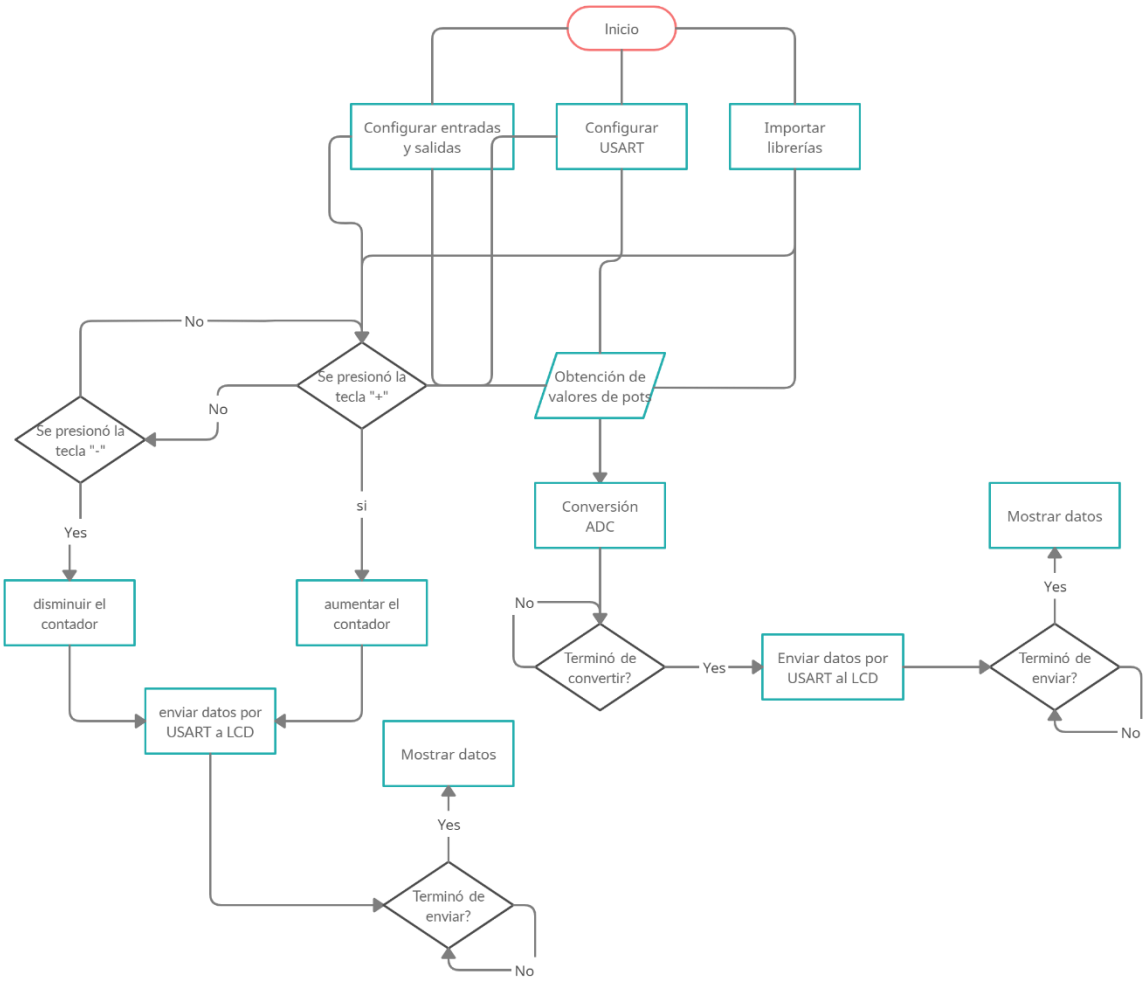
Universidad del Valle de Guatemala
Electrónica Digital 2
Sección 30

