

Microcontroladores

Semestre: 2021-2

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,

`-- delay-ms(100)` ✓

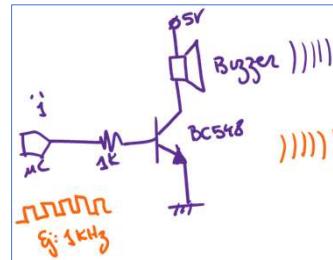
`unsigned int tiempo = 15;` ✗
`-- delay-ms(Tiempo);`

2

Preguntas previas:

- Dos tipos de buzzer:

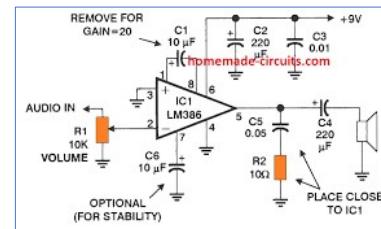
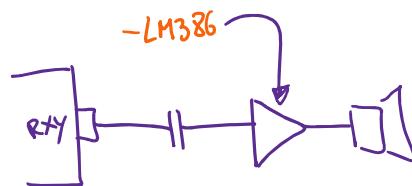
- Los activos con nivel lógico (le colocas energía DC y empiezan a sonar)
- Los que necesitan una señal oscilante para sonar (mismo parlante)



3

Preguntas previas:

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



4

Agenda

- Uso del sensor de temperatura LM35 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 $\pm 1/4^\circ\text{C}$ at room temperature and $\pm 3/4^\circ\text{C}$ 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 $\pm 1/4^\circ\text{C}$ typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

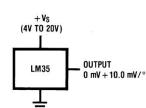
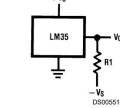


FIGURE 1. Basic Centigrade Temperature Sensor
(-55°C to +150°C)

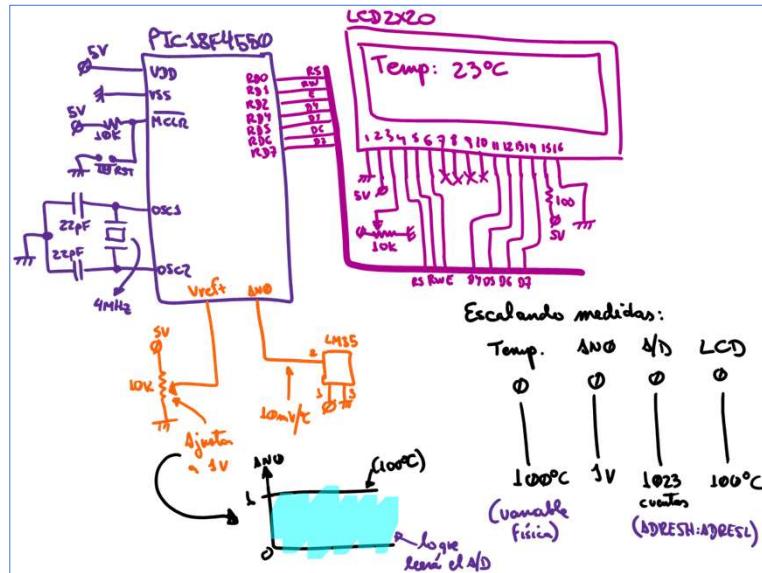


Choose $R_1 = -V_g/50 \mu\text{A}$
 $V_{OUT} = 1,500 \text{ mV at } +150^\circ\text{C}$
 $= +250 \text{ mV at } +25^\circ\text{C}$
 $= -550 \text{ mV at } -55^\circ\text{C}$

DS005516-4

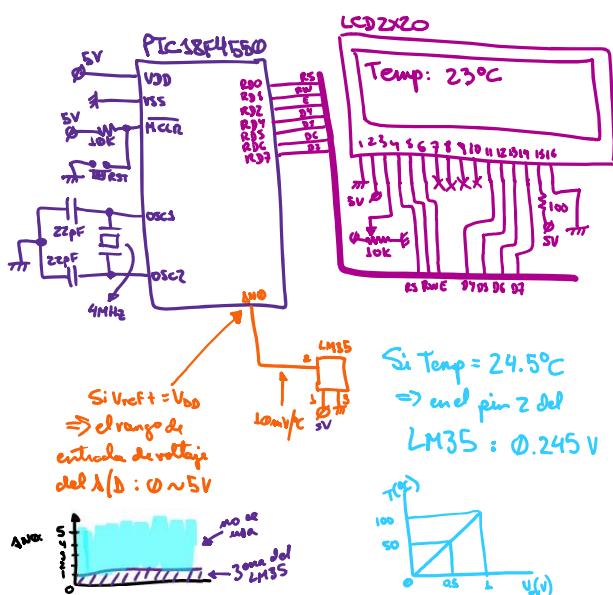
6

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 será:
 $\frac{5}{1024} = 4.88 \text{ mV}$

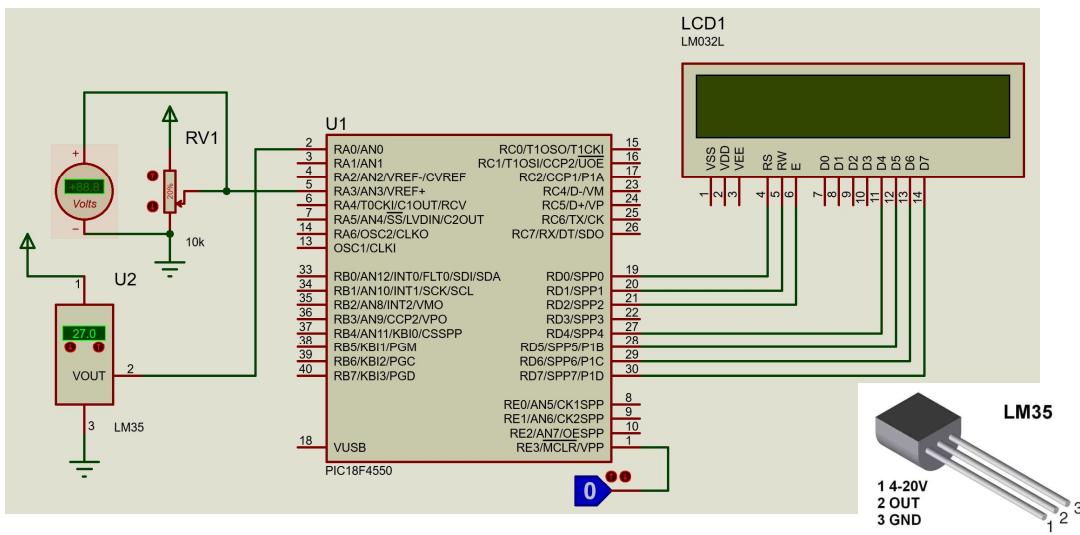
Si el LM35 tiene un rango de 0-1V,
cuantos escalones empleará?
- Solo 205 escalones

Si ajustamos el $Vref(+)$ a 1V entonces
el rango de entrada será 0-1V, al
valor del escalón será:
 $\frac{1}{1024} = 0.9765 \text{ mV}$

