Universidad del Valle de Guatemala

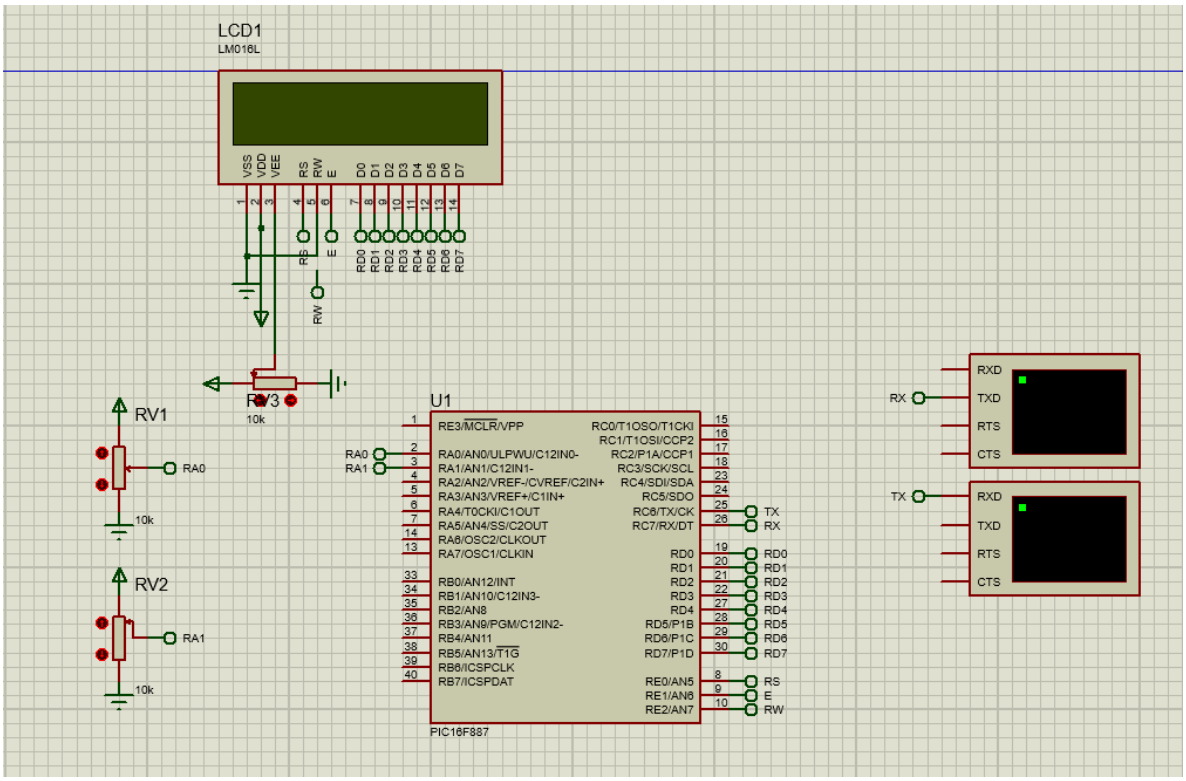
Digital 2 Sección 10

Rodrigo García 19085

Reporte Laboratorio 3

Link a Repositorio: https://github.com/gar19085/Digital2-Gar19085.git

Esquemático:

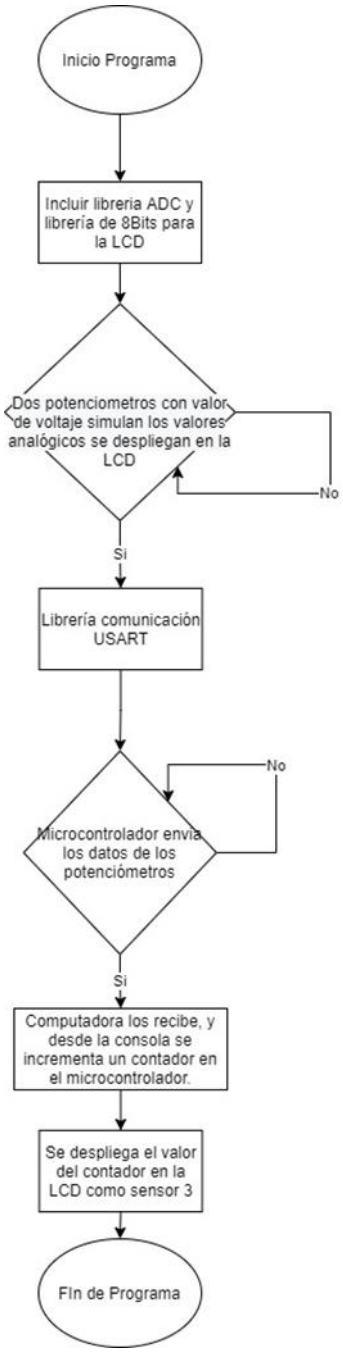


Descripción:

Configuración de Puertos:

Se utilizo el puerto D para poder conectar todos los pines del LCD para así poder escribir en el mismo, luego en los pines RE0-RE2 se habilitaron para así poder utilizar los pines E y RS para poder habilitar la LCD. En el puerto A solo se utilizaron los primeros dos pines ya que es en donde se habilitaron dos configuraciones ADC para que se encendieran el ANS0 y ANS1, los cuales funcionan mediante un toggle. En el puerto C se habilitaron los pines RC6 y RC7 para así poder configurar correctamente la enviada y recibida de datos necesarios entre la computadora y el PIC.

Seudocódigo:



Código:

PRINCIPAL

```
/*
 * File:   Lab031.c
 * Author: RODRIGO GARCIA
 *
 * Created on 4 de febrero de 2021, 05:04 PM
 */
```

```
// PIC16F887 Configuration Bit Settings

// 'C' source line config statements

// CONFIG1
#pragma config FOSC = INTRC_NOCLKOUT// Oscillator Selection bits (INTOSCIO o
scillator: I/O function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLK
IN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled a
nd can be enabled by SWDTEN bit of the WDTCON register)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config MCLRE = OFF      // RE3/MCLR pin function select bit (RE3/MCL
R pin function is digital input, MCLR internally tied to VDD)
#pragma config CP = OFF         // Code Protection bit (Program memory code
protection is disabled)
#pragma config CPD = OFF        // Data Code Protection bit (Data memory cod
e protection is disabled)
#pragma config BOREN = OFF      // Brown Out Reset Selection bits (BOR disab
led)
#pragma config IESO = OFF       // Internal External Switchover bit (Interna
l/External Switchover mode is disabled)
#pragma config FCMEN = OFF      // Fail-
Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)
#pragma config LVP = OFF        // Low Voltage Programming Enable bit (RB3 p
in has digital I/O, HV on MCLR must be used for programming)

// CONFIG2
#pragma config BOR4V = BOR40V   // Brown-out Reset Selection bit (Brown-
out Reset set to 4.0V)
#pragma config WRT = OFF        // Flash Program Memory Self Write Enable bi
ts (Write protection off)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>
#include <stdio.h>
#include <stdint.h>
#include <pic16f887.h>
#include "LCD.h"
#include "USART.h"
#include "ADC.h"
#include "Oscilador.h"
```

```
#define _XTAL_FREQ 4000000

uint8_t valorPOT1 = 0;
uint8_t valorPOT2 = 0;
uint8_t FLAG = 0;
uint8_t FLAG2 = 0;
uint8_t ADCGO = 0;
uint8_t CONTADOR;
uint8_t CONT = 0;


int CONT1;
int CONT2;
int CONT3;
int POT1;
int POT12;
int POT13;
int POT2;
int POT22;
int POT23;
char TURN = 0;
char FLG1;
char FLG2;
char VOLTAGE1;
char VOLTAGE2;

void Setup(void);
void ADCG(void);
void TURNO(void);
void INFOCONT(void);
void INFOADC1(void);
void INFOADC2(void);
const char* STRING(char C1, char C2, char C3);
const char* STRINGPOT1(char C1, char C2, char C3);
const char* STRINGPOT2(char C1, char C2, char C3);
//void TOGGLEADC(char FLG, char FLG2, char TURN1);

/*
 INTERRUPCIONES
 */

void __interrupt() isr(void){
        if(PIR1bits.ADIF==1){ //CONFIGURACIÓN PARA LAS INTERRUPCIONES DEL AD
C
            PIR1bits.ADIF = 0;
```

```
    }
        if(PIR1bits.RCIF == 1){
            CONTADOR = RCREG;
    }

        if(INTCONbits.TMR0IF == 1){   //CONFIGURACIÓN PARA UTILIZAR LA NTERR
UPCIÓN DE TMR0
            TMR0=236;
            ADCGO++;
            INTCONbits.TMR0IF = 0;
        }
}

/*
 MAIN LOOP
 */

void main(void) {
    initOsc(8);
    Setup();
    Conf_TXR();
    Conf_RXT();
    LCD_init();
    LCD_Cmd(0x8A);
    LCD_Goto(1,1);
    LCD_Print("VOLT1");
    LCD_Goto(7,1);
    LCD_Print("VOLT2");
    LCD_Goto(13,1);
    LCD_Print("CONT");
    while(1){
            if(TURN==0){