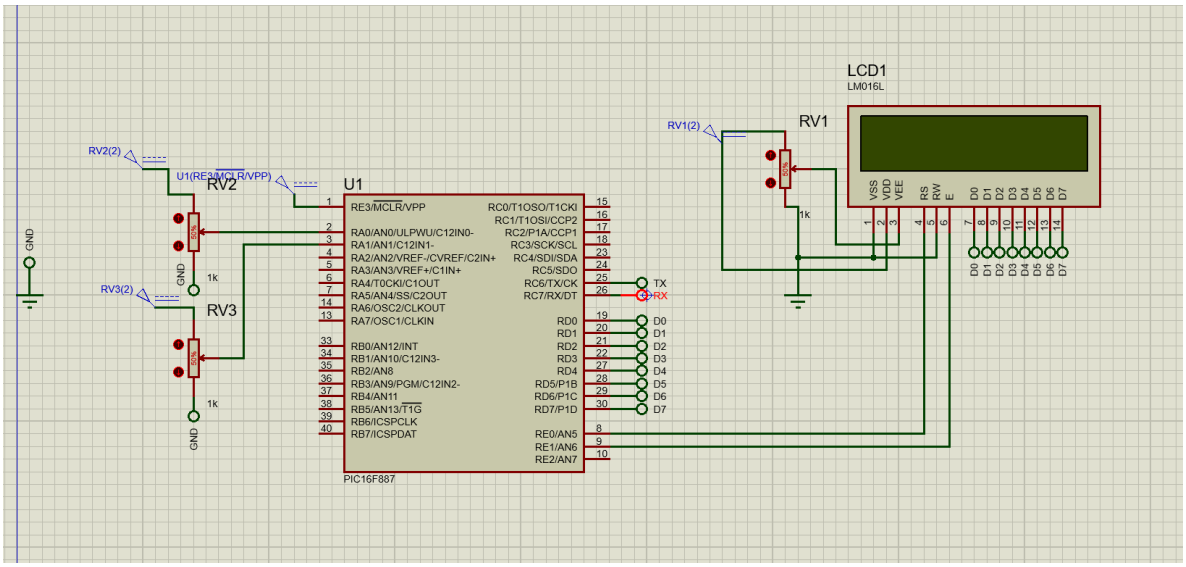
Cristopher René Sagastume 18640

4/02/2021

Pseudocódigo



Esquemático Proteus



Código comentado

Main

```
/*
 * File:   Carrera.c
 * Author: Cristopher Sagastume 18640
 *
 * Created on 22 de enero de 2021, 08:46 AM
 */
//******************************************************************************
// PALABRA DE CONFIGURACIÓN
//******************************************************************************

// CONFIG1
#pragma config FOSC = INTRC_CLKOUT// Oscillator Selection bits (INTOSC oscillator: CLKOUT function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLKIN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT enabled)
#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 = OFF      // Brown Out Reset Selection bits (BOR enabled)
#pragma config IESO = OFF       // Internal External Switchover bit (Internal/External Switchover mode is enabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is enabled)
#pragma config LVP = OFF        // Low Voltage Programming Enable bit (RB3/PGM pin has PGM function, low voltage programming enabled)

// 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)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.
#include <stdio.h>
#include <xc.h>
#include<stdint.h>
#include "ADC.h"
#include "LCD.h"
#include "USART.h"
#define _XTAL_FREQ 8000000
//*****************************************************************************
//  Variables
//*****************************************************************************
uint8_t flag = 1;
uint8_t turno = 1;
uint8_t retorno;
uint8_t valor_AN0;
uint8_t valor_AN1;
double valor1;
double valor2;
```

```
float x;
char s[20];
uint8_t count = 0;
uint8_t recibido;
uint8_t enviado;
//*****************************************************************************
//  Prototipo de funciones
//*****************************************************************************
void setup(void);
void __interrupt() ISR(void);
double conversor(uint8_t x);

//*****************************************************************************
//  Ciclo Principal
//*****************************************************************************

void main(void) {
   setup();
   Config_USARTT();
   recibir();
   Lcd_Init();
   Lcd_Clear();
   while (1) {
      Lcd_Set_Cursor(1, 1); //colocamos el cursor en posicón 1,1
      Lcd_Write_String("S1:   S2:    S3:"); //escribimos los encabezados
      retorno = ADC_con(flag, turno); //convertimos los valores de ADC
      valor1 = conversor(valor_AN0); //convertimos la resolución del ADC a
      //valores de voltaje en decimal
      valor2 = conversor(valor_AN1);
      Lcd_Set_Cursor(2, 1);//colocamos el cursor en posición 2,1
      sprintf(s, "%3.2fV", valor1); //guardamos los valores de la conversión
```

```
      //en el array s con un formato de 2 decimales en caracter flotante
      Lcd_Write_String(s); //escribimos los valores en la LCD
      //       enviar(s);
      Lcd_Set_Cursor(2, 7);
      sprintf(s, "%3.2fV", valor2);
      Lcd_Write_String(s);
      enviar(s);
      __delay_ms(300);
      if (recibido == 0x2B) { //0x2B == signo "+" revisamos si lo recibimos
         count++; //aumentamos el contador
         recibido = 0;//vaciamos la variable de recibido
      }
      else if (recibido == 0x2D){//0x2D == signo "-" revisamos si lo recibimos
         count--;//disminuimos el contador
         recibido = 0; //vaciamos la variable de recibido
      }
      sprintf(s, "%d", count);//se guarda el valor de contador en s con
      //formado de decimal
      Lcd_Set_Cursor(2, 14);
      Lcd_Write_String(s);
      //       enviar(s);
      if (count >= 0 && count < 10) {//limpiamos los espacios de decenas y
         //centenas si en caso no se necesitaran
         Lcd_Set_Cursor(2, 15);
         Lcd_Write_String("   ");

      }
   }
}
//*****************************************************************************
//  Configuración
```

