Universidad del Valle de Guatemala

Digital 2

Kurt Kellner

**PRE-LABORATORIO No. 3**

**LCD:**

/\*

\* Project: LCD

\* File: LCD.c

\* Author: Pablo Rene Arellano Estrada

\* Carnet: 151379

\* Created: February 8, 2021,

\*/

//============================================================================\*/

// LIBRERIAS

//============================================================================\*/

#define RS PORTCbits.RC0

#define RW PORTCbits.RC1

#define E PORTCbits.RC2

#define D0 PORTDbits.RD0

#define D1 PORTDbits.RD1

#define D2 PORTDbits.RD2

#define D3 PORTDbits.RD3

#define D4 PORTDbits.RD4

#define D5 PORTDbits.RD5

#define D6 PORTDbits.RD6

#define D7 PORTDbits.RD7

#include <xc.h> //

#include <stdint.h> // Variables de ancho definido

#include <stdio.h> // Variables

#include "ADC\_lib.h" // Libreria Personalizada ADC

#include "LCD\_8bits.h"

//============================================================================\*/

// PALABRA DE CONFIGURACION

//============================================================================\*/

// CONFIG1

#pragma config FOSC = INTRC\_NOCLKOUT// Oscilador interno

#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)

// DEFINE

#define \_XTAL\_FREQ 8000000

//============================================================================\*/

// VARIABLES

//============================================================================\*/

uint16\_t i = 0; // Variables Configuracion ADC

uint8\_t adc\_value = 0;

//============================================================================\*/

// PROTOTIPO DE FUNCIONES

//============================================================================\*/

void setup(void);

void osc\_config (void);

void interrup\_config (void);

void tmr0\_config(void);

void adc\_config (void);

void USART\_config(void);

//============================================================================\*/

// INTERRUPCIONES

//============================================================================\*/

//void \_\_interrupt() ISR(void)

//{

// La interrupcion global GIE inicia automaticamente con GIE = 0

//if (INTCONbits.TMR0IF == 1) // Si hay desboradmiento de TIMER0 la bandera se levanta y se revisa

//{

// INTCONbits.TMR0IF = 0; // Se apaga la bandera manualmente

// TMR0 = 10;

//} // La interrupcion global GIE finaliza automaticamente con GIE = 1 para la siguiente

//}

//============================================================================\*/

// CICLO PRINCIPAL

//============================================================================\*/

void main(void)

{

setup(); // Funciones de Configuracion

osc\_config();

interrup\_config();

tmr0\_config();

adc\_config ();

unsigned int a;

TRISD = 0b00000000;

Lcd\_Init();

while (1) // LOOP PRINCIPAL \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{

Lcd\_Clear();

Lcd\_Set\_Cursor(1,1);

Lcd\_Write\_String("LCD Library for");

Lcd\_Set\_Cursor(2,1);

Lcd\_Write\_String("MPLAB XC8");

\_\_delay\_ms(2000);

Lcd\_Clear();

Lcd\_Set\_Cursor(1,1);

Lcd\_Write\_String("Developed By");

Lcd\_Set\_Cursor(2,1);

Lcd\_Write\_String("electroSome");

\_\_delay\_ms(2000);

Lcd\_Clear();

Lcd\_Set\_Cursor(1,1);

Lcd\_Write\_String("www.electroSome.com");

for(a=0;a<15;a++)

{

\_\_delay\_ms(300);

Lcd\_Shift\_Left();

}

for(a=0;a<15;a++)

{

\_\_delay\_ms(300);

Lcd\_Shift\_Right();

}

Lcd\_Clear();

Lcd\_Set\_Cursor(2,1);

Lcd\_Write\_Char('H');

Lcd\_Write\_Char('o');

Lcd\_Write\_Char('l');

Lcd\_Write\_Char('a');

//Lcd\_Set\_Cursor(1,1);

//Lcd\_Write\_String('Hola Mundo');

\_\_delay\_ms(2000);

}

}

//============================================================================\*/

// CONFIGURACION

//============================================================================\*/

void setup(void)

{

ANSEL = 1; // Puerto A analogico

TRISA = 1; // Puerto A como entrada analogica

PORTA = 0; // Puerto A entrada apagado

ANSELH = 0; // Puerto B digital

TRISB = 0; //

PORTB = 0; // Puerto B RB0 y RB1 entrada igual a 0

TRISC = 0; // Puerto C salida leds

PORTC = 0; // Puerto C salida leds apagados

TRISD = 0; // Puerto D salida display

PORTD = 0; // Puerto D salida apagados

TRISE = 0; // Puerto E salida transistores y alarma

PORTE = 0; // Puerto E salida apagado

}

void interrup\_config (void)