                valorPOT1 = ADRESH;
                initADC(1,0);
                TURN=1;
            }
            else if(TURN==1){
                valorPOT2 = ADRESH;
                initADC(1,1);
                TURN = 0;
            }
        ADCG();
        INFOCONT();
        INFOADC1();
        INFOADC2();
```

```
        VOLTAGE1 = valorPOT1;
        VOLTAGE2 = valorPOT2;
        TRANSMITIR(POT1);
        TRANSMITIR(POT12);
        TRANSMITIR(POT13);
        LCD_Goto(2,2);
        LCD_Print(STRINGPOT1(POT1, POT12, POT13));
        TRANSMITIR(POT2);
        TRANSMITIR(POT22);
        TRANSMITIR(POT23);
        LCD_Goto(8,2);
        LCD_Print(STRINGPOT2(POT2, POT22, POT23));
        LCD_Goto(6,2);
        LCD_Print("V");
        LCD_Goto(12,2);
        LCD_Print("V");
                if(CONTADOR==0x2b){
                    FLG1=1; //SE ACTIVA MI BANDERA
                }
                if(FLG1 == 1 && CONTADOR != 0x2b){
                    FLG1=0; //SE APAGA EL FLAG1
                    PORTB++;
                    CONT++;
                }
                if(CONTADOR==0x2d){ //MISMO PROCEDIMIENTO SOLO QUE ESTE REST
A VALORES AL CONTADOR
                    FLG2=1;
                }
                if(FLG2==1 && CONTADOR != 0x2d){
                    FLG2=0;
                    PORTB--;
                    CONT--;
                }
        LCD_Goto(14,2);
        LCD_Print(STRING(CONT1, CONT2, CONT3));
  }
}

/*
 CONFIGURACIÓN PRINCIPAL
 */

void Setup(void){
    PORTA = 0;//LIMPIEZA DE PUERTOS
    PORTB = 0;
```

```
    PORTC = 0;
    PORTD = 0;
    PORTE = 0;

    ANSEL = 0b00000011;//INDICO EL PRIMER PIN COMO ANALOGO
    ANSELH = 0;

    TRISA = 0b00000011;
    TRISB = 0;
    TRISC = 0b10000000;
    TRISD = 0;
    TRISE = 0;
    OPTION_REG = 0b00000011;
    INTCONbits.GIE = 1;//HABILITO LAS INTERRUPCIONES NECESARIAS, LA GLOBAL P
RINCIPALMENTE
    INTCONbits.PEIE = 1; //HABILITA LOS PERIPHERAL INTERRUPTS
    PIE1bits.ADIE = 1; //HABILILTO LAS INTERRUPCIONES DEL ADC
    PIR1bits.ADIF = 0;
    INTCONbits.T0IE = 1; //HABILITO LAS INTERRUPCIONES DEL TMR0
    INTCONbits.T0IF = 0;
}

/*
 MAPEO DE INFORMACIÓN
 */

void INFOCONT(void){
    CONT1 = CONT/100;
    CONT2 = ((CONT-(CONT1*100))/10);
    CONT3 = (CONT-(CONT1*100))-(CONT2*10);
    CONT1 = CONT1+0x30;
    CONT2 = CONT2+0x30;
    CONT3 = CONT3+0x30;
}

const char* STRING(char C1, char C2, char C3){
    char TEMP[3];
    TEMP[0] = C1;
    TEMP[1] = C2;
    TEMP[2] = C3;
    return TEMP;
}

void INFOADC1(void){
    POT1 = VOLTAGE1/51;
```

```
    POT12 = (VOLTAGE1-(POT1*51))/10;
    POT13 = (VOLTAGE1-(POT1*51))-(POT12*10);
    POT1 = POT1+0x30;
    POT12 = POT12+0x30;
    POT13 = POT13+0x30;
}

const char* STRINGPOT1(char C1, char C2, char C3){
    char TEMP[4];
    TEMP[0] = C1;
    TEMP[1] = 0x2E;
    TEMP[2] = C2;
    TEMP[3] = C3;
    return TEMP;
}

void INFOADC2(void){
    POT2 = VOLTAGE2/51;
    POT22 = (VOLTAGE2-(POT2*51))/10;
    POT23 = (VOLTAGE2-(POT2*51))-(POT22*10);
    POT2 = POT2+0x30;
    POT22 = POT22+0x30;
    POT23 = POT23+0x30;
}

const char* STRINGPOT2(char C1, char C2, char C3){
    char TEMP[4];
    TEMP[0] = C1;
    TEMP[1] = 0x2E;
    TEMP[2] = C2;
    TEMP[3] = C3;
    return TEMP;
}

/*
 DELAY DE ADQUISICION
 */

void ADCG(void){//GENERO UN DELAY DE ADQUISICIÓN EL CUAL FUNCIONA DE LA SIGU
IENTE MANERA
    if(ADCGO > 20){ //CUANDO ADCGO SEA MÁS GRANDE QUE 20 YA QUE ESTE VA A ES
TAR SUMANDOSE CONSTANTEMENTE EN LA INTERRUPCIÓN
        ADCGO = 0; //SE SETEA EN 0 NUEVAMENTE
        ADCON0bits.GO_nDONE = 1; //SE HABILITA EL GO DEL ADC PARA QUE LA CON
FIGURACIÓN ADC FUNCIONE CORRECTAMENTE
```

```
    }                           //DE ESTA MANERA PUEDE VOLVER A COMENZAR NU
EVAMENTE SIN PROBLEMAS
}
```

