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
}
```