Universidad del Valle de Guatemala
Electrónica Digital 2 – Sección 11
Pablo Mazariegos

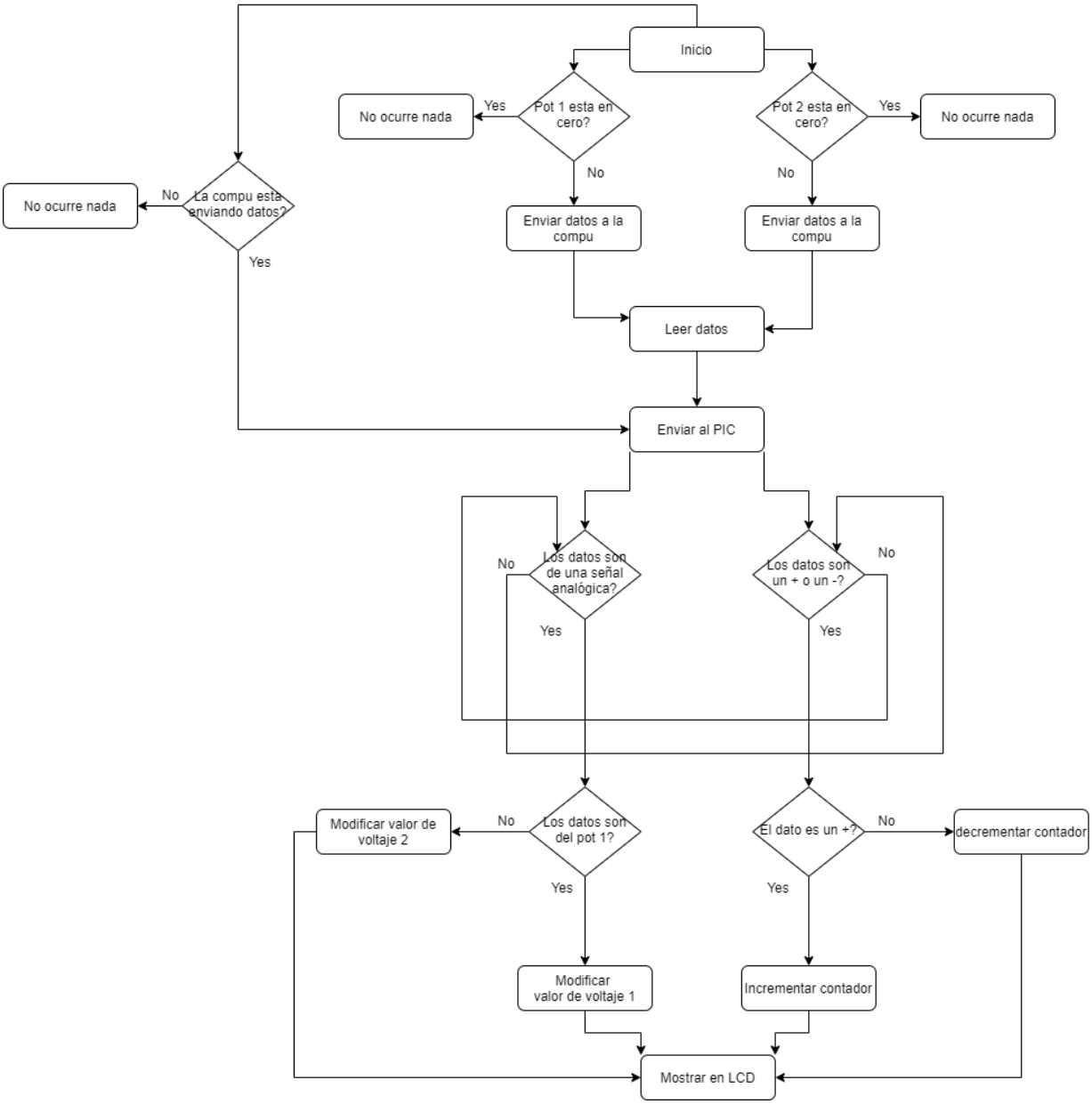
Francisco Montúfar
Carné 19379
11/02/2021

Laboratorio No. 3

LCD 16x2

Link de GitHub: https://github.com/mon19379/DIGITAL2.git

Pseudocódigo:



Descripción:

El laboratorio consiste en la implementación de una pantalla LCD 16x2 y el módulo de comunicación serial. Lo que se quería realizar era que la pantalla LCD mostrara los valores de voltaje de 2 potenciómetros y el valor de un contador. Para poder realizar eso, se utilizó la comunicación serial, para poder enviar y recibir datos.

