```
/*********************************************************************
*                                                                    *
*           Librer¡a de Funciones del Display  HD44780               *
*            Configuraci¢n de e/s de la placa de test                *
*                                                                    *
*********************************************************************/

#pragma ROM (LARGE)     //   Las llamadas a CALL y JMP se codifican como
                        //  LCALL y LJMP. Esto permite acceder a lo largo
                        //  de todo el espacio de memoria direccionable.
                        //  El c¢digo de programa se puede localizar en los
                        // 64 Kbytes de memoria de c¢digo ROM.

#pragma small   //  Todas las variables, los segmentos de datos locales y los
                // procedimientos se obligan a residir en la parte de memoria
                // interna del microcontrolador. El acceso a ,stas, es pues muy
                // eficiente, y su desventaja es que presenta un zona de memo-
                // -ria reducida.

#pragma ot(5,SIZE)      // Direccitva que obliga al compilador a realizar una
                        // optimizaci¢n del c¢digo generado. En este caso se
                        // fuerza la optimizaci¢n en su m ximo nivel (=5) y
                        // se realizar  un ,nfasis especial en la optimizaci¢n
                        // del tama¤o del c¢digo.
                        //
                        // Para realizarlo en la velocidad, se utilizar  la
                        // siguiente directiva.
                        //   #pragma ot(5,SPEED)

#pragma DEBUG    //  Esta directiva indica al compilador que incluya la infor-
                 // -maci¢n necesaria para realizar el 'debugeo' en una
                 // herramienta SW. Esta informaci¢n contiene las definiciones
                 // de las variables locales y globales y su direcci¢n, as¡
                 // como los nombres de las funciones y sus n£meros de l¡nea.

#pragma SYMBOLS  //  Mediante esta directiva se genera un listado completo de todos
                 // los s¡mbolos usados en y por el modulo del programa al ser
                 // compilado. Para cada objeto simb¢lico se indican la descrip-
                 // -ci¢n de memoria, su categor¡a, el offset y el tama¤o del
                 // objeto.

#pragma LISTINCLUDE     //  Esta directiva permite incluir el contenido de
                        // todos los ficheros incluidos (#include <fich>) en
                        // el programa.

#pragma CODE    //  Esta directiva permite incluir una lista de los mnem¢nicos
                // de ensamblador al fichero de listado (fich.lst) El c¢digo
                // ensamblador se presenta paa cada funci¢n del programa.

//#pragma SRC

#include <reg52.h>

sbit E     = P0^5  ;     // Lineas de control del display
sbit RS    = P0^7  ;     // Lineas de control del display
sbit RW    = P0^6  ;     // Lineas de control del display


#define  n_final  120   // 120

// *******************************************************************
//    En todas las funciones de esta librer¡a se a inhabilitado la entra-
//  de las interrupciones externas ya que pueden producir una programaci¢n
//  incorrecta de LCD. A tal efecto se incluye al principio de cada funci¢n
//
//   void XXXXX( void )
//
//   EA=0;        // Se deshabilitan todas las interrupciones
//   ...
//   EA=0x87;     // Se vuelven a habilitar las INT0,TIMER0 e INT1
//

void dato_display( char dato )  //   Este proc es identico a putchar y
{                               //  no devuelve nada
  int n;

  EA=1;

  P2 = dato;
  RW = 0;
  RS = 1;
  E = 1;
  E = 0;
  for( n = 0; n < n_final; n ++ );

  EA=0x87;
}


void control_display( dato )
char dato;
{
  unsigned char n,aux;

  EA=1;

  aux=P2;
  P2=dato;
  RW = 0;
  RS = 0;
  E = 1;
  E = 0;
  for( n = 0; n < n_final; n ++ );
  P2=aux;

  EA=1;
}



void init_display()
{
  int n;

  EA=1;

  E = 0;          /* Deseleccionamos el display */
  RW = 0;
  RS = 0;

  for( n = 0; n < 5000; n ++ );   //5000

  control_display( 0x3f );      // Selecci¢n del interfaz de 8 l¡neas (0x10)
                                // Visualizaci¢n en 2 l¡neas          (0x08)
                                // Fuentes de caracteres 5*10 puntos  (0x04)
                                // RESTO  (0x000---XX)
  control_display( 0x6  );  //6
