UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

LABORATORIO DE MICROCOMPUTADORAS

GRUPO 04

ROJAS MÉNDEZ GABRIEL

314141712

CARREÓN GUZMÁN MARIANA IVETTE

312103914

PRÁCTICA 10: PROGRAMACIÓN EN C

CONVERTIDOR A/D E INTERRUPCIONES.

SEMESTRE 2022-1

29/11/2021

Objetivos:

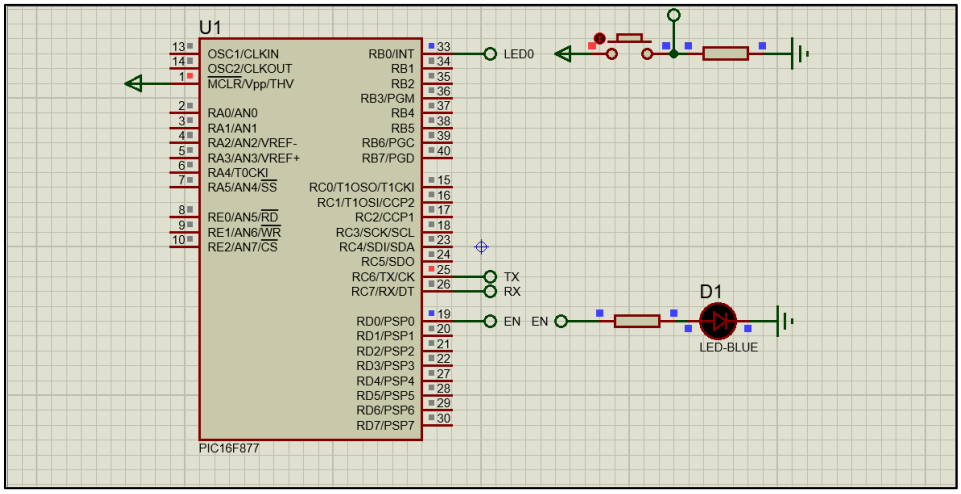
Realización de programas usando programación en lenguaje C, utilización del puerto serie, convertidor analógico digital e introducción a aplicaciones con interrupciones.

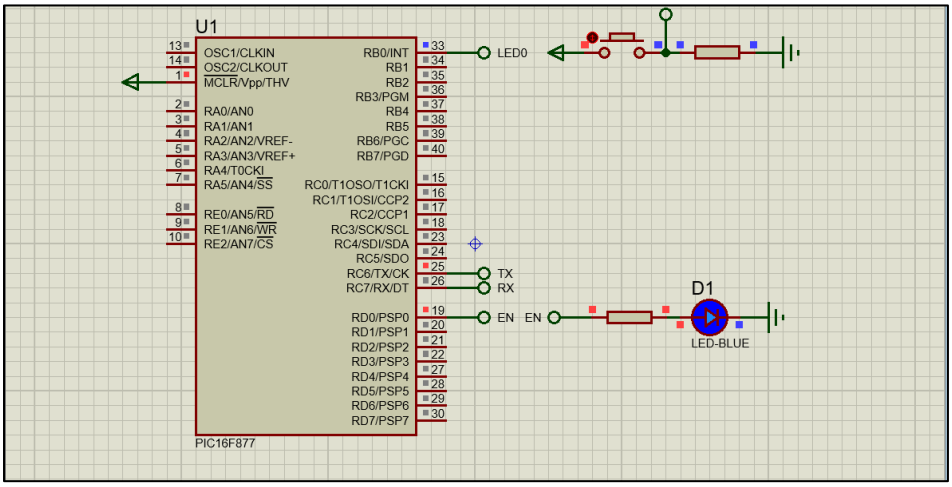
Desarrollo:

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

Ejercicio 1.c* | Ejercicio 2.c | Ejercicio 3.c | Ejercicio 4.c | Ejercicio 5.c

```
1  #INCLUDE <16F877.H>              //Librería del microcontrolador
2  #fuses HS,NOWDT,NOPROTECT
3  #use delay(clock=20000000)       //Frecuencia de oscilación
4  #use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)  //Configuración y habilitación
5  #org 0x1F00, 0x1FFF void loader16F877(void) {}  //de comunicación serial
6
7  #int_EXT                         //Rutina de interrupción por detección de
8  VOID ext_int(){                  //flanco de RB0
9     output_toggle(PIN_D0);        //Si el pin D0 está en 0 cambia a 1 y viceversa
10 }
11
12 void main() {
13    ext_int_edge(L_TO_H);         //Configura el flanco a detectar (subida)
14    enable_interrupts(INT_EXT);   //Habilita interrupción por detección de flanco de RB0.
15    enable_interrupts(GLOBAL);    //Habilita interrupciones generales
16    output_low(PIN_D0);           //Se garantiza que el PIN D0 inicie apagado
17    while( TRUE ) {}              //Ciclo infinito
18 }
```





Este ejercicio consistía en la habilitación de la interrupción por flanco de RB0, por lo tanto, cada que el puerto B0 se recibía un cambio de voltaje mediante el push button, en el puerto D0 se aplicaba un toggle el cual mediante cada interrupción cambiaba de 0 a 1 el estado del puerto o viceversa.