Código:

Main:

```
/*
 * File:   newmain.c
 * Author: franc
 *
 * Created on 4 de febrero de 2021, 12:51 PM
 */
```

```
#include <xc.h>
#include <stdint.h>
#include <pic16f887.h>
#include "LCD.h"
#include "adc2.h"
#include "osc.h"
#include "usart.h"
//******************************************************************************
// Palabra de configuración
//******************************************************************************
// CONFIG1
```

#pragma config FOSC = INTRC_NOCLKOUT // 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)

```
#define _XTAL_FREQ 4000000      //SE CONFIGURA EL OSCILADOR EXTERNO


//*****************************************************************************
//Variables
//*****************************************************************************
uint8_t pot1 = 0;
uint8_t pot2 = 0;
uint8_t FLAGADC = 0;
uint8_t CONTADC = 0;
uint8_t TOGGLE = 0;
uint8_t REC = 0;
uint8_t CP1 = 0;
uint8_t DP1 = 0;
uint8_t UP1 = 0;
uint8_t CP2 = 0;
uint8_t DP2 = 0;
uint8_t UP2 = 0;
uint8_t C1 = 0;
uint8_t D1 = 0;
uint8_t U1 = 0;
uint8_t C2 = 0;
uint8_t D2 = 0;
uint8_t U2 = 0;
uint8_t SEND = 0;
uint8_t CONT = 0;
```

```
uint8_t R1 = 0;
uint8_t R2 = 0;
uint8_t CONTC = 0;
uint8_t CONTD = 0;
uint8_t CONTU = 0;
uint8_t CO1 = 0;
uint8_t CO2 = 0;
uint8_t CO3 = 0;



//*****************************************************************************
// Prototipos de funciones
//*****************************************************************************
void Setup(void);
void pots(void);
void mandar(void);
void map(void);
void recibir(void);
void map2(void);
//*****************************************************************************
//Interrupción
//*****************************************************************************
```

```
void __interrupt() ISR(void) {

    if (INTCONbits.T0IF == 1) {
        TMR0 = 236;
        CONTADC++;
        INTCONbits.T0IF = 0;
    }

    if (PIR1bits.ADIF == 1) {
        pots();
        PIR1bits.ADIF = 0;
    }

    if (PIR1bits.RCIF == 1) {
        REC = RCREG;
        recibir();
    }

    if (PIR1bits.TXIF == 1) {
        mandar();
        SEND++;
        PIE1bits.TXIE = 0;

    }
```

```
}
//*****************************************************************************
//Ciclo pincipal
//*****************************************************************************

void main(void) {

    Setup();
    Lcd_Set_Cursor(1, 1);
    Lcd_Write_String("V1");
    Lcd_Set_Cursor(1, 7);
    Lcd_Write_String("V2");
    Lcd_Set_Cursor(1, 13);
    Lcd_Write_String("CONT");




    //*****************************************************************************
    // Loop principal
    //*****************************************************************************
    while (1) {
        map();
```

```
map2();
if (CONTADC > 20) {
    ADCON0bits.GO_nDONE = 1;
    CONTADC = 0;
    PIE1bits.TXIE = 1;



}

Lcd_Set_Cursor(2, 1);
Lcd_Write_Char(C1);
Lcd_Set_Cursor(2, 2);
Lcd_Write_String(".");
Lcd_Write_Char(D1);
Lcd_Set_Cursor(2, 4);
Lcd_Write_Char(U1);



Lcd_Set_Cursor(2, 7);
Lcd_Write_Char(C2);
Lcd_Set_Cursor(2, 8);
Lcd_Write_String(".");
Lcd_Set_Cursor(2, 9);
Lcd_Write_Char(D2);
Lcd_Set_Cursor(2, 10);
```

```
        Lcd_Write_Char(U2);

        Lcd_Set_Cursor(2, 13);
        Lcd_Write_Char(CO1);
        Lcd_Set_Cursor(2, 14);
        Lcd_Write_Char(CO2);
        Lcd_Set_Cursor(2, 15);
        Lcd_Write_Char(CO3);









    }
}
