Website Quick Reference

- Lakeview Research home
- Contact Info
  - News for the Press
- About Jan

Books

- *USB Complete*
  - *Serial Port Complete*
- *Parallel Port Complete*
- *The Microcontroller Idea Book*
- *Making Printed Circuit Boards*

- **How to Order**

ResourcePages

- USB
  - Serial PortS
- Parallel Port
  - 8052-Basic Microcontrollers
- PC Boards
  - Magazine Articles

Jan's Parallel Port FAQ

Answers to some frequently asked questions about the PC's parallel port

by Jan Axelson

This FAQ contains my answers to parallel-port questions that I have answered in newsgroups, forums, or direct email. Because many of the same topics come up again and again, I've collected the most common ones in this document.

Most are specific, technical questions relating to programming, interfacing, or using the parallel port. For more basic information about the parallel port and its many modes, see the other information and links at Lakeview Research's web site: http://www.lvr.com

Thanks to the questioners for some thought-provoking queries!

Latest update: 5/3/99

Jan Axelson
jan@lvr.com

General Interfacing

Q: When I write a byte to a port address, I don't see the value at the connector/cable/peripheral.

A: If you don't see what you expect at the other end, the reason is probably one of the following

- You're writing to the wrong address. Remember that 378 in hexadecimal is 888 in decimal.
- You're looking on the wrong pin. Verify the wiring of any cable you're using.
- You're running NT and trying to access a port directly. NT requires a kernel-mode driver.
- External circuits connected to the port are preventing the bits from toggling. Disconnect the circuits and measure again.
- You forgot that Control port bits 0, 1, and 3 are inverted between the port register and the connector pins. When you write to the Control port, these bits will be the inverse of what you write.
- The port supports PS/2 mode and you've written 1 to Control port, bit 5, disabling the Data port's outputs. Write 0 to this bit to enable the outputs.
- You're trying to connect two PCs' parallel ports and you aren't using the required LapLink-type cable and software that transmits four bits at a time.
- A low-level Windows driver has prevented access to the port. This is rarely a problem with the parallel port, however.

Q: When I read a port, the value I read doesn't match the value at the connector/cable/peripheral.

A: If you don't read what you expect, the reason is probably one of the following:

- You're reading the wrong address. Remember that 378 in hexadecimal is 888 in decimal.
- You're looking on the wrong pin. Verify the wiring of any cable you're using.
- You're running NT and trying to access a port directly. NT requires a kernel-mode driver.
- You forgot that Control port bits 0, 1, and 3 and Status port bit 7 are inverted between the port register and the connector pins. When you read these ports, these bits will be the inverse of the logic states at the connector.
- You're reading the data port and your port doesn't support PS/2 mode. You can't disable the data outputs, and will read the last value written to the port, rather than the logic state at the connector.
- The port supports PS/2 mode and you haven't written 1 to Control port, bit 5, disabling the Data port's outputs. Write 1 to this bit to enable reading external signals on the data lines.
- A low-level Windows driver has prevented access to the port. This is rarely a problem with the parallel port, however.

Q: I need to make an interface to emulate a printer that is on-line and with paper. I have a program that needs three printers, but two of the outputs don't interest me and if I don't have the printers connected, the program doesn't work.

A: To fool the software into thinking that a printer is connected, try this: Tie -Error (pin 15) high (+5V). Tie PaperEmpty (pin 12) low (0V). Tie Busy (pin 11) low (0V). Matthew Chapman also recommends tying -Select (pin 17) low and pin 12 (Select In) high.

Also from Matthew Chapman: In VB, do not use ' Printer.Print Chr(xxx) ' followed by ' Printer.EndDoc ' (to start printing), because that will start endlessly spooling and won't reach the final stage, which is to output the byte for printing. Use ' Text1.Text = "" ' and ' Text2.Text = Chr(255) ' in the ' Form_Load ' procedure and ' Text1.SelPrint (Printer.hDC) ' followed by ' Text2.SelPrint (Printer.hDC) ' in the ' Command1_Click ' procedure to output the byte value of 255 to the printer port (regardless of which printer driver you are using) whenever you click on the command button.

Q: I'm trying to input a stream of data into pin 12 but I get garbage on other computers and mine works fine. I'm using the output pin number 8 on a 74F299 to serially load the data into the computer. I use the data and control ports for control signals, and these work fine.

A: Sending data at high speeds over a long cable can cause problems. The components used to implement the parallel port can vary from system to system, so a marginal circuit may work with some ports & not others. You might want to try these:

- Use a slower logic family (LSTTL, HCTMOS). (Decreases transmission-line effects.)

