Laboratorio de Microcomputadoras
Práctica No. 10
Programación C
Convertidor A/D, Temporizadores e Interrupciones

Objetivo. Realización de programas usando programación en lenguaje C, utilización del puerto serie, convertidor analógico digital e introducción a aplicaciones con interrupciones.

Introducción

Convertidor A/D

Tal como se estudió y experimentó en la practica 6, el PIC16F877(A) contiene un convertidor A/D de 10 bits, dispone de 8 canales ubicados en los puertos A y E.

Las funciones y directivas a utilizar son:

a. #device ADC=Tamaño del resultado; agregar después de #include <16f877.h>
    a. #device ADC=8 // resolución de 8 bits
    b. #device ADC=10 // resolución de 10 bits
b. Setup_port_a(ALL_ANALOG); // Define el puerto A como analógico
c. Setup_adc(ADC_CLOCK_INTERNAL); // Define frecuencia de muestreo del convertidor A/D
d. Set_adc_channel(num); // Configura el canal a usar
e. Delay_us(20); // Retardo de 20 µs
f. Read_adc(); // Obtener el resultado de la conversión

Interrupciones

Una interrupción es la solicitud al procesador para dejar momentáneamente la ejecución secuencial del programa para atender una petición de interrupción, una vez que se ha reconocido, deberá almacenar el entorno del procesador en la pila, para ser recuperado previo a salir de la rutina de interrupción, para continuar con las instrucciones pendientes. El procesador tiene 14 fuentes de interrupción, estas son:

a. Desbordamiento del TIMER0
b. Detección de flanco de RB0
c. Cambio de nivel de RB4-RB7
d. Desbordamiento del TIMER1
e. TIMER2
f. Módulo CCP1 (Captura y comparación)
g. Módulo CCP2 (Captura y comparación)
h. Escritura en memoria EEPROM
i. Comunicación Serial Síncrona
j. Colisión del bus en la Comunicación Serial Síncrona
k. Recepción de datos en la Comunicación Serial Asíncrona
l. Transferencia de datos en la Comunicación Serial Asíncrona
m. Terminación completa del convertidor A/D
n. Transferencia paralela de datos

Para ser atendida una petición de interrupción se debe cumplir con los siguientes requisitos (usando el compilador de C):

a. Habilitar interrupciones particulares
b. Habilitar interrupciones generales
c. Definir la rutina de interrupción

Funciones en C para control de interrupciones:

- a. Habilitar interrupciones particulares
  - enable_interrups(FUENTE)

- b. Habilitar interrupciones generales
  - enable_interrups(GLOBAL)

- c. Definir la rutina de interrupción; incluir previo a la función principal (main)

```
#int_evento // Fuente de interrupción
Función_de_interrupción(){
                          //código de la rutina de interrupción
                         }
```

- Fuentes de interrupción usadas en esta práctica
  - #INT_RB ; //Cambio de nivel del los cuatro bits más significativos del puerto B
  - #INT_RTCC; //Desbordamiento del TIMER0
  - #INT_RDA; //Recepción de datos por el puerto serie asíncrona
  - #INT_EXT; // Detección de flanco por RB0

TIMER0

El TIMER0 puede ser empleado para trabajar bajo dos modalidades:

a. Temporizador
b. Acumulador de pulsos ocurridos en RA4

Como temporizador incrementa el registro de 8 bits **TMR0**, cada *x* cantidad de ciclos de reloj, establecido como pre_divisor (desde 2 hasta 256), cuando este registro pasa de 0xFF a 0x00 se prende la bandera T0IF; con lo que se genera la petición de interrupción.

- set_timer0(0) ;inicia el timer0 en 0x00
- setup_counters(RTCC_INTERNAL,RTCC_DIV_256);
  - Configura el tiempo de activación de TOIF (**T_{T0IF}**)
    - **T_{T0IF}**=T_{INT}(255)(256)
      - **T_{INT}** : es el periodo de oscilación interna (0.2 µs ; para oscilador de 20 MHz)
      - 255 es el pre-escalador del TIMER0
      - 256 la cantidad de pulsos requeridos para generar el desbordamiento del TIMER0, a partir del valor inicial establecido en la función set_timer0(0)

- Con la configuración de la función descrita y sustituyendo valores
  - **TT0IF**= 13.10 ms

- enable_interrupts(INT_RTCC); habilita la interrupción TIMER0

Declaración de la interrupción del TIMER 0:

```
#int_rtcc
clock_isr(){
        //código de la rutina
    }
```

Interrupción por cambio de nivel de los pines RB4 a RB7

Se puede generar la petición de interrupción cuando cualesquiera de los pines RB4, RB5, RB6 o RB7 cambia de nivel; ya sea de alto a bajo o de bajo a alto.

