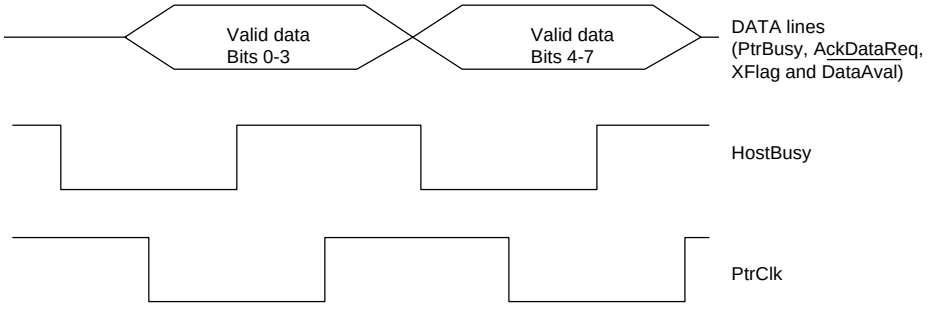

Figure 19.2 Nibble mode data transfer cycle.

19.5 Byte mode

The byte mode is often known as a bi-directional port and it uses bi-directional data lines. It has the advantage over nibble mode in that it only takes a single cycle to transfer a byte. Unfortunately, it is only compatible with newer ports. Table 19.2 defines the names of the signal in the nibble mode and Figure 19.3 shows the handshaking for this mode.

The byte mode has the following sequence:

1. The host (PC) indicates that it is ready to receive data by setting HostBusy low.
2. The peripheral then places the byte on the status lines.
3. The peripheral indicates that the data is valid on the status line by setting PtrClk low.
4. The host then reads from the data lines and sets HostBusy high to indicate that it has received the nibble, but it is not yet ready for another nibble.
5. The peripheral sets PtrClk high as an acknowledge to the host.
6. The host then acknowledges the transfer by pulsing HostClk.

Table 19.2 Byte mode signals.

Compatibility signal name	*Byte mode name*	*In/out*	*Description*
$\overline{\text{STROBE}}$	HostClk	O	Used as an acknowledgment signal. It is pulsed low after each transferred byte.
$\overline{\text{AUTO FEED}}$	HostBusy	O	It is set low to indicate that the host is ready for nibble and set high when the nibble has been received.
$\overline{\text{SELECT INPUT}}$	1284Active	O	Set high when the host is transferring data.
$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	O	Not used.
$\overline{\text{ACK}}$	PtrClk	I	Indicates valid data byte. It is set low to indicate that there is valid data on the data lines and then set high when the HostBusy going high.
BUSY	PtrBusy	I	Busy status (for forward direction).
PE	AckDataReq	I	Same as $\overline{\text{DataAvail}}$.
SELECT	Xflag	I	Not used.
$\overline{\text{ERROR}}$	$\overline{\text{DataAvail}}$	I	Indicates that there is reverse data available.
D0–D7	D0–D7	I/O	Input/output data lines.

19.6 EPP

The Enhanced Parallel Port (EPP) mode defines standard bi-directional communications and is used by many peripherals, such as CD-ROMs, tape drives, external hard disks, and so on.

The EPP protocol provides four types of data transfer cycles:

1. Data read and write cycles. These involve transfers between the host and the peripheral.
2. Address read and write cycle. These pass address, channel, or command and control information.

Table 19.3 defines the names of the signals in the nibble mode and Figure 19.4 shows the handshaking for this mode. The $\overline{\text{WRITE}}$ signal occurs automatically when the host writes data to the output lines.

The data write cycle has the following sequence:

1. Program executes an I/O write cycle to the base address port + 4 (EPP Data Port); see Table 19.4. Then the following occurs with hardware:
2. The $\overline{\text{WRITE}}$ line is set low which puts the data on the data bus.
3. The $\overline{\text{DATASTB}}$ is then set low.
4. The host waits for peripheral to set the $\overline{\text{WAIT}}$ line high.
5. The $\overline{\text{DATASTB}}$ and $\overline{\text{WRITE}}$ are then set high and the cycle ends.

