Autor: Pedro I. López
Contacto: dreilopz@gmail.com | www.dreilopz.me

Fecha: Febrero 2012.

En ninguna circunstancia el autor se hace responsable de cualquier daño a cualquier persona o hardware causado por realizar lo descrito en este documento.

Práctica 10

Entradas y salidas digitales de un sistema DAQ

Objetivo

Al término de esta práctica el estudiante será capaz de utilizar un puerto digital y configurarlo como entrada o salida y conocerá las diferentes utilidades de las I/O Digitales.

Desarrollo

En esta sección se realizará un ejercicio básico de lectura y escritura de puertos digitales, el estudiante lograra observar el comportamiento de una entrada y una salida digital. Para lograr el objetivo del ejercicio la idea original era implementar un sistema de entradas y salidas digitales con la tarjeta DAQ6, la NI ELVIS y programa codificado en LabView.

Sin embargo, para lograr tal objetivo, se necesitaban unas librerías para codificar el instrumento virtual, librerías que no encontré disponibles en todo el curso (*Digital IO*, ver documento de laboratorio, página 140).

Aún así, decidí realizar la práctica con otros recursos, manteniendo el objetivo de la misma. En este mismo semestre y materia, mi equipo trabaja en un proyecto de interfaz USB para las tarjetas de adquisición de datos del laboratorio de la materia. Entonces la práctica consistió en construir una herramienta de software que escriba un byte de salida y lea uno de entrada.

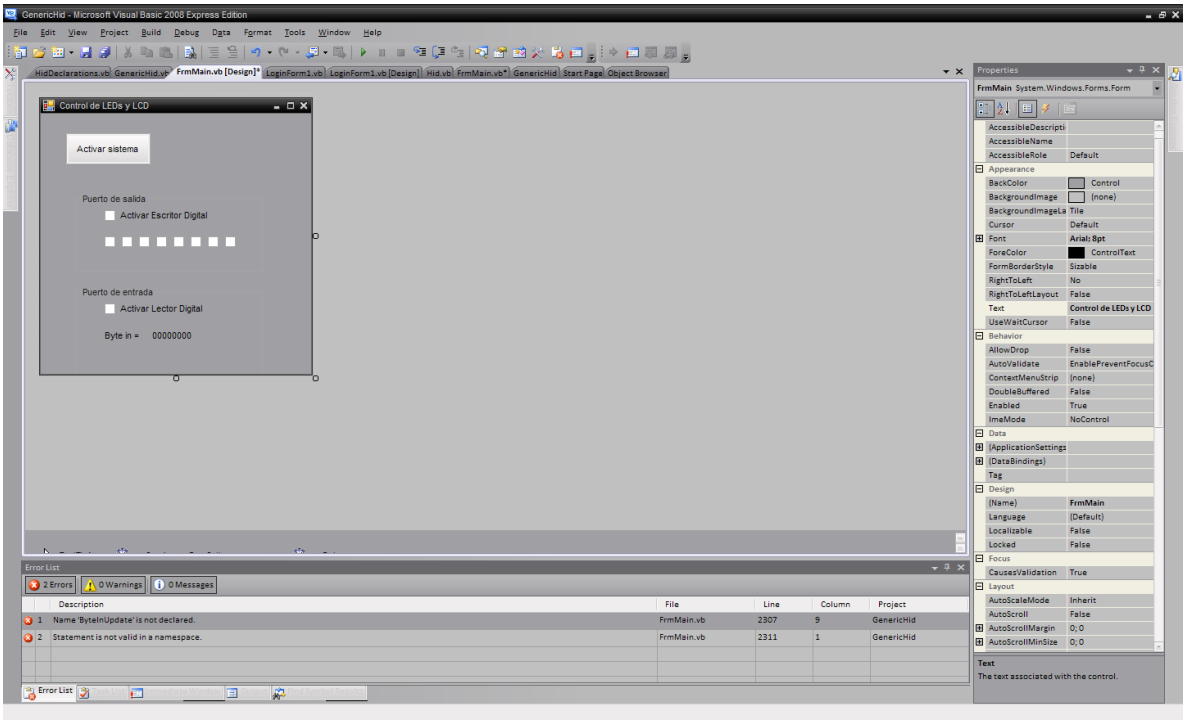
El programa para PC fue codificado en **Visual Basic 2008 Express Edition** debido a que tal paquete es muy útil para programar aplicaciones de usuario sencillas. El diseño es el siguiente. La PC será el host de la comunicación USB, y establecerá una comunicación digital con el microcontrolador para enviar y recibir comando que operen el lector y escritor digital. El microcontrolador es un PIC18F4550, y es el dispositivo USB, configurado para ser reconocido como un HID (*Human Interface Device*).