- Funciones para uso de PB4 al PB7

  - enable_interrupts(INT_RB); //Habilita interrupción por cambio de nivel en los cuatro bits más significativos del puerto B

Declaración de interrupción por cambio de nivel RB4-RB7

```
#int_rb
port_rb(){
    //código de la rutina de interrupción
      }
```

Interrupción por detección de flanco de RB0

Cuando ocurre un flanco ya sea de subida o de bajada previamente definida, el procesador podrá generar la petición de interrupción.

Funciones empleadas:

ext_int_edge(flanco); // Configura el flanco a detectar; donde flanco= H_TO_L o L_TO_H

Definición de la rutina de interrupción por detección de flanco de RB0

```
#int_ext
detecta_rb0(){
        //código de la rutina de interrupción
    }
```

Interrupción por recepción de datos del puerto serie asíncrono

Configuración de la comunicación asíncrona:

```
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
```

Definición de la rutina de interrupción por recepción de datos por el puerto serie asíncrono.

```
#int_rda
recepción_serie(){
                    //código de la rutina de interrupción
                }
```

Finalmente, el programa que empleará las interrupciones tendrá la siguiente forma:

```
#include <16f877.h>
#device adc=8 //en caso de emplear el conv. A/D indica resolución de 8 bits
...             //configuración general
…
            //declaración de variables
#int_rtcc    // rutina de interrupción del timer0
clock_isr(){
        //código de la rutina de interrupción
        }

main()
{
set_timer0(0); // Inicia TIMER0 en 00H
setup_counters(RTCC_INTERNAL,RTCC_DIV_256); //Fuente de reloj y pre-divisor
 enable_interrupts(INT_RTCC); //Habilita interrupción por TIMER0
enable_interrupts(GLOBAL); //Habilita interrupciones generales

  while(1){
                    // programa principal
            }
}
```

Desarrollo. Realizar los siguientes ejercicios.

1.- Escribir, comentar, indicar que hace; comprobar el funcionamiento del siguiente programa.

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

int contador;

#int_EXT
ext_int()
{
contador++;
output_d(contador);
}

void main() {
   ext_int_edge(L_TO_H);
   enable_interrupts(INT_EXT);
 enable_interrupts(GLOBAL);
 output_d(0x00);
  while( TRUE ) {}

