Universidad Del Valle De Guatemala 01/02/2021

Digital 2 Helder Ovalle Barrios

Sección: 18349

Labrotorio #3

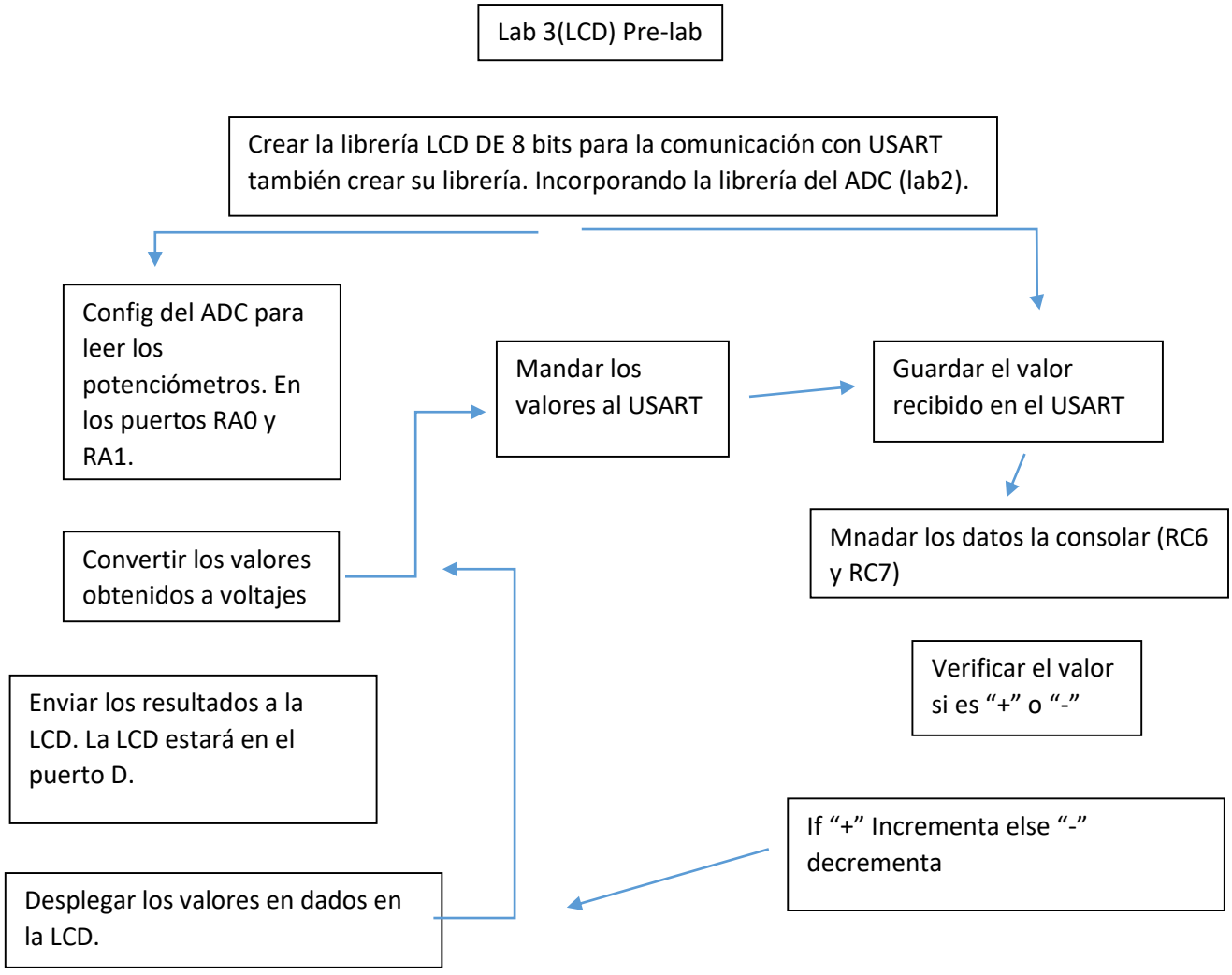
Link de github:

https://github.com/Helder1121/Labsdigitaldos

Link del video:

https://www.youtube.com/watch?v=3-n3GE8pubo

Diagrama de flujo:



Progra comentada:

```
/*
 * File:   lab03.c
 * Author: Helder Ovalle
 *
 * Created on 8 de febrero de 2021, 10:35 AM
 */
//******************************************************************************
// 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 MCLR)
#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 = ON       // 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)

//*****************************************************************************
// Importación de librerías
//*****************************************************************************
#include <xc.h>
#include <stdint.h>
//Permite realizar los prints
#include <stdio.h>
#include <pic16f887.h>
#include "LCD.h"
#include "ADC.h"
#include "USART.h"
#define _XTAL_FREQ 8000000

//*****************************************************************************
// Variables
//*****************************************************************************
char data[16];//Variable mostrara los valos en la lcd
float volt, volt2;//variable para los voltajes en los pots
char LecUSART = 0;
char entregado = 0;
uint8_t contador = 0;
//uint8_t valorADC1;
//uint8_t valorADC2;

//*****************************************************************************
```

```
// Prototipos de funciones
//*****************************************************************************
void config_P();
float ADC_1(void);
float ADC_2(void);
void Enviar_1(void);
void Enviar_2(void);
//float Volts_Bina(uint8_t b);

//Interrupcion del RCIF
void __interrupt() ISR(){
   if (RCIF == 1){
      RCIF = 0;
      LecUSART = Read_USART();
      if(LecUSART=='+'){
         contador++;}
      else if(LecUSART=='-'){
         contador--;}
   }
}
//*****************************************************************************
// Ciclo principal
//*****************************************************************************
void main(void){
   config_P();
   config_ADC();
   _baudios();
   config_txsta();
   config_rcsta();
```

```
Lcd_Init();
LCD_Limpia();
//*****************************************************************************
// Loop principal
//*****************************************************************************
while(1){
   LCD_Limpia();
   Lcd_Set_Cursor(1, 1);
   Lcd_Write_String("S1   S2   CONT ");
   //Lo que se muestra en la LCD en la primera fila
   ADC_1();
   ADC_2();
   sprintf(data, "%1.2f   %1.2f   %d" ,volt,volt2,contador);
   //Despliega en dos decimales el voltaje de 0V-5V
   Lcd_Set_Cursor(2, 1);
   Lcd_Write_String(data);//Mostrara el valor en la LCD
   Write_USART_String("S1    S2    CONT");
   //Mensaje que se muestra en la terminal en la segunda linea
   Write_USART(13);
   Write_USART(10);
   //Saltar lineas
   Write_USART_String(data);
   Write_USART(13);
   Write_USART(10);
   //Saltar lineas
   if (RCIF == 1){
      entregado = RCREG;
      if(entregado == '+'){contador = contador +1;}
      if(entregado == '-'){contador = contador -1;}
```

```
      }
      __delay_ms(500);
   }
}
//*****************************************************************************
// Configuración
//*****************************************************************************
void config_P(){
   //Configuracion de los puertos
   TRISD = 0;
   TRISE = 0;
   TRISA = 3;
   //TRISCbits.TRISC7 = 1;
   //TRISCbits.TRISC6 = 0;
   ANSEL = 3;//Para los potenciometros
   ANSELH = 0;
   //Steo los puertos
   PORTD = 0;
   PORTE = 0;
   PORTC = 0;
}
//*****************************************************************************
// Funciones
//*****************************************************************************
float ADC_1(void){
   Canal_ADC(0);//canal 0
   //Configuracion bits ADCON0
   ADCON0bits.ADCS0 = 1;//Clock ADC conversion
   ADCON0bits.ADCS1 = 0;
```

```
    ADCON0bits.ADON = 1;//Habilitamos el ADC
    __delay_ms(0.25);//Para la conversion
    ADCON0bits.GO = 1;//Inicia la conversion
    while (ADCON0bits.GO == 1){
        volt = ((ADRESH * 5.0)/255);//Conversion de 0V-5V
    }
}
float ADC_2(void){
    Canal_ADC(1);//Canal 1
    //Configuracion bits ADCON0
    ADCON0bits.ADCS0 = 1;//Clock ADC conversion
    ADCON0bits.ADCS1 = 0;
    ADCON0bits.ADON = 1;//Habilitamos el ADC
    __delay_ms(0.25);//Para la conversion
    ADCON0bits.GO = 1;//Inicia la conversion
    while (ADCON0bits.GO == 1){
        volt2 = ((ADRESH * 5.0)/255); //Conversion de 0V-5V
    }
}
void Enviar_1(void){//Envio de datos
    TXREG = volt;
    while (TXSTAbits.TRMT == 1){//Retorna y envia el voltaje a ADC1
        return;
    }
}
void Enviar_2(void){//Envio de datos
    TXREG = volt2;
    while (TXSTAbits.TRMT == 1){//Retorna y envia el voltaje a ADC2
        return;
```

```
    }
}
```

