Simple Parallel Port Interfacing

by xxxtoytech
Simple Parallel Port Interface, 8 output

The first project took about 30 minutes to build on a piece of perf board and cost less than $5; see photo at right (shown with batteries). It is from an article by James M. Conrad and Jonathan W. Mills in Circuit Cellar #108, July 1999 “A PC-Based Controller for the Stiquito Robot”. That article suggests you try Radio Shack perf board (Cat. # 276-150) The ULN2803 is a darlington transistor array with built-in resistors for 5 volt CMOS operation. To turn on an output transistor write a 1 to that bit (0 to 7). To get an idea of what is going on, hook up an LED (Light Emitting Diode) with a 220 ohm resistor between the output of the ULN2803 and a 4.5 volt supply (3 AA batteries in series). This circuit is shown as the LED Output Tester on page 4. When transistor is on, the LED will glow. By adding opto-isolators (devices that electrically isolate parts of a circuit by using an LED and photodetector such as a transistor or triac) you can power devices of different voltages.

Use the SIMTEST.BAS program to test this 8 output parallel port interface. The program is designed to run in the MS-DOS QBASIC interpreter. Program steps through all control codes from 0 to 255 (1 byte or 8 bits).

Parallel Port Interface, 8 inputs 8 outputs

Construction of the board took an afternoon, including checking the board. In the photo below right is the finished board with the DB-25 connector at the top and the power supply along the left side. The connector at the bottom center is for connecting the inputs, power supply connection is at the bottom left. For testing I used an old 80386 computer and the QBASIC interpreter that comes with MS-DOS. Using an old computer makes life a bit easier since you’re not as concerned with damaging the motherboard etc. The MS-DOS OS gives you access at a low level to the ports, the QBASIC language, although somewhat ugly, is simple to use for testing. Connect to the computer printer port with a standard DB-25 male/female cable from your computer store (I paid about $5 for a 6 foot cable).

The outputs/outputs are set or read independent of each other (unlike the simple interface above). To use this interface Data Bits 0 to 2 are set with the address of the input/output you want to read/set Data Bit 3 selects input or output (1 for output, 0 for input). Data Bit 4 is only used for the outputs and sets the output (0 is off, 1 is on). You then use the Printer Control Register to send an nStrobe signal to the interface to either set the output or read an input. In the case of an input, the value (either 0 or 1) is stored in a latch and can be read by reading the PError (Bit 5) in the Printer Status Register. Reading is simple, just AND the value of the status register with 20 hex (32 decimal). If 0 then the input read is OFF, otherwise the input read is ON. See the TEST.BAS program on page 6-7 and the parallel port info on page 3 to get a better grasp of how this works.

Wire the switches to the inputs of the CD4051 so that when the switch is closed it brings the output to ground. You can also interface other circuits to the inputs by using a general purpose NPN transistor (i.e. 2N3904) to ground the input. To approximate analog inputs, sample the switches over time.

Voltage on the outputs can be up to 500mA at 50 volts, however, it’s a good idea to use opto-isolators along with the interface board’s built-in 5 volt power supply, to isolate the board from high voltages and noise. When using opto-isolators with the board's 5 volt supply use a resistor to limit the current to isolator's ILED (about 270 ohms). You can control whatever you want with the outputs depending upon what type of opto-isolator you use. You can use the LED Output Tester circuit shown on page 4 along with the TEST.BAS program on pages 6-7 for testing the outputs. Use switches between ground and the inputs on CD4051 for testing the inputs.

While it is far from state of the art, this interface is cheap with available components and simple to build (even for the beginner), easy to use and trouble-shoot and versatile.
Using the Parallel Port to send data to the control unit.

To send data to the device, the data byte is first placed in the printer data register for the parallel port that the device is connected to. This is at the base address of the parallel port. The computer pulses strobe line to inform the device that the data is available on pins 2-9. After being read, the ack(nowledge) line (pin 10 on D shell connector) goes low (negative pulse). This pulse is about 10 microseconds in length. The three registers we are concerned with are:

**Printer Data Register** - holds byte going to computer, bits correspond to the pin 2-9 on the 25 pin D-shell connector. Pin 2 corresponds to the least significant bit (LSB), pin 9 corresponds to the most significant bit (MSB). The printer data register is located at the base address of the port to which the device is connected.

**Printer Status Register** - holds the return values from the printer. These correspond to the signals on the various pins (see information on pin signal assignment that follows). The printer status register is offset by one port from the base port (base port address + 1). The byte assignments are below.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>not used</td>
</tr>
<tr>
<td>1</td>
<td>not used</td>
</tr>
<tr>
<td>2</td>
<td>not used</td>
</tr>
<tr>
<td>3</td>
<td>nFault</td>
</tr>
<tr>
<td>4</td>
<td>Select</td>
</tr>
<tr>
<td>5</td>
<td>PError</td>
</tr>
<tr>
<td>6</td>
<td>nAck</td>
</tr>
<tr>
<td>7</td>
<td>Busy</td>
</tr>
</tbody>
</table>