}
```

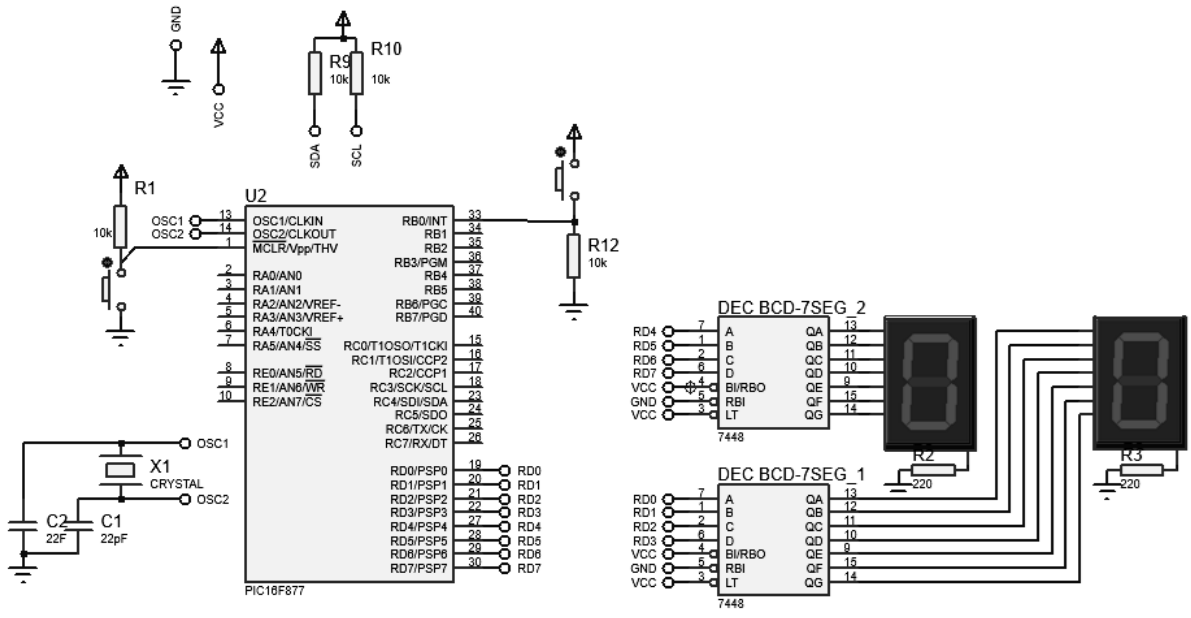

Figura 10.1 Circuito actividad 1

2.- Escribir un programa, el cual obtenga una señal analógica a través del canal de su elección; el resultado de la conversión deberá ser desplegado en tres diferentes dispositivos, de acuerdo a la tabla 9.1.

El resultado debe ser desplegado de acuerdo a:

Periférico	Formato del despliegue	Dispositivo	Formato del despliegue
Puerto paralelo	Decimal	Disp. 7SEG	Visible 00 - 99
I2C	Voltaje	LCD	Vin= 5.00 V
Puerto serie	Decimal, hexadecimal	Terminal	Decimal=1023, Hexadecimal=0x3FF

Tabla 10.1 Formatos de resultados y periféricos

