Universidad del Valle de Guatemala

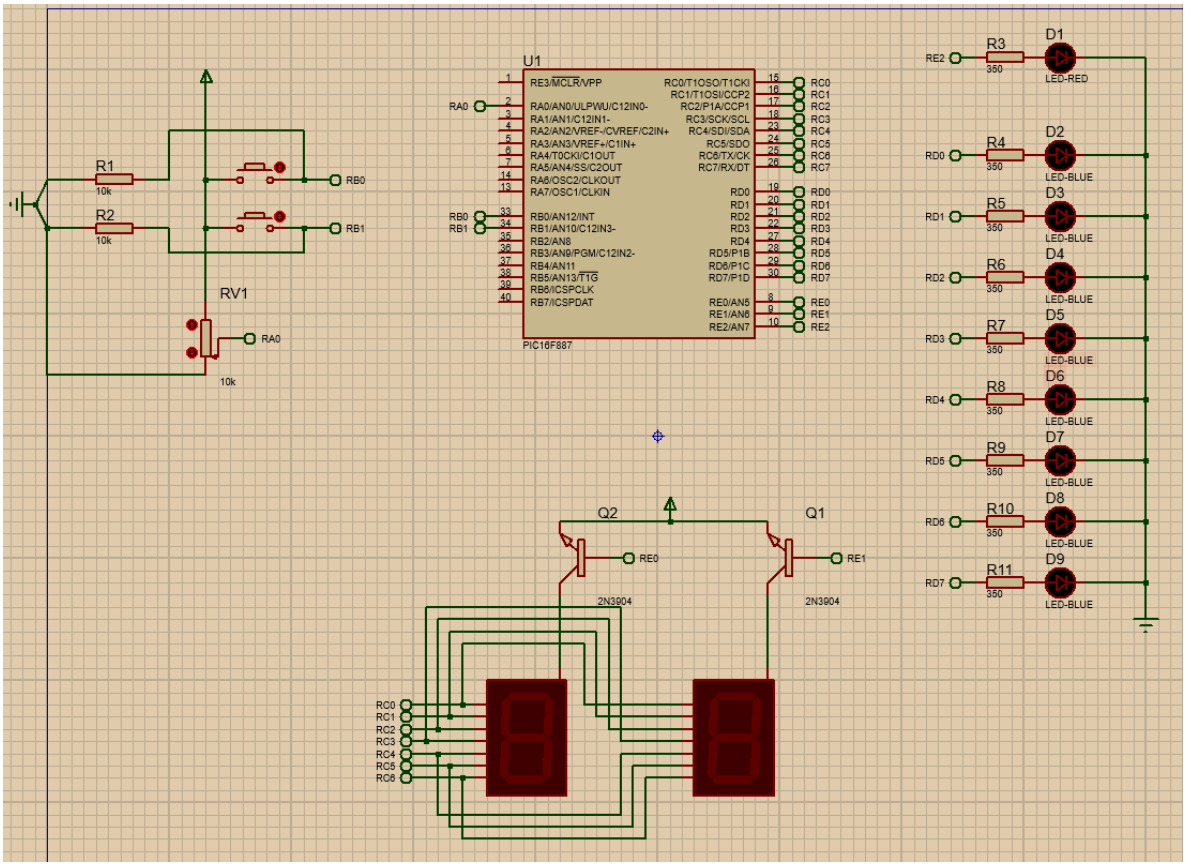
Digital 2 Sección 10

Rodrigo García 19085

Reporte Laboratorio 2

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

Esquemático:



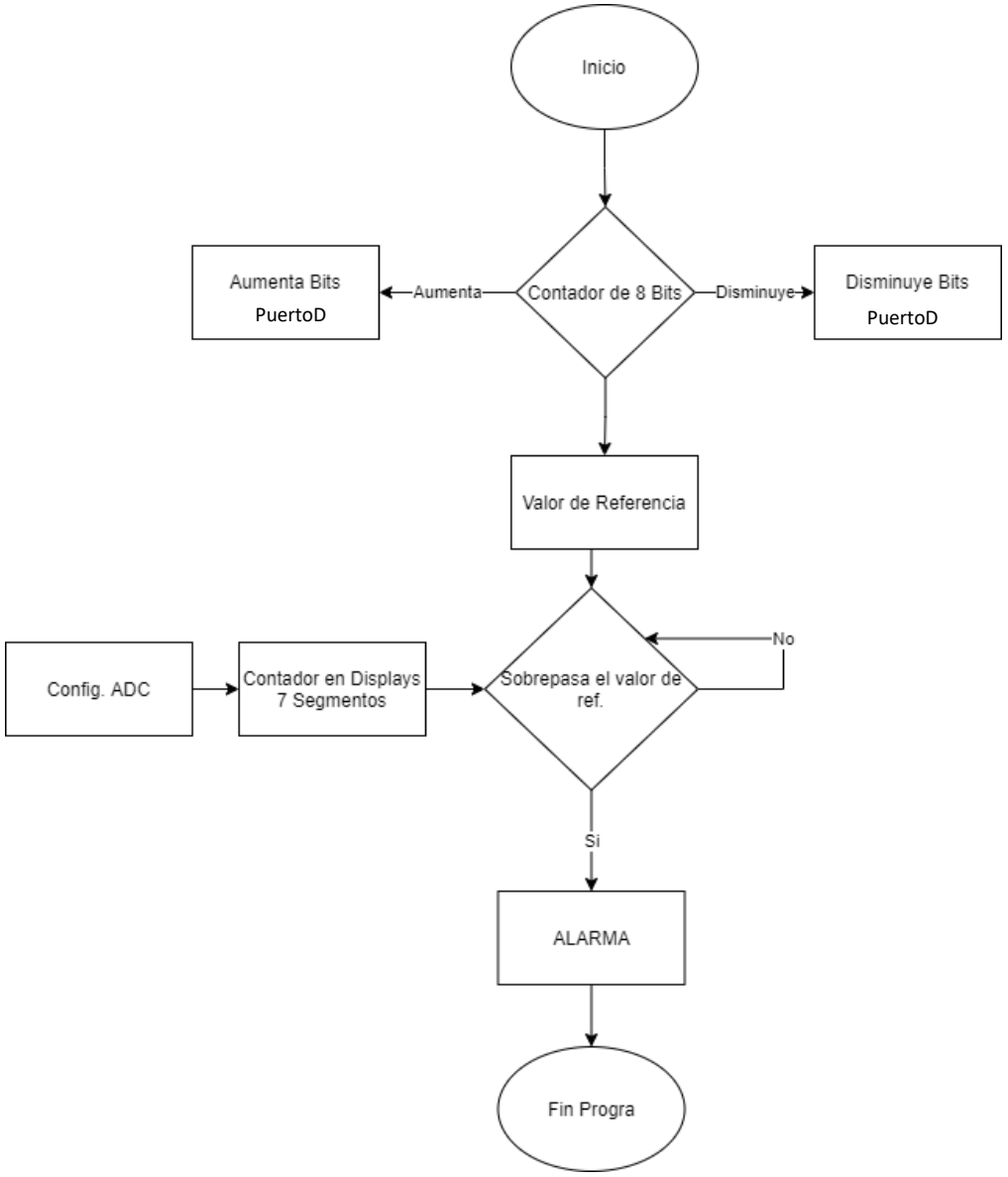
Descripción:

Configuración de Puertos:

Su utilizaron los puertos A, B, C, D, y E. En el puerto C se conectaron los displays de 7 segmentos de anodo común, los cuales se multiplexaron utilizando transistores los cuales se conectan al común de los displays, y a los pinres RE0 y RE1 del puerto E para así realizar el toggle en la interrupción del TMR0. En el puerto D están conectados los leds que funcionan como contador, y la alarma que también es un led esta conectada en el pin RE2. El potenciómetro que se utiliza para cambiar los valores de los displays mediante el ADC esta conectado en el pin RA0 donde se configuro el ADC de manera en que se habilita el canal ANS0 para que funcione correctamente. Luego en el puerto B se utilizaron los pines RB0 y RB1 para los push buttons que sirven para aumentar y disminuir el valor del contador, en ese mismo puerto B se habilitaron las interrupciones necesarias, se

configuraron los pull-ups y se configuraron los pines como entradas para leer correctamente los push buttons.

