

Modificado por Luis Frino [www.frino.com.ar](http://www.frino.com.ar)

Fuente [www.micro1.com.ar](http://www.micro1.com.ar)

## REGISTROS DE INTERRUPCIÓN Y BANDERAS

Cada causa de interrupción actúa con dos señales. Una de ellas actúa como señalizador o bandera que indica si se ha producido o no la interrupción, mientras que la otra funciona como permiso o prohibición de la interrupción en sí. Los PIC 16C84 y 16C71 disponen de 4 fuentes de interrupción válidas o no por la puesta a 1 de los correspondientes bits del registro **INTCON (0x0B ó 0x8B)**.

GIE	EEIE / A/D	TOIE	INTE	RBIE	TOIF	INTF	RBIF
-----	---------------	------	------	------	------	------	------

registro INTCON

Bit 0

Bit 7

El bit **GIE** (Global Interrupt Enable) habilita todas las interrupciones. Cada tipo de interrupción tiene, a su vez, otro bit que la habilita o deshabilita. Las interrupciones son capaces de despertar al chip de su estado de reposo. El bit GIE se borra en cuanto se está atendiendo una interrupción, para evitar que se atienda otra. Volverá a valer 1 si se vuelve de la interrupción mediante RETFIE, como ya ha sido explicado varias veces. Preferimos reiterarnos por ser motivo de posibles problemas.

Todas las interrupciones saltan a la dirección 0x04, por lo que será labor del programador identificar la causa de interrupción.

El bit **RBIE** habilita la interrupción RB, es decir, interrupción ante cambios en las patas RB4-RB7. **RBIF** es la bandera que indica que se ha producido esta interrupción.

El bit **INTE** activa la interrupción por la pata INT/RB0. El bit **INTF** es la bandera que indica si se ha producido esta interrupción.

El bit **TOIE** habilita la interrupción por desbordamiento del TMR0. El bit **TOIF** es la bandera que indica si se ha producido la interrupción.

El **bit 6** del registro INTCON es distinto para el 16C71 (ADIE) y para el 16C84 (EEIE). En el 16C71 activa las interrupciones procedentes del convertor A/D, y el 16C84 las procedentes de la E<sup>2</sup>PROM. Sus respectivas banderas están en el registro ADCON1 ó EECN1.

## EJEMPLO DEL MANEJO DE INTERRUPTACIONES. EL TMR0.

Vamos a hacer en el programa **parpadeo.asm** que, gracias a la interrupción del TMR0, que un LED parpadee con una frecuencia de 200 ms. Una vez inicializadas las puertas, el predivisor de TMR0 se carga con 78 y se habilita la interrupción (GIE+TOIE)

El número de cuentas es FF-N, siendo N el número con el que se carga TMR0. Como se carga con 78, N= 256-78=178. Si ponemos el divisor de frecuencia del TMR0 a 128 (bit PS0, PS1 y PS2 del registro OPTION a 1 1 0) y con un oscilador de 4 MHz, donde el ciclo de instrucción es de un μsegundo, ocurre que...

$$\text{Tiempo Total} = N * \text{valor predivisor} * \text{ciclo instrucción}$$

$$\text{Tiempo Total} = 178 * 128 * 1 \mu\text{s} = 9984 \mu\text{s} = 9.98 \text{ ms}$$

Mediante el programa principal comprobamos que el valor del contador es 20 (200 ms). Si es así el LED es conmutado, encendiéndolo o apagándolo según su estado anterior.

No tenemos en cuenta el WatchDog, por lo que conviene deshabilitarlo, y nos son indiferentes los valores del W en todo momento, por lo que nos es igual guardar su valor o no al saltar a la rutina de interrupción.

```

LIST P = 16F84          ;Indicamos el modelo de PIC a utilizar

; Definición de registros

portb    EQU    0x06    ; Puerto B
TRISBEQU EQU    0X06    ; y TRISB en banco 1
estado   EQU    0X03    ; La dirección del registro de estado es la 0x03
intconEQU EQU    0x0B    ; Registro controlador de interrupciones
opcion   EQU    0x81    ; Registro OPTION
tmr0     EQU    0x01    ; Registro del Timer0 (TMR0)

; Definición de bits

bancoEQU EQU    0X05    ; Bit del registro de estado correspondiente
                        ; al banco de datos. En ESTADO
Z      EQU    0X02    ; Bit indicador de que el registro W está a cero. ESTADO
t0if   EQU    0x02    ; Bit de INTCON que indica que se produjo
                        ; interrupción por desbordamiento del timer0
t0ie   EQU    0x05    ; Bit de INTCON que habilita o no la interrupción
                        ; por desbordamiento del timer0

; Definición de constantes

w      EQU    0        ; Destino de operación = w
f      EQU    1        ; Destino de operación = registro

; Definición de variables

contador EQU    0X0C    ; Contador

; Comienzo del programa.

ORG 0X00          ; Cubrimos el vector de reset

                GOTO inicio      ; Saltamos a la primera dirección tras
                ; el vector de interrupción

ORG 0x04          ; Cubrimos el vector de interrupción

                GOTO RSI        ; Y saltamos a la rutina de servicio de interrupción

; ***** Inicialización de variables *****

ORG 0X05

inicio BSF estado,banco ; Seleccionamos el banco 1
                MOVLW 0x06    ; En binario 0000 0110
                MOVWF opcion  ; Ponemos el predivisor a 128

; *****
; *** OPTION.7 = 0 Resistencias de polarización deshabilitadas ***
; *** OPTION.6 = 0 Interrupción externa por flanco bajada (no se usa) **
; *** OPTION.5 = 0 Fuente de reloj interna ***