- Buffer the inputs and outputs that connect to the cable. Use a 74HC244 or 74LS244 buffer/driver or similar. (For isolation, strong drivers, Schmitt-trigger inputs.)
- Slow the clock that controls the bit transmissions.
- Use a short cable (6 feet or less).

Q: The status register on the parallel port is used for input. Why do I read a logic "1" wit a logic probe at these pins on the DB25 connector?

A: This just means that the inputs have pull-up resistors. Not unusual, and not a problem.

Q: Is the parallel port based on the 8255 Programmable Peripheral Interface?

A: There's no relation between the 8255 chip and the parallel port. The original IBM PC did contain an 8255, but it had nothing to do with the parallel-port circuits, which used ordinary logi gates and latches.

Q: We have a device that sends data to a line printer. We'd like to send the data to a PC instead. How can we do this?

A: The easiest solution is to buy a parallel-to-serial converter and read the data at the new PC's serial port. Converters are available from B &B, Jameco, JDR, and other sources. Be sure the converter can convert parallel input to serial output.

Q: I am in need of a parallel to serial to parallel conversion for a data acquisition application. The data acquisition is controlled through a centronics parallel port. The application requires a wireless connection between the DAQ and conputer and this is to b accomplished using 19.2kbps wireless modems that is RS-232 connected. Commercially available products seem only to provide conversions one way OR the other based on dip switch settings. Is there something availalbe that I have missed which is smart enough to know what direction to convert on the fly? The data acquisition system interprets the control and data lines and drives the status lines, is it possible to split the signals and ther rejoin them?

A: The Centronics-to-RS-232 converters that I've seen are for use only with a conventional Centronics interface - in other words, they convert the 8 Data bits only, either converting paralle inputs to serial outputs, or a serial input to parallel outputs. Your data-acquisition unit is sending data on the Status lines (probably 4 data bits and a strobe), and the converters can't handle this arrangement. One solution would be to program a microcontroller to do the conversions between the modem and data-acquisition device. Or (if possible) switch to a data-acquistion uni with a serial interface.

Q: I'd like to power a low component count device via an output port bit on the pc's parallel port. Does anyone know the current sourcing spec for an output port bit?

A: The data outputs on the original parallel port were 74LS374 outputs, which can source 2.6 milliamps at 2.4 guaranteed, 3.1V typical. The data outputs on most parallel ports are at least as strong as the original port's, but no guarantees. If you need 5V, you can use one of the high-efficiency step-up regulators from Maxim & others (Max756). The newer port controller chips often have IEEE-1284 Level 2 outputs, which can source 12 milliamps at 2.5V. The

Control outputs were originally open-collector TTL with 4.7K pullups, so these are less useful a current sources.

Speed

Q: How can Parallel Port ISA card support quoted rates of data transfer in excess of 1MB/sec ? As far a I know every in/out command for an ISA bus address takes at least 1 microsecond to perform REGARDLESS of processor speed ?

A: A simple parallel-port read or write does take one ISA-bus cycle, and on most systems, the bus speed is around 1.3 Mhz, so one cycle is a little under 1 microsecond. A complete data transfer on the original parallel port usually takes at least 4 cycles, however: check the Busy status, write the data, bring Strobe low, then high. On ports that support EPP and ECP modes, in these modes the port hardware does the handshake, and a complete transfer can take place i 1 bus cycle.

There are at least two ways of getting faster data transfers: Some ports support a "fast mode" that uses the ISA bus's NOWS (no wait state) signal to cause the CPU to skip 3 wait states. This mode is twice as fast, or around 2.7 Mhz. ECPs also include hardware support for data decompressing (& sometimes compressing, though that's often left for software), so the effective rate of data transmission can be much faster than the number of bytes written or read per second. Also, ECPs may be able to transfer data between the host's and peripheral's ECP buffers at rates greater than 1 Mhz, though the buffers are typically just 8 bytes.

Using Interrupts

Q: I'm confused about printer ports and IRQs. I've been told several contradictory things. I've been told that device drivers for don't actually use interrupts, so you can set multi-port cards such as byterunner's 4 serial/3 parallel port card to either all use the same IRQ for the parallel ports or none at all and everything should work. I've also been told that every printer port must absolutely use separate IRQ's.

A: You can print from DOS, Windows 3.1, and Windows 95/98 without using parallel-port interrrupts. It's easy enough to test: select no IRQ level on your port, then print to it. But - there are situations where you might need or want to use parallel port interrupts:

1. Under Windows 95/98, if your PC's port and your printer both support ECP transfers, W95 will print using ECP mode (faster) if ECP mode is enabled at the port (for motherboard ports, i the CMOS setup), if the PC's port is identified as ECP in the Device Manager, and if the port has an assigned interrupt and DMA channel (see Control Panel, System, Device Manager, Resources).