Para controlar las tareas de mantenimiento de bajo nivel del protocolo USB, se utilizan unas librerías que ofrecen un stack para operar con dispositivos USB. Las librerías son el stack MCHPFSUSB2.2, liberado por Microchip gratuitamente. A continuación el código de la aplicación VBNET.

```
    Public Sub ByteOutUpdate()
        outputReportBuffermask(0) = 3

        EnableSystem()
        If master_enable Then
            If byte_out_enable Then
                outputReportBuffermask(1) = 1
            Else
                outputReportBuffermask(1) = 2
            End If
        Else
            outputReportBuffermask(1) = 2
        End If

        byte_out = 0
        For index As Integer = 0 To 7
            byte_out = byte_out + byte_out_bits(index)
        Next
        outputReportBuffermask(2) = byte_out

        For index As Integer = 3 To 7
            outputReportBuffermask(index) = 0
```

```
        Next

        ExchangeInputAndOutputReports()
    End Sub

    Private Sub daqp0Init()

        master_enable = False
        byte_out_enable = False
        For index As Integer = 0 To 7
            byte_out_bits(index) = 0
        Next
        EnableSystem()
        MasterEnableUpdate()

    End Sub

    Private Sub byte_out0_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles byte_out0.CheckedChanged
        If byte_out0.Checked Then
            byte_out_bits(0) = 1
        Else
            byte_out_bits(0) = 0
        End If
        ByteOutUpdate()
    End Sub


    Private Sub byte_out1_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles byte_out1.CheckedChanged
        If byte_out1.Checked Then
            byte_out_bits(1) = 2
        Else
            byte_out_bits(1) = 0
        End If
        ByteOutUpdate()
    End Sub

    Private Sub byte_out2_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles byte_out2.CheckedChanged
        If byte_out2.Checked Then
            byte_out_bits(2) = 4
        Else
            byte_out_bits(2) = 0
        End If
        ByteOutUpdate()
    End Sub

    Private Sub byte_out3_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles byte_out3.CheckedChanged
        If byte_out3.Checked Then
            byte_out_bits(3) = 8
        Else
            byte_out_bits(3) = 0
        End If
        ByteOutUpdate()
    End Sub

    Private Sub byte_out4_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles byte_out4.CheckedChanged
        If byte_out4.Checked Then
            byte_out_bits(4) = 16
        Else
            byte_out_bits(4) = 0
        End If
        ByteOutUpdate()
    End Sub

    Private Sub byte_out5_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles byte_out5.CheckedChanged
        If byte_out5.Checked Then
            byte_out_bits(5) = 32
        Else
            byte_out_bits(5) = 0
        End If
        ByteOutUpdate()
    End Sub

    Private Sub byte_out6_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles byte_out6.CheckedChanged
        If byte_out6.Checked Then
            byte_out_bits(6) = 64
        Else
            byte_out_bits(6) = 0
        End If
        ByteOutUpdate()
    End Sub

    Private Sub byte_out7_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles byte_out7.CheckedChanged
        If byte_out7.Checked Then
            byte_out_bits(7) = 128
        Else
            byte_out_bits(7) = 0
        End If
        ByteOutUpdate()
    End Sub


    Private Sub byteoutenable_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles byteoutenable.CheckedChanged
        If byteoutenable.Checked Then
            byte_out_enable = True
            byteoutenable.Text = "Desactivar Escritor Digital"
        Else
            byte_out_enable = False
            byteoutenable.Text = "Activar Escritor Digital"
        End If
```

```
        ByteOutUpdate()

    End Sub

    Private Sub masterenable_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
masterenable.Click
        master_enable = Not (master_enable)
        If master_enable Then
            masterenable.Text = "Desactivar sistema"
            EnableSystem()
        Else
            masterenable.Text = "Activar sistema"
            EnableSystem()
        End If

        MasterEnableUpdate()

    End Sub

    Private Sub MasterEnableUpdate()
        If master_enable Then
            outputReportBuffermask(0) = 2
        Else
            outputReportBuffermask(0) = 1
        End If
        ExchangeInputAndOutputReports()
    End Sub

    Private Sub EnableSystem()
        If master_enable Then
            If byte_out_enable Then
                byte_out0.Enabled = True
                byte_out1.Enabled = True
                byte_out2.Enabled = True
                byte_out3.Enabled = True
                byte_out4.Enabled = True
                byte_out5.Enabled = True
                byte_out6.Enabled = True
                byte_out7.Enabled = True
            Else
                byte_out0.Enabled = False
                byte_out1.Enabled = False
                byte_out2.Enabled = False
                byte_out3.Enabled = False
                byte_out4.Enabled = False
                byte_out5.Enabled = False
                byte_out6.Enabled = False
                byte_out7.Enabled = False
            End If
        Else
            byte_out0.Enabled = False
            byte_out1.Enabled = False
            byte_out2.Enabled = False
            byte_out3.Enabled = False
            byte_out4.Enabled = False
            byte_out5.Enabled = False
            byte_out6.Enabled = False
            byte_out7.Enabled = False
        End If

    End Sub


    Private Sub byteinenable_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles byteinenable.CheckedChanged
        If byteinenable.Checked Then
            byte_in_enable = True
            byteinenable.Text = "Desactivar Lector Digital"
        Else
            byte_in_enable = False
            byteinenable.Text = "Activar Lector Digital"
        End If
        ByteInUpdate()
    End Sub
End Class

Private Sub ByteInUpdate()
    outputReportBuffermask(0) = 4

    EnableSystem()
    If master_enable Then
        If byte_out_enable Then
            outputReportBuffermask(1) = 1
        Else
            outputReportBuffermask(1) = 2
        End If
    Else
        outputReportBuffermask(1) = 2
    End If

    byte_out = 0
    For index As Integer = 0 To 7
        byte_out = byte_out + byte_out_bits(index)
    Next
    outputReportBuffermask(2) = byte_out

    For index As Integer = 3 To 7
        outputReportBuffermask(index) = 0
    Next

    ExchangeInputAndOutputReports()
End Sub
```



