

```

/*****
/* Program      : LCD44780.H
/* Function     : LCD Utility Control Procedures Header File for HD44780 Controller
/* Author      : John F. Fitter B.E.
/*
/*              Copyright © 1998 Eagle Air Australia Pty. Ltd. All rights reserved
/*
*****/

#ifndef _LCD44780_H
#define _LCD44780_H

#define CMD_REG          0
#define DATA_REG       1
#define WRITE_DIR       0
#define READ_DIR        1

#define RIGHT           1
#define LEFT            0

#define BL_ON           1
#define BL_OFF          0

// LCD defines for calling lcd_gotoxy with line number, position and cursor status
// packed into one byte.
#define LINE0           0
#define LINE1           0x20
#define LINE2           0x40
#define LINE3           0x60
#define CSRON           0x80

// LCD control defines
#define LCDPORT          PORTD // control and data port for lcd
#define LCDPORTDIR      TRISD // direction register for lcd
#define LCDLONBL         // define lcd comms on low nibble
// #define LCD1LINES      // define number of display lines
// #define LCD2LINES      // (only define one of these)
#define LCD4LINES
#define LCDNCHARS       20 // define number of chars per line

// LCD macros
#define lcd_set_clk_hi() lcd.en=1;delay_500ns()
#define lcd_set_clk_lo() lcd.en=0
#define lcd_select_reg(reg) lcd.rs=reg
#define lcd_select_dir(dir) lcd.rw=dir
#define lcd_get_data()    lcd.data
#define lcd_toggle_clk()  lcd_set_clk_hi();lcd_set_clk_lo()

// Because the backlight drive is from the lcd port and the backlight is interrupt driven,
// a backlight interrupt can come during a write of the lcd data bits. The write is actually
// a read, clear the data bits, or in the new data, and then write. If the backlight
// interrupt changes the backlight bit while this is happening, the write will reset the bit
// to the old value. The effect of this is random flashing of the backlight. To correct this,
// the interrupts need to be disabled during the write. A better alternative is to manipulate
// the port bits directly since invalid data on the port will not effect the lcd. The
// interrupts can now be left running.
#ifdef LCDLONBL
#define lcd_set_data(dat) LCDPORT&=0xf0;LCDPORT|=dat&0xf
#else
#define lcd_set_data(dat) LCDPORT&=0xf;LCDPORT|=dat<<4
#endif // _LCDLONBL

// LCD backlight defines
// The backlight is driven from timer 2 using the pr2 compare register. The prescaler is set to
// 16 and the post scaler is set to 10 for 16MHz clock and 5 for 8MHz clock. The backlight is
// thus on for 40*(pr2+1) uS. pr2 can range from 0 to 24 giving on times from 40 to 1000uS.
// If the off time is set to a pr2 value of 24-(on value) then a 1kHz backlight drive will
// result with duty cycle adjustable from 4 to 100% in 24 increments.

#define LCDBLPLOG // backlight logic (positive logic)

// LCD Backlight macros
#define enable_lcd_bl(state) flg_lcd_bl_en=!state
#define lcd_bl_isenabled()  flg_lcd_bl_en
#define reset_lcd_bl()      bl_ontime=read_1306(BLONADDR);\
                             if(!bl_ontime|| (bl_ontime>23)){bl_ontime=18;write_1306(BLONADDR,20);}
#define brtn_lcd_bl(dirn)  bl_ontime=(bl_ontime+(dirn?24:22))%23;write_1306(BLONADDR,bl_ontime)
#define set_bl_ontime()    PR2=bl_ontime

```

```

#define set_bl_offtime()          PR2=22-bl_ontime
#ifdef _FAST_CLOCK
#define init_lcd_bl()             bl_ontime=18;\
                                  set_bl_ontime();\
                                  enable_lcd_bl(true);\
                                  set_lcd_bl(true);\
                                  setup_timer_2(T2_DIV_BY_16,10);\
                                  enable_interrupts(INT_TIMER2)
#else
#define init_lcd_bl()             bl_ontime=18;\
                                  set_bl_ontime();\
                                  enable_lcd_bl(true);\
                                  set_lcd_bl(true);\
                                  setup_timer_2(T2_DIV_BY_16,5);\
                                  enable_interrupts(INT_TIMER2)
#endif // _FAST_CLOCK
#ifdef LCDBLPL0G
#define lcd_bl_ison()             flg_lcd_bl_hi
#define set_lcd_bl(state)         lcd.bl=flg_lcd_bl_en&state;flg_lcd_bl_hi=!!state
#else
#define lcd_bl_ison()             flg_lcd_bl_hi
#define set_lcd_bl(state)         lcd.bl=!(flg_lcd_bl_en&state);flg_lcd_bl_hi=!!state
#endif // LCDBLPL0G

// LCD specific defines
#define LCD_CLR                    1 // clear display
#define LCD_HOME                   2 // return home
#define LCD_EMSET                   4 // entry mode set
#define LCD_DISPON                  8 // display on/off control
#define LCD_SHIFT                   0x10 // cursor/display shift
#define LCD_FNSET                   0x20 // function set
#define LCD_CG_SET                  0x40 // CG RAM address set
#define LCD_DD_SET                  0x80 // DD RAM address set
#define LCD_BUSY                    0x80 // Busy flag
#define LCD_EM_INC                   2 // entry mode increment
#define LCD_EM_SH                    1 // entry mode accompanies shift
#define LCD_DISP_ALL                 4 // display on/off all display
#define LCD_DISP_CSR                 2 // display on/off cursor
#define LCD_DISP_BLNK                1 // display on/off cursor blink
#define LCD_DISP_SHFT               8 // display shift
#define LCD_SHFT_RT                  4 // shift to right
#define LCD_8BIT                    0x10 // 8 bit data
#define LCD_2LINE                    8 // 2 lines
#define LCD_4LINE                   0x0c // 4 lines
#define LCD_5X10                     4 // 5x10 pixels per character
#define STRT_LINE2                   0x40 // address of line 2 is 0x40 always
#define STRT_LINE3                   LCDNCHARS
#define STRT_LINE4                   STRT_LINE2+LCDNCHARS
#define LCD_COMMAND                  0 // lcd_send_byte cmd/data parameter
#define LCD_DATA                     1

// LCD structures
#define LCD_WRITE                    0; // lcd data directions for write
#ifdef LCDLONBL
#define LCD_READ                    0xf; // lcd data directions for read
// data bus is low nibble
// LCD data pins 0..3
// read/write pin 4
// register select pin 5
// enable pin 6
// backlight drive pin 7
struct lcd_pin_map {
    unsigned data :4;
    unsigned rw :1;
    unsigned rs :1;
    unsigned en :1;
    unsigned bl :1;
};
#else
#define LCD_READ                    0xf0; // lcd data directions for read
// data bus is high nibble
// read/write pin 0
// register select pin 1
// enable pin 2
// backlight drive pin 7
// LCD data pins 4..7
struct lcd_pin_map {
    unsigned rw :1;
    unsigned rs :1;
    unsigned en :1;
    unsigned bl :1;
    unsigned data :4;
};
#endif // _LCDLONBL

// Static variables
// map lcd data & control port to lcd backlight drive lcd backlight drive direction
static volatile struct lcd_pin_map lcd @ (unsigned)&LCDPORT;

```

```
#ifndef _LCD44780_C
bank1 volatile bit flg_lcd_bl_en; // lcd backlight is enabled
bank1 volatile bit flg_lcd_bl_hi; // backlight is on
bank1 volatile unsigned char bl_ontime; // backlight on period
bank1 unsigned char linenum; // display cursor line number (0 to 3)
bank1 unsigned char linepos; // display cursor char position number
bank1 bit cursor_on; // flag cursor is being displayed
bank1 bit disp_blank; // flag display is blanked
#else
extern bank1 volatile bit flg_lcd_bl_en;
extern bank1 volatile bit flg_lcd_bl_hi;
extern bank1 volatile unsigned char bl_ontime;
extern bank1 unsigned char linenum;
extern bank1 unsigned char linepos;
extern bank1 bit cursor_on;
extern bank1 bit disp_blank;
#endif // _LCD44780_C

// Function prototypes
extern void lcd_send_byte(unsigned char, unsigned char);
extern void init_lcd();
extern unsigned char lcd_gotoxy(unsigned char);
extern void lcd_blank_display(unsigned char);
extern void putchar(char);
//extern void lcd_shift(signed char);
extern void lcd_write_cgram(unsigned char, unsigned char);
extern void clear_line(unsigned char);

#endif // _LCD44780_H

// ***** EOF LCD44780.H *****
```