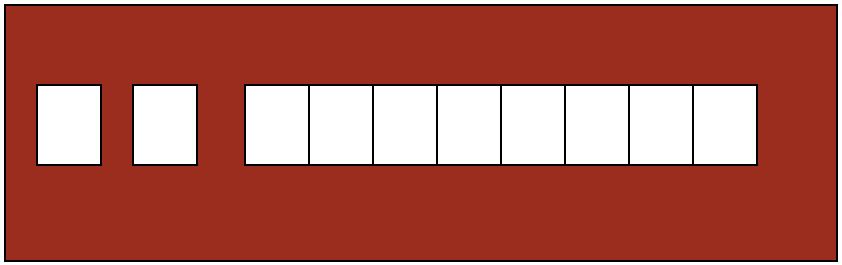
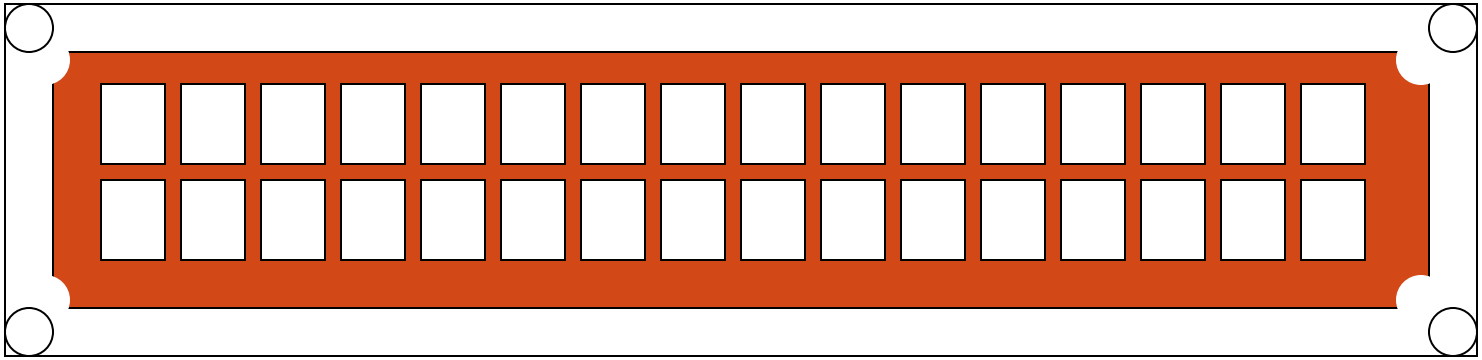
Control del LCD

Fernando Remiro

Control del LCD

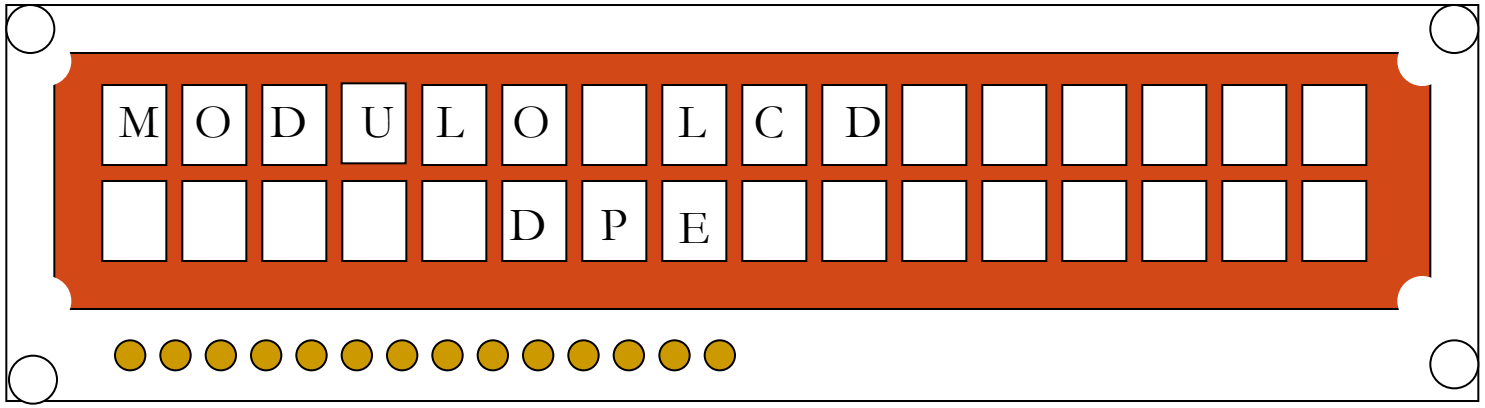




CARACTERÍSTICAS

- Consumo muy reducido, del orden de los 7,5 mW.
- Pantalla de caracteres ASCII, además de los caracteres japoneses Kanji, caracteres griegos y símbolos matemáticos.
- Desplazamiento de los caracteres hacia la izquierda o a la derecha.
- Memoria de 40 caracteres por línea de pantalla, visualizándose 16 caracteres por línea.
- Movimiento del cursor y cambio de su aspecto.
- Permite que el usuario pueda programar ocho caracteres.
- Pueden ser gobernados de dos formas principales:
  - Conexión con bus de 4 bits.
  - Conexión con bus de 8 bits.

PATILLAJE



14 1

1	VSS
2	Vdd
3	Vo
4	RS
5	R/W
6	E
7:14	Datos

Funciones de los pines del LCD

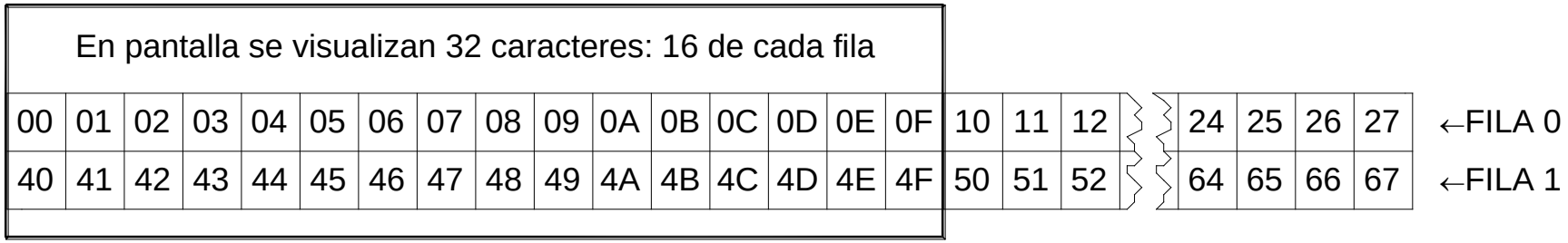
SEÑAL	DEFINICIÓN	PINES	FUNCIÓN
DB0.. ..DB7	Data Bus	7... ...14	Bus de Datos.
E	Enable	6	E=0, LCD no habilitado E=1, LCD habilitado
R/W	Read/Write	5	R/W=0, escribe en LCD R/W=1, lee del LCD
RS	Register Select	4	R/S=0, Modo Comando R/S=1, Modo Carácter
V_{LC}	Liquid Crystal driving Voltage	3	
V_{DD}	Power Supply Voltage	2	
V_{SS}	Ground	1	

DDRAM

- Es la zona de memoria donde se almacenan los caracteres que se pueden representar
- Tiene una capacidad de 80 Bytes, 40 por línea
- Solo se pueden presentar 32 (16 columnas por 2 líneas).

DDRAM

- Existe una correspondencia entre las filas de la pantalla y las posiciones consecutivas de la memoria



La DDRAM tiene un tamaño de 80 bytes (40 en cada fila), de los cuales se visualizan 32

- De las 80 posibles, las dos direcciones más importantes son:
  - Dirección 00h, que es el comienzo de la primera línea
  - Dirección 40h, que es el comienzo de la segunda línea

La CGROM

- Es la zona de memoria no volátil donde se almacena una tabla con los 192 caracteres que pueden ser visualizados.
- Cada uno de los caracteres tiene una representación binario de ocho bits.
- Para visualizar un carácter debe recibir por el bus de datos el código correspondiente. Por ejemplo para representar la ‘A’ el numero binario b’01000001’.
- Se pueden definir ocho nuevos caracteres de usuario, no incluidos en su tabla interna. Estos caracteres se guardan en la zona de memoria CGRAM.


Upper 4 Bits
Lower 4 Bits
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
xxxx0000
CG RAM (1)
xxxx0001
(2)
xxxx0010
(3)
xxxx0011
(4)
xxxx0100
(5)
xxxx0101
(6)
xxxx0110
(7)
xxxx0111
(8)
xxxx1000
(1)
xxxx1001
(2)
xxxx1010
(3)
xxxx1011
(4)
xxxx1100
(5)
xxxx1101
(6)
xxxx1110
(7)
xxxx1111
(8)

Modos de funcionamiento (1)

- Modo Comando:
  - El LCD recibe por el bus de datos instrucciones como “Borrar el Display”, “Mover el cursor”, “Desplazar a la izquierda”,etc.
  - Para trabajar en modo Comando,
    - El pin RS debe estar a “0”
    - El pin R/W debe esta a “0”
    - Cada operación tarda 1.64 ms

Modos de funcionamiento (2)

- Modo Carácter o Dato:
  - Cuando el LCD recibe por el bus de datos un carácter ASCII a visualizar.
  - Para trabajar en modo Carácter
    - El pin RS debe estar a “1”
    - El pin R/W debe de estar a “0”
    - Una operación de este tipo tarda unos 40 µs.

Modos de funcionamiento (3)

- Modo de lectura del "*Busy Flag*" o LCD ocupado
  - El bit 7 del bus de datos del LCD informa al microcontrolador de que está ocupado. Este bit se denomina Busy Flag.
  - Para trabajar en este modo
  - RS debe de estar a "0"
  - R/W debe de estar a"1"
  - Si el bit 7 del bus de datos = "1" LCD Ocupado
  - Si el bit 7 del bus de datos ="0" LCD Libre

Modos de funcionamiento (4)

- Para un control sencillo, se pueden realizar pausas después de cada instrucción o envío de datos para no tener que lee el registro de estado.
- La ventaja de este sistema es que la línea R/W no es necesaria y puede conectarse directamente a masa.
- Los retardos empleados deberán de ser mayores de 1,64 ms si se trabaja en modo comando y mayor de 40 μs si se trabaja en modo dato.