f10- 1. Captura de proceso de codificación de aplicación VBNET para PC

Como se puede apreciar en *f10-1* con el número de pestañas abiertas, la comunicación USB utiliza muchas definiciones y procedimientos que forman la mayor parte del código, por eso se omiten en el código anterior, de igual manera se hará al presentar el código del PIC.

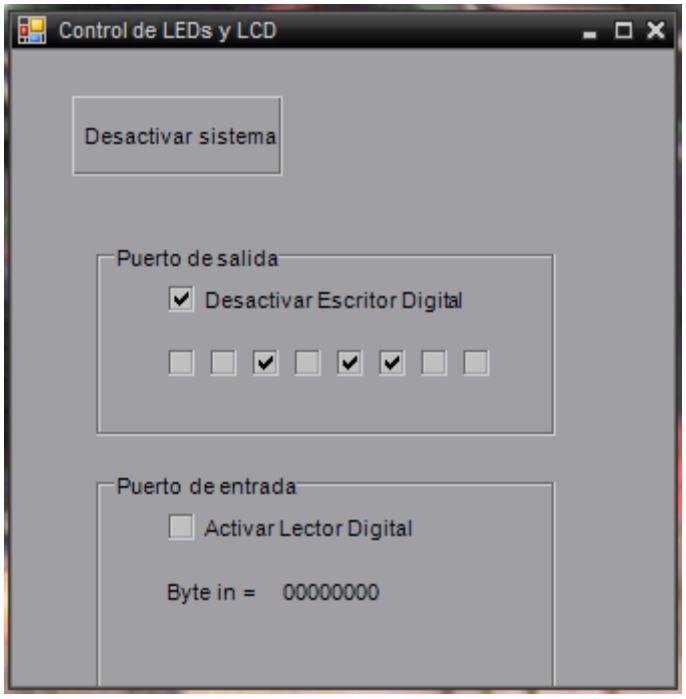
```
/** INCLUDES *******************************************************/
#include "GenericTypeDefs.h"
#include "Compiler.h"
#include "usb_config.h"
#include "./USB/usb_device.h"
#include "./USB/usb.h"
#include "generic_hid.h"
#include <p18f4550.h>
#include <delays.h>
#include <xlcd.h>
#include <stdlib.h>
#include "daqp0.h"
#include "HardwareProfile.h"
//#include "generic_hid.c"

/** VARIABLES ******************************************************/
char ade, boe, fge, me, boe_past, bie;
unsigned char bi[8], ad, byte_out_, byte_in_, bo[8], byte_in_bits[8];
/*********************************************************************/
/*********************************************************************/


/*********************************************************************
 * Function:        void ConfigByteIn()
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 * Overview:
 *
 * * Note:
 *
 ********************************************************************/
void ConfigByteIn(void)
{
    int i;

    switch(bi[1])
    {
              case bienabled:
                       bie=1;
              break;

              case binenabled:
                       bie=0;
              break;
```

```
    }
}


/*********************************************************************************
 * Function:        void ByteIn(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        Rutina de tareas principales
 *
 *
 * Note:
 *
 ********************************************************************************/
void ByteIn(void)
{
    int i;

    byte_in_=0;

    byte_in_bits[0]=BI0;
    byte_in_bits[1]=BI1;
    byte_in_bits[2]=BI2;
    byte_in_bits[3]=BI3;
    byte_in_bits[4]=BI4;
    byte_in_bits[5]=BI5;
    byte_in_bits[6]=BI6;
    byte_in_bits[7]=BI7;

    if(me==1)
    {
              if(bie==1)
              {
                        for(i=0;i<=7;i++)
                        {
                                  if(byte_in_bits[i]==1)
                                  {
                                            switch(i)
                                            {
                                                      case 0:
                                                                byte_in_bits[i]=1;
                                                      break;
                                                      case 1:
                                                                byte_in_bits[i]=2;
                                                      break;
                                                      case 2:
                                                                byte_in_bits[i]=4;
                                                      break;
                                                      case 3:
                                                                byte_in_bits[i]=8;
                                                      break;
                                                      case 4:
                                                                byte_in_bits[i]=16;
                                                      break;
                                                      case 5:
                                                                byte_in_bits[i]=32;
                                                      break;
                                                      case 6:
                                                                byte_in_bits[i]=64;
                                                      break;
                                                      case 7:
                                                                byte_in_bits[i]=128;
                                                      break;
                                            }
                                  }
                                  else
                                  {
                                            byte_in_bits[i]=0;
                                  }
                        }
              }
              else
              {
                        for(i=0;i<=7;i++)
                        {
                                  byte_in_bits[i]=0;
                        }
              }
    }
    else
    {
              for(i=0;i<=7;i++)
              {
                        byte_in_bits[i]=0;
              }
    }

    for(i=0;i<=7;i++)
    {
              byte_in_=byte_in_bits[0];
    }
}

/*********************************************************************************
 * Function:        void ConfigByteOut()
 *
 * PreCondition:    None
 *
```