//*****************************************************************************
//Configuracion
//*****************************************************************************
```

```
void Setup(void) {
   TRISD = 0;
   TRISE = 0; //PUERTO E SALIDAS
   initOsc(6); //SE LLAMA LA CONFIG DEL OSCILADOR
   configADC2(1, 12); //SE LLAMA LA CONFIG DEL ADC
   usart();
   Lcd_Init();
   Lcd_Cmd(0x8A);
   ANSEL = 0; // ENTRADAS DIGITALES Y BIT 0 ANALÓGICA
   ANSELH = 0b00000011;
   PORTA = 0; //PUERTO A EN 0
   PORTB = 0; //PUERTO B EN 0
   PORTC = 0; //PUERTO C EN 0
   PORTD = 0; //PUERTO D EN 0
   PORTE = 0; //PUERTO E EN 0
   //PINES RA0 Y RA2 COMO ENTRADAS, LOS DEMAS COMO SALIDAS
   TRISC = 0b10000000; //PUERTO C SALIDAS
   TRISA = 0; //PUERTO A SALIDAS
   TRISB = 0b00000011; //PUERTO B
   OPTION_REG = 0b10000111; //SE APAGAN LAS PULLUPS DEL PUERTO B
   INTCONbits.GIE = 1; //SE HABILITAN LAS INTERRUPCIONES GLOBALES
   INTCONbits.T0IE = 1; //SE HABILITA LA INTERRUPCION DEL TIMER0
```

```
   INTCONbits.PEIE = 1; //SE HABILITAN LAS INTERRUPCIONES PERIFERICAS
   PIE1bits.ADIE = 1; //SE HABILITA LA INTERRUPCION DEL ADC
   INTCONbits.T0IF = 0; // SE LIMPIA LA BANDERA DE INTERRUPCION DEL TIMER 0
   PIR1bits.ADIF = 0; //SE LIMPIOA LA BANDERA DE INTERRUPCION DEL ADC
   PIR1bits.TXIF = 0;
   PIE1bits.TXIE = 1;
   PIE1bits.RCIE = 1;
   PIR1bits.RCIF = 0;




}

//*****************************************************************************
// Subrutinas
//*****************************************************************************

void pots(void) {

   if (TOGGLE == 0) {
      configADC2(1, 12);
      pot1 = ADRESH;
      TOGGLE = 1;
```

```
    } else {
        configADC2(1, 10);
        pot2 = ADRESH;
        TOGGLE = 0;
    }
}

void map(void) {
    CP1 = ((pot1) / 51);
    DP1 = (((pot1 * 100) / 51)-(CP1 * 100)) / 10;
    UP1 = (((pot1 * 100) / 51)-(CP1 * 100)-(DP1 * 10));

    CP2 = ((pot2) / 51);
    DP2 = (((pot2 * 100) / 51)-(CP2 * 100)) / 10;
    UP2 = (((pot2 * 100) / 51)-(CP2 * 100)-(DP2 * 10));

    C1 = (CP1 + 0x30);
    D1 = (DP1 + 0x30);
    U1 = (UP1 + 0x30);

    C2 = (CP2 + 0x30);
    D2 = (DP2 + 0x30);
    U2 = (UP2 + 0x30);
}
```

```
void mandar(void) {
   switch (SEND) {

      case 0:
         TXREG = 0x28;
         break;

      case 1:
         TXREG = C1;
         break;

      case 2:
         TXREG = 0x2E;
         break;
      case 3:
         TXREG = D1;
         break;
      case 4:
         TXREG = U1;
         break;
      case 5:
         TXREG = 0x29;
         break;

      case 6:
```

```
    TXREG = 0x2C;
    break;

case 7:
    TXREG = 0x20;
    break;

case 8:
    TXREG = 0x28;
    break;

case 9:
    TXREG = C2;
    break;
case 10:
    TXREG = 0x2E;
    break;
case 11:
    TXREG = D2;
    break;
case 12:
    TXREG = U2;

    break;
```

```
        case 13:
            TXREG = 0x29;



        case 14:
            TXREG = 0x0D;
            SEND = 0;
            break;
    }

}

void recibir(void) {
    if (REC == 43) {
        R1 = 1;



    }

    if (REC != 43 && R1 == 1) {
        R1 = 0;
        CONT++;
    }

    if (REC == 45) {
```

```
        R2 = 1;



    }

    if (REC != 45 && R2 == 1) {
        R2 = 0;
        CONT--;
    }
}

void map2 (void){
    CONTC = (CONT/100);
    CONTD = (CONT- (CONTC*100))/10;
    CONTU = (CONT - (CONTC*100)-(CONTD*10));

    CO1 = (CONTC + 0x30);
    CO2 = (CONTD + 0x30);
    CO3 = (CONTU + 0x30);
}
```

