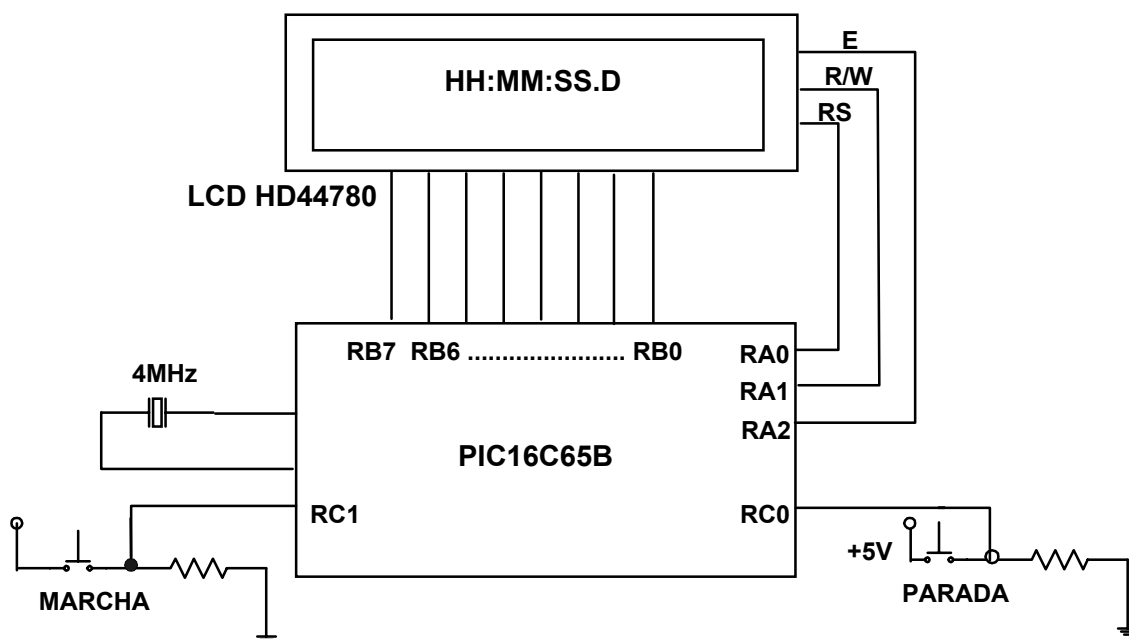


Práctica 8 - Microcontroladores PIC: Interface con driver de LCD

Subprogramas para interface a 8 bits y 4 bits con LCD

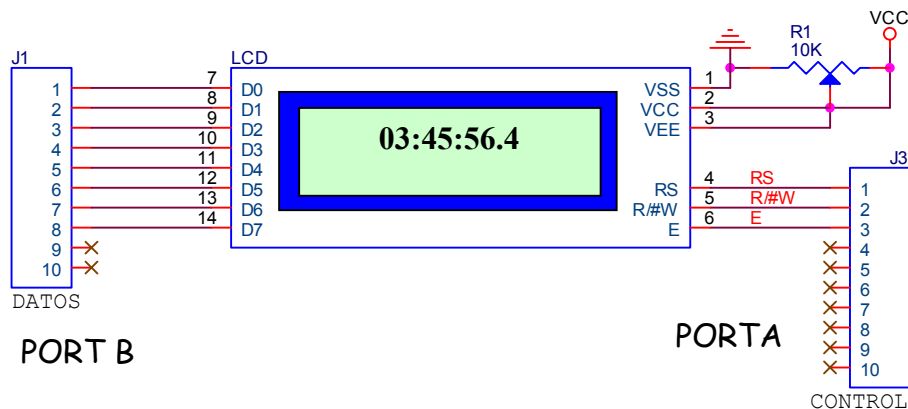
Se trata de realizar el diseño del programa encargado de gestionar **una pantalla de cristal líquido matricial de puntos con driver HD44780 o compatible** para implementar un cronómetro digital que dispone de pulsadores de **MARCHA (RC1)** y **PARADA (RC0)**. El crono tiene capacidad de representar **Horas:Minutos:Segundos.Décimas de Segundo** en pantalla y hasta completar 24 horas



El diseño se basa en el empleo de un microcontrolador PIC16C65B con un oscilador de 4MHz

Aunque las 8 líneas de datos del LCD están conectadas a las 8 líneas de datos del PORTB, resulta posible utilizar únicamente las 4 más altas si así se define el interface.