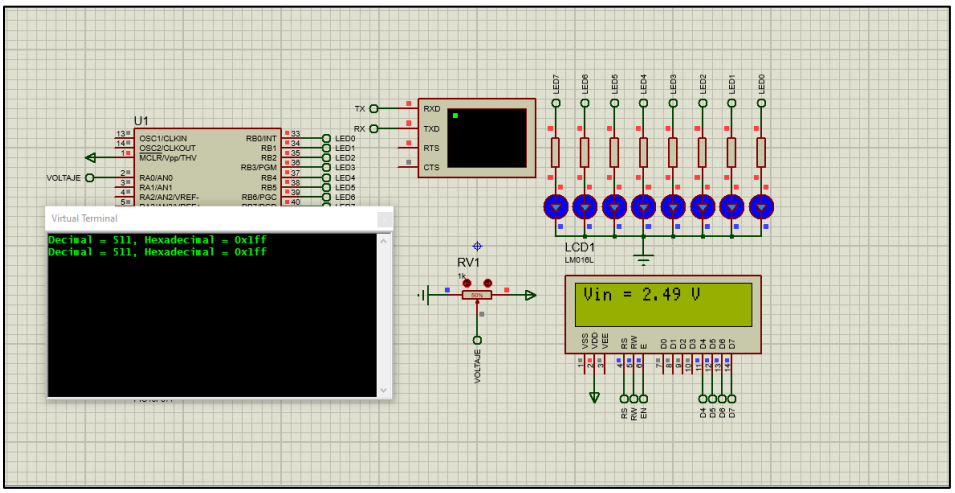
2.- Escribir un programa, el cual obtenga una señal analógica a través del canal de su elección; el resultado de la conversión deberá ser desplegado en tres diferentes dispositivos, de acuerdo con la tabla 10.1.

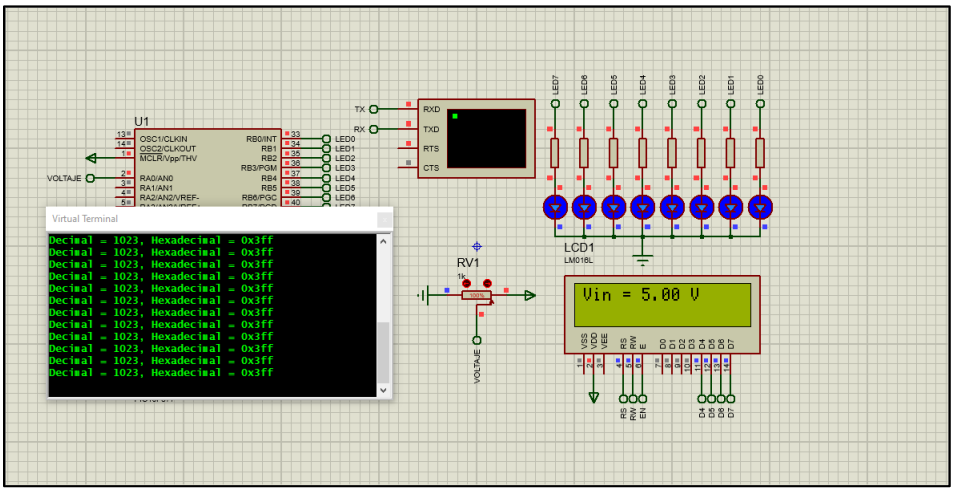
El resultado debe ser desplegado de acuerdo con:

Periférico	Formato del despliegue	Dispositivo	Formato del despliegue
Puerto paralelo	Binario	Leds	11111111
Puerto paralelo	Voltaje	LCD	Vin= 5.00 V
Puerto serie	Decimal, hexadecimal	Terminal	Decimal=1023, Hexadecimal=0x3FF

