

Microcontroladores

Semestre: 2021-1

Profesor: Kalun José Lau Gan

Semana 10: - Manejo de LM35 con A/D y LCD
- Manejo de interrupciones en XC8

1

Preguntas previas:

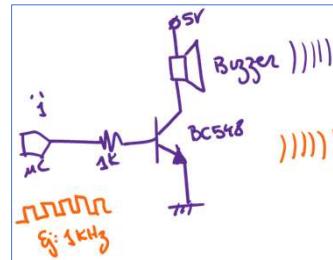
- `__delay_ms()` se puede colocar un segundo?
 - Si: `__delay_ms(1000);`
- Se puede hacer retardo en microsegundos?
 - Si: `__delay_us(10);`
- Recordar que la instrucción `__delay_us(x)`, el valor x es una constante, no pueden colocarle una variable,
- Importante: el 14 de mayo salió la nueva versión de MPLABX, la v5.50

2

Preguntas previas:

- Dos tipos de buzzer:

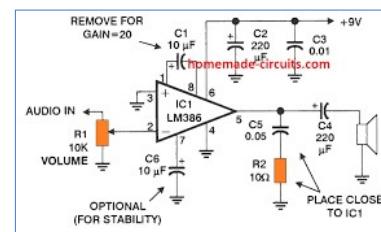
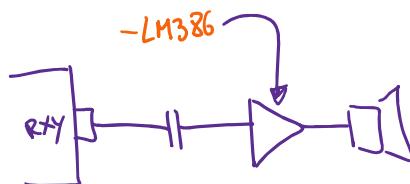
- Los activos con nivel lógico (le colocas energía DC y empiezan a sonar)
- Los que necesitan una señal oscilante para sonar (mismo parlante)



3

Preguntas previas:

- ¿Cómo conectar un parlante de 8Ω al microcontrolador?



4

Agenda

- Uso del sensor de temperatura en el microcontrolador PIC18F4550 en XC8
- Caracteres personalizados en el LCD en XC8

5

Sensor LM35

- Mide temperatura en °C
- Rango de temperatura:
-55°C-150°C
- Salida analógica 10mV/°C
- Alimentación desde
4VDC hasta 20VDC
- Respuesta lineal

National Semiconductor

November 2000

LM35 Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of $\pm\frac{1}{4}^\circ\text{C}$ at room temperature and $\pm\frac{3}{4}^\circ\text{C}$ over a full -55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 μA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a -55° to +150°C temperature range, while the LM35C is rated for a -40° to +110°C range (-10° with improved accuracy). The LM35 series is available pack-

aged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteed (at +25°C)
- Rated for full -55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 μA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only $\pm\frac{1}{4}^\circ\text{C}$ typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

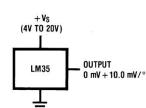
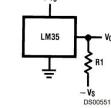


FIGURE 1. Basic Centigrade Temperature Sensor
(-55°C to +150°C)

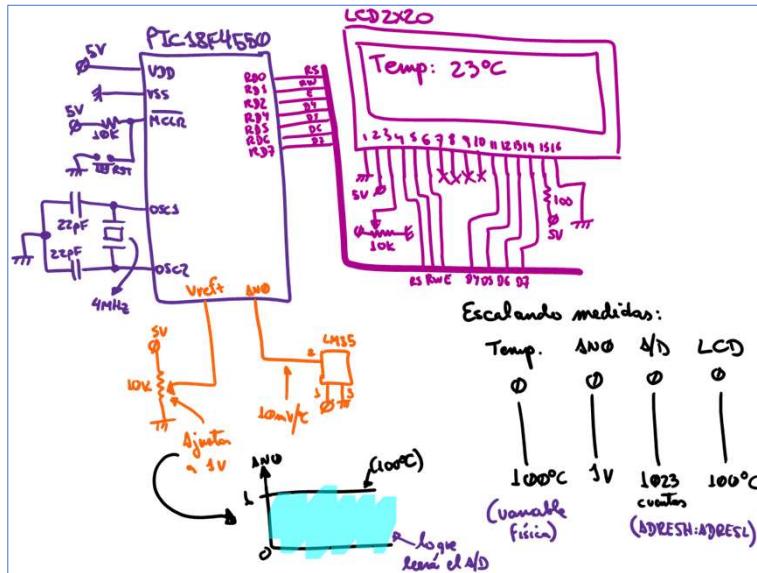


Choose $R_1 = -V_g/50 \mu\text{A}$
 $V_{out} = 1,500 \text{ mV at } +150^\circ\text{C}$
 $= +250 \text{ mV at } +25^\circ\text{C}$
 $= -550 \text{ mV at } -55^\circ\text{C}$

