Muhammad Sharjeel and Naafiul Hossain

115185427

115107623

Lab 7: Multi-file Embedded C Program

ESE 381 Section L02

Bench 7

Breadboard: K2

# 381 PreLab 7 Table of Contents

# temp_mass_modular

AUTHOR
Version 1.1

# Table of Contents

Table of contents

# Description

This project involves a temperature measurement system using the LM75 temperature sensor. The system reads temperature data and displays it on an LCD. It involves managing SPI and I2C communication protocols.

## Libraries Used in this Project

### LM75 Library

This library handles interaction with the LM75 temperature sensor.

### LCD Library

This library manages the LCD display to show temperature data.

# File Index

## File List

Here is a list of all files with brief descriptions:

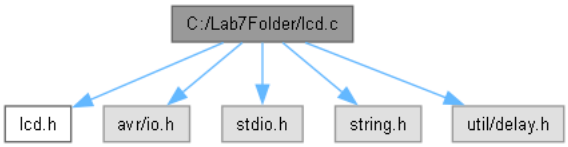
# File Documentation

## C:/Lab7Folder/lcd.c File Reference

LCD display management for the temperature measurement system.

```
#include "lcd.h"
#include <avr/io.h>
#include <stdio.h>
#include <string.h>
#include <util/delay.h>
```

Include dependency graph for lcd.c:




### Macros

- #define **F_CPU** 4000000UL

### Functions

- void **init_spi0_SerLCD** (void)
  *Initializes the SPI interface for LCD communication.*

- void **write_spi0_SerLCD** (unsigned char data)
  *Sends a byte of data to the LCD via SPI.*

- void **select_SS** (void)
  *Selects the LCD as the SPI slave device.*

- void **deselect_SS** (void)
  *Deselects the LCD as the SPI slave device.*

- void **update_SerLCD** (void)
  *Updates the content displayed on the LCD.*

- void **clear_display_buffs** (void)
  *Clears the display buffers.*

### Variables

- char **dsp_buff1** [21]
- char **dsp_buff2** [21]
- char **dsp_buff3** [21]
- char **dsp_buff4** [21]

---

## Detailed Description

LCD display management for the temperature measurement system.

This file contains all the functions necessary for initializing and managing the LCD display via SPI communication, including sending data to the display, clearing display buffers, and updating the display with new information.

**Author**

Naafiul Hossain

**Date**

2025-04-02

---

## Macro Definition Documentation

### #define F_CPU 4000000UL

---

## Function Documentation

### void clear_display_buffs (void )

Clears the display buffers.

Fills the display buffer arrays with spaces to clear previous content, and sets the end character to null to properly terminate the string for display.

```
clear_display_buffs(); // Clear the display buffers before writing new content
```

Here is the caller graph for this function:


### void deselect_SS (void )

Deselects the LCD as the SPI slave device.

Deactivates the slave select (SS) line specific to the LCD to end SPI communication.

```
deselect_SS(); // Deactivate the LCD SS line after sending data
```

Here is the caller graph for this function:


### void init_spi0_SerLCD (void )

Initializes the SPI interface for LCD communication.

Sets up the SPI0 hardware module for communication with the LCD using master mode. Configures the direction of SPI pins and initializes SPI control registers.

```
init_spi0_SerLCD();
```

Here is the caller graph for this function:


### void select_SS (void )

Selects the LCD as the SPI slave device.

Activates the slave select (SS) line specific to the LCD to initiate SPI communication.

```
select_SS(); // Activate the LCD SS line before sending data
```

Here is the caller graph for this function:

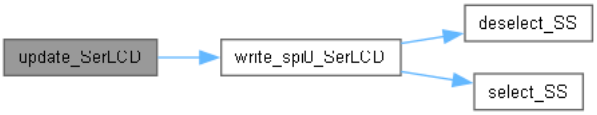

### void update_SerLCD (void )

Updates the content displayed on the LCD.

Sends the contents of display buffers to the LCD via SPI, updating each line of the display. This function should be called whenever the display data needs to be refreshed.

```
clear_display_buffs();   // Clear the display buffers
sprintf(dsp_buff1, "Temperature: %dC", temp); // Prepare line 1
sprintf(dsp_buff2, "Status: %s", status);     // Prepare line 2
update_SerLCD();         // Send the updated buffer to the LCD
```

Here is the call graph for this function:




Here is the caller graph for this function:


### void write_spi0_SerLCD (unsigned char data)

Sends a byte of data to the LCD via SPI.