```

```

; *** OPTION.4 = 0 Flanco de señal externa (no lo usamos) ***
; *** OPTION.3 = 0 Divisor asignado al TMR0 ***
; *** OPTION.2 = 1 OPTION.1= 1 OPTION.0= 0 División por 128 ***
; ****
;
; CLRF TRISB ; La puerta B es toda de salida
; BCF estado,banco ; Volvemos a la página 0
; CLRF portb ; Borramos todos los LEDS
; CLRF contador ; Contador = 0
; MOVLW 0xB2 ; Cargamos el timer con 78 decimal (0xB2)
; MOVWF tmr0
; MOVLW 0xA0 ; 1010 0000 en binario
; MOVWF intcon ; Habilitamos GIE y T0IE (interrupción del timer0)

; ***** Cuerpo principal *****
; ***** Mira si hay 20 cuentas de 10 ms *****
; ***** Y, si las hay, cambia el LED *****
;
; Bucle MOVF contador,w; Se incrementa cada 10 ms en uno al
; ; producirse la interrupción
; XORLW 0x14 ; Ha llegado a 200 ms si llevamos 20 (0x14) cuentas
; BTFSS estado,Z ; Si es así, salta para cambiar el LED
; GOTO Bucle ; Si no es así, prueba otra vez
; CLRF contador ; El contador vuelve a 0 para iniciar el nuevo ciclo
; BTFSS portb,1 ; Está encendido ? Si sí, apaga
; GOTO Encien
; Apaga BCF portb,1 ; Apaga el LED
; GOTO Bucle
; Encien BSF portb,1 ; Enciende el LED
; GOTO Bucle

; ***** RSI: Rutina de servicio de interrupción *****
; ***** Salta al desbordarse el TMR0, cada 10 ms *****
;
RSI BTFS intcon,t0if ; Salta si la interrupción es TMR0
RETfie ; Interrupción desconocida, regresar
; ; (es un mecanismo de seguridad).
INCF contador,f ; El contador es incrementado cada 10 ms
MOVLW 0xB2 ; Recarga el valor inicial del TMR0
MOVWF tmr0
BCF intcon,t0if ; Borra la bandera de interrupción
BSF INTCON,t0ie ; Habilita de nuevo la interrupción
RETfie

END

```

### 5.3.6 Manejo de una pantalla LCD. Creación de una librería.

#### INTRODUCCIÓN

Una LCD estándar es una pantalla de cristal líquido con una matriz de 16, 32, 40 u 80 caracteres de 5x7 pixeles, contando, además, con un microcontrolador (generalmente el Hitachi 44780) que lo gobierna. Normalmente cada línea contiene entre 8 y 80 caracteres, y suelen ser capaces de mostrar caracteres ASCII, japoneses, griegos...; o símbolos matemáticos. Su bus de conexión puede ser de 4 u 8 bits.

El consumo de este tipo de módulos es muy bajo (7.5mW), y, gracias a su sencillo manejo, son ideales para dispositivos que requieren una visualización pequeña o media.



La secuencia de escritura debe seguir los siguientes pasos:

- 1) Línea I/D a 0 o a 1, según se trate de comandos o datos
- 2) Línea R/W a 0 (1 en caso de escritura)
- 3) Línea EN a 1 (se habilita la LCD)
- 4) Escritura de datos en el bus DB.
- 5) Línea EN a 0 (deshabilitación de la LCD)

La misma secuencia en un módulo de 4 bits cambiaría:

- 1) Línea I/D a 0 o a 1, según se trate de comandos o datos
- 2) Línea R/W a 0 (1 en caso de escritura)
- 3) Línea EN a 1 (se habilita la LCD)
- 4) Escritura en los 4 bits de mayor peso del DB de la LCD.
- 5) Línea EN = 0
- 6) Línea EN = 1
- 7) Escribir de nuevo los 4 bits de menor peso
- 8) Línea EN = 0 (deshabilitación de la LCD).

Las dos secuencias de 4 bits se concatenarían dentro del LCD para formar 8 bits.