Comandos de Control

COMANDO	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Clear Display	0	0	0	0	0	0	0	0	0	1
Return Home	0	0	0	0	0	0	0	0	1	*
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S
Display Control	0	0	0	0	0	0	1	D	C	B
Cursor and display Shift	0	0	0	0	0	1	S/C	R/L	*	*
Function Set	0	0	0	0	1	DL	N	F	*	*
Set CGRAM Address	0	0	0	1	CGRAM Address					
Set DDRAM Address	0	0	1	DDRAM Address						
Read Busy Flag	0	1	BF	DDRAM Address						
Write RAM	1	0	Write Data							
Read RAM	1	1	Read Data							

Comandos de control

- *Clear Display* (0 0 0 0 0 0 0 1). Borra pantalla y devuelve el cursor a la posición inicial (dirección 0 de la DDRAM).
- *Return Home* (0 0 0 0 0 0 1 x). “Cursor a casa” (dirección origen). Devuelve el cursor la posición original de la DDRAM (dirección 00H), quedando intacto su contenido.

Comandos de control

- *Entry Mode Set* (0 0 0 0 0 1 I/D S). Modo Entrada. Establece las características de escritura de los datos:
  - S = 0. *Shift*. El display no se desplaza al escribir un nuevo carácter.
  - S = 1. *Shift*. El display se desplaza al escribir un nuevo carácter. La pantalla se desplaza en el sentido indicado por el bit "I/D" cuando el cursor llega al filo de la pantalla.
  - I/D = 1. *Increment/Decrement*. Incremento automático de la posición del cursor. La posición de la DDRAM se incrementa automáticamente tras cada lectura o escritura a la misma,
  - I/D = 0. *Increment/Decrement*. Decremento de la posición del cursor. Se decrementa el puntero de la DDRAM.

Comandos de control

- *Display Control* (0 0 0 0 1 D C B). Control de la pantalla:
  - B = 0. *Blink OFF*, no hay efecto de parpadeo del cursor.
  - B = 1. *Blink ON*, efecto de parpadeo del cursor rectangular.
  - C = 0. *Cursor OFF*, el cursor no se visualiza.
  - C =1. *Cursor ON*, el cursor es visualizado.
  - D = 0. *Display OFF*, el display se apaga.
  - D = 1. *Display ON*, el display se enciende.

Comandos de control

- *Cursor and Display Shift* (0 0 0 1 S/C R/L x x). Control de los desplazamientos del cursor y de la pantalla:
  - R/L = 0. *Left*. A la izquierda.
  - R/L = 1. *Right*. A la derecha.
  - S/C = 0. El efecto de desplazamiento se aplica solo sobre el cursor sin alterar el contenido de la DDRAM.
  - S/C = 1. El efecto de desplazamiento se aplica sobre todo el display.

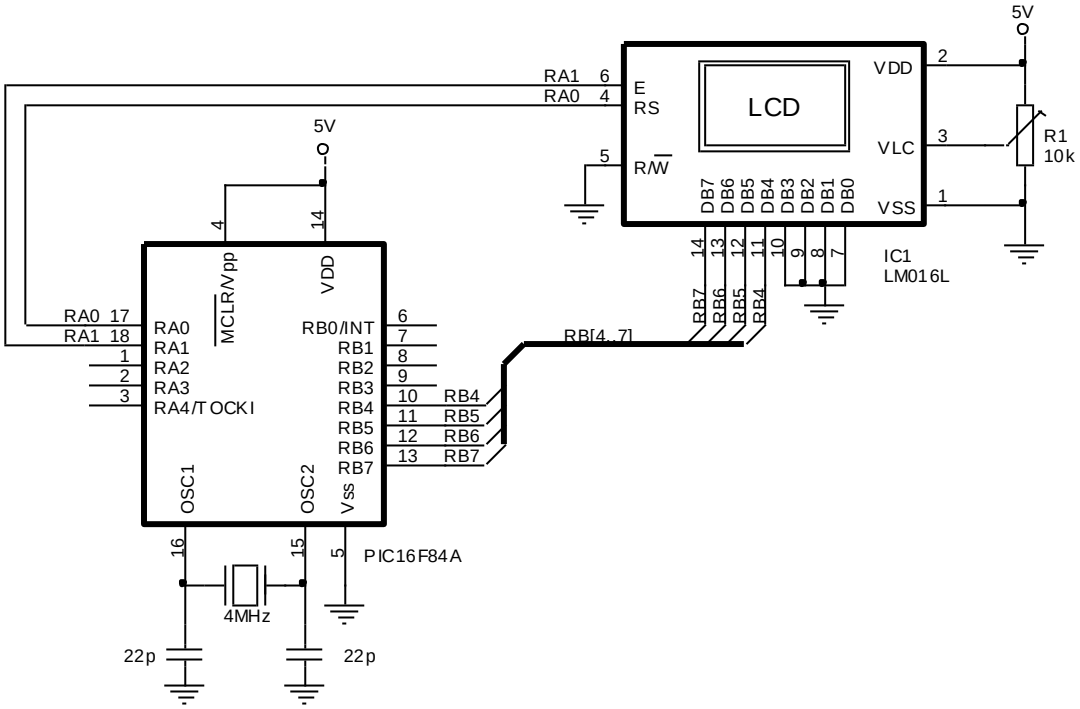
Comandos de control

- *Function Set* (0 0 1 DL N F x x x). Características de control hardware de la pantalla LCD.
  - F = 0. *Font*. Caracteres de 5 x 7 puntos.
  - F = 1. *Font*. Caracteres de 5 x 10 puntos.
  - N = 0. *Number Line*. Pantalla LCD de 1 línea.
  - N = 1. *Number Line*. Pantalla LCD de 2 líneas.
  - DL = 0. *Data Length*. Comunicación con 4 bits. Se indicar al LCD que solamente se van a utilizar las líneas DB7, DB6, DB5 y DB4 para enviarle los datos, y que se hará enviando primero el nibble alto, y a continuación el nibble bajo del dato
  - DL = 1. *Data Length*. Comunicación con 8 bits.

Comandos de control

- *Set CGRAM Address*. Se va a escribir sobre la dirección CGRAM señalada.
- *Set DDRAM Address* (1 d d d d d d d). Se va a escribir sobre la dirección DDRAM señalada. Esta instrucción se utiliza para modificar el puntero a la DDRAM. Así por ejemplo, si la dirección es la 00h se escribirá en la primera línea
- *Read Busy Flag*. Lee el BF indicando si hay una operación interna en curso y lee además, el contenido de la dirección DDRAM apuntada.

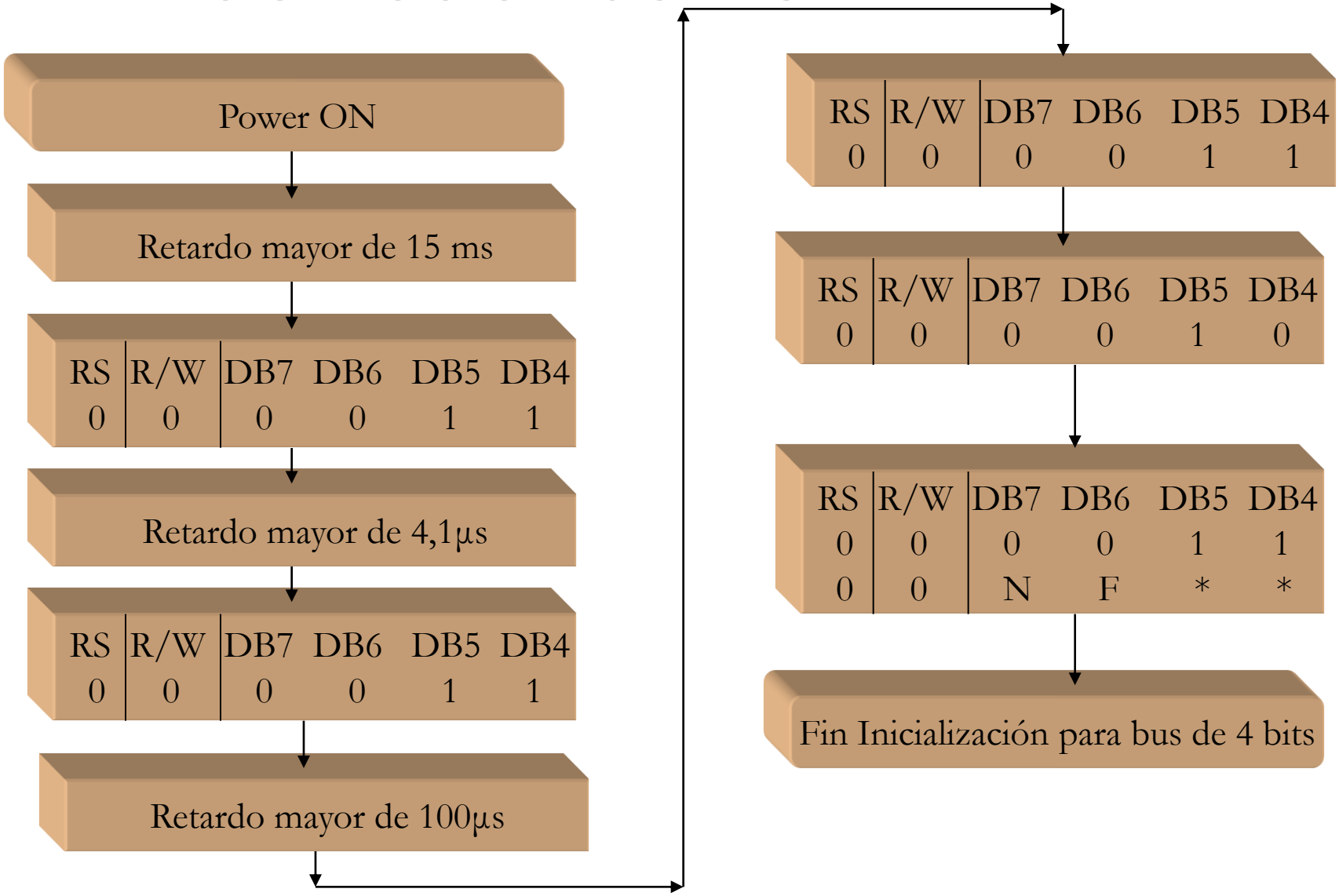
Conexión del LCD mediante 4 bits



Librería de Subrutinas

- "LCD_Inicializa".
  - Inicializa el modulo LCD para su correcto funcionamiento. Configura funciones de LCD, produce un reset por software, borra la memoria DDRAM y enciende la pantalla.
  - Es necesario ejecutar esta subrutina al principio de los programas que vayan a utilizar la visualización mediante LCD.
  - El fabricante especifica que para garantizar una correcta inicialización, debe realizarse como indica.

