Controlando desde el PC con el PIC-IO

Paul Aguayo S., paguayo@olimex.cl

25 de noviembre de 2004

Índice

1. Introducción

El PIC-IO es una pequeña pero poderosa tarjeta de desarrollo que permite controlar 4 puertas de entradas optoacopladas y 4 relés de salida de 10A a 220V lo que permite encender o apagar la mayoría de los artefactos domésticos como lámparas, motores pequeños, aspersores, etc.

Aprovechando esta tarjeta que es practicamente un modulo discreto de un PLC podemos hacer control utilizando nuestro PC. La ideas es comunicar el PC con la tarjeta utilizando el puerto serial. Las señales de entrada pueden ser analizadas por nuestro pic y generar una salida y tambien pueden ser monitoreadas en un PC.

A continuación se mostrará parte del trabajo para ayudar al desarrollo de este tipo de sistemas. Lo que se implementará será una interfaz windows que permite encender y apagar los relés de la tarjeta presionando botones en la pantalla.

2. Materiales

Utilizaremos en esta oportunidad:

- Tarjeta de desarrollo: PIC I/O
- Microcontrolador: PIC16F628
- Programador: PIC PG1
- Cable serial
- Visual Basic 6.0
- stdio.c, del tutorial de comunicación serial

3. Objetivos

Lo que deseamos hacer es enviar un comando desde el PC hacia el PIC. Este comando es interpretado por el PIC de manera que este hace algo con el, en nuestro caso encender los relés.

Para esto vamos a adotar las siquientes reglas para el comando que enviaremos desde el PC:
O<numero de la linea><estado><cr>
Donde:

- O es la letra 'O' mayuscula (Output)
- <numero de la linea>es el numero de la linea que identifica al relé que queremos activar. Los valores admitidos son 1,2,3 y 4
- <estado>los valores admitidos son 'A'=Activado, 'D'=Desactivado
- <cr>es el codigo ASCII del Carriage Return (0x0D en hexagesimal o 13 en decimal)

Ejemplos:

- *O1A<enter>*
  activará el relé 1
- *O4D<enter>*
  desactivará el relé 4

Estos comandos pueden ser enviados a la tarjeta utilzando el programa Hyperterminal con las siguiente configuración *9600,8,N,1*. La luz de estado de la tarjeta debiera parpadear mientras el programa se encuentra en su operación normal.

Veamos como se hace lo anterior en C, por que lo hacemos en C? por que la idea es que sea entendible, una vez que estemos seguros que nuestro programa en C funciona podemos depurar el ASM para optimizar el código.

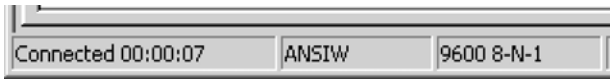


Figura 1: Configuración del Hyperterminal

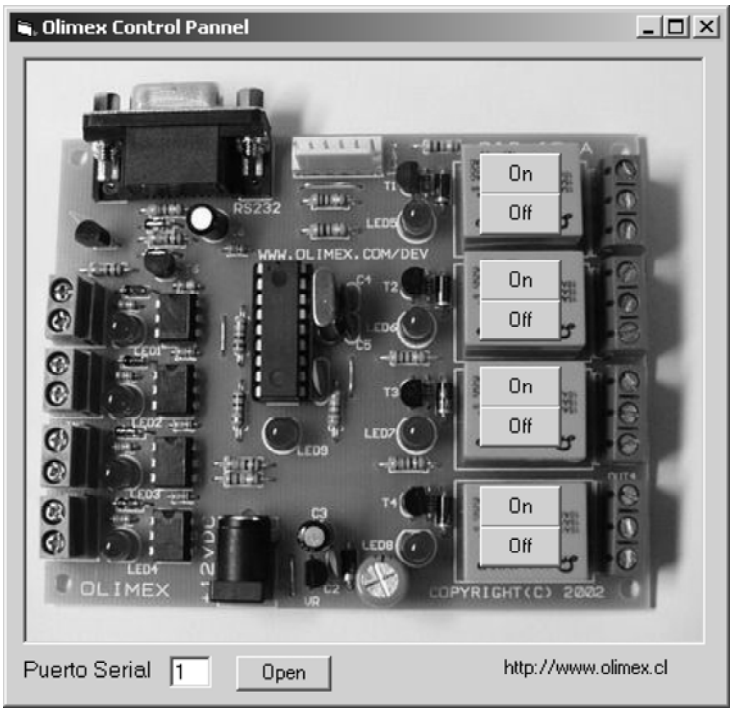
```
#define HS_Osc
#define Serial_Out_BB   RB2
#define Serial_In_BB   RB7
#define Baud_9600

#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)
            {
                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++;
  }

}
```

4. Interfaz con Visual Basic 6.0



Para utilizar esta interfaz, primero debes asegurarte de que no haya otra aplicación que esté utilizando el puerto COM. Luego debes abrir el puerto de comunicación y hacer click en los botones que activan y desactivan los relés.

Los archivos se encuentran disponibles para ser modificados en www.olimex.cl/pic-io.zip