```
#ifndef LCD_H
#define LCD_H
/*************************************************************************
*
 Title      :   C include file for the HD44780U LCD library (lcd.c)
 Author:    Peter Fleury <pfleury@gmx.ch>  http://jump.to/fleury
 File:         $Id: lcd.h,v 1.13.2.2 2006/01/30 19:51:33 peter Exp $
 Software:  AVR-GCC 3.3
 Hardware:  any AVR device, memory mapped mode only for
AT90S4414/8515/Mega
*************************************************************************
**/

/**
 @defgroup pfleury_lcd LCD library
 @code #include <lcd.h> @endcode

 @brief Basic routines for interfacing a HD44780U-based text LCD display

 Originally based on Volker Oth's LCD library,
 changed lcd_init(), added additional constants for lcd_command(),
 added 4-bit I/O mode, improved and optimized code.

 Library can be operated in memory mapped mode (LCD_IO_MODE=0) or in
 4-bit IO port mode (LCD_IO_MODE=1). 8-bit IO port mode not supported.

 Memory mapped mode compatible with Kanda STK200, but supports also
 generation of R/W signal through A8 address line.

 @author Peter Fleury pfleury@gmx.ch http://jump.to/fleury

 @see The chapter <a
href="http://homepage.sunrise.ch/mysunrise/peterfleury/avr-lcd44780.html"
target="_blank">Interfacing a HD44780 Based LCD to an AVR</a>
      on my home page.

*/

/*@{*/

#if (__GNUC__ * 100 + __GNUC_MINOR__) < 303
#error "This library requires AVR-GCC 3.3 or later, update to newer AVR-
GCC compiler !"
#endif

#include <inttypes.h>
#include <avr/pgmspace.h>

/**
 *  @name  Definitions for MCU Clock Frequency
 *  Adapt the MCU clock frequency in Hz to your target.
 */
#define XTAL 4000000              /**< clock frequency in Hz, used to
calculate delay timer */
```

```
/**
 * @name  Definition for LCD controller type
 * Use 0 for HD44780 controller, change to 1 for displays with KS0073
controller.
 */
#define LCD_CONTROLLER_KS0073 0  /**< Use 0 for HD44780 controller, 1 for
KS0073 controller */

/**
 *  @name  Definitions for Display Size
 *  Change these definitions to adapt setting to your display
 */
#define LCD_LINES           2     /**< number of visible lines of the
display */
#define LCD_DISP_LENGTH    16     /**< visibles characters per line of
the display */
#define LCD_LINE_LENGTH  0x40     /**< internal line length of the
display    */
#define LCD_START_LINE1  0x00     /**< DDRAM address of first char of
line 1 */
#define LCD_START_LINE2  0x40     /**< DDRAM address of first char of
line 2 */
#define LCD_START_LINE3  0x14     /**< DDRAM address of first char of
line 3 */
#define LCD_START_LINE4  0x54     /**< DDRAM address of first char of
line 4 */
#define LCD_WRAP_LINES      0     /**< 0: no wrap, 1: wrap at end of
visibile line */


#define LCD_IO_MODE      1         /**< 0: memory mapped mode, 1: IO port
mode */
#if LCD_IO_MODE
/**
 *  @name Definitions for 4-bit IO mode
 *  Change LCD_PORT if you want to use a different port for the LCD pins.
 *
 *  The four LCD data lines and the three control lines RS, RW, E can be
on the
 *  same port or on different ports.
 *  Change LCD_RS_PORT, LCD_RW_PORT, LCD_E_PORT if you want the control
lines on
 *  different ports.
 *
 *  Normally the four data lines should be mapped to bit 0..3 on one
port, but it
 *  is possible to connect these data lines in different order or even on
different
 *  ports by adapting the LCD_DATAx_PORT and LCD_DATAx_PIN definitions.
 *
 */
#define LCD_PORT         PORTA        /**< port for the LCD lines   */
```

```
#define LCD_DATA0_PORT   LCD_PORT     /**< port for 4bit data bit 0 */
#define LCD_DATA1_PORT   LCD_PORT     /**< port for 4bit data bit 1 */
#define LCD_DATA2_PORT   LCD_PORT     /**< port for 4bit data bit 2 */
#define LCD_DATA3_PORT   LCD_PORT     /**< port for 4bit data bit 3 */
#define LCD_DATA0_PIN    3            /**< pin for 4bit data bit 0  */
#define LCD_DATA1_PIN    2            /**< pin for 4bit data bit 1  */
#define LCD_DATA2_PIN    1            /**< pin for 4bit data bit 2  */
#define LCD_DATA3_PIN    0            /**< pin for 4bit data bit 3  */
#define LCD_RS_PORT      LCD_PORT     /**< port for RS line         */
#define LCD_RS_PIN       6            /**< pin  for RS line         */
#define LCD_RW_PORT      LCD_PORT     /**< port for RW line         */
#define LCD_RW_PIN       5            /**< pin  for RW line         */
#define LCD_E_PORT       LCD_PORT     /**< port for Enable line     */
#define LCD_E_PIN        4            /**< pin  for Enable line     */

#elif defined(__AVR_AT90S4414__) || defined(__AVR_AT90S8515__) ||
defined(__AVR_ATmega64__) || \
      defined(__AVR_ATmega8515__)|| defined(__AVR_ATmega103__) ||
defined(__AVR_ATmega128__) || \
      defined(__AVR_ATmega161__) || defined(__AVR_ATmega162__)
/*
 *  memory mapped mode is only supported when the device has an external
data memory interface
 */
#define LCD_IO_DATA      0xC000    /* A15=E=1, A14=RS=1
*/
#define LCD_IO_FUNCTION  0x8000    /* A15=E=1, A14=RS=0
*/
#define LCD_IO_READ      0x0100    /* A8 =R/W=1 (R/W: 1=Read, 0=Write
*/
#else
#error "external data memory interface not available for this device, use
4-bit IO port mode"

#endif


/**
 *  @name Definitions for LCD command instructions
 *  The constants define the various LCD controller instructions which
can be passed to the
 *  function lcd_command(), see HD44780 data sheet for a complete
description.
 */

/* instruction register bit positions, see HD44780U data sheet */
#define LCD_CLR               0      /* DB0: clear display
*/
#define LCD_HOME              1      /* DB1: return to home position
*/
#define LCD_ENTRY_MODE        2      /* DB2: set entry mode
*/
#define LCD_ENTRY_INC         1      /*   DB1: 1=increment, 0=decrement
*/
```

```
#define LCD_ENTRY_SHIFT       0      /*   DB2: 1=display shift on
*/
#define LCD_ON                3      /* DB3: turn lcd/cursor on
*/
#define LCD_ON_DISPLAY        2      /*   DB2: turn display on
*/
#define LCD_ON_CURSOR         1      /*   DB1: turn cursor on
*/
#define LCD_ON_BLINK          0      /*     DB0: blinking cursor ?
*/
#define LCD_MOVE              4      /* DB4: move cursor/display
*/
#define LCD_MOVE_DISP         3      /*   DB3: move display (0-> cursor)
?  */
#define LCD_MOVE_RIGHT        2      /*   DB2: move right (0-> left) ?
*/
#define LCD_FUNCTION          5      /* DB5: function set
*/
#define LCD_FUNCTION_8BIT     4      /*   DB4: set 8BIT mode (0->4BIT
mode) */
#define LCD_FUNCTION_2LINES   3      /*   DB3: two lines (0->one line)
*/
#define LCD_FUNCTION_10DOTS   2      /*   DB2: 5x10 font (0->5x7 font)
*/
#define LCD_CGRAM             6      /* DB6: set CG RAM address
*/
#define LCD_DDRAM             7      /* DB7: set DD RAM address
*/
#define LCD_BUSY              7      /* DB7: LCD is busy
*/

/* set entry mode: display shift on/off, dec/inc cursor move direction */
#define LCD_ENTRY_DEC            0x04   /* display shift off, dec cursor
move dir */
#define LCD_ENTRY_DEC_SHIFT      0x05   /* display shift on,  dec cursor
move dir */
#define LCD_ENTRY_INC_           0x06   /* display shift off, inc cursor
move dir */
#define LCD_ENTRY_INC_SHIFT      0x07   /* display shift on,  inc cursor
move dir */

/* display on/off, cursor on/off, blinking char at cursor position */
#define LCD_DISP_OFF             0x08   /* display off
*/
#define LCD_DISP_ON              0x0C   /* display on, cursor off
*/
#define LCD_DISP_ON_BLINK        0x0D   /* display on, cursor off, blink
char     */
#define LCD_DISP_ON_CURSOR       0x0E   /* display on, cursor on
*/
#define LCD_DISP_ON_CURSOR_BLINK 0x0F   /* display on, cursor on, blink
char      */

/* move cursor/shift display */
```

```
#define LCD_MOVE_CURSOR_LEFT     0x10   /* move cursor left  (decrement)
*/
#define LCD_MOVE_CURSOR_RIGHT    0x14   /* move cursor right (increment)
*/
#define LCD_MOVE_DISP_LEFT       0x18   /* shift display left
*/
#define LCD_MOVE_DISP_RIGHT      0x1C   /* shift display right
*/

/* function set: set interface data length and number of display lines */
#define LCD_FUNCTION_4BIT_1LINE  0x20   /* 4-bit interface, single line,
5x7 dots */
#define LCD_FUNCTION_4BIT_2LINES 0x28   /* 4-bit interface, dual line,
5x7 dots */
#define LCD_FUNCTION_8BIT_1LINE  0x30   /* 8-bit interface, single line,
5x7 dots */
#define LCD_FUNCTION_8BIT_2LINES 0x38   /* 8-bit interface, dual line,
5x7 dots */


#define LCD_MODE_DEFAULT     ((1<<LCD_ENTRY_MODE) | (1<<LCD_ENTRY_INC) )



/**
 *  @name Functions
 */


/**
 @brief    Initialize display and select type of cursor
 @param    dispAttr \b LCD_DISP_OFF display off\n
                    \b LCD_DISP_ON display on, cursor off\n
                    \b LCD_DISP_ON_CURSOR display on, cursor on\n
                    \b LCD_DISP_ON_CURSOR_BLINK display on, cursor on
flashing
 @return  none
*/
extern void lcd_init(uint8_t dispAttr);


/**
 @brief    Clear display and set cursor to home position
 @param    void
 @return   none
*/
extern void lcd_clrscr(void);


/**
 @brief    Set cursor to home position
 @param    void
 @return   none
*/
```

```
extern void lcd_home(void);


/**
 @brief    Set cursor to specified position

 @param    x horizontal position\n (0: left most position)
 @param    y vertical position\n   (0: first line)
 @return   none
*/
extern void lcd_gotoxy(uint8_t x, uint8_t y);


/**
 @brief    Display character at current cursor position
 @param    c character to be displayed
 @return   none
*/
extern void lcd_putc(char c);


/**
 @brief    Display string without auto linefeed
 @param    s string to be displayed
 @return   none
*/
extern void lcd_puts(const char *s);


/**
 @brief    Display string from program memory without auto linefeed
 @param    s string from program memory be be displayed
 @return   none
 @see      lcd_puts_P
*/
extern void lcd_puts_p(const char *progmem_s);


/**
 @brief    Send LCD controller instruction command
 @param    cmd instruction to send to LCD controller, see HD44780 data
sheet
 @return   none
*/
extern void lcd_command(uint8_t cmd);


/**
 @brief    Send data byte to LCD controller

 Similar to lcd_putc(), but without interpreting LF
 @param    data byte to send to LCD controller, see HD44780 data sheet
 @return   none
*/
```

```
extern void lcd_data(uint8_t data);


/**
 @brief macros for automatically storing string constant in program
memory
*/
#define lcd_puts_P(__s)         lcd_puts_p(PSTR(__s))

/*@}*/
#endif //LCD_H
```