Inicialización del LCD



Librería de Subrutinas

- "LCD_Carácter". Visualiza en la posición actual del cursor el código ASCII del dato contenido en el registro W.
- "LCD_Borra". Borra toda la pantalla y pone el cursor a principio de la línea 1.
- "LCD_Linea1". Envía el cursor al principio de la línea 1.
- "LCD_Linea2". Envía el cursor al principio de la línea 2.
- "LCD_PosicionLinea1". Envía el cursor a la posición de la línea 1 indicada por (W). Por ejemplo si (W)=0x08, al ejecutar esta subrutina el cursor se irá al centro de la línea en una pantalla de 16 caracteres.

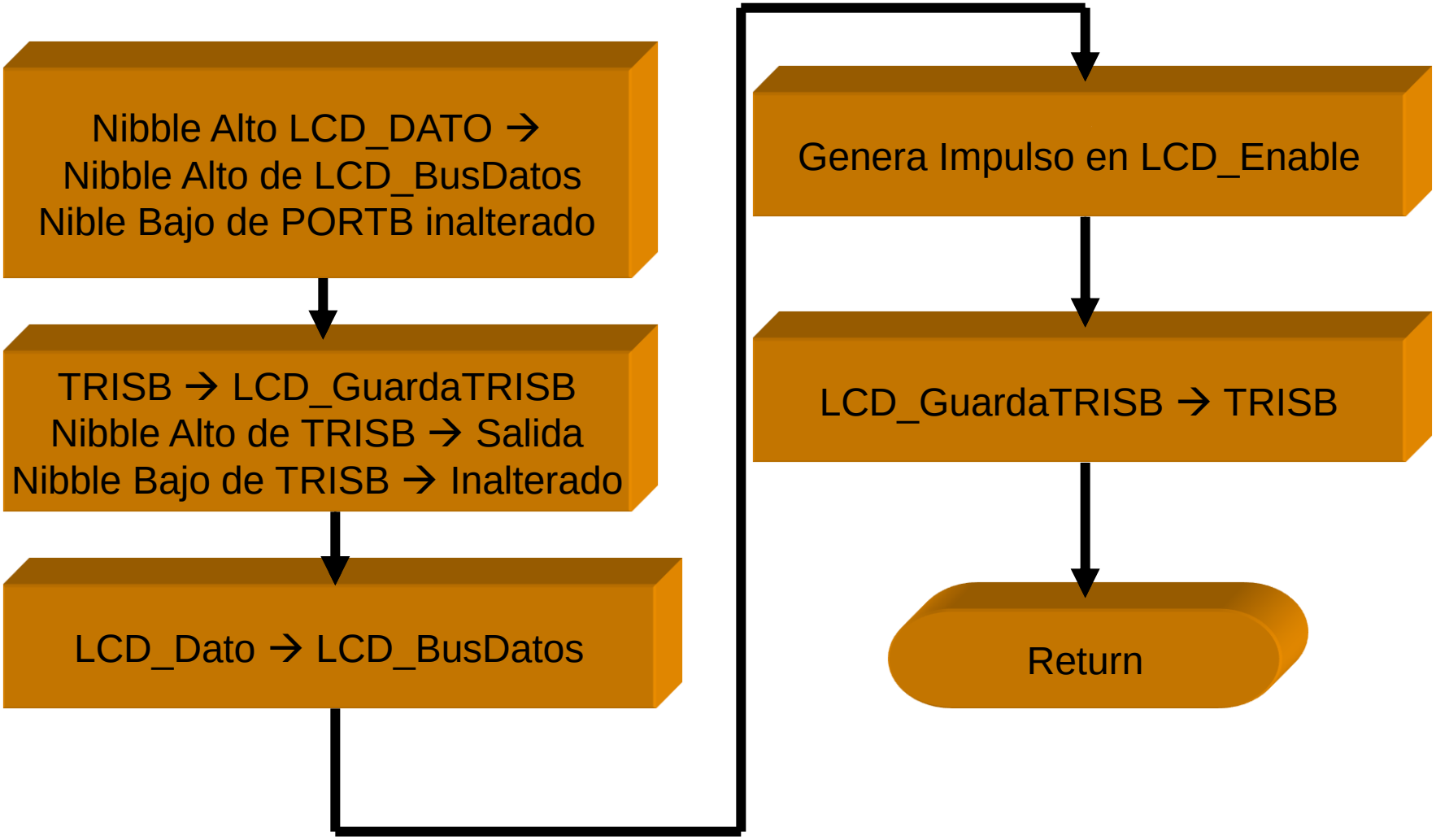
Librería de Subrutinas

- "LCD_PosicionLinea2". Igual que el anterior para la línea 2.
- "LCD_LineaEnBlanco". Visualiza una línea en blanco.
- "LCD_DosEspaciosBlanco". Visualiza dos espacios en blanco.

```
;*****************************************************************************************************
;LCD_4bits.inc
; Estas subrutinas permiten realizar las tareas básicas de control de un módulo LCD de 2
; líneas por 16 caracteres, compatible con el modelo LM016L.
;; El visualizador LCD está conectado al Puerto B del PIC mediante un bus de 4 bits. Las
; conexiones son:
;    -          Las 4 líneas superiores del módulo LCD, pines <DB7:DB4>  se conectan a las 4
;               líneas superiores del Puerto B del PIC, pines <RB7:RB4>.
;    -          Pin RS del LCD a la línea RA0 del PIC.
;    -          Pin R/W del LCD a la línea RA1 del PIC, o a masa.
;    -          Pin Enable del LCD a la línea RA2 del PIC.
;
; Se utilizan llamadas a subrutinas de retardo de tiempo localizadas en la librería RETARDOS.INC.
; ZONA DE DATOS ********************************************************************
               CBLOCK
               LCD_Dato
               LCD_GuardaDato
               LCD_GuardaTRISB
               LCD_Auxiliar1
               LCD_Auxiliar2
               ENDC
LCD_CaracteresPorLinea          EQU         .16          ; Número de caracteres por línea de la pantalla.

#DEFINE  LCD_PinRS              PORTA,0
#DEFINE  LCD_PinRW              PORTA,1
#DEFINE  LCD_PinEnable          PORTA,2
#DEFINE  LCD_BusDatos           PORTB
```

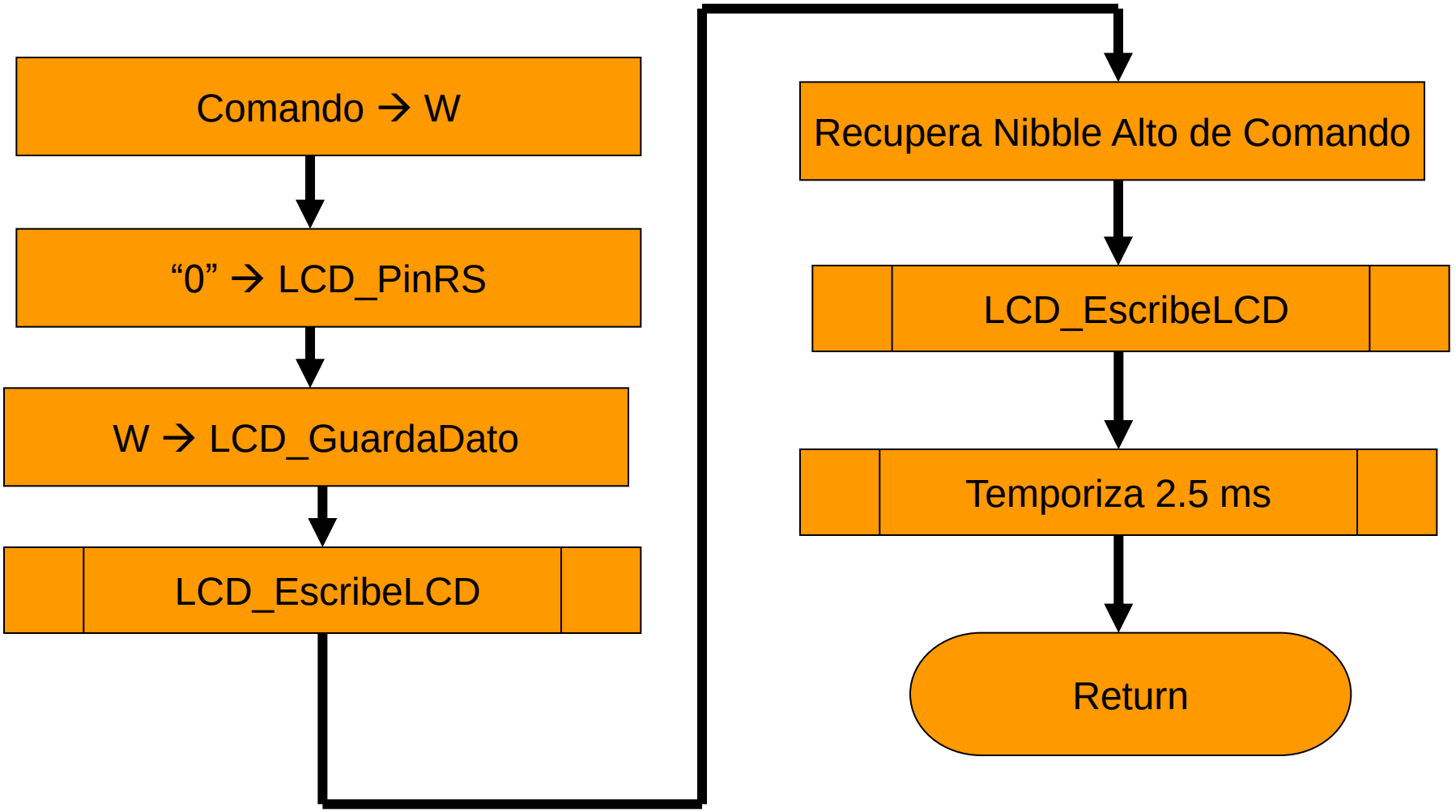
```
; Subrutina "LCD_Inicializa" ---------------------------------------------------------
;; Inicialización del módulo LCD: Configura funciones del LCD, produce reset por software,
; borra memoria y enciende pantalla. El fabricante especifica que para garantizar la
; configuración inicial hay que hacerla como sigue:
LCD_Inicializa
        bsf     STATUS,RP0              ; Configura las líneas conectadas al pines RS,
        bcf     LCD_PinRS               ; R/W y E.
        bcf     LCD_PinEnable
        bcf     LCD_PinRW
        bcf     STATUS,RP0
        bcf     LCD_PinRW               ; En caso de que esté conectado le indica
                                        ; que se va a escribir en el LCD.
        bcf     LCD_PinEnable           ; Impide funcionamiento del LCD poniendo E=0.
        bcf     LCD_PinRS               ; Activa el Modo Comando poniendo RS=0.
        call    Retardo_20ms
        movlw   b'00110000'
        call    LCD_EscribeLCD          ; Escribe el dato en el LCD.
        call    Retardo_5ms
        movlw   b'00110000'
        call    LCD_EscribeLCD
        call    Retardo_200micros
        movlw   b'00110000'
        call    LCD_EscribeLCD
        movlw   b'00100000'             ; Interface de 4 bits.
        call    LCD_EscribeLCD
; Ahora configura el resto de los parámetros:
        call    LCD_2Lineas4Bits5x7     ; LCD de 2 líneas y caracteres de 5x7 puntos.
        call    LCD_Borra               ; Pantalla encendida y limpia. Cursor al principio
        call    LCD_CursorOFF           ; de la línea 1. Cursor apagado.
        call    LCD_CursorIncr          ; Cursor en modo incrementar.
        return
```

