DAGE Scientific . . . using the Parallel Port

Every PC user has at least one parallel port. Learn how to use and program this standard port as a byte wide input and output device. The best way to approach this project is to build the test circuit, run the program, and then read this page to find out *how-it-works*.

| Home | Down Load Schematics | Test Circuit | Control Software |

Commercial products have long used this port to transfer large quantities of data. Computers are interfaced, digital pictures are entered, even external hard drives transfer data through this port. While some may claim this port is limited in speed, look at what can be done with a little imagination.

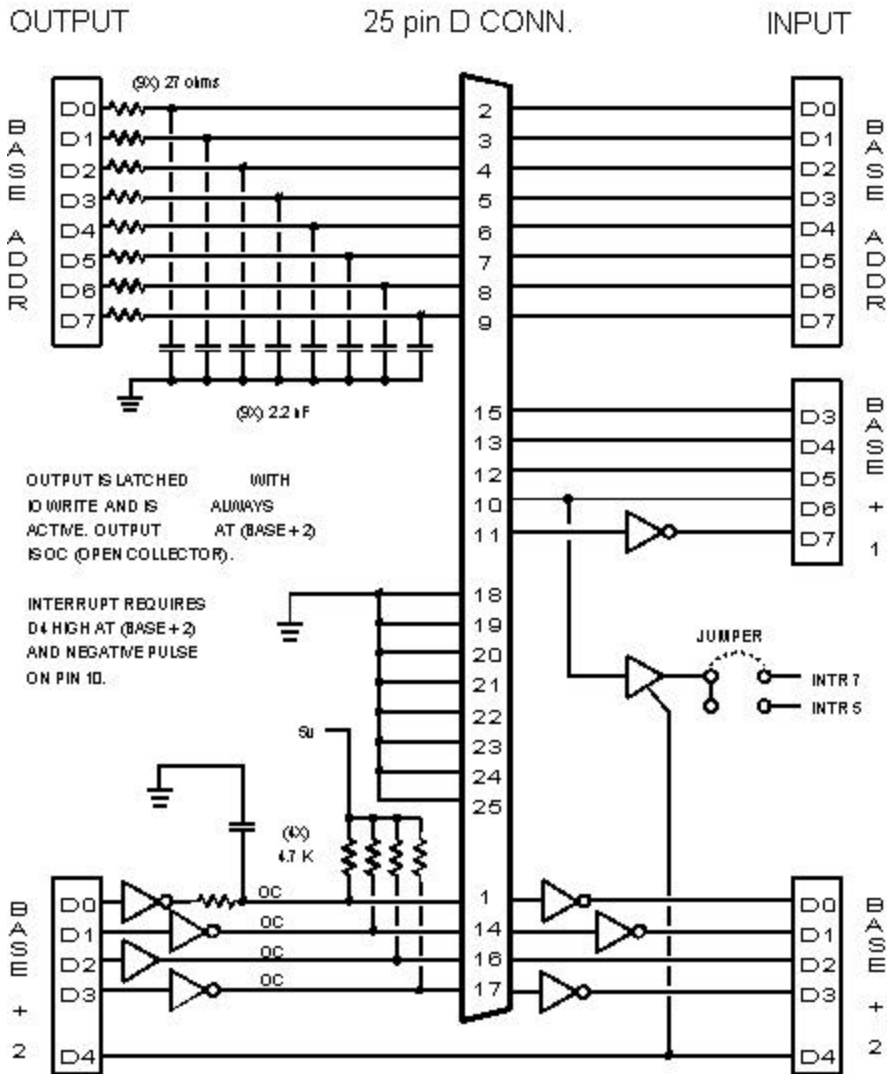
The standard PC parallel port is accessible through a 25 pin D connector at the back of your computer. Each pin and its associated circuitry is examined to provide the interfacing background you will need to use this port successfully. A sample breadboard hookup, driven with a Qbasic program will turn your parallel port into a bi-directional port with 8 bits in and 8 bits out.

We refer to the original parallel port as the *standard* , in contrast to the newer ports that are called *enhanced* .

Shown below is the *PC PARALLEL PORT INTERFACE* . This schematic, as well as the *PC PARALLEL TEST CIRCUIT* may be downloaded in the .wri format and printed. A hardcopy may be easier to follow.

| Download PC parallel port schematics | Up Top |

PC Parallel Port Interface



PARALLEL PORT HARDWARE

Many years ago, IBM designed the parallel port to drive printers. A standard 'D' 25 pin male connector was available on the back of the PC that connected to a printer. Their sole purpose was to interface with the de facto Centronics printer. Instead of building a clean direct interface that relied on software to invert signals, they used hardware inverters in a most unusual fashion. Here an inverter, there an inverter, everywhere an inverter. But it's the standard, everybody has one so lets use it.