**Printer Control Register** - sends commands to the printer. This register is offset by two ports from the base address (base port address + 2). Although the programmer can handle these directly, for slow devices at low rates of data transfer it is easier to let low level BIOS or DOS functions handle the commands. For those who are interested the commands follow.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>nStrobe</td>
</tr>
<tr>
<td>1</td>
<td>nAutoFd</td>
</tr>
<tr>
<td>2</td>
<td>nInit</td>
</tr>
<tr>
<td>3</td>
<td>nSelectIn</td>
</tr>
<tr>
<td>4</td>
<td>interrupt</td>
</tr>
<tr>
<td>5</td>
<td>not used</td>
</tr>
<tr>
<td>6</td>
<td>not used</td>
</tr>
<tr>
<td>7</td>
<td>not used</td>
</tr>
</tbody>
</table>

Use IBM compatibility mode (default for parallel port) 25-pin D-Shell connector (female on computer, male on device)

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>nStrobe</td>
<td>from computer, negative pulse tells device data is ready</td>
</tr>
<tr>
<td>2</td>
<td>Data 0</td>
<td>data from computer (least significant bit)</td>
</tr>
<tr>
<td>3</td>
<td>Data 1</td>
<td>data from computer</td>
</tr>
<tr>
<td>4</td>
<td>Data 2</td>
<td>data from computer</td>
</tr>
<tr>
<td>5</td>
<td>Data 3</td>
<td>data from computer</td>
</tr>
<tr>
<td>6</td>
<td>Data 4</td>
<td>data from computer</td>
</tr>
<tr>
<td>7</td>
<td>Data 5</td>
<td>data from computer</td>
</tr>
<tr>
<td>8</td>
<td>Data 6</td>
<td>data from computer</td>
</tr>
<tr>
<td>9</td>
<td>Data 7</td>
<td>data from computer (most significant bit)</td>
</tr>
<tr>
<td>10</td>
<td>nAck</td>
<td>acknowledge, device sends negative pulse to computer</td>
</tr>
<tr>
<td>11</td>
<td>Busy</td>
<td>device sets this high to tell computer it is busy</td>
</tr>
<tr>
<td>12</td>
<td>PError</td>
<td>from device to computer</td>
</tr>
<tr>
<td>13</td>
<td>Select</td>
<td>from device to computer</td>
</tr>
<tr>
<td>14</td>
<td>nAutoFd</td>
<td>from computer to device</td>
</tr>
<tr>
<td>15</td>
<td>nFault</td>
<td>from device to computer</td>
</tr>
<tr>
<td>16</td>
<td>nInit</td>
<td>from computer to device, negative pulse resets device</td>
</tr>
<tr>
<td>17</td>
<td>nSelectIn</td>
<td>from computer to device</td>
</tr>
<tr>
<td>18-25</td>
<td>Ground</td>
<td>signal ground</td>
</tr>
</tbody>
</table>
REM - SIMTEST.BAS - This program is a simple test program for an
REM - 8 output parallel port interface designed to run in the
REM - MS-DOS QBASIC interpreter. Program steps through all control
REM - codes from 0 to 255 (1 byte or 8 bits)

DIM pport%(3)
pport%(0) = &H278
pport%(1) = &H378
pport%(2) = &H3BC
cport% = pport%(0)
togg% = 0
CLS
PRINT : PRINT
PRINT “Simple Parallel Port Interface Test”
PRINT “Press ESC to exit program”
PRINT “Press P to change Port”
PRINT “Press T to step through outputs”
OUT cport%, togg%
PRINT “All outputs are OFF”
PRINT “Current port is “; HEX$(cport%)

REM - main loop of program -
DO
ks$ = INKEY$
SELECT CASE ks$
CASE CHR$(27)
EXIT DO
CASE “P”, “p”
PRINT “Change Port”
PRINT “Enter 2 for port 0278h”
PRINT “Press 3 for port 0378h (default port)”
PRINT “Press B for port 03BCh”
INPUT PortSelect$
SELECT CASE PortSelect$:
CASE “2”
cport% = pport%(0)
CASE “3”
cport% = pport%(1)
CASE “B”: CASE “b”
cport% = pport%(2)
CASE ELSE
PRINT “Not a valid choice, Press P to try again”
END SELECT
PRINT “Port Selected”, “0”; HEX$(cport%); “h”
togg% = 0
CASE “T”, “t”
REM - The following OUT writes a data byte to Printer Data Register
OUT cport%, togg%
PRINT “Output “; togg%
togg% = togg% + 1
IF togg% = 256 THEN togg% = 0
END SELECT
LOOP
REM - End of program
Parallel Port Interface
8 inputs, 8 outputs

Notes: IC1, IC2, IC3 and IC4 are powered from the +5 volt power supply.
Connect as follows:

IC: +5 volts | Ground
---|---
IC1 CD4093 | Pin 14 | Pin 7
IC2 CD4042 | Pin 16 | Pin 8
IC3 CD4099 | Pin 16 | Pin 9
IC4 CD4051 | Pin 16 | Pin 10

A 0.1 μF capacitor should be placed between the supply and ground of each IC1, IC2, IC3 and IC4.