LCD_EscribeLCD



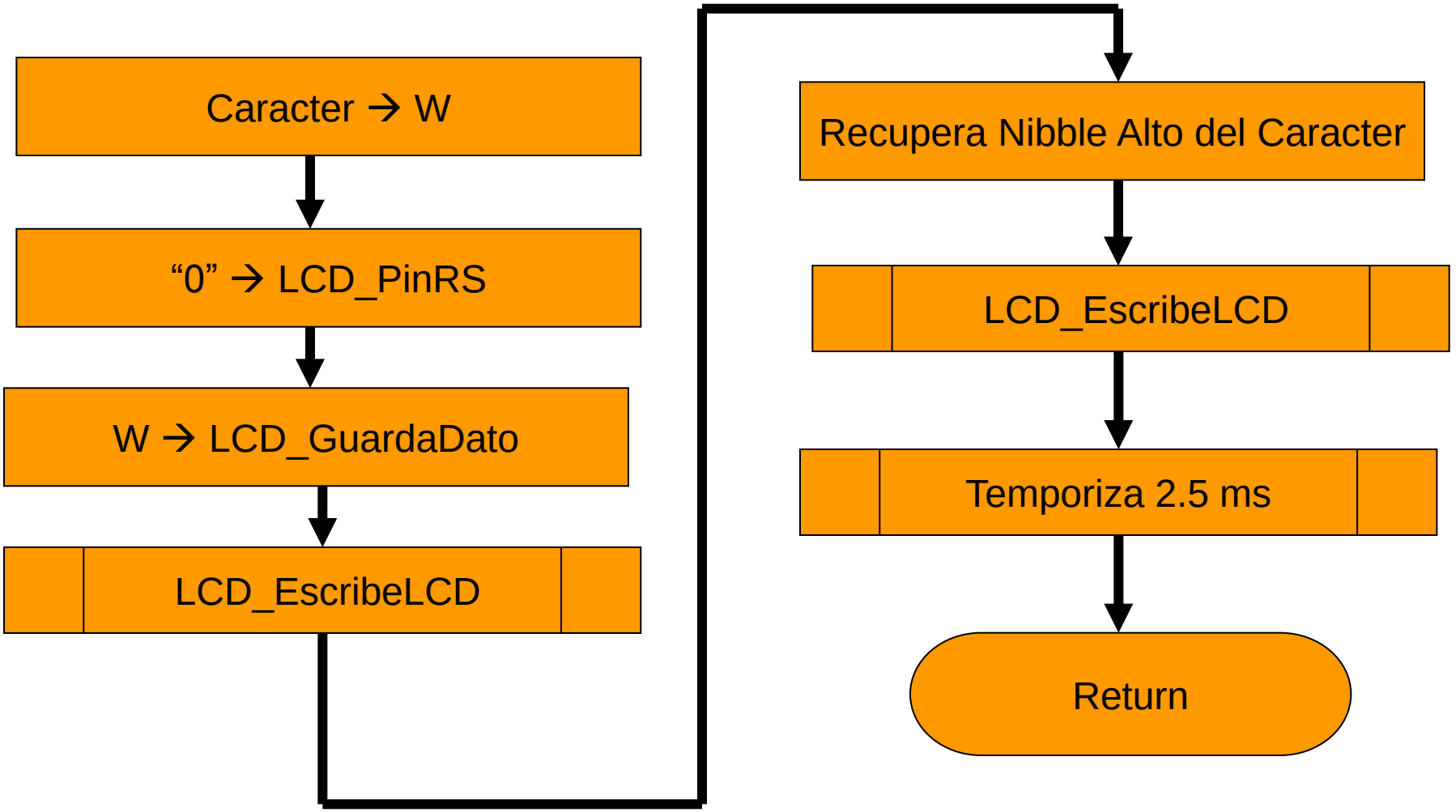
```
; Subrutina "LCD_EscribeLCD" ---------------------------------------------------------
; Envía el dato del registro de trabajo W al bus de dato y produce un pequeño pulso en el pin
; Enable del LCD. Para no alterar el contenido de las líneas de la parte baja del Puerto B que
; no son utilizadas para el LCD (pines RB3:RB0), primero se lee estas líneas y después se
; vuelve a enviar este dato sin cambiarlo.

LCD_EscribeLCD
        andlw   b'11110000'; Se queda con el nibble alto del dato que es el
        movwf   LCD_Dato            ; que hay que enviar y lo guarda.
        movf    LCD_BusDatos,W      ; Lee la información actual de la parte baja
        andlw   b'00001111'; del Puerto B, que no se debe alterar.
        iorwf   LCD_Dato,F          ; Enviará la parte alta del dato de entrada
                                    ; y en la parte baja lo que había antes.
        bsf     STATUS,RP0          ; Acceso al Banco 1.
        movf    TRISB,W             ; Guarda la configuración que tenía antes TRISB.
        movwf   LCD_GuardaTRISB
        movlw   b'00001111'; Las 4 líneas inferiores del Puerto B se dejan
        andwf   PORTB,F             ; como estaban y las 4 superiores como salida.
        bcf     STATUS,RP0          ; Acceso al Banco 0.
;
        movf    LCD_Dato,W          ; Recupera el dato a enviar.
        movwf   LCD_BusDatos        ; Envía el dato al módulo LCD.
        bsf     LCD_PinEnable       ; Permite funcionamiento del LCD mediante un pequeño
        bcf     LCD_PinEnable       ; pulso y termina impidiendo el funcionamiento del LCD.
        bsf     STATUS,RP0          ; Acceso al Banco 1. Restaura el antiguo valor en
        movf    LCD_GuardaTRISB,W   ; la configuración del Puerto B.
        movwf   PORTB               ; Realmente es TRISB.
        bcf     STATUS,RP0          ; Acceso al Banco 0.
        return
```

Escritura de un comando



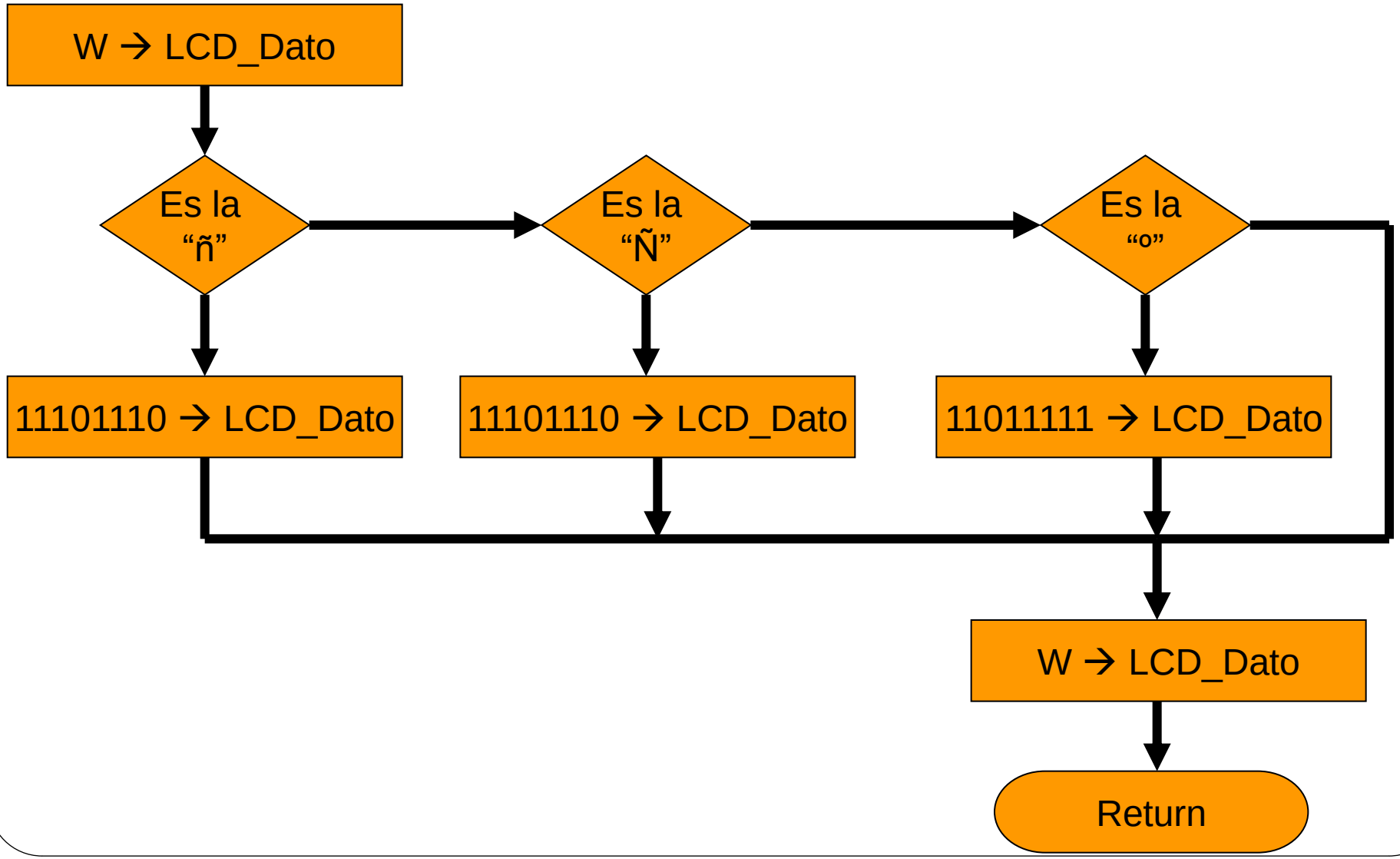
```
; Subrutinas variadas para el control del módulo LCD ------------------------------------
;Los comandos que pueden ser ejecutados son:
LCD_CursorIncr                                  ; Cursor en modo incrementar.
        movlw   b'00000110'
        goto    LCD_EnviaComando
LCD_Linea1                                      ; Cursor al principio de la Línea 1.
        movlw   b'10000000'                     ; Dirección 00h de la DDRAM
        goto    LCD_EnviaComando
LCD_Linea2                                      ; Cursor al principio de la Línea 2.
        movlw   b'11000000'                     ; Dirección 40h de la DDRAM
        goto    LCD_EnviaComando
LCD_PosicionLinea1                              ; Cursor a posición de la Línea 1, a partir de la
        iorlw   b'10000000'                     ; dirección 00h de la DDRAM más el valor del
        goto    LCD_EnviaComando                ; registro W.
LCD_PosicionLinea2                              ; Cursor a posición de la Línea 2, a partir de la
        iorlw   b'11000000'                     ; dirección 40h de la DDRAM más el valor del
        goto    LCD_EnviaComando                ; registro W.
LCD_OFF                                         ; Pantalla apagada.
        movlw   b'00001000'
        goto    LCD_EnviaComando
LCD_CursorON                                    ; Pantalla encendida y cursor encendido.
        movlw   b'00001110'
        goto    LCD_EnviaComando
LCD_CursorOFF                                   ; Pantalla encendida y cursor apagado.
        movlw   b'00001100'
        goto    LCD_EnviaComando
LCD_Borra                               ; Borra toda la pantalla, memoria DDRAM y pone el
        movlw   b'00000001'             ; cursor a principio de la línea 1.
        goto    LCD_EnviaComando
LCD_2Lineas4Bits5x7                     ; Define la pantalla de 2 líneas, con caracteres
        movlw   b'00101000'             ; de 5x7 puntos y conexión al PIC mediante bus de
;       goto    LCD_EnviaComando        ; 4 bits.
```