DOS supports up to three parallel ports that are assigned the handles of LPT1, LPT2, and LPT3. Each port requires three consecutive IO addresses to select all the possibilities. They will be referred to as Base, Base +1, and Base + 2.

Parallel port is a misnomer. Actually there are five ports, consisting of two output ports and three input ports.

At base address, eight bits are available as output on pins 2-9. They are hard wired to an eight bit input at the same address. The output is latched in the usual manner with the IO write pulse and is

always active. The original IBM card used a 74LS374 tri-state device that has it output enable pin hardwired active. This means the input can only read the output so it's not usable except to check that the output is correct. The original IBM card could be reworked to control the output enable pin that would then free up the input, but this is not recommended. Resistor and capacitors suppress ringing but should not limit this ports speed. When using CMOS devices driven from this output, it's recommended that pull-up resistors be used.

Chances are that you don't have an original IBM card. The newer ports combine the IBM circuitry into a single chip and change the series resistor to a pull-up. They're functional equal. Use this base address as a straight forward eight bit output only.

At base + 1, there are five input bits from D3 to D7. They are gated on the bus with IO read. Note that bit D7 (pin 11) is inverted. Software can invert this bit if necessary and will be demonstrated later. Bit D6 (pin 10) can also be used to generate a hardware interrupt. Several conditions must be met before an interrupt occurs. This pin can be used as a data input without concerns of inadvertently causing an interrupt. If a hardware interrupt is desired, this is the pin to use.

At base + 2 several options exist. This is a four bit output or a four bit input, or can be configured as any mixture of input and output. This is possible because the output is open collector. By sending data to this port to make an output pin high, allows that pin to be driven as an input. The open collectors are pulled high with 4.7 K resistors. Open collectors are not driven high but float high due to the pull up resistor charging any capacitance in the circuit. The speed on this port will not be as fast as on the base address particularly bit D0 that has an added external capacitor. Test any application that requires maximum output speed from this port.

Interrupts are normally open collector activated with a device pulling the line low. Any card in the ISA slot can pull an interrupt line low. This parallel port card uses a different approach. Bit D4 at base + 2 controls a tri-state device. When D4 is high, pin's 10 logic state is passed to the hardware interrupt number 7 (default) or number 5. Remember that all outputs are latched and remain active. If bit D4 is set high, and input pin 10 is high, this could disable other cards from using the selected interrupt. Your software program should set bit D4 low when not controlling the interrupt.

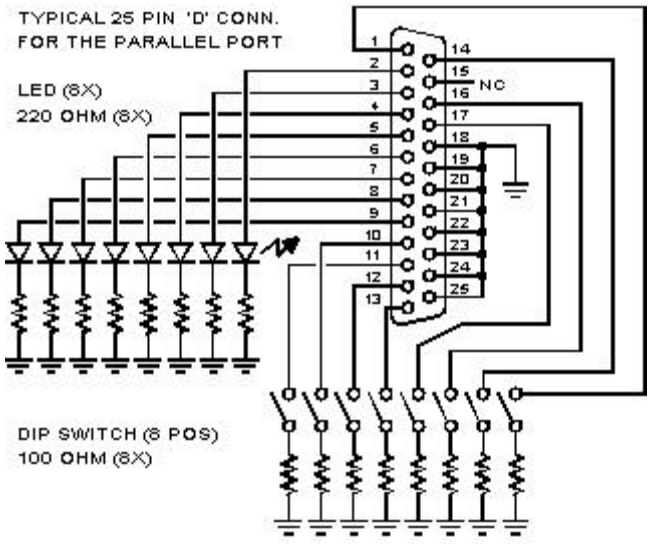
To summarize: the parallel port is capable of eight to 12 output bits, and five to nine input bits. One of the input lines can be used as a hardware interrupt. All inputs and outputs will behave as 74LS devices.

