Índice

Práctica 1 a)

Hay que escribir el programa a ejecutar por el microcontrolador indicado en el esquema, de modo que se enciendan los 8 leds de forma consecutiva. El tiempo de encendido de cada led debe ser igual a 0.2 segundos y el tiempo que transcurre desde que se apaga un led hasta que se encienda el siguiente debe ser igual a 0.1 segundos.

```
void main() {
                        ADCON1=0x07;
                        TRISC.B0=0;
                        PORTC.B0=0;
                        TRISC.B1=0;
                        PORTC.B1=0;
                        TRISC.B2=0;
                        PORTC.B2=0;
                        TRISC.B3=0;
                        PORTC.B3=0;
                        TRISC.B4=0;
                        PORTC.B4=0;
                        TRISC.B5=0;
                        PORTC.B5=0;
                        TRISC.B6=0;
                        PORTC.B6=0;
                        TRISC.B7=0;
                        PORTC.B7=0;
                        while(1)
                        {
                         PORTC.B0=1;
                         delay_ms(200);
                         PORTC.B0=0;
                         delay_ms(100);
                         PORTC.B1=1;
                         delay_ms(200);
                         PORTC.B1=0;
                         delay_ms(100);
                         PORTC.B2=1;
                         delay_ms(200);
                         PORTC.B2=0;
                         delay_ms(100);
                         PORTC.B3=1;
                         delay_ms(200);
                         PORTC.B3=0;
                         delay_ms(100);
                         PORTC.B4=1;
                         delay_ms(200);
                         PORTC.B4=0;
```

```
                delay_ms(100);
                PORTC.B5=1;
                delay_ms(200);
                PORTC.B5=0;
                delay_ms(100);
                PORTC.B6=1;
                delay_ms(200);
                PORTC.B6=0;
                delay_ms(100);
                PORTC.B7=1;
                delay_ms(200);
                PORTC.B7=0;
                delay_ms(100);

            }
}
```

Práctica 1 b)

Hay que escribir el programa a ejecutar por el microcontrolador indicado en el esquema de la página anterior, de modo que se enciendan los 8 leds por parejas de forma consecutiva. La secuencia de encendido irá desde la pareja de leds D1-D8 hasta la pareja de leds D4-D5, primero en un sentido y luego en el otro, de forma consecutiva. El tiempo de encendido de cada pareja de leds debe ser igual a 0.4 segundos y el tiempo que transcurre desde que se apaga una pareja de leds hasta que se encienda la siguiente pareja de leds debe ser igual a 0.2 segundos.

```
void main() {
                        ADCON1=0x07;
                        TRISC=0;
                        PORTC=0;

                        while(1)
                        {
                         PORTC.B0=1;
                         PORTC.B7=1;
                         delay_ms(400);
                         PORTC.B0=0;
                         PORTC.B7=0;
                         delay_ms(200);
                         PORTC.B1=1;
                         PORTC.B6=1;
                         delay_ms(400);
                         PORTC.B1=0;
                         PORTC.B6=0;
                         delay_ms(200);
                         PORTC.B2=1;
                         PORTC.B5=1;
                         delay_ms(400);
                         PORTC.B2=0;
                         PORTC.B5=0;
                         delay_ms(200);
                         PORTC.B3=1;
                         PORTC.B4=1;
                         delay_ms(400);
                         PORTC.B3=0;
                         PORTC.B4=0;
                         delay_ms(200);
                         PORTC.B3=1;
                         PORTC.B4=1;
                         delay_ms(400);
                         PORTC.B3=0;
                         PORTC.B4=0;
                         delay_ms(200);
                          PORTC.B2=1;
                         PORTC.B5=1;
```

```
                delay_ms(400);
                PORTC.B2=0;
                PORTC.B5=0;
                delay_ms(200);
                PORTC.B1=1;
                PORTC.B6=1;
                delay_ms(400);
                PORTC.B1=0;
                PORTC.B6=0;
                delay_ms(200);
                 PORTC.B0=1;
                PORTC.B7=1;
                delay_ms(400);
                PORTC.B0=0;
                PORTC.B7=0;
                delay_ms(200);

               }
}
```