Escritura de un caracter



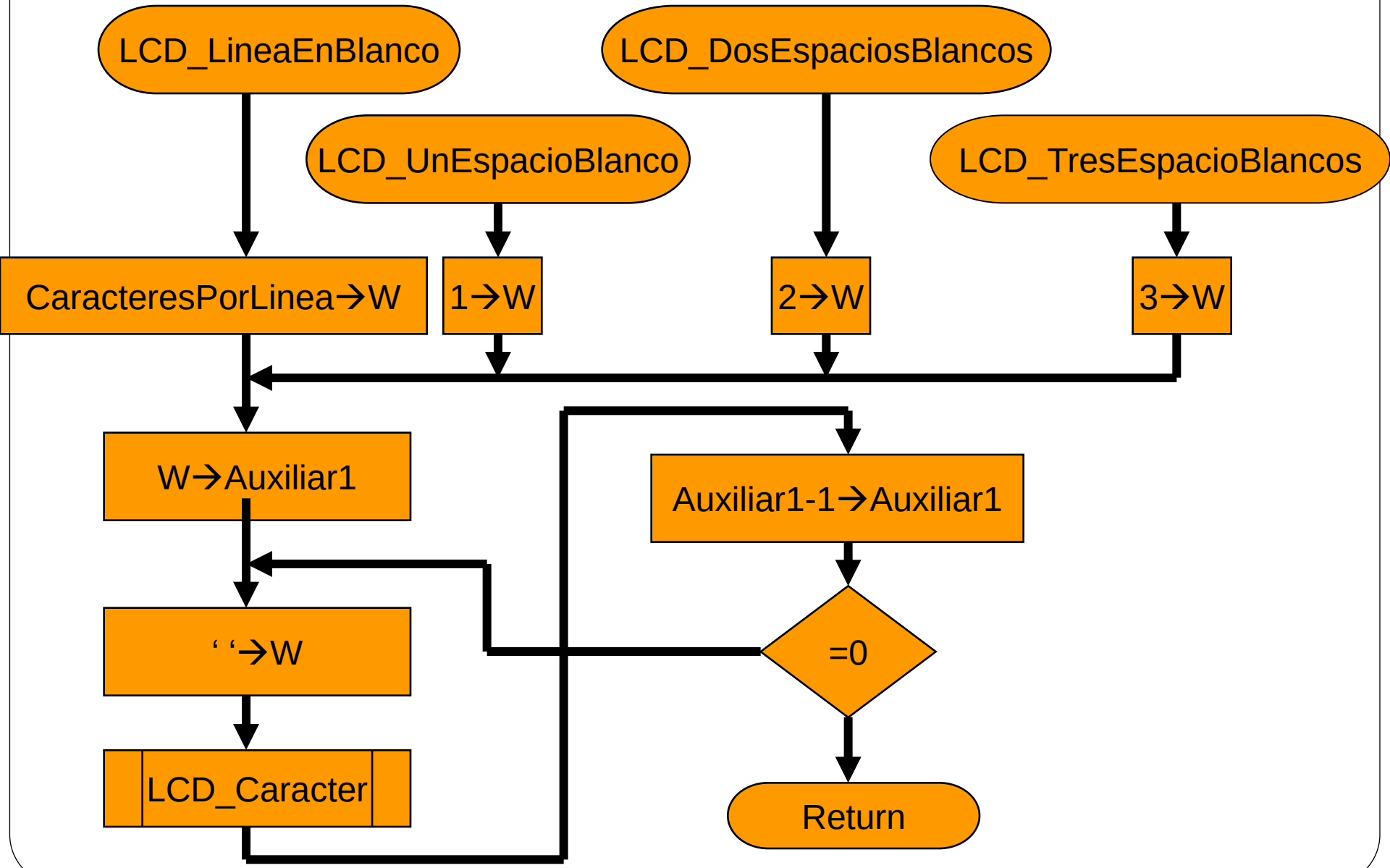
```
; Subrutinas "LCD_EnviaComando" y "LCD_Caracter" ---------------------------------
;
; "LCD_EnviaComando". Escribe un comando en el registro del módulo LCD. La palabra de
; comando ha sido entregada a través del registro W.  Trabaja en Modo Comando.
; "LCD_Caracter". Escribe en la memoria DDRAM del LCD el carácter ASCII introducido a
; a través del registro W. Trabaja en Modo Dato.
;
LCD_EnviaComando
        bcf     LCD_PinRS               ; Activa el Modo Comando, poniendo RS=0.
        goto    LCD_Envia
LCD_Caracter
        bsf     LCD_PinRS               ; Activa el "Modo Dato", poniendo RS=1.
        call    LCD_CodigoCGROM  ;       Obtiene el código para correcta visualización.
LCD_Envia
        movwf   LCD_GuardaDato          ; Guarda el dato a enviar.
        call    LCD_EscribeLCD          ; Primero envía el nibble alto.
        swapf   LCD_GuardaDato,W        ; Ahora envía el nibble bajo. Para ello pasa el
                                ; nibble bajo del dato a enviar a parte alta del byte.

        call    LCD_EscribeLCD  ; Se envía al visualizador LCD.
        btfss   LCD_PinRS       ; Debe garantizar una correcta escritura manteniendo
        call    Retardo_2ms     ; 2 ms en modo comando y 50 µs en modo cáracter.
        call    Retardo_50micros
        return
```

