ANEXO 3: PROGRAMACIÓN DEL PIC16F737 EN MPLAB

En este anexo vamos a exponer el programa completo mediante el cual se han realizado todas las aplicaciones explicadas en la memoria por el PIC, y a las cuales se hacen referencia a lo largo de la memoria. Repasaremos también brevemente cuál es el funcionamiento de cada una.

```
        list            p=16f737            ; list directive to define processor
        #include        <p16f737.inc>       ; processor specific variable definitions


grupo1 udata   0x20

LCDSETCGRAM             equ 40h         ; Direcciones de memoria del display donde
se almacenan los nuevos caracteres
LCDSETCGRAM1            equ 48h
LCDSETCGRAM2            equ 50h
LCDSETCGRAM3            equ 58h
LCDSETCGRAM4            equ 60h
LCDSETCGRAM5            equ 68h
LCDSETCGRAM6            equ 70h
LCDSETCGRAM7            equ 78h
LCDCLEAR                equ 01h
LCDCONFIG               equ 0Eh
LCDWRITE                equ 38h
LCDSETDDRAM1            equ 80h
LCDSETDDRAM2            equ 0c0h
LCDMOVECURSOR           equ 14h
LCDDISPLAY              equ 08h
Contador                res 1           ; Variables auxiliares
Contador2               res 1
DelayL                  res 1           ; Variables para simular esperas
DelayH                  res 1
LCDENTRYMODE            equ 07h
DisData                 res 1
Central                 res 1
Delantero_derecho       res 1           ; Variables que almacenan el estado del
volumen de los canales
Delantero_izquierdo     res 1
Trasero_derecho         res 1
Trasero_izquierdo       res 1
```

```
Barras                  res 1       ; Variable que almacena el número de
barras a representar por pantalla
posicion                res 1
auxiliar                res 1
auxiliar2               res 1

        org     00000h
                goto    start

        org     00030h              ; Beginning of program EPROM
```

El comienzo del programa es la etiqueta START, a partir de la cual comienzan a configurarse todos los registros del PIC. Estas primeras líneas ya las hemos comentado sobradamente en la memoria, con lo que no nos detendremos más.

START

```
        BCF         STATUS, RP0
        BCF         STATUS, RP1
        CLRF        PORTB           ; Initialize PORTB by clearing output data
latches
        CLRF        PORTC           ; Initialize PORTC by clearing output data
latches
        CLRF        PORTA           ; Initialize PORTA by clearing output data
latches
        BSF         STATUS, RP0     ; Select Bank 1
        CLRF        TRISB           ; Configure all pins as digital outputs
        MOVLW       B'00000011'
        movwf       TRISC
        movlw       B'11111111'
        movwf       TRISA
        movlw       0fh             ; = all digital
        movwf       ADCON1
        bsf         SSPSTAT,6   ; Transmit occurs on transition from active to Idle
clock state
        BCF         STATUS, RP0     ; Select Bank 0
        clrf        SSPCON
        movlw       B'00100001'
        movwf       SSPCON          ; Enables serial port and configures SCK,
SDO, SDI and SS as serial port pins
        call        Clock
        movlw       38h             ; 2-line mode, 5x8 bits
        call        COMMAND
```

```
call    Clock
movlw   38h
call    COMMAND
call    Clock
movlw   38h
call    COMMAND
call    Clock
movlw   0ch             ; display on, cursor off, blink off
call    COMMAND
call    Clock
movlw   01h             ; display clear
call    COMMAND
call    Clock
call    Clock
call    Clock
call    Clock
call    Clock
movlw   06h             ; increment mode, entire shift off
call    COMMAND
call    DISPLAY
MOVLW   80h
call    COMMAND
MOVLW   0fh             ; default value set in the middle
MOVWF   Central
MOVLW   0fh             ; default value set in the middle
MOVWF   Delantero_derecho
MOVLW    0fh            ; default value set in the middle
MOVWF   Delantero_izquierdo
MOVLW    0fh            ; default value set in the middle
MOVWF   Trasero_derecho
MOVLW   0fh             ; default value set in the middle
MOVWF   Trasero_izquierdo
```