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

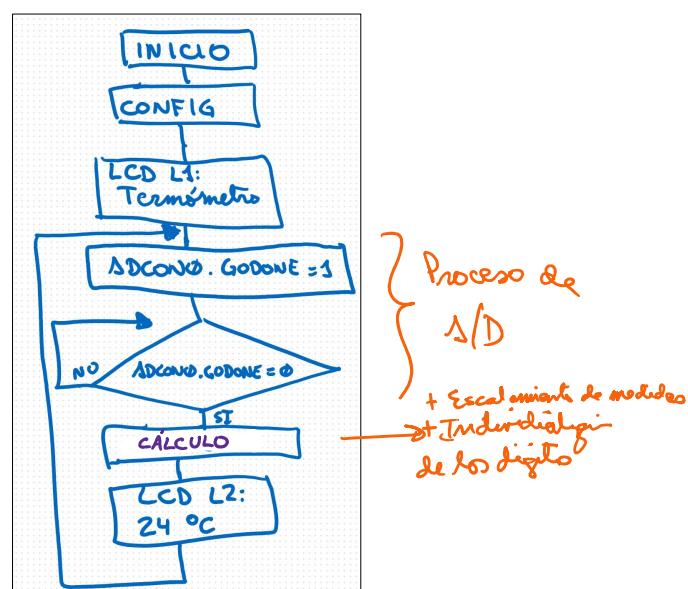
8

Interface del LM35 con el PIC18F4550



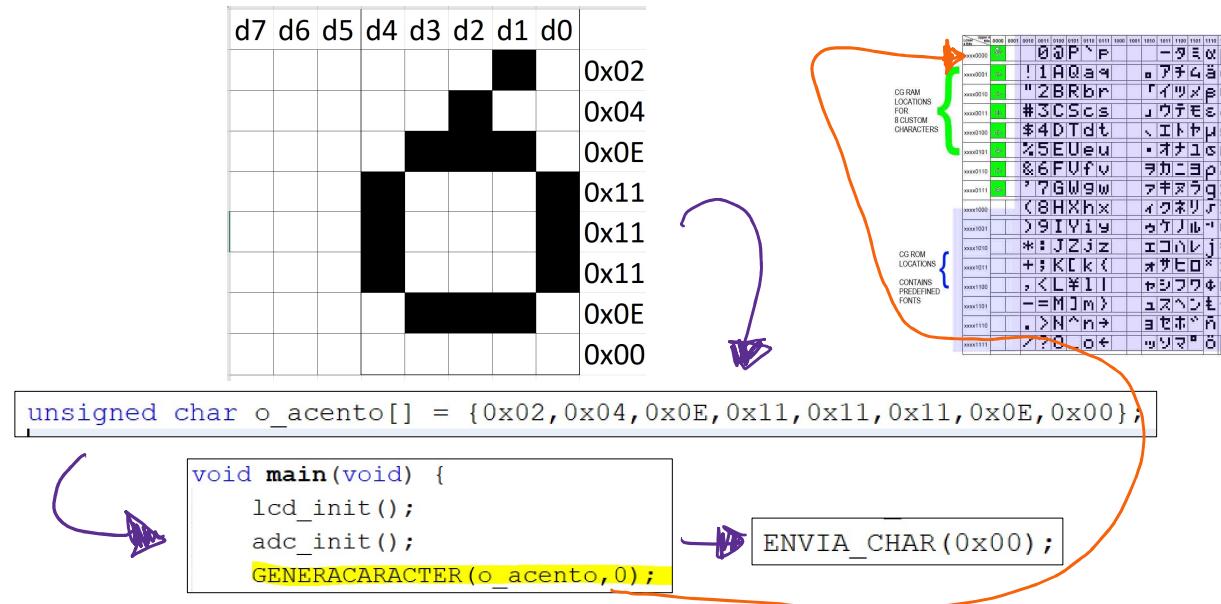
9

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 }

```

Float

Escalamiento

10.24?

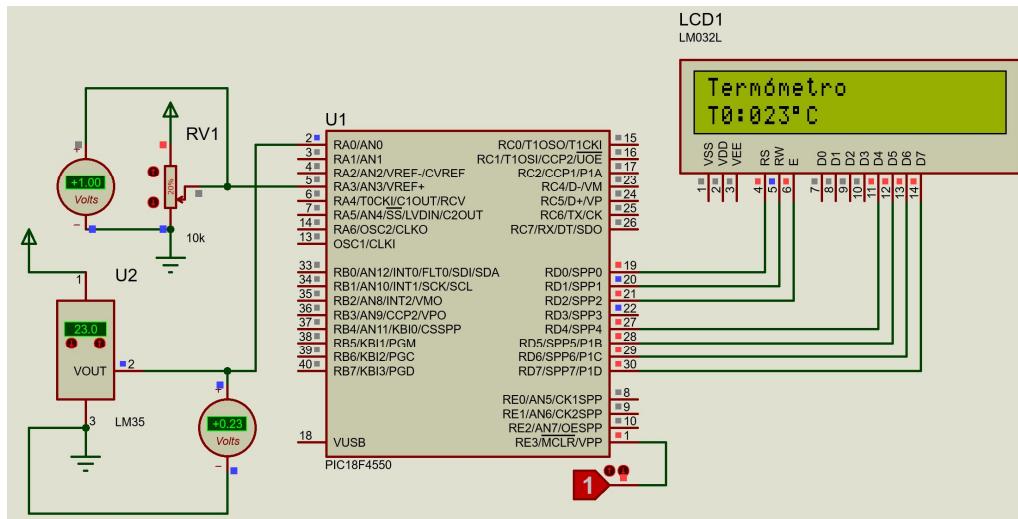
```

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         --delay_ms(500);
74     }
}

```

12

Simulación



Modificación al ejemplo anterior:

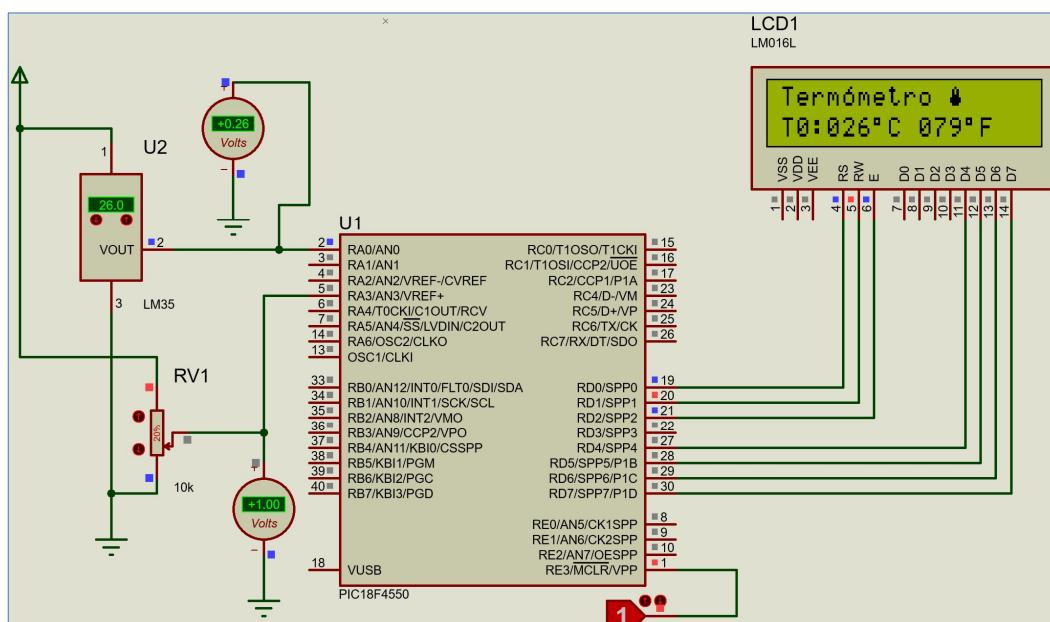
```

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 y XC* PIC Assembler
- 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*/
}
```

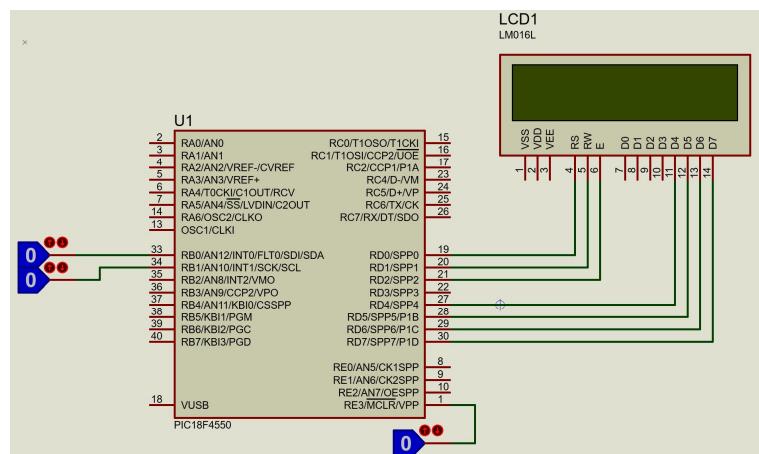
```
void __interrupt(high_priority) High_ISR(void)
{
    /*Aquí se escribe la rutina de alta prioridad*/
}