Seudocódigo:



Código:

PRINCIPAL

```
/*
 * File:   Lab02.c
 * Author: Rodrigo García 19085
 *
 * Created on 28 de enero de 2021, 04:40 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>
//INCLUYO LAS LIBRERIAS CREADAS POR EL USUARIO(YO RODRIGO GARCIA))
#include "7Segmentos.h"
#include "Oscilador.h"
#include "ADC.h"
```

```
/*
 VARIABLES
 */

#define SUM RB0
#define RES RB1
#define ALARMA RE2
#define POT RA0

char FLAG1;
char FLAG2;
uint8_t valorADC = 0;
uint8_t ADCGO = 0;
uint8_t HIGH = 0;
uint8_t LOW = 0;
char FLG;


/*
 FUNCIONES
 */
void Setup(void);
void ADCG(void);


void __interrupt() isr(void){
    if (INTCONbits.TMR0IF == 1){   //CONFIGURACIÓN PARA UTILIZAR LA NTERRUPCIÓN DE TMR0
        TMR0=236;
        INTCONbits.TMR0IF = 0;
        ADCGO++; //SE VA SUMANDO CADA VEZ ESTA VARIABLE YA QUE ES UN DELAY DE ADQUISICIÓN EXTRA
        RE0 = 0; //LIMPIOO LOS PINES EN DONDE SE ENCUENTRAN LOS TRANSISTORES PARA ASÍ
        RE1 = 0; //INICIAR LA MULTIPLEXACIÓN DE LOS DISPLAYS
        if(FLG == 0){ //SI MI BANDERA SE ENCUENTRA EN 0
            FLG = 1;  //ENCIENDO LA BANDERA
            RE0 = 1; //ENCIENDO EL TRANSISTOR CORRESPONDIENTE AL PRIMER DISPLAY
            RE1 = 0; //
            HIGH=valorADC; //LUEGO SEPARO EL VALOR EN DONDE SE GUARDA LA INFO DEL ADC(valorADC)
            HIGH=HIGH & 0b11110000; //SEPARO LA INFORMACIÓN EN NIBBLES HIGH Y LOW REALIZO UN AND
```

```
            HIGH=HIGH>>4;     //ENTONCES MUEVO PARA LA DERECHA LOS BITS MÁS
SIGNIFICATIVOS
            LOW=valorADC & 0b00001111; //HAGO LO MISMO PERO CON EL NIBBLE LO
W
            Display7(HIGH);    //LLAMO A LA FUNCION DE MI LIBRERÍA QUE CONTI
ENE LA TABLA PARA LOS VALORES DE MI DISPLAY DE 7 SEGMENTOS
        }
        if(FLG == 1){  //REALIZO LO MISMO SOLO QUE PARA EL SEGUNDO DISPLAY U
TILIZANDO EL NIBBLE LOW
            FLG = 0;
            RE0 = 0;
            RE1 = 1;
            LOW=valorADC;
            LOW=LOW &0b00001111;
            LOW=LOW<<4;
            HIGH=valorADC & 0b11110000;
            Display7(LOW);
        }
    }

    if(INTCONbits.RBIF == 1){ //CONFIGURACIÓN DE INTERRUPCIÓN EN EL PUERTOB
                if(SUM == 1){ //SI MI PUSH BUTTON DE SUMA ESTA EN 1
                    FLAG1=1; //SE ACTIVA MI BANDERA
                }
                if(FLAG1 == 1 && SUM == 0){ //SI MI FLAG1 = 1 Y MI PUSH BUTT
ON EN 0
                    FLAG1=0; //SE APAGA EL FLAG1
                    PORTD++; //SE LE SUMA 1 AL PUERTO D EN DONDE SE ENCENTRA
 EL CONTADOR
                }