Ejercicio 1.c | Ejercicio 2.c | Ejercicio 3.c | Ejercicio 4.c | Ejercicio 5.c

```c
#include <16f877.h>              //Librería del microcontrolador
#fuses HS,NOPROTECT, NOWDT, NOLVP
#device ADC = 10                 //Convertidor A/D indica resolución de 10 bits
#use delay(clock=20000000)       //Frec. de Osc. 20Mhz
#define use_portd_lcd true
#include <lcd.c>
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7) //Configuracion del Puerto SERIAL
#org 0x1F00, 0x1FFF void loader16F877(void) {}

int16 var; // Variable tipo entero de 16 bits

void main(){
   lcd_init();                   //Inicialización del LCD
   setup_port_a(ALL_ANALOG);     //Configuración del Puerto A como analógico
   setup_adc(ADC_CLOCK_INTERNAL);//Relog interno de Conv. A/D
   set_adc_channel(0);           //Selecion del canal 0
   delay_us(20);
    while(TRUE){                 // Ciclo while infinito
        var = read_adc();        // Se asigna a var la lectura del Canal 0
        output_b(var);           // Se muestra  var en el puerto B
        printf("Decimal = %lu, Hexadecimal = 0x%3lx\n\r",var, var);
        lcd_gotoxy(1,1);
        printf(lcd_putc,"Vin = %2.2f V\n", var/204.60);
        delay_ms(1000);          // Retardo de 20us (microsegundos)
    }
}
```





Para poder desarrollar este ejercicio se siguió el siguiente algoritmo:

1) Definir la resolución del convertidor A/D, como el ejercicio solicitaba un formato de 1023 cuando hubiera 5 volts, entonces se debían definir 10 bits.
2) Definir al puerto A como analógico.
3) Definir la frecuencia de muestreo del convertidor A/D.
4) Configurar que canal usar.
5) Retardo de 20 microsegundos.
6) Leer el resultado de la conversión.
7) Para imprimir en el LCD el formato solicitado el resultado de la conversión habrá que dividirla entre 204.6 para que así el 1023 sea igual a 5 volts y los resultados de las conversiones sean equivalentes.

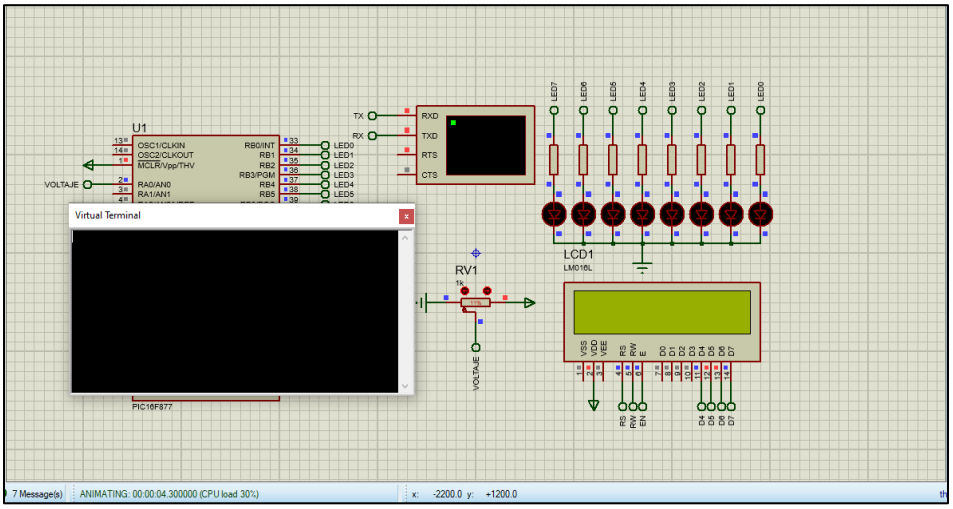
3.- Utilizando la interrupción del TIMER0, realizar un programa que transmita el resultado de la conversión cada 10 segundos, usar el mismo formato del ejercicio anterior.