Esquema de la placa de circuito impreso con el LCD:



Planteamiento:

Se utiliza el TMR1 para realizar una temporización de 0,1 s que generará una interrupción, en esa interrupción se modifican las posiciones que almacenan Décimas, Segundos, Minutos y Horas y se envían al LCD como caracteres a representar. Al LCD sólo se envían los caracteres cuando se necesita cambiarlos y se envían todos, también podría ser posible ubicar puntero de direcciones de la DDRAM del LCD y enviar sólo los que cambian.

El programa principal es un bucle donde se exploran permanentemente el estado de los pulsadores de MARCHA y de PARADA, si alguno de ellos está activado, se para el TMR1 ó se pone en marcha

El TMR1 debe precargarse con un valor tal que:

$$0,1s = 4/4MHz * 8 * (65536 - \text{Precarga})$$

de donde Precarga=53036 (0xCF2C)

Para el envío de caracteres y comandos al LCD se utilizan los subprogramas presentes en el fichero S_LCD8B.ASM si el interface se realiza con 8 bits o bien S_LCD4B.ASM si es de 4 bits. Estos ficheros se deben incluir en el fichero del código fuente.

Algoritmo utilizado:

INICIALIZACIÓN

- Puertos: PORTB inicialmente de salida para enviar datos y comandos al LCD, pero serán de entrada en algunos intervalos para leer estado del LCD
- PORTA será de salida en sus 3 bits más bajos para líneas de control del LCD (RS,R/W y E) y son salidas permanentemente
- PORTC de entrada en sus 2 líneas más bajas, el resto no se usa
- Inicializamos el LCD: tras esperar el arranque (15ms de temporización software) definimos interface de 8 bits (ó 4) y 2 líneas de pantalla, desactivamos pantalla
- TMR1: modo temporizador, prescaler de 8 y parado
- Precargar TMR1H y TMR1L para que, tras ponerlo en marcha, desborde al cabo de 0,1s
- INTERRUPCIONES: habilitar las de TMR1
- Variables del algoritmo a cero: DECIMAS, SEGUNDOS, MINUTOS Y HORAS
- Activamos interrupciones globales y de periféricos (GIE y PEIE)
- Llamamos al subprograma encargado de enviar los caracteres ASCII de lo que queremos sacar en el LCD (SACATIEMPO) al principio

BUCLE PRINCIPAL

- Si RCO está a 1 (pulsada) entonces paramos TMR1
- Si RC1 está a 1 (pulsada) entonces ponemos en marcha TMR1
- Volvemos al principio del bucle principal

PROGRAMA DE TRATAMIENTO DE LA INTERRUPCIÓN DE TMR1

- Se comprueba que el flag TMR1IF está a 1
- Salvaguarda del contexto (W y STATUS)
- Precarga de TMR1H y TMR1L para que siga contando
- Incremento de DECIMAS
- Si hemos llegado a 10, ponemos a 0 DECIMAS e incrementamos SEGUNDOS
si no, vamos directamente a la salida del programa de tratamiento
- Se incrementa SEGUNDOS en una unidad
- Se extraen los 4 bits más bajos (unidades)

- Si unidades=0x0A entonces sumamos 6 a SEGUNDOS (para hacer el ajuste a BCD)
si no, vamos directamente a la salida del programa de tratamiento
- Se extraen los 4 bits más altos de SEGUNDOS (decenas de segundo)
- Si hemos llegado a 6 entonces se pone a cero SEGUNDOS y se incrementan los minutos (MINUTOS)
si no, vamos directamente a la salida del programa de tratamiento
- Se extraen los 4 bits más bajos de MINUTOS (unidades de minuto)
- Si unidades=0x0A entonces se suma 6 a MINUTOS (para hacer el ajuste a BCD)
si no, vamos directamente a la salida del programa de tratamiento
- Se extraen los 4 bits más altos de MINUTOS (decenas de minuto)
- Si se ha llegado a 6 entonces ponemos a cero MINUTOS y se incrementan las horas (HORAS)
si no, vamos directamente a la salida del programa de tratamiento
- Se extraen los 4 bits más bajos de HORAS (unidades de hora)
- Si unidades=0x0A entonces se suma 6 a HORAS (para hacer el ajuste a BCD)
- Salida del programa de tratamiento:
 - Se comparan las HORAS con 24 si las hemos alcanzado, ponemos HORAS a cero
 - Llamamos al Subprograma que saca el tiempo total para actualizar el LCD
 - Se pone el flag a cero (TMR1IF=0)
 - Recuperación del contexto
 - Retorno de interrupción