Al resetear una LCD o encenderla ésta se queda a la espera de instrucciones. Usualmente se suele empezar encendiendo la pantalla, colocando el cursor y configurando la escritura de derecha a izquierda.

La LCD contiene una RAM propia en la que almacena los datos, que se denomina DDRAM. Independientemente del número de caracteres visibles, la DDRAM contará con 80 posiciones. Los caracteres no visibles se visualizarán provocando un desplazamiento.

La utilización de la LCD es lenta. Una escritura o lectura puede tardar entre 40 y 120  $\mu$ segundos; otras instrucciones pueden llegar a los 5 ms. Para lograr que el PIC no necesite esperar tiene una instrucción de 1 $\mu$ seg que lee la dirección del contador y una bandera interior de ocupado. Cuando la bandera de ocupado (BF) está a 1, la LCD no puede leer ni escribir.

En nuestro ejemplo, del que a continuación mostramos el esquema, las líneas de datos se comparten con el teclado y una barra de diodos. Compartir la puerta B es una de las ventajas del PIC, puesto que le da una gran capacidad de reconfiguración, por su sencillez y rapidez.

### **HABILITACIÓN DE UNA LCD. RUTINA LCD\_HABILITA**

La línea EN de habilitación de una LCD necesita estar activada durante, al menos, 500 ns. La rutina LCD\_Habilita de la LCD.LIB se asegura de que así sea, siendo LCDE = PORTA.2, es decir, EN:

LCD\_Habilita ; Envía un impulso de habilitación de 500 ns a la LCD para  
; completar la operación de escribir un registro o un carácter  
; La instrucción NOP sólo es necesaria para procesadores de  
; una velocidad superior a 8 MHz. Si el procesador es de  
; más de 16 MHz, se debe añadir un segundo NOP

BSF LCDE ; Pone a 1 la línea EN (habilita la LCD)

```

NOP          ; Pausa para 250 ns extra
              ; (según velocidad del micro)
BCF LCDE     ; Pone a 0 la línea EN (deshabilita la LCD)
RETURN

```

### SELECCIÓN DE MODO (COMANDO/DATOS).

La línea I/D selecciona entre el modo comando si vale 0 o el modo datos si es 1. Una llamada tipo CALL LCD\_Comando asegurará dicho modo antes de una habilitación de la LCD; lo mismo sucederá con LCD\_Carácter.

```

LCD_Comando  ; Carga W con una constante software LCD de la tabla de
              ; igualdades. LCD_Comando saca el comando a la LCD y
              ; activa la línea de comando de la LCD, y la propia LCD
              ; mediante la llamada a LCD_Habilita, completando así
              ; el comando.

```

```

BCF LCDModo  ; Entra en modo registro
MOVWF portb  ; y envía W a LCD en puertoB.
              ; W, por tanto, habrá de tener ya el valor
              ; del comando antes de que el programa
              ; invoque a LCD_Comando
CALL LCD_Chequea ; Chequea la bandera de LCD ocupada
GOTO LCD_Habilita ; Envía el comando

```

```

LCD_Carácter ; Carga W con el código ASCII del carácter que desea
              ; enviar a la LCD. Activará para ello el modo datos
              ; y, posteriormente, la LCD mediante una llamada a
              ; LCD_Habilita para completar el envío del mismo.

```

```

BCF LCDModo  ; Envía modo registro
MOVWF portb  ; y envía W a LCD en puertoB
              ; W, por tanto, habrá de tener ya el
              ; valor del carácter antes de la llamada a
              ; esta rutina.
CALL LCD_Chequea ; Chequea la bandera de LCD ocupada
BSF LCDModo  ; Envía modo datos
GOTO LCD_Habilita ; y envía el carácter

```

Debemos recordar que es la línea R/W la que determina si se lee o escribe, debiendo estar debidamente activada según nuestros deseos antes de cualquier intento de acceso a la LCD.

## COMANDO DE LA PANTALLA LCD

Aunque pueden variar, en el caso que nos ocupa y en el estándar los comandos de la LCD son:

Comando	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Borra Pantalla	0	0	0	0	0	0	0	0	0	1
Cursor a Casa	0	0	0	0	0	0	0	0	1	*
Modo Introducción	0	0	0	0	0	0	0	1	I/D	S
Pantalla On/Off	0	0	0	0	0	0	1	D	C	B
Modo Desplazamiento	0	0	0	0	0	1	S/C	R/L	*	*
Función	0	0	0	0	1	DL	líneas	Font	*	*
Dirección CGRAM	0	0	0	1	Dirección CGRAM					
Dirección DDRAM	0	0	1	Dirección DDRAM						
Lectura ocupado y dirección contador	0	1	BF	Dirección DDRAM						
Escribe RAM	1	0	Escribe Dato							
Lee RAM	1	1	Lee Dato							

**Borra Pantalla:** La borra y sitúa el cursor en su posición inicial (la 0).

**Cursor a Casa:** El cursor va a la posición inicial (la 0), pero sin borrar nada.

**Modo instrucción:** Configura la dirección del cursor I/D. Cuando I=1 incrementa la posición del cursor, y D=0 la decremента. Mientras S=1 significa que hay desplazamiento en la pantalla. La operación se ejecuta durante la I/O de los datos.

