Interfacing the Enhanced Parallel Port

Table of Contents

EPP - Enhanced Parallel Port

The Enhanced Parallel Port (EPP) was designed in a joint venture between Intel, Xircom & Zenith Data Systems. EPP Ports were first specified in the EPP 1.7 standard, and then later included in the IEEE 1284 Standard released in 1994. EPP has two standards, EPP 1.7 and EPP 1.9. There are differences between the two standards which may affect the operation of devices. This is further discussed latter. EPP has a typical transfer rate in the order of 500KB/S to 2MB/S. This is achieved by allowing the hardware contained in the port to generate handshaking, strobing etc, rather that have the software do it, which was the case with Centronics.

For the hobbyist, EPP is more commonly used than ECP. EPP differs from ECP by the fact that the EPP Port generates and controls all the transfers to and from the peripheral. ECP on the other hand requires the peripheral to negotiate a reverse channel and control the handshaking. This is harder to achieve with common glue logic, thus really requires a dedicated controller or ECP Peripheral Chip.

EPP Hardware Properties

When using EPP mode, a different set of tasks and labels are assigned to each line. These are listed below in Table 4. It's very common to see both the SPP and EPP names interchanged in Parallel Port Data Sheets and Literature. This can make it very hard to focus on what is exactly happening. Therefore all the documentation here will use the EPP names.

Pin	SPP Signal	EPP Signal	In/Out	Function
1	Strobe	Write	Out	A low on this line indicates a Write, High indicates a Read
2-9	Data 0-7	Data 0-7	In-Out	Data Bus. Bi-directional
10	Ack	Interrupt	In	Interrupt Line. Interrupt occurs on Positive (Rising) Edge.
11	Busy	Wait	In	Used for handshaking. A EPP cycle can be started when low, and finished when high.
12	Paper Out / End	Spare	In	Spare - Not Used in EPP Handshake
13	Select	Spare	In	Spare - Not Used in EPP Handshake
14	Auto Linefeed	Data Strobe	Out	When Low, indicates Data transfer
15	Error / Fault	Spare	In	Spare - Not used in EPP Handshake
16	Initialize	Reset	Out	Reset - Active Low
17	Select Printer	Address Strobe	Out	When low, indicates Address transfer
18-25	Ground	Ground	GND	Ground

Table 1. Pin Assignments For Enhanced Parallel Port Connector.

Paper Out, Select and Error are not defined in the EPP handshake. These lines can be utilised in any way by the user. The status of these lines can be determined at anytime by viewing the SPP Status Register. Unfortunately there are no spare output's. This can become a hassle regularly.

The EPP Handshake

In order to perform a valid exchange of data using EPP we must follow the EPP handshake. As the hardware does all the work, this handshake only requires to be used for your hardware and not for software as the case with SPP. To initiate an EPP cycle your software needs to perform only one I/O operation to the relevant EPP Register. Details on this, later.

EPP Data Write Cycle

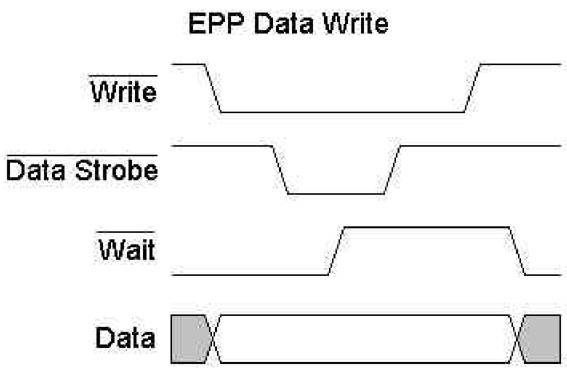

Figure 1. Enhanced Parallel Port Data Write Cycle.

1. **Program writes to EPP Data Register. (Base + 4)**
2. nWrite is placed low. (Low indicates write operation)
3. **Data is placed on Data Lines 0-7.**
4. nData Strobe is asserted if Wait is Low (O.K. to start cycle)
5. **Host waits for Acknowledgment by nWait going high (O.K. to end cycle)**
6. nData Strobe is de-asserted.
7. EPP Data Write Cycle Ends.