La rutina principal comienza a partir de la etiqueta LOOP, y es un bucle infinito en el que se va comprobando una vez detrás de otra si se ha producido alguna pulsación de algún botón. Si esto es así, actualiza las diferentes variables según el botón pulsado, y se transmiten los nuevos datos a los potenciómetros digitales. Al principio de cada iteración, se llama a la función REFRESH, que es la que se encarga de representar los diferentes datos por pantalla.

```
LOOP    call    REFRESH
        btfss   PORTA,1         ; Has UP been pressed?      (Normally low,
goes high when pressed.)
```

```
                goto    DOWN
WAIT1           btfsc   PORTA,1
                goto    WAIT1
                incf    Central
                incf    Delantero_derecho
                incf    Delantero_izquierdo
                incf    Trasero_derecho
                incf    Trasero_izquierdo
                call    TRANSMISION
DOWN            btfss   PORTA,0         ; Has DOWN been pressed? (Normally low,
goes high when pressed.)
                goto    REAR
WAIT2           btfsc   PORTA,0
                goto    WAIT2
                decf    Central
                decf    Delantero_derecho
                decf    Delantero_izquierdo
                decf    Trasero_derecho
                decf    Trasero_izquierdo
                call    TRANSMISION
REAR            btfss   PORTA,2         ; Has REAR been pressed? (Normally low,
goes high when pressed.)
                goto    FRONT
WAIT3           btfsc   PORTA,2
                goto    WAIT3
                incf    Trasero_derecho
                incf    Trasero_izquierdo
                decf    Central
                decf    Delantero_derecho
                decf    Delantero_izquierdo
                call    TRANSMISION
FRONT           btfss   PORTA,3         ; Has FRONT been pressed? (Normally
low, goes high when pressed.)
                goto    LEFT
WAIT4           btfsc   PORTA,3
                goto    WAIT4
                incf    Central
                incf    Delantero_derecho
                incf    Delantero_izquierdo
                decf    Trasero_derecho
                decf    Trasero_izquierdo
                call    TRANSMISION
LEFT            btfss   PORTA,4         ; Has LEFT been pressed? (Normally low,
goes high when pressed.)
                goto    RIGHT
```

```
WAIT5   btfsc   PORTA,4
        goto    WAIT5
        incf    Delantero_izquierdo
        incf    Trasero_izquierdo
        decf    Delantero_derecho
        decf    Trasero_derecho
        call    TRANSMISION
RIGHT   btfss   PORTA,5          ; Has RIGHT been pressed? (Normally low,
goes high when pressed.)
        goto    LOOP
WAIT6   btfsc   PORTA,5
        goto    WAIT6
        incf    Delantero_derecho
        incf    Trasero_derecho
        decf    Trasero_izquierdo
        decf    Delantero_izquierdo
        call    TRANSMISION
        goto    LOOP
```

La función TRANSMISION, como su propio nombre indica, se encarga de transmitirlos diferentes datos almacenados en las variables de volumen de los diferentes canales. Para ello hace uso del registro SSPBUF, que es el que el fabricante del PIC nos proporciona para realizar la comunicación por el puerto serie. Se añaden bucles de espera para evitar que se escriba sobre SSPBUF antes de que éste haya mandado todos los bits, y se llama a la función CAMBIO, que es la que se encarga de transformar los datos para que los entienda el DS1802.

TRANSMISION

```
        bsf     PORTC,2
        MOVLW   B'11101111'      ; data transmision
        MOVWF   SSPBUF
        movlw   8
        movwf   auxiliar
DATO1   decfsz  auxiliar,1       ; Check collision
        goto    DATO1
        MOVLW   3fh
        MOVWF   auxiliar
        MOVF    Central,W
        call    CAMBIO           ; data transform for the DS1802
        MOVWF   SSPBUF           ; data transmision
        movlw   8
```

```
        movwf   auxiliar
DATO2   decfsz  auxiliar,1          ; Check collision
        goto    DATO2
        MOVLW   3fh
        MOVWF   auxiliar
        MOVF    Trasero_derecho,W
        call    CAMBIO
        MOVWF   SSPBUF
        movlw   8
        movwf   auxiliar
DATO3   decfsz  auxiliar,1
        goto    DATO3
        MOVLW   3fh
        MOVWF   auxiliar
        MOVF    Trasero_izquierdo,W
        call    CAMBIO
        MOVWF   SSPBUF
        movlw   8
        movwf   auxiliar
DATO4   decfsz  auxiliar,1
        goto    DATO4
        MOVLW   3fh
        MOVWF   auxiliar
        MOVF    Delantero_derecho,W
        call    CAMBIO
        MOVWF   SSPBUF
        movlw   8
        movwf   auxiliar
DATO5   decfsz  auxiliar,1
        goto    DATO5
        MOVLW   3fh
        MOVWF   auxiliar
        MOVF    Delantero_izquierdo,W
        call    CAMBIO
        MOVWF   SSPBUF
        movlw   32
        movwf   auxiliar
DATO6   decfsz  auxiliar,1
        goto    DATO6
        bcf     PORTC,2

        RETURN
```

