```
/*****************************************************************************************/
/* Program        : LCD44780.C                                                           */
/* Function       : LCD Control Procedures for HD44780 Controller                        */
/* Author         : John F. Fitter B.E.                                                  */
/*                                                                                       */
/* Description    : Procedures to drive the Hitachi LCD module - 1 to 4 line x nn char   */
/*                  - in 4 bit mode                                                      */
/*                                                                                       */
/* Useage         : lcd_init()           Must be called before any other function       */
/*                  putch(chr)           Display chr at the current cursor position     */
/*                                       This is stdout for the Hitech compiler         */
/*                                       The following have special meaning;            */
/*                                           \f  Clear display                          */
/*                                           \n  Goto start of next line               */
/*                                           \r  Goto start of current line             */
/*                                           \b  Backspace one position                 */
/*                  lcd_gotoxy(c)       Set cursor position on LCD, Upper left is (0,0)  */
/*                                           c : bits 0..4 = character position in line  */
/*                                           c : bits 5..6 = line number                 */
/*                                           c : bit  7    = cursor on status            */
/*                                       In the case of single line displays the lower 6 bits*/
/*                                       are the character position.                     */
/*                  lcd_shift(n)         Shifts display n chars, +ve for right, -ve for left */
/*                  lcd_write_cgram(c,p) Writes a cgram character line pattern to the cgram. */
/*                                           c = character position to write             */
/*                                           p = character pattern to write              */
/*                  lcd_blank_display(b) Blank or restore the display restoring the cursor   */
/*                                       status also. b is true to blank the display.    */
/*                                                                                       */
/* Rev No.   Rev date   Test date   Test platform     Description                        */
/* -------   --------   ---------   ---------------   ----------------------------------    */
/*     00    6jun98                                  Original in HiTech C               */
/*     01    25jun98                                 Total rewrite of lcd_gotoxy        */
/*                                                   Improved lcd_shift                 */
/*                                                                                       */
/*           Copyright © 1998 Eagle Air Australia Pty. Ltd. All rights reserved          */
/*****************************************************************************************/

#define _LCD44780_C

#include <commdefs.h>
#include "main.h"
#include "lcd44780.h"
#include "1306spi.h"
#include "delays.h"
#include "serial.h"
#include "25cxxspi.h"
#include "e2data.h"

void lcd_send_byte(unsigned char address, unsigned char chr) {
    unsigned char rVal;

    LCDPORTDIR = LCD_READ;                              // set data/con port directions
    lcd_select_reg(CMD_REG);                            // select command register
    lcd_select_dir(READ_DIR);                           // set lcd to read

    do {                                                // wait until lcd is ready
        lcd_set_clk_hi();                               // toggle data clock
        rVal = lcd_get_data();                          // read high nibble from data bus
        lcd_set_clk_lo();
        delay_500ns();
        lcd_toggle_clk();                               // toggle data clock
    } while(!!(rVal & (HINIBBLE(LCD_BUSY))));

    LCDPORTDIR = LCD_WRITE;                             // set data/con port directions
    lcd_select_reg(address);                            // select desired register
    lcd_select_dir(WRITE_DIR);                          // set to write
    lcd_set_data(HINIBBLE(chr));                        // send the high nibble
    lcd_toggle_clk();                                   // toggle data clock
    lcd_set_data(chr);                                  // send the low nibble
    lcd_toggle_clk();                                   // toggle data clock
}

// Procedure to initialise the lcd for 4 bit operation.
// This procedure implements the "Initialisation by instruction" as described for
// Philips PCF2116X
```

```
void init_lcd() {
    unsigned char n;

    LCDPORTDIR = LCD_WRITE;                                 // set data/con & enable port dir'ns
    lcd_set_clk_lo();                                       // set data clock low (enable pin)
    lcd_select_reg(CMD_REG);                                // select command register
    lcd_select_dir(WRITE_DIR);                              // set to write
    delay_ms(15);                                           // wait > 15ms Vdd rises above Vpor
    lcd_set_data(HINIBBLE(LCD_FNSET | LCD_8BIT));           // put nibble on data bus
    for (n = 0; n < 3; ++n) {                               // set mode to 8 bit data 3 times
        lcd_toggle_clk();                                   // toggle data clock
        delay_ms(5);                                        // 5ms delay
    }
    #ifdef LCD1LINES && (LCDNCHARS > 16)
    lcd_set_data(HINIBBLE(LCD_FNSET));                      // set mode to 4 bit data and 1 line
    lcd_toggle_clk();                                       // toggle data clock
    lcd_send_byte(LCD_COMMAND, LCD_FNSET);
    #else
    #ifdef LCD4LINES
    lcd_set_data(HINIBBLE(LCD_FNSET | LCD_4LINE));          // set mode to 4 bit data and 4 lines
    lcd_toggle_clk();                                       // toggle data clock
    lcd_send_byte(LCD_COMMAND, LCD_FNSET | LCD_4LINE);
    #else
    lcd_set_data(HINIBBLE(LCD_FNSET | LCD_2LINE));          // set mode to 4 bit data and 2 lines
    lcd_toggle_clk();                                       // toggle data clock
    lcd_send_byte(LCD_COMMAND, LCD_FNSET | LCD_2LINE);      // (applies to 16x1 also - odd one!!)