Librería USART

```
#include <xc.h>
#include <pic16f887.h>
#include "USART.h"

void _baudios(void){
    SPBRG = 12; //9600 baudios para 8MHZ
}
//Configuracion dada en el datasheet
void config_txsta(void){
    TXSTAbits.CSRC = 0;//Clock terminal
    TXSTAbits.TX9 = 0;//8 bits de transmicion
    TXSTAbits.TXEN = 1;//Transmicion habilitada
    TXSTAbits.SYNC = 0;//modo asincrono
    TXSTAbits.BRGH = 0;//low speed
    TXSTAbits.TRMT = 0;//Tsr full
    TXSTAbits.TX9D = 0;
}
//Configuracion dada en el datasheet
void config_rcsta(void){
    RCSTAbits.SPEN = 1;//Se habilita el puerto serial
    RCSTAbits.RX9 = 0;
    RCSTAbits.SREN = 0;
    RCSTAbits.CREN = 1;//Recibir habilitadp
    RCREG = 0;
}
//Extraido de https://electrosome.com/uart-pic-microcontroller-mplab-xc8/
```

```
void Write_USART(uint8_t a){
   while(!TRMT);
   TXREG=a;
}
void Write_USART_String(char *a){
   uint8_t i;
   for(i=0;a[i]!='\0';i++){
      Write_USART(a[i]);
   }
}
uint8_t Read_USART(){
  while(!RCIF);
  return RCREG;
}
```

Librería LCD

```
//Libreria de Pablo Mazariegos en clase de 4 bits modificada a unos de 8bits

#include <xc.h>
#include <stdint.h>
#include "LCD.h"
#define _XTAL_FREQ 8000000

//Funcion para indicar el caracter segun sea el tamaño del mismo.
void Puerto(uint8_t x){
          if(x & 1){D0 = 1;}else{D0 = 0;}
   if(x & 2){D1 = 1;}else{D1 = 0;}
   if(x & 4){D2 = 1;}else{D2 = 0;}
   if(x & 8){D3 = 1;}else{D3 = 0;}
   if(x & 16){D4 = 1;}else{D4 = 0;}
```

```
    if(x & 32){D5 = 1;}else{D5 = 0;}
    if(x & 64){D6 = 1;}else{D6 = 0;}
    if(x & 128){D7 = 1;}else{D7 = 0;}
}

//Funcion para imprimir el caracter
void LCD_CMD(char a){
    RS = 1;//Las direcciones a los caracteres
    Puerto(a);
    EN = 1;//Mandar el valor
    __delay_us(5);
    EN = 0;//Verificar si el valor de carac llego
    __delay_us(5);
    __delay_us(50);
}
//Funcion para mandar los datos a la LCD
void datosLCD(uint8_t x){
    RS = 0;//Modifica el contraste de la patalla
    Puerto(x);
    EN = 1;//Mandar el valor
    __delay_us(5);
    EN = 0;//Verificar si el valor de carac llego
    __delay_us(5);
    __delay_ms(2);
}
//Funcion para limpiar la LCD
void LCD_Limpia(void){
    datosLCD(0);
    datosLCD(1);
```

```
}
//Funcion para iniciar la LCD
//En base de la presentacion de clase.
void Lcd_Init(){
   __delay_ms(20);
   datosLCD (0x30);
   __delay_ms(5);
   datosLCD (0x30);
   __delay_us(100);
   datosLCD (0x30);
   __delay_us(100);
   datosLCD (0x38);
   __delay_us(60);
   datosLCD (0x08);
   __delay_us(60);
   datosLCD (0x01);
   __delay_ms(5);
   datosLCD (0x06);
   __delay_us(60);
   datosLCD (0x0C);
   __delay_us(60);
}
//Funcion para configurar el cursor
void Lcd_Set_Cursor(uint8_t x, uint8_t y){
        uint8_t a;
        if(x == 1){//Linea que se coloca arriba
          a = 0x80 + y;//direccion(hexadecimal) y posicion para colocarlo en la fila
              //adecuada para ire leyendo adecuadamente
                  datosLCD(a);
```

```
   }
          else if(x == 2){//Linea que se coloca abajo
            a = 0xC0 + y;//direccion(hexadecimal) y posicion para colocarlo en la fila
                //adecuada para ire leyendo adecuadamente
                    datosLCD(a);
   }
}
//Funcion para mandar un string
void Lcd_Write_String(char *a){
   //funcion para poder imprimir texto usando el puntero
   //para guardar la direccion del registro o valor de a
          int i;
          for(i=0;a[i]!='\0';i++)
            LCD_CMD(a[i]);
}
```