EPP Address Write Cycle

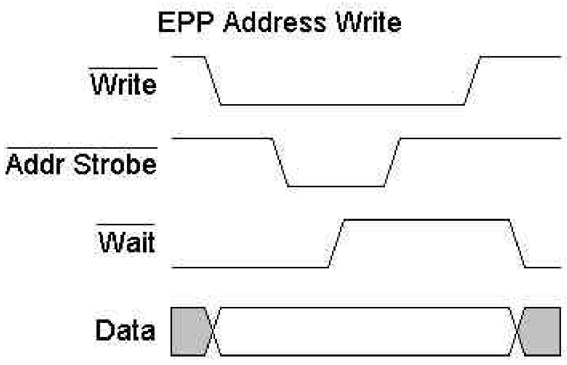

Figure 2. Enhanced Parallel Port Address Write Cycle.

1. **Program writes address to EPP's Address Register (Base + 3)**
2. Write is placed low. (Low indicates write operation)
3. **Address is placed on Data Lines 0-7.**
4. nAddress Strobe is asserted if Wait is Low (O.K. to start cycle)
5. **Host waits for Acknowledgment by wait going high (O.K. to end cycle)**
6. nAddress Strobe is De-asserted.
7. **EPP Address Write Cycle Ends.**

EPP Data Read Cycle

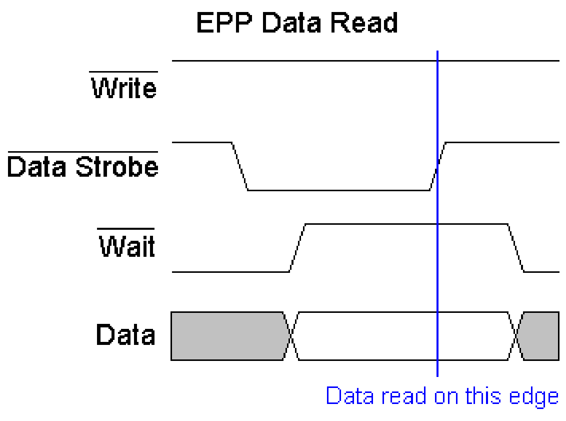

Figure 3. Enhanced Parallel Port Data Read Cycle.

1. **Program reads EPP Data Register. (Base + 4)**
2. nData Strobe is asserted if Wait is Low (O.K. to start cycle)
3. **Host waits for Acknowledgment by nWait going high**
4. Data is read from Parallel Port Pins.
5. **nData Strobe is de-asserted.**
6. EPP Data Read Cycle Ends.

EPP Address Read Cycle

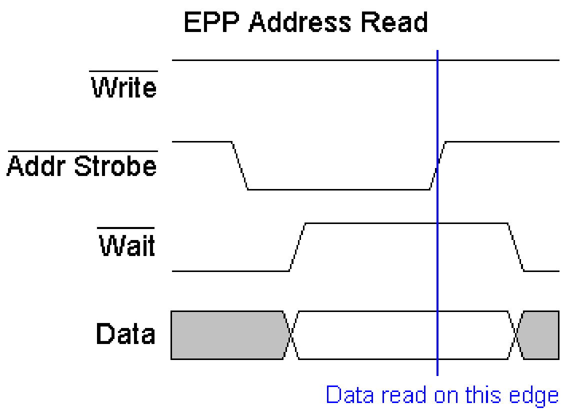

Figure 4. Enhanced Parallel Port Address Read Cycle.

1. **Program reads EPP Address Register. (Base + 3)**
2. nAddr Strobe is asserted if Wait is Low (O.K. to start cycle)
3. **Host waits for Acknowledgment by nWait going high**
4. Data is read from Parallel Port Pins.
5. **nAddr Strobe is de-asserted.**
6. EPP Address Read Cycle Ends.