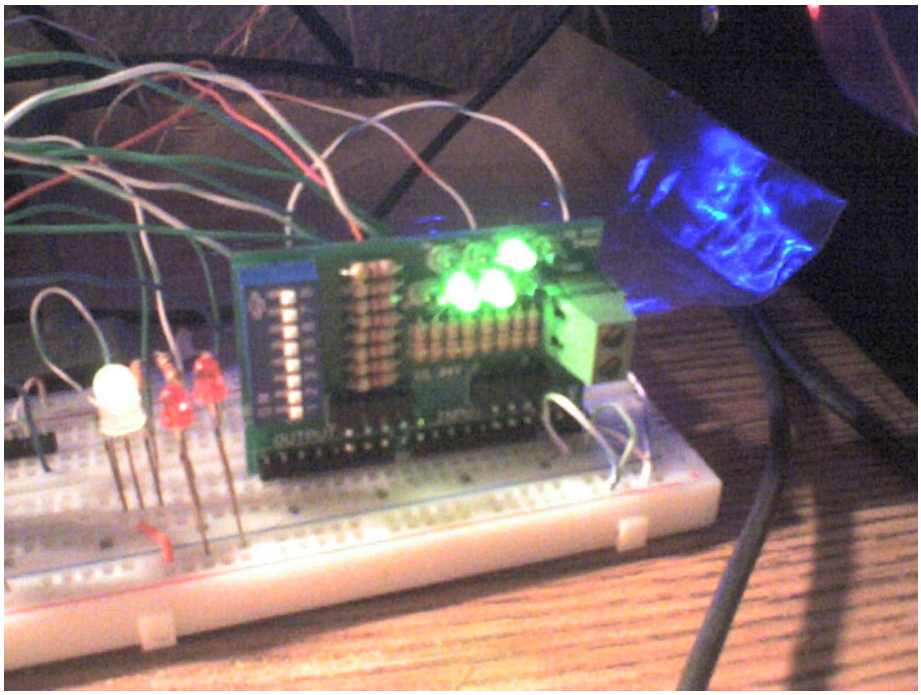
```
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 * Overview:
 *
 * * Note:
 *
 ********************************************************************************/

void ConfigByteOut(void)
{
    switch(bi[1])
    {
              case boenabled:
                        boe=1;
                        byte_out_=bi[2];
              break;

              case bonenabled:
                        boe=0;
              break;
    }
}

/********************************************************************************
 * Function:        void ByteOut(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        Rutina de tareas principales
 *
 *
 * Note:
 *
 ********************************************************************************/
void ByteOut(void)
{
    if(me==1)
    {
              if(boe==1)
              {
                        BO0=(byte_out_&0b00000001);
                        BO1=(byte_out_&0b00000010)>>1;
                        BO2=(byte_out_&0b00000100)>>2;
                        BO3=(byte_out_&0b00001000)>>3;
                        BO4=(byte_out_&0b00010000)>>4;
                        BO5=(byte_out_&0b00100000)>>5;
                        BO6=(byte_out_&0b01000000)>>6;
                        BO7=(byte_out_&0b10000000)>>7;
              }
              else
              {
                        BO0=0; //byte_out in zeroes
                        BO1=0;
                        BO2=0;
                        BO3=0;
                        BO4=0;
                        BO5=0;
                        BO6=0;
                        BO7=0;
              }
    }
    else
    {
              BO0=0; //byte_out in zeroes
              BO1=0;
              BO2=0;
              BO3=0;
              BO4=0;
              BO5=0;
              BO6=0;
              BO7=0;
    }
}

/********************************************************************************
 * Function:        void Gate()
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 * Overview:
 *
 * * Note:
 *
 ********************************************************************************/
void Gate(void)
{
    int i;

    for(i=0; i<=7; i++)
    {
              bi[i]=hid_report_out[i];
    }
```

```
    switch(bi[0])
    {
              case unimplemented:
                        Nop();
              break;

              case menabled:
                        me=1;
              break;

              case mnenabled:
                        me=0;
              break;

              case byte_out:
                        ConfigByteOut();
              break;

              case byte_in:
                        ConfigByteIn();
              break;
    }
}



/*******************************************************************************
 * Function:        void daqp0(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        Rutina de tareas principales
 *
 *
 * Note:
 *
 ******************************************************************************/

void daqp0(void)
{
    Gate();
    ByteOut();
    ByteIn();
}
```

