Universidad Nacional Autónoma de México

Facultad de Ingeniería

Semestre 2023-2

Laboratorio de Microcomputadoras

Práctica 10:

Programación en C

Convertidor A/D, Temporizadores e Interrupciones

Profesor:

M.I. Ruben Anaya García

Alumnos:	Número de cuenta:
Barreiro Valdez Alejandro	317520888
Zepeda Baeza Jessica	317520747

Grupo Laboratorio: 4

Grupo Teoría: 1

Fecha de realización: 24 de mayo de 2023

Fecha de entrega: 30 de mayo de 2023

Desarrollo

En esta práctica se realizan programas en lenguaje C, utilizando el Convertidor Analógico/Digital del PIC16F877 e implementando interrupciones que provocan una cierta salida. En caso del Convertidor A/C se aprendieron las instrucciones en C que: definen la resolución del convertidor (8 o 10), definen el o los puertos que se quieren manejar como analógicos , definen la frecuencia de muestreo del convertidor, configura el canal a usar y obtienen el resultado de la conversión.

En el caso de las interrupciones, primero se definió el concepto y se enumeraron las 14 fuentes de interrupción que el PIC16F877A tiene. Se vio que para generar interrupciones se deben de habilitar las interrupciones generales, las particulares y definir las rutinas de interrupción de aquellas que se habilitarán. Después se aprendieron las instrucciones en C que realizan estas acciones y las directivas de algunas fuentes de interrupción como TIMER0, cambio de nivel de RB4 a RB7, detección de flanco de RB0 y recepción de datos por puerto serie. En el caso del TIMER0, también se aprendió la forma de configurar el predivisor y la fuente de reloj a utilizar. Básicamente las interrupciones se implementan como bloques de código en forma de funciones que se realizan al suceder la interrupción.

Los programas o ejercicios desarrollados fueron:

1. Comentar y probar un programa ejemplo que implementa interrupciones por detección de flanco de RB0.
2. Un programa que recibe un dato analógico y su resultado se despliega en diferentes formatos y dispositivos, como muestra la siguiente tabla.

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

3. Agregar al programa anterior interrupciones utilizando el TIMER0 para mostrar la conversión cada 10 segundos.
4. Un programa que cada 250 segundos muestra un contador binario en LEDs, cada 10 segundos muestra en el LCD el voltaje ingresado por un potenciómetro en el canal 0 del Convertidor A/D y cada 25 segundos muestra en terminal los nombres de los integrantes del equipo, su grupo de teoría, de laboratorio y número de cuenta.
5. Un programa que atiende interrupciones y realiza una cierta acción en un cierto dispositivo, según la interrupción generada, como muestra la siguiente tabla. El programa principal ejecuta un contador ascendente y descendente de 0 a 20 y 20 a 0 con retardos de 1 segundo.

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

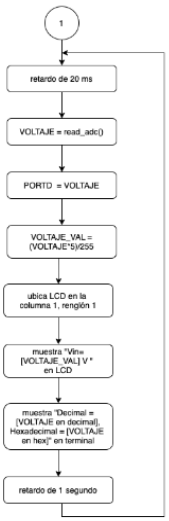
Algoritmos

Configuraciones iniciales

Para los programas realizados se incluyeron distintas directivas e instrucciones iniciales para configurar el microcontrolador. Para empezar, se importa la biblioteca que maneja el PIC16F877, se define el oscilador y el reloj interno, se configura la recepción y transmisión de datos por el puerto serie, se configura el convertidor A/D, la comunicación i2c, se importa la biblioteca que maneja el LCD por i2c y se inicializa y por último, se definen las localidades donde no se puede escribir código. Esto se muestra en la siguiente imagen:

Ejercicio 2

Para el ejercicio 2 se realizó la siguiente carta ASM:

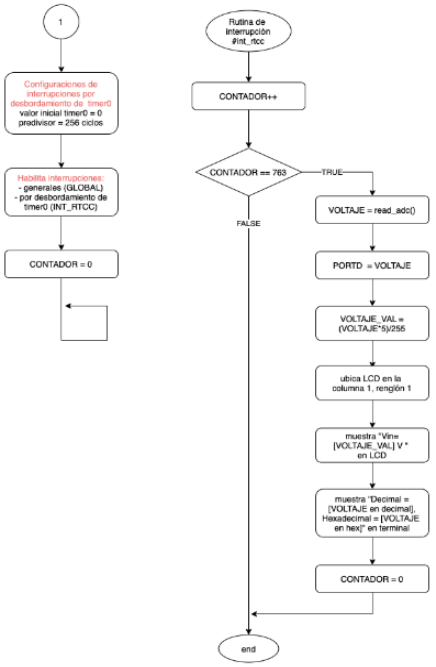


Donde se genera un retardo de 20 ms para permitir una correcta conversión, luego se guarda el valor de la conversión en una variable VOLTAJE y ese mismo valor se muestra en

el Puerto D (que contiene un BCD de 7 segmentos). También se calcula el voltaje que ese valor representa mediante una regla de 3 y se almacena en la variable VOLTAJE_VAL. Para mostrar este valor en el LCD, primero se ubica en la columna 1, renglón 1 y luego se manda la cadena y dato a mostrar. Lo último es mandar el valor de VOLTAJE a la terminal como número decimal y hexadecimal, lo cual se logra cambiando el formato del número al llamar la instrucción: %d y %x respectivamente. Después se hace un retardo de 1 segundo y se vuelve a realizar el programa desde el retardo de 20 ms.

Ejercicio 3

Para el ejercicio 3 se realizó la siguiente carta ASM:

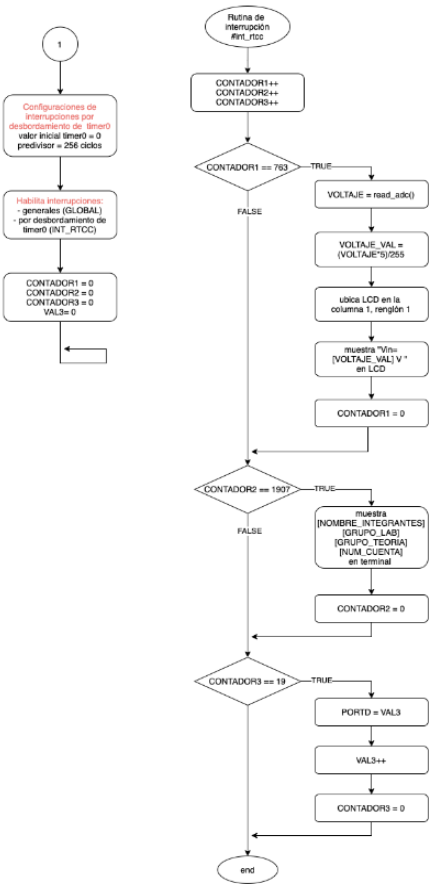


Se implementó el código anterior pero ahora solo mostrando los valores cada 10 segundos utilizando la interrupción por desbordamiento de TIMER0. En la función *main* se agregaron las configuraciones de la interrupción como el valor del predivisor y la tiempo inicial del TIEMR0. Debido a que se escogió un predivisor de 256 y un tiempo inicial de 0, la interrupción se genera cada 0.013 segundos. También se habilitaron las interrupciones generales y la particular, en este caso INT_RTCC. Luego se inicializa un contador en 0 y se genera un loop infinito que no hace nada hasta que sucede la interrupción.

Dentro de la rutina de interrupción del TIMER0 se incrementa el contador y en caso de que su valor sea 763 (han pasado 10 segundos) hace el bloque de código del ejercicio anterior para mostrar la conversión y regresa el valor de contador a 0 para volver a contar 10 segundos.

Ejercicio 4

Para el ejercicio 4 se realizó la siguiente carta ASM:



El programa principal se mantiene igual con las configuraciones de la interrupción por desbordamiento de TIMER0; pero en este caso se inicializan 3 contadores y una variable VAL3 en 0. Los contadores llevaran las cuentas de distintos tiempos. De igual forma se hace un loop infinito donde no sucede nada hasta que se activa la interrupción.