LCD_CodigoCGROM



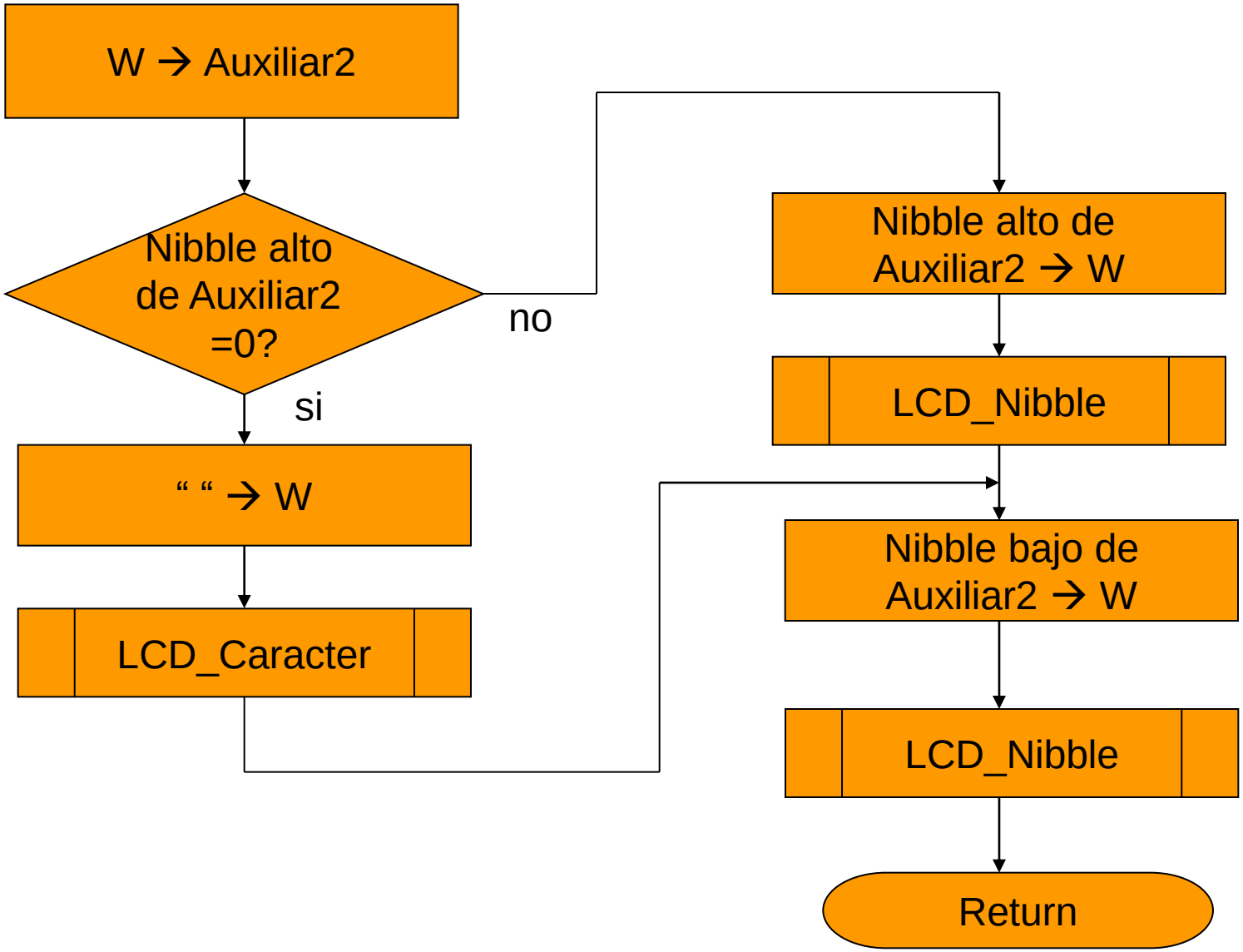
```
; Subrutina "LCD_CodigoCGROM" ----------------------------------------------------------
; A partir del carácter ASCII número 127 los códigos de los caracteres definidos en la
; tabla CGROM del LM016L no coinciden con los códigos ASCII. Así por ejemplo, el código
; ASCII de la "Ñ" en la tabla CGRAM del LM016L es EEh.
; Esta subrutina convierte los códigos ASCII de la "Ñ", "º" y otros, a códigos CGROM para que
; que puedan ser visualizado en el módulo LM016L.
; Entrada:      En (W) el código ASCII del carácter que se desea visualizar.
; Salida:       En (W) el código definido en la tabla CGROM.
LCD_CodigoCGROM
                movwf   LCD_Dato            ; Guarda el valor del carácter y comprueba si es
LCD_EnheMinuscula                           ; un carácter especial.
                sublw   'ñ'                         ; ¿Es la "ñ"?
                btfss   STATUS,Z
                goto    LCD_EnheMayuscula   ; No es "ñ".
                movlw   b'11101110'         ; Código CGROM de la "ñ".
                movwf   LCD_Dato
                goto    LCD_FinCGROM
LCD_EnheMayuscula
                movf    LCD_Dato,W          ; Recupera el código ASCII de entrada.
                sublw   'Ñ'                         ; ¿Es la "Ñ"?
                btfss   STATUS,Z
                goto    LCD_Grado           ; No es "Ñ".
                movlw   b'11101110'         ; Código CGROM de la "ñ". (No hay símbolo para
                movwf   LCD_Dato            ; la "Ñ" mayúscula en la CGROM).
                goto    LCD_FinCGROM
LCD_Grado
                movf    LCD_Dato,W          ; Recupera el código ASCII de entrada.
                sublw   'º'                         ; ¿Es el símbolo "º"?
                btfss   STATUS,Z
                goto    LCD_FinCGROM                ; No es "º".
                movlw   b'11011111'         ; Código CGROM del símbolo "º".
                movwf   LCD_Dato
LCD_FinCGROM
                movf    LCD_Dato,W          ; En (W) el código buscado.
                return
```

LCD_DosEspaciosBlanco y LCD_LineaBlanco



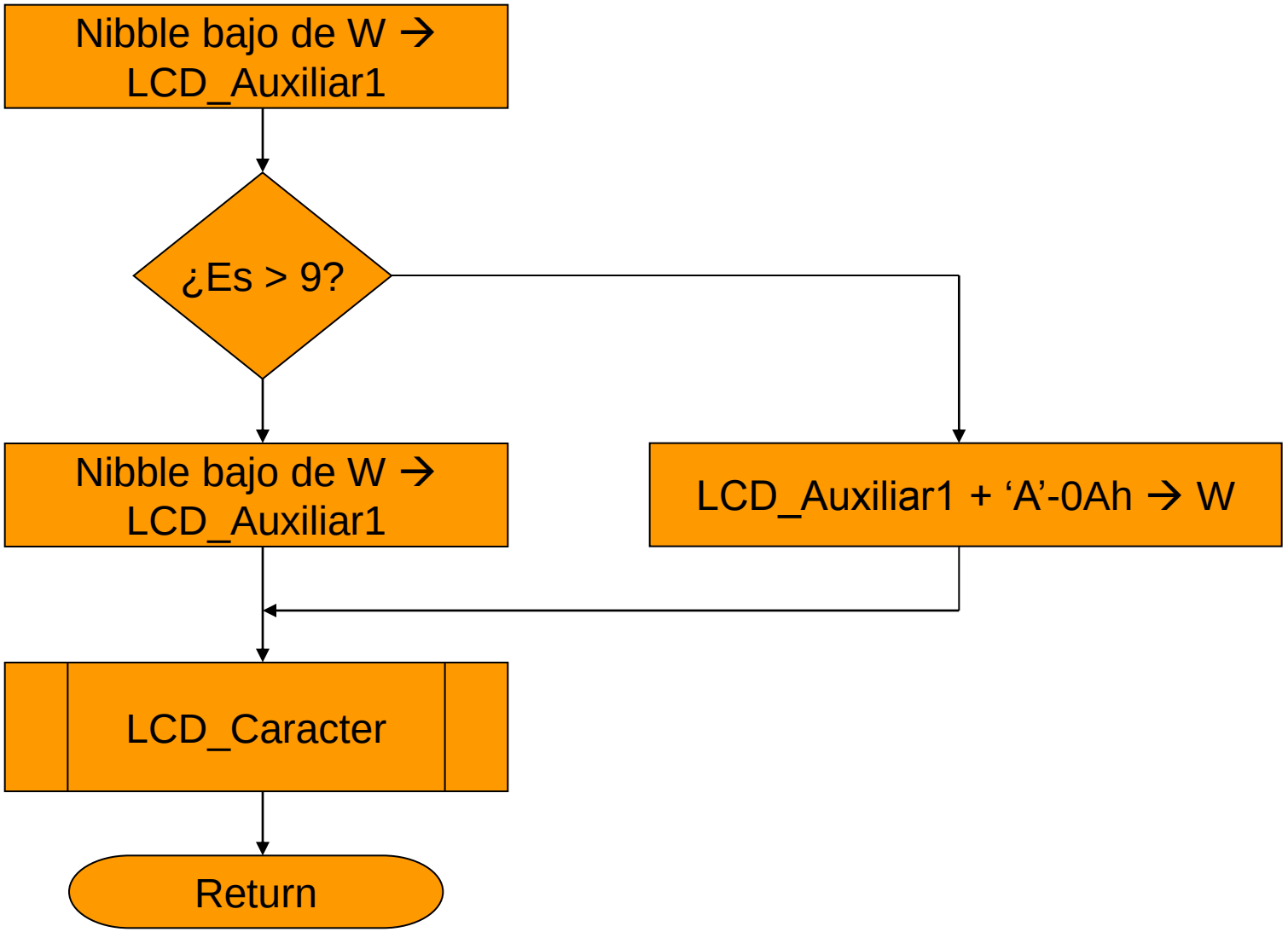
```
; Subrutina "LCD_DosEspaciosBlancos" y "LCD_LineaBlanco" ------------------------------
;
; Visualiza espacios en blanco.

LCD_LineaEnBlanco
                movlw       LCD_CaracteresPorLinea
                goto        LCD_EnviaBlancos
LCD_UnEspacioBlanco
                movlw       .1
                goto        LCD_EnviaBlancos
LCD_DosEspaciosBlancos
                movlw       .2
                goto        LCD_EnviaBlancos
LCD_TresEspaciosBlancos
                movlw       .3
LCD_EnviaBlancos
                movwf       LCD_Auxiliar1           ; (LCD_Auxiliar1) se utiliza como contador.
LCD_EnviaOtroBlanco
                movlw       ' '                     ; Esto es un espacio en blanco.
                call        LCD_Caracter            ; Visualiza tanto espacios en blanco como se
                decfsz      LCD_Auxiliar1,F         ; haya cargado en (LCD_Auxiliar1).
                goto        LCD_EnviaOtroBlanco
                return
```