**Pantalla On/Off:** Coloca en movimiento al cursor o genera desplazamiento en la pantalla. D para toda la pantalla, C para cursor On/Off, y B hace parpadear el cursor.

**Desplazamiento Cursor/Pantalla:** S/C indica el movimiento del cursor o desplazamiento en la pantalla, R/L la dirección a derecha o izquierda. No se varía el contenido de la DDRAM.

**Función:** DL indica la longitud de datos del interfaz; N el número de líneas de la pantalla y F el tipo de caracteres.

**Dirección CGRAM:** Coloca el dato enviado o recibido en la CGRAM después de este comando.

**Dirección DDRAM:** Coloca el dato enviado o recibido en la DDRAM después de la ejecución de este comando.

**Bandera de ocupado BF:** Lee BF indicando si hay una operación interna en curso y lee, además, el contenido de la dirección contador.

**Escribe RAM:** Escribe un dato en la RAM (ya sea DDRAM o CGRAM).

**Lee RAM:** Lee datos de la RAM (ya sea DDRAM o CGRAM).

Nombre Bit	Estado y Funcionamiento	
I/D	0 = Decrementa posición cursor	1 = Incrementa posición cursor
S	0 = Sin desplazamiento	1 = Con desplazamiento
D	0 = Pantalla Off	1 = Pantalla On
C	0 = Cursor Off	1 = Cursor On
B	0 = Parpadeo cursor Off	1 = Parpadeo cursor On
S/C	0 = Mueve el cursor	1 = Desplaza la pantalla
R/L	0 = Desplaza a la izquierda	1 = Desplaza a la derecha
DL	0 = Interfaz de 4 bits	1 = interfaz de 8 bits
Líneas	0 = 1 línea datos visible	1 = 2 líneas datos visibles
Font	0 = 5 x 7 pixeles	1 = 5 x 10 pixel
BF	0 = Puede aceptar instrucción	1 = Operación interna en curso

**DEFINICIÓN DE VALORES DE CONSTANTES DE COMANDOS PARA UTILIZAR EN LAS  
RUTINAS DE LCD.LIB**

Nombre constante	Valor	Significado
LCDLinea1	0x80	Coloca cursor en la posición 1 línea 1
LCDLinea2	0x0C	Coloca el cursor en la posición 1 línea 2
LCDCLR	0x01	Borra la pantalla + LCDLinea1
LCDCasa	0x02	Como LCDLinea1
LCDInc	0x06	El cursor incrementa su posición tras cada carácter
LCDDec	0x04	El cursor decrementa su posición tras cada carácter
LCDOn	0x0C	Enciende la pantalla
LCDOff	0x08	Apaga la pantalla
CursOn	0x0E	Enciende pantalla mas cursor
CursOff	0x0C	Apaga pantalla mas cursor
CursBlink	0x0F	Enciende la pantalla con cursor parpadeando
LCDIzda	0x18	Desplaza los caracteres mostrados a la izquierda
LCDDecha	0x1C	Desplaza los caracteres mostrados a la derecha
CursIzda	0x10	Mueve el cursor una posición a la izquierda
CursDecha	0x14	Mueve el cursor una posición a la derecha
LCDFuncion	0x38	Programa una interface 8 bits, pantalla 2 líneas, fuente 5x7 pixeles
LCDCGRAM	0x40	Programa el generador de caracteres del usuario RAM

## DIRECCIONADO DE LA RAM

La RAM de una LCD no tiene direccionamiento continuo y lineal, pues el mapa depende de los caracteres y líneas que tenga el módulo.

Tamaño Pantalla	Visible	
Una Línea	Posición Carácter	Dirección DDRAM
1 x 8	00 – 07	0x00 – 0x07
1 x 16	00 – 15	0x00 – 0x0F
1 x 20	00 – 19	0x00 – 0x13
1 x 24	00 – 23	0x00 – 0x17
1 x 32	00 – 31	0x00 - 0x1F
1 x 40	00 – 39	0x00 - 0x27

Tamaño Pantalla	Visible	
Dos Línea	Posición Carácter	Dirección DDRAM
1 x 16	00 – 15	0x00 – 0x0F + 0x40 – 0x4F
1 x 20	00 – 19	0x00 – 0x13 + 0x40 – 0x53
1 x 24	00 – 23	0x00 – 0x17 + 0x40 – 0x57
1 x 32	00 – 31	0x00 - 0x1F + 0x40 – 0x5F
1 x 40	00 – 39	0x00 - 0x27 + 0x40 – 0x67