TEST CIRCUIT FOR BYTE IN/OUT

The goal of this circuit is to use the minimum number of parts that will provide a functional test of all eight bits. Each input bit can be individually controlled, and visually displayed on the eight output LEDs. The display LEDs are driven with the source current only from the output. The preferred method would be to power the LEDs and use the sink current to light them. No power (Vcc) is available on the parallel plug, so external power would be required. It was found empirically that on the original IBM port and several clones, that each card produced ample source current for clear indication from the LEDs.

| Up Top |

PC Parallel Port Test Circuit



TEST CIRCUIT FOR BYTE IN/OUT

The circuit can best be built on a solderless breadboard with short jumpers (about 8") soldered to a 25 pin male 'D' connector. An extension cable (male on one end, female on the other) will allow the circuit to be placed in front of the computer for easy access. To simplify the circuit wiring, use a 10 position LED bargraph, an eight position DIP switch, and two 10 pin SIP resistors of the value shown. Installation of these parts requires only one jumper. Individual LEDs and resistors also work with just a little more wiring. The whole project should be easily constructed within an hour.

Program in QBASIC

To use the following Qbasic program select it with the mouse and use your menu commands to copy it to the clip board (Edit-Copy). Then open your Windows Word processor (Write, Wordpad or Word for Windows) and paste from clip board. Save as PP_TEST8.BAS as a text file only.

| Up Top |

```
CLS

'  BIOS places address of parallel ports starting at absolute address &H408
'  LPT1 low byte at &h408, high byte at &H409
'  LPT2 low byte at &H40A, high byte at &H40B
'  LPT3 low byte at &H40C, high byte at &H40D
'  Possible values are &H0278, &H0378, &H03BC, and 0 if not installed

'  Find printer ports if installed
DEF SEG = 0
lpt1 = PEEK(&H408) + 256 * PEEK(&H409)
lpt2 = PEEK(&H40A) + 256 * PEEK(&H40B)
lpt3 = PEEK(&H40C) + 256 * PEEK(&H40D)
DEF SEG

IF lpt1 = 0 THEN
```

```
        PRINT "1.  lpt1 is not installed."
ELSE
        PRINT "1.  lpt1 is installed at address "; lpt1; " decimal."
END IF

IF lpt2 = 0 THEN
        PRINT "2.  lpt2 is not installed."
ELSE
        PRINT "2.  lpt2 is installed at address "; lpt2; " decimal."
END IF

IF lpt3 = 0 THEN
        PRINT "3.  lpt3 is not installed."
ELSE
        PRINT "3.  lpt3 is installed at address "; lpt3; " decimal."
END IF

'  Input desired port
PRINT

DO
    INPUT "Which LPT port will you use; 1, 2, or 3?  ", N%

'  Assign Base0% to selected port address
'  Base is a keyword in Qbasic so Base0% is used instead

    IF N% = 1 THEN Base0% = lpt1
    IF N% = 2 THEN Base0% = lpt2
    IF N% = 3 THEN Base0% = lpt3

    IF Base0% = 0 THEN
        PRINT "That port doesn't exits, try again."
        N% = 0
    END IF

LOOP WHILE N% > 3 OR N% < 1

'  ********** Start IN to OUT  *********

Base1% = Base0% + 1  'assign addresses
Base2% = Base0% + 2

'  activate Base2% for high open collectors and disable interrupt
OUT Base2%, 4

PRINT : PRINT "Input is now controlling output;"
PRINT "press any key to cycle LEDs..."
DO WHILE INKEY$ = ""
    B1% = INP(Base1%) AND &HF0
    B2% = INP(Base2%) AND &HF
    FullByte% = B1% OR B2%
    FullByte% = FullByte% XOR &H8B

    OUT Base0%, FullByte%

LOOP

'  ********** Start LEDs to cycle ******
```

```
'  TON can range from 500 to 2000 to vary the cycle speed
TON = 500

PRINT : PRINT "press any key to quit..."
DO WHILE INKEY$ = ""
    FOR C = 1 TO 7
        OUT Base0%, 2 ^ C
        FOR T = 1 TO TON: NEXT T
    NEXT C

    FOR C = 6 TO 0 STEP -1
        OUT Base0%, 2 ^ C
        FOR T = 1 TO TON: NEXT T
    NEXT C

    OUT Base0%, 0
LOOP

OUT Base0%, 0   'shut off all LEDs

END
```

During initialization, BIOS checks for three parallel ports at IO addresses 378, 278, and 3BC hex. It places these IO addresses in memory starting at 0000:0408 hex with lower byte first, followed by the higher byte. If a parallel port is not installed it places zeros in that location.

The table below shows how memory would look if your computer has two parallel ports installed at 378 and 278 hex.