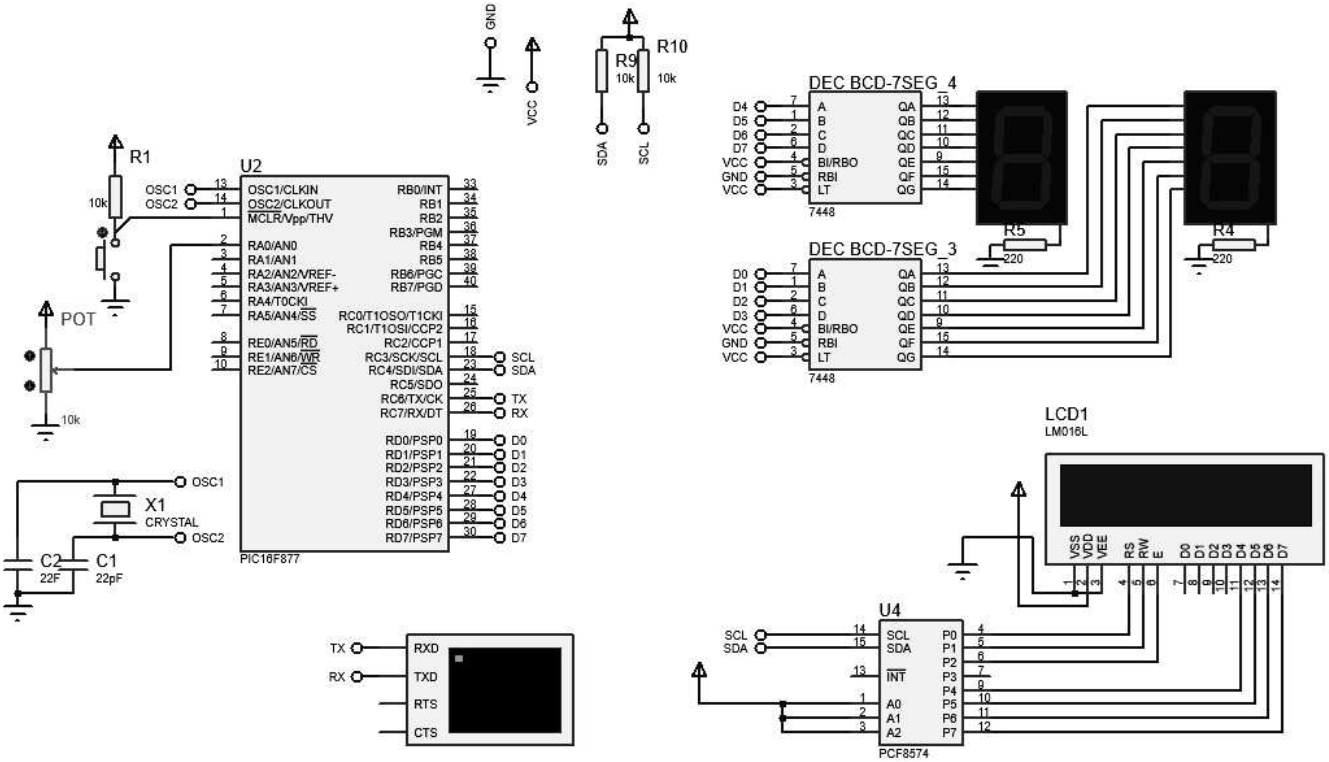

Figura 10.2 Circuito actividad 2; Entrada analógica, display 7 SEG, LCD I2C, Terminal.

3.- Utilizando la interrupción del TIMER0, realizar un programa que transmita el resultado de la conversión cada 10 segundos, usar el mismo formato del ejercicio anterior.

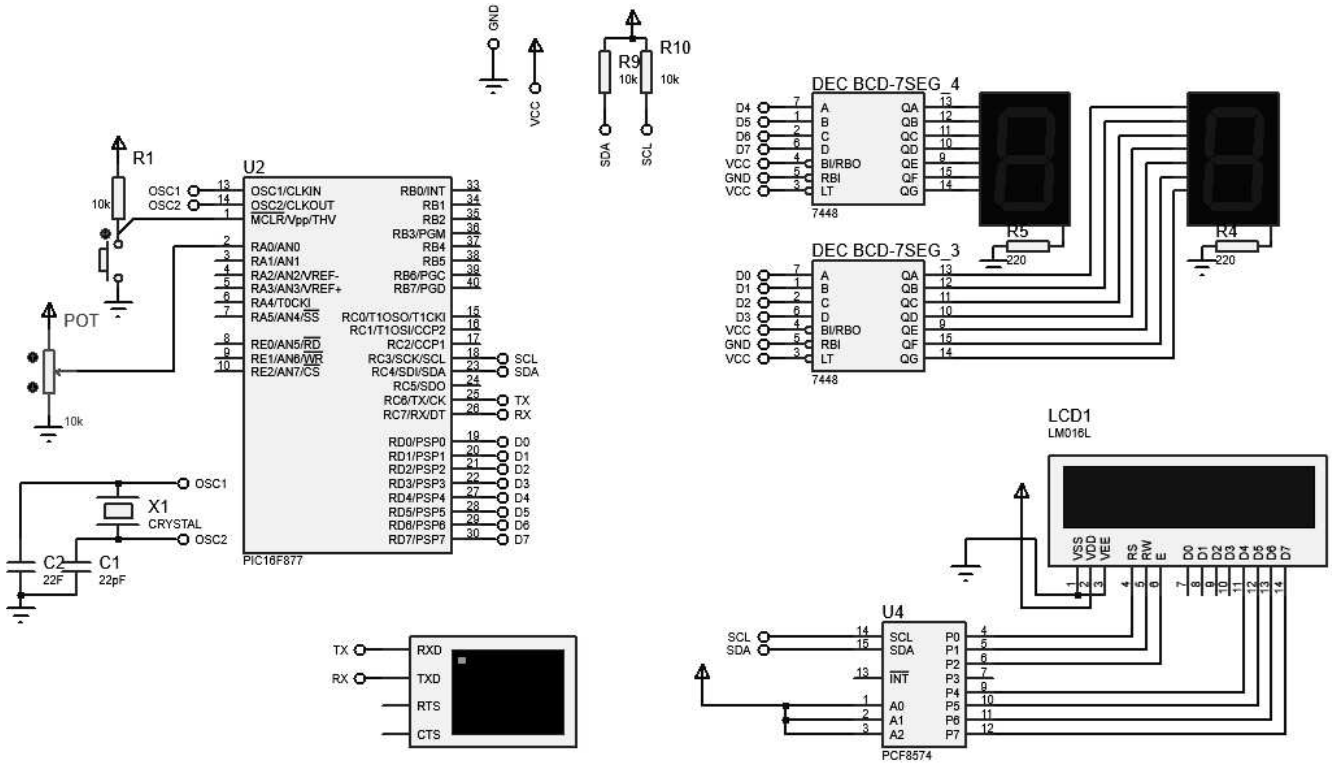

Figura 10.3 Circuito actividad 3.

4.- Realizar un programa que muestre un contador binario de 8 bits en un puerto paralelo (usar leds y retardos de 250 ms), cada 10 segundos muestre el voltaje de la señal que ingrese en el canal 0 del convertidor A/D en