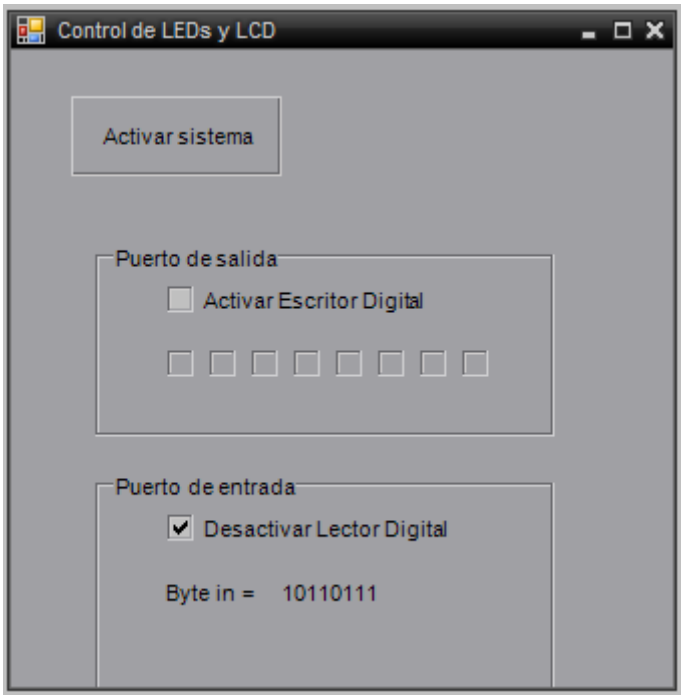
El código del PIC fue compilado con MC18 y ensamblado con MPASM en MPLAB.