Ejercicio 1.c | Ejercicio 2.c | Ejercicio 3.c | Ejercicio 4.c | Ejercicio 5.c

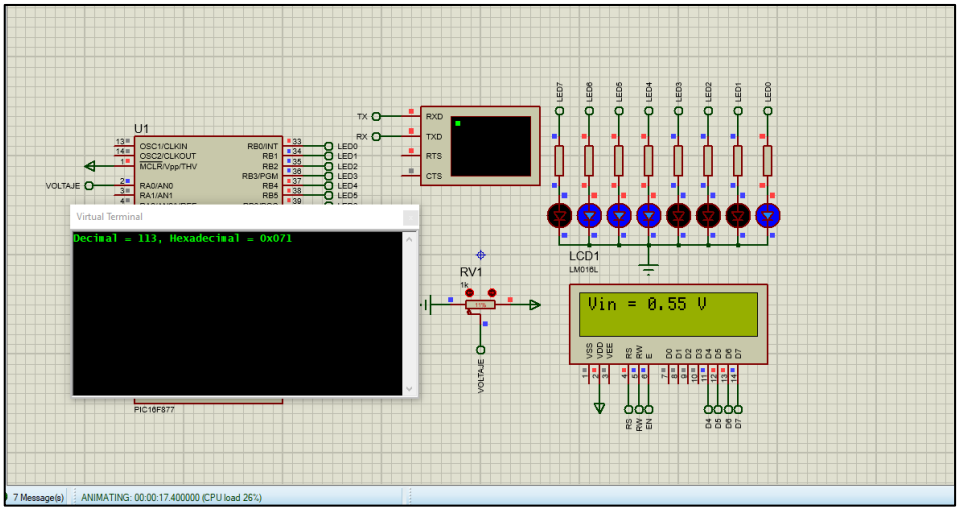
```
1  #include <16f877.h>                // Librería del microcontrolador
2  #device ADC=10                     // Convertidor A/D indica resolución de 10 bits
3  #fuses HS,NOPROTECT,NOWDT,NOLVP
4  #use delay(clock=20000000)         //Frec. de Osc. 20Mhz
5  #define use_portd_lcd true         //Se define al puerto D como enlace al LCD
6  #include <lcd.c>
7  #use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)//Configuración del puerto SERIAL
8  #org 0x1F00, 0x1FFF void loader16F877(void) {}
9
10 int16 var;                         // Variable tipo entero
11 int16 contador=0;
12
13 #int_rtcc                          //Desbordamiento del TIMER0
14 void clock_isr(){                  // Rutina de interrupcion por Timer0
15     contador++;                    // Incrementa contador
16     if(contador==763){             //763 * 10.13 us ~~ 10 s
17        output_b(var);              //Se muestra  resultado como salida
18        printf("Decimal = %lu, Hexadecimal = 0x%3lx\n\r",var, var);
19        lcd_gotoxy(1,1);
20        printf(lcd_putc,"Vin = %2.2f V", var/204.60);
```

Ejercicio 1.c | Ejercicio 2.c | Ejercicio 3.c | Ejercicio 4.c | Ejercicio 5.c

```
        delay_ms(1000);
        contador=0;                 // Inicializa contador, se reinicia
    }
}

void main(){
    lcd_init();
    setup_port_a(ALL_ANALOG);     //Configuración del Puerto A como analógico
    setup_adc(ADC_CLOCK_INTERNAL);  //Configuración del reloj interno ADC
    set_adc_channel(0);             //Configuración del Canal 0
    set_timer0(0);                  // Inicia el timer0 en 00H
    setup_counters(RTCC_INTERNAL,RTCC_DIV_256); //Fuente de reloj y predivisor
    enable_interrupts(INT_RTCC);    // Habilita la interrupción TIMER0
    enable_interrupts(GLOBAL);      // Habilita interrupciones generales

    while(TRUE){                    // Ciclo while infinito
        delay_us(20);               // Retardo de 20us (microsegundos)
        var = read_adc();           // resultado=valor de ADC
        delay_us(20);               // Retardo de 20us (microsegundos)
    }
}
```



Como se puede alcanzar en esta primera imagen, el reloj va en 4 segundos y aún no se ha desplegado nada.



Y para esta segunda imagen se puede notar que el reloj del simulador va en 17 segundos y ya hay información de la conversión en las distintas salidas establecidas.

Para la realización de este ejercicio se siguió el siguiente algoritmo:

1) Declarar la función INT_RTCC para el desbordamiento del TIMER0.
2) Incrementar un contador mientras se este ejecutando la función de desbordamiento.
3) Si contador == 763 (763 * 10.13 us = 9.99 s)
    a. Mostrar el resultado en el puerto B
    b. Mostar resultado en la terminal virtual.
    c. Mostar resultado en el LCD
4) Se configura el puerto A como analógico.
5) Configurar el reloj interno del A/D.
6) Configurar el canal de recepción de la señal análoga.
7) Se inicializa el TIMER0;
8) Se establece el reloj y el predivisor.
9) Habilitar la interrupción del TIMER0.

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.