## LISTADO DE LA LIBRERÍA

; LCD.LIB

```

, *****
, ***   LCD.LIB proporciona las siguientes funciones:   ***
, ***                                                     ***
, ***   - Configuración de las puertas                   ***
, ***   - Comandos en modo registro                     ***
, ***   - Exploración de LCD Busy - Ocupado             ***
, ***   - LCD Enable (habilitación)                    ***
, *****

```

; Variables

; Ninguna

; Requisitos

; Dos niveles libres de pila para llamadas anidadas.  
; La rutina LCD\_Inic necesita 5 ms para inicializar  
; el LCD. Pausas mayores son aceptables. LCD\_Inic  
; llama a Pausa\_5ms que debe ser incluida en la llamada  
; del programa. Si el programa que usa esta librería  
; tiene un bucle de retardo mayor de 5 ms, puede usarlo  
; colocándole la etiqueta Pausa\_5ms la etiqueta de su rutina.

; Uso

; Para iniciar la LCD después del encendido:

; 1.- Llama LCD\_Port, que inicializa Puerta A y Puerta B para la LCD  
; 2.- Llame LCD\_Inic que inicializa el controlador de la LCD

; Para escribir un comando o un carácter en la pantalla LCD:

; 1.- Llame LCD\_Port que inicializa Puerta A y Puerta B para la LCD  
; 2.- Mueva un comando LCD o un carácter ASCII a W  
; 3.- Llame LCD\_Comando ó LCD\_Caracter para enviar un comando o carácter

;           respectivamente a la pantalla LCD.

; \*\*\*\*\* Definición de Constantes

;       \*\*\*\*\* Correspondientes a registros del PIC

estado       EQU    0X03           ; La dirección del registro de estado es la 0x03  
portA        EQU    0x05           ; Puerto A  
portB        EQU    0x06           ; Puerto B  
TRISA        EQU    0X05           ; La dirección del registro TRISA en banco 1  
TRISB        EQU    0X06           ; y de TRISB en banco 1

; \*\*\*\*\* Correspondientes a bits de registros del PIC

banco        EQU    0X05           ; Bit del registro de estado correspondiente  
                                  ;           al banco de datos. En ESTADO

; \*\*\*\*\* Correspondencias de los pines con líneas hardware

LCD           EQU    0x05           ; La LCD está en el puerto A  
LCDModo       EQU            0           ; Selecciona Registro LCD  
LCDRW EQU        1           ; Lectura / Escritura LCD  
LCDE         EQU    2           ; Habilita LCD

; \*\*\*\*\* Comandos de Software para la LCD

LCDLinea1    EQU    0x80    ; Dirección comienzo línea1  
LCDLinea2    EQU    0x0C    ; Dirección comienzo línea2  
LCDCLR EQU    0x01    ; Borra pantalla, cursor a casa  
LDCasaEQU    0x02    ; Cursor a casa, DDRAM sin cambios  
LCDInc       EQU    0x06    ; Modo incrementa cursor  
LCDDec EQU    0x04    ; Modo decrementa cursor  
LCDOn        EQU    0x0C    ; Pantalla On  
LCDOff       EQU    0x08    ; Pantalla Off  
CursOn EQU    0x0E    ; Pantalla On, cursor On  
CursOff EQU   0x0C    ; Pantalla On, cursor Off  
CursBlink    EQU    0x0F    ; Pantalla On, Cursor parpadeante  
LCDIzda EQU   0x10    ; Mueve cursor a la izquierda  
LCDDecha     EQU    0x14    ; Mueve cursor a la derecha  
LCDFuncion   EQU    0x38    ; Inicializa registro función  
LCDCGRAM     EQU    0x40    ; Dirección origen CGRAM

; \*\*\*\*\* RUTINAS \*\*\*\*\*

ORG 0x05     ; Empezamos aquí por seguridad.  
                  ; En la rutina principal del programa habrá que definir ORG 0  
                  ; El resto del programa irá a continuación de estas rutinas

LCD\_Port     ; \*\*\*\*\*  
                  ; \*\*\* Inicializa los buffers triestado de la Puerta B como salidas \*\*\*  
                  ; \*\*\* para líneas de datos de la LCD. Coloca las líneas de la       \*\*\*  
                  ; \*\*\* puerta A como salidas: I/D, E/S, En.                       \*\*\*  
                  ; \*\*\*\*\*

BSF estado,banco; Pasamos a la página 1  
MOVLW 0xF8       ; En binario 1111 1000, para poner RA2, RA1 y RA0  
ANDWF TRISA       ; como salidas del puerto A  
CLRF TRISB       ; Y todo el puerto B como salidas de datos  
BCF estado,banco   ; Volvemos a la página 0, para poder tocar los puertos  
BCF LCD,LCDE     ; Y deshabilitamos, por si las moscas, la LCDE