Práctica 2

Hay que diseñar un contador de módulo 60. El contenido del contador se visualizará en un doble display de 7 segmentos de cátodo común. Las señales a, b, c, d, e, f y g (ver página siguiente) son comunes a ambos displays, por lo que se debe activar alternativamente el display de unidades y el de decenas, de modo que parezca que siempre están activos. El contador debe incrementar su contenido cada segundo (periodo = 1seg).

```
/* Estructura básica de un programa */

//declaracion de variables globales

//declaracion (y definicion) de funciones

//declaracion y definicion de la ISR (rutina de servicio de
interrupciones)


void main()
{
 //declaracion de variables
 unsigned short
numeros[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7C,0x07,0x7F,0x67};
 int dec;
 int unid;
 int cont;

 ADCON1 = 0x07; //configuración de los canales analógicos (AN)
como digitales

 // configuracion de puertos
 TRISD = 0; //se declara RD0 como una salida digital
 PORTD = 0; //se pone el terminal RD0 a 0

 TRISA = 0; //se declara RA0 como una salida digital
 PORTA = 0; //se pone el terminal RA0 a 0

 //configuracion e inicializacion de los módulos del PIC que se
vayan a utilizar

 //configuración de interrupciones (si se utilizan)


 while(1) //bucle infinito
 {
```

```
  for(unid=0;unid<6;unid++){
                              for(dec=0;dec<10;dec++){

                                for(cont=0;cont<25;cont++){
                                PORTD = numeros[dec];
                                PORTA.B0 = 1 ;
                                delay_ms(20);
                                PORTA.B0 = 0;
                                //delay_ms(20);
                                PORTD = numeros[unid];
                                PORTA.B1 = 1 ;

                                delay_ms(20);
                                PORTA.B1 = 0;


}
                              }
                             /* PORTA.B0 = 1; //se pone el terminal
RC0 a 1 (el display de la derecha)
                              PORTD.B0 = 1; //se pone el terminal
RC0 a 1
                              delay_ms(20); //se introduce un
retardo de 200ms en la ejecución del código        */
 }


 }
}
// Componentes ISIS: PIC18F452, RES, LED-BLUE
```

Práctica 3 a)

El diodo emisor de luz (led) conectado a la patilla RB1 debe cambiar de estado (encendido/apagado) cada vez que se presiona el pulsador conectado al terminal RB0. En este apartado, se trata de resolver este problema haciendo que el microcontrolador observe periódicamente si el pulsador está presionado o no (técnica de polling ≡ observación periódica).

```
/* Estructura básica de un programa */

//declaracion de variables globales

//declaracion (y definicion) de funciones

//declaracion y definicion de la ISR (rutina de servicio de
interrupciones)


void main()
{
 //declaracion de variables
 unsigned short ant = 1;

 ADCON1 = 0x07; //configuración de los canales analógicos (AN)
como digitales

 // configuracion de puertos
 TRISB.B0 = 1; //se declara B0 como una entrada digital
 TRISB.B1 = 0; //se declara B1 como una salida digital

 //configuracion e inicializacion de los módulos del PIC que se
vayan a utilizar


 //configuración de interrupciones (si se utilizan)
  PORTB.B0 = 1;
  PORTB.B1 = 0;
  RBPU_bit = 0;

 while(1) //bucle infinito
 {
  delay_ms(100);
  if((ant == 1) && (PORTB.B0 == 0)){
           PORTB.B1 = !PORTB.B1;
           ant = 0;
  } else if((ant == 0) && (PORTB.B0 == 1))
   ant = 1;
```

```
 }
}
// Componentes ISIS: PIC18F452, RES, LED-BLUE
```

Práctica 3 b)

Hay que hacer que el diodo led conectado al terminal RB1 cambie de estado (encendido/apagado) cada vez que se presiona el pulsador conectado al terminal RB0. La diferencia con el apartado anterior reside en que ahora hay que utilizar la interrupción (INT0) para detectar los cambios de estado del pulsador.

```
/* Estructura básica de un programa */

//declaracion de variables globales

//declaracion (y definicion) de funciones

//declaracion y definicion de la ISR (rutina de servicio de
interrupciones)

void interrupt()
{
      INTCON.INT0IF=0;
      PORTB.B1 = !PORTB.B1;
}
void main()
{
 //declaracion de variables
 unsigned short ant = 1;

 ADCON1 = 0x07; //configuración de los canales analógicos (AN)
como digitales

 // configuracion de puertos
 TRISB.B0 = 1; //se declara B0 como una entrada digital
 TRISB.B1 = 0; //se declara B1 como una salida digital

 //configuracion e inicializacion de los módulos del PIC que se
vayan a utilizar

  RBPU_bit = 0;
 //configuración de interrupciones (si se utilizan)
  PORTB.B1 = 0;
  INTCON2.INTEDG0=1;
  INTCON.INT0IF=0;
  INTCON.INT0IE=1;
  INTCON.GIE=1;
```

```
  while(1);
}
// Componentes ISIS: PIC18F452, RES, LED-BLUE
```