2. Drivers for other parallel-port devices (drives, scanners, etc.) may use interrupts, or they may work better (faster) with interrupts.

3. Some alternate print spoolers can use interrupts. LaserTools' PrintCache was one, though I haven't seen recent versions.

Bidirectional Ports

Q: I am trying to obtain the status of 8 switches via the parallel port. What I need is I think first a way to tell if I can tri-state the data-port lines port on my computer, so that I can read external signals on the data lines. Does anyone know how to tell if the port will support setting Control Port bit 5 to tristate the outputs?

A: To find out if the data port is bidirectional: 1. Set bit 5 of the control register (at base address+2). 2. With nothing connected to the port, write a couple of values to the data port, an read each back after you write it. If the reads DON'T match the writes, your port is probably bidirectional. Setting C5 disabled the data outputs and you're reading the open inputs of the data-port buffer. If the reads DO match the writes, your port isn't bidirectional. The data output are still enabled, you're reading back what you wrote, and you won't be able to read external signals.

On many ports, before you can use the data lines for input, you have to configure the port for "PS/2 mode", usually in the CMOS setup or a software utility. This writes to configuration registers in the chip that controls the port. The specifics of how to access the registers vary with the chip, so it's impossible to give a single programming procedure that will work for all. Some parallel ports (usually on expansion boards) have configuration jumpers on the circuit board to set the mode. (Check your board's documentation.) If the port is configured as ECP, select the PS/2 mode in the ECP's ECR (as described below).

Q: I've been searching for info on enabling PS/2, or Byte Mode, as described in the IEEE 1284 specification for all the different chips and boards. They all make the mode possible but the procedure is different for each chip and even for each board brand implementation!

A: If you've created a product that connects to the parallel port, and you have software capable of using Byte ("PS/2") mode, or another of the new modes with your device, you could do the following:

1. Ask users to enable the new modes at their port. It's just about impossible to do this automatically, because there's no way to detect which port-controller chip is used. If the controller chip is on the motherboard, there's normally a parallel-port option in the CMOS setup. If the port is on an expansion card, there should be jumpers or configuration software tha came with the board. On some boards, the new modes are hard-wired-enabled, so nothing needs to be done. (The user shouldn't have to select a specific mode, just configure the port so that the advanced modes are available.) Example: some (or all) Dell XPS P90 motherboard parallel ports allow selecting AT or PS/2 type port. The AT option locks out everything except "original" Centronics-type output. The PS/2 option puts the port in the controller's ECP configuration, with the PS/2 submode selected, so all of the other modes except EPP are available via the ECP's ECR register. The documentation doesn't explain this; the only way to determine it is to select the different options and examine the contents of the configuration registers. To use EPP mode, you do have to access the configuration registers.

2. Once the new modes are enabled, your software can try out the different modes and use the best of the available options.

Q: I am trying to interface a device to a parallel port, using PS/2 mode (simple bidirectional) and reading data on the parallel port's Data lines. I have successfully used this device on older AST 386 notebook ports as well as a modified SPP in a desktop computer. When in the PS/2 input mode, the input impedance of the older computers is fairly high, as I would expect if the output latch is truly tri-stated.

The problem I am chasing is with a new Toshiba 420 notebook computer, which professes to have a port which can be configured as a standard bi-directional or an ECP. After I have configured (through setup) the Toshiba port for bidirectional, when I switch from th compatibility mode to the byte (PS/2) mode by writing 2Xh to the port's Control register, each data pin is pulled up rather strongly to +5 vdc. It requires less than 270 ohms from any data pin to ground (sinking about 4 ma) to be able to read a 0 at the data register. This is certainly not my idea of "tri-state".

Have you ever dealt with such a port and would you believe this could be normal operation?

A: Several people have described this problem to me, always with Toshiba laptops. One possibility is that the Data lines have pullup resistors. The IEEE-1284 standard actually recommends using 1.2K pullups on inputs, "to ensure operation with Level 1 [original] and compatible devices." And an input with a 1.2K pullup would require a driver capable of sinking several milliamperes.

Using EPP & ECP

Q: Does the ECP parallel port specification require the use of a DMA channel?

A: No, but a specific device driver may make use of it if available. For example, if your PC and printer both support ECP mode, but your printer's port has no DMA channel assigned, the Windows 95 printer driver will use Fast Centronics mode instead of ECP. This is faster than SPP (original) mode, but not as fast as ECP mode.

Q: All the I/O cards I've seen require you to select a DMA channel when you jumper the parallel port in ECP mode. Yet, you can set up the "ECP Port" in Win95 with or without any DMA channel (and with or without any IRQ, for that matter). So, if I set up the parallel port on my I/O card to ECP mode with a DMA channel, but set up the "ECP Port" option in Win95 NOT to use any DMA channel, can I still use that DMA channel fo another device?