LCD

```
/*
 * File:   LCD.c
 * Author: rodri
 * REFERENCIA: https://simple-circuit.com/pic-microcontroller-mplab-xc8-lcd/
 * https://electrosome.com/lcd-pic-mplab-xc8/
 * Created on 4 de febrero de 2021, 10:48 AM
 */

#include <xc.h>
#include <stdint.h>
#include "LCD.h"

#define LCD_FIRST_ROW           0x80
#define LCD_SECOND_ROW          0xC0
#define LCD_THIRD_ROW           0x94
#define LCD_FOURTH_ROW          0xD4
#define LCD_CLEAR               0x01
#define LCD_RETURN_HOME         0x02
#define LCD_ENTRY_MODE_SET      0x04
#define LCD_CURSOR_OFF          0x0C
#define LCD_UNDERLINE_ON        0x0E
#define LCD_BLINK_CURSOR_ON     0x0F
#define LCD_MOVE_CURSOR_LEFT    0x10
#define LCD_MOVE_CURSOR_RIGHT   0x14
#define LCD_TURN_ON             0x0C
#define LCD_TURN_OFF            0x08
#define LCD_SHIFT_LEFT          0x18
#define LCD_SHIFT_RIGHT         0x1E


char a;

//LCD module connections
#define LCD_RS   RE0
#define LCD_EN   RE1
#define LCD_RW   RE2
#define LCD_D0   RD0
#define LCD_D1   RD1
```

```
#define LCD_D2   RD2
#define LCD_D3   RD3
#define LCD_D4   RD4
#define LCD_D5   RD5
#define LCD_D6   RD6
#define LCD_D7   RD7
#define LCD_RS_DIR  TRISE0
#define LCD_EN_DIR  TRISE1
#define LCD_RW_DIR  TRISE2
#define LCD_D0_DIR  TRISD0
#define LCD_D1_DIR  TRISD1
#define LCD_D2_DIR  TRISD2
#define LCD_D3_DIR  TRISD3
#define LCD_D4_DIR  TRISD4
#define LCD_D5_DIR  TRISD5
#define LCD_D6_DIR  TRISD6
#define LCD_D7_DIR  TRISD7
//End LCD module connections
#define _XTAL_FREQ 8000000


void LCD_init(void);
void LCD_Goto(uint8_t col, uint8_t row);
void LCDPutC(char LCD_Char);
void LCD_Print(char *LCD_Str);
void LCD_Cmd(uint8_t Command);
void LCD_PORT(char a);

void LCD_PORT(char a){
    PORTD = a;
}

void LCD_init(void){
  LCD_Cmd(0x38);
  LCD_Cmd(0x0c);
  LCD_Cmd(0x06);
  LCD_Cmd(0x80);
}

void LCD_Goto(uint8_t col, uint8_t row){
switch(row)
  {
    case 2:
      LCD_Cmd(LCD_SECOND_ROW + col - 1);
      break;
```

```
    case 3:
      LCD_Cmd(LCD_THIRD_ROW  + col - 1);
      break;
    case 4:
      LCD_Cmd(LCD_FOURTH_ROW + col - 1);
    break;
    default:      // case 1:
      LCD_Cmd(LCD_FIRST_ROW  + col - 1);
  }
}

void LCDPutC(char LCD_Char){
  LCD_RS = 1;
  LCD_PORT(LCD_Char);
  LCD_EN = 1;
  __delay_us(40);
  LCD_EN = 0;
  LCD_RS = 0;
}
```

```
/*
 * File:
 * Author: RODRIGO GARCÍA
 * Comments:
 * Revision history:
 */

// This is a guard condition so that contents of this file are not included
// more than once.
#ifndef _LCD_H_
#define _LCD_H_

#include <xc.h> // include processor files - each processor file is guarded.

#include <stdint.h>

void LCD_init(void);
void LCD_Goto(uint8_t col, uint8_t row);
void LCDPutC(char LCD_Char);
void LCD_Print(char *LCD_Str);
void LCD_Cmd(uint8_t Command);
void LCD_PORT(char a);
//void LCD_WRITENIBBLE(uint8_t n);
```

```
#endif  /* _LCD_H_ */
```

