Controlling the PIC-IO board from your PC

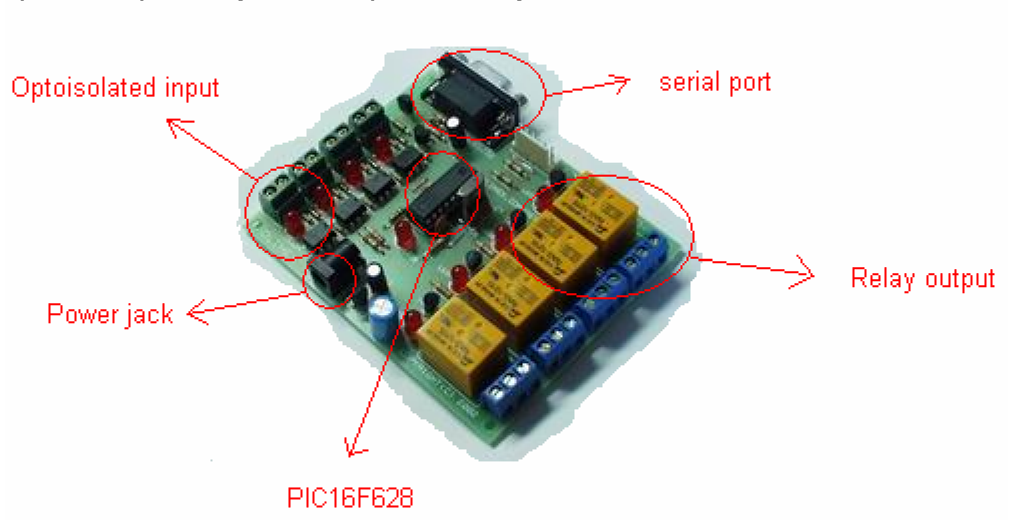
Por. Paul Aguayo S.
www.olimex.cl

Introduction

PIC-IO from Olimex is a small but powerful development board who let you control 4 optoisolated input and 4 relay outputs (220V,10A) with this features is possible to turn on and off almost any electronic device at home.

This board is almost a discrete PLC module and it has a serial port RS232 that allow you to communicate with a PC. The signals from de input channels can be analyzed by the PIC and this one could generate a output.

In this short tutorial we will show how to use the PIC-IO board to communicate it with a graphical interface that let you turn on and off the outputs from a PC. If you work a little bit in this example you could control devices (lamps, small motors, sprinklers) from your computer easily.



What do I need?

1 development board PIC-IO
1 Microcontroller 16F628 (can be any other compatible with PIC-IO, just be sure to compile the code for the model that you are using)
1 PIC-PG1 programmer (to program the pic)
1 serial cable (to work comfortable)
12V power supply
Visual Basic 6.0
stdio.c from http://www.olimex.cl/soft/pic/stdio.c

Objective

What we want to do is send a command from the PC to the PIC. This command has to be interpreted by the PIC. The PIC will activate an output depending of what message it received.

PIC side

We need to define a communication "protocol" between the PC and the PIC-IO board.

I will use this "protocol":

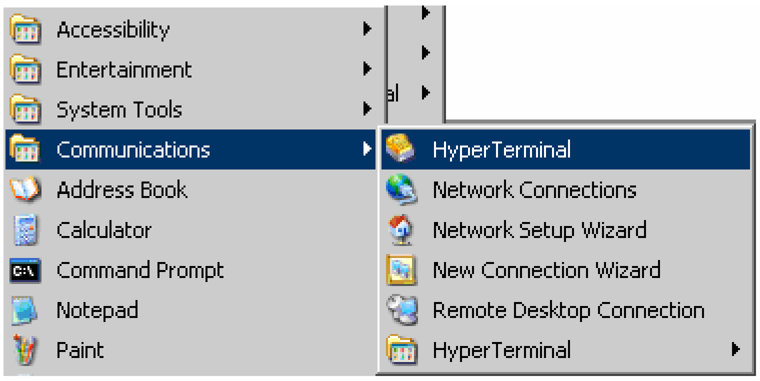
O <output channel number> <output channel state>

Where:

O is the capital letter O.
<output channel number> is the number of the channel that identify the output that we want to turn on or off. The admissible values can be 1, 2, 3 or 4
<output channel state> is the estate of the channel. The state can be 'A' meaning activated or 'D' meaning deactivated.

Let see how the "protocol" works with 2 examples: the command O1A will activate the output channel number 1. The command O4D will deactivate the output channel number 4.