Ejercicio 1.c | Ejercicio 2.c | Ejercicio 3.c | Ejercicio 4.c | Ejercicio 5.c

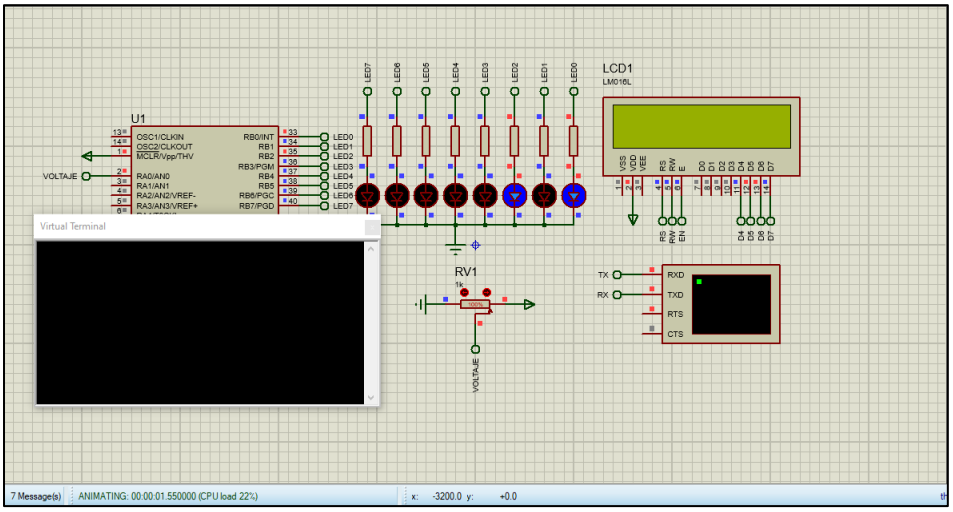
```
1  #include <16f877.h>                //Librería de microcomtrolador
2  #device ADC = 10                   //Resolución Conv A/D
3  #fuses HS,NOPROTECT,
4  #use delay(clock=20000000)         //Frec. de Osc. 20Mhz
5  #define use_portd_lcd true         //Se define el puerto D como enlace al LCD
6  #include <lcd.c>
7  #use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)   //Configuración Puerto SERIAL
8  #org 0x1F00, 0x1FFF void loader16F877(void) {}
9
10 int16 var;
11 int16 contador=0;
12 int16 contador1=0;
13 int16 contador2=0;
14 int aux = 0;
15
16 #int_rtcc
17 void clock_isr(){                  //Rutina de interrupción por TIMER0.
18    contador++;                     //Contadores que permitirán las interrupciones
19    contador1++;                    //independientes de cada uno de los casos
20    contador2++;                    //solicitados.
21
22    if(contador2==19){              //19 * 13us = 248 ms
23       output_b(aux);               //Se imprime el contador de 8 bits en puerto
24       aux++;                       //B
25       contador2 = 0;
26    }
27
28    if(contador1==763){             //763 * 10.13 us = 9.99 s
29       lcd_gotoxy(1,1);             //Se imprime la conversión en el LCD
30       printf(lcd_putc,"Vin = %2.2f V", var/204.60);//1023 / 204.0 = 5
31       contador1 = 0;
32    }
```

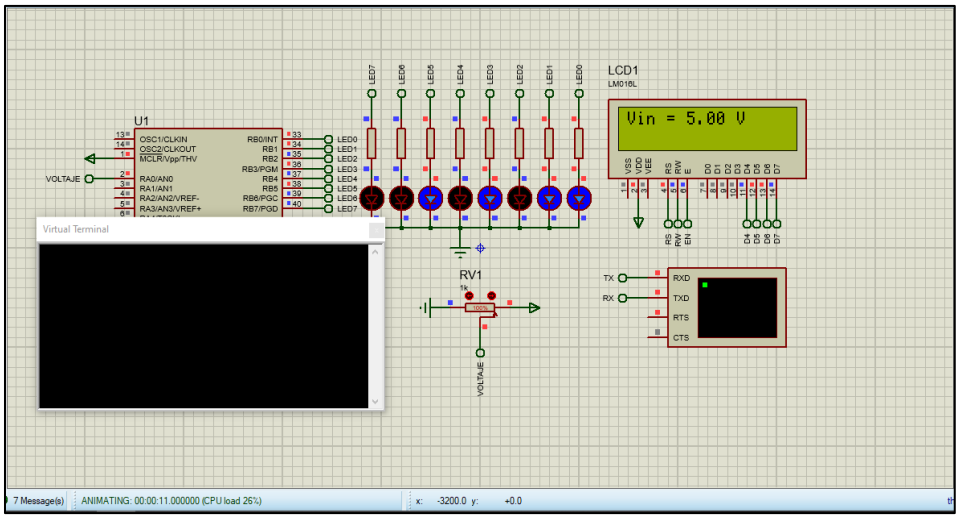
Ejercicio 1.c | Ejercicio 2.c | Ejercicio 3.c | Ejercicio 4.c | Ejercicio 5.c