Práctica 3 c)

Se trata de diseñar el circuito que indica el 'turno' o la 'vez' en los centros de salud, centros comerciales (frutería, carnicería, etc.). La idea es que cada vez que se presione un pulsador, el número decimal representado en un doble display de 7 segmentos incremente su contenido en 1 unidad. Hay que proponer un circuito a partir de los circuitos de las prácticas anteriores.

Nota 1: Se trata de implementar un contador de módulo 100 del mismo tipo que los estudiados en SD (modo de contaje ascendente), con la diferencia de que en éste caso hay que ver/representar el contenido del contador en base 10, en un doble display de 7 segmentos.

Nota 2: Lo primero que hay que decidir es si la CPU del microcontrolador puede observar el estado del pulsador y controlar, al mismo tiempo, el doble display de 7 segmentos. No se puede utilizar la técnica de polling (en lo que queda de curso).

```
unsigned short i = 0;
unsigned short j = 0;

void interrupt() {
     INTCON.INT0IF=0;
     if(i < 9){
              i++;
     }else if(j<9){
           i=0;
         j++;
    }else{
          i=0;
          j=0;
    }
}

void main()
{
  //declaración de variables
  unsigned short
numeros[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};
  unsigned short l;
  ADCON1 = 0x07;

  //configuración de los canales analógicos(AN) como
digitales(PIC18F452)

  //configuración de puertos

  TRISA = 0;
```

```
  TRISB.B0 = 1;
  RBPU_bit = 0;
  TRISD = 0;
  PORTD = 0;


  //configuración e inicialización delos módulos del PIC que se
utilicen (si se utilizan)

  //configuración de interrupciones (si se utilizan)

  INTCON2.INTEDG0 = 1;
  INTCON.INT0IF = 0;
  INTCON.INT0IE = 1;
  INTCON.GIE = 1;

  //instrucciones

  while(1){


          PORTD = numeros[i];
          PORTA.B0 = 1 ;
           delay_ms(20);
           PORTA.B0 = 0;
           PORTD = numeros[j];
           PORTA.B1 = 1 ;
           delay_ms(20);
           PORTA.B1 = 0;

                           }
                           }
```

Práctica 4 b)

Hay que controlar el funcionamiento de un motor paso a paso (Stepper motor) de tipo unipolar, configurado con pasos de 45º, utilizando un microcontrolador PIC18F452 (ver página siguiente). Componentes ISIS: PIC18F452, Motor-stepper, ULN2003A, RES, SW-SPST-MOM, DC generator.

El control del motor que hay que realizar consiste en lo siguiente:

1º: Cuando se inicia ISIS, el eje del motor está siempre en +0º (ver punto amarillo). Hay que hacer que el rotor se sitúe en −90º (posición inicial)

2º: Si se pulsa el botón SW1, el motor debe realizar un giro a izquierdas de 135º y detenerse.

3º: Una vez que el eje del motor ha realizado el giro anterior de 135º, si se pulsa otra vez el botón SW1, el motor debe realizar un giro a derechas de 360º y detenerse.