USART

```
#include <xc.h>
#include <stdint.h>
#include "USART.h"

void Conf_TXR(void);
void Conf_RXT(void);

void Conf_TXR(void){
    TXSTAbits.SYNC = 0;
    TXSTAbits.TXEN = 1;
    TXSTAbits.BRGH = 1;
    TXSTAbits.TX9 = 0;
    BAUDCTLbits.BRG16 = 0;
    SPBRG = 25;
}

void Conf_RXT(void){
    RCSTAbits.SPEN = 1;
    RCSTAbits.CREN = 1;
    RCSTAbits.FERR = 0;
    RCSTAbits.OERR = 0;
    RCSTAbits.RX9 = 0;
    PIE1bits.RCIE = 1;
}

void TRANSMITIR(char *VAL){
    TXREG = VAL;
    TXREG = 0x2E;
}
```

```
/*
 * File:
 * Author: RODRIGO GARCIA
 * Comments:
 * Revision history:
 */

// This is a guard condition so that contents of this file are not included
```

```
// more than once.
#ifndef _USART_
#define _USART_

#include <xc.h> // include processor files - each processor file is guarded.

#include <stdint.h>

void Conf_TXR(void);
void Conf_RXT(void);
void TRANSMITIR(char *VAL);

#endif  /* XC_HEADER_TEMPLATE_H */
```