The important parameter is that it takes just one memory mapped I/O operation to transfer data. This gives transfer rates of up to 2 million bytes per second. While it is not as fast as a peripheral transferring over the ISA, it has the advantage that the peripheral can transfer data at a rate that is determined by the peripheral (ISA has a fixed transfer rate).

19.6.1 EPP registers

Several extra ports are defined; these are the EPP address register and the EPP data register. The EPP address register has an offset of 3 bytes from the base address and the EPP data register is offset by 4 bytes. Table 19.4 defines the registers.

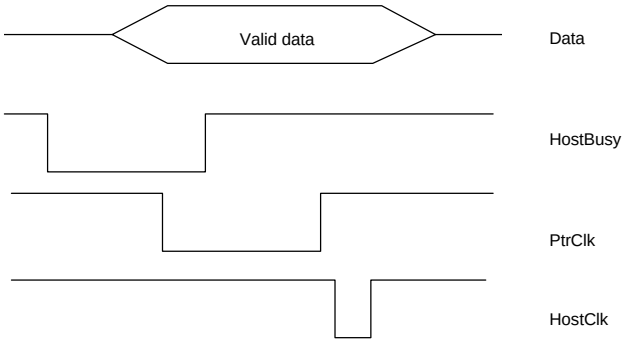

Figure 19.3 Byte mode data transfer cycle.

Table 19.3 EPP mode signals.

Compatibility signal name	*EPP mode name*	*In/out*	*Description*
$\overline{\text{STROBE}}$	$\overline{\text{WRITE}}$	O	A low for a write operation while a high indicates a read operation.
$\overline{\text{AUTO FEED}}$	$\overline{\text{DATASTB}}$	O	Indicates a data read or write operation.
$\overline{\text{SELECT INPUT}}$	$\overline{\text{ADDRSTROBE}}$	O	Indicates an address read or write operation.
$\overline{\text{INIT}}$	$\overline{\text{RESET}}$	O	Peripheral reset when low.
$\overline{\text{ACK}}$	$\overline{\text{INTR}}$	I	Peripheral sets this line low when it wishes to interrupt to the host.
BUSY	$\overline{\text{WAIT}}$	I	When it is set low it indicates that it is valid to start a cycle, else if it is high then it is valid to end the cycle.
PE	User defined	I	Can be set by each peripheral.
SELECT	User defined	I	Can be set by each peripheral.
$\overline{\text{ERROR}}$	User defined	I	Can be set by each peripheral.
D0–D7	AD0–AD7	I/O	Bi-directional address and data lines.

Table 19.4 EPP register definitions.

Port Name	*I/O address*	*Read/ write*	*Description*
Data register	BASE_AD	W	
Status register	BASE_AD +1	R	
Control register	BASE_AD +2	W	
EPP address port	BASE_AD+3	R/W	Generates EPP address read or write cycle.
EPP data port	BASE_AD+4	R/W	Generates EPP data read or write cycle.

19.7 ECP

The extended capability port (ECP) protocol was proposed by Hewlett Packard and Microsoft as an advanced mode for communication with printer and scanner type peripherals. It provides a high performance bi-directional data transfer between a host and a peripheral.

ECP provides the following cycle types in both the forward and reverse directions:

- Data cycles.
- Command cycles. The command cycles are divided into 2 types: Run Length Count and Channel address.

It supports several enhancements, such as:

- Run Length Encoding (RLE). This allows for real-time compression with compression ratios of up to 64:1. RLE allows multiple occurrences of a sequence to be sent as a short code. Typically graphics images and video information have long sequences of the same data.
- Forward and reverse channel FIFOs.
- DMA.
- Programmed I/O with a standard addressing structure.
- Channel addressing. This supports many logical devices connected to a single parallel port connection. Each of the devices can have its own connection. Typically a FAX, modem, printer and CD-ROM drive could be connected to a single parallel port connection.

In the ECP protocol the signal lines have been renamed to be consistent with an ECP handshake. Table 19.5 describes these signals.

Table 19.5 ECP mode signals.