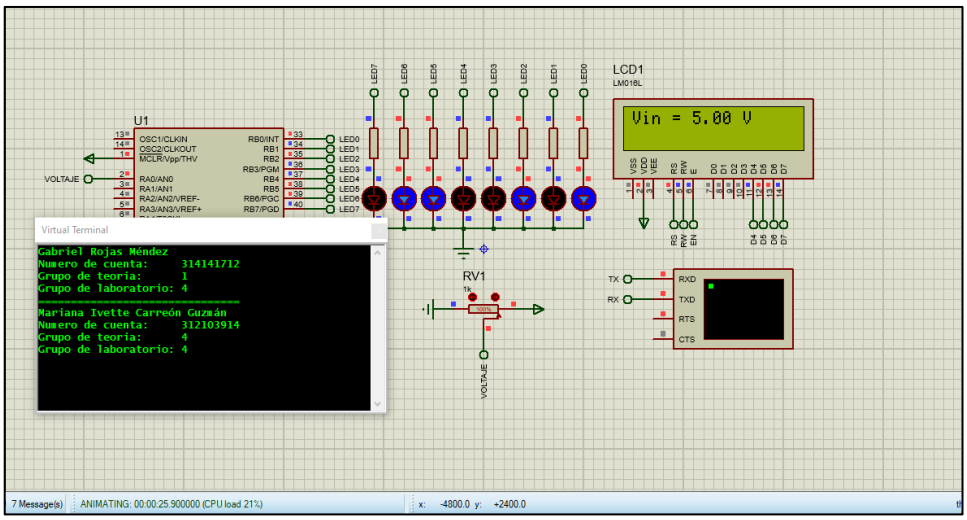
```
34    if(contador==1908){             //1908 * 13.10 us = 24.99 s
35      printf("Gabriel Rojas Méndez\n\r");
36      printf("Numero de cuenta:     314141712\n\r");
37      printf("Grupo de teoria:      1\n\r");
38      printf("Grupo de laboratorio: 4\n\r");
39      printf("==============================\n\r");
40      printf("Mariana Ivette Carreón Guzmán\n\r");
41      printf("Numero de cuenta:     312103914\n\r");
42      printf("Grupo de teoria:      4\n\r");
43      printf("Grupo de laboratorio: 4\n\r");
44      contador=0;
45    }
46 }
```

Ejercicio 1.c | Ejercicio 2.c | Ejercicio 3.c | Ejercicio 4.c | Ejercicio 5.c

```
48  void main(){
49     lcd_init();
50     setup_port_a(ALL_ANALOG);              //Configura PORTA como analógico
51     setup_adc(ADC_CLOCK_INTERNAL);         //Relog interno de Conv. A/D
52     set_adc_channel(0);                    //Selección del Canal 0
53     set_timer0(0);                         //Inicia el timer0 en 00H
54     setup_counters(RTCC_INTERNAL,RTCC_DIV_256);
55     enable_interrupts(INT_RTCC);           //Habilita la interrupción TIMER0
56     enable_interrupts(GLOBAL);             //Habilita interrupciones generales
57
58     while(TRUE){
59        delay_us(20);                       //Retardo
60        var=read_adc();                     //Lee el resultado de la Conversión
61        delay_us(20);                       //Retardo
62      }
63  }
```







En las ilustraciones anteriores se puede ver como la primera imagen tiene un tiempo en el simulador de 1 segundo y sólo están encendidos los leds que forman parte del contador de 8 bits solicitado. La segunda imagen tiene un tiempo de 11 segundos y ya se encuentra desplegada la información de la conversión del A/D en el LCD, y finalmente, la tercera imagen tiene un tiempo de 25 segundos y en la terminal se alcanza a distinguir que se han desplegado los datos solicitados. Para el desarrollo de este ejercicio se empleó el algoritmo anterior solo que se modificaron las estructuras condicionales para poder hacer los despliegues de información en los tiempos dados.

1) Declarar la función INT_RTCC para el desbordamiento del TIMER0.
2) Incrementar tres contadores mientras se esté ejecutando la función de desbordamiento para permitir la ejecución independiente de cada interrupción.
3) Si contador2 == 19 (19 * 10.13 us = 248 ms)
    a. Incrementar aux en 1.
    b. Mostrar aux en el puerto B.
4) Si contador1 == 763 (763 * 13.10 us = 9.99 s)
    a. Mostar resultado en el LCD.
5) Si contador == 1908 (1908 * 13.10 us = 24.99 s)
    a. Desplegar datos de los integrantes en la terminal.

5.- Realizar un programa que permita atender las interrupciones indicadas en la tabla 10.2 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 de 20 a 0 de manera indefinida con retardos de un segundo (usar display de 7 segmentos).