These commands have to be sent through the serial port. To test the communication we can use the HyperTerminal program that came with Windows. For our purpose we will use the 9600,8,N,1 configuration



Connected 00:00:07 | ANSIW | 9600 8-N-1

Let's see how we can use the PIC to accomplish this job. All the C code is in this document you just have copy and compile it. Why use C? Because is easiest than Asm :). We used cc5x to compile the following code. If you don't know how to compile C code for PIC microcontrollers I'll recommend you to read this: http://www.sparkfun.com/tutorial/Setup_Space/setting_up_your_space.htm

What we do with this code is use interrupt to know when the PIC receives data from the PC. We saved the last 3 characters and when the PIC received a 0x0D (enter in ASCII code) we check if the sequence is one of the allowed sequences.

```
#define HS_Osc
#define Serial_Out_BB   RB2
#define Serial_In_BB   RB7
#define Baud_9600

#include "c:\cc5x\16F628.h" //Compiler specific header file
#include "c:\cc5x\int16CXX.H" //General Interrupts header file

#pragma origin 4 //Manditory when interrupts are used

#define TRUE        1
#define FALSE       0

bit print_it;
bit start_record;

uns8 data_in;
uns8 data_last1;
uns8 data_last2;
uns8 data_last3;

uns8 memory_array[8];
uns8 mem_spot;

interrupt serverX(void)
{
    int_save_registers
    char sv_FSR = FSR;  // save FSR if required

    if(RCIF) //UART Recieve Interrupt
    {
        data_in = RCREG;

        print_it = TRUE;

        //No clearing RCIF, must clear RCREG
        RCREG = 0;
    }

    FSR = sv_FSR;               // restore FSR if saved
    int_restore_registers
}

#include "c:\cc5x\Delay.c"   // Delays
#include "c:\cc5x\stdio.c"   // Software and Hardware based RS232 Signals

#pragma config |= 0x3F02 //HS Oscillator, CProtect off, WDT off

void main()
{
```

```
PORTA = 0b.0000.0000;
TRISA = 0b.0000.0000;  //0 = Output, 1 = Input

PORTB = 0b.0000.0000;
TRISB = 0b.0000.0010;  //0 = Output, 1 = Input - RB1 on the 16F628 is RX of the UART

uns8 x;

RB5 = 1; //Turn on the on-board LED
mem_spot = 0;
uns16 button_monitor = 0;

enable_uart_TX(0); //Turn on Transmit UART with TX Interupts Disabled
enable_uart_RX(1); //Turn on Receive UART with RX Interupts Enabled

char aux;
aux = 0x0000;


while(1)
{

    if (print_it == TRUE) //We have something to record
    {
        GIE = 0;

        print_it = FALSE;
        button_monitor = 0; //Reset the LED blinker


        //Store the incoming data to the memory array
        memory_array[mem_spot] = data_in;
        mem_spot++;

        if (mem_spot > 8) mem_spot = 0;


        if (data_in == 0x0D && mem_spot>3) //0x0D is ENTER in ASCII code
        {
            rs_out(12);

            data_last1 = memory_array[mem_spot-2];
            data_last2 = memory_array[mem_spot-3];
            data_last3 = memory_array[mem_spot-4];

            if(data_last3 == 'O')
            {
                switch (data_last2) {
                case '1':
                    if (data_last1 == 'A')
                    {
                        aux |= 0x08;

                    }
                    if (data_last1 == 'D')
                    {
                        aux &= 0xF7;

                    }
                    PORTA = aux;
                break;

               case '2':
                    if (data_last1 == 'A')
                    {
                        aux |= 0x04;

                    }
                    if (data_last1 == 'D')
                    {
                            aux &= 0xFB;

                        }
                        PORTA = aux;
                    break;

                  case '3':
                        if (data_last1 == 'A')
                        {
                            aux |= 0x02;

                        }
                        if (data_last1 == 'D')
                        {
                            aux &= 0xFD;
                        }
                        PORTA = aux;
                    break;

                   case '4':
                        if (data_last1 == 'A')
                        {
                            aux |= 0x01;

                        }
                        if (data_last1 == 'D')
                        {
                            aux &= 0xFE;

                        }
                        PORTA = aux;
                    break;

                    default:
                    break;
                    }

                }
             mem_spot = 0;
            }

            GIE = 1;

            CREN = 0;
            CREN = 1;
        }

        //Blinks the LED when the counter called button_monitor rolls over
        if(button_monitor == 0xFFFF)
            RB5 ^= 1;

        button_monitor++;
    }

}
```

PC side.

As we said we can use the HyperTerminal to send code to the PIC, but what we really like is a user friendly interface :). To do that we used Visual Basic 6 to program a simple graphic interface. It has on/off buttons for each relay and a COM port selector.