ADC

```
/*
 * File:
 * Author: RODRIGO GARCIA
 * Comments:
 * Revision history:
 */

// This is a guard condition so that contents of this file are not included
// more than once.
#ifndef _ADC_H_
#define _ADC_H_

#include <xc.h> // include processor files - each processor file is guarded.

#include<stdint.h>

void initADC(uint8_t frec, uint8_t can);

#endif  /* XC_HEADER_TEMPLATE_H */
```

```
/*
 * File:   Contador.c
 * Author: RODRIGO GARCIA
 *
 * Created on 1 de febrero de 2021, 09:52 PM
 */
```

```c
/*
 INCLUIR LIBRERIAS CREADAS
 */

#include <xc.h>
#include <stdint.h>
#include <pic16f887.h>
#include "ADC.h"


void initADC(uint8_t frec, uint8_t can){
    switch(frec){
        case 0:
         ADCON0bits.ADCS = 0b00;  //FOSC/2
        break;
        case 1:
         ADCON0bits.ADCS = 0b01; //FOSC/8
        break;
        case 2:
         ADCON0bits.ADCS = 0b10; //FOSC/32
        break;
        case 3:
         ADCON0bits.ADCS = 0b11; //FRc (500kHz)
        break;
        default:
         ADCON0bits.ADCS = 0b00;
        break;
    }
    switch(can){
        case 0:
         ADCON0bits.CHS = 0b0000; //CANAL AN0
        break;
        case 1:
         ADCON0bits.CHS = 0b0001; //CANAL AN1
        break;
        case 2:
         ADCON0bits.CHS = 0b0010; //CANAL AN2
        break;
        case 3:
         ADCON0bits.CHS = 0b0011; //CANAL AN3
        break;
        case 4:
         ADCON0bits.CHS = 0b0100; //CANAL AN4
        break;
        case 5:
```

```
         ADCON0bits.CHS = 0b0101; //CANAL AN5
        break;
        case 6:
         ADCON0bits.CHS = 0b0110; //CANAL AN6
        break;
        case 7:
         ADCON0bits.CHS = 0b0111; //CANAL AN7
        break;
        case 8:
         ADCON0bits.CHS = 0b1000; //CANAL AN8
        break;
        case 9:
         ADCON0bits.CHS = 0b1001; //CANAL AN9
        break;
        case 10:
         ADCON0bits.CHS = 0b1010; //CANAL AN10
        break;
        case 11:
         ADCON0bits.CHS = 0b1011; //CANAL AN11
        break;
        case 12:
         ADCON0bits.CHS = 0b1100; //CANAL AN12
        break;
        case 13:
         ADCON0bits.CHS = 0b1101; //CANAL AN13
        break;
        case 14:
         ADCON0bits.CHS = 0b1110; //CVref
        break;
        case 15:
         ADCON0bits.CHS = 0b1111; //Fixed Ref
        break;
        default:
         ADCON0bits.CHS = 0b0000;
        break;
    }
    ADCON0bits.GO = 0; //CONVERSIÓN STATUS BIT EN 0
    ADCON0bits.ADON = 1; //ENABLE BIT DEL ADC EN 1
    ADCON1=1;
}
```

OSCILADOR

```
/*
 * File:
```

```
 * Author: RODRIGO GARCÍA
 * Comments:
 * Revision history:
 */

// This is a guard condition so that contents of this file are not included
// more than once.
#ifndef __OSCILADOR_H_
#define __OSCILADOR_H_

#pragma config FOSC = INTRC_NOCLKOUT// Oscillator Selection bits (INTOSCIO o
scillator: I/O function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLK
IN)

#include <xc.h> // include processor files - each processor file is guarded.

#include<stdint.h>
//*************************************************************
// Prototipo de la función para inicializar Oscilador Interno
// Parametros: Opción de frecuencia a utilizar ver pág. 62 Manual
//*************************************************************
void initOsc(uint8_t frec);

#endif  /* XC_HEADER_TEMPLATE_H */
```

```
/*
 * File:   Osciladorc
 * Author: RODRIGO GARCIA
 * Author Original: Pedro Mazariegos
 * Repositorio de GitHUB de el: https://github.com/pdmazariegos-
uvg/ie3027/tree/master/Ejemplos
 *
 * Created on 1 de febrero de 2021, 10:52 PM
 */

#include <stdint.h>
#include <pic16f887.h>
#include "Osciladorh"
//*************************************************************
// Función para inicializar Oscilador Interno
// Parametros: Opción de frecuencia a utilizar ver pág. 62 Manual
//*************************************************************
void initOsc(uint8_t frec){
    switch(frec){
        case 0:                             // 31 KHz
```

```
            OSCCONbits.IRCF0 = 0;
            OSCCONbits.IRCF1 = 0;
            OSCCONbits.IRCF2 = 0;
            break;
        case 1:                             // 125 KHz
            OSCCONbits.IRCF0 = 1;
            OSCCONbits.IRCF1 = 0;
            OSCCONbits.IRCF2 = 0;
            break;

            /*************************************************
             * Acá se debería de programar para todas las demás
             * frecuencias, colocando un caso por cada una de
             * las opciones que tiene el microcontrolador
             *************************************************/

        case 7:                             // 8 MHz
            OSCCONbits.IRCF0 = 1;
            OSCCONbits.IRCF1 = 1;
            OSCCONbits.IRCF2 = 1;
            break;
        case 8:                            // 4 MHz
            OSCCONbits.IRCF0 = 0;
            OSCCONbits.IRCF1 = 1;
            OSCCONbits.IRCF2 = 1;
            break;
    }

    OSCCONbits.SCS = 1;      // Se utilizará el reloj interno para el sistem
a
}
```