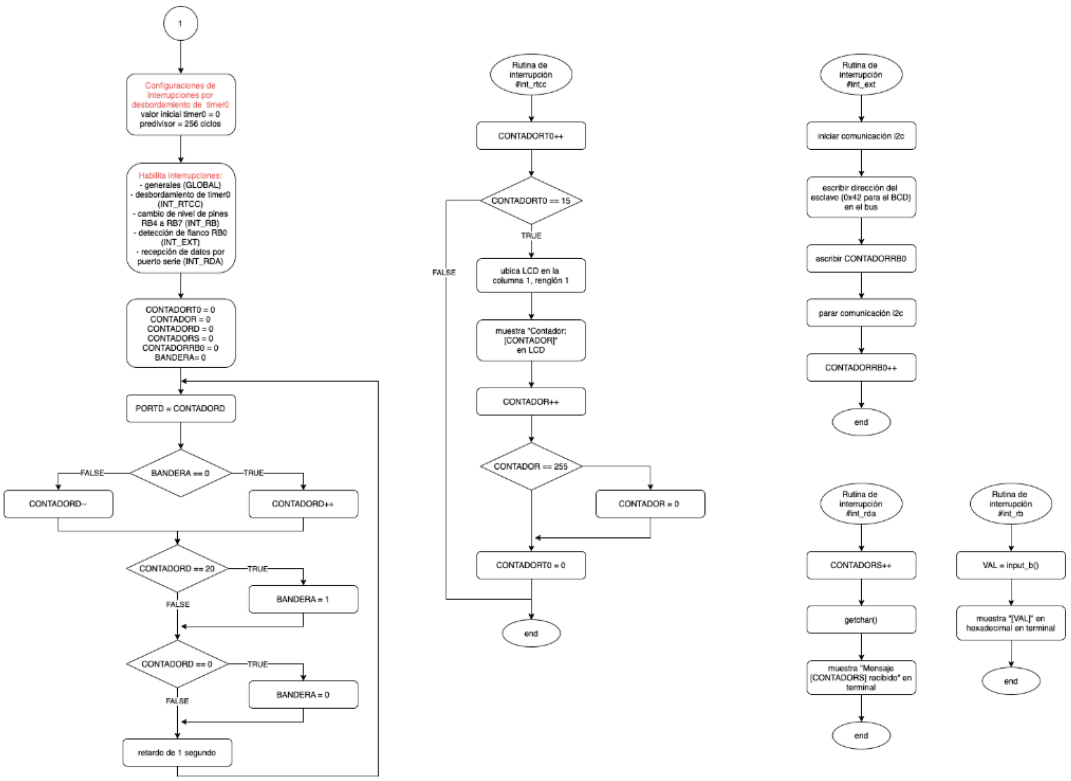
En la rutina de interrupción se incrementan en 1 los tres contadores y luego se tienen 3 *if* que verifican el valor de cada contador. En el primer *if* se verifica si CONTADOR1 ya llegó a 763 (10 segundos) y en caso de ser así hace la conversión del voltaje, la regla de 3 y muestra el valor del voltaje en el LCD (como en los ejercicios pasados). Al final vuelve a hacer CONTADOR1 igual a 0 para repetir la interrupción en 10 segundos.

El segundo *if* verifica si CONTADOR2 es 1907 (25 segundos) y en caso de serlo imprime en terminal los nombres de los integrantes del equipo, números de cuenta, grupo de teoría y de laboratorio. Al final le asigna un 0 a CONTADOR2 para volver a contar 25 segundos.

El segundo *if* verifica si CONTADOR3 es 19 (250 milisegundos) y en caso de serlo se pasa el valor de VAL3 al Puerto D (donde hay un BCD de 7 segmentos), se incrementa VAL3 y CONTADOR3 se vuelve a hacer 0 para contar otros 250 milisegundos.

Ejercicio 5

Para el último ejercicio se realizó la siguiente carta ASM:



Para empezar en el programa principal se agregan instrucciones para habilitar las interrupciones: por desbordamiento de TIMER0 (INT_RTCC), detección de flanco de RB0 (INT_EXT), recepción de datos del puerto serie (INT_RDA) y cambio de nivel en los pines de RB4 a RB7 (INT_RB). Después se declaran distintos contadores:

- CONTADORT0 (para contar los 200 ms utilizando la interrupción del TIMER0)
- CONTADOR (contador de 8 bits que incrementa cada 200 ms)
- CONTADORD (contador decimal que va de 0 a 20 y luego de 20 a 0)
- CONTADORS (lleva la cuenta de mensajes recibidos por el puerto serie)
- CONTADORRB0 (contador de las veces que RB0 ha pasado de bajo a alto)

También se tiene una variable VAL que almacenará el valor del Puerto B cuando RB4, 5, 6 o 7 cambien de valor. Y una variable BANDERA para indicar si el contador de 0 a 20 va en orden ascendente o descendente.