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

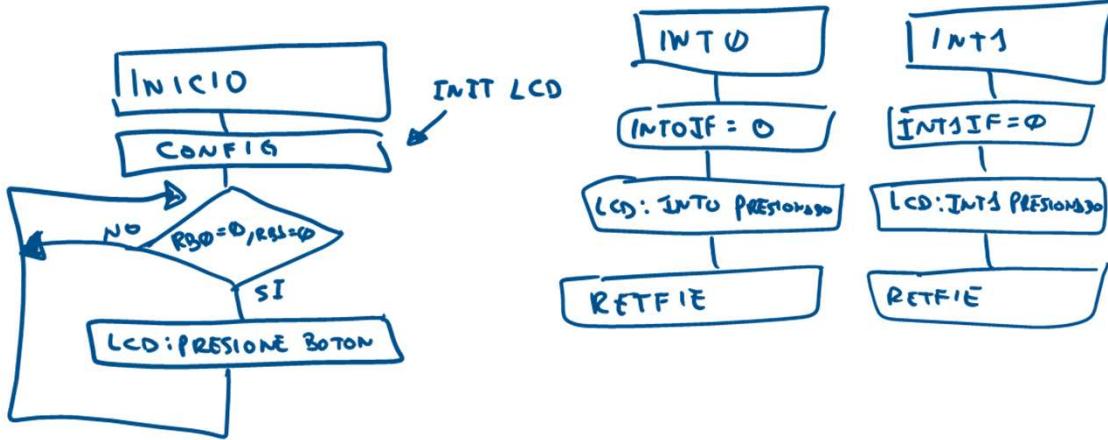
- En la pantalla se mostrará si se ha presionado las entradas INT0 o INT1.



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.IPEN = 1;           //Prioridades habilitadas
18     INTCONbits.INT0IE = 1;       //Int0 habilitada
19     INTCON3bits.INT1IE = 1;       //Int1 habilitada
20     INTCON3bits.INT1IP = 0;      //Int1 se vaya a low priority
21     INTCONbits.GIEH = 1;         //Interruptor global de ints HP activado
22     INTCONbits.GIEL = 1;         //Interruptor global de ints LP 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.INT1IP = 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     }

```

```

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     }

```

- Código con borrado de pantalla:

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

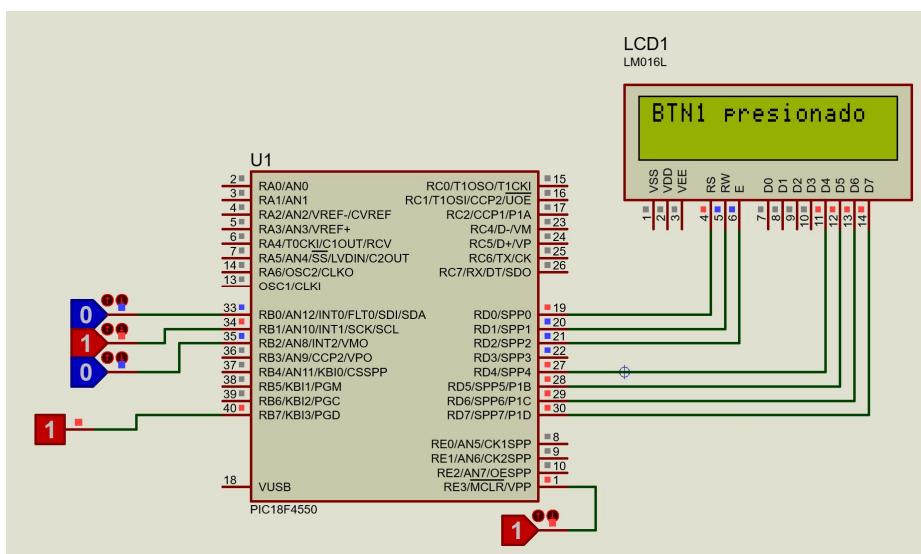
```