A: Yes.

Q: Alternatively, if I jumper my I/O card to EPP mode (which doesn't use DMA channel), should I still set it up in Win95 as an "ECP Port", or as a regular "Printer Port"?

A: Use the ECP setting unless you have problems with it.

Q: Which mode (ECP or EPP) would work best with parallel port devices such as the

Snappy and the QuickCam Color?

A: As a general rule, because of its FIFOs and DMA support, ECP is good at transferrring big blocks of data quickly (scanners, printers). EPP is good for links that switch directions frequentl (drives). Specifically, it depends on the driver for the device, so you might want to experiment if both options are available.

Q: Do I need to use parallel cables and switchboxes which are SPECIFICALLY labeled to support ECP/EPP to connect my parallel devices, or will any cable and switchbox work with all ECP/EPP ports and devices (as long as they have the 25 pins wired straight-through)?

A: A cable that is "IEEE 1284-compliant" meets certain standards for shielding, cable capacitance, etc. Over short links (5-6 feet), any cable will usually work, but you can run into problems with longer cables and higher speeds. Get the better cable if you have a choice.

Q: I have been communicating in ECP mode with another device. PC to peripheral transfers work fine. However in peripheral-to-PC transfers, sometimes the data repeats four times. For instance, If my peripheral sends a 0, I get four 0s instead of one.

A: My ECP controller on the PC side was configured for RLE data compression. RLE is enabled depending of the state of S7: S7 low is RLE disable, and S7 high is RLE enable. With RLE, the sending device sends the number of times to repeat a byte followed by the byte, and the receiving device automatically creates the requested number of repetitions. Francois Blouin provided this question and the answer.

Mode Problems

Q: I am having a problem using a parallel port device I designed and built,with the newer bidirectional parallel ports. The device works fine with the old IBM style ports but gags when I try to use it with the newer styleprinter ports. Is there a standard way to force a printer port into the old IBM configuration ?

A: There is no single standard. There are a couple of different approaches you could take.

One way is to disable the newer modes entirely. Depending on the port, you can do this in the CMOS setup, or with a jumper, or a utility on disk. Select "AT" type, not PS/2, ECP, or EPP. You can do the same thing by programming the port chip directly, but how to do it varies with the chip. This method is fine if you use the port only for your OScope. If you use the same port with other devices, they won't be able to use the newer modes unless you re-enable them.

The second option is to get a PS/2, EPP, or ECP port to act like an old-style, "original," SPP, without disabling the new modes entirely. A PS/2 (simple bidirectional) port or EPP will act like an SPP as long as you don't set bit 5 of the control port (at base address + 2) to 1. Setting this bit to 1 disables the data outputs and enables you to read external signals at the data port. On a SPP, the bit has no effect. (EPP transfers are done by writing to different port addresses; if you write to the base address, it acts like an SPP.)

ECPs are a little more complicated. An ECP can operate in several modes. You select the mod by writing to bits 7, 6, and 5 of the ECR (extended control register) at (base address + 402h). (Example: for a port at 378h, the ECR is at 77Ah.) If you want the port to act like an SPP, use mode 000 (SPP). In mode 011 (ECP), writing to the data port (the base address) causes the port to try to do an ECP Address Write cycle, with automatic handshaking, which you don't want.

To test if a port is ECP, read the ECR at (base address + 402h) and verify that bit 0 (fifo empty =1 and bit 1 (fifo full) =0. Then write 34h to the ECR and read it back. Bits 0 and 1 are read-only, so if you read 35h, you almost certainly have an ECP. (This is the test described in Microsoft's ECP document, on the MS developer's CD-ROM.)

So what can you do if you have an unknown port and want it to emulate an SPP? First, test for the presence of an ECP and if it exists, select mode 000 in the ECR. Otherwise, just remember to keep control bit 5 = 0.

One other thing: in the advanced modes, for faster performance, the control-port outputs often change from open-collector/open-drain type to push-pull (totem-pole) type. This means that in an EPP, and in all of the ECP modes except SPP, you can't count on being able to write "1" to control output and then use it as an input bit (as some projects do). Also, some ports apparently have push-pull control outputs in all of the modes, so for compatibility with all ports, don't use the control bits as inputs.

Accessing Ports Under Windows

Q: How do you output a byte to the parallel port under Windows 95? I'm using Delphi/Visual C++/Visual Basic. The faqs I've found say to write directly to the portregister, but I understand that Windows prevents this. Is this correct?

