INFORME DE LABORATORIO 1: MICROCONTROLADORES

Luis Miguel Rincon Pinilla
e-mail: lmrinconp@itc.edu.co
Cristian Camilo Pianda Rodríguez
e-mail: ccpiandar@itc.edu.co

RESUMEN: *En este informe se detalla los procedimientos seguidos y los resultados que se obtuvieron como parte de la primera interacción con el entorno de programación MPLAB IDE en el marco del primer laboratorio de Microcontroladores. En el cual se adquirieron las destrezas necesarias para la programación del microcontrolador PIC18F4550 por medio de la modificación de sus registros de configuración.*

PALABRAS CLAVE: C++, Compilador, Microcontrolador, Oscilador, Programación, Registro.

1 INTRODUCCIÓN

El medio de conexión para monitorear o controlar el hardware externo a los microcontroladores son las entradas y salidas digitales con las que cuenta. Por lo cual es necesario comprender la manera en que se configuran cada uno de los puertos con los que cuenta el PIC, como entradas o salidas dentro de la solución de un problema determinado.

2 OBJETIVOS

- Configurar el entorno de programación de MPLAB XC8 para la programación del PIC 18F4550.
- Inicializar adecuadamente los puertos digitales de entrada y salida para el PIC 18F4550
- Utilizar los puertos digitales de entrada y salida para recibir órdenes de un usuario y enviar información a través de estos.

3 ELEMENTOS UTILIZADOS

Se Utilizaron los siguientes elementos durante el laboratorio:

- Botones. (Simulador)
- Fuente de voltaje D.C. (Simulador)
- LEDs. (Simulador)
- Microcontrolador PIC18F4550. (Simulador)
- MPLAB X IDE 5.40.
- Osciloscopio. (Simulador)
- Proteus 8.9.
- Resistencias varias. (Simulador)

4 DESARROLLO

A continuación se describe el proceso de diseño y programación del sistema que da solución al problema planteado.

4.1 MONTAJE DEL CIRCUITO

Primeramente, se realizó el montaje del circuito con un microcontrolador PIC18F4550 (*Fig. 1*) con las siguientes características:

- Se conecta un osciloscopio al pin 3 (RA1).
- Se conecto un pulsador con una resistencia de pull-down al pin 4 (RA2).
- Se conectaron 4 Leds (Contador BCD) a los pines 5, 6, 7, 14 (RA3-RA6).
- Se conectaron 8 leds a los pines correspondientes al puerto B.
- Se conectaron 8 leds a los pines correspondientes al puerto D.

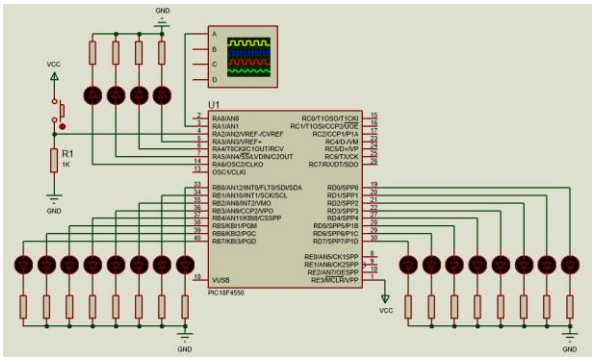


Figura 1. Montaje del Circuito.

4.2 REGISTROS DE CONFIGURACIÓN

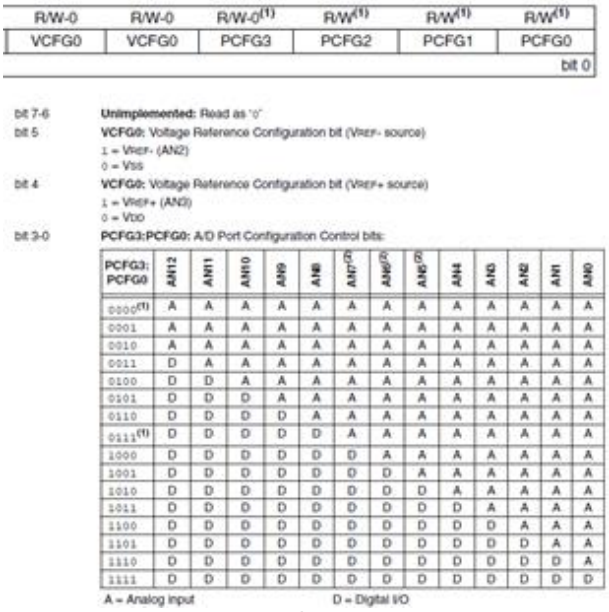
Al completar el montaje del circuito en el simulador se encontró un desafío particular con el puerto RA2, al realizarse la lectura del bit RA2 no se generaba ningún cambio en el mismo al presionar el pulsador conectado.