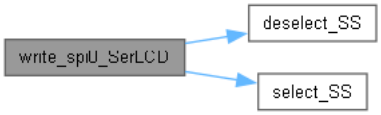
This function transmits a single byte to the LCD using SPI communication, ensuring the slave select line is appropriately managed before and after the transmission.

**Parameters**

| | |
|---|---|
| *data* | The data byte to be sent to the LCD. |

```
write_spi0_SerLCD('H'); // Send character 'H' to the LCD
```

Here is the call graph for this function:




Here is the caller graph for this function:

## Variable Documentation

**char dsp_buff1[21]**

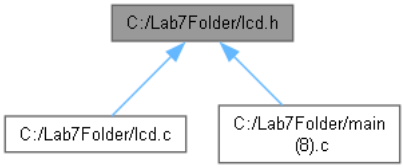
**char dsp_buff2[21]**

**char dsp_buff3[21]**

**char dsp_buff4[21]**

# C:/Lab7Folder/lcd.h File Reference

This graph shows which files directly or indirectly include this file:




## Functions

- void **init_spi0_SerLCD** (void)
  *Initializes the SPI interface for LCD communication.*

- void **write_spi0_SerLCD** (unsigned char data)
  *Sends a byte of data to the LCD via SPI.*

- void **update_SerLCD** (void)
  *Updates the content displayed on the LCD.*

- void **clear_display_buffs** (void)
  *Clears the display buffers.*

- void **select_SS** (void)
  *Selects the LCD as the SPI slave device.*

- void **deselect_SS** (void)
  *Deselects the LCD as the SPI slave device.*

---

## Function Documentation

### void clear_display_buffs (void )

Clears the display buffers.

Fills the display buffer arrays with spaces to clear previous content, and sets the end character to null to properly terminate the string for display.

```
clear_display_buffs(); // Clear the display buffers before writing new content
```

Here is the caller graph for this function:


### void deselect_SS (void )

Deselects the LCD as the SPI slave device.

Deactivates the slave select (SS) line specific to the LCD to end SPI communication.

```
deselect_SS(); // Deactivate the LCD SS line after sending data
```

Here is the caller graph for this function:


### void init_spi0_SerLCD (void )

Initializes the SPI interface for LCD communication.

Sets up the SPI0 hardware module for communication with the LCD using master mode. Configures the direction of SPI pins and initializes SPI control registers.

```
init_spi0_SerLCD();
```

Here is the caller graph for this function:


### void select_SS (void )

Selects the LCD as the SPI slave device.

Activates the slave select (SS) line specific to the LCD to initiate SPI communication.

```
select_SS(); // Activate the LCD SS line before sending data
```

Here is the caller graph for this function:

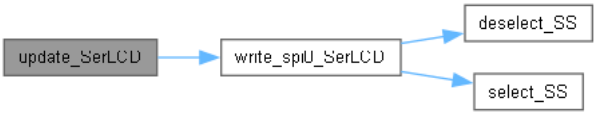

### void update_SerLCD (void )

Updates the content displayed on the LCD.

Sends the contents of display buffers to the LCD via SPI, updating each line of the display. This function should be called whenever the display data needs to be refreshed.

```
clear_display_buffs();   // Clear the display buffers
sprintf(dsp_buff1, "Temperature: %dC", temp); // Prepare line 1
sprintf(dsp_buff2, "Status: %s", status);     // Prepare line 2
update_SerLCD();         // Send the updated buffer to the LCD
```

Here is the call graph for this function:




Here is the caller graph for this function:


### void write_spi0_SerLCD (unsigned char data)

Sends a byte of data to the LCD via SPI.

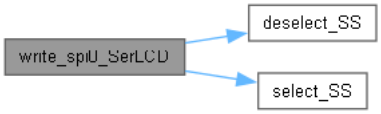
This function transmits a single byte to the LCD using SPI communication, ensuring the slave select line is appropriately managed before and after the transmission.

**Parameters**

| | |
|---|---|
| *data* | The data byte to be sent to the LCD. |

```
write_spi0_SerLCD('H'); // Send character 'H' to the LCD
```

Here is the call graph for this function:




Here is the caller graph for this function:

## lcd.h

Go to the documentation of this file.