A: You can write directly to the port registers under Window 95/98 (but not under NT). If the port isn't being used by anything else, there should be no problem. Here's how to read and write to ports in various languages:

Visual Basic has no built-in way to access ports. A solution is to use a DLL written in Delphi or C that enables you to use inp and out routines in Visual-Basic programs. See http://www.lvr.com for inpout32 and other options.

In 16-bit Delphi, use inport and outport. In 32-bit Delphi (Delphi 2 and higher), use in-line assembly code. Here is Delphi source code for routines for reading and writing to ports.

```
library inpout32;
uses SysUtils;

{write a byte to a port}
procedure
Out32(PortAddress:smallint;Value:smallint);stdcall;export

var
ByteValue:Byte;
```

```
begin
ByteValue:=Byte(Value);
{in-line assembly code begins here}
asm
[preserve the contents of the dx register]
push dx
[write ByteValue to PortAddress]
mov dx,PortAddress
mov al, ByteValue
out dx,al
{restore the contents of dx}
pop dx
end;
end;]

{read a byte from a port}
function
Inp32(PortAddress:smallint):smallint;stdcall;export;

var
ByteValue:byte;

begin
{in-line assembly code begins here}
asm
[preserve the contents of the dx register]
push dx
[read ByteValue from PortAddress]
mov dx, PortAddress
in al,dx
mov ByteValue,al
{restore the contents of dx}
pop dx
end;
Inp32:=smallint(ByteValue) and $00FF;
end;

Exports
Inp32,
Out32;

begin
end.
```

In C, use inp and outp, if available. Some 32-bit compilers don't support these. In that case, use in-line assembly code, or an inpout DLL.

Jody Rice has provided this code for using inpout32 with Visual C++ v6:

```
typedef UINT (CALLBACK* LPFNDLLFUNC1)(INT,INT);
```

```
typedef UINT (CALLBACK* LPFNDLLFUNC2)(INT);
HINSTANCE hDLL; // Handle to DLL
LPFNDLLFUNC1 Output; // Function pointer
LPFNDLLFUNC2 Input; // Function pointer
INT Addr;
INT AddrIn;
INT Value;
hDLL = LoadLibrary("Inpout32");
if (hDLL != NULL)
{
  Output = (LPFNDLLFUNC1)GetProcAddress(hDLL,"Out32");
  Input = (LPFNDLLFUNC2)GetProcAddress(hDLL,"Inp32");
  if (!Output || !Input)
  {
    // handle the error FreeLibrary(hDLL);
  }
}
Addr = 0x378;
AddrIn = 0x379;
Value = 0;
Output(Addr, Value);
INT somenum = Input(Addr);
```

Q: In Visual Basic, I use MSComm to read and write to serial ports. How do I do the sam for parallel ports?

Unfortunately, things aren't as simple with the parallel port. If all you need is simple line-printer-type output (check the status lines and send each byte with a strobe), you may be able to use the Generic printer object, or the Createfile and Writefile API calls, or (16-bit only) Open "LPT1".

From Chris Wells: The lpt.vxd driver for Windows 9x may enable you read external signals on the data lines. Use SetCommTimeouts() to specify a ReadTotalTimeoutConstant of 1000 or so and call ReadFile.

For anything more complicated (PC-to-PC transfer or anything that requires reading the port), you're pretty much on your own. Windows and VB have no built-in support. There are a few vendors of parallel-port drivers, but these tend to be very expensive (thousands of $$) and are limited to the IEEE-1284 modes.

To write your own driver (which can be as simple as a routine in your application), there are various tools that enable reading and writing to ports. Under W3.x and W95, you can use an *inpout* DLL, though this offers no system-level protection. The alternative is a Vxd (W95), kernel-mode driver (NT), or an Ocx or Active-X Control that can communicate with these.

Q: I need to use the parallel port to control a device. I'm told that the best way is to use a driver like printers and data acquisition cards use. How do I write a driver for use with Visual Basic?

A: In the most general sense, a driver is just a program or set of routines that communicates witl

a specific piece of hardware, such as a parallel port. It can be as simple or sophisticated as you like. Under Windows 3.x or Windows 95, *driver* often means a VxD (virtual device driver). Advantages of using a VxD are the ability to protect a port by controlling access, and faster hardware-interrupt response. There's no way to write a VxD in Visual Basic. Most developers use Microsoft's Device Developers Kit (DDK), which includes an assembler, examples, and much information about driver development. Some C compilers also support VxD creation. To write a VxD, you need to be an experienced C or assembly-language programmer with a good understanding of Windows programming - or you will be when you're finished!

If all you need is the ability to register a port in the system registry (to control access to it), to read and write to the port, and to detect hardware interrupts, you can use one of the commercia OCX's designed for use with (32-bit) Visual Basic.