Librería ADC

```
#include <xc.h>
#include <stdint.h>
#include "ADC.h"
#define _XTAL_FREQ 8000000

void config_ADC(void){
   ADCON1 = 0b00000000;//Justificado a la izquierda
}

unsigned Canal_ADC(unsigned short x){ //Fosc/8,datasheet
   switch(x){
      //Canal analogico
      case 0:
```

```
    ADCON0bits.CHS3 = 0;
    ADCON0bits.CHS2 = 0;
    ADCON0bits.CHS1 = 0;
    ADCON0bits.CHS0 = 0;//Canal00
    break;
  case 1:
    ADCON0bits.CHS3 = 0;
    ADCON0bits.CHS2 = 0;
    ADCON0bits.CHS1 = 0;
    ADCON0bits.CHS0 = 1;//Canal1
    break;
  case 2:
    ADCON0bits.CHS3 = 0;
    ADCON0bits.CHS2 = 0;
    ADCON0bits.CHS1 = 1;
    ADCON0bits.CHS0 = 0;//Canal2
    break;
  case 3:
    ADCON0bits.CHS3 = 0;
    ADCON0bits.CHS2 = 0;
    ADCON0bits.CHS1 = 1;
    ADCON0bits.CHS0 = 1;//Canal3
    break;
  case 4:
    ADCON0bits.CHS3 = 0;
    ADCON0bits.CHS2 = 1;
    ADCON0bits.CHS1 = 0;
    ADCON0bits.CHS0 = 0;//Canal4
    break;
```

```
case 5:
   ADCON0bits.CHS3 = 0;
   ADCON0bits.CHS2 = 1;
   ADCON0bits.CHS1 = 0;
   ADCON0bits.CHS0 = 1;//Canal5
   break;
case 6:
   ADCON0bits.CHS3 = 0;
   ADCON0bits.CHS2 = 1;
   ADCON0bits.CHS1 = 1;
   ADCON0bits.CHS0 = 0;//Canal6
   break;
case 7:
   ADCON0bits.CHS3 = 0;
   ADCON0bits.CHS2 = 1;
   ADCON0bits.CHS1 = 1;
   ADCON0bits.CHS0 = 1;//Canal7
   break;
case 8:
   ADCON0bits.CHS3 = 1;
   ADCON0bits.CHS2 = 0;
   ADCON0bits.CHS1 = 0;
   ADCON0bits.CHS0 = 0;//Canal8
   break;
case 9:
   ADCON0bits.CHS3 = 1;
   ADCON0bits.CHS2 = 0;
   ADCON0bits.CHS1 = 0;
   ADCON0bits.CHS0 = 1;//Canal9
```

```
    break;
  case 10:
    ADCON0bits.CHS3 = 1;
    ADCON0bits.CHS2 = 0;
    ADCON0bits.CHS1 = 1;
    ADCON0bits.CHS0 = 0;//Canal10
    break;
  case 11:
    ADCON0bits.CHS3 = 1;
    ADCON0bits.CHS2 = 0;
    ADCON0bits.CHS1 = 1;
    ADCON0bits.CHS0 = 1;//Canal11
    break;
  case 12:
    ADCON0bits.CHS3 = 1;
    ADCON0bits.CHS2 = 1;
    ADCON0bits.CHS1 = 0;
    ADCON0bits.CHS0 = 0;//Canal12
    break;
  case 13:
    ADCON0bits.CHS3 = 1;
    ADCON0bits.CHS2 = 1;
    ADCON0bits.CHS1 = 0;
    ADCON0bits.CHS0 = 1;//Canal13
    break;
  case 14:
    ADCON0bits.CHS3 = 1;
    ADCON0bits.CHS2 = 1;
    ADCON0bits.CHS1 = 1;
```

```
            ADCON0bits.CHS0 = 0;//CVref
            break;
        case 15:
            ADCON0bits.CHS3 = 1;
            ADCON0bits.CHS2 = 1;
            ADCON0bits.CHS1 = 1;
            ADCON0bits.CHS0 = 1;//Fixed Ref
            break;
        default:
            ADCON0bits.CHS3 = 0;
            ADCON0bits.CHS2 = 0;
            ADCON0bits.CHS1 = 0;
            ADCON0bits.CHS0 = 0;//Canal 0
            break;
    }
}
```