    #endif
    #endif
    lcd_send_byte(LCD_COMMAND, LCD_DISPON | LCD_DISP_ALL); // display on
    lcd_send_byte(LCD_COMMAND, LCD_CLR);                    // clear the display
    lcd_send_byte(LCD_COMMAND, LCD_EMSET | LCD_EM_INC);     // entry mode to increment
    LCDPORTDIR = LCD_READ;                                  // set data/con port dir'ns
    linenum = 0;                                            // set line number to first line
    disp_blank = false;                                     // flag display as visible
    cursor_on  = false;                                     // flag the cursor as off
}

// Moves the cursor to posline. The first character position is 0 and the first line is 0
// If the cursor bit is set then a blinking cursor location is shown else the blinking
// is removed. The procedure returns the old cursor status.
// Definition of posline : bits 0..4 = character position in line (left = 0)
//                         bits 5..6 = line number (top = 0)
//                         bit  7    = cursor status (on = 1)
// In the case of single line displays the lower 6 bits are the character position.

unsigned char lcd_gotoxy(unsigned char posline) {

    unsigned char address, cstat;

    cstat = cursor_on;
    if(c_status.prt_to_lcd) {
        #ifdef LCD1LINES                                    // this method is faster than using the
            linenum = 0;                                    // modulus operator but more verbose
            address = posline & 0x3f;
        #else
            linepos = posline & 0x1f;                       // save the line char position
            linenum = (posline >> 5) & 3;                   // save the line number
            #ifdef LCD2LINES
                if(linenum > 1) linenum = 0;                // 2 line display ?
            #else
                if(linenum > 3) linenum = 0;                // 4 line display ?
            #endif  // LCD2LINES
                address = linepos;
                if(linenum == 1)      address += STRT_LINE2; // set ram address to (pos,line)
                else if(linenum == 2) address += STRT_LINE3;
                else if(linenum == 3) address += STRT_LINE4;
        #endif  // LCD1LINES
        lcd_send_byte(LCD_COMMAND, LCD_DD_SET | address);
        if(!!(posline & 0x80) != cursor_on) {               // has the cursor status changed ?
            cursor_on = !!(posline & 0x80);
            if(disp_blank) lcd_send_byte(LCD_COMMAND, LCD_DISPON);
            else if(cursor_on) lcd_send_byte(LCD_COMMAND, LCD_DISPON |
                LCD_DISP_ALL | LCD_DISP_BLNK);
            else lcd_send_byte(LCD_COMMAND, LCD_DISPON|LCD_DISP_ALL);
        }
    }
    return cstat;
```

```
}

// Blank or restore the display restoring the cursor status also.

void lcd_blank_display(unsigned char blank) {
    disp_blank = blank;
    if(disp_blank) lcd_send_byte(LCD_COMMAND, LCD_DISPON);
    else if(cursor_on) lcd_send_byte(LCD_COMMAND, LCD_DISPON | LCD_DISP_ALL | LCD_DISP_BLNK);
    else lcd_send_byte(LCD_COMMAND, LCD_DISPON|LCD_DISP_ALL);
}

// If the print flag is set to lcd then writes a character to the display at cursor. Backspace,
// newline and formfeed are recognised. Formfeed clears the display, newline moves the cursor
// to the start of the second line. Programmable characters have codes between 0x0f and 0x1f.
// They are mapped to cgram, 0x0 to 0xf. This makes all of the cgram available to programmable
// characters and also allows string printing without premature termination on 0.
// If the print flag is set to serial then the character is sent directly to the serial port.

void putch(char chr) {
    if(c_status.prt_to_lcd) {                           // printing to lcd
        if(chr == '\f') {
            lcd_send_byte(LCD_COMMAND, LCD_CLR);        // formfeed clears the display
            linenum = 0;                                // set line number to first line
        }
        // Newline to start of next line.
        else if(chr == '\n') lcd_gotoxy((cursor_on ? 0x80 : 0) | ((linenum + 1) << 5));

        // Return to start of current line.
        else if(chr == '\r') lcd_gotoxy((cursor_on ? 0x80 : 0) | (linenum << 5));

        // Backspace.
        else if(chr == '\b') lcd_send_byte(LCD_COMMAND, LCD_SHIFT);

        // Other characters.
        else {
            if((chr < 0x20)&&(chr > 0x0f)) chr -= 0x10; // translate special characters
            lcd_send_byte(LCD_DATA, chr);               // write at current cursor position
        }
    }else {                                             // printing to usart
        if(chr == 0x11) chr = xl_super2;                // translate superscript 2
        putch_ser(chr);                                 // send char to serial port
        if(chr == '\n') putch_ser('\r');                // make cr/lf pair if required
    }
}

// Clear m characters on the lcd by writing m spaces then m backspaces.

void clear_line(unsigned char m) {
    unsigned char n = m;

    if(c_status.prt_to_lcd) {
        while(n--) lcd_send_byte(LCD_DATA, ' ');
        while(m--) lcd_send_byte(LCD_COMMAND, LCD_SHIFT);
    }
}

// Shift the screen left or right. npos is the number of character positions to shift,
// +ve for right and -ve for left.
/*
void lcd_shift(signed char npos) {
    unsigned char command;

    command = LCD_SHIFT | LCD_DISP_SHFT;
    if(npos > 0) command |= LCD_SHFT_RT;
    else npos = -npos;
    while(npos--) lcd_send_byte(LCD_COMMAND, command);
}
*/

// Writes a cgram character line pattern to the cgram.

void lcd_write_cgram(unsigned char address, unsigned char pattern) {

    lcd_send_byte(LCD_COMMAND, (address & 0x3f) | LCD_CG_SET); // set cgram addr.
    lcd_send_byte(LCD_DATA, pattern);
}
```

```
// ***** EOF LCD44780.C *****
```