Q: My company has a product that connects to the parallel port. The process of finding the port address via WIN95's properties window is too much for some customers - we'd like the program to be able to find the address itself without needing the customer to find and enter the port
address. How can we find a parallel port's address under software control?

Here are three options:

1. Write a couple of values to each of the addresses and read back the result. If the values match, the port exists. This won't detect a port where access is denied by a system-level driver, or a port in PS/2-input mode (though you can also test for this). Requires an Inpout DLL or other driver for port I/O.

2. 16-bit only: Use Vbasm's Peek function to read the addresses stored in the BIOS data area at 40:08 through 40:0D. May not include all ports.

3. Under Windows 95, the port addresses are stored in the registry, under

[HKEY_LOCAL_MACHINE\Enum\Root\
[HKEY_LOCAL_MACHINE\Enum\BIOS\
[HKEY_DYN_DATA\Config Manager\Enum\

You can read the registry data with API calls.

From Dominic On: Option 3 is difficult in Windows 95. I tested many PCs and found that the byte location of the port address is different from one PC to another. But in windows NT it's OK. Based on the 3 options above, I came up with a solution: 1. Use an API call to detect which platform is running. (95 or NT) 2. If 95, use Toolhelp32ReadProcessMemory() function to find the port address. 3. If NT, read the port address from the registry. My application/DLL runs OK in windows 95. But when it runs in NT, I receive an error that Toolhelp32ReadProcessMemory() is not found in the Kernel32. Therefore, when I release my program/DLL, I have to compile conditionally in 2 versions (Windows 95 and NT).

And Bill McCarthy contributes this code that shows how to use Toolhelp32ReadProcessMemory to obtain port addresses:

```
Declare Function Toolhelp32ReadProcessMemory Lib
"KERNEL32" _
  (ByVal th32ProcessID As Long, _
  ByVal lpBaseAddress As Long, _
  lpBuffer As Any, _
  ByVal cbRead As Long, _
  ByRef lpNumberOfBytesRead As Long)
As Long

Sub Main()
Dim i As Long, rtn As Long
Dim portAddresses(3) As Integer
Dim cbLen As Long

cbLen = 8
rtn = Toolhelp32ReadProcessMemory _
   (0&, _
   &H408&, _
   portAddresses(0), _
   8, _
   cbLen)
'Debug.Print rtn, cbLen

For i = 0 To 3
  If portAddresses(i) = 0 Then Exit For
  Debug.Print _
  "port address " & i & " = &H" & Hex$(portAddresses(i))
Next i

End Sub
```

What Type of Port?

Q: How can I tell what type of port I have?

A: These are some ways to detect what type of port a system has: SPP (original), PS/2 (simple bidirectional), EPP, or ECP. The explanations assume that you have some familiarity with the different port types and how to access port registers. These tests detect only what type of port i currently enabled! If the advanced modes are disabled on the port controller chip, the tests won't detect them.

On many ports that support advanced modes, you can configure the port either in the CMOS setup, or with jumpers, or with configuration software that comes with the port. Most have an option that causes the port to emulate the original SPP, plus one or more options that enable the advanced modes. If the port is configured as an SPP, the advanced modes will be locked out and the port will fail any tests for PS/2, EPP, or ECP abilities.

Detecting an ECP

In testing a port, you might think that the first step would be to test for an SPP, and work your

way on up from there. But if the port is an ECP, and it happens to be in its internal SPP-emulation mode, the port will fail the PS/2 (bidirectional) test. For this reason, I begin by testing for an ECP, and work down from there. This is the method Microsoft's ECP document (in the Developer's Network CD-ROM) recommends for detecting an ECP: 1. Read the ECP's extended control register (ECR) at base address + 402h and verify that bit 0 (fifo empty) =1 an bit 1 (fifo full) =0. These bits should be distinct from bits 0 and 1 in the port's control register (at base address + 2). You can verify this by toggling one of the bits in the control register, and seeing that the corresponding bit in the ECR doesn't change. 2. A further test is to write 34h to the ECR and read it back. Bits 0 and 1 in the ECR are read-only, so if you read 35h, you almos certainly have an ECP. If an ECP exists, you can read and set the ECP's internal mode in the ECR. (See below.)

Detecting an EPP

In addition to the SPP's three registers, an EPP has four additional registers, at base address + through base address + 6. These additional registers provide a way to test for the presence of a EPP, by writing a couple of values to one of the EPP registers and reading them back, much lik you would test for an SPP. If the reads are successful, the register exists and you probably have an EPP. I'm not sure if this test works on all EPPs. Because the EPP handshake doesn't complete, there's no guarantee of the contents of the register after the transfer times out. But on the tests I've done, I was able to read back the values written. Be sure to clear the EPP timeout bit (bit 0 of the status port, at base address + 1) after each read or write. Unfortunately, the method for clearing the bit varies with the controller chip. On some ports, you clear the bit by writing 1 to it. On others, simply reading the status register clears the bit. And, though I haven't seen any controllers that clear the bit in the conventional way, by writing 0 to it, you may as well do that too, just to be safe.