Compatibility signal name	*ECP mode name*	*In/out*	*Description*
$\overline{\text{STROBE}}$	HostClk	O	Along with PeriphAck it is used to transfer data or address information in the forward direction.
$\overline{\text{AUTO FEED}}$	HostAck	O	Gives Command/Data status in the forward direction.
$\overline{\text{SELECT INPUT}}$	1284Active	O	Set to a high when host is a transfer mode.
$\overline{\text{INIT}}$	$\overline{\text{ReverseRequest}}$	O	Active low puts the channel into the reverse direction.
$\overline{\text{ACK}}$	PeriphClk	I	Along with HostAck it is used to transfer data in the reverse direction.
BUSY	PeriphAck	I	Along with HostClk it is used to transfer data or address information in the forward direction.
PE	nAckReverse	I	Active low to acknowledge nReverseRequest.
SELECT	Xflag	I	Extensibility flag.
$\overline{\text{ERROR}}$	nPeriphRequest	I	Active low to indicate the availability of reverse data.
D0–D7	Data[8:1]	I/O	Data lines.

Figure 19.4 shows two forward transfer cycles, a data cycle followed by a command cycle. An active HostAck signal indicates that it is a data cycle, else it is a command cycle. In the command cycle the data byte represents either:

- RLE count. If the most significant bit of the data byte is a 0 then the rest of the bytes represent the Run Length Count (0–127).
- Channel address. If the most significant bit of the data byte is a 1 then the rest of the bytes represent a channel address.

Forward Transfer phase is as follows:

1. Host sets the HostAck signal to identify if the transfer is a data or a command cycle Host puts its data on the data bus. (A).
2. Host sets HostClk low to indicate valid data. (B).
3. The peripheral sets PeriphAck high to acknowledge the transfer. (C).
4. Host sets HostClk high which clocks data into the peripheral. (D).
5. Peripheral sets PeriphAck low to indicate that it is ready for more data. (E).

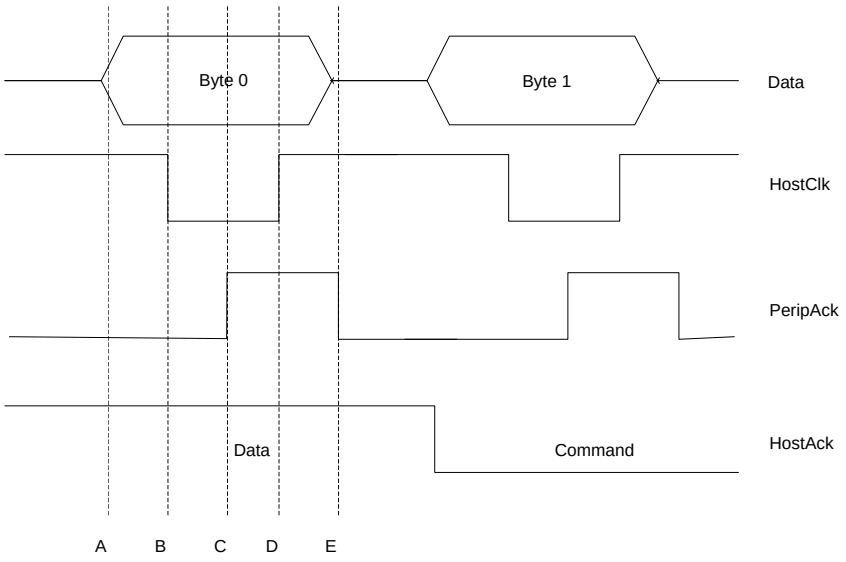

Figure 19.4 ECP forward data and Command cycle.

The reverse transfer is as follows:

1. Host identifies a reverse channel transfer by setting nReverseRequest low.
2. Peripheral acknowledges this by setting nAckReverse low.
3. Peripheral puts its data on the data bus and sets PeriphAck high.
4. Peripheral sets PeriphClk low to indicate valid data.
5. Host acknowledges this by setting HostAck high.
6. Peripheral sets PeriphClk high which clocks data into the host.
7. Host sets HostAck low to indicate that it is ready for the next transfer.