DS005516-4

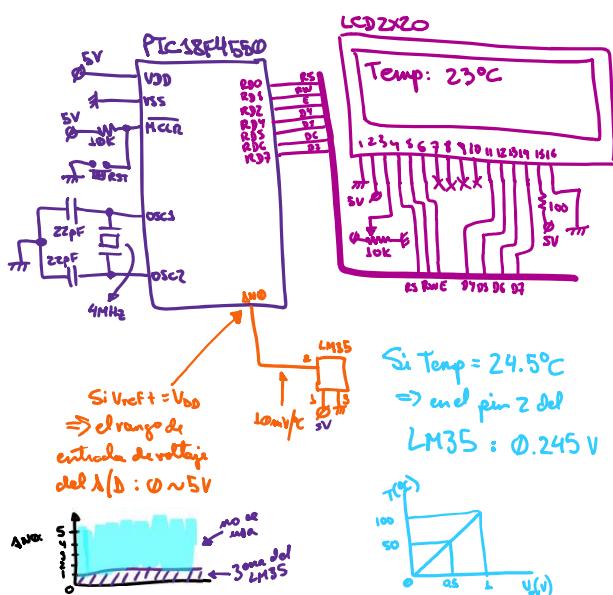
6

Interface del LM35 con el PIC18F4550



7

Interface del LM35 con el PIC18F4550



El A/D del uC: 10 bits de resolución
 $2^{10} = 1024$ escalones

Si el rango de entrada: $0-5V$,
el valor de cada escalón sera:

$$\frac{5}{1024} = 4.88mV$$

Si el LM35 tiene un rango de $0-1V$,
cuantos escalones empleara?
-Solo 205 escalones

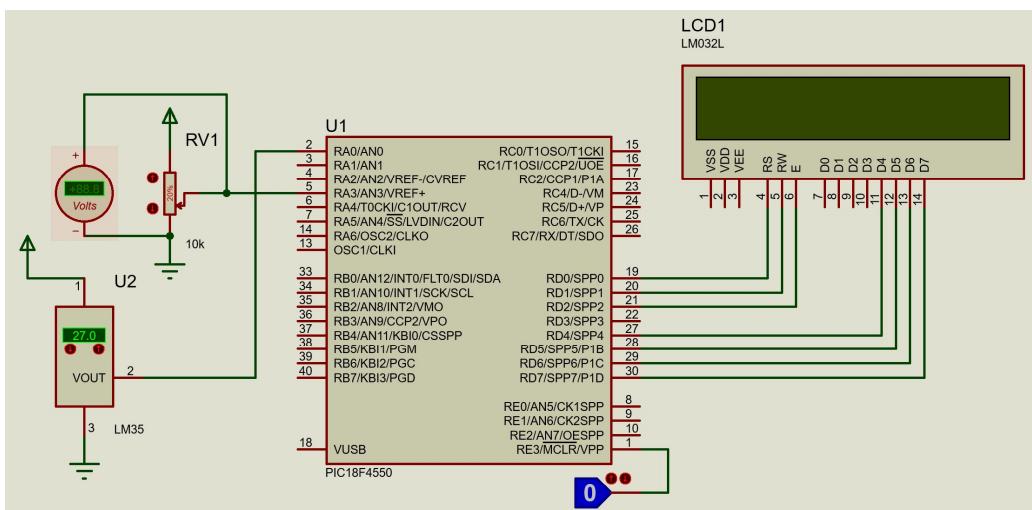
Si ajustamos el $V_{ref(+)}$ a $1V$ entonces
el rango de entrada sera $0-1V$, al
valor del escalón sera:

$$\frac{1}{1024} = 0.9765 mV$$

\Rightarrow Ahora si utilizamos todos los
escalones de 10 bits del A/D para
medir el LM35.