LCD_ByteCompleto y LCD_Byte



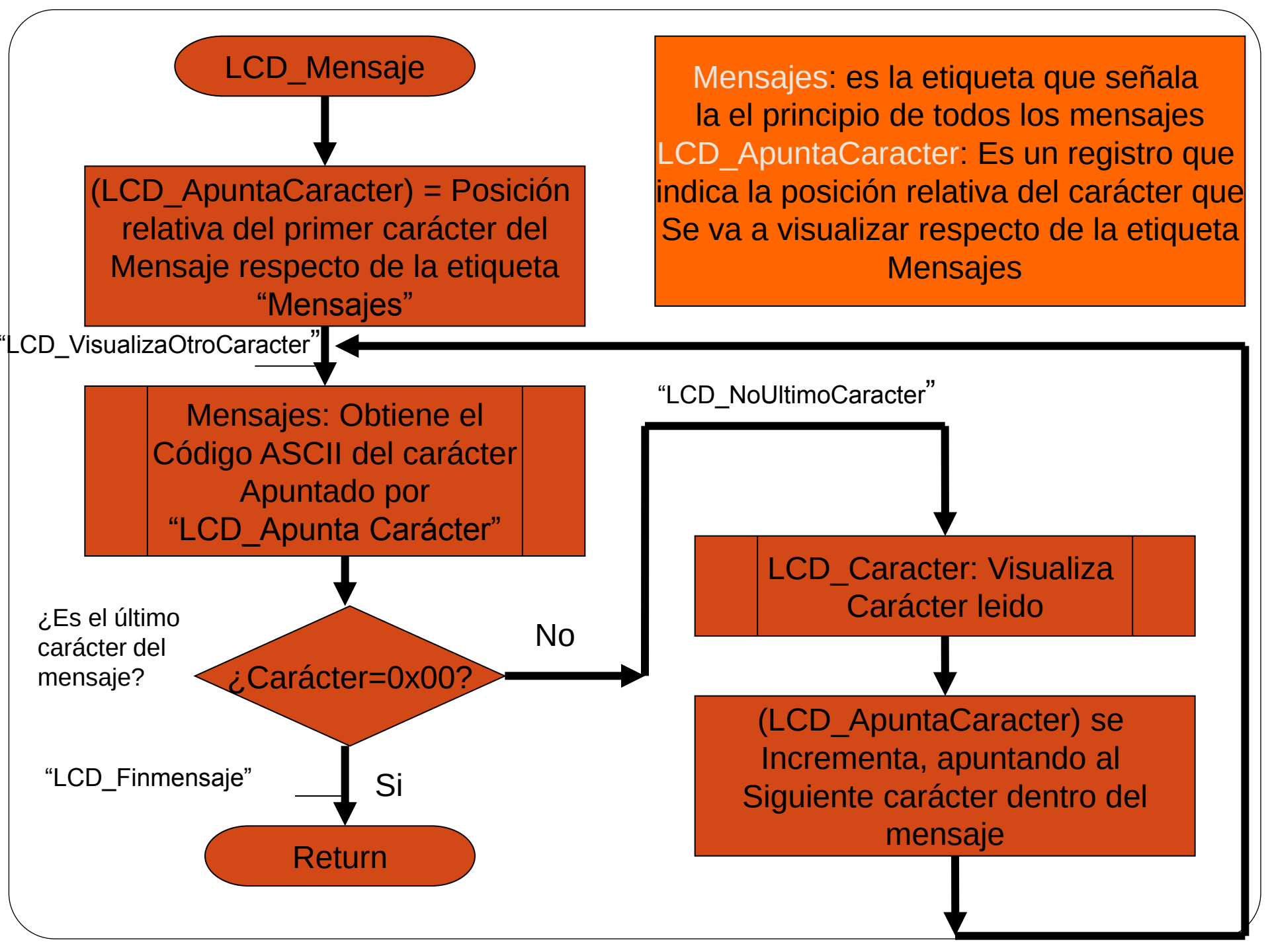
```
; Subrutinas "LCD_ByteCompleto" y "LCD_Byte" ----------------------------------------
;
; Subrutina "LCD_ByteCompleto", visualiza el byte que almacena el registro W en el
; lugar actual de la pantalla. Por ejemplo, si (W)=b'10101110' visualiza "AE".
;
; Subrutina "LCD_Byte" igual que la anterior, pero en caso de que el nibble alto sea cero
; visualiza en su lugar un espacio en blanco. Por ejemplo si (W)=b'10101110' visualiza "AE"
; y si (W)=b'00001110', visualiza " E" (un espacio blanco delante).
;
; Utilizan la subrutina "LCD_Nibble" que se analiza más adelante.
;
LCD_Byte
            movwf   LCD_Auxiliar2               ; Guarda el valor de entrada.
            andlw   b'11110000'         ; Analiza si el nibble alto es cero.
            btfss   STATUS,Z                    ; Si es cero lo apaga.
            goto    LCD_VisualizaAlto           ; No es cero y lo visualiza.
            movlw   ' '                         ; Visualiza un espacio en blanco.
            call    LCD_Caracter
            goto    LCD_VisualizaBajo

LCD_ByteCompleto
            movwf   LCD_Auxiliar2               ; Guarda el valor de entrada.
LCD_VisualizaAlto
            swapf   LCD_Auxiliar2,W             ; Pone el nibble alto en la parte baja.
            call    LCD_Nibble                  ; Lo visualiza.
LCD_VisualizaBajo
            movf    LCD_Auxiliar2,W             ; Repite el proceso con el nibble bajo.
;           call    LCD_Nibble                  ; Lo visualiza.
;           return
```

LCD_Nibble


Nibble bajo de W → LCD_Auxiliar1
¿Es > 9?
Nibble bajo de W → LCD_Auxiliar1
LCD_Auxiliar1 + ‘A’-0Ah → W
LCD_Caracter
Return

```
; Subrutina "LCD_Nibble" ---------------------------------------------------------
;
; Visualiza en el lugar actual de la pantalla, el valor hexadecimal que almacena en el nibble
; bajo del registro W. El nibble alto de W no es tenido en cuenta. Ejemplos:
; - Si (W)=b'01010110', se visualizará "6".
; - Si (W)=b'10101110', se visualizará "E".
;
LCD_Nibble
            andlw   b'00001111'         ; Se queda con la parte baja.
            movwf   LCD_Auxiliar1               ; Lo guarda.
            sublw   0x09                        ; Comprueba si hay que representarlo con letra.
            btfss   STATUS,C
            goto    LCD_EnviaByteLetra
            movf    LCD_Auxiliar1,W
            addlw   '0'                         ; El número se pasa a carácter ASCII sumándole
            goto    LCD_FinVisualizaDigito      ; el ASCII del cero y lo visualiza.
LCD_EnviaByteLetra
            movf    LCD_Auxiliar1,W
            addlw   'A'-0x0A                    ; Sí, por tanto, se le suma el ASCII de la 'A'.
LCD_FinVisualizaDigito
            goto    LCD_Caracter                ; Y visualiza el carácter. Se hace con un "goto"
                                                ; para no sobrecargar la pila.
```

```
;************************************ LCD_01.asm ************************************
;; En la pantalla LCD se visualizará el mensaje "HOLA". Al terminar de escribir la frase
; el PIC entrará en modo "Standby" o "Bajo Consumo" mediante la instrucción "sleep".
; ZONA DE DATOS ********************************************************************
            LIST    P=16F876            ; Tipo de procesador.
            INCLUDE <P16F876.INC>       ; Definición de registros.
            CBLOCK  0x20
            ENDC
; ZONA DE CÓDIGOS ******************************************************************
            ORG     0x05
INICIO      bsf     STATUS,RP0          ;Configura el PORTA como E/S digitales
            movlw   b'00000110'
            movwf   ADCON1
            bcf     STATUS,RP0
            call    LCD_Inicializa
            movlw   'H'
            call    LCD_Caracter
            movlw   'O'
            call    LCD_Caracter
            movlw   'L'
            call    LCD_Caracter
            movlw   'A'
            call    LCD_Caracter
            sleep                               ; Entra en modo "Bajo Consumo".
            INCLUDE <LCD_4BIT.INC>              ; Subrutinas de control de la LCD.
            INCLUDE <RETARDOS.INC>              ; Subrutinas de retardo.

            ORG     0x1F00
            bcf     PCLATH,4
            bcf     PCLATH,3            ;Selecciona la página 0
            goto    INICIO              ;Salto a la dirección de INICIO del programa
            END                                 ; Fin del programa.
```


LCD_Mensaje
(LCD_ApuntaCaracter) = Posición relativa del primer carácter del Mensaje respecto de la etiqueta "Mensajes"
Mensajes: es la etiqueta que señala la el principio de todos los mensajes
LCD_ApuntaCaracter: Es un registro que indica la posición relativa del carácter que Se va a visualizar respecto de la etiqueta Mensajes
"LCD_VisualizaOtroCaracter"
Mensajes: Obtiene el Código ASCII del carácter Apuntado por "LCD_Apunta Carácter"
¿Es el último carácter del mensaje?
¿Carácter=0x00?
No
"LCD_NoUltimoCaracter"
LCD_Caracter: Visualiza Carácter leido
(LCD_ApuntaCaracter) se Incrementa, apuntando al Siguiente carácter dentro del mensaje
"LCD_Finmensaje"
Si
Return