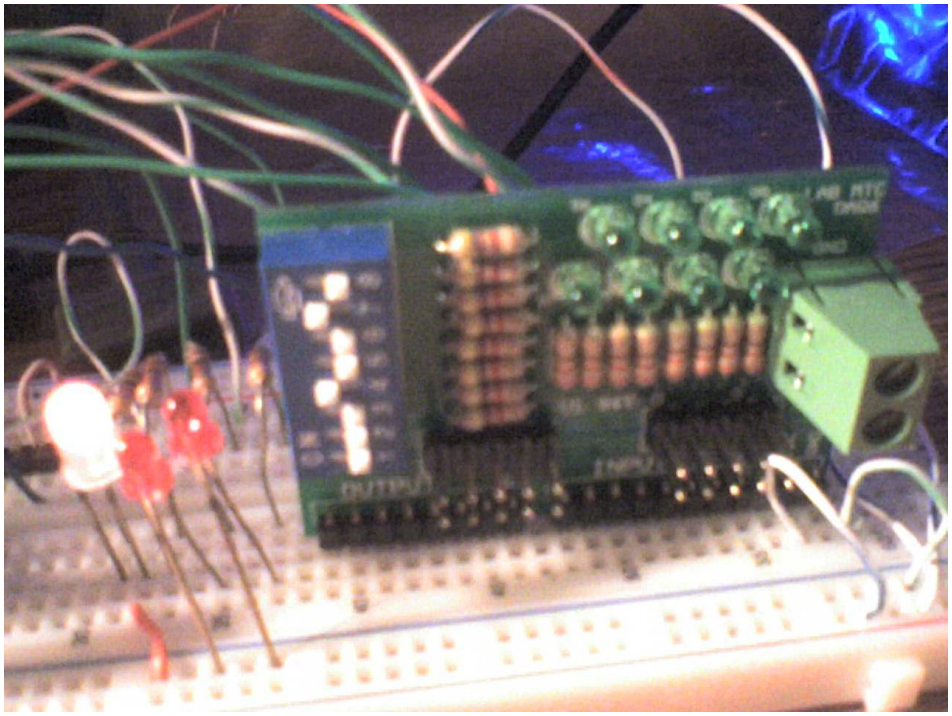
f10- 2. Programa en ejecución escribiendo un 00101100



f10- 3. Tarjeta ejecutando comando del PIC



f10- 4. Programa en ejecución leyendo un 10110111



f10- 5. Tarjeta enviando el byte de entrada

Las siguientes líneas son definiciones de las entradas y salidas, para poder apreciar como fue conectada la DAQ6 al PIC18F4550.

```
//Funciones

void daqp0(void);

void TransactionOut(void);

void Gate(void);

void ConfigByteOut(void);

void ByteOut(void);

void ConfigByteIn(void);

void ByteIn(void);

void ADConversion(void);


/** DECLARATIONS ******************************************************/
//pines de ByteOut
#define BO0    LATBbits.LATB0
#define BO1    LATBbits.LATB1
#define BO2    LATBbits.LATB2
#define BO3    LATBbits.LATB3
#define BO4    LATBbits.LATB4
#define BO5    LATBbits.LATB5
#define BO6    LATBbits.LATB6
#define BO7    LATBbits.LATB7
#define TRISBO0          TRISBbits.TRISB0
#define TRISBO1          TRISBbits.TRISB1
#define TRISBO2          TRISBbits.TRISB2
#define TRISBO3          TRISBbits.TRISB3
#define TRISBO4          TRISBbits.TRISB4
#define TRISBO5          TRISBbits.TRISB5
#define TRISBO6          TRISBbits.TRISB6
#define TRISBO7          TRISBbits.TRISB7

//pines de ByteIn
#define BI0    mLED_3
#define BI1    mLED_4
#define BI2    LATDbits.LATD4
#define BI3    LATDbits.LATD5
#define BI4    LATDbits.LATD6
#define BI5    LATDbits.LATD7
#define BI6    LATCbits.LATC6
#define BI7    LATCbits.LATC7
#define TRISBI0          TRISDbits.TRISD2
#define TRISBI1          TRISDbits.TRISD3
#define TRISBI2          TRISDbits.TRISD4
```

```
#define TRISBI3         TRISDbits.TRISD5
#define TRISBI4         TRISDbits.TRISD6
#define TRISBI5         TRISDbits.TRISD7
#define TRISBI6         TRISCbits.TRISC6
#define TRISBI7         TRISCbits.TRISC7

//tercer conjunto de pines
//#define BO0 LATEbits.LATE0
//#define BO1 LATEbits.LATE1
//#define BO2 LATEbits.LATE2
//#define BO3 LATAbits.LATA0
//#define BO4 LATAbits.LATA2
//#define BO5 LATAbits.LATA3
//#define BO6 LATAbits.LATA4
//#define BO7 LATAbits.LATA5
//#define TRISBO0       TRISEbits.TRISE0
//#define TRISBO1       TRISEbits.TRISE1
//#define TRISBO2       TRISEbits.TRISE2
//#define TRISBO3       TRISAbits.TRISA0
//#define TRISBO4       TRISAbits.TRISA2
//#define TRISBO5       TRISAbits.TRISA3
//#define TRISBO6       TRISAbits.TRISA4
//#define TRISBO7       TRISAbits.TRISA5

#define unimplemented   0x00
#define mnenabled               0x01
#define menabled                0x02
#define byte_out                0x03
#define byte_in                 0x04
#define check                   0x06

#define trivial                         0x01

#define boenabled               0x01
#define bonenabled              0x02

#define bienabled               0x01
#define binenabled              0x02

#define good                    0xab
#define bad                             0xac
```