SUBPROGRAMA SACATIEMPO

Se encarga de tomar los valores almacenados en DECIMAS, SEGUNDOS, MINUTOS Y HORAS y enviarlos al LCD en formato ASCII, se intercalan también caracteres de separación ':' (dos puntos) entre horas y minutos y entre minutos y horas y '.' (punto) entre segundos y décimas

- Se coloca el cursor en "casa" (esquina superior izquierda)
- Enviamos 3 espacios en blanco para "centrar" en pantalla
- Extraemos decenas de horas, le añadimos el ASCII del cero y lo enviamos al LCD con el subprograma LCDPUTCHAR
- Lo mismo con las unidades de horas

- Enviamos el ASCII de ':'
- Envío del ASCII de las decenas de minutos
- Envío del ASCII de las unidades de minutos
- Enviamos el ASCII de ':'
- Envío del ASCII de las decenas de segundos
- Envío del ASCII de las unidades de segundos
- Enviamos el ASCII de '.'
- Envío del ASCII de las décimas de segundo
- Retorno de subprograma

Código Fuente de la Aplicación para interface de 8 bits:

```
*****  
;  
;  
; FICHERO REL_LCD1.ASM (LCD a 8 bits)  
;  
; Ejemplo de manejo de un LCD con driver HD44780 para realizar  
; un cronómetro digital con salida: HH:MM:SS.D  
;  
; Se emplea un oscilador de frecuencia 4MHz  
;  
; Para contabilizar DECIMAS usaremos TMR1 para que genere una  
; interrupción cada décima de segundo, en el programa de tratamiento  
; de la interrupción actualizamos variables y las enviamos al display  
;  
; Realizado por Fernando Nuño García  
;  
=====  
; Se utiliza un interface de 8 bits y los subprogramas de manejo del LCD  
; correspondientes que están en el fichero S_LCD8B.ASM y que se incluirá  
; a su vez en el presente fichero  
;  
; En este programa principal se definen los puertos de datos y de control  
; y se les asocian las etiquetas que se utilizan en el fichero de subprogramas:  
  
; El puerto de datos es el PORTB y el de control las 3 líneas más bajas de PORTA  
;  
; Se emplean además dos pulsadores conectados al PORTC:  
;  
; Para puesta en marcha del cronómetro en RC0 (si está pulsado a 1)  
; Para la parada del cronómetro en RC1 (si pulsado a 1)  
;  
  
list p=PIC16C65 ;Definimos MCU a utilizar  
include "p16c65b.inc" ;Incluimos fichero de etiquetas  
  
;Definición de líneas a utilizar  
  
LCD_DATA EQU PORTB ;DATOS se conecta a PORTB  
LCD_DATA_TRIS EQU TRISB ;  
LCD_CTRL EQU PORTA ;CONTROL se conecta a PORTA  
LCD_CTRL_TRIS EQU TRISA ;  
  
LCD_LINE0 EQU 0x000 ;dirección comienzo línea superior DDRAM  
LCD_LINE1 EQU 0x040 ;dirección comienzo línea inferior DDRAM  
  
; También se deben definir los Bits de control:  
  
LCD_E EQU 2 ; Enable será el bit 2 del PORTA  
LCD_RW EQU 1 ; Read/Write bit 1 del PORTA  
LCD_RS EQU 0 ; RS será el bit 0 del PORTA  
  
; Los bits de datos por orden:  
  
DB7 EQU 7 ; bit 7 (MSB)  
DB6 EQU 6 ; bit 6  
DB5 EQU 5 ; bit 5  
DB4 EQU 4 ; bit 4  
DB3 EQU 3 ; bit 3  
DB2 EQU 2 ; bit 2  
DB1 EQU 1 ; bit 1  
DB0 EQU 0 ; bit 0 (LSB)  
  
Banco_0_RAM EQU 0x20 ; comienzo de la RAM de propósito general  
; del Banco0  
  
; Bloque de variables en RAM  
  
cblock Banco_0_RAM  
W_TEMP ;almacén temporal para el registro W  
STATUS_TEMP ;almacén temporal para el registro STATUS  
SEGUNDOS ;segundos (DECENAS:UNIDADES)  
MINUTOS ;minutos (DECENAS:UNIDADES)  
HORAS ;horas (DECENAS:UNIDADES)  
DECIMAS ;contador de DECIMAS de segundo  
LCD_TEMP ;variable temporal de uso interno en las subrutinas  
;del LCD
```

```

    AUX            ;variable auxiliar

    DELAY          ;Para temporización software
    X_DELAY        ;para temporización software

    endc

    org    0
    goto   INICIO      ;Salto a posición de inicio
    org    0x04
    goto   PTI         ;Salto a Programa de Tratamiento de Interrupción

    include "S_LCD8B.ASM" ;Incluimos fichero de subprogramas de manejo
                        ;del LCD

INICIO bsf     STATUS,RP0      ;Paso al banco 1
      clrf    LCD_DATA_TRIS   ;Defino PORTB de salidas
      movlw   b'11111000'     ;En el PORTA las 3 menos significativas
      movwf   LCD_CTRL_TRIS   ;como salidas para E, RS y R/W

      bsf     PIE1,TMR1IE     ;Activamos máscara interrupciones TMR1 en PIE1

      bcf     STATUS,RP0      ;Volvemos al banco 0

      call    LCDINIT         ;Inicializamos el LCD para definir funcionamiento
                        ;interface de 8 bits y 2 líneas visibles

      movlw   b'00110000'     ;Configuramos TMR1 como temporizador
      movwf   T1CON           ;prescaler a 8 y parado inicialmente

; Realizaremos la precarga de TMR1 para que desborde cada décima de segundo
; (4/4MHz)*8*(65536-53036)=0,1s, luego precarga 53036=0xCF2C

      movlw   0xCF            ;Precargamos TMR1H
      movwf   TMR1H           ;con 0xCF
      movlw   0x2C            ;y TMR1L
      movwf   TMR1L           ;con 0x2C

      clrf    DECIMAS         ;Inicializamos DECIMAS
      clrf    SEGUNDOS        ; SEGUNDOS
      clrf    MINUTOS         ; MINUTOS
      clrf    HORAS           ; HORAS

      clrf    PIR1            ;Ponemos a 0 el flag de TMR1

      movlw   b'11000000'     ;Activamos interrupciones con
      movwf   INTCON           ;las máscaras GIE y PEIE

      call    SACATIEMPO      ;Sacamos el tiempo al LCD al principio

BUCLE  btfsc   PORTC,0         ;Exploramos RC0
      bcf     T1CON,TMR1ON     ;Si está a 1 paramos TMR1
      btfsc   PORTC,1         ;Exploramos RC1
      bsf     T1CON,TMR1ON     ;Si está a 1 activamos TMR1
      goto   BUCLE            ;Volvemos al bucle principal

; Subprograma que se encarga de sacar en el LCD el tiempo transcurrido

SACATIEMPO
    call    LCDHOME           ;Cursor a esquina superior izquierda

    movlw   0x03              ;Mandamos 3 espacios en blanco
    movwf   AUX               ;para "centrar" la hora
BLANCOS movlw   0x20           ;El espacio en blanco es 0x20 en ASCII
                        ;se podía haber puesto también: A' '
    call    LCDPUTCHAR        ;se carga en W y se llama al subprograma
    decfsz  AUX,F             ;si no hemos enviado 3 blancos repetimos
    goto   BLANCOS

                        ;Empezamos por la izquierda del LCD
    movf    HORAS,W           ;Tomo las horas
    andlw   0xF0              ;y extraigo las decenas
    movwf   AUX               ;se las paso a la variable AUX
    swapf   AUX,W             ;y las coloco en la parte baja de W
    addlw   A'0'              ;le sumo el ASCII del 0
    call    LCDPUTCHAR        ;y llamo a enviar caracter

    movf    HORAS,W           ;Tomo las horas