La función COMMAND es la usada para enviar cualquier tipo de comando al display. Se le pasa por el registro W el comando a ejecutar, y ésta se encarga de enviarlo, activar el ENABLE, y esperar un tiempo para asegurarse que el display ejecuta la orden requerida, y no se le manda otra. Las esperas están calculadas con holgura, y aunque son elevadas, no afectan al rendimiento de la aplicación.

```
COMMAND                 ; Function to transmit differents commands to the LCD

                movwf   PORTB
                bsf     PORTC,7
                nop
                bcf     PORTC,7
                movlw   16h
                movwf   DelayL
kdloop2         decfsz   DelayL
                goto    kdloop2

                RETURN
```

La función CLOCK se ha realizado para meter un bucle de espera largo en las instrucciones iniciales que se le envían al LCD, ya que éstas lo requieren así.

```
CLOCK           movlw    8fh            ; Function to wait
                movwf    DelayL
                movwf    Contador
kdloop          nop
                decfsz   DelayL
                goto     kdloop
                movwf    DelayL
                decfsz   Contador
                goto     kdloop

                RETURN
```

La función ESCRIBE se utiliza para escribir diferentes caracteres en la pantalla del LCD, para ello activa tanto el ENABLE como el WRITE del LCD, a la vez que manda por el puerto B el dato en cuestión. Este dato le tiene que ser proporcionado a la

función a través del registro W. También se le añadió un bucle de espera por precaución a que no se pisaran unos datos con otros.

```
ESCRIBE                 ; Function to write characters in the display

                bsf     PORTC,6
                movwf   PORTB
                bsf     PORTC,7
                nop
                bcf     PORTC,7
                bcf     PORTC,6
                movlw   40h
                movwf   DelayL
kdloop5         decfsz  DelayL
                goto    kdloop5

                RETURN
```

La función DISPLAY es la que se encarga de escribir en la memoria del LCD los datos nuevos que creamos para esta aplicación. Para ello realiza las siguientes tareas, primero sitúa el puntero del LCD en la dirección de memoria de la CGRAM adecuada, después va escribiendo los diferentes datos en ella por filas, de tal forma que si queremos una fila rellena llama a la función FILA1, y si la queremos vacía llama a la función FILA1

```
DISPLAY                 ; To store new characters into the memory of the LCD

                movlw   LCDSETCGRAM
                call    COMMAND
                call    FILA0
                call    FILA0
                call    FILA0
                call    FILA0
                call    FILA0
                call    FILA0
                call    FILA0
                call    FILA0
                call    FILA1
                movlw   LCDSETCGRAM1
                call    COMMAND
                call    FILA0
                call    FILA0
                call    FILA0
                call    FILA0
```

```
call    FILA0
call    FILA0
call    FILA1
call    FILA1
movlw   LCDSETCGRAM2
call    COMMAND
call    FILA0
call    FILA0
call    FILA0
call    FILA0
call    FILA0
call    FILA1
call    FILA1
call    FILA1
movlw   LCDSETCGRAM3
call    COMMAND
call    FILA0
call    FILA0
call    FILA0
call    FILA0
call    FILA1
call    FILA1
call    FILA1
call    FILA1
movlw   LCDSETCGRAM4
call    COMMAND
call    FILA0
call    FILA0
call    FILA0
call    FILA1
call    FILA1
call    FILA1
call    FILA1
call    FILA1
movlw   LCDSETCGRAM5
call    COMMAND
call    FILA0
call    FILA0
call    FILA1
call    FILA1
call    FILA1
call    FILA1
call    FILA1
call    FILA1
movlw   LCDSETCGRAM6
```

```
        call    COMMAND
        call    FILA0
        call    FILA1
        call    FILA1
        call    FILA1
        call    FILA1
        call    FILA1
        call    FILA1
        call    FILA1
        movlw   LCDSETCGRAM7
        call    COMMAND
        call    FILA1
        call    FILA1
        call    FILA1
        call    FILA1
        call    FILA1
        call    FILA1
        call    FILA1
        call    FILA1

        RETURN
```