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

void setup(void) {
   TRISD = 0b00000000;// puerto D como salida
   TRISC = 0b10000000; //activamos el RX como entrada
   TRISE = 0b00000000;
   ANSEL = 0b00000011;
   PORTC = 0;//limpiamos puertos
   PORTD = 0;
   PORTE = 0;
}



//*****************************************************************************
//  Funciones
//*****************************************************************************

double conversor(uint8_t val) {
   x = 0.0195 * val; //5V/256bits=0.0195 convertir de bits a voltaje
   return (x);
}



//*****************************************************************************
//  Interrupciones
//*****************************************************************************

void __interrupt() ISR(void) {
   if (PIR1bits.ADIF == 1) {//verificamos si fue interrupt ADC
      if (retorno == 0) { //verificamos si fue del AN0
         valor_AN0 = ADRESH; //guardamos el valor
```

```
            turno = 0;//cambiamos a conversión de AN1
            flag = 1;
        } else if (retorno == 1) {//verificamos si fue del AN1
            valor_AN1 = ADRESH;
            turno = 1;//cambiamos a conversión de AN0
            flag = 1;//se setea flag para indicar que puede volver a convertir
        }
//        flag = 1;

        PIR1bits.ADIF = 0; //apagamos la bandera de ADC
    }
    if (PIR1bits.RCIF == 1){//verificamos si fue interrupcion de la recepción USART
        if (RCSTAbits.OERR == 1) {//verificamos si hubo algún error de overrun
            RCSTAbits.CREN = 0;
            __delay_us(300);
        } else {
            recibido = RCREG; //guardamos el valor recibido en una variable
        }
    }
}
```

ADC

```
/*
 * File:   ADC.c
 * Author: SAGASTUME
 *
 * Created on 2 de febrero de 2021, 11:46 PM
 */

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

```
#define _XTAL_FREQ 8000000

uint8_t ADC_con(uint8_t flag, uint8_t turno) {
   uint8_t var;
   ADCON1bits.ADFM = 0; //se justifica la resolución del ADC a la izquierda  en ADRESH
   ANSEL = 0b00000011; //se configura el RA0 como entrada analógica
   INTCON = 0b11101000; //se configuran las interrupciones GIE, PIE, T0IE y RBIE
   //ADCON0 = 0b01000001;//frecuencia de oscilacion 1/8 canal analógico AN0 y
   //encender ADC
   PIE1bits.ADIE = 1; //se configura la interrupcion del ADC
   PIR1bits.ADIF = 0; //se apaga la bandera de interrupcion ADC
   if (flag == 1 && turno == 1) {
      ADCON0 = 0b01000001;//frecuencia de oscilacion 1/8 canal analógico AN0 y
   //encender ADC
      __delay_us(20);
      ADCON0bits.GO = 1; //se indica que empiece a convertir al ADC
      flag = 0;
      var=0;

   } else if (flag == 1 && turno == 0) {
      ADCON0 = 0b01000101;//frecuencia de oscilacion 1/8 canal analógico AN1 y
   //encender ADC
      __delay_us(20);
      ADCON0bits.GO = 1; //se indica que empiece a convertir al ADC
      flag = 0;
      var=1;
   }
   return (var);
}
```