  control_display( 0xe  );  //e
  control_display( 0x1  );
 // control_display( 0x2  );

  EA=1;
}

void borrar_display()
{
  int n;

  EA=1;

  E = 0;         //  Deseleccionamos el display
  RW = 0;
  RS = 0;

  for( n = 0; n < 5000; n ++ );
  control_display( 0x1  );

  EA=1;
}


void locate(short int fila, short int col )
{
 short int n ;
 EA=1;
 n = col;
 if( fila == 1 ) n = n + 0x40 ;
 n = n & 0x7f ;
 n = n + 0x80 ;
 control_display( n );
```

```
 EA=1;
}



/*********************************************************************/
/*  This file is part of the C-51 Compiler package                   */
/*  Copyright KEIL ELEKTRONIK GmbH 1990                              */
/*********************************************************************/
/*                                                                   */
/*  PUTCHAR.C:  Rutina modificada para que atienda el LCD HD44780    */
/*                                                                   */
/*********************************************************************/

#include <reg52.h>

sbit E     = P0^5  ;     // Lineas de control del display
sbit RS    = P0^7  ;     // Lineas de control del display
sbit RW    = P0^6  ;     // Lineas de control del display

char putchar (char c)

{
  int n;
  P2=c;
  RW=0;
  RS=1;
  E=1;
  for( n = 0 ; n < 20 ; n++ );
  E=0;
  for( n = 0 ; n < 20 ; n++ );
  return 0;
}


/* ------------------------------------------------------------------------ */
/*                                                                          */
/*    C¢digo fuente del RELOJ dise¤ado pare el Emulador de Ceibo EB-51      */
/*                                                                          */
/* ------------------------------------------------------------------------ */
/*                                                                          */
/*        Por Andr,s Rold n Aranda     Univ. Polot,cnica Superior           */
/* ------------------------------------------------------------------------ */

#pragma ROM (LARGE)     //   Las llamadas a CALL y JMP se codifican como
                        //  LCALL y LJMP. Esto permite acceder a lo largo
                        //  de todo el espacio de memoria direccionable.
                        //  El c¢digo de programa se puede localizar en los
                        // 64 Kbytes de memoria de c¢digo ROM.

#pragma small   //  Todas las variables, los segmentos de datos locales y los
                // procedimientos se obligan a residir en la parte de memoria
                // interna del microcontrolador. El acceso a ,stas, es pues muy
                // eficiente, y su desventaja es que presenta un zona de memo-
                // -ria reducida.

#pragma ot(5,SIZE)      // Direccitva que obliga al compilador a realizar una
                        // optimizaci¢n del c¢digo generado. En este caso se
                        // fuerza la optimizaci¢n en su m ximo nivel (=5) y
                        // se realizar  un ,nfasis especial en la optimizaci¢n
                        // del tama¤o del c¢digo.
                        //
                        // Para realizarlo en la velocidad, se utilizar  la
                        // siguiente directiva.
                        //   #pragma ot(5,SPEED)

#pragma DEBUG    //  Esta directiva indica al compilador que incluya la infor-
                 // -maci¢n necesaria para realizar el 'debugeo' en una
                 // herramienta SW. Esta informaci¢n contiene las definiciones
                 // de las variables locales y globales y su direcci¢n, as¡
                 // como los nombres de las funciones y sus n£meros de l¡nea.

#pragma SYMBOLS  //  Mediante esta directiva se genera un listado completo de todos
                 // los s¡mbolos usados en y por el modulo del programa al ser
                 // compilado. Para cada objeto simb¢lico se indican la descrip-
                 // -ci¢n de memoria, su categor¡a, el offset y el tama¤o del
                 // objeto.

#pragma LISTINCLUDE     //  Esta directiva permite incluir el contenido de
                        // todos los ficheros incluidos (#include <fich>) en
                        // el programa.

#pragma CODE    //  Esta directiva permite incluir una lista de los mnem¢nicos
                // de ensamblador al fichero de listado (fich.lst) El c¢digo
                // ensamblador se presenta paa cada funci¢n del programa.