                if(RES==1){ //MISMO PROCEDIMIENTO SOLO QUE ESTE RESTA VALORE
S AL CONTADOR
                    FLAG2=1;
                }
                if(FLAG2==1 && RES==0){
                    FLAG2=0;
                    PORTD--;
                }
        INTCONbits.RBIF = 0;

    }
    if(PIR1bits.ADIF==1){ //CONFIGURACIÓN PARA LAS INTERRUPCIONES DEL ADC
        PIR1bits.ADIF = 0;
        valorADC = ADRESH; //LE INDICO A MI VARIABLE valorADC EL VALRO DE AD
RESH QUE CORRESPONDE
```

```
                            //AL VALOR DEL ADC
    }
}

/*
 MAIN LOOP
 */

void main(void) {
    initOsc(8);//LLAMO LA FUNCIÓN DE MI LIBRERIA DEL OSCILADOR
    Setup();   //LLAMO A MI FUNCIÓN SETUP
    initADC(1,0); //LLAMO A LA FUNCÓN DE MI LIBRERIA DE ADC PAR ASÍ CONFIGUR
ARLO
        while(1){
            ADCG(); //LLAMO A MI RUTINA QUE SE ENCARGA DEL DELAY DE ADQUISIC
IÓN
            if(valorADC>PORTD){ //COMPARO LOS VALORES DE MI PUERTO Y DE LA V
ARIABLE QUE CONTIENE EL VALOR DE ADRESH
                ALARMA=1; //SE ENCIENDE LA ALARMA SI LA VARIABLE TIENE UN VA
LOR MÁS GRANDE QUE EL CONTADOR
            }
            if(valorADC<PORTD){ //SI ES MÁS PEQUEÑO SE APAGA LA ALARMA
                ALARMA=0;
            }
        }
    }

void Setup(void){
    PORTA = 0;//LIMPIEZA DE PUERTOS
    PORTB = 0;
    PORTC = 0;
    PORTD = 0;
    PORTE = 0;

    ANSEL = 0b00000001;//INDICO EL PRIMER PIN COMO ANALOGO
    ANSELH = 0;

    TRISA = 0b00000001;
    TRISB = 0b00000011;
    TRISC = 0;
    TRISD = 0;
    TRISE = 0;
    OPTION_REG = 0b00000011;
    INTCONbits.GIE = 1;//HABILITO LAS INTERRUPCIONES NECESARIAS, LA GLOBAL P
RINCIPALMENTE
```

```
    INTCONbits.T0IE = 1; //HABILITO LAS INTERRUPCIONES DEL TMR0
    INTCONbits.T0IF = 0;
    INTCONbits.PEIE = 1; //HABILITA LOS PERIPHERAL INTERRUPTS
    PIE1bits.ADIE = 1; //HABILILTO LAS INTERRUPCIONES DEL ADC
    PIR1bits.ADIF = 0;

    IOCBbits.IOCB0 = 1;//HABILITO LOS INTERRUPTS ON CHANGE PARA LOS PUSH
    IOCBbits.IOCB1 = 1;
    INTCONbits.RBIE = 1; //HABILILTO LAS INTERRUPCIONES DEL PUERTO B
    INTCONbits.RBIF = 0;
}


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
}
```

LIBRERIAS UTILIZADAS

7 SEGMENTOS

```
/*
 * 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 _7SEGMENTOS_H_
#define _7SEGMENTOS_H_

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

```
void Display7(uint8_t Table);

#endif  /* XC_HEADER_TEMPLATE_H */
```

```
/*
 * File:   7segmento.c
 * Author: RODRIGO GARCIA
 *
 * Created on 1 de febrero de 2021, 09:15 PM
 */