La función FILA0 es llamada por la función DISPLAY, y como ya hemos dicho antes, rellena una fila de memoria de CGRAM con espacios en blanco, para ello escribe el carácter 00h en el puerto B, y activa tanto la señal ENABLE como la señal WRITE

```
FILA0                   ; Writes a line full of 0 in the LCD

        MOVLW   00h
        MOVWF   PORTB
        bsf     PORTC,6
        bsf     PORTC,7
        nop
        nop
        bcf     PORTC,7
        bcf     PORTC,6
        movlw   20h
        movwf   DelayL
kdloop3 decfsz  DelayL
        goto    kdloop3

        RETURN
```

La función FILA1 es similar a la anterior, su única diferencia es que en vez de escribir el carácter 00h escribe el 1fh, con lo que se rellena la fila entera con una barra.

```
FILA1                   ; Writes a line full of 1 in the LCD

        MOVLW   1fh
        MOVWF   PORTB
        bsf     PORTC,6
        bsf     PORTC,7
        nop
        nop
        bcf     PORTC,7
        bcf     PORTC,6
        movlw   20h
        movwf   DelayL
kdloop4 decfsz  DelayL
        goto    kdloop4

        RETURN
```

La función REFRESH se encarga de sacar por la pantalla del LCD los datos almacenados en el PIC. Se le llama cada vez que se produce una iteración en el bucle principal. Su funcionamiento es el siguiente: primero calcula el número de barras que tiene que representar, y en función de este se comporta de dos maneras diferentes. Si el número de barras es mayor que 8, necesitamos representar caracteres en las dos filas del display; lo que hace la función es representar el la línea de abajo todas las barras llenas, restarle 8 al número de barras a representar, y representar el resto arriba. Si el número de barras es menor que 8 (o lo que es lo mismo, que el número a representar sea menor que 32), entonces se representan las barras que sean abajo. Además, se representa el carácter en blanco arriba, puesto que si no, puede ser que se guarde algún carácter antes usado, y no se represente de forma correcta. Este proceso se repite con el resto de caracteres.

```
REFRESH                 ; Refresh the display

        MOVFW   Central
        call    CONT_BARRAS
        MOVLW   0c3h
        BTFSS   Central,5       ; is Central < 32 ?
        goto    LA1
        MOVLW   83h
        call    LINEA_ARRIBA
```

```
            MOVLW   0c3h
            call    LINEA_LLENA
            goto    FINLINEA1
LA1         call    LINEA_ABAJO
            MOVLW   83h
            call    LINEA_VACIA
FINLINEA1   MOVFW   Delantero_izquierdo
            call    CONT_BARRAS
            MOVLW   0c6h
            BTFSS   Delantero_izquierdo,5
            goto     LA2
            MOVLW   86h
            call    LINEA_ARRIBA
            MOVLW   0c6h
            call    LINEA_LLENA
            goto    FINLINEA2
LA2         call    LINEA_ABAJO
            MOVLW   86h
            call    LINEA_VACIA
FINLINEA2   MOVFW   Delantero_derecho
            call    CONT_BARRAS
            MOVLW   0c9h
            BTFSS   Delantero_derecho,5
            goto    LA3
            MOVLW   89h
            call    LINEA_ARRIBA
            MOVLW   0c9h
            call    LINEA_LLENA
            goto    FINLINEA3
LA3         call    LINEA_ABAJO
            MOVLW   89h
            call    LINEA_VACIA
FINLINEA3   MOVFW   Trasero_izquierdo
            call    CONT_BARRAS
            MOVLW   0cch
            BTFSS   Trasero_izquierdo,5
            goto    LA4
            MOVLW   8ch
            call    LINEA_ARRIBA
            MOVLW   0cch
            call    LINEA_LLENA
            goto    FINLINEA4
LA4         call    LINEA_ABAJO
            MOVLW   8ch
            call    LINEA_VACIA
```

```
FINLINEA4   MOVFW     Trasero_derecho
            call      CONT_BARRAS
            MOVLW     0cfh
            BTFSS     Trasero_derecho,5
            goto      LA5
            MOVLW     8fh
            call      LINEA_ARRIBA
            MOVLW     0cfh
            call      LINEA_LLENA
            goto      FINLINEA5
LA5         call      LINEA_ABAJO
            MOVLW     8fh
            call      LINEA_VACIA

FINLINEA5   RETURN
```