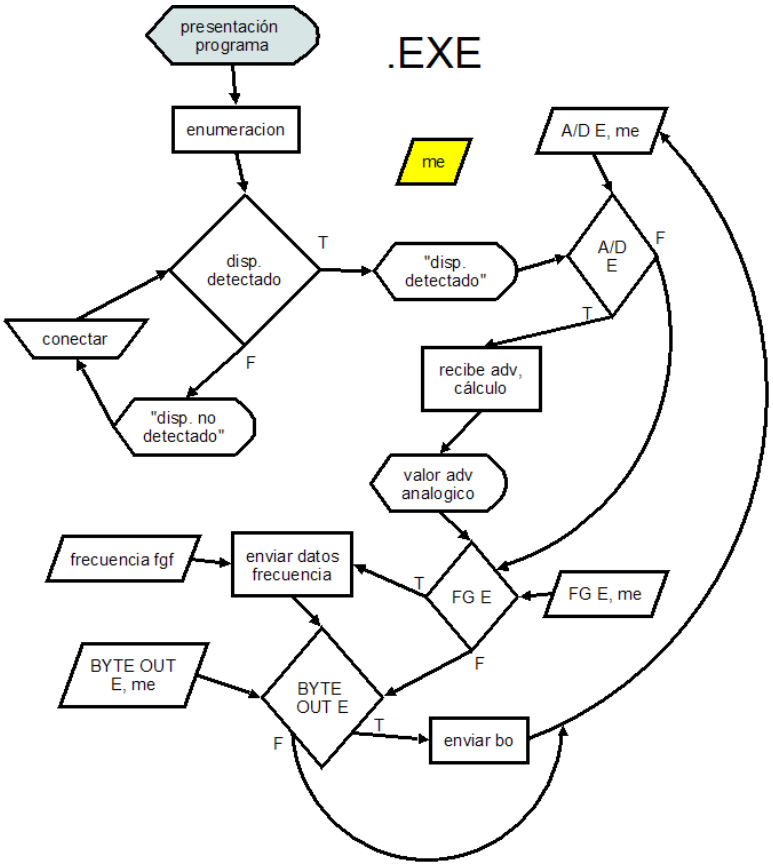
Reporte

Describa la actividad realizada en la experimentación
(Ver sección *Desarrollo*).

Realice el diagrama de flujo y de bloques para el programa
Este es el diagrama de flujo (*f10-6*) del programa usado, con las modificaciones acorde al proyecto de Adquisición de Datos (el Byte In viene de la salida de un convertidor A/D, y hay otro Byte de Salida que controla un convertidor D/A). Diagrama de bloques en *f10-7*.

¿Cómo se realizó la configuración de entradas y salidas?
Con las instrucciones de configuración de puertos del microcontrolador, que se pueden ver en el código siguiente.

```
/*********************************************************************
 * Function:        void UserInit(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        This routine should take care of all of the demo code
 *                  initialization that is required.
 *
 * Note:
 *
 ********************************************************************/
void UserInit(void)
{
    //Initialize all of the LED pins
    mInitAllLEDs();

    old_sw2 = sw2;
    old_sw3 = sw3;

     //todo digital, expcepto RA1

     //initialize PORTB
```