Beware #1: on SMC's chips (& maybe others), a set timeout bit can make the port unusable in any mode, until a hard reset (or clearing the bit)!

Beware #2: Don't test for an EPP at address 3BCh. The added EPP registers aren't available at this address, and may be used for video.

Detecting an SPP

To test for an SPP, use the tried and true method of writing two values to the data port and reading them back. If the values match, the port exists. Otherwise, the port doesn't exist, or it's not working properly. Also note that the port-test routine only verifies the existence of the data port. It doesn't test the status and control lines. The other port types (EPP, ECP. PS/2) should also pass this test.

Detecting a PS/2-type Port

To test for simple bidirectional ability, first try to put the port in input mode by writing 1 to bit 5 in the port's control register (at base address + 2). If the port is bidirectional, this tri-states the data port's outputs. Then write two values to the data port and read each back. If the outputs have been tri-stated, the reads won't match what was written, and the port is almost certainly bidirectional. If the reads do match the values written, you're reading back what you wrote, which tells you that the data-port outputs weren't disabled and the port isn't bidirectional. An

ECP in its internal PS/2 mode and some EPPs will pass this bidirectional test. Test for a PS/2-type port only after you've verified that an SPP, EPP, or ECP exists. Because the PS/2 test uses the failure of a port read to determine that a port is bidirectional, a non-existent port wi pass the test!

ECP Modes

Q: Can an ECP emulate other port types?

A: An ECP has several internal modes. In addition to ECP mode, an ECP can emulate an SPP (original) or PS/2-type (Byte-mode, or simple bidirectional) port. Many ECPs also support EP emulation. Fast Centronics is an additional mode that gives faster performance with many SPP-type peripherals.

ECP internal modes:

000 SPP
001 Byte
010 Fast Centronics
011 ECP
100 EPP
101 Reserved
110 Test
111 Config

Set the mode in bits 7, 6, and 5 of the ECR at base address + 402h. For example, to set an ECP at 378h to ECP mode in Basic:

```
'The ECR is at 77Ah (378h + 402h)
EcrAddress=&h77A
'The code for the selected mode (EPP) from the table above:
EcpMode=3
'Read the ECR
ECRData=Inp(EcrAddress)
'Set the highest 3 bits to match the selected mode.
ECRData=(ECRData AND &h1F) + EcpMode * &h20
'Write the result back to the ECR.
Out EcrAddress, ECRData
```

Cables

Q: I am tring to find the pin-outs and pin descriptions for an IEEE-1284 cable or ECP cable.

A: It depends on what you mean:

1. A cable labeled "IEEE-1284 compliant" has to meet certain standards defined in the IEEE 1284 standard, such as double shielding and having each signal wire in a twisted pair with its

ground return. The connectors may be the traditional 25-pin D-subs, 36-contact Centronics-type, or the new, more compact 36-contact IEEE-1284C type. If the connectors are D-subs or Centronics, some of the pins have multiple ground wires. (The Centronics connector has 36 contacts, but the conventional use of it doesn't allow a ground return for every signal wire.)

2. If you're just looking for a cable that will do ECP transfers, you don't need an IEEE-1284-compliant cable, but it may give you faster transfers or allow a longer link.

3. See Lakeview Research's web site for the connector pinouts.

Q: Where can I find the wiring diagram of an ECP parallel cable for connecting a pair of computers to do file transfer at ECP port speed ?

A: The PC's parallel port was designed for PC-to-peripheral links, where the ports on the PC and peripheral complement each other. On the PC's port, the data and control bits are outputs, and the status bits are inputs, and on the peripheral, the data and control bits are inputs, and the status bits are outputs. If you want to connect the parallel ports of two PCs, you need a special cable that connects inputs on one port to outputs on the other. With old-style ports, the conventional way to do this is to cross-connect the status lines with 5 of the data lines, and have the software write a nibble at a time to the data lines and read the data on the status lines. (The other two lines do handshaking.)

Parallel ports that support ECP mode (or PS/2 (Byte) mode or EPP mode) have bidirectional data lines, and it would be nice to be able to use these for faster data transfers between PCs. For the handshake, you would need to cross-connect the status and control lines so that each output connects to an appropriate input.