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 = XTPPLL_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 PBADEN = OFF // 50 void main(void) {
20 #pragma config MCLRE = ON // 51 configuracion();
21 #pragma config LVP = OFF // 52 lcd_init();
22 // 53 ESCRIBE_MENSAJE("Presiona boton",14);
23 // 54 while (1) {
24 // 55 LATBbits.LB7 = 1;
25 // 56 _delay_ms(100);
26 // 57 LATBbits.LB7 = 0;
27 // 58 _delay_ms(100);
28 // 59 }
29
30 void configuracion(void) {
31 //Aqui colocare las config
32 RCONbits.IPEN = 1; //INT1IF
33 INTCON3bits.INT1IP = 0; //INT2IF
34 INTCON3bits.INT2IP = 0;
35 INTCON3bits.INT2IE = 1;
36 INTCON3bits.INT1IE = 1; //INT0IE
37 INTCONbits.INT0E = 1;
38 INTCONbits.GIEL = 1;
39 INTCONbits.GIEH = 1;
40 TRISB = 0x7F;
41 }
42
43 void _interrupt(low_priority) Int1_ISR(void) {
44 //POS_CURSOR(2,0);
45 if (INTCON2bits.INTEDG1 == 1) {
46 CURSOR_HOME();
47 ESCRIBE_MENSAJE("BTN1 presionado",15);
48 INTCON2bits.INTEDG1 = 0;
49 }
50 else {
51 CURSOR_HOME();
52 ESCRIBE_MENSAJE("BTN1 soltado ",15);
53 INTCON2bits.INTEDG1 = 1;
54 }
55 INTCON3bits.INT1IF = 0;
56
57 void _interrupt(high_priority) Int0_ISR(void) {
58 //POS_CURSOR(2,0);
59 if (INTCON2bits.INTEDG0 == 1) {
60 CURSOR_HOME();
61 ESCRIBE_MENSAJE("BTN0 presionado",15);
62 INTCON2bits.INTEDG0 = 0;
63 }
64 else {
65 CURSOR_HOME();
66 ESCRIBE_MENSAJE("BTN0 soltado ",15);
67 INTCON2bits.INTEDG0 = 1;
68 }
69 INTCON3bits.INT2IF = 0;
70
71 void _interrupt(low_priority) Int2_ISR(void) {
72 //POS_CURSOR(2,0);
73 if (INTCON3bits.INTEDG2 == 1) {
74 CURSOR_HOME();
75 ESCRIBE_MENSAJE("BTN2 presionado",15);
76 INTCON3bits.INTEDG2 = 0;
77 }
78 else {
79 CURSOR_HOME();
80 ESCRIBE_MENSAJE("BTN2 soltado ",15);
81 INTCON3bits.INT2IF = 1;
82 }
83 INTCON3bits.INT2IF = 0;
84
85 void _interrupt(low_priority) Int0_ISR(void) {
86 //POS_CURSOR(2,0);
87 if (INTCON3bits.INT1IF == 1) {
88 if (INTCON2bits.INTEDG1 == 1) {
89 CURSOR_HOME();
90 ESCRIBE_MENSAJE("BTN1 presionado",15);
91 INTCON2bits.INTEDG1 = 0;
92 }
93 else {
94 CURSOR_HOME();
95 ESCRIBE_MENSAJE("BTN1 soltado ",15);
96 INTCON2bits.INTEDG1 = 1;
97 }
98 INTCON3bits.INT1IF = 0;
99
100 void _interrupt(low_priority) Int1_ISR(void) {
101 //POS_CURSOR(2,0);
102 if (INTCON2bits.INTEDG0 == 1) {
103 CURSOR_HOME();
104 ESCRIBE_MENSAJE("BTN0 presionado",15);
105 INTCON2bits.INTEDG0 = 0;
106 }
107 else {
108 CURSOR_HOME();
109 ESCRIBE_MENSAJE("BTN0 soltado ",15);
110 INTCON2bits.INTEDG0 = 1;
111 }
112 INTCON3bits.INT2IF = 0;
113 }
114
115 void _interrupt(low_priority) Int2_ISR(void) {
116 //POS_CURSOR(2,0);
117 if (INTCON3bits.INT2IF == 1) {
118 if (INTCON2bits.INTEDG2 == 1) {
119 CURSOR_HOME();
120 ESCRIBE_MENSAJE("BTN2 presionado",15);
121 INTCON2bits.INTEDG2 = 0;
122 }
123 else {
124 CURSOR_HOME();
125 ESCRIBE_MENSAJE("BTN2 soltado ",15);
126 INTCON2bits.INTEDG2 = 1;
127 }
128 INTCON3bits.INT2IF = 0;
129 }
130
131 void _interrupt(low_priority) Int0_ISR(void) {
132 //POS_CURSOR(2,0);
133 if (INTCON3bits.INT0IF == 1) {
134 CURSOR_HOME();
135 ESCRIBE_MENSAJE("BTN0 presionado",15);
136 INTCON3bits.INT0IF = 0;
137 }
138 else {
139 CURSOR_HOME();
140 ESCRIBE_MENSAJE("BTN0 soltado ",15);
141 INTCON3bits.INT0IF = 1;
142 }
143 INTCON3bits.INT0IF = 0;
144 }
145
146 void _interrupt(low_priority) Int1_ISR(void) {
147 //POS_CURSOR(2,0);
148 if (INTCON3bits.INT1IF == 1) {
149 CURSOR_HOME();
150 ESCRIBE_MENSAJE("BTN1 presionado",15);
151 INTCON3bits.INT1IF = 0;
152 }
153 else {
154 CURSOR_HOME();
155 ESCRIBE_MENSAJE("BTN1 soltado ",15);
156 INTCON3bits.INT1IF = 1;
157 }
158 INTCON3bits.INT1IF = 0;
159 }
160
161 void _interrupt(low_priority) Int2_ISR(void) {
162 //POS_CURSOR(2,0);
163 if (INTCON3bits.INT2IF == 1) {
164 CURSOR_HOME();
165 ESCRIBE_MENSAJE("BTN2 presionado",15);
166 INTCON3bits.INT2IF = 0;
167 }
168 else {
169 CURSOR_HOME();
170 ESCRIBE_MENSAJE("BTN2 soltado ",15);
171 INTCON3bits.INT2IF = 1;
172 }
173 INTCON3bits.INT2IF = 0;
174 }
175
176 void _interrupt(low_priority) Int0_ISR(void) {
177 //POS_CURSOR(2,0);
178 if (INTCON3bits.INT0IF == 1) {
179 CURSOR_HOME();
180 ESCRIBE_MENSAJE("BTN0 presionado",15);
181 INTCON3bits.INT0IF = 0;
182 }
183 else {
184 CURSOR_HOME();
185 ESCRIBE_MENSAJE("BTN0 soltado ",15);
186 INTCON3bits.INT0IF = 1;
187 }
188 INTCON3bits.INT0IF = 0;
189 }
190
191 void _interrupt(low_priority) Int1_ISR(void) {
192 //POS_CURSOR(2,0);
193 if (INTCON3bits.INT1IF == 1) {
194 CURSOR_HOME();
195 ESCRIBE_MENSAJE("BTN1 presionado",15);
196 INTCON3bits.INT1IF = 0;
197 }
198 else {
199 CURSOR_HOME();
200 ESCRIBE_MENSAJE("BTN1 soltado ",15);
201 INTCON3bits.INT1IF = 1;
202 }
203 INTCON3bits.INT1IF = 0;
204 }
205
206 void _interrupt(low_priority) Int2_ISR(void) {
207 //POS_CURSOR(2,0);
208 if (INTCON3bits.INT2IF == 1) {
209 CURSOR_HOME();
210 ESCRIBE_MENSAJE("BTN2 presionado",15);
211 INTCON3bits.INT2IF = 0;
212 }
213 else {
214 CURSOR_HOME();
215 ESCRIBE_MENSAJE("BTN2 soltado ",15);
216 INTCON3bits.INT2IF = 1;
217 }
218 INTCON3bits.INT2IF = 0;
219 }
220
221 void _interrupt(low_priority) Int0_ISR(void) {
222 //POS_CURSOR(2,0);
223 if (INTCON3bits.INT0IF == 1) {
224 CURSOR_HOME();
225 ESCRIBE_MENSAJE("BTN0 presionado",15);
226 INTCON3bits.INT0IF = 0;
227 }
228 else {
229 CURSOR_HOME();
230 ESCRIBE_MENSAJE("BTN0 soltado ",15);
231 INTCON3bits.INT0IF = 1;
232 }
233 INTCON3bits.INT0IF = 0;
234 }
235
236 void _interrupt(low_priority) Int1_ISR(void) {
237 //POS_CURSOR(2,0);
238 if (INTCON3bits.INT1IF == 1) {