El programa principal muestra el CONTADORD en el Puerto D (BCD de 7 segmentos) y luego dependiendo el valor de BANDERA, aumenta (valor 0) o decrementa (valor 1) CONTADORD. Luego verifica si CONTADORD ya llegó a 20 para cambiar el valor de BANDERA a 1 (cuenta descendente) o si ya llegó a 0 para cambiar el valor de BANDERA a 0 (cuenta ascendente). Por último se hace un retardo de 1 segundo y regresa el flujo a mostrar el valor de CONTADORD en el Puerto D. Esto se realiza infinitamente y en caso de

suceder alguna interrupción se realizan sus respectivas rutinas y sigue el programa principal.

La rutina de interrupción por desbordamiento de TIMER0 comienza incrementando en 1 el CONTADORT0, luego verifica si su valor es 15 (200 ms) y en caso de serlo imprime en el renglón 1, columna 1 del LCD: “Contador: “ y el valor de CONTADOR. Luego incrementa CONTADOR y verifica su valor. En caso de ser 256, CONTADOR regresa a ser 0; de forma que se genera un contador de 8 bits que va de 0 a 255. Por último, CONTADORT0 también vuelve a ser 0 para contar otra vez 200 ms.

La rutina de interrupción por detección de flanco RB0 inicia la comunicación i2c, pasa la dirección del BCD de 7 segmentos conectado a i2c, le pasa el dato de CONTADORRB0 y para la comunicación i2c. Por último incrementa CONTADORRB0.

La rutina de interrupción por cambio de nivel de RB4 a RB7 obtiene el valor del Puerto B en la variable VAL e imprime los 4 bits más significativos en terminal.

La rutina de interrupción por recepción de datos del puerto serie incrementa CONTADORS en 1 e imprime en terminal “Mensaje # recibido” donde # es el valor de CONTADORS. También obtiene el caracter enviado mediante *getchar()* para limpiar el buffer y permitir más recepciones.

Programas comentados

Ejercicio 1

```
1   #include <16F877.h>
2   #fuses HS,NOWDT,NOPROTECT
3   #use delay(clock=20000000)
4   #use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
5
6   int contador;
7
8   #int_EXT                        //Interrupción en RB0
9   ext_int(){
10      contador++;                 //Se incrementa el contador
11      output_d(contador);         //Salida del contador en puerto D
12  }
13  void main() {
14      ext_int_edge(L_TO_H);       //Configura flanco a detectar
15      enable_interrupts(INT_EXT); //Habilitando interrupción en RB0
16      enable_interrupts(GLOBAL);  //Habilitando interrupciones globales
17      output_d(0x00);             //Salida de 0 en el puerto D
18      while( TRUE ) {}            //Loop infinito
19  }
```

Ejercicio 2

```c
#include <16F877.h>
#device adc=10                                          //Convertidor a 10 bits
#fuses HS,NOWDT,NOPROTECT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#use i2c(MASTER,SDA=PIN_C4,SCL=PIN_C3,SLOW,NOFORCE_SW)  //Se utiliza I2C LCD
#include <i2c_LCD.c>
#org 0x1F00,0x1FFF void loader16F877(void){}

int voltaje;                                            //Valor de voltaje leído
float voltaje_val;                                      //Voltaje convertido a V

void main() {
   Setup_port_a(ALL_ANALOG);                            //Puerto A analógico
   Setup_adc(ADC_CLOCK_INTERNAL);                       //Define frecuencia de muestreo
   Set_adc_channel(0);                                  //Se utiliza el canal 0

   lcd_init(0x4E, 16, 2);                               //Se inicializa LCD en dirección 4E

   while( TRUE ) {
      delay_us(20);                                     //Retardo de 20us
      voltaje = read_adc();                             //Se lee voltaje de convertidor

      output_d(voltaje);                                //Muestra voltaje en puerto D

      voltaje_val = (voltaje*5.0)/255.0;                //Conversión a V

      lcd_gotoxy(1,1);
      printf(lcd_putc,"Vin = %f V",voltaje_val);        //Muestra voltaje en V en LCD

      printf("Decimal = %u, Hexadecimal = %x \n",voltaje,voltaje);   //Valor decimal y hexadecimal en terminal
      delay_ms(1000);                                   //Retardo de 1 segundo
      }
}
```