LCD

```
/*
 * File:   LCD.c
 * Author: electroSome.com
 * Editor: Cristopher Sagastume
 *
 * modified on 6 de febrero de 2021, 01:03 AM
 */

#include<stdint.h>
#include <xc.h>
#define _XTAL_FREQ 8000000
#define RS PORTEbits.RE0
#define EN PORTEbits.RE1
//LCD Functions Developed by electroSome

void Lcd_Port(char a) {
   PORTD = a; //se carga a al puerto D completo
}

void Lcd_Cmd(char a) {
   RS = 0; // => RS = 0 se mandan comandos
   Lcd_Port(a);
   EN = 1; // => E = 1
   __delay_ms(4);
   EN = 0; // => E = 0
}

void Lcd_Clear(void) {
   Lcd_Cmd(0);
```

```
    Lcd_Cmd(1);
}

void Lcd_Set_Cursor(char a, char b) {
    char temp;
    if (a == 1) {
        temp = 0x80 + b - 1; //dirección de posicion de DDRAM al puerto
        Lcd_Cmd(temp);
    } else if (a == 2) {
        temp = 0xC0 + b - 1;
        Lcd_Cmd(temp);
    }
}

void Lcd_Init(void) {
    Lcd_Port(0x00);
    __delay_ms(20);
    Lcd_Cmd(0x3F); //0b00111111 valores de iniciación de LCD
    __delay_ms(10);
    Lcd_Cmd(0x3F);//0b00111111
    __delay_us(200);
    Lcd_Cmd(0x3F);//0b00111111
    ////////////////////////////////////////////////////
    Lcd_Cmd(0x38);//0b00111000 modo de funcionamiento
    Lcd_Cmd(0x0C);//0b00001100 encender visualizador
    Lcd_Cmd(0x06);//0b00000110 modo de entrada
}

void Lcd_Write_Char(char a) {
    RS = 1; // => RS = 1 envio de datos
    Lcd_Port(a); //cargando los datos al puerto
```

```
    EN = 1;
    __delay_us(40); //transición EN:1->0 para indicar que se mandan datos/comandos
    EN = 0;
    __delay_us(40);
}

void Lcd_Write_String(char *a) {
    int i;
    for (i = 0; a[i] != '\0'; i++) //ciclo cargar direcciones/valores a LCD
        Lcd_Write_Char(a[i]);
}
```

USART

```
/*
 * File:   USART.c
 * Author: SAGASTUME
 *
 * Created on 6 de febrero de 2021, 12:41 AM
 */



#include <xc.h>
#define _XTAL_FREQ 8000000
void Config_USARTT(void) {
    TXSTAbits.SYNC = 0; //modo asincrono
    TXSTAbits.TXEN = 1; //activamos la transmisión
    TXSTAbits.BRGH = 0; //velocidad baja de baud rate (velocidad de trabajo de la comuniación)
    BAUDCTLbits.BRG16 = 1; //geneardor de 16 bits de baud rate
    SPBRG = 25; // tener un baudrate a 19230
    RCSTAbits.SPEN = 1; //se activa la comunación del RX/TX
    INTCONbits.GIE = 1; //se activan la interrupción global
```

```
    return;
}

void enviar(char *valor) {
    TXREG = valor[0];
    while (TRMT == 0) {//bucle si TRMT sigue lleno
    }
    TXREG = valor[1];//si TRMT esta vacio le cargamos otro valor al TXREG->TRMT
    while (TRMT == 0) {
    }
    TXREG = valor[2];
    while (TRMT == 0) {
    }
    TXREG = valor[3];
    while (TRMT == 0) {
    }
    TXREG = 0x20;
    while (TRMT == 0) {
    }
}

void recibir(void) {
    RCSTAbits.CREN = 1; //activamos el recibidor contunio asincrono
    RCSTAbits.FERR = 0; //apagamos el error de frame bit
    PIE1bits.RCIE = 1; //ecendemos la interrupción de recepción de la comunicación USART
    RCSTAbits.OERR = 0; //apagamos el overrun error

}
```

Link del repositorio

https://github.com/sag18640/Electronica_Digital_2.git