239 CURSOR_HOME();
240 ESCRIBE_MENSAJE("BTN1 presionado",15);
241 INTCON3bits.INT1IF = 0;
242 }
243 else {
244 CURSOR_HOME();
245 ESCRIBE_MENSAJE("BTN1 soltado ",15);
246 INTCON3bits.INT1IF = 1;
247 }
248 INTCON3bits.INT1IF = 0;
249 }
250
251 void _interrupt(low_priority) Int2_ISR(void) {
252 //POS_CURSOR(2,0);
253 if (INTCON3bits.INT2IF == 1) {
254 CURSOR_HOME();
255 ESCRIBE_MENSAJE("BTN2 presionado",15);
256 INTCON3bits.INT2IF = 0;
257 }
258 else {
259 CURSOR_HOME();
260 ESCRIBE_MENSAJE("BTN2 soltado ",15);
261 INTCON3bits.INT2IF = 1;
262 }
263 INTCON3bits.INT2IF = 0;
264 }
265
266 void _interrupt(low_priority) Int0_ISR(void) {
267 //POS_CURSOR(2,0);
268 if (INTCON3bits.INT0IF == 1) {
269 CURSOR_HOME();
270 ESCRIBE_MENSAJE("BTN0 presionado",15);
271 INTCON3bits.INT0IF = 0;
272 }
273 else {
274 CURSOR_HOME();
275 ESCRIBE_MENSAJE("BTN0 soltado ",15);
276 INTCON3bits.INT0IF = 1;
277 }
278 INTCON3bits.INT0IF = 0;
279 }
280
281 void _interrupt(low_priority) Int1_ISR(void) {
282 //POS_CURSOR(2,0);
283 if (INTCON3bits.INT1IF == 1) {
284 CURSOR_HOME();
285 ESCRIBE_MENSAJE("BTN1 presionado",15);
286 INTCON3bits.INT1IF = 0;
287 }
288 else {
289 CURSOR_HOME();
290 ESCRIBE_MENSAJE("BTN1 soltado ",15);
291 INTCON3bits.INT1IF = 1;
292 }
293 INTCON3bits.INT1IF = 0;
294 }
295
296 void _interrupt(low_priority) Int2_ISR(void) {
297 //POS_CURSOR(2,0);
298 if (INTCON3bits.INT2IF == 1) {
299 CURSOR_HOME();
300 ESCRIBE_MENSAJE("BTN2 presionado",15);
301 INTCON3bits.INT2IF = 0;
302 }
303 else {
304 CURSOR_HOME();
305 ESCRIBE_MENSAJE("BTN2 soltado ",15);
306 INTCON3bits.INT2IF = 1;
307 }
308 INTCON3bits.INT2IF = 0;
309 }
310
311 void _interrupt(low_priority) Int0_ISR(void) {
312 //POS_CURSOR(2,0);
313 if (INTCON3bits.INT0IF == 1) {
314 CURSOR_HOME();
315 ESCRIBE_MENSAJE("BTN0 presionado",15);
316 INTCON3bits.INT0IF = 0;
317 }
318 else {
319 CURSOR_HOME();
320 ESCRIBE_MENSAJE("BTN0 soltado ",15);
321 INTCON3bits.INT0IF = 1;
322 }
323 INTCON3bits.INT0IF = 0;
324 }
325
326 void _interrupt(low_priority) Int1_ISR(void) {
327 //POS_CURSOR(2,0);
328 if (INTCON3bits.INT1IF == 1) {
329 CURSOR_HOME();
330 ESCRIBE_MENSAJE("BTN1 presionado",15);
331 INTCON3bits.INT1IF = 0;
332 }
333 else {
334 CURSOR_HOME();
335 ESCRIBE_MENSAJE("BTN1 soltado ",15);
336 INTCON3bits.INT1IF = 1;
337 }
338 INTCON3bits.INT1IF = 0;
339 }
340
341 void _interrupt(low_priority) Int2_ISR(void) {
342 //POS_CURSOR(2,0);
343 if (INTCON3bits.INT2IF == 1) {
344 CURSOR_HOME();
345 ESCRIBE_MENSAJE("BTN2 presionado",15);
346 INTCON3bits.INT2IF = 0;
347 }
348 else {
349 CURSOR_HOME();
350 ESCRIBE_MENSAJE("BTN2 soltado ",15);
351 INTCON3bits.INT2IF = 1;
352 }
353 INTCON3bits.INT2IF = 0;
354 }
355
356 void _interrupt(low_priority) Int0_ISR(void) {
357 //POS_CURSOR(2,0);
358 if (INTCON3bits.INT0IF == 1) {
359 CURSOR_HOME();
360 ESCRIBE_MENSAJE("BTN0 presionado",15);
361 INTCON3bits.INT0IF = 0;
362 }
363 else {
364 CURSOR_HOME();
365 ESCRIBE_MENSAJE("BTN0 soltado ",15);
366 INTCON3bits.INT0IF = 1;
367 }
368 INTCON3bits.INT0IF = 0;
369 }
370
371 void _interrupt(low_priority) Int1_ISR(void) {
372 //POS_CURSOR(2,0);
373 if (INTCON3bits.INT1IF == 1) {
374 CURSOR_HOME();
375 ESCRIBE_MENSAJE("BTN1 presionado",15);
376 INTCON3bits.INT1IF = 0;
377 }
378 else {
379 CURSOR_HOME();
380 ESCRIBE_MENSAJE("BTN1 soltado ",15);
381 INTCON3bits.INT1IF = 1;
382 }
383 INTCON3bits.INT1IF = 0;
384 }
385
386 void _interrupt(low_priority) Int2_ISR(void) {
387 //POS_CURSOR(2,0);
388 if (INTCON3bits.INT2IF == 1) {
389 CURSOR_HOME();
390 ESCRIBE_MENSAJE("BTN2 presionado",15);
391 INTCON3bits.INT2IF = 0;
392 }
393 else {
394 CURSOR_HOME();
395 ESCRIBE_MENSAJE("BTN2 soltado ",15);
396 INTCON3bits.INT2IF = 1;
397 }
398 INTCON3bits.INT2IF = 0;
399 }
400
401 void _interrupt(low_priority) Int0_ISR(void) {
402 //POS_CURSOR(2,0);
403 if (INTCON3bits.INT0IF == 1) {
404 CURSOR_HOME();
405 ESCRIBE_MENSAJE("BTN0 presionado",15);
406 INTCON3bits.INT0IF = 0;
407 }
408 else {
409 CURSOR_HOME();
410 ESCRIBE_MENSAJE("BTN0 soltado ",15);
411 INTCON3bits.INT0IF = 1;
412 }
413 INTCON3bits.INT0IF = 0;
414 }
415
416 void _interrupt(low_priority) Int1_ISR(void) {
417 //POS_CURSOR(2,0);
418 if (INTCON3bits.INT1IF == 1) {
419 CURSOR_HOME();
420 ESCRIBE_MENSAJE("BTN1 presionado",15);
421 INTCON3bits.INT1IF = 0;
422 }
423 else {
424 CURSOR_HOME();
425 ESCRIBE_MENSAJE("BTN1 soltado ",15);
426 INTCON3bits.INT1IF = 1;
427 }
428 INTCON3bits.INT1IF = 0;
429 }
430
431 void _interrupt(low_priority) Int2_ISR(void) {
432 //POS_CURSOR(2,0);
433 if (INTCON3bits.INT2IF == 1) {
434 CURSOR_HOME();
435 ESCRIBE_MENSAJE("BTN2 presionado",15);
436 INTCON3bits.INT2IF = 0;
437 }
438 else {
439 CURSOR_HOME();
440 ESCRIBE_MENSAJE("BTN2 soltado ",15);
441 INTCON3bits.INT2IF = 1;
442 }
443 INTCON3bits.INT2IF = 0;
444 }
445
446 void _interrupt(low_priority) Int0_ISR(void) {
447 //POS_CURSOR(2,0);
448 if (INTCON3bits.INT0IF == 1) {
449 CURSOR_HOME();
450 ESCRIBE_MENSAJE("BTN0 presionado",15);
451 INTCON3bits.INT0IF = 0;
452 }
453 else {
454 CURSOR_HOME();
455 ESCRIBE_MENSAJE("BTN0 soltado ",15);
456 INTCON3bits.INT0IF = 1;
457 }
458 INTCON3bits.INT0IF = 0;
459 }
460
461 void _interrupt(low_priority) Int1_ISR(void) {
462 //POS_CURSOR(2,0);
463 if (INTCON3bits.INT1IF == 1) {
464 CURSOR_HOME();
465 ESCRIBE_MENSAJE("BTN1 presionado",15);
466 INTCON3bits.INT1IF = 0;
467 }
468 else {
469 CURSOR_HOME();