el LCD y cada 25 segundos despliegue en la terminal los nombres, número de cuenta, grupo de teoría y laboratorio del o los integrantes del equipo.

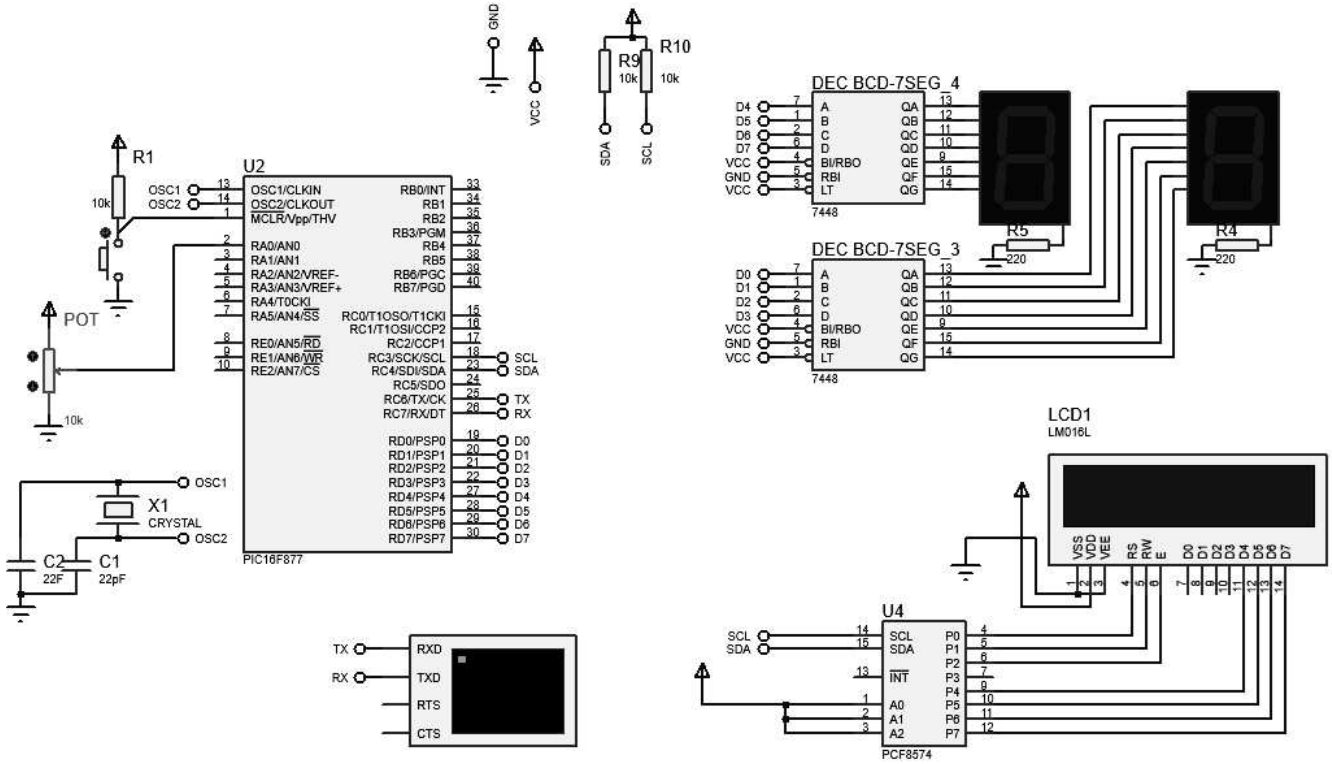

Figura 10.4 Circuito actividad 4

5.- Realizar un programa que permita atender las interrupciones indicadas en la tabla 9.2 y realice las acciones indicadas en esta, usando el dispositivo que considere para cada caso. El programa principal ejecutará un contador decimal ascendente y descendente, de 0 a 20 y 20 a 0 de manera indefinida con retardos de 1 segundo (usar display de 7 segmentos)

Interrupción	Acción	Periférico	Dispositivo