Ejercicio 3

C P10-EJ3.C

```c
#include <16F877.h>
#device adc=10                                          //Convertidor a 10 bits
#fuses HS,NOWDT,NOPROTECT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#use i2c(MASTER,SDA=PIN_C4,SCL=PIN_C3,SLOW,NOFORCE_SW)
#include <i2c_LCD.c>
#org 0x1F00,0x1FFF void loader16F877(void){}

int voltaje;                                            //Valor de voltaje del convertidor
long contador;                                          //Valor de contador para la interrupción
float voltaje_val;                                      //Valor calculado de V

#int_rtcc                                               //Interrupción por Timer0
clock_isr(){
   contador++;                                          //Incrementar el contador

   if(contador == 763){                                 //Realizar cada segundo
      voltaje = read_adc();                             //Se lee voltaje de convertidor

      output_d(voltaje);                                //Muestra voltaje en puerto D

      voltaje_val = (voltaje*5.0)/255.0;                //Conversión a V

      lcd_gotoxy(1,1);
      printf(lcd_putc,"Vin = %f V",voltaje_val);        //Muestra voltaje en V en LCD

      printf("Decimal = %u, Hexadecimal = %x \n",voltaje,voltaje);   //Valor decimal y hexadecimal en terminal

      contador = 0;
   }
}

void main() {
   set_timer0(0);                                       //Inicia Timer en 00h
   setup_counters(RTCC_INTERNAL,RTCC_DIV_256);          //Fuente de reloj y pre-divisor
   enable_interrupts(INT_RTCC);                         //Habilita interrupción de Timer0
   enable_interrupts(GLOBAL);                           //Habilita interrupciones globales

   Setup_port_a(ALL_ANALOG);                            //Puerto A analógico
   Setup_adc(ADC_CLOCK_INTERNAL);                       //Define frecuencia de muestreo
   Set_adc_channel(0);                                  //Se utiliza el canal 0

   lcd_init(0x4E, 16, 2);                               //Se inicializa LCD en dirección 4E
   contador = 0;                                        //Inicializa contador en 0

   while( TRUE ) {}                                     //Loop infinito
}
```

Ejercicio 4

```c
#include <16F877.h>
#device adc=10                                                  //Convertidor a 10 bits
#fuses HS,NOWDT,NOPROTECT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#use i2c(MASTER,SDA=PIN_C4,SCL=PIN_C3,SLOW,NOFORCE_SW)
#org 0x1F00,0x1FFF void loader16F877(void){}
#include <i2c_LCD.c>

int voltaje;                                                    //Voltaje del convertidor
long contador1,contador2,contador3;                             //Tres contadores
float voltaje_val;                                              //Valor calculado de voltaje
long val3;                                                      //Valor del conteo

#int_rtcc                                                       //Interrupción por Timer0
clock_isr(){
   contador1++;
   contador2++;
   contador3++;                                                 //Se incrementa el valor de todos los contadores

   if(contador1 == 763){                                        //Cada diez segundos
      voltaje = read_adc();                                     //Se lee el valor del convertidor

      voltaje_val = (voltaje*5.0)/255.0;                        //Se calcula el valor en V

      lcd_gotoxy(1,1);
      printf(lcd_putc,"Vin = %f V",voltaje_val);                //Se muestra en LCD el voltaje calculado

      contador1 = 0;                                            //Se reinicia el contador
   }

   if(contador2 == 1907){                                       //Cada 25 segundos
      printf("\nAlejandro Barreiro Valdez\tNum. cuenta: 317520888\n");
      printf("Jessica Zepeda Baeza\tNum. cuenta: 317520747\n");
      printf("Grupo Teor♦a 1\n");
      printf("Grupo Laboratorio 4\n");                          //Valores a imprimir en la terminal

      contador2 = 0;                                            //Se reinicia el contador
   }

   if(contador3 == 19){                                         //Cada 250 ms
      output_d(val3);                                           //Se muestra el conteo
      val3++;                                                   //Se incrementa el conteo
      contador3 = 0;                                            //Se reinicia el contador
   }

}

void main() {
   set_timer0(0);                                               //Inicia Timer en 00h
   setup_counters(RTCC_INTERNAL,RTCC_DIV_256);                  //Fuente de reloj y pre-divisor
   enable_interrupts(INT_RTCC);                                 //Habilita interrupción de Timer0
   enable_interrupts(GLOBAL);                                   //Habilita interrupciones globales

   Setup_port_a(ALL_ANALOG);                                    //Puerto A analógico
   Setup_adc(ADC_CLOCK_INTERNAL);                               //Define frecuencia de muestreo
   Set_adc_channel(0);                                          //Se utiliza el canal 0

   lcd_init(0x4E, 16, 2);                                       //Inicializa LCD en dirección 4E
   contador1 = 0;
   contador2 = 0;
   contador3 = 0;                                               //Se inicializan los contadores
   val3 = 0;                                                    //Inicializa el conteo

   while( TRUE ) {                                              //Loop infinito
   }
}
```