A partir de dicho momento, siempre que se vuelva a pulsar el botón SW1, el motor deberá realizar un giro a derechas de 360º.

```
unsigned short sw = 0;

void interrupt(){

if(sw==0){
          PORTC.B2 = 1;
          delay_ms(200);
          PORTC = 0;
          PORTC.B3 = 1;
          delay_ms(200);
          PORTC = 0;
          PORTC.B0 = 1;
          PORTC.B1 = 1;
          delay_ms(200);
          PORTC = 0;
          sw=1;

}else{
          PORTC.B0 = 1;
          delay_ms(200);
          PORTC = 0;
          PORTC.B3 = 1;
          delay_ms(200);
          PORTC = 0;
```

```
            PORTC.B2 = 1;
            delay_ms(200);
            PORTC = 0;
            PORTC.B1 = 1;
            delay_ms(200);
            PORTC = 0;
            PORTC.B0 = 1;
            delay_ms(200);
            PORTC = 0;
            PORTC.B3 = 1;
            delay_ms(200);
            PORTC = 0;
            PORTC.B2 = 1;
            delay_ms(200);
            PORTC = 0;
            PORTC.B0 = 1;
            PORTC.B1 = 1;
            delay_ms(200);
            PORTC = 0;
}
            INTCON.INT0IF = 0;
}


void main() {

TRISC = 0;
PORTC = 0;
INTCON2.INTEDG0 = 0;
INTCON.INT0IF = 0;
INTCON.INT0IE = 1;
INTCON.GIE = 1;

PORTC.B0=1;
delay_ms(200);
PORTC.B0 = 0;
PORTC.B1 = 1;
delay_ms(200);
PORTC.B2 = 1;
delay_ms(100);

while(1){
asm nop;
}


}
```

Práctica 5 a)

Introducir datos por medio de un teclado y visualizarlos en un LCD.

```
#include "Tecla12INT.h"

sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;

unsigned short key;
unsigned short x;

void interrupt(){
     key = tecla();
     Lcd_Chr(1,16,key);
     x=PORTB; //hay que leer el puerto B para poder borrar el bit
RBIF (define x global)
     INTCON.RBIF=0;//Al borrar el bit RBIF despues de llamar a la
funcion tecla, nos
}


void main() {
    TRISB = 0xF0;
    PORTB = 0;



    //configuración de interrupciones (si se utilizan)
  x = PORTB;

  INTCON2.RBPU = 0;
  INTCON.RBIF = 0;
  INTCON.RBIE = 1;
  INTCON.GIE = 1;
```

```
  RBPU_bit = 0;
  //Activar LCD
  Lcd_Init();
  while(1); //bucle infinito
}
```

Práctica 5 b)

El objetivo de este ejercicio es comprender y practicar las interrupciones por cambio de nivel de los terminales RB4-RB7. En el circuito de la página siguiente, el Lcd debe mostrar en todo momento el número de veces que se ha pulsado el botón SW1 desde que se ha puesto en funcionamiento el circuito. El sistema debe comenzar a contar en 0 (carácter 48 en ASCII) y una vez que se llegue a 99, cuando se presione otra vez el botón, el sistema debe pasar a 0 (se trata de implementar un contador de módulo 100).

```
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;

unsigned short sw = 0;
unsigned short x;
unsigned short num = 0;
unsigned short txt;

void interrupt(){
  if(sw == 0){
         sw  = 1;
  } else{
   sw = 0;
   num++;
   if(num == 100){
     num = 0;
   }
   ByteToStr(num,txt);
   Lcd_out(1,1,txt);
 }
 x=PORTB; //hay que leer el puerto B para poder borrar el bit RBIF
(define x global)
```

```
 INTCON.RBIF=0;//Al borrar el bit RBIF despues de llamar a la
funcion tecla, nos
}

void main() {
    //Activar LCD
    Lcd_Init();
    TRISB = 0xF0;
    PORTB = 0;

    //configuración de interrupciones (si se utilizan)
    x = PORTB;

    INTCON2.RBPU = 0;
    //Activar LCD
    Lcd_Init();
    INTCON.RBIF = 0;
    INTCON.RBIE = 1;
    INTCON.GIE = 1;
    RBPU_bit = 0;

    while(1); //bucle infinito
}
```

Práctica 5 c)