Note If implementing EPP 1.7 Handshake (Pre IEEE 1284) the Data and Address Strobes can be asserted to start a cycle regardless of the wait state. EPP 1.9 will only start a cycle once wait is low. Both EPP 1.7 and EPP 1.9 require the wait to be high to finish a cycle.

The EPP's Software Registers.

The EPP Port also has a new set of registers. However 3 of them have been inherited from the Standard Parallel Port. On the next page is a table showing the new and existing registers.

Address	Port Name	Read/Write
Base + 0	Data Port (SPP)	Write
Base + 1	Status Port (SPP)	Read
Base + 2	Control Port (SPP)	Write
Base + 3	Address Port (EPP)	Read/Write
Base + 4	Data Port (EPP)	Read/Write
Base + 5	Undefined (16/32bit Transfers)	-
Base + 6	Undefined (32bit Transfers)	-
Base + 7	Undefined (32bit Transfers)	-

Table 2 EPP Registers

As you can see, the first 3 addresses are exactly the same than the Standard Parallel Port Register and behave in exactly the same way. Therefore if you used a Enhanced Parallel Port, you can output data to Base + 0 in exactly the same fashion than you would if it was a Standard Parallel Port (SPP). If you were to connect a printer, and use compatibility mode then you would have to check to see if the port is busy and then assert & de-assert the strobe using the Control and Status Port, then wait for the Ack.

If you wish to communicate with a EPP compatible device then all you have to do, is place any data you wish to send in the EPP Data Register at Base + 4 and the card will generate all the necessary handshaking required. Likewise if you wish to send an address to your device, then you use the EPP Address Register at offset +3.

Both the EPP Address Register and the EPP Data Register are read / write, thus to read data from your device, you can use the same registers. However the EPP Printer Card has to initiate a read Cycle as both the nData Strobe and nAddress Strobe are outputs. Your device can signal a read request via the use of the interrupt and have your ISR perform the Read Operation.

The Status Port has one little modification. Bit 0, which was reserved in the SPP register set, now becomes the EPP Time-out Bit. This bit will be set when an EPP time-out occurs. This happens when the nWait line is not deasserted within approximately 10uS (depending upon the port) of the IOW or IOR line being asserted. The IOW and IOR are the I/O Read and Write lines present on the ISA Bus.

The EPP mode is very depended of the ISA bus timing. When a read cycle is performed, the port must undertake the appropriate Read/Write handshake and return the data in that ISA cycle. Of course this doesn't occur within one ISA cycle, thus the port uses the IOCHRDY (I/O Channel Ready) on the ISA bus to introduce wait states, until the cycle completes. Now imagine if a EPP Read or Write is started with no peripheral connected? The port never gets an acknowledgment (nWait), thus keeps sending requests for wait states, and your computer locks up. Therefore the EPP implements a type of watchdog, which times out after approximately 10uS.

The three registers, Base + 5, Base + 6 and Base + 7 can be used for 16 and 32 bit read/write operations if your port supports it. This can further reduce your I/O operations. The Parallel Port can only transport 8 bits at a time, thus any 32 or 16 bit word written to the Parallel Port will be split into byte size blocks and sent via the Parallel Port's 8 data lines.

EPP Programming Considerations.

EPP only has two main registers and a Time-out Status Flag, What could there possibly be to set up?

Before you can start any EPP cycles by reading and writing to the EPP Data and Address Ports, the port must be configured correctly. In the idle state, an EPP port should have it's nAddress Strobe, nData Strobe, nWrite and nReset lines inactive, high. Some ports require you to set this up before starting any EPP Cycle. Therefore our first task is to manually initialise these lines using the SPP Registers. Writing XXXX0100 to the control port will do this.

On some cards, if the Parallel Port is placed in reverse mode, a EPP Write cycle cannot be performed. Therefore it is also wise to place the Parallel Port in forward mode before using EPP. Clearing Bit 5 of the Control Register should result in an more enjoyable programming session, without tearing your hair out.

The EPP Time-out bit we have already discussed. When this bit is set, the EPP port may not function correctly. A common scenario is always reading 0xFF from either the Address or Data Cycles. This bit should be cleared for reliable operation, and constantly checked.