```
// ------------------------------------------------------------------------------------
// DisplayDriver.c file for a 16*2 char LCD interfaced with the ATMEL ATMega32 MCU
// By - Nandan Banerjee
// Updated by - Carlos E. Marciales
// Date - 08/02/2010
// Compiled using AVR-GCC (WinAVR)
// IDE - AVRStudio 4.18

// 8-bit communication mode for the HD44780
// Be sure to set the Frequency field in project properties
// This was targeted to an ATmega32

#include <avr/io.h>
#include <avr/pgmspace.h>
#include <util/delay.h>

// This implementation uses Port D for the Data port
#define D0 PD0
#define D1 PD1
#define D2 PD2
#define D3 PD3
#define D4 PD4
#define D5 PD5
#define D6 PD6
#define D7 PD7
#define DATA_PORT PORTD
#define DDRDATA_PORT DDRD
// This implementation uses Port A for the Control port
#define RS PA7
#define RW PA6 // Will be 0 most of the time since we will be writing
#define E  PA5
#define COMM_PORT PORTA
#define DDRCOMM_PORT DDRA

// This function clears the RS line to write a command
void lcd_set_write_instruction() {
  COMM_PORT &= ~(1<<RS);
  _delay_us(50);
}

// This function sets the RS line to write data
void lcd_set_write_data() {
  COMM_PORT |= (1<<RS);
  _delay_us(50);
}

// This function writes a byte to the LCD
void lcd_write_byte(char c) {
  DATA_PORT = c;         // Place data on Data Port
  COMM_PORT |= (1<<PA5); // Toggle the E line to latch the data in LCD
  _delay_us(50);
  COMM_PORT &= ~(1<<PA5);
  _delay_us(50);
```

```
}

// This function clears LCD and sends address to beginning of first line
void lcd_clear_and_home() {
  lcd_set_write_instruction();
  lcd_write_byte(0x01);
  _delay_us(50);
  lcd_write_byte(0x02);
  _delay_us(50);
}

// This function ends address to beginning of first line
void lcd_home() {
  lcd_set_write_instruction();
  lcd_write_byte(0x02);
  _delay_ms(50);
}

// This function moves cursor to a given line and position
//  line is either 0 (first line) or 1 (second line)
//  pos is the character position from 0 to 15.
void lcd_goto(uint8_t line, uint8_t pos)
{
uint8_t position = 0;
  lcd_set_write_instruction();
  switch(line)
  {
    case 0: position = 0;
    break;
    case 1: position = 0x40;
    break;
  }
  lcd_write_byte(0x80 | (position + pos));
}

// This function moves cursor to 1st character of 1st line
void lcd_line_one() { lcd_goto(0, 0); }

// This function moves cursor to 1st character if 2nd line
void lcd_line_two() { lcd_goto(1, 0); }

// This function writes a character to the LCD
void lcd_write_data(char c) {
  lcd_set_write_data();
  lcd_write_byte(c);
}

// This function writes a string (in SRAM) of given length to the LCD
void lcd_write_string(char *x, uint8_t len ) {
  while (--len > 0)
    lcd_write_data(*x++);
}
```

```
// Same as above, but the string is located in program memory,
//  so "lpm" instructions are needed to fetch it, and a \0
//  must be defined at the end of the string to terminate it.
void lcd_write_string_p(const char *s)
{
  char c;

  for (c = pgm_read_byte(s); c; ++s, c = pgm_read_byte(s))
    lcd_write_data(c);
}

// This function initializes the LCD
void lcd_init() {

  lcd_set_write_instruction();

  lcd_write_byte(0x38); // Set Data length as 8 (DL bit set) and
  _delay_us(50);         // no. of display lines to 2 (N bit set)

  lcd_write_byte(0x0c); // Enable LCD (D bit set)
  _delay_us(50);

  lcd_write_byte(0x01); // Clear the LCD display
  _delay_us(50);

  lcd_write_byte(0x06); // Set entry mode: Increment cursor by 1
  _delay_us(50);         // after data read/write (I/D bit set)

  lcd_write_byte(0x14); // Cursor shift
  _delay_us(50);

  lcd_clear_and_home(); // LCD cleared and cursor is brought to
  _delay_us(50);         // the beginning of 1st line

}

// ---------------------------------------------------------------------------
//
// How to use the LCD routines
//
// ---------------------------------------------------------------------------
static uint16_t Tset=0x11F, Tin=0x123;

int main () {

DATA_PORT    = 0x00; // Initialize ports
DDRDATA_PORT = 0xFF;
COMM_PORT    = 0x00;
DDRCOMM_PORT = 0xFF;

//Initialize the LCD
  lcd_init();
```

```
// Write messages created in SRAM space
static char MsgHello[]  = "Hello AVR World!";
static char MsgPrompt[] = "Set=   Inside=  ";
  lcd_line_one();
  lcd_write_string(MsgHello, sizeof(MsgHello));
  lcd_line_two();
  lcd_write_string(MsgPrompt, sizeof(MsgPrompt));

// Write a string from program space so that it doesn't take up SRAM space:
  lcd_line_one();
  lcd_write_string_p(PSTR("Hello AVR World!\0"));
  lcd_line_two();
  lcd_write_string_p(PSTR("Set=   Inside=  \0"));

// Write numbers to the display at given locations
// Tset and Tin are ADC values taken with the 2.56V reference voltage. This gives
//  actual temp from an LM34DZ temp sensor in 10mV/°F by dividing the result
//  by 4 (Tset>>2 is shift right twice). Then convert that number into 2 ASCII digits.
uint8_t temp;
  if (Tset >= 0x18F) Tset = 0x18F; //set max temp at 99°F
// put cursor after the 1st "="
  lcd_goto(1,4);
  temp=(Tset>>2)/10;
  lcd_write_data(temp + 0x30);
  lcd_write_data((Tset>>2) - (temp*10) + 0x30);
// put cursor after second "="
  lcd_goto(1,14);
  temp=(Tin>>2)/10;
  lcd_write_data(temp + 0x30);
  lcd_write_data((Tin>>2) - (temp*10) + 0x30);
}
//
// ------------------------------------------------------------------------------
```