RETURN

LCD\_Inic

```
;  
*****  
; *** Inicialización de la LCD según el manual de datos de Optrex. ***  
; *** Configura funciones de LCD para pantalla DMC16207 ***  
; *** Produce reset por software, borra la memoria y ***  
; *** activa la pantalla. ***  
;  
*****
```

```
MOVLW LCDFuncion ; Carga W con orden inicia LCD  
CALL LCD_Comando ; y lo envía a la LCD  
CALL Pausa_5ms ; ... y espera  
MOVLW LCDFuncion ; Repite la operación.  
CALL LCD_Comando  
CALL Pausa_5ms
```

```
; Si desea otra configuración de pantalla habrá de cambiar la siguiente  
; tanda de órdenes. Aquí encendemos la LCD, quitamos el cursor  
; y borramos  
; la pantalla, provocando que con cada carácter el cursor avance hacia la  
; derecha.
```

```
BCF LCD,LCDModo ; Entramos en modo registro  
MOVWF portB ; y envía W a LCD en la puerta B  
CALL CLD_Chequea ; Explora la bandera de ocupado  
BSF LCD,LCDModo ; Entra en modo ASCII  
GOTO LCD_Habilita ; Envía carácter ASCII
```

LCD\_Comando

```
;  
*****  
; *** Carga W con una constante software LCD de la tabla anterior ***  
; *** y saca el comando a la LCD, pulsando la línea de habilitación ***  
; *** con LCD_Habilita para completar el comando ***  
;  
*****
```

```
BCF LCD,LCDModo ; Entra en modo registro  
MOVWF portB ; y envía el comando a la puerta B  
CALL LCD_Chequea ; Chequea la bandera de ocupado  
GOTO LCD_Habilita ; Y envía el comando
```

LCD\_Caracter

```
;  
*****  
; *** Carga W con el código del carácter ASCII para enviarlo a la ***  
; *** LCD. Después activa la LCD con LCD_Habilita para ***  
; *** completar el envío ***  
;  
*****
```

```
BCF LCD,LCDModo ; Entra en modo registro  
MOVWF portB ; y envía W a la LCD en la puerta B  
CALL LCD_Chequea ; explora la bandera de LCD ocupado  
BSF LCD,LCDModo ; Entra en modo ASCII  
GOTO LCD_Habilita ; Y envía en carácter ASCII
```

LCD\_Chequea

```
;  
*****  
; *** Explora el estado de la bandera Busy (ocupado) de la LCD ***  
; *** Y espera que termine cualquier comando previo antes de ***
```

```

; *** volver a la rutina que le llamó ***
;
;
*****
BSF LCD,LCDModo ; Coloca la LCD en modo Lectura
BSF estado,banco ; Coloca la página 1 de registros
MOVLW 0xFF ; Configura como entrada la puerta B
MOVWF TRISB ; cambiando el trisB
BCF estado,banco ; Vuelve a la página 0 para leer el puerto B
BSF LCD,LCDE ; Habilita la LCDE
NOP ; Pausa para a 8 MHz esperar la estabilidad
; de salidas LCD
Bucle BTFSC portB,7 ; Explora el bit de ocupado LCD y
GOTO Bucle ; espera a que valga 1 (si es 0, está ocupado).
BCF LCD,LCDE ; Deshabilita la LCD
BCF estado,banco ; Coloca la página 1 de registros
CLRF TRISB ; Coloca de nuevo el puerto B
; como de todo salidas
BCF estado,banco ; Y regresa a la página 0
BCF LCD,LCDModo ; Pone la LCD en modo escritura
RETURN ; Aquí la LCD ya está libre

```

```

LCD_Habilita
;
*****
; *** Envía un pulso de habilitación a la LCD de 500 ns para completar ***
; *** la operación de escribir un registro o un carácter. ***
; *** la instrucción NOP sólo es necesaria para procesadores de una ***
; *** velocidad mayor de 8 Mhz. Para procesadores a más de 16 MHz, ***
; *** añadir un segundo NOP ***
; *****
; *****
; *****
BSF LCD, LCDE ; Pone a 1 la línea Enable (habilita)
NOP ; y espera 1 ciclo (250 ns a 8 MHz)
BCF LCD,LCDE ; Pone a 0 la línea Enable (deshabilita)
RETURN

```

Si usted pasa el corrector a la librería observará que le dará 3 errores. Dos de ellos están en la rutina Pausa\_5ms, inexistente. Esta rutina depende grandemente del micro, la implementación del mismo, y la velocidad de su reloj, por lo que, de emplear la librería, debería crearla usted mismo en su programa. El otro indica que falta la directiva END. No la ponga. Las librerías no deben acabar con un END. En el siguiente punto veremos una sencilla aplicación para utilizar esta librería.

### 5.3.7 Uso de una librería: LCD.LIB

El uso de una librería es bien sencillo si conocemos sus variables y sus rutinas internas. Aprovecharemos la creada en el apartado anterior para crear un pequeño programa que sitúe en la LCD la palabra PIC.