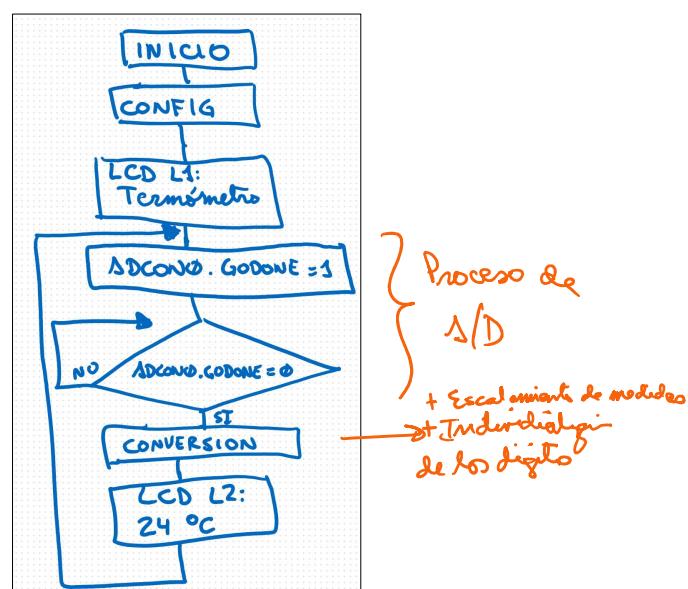
8

Interface del LM35 con el PIC18F4550



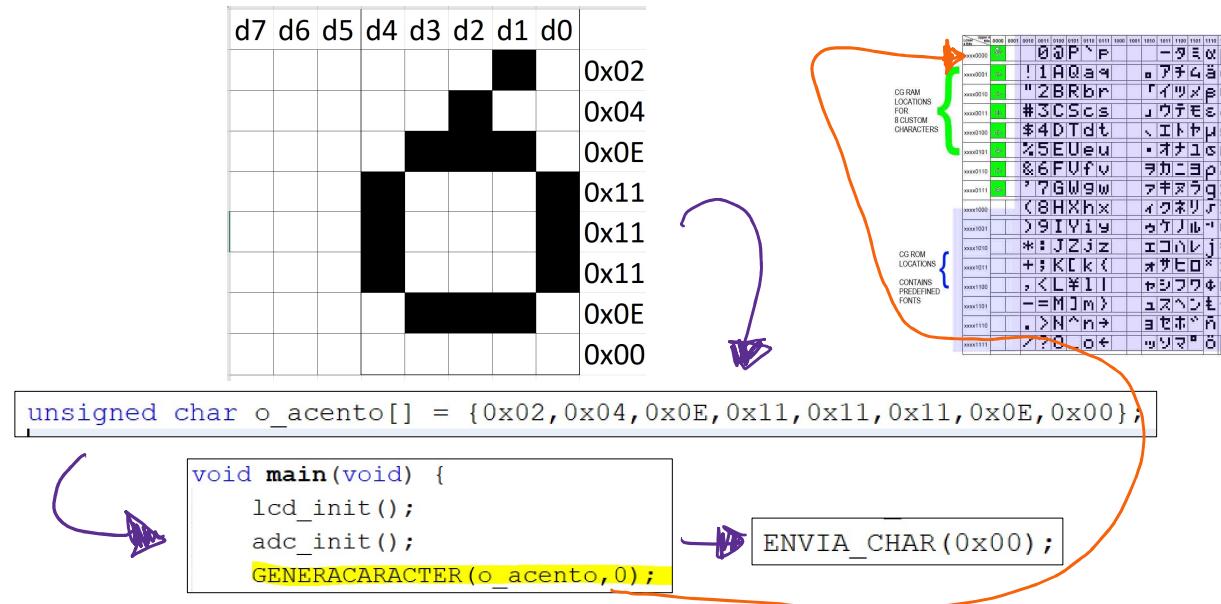
9

Diagrama de flujo del programa:



10

Caracteres personalizados en el display



11

Interface del LM35 con el PIC18F4550

- Código ejemplo:

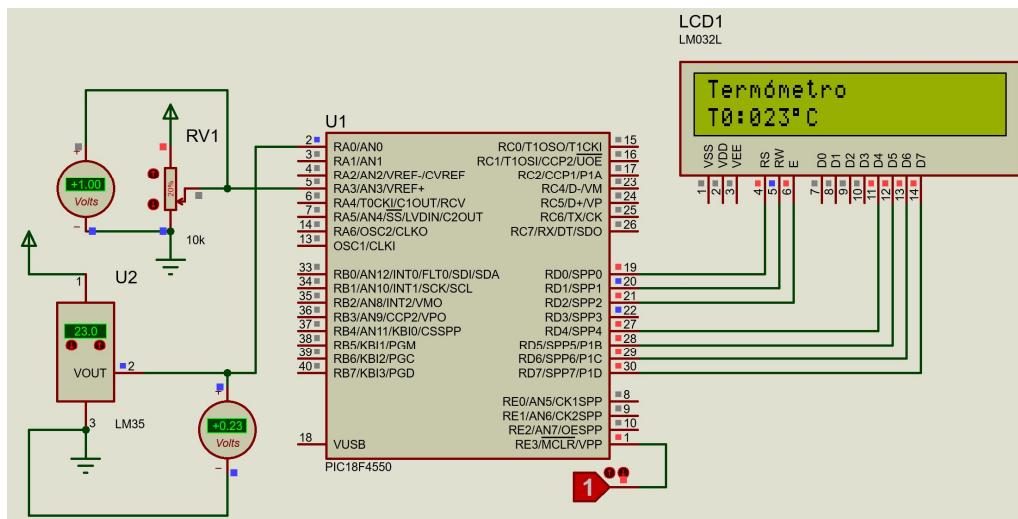
```

14 #define _XTAL_FREQ 48000000UL
15
16 //Declaracion de variables globales
17 unsigned int d_millar = 0;
18 unsigned int millar = 0;
19 unsigned int centena = 0;
20 unsigned int decena = 0;
21 unsigned int unidad = 0;
22 unsigned int lm35raw = 0;
23 unsigned char o_acento[] = {0x02, 0x04, 0x0E, 0x11, 0x11, 0x0E, 0x00};
24
25 //Funcion para inicializar el LCD
26 void lcd_init(void){
27     TRISE = 0x00;
28     LCD_CONFIG();
29     _delay_ms(15);
30     BORRAR_LCD();
31     CURSOR_HOME();
32     CURSOR_ONOFF(OFF);
33 }
34
35 //Funcion para inicializar el ADC
36 void adc_init(void){
37     ADCON2 = 0xA4; //ADFM=1
38     ADCON1 = 0x1B; //Vref+
39     ADCON0 = 0x01; //ADON=1
40 }
41
42 void convierte(unsigned int numero){
43     d_millar = numero / 10000;
44     millar = (numero % 10000) / 1000;
45     centena = (numero % 1000) / 100;
46     decena = (numero % 100) / 10;
47     unidad = numero % 10;
48 }
49
50 void main(void) {
51     lcd_init();
52     adc_init();
53     GENERACARACTER(o_acento, 0);
54     POS_CURSOR(1, 0);
55     ESCRIBE_MENSAJE("Term", 4);
56     ENVIA_CHAR(0x00);
57     ESCRIBE_MENSAJE("metro", 5);
58     while(1){
59         ADCON0bits.GODONE = 1;
60         while (ADCON0bits.GODONE == 1);
61         lm35raw = (ADRESH << 8) + ADRESL;
62         POS_CURSOR(2, 0);
63         ESCRIBE_MENSAJE("T0:", 3);
64         lm35raw = lm35raw / 10;
65         convierte(lm35raw);
66         ENVIA_CHAR(d_millar+0x30);
67         ENVIA_CHAR(millar+0x30);
68         ENVIA_CHAR(centena+0x30);
69         ENVIA_CHAR(decena+0x30);
70         ENVIA_CHAR(unidad+0x30);
71         ENVIA_CHAR(0xF);
72         ESCRIBE_MENSAJE("C", 1);
73     }
74 }

```

12

Simulación



13

Modificación al ejemplo anterior:

- En el ejemplo anterior se tenía un error significativo entre el valor de la variable física y el valor mostrado en el LCD, esto debido al manejo de números enteros en la operación de escalamiento.
- Para solucionarlo se está empleando una variable de punto flotante y se empleará el valor de 10.24 para el escalamiento.
- Otra variable “float” se empleará para obtener el valor de la temperatura en grados Fahrenheit.
- El uso de variables y operaciones con flotantes hará que se consuma mayores recursos en el microcontrolador tanto en tiempo de procesamiento como en espacio ocupado en la memoria de programa.

14

Modificación al ejemplo anterior:

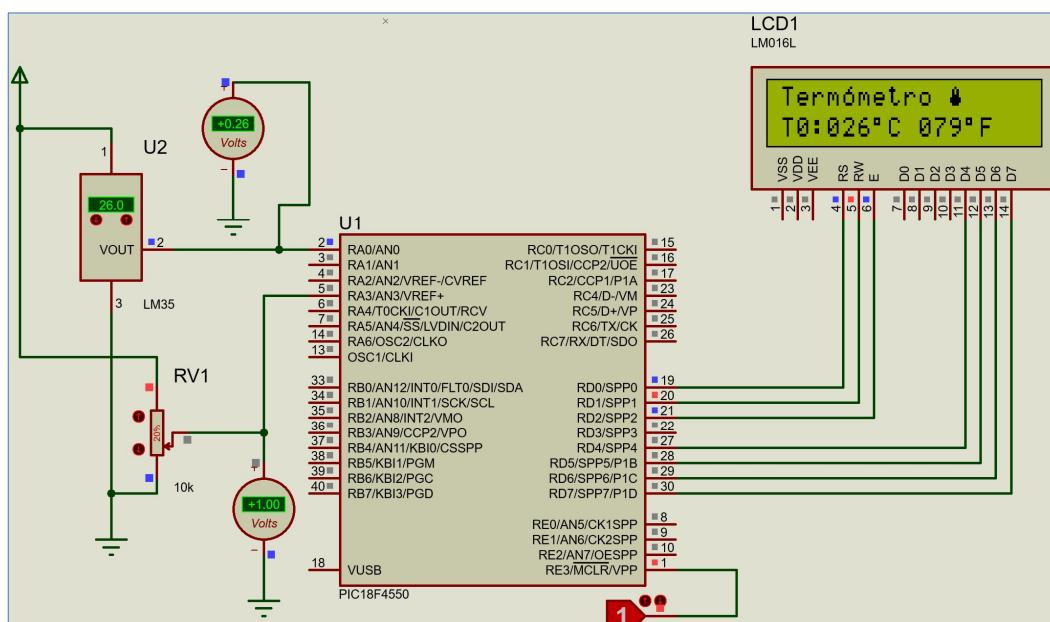
```

1 #include <xc.h>
2 #include <stdio.h>
3 #include "maincode.h"
4 #include "LCD.h"
5 #define _XTAL_FREQ 48000000UL
6
7 //declaracion de variables globales
8 unsigned int lm35raw = 0;
9 unsigned int millar = 0;
10 unsigned int centena = 0;
11 unsigned int decena = 0;
12 unsigned int unidad = 0;
13 unsigned char o_acento[] = {0x02, 0x04, 0x0Ee, 0x11, 0x11, 0x0E, 0x00};
14 unsigned char thermo[] = {0x04, 0x0A, 0x0A, 0x0E, 0x1F, 0x1F, 0x0E, 0x00};
15 float n_temp_c = 0;
16 float n_temp_f = 0;
17
18 void convierte(unsigned int numero){
19     millar = (numero % 1000) / 1000;
20     centena = (numero % 100) / 100;
21     decena = (numero % 10) / 10;
22     unidad = numero % 10;
23 }
24
25 void lcd_init(void) {
26     TRISD = 0x00;
27     __delay_ms(50);
28     LCD_CONFIG();
29     __delay_ms(15);
30     BORRAR_LCD();
31     CURSOR_HOME();
32     CURSOR_ONOFF(0xFF);
33     GENERACARACTER(o_acento, 0);
34     GENERACARACTER(thermo, 1);
35 }
36
37 void adc_init(void) {
38     ADCON2 = 0xA4;
39     ADCON1 = 0x1B;
40     ADCON0 = 0x01;
41 }
42
43 void main(void) {
44     lcd_init();
45     adc_init();
46     POS_CURSOR(1, 0);
47     ESCRIBE_MENSAJE("Term", 4);
48     ENVIA_CHAR(0x00);
49     ESCRIBE_MENSAJE("metro ", 6);
50     ENVIA_CHAR(0x01);
51     while(1) {
52         ADCON0bits.GODONE = 1; //toma de una muestra en AN0
53         while(ADCON0bits.GODONE == 1);
54         POS_CURSOR(2, 0);
55         ESCRIBE_MENSAJE("T0:", 3);
56         lm35raw = (ADRESH << 8) + ADRESL; //ADRESH:ADRESL
57         n_temp_c = lm35raw / 10.24;
58         n_temp_f = (n_temp_c * 9 / 5) + 32;
59         convierte(n_temp_c);
60         ENVIA_CHAR(centena+0x30);
61         ENVIA_CHAR(decena+0x30);
62         ENVIA_CHAR(unidad+0x30);
63         ENVIA_CHAR(0x0F);
64         ESCRIBE_MENSAJE("C ", 2);
65         convierte(n_temp_f);
66         ENVIA_CHAR(centena+0x30);
67         ENVIA_CHAR(decena+0x30);
68         ENVIA_CHAR(unidad+0x30);
69         ENVIA_CHAR(0x0F);
70         ESCRIBE_MENSAJE("F", 1);
71     }
72 }

```

15

Modificación al ejemplo anterior:



16

Manejo de las interrupciones en XC8

- Similar que en MPASM.
- Tener en cuenta que en versiones anteriores del XC8 se tenía una sintaxis diferente para las interrupciones (cuadro de la izquierda)

```
void interrupt high_priority Alta_p(void)
{
    /*Ingrese el código de la rutina de servicio de
     *interrupción de alta prioridad aquí*/
}

void interrupt low_priority Baja_p(void)
{
    /*Ingrese el código de la rutina de servicio de
     *interrupción de baja prioridad aquí*/
}
```

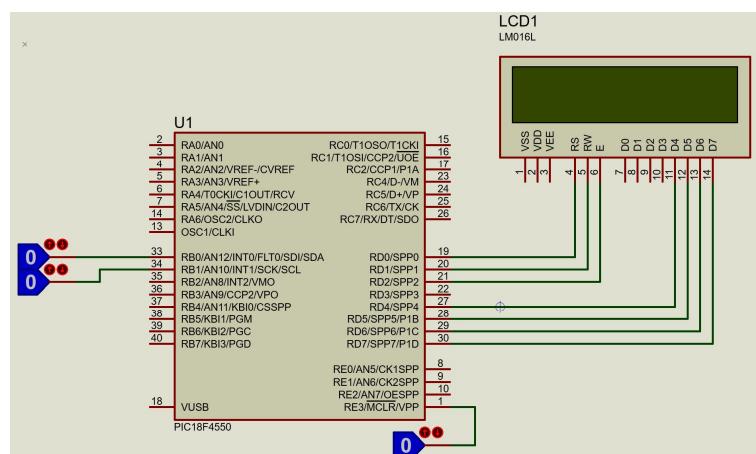
```
void __interrupt(high_priority) High_ISR(void)
{
    /*Aquí se escribe la rutina de alta prioridad*/
}

void __interrupt(low_priority) Low_ISR(void)
{
    /*Aquí se escribe la rutina de baja prioridad*/
}
```

17

Ejemplo: INT0 (high priority) e INT1 (low priority) en XC8

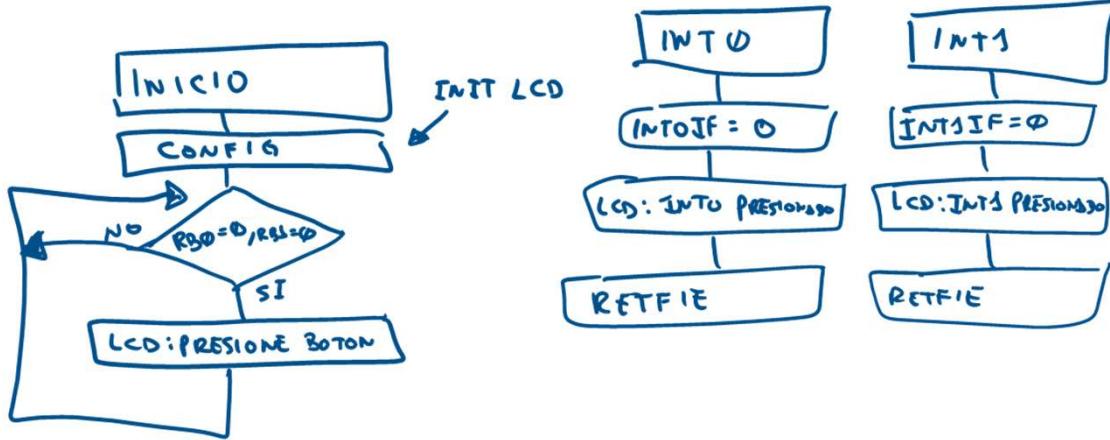
- En la pantalla se mostrará si se ha presionado las entradas INT0 o INT1.



18

Ejemplo: INT0 (high priority) e INT1 (low priority) en XC8

- Diagrama de flujo:



19

Ejemplo: INT0 (high priority) e INT1 (low priority) en XC8

- Código inicial:

```

16 void init_conf(void) {
17     RCONbits.IREN = 1;           //prioridades habilidadadas
18     INTCONbits.INT0IE = 1;       //Int0 habilidadada
19     INTCON3bits.INT1IE = 1;       //Int1 habilidadada
20     INTCON3bits.INT1IP = 0;      //Int1 se va a low priority
21     INTCONbits.GIEH = 1;         //Interruptor global de ints HF activado
22     INTCONbits.GIEL = 1;         //Interruptor global de ints LF activado
23 }
24
25 void lcd_conf(void) {
26     TRISD = 0x00;
27     __delay_ms(50);
28     LCD_CONFIG();
29     __delay_ms(15);
30     BORRAR_LCD();
31     CURSOR_HOME();
32     CURSOR_ONOFF(OFF);
33 }
34
35 void main(void) {
36     init_conf();
37     lcd_conf();
38     POS_CURSOR(1,0);
39     ESCRIBE_MENSAJE("Detector UPC2021",16);
40     while(1){
41     };
42 }
  