```

```

andlw 0x0F          ;y extraigo las unidades
addlw A'0'         ;le sumo el ASCII del 0
call LCDPUTCHAR   ;y lo envío al LCD

movlw ':'          ;cargo los 2 puntos
call LCDPUTCHAR   ;y lo envío al LCD

movf MINUTOS,W    ;Tomo los minutos
andlw 0xF0        ;y extraigo las decenas
movwf AUX         ;se las paso a la variable AUX
swapf AUX,W       ;y las coloco en la parte baja de W
addlw A'0'        ;le sumo el ASCII del 0
call LCDPUTCHAR   ;y llamo a enviar caracter

movf MINUTOS,W    ;Tomo los minutos
andlw 0x0F        ;y extraigo las unidades
addlw A'0'        ;le sumo el ASCII del 0
call LCDPUTCHAR   ;y lo envío al LCD

movlw ':'          ;cargo los 2 puntos
call LCDPUTCHAR   ;y lo envío al LCD

movf SEGUNDOS,W   ;Tomo los segundos
andlw 0xF0        ;y extraigo las decenas
movwf AUX         ;se las paso a la variable AUX
swapf AUX,W       ;y las coloco en la parte baja de W
addlw A'0'        ;le sumo el ASCII del 0
call LCDPUTCHAR   ;y llamo a enviar caracter

movf SEGUNDOS,W   ;Tomo los segundos
andlw 0x0F        ;y extraigo las unidades
addlw A'0'        ;le sumo el ASCII del 0
call LCDPUTCHAR   ;y lo envío al LCD

movlw '.'         ;cargo el punto
call LCDPUTCHAR   ;y lo envío al LCD

movf DECIMAS,W    ;Tomo las décimas
addlw A'0'        ;le sumo el ASCII del 0
call LCDPUTCHAR   ;y lo envío al LCD

return            ;salimos del subprograma

;Programa de Tratamiento de la Interrupción de TMR1

PTI    btfss PIR1,TMR1IF ;Comprobamos que TMR1IF=1
        retfie           ;en caso contrario retornamos

        movwf W_TEMP     ;Salvamos W
        swapf STATUS,W   ;y STATUS
        movwf STATUS_TEMP ;a la entrada de la interrupción

        movlw 0xCF       ;Precargamos TMR1H
        movwf TMR1H
        movlw 0x2C       ;y también TMR1L
        movwf TMR1L

        incf DECIMAS     ;Incrementamos DECIMAS
        movlw 0x0A       ;Cargamos W con 0x0A
        xorwf DECIMAS,W  ;Realizamos una EXOR
        btfss STATUS,Z   ;Si DECIMAS llegó 0x0A (10) cambiamos SEGUNDOS
        goto SALIR       ;si no, vamos al final

        clrf DECIMAS     ;Ponemos DECIMAS a cero
        incf SEGUNDOS    ;Incrementamos segundos

        movf SEGUNDOS,W  ;Me quedo con
        andlw 0x0F       ;las unidades
        xorlw 0x0A       ;compruebo si hemos llegado a a (10)
        btfss STATUS,Z   ;Si es así hay que cambiar decenas
        goto SALIR       ;en caso contrario salimos
        movlw 0x06       ;Para corregir sumamos 6 a las unidades
        addwf SEGUNDOS,F ;para tenerlo en BCD

        movlw 0xF0       ;Extraigo los 4 bits altos
        andwf SEGUNDOS,W ;decenas de segundos
        xorlw 0x60       ;compruebo si hemos llegado a 6

