Universidad del Valle de Guatemala

Electrónica Digital II

Rodrigo Díaz, 18265

Lab #2

# Pseudocódigo:

Parte 1

Parte 2

Parte 3

Parte 4

# Código Comentado

## Main

/\*

\* File: main.c

\* Author: Rodrigo Díaz

\* Digital 2

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Importación de librerías

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// se incluyen las librerias que se utilizan y las creadas para el lab

#include <xc.h>

#include <stdint.h>

#include <pic16f887.h>

#include "ADC.h"

#include "TMR0.h"

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Palabra de configuración

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// CONFIG1

#pragma config FOSC = XT // Oscillator Selection bits (XT oscillator: Crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN)

#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled and 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/MCLR 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 code protection is disabled)

#pragma config BOREN = OFF // Brown Out Reset Selection bits (BOR disabled)

#pragma config IESO = OFF // Internal External Switchover bit (Internal/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 pin 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 bits (Write protection off)

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Variables

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// la variable XTAL FREQ es necesaria para que funcionen los delays

#define \_XTAL\_FREQ 8000000

// se crean las variables a utilizar con su tamano

uint8\_t cambio = 0;

int8\_t swap = 0;

uint8\_t valor\_adc = 0;

int8\_t adc\_low = 0;

int8\_t adc\_high = 0;

int8\_t display\_1 = 0;

int8\_t display\_2 = 0;

// se crea un array con todos los valores para encender los pines del 7seg

int8\_t segmentos[16]={0b00111111,0b00000110,0b01011011,0b01001111,0b01100110,0b01101101,0b01111101,0b00000111,0b01111111,0b01101111,0b01110111,0b01111100,0b00111001,0b01011110,0b01111001,0b01110001};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Interrupción

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void \_\_interrupt() ISR(void) {

// primero se revisa la bandera de la interrupcion on change del PORTB

if (INTCONbits.RBIF == 1) {

// se realiza un antirebote

if (PORTBbits.RB0 == 0) {

\_\_delay\_ms(50);

if (PORTBbits.RB0 == 1) {

// se incrementa el PORTC con un boton y se apaga la bandera

PORTC ++;

INTCONbits.RBIF = 0;

}

}

// se realiza antirebote para el boton de decremento

if (PORTBbits.RB1 == 0) {

\_\_delay\_ms(50);

if (PORTBbits.RB1 == 1) {

// se decrementa el PORTC y se apaga la bandera

PORTC --;

INTCONbits.RBIF = 0;

}

}

}

// luego se revisa la bandera el interrupt del ADC

if (PIR1bits.ADIF == 1) {

// se pasa el valor del ADRESH a la variable

valor\_adc = ADRESH;

// se separan los nibbles, basandose en el codigo de

//https://www.geeksforgeeks.org/swap-two-nibbles-byte/#:~:text=To%20swap%20the%20nibbles%2C%20we,in%20a%20typical%20C%20compiler.

adc\_low = valor\_adc & 0b00001111;

swap = ((valor\_adc & 0b00001111)<<4 | (valor\_adc & 0b11110000)>>4);

adc\_high = swap & 0b00001111;

// se apaga la bandera

PIR1bits.ADIF = 0;

}

// por ultimo, se revisa la bandera de interrupcion del TMR0

if (INTCONbits.T0IF == 1) {

// se realiza un cambio en los bits de los transistores y se

// despliega el valor respectivo en el PORTD

if (PORTEbits.RE0 == 1){

PORTEbits.RE0 = 0;

PORTEbits.RE1 = 1;

PORTD = display\_2;

} else {

PORTEbits.RE0 = 1;

PORTEbits.RE1 = 0;

PORTD = display\_1;

}

// se vuelve a asignar el valor del TMR0 y se apaga la bandera

TMR0 = 176;

INTCONbits.T0IF = 0;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Prototipos de funciones

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void setup(void);

void adc(void);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Ciclo principal

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void main(void) {

// mando a llamar a la funcion de setup, y a las initadc y initmr0 de las

//librerias creadas

setup();

initADC();

initTMR0();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Loop principal

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// como es un while(1) siempre se va a repetir este loop.

while (1) {

// se manda a llamar la funcion del adc

adc();

// se asigna un valor del array para los valores del disp 7seg

display\_1 = segmentos[adc\_high];

display\_2 = segmentos[adc\_low];

// se tiene una alarma visual cuando el valor analogico sobrepasa al

// valor del contador

if (valor\_adc > PORTC) {

PORTEbits.RE2 = 1;

} else {

PORTEbits.RE2 = 0;

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Configuración

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void setup(void) {

// Todos los bits utilizados se configuran como salidas, menos los primeros

// 2 bits del puerto B y el primero del A, debido a que allí estan los push/POT. Ansel y Anselh

// se ponen en 1 solamente donde hayan entradas digitales.

TRISE = 0;

PORTE = 0;

ANSEL = 0b00000001;

ANSELH = 0;

TRISB = 0b00000011;

PORTB = 0;

TRISC = 0;

PORTC = 0;

TRISD = 0;

PORTD = 0;

PORTA = 0;

TRISA = 0b00000001;

// se configuran las interrupciones on change del puerto B

INTCONbits.GIE = 1;

INTCONbits.RBIE = 1;

INTCONbits.RBIF = 0;

IOCB = 0b00000011;

//INTCONbits.PEIE = 1;

//PIE1bits.ADIE = 1;

//PIR1bits.ADIF = 0;

//ADCON0 = 0b01000001;

//ADCON1 = 0b00000000;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Funciones

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// esta funcion incluye el delay para antes de la conversion del ADC y encender

// el bit de GO

void adc(void) {

\_\_delay\_us(8);

ADCON0bits.GO\_DONE = 1;

}

## Librería ADC

# include "ADC.h"

void initADC(void){

//INTCONbits.GIE = 1;

// se realiza el setup para utilizar interrupciones y apagar la bandera

// se tiene un corrimiento hacia la izquierda para obtener los valores

// mas significativos.

INTCONbits.PEIE = 1;

PIE1bits.ADIE = 1;

PIR1bits.ADIF = 0;

ADCON0 = 0b01000001;

ADCON1 = 0b00000000;

}

## Librería TMR0

# include "TMR0.h"

void initTMR0(void){ //Interrupciones cada 2,5 ms

// se configuran los bits para tener interrupciones en el TMR0

INTCONbits.GIE = 1;

INTCONbits.PEIE = 1;

INTCONbits.T0IE = 1;

INTCONbits.T0IF = 0;

// se configura un prescaler de 64

OPTION\_REG = 0b10000101;

// con la calculadora proporcionada, se calcula el valor del TMR0

TMR0 = 176;

}

# LINK repositorio GitHub

<https://github.com/RodDia2/Labs_Digital_2>