{

INTCONbits.GIE = 1; // Interrupciones globales habilitadas

INTCONbits.PEIE = 0; // Interrupciones periferias deshabilidatas

INTCONbits.T0IE = 1; // Interrupcion del Timer0 habilitada

INTCONbits.INTE = 0; // Interrupcion externa INT deshabilitada

INTCONbits.RBIE = 1; // Interrupcion del Puerto B habilitadas

INTCONbits.T0IF = 0; // Bandera de Interrupcion del Timer 0

INTCONbits.INTF = 0; // Bandera de interrupcion del INT

INTCONbits.RBIF = 0; // Bandera de interrrupcion del Puerto B

IOCB = 0b00000011; // Interrup on Change enable

}

void osc\_config (void)

{

OSCCONbits.IRCF2 = 1; // Oscilador en 4Mhz

OSCCONbits.IRCF1 = 1;

OSCCONbits.IRCF2 = 0;

OSCCONbits.OSTS = 0; // Oscilador interno

OSCCONbits.HTS = 0;

OSCCONbits.LTS = 1;

OSCCONbits.SCS = 0; // Oscilador basado en el reloj

}

void tmr0\_config (void)

{

OPTION\_REGbits.nRBPU = 1; // PORTB pull-ups habilitados

OPTION\_REGbits.T0CS = 0; // TIMER0 como temporizador, no contador

OPTION\_REGbits.PSA = 0; // Modulo de TIMER con prescaler, no se usa WDT

OPTION\_REGbits.PS2 = 0; // Prescaler en 8

OPTION\_REGbits.PS1 = 1;

OPTION\_REGbits.PS0 = 0;

TMR0 = 10; // Valor del TIMER0 para un delay de 0.246 seg.

}

//============================================================================\*/

// FUNCIONES CON LIBRERIA

//============================================================================\*/

void adc\_config (void)

{

initADC (0); // Configuracion de ADC en libreria

}

void USART\_config(void) // Valor del pic a compu de dos potensiometros

{

USART\_lib\_config();

}

//============================================================================\*/

// FUNCIONES

//============================================================================\*/

// void USART () // Valor del pic a compu de dos potensiometros

//{

// ADCONL = 0;

// ADCON0 = 0;

// ADCON0bits.GO\_DONE = 1; // Se inicia el GO\_DONE para iniciar conversion

// \_\_delay\_ms(10); // Se da tiempo para el Acquisition Time Example

// if (ADCON0bits.GO\_DONE == 0) // Si ya termino la conversion

// {

// ADCON0bits.GO\_DONE = 1; // Se inicia el GO\_DONE para iniciar nuevamente

// adc\_value\_1 = ADRESL; // Se Coloca el valor del registro de la conversion en una variable

// PIR1bits.ADIF = 0;

// if (PIR1bits.TXIF = 1)

// {

// TXREG = adc\_value\_1;

// }

// }

// ADCONL = 0;

// ADCON0 = 0;

// ADCON0bits.GO\_DONE = 1; // Se inicia el GO\_DONE para iniciar conversion

// \_\_delay\_ms(10); // Se da tiempo para el Acquisition Time Example

// if (ADCON0bits.GO\_DONE == 0) // Si ya termino la conversion

// {

// ADCON0bits.GO\_DONE = 1; // Se inicia el GO\_DONE para iniciar nuevamente

// adc\_value\_2 = ADRESL; // Se Coloca el valor del registro de la conversion en una variable

// PIR1bits.ADIF = 0;

// if (PIR1bits.TXIF = 1)

// {

// TXREG = adc\_value\_2;

// }

// }

//}

// void Conversion\_voltaje () // Conversion de Binario a Voltaje

//{

// int voltaje;

// voltaje = ADRESL \* 5 / 256;

//}

// void contador () // Valor de Compu a LCD

//{

// if (valor "+")

// {

// contador = contador + 1;

// }

// if (valor "-")

// {

// contador = contador - 1;

// }

//}

**ADC\_LIB.h:**

/\*

\* Project: Interrupciones y Librerias

\* File: ADC\_lib.h

\* Author: Pablo Rene Arellano Estrada

\* Carnet: 151379

\* Created: February 9, 2021,

\* Libreria https://electrosome.com/lcd-pic-mplab-xc8/

\* Autor: Ligo George

\* Autor 2: Pablo Mazariegos (Canvas)