470 ESCRIBE_MENSAJE("BTN1 soltado ",15);
471 INTCON3bits.INT1IF = 1;
472 }
473 INTCON3bits.INT1IF = 0;
474 }
475
476 void _interrupt(low_priority) Int2_ISR(void) {
477 //POS_CURSOR(2,0);
478 if (INTCON3bits.INT2IF == 1) {
479 CURSOR_HOME();
480 ESCRIBE_MENSAJE("BTN2 presionado",15);
481 INTCON3bits.INT2IF = 0;
482 }
483 else {
484 CURSOR_HOME();
485 ESCRIBE_MENSAJE("BTN2 soltado ",15);
486 INTCON3bits.INT2IF = 1;
487 }
488 INTCON3bits.INT2IF = 0;
489 }
490
491 void _interrupt(low_priority) Int0_ISR(void) {
492 //POS_CURSOR(2,0);
493 if (INTCON3bits.INT0IF == 1) {
494 CURSOR_HOME();
495 ESCRIBE_MENSAJE("BTN0 presionado",15);
496 INTCON3bits.INT0IF = 0;
497 }
498 else {
499 CURSOR_HOME();
500 ESCRIBE_MENSAJE("BTN0 soltado ",15);
501 INTCON3bits.INT0IF = 1;
502 }
503 INTCON3bits.INT0IF = 0;
504 }
505
506 void _interrupt(low_priority) Int1_ISR(void) {
507 //POS_CURSOR(2,0);
508 if (INTCON3bits.INT1IF == 1) {
509 CURSOR_HOME();
510 ESCRIBE_MENSAJE("BTN1 presionado",15);
511 INTCON3bits.INT1IF = 0;
512 }
513 else {
514 CURSOR_HOME();
515 ESCRIBE_MENSAJE("BTN1 soltado ",15);
516 INTCON3bits.INT1IF = 1;
517 }
518 INTCON3bits.INT1IF = 0;
519 }
520
521 void _interrupt(low_priority) Int2_ISR(void) {
522 //POS_CURSOR(2,0);
523 if (INTCON3bits.INT2IF == 1) {
524 CURSOR_HOME();
525 ESCRIBE_MENSAJE("BTN2 presionado",15);
526 INTCON3bits.INT2IF = 0;
527 }
528 else {
529 CURSOR_HOME();
530 ESCRIBE_MENSAJE("BTN2 soltado ",15);
531 INTCON3bits.INT2IF = 1;
532 }
533 INTCON3bits.INT2IF = 0;
534 }
535
536 void _interrupt(low_priority) Int0_ISR(void) {
537 //POS_CURSOR(2,0);
538 if (INTCON3bits.INT0IF == 1) {
539 CURSOR_HOME();
540 ESCRIBE_MENSAJE("BTN0 presionado",15);
541 INTCON3bits.INT0IF = 0;
542 }
543 else {
544 CURSOR_HOME();
545 ESCRIBE_MENSAJE("BTN0 soltado ",15);
546 INTCON3bits.INT0IF = 1;
547 }
548 INTCON3bits.INT0IF = 0;
549 }
550
551 void _interrupt(low_priority) Int1_ISR(void) {
552 //POS_CURSOR(2,0);
553 if (INTCON3bits.INT1IF == 1) {
554 CURSOR_HOME();
555 ESCRIBE_MENSAJE("BTN1 presionado",15);
556 INTCON3bits.INT1IF = 0;
557 }
558 else {
559 CURSOR_HOME();
560 ESCRIBE_MENSAJE("BTN1 soltado ",15);
561 INTCON3bits.INT1IF = 1;
562 }
563 INTCON3bits.INT1IF = 0;
564 }
565
566 void _interrupt(low_priority) Int2_ISR(void) {
567 //POS_CURSOR(2,0);
568 if (INTCON3bits.INT2IF == 1) {
569 CURSOR_HOME();
570 ESCRIBE_MENSAJE("BTN2 presionado",15);
571 INTCON3bits.INT2IF = 0;
572 }
573 else {
574 CURSOR_HOME();
575 ESCRIBE_MENSAJE("BTN2 soltado ",15);
576 INTCON3bits.INT2IF = 1;
577 }
578 INTCON3bits.INT2IF = 0;
579 }
580
581 void _interrupt(low_priority) Int0_ISR(void) {
582 //POS_CURSOR(2,0);
583 if (INTCON3bits.INT0IF == 1) {
584 CURSOR_HOME();
585 ESCRIBE_MENSAJE("BTN0 presionado",15);
586 INTCON3bits.INT0IF = 0;
587 }
588 else {
589 CURSOR_HOME();
590 ESCRIBE_MENSAJE("BTN0 soltado ",15);
591 INTCON3bits.INT0IF = 1;
592 }
593 INTCON3bits.INT0IF = 0;
594 }
595
596 void _interrupt(low_priority) Int1_ISR(void) {
597 //POS_CURSOR(2,0);
598 if (INTCON3bits.INT1IF == 1) {
599 CURSOR_HOME();
600 ESCRIBE_MENSAJE("BTN1 presionado",15);
601 INTCON3bits.INT1IF = 0;
602 }
603 else {
604 CURSOR_HOME();
605 ESCRIBE_MENSAJE("BTN1 soltado ",15);
606 INTCON3bits.INT1IF = 1;
607 }
608 INTCON3bits.INT1IF = 0;
609 }
610
611 void _interrupt(low_priority) Int2_ISR(void) {
612 //POS_CURSOR(2,0);
613 if (INTCON3bits.INT2IF == 1) {
614 CURSOR_HOME();
615 ESCRIBE_MENSAJE("BTN2 presionado",15);
616 INTCON3bits.INT2IF = 0;
617 }
618 else {
619 CURSOR_HOME();
620 ESCRIBE_MENSAJE("BTN2 soltado ",15);
621 INTCON3bits.INT2IF = 1;
622 }
623 INTCON3bits.INT2IF = 0;
624 }
625
626 void _interrupt(low_priority) Int0_ISR(void) {
627 //POS_CURSOR(2,0);
628 if (INTCON3bits.INT0IF == 1) {
629 CURSOR_HOME();
630 ESCRIBE_MENSAJE("BTN0 presionado",15);
631 INTCON3bits.INT0IF = 0;
632 }
633 else {
634 CURSOR_HOME();
635 ESCRIBE_MENSAJE("BTN0 soltado ",15);
636 INTCON3bits.INT0IF = 1;
637 }
638 INTCON3bits.INT0IF = 0;
639 }
640
641 void _interrupt(low_priority) Int1_ISR(void) {
642 //POS_CURSOR(2,0);
643 if (INTCON3bits.INT1IF == 1) {
644 CURSOR_HOME();
645 ESCRIBE_MENSAJE("BTN1 presionado",15);
646 INTCON3bits.INT1IF = 0;
647 }
648 else {
649 CURSOR_HOME();
650 ESCRIBE_MENSAJE("BTN1 soltado ",15);
651 INTCON3bits.INT1IF = 1;
652 }
653 INTCON3bits.INT1IF = 0;
654 }
655
656 void _interrupt(low_priority) Int2_ISR(void) {
657 //POS_CURSOR(2,0);
658 if (INTCON3bits.INT2IF == 1) {
659 CURSOR_HOME();
660 ESCRIBE_MENSAJE("BTN2 presionado",15);
661 INTCON3bits.INT2IF = 0;
662 }
663 else {
664 CURSOR_HOME();
665 ESCRIBE_MENSAJE("BTN2 soltado ",15);
666 INTCON3bits.INT2IF = 1;
667 }
668 INTCON3bits.INT2IF = 0;
669 }
670
671 void _interrupt(low_priority) Int0_ISR(void) {
672 //POS_CURSOR(2,0);
673 if (INTCON3bits.INT0IF == 1) {
674 CURSOR_HOME();
675 ESCRIBE_MENSAJE("BTN0 presionado",15);
676 INTCON3bits.INT0IF = 0;
677 }
678 else {
679 CURSOR_HOME();
680 ESCRIBE_MENSAJE("BTN0 soltado ",15);
681 INTCON3bits.INT0IF = 1;
682 }
683 INTCON3bits.INT0IF = 0;
684 }
685
686 void _interrupt(low_priority) Int1_ISR(void) {
687 //POS_CURSOR(2,0);
688 if (INTCON3bits.INT1IF == 1) {
689 CURSOR_HOME();
690 ESCRIBE_MENSAJE("BTN1 presionado",15);
691 INTCON3bits.INT1IF = 0;
692 }
693 else {
694 CURSOR_HOME();
695 ESCRIBE_MENSAJE("BTN1 soltado ",15);
696 INTCON3bits.INT1IF = 1;
697 }
698 INTCON3bits.INT1IF = 0;
699 }
700
701 void _interrupt(low_priority) Int2_ISR(void) {
702 //POS_CURSOR(2,0);