Pocas cosas debemos hacer resaltar para un paso tan sencillo, pero es importante saber que las rutinas del programa **LCDPIC.asm** se situarán a continuación de las de la librería. Como no puede predecir, a priori, en qué dirección de memoria acabará la librería (y comprobarlo a mano es pesado) no se debe comenzar el

mismo como habitualmente lo hacemos (con una directiva ORG). Sin embargo sí debemos emplear una para colocar el vector de reset, pero esta irá al final del programa, para no interferir.

La rutina que ejecuta la pausa de 5 ms está basada en la que empleamos en parpadeo.asm.

En cualquier caso hemos querido hacer notar el uso de dos directivas más de ensamblador y que, curiosamente, no hemos encontrado en bibliografías distintas de las suministradas por el fabricante. Estas son #DEFINE y macro.

**#DEFINE** es empleado para crear sustituciones dentro del texto del programa que lo simplifiquen. Nosotros hemos pensado que un cambio de banco es más evidente y comprensible dentro del programa si se escribe como BANCOx (siendo x el número de banco) que con la instrucción completa (BCF estado,banco). La forma correcta es #DEFINE NOMBRE TEXTO, con lo que, cada vez que el compilador encuentre la orden NOMBRE, la sustituirá por el texto. El problema que se nos plantea es que, si bien es más flexible que la directiva EQU, puesto que esta sólo nos permitía asignar un valor, sólo se nos permite con #DEFINE una línea de texto, y esta debe ser fija.

Este problema se soluciona mediante **macro**. Esta directiva tiene la siguiente forma:

```
NOMBRE macro ARGUMENTO1, ARGUMENTO2, ETC
    TEXTO
    TEXTO...
endm
```

De este modo NOMBRE será sustituido como comando por la secuencia completa definida tras macro hasta endm, y los sucesivos argumentos serán, a su vez, sustituidos dentro del texto.

En nuestro ejemplo se repetía por tres veces la escritura de un carácter, cada vez distinto, y sólo se requerían dos líneas para cada una, por lo que no merecía la pena crear una rutina para simplificarlo. Fue en cada caso sustituida por una única línea del tipo PON\_ASCII 0x40, que sitúa en la LCD el carácter 0x40. Lo hicimos como sigue:

```
PON_ASCII macro ASCII
    MOVLW ASCII
    CALL LCD_Caracter
Endm
```

Otra directiva que ayuda mucho para determinados programas es **if**. Supongamos, por ejemplo, que nuestro programa debe estar diseñado para funcionar bajo dos micros, uno a 2 MHz y otro a 4 MHz. ¿Cómo solucionaremos entonces el problema de la necesaria pausa de 5 milisegundos? Podemos sacar una copia modificada del programa o hacerlo mediante un if. Tenga en cuenta que, en este caso, lo empleamos para que

usted vea su uso, y, por eso, parece un poco forzado. Su verdadera utilidad se encontrará a la hora de crear librerías más o menos universales. Abra, como curiosidad, cualquiera de las librerías \*.inc que se suministran junto al MPLAB y lo comprobará.

Su forma de uso es:

```
IF NOMBRE OPERADOR VALOR
    COMANDOS1
ELSE
    COMANDOS2
ENDIF
```

En donde nombre será una etiqueta definida previamente, el operador será = (igual), >=, <=, >, <, != (distinto). COMANDOS1 se ejecutará si se cumple NOMBRE OPERADOR VALOR, y COMANDOS2 se ejecutará si no se cumple.

Las directivas **ifdef nombre** y **ifndef nombre** funcionan de idéntica manera, pero en caso de que nombre haya sido definido o no, respectivamente.