Luego de varias horas de consulta en el manual del microcontrolador se encontró que el registro correspondiente a RA2 como entrada tenía la siguiente condición “PORTA<2> data input. Disabled when analog functions enabled; disabled when CVREF output enabled.” Lo cual significa que la entrada digital va a estar deshabilitada mientras la conversión A/D estuviera habilitada.

Para deshabilitar la entrada análoga y la función Vref del bit RA2 era necesario modificar el registro **ADCON1** (*Fig. 2*) que se configuró de la siguiente manera:

- Bits 6 y 7 no es necesaria su modificación pues no están implementados.
- Bit 5 en 0 para usar Vss en vez de RA2 como Vref-.

- Bit 4 en 0 para usar Vdd en vez de RA3 como Vref+.
- Bits 3-0 como 1111 para usar todas las entradas AN como digitales.



R/W-0	R/W-0	R/W-0(1)	R/W(1)	R/W(1)	R/W(1)
VCFG0	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
					bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **VCFG0:** Voltage Reference Configuration bit (VREF- source)
1 = VREF- (AN2)
0 = VSS

bit 4 **VCFG0:** Voltage Reference Configuration bit (VREF+ source)
1 = VREF+ (AN3)
0 = VDD

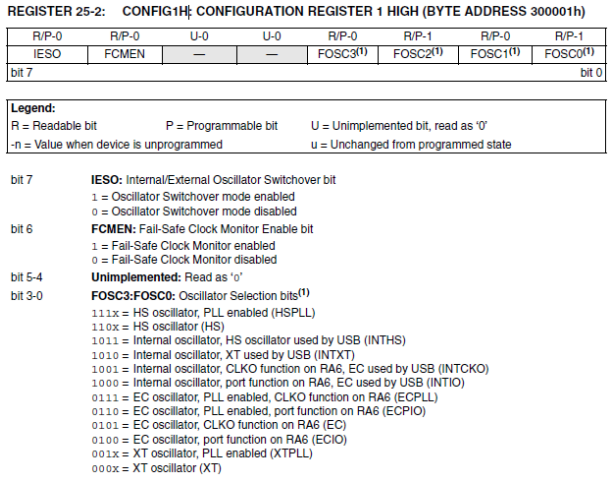
bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits:

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7(2)	AN6(2)	AN5(2)	AN4	AN3	AN2	AN1	AN0
0000(1)	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111(1)	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O

Figura 2. Configuración ADCON1 (Pag. 262)

El segundo desafío que se encontró fue la implementación del contador BCD de 4 bits en el puerto A, para facilitar la conexión en 4 bits consecutivos del puerto era necesario usar RA3-RA6 pero RA6 no se habilita como salida debido a que es uno de los pines asignados al oscilador externo, luego de revisar el manual se encontró que el PIC18F4550 cuenta con un oscilador interno pero para activarlo era necesario modificar el registro **CONFIG1H** (*Fig. 3*), el cual configuramos con la declaración #pragma config FOSC = INTOSCIO_EC.



REGISTER 25-2: CONFIG1H: CONFIGURATION REGISTER 1 HIGH (BYTE ADDRESS 300001h)

R/P-0	R/P-0	U-0	U-0	R/P-0	R/P-1	R/P-0	R/P-1
IESO	FCMEN	—	—	FOSC3(1)	FOSC2(1)	FOSC1(1)	FOSC0(1)
bit 7							bit 0

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
-n = Value when device is unprogrammed		u = Unchanged from programmed state

bit 7 **IESO:** Internal/External Oscillator Switchover bit
1 = Oscillator Switchover mode enabled
0 = Oscillator Switchover mode disabled

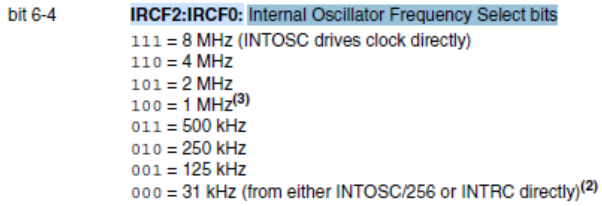
bit 6 **FCMEN:** Fail-Safe Clock Monitor Enable bit
1 = Fail-Safe Clock Monitor enabled
0 = Fail-Safe Clock Monitor disabled