En este apartado se trata de modificar el código escrito para el apartado anterior de modo que cuando se ponga en funcionamiento el circuito, en la pantalla aparezca lo que se indica a continuación (sin que haya que pulsar el botón SW1)

```
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;

unsigned short sw = 0;
unsigned short x;
unsigned short num = 0;
unsigned short txt;
unsigned short cont = 0;
//unsigned char turno[] = {"Turno:   "};

void interrupt(){

  if(cont == 0){
   Lcd_out_CP("turno:   0");
   cont++;
  }

  if(sw == 0){
   Lcd_out_CP("turno: ");
   if(num<1)
   ByteToStr(0,txt);
   Lcd_out(1,14,txt);
         sw  = 1;
  } else{
   sw = 0;
   num++;
```

```
   if(num == 100){
     num = 0;
   }
   ByteToStr(num,txt);
   Lcd_out(1,14,txt);
   Lcd_Cmd(_LCD_CURSOR_OFF);

 }
 x=PORTB; //hay que leer el puerto B para poder borrar el bit RBIF
(define x global)
 INTCON.RBIF=0;//Al borrar el bit RBIF despues de llamar a la
funcion tecla, nos
}

void main() {
    //Activar LCD
    Lcd_Init();
    TRISB = 0xF0;
    PORTB = 0;
    INTCON2.RBPU = 0;
    //Activar LCD
    Lcd_Init();
    //configuración de interrupciones (si se utilizan)
    x = PORTB;
    INTCON.RBIF = 1;
    INTCON.RBIE = 1;
    INTCON.GIE = 1;

    while(1); //bucle infinito
}
```

Práctica 6 a)

Se trata de generar una señal digital de frecuencia 1KHz, con un ciclo de trabajo igual a 0.3 (ver página siguiente). Para resolver este problema hay que utilizar el TIMER0, de modo que cada vez que produzca una interrupción (por overflow) cambie el estado del terminal RC0 del PORTC. (mira la página siguiente)

```
void interrupt()
{
     INTCON.TMR0IF = 0;
     if(PORTC.B0 == 1)
     {
          PORTC.B0 = 0;
          T0CON = 0XC2;
          TMR0L = 81;

          }else
          {
               PORTC.B0 = 1;
               T0CON = 0XC1;
               TMR0L = 106;
          }
}

void main() {
     TRISC.B0 = 0;
     PORTC.B0 = 0;
     T0CON = 0XC1;
     INTCON.TMR0IF = 0;
     INTCON.TMR0IE = 1;
     TMR0L = 106;
     INTCON.GIE = 1;
     while(1);
}
```

Práctica 6 c)

Construir un temporizador de 1 minuto, no redisparable. Para ello se utilizará la interrupción INT1 y el Timer0. Cada vez que se presiona el pulsador, la salida RC0 debe ponerse a nivel alto durante 1 minuto (el led deberá estar encendido durante 60 segundos).

```
unsigned short i=0;
void interrupt()
{
        if((INTCON3.INT1IF)&&(INTCON3.INT1IE))
        {
                        PORTC.B0=1;
                        T0CON=0x87;
                        TMR0H=(18661>>8);
                        TMR0L=18661;
                        INTCON3.INT1IF=0;
                        INTCON3.INT1IE=0;
        }
         if((INTCON.TMR0IF)&&(INTCON.TMR0IE))
         {
                INTCON.TMR0IF = 0;
                i++;
                if(i<10){
                        TMR0H=(18661>>8);
                        TMR0L=18661;
            }else{
                PORTC.B0=0;
                T0CON=0;
                INTCON3.INT1IF=0;
                INTCON3.INT1IE=1;
                i=0;
            }
         }

}

void main() {
     TRISC.B0 = 0;
     PORTC.B0 = 0;
     TRISB.B1 = 1;
     INTCON2.INTEDG1 = 1;
     INTCON3.INT1IF = 0;
     INTCON3.INT1IE = 1;
     INTCON.TMR0IF = 0;
     INTCON.TMR0IE = 1;
     INTCON.GIE = 1;
```

```
     while(1);
}
```

Práctica 7 a)

Esta práctica consiste en construir un medidor de tensiones continuas (un voltímetro). La tensión (V1) a medir deberá muestrearse con una frecuencia de 1Hz. En este apartado se puede utilizar la función delay_ms()

```
unsigned int aux=0;

char txt[14];

float alfa;



sbit LCD_RS at RD2_bit;

sbit LCD_EN at RD3_bit;

sbit LCD_D7 at RD7_bit;

sbit LCD_D5 at RD5_bit;

sbit LCD_D6 at RD6_bit;

sbit LCD_D4 at RD4_bit;


sbit LCD_RS_Direction at TRISD2_bit;

sbit LCD_EN_Direction at TRISD3_bit;

sbit LCD_D7_Direction at TRISD7_bit;

sbit LCD_D5_Direction at TRISD5_bit;

sbit LCD_D6_Direction at TRISD6_bit;

sbit LCD_D4_Direction at TRISD4_bit;



void interrupt(){

     PIR1.ADIF=0;
```

```
      aux=ADRESL;
      aux=aux+(ADRESH<<8);
      alfa= aux*0.004887585;
      FloatToStr(alfa,txt);
      Lcd_Cmd(_LCD_CLEAR);
      LCD_out(1,1,txt);
      delay_ms(1000);
      ADCON0.b2=1;

}

void main() {

                  TRISA.B0=1;
                  ADCON1=0xDE;
                  ADCON0=0x41;
                  LCD_init();
                  PIR1.ADIF = 0;
                  PIE1.ADIE=1;
                  INTCON.PEIE=1;
                  INTCON.GIE=1;
                  ADCON0.B2 = 1;

                  while(1);

}
```