ECP registers

ECP mode has a standard set of I/O registers using a number of modes (as defined in Table 19.6). The additional registers have been added at an offset of 400h from the base port, as given in Table 19.7. Note that only extra three I/O addresses have been added. It can be seen from Figure 19.5 that the ECP driver uses the address 378h to 37Ah and 778h to 77Ah (offset from the base register by 400h). The configuration of the ECP is setup using the ECR register.

Table 19.6 ECR Register Modes

Mode	*Description*
000	SPP mode
001	Bi-directional mode (Byte mode)
010	Fast Centronics
011	ECP Parallel Port mode
100	EPP Parallel Port mode (note 1)
101	(reserved)
110	Test mode
111	Configuration mode

Table 19.7 ECP register description.

Offset	*Read/Write*	*ECP Mode*	*Function*
000	R/W	000–001	Data register
000	R/W	011	ECP address FIFO
001	R/W	All	Status register
002	R/W	All	Control register
400	R/W	010	Parallel port data FIFO
400	R/W	011	ECP data FIFO
400	R/W	110	Test FIFO
400	R	111	Configuration register A
401	R/W	111	Configuration register B
402	R/W	All	Extended control register

19.8 1284 Negotiation

The negotiation mode allows the host to determine the attached peripherals and the method used to control them. It has been designed so that is does not affect older devices, which do not respond to the negotiation phase.

In the negotiation phase, the host places a request on the data lines, such as:

- Setup a mode.
- Request a device ID.

It then goes into a negotiation sequence and uses an extensibility byte, as defined in Table 19.8.

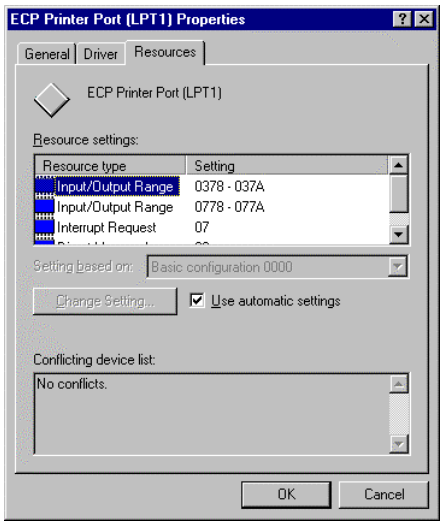

Figure 19.5 ECP register settings.

The negotiation is as follows:

1. Host puts the required extensibility byte on the data bus.
2. Host indicates the negotiation phase by setting nSelectIn high and nAutoFeed.
3. Compliant peripherals respond by setting nAck low, nError, PE high and Select high.
4. Host sets nStrobe low which clocks the extensibility byte into the peripheral.
5. Host sets nStrobe high and nAutoFeed high to acknowledge the transfer.
6. Peripheral then sets PE low, nError low and Select high.
7. Peripheral sets nAck high to signal that the negotiation sequence is over.

Table 19.8 Extensibility byte bit values.

Bit	*Description*	*Valid bit values*
8	Request Extensibility Link	1000 0000
7	Request EPP Mode	0100 0000
6	Request ECP Mode with RLE	0011 0000
5	Request ECP Mode without RLE	0001 0000
4	Reserved	0000 1000
3	Request Device ID	Return data using mode: Nibble Mode 0000 0100 Byte Mode 0000 0101 ECP Mode without RLE 0001 0100 ECP Mode with RLE0011 0100
2	Reserved	0000 0010
1	Byte Mode	0000 0001
none	Nibble Mode	0000 0000

19.9 Exercises

19.9.1 Discuss the different modes which the parallel port can be put into.

19.9.2 Why is nibble mode hardly ever used in PC systems? Show the handshaking that occurs in this mode.

19.9.3 Discuss the operation of the handshaking signals used in the compatibility mode.

19.9.4 Explain how, in ECP mode, several devices can be connected onto the same bus.

19.9.5 Explain how, in ECP mode, data can be compressed. Give an example of the type of data that is likely to have compression rates.

19.9.6 Explain how ECP mode uses the standard printer port addresses and extra registers.

19.9.7 Explain how ECR register is used to setup the mode in which the ECP mode is used.