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **FOSC3:FOSC0:** Oscillator Selection bits(1)
111x = HS oscillator, PLL enabled (HSPLL)
110x = HS oscillator (HS)
1011 = Internal oscillator, HS oscillator used by USB (INTHS)
1010 = Internal oscillator, XT used by USB (INTXT)
1001 = Internal oscillator, CLKO function on RA6, EC used by USB (INTCKO)
1000 = Internal oscillator, port function on RA6, EC used by USB (INTIO)
0111 = EC oscillator, PLL enabled, CLKO function on RA6 (ECPLL)
0110 = EC oscillator, PLL enabled, port function on RA6 (ECPIO)
0101 = EC oscillator, CLKO function on RA6 (EC)
0100 = EC oscillator, port function on RA6 (ECIO)
001x = XT oscillator, PLL enabled (XTPLL)
000x = XT oscillator (XT)

Figura 3. Configuración CONFIG1H (Pag. 290)

Finalmente encontramos que lo tiempos de delay en la simulación no coincidían con los que se usaban en el código, para solucionarlo encontramos que era necesario modificar el registro OSCCON específicamente en los bits para IRCF (Internal Oscillator Frequency Select bits) los cuales se configuraron en 111 para seleccionar el reloj del microcontrolador en 8 MHz.



bit 6-4 **IRCF2:IRCF0:** Internal Oscillator Frequency Select bits
111 = 8 MHz (INTOSC drives clock directly)
110 = 4 MHz
101 = 2 MHz
100 = 1 MHz(3)
011 = 500 kHz
010 = 250 kHz
001 = 125 kHz
000 = 31 kHz (from either INTOSC/256 or INTRC directly)(2)

Figura 3. Configuración OSCON:IRCF (Pag. 34)

4.3 PROGRAMACIÓN

Para la selección de los 4 modos de operación se uso switch case configurado de la siguiente manera:

- Case 1: Mover un "1" de izquierda a derecha y de derecha a izquierda a lo largo del puerto D del PIC 18F4550, indefinidamente. Utilizar leds para visualizar el desplazamiento del bit.
- Case 2: Encender un led del puerto cada 2 segundos de forma consecutiva de tal manera que al finalizar el bit 8 del puerto B se tenga una línea de leds encendidos. Al encender todos los leds debe empezar a apagar de a un led cada vez, hasta apagar completamente el puerto.
- Case 3: Utilizar 4 pines del puerto A para mostrar un conteo en código BCD que debe ir cambiando cada 1 segundo.
- Case 4: Utilizar el pin A1 como salida para generar una señal cuadrada de 500ms en estado bajo y 500 ms en estado alto.
- Default: Reinicia el contador a 0.

El código de programación en C que creamos para el microcontrolador es el siguiente:

```
#include <xc.h>
  #include <math.h>
  #define _XTAL_FREQ 8000000
  #pragma config FOSC = INTOSCIO_EC

  void main(void) {
    OSCCONbits.IRCF=0b111;
    int mode=0;
    ADCON1=0b00001111;
    TRISA=0b00000100;
    TRISB=0b00000000;
    TRISD=0b00000000;
    PORTA=0b00000000;
    PORTB=0b00000000;
    PORTD=0b00000000;

    while(1){
    if (PORTAbits.RA2==1){
      mode++;
    }

    switch (mode) {
      case 1:
        for (int i=7;i>=0;i--){
```

```
            PORTD=pow(2,i);
            __delay_ms(100);
        }
        for (int i=0;i<=7;i++){
            PORTD=pow(2,i);
            __delay_ms(100);
        }
    break;
    case 2 :
        PORTD=0x00;
        for (int i=0;i<=7;i++){
            PORTB+=pow(2,i);
            __delay_ms(2000);
        }
        for (int i=7;i>=0;i--){
            PORTB-=pow(2,i);
            __delay_ms(2000);
        }
    break;
    case 3:
        for (int i=0;i<=128;i+=8){
            PORTA=i;
            __delay_ms(1000);
        }
    break;
    case 4:
        PORTAbits.RA1=1;
        __delay_ms(500);
        PORTAbits.RA1=0;
        __delay_ms(500);
    break;
    default :
        mode=0;
        __delay_ms(200);
    break;
  }
  }
  return;
}
```

5 Enlace del Video

El video de la simulación se encuentra en:
https://youtu.be/eX8sGAImPqA

6 CONCLUSIONES

Lo expuesto en este informe de laboratorio permite arribar a las siguientes conclusiones:

- La configuración del entorno de programación de MPLAB para el PIC escogido se logró a través de la consulta del manual de referencia de este en el cual se encontró la manera de modificar los registros para cumplir con los requerimientos del problema planteado.
- Con la configuración de los registros se logró habilitar los puertos del microcontrolador para desarrollar la interfaz hombre-maquina requerida para el funcionamiento del automatismo.