Práctica 7 b)

En este apartado hay que construir un termómetro basado en el uso de un sensor LM35. El periodo de muestreo de la tensión Vout proporcionada por el sensor debe ser de 1,5 segundos. La temperatura se representará en la pantalla de cristal líquido en grados Celsius. En esta apartado no se pueden utilizar las funciones delay_ms() y delay_us(). Utiliza un Counter timer para comprobar que las temporizaciones son de 1,5 seg.

```
unsigned int aux=0;
char txt[14];
float alfa;
char x=0;

//Configuracioón del LCD
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D7 at RD7_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D4 at RD4_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D7_Direction at TRISD7_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D4_Direction at TRISD4_bit;

//Configuramos la interrupción
void interrupt()
{
  if(INTCON.TMR0IF)//Comprueba el flag de interrupción del timer
está ejecutada(Por eso se iguala a 1)
      {
        INTCON.TMR0IF = 0;//Deshabilita el flag interrupción
        TMR0H = (18661>>8);//Parte alta
        TMR0L = 18661;//Parte baja
        ADCON0.b2 = 1;//Activa conversión(de analógico a digital)
        PORTC.B0 = !PORTC.B0;
      }

   if(PIR1.ADIF)//Si finaliza la conversión
      {
           PIR1.ADIF = 0;//Pones la conversión sin finalizar
           aux = ADRESL;//Resultado del AD
           aux = aux+(ADRESH<<8);//Resultado del AD
           alfa = aux*0.48875855;//T=100*Vout
           floatToStr(alfa,txt);//Convertir para que se muestre en
LCD
           Lcd_Cmd(_LCD_CLEAR);
           LCD_out(1,1,txt);//Lo muestra
```

```
      }

}

void main() {
                TRISC=0;
                PORTC = 0;
                TRISE.B1=1;//Se activa como entrada el bit 1 del
E
                ADCON1=0xC0;//Registro ADCON
                ADCON0=0x71;//Registro ADCON
                LCD_init();//Inicia el LCD
                T0CON=0x85;//Configurar timer0
                TMR0H = (18661>>8);//Parte alta
                TMR0L = 18661;//Parte baja
                INTCON.TMR0IF = 0;//Deshabilita el flag
interrupción
                INTCON.TMR0IE = 1;//Habilita la interrupción.
                PIR1.ADIF = 0;//Deshabilito el flag de la
conversión
                PIE1.ADIE = 1;//Habilito la conversión
                INTCON.PEIE = 1;//Es de tipo core
                INTCON.GIE = 1;//Se habilita las interrupciones
                ADCON0.b2 = 1;//Activa conversión(de analógico a
digital)

                while(1);

}
```

Práctica 7 c)