\*/

#ifndef LCD\_8bits\_H

#define LCD\_8bits\_H

#ifndef \_XTAL\_FREQ

#define \_XTAL\_FREQ 8000000

#endif

#ifndef RS

#define RS PORTCbits.RC0

#endif

#ifndef RW

#define RW PORTCbits.RC1

#endif

#ifndef E

#define E PORTCbits.RC2

#endif

#ifndef D0

#define D0 PORTDbits.RD0

#endif

#ifndef D1

#define D1 PORTDbits.RD1

#endif

#ifndef D2

#define D2 PORTDbits.RD2

#endif

#ifndef D3

#define D3 PORTDbits.RD3

#endif

#ifndef D4

#define D4 PORTDbits.RD4

#endif

#ifndef D5

#define D5 PORTDbits.RD5

#endif

#ifndef D6

#define D6 PORTDbits.RD6

#endif

#ifndef D7

#define D7 PORTDbits.RD7

#endif

//------------------------------------------------------------------------------

// Funciones de Conversion ADC

//------------------------------------------------------------------------------

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

#include <stdint.h> // Variables de ancho definido

void Lcd\_Port (char a);

void Lcd\_Cmd (char a);

void Lcd\_Init(); // Prototipo de funcion

void Lcd\_Clear();

void Lcd\_Set\_Cursor(char a, char b);

void Lcd\_Write\_String(char \*a);

void Lcd\_Shift\_Left();

void Lcd\_Shift\_Right();

void Lcd\_Write\_Char(char a);

void Lcd\_Write\_Char\_4(char a);

#endif /\* LCD\_H \*/

**ADC\_LIB.c:**

/\*

\* Project: Interrupciones y Librerias

\* File: ADC\_lib.h

\* Author: Pablo Rene Arellano Estrada

\* Carnet: 151379

\* Created: February 9, 2021,

\* Libreria https://electrosome.com/lcd-pic-mplab-xc8/

\* Autor: Ligo George

\* Autor 2: Pablo Mazariegos (Canvas)

\*/

#include "LCD\_8bits.h"

void Lcd\_Port (char a)

{

PORTD = a;

}

void Lcd\_Cmd (char a)

{

RS = 0; // Comando hacia LCD

Lcd\_Port(a);

E = 1;

\_\_delay\_ms(4);

E = 0;

}

void Lcd\_Init()

{

Lcd\_Port(0b0000000); // puerto

\_\_delay\_ms(20);

Lcd\_Cmd(0b00110000);

RS = 0;

RW = 0;

\_\_delay\_ms(5);

Lcd\_Cmd(0b00110000);

RS = 0;

RW = 0;

\_\_delay\_ms(11);

Lcd\_Cmd(0b00110000);

RS = 0;

RW = 0;

Lcd\_Cmd(0b00111000);

Lcd\_Cmd(0b00001000);

Lcd\_Cmd(0b00000001);

Lcd\_Cmd(0b00000110);

}

void Lcd\_Clear()

{

Lcd\_Cmd(0);

Lcd\_Cmd(1);

}

void Lcd\_Set\_Cursor(char a, char b)

{

char temp, z, y;

if (a == 1)

{

temp = 0b00010000 + b - 1;

z = temp >> 4;

y = temp & 0x0F;

Lcd\_Cmd(z);

Lcd\_Cmd(y);

}

else if (a == 2)

{

temp = 0b11000000 + b - 1;

z = temp >> 4;

y = temp & 0x0F;

Lcd\_Cmd(z);

Lcd\_Cmd(y);

}

}

void Lcd\_Write\_String(char \*a)

{

int i;

for (i = 0; a[i] != '\0'; i++)

Lcd\_Write\_Char(a[i]);

}

void Lcd\_Shift\_Left()

{

Lcd\_Cmd(0x01);

Lcd\_Cmd(0x0C);

}

void Lcd\_Shift\_Right()

{

Lcd\_Cmd(0x01);

Lcd\_Cmd(0x08);

}

void Lcd\_Write\_Char(char a)

{

RS = 1;

Lcd\_Port(a);

E = 1;

\_\_delay\_us(40);

E = 0;

}

**USART.h:**

/\*

\* Project: LCD

\* File: USART.h

\* Author: Pablo Rene Arellano Estrada

\* Carnet: 151379

\* Created: February 9, 2021,

\*/

#ifndef USART\_H

#define USART\_H

//------------------------------------------------------------------------------

// Funciones de Conversion ADC

//------------------------------------------------------------------------------

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

#include <stdint.h> // Variables de ancho definido

void USART\_lib\_config(); // Prototipo de funcion

#endif /\* ADC\_lib\_H \*/

**USART.c:**

/\*

\* Project: Interrupciones y Librerias

\* File: ADC\_lib.h

\* Author: Pablo Rene Arellano Estrada

\* Carnet: 151379

\* Created: February 9, 2021,

\* Libreria https://electrosome.com/lcd-pic-mplab-xc8/

\* Autor: Ligo George

\* Autor 2: Pablo Mazariegos (Canvas)

\*/

#include "USART.h"

void USART\_lib\_config() // Valor del pic a compu de dos potensiometros

{

TXSTAbits.TX9 = 0;

TXSTAbits.SYNC = 0;

TXSTAbits.BRGH = 0;

TXSTAbits.TXEN = 1;

SPBRG = .12;

SPBRGH = 0;

RCSTAbits.RX9 = 0;