Let see the routine for the On/Off button:

```
Private Sub OutOn_Click(Index As Integer)
    MSComm1.Output = "O"
  Delay
    MSComm1.Output = Chr$(Index + &H30 + 1)
 Delay
    MSComm1.Output = "A"
 Delay
    MSComm1.Output = Chr$(13)
    LedOn (4 + Index)
End Sub
```

As you can see we are sending the characters on by one to the PIC between each character we have to wait a little time in order to let de PIC do his job.

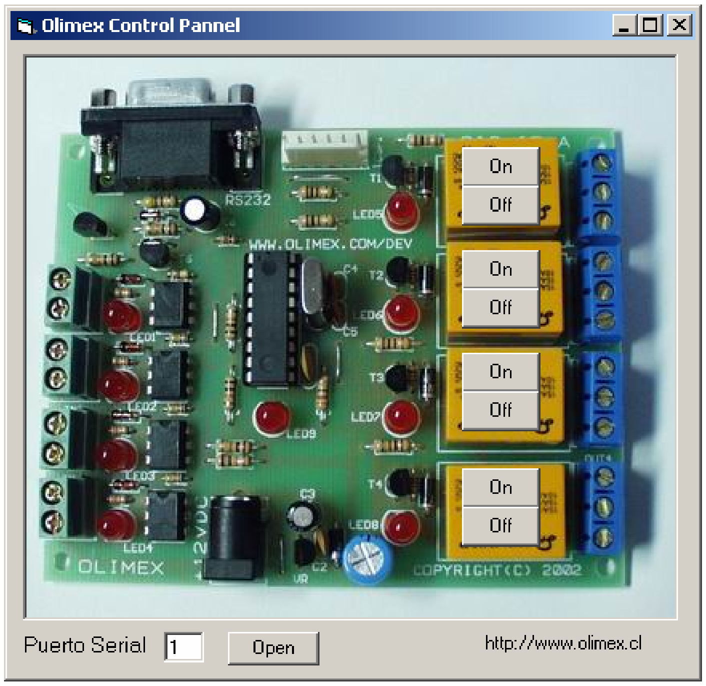
Let's take a look to this instruction:

```
MSComm1.Output = Chr$(Index + &H30 + 1)
```

Each button has a number associated to it when you call the routine the number of the button is given as parameter. The instruction above converts this number to an ASCII number before send it through the serial port. (take a look to the ASCII code here www.asciitable.com)

```
MSComm1.Output = Chr$(13)
```

Send an enter to the serial port (see the ASCII table 13 = 0x0D = enter)



To manage the serial port in VB6 we use this routine:

```
Private Sub Command1_Click()
    If Command1.Caption = "Open" Then
        MSComm1.CommPort = Text1.Text
        MSComm1.PortOpen = True
        Command1.Caption = "Close"
    Else
        MSComm1.PortOpen = False
        Command1.Caption = "Open"
    End If
End Sub

Private Sub CheckRxData(incomingChars As String)

    crPos = InStr(1, incomingChars, Chr$(13), vbBinaryCompare)
    If crPos <> 0 Then
        rxLine = Mid(incomingChars, 1, crPos - 1)

        For i = 2 To 5
            If Mid(rxLine, i, 1) = "A" Then
                LedOn (i - 2)
            Else
                LedOff (i - 2)
            End If
        Next i
    End If
End Sub


Private Sub MSComm1_OnComm()

    Select Case MSComm1.CommEvent
      Case comEventBreak
        Debug.Print "comEventBreak"
      Case comEventFrame
        Debug.Print "comEventFrame"
      Case comEventOverrun
        Debug.Print "comEventOverrun"
      Case comEventRxOver
        Debug.Print "comEventRxOver"
      Case comEventRxParity
        Debug.Print "comEventRxParity"
      Case comEventTxFull
        Debug.Print "comEventTxFull"
      Case comEventDCB
        Debug.Print "comEventDCB"
      Case comEvCD
        Debug.Print "comEvCD"
      Case comEvCTS
        Debug.Print "comEvCTS"
      Case comEvDSR
        Debug.Print "comEvDSR"
      Case comEvRing
        Debug.Print "comEvRing"
      Case comEvReceive
        CheckRxData (MSComm1.Input)
      Case comEvSend
        Debug.Print "comEvSend"
      Case comEvEOF
        Debug.Print "comEvEOF"
    End Select
End Sub
```

All the necessary files can be downloaded from http://www.olimex.cl/pic-io.zip
To use the interface you have to be sure that the COM port that you select is free. Once you select the port you have to click in the “open” button and then you can click any of the on/off buttons.

Have fun!