Modificar el código y el circuito del apartado 7 b) de modo que al pulsar un botón el sistema cambie la escala de la temperatura representada (grados Celsius, Farenheit y Kelvin). En esta apartado no se pueden utilizar las funciones delay_ms(), delay_us(), ADC_Get_Sample(), ADC_Read(). Tampoco se puede utilizar la técnica de polling

```
unsigned short aux;
char txt[14], let;
unsigned out;
char cont = 2;
// Lcd pinout settings
 sbit LCD_RS at RD2_bit;
 sbit LCD_EN at RD3_bit;
 sbit LCD_D7 at RD7_bit;
 sbit LCD_D6 at RD6_bit;
 sbit LCD_D5 at RD5_bit;
 sbit LCD_D4 at RD4_bit;
 // Pin direction
 sbit LCD_RS_Direction at TRISD2_bit;
 sbit LCD_EN_Direction at TRISD3_bit;
 sbit LCD_D7_Direction at TRISD7_bit;
 sbit LCD_D6_Direction at TRISD6_bit;
 sbit LCD_D5_Direction at TRISD5_bit;
 sbit LCD_D4_Direction at TRISD4_bit;

void interrupt(){
        if(INTCON.TMR0IF){
                TMR0H = (18661>>8);
                TMR0L = 18661;
                ADCON0.B2 = 1;
```

```
         INTCON.TMR0IF = 0;
}
if(PIR1.ADIF){

         aux= ADRESL + (ADRESH << 8);
         out = (4.88e-1)*aux;

        if(cont<=2){
                   let = 'C';

        }else if(cont<=4){
                   out=out+273,15;
                   let = 'K';
        }else{
              out = ((1)*out)+32; //Al poner 1.8 da Demo
         Limit
                   let = 'F';
        }
         LCD_cmd(_LCD_CLEAR);
         FloatToStr(out, txt);
         Lcd_out(1,1, txt);
         Lcd_Chr_cp(223);
         Lcd_Chr_cp(let);

         PIR1.ADIF = 0;

}
if(INTCON.RBIF){
PORTB.B1=!PORTB.B1;
cont++;
        if(cont >= 7){
```

```
                        cont=0;
                }
                ADCON0.B2 = 1;

        INTCON.RBIF = 0;
        }
}
void main() {
     ADCON1 = 0xC0;
     ADCON0 = 0x71;

     TRISE.B1 = 1;
     TRISB.B4 = 1;


     Lcd_Init();
     T0CON = 0X85;
     TMR0H = (18661>>8);
     TMR0L = 18661;

     RCON.IPEN = 0;
     INTCON.RBIF = 0;   // se pone el flaga 0
     INTCON.RBIE = 1;    // se habilita la interrupción por cambio
     de nivel
     INTCON.TMR0IF = 0;   // se pone el flaga 0
     INTCON.TMR0IE = 1;    // se habilita la interrupción del
Timer 0

     PIR1.ADIF = 0;   //el bit PIR1.ADIF se pone a 1 siempre que
el convertidor AD finaliza una conversión
     PIE1.ADIE = 1;   /*se habilitan las interrupciones del
convertidor AD.
```

```
                        Siempre que el bit (flag) PIR1.ADIF se
ponga a 1, el micro dejara de ejecutar el codigo

                        que este ejecutando en ese momento para
ejecutar la rutina asociada a la finalizacion de

                        una conversion AD*/


     INTCON.PEIE = 1;

     INTCON.GIE = 1;    // se habilitan las interrupciones en
general



     ADCON0.B2 = 1;

     while(1){

     asm nop;


     }

}
```

Examen Mayo Ej 19

En este ejercicio se trata de diseñar un sistema que mida el tiempo que transcurre entre dos pulsaciones consecutivas de un botón, utilizando el circuito indicado en la parte derecha. Supon que el tiempo a medir nunca va a ser superior a 500mseg.

El sistema debe calcular el tiempo pedido con la mayor precisión posible, teniendo en cuenta que la frecuencia de la señal de reloj del PIC18F452 es de 12MHz. El tiempo medido debe representarse, en segundos, en la pantalla de cristal liquido (Lcd). Escribe en lenguaje C el código a ejecutar por el microprocesador.

```
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D7 at RD7_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D7_Direction at TRISD7_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D4_Direction at TRISD4_bit;
char flag=0;
unsigned int x = 0;
float aux = 0;
char txt[10];
void interrupt()
{
if(INTCON.INT0IF ==1){
if(flag==1){
  x = TMR0L + (TMR0H<<8);
  TMR0H =0;
  TMR0L=0;
  aux = x*1.07e-5;
```

```
  FloatToStr(aux,txt);
  Lcd_cmd(_lcd_clear);
  Lcd_out(1,1,txt);

  flag=0;
}
if(flag==0){
  T0CON=0x84;
  flag=1;
}
 INTCON.INT0IF = 0;
}
}
void main()
{
    TRISB.B0 = 1;
    RBPU_bit=0;
    TMR0H=0;
    TMR0L=0;
    T0CON=0;
    Lcd_Init();
    INTCON.TMR0IF = 0;
    INTCON.TMR0IE = 1;
    INTCON2.INTEDG0 = 1;
    INTCON.INT0IF = 0;
    INTCON.INT0IE = 1;
    INTCON.GIE = 1;
 while(1){
}
}
```