Ejercicio 5

C P10-EJ5.C

```c
#include <16F877.h>
#device adc=10                                                  //Convertidor a 10 bits
#fuses HS,NOWDT,NOPROTECT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#use i2c(MASTER,SDA=PIN_C4,SCL=PIN_C3,SLOW,NOFORCE_SW)
#include <i2c_LCD.c>
#org 0x1F00,0x1FFF void loader16F877(void){}

int voltaje,bandera,contador,val;
/*
voltaje:    Voltaje del convertidor
bandera:    Modo ascendente o descendente
contador:   Contador de salida
val:        Entrada del puerto B
*/
long contadorT0,contadorD,contadorRB0,contadorS;                //Contadores interrupciones
float voltaje_val;                                              //Voltaje calculado

```

```
20
21  void escribir_i2c(){                                    //Escribir en I2C de BCD
22  |   i2c_start();                                        //Inicia comunicación enviando bit S
23  |   i2c_write(0x42);                                    //Escribir en dirección 42 (BCD)
24  |   i2c_write(contadorRB0);                             //Escribir el contador
25  |   i2c_stop();                                         //Finaliza comunicación con bit de paro
26  }
27
28  #int_rtcc                                               //Interrupción Timer0
29  clock_isr(){
30  |   contadorT0++;                                      //Incrementa contador
31  |
32  |   if(contadorT0 == 15){                               //Cada 200ms
33  |   |   lcd_gotoxy(1,1);
34  |   |   printf(lcd_putc,"Contador: %x\n",contador);     //Salida de contador en LCD
35  |   |   contador++;                                     //Se incrementa conteo
36  |   |   if(contador == 255)                             //Si el valor es 255
37  |   |   |   contador = 0;                               //Reinicia contador de salida
38  |   |   contadorT0 = 0;                                 //Reinicia contador de interrupción
39  |   }
40  }
41
42  #int_ext                                                //Interrupción en R0
43  detecta_rb0(){
44  |   escribir_i2c();                                     //Escribe contador en BCD
45  |   contadorRB0++;                                      //Incrementa contador interrupción
46  }
47
48  #int_rda                                                //Interrupción recepción de datos
49  recepcion_serie(){
50  |   contadorS++;                                        //Incrementa contador interrupción
51  |   getchar();                                          //Se recibe valor de recepción
52  |   printf("Mensaje %lu recibido/n",contadorS);         //Mensaje recibido en terminal
53  }
54
```

```
55  #int_rb                                                 //Interupción 4 bits más significativos
56  port_rb(){
57  |   val = input_b();                                    //Lee la entrada del puerto B
58  |   printf("%x \n",val/0x10);                           //Muestra los 4 bits más significativos en hexadecimal
59  }
60
61  void main() {
62  |   set_timer0(0);                                      //Inicia Timer en 00h
63  |   setup_counters(RTCC_INTERNAL,RTCC_DIV_256);         //Fuente de reloj y pre-divisor
64  |   enable_interrupts(INT_RTCC);                        //Habilita interrupción de Timer0
65  |   enable_interrupts(GLOBAL);                          //Habilita interrupciones globales
66  |   enable_interrupts(INT_EXT);                         //Habilita interrupción en RB0
67  |   enable_interrupts(INT_RDA);                         //Habilita interrupción recepción de datos
68  |   enable_interrupts(INT_RB);                          //Habilita interrupción 4 bits más significativos B
69  |   ext_int_edge(L_TO_H);                               //Configura flanco a detectar
70  |
71  |   Setup_port_a(ALL_ANALOG);                           //Puerto A analógico
72  |   Setup_adc(ADC_CLOCK_INTERNAL);                      //Define frecuencia de muestreo
73  |   Set_adc_channel(0);                                 //Se utiliza el canal 0
74  |
75  |   lcd_init(0x4E, 16, 2);                              //Inicializa LCD en dirección 4E
76  |
77  |   contadorT0 = 0;
78  |   contador = 0;
79  |   contadorD = 0;
80  |   contadorS = 0;
81  |   contadorRB0 = 0;                                    //Contadores para interrupciones en 0
82  |   bandera = 0;                                        //Bandera para cuenta ascendente y descendente
83  |
84  |   while( TRUE ) {                                     //Loop infinito
85  |   |   output_d(contadorD);                            //Mostrar contador en Puerto D
86  |   |   if(bandera == 0)                                //Bandera en modo ascendente
87  |   |   |   contadorD++;                                //Incrementar contador
88  |   |   else                                            //Bandera en modo descendente
89  |   |   |   contadorD--;                                //Decrementar contador
90  |   |   if (contadorD == 20)                            //Si se llega a 20
91  |   |   |   bandera = 1;                                //Cambiar a modo descendente
92  |   |   if (contadorD == 0)                             //Si se llega a 0
93  |   |   |   bandera = 0;                                //Cambiar a modo ascendente
94  |   |   delay_ms(1000);                                 //Retardo de 1 segundo
95  |   }
96  }
```