Ninguna	Contador	**Puerto D**	**Display 7 SEG**
RB0	Despliegue la cuenta de las veces que ha sido activada	**I2C**	**Display 7 SEG**
Recepción de datos del puerto serie	Cada que llegue un dato muestre un mensaje y las veces que ha ocurrido este evento	**USART**	**Terminal**
RB4 – RB7	Cuando alguna de los pines cambie de bajo a alto, indique en cual de ellos ha ocurrido	**USART**	**Terminal**
Desbordamiento TIMER0	Contador de ocho bits; cambio cada 200 ms	**I2C**	**LCD**

Tabla 10.2 Acciones actividad 5

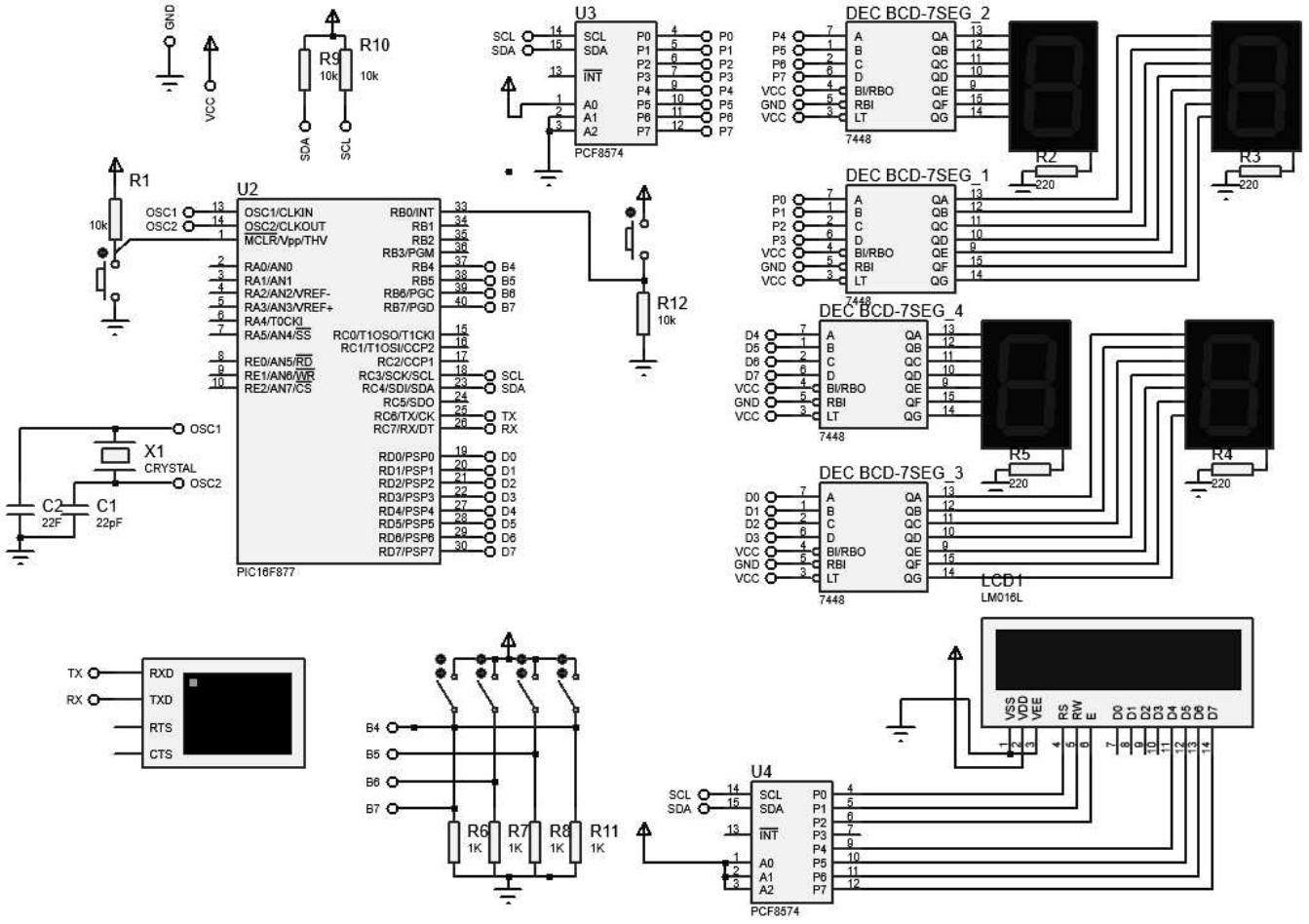

Figura 10.5 Circuito actividad No. 5