### Listado del programa

```
include <LCD.LIB>

; Definición de registros

estado EQU 0X03 ; La dirección del registro de estado es la 0x03
intcon EQU 0x0B ; Registro controlador de interrupciones
opcion EQU 0x81 ; Registro OPTION
tmr0 EQU 0x01 ; Registro del Timer0 (TMR0)

; Definición de bits

banco EQU 0X05 ; Bit del registro de estado correspondiente al banco de datos.
;En ESTADO
Z EQU 0X02 ; Bit indicador de que el registro W está a cero. En ESTADO
t0if EQU 0x02 ; Bit de INTCON que indica que se produjo interrupción
; por desbordamiento del timer0
t0ie EQU 0x05 ; Bit de INTCON que habilita o no la interrupción
; por desbordamiento del timer0

; Definición de constantes

w EQU 0 ; Destino de operación = w
f EQU 1 ; Destino de operación = registro

; Definición de variables

contador EQU 0X0C ; Contador

; Definiciones para el ensamblador

#define BANCO0 BCF estado,banco ; Sirve para situarse en banco 0
#define BANCO1 BSF estado,banco ; Sirve para situarse en banco 1
```

```

; Definición de macros

PON_ASCII macro ASCII
    MOVLW ASCII
    CALL LCD_Character
endm

; ***** CUERPO DEL PROGRAMA *****

inic    ; Preparamos el timer para la pausa, como en parpadeo.asm
    BSF estado,banco    ; Seleccionamos el banco 1
    MOVLW 0x06          ; En binario 0000 0101
    MOVWF opcion        ; Ponemos el predivisor a 64
    ;*****
    ; *** OPTION.7 = 0 Resistencias de polarización deshabilitadas ***
    ; *** OPTION.6 = 0 Interrupción externa por flanco bajada (no se usa) ***
    ; *** OPTION.5 = 0 Fuente de reloj interna ***
    ; *** OPTION.4 = 0 Flanco de señal externa (no lo usamos) ***
    ; *** OPTION.3 = 0 Divisor asignado al TMRO ***
    ; *** OPTION.2 = 1 OPTION.1= 0 OPTION.1= 0 División por 64 ***
    ;*****
    BANCO0              ; Y volvemos al banco 0

    ; **** Comenzamos con la LCD

    CALL LCD_Port      ; Inicializa los puertos, para acoplarlos
                        ; al diseño especificado de la LCD.
    CALL LCD_Inic      ; Inicializa los valores de la LCD y
                        ; la enciende tal cuál la necesitamos:
                        ; Resetea la LCD, borra la memoria y activa la pantalla
    PON_ASCII 0x50 ; Carácter ASCII de la P mayúscula (80 decimal)
    PON_ASCII 0x49 ; Carácter ASCII de la I mayúscula (73 decimal)
    PON_ASCII 0x43 ; Carácter ASCII de la C mayúscula (67 decimal)

Pausa_5ms
    if velocidad = = 4    ; Para un micro a 4 MHZ
        MOVLW 0xB2      ; Cargamos el timer con 78 decimal (0xB2)
    else
        MOVLW 0xD8      ; Si no deducimos que funciona a 2 MHz
                        ; y cargamos el timer con 39 (la mitad)
    endif
    MOVWF tmr0
    MOVLW 0xA0          ; 1010 0000 en binario
    MOVWF intcon        ; Habilitamos GIE y TOIE (interrupción del timer0)
                        ; Deshabilitamos TOIF (bandera de salto producido)
espera    BTFSS intcon,t0if ; Esperamos a que la bandera se active
    GOTO espera
    RETURN

; ***** RSI: Rutina de servicio de interrupción *****
; ***** Salta al desbordarse el TMRO, cada 5 ms *****

RSI    RETURN ; Queda deshabilitada la interrupción mientras no sea necesaria
        ; No se borra la bandera GIE ni la TOIF

ORG 0X00    ; Cubrimos el vector de reset

    GOTO inic    ; Saltamos a la primera dirección tras el vector de interrupción

ORG 0x04    ; Cubrimos el vector de interrupción

    GOTO RSI    ; Y saltamos a la rutina de servicio de interrupción

END

```

Pruebe usted con otras variantes, como, por ejemplo, hacer parpadear PIC, o desplazarlo por la pantalla.

### 5.3.8 El Watchdog

Conocer la existencia y el manejo del **Watchdog** es fundamental, en muchos casos. Esta herramienta es un contador de 8 bits que, al desbordarse, produce el reseteo del micro. La única forma de evitar este reseteo es, por tanto, borrarlo por software cada cierto tiempo con la instrucción **CLRWDT**, que devuelve su valor a 0. Su velocidad normal es la de una cuenta por cada ciclo de instrucción, aunque puede asignársele el preescaler para reducir su frecuencia (vea el registro **option** para aprender el uso de este divisor).

Su utilización es opcional, y se activa (o no) durante el proceso de grabación del micro. Todos los grabadores que conocemos y hemos usado tienen en sus menús o funciones la opción específica.

Sirve para evitar posibles problemas de grabación no controlados o controlables por la razón que sea, como, por ejemplo, bucles infinitos, esperas exageradamente largas de alguna determinada entrada, etc., y es especialmente interesante en ambientes con mucho ruido, ya que éste puede afectar al PC, mandándolo a ejecutar una línea al azar.

### 5.3.9 Notas para el profesor sobre la elaboración de estos programas

Todos los programas de este apartado están sacados de los libros "µcontroladores PIC: Teoría y Práctica", y "µControladores PIC, la solución en un chip", de J. M<sup>a</sup> Angulo. Su documentación, muchas veces insuficiente, bajo nuestro criterio, ha sido ampliada, en todos los casos. Han sido adaptados al entorno MPLAB (pues estaban orientados a programas de MS-DOS) y simulados para comprobar que carecían de errores, así como corregidos, cuando era pertinente. Los comentarios sobre los mismos, lejos de ser sacados de los mencionados libros, son basados en sus respectivas pruebas. Todos los programas tienen modificaciones sobre los originales para demostrar nuestra