Librerías

```
/*
 * File:   LCD.c
```

```
 * Author: Extraído de electrosome.com
 *
 * Created on 4 de febrero de 2021, 12:52 PM
 */

#include <xc.h>
#include <stdint.h>
#include "LCD.h"

void Lcd_Port(char a) {
    PORTD = a;
}

void Lcd_Cmd(char a) {
    Lcd_Port(a);
    RS = 0; // => RS = 0

    EN = 1; // => E = 1
    __delay_ms(5);
    EN = 0; // => E = 0
}

Lcd_Clear() {
    Lcd_Cmd(0);
    Lcd_Cmd(1);
```

```
}

void Lcd_Set_Cursor(char a, char b) {
    char temp;
    if (a == 1) {
        temp = 0x80 + b - 1;
        Lcd_Cmd(temp);
    } else if (a == 2) {
        temp = 0xC0 + b - 1;
        Lcd_Cmd(temp);
    }
}

void Lcd_Init() {

    //////////////////////////////////////////////////////

    Lcd_Cmd(0x38);
    Lcd_Cmd(0x0C);
    Lcd_Cmd(0x06);
    Lcd_Cmd(0x80);
}

void Lcd_Write_Char(char a) {
    RS = 1;             // => RS = 1
```

```
  Lcd_Port(a);            //Data transfer
  EN = 1;
  __delay_us(40);
  EN = 0;
  RS = 0;
}

void Lcd_Write_String(char *a) {
  int i;
      for(i=0;a[i]!='\0';i++)
        Lcd_Write_Char(a[i]);
}

void Lcd_Shift_Right() {
   Lcd_Cmd(0x1C);

}

void Lcd_Shift_Left() {
   Lcd_Cmd(0x18);

}

#include <pic16f887.h>
```

```
#include "usart.h"

void usart(void){

    //CONFIG TX
    TXSTAbits.TX9 = 0; //TRANSMISION DE 8 BITS
    TXSTAbits.SYNC = 0; //ASINCRONO
    TXSTAbits.BRGH = 1; //HIGH SPEED
    BAUDCTLbits.BRG16 = 0; //BAUD RATE DE 8 BITS
    SPBRGH = 0;
    SPBRG = 25;
    PIE1bits.TXIE = 1;
    TXSTAbits.TXEN = 1;

    //CONFIG RX
    RCSTAbits.SPEN = 1;
    RCSTAbits. RX9 = 0;
    RCSTAbits.CREN = 1;




}
/*
 * File:   adc2.c
```

```
 * Author: franc
 *
 * Created on 4 de febrero de 2021, 06:25 PM
 */

#include <pic16f887.h>
#include <xc.h>
#include "adc2.h"
#define _XTAL_FREQ 4000000

//*****************************************************************************
// CONFIGURACION DEL ADC
//*****************************************************************************

void configADC2(uint8_t fosc, uint8_t chan) {

   switch (fosc) {

      case 0:
         ADCON0bits.ADCS = 0b00;
         break;

      case 1:
         ADCON0bits.ADCS = 0b01;
         break;
```

```
    case 2:
        ADCON0bits.ADCS = 0b10;
        break;

    case 3:
        ADCON0bits.ADCS = 0b11;
        break;

    default:
        ADCON0bits.ADCS = 0b00;
        break;
}
switch (chan) {
    case 0:
        ADCON0bits.CHS = 0b0000;
        break;

    case 1:
        ADCON0bits.CHS = 0b0001;
        break;

    case 2:
        ADCON0bits.CHS = 0b0010;
        break;
```

```
case 3:
   ADCON0bits.CHS = 0b0011;