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

Ejercicio 5.c | i2c_Flex_LCD.c

```
1  #include <16F877.h>                                  //Librería de microcomtrolad
2  #fuses HS,NOPROTECT,NOWDT
3  #use delay(clock=20000000)                           //Frec. de Osc. 20Mhz
4  #use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)       //Configuración Puerto SERIA
5  #org 0x1F00, 0x1FFF void loader16f877(void) {}
6  #use i2c(MASTER, SDA=PIN_C4, SCL=PIN_C3,SLOW, NOFORCE_SW)
7  #include <i2c_Flex_LCD.c>
8
9  byte const display[21]={0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,
10 0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x20};
11
12 int aux = 0, bandera = 0;
13 int contador = 0, conteo = 0;    //Variables TIMER0
14 int cambiob0 = 0, cambiob4 = 0, cambiob5 = 0, cambiob6 = 0,cambiob7 = 0;
15 char dato;
16
17 void escribir_i2c(){
18    i2c_start();
19    i2c_write(0x42);
20    i2c_write(conteo);
21    i2c_stop();
22 }
23
24 #int_rtcc
25 void clock_isr(){
26    contador++;
27    if(contador == 15){
28       escribir_i2c();
29       conteo++;
30       contador = 0;
31    }
32 }
```

Ejercicio 5.c | i2c_Flex_LCD.c

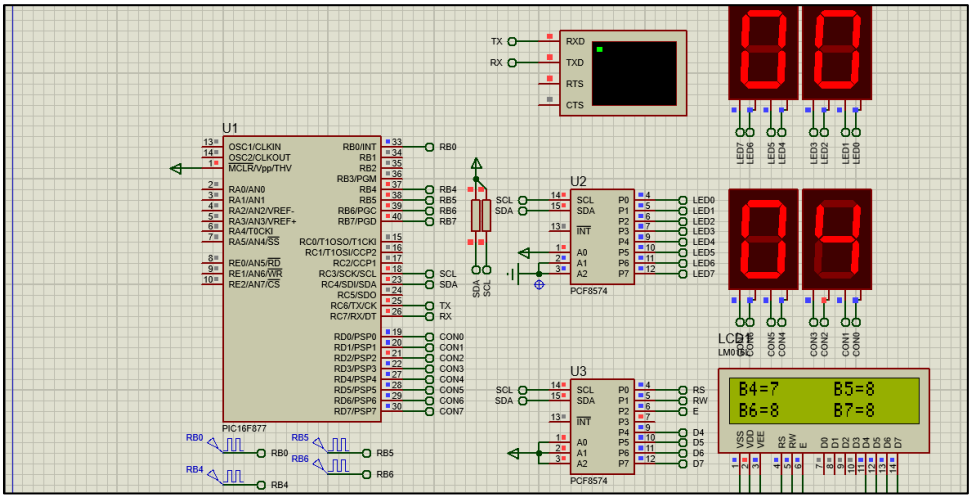
```
#int_rda
void recepcion_serie(){
   dato = getchar();
   printf("\n\r");
   printf("Se recibio el dato: %c\n\r", dato);
   }

#int_rb
void cambio_bits(){
   if(input(Pin_B4) == 1){
         cambiob4++;
         lcd_gotoxy(1,1);
         printf(LCD_PUTC, "B4=%d\n", cambiob4);
      }
   if(input(Pin_B5) == 1){
         cambiob5++;
         lcd_gotoxy(10,1);
         printf(LCD_PUTC, "B5=%d\n", cambiob5);
      }
   if(input(Pin_B6) == 1){
         cambiob6++;
         lcd_gotoxy(1,2);
         printf(LCD_PUTC, "B6=%d\n", cambiob6);
      }
   if(input(Pin_B7) == 1){
         cambiob7++;
         lcd_gotoxy(10,2);
         printf(LCD_PUTC, "B7=%d\n", cambiob7);
      }
}
```