```

```
    btfss STATUS,Z      ;Si es así hay que resetear SEGUNDOS
                        ;e incrementar MINUTOS
    goto SALIR          ;En caso contrario salimos
    clrf SEGUNDOS

    incf MINUTOS        ;Incrementamos MINUTOS

    movf MINUTOS,W     ;Me quedo con
    andlw 0x0F          ;las unidades
    xorlw 0x0A         ;compruebo si hemos llegado a A (10)
    btfss STATUS,Z     ;Si es así hay que cambiar decenas
    goto SALIR         ;en caso contrario salimos
    movlw 0x06         ;Para corregir sumamos 6 a las unidades
    addwf MINUTOS,F    ;para tenerlo en BCD

    movlw 0xF0         ;Extraigo los 4 bits altos
    andwf MINUTOS,W    ;decenas de MINUTOS
    xorlw 0x60         ;compruebo si hemos llegado a 6
    btfss STATUS,Z     ;Si es así hay que resetear MINUTOS
                        ;e incrementar HORAS
    goto SALIR         ;En caso contrario salimos
    clrf MINUTOS

    incf HORAS         ;Incrementamos HORAS

    movf HORAS,W       ;Me quedo con
    andlw 0x0F          ;las unidades
    xorlw 0x0A         ;compruebo si hemos llegado a a (10)
    btfss STATUS,Z     ;Si es así hay que cambiar decenas
    goto SALIR         ;en caso contrario salimos
    movlw 0x06         ;Para corregir sumamos 6 a las unidades
    addwf HORAS,F      ;para tenerlo en BCD

                        ;Al salir compruebo que no hemos llegado a 24h
SALIR movf HORAS,W     ;Tomo las HORAS totales
      xorlw 0x24       ;compruebo si hemos llegado a 24
      btfsc STATUS,Z  ;Si es así hay que
      clrf HORAS      ;resetear las HORAS

    call SACATIEMPO    ;Actualizo el LCD con los nuevos valores

    bcf PIR1,TMR1IF    ;Pongo a 0 el flag para el siguiente desborde

    swapf STATUS_TEMP,W ;Recuperamos STATUS con swapf
    movwf STATUS
    swapf W_TEMP,F     ;Y W con doble swap
    swapf W_TEMP,W

    retfie             ;Salimos del programa de tratamiento

    END                ;Fin de Fichero
```