   break;

case 4:
   ADCON0bits.CHS = 0b0100;
   break;

case 5:
   ADCON0bits.CHS = 0b0101;
   break;

case 6:
   ADCON0bits.CHS = 0b0110;
   break;

case 7:
   ADCON0bits.CHS = 0b0111;
   break;

case 8:
   ADCON0bits.CHS = 0b1000;
   break;
```

```
case 9:
   ADCON0bits.CHS = 0b1001;
   break;

case 10:
   ADCON0bits.CHS = 0b1010;
   break;

case 11:
   ADCON0bits.CHS = 0b1011;
   break;

case 12:
   ADCON0bits.CHS = 0b1100;
   break;

case 13:
   ADCON0bits.CHS = 0b1101;
   break;

case 14:
   ADCON0bits.CHS = 0b1110;
   break;

case 15:
```

```
            ADCON0bits.CHS = 0b1111;
            break;

        default:
            ADCON0bits.CHS = 0b0000;
            break;


    }

    __delay_ms(10);


    ADCON0bits.GO_nDONE = 1;



    ADCON0bits.ADON = 1;
    ADCON1 = 0;

}

/*
 * File:   adc2.c
 * Author: franc
```

```
 *
 * Created on 4 de febrero de 2021, 06:25 PM
 */

#include <pic16f887.h>
#include <xc.h>
#include "adc2.h"
#define _XTAL_FREQ 4000000

//*****************************************************************************
// CONFIGURACION DEL ADC
//*****************************************************************************

void configADC2(uint8_t fosc, uint8_t chan) {

   switch (fosc) {

      case 0:
         ADCON0bits.ADCS = 0b00;
         break;

      case 1:
         ADCON0bits.ADCS = 0b01;
         break;
```

```
    case 2:
        ADCON0bits.ADCS = 0b10;
        break;

    case 3:
        ADCON0bits.ADCS = 0b11;
        break;

    default:
        ADCON0bits.ADCS = 0b00;
        break;
}
switch (chan) {
    case 0:
        ADCON0bits.CHS = 0b0000;
        break;

    case 1:
        ADCON0bits.CHS = 0b0001;
        break;

    case 2:
        ADCON0bits.CHS = 0b0010;
        break;
```

```
case 3:
   ADCON0bits.CHS = 0b0011;
   break;

case 4:
   ADCON0bits.CHS = 0b0100;
   break;

case 5:
   ADCON0bits.CHS = 0b0101;
   break;

case 6:
   ADCON0bits.CHS = 0b0110;
   break;

case 7:
   ADCON0bits.CHS = 0b0111;
   break;

case 8:
   ADCON0bits.CHS = 0b1000;
   break;

case 9:
```

```
    ADCON0bits.CHS = 0b1001;
    break;

case 10:
    ADCON0bits.CHS = 0b1010;
    break;

case 11:
    ADCON0bits.CHS = 0b1011;
    break;

case 12:
    ADCON0bits.CHS = 0b1100;
    break;

case 13:
    ADCON0bits.CHS = 0b1101;
    break;

case 14:
    ADCON0bits.CHS = 0b1110;
    break;

case 15:
    ADCON0bits.CHS = 0b1111;
```

```
            break;

        default:
            ADCON0bits.CHS = 0b0000;
            break;


    }

    __delay_ms(10);


    ADCON0bits.GO_nDONE = 1;



    ADCON0bits.ADON = 1;
    ADCON1 = 0;

}
```