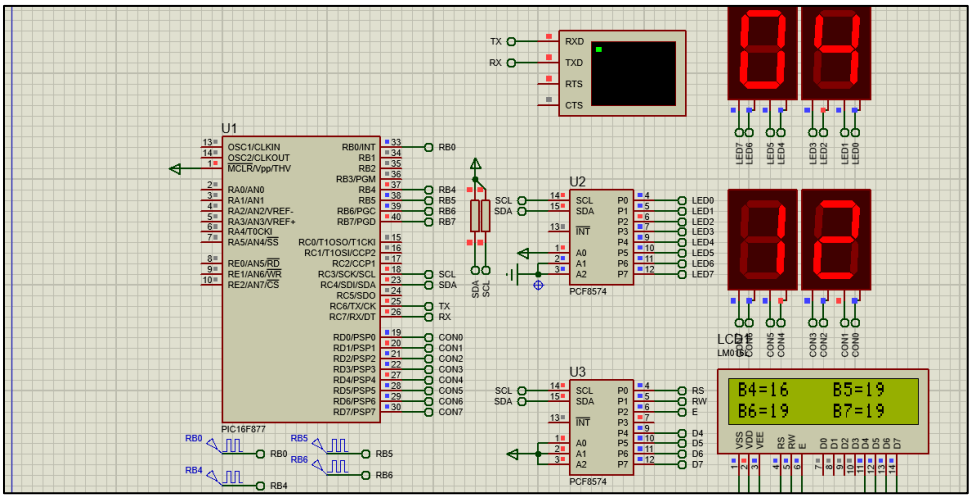
Ejercicio 5.c | i2c_Flex_LCD.c

```
#int_ext
void state_b0(){
      if(input(Pin_B0) == 1){
         cambiob0++;
         printf("El pin B0 ha cambiado %d\n\r", cambiob0);
      }
}

void main(){
  set_timer0(0);
  setup_counters(RTCC_INTERNAL,RTCC_DIV_256);
  enable_interrupts(INT_RTCC);
  enable_interrupts(INT_RDA);
  enable_interrupts(INT_RB);
  enable_interrupts(INT_EXT);
  enable_interrupts(GLOBAL);
  lcd_init(0x4E,16,2);          //Inicializa el LCD y espeficia su dirección
  lcd_backlight_led(ON);        //Configura la iluminación de fondo.

   while(TRUE){
      if(aux <= 19 && bandera == 0){
         output_d(display[aux]);
         aux++;
         delay_ms(1000);
      }
      else{
         bandera = 1;
      }
      if(aux != 0 && bandera == 1){
         output_d(display[aux]);
         aux--;
```

Ejercicio 5.c | i2c_Flex_LCD.c

```
96              delay_ms(1000);
97          }
98          else{
99              bandera = 0;
100         }
101     }
102 }
103
```





Para el desarrollo de este ejercicio, no hubo tanto problema, sólo fue cuestión de planificar la forma en la que actuaría cada uno de los elementos para así mostrar cada una de las interrupciones implementadas. El algoritmo que se siguió de forma muy general fue el siguiente:

Algoritmo:

1. Definir un arreglo con los números del uno al 20 para ser mostrados en la función principal del programa.
2. Declarar cada una de las 4 funciones establecidas en los requerimientos.
    a. Para la interrupción de desbordamiento del TIMER0: si contador = 16 entonces llamar a función escribir para poder transmitir los datos al display 7 segmentos con comunicación I2C.
    b. Para interrupción por recepción de datos por el puerto serie:
        i. Capturar dato.
        ii. Regresar dato.
    c. Para interrupción por detección de flanco RB0.
        i. Si la entrada en pin B0 = a 1 entonces aumentar cambio.
        ii. Imprimir en terminal virtual cambio.
    d. Para interrupción por cambio de nivel en pines del B4 al B7:
        i. Si entrada en pin B4 = 1 aumentar variable cambiob4
        ii. Imprimir en el LCD.
        iii. Si entrada en pin B5 = 1 aumentar variable cambiob5
        iv. Imprimir en el LCD.
        v. Si entrada en pin B6 = 1 aumentar variable cambiob6
        vi. Imprimir en el LCD.
        vii. Si entrada en pin B7 = 1 aumentar variable cambiob7
        viii. Imprimir en el LCD.
3. Para cuerpo principal del programa.
    a. Si auxiliar <= 19 y bandera = 0 entonces
        i. Mandar a puerto B
        ii. Aumentar el auxiliar
    b. Sino
        i. Bandera = 1
    c. Si auxiliar ¡= 0 y bandera = 1

i. Mandar a puerto B.
ii. Decrementar el auxiliar.

d. Sino
i. Bandera = 0.

Conclusiones:

Carreón Guzmán Mariana:

A lo largo de esta práctica pude seguir programando en el lenguaje C, aprendí cómo es que se manejan los puertos los puertos serie, así como la forma en la que puedo ir realizando interrupciones dentro de mi código, aunque esto se había mencionado en prácticas pasadas pude terminar de comprender este concepto. Por otra parte, el manejo del convertidor A/D es algo que se me complica, pero pude ver cómo es que se debía implementar al momento de analizar el algoritmo y así poder resolver los ejercicios planteados.

Rojas Méndez Gabriel:

Con la elaboración de esta práctica pude ver lo que son las interrupciones y la manera tan distinta como actúan en lenguaje C, además, la forma de trabajar con el convertidor A/D. Aunque el último ejercicio fue un verdadero reto debido a que para su desarrollo se emplearon varios conceptos vistos en las prácticas anteriores, y es que para mostrar los resultados solicitados se tuvo que emplear la comunicación I2C y el expansor de puertos, pero de esta manera se pudieron poner en práctica los conocimientos adquiridos.