Conclusiones y/o comentarios

Zepeda Baeza Jessica:

Considero que esta práctica fue mucho más compleja que las anteriores ya que hizo uso de todos los conceptos vistos a lo largo de las prácticas. Aunque en sí la práctica se enfocó en

el uso de interrupciones y del convertidor analógico digital, los ejercicios requerían la implementación de comunicación por puerto serie, de comunicación i2c, del uso de puertos paralelos, del uso de instrucciones y ciclos sencillos en C y más. Se observaron las diferencias de implementar el convertidor A/D en ensamblador y en C y las formas de hacer las mismas configuraciones como la definición de resolución, la elección de puertos analógicos y digitales y la elección del canal a usar. A pesar de que siempre se proporcionan códigos ejemplo, siento que en esta práctica sí fue necesario entender cómo funciona cada protocolo o puerto para poder utilizarlo de acuerdo a lo que se pedía. En mi caso, los programas realizados me ayudaron a visualizar cómo se conforman sistemas más complejos pero también me ayudaron a entender un poco más cómo funciona el lenguaje C. Es decir que tipo de valores admite cada tipo de dato y cómo se guardan en memoria, qué se guarda y donde al recibir un dato y más.

Barreiro Valdez Alejandro:

En esta práctica se utilizaron conceptos de prácticas pasadas como programación en C para el PIC16F877A, uso del protocolo I2C para ampliación de puertos, salidas y entradas en el puerto serie asíncrono y manejo de directivas y funciones especiales para la programación en C. Además, se agregaron nuevos conceptos para generar cinco ejercicios que combinan todos estos temas. Los nuevos conceptos que se agregaron fueron las interrupciones y el convertidor analógico/digital en C. El tema del convertidor se había cubierto en otra práctica para el ensamblador pero en está práctica se pudo ver cómo se realiza en C. Para C se definen directivas para el número de bits a convertir, la frecuencia del muestreo y el puerto que se quiere definir como analógico. Para las interrupciones se utilizaron diferentes interrupciones como el Timer0, la recepción de mensajes, el cambio de estado de los cuatro bits más significativos del puerto B y el cambio de estado de RB0. En C se definen las interrupciones generales y las específicas para poder hacer uso de ellas. Para las interrupciones que se mencionaron se escribe una función que se ejecutará cada vez que se presenta cada una de las interrupciones. Con todos los ejercicios se logró entender y aplicar cada uno de los temas mencionados en el microcontrolador PIC16F877A utilizando programación en C.