703 if (INTCON3bits.INT2IF == 1) {
704 CURSOR_HOME();
705 ESCRIBE_MENSAJE("BTN2 presionado",15);
706 INTCON3bits.INT2IF = 0;
707 }
708 else {
709 CURSOR_HOME();
710 ESCRIBE_MENSAJE("BTN2 soltado ",15);
711 INTCON3bits.INT2IF = 1;
712 }
713 INTCON3bits.INT2IF = 0;
714 }
715
716 void _interrupt(low_priority) Int0_ISR(void) {
717 //POS_CURSOR(2,0);
718 if (INTCON3bits.INT0IF == 1) {
719 CURSOR_HOME();
720 ESCRIBE_MENSAJE("BTN0 presionado",15);
721 INTCON3bits.INT0IF = 0;
722 }
723 else {
724 CURSOR_HOME();
725 ESCRIBE_MENSAJE("BTN0 soltado ",15);
726 INTCON3bits.INT0IF = 1;
727 }
728 INTCON3bits.INT0IF = 0;
729 }
730
731 void _interrupt(low_priority) Int1_ISR(void) {
732 //POS_CURSOR(2,0);
733 if (INTCON3bits.INT1IF == 1) {
734 CURSOR_HOME();
735 ESCRIBE_MENSAJE("BTN1 presionado",15);
736 INTCON3bits.INT1IF = 0;
737 }
738 else {
739 CURSOR_HOME();
740 ESCRIBE_MENSAJE("BTN1 soltado ",15);
741 INTCON3bits.INT1IF = 1;
742 }
743 INTCON3bits.INT1IF = 0;
744 }
745
746 void _interrupt(low_priority) Int2_ISR(void) {
747 //POS_CURSOR(2,0);
748 if (INTCON3bits.INT2IF == 1) {
749 CURSOR_HOME();
750 ESCRIBE_MENSAJE("BTN2 presionado",15);
751 INTCON3bits.INT2IF = 0;
752 }
753 else {
754 CURSOR_HOME();
755 ESCRIBE_MENSAJE("BTN2 soltado ",15);
756 INTCON3bits.INT2IF = 1;
757 }
758 INTCON3bits.INT2IF = 0;
759 }
760
761 void _interrupt(low_priority) Int0_ISR(void) {
762 //POS_CURSOR(2,0);
763 if (INTCON3bits.INT0IF == 1) {
764 CURSOR_HOME();
765 ESCRIBE_MENSAJE("BTN0 presionado",15);
766 INTCON3bits.INT0IF = 0;
767 }
768 else {
769 CURSOR_HOME();
770 ESCRIBE_MENSAJE("BTN0 soltado ",15);
771 INTCON3bits.INT0IF = 1;
772 }
773 INTCON3bits.INT0IF = 0;
774 }
775
776 void _interrupt(low_priority) Int1_ISR(void) {
777 //POS_CURSOR(2,0);
778 if (INTCON3bits.INT1IF == 1) {
779 CURSOR_HOME();
780 ESCRIBE_MENSAJE("BTN1 presionado",15);
781 INTCON3bits.INT1IF = 0;
782 }
783 else {
784 CURSOR_HOME();
785 ESCRIBE_MENSAJE("BTN1 soltado ",15);
786 INTCON3bits.INT1IF = 1;
787 }
788 INTCON3bits.INT1IF = 0;
789 }
790
791 void _interrupt(low_priority) Int2_ISR(void) {
792 //POS_CURSOR(2,0);
793 if (INTCON3bits.INT2IF == 1) {
794 CURSOR_HOME();
795 ESCRIBE_MENSAJE("BTN2 presionado",15);
796 INTCON3bits.INT2IF = 0;
797 }
798 else {
799 CURSOR_HOME();
800 ESCRIBE_MENSAJE("BTN2 soltado ",15);
801 INTCON3bits.INT2IF = 1;
802 }
803 INTCON3bits.INT2IF = 0;
804 }
805
806 void _interrupt(low_priority) Int0_ISR(void) {
807 //POS_CURSOR(2,0);
808 if (INTCON3bits.INT0IF == 1) {
809 CURSOR_HOME();
810 ESCRIBE_MENSAJE("BTN0 presionado",15);
811 INTCON3bits.INT0IF = 0;
812 }
813 else {
814 CURSOR_HOME();
815 ESCRIBE_MENSAJE("BTN0 soltado ",15);
816 INTCON3bits.INT0IF = 1;
817 }
818 INTCON3bits.INT0IF = 0;
819 }
820
821 void _interrupt(low_priority) Int1_ISR(void) {
822 //POS_CURSOR(2,0);
823 if (INTCON3bits.INT1IF == 1) {
824 CURSOR_HOME();
825 ESCRIBE_MENSAJE("BTN1 presionado",15);
826 INTCON3bits.INT1IF = 0;
827 }
828 else {
829 CURSOR_HOME();
830 ESCRIBE_MENSAJE("BTN1 soltado ",15);
831 INTCON3bits.INT1IF = 1;
832 }
833 INTCON3bits.INT1IF = 0;
834 }
835
836 void _interrupt(low_priority) Int2_ISR(void) {
837 //POS_CURSOR(2,0);
838 if (INTCON3bits.INT2IF == 1) {
839 CURSOR_HOME();
840 ESCRIBE_MENSAJE("BTN2 presionado",15);
841 INTCON3bits.INT2IF = 0;
842 }
843 else {
844 CURSOR_HOME();
845 ESCRIBE_MENSAJE("BTN2 soltado ",15);
846 INTCON3bits.INT2IF = 1;
847 }
848 INTCON3bits.INT2IF = 0;
849 }
850
851 void _interrupt(low_priority) Int0_ISR(void) {
852 //POS_CURSOR(2,0);
853 if (INTCON3bits.INT0IF == 1) {
854 CURSOR_HOME();
855 ESCRIBE_MENSAJE("BTN0 presionado",15);
856 INTCON3bits.INT0IF = 0;
857 }
858 else {
859 CURSOR_HOME();
860 ESCRIBE_MENSAJE("BTN0 soltado ",15);
861 INTCON3bits.INT0IF = 1;
862 }
863 INTCON3bits.INT0IF = 0;
864 }
865
866 void _interrupt(low_priority) Int1_ISR(void) {
867 //POS_CURSOR(2,0);
868 if (INTCON3bits.INT1IF == 1) {
869 CURSOR_HOME();
870 ESCRIBE_MENSAJE("BTN1 presionado",15);
871 INTCON3bits.INT1IF = 0;
872 }
873 else {
874 CURSOR_HOME();
875 ESCRIBE_MENSAJE("BTN1 soltado ",15);
876 INTCON3bits.INT1IF = 1;
877 }
878 INTCON3bits.INT1IF = 0;
879 }
880
881 void _interrupt(low_priority) Int2_ISR(void) {
882 //POS_CURSOR(2,0);
883 if (INTCON3bits.INT2IF == 1) {
884 CURSOR_HOME();
885 ESCRIBE_MENSAJE("BTN2 presionado",15);
886 INTCON3bits.INT2IF = 0;
887 }
888 else {
889 CURSOR_HOME();
890 ESCRIBE_MENSAJE("BTN2 soltado ",15);
891 INTCON3bits.INT2IF = 1;
892 }
893 INTCON3bits.INT2IF = 0;
894 }
895
896 void _interrupt(low_priority) Int0_ISR(void) {
897 //POS_CURSOR(2,0);
898 if (INTCON3bits.INT0IF == 1) {
899 CURSOR_HOME();
900 ESCRIBE_MENSAJE("BTN0 presionado",15);
901 INTCON3bits.INT0IF = 0;
902 }
903 else {
904 CURSOR_HOME();
905 ESCRIBE_MENSAJE("BTN0 soltado ",15);
906 INTCON3bits.INT0IF = 1;
907 }
908 INTCON3bits.INT0IF = 0;
909 }
910
911 void _interrupt(low_priority) Int1_ISR(void) {
912 //POS_CURSOR(2,0);
913 if (INTCON3bits.INT1IF == 1) {
914 CURSOR_HOME();
915 ESCRIBE_MENSAJE("BTN1 presionado",15);
916 INTCON3bits.INT1IF = 0;
917 }
918 else {
919 CURSOR_HOME();
920 ESCRIBE_MENSAJE("BTN1 soltado ",15);
921 INTCON3bits.INT1IF = 1;
922 }
923 INTCON3bits.INT1IF = 0;
924 }
925
926 void _interrupt(low_priority) Int2_ISR(void) {
927 //POS_CURSOR(2,0);
928 if (INTCON3bits.INT2IF == 1) {
929 CURSOR_HOME();
930 ESCRIBE_MENSAJE("BTN2 presionado",15);
931 INTCON3bits.INT2IF = 0;
932 }
933 else {