//#pragma SRC


#include <reg52.h>   // Registros del 8052
#include <stdio.h>   // Procedimientos estandard de e/s

#define TIMER0_COUNT 0x0FA18  // 10000h -(12.000.000 / (12/T - 17 )


sbit LED1  = P1^3  ;     // L¡neas de control del LED_1
sbit LED2  = P1^2  ;     // L¡neas de control del LED_2

sbit E     = P0^5  ;     // Lineas de control del display
sbit RS    = P0^7  ;     // Lineas de control del display
sbit RW    = P0^6  ;     // Lineas de control del display

int cambio_de_hora=0;
bit     led1_on=1,
        led2_on=0,
        cambio_horario=0,
        reloj_activo=1,
        reloj_activo_en_LCD=1;

struct RTC {   int hora;
               int minutos;
               int segundos;
               unsigned int msec;
            };

static struct RTC reloj;

extern void init_display();
extern void borrar_display();
extern void locate(short int fila, short int col );
extern void dato_display( char dato );

void delay( unsigned int retardo)    // Procedimiento para introducir una espera
{
 unsigned int aux;
 for (aux=0;aux<retardo;aux++);
}

void actualizacion_cambio_de_hora( void )
{
    borrar_display();
    locate(0,0);
    printf("          Hora          %d:%d:%d", reloj.hora, reloj.minutos, reloj.segundos);
}

void refresco_LCD_con_hora( void )
{
    borrar_display();
    locate(1,6);
    actualizacion_cambio_de_hora();
}


static void interrupcion_0 (void) interrupt 0 using 2
{
  unsigned char n;

    EA=1;
    borrar_display();
    locate(1,1);
    // printf("      Se ha pulsado      Int 0.");
    printf("      TIME STOPPED.");
    delay(31000);
    delay(31000);
    refresco_LCD_con_hora();
    reloj_activo=~reloj_activo;

    EA=1;
}
```

```
static void interrupcion_1 (void) interrupt 2 using 2
{
  unsigned char n;

    EA=1;
    borrar_display();
    locate(0,0);
    //printf("     Se ha pulsado      Int 1.");
    printf("     ELAPSED TIME.");
    delay(31000);
    delay(31000);
    refresco_LCD_con_hora();
    reloj_activo_en_LCD=~reloj_activo_en_LCD;

    EA=1;
}


static void timer0 (void) interrupt 1 using 2
{

 EA=0;

   reloj.msec++;


 TR0=0;                          // Paramos el timer0
 TL0= TL0 + (TIMER0_COUNT & 0x00FF);
 TH0= TH0 + (TIMER0_COUNT >> 8);
 TR0=1;                          // Arrancamos el timer otra vez para 10 ms

 EA=1;
}

void timer0_init (void)
{
 EA=0;                           // Deshabilitamos las interrupciones
 TR0=0;                          // Paramos el timer0
 TMOD &= ~0x0F;                    // Borramos los bits del timer0
 TMOD |=0x01;                    // Ponemos el timer0 en contador de 16 bits
 TL0=(TIMER0_COUNT & 0x00FF);
 TH0=(TIMER0_COUNT >> 8);
 PT0=0;                          // Ponemos baja prioridad al timer0
 ET0=1;                          // Habilitamos la posibilid de int_timer0
 TR0=1;                          // Arrancamos el timer0
 EA=1;                           // Habilitamos interrupciones
}

void actualizar_valor_horario(bit reloj_activado, bit reloj_activado_en_LCD)
{
 if (reloj_activado)
 {
   if (reloj.msec>=1000)
    {
     reloj.msec=0;
     reloj.segundos++;

     if (reloj_activado_en_LCD)
      {
       cambio_horario=1;
      }

     if (reloj.segundos==60)
      {
       reloj.segundos=0;
       reloj.minutos++;
       if (reloj.minutos==60)
        {
         reloj.minutos=0;
         reloj.hora++;
        }
      }
    }

 }

}

main ()
{
  E     = 0;              // Lineas de control del display
  RS    = 0;              // Lineas de control del display
  RW    = 0;              // Lineas de control del display

  PT0=1;              // Alta prioridad al Timer0
  IT0=0;              // La INT0 es activa por flanco de bajada.
  IT1=0;              // La INT1 es activa por flanco de bajada.
  IE=0x87;            // Habilito INT_0, TImer0 e INT_1
  timer0_init ();
  init_display();
  delay(31000);

  while(1)
  {

   if (cambio_horario==1)
    {
     actualizacion_cambio_de_hora();
     cambio_horario=0;
    }
   actualizar_valor_horario(reloj_activo, reloj_activo_en_LCD);

   // Al pulsar INT0 se actua sobre el start/stop del reloj del LCD.
   // Al pulsar INT1 se actua sobre el laps del reloj del LCD.

  }



}

/* ------------------------------------------------------------------------ */
/*                                                                          */
/*    C¢digo fuente del RELOJ dise¤ado pare el Emulador de Ceibo EB-51      */
/*                                                                          */
/*                Por   Andr,s Rold n Aranda  ¸ 1998                        */
/* ------------------------------------------------------------------------ */
```