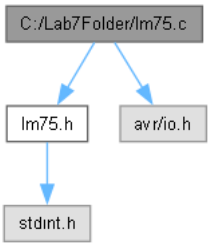
```
1 /*
2  * lcd.h
3  *
4  * Created: 4/2/2025 1:20:53 AM
5  *  Author: Naafiul Hossain
6  */
7 #ifndef LCD H
8 #define LCD_H
9
10 void init_spi0_SerLCD(void);
11 void write_spi0_SerLCD(unsigned char data);
12 void update_SerLCD(void);
13 void clear display buffs(void);
14 void select_SS(void);  // Add this line
15 void deselect_SS(void);  // Add this line
16
17 #endif // LCD_H
```

# C:/Lab7Folder/lm75.c File Reference

Functions to interact with the LM75 temperature sensor via I2C.

```
#include "lm75.h"
#include <avr/io.h>
```

Include dependency graph for lm75.c:




## Functions

- void **TWI0_LM75_init** (void)
  *Initializes the I2C interface for communication with the LM75 sensor.*

- uint16_t **TWI0_LM75_read** (uint8_t saddr)
  *Reads the temperature data from the LM75 sensor.*

- int **TWI0_LM75_write** (uint8_t saddr, uint8_t raddr, uint8_t data)
  *Writes a data byte to a specific register of the LM75 sensor.*

---

## Detailed Description

Functions to interact with the LM75 temperature sensor via I2C.

This file contains functions to initialize the I2C communication for the LM75 sensor, read temperature data from it, and write configurations to it.

**Author**

Naafiul Hossain

**Date**

2025-04-02

---

## Function Documentation

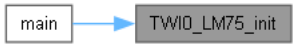
### void TWI0_LM75_init (void )

Initializes the I2C interface for communication with the LM75 sensor.

Sets the master baud rate and control register to prepare the I2C interface for communication. It also forces the I2C bus state to idle to ensure a clean start.

```
TWI0_LM75_init(); // Initialize the I2C bus for LM75 communication
```

Here is the caller graph for this function:




### uint16_t TWI0_LM75_read (uint8_t saddr)

Reads the temperature data from the LM75 sensor.

Sends the read command to the LM75 sensor and reads back two bytes of temperature data. It manages the I2C bus state throughout the operation to ensure proper reception of data.

**Parameters**

| | |
|---|---|
| *saddr* | The slave address of the LM75 sensor. |

**Returns**

The 16-bit raw temperature data read from the sensor.

```
uint16_t temperature = TWI0_LM75_read(LM75_ADDR);
```

Here is the caller graph for this function:


### int TWI0_LM75_write (uint8_t saddr, uint8_t raddr, uint8_t data)

Writes a data byte to a specific register of the LM75 sensor.

This function is used to configure the LM75 sensor by writing to its registers. It handles the entire write operation including sending the slave address, register address, and the data byte, followed by issuing a stop condition.

**Parameters**

| | |
|---|---|
| *saddr* | The slave address of the LM75 sensor. |
| *raddr* | The register address to write to. |
| *data* | The data byte to write to the register. |

**Returns**

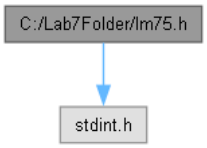
Always returns 0 indicating success.

```
TWI0_LM75_write(LM75_ADDR, LM75_CONFIG_REGISTER, new_config_value);
```

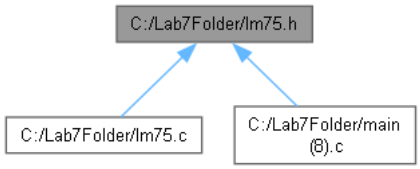
# C:/Lab7Folder/lm75.h File Reference

```
#include <stdint.h>
```

Include dependency graph for lm75.h:




This graph shows which files directly or indirectly include this file:




## Functions

- void **TWI0_LM75_init** (void)
  *Initializes the I2C interface for communication with the LM75 sensor.*

- uint16_t **TWI0_LM75_read** (uint8_t saddr)
  *Reads the temperature data from the LM75 sensor.*

- int **TWI0_LM75_write** (uint8_t saddr, uint8_t raddr, uint8_t data)
  *Writes a data byte to a specific register of the LM75 sensor.*

---

## Function Documentation

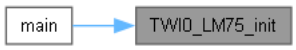
### void TWI0_LM75_init (void )

Initializes the I2C interface for communication with the LM75 sensor.

Sets the master baud rate and control register to prepare the I2C interface for communication. It also forces the I2C bus state to idle to ensure a clean start.

```
TWI0_LM75_init(); // Initialize the I2C bus for LM75 communication
```

Here is the caller graph for this function:




### uint16_t TWI0_LM75_read (uint8_t saddr)

Reads the temperature data from the LM75 sensor.