La función CONT_BARRAS recibe por el registro W el dato correspondiente, y calcula el número de barras a representar dividiendo éste por 4, devolviendo el resultado en la variable Barras.

```
CONT_BARRAS                 ; Calculates the charater to write on the display

            clrf      Barras
            incf      Barras
            MOVWF     Contador
CALC        BTFSC     Contador,7
            goto      CALCULAR
            goto      COMPR
COMPR       BTFSC     Contador,6
            goto      CALCULAR
            goto      COMPR2
COMPR2      BTFSC     Contador,5
            goto      CALCULAR
            goto      COMPR3
COMPR3      BTFSC     Contador,4
            goto      CALCULAR
            goto      COMPR4
COMPR4      BTFSC     Contador,3
            goto      CALCULAR
            goto      COMPR5
COMPR5      BTFSS     Contador,2
            goto      FINCALC
CALCULAR    incf      Barras
```

```
            MOVLW   04h
            MOVWF   Contador2
SUB         decf    Contador
            decfsz  Contador2
            goto    SUB
            goto    CALC

FINCALC     RETURN
```

Tal y como hemos comentado antes, la función LINEA_ABAJO representa en la línea de abajo del display el número de barras almacenado en la variable Barras. Anteriormente se ha tenido que colocar el puntero en el lugar adecuado.

```
LINEA_ABAJO         ; Writes the character set on the W registrer in the second
line of the display

            Call    COMMAND
            decf    Barras
            MOVFW   Barras
            call    ESCRIBE

            RETURN
```

Esta función, denominada LINEA_ARRIBA, realiza lo mismo que la anterior, pero con la salvedad de que tiene que restarle 8 a las barras a representar, puesto que éstas son las que se han representado con anterioridad en la 2ª línea del display.

```
LINEA_ARRIBA        ; Writes the character set on the W registrer in the first line
of the display

            Call    COMMAND
            MOVLW   8
            SUBWF   Barras
            MOVFW   Barras
            call    ESCRIBE

            RETURN
```

La función LINEA_LLENA escribe el carácter que representa a todas la barras de una posición, en la dirección de la pantalla donde esté apuntando el puntero en ese momento

```
LINEA_LLENA     ; Writes the character "full lines" in the second line of the
display

                call    COMMAND
                MOVLW   07h                     ; Todas las barras
                call    ESCRIBE

                RETURN

LINEA_VACIA     ; Writes the character zero in the second line of the display

                call    COMMAND
                MOVLW   20h
                call    ESCRIBE

                RETURN
```

Para concluir, la función CAMBIO, que es la que se encarga de adaptar los datos a enviar al DS1802. Para ello, primero se le resta a 64 el valor del dato, puesto que el potenciómetro entiende los números de forma diferente a como los tratamos posteriormente. Por otro lado, hay que darle la vuelta a todos los bits, de otra manera, el PIC y el potenciómetro no se entenderían. Mediante una resta y varios saltos condicionantes se realizan estas acciones.

```
CAMBIO          ; Changes the character set in the W registrer, for being
understand by the DS1802

                SUBWF   auxiliar
SHIFT           btfsc   auxiliar,7
                goto    shift7
                bcf     auxiliar2,0
                goto    bit6
shift7          bsf     auxiliar2,0
bit6            btfsc   auxiliar,6
                goto    shift6
                bcf     auxiliar2,1
```

```
            goto    bit5
shift6      bsf     auxiliar2,1
bit5        btfsc   auxiliar,5
            goto    shift5
            bcf     auxiliar2,2
            goto    bit4
shift5      bsf     auxiliar2,2
bit4        btfsc   auxiliar,4
            goto    shift4
            bcf     auxiliar2,3
            goto    bit3
shift4      bsf     auxiliar2,3
bit3        btfsc   auxiliar,3
            goto    shift3
            bcf     auxiliar2,4
            goto    bit2
shift3      bsf     auxiliar2,4
bit2        btfsc   auxiliar,2
            goto    shift2
            bcf     auxiliar2,5
            goto    bit1
shift2      bsf     auxiliar2,5
bit1        btfsc   auxiliar,1
            goto    shift1
            bcf     auxiliar2,6
            goto    bit0
shift1      bsf     auxiliar2,6
bit0        btfsc   auxiliar,0
            goto    shift0
            bcf     auxiliar2,7
            goto    FIN
shift0      bsf     auxiliar2,7
FIN         MOVF    auxiliar2,W

            RETURN


        END
```