934 CURSOR_HOME();
935 ESCRIBE_MENSAJE("BTN2 soltado ",15);
936 INTCON3bits.INT2IF = 1;
937 }
938 INTCON3bits.INT2IF = 0;
939 }
940
941 void _interrupt(low_priority) Int0_ISR(void) {
942 //POS_CURSOR(2,0);
943 if (INTCON3bits.INT0IF == 1) {
944 CURSOR_HOME();
945 ESCRIBE_MENSAJE("BTN0 presionado",15);
946 INTCON3bits.INT0IF = 0;
947 }
948 else {
949 CURSOR_HOME();
950 ESCRIBE_MENSAJE("BTN0 soltado ",15);
951 INTCON3bits.INT0IF = 1;
952 }
953 INTCON3bits.INT0IF = 0;
954 }
955
956 void _interrupt(low_priority) Int1_ISR(void) {
957 //POS_CURSOR(2,0);
958 if (INTCON3bits.INT1IF == 1) {
959 CURSOR_HOME();
960 ESCRIBE_MENSAJE("BTN1 presionado",15);
961 INTCON3bits.INT1IF = 0;
962 }
963 else {
964 CURSOR_HOME();
965 ESCRIBE_MENSAJE("BTN1 soltado ",15);
966 INTCON3bits.INT1IF = 1;
967 }
968 INTCON3bits.INT1IF = 0;
969 }
970
971 void _interrupt(low_priority) Int2_ISR(void) {
972 //POS_CURSOR(2,0);
973 if (INTCON3bits.INT2IF == 1) {
974 CURSOR_HOME();
975 ESCRIBE_MENSAJE("BTN2 presionado",15);
976 INTCON3bits.INT2IF = 0;
977 }
978 else {
979 CURSOR_HOME();
980 ESCRIBE_MENSAJE("BTN2 soltado ",15);
981 INTCON3bits.INT2IF = 1;
982 }
983 INTCON3bits.INT2IF = 0;
984 }
985
986 void _interrupt(low_priority) Int0_ISR(void) {
987 //POS_CURSOR(2,0);
988 if (INTCON3bits.INT0IF == 1) {
989 CURSOR_HOME();
990 ESCRIBE_MENSAJE("BTN0 presionado",15);
991 INTCON3bits.INT0IF = 0;
992 }
993 else {
994 CURSOR_HOME();
995 ESCRIBE_MENSAJE("BTN0 soltado ",15);
996 INTCON3bits.INT0IF = 1;
997 }
998 INTCON3bits.INT0IF = 0;
999 }
1000
1001 void _interrupt(low_priority) Int1_ISR(void) {
1002 //POS_CURSOR(2,0);
1003 if (INTCON3bits.INT1IF == 1) {
1004 CURSOR_HOME();
1005 ESCRIBE_MENSAJE("BTN1 presionado",15);
1006 INTCON3bits.INT1IF = 0;
1007 }
1008 else {
1009 CURSOR_HOME();
1010 ESCRIBE_MENSAJE("BTN1 soltado ",15);
1011 INTCON3bits.INT1IF = 1;
1012 }
1013 INTCON3bits.INT1IF = 0;
1014 }
1015
1016 void _interrupt(low_priority) Int2_ISR(void) {
1017 //POS_CURSOR(2,0);
1018 if (INTCON3bits.INT2IF == 1) {
1019 CURSOR_HOME();
1020 ESCRIBE_MENSAJE("BTN2 presionado",15);
1021 INTCON3bits.INT2IF = 0;
1022 }
1023 else {
1024 CURSOR_HOME();
1025 ESCRIBE_MENSAJE("BTN2 soltado ",15);
1026 INTCON3bits.INT2IF = 1;
1027 }
1028 INTCON3bits.INT2IF = 0;
1029 }
1030
1031 void _interrupt(low_priority) Int0_ISR(void) {
1032 //POS_CURSOR(2,0);
1033 if (INTCON3bits.INT0IF == 1) {
1034 CURSOR_HOME();
1035 ESCRIBE_MENSAJE("BTN0 presionado",15);
1036 INTCON3bits.INT0IF = 0;
1037 }
1038 else {
1039 CURSOR_HOME();
1040 ESCRIBE_MENSAJE("BTN0 soltado ",15);
1041 INTCON3bits.INT0IF = 1;
1042 }
1043 INTCON3bits.INT0IF = 0;
1044 }
1045
1046 void _interrupt(low_priority) Int1_ISR(void) {
1047 //POS_CURSOR(2,0);
1048 if (INTCON3bits.INT1IF == 1) {
1049 CURSOR_HOME();
1050 ESCRIBE_MENSAJE("BTN1 presionado",15);
1051 INTCON3bits.INT1IF = 0;
1052 }
1053 else {
1054 CURSOR_HOME();
1055 ESCRIBE_MENSAJE("BTN1 soltado ",15);
1056 INTCON3bits.INT1IF = 1;
1057 }
1058 INTCON3bits.INT1IF = 0;
1059 }
1060
1061 void _interrupt(low_priority) Int2_ISR(void) {
1062 //POS_CURSOR(2,0);
1063 if (INTCON3bits.INT2IF == 1) {
1064 CURSOR_HOME();
1065 ESCRIBE_MENSAJE("BTN2 presionado",15);
1066 INTCON3bits.INT2IF = 0;
1067 }
1068 else {
1069 CURSOR_HOME();
1070 ESCRIBE_MENSAJE("BTN2 soltado ",15);
1071 INTCON3bits.INT2IF = 1;
1072 }
1073 INTCON3bits.INT2IF = 0;
1074 }
1075
1076 void _interrupt(low_priority) Int0_ISR(void) {
1077 //POS_CURSOR(2,0);
1078 if (INTCON3bits.INT0IF == 1) {
1079 CURSOR_HOME();
1080 ESCRIBE_MENSAJE("BTN0 presionado",15);
1081 INTCON3bits.INT0IF = 0;
1082 }
1083 else {
1084 CURSOR_HOME();
1085 ESCRIBE_MENSAJE("BTN0 soltado ",15);
1086 INTCON3bits.INT0IF = 1;
1087 }
1088 INTCON3bits.INT0IF = 0;
1089 }
1090
1091 void _interrupt(low_priority) Int1_ISR(void) {
1092 //POS_CURSOR(2,0);
1093 if (INTCON3bits.INT1IF == 1) {
1094 CURSOR_HOME();
1095 ESCRIBE_MENSAJE("BTN1 presionado",15);
1096 INTCON3bits.INT1IF = 0;
1097 }
1098 else {
1099 CURSOR_HOME();
1100 ESCRIBE_MENSAJE("BTN1 soltado ",15);
1101 INTCON3bits.INT1IF = 1;
1102 }
1103 INTCON3bits.INT1IF = 0;
1104 }
1105
1106 void _interrupt(low_priority) Int2_ISR(void) {
1107 //POS_CURSOR(2,0);
1108 if (INTCON3bits.INT2IF == 1) {
1109 CURSOR_HOME();
1110 ESCRIBE_MENSAJE("BTN2 presionado",15);
1111 INTCON3bits.INT2IF = 0;
1112 }
1113 else {
1114 CURSOR_HOME();
1115 ESCRIBE_MENSAJE("BTN2 soltado ",15);
1116 INTCON3bits.INT2IF = 1;
1117 }
1118 INTCON3bits.INT2IF = 0;
1119 }
1120
1121 void _interrupt(low_priority) Int0_ISR(void) {
1122 //POS_CURSOR(2,0);
1123 if (INTCON3bits.INT0IF == 1) {
1124 CURSOR_HOME();
1125 ESCRIBE_MENSAJE("BTN0 presionado",15);
1126 INTCON3bits.INT0IF = 0;
1127 }
1128 else {
1129 CURSOR_HOME();
1130 ESCRIBE_MENSAJE("BTN0 soltado ",15);
1131 INTCON3bits.INT0IF = 1;
1132 }
1133 INTCON3bits.INT0IF = 0;
1134 }
1135
1136 void _interrupt(low_priority) Int1_ISR(void) {
1137 //POS_CURSOR(2,0);
1138 if (INTCON3bits.INT1IF == 1) {
1139 CURSOR_HOME();
1140 ESCRIBE_MENSAJE("BTN1 presionado",15);

```

23

Simulación



24

Cuestionario:

- Implementar una aplicación en la cual existan cuatro fuentes de interrupción: dos en alta prioridad y dos en baja prioridad.
- Desarrollar una solución para implementar una rutina de retardo de periodo cargable con una variable y empleando como base las funciones __delay_ms() y __delay_us().

25

Fin de la sesión

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

26