Universidad del Valle de Guatemala
Electrónica Digital 2
Sección 30

Cristopher René Sagastume 18640

29/01/2021

Pseudocódigo

- Inicio
- Revisar si se presionó el botón de incrementar contador (con respectivo anti rebote)
    - Si se presionó aumentar el contador en 1 y cargar el valor al puerto de salida de leds.
- Revisar si se presionó el botón de decrementar contador (con respectivo anti rebote)
    - Si se presionó disminuir el contador en 1 y cargar el valor al puerto de salida de leds.
- Cargar los valores analógicos del pot al ADC y almacenar el resultado de la conversión en un registro de 8 bits
- Desplegar el valor obtenido de la conversión del ADC en los displays multiplexados.
- Establecer un valor de referencia en el contador el cual al se debe verificar si el valor del contador es menor al del display debe encenderse la alarma visual (led)
- Fin.

Código comentado

Main

```
/*
 * File:   Carrera.c
 * Author: Cristopher Sagastume 18640
 *
 * Created on 29 de enero de 2021, 08:46 AM
 */
//******************************************************************************
// PALABRA DE CONFIGURACIÓN
```

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

// CONFIG1
#pragma config FOSC = INTRC_CLKOUT
#pragma config WDTE = OFF
#pragma config PWRTE = OFF
#pragma config MCLRE = OFF
#pragma config CP = OFF
#pragma config CPD = OFF
#pragma config BOREN = OFF
#pragma config IESO = OFF
#pragma config FCMEN = OFF
#pragma config LVP = OFF

// CONFIG2
#pragma config BOR4V = BOR40V
#pragma config WRT = OFF

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.
#include<stdint.h>
#include <xc.h>
#include "ADC.h"
#include "Multiplexado.h"

//*****************************************************************************
//  Variables
```

```
//*****************************************************************************
uint8_t count;
uint8_t flag = 1;
uint8_t valor_MSB;
uint8_t valor_LSB;
uint8_t comparacion;
uint8_t seg1;
uint8_t flag2;
uint8_t seg2;
uint8_t tabla_7seg[] = {0b00111111, 0b00000110, 0b01011011, 0b01001111,
   0b01100110, 01111101, 0b00000111, 0b01111111, 0b01101111, 0b01110111,
   0b01111100, 0b00111001, 0b01011110, 0b01111001, 0b01110001};//tabla displays
#define _XTAL_FREQ 8000000


//*****************************************************************************
//  Prototipo de funciones
//*****************************************************************************
void setup(void);
void __interrupt() ISR(void);


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

void main(void) {
```

```
    setup();
    count = 0;
    flag = 1;
    Multiplex();
    while (1) {
        ADC_con(flag);
        //se le da el valor constantemente a puerto D
        PORTD = count;
        seg1 = tabla_7seg[valor_MSB];
        seg2 = tabla_7seg[valor_LSB];
        //comparación para alarma visual si la resolución del ADC es mayor
        //al contador de 8 bits
        if (comparacion > count) {
            PORTBbits.RB0 = 1;
        } else {
            PORTBbits.RB0 = 0;
        }
    }

}



//**********************************************************************
//  Configuración
//**********************************************************************

void setup(void) {
```

```
   TRISD = 0b00000000; //puerto D como salida contador leds
   TRISC = 0b00000000; //puerto C como salida display
   TRISB = 0b11111110; //puerto B como entradas a excepción del pin 0 como-
   //salida alarma
   TRISE = 0b00000000; //puerto E como salida transistores
   IOCBbits.IOCB6 = 1;
   IOCBbits.IOCB7 = 1;
   PORTB = 0; //limpiamos puertos
   PORTC = 0;
   PORTD = 0;
   PORTE = 0;
   flag2 = 0;
}



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

void __interrupt() ISR(void) {
   if (PIR1bits.ADIF == 1) {//verificamos si fue interrupt ADC
      flag = 1;
      valor_MSB = ADRESH >> 4; //11110000 00001111
      valor_LSB = ADRESH & 0b00001111;
      comparacion = ADRESH;
      PIR1bits.ADIF = 0;//apagamos la bandera de ADC
   }
```

```
if (INTCONbits.T0IF == 1) { //verificamos si fue tmr0 interrupt
   TMR0 = 10;
   if (PORTEbits.RE0 == 1) {
      PORTEbits.RE0 = 0;
      PORTEbits.RE1 = 1;
      PORTC = seg2;
   } else {
      PORTEbits.RE1 = 0;
      PORTEbits.RE0 = 1;
      PORTC = seg1;
   }
   INTCONbits.T0IF = 0;//apagamos la bandera de interrupt tmr0
}
if (INTCONbits.RBIF == 1) {//verificamos si fue interrupt on change
   if (PORTBbits.RB6 == 0) {//antirebote aumentar
      while (flag2 == 0) {
         if (PORTBbits.RB6 == 1) {
            flag2 = 1;
         }
      }
      flag2 = 0;
      count++;

   } else if (PORTBbits.RB7 == 0) { //antirebote disminuir
      while (flag2 == 0) {
         if (PORTBbits.RB7 == 1) {
            flag2 = 1;
```

```
            }
        }
        flag2 = 0;
        count--;

    }
  }
  INTCONbits.RBIF = 0; //apagamos la bandera de interrupt on change
}
```

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



void ADC_con(uint8_t flag) {
    //se justifica la resolución del ADC a la izquierda  en ADRESH
  ANSEL = 0b00000001; //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) {
        __delay_us(20);
        ADCON0bits.GO = 1; //se indica que empiece a convertir al ADC
        flag = 0;



    }


}
```

MULTIPLEXADO

```
/*
 * File:   Multiplexado.c
 * Author: SAGASTUME
 *
 * Created on 3 de febrero de 2021, 12:55 PM
 */



#include <xc.h>


void Multiplex(void) {
    OPTION_REG= 0b00000111; // Se coloca prescaler como 1:256
    TMR0=10; //se pre carga el tmr0 con 10 ticks para tener interrupcion
```

```
    // cada 31ms para observar mejor los displays
}
```

Link del repositorio

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