```



```

44 void __interrupt(high_priority) INT0_ISR(void) {
45     POS_CURSOR(2,0);
46     ESCRIBE_MENSAJE("Presionaste INT0",16);
47     INTCONbits.INT0IF = 0;           //bajamos la bandera de INT0
48 }
49
50 void __interrupt(low_priority) INT1_ISR(void) {
51     POS_CURSOR(2,0);
52     ESCRIBE_MENSAJE("Machucaste INT1 ",16);
53     INTCON3bits.INT1IF = 0;          //BAjamos la bandera de INT1
54 }
  
```

20

Ejemplo: INT0 (high priority) e INT1 (low priority) en XC8

```

16     unsigned char indicador = 0;
17
18     void init_conf(void){
19         RCONbits.IREN = 1;           //Prioridades habilitadas
20         INTCONbits.INT0IE = 1;      //INT0 habilitada
21         INTCON3bits.INT1IE = 1;      //INT1 habilitada
22         INTCON3bits.INT1IF = 0;      //INT1 se vaya a low
23         INTCONbits.GIEH = 1;        //Interruptor global
24         INTCONbits.GIEL = 1;        //Interruptor global
25     }
26
27     void lcd_conf(void){
28         TRISE = 0x00;
29         _delay_ms(50);
30         LCD_CONFIG();
31         _delay_ms(15);
32         BORRAR_LCD();
33         CURSOR_HOME();
34         CURSOR_ONOFF(OFF);
35     }
36
37     void main(void) {
38         init_conf();
39         lcd_conf();
40         POS_CURSOR(1,0);
41         ESCRIBE_MENSAJE("Detector UPC2021",16);
42         while(1){
43             if (indicador == 1){
44                 indicador = 0;
45                 _delay_ms(3000);
46                 POS_CURSOR(2,0);
47                 ESCRIBE_MENSAJE(" ",16);
48             }
49         }
50     }

```

- Código con borrado de pantalla:

```

52     void __interrupt(high_priority) INT0_ISR(void){
53         POS_CURSOR(2,0);
54         ESCRIBE_MENSAJE("Presionaste INT0",16);
55         INTCONbits.INT0IF = 0;
56         indicador = 1;
57     }
58
59     void __interrupt(low_priority) INT1_ISR(void){
60         POS_CURSOR(2,0);
61         ESCRIBE_MENSAJE("Machucaste INT1 ",16);
62         INTCON3bits.INT1IF = 0;
63         indicador = 1;
64     }

```

21

Modificación al ejemplo anterior

- Ahora se tendrán habilitadas las tres interrupciones externas (INT0, INT1 e INT2)
- La INT1 y la INT2 estarán en el vector de interrupción de baja prioridad.