DOS Handle	Handle Address	Low Byte	High Byte	Decimal
LPT1	0000:0408/9	78	03	888
LPT2	0000:040A/B	78	02	632
LPT3 (not installed)	0000:040C/D	00	00	0

The first half of the QBasic program goes out and 'PEEK's at memory and assigns active parallel port addresses to the DOS handles. It then allows you to select which port to use. You may skip this part if you are only going to run this program on a single computer, and you know the address that will be used. Simply assign that address to variable Base0%. Use the statement [Base0% = 888] if your parallel port is at address 0378 Hex.

The program listed chooses a particular configuration for the port, i.e., eight bits out and eight bit in. To understand the next part of the program shown as *Start IN to OUT* refer to the PC PARALLEL PORT INTERFACE schematic.

The lower four input bits will come from Base + 2. Before they can be used as input, the output four bits (D0 to D3) at address Base + 2 must be set high. Notice that bits D0, D1, and D3 are hardware inverted. Output zero for these three will make the signals at the input pins high. Bit D2 is not inverted so it must be a one. The interrupt can be disabled by setting bit D4 to zero. Bits D5 - D7 are not connected so they may be any value. For simplicity, make them zero also. QBasic outputs only a full 8 bits. The byte that is needed to accomplish all of the above is [00000100] binary or

decimal 4. The QBasic statement is [OUT 4, Base2%]. That's it; a short statement with a lot of meaning. The four inputs at Base + 2 can now be used in any manner knowing that the interrupt is disabled.

The upper four bits will come from Base + 1. Nothing needs to be done before using them as inputs. Fortunately they are also in the proper order of D4 - D7.

The program must input one set of four bits at a time. If you have control over the inputs, hold them long enough for both IO reads to occur. If you don't have control and the signals are changing rapidly, an external latch must be used to capture all eight bits at the same time.

The object now is to input two sets of four bits, add them into one byte and then undo all the bits that were inverted by the hardware. The lower four bits of input at Base + 1 may be any value. By using what is called an 'AND MASK' the lower four bits can be set to zero without affecting the upper four bits. The bits that need to be saved are 'AND' with ones while the bits to be set to zero are 'AND' with zero. These bit operators act on the variable one bit at a time with the result mirroring the truth table of the operator. The statement that performs the above is [B1% = INP(Base1%) AND &HF0]. The value input to the computer at address base + 1 is first stored in memory location B1%. It then is 'AND' with F0 hex and stored back into the same place.

This is shown below in table form. The X indicates bits that we need while the ? are unknown and need to be set to zero. The blue color bits are true while the red color bits are inverted and need to be changed. We begin by setting the unknown bits to zero.

B1% = INP(Base1%) AND &HF0

INP(BASE + 1)							
X	X	X	X	?	?	?	?
AND &HF0							
1	1	1	1	0	0	0	0
EQUALS							
B1%							
X	X	X	X	0	0	0	0

The same procedure is used with the input from Base + 2 except that in this case the upper four bits are masked to zero [B2% = INP(Base2%) AND &H0F].

B2% = INP(Base2%) AND &H0F

INP(BASE + 2)							
?	?	?	?	Y	Y	Y	Y
AND &H0F							
0	0	0	0	1	1	1	1
EQUALS							
B2%							
0	0	0	0	Y	Y	Y	Y

We now have two bytes that need to be combined. Do not add them; addition is not a bit operator. It will work in this example but only under special circumstances. Instead use the 'OR' bit operator. [FullByte = B1% OR B2%].

FullByte = B1% OR B2%

B1%							
X	X	X	X	0	0	0	0
OR B2%							
0	0	0	0	Y	Y	Y	Y
EQUALS FULLBYTE							
X	X	X	X	Y	Y	Y	Y

Bits D0, D1, D3, and D7 need to be inverted. The 'XOR' operator will be used to change these four bits; use a one to invert a bit and a zero to pass the bit unchanged. [FullByte = FullByte XOR &H8B].

FullByte = FullByte XOR &H8B

FULLBYTE							
X	X	X	X	Y	Y	Y	Y
XOR							
1	0	0	0	1	0	1	1
EQUALS CORRECTED FULLBYTE							
X	X	X	X	Y	Y	Y	Y

The only thing left is to output this corrected byte to the LEDs. [OUT Base0%, FullByte].

The program loops until any key is pressed. The last part of the program will cycle the LEDs back and forth making a rhythmic display. Why not have a little fun?

The parallel port certainly has its idiosyncrasy's and may not be the easiest means for interfacing, but it's a start. Armed with this information, what can you devise to feed through this port?

| Up Top | Home |

To contact *DAGE* Scientific:
Phone 209 772-2076

FAX 707 924-7116

or

Send us e-mail