Sends the read command to the LM75 sensor and reads back two bytes of temperature data. It manages the I2C bus state throughout the operation to ensure proper reception of data.

**Parameters**

| | |
|---|---|
| *saddr* | The slave address of the LM75 sensor. |

**Returns**

The 16-bit raw temperature data read from the sensor.

```
uint16_t temperature = TWI0_LM75_read(LM75_ADDR);
```

Here is the caller graph for this function:


### int TWI0_LM75_write (uint8_t saddr, uint8_t raddr, uint8_t data)

Writes a data byte to a specific register of the LM75 sensor.

This function is used to configure the LM75 sensor by writing to its registers. It handles the entire write operation including sending the slave address, register address, and the data byte, followed by issuing a stop condition.

**Parameters**

| | |
|---|---|
| *saddr* | The slave address of the LM75 sensor. |
| *raddr* | The register address to write to. |
| *data* | The data byte to write to the register. |

**Returns**

Always returns 0 indicating success.

```
TWI0_LM75_write(LM75_ADDR, LM75_CONFIG_REGISTER, new_config_value);
```

## lm75.h

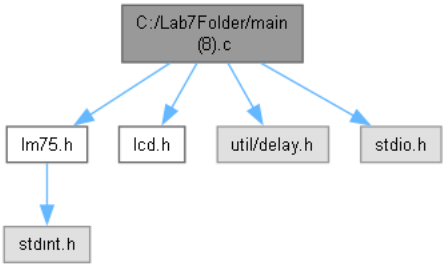
Go to the documentation of this file.

```
1 /*
2  * lm75.h
3  *
4  * Created: 4/2/2025 12:47:55 AM
5  *  Author: Naafiul Hossain
6  */
7
8
9 #ifndef LM75_H
10 #define LM75_H
11
12 #include <stdint.h>
13
14 void TWI0_LM75_init(void);
15 uint16_t TWI0_LM75_read(uint8_t saddr);
16 int TWI0_LM75_write(uint8_t saddr, uint8_t raddr, uint8_t data);
17
18 #endif
```

# C:/Lab7Folder/main (8).c File Reference

```
#include "lm75.h"
#include "lcd.h"
#include <util/delay.h>
#include <stdio.h>
```

Include dependency graph for main (8).c:




## Macros

- #define **F_CPU** 4000000UL
- #define **LM75_ADDR** 0x48

## Functions

- int **main** (void)

---

## Macro Definition Documentation

### #define F_CPU 4000000UL

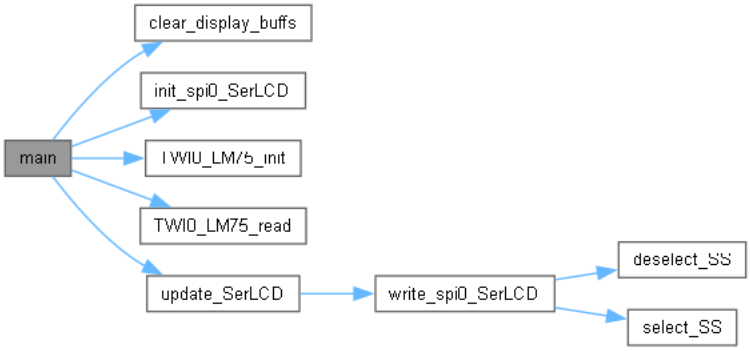
### #define LM75_ADDR 0x48

---

## Function Documentation

### int main (void )

Here is the call graph for this function:



# Index

INDEX

```c

/**
 * @file main.c
 * @author Naafiul Hossain
 * @date 2025-03-30
 * @brief Main file for the temperature measurement system.
 *
 * @mainpage Description
 * This project involves a temperature measurement system using the LM75 temperature sensor.
 * The system reads temperature data and displays it on an LCD. It involves managing SPI and I2C
 * communication protocols.
 *
 * @section library_sec Libraries Used in this Project
 * @subsection library1 LM75 Library
 * This library handles interaction with the LM75 temperature sensor.
 *
 * @subsection library2 LCD Library
 * This library manages the LCD display to show temperature data.
 */


#include "lm75.h"
#include "lcd.h"
#include <util/delay.h>
#include <stdio.h>

#define F_CPU 4000000UL  // Clock frequency for delay functions
#define LM75_ADDR 0x48     // LM75 I2C address

int main(void) {
    init_spi0_SerLCD();
    TWI0_LM75_init();
    clear_display_buffs();

    while (1) {
        uint16_t LM75_temp_reg = TWI0_LM75_read(LM75_ADDR);
        int16_t LM75_temp_celsius = LM75_temp_reg >> 7;
        float temperature_celsius = LM75_temp_celsius / 2.0f;
        float temperature_fahrenheit = temperature_celsius * 9.0f / 5.0f + 32.0f;

        // Suppose you want to use temperature_fahrenheit, here's how you might log it:
        // printf("Temperature: %.1f F\n", temperature_fahrenheit);

        // Update LCD display with temperature data
        update_SerLCD();
```

```
46
47          _delay_ms(1000); // Delay for readability
48      }
49      return 0;
50  }
51
52
```

```c
/**
 * @file lm75.h
 * @brief Interface for LM75 temperature sensor operations over TWI0.
 *
 * This header defines the functions necessary for interacting with the LM75
 * temperature sensor using the TWI0 (Two-Wire Interface 0). It includes
 * initialization, reading temperature data, and writing configurations to the sensor.
 *
 * @author Naafiul Hossain
 * @date 2025-04-02
 */

#ifndef LM75_H
#define LM75_H

#include <stdint.h>

/**
 * Initializes the TWI0 interface for LM75 usage.
 */
void TWI0_LM75_init(void);

/**
 * Reads the temperature or configuration from the LM75 sensor.
 * @param saddr The sensor's address.
 * @return The 16-bit value read from the sensor.
 */
uint16_t TWI0_LM75_read(uint8_t saddr);

/**
 * Writes data to a register on the LM75 sensor.
 * @param saddr The sensor's address.
 * @param raddr The register address to write to.
 * @param data The data to write to the register.
 * @return 0 if successful, non-zero error code otherwise.
 */
int TWI0_LM75_write(uint8_t saddr, uint8_t raddr, uint8_t data);

#endif // LM75_H


```

```c
/**
 * @file lm75.c
 * @brief Functions to interact with the LM75 temperature sensor via I2C.
 *
 * This file contains functions to initialize the I2C communication for the LM75 sensor,
 * read temperature data from it, and write configurations to it.
 *
 * @author Naafiul Hossain
 * @date 2025-04-02
 */

#include "lm75.h"
#include <avr/io.h>
/**
 * @brief Initializes the I2C interface for communication with the LM75 sensor.
 *
 * Sets the master baud rate and control register to prepare the I2C interface
 * for communication. It also forces the I2C bus state to idle to ensure a clean start.
 *
 * @code
 * TWI0_LM75_init(); // Initialize the I2C bus for LM75 communication
 * @endcode
 */

void TWI0_LM75_init(void) {
    TWI0_MBAUD = 0x01;
    TWI0_MCTRLA = 0x01;
    TWI0.MSTATUS = 0x01;  // Force bus state to idle
}
/**
 * @brief Reads the temperature data from the LM75 sensor.
 *
 * Sends the read command to the LM75 sensor and reads back two bytes of temperature data.
 * It manages the I2C bus state throughout the operation to ensure proper reception of data.
 *
 * @param saddr The slave address of the LM75 sensor.
 * @return The 16-bit raw temperature data read from the sensor.
 *
 * @code
 * uint16_t temperature = TWI0_LM75_read(LM75_ADDR);
 * @endcode
 */
uint16_t TWI0_LM75_read(uint8_t saddr) {
    uint8_t temp_reg_high, temp_reg_low;
```

```
45      while((TWI0.MSTATUS & 0x03) != 0x01); // Wait until idle
46
47      TWI0.MADDR = ((saddr << 1) | 0x01); // Send slave address and read command
48      while((TWI0.MSTATUS & 0x80) == 0); // Wait for RIF flag, byte received
49      temp_reg_high = TWI0.MDATA; // Clear the RIF flag
50
51      TWI0.MCTRLB = 0x02; // Issue ACK followed by a byte read operation
52      while((TWI0.MSTATUS & 0x80) == 0); // Wait for next byte
53      temp_reg_low = TWI0.MDATA; // Clear the RIF flag
54
55      TWI0.MCTRLB = 0x07; // Issue NACK followed by a stop
56      return (uint16_t)((temp_reg_high << 8) | temp_reg_low);
57  }
58  /**
59   * @brief Writes a data byte to a specific register of the LM75 sensor.
60   *
61   * This function is used to configure the LM75 sensor by writing to its
       registers.
62   * It handles the entire write operation including sending the slave address,
       register address,
63   * and the data byte, followed by issuing a stop condition.
64   *
65   * @param saddr The slave address of the LM75 sensor.
66   * @param raddr The register address to write to.
67   * @param data The data byte to write to the register.
68   * @return Always returns 0 indicating success.
69   *
70   * @code
71   * TWI0_LM75_write(LM75_ADDR, LM75_CONFIG_REGISTER, new_config_value);
72   * @endcode
73   */
74
75  int TWI0_LM75_write(uint8_t saddr, uint8_t raddr, uint8_t data) {
76      while((TWI0.MSTATUS & 0x03) != 0x01); // Wait until idle
77
78      TWI0.MADDR = saddr << 1; // Send address for write
79      while((TWI0.MSTATUS & 0x40) == 0); // Wait until address sent
80
81      TWI0.MDATA = raddr; // Send memory address
82      while((TWI0.MSTATUS & 0x40) == 0); // Wait until memory address sent
83
84      TWI0.MDATA = data; // Send data
85      while((TWI0.MSTATUS & 0x40) == 0); // Wait until data sent
86
87      TWI0.MCTRLB |= 0x03; // Issue a stop
88      return 0;
89  }
90
```

```c
/**
 * @file lcd.h
 * @brief SPI communication interface for SerLCD display management.
 *
 * This header file provides the definitions for the SPI communication functions
 * used to interact with a SerLCD display. It includes functions for initializing
 * the SPI interface, sending data, updating the display, clearing buffers, and
 * managing the slave select line.
 *
 * @author Naafiul Hossain
 * @date 2025-04-02
 */

#ifndef LCD_H
#define LCD_H

/**
 * Initializes the SPI0 interface for SerLCD display.
 */
void init_spi0_SerLCD(void);

/**
 * Sends a byte of data to the SerLCD display over SPI0.
 * @param data The data byte to send.
 */
void write_spi0_SerLCD(unsigned char data);

/**
 * Updates the SerLCD display with new data.
 */
void update_SerLCD(void);

/**
 * Clears the display buffers of the SerLCD.
 */
void clear_display_buffs(void);

/**
 * Selects the Slave Select (SS) line, enabling the SerLCD to listen for SPI data.
 */
void select_SS(void);

/**
 * Deselects the Slave Select (SS) line, disabling the SerLCD from listening for SPI data.
```