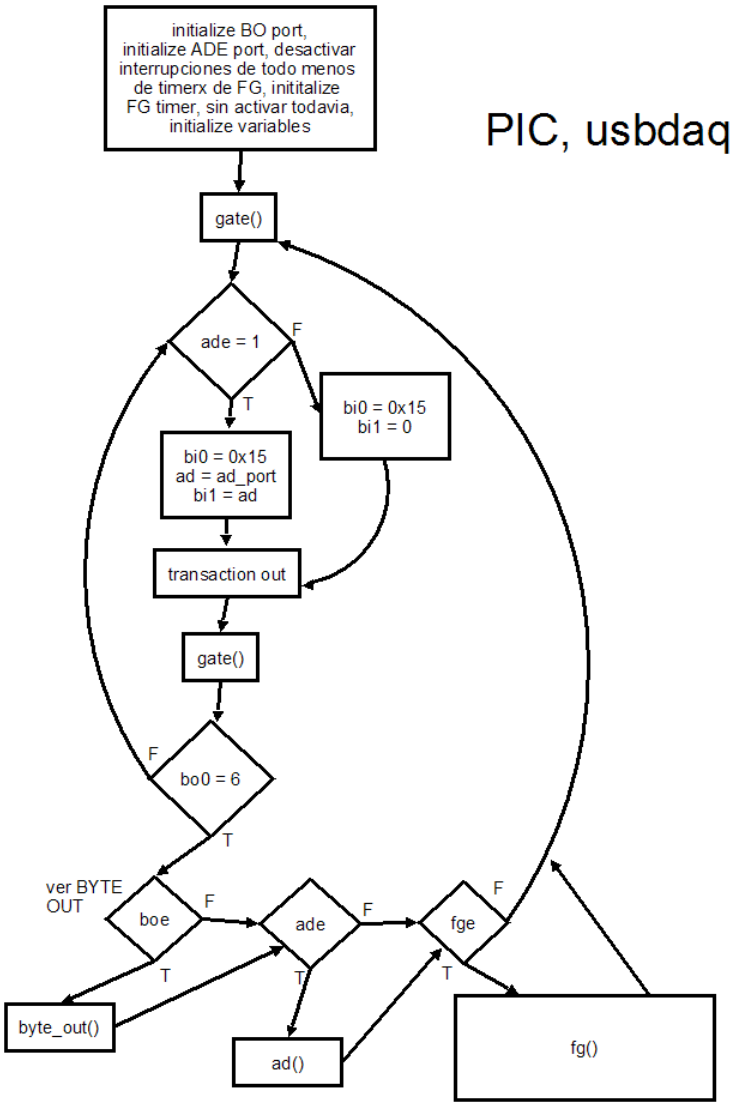

```
PORTB=0;
          LATB=0;
          ADCON1=0x0e;

          //initialize PORTD
          PORTD=0;
          LATD=0;
          TRISD=0;

          //initialize PORTE
          PORTE=0;
          LATE=0;
          ADCON1bits.PCFG0=1;
          ADCON1bits.PCFG1=0;
          ADCON1bits.PCFG2=1;
          ADCON1bits.PCFG3=1;
          CMCON=0x07;

          //ByteOut  PORT  configured  as
output
          TRISBO0=0;
          TRISBO1=0;
          TRISBO2=0;
          TRISBO3=0;
          TRISBO4=0;
          TRISBO5=0;
          TRISBO6=0;
          TRISBO7=0;

          //ByteIn  PORT  configured  as
input
          TRISBI0=1;
          TRISBI1=1;
          TRISBI2=1;
          TRISBI3=1;
          TRISBI4=1;
          TRISBI5=1;
          TRISBI6=1;
          TRISBI7=1;

          Gate();

}//end UserInit
```

f10- 6. Diagrama de flujo

Conclusión

Un sistema de adquisición siempre cuenta con terminales de entrada y salida digital. A veces las terminales digitales son de propósito general, y otra veces tienen una función específica, entre las que pueden estar establecer lazos de comunicación digital con protocolo, para conversión de datos A/D o D/A, pin especiales como de reset o de interrupción, módulo de captura de datos o PWM, etcétera).

Es importante saber un funcionamiento general o universal de las entradas y salidas de un sistema, para poder configurarlas correctamente cuando se ofrezca trabajar con ellas. Después de iniciar el sistema normalmente, hay que cerciorarnos que la terminal se encuentra libre, y configurar para que trabaje como digital (en caso de que no esté así por default, por ejemplo si es un canal de adquisición analógica de señales). Después se escoge si se desea que se entrada o salida, y se configura para tal función. Después de un reset del puerto y una pequeña demora para estabilización, la terminal está lista para utilizarse. Ahora se puede disponer de ella para ejecutar tareas simples como leer estados lógicos estacionarios o tareas complicadas como controlar un convertidor D/A o un servomotor.



f10- 7. Diagrama de bloques

Bibliografía

- **Documento de práctica de laboratorio**
  Práctica 10 – Entradas y salidas digitales de un sistema DAQ
  Laboratorio de Adquisición de Datos
  FIME – UANL

- **PIC18F4550 data sheet**
  Microchip
  2007

- **MCHPFSUSB Firmware User's Guide.**
  Microchip
  2007