One problem is that the default state for the data lines on both of the PCs' ports is output. If you connect the data lines of two ports to each other, you have outputs connected to outputs, and the resulting currents may destroy the port circuits. Some port controllers have protected outputs, but relying on this isn't the greatest solution. Other possibilities: Add an electronic switch that keeps the data lines from connecting until the software explicitly requests it. Be very, very careful not to plug in the cable until at least one data port has been configured as input. Use Parallel Technologies' Universal Cable (http://www.lpt.com/lpt/) which contains active circuits that handle the signal switching for you.

With that said, there is a document on the Microsoft Developer's Network CD-ROM titled, "Extended Capabilities Port: Specifications." A lot of the information in it is reprinted word-for-word in the controllers' data sheets, but it does also describe a method of connecting two PC-side ECPs to perform a "compliance test." This is the recommended wiring from this document:

Make these connections (PC "A" to PC "B"):
nStrobe (pin 1) to nAck (pin 10)
nAck (pin 10) to nStrobe (pin 1)
Busy (pin 11) to nAutoFd (pin 14)
nAutoFd (pin 14) to Busy (pin 11)
Data (pins 2-9) to Data (pins 2-9)

*nInit (pin 16) to PError (pin 12)
*PError (pin 12) to nInit (pin 16)
*nFault (pin 15) to nSelectIn (pin 17)
*nSelectIn (pin 17) to nFault (pin 15)
*SelectIn (pin 17) to Select (pin 13) on both PC A and PC B

* indicates a connection that isn't essential for data transfer. Pin numbers are for 25-pin D-subs.

Q: I'm experiencing some problems with my printer which may be caused by the fact that I do not use a BI-directional cable. I just bought a new cable, but I do not know for sure that it's indeed a BI-directional cable. Does anyone have the specs (pin-layout), so that I can check it, or is there another 'trick' to see whether the cable is BI-directional?

A: The term Bidirectional printer cable is confusing, because all PC printer cables have the same 17 signal lines. A PC with a bidirectional Data port can receive as well as send data over the 8 Data lines. Some printers can use a bidirectional port to send detailed status information back to the PC. But the signal lines are the same.

All printer cables should have at least 25 wires and be shielded. There are 17 signal lines and 8 ground wires. Some cheaper cables skimp by using just one ground wire, but even the cheap cables usually work OK if the printer doesn't use high-speed modes.

Some printers support a high-speed ECP mode, and other peripherals may use a different high-speed mode called EPP. A cable labeled "IEEE-1284-compliant" is designed to guarantee performance at ECP and EPP speeds. (Each signal wire forms a twisted pair with its ground return, and the cable has two shielding layers.) These cables might also be advertised as "bidirectional" cables, because ECP and EPP are bidirectional interfaces. But don't confuse this type of cable with a "Laplink" cable, which is used to connect two PCs' parallel ports and is a different animal altogether. Also, some printers and other peripherals support an older, slower bidirectional mode called "PS/2-type."

Q: Where can I get the new IEEE-1284-C connectors?

A: These are some manufacturers and part numbers listed in the IEEE-1284 standard:

PC mount Receptacle

AMP
2-175925-2
2-176971-5
2-178238-5

3M
10236-52A2VC
10236-5202VC

Molex
52311-3611
52311-3621

Cable Plug

AMP
2-175677-5

3M
10136-6000EC

Molex
52316-3611

General Printer Problems

Q: I am having trouble getting my printer to work. Everytime I try to print a test page I get a three line mess of numbers. I have a Panasonic KX-P4410 LASER PRINTER. I have tried many times to install a new driver for it. I have uninstalled and then installed it again but I keep getting the same getting that same message of some sorts everytime. I am working under Win95

A: I had a similar problem - a newly installed HP 6MP printer that spewed out a few lines of garbage on every bootup. A search of Dejanews brought up the advice to rename or delete this file:

\windows\system\iosubsys\drvwppqt.vxd

& it solved the problem. The file is apparently used by Microsoft Backup. If you don't use Backup, you don't need the file.

Q: How can I determine printer status (on/off-line, paper out, etc.)?

A: You can do this with an API call that retrieves a PRINTER_INFO_2 structure. To find out how, see article Q160129, *HOW TO: Get the Status of a Printer and a Print Job*, at http//support.microsoft.com/support .

However, you can read the status only after sending a job to the print spooler:

"The Spooler does not query for the state of a physical printer to which a Printer is connected. Instead, the state of a physical printer determines the success of a print job at the time it is despooled over the port monitor. If some error occurs in this process, the error is reported by the port monitor and recorded in a print job's status information. The Spooler, in turn, propagates reasonable error information to the Printer Queue. Consequently, a system Printer reports no status when the Printer queue is empty. In this state, the Printer is assumed ready to accept print jobs."