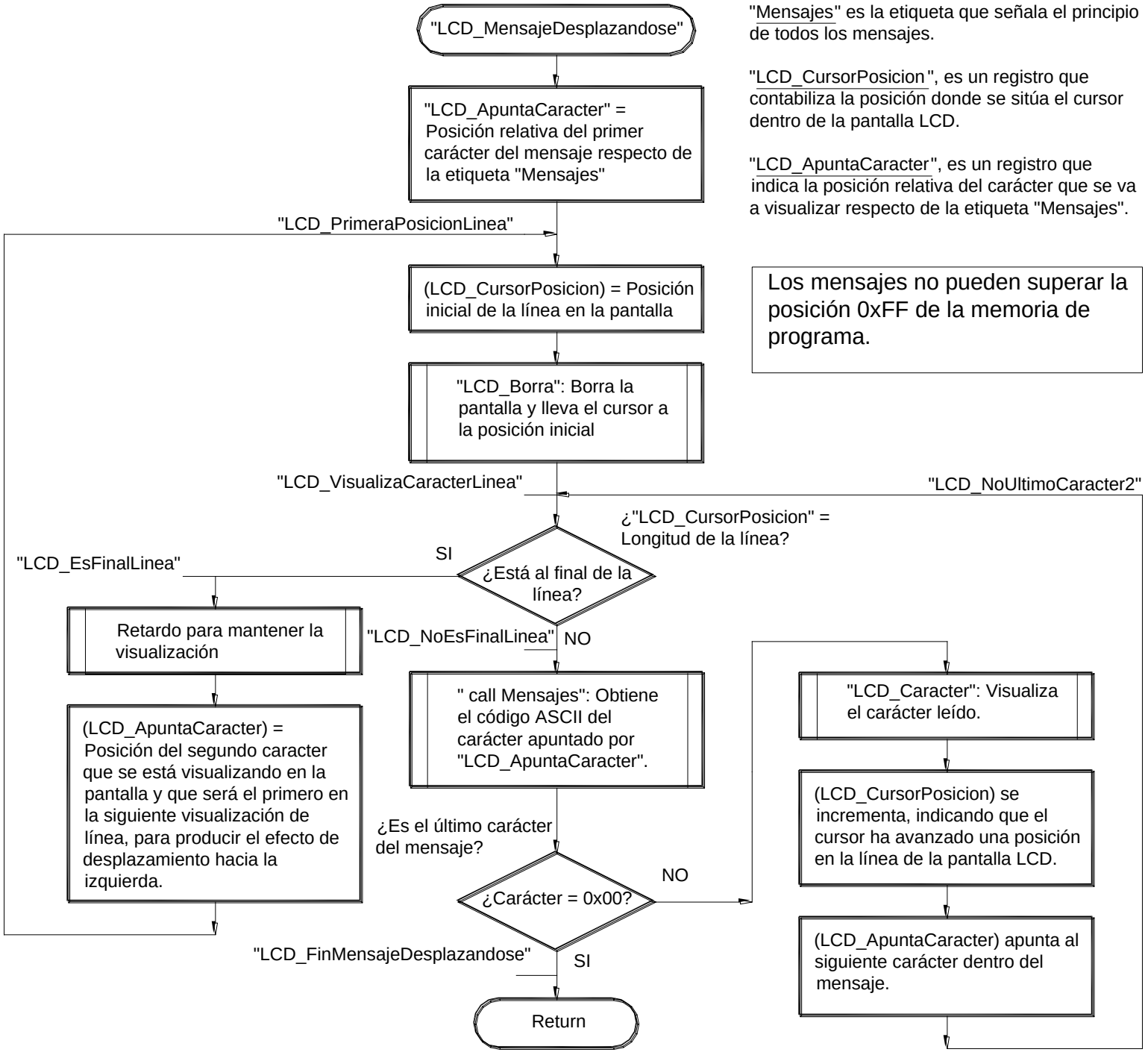
LCD_MENS.INC

```
; Librería de subrutinas para el manejo de mensajes a visualizar en un visualizador LCD.

        CBLOCK
        LCD_ApuntaCaracter      ; Indica la posición del carácter a visualizar
                                ; respecto del comienzo de todos los mensajes,
                                ; (posición de la etiqueta "Mensajes").
        LCD_ValorCaracter       ; Código ASCII del carácter a
        ENDC                    ; visualizar.

; Los mensajes tienen que estar situados dentro de las 256 primeras posiciones de la
; memoria de programa, es decir, no pueden superar la dirección 0FFh.
```

```
; Subrutina "LCD_Mensaje" ---------------------------------------------------------------
; Visualiza por pantalla el mensaje apuntado por el registro W.  Los mensajes deben localizarse dentro de una
; zona encabezada por la etiqueta "Mensajes" y que  tenga la siguiente estructura:
; Mensajes                                        ; ¡Etiqueta obligatoria!
;               addwf       PCL,F
; Mensaje0                                        ; Posición inicial del mensaje.
;               DT ".. ..", 0x00                  ; Mensaje terminado en 0x00.
; Mensaje1
;               ...
;               ...
; FinMensajes
; La llamada a esta subrutina se realizará siguiendo este ejemplo:
;               movlw       Mensaje0                        ; Carga la posición del mensaje.
;               call        LCD_Mensaje                     ; Visualiza el mensaje.
LCD_Mensaje
                movwf       LCD_ApuntaCaracter              ; Posición del primer carácter del mensaje.
                movlw       Mensajes                        ; Halla la posición relativa del primer carácter
                subwf       LCD_ApuntaCaracter,F            ; del mensaje respecto de etiqueta "Mensajes".
                decf        LCD_ApuntaCaracter,F            ; Compensa la posición que ocupa "addwf PCL,F".
LCD_VisualizaOtroCaracter
                movf        LCD_ApuntaCaracter,W
                call        Mensajes                        ; Obtiene el código ASCII del carácter apuntado.
                movwf       LCD_ValorCaracter               ; Guarda el valor de carácter.
                movf        LCD_ValorCaracter,F             ; Lo único que hace es posicionar flag Z. En caso
                btfsc       STATUS,Z                        ; que sea "0x00", que es código indicador final
                goto        LCD_FinMensaje                  ; de mensaje, sale fuera.
LCD_NoUltimoCaracter
                call        LCD_Caracter                    ; Visualiza el carácter ASCII leído.
                incf        LCD_ApuntaCaracter,F            ; Apunta a la posición del siguiente carácter
                goto        LCD_VisualizaOtroCaracter       ; dentro del mensaje.
LCD_FinMensaje
                return                                      ; Vuelve al programa principal.
```

```
;********************************** Mensaje_02.asm ***********************************
;En la pantalla del módulo LCD se visualiza un mensaje de menos de 16 caracteres grabado
; en la memoria ROM mediante la directiva DT. Utiliza la subrutina LCD_Mensaje de la
; librería LCD_MENS.INC
            LIST        P=16F876
            INCLUDE <P16F876A.INC>
            CBLOCK  0x20
            ENDC
; ZONA DE CÓDIGOS ***************************************************************
            ORG         0x05
Inicio      bsf         STATUS,RP0
            movlw       b'00000110'
            movwf       ADCON1
            bcf         STATUS,RP0
            call        LCD_Inicializa
            movlw       Mensaje0                    ; Apunta dónde se encuentra el mensaje.
            call        LCD_Mensaje                 ; Visualiza el mensaje.
            sleep                                   ; Pasa a modo bajo consumo.
; Mensajes ----------------------------------------------------------------------
Mensajes
            addwf       PCL,F
Mensaje0
            DT "Hola!, que tal?   ", 0x00
            INCLUDE  <LCD_4BIT.INC>
            INCLUDE  <LCD_MENS.INC>
            INCLUDE  <RETARDOS.INC>
            ORG         0x1F00
            bcf         PCLATH,4
            bcf         PCLATH,3                    ;Selecciona la página 0
            goto        Inicio                      ;Salto a la dirección de INICIO del programa
            END
```

```
; Subrutina "LCD_MensajeMovimiento" ----------------------------------------------------
;Visualiza un mensaje de mayor longitud que los 16 caracteres que pueden
;representarse en una línea, por tanto se desplaza
; a través de la pantalla.  En el mensaje debe dejarse 16 espacios en blanco, al
;principio y al final para conseguir que el desplazamiento del mensaje sea lo
;más legible posible.
```



"Mensajes" es la etiqueta que señala el principio de todos los mensajes.

"LCD_CursorPosicion", es un registro que contabiliza la posición donde se sitúa el cursor dentro de la pantalla LCD.

"LCD_ApuntaCaracter", es un registro que indica la posición relativa del carácter que se va a visualizar respecto de la etiqueta "Mensajes".

Los mensajes no pueden superar la posición 0xFF de la memoria de programa.

```
; Subrutina "LCD_MensajeMovimiento" ---------------------------------------------------
;               CBLOCK
                LCD_CursorPosicion                      ; Contabiliza la posición del cursor dentro de la
                ENDC                                    ; pantalla LCD
LCD_MensajeMovimiento
                movwf   LCD_ApuntaCaracter      ; Posición del primer carácter del mensaje.
                movlw   Mensajes                ; Halla la posición relativa del primer carácter
                subwf   LCD_ApuntaCaracter,F    ; del mensaje respecto de la etiqueta "Mensajes".
                decf    LCD_ApuntaCaracter,F    ; Compensa la posición que ocupa "addwf PCL,F".
LCD_PrimeraPosicion
                clrf    LCD_CursorPosicion      ; El cursor en la posición 0 de la línea.
                call    LCD_Borra               ; Se sitúa en la primera posición de la línea 1 y
LCD_VisualizaCaracter                           ; borra la pantalla.
                movlw   LCD_CaracteresPorLinea  ; ¿Ha llegado a final de línea?
                subwf   LCD_CursorPosicion,W
                btfss   STATUS,Z
                goto    LCD_NoEsFinalLinea
LCD_EsFinalLinea
                call    Retardo_200ms           ; Lo mantiene visualizado durante este tiempo.
                call    Retardo_200ms
                movlw   LCD_CaracteresPorLinea-1; Apunta a la posición del segundo carácter visualizado
                subwf   LCD_ApuntaCaracter,F    ; en pantalla, que será el primero en la siguiente
                goto    LCD_PrimeraPosicion      ; visualización de línea, para producir el efecto
LCD_NoEsFinalLinea                              ; de desplazamiento hacia la izquierda.
                movf    LCD_ApuntaCaracter,W
                call    Mensajes                ; Obtiene el ASCII del carácter apuntado.
                movwf   LCD_ValorCaracter       ; Guarda el valor de carácter.
                movf    LCD_ValorCaracter,F     ; Lo único que hace es posicionar flag Z. En caso
                btfsc   STATUS,Z                ; que sea "0x00", que es código indicador final
                goto    LCD_FinMovimiento       ; de mensaje, sale fuera.
LCD_NoUltimoCaracter2
                call    LCD_Caracter            ; Visualiza el carácter ASCII leído.
                incf    LCD_CursorPosicion,F    ; Contabiliza el incremento de posición del  cursor en la pantalla.
                incf    LCD_ApuntaCaracter,F    ; Apunta a la siguiente posición por visualizar.
                goto    LCD_VisualizaCaracter   ; Vuelve a visualizar el siguiente carácter
LCD_FinMovimiento                               ; de la línea.
                return                          ; Vuelve al programa principal.
```

```
;********************************** Mensaje_07.asm **********************************
; El módulo LCD visualiza un mensaje largo (más de 16 caracteres) que se desplaza a lo largo
; de la pantalla. Se utiliza la subrutina LCD_MensajeMovimiento de la librería LCD_MENS.INC.
            LIST        P=16F876
            INCLUDE  <P16F876.INC>
            CBLOCK  0x20
            ENDC
; ZONA DE CÓDIGOS ***************************************************************
            ORG         0x05
Inicio      bsf         STATUS,RP0
            movlw       b'00000110'
            movwf       ADCON1
            bcf         STATUS,RP0
            call        LCD_Inicializa                  ; Prepara la pantalla.
Principal   movlw       Mensaje0                        ; Apunta al mensaje.
            call        LCD_MensajeMovimiento
            goto        Principal                       ; Repite la visualización.
; "Mensajes" --------------------------------------------------------------------
Mensajes    addwf       PCL,F
Mensaje0                                                ; Posición inicial del mensaje.
            DT "                "                       ; Espacios en blanco al principio para mejor
            DT "Estudia un Ciclo Formativo "            ; visualización.
            DT "de ELECTRONICA."
            DT "                ", 0x0                  ; Espacios en blanco al final.
            INCLUDE  <LCD_MENS.INC>                     ; Subrutina LCD_MensajeMovimiento.
            INCLUDE  <LCD_4BIT.INC>                     ; Subrutinas de control del LCD.
            INCLUDE  <RETARDOS.INC>                     ; Subrutinas de retardos.
            ORG         0x1F00
            bcf         PCLATH,4
            bcf         PCLATH,3  ;Selecciona la página 0
            goto        Inicio
            END                                         ; Fin del programa.
```