22

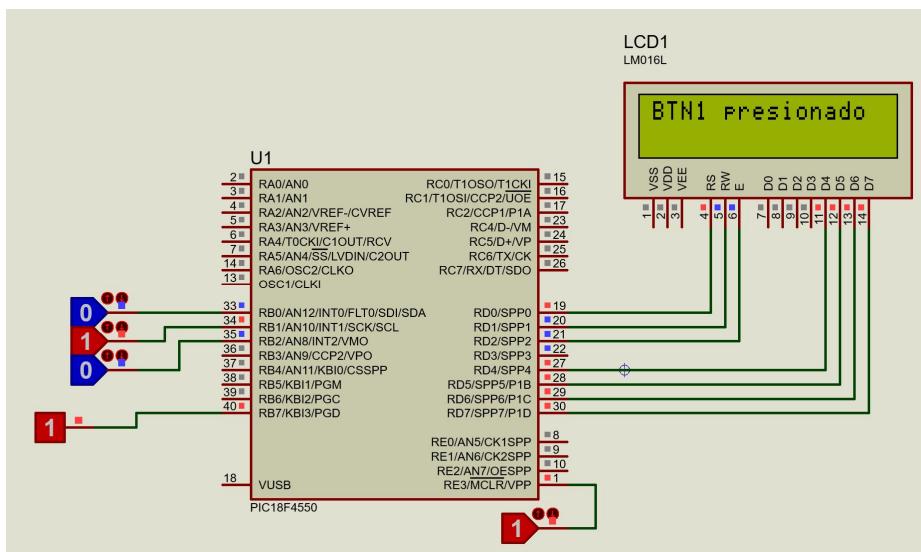
Código en XC8

```

10  #pragma config PLLDIV = 1           // 41 void lcd_init(void) {
11  #pragma config CPUDIV = OSC1_PLL2  // 42   TRISD = 0x00;           // Puerto I
12  #pragma config FOSC = XTPLL_XT  // 43   LCD_CONFIG();
13  #pragma config PWRT = ON          // 44   _delay_ms(15);
14  #pragma config BOR = OFF          // 45   BORRAR_LCD();
15  #pragma config BORV = 3           // 46   CURSOR_HOME();
16  #pragma config WDT = OFF          // 47   CURSOR_ONOFF(OFF);
17  #pragma config WDTPS = 32768      // 48 }
18  #pragma config CCP2MX = ON         // 49
19  #pragma config CCP2BDM = OFF       // 50 void main(void) {
20  #pragma config MCLRE = ON         // 51   configuracion();
21  #pragma config LVP = OFF          // 52   lcd_init();
22  #define _XTAL_FREQ 48000000UL    // 53   ESCRIBE_MENSAJE("Presiona boton",14);
23  #include <xc.h>                // 54   while (1) {
24  #include "LCD.h"                // 55     LATBbits.LB7 = 1;
25                                // 56     _delay_ms(100);
26                                // 57     LATBbits.LB7 = 0;
27                                // 58     _delay_ms(100);
28  void configuracion(void) {        // 59 }
29    //Aqui colocare las config.      // 60 void _interrupt(high_priority) Int0_ISR(void) {
30    RCONbits.IPEN = 1;             // 61   //POS_CURSOR(2,0);
31    INTCON3bits.INT1IF = 0;         // 62   if (INTCON2bits.INTEDG0 == 1) {
32    INTCON3bits.INT2IF = 0;         // 63     CURSOR_HOME();
33    INTCON3bits.INT2IE = 1;         // 64     ESCRIBE_MENSAJE("BTN0 presionado",15);
34    INTCON3bits.INT1IE = 1;         // 65     INTCON2bits.INTEDG0 = 0;
35    INTCON3bits.INT0IF = 1;         // 66   } else {
36    INTCONbits.GIEL = 1;           // 67     CURSOR_HOME();
37    INTCONbits.GIEH = 1;           // 68     ESCRIBE_MENSAJE("BTN0 soltado ",15);
38    TRISB = 0x7F;                 // 69     INTCON2bits.INTEDG0 = 1;
39  }                                // 70   INTCONbits.INT0IF = 0;
40                                // 71 }
41                                // 72 }
42                                // 73 }
43                                // 74 }
44                                // 75 }
45                                // 76 }
46                                // 77 void _interrupt(low_priority) Int1_ISR(void) {
47                                // 78   //POS_CURSOR(2,0);
48                                // 79   if (INTCON3bits.INT1IF == 1) {
49                                // 80     if (INTCON2bits.INTEDG1 == 1) {
50                                // 81       CURSOR_HOME();
51                                // 82       ESCRIBE_MENSAJE("BTN1 presionado",15);
52                                // 83       INTCON2bits.INTEDG1 = 0;
53                                // 84     } else {
54                                // 85       CURSOR_HOME();
55                                // 86       ESCRIBE_MENSAJE("BTN1 soltado ",15);
56                                // 87       INTCON2bits.INTEDG1 = 1;
57                                // 88     }
58                                // 89   }
59                                // 90   INTCON3bits.INT1IF = 0;
60                                // 91 }
61                                // 92   if (INTCON3bits.INT2IF == 1) {
62                                // 93     if (INTCON2bits.INTEDG2 == 1) {
63                                // 94       CURSOR_HOME();
64                                // 95       ESCRIBE_MENSAJE("BTN2 presionado",15);
65                                // 96       INTCON2bits.INTEDG2 = 0;
66                                // 97     } else {
67                                // 98       CURSOR_HOME();
68                                // 99       ESCRIBE_MENSAJE("BTN2 soltado ",15);
69                                // 100      INTCON2bits.INTEDG2 = 1;
70                                // 101    }
71                                // 102  }
72                                // 103 }
73                                // 104 }
74                                // 105 }
```

23

Simulación



24

Cuestionario:

25

Fin de la sesión

- Destinar una hora el sábado y una hora el domingo para repasar el curso.

26