#include "7Segmentos.h"

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


void Display7(uint8_t Table){
    switch(Table){
        case 0:
            PORTC=0; //Numero 0
            RC0=0;
            RC1=0;
            RC2=0;
            RC3=0;
            RC4=0;
            RC5=0;
            RC6=1;
            RC7=0;
            break;
        case 1:
            PORTC=0; //Numero 1
            RC0=1;
            RC1=0;
            RC2=0;
            RC3=1;
            RC4=1;
            RC5=1;
            RC6=1;
            RC7=0;
            break;
        case 2:
            PORTC=0; //Numero 2
            RC0=0;
            RC1=0;
```

```
            RC2=1;
            RC3=0;
            RC4=0;
            RC5=1;
            RC6=0;
            RC7=0;
            break;
        case 3:
            PORTC=0; //Numero 3
            RC0=0;
            RC1=0;
            RC2=0;
            RC3=0;
            RC4=1;
            RC5=1;
            RC6=0;
            RC7=0;
            break;
        case 4:
            PORTC=0;//Numero 4
            RC0=1;
            RC1=0;
            RC2=0;
            RC3=1;
            RC4=1;
            RC5=0;
            RC6=0;
            RC7=0;
            break;
        case 5:
            PORTC=0; //Numero 5
            RC0=0;
            RC1=1;
            RC2=0;
            RC3=0;
            RC4=1;
            RC5=0;
            RC6=0;
            RC7=0;
            break;
        case 6:
            PORTC=0; //Numero 6
            RC0=0;
            RC1=1;
            RC2=0;
```

```
            RC3=0;
            RC4=0;
            RC5=0;
            RC6=0;
            RC7=0;
            break;
        case 7:
            PORTC=0; //Numero 7
            RC0=0;
            RC1=0;
            RC2=0;
            RC3=1;
            RC4=1;
            RC5=1;
            RC6=1;
            RC7=0;
            break;
        case 8:
            PORTC=0; //Numero 8
            RC0=0;
            RC1=0;
            RC2=0;
            RC3=0;
            RC4=0;
            RC5=0;
            RC6=0;
            RC7=0;
            break;
        case 9:
            PORTC=0; //Numero 9
            RC0=0;
            RC1=0;
            RC2=0;
            RC3=1;
            RC4=1;
            RC5=0;
            RC6=0;
            RC7=0;
            break;
        case 10:
            PORTC=0; //LETRA A
            RC0=0;
            RC1=0;
            RC2=0;
            RC3=1;
```

```
            RC4=0;
            RC5=0;
            RC6=0;
            RC7=0;
            break;
        case 11:
            PORTC=0; //LETRA b
            RC0=1;
            RC1=1;
            RC2=0;
            RC3=0;
            RC4=0;
            RC5=0;
            RC6=0;
            RC7=0;
            break;
        case 12:
            PORTC=0; //LETRA C
            RC0=0;
            RC1=1;
            RC2=1;
            RC3=0;
            RC4=0;
            RC5=0;
            RC6=1;
            RC7=0;
            break;
        case 13:
            PORTC=0; //LETRA d
            RC0=1;
            RC1=0;
            RC2=0;
            RC3=0;
            RC4=0;
            RC5=1;
            RC6=0;
            RC7=0;
            break;
        case 14:
            PORTC=0; //LETRA E
            RC0=0;
            RC1=1;
            RC2=1;
            RC3=0;
            RC4=0;
```

```
            RC5=0;
            RC6=0;
            RC7=0;
            break;
        case 15:
            PORTC=0; //LETRA F
            RC0=0;
            RC1=1;
            RC2=1;
            RC3=1;
            RC4=0;
            RC5=0;
            RC6=0;
            RC7=0;
            break;
    }
}
```

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
 */


/*
 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:   Oscilador.c
 * 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 "Oscilador.h"
//*****************************************************************
// 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
}
```