RCSTAbits.CREN = 1;

RCSTAbits.SPEN = 1;

PIR1bits.RCIF = 0;

PIR1bits.TXIF = 0;

}

**ADC\_lib.h:**

/\*

\* Project: Interrupciones y Librerias

\* File: ADC\_lib.h

\* Author: Pablo Rene Arellano Estrada

\* Carnet: 151379

\* Created: February 9, 2021,

\*/

#ifndef ADC\_lib\_H

#define ADC\_lib\_H

//------------------------------------------------------------------------------

// Funciones de Conversion ADC

//------------------------------------------------------------------------------

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

#include <stdint.h> // Variables de ancho definido

void initADC (uint8\_t CHS); // Prototipo de funcion

#endif /\* ADC\_lib\_H \*/

**ADC\_lib.c:**

/\*

\* Project: Interrupciones y Librerias

\* File: ADC\_lib.c

\* Author: Pablo Rene Arellano Estrada

\* Carnet: 151379

\* Created: February 9, 2021,

\*/

#include "ADC\_lib.h" // Se incluye header

void initADC (uint8\_t CHS)

{

switch (CHS) // Menu para elegir canal

{

case 0: //AN0

ADCON0bits.CHS3 = 0;

ADCON0bits.CHS2 = 0;

ADCON0bits.CHS1 = 0;

ADCON0bits.CHS0 = 0;

break;

case 1: //AN1

ADCON0bits.CHS3 = 0;

ADCON0bits.CHS2 = 0;

ADCON0bits.CHS1 = 0;

ADCON0bits.CHS0 = 1;

break;

case 2: //AN2

ADCON0bits.CHS3 = 0;

ADCON0bits.CHS2 = 0;

ADCON0bits.CHS1 = 1;

ADCON0bits.CHS0 = 0;

break;

case 3: //AN3

ADCON0bits.CHS3 = 0;

ADCON0bits.CHS2 = 0;

ADCON0bits.CHS1 = 1;

ADCON0bits.CHS0 = 1;

break;

case 4: //AN4

ADCON0bits.CHS3 = 0;

ADCON0bits.CHS2 = 1;

ADCON0bits.CHS1 = 0;

ADCON0bits.CHS0 = 0;

break;

case 5: //AN5

ADCON0bits.CHS3 = 0;

ADCON0bits.CHS2 = 1;

ADCON0bits.CHS1 = 0;

ADCON0bits.CHS0 = 1;

break;

case 6: //AN6

ADCON0bits.CHS3 = 0;

ADCON0bits.CHS2 = 1;

ADCON0bits.CHS1 = 1;

ADCON0bits.CHS0 = 0;

break;

case 7: //AN7

ADCON0bits.CHS3 = 0;

ADCON0bits.CHS2 = 1;

ADCON0bits.CHS1 = 1;

ADCON0bits.CHS0 = 1;

break;

case 8: //AN8

ADCON0bits.CHS3 = 1;

ADCON0bits.CHS2 = 0;

ADCON0bits.CHS1 = 0;

ADCON0bits.CHS0 = 0;

break;

case 9: //AN9

ADCON0bits.CHS3 = 1;

ADCON0bits.CHS2 = 0;

ADCON0bits.CHS1 = 0;

ADCON0bits.CHS0 = 1;

break;

case 10: //AN10

ADCON0bits.CHS3 = 1;

ADCON0bits.CHS2 = 0;

ADCON0bits.CHS1 = 1;

ADCON0bits.CHS0 = 0;

break;

case 11: //AN11

ADCON0bits.CHS3 = 1;

ADCON0bits.CHS2 = 0;

ADCON0bits.CHS1 = 1;

ADCON0bits.CHS0 = 1;

break;

case 12: //AN12

ADCON0bits.CHS3 = 1;

ADCON0bits.CHS2 = 1;

ADCON0bits.CHS1 = 0;

ADCON0bits.CHS0 = 0;

break;

case 13: //AN13

ADCON0bits.CHS3 = 1;

ADCON0bits.CHS2 = 1;

ADCON0bits.CHS1 = 0;

ADCON0bits.CHS0 = 1;

break;

default:

ADCON0bits.CHS3 = 1;

ADCON0bits.CHS2 = 1;

ADCON0bits.CHS1 = 1;

ADCON0bits.CHS0 = 0;

break;

}

ADCON0bits.ADCS1 = 1; // Frecuencia de Oscilacion / 32

ADCON0bits.ADCS0 = 0;

ADCON0bits.GO\_DONE= 0; // Conversion apagada al principio

ADCON0bits.ADON = 1; // La conversion esta habilitada

ADCON1bits.ADFM = 1; // Justificado a la derecha

ADCON1bits.VCFG1 = 0; // Voltaje = 5V

ADCON1bits.VCFG0 = 0; // Tierra = 0V

}