Muhammad Sharjeel and Naafiul Hossain

115185427

115107623

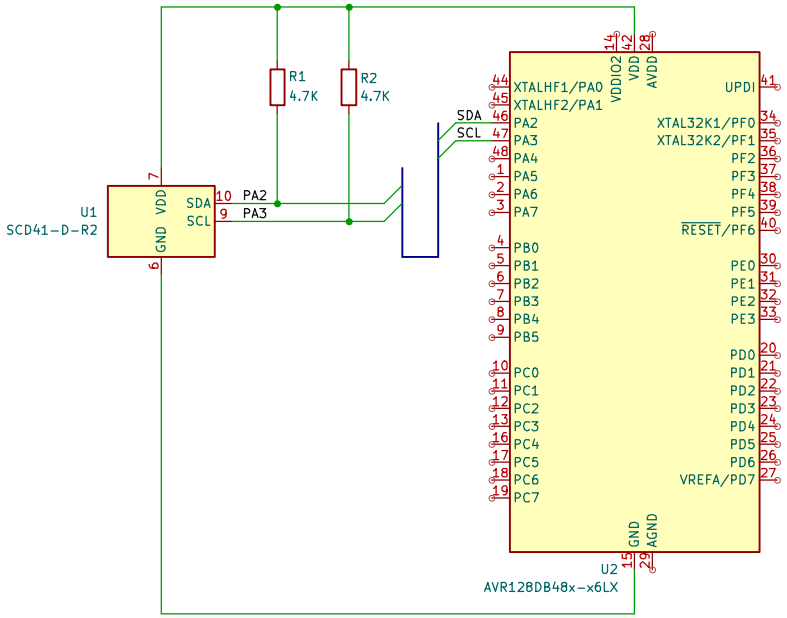
Pre-Lab 10: Air Monitoring System I -

Basic Operation of SCD41 CO2, Humidity, and Temperature Sensor

ESE 381 Section L02

Bench 7

Breadboard: K2



Sheet: /
File: ESE 381_Lab 10_Prelab_Task 1_Muhammad Sharjeel_Naafiul Hossain.kicad_sch

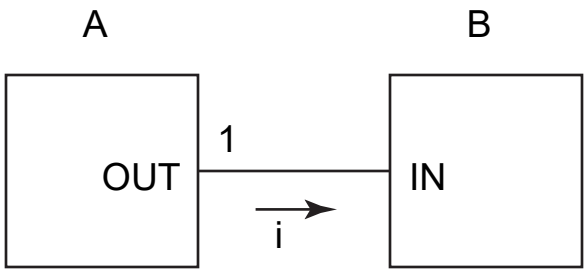
Title: ESE 381 Lab 10 Prelab Task 1 SecL02 Sharjeel Hossain

Size: A4	Date: 2025-04-23	**Rev: 1.0**
KiCad E.D.A. 8.0.8		Id: 1/1

Interface Checklist (revised 10/09/17)
ESE280 Ken Short

Naafiul Hossain-Group 16

A=AVR128DB48
B=SDC41



VOH min (LA) > VIH min (CB)
→ 3.29 V [3.3 − 10k × 1 µA]
→ 2.145 V [0.65 × VDD

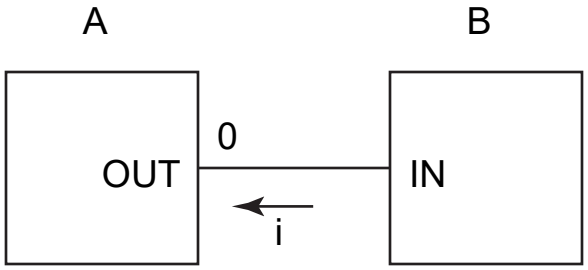
IOH min (A) > IIH max (B)
→ (3.3 − 2.145) / 10k
→ Assume 1 µA

$V_{OHmin(A)} > V_{IHmin(B)}$

3.29V > 2.145V

$I_{OHmax(A)} > I_{IHmax(B)}$

115.5uA > 1nA



VOL max (LA) < VIL max (CB)
→ 0.99 V
→ 0.99 V

IOL max (A) > IIL max (B)
→ 6 mA
→ 1 mA + (3.3 / 10k)

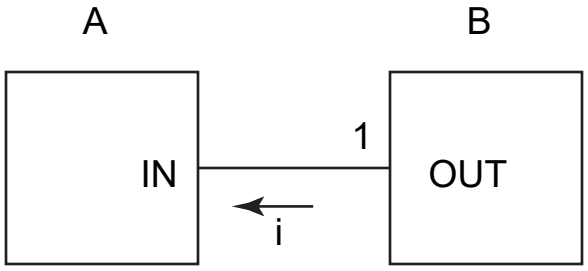
$V_{OLmax(A)} < V_{ILmax(B)}$

0.99v < 0.99v

$I_