Bridge Rectifier 1A
220 μF 35 v
220 Ω
1000 μF 10 v
REM - TEST.BAS - This program is a simple test program for an 8 input, 8 output
REM parallel port device designed to run in the MS-DOS QBASIC interpreter

DIM pport%(3)
pport%(0) = &H278
pport%(1) = &H378
pport%(2) = &H3BC
cport% = pport%(0)
togg% = 8

REM - Turn off all outputs
FOR i = 0 TO 7 STEP 1
OUT cport%, togg%
OUT (cport% + 2), 1
togg% = togg% + 1
NEXT i
togg% = 24

CLS : PRINT : PRINT
PRINT ªSimple Parallel Port Interface Device Testª
PRINT ª----------------------------------------------------ª
PRINT ªPress ESC to exit programº
PRINT ªPress H for Helpº
PRINT ªPress P to change Portº
PRINT ªPress S to Scan inputsº
PRINT ªPress T to Toggle outputs on and offº
PRINT : PRINT ªAll outputs are OFFº
PRINT ªCurrent port is ª; HEX$(cport%)

DO
ks$ = INKEY$
SELECT CASE ks$
CASE CHRS$(27)
   EXIT DO
CASE ªHº, ªhº
   PRINT ªPress ESC to exit programº
   PRINT ªPress H for Helpº
   PRINT ªPress P to change Portº
   PRINT ªPress S to Scan inputsº
   PRINT ªPress T to Toggle outputs on and offº
CASE ªPº, ªpº
   PRINT ªChange Portº
   PRINT ªEnter 2 for port 0278hº
   PRINT ªPress 3 for port 0378h (default port)º
   PRINT ªPress B for port 03BChº
   INPUT PortSelect$
   SELECT CASE PortSelect$
   CASE ª2º
      cport% = pport%(0)
   CASE ª3º
      cport% = pport%(1)
   CASE ªBº: CASE ªbº
      cport% = pport%(2)
   CASE ELSE
      PRINT ªNot a valid choice, Press P to try againº
   END SELECT
   PRINT ªPort Selectedº, ª0º; HEX$(cport%)
   togg% = 8
END SELECT
PRINT ªPort Selected”, ª0º; HEX$(cport%); ºh”
togg% = 8
CASE ‘S’, ‘s’
  PRINT ‘Scan Inputs’
  PRINT “——————”
  FOR i = 0 TO 7 STEP 1
    OUT cport%, i
    OUT (cport% + 2), 1
    x% = INP(cport% + 1) AND &H20
    IF x% > 0 THEN PRINT “HIGH #”; (i + 1) ELSE PRINT “LOW #”; (i + 1)
  NEXT i
CASE ‘T’, ‘t’
  IF togg% > 23 THEN PRINT ‘Output ‘; (togg% - 23); “ ON” ELSE PRINT “Output ‘; (togg% - 7); “OFF”
  REM - The following OUT writes a data byte to Printer Data Register
  OUT cport%, togg%
  REM - The following OUT writes a control byte to the
  REM - Printer Control Register which sends an nStrobe signal
  REM - to Pin 1 of the DB-25 connector
  OUT (cport% + 2), 1
  togg% = togg% + 1
  IF togg% = 16 THEN togg% = 24
  IF togg% = 32 THEN togg% = 8
END SELECT
LOOP
REM - End of program

Codes for Addressing the 8 input/output interface
Codes for each input/output numbered 0 through 7 (X0-X7 on the CD4051 inputs, O1-O8 outputs on the ULN2803) first in decimal numbering followed by the hex code in brackets. The read input code places the value of the input (0 or 1) in a latch. To actually get the value look at the PError value (Bit 5) of the Printer Status Register.

<table>
<thead>
<tr>
<th>I/O</th>
<th>Set Output LOW</th>
<th>Set Output HIGH</th>
<th>Read Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8 (08h)</td>
<td>24 (18h)</td>
<td>0 (00h)</td>
</tr>
<tr>
<td>1</td>
<td>9 (09h)</td>
<td>25 (19h)</td>
<td>1 (01h)</td>
</tr>
<tr>
<td>2</td>
<td>10 (0Ah)</td>
<td>26 (1Ah)</td>
<td>2 (02h)</td>
</tr>
<tr>
<td>3</td>
<td>11 (0Bh)</td>
<td>27 (1Bh)</td>
<td>3 (03h)</td>
</tr>
<tr>
<td>4</td>
<td>12 (0Ch)</td>
<td>28 (1Ch)</td>
<td>4 (04h)</td>
</tr>
<tr>
<td>5</td>
<td>13 (0Dh)</td>
<td>29 (1Dh)</td>
<td>5 (05h)</td>
</tr>
<tr>
<td>6</td>
<td>14 (0 Eh)</td>
<td>30 (1 Eh)</td>
<td>6 (06h)</td>
</tr>
<tr>
<td>7</td>
<td>15 (0Fh)</td>
<td>31 (1Fh)</td>
<td>7 (07h)</td>
</tr>
</tbody>
</table>