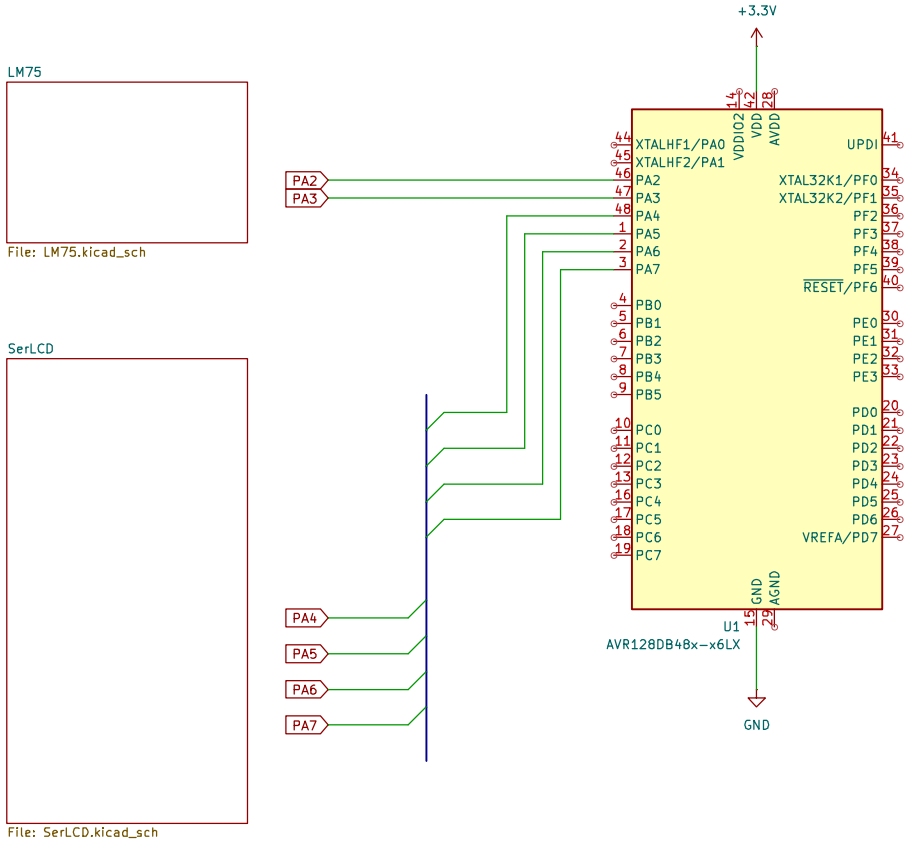
```
45   */
46  void deselect_SS(void);
47
48  #endif // LCD_H
49
```

```c

/**
 * @file lcd.c
 * @brief LCD display management for the temperature measurement system.
 *
 * This file contains all the functions necessary for initializing and managing
 * the LCD display via SPI communication, including sending data to the display,
 * clearing display buffers, and updating the display with new information.
 *
 * @author Naafiul Hossain
 * @date 2025-04-02
 */

#include "lcd.h"
#include <avr/io.h>
#include <stdio.h>
#include <string.h>
#define F_CPU 4000000UL  // Clock frequency for delay functions
#include <util/delay.h>


char dsp_buff1[21]; // Buffer for LCD display line 1 - Global variable, static storage, no linkage
char dsp_buff2[21]; // Buffer for LCD display line 2 - Global variable, static storage, no linkage
char dsp_buff3[21]; // Buffer for LCD display line 3 - Global variable, static storage, no linkage
char dsp_buff4[21]; // Buffer for LCD display line 4 - Global variable, static storage, no linkage


/**
 * @brief Initializes the SPI interface for LCD communication.
 *
 * Sets up the SPI0 hardware module for communication with the LCD using master mode.
 * Configures the direction of SPI pins and initializes SPI control registers.
 *
 * @code
 * init_spi0_SerLCD();
 * @endcode
 */

void init_spi0_SerLCD(void) {
    PORTA.DIRSET = PIN7_bm | PIN6_bm | PIN4_bm; // Set SPI pins as output
    PORTA.DIRCLR = PIN5_bm; // Set MISO pin as input
    SPI0.CTRLA = SPI_MASTER_bm | SPI_PRESC_DIV16_gc | SPI_ENABLE_bm; // Enable
```

```
          SPI, master mode
43      SPI0.CTRLB = SPI_SSD_bm | SPI_MODE_0_gc; // Set SPI mode 0
44  }
45  /**
46   * @brief Sends a byte of data to the LCD via SPI.
47   *
48   * This function transmits a single byte to the LCD using SPI communication,
49   * ensuring the slave select line is appropriately managed before and after
       the transmission.
50   *
51   * @param data The data byte to be sent to the LCD.
52   *
53   * @code
54   * write_spi0_SerLCD('H'); // Send character 'H' to the LCD
55   * @endcode
56   */
57  void write_spi0_SerLCD(unsigned char data) {
58      select_SS();
59      SPI0.DATA = data;
60      while (!(SPI0.INTFLAGS & SPI_IF_bm));
61      deselect_SS();
62  }
63
64  /**
65   * @brief Selects the LCD as the SPI slave device.
66   *
67   * Activates the slave select (SS) line specific to the LCD to initiate SPI
       communication.
68   *
69   * @code
70   * select_SS(); // Activate the LCD SS line before sending data
71   * @endcode
72   */
73  void select_SS(void) {
74      PORTA.OUT &= ~PIN7_bm; // Select slave (active low)
75  }
76  /**
77   * @brief Deselects the LCD as the SPI slave device.
78   *
79   * Deactivates the slave select (SS) line specific to the LCD to end SPI
       communication.
80   *
81   * @code
82   * deselect_SS(); // Deactivate the LCD SS line after sending data
83   * @endcode
84   */
85  void deselect_SS(void) {
86      PORTA.OUT |= PIN7_bm; // Deselect slave
87  }
```

```
 88 /**
 89  * @brief Updates the content displayed on the LCD.
 90  *
 91  * Sends the contents of display buffers to the LCD via SPI, updating each
       line of the display.
 92  * This function should be called whenever the display data needs to be
       refreshed.
 93  *
 94  * @code
 95  * clear_display_buffs();   // Clear the display buffers
 96  * sprintf(dsp_buff1, "Temperature: %dC", temp); // Prepare line 1
 97  * sprintf(dsp_buff2, "Status: %s", status);     // Prepare line 2
 98  * update_SerLCD();         // Send the updated buffer to the LCD
 99  * @endcode
100  */
101
102 void update_SerLCD(void) {
103     write_spi0_SerLCD(0xFE);
104     write_spi0_SerLCD(0x80); // First line
105     for (uint8_t i = 0; i < 20; i++) {
106         write_spi0_SerLCD(dsp_buff1[i]);
107     }
108
109     write_spi0_SerLCD(0xFE);
110     write_spi0_SerLCD(0xC0); // Second line
111     for (uint8_t i = 0; i < 20; i++) {
112         write_spi0_SerLCD(dsp_buff2[i]);
113     }
114
115     write_spi0_SerLCD(0xFE);
116     write_spi0_SerLCD(0x94); // Third line
117     for (uint8_t i = 0; i < 20; i++) {
118         write_spi0_SerLCD(dsp_buff3[i]);
119     }
120
121     write_spi0_SerLCD(0xFE);
122     write_spi0_SerLCD(0xD4); // Fourth line
123     for (uint8_t i = 0; i < 20; i++) {
124         write_spi0_SerLCD(dsp_buff4[i]);
125     }
126 }
127 /**
128  * @brief Clears the display buffers.
129  *
130  * Fills the display buffer arrays with spaces to clear previous content,
131  * and sets the end character to null to properly terminate the string for
       display.
132  *
133  * @code
```

```
134   * clear_display_buffs(); // Clear the display buffers before writing new
        content
135   * @endcode
136   */
137
138  void clear_display_buffs(void) {
139      memset(dsp_buff1, ' ', 20);
140      dsp_buff1[20] = '\0';
141      memset(dsp_buff2, ' ', 20);
142      dsp_buff2[20] = '\0';
143      memset(dsp_buff3, ' ', 20);
144      dsp_buff3[20] = '\0';
145      memset(dsp_buff4, ' ', 20);
146      dsp_buff4[20] = '\0';
147  }
148
```

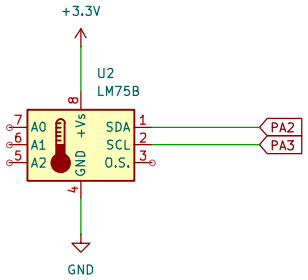



Embedded system involving serial communication between
the LM75 temperature and the AVR128DB48 microcontroller,
as well as between the microcontroller and the SerLCD display.
For the off page connections, netlists were used.

Sheet: /
File: ESE 381_Lab 7_Prelab_Task 1_Muhammad Sharjeel_Naafiul Hossain.kicad_sch

**Title: ESE 381_Lab 7_Prelab_Task 1_Sharjeel_Hossain**

| Size: A4 | Date: 2025-04-02 | **Rev: 1.0** |
|---|---|---|
| KiCad E.D.A. 8.0.8 | | Id: 1/3 |

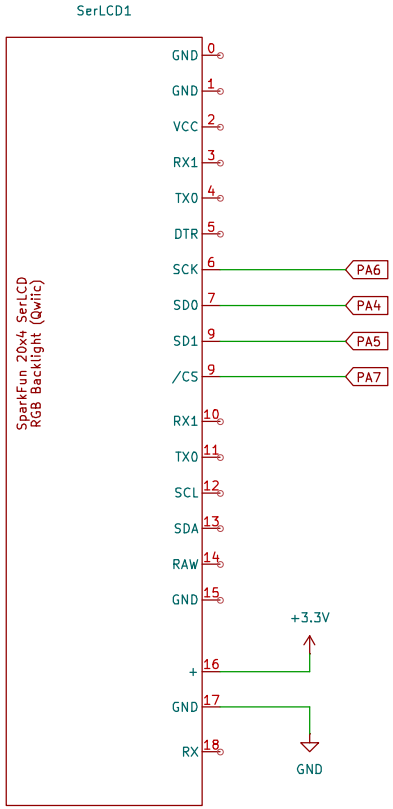



The schematic of the LM75 temperature sensor configured
for I2C communication with the AVR128DB48 microcontroller.

Sheet: /LM75/
File: LM75.kicad_sch

**Title: ESE 381_Lab 7_Prelab_Task 1_Sharjeel_Hossain**

| Size: A4 | Date: 2025-04-02 | **Rev: 1.0** |
|---|---|---|
| KiCad E.D.A. 8.0.8 | | Id: 2/3 |




The schematic of the SerLCD display configured for
SPI communication with the AVR128DB48 microcontroller.

Sheet: /SerLCD/
File: SerLCD.kicad_sch

**Title: ESE 381_Lab 7_Prelab_Task 1_Sharjeel_Hossain**

| Size: A4 | Date: 2025-04-02 | **Rev: 1.0** |
|---|---|---|
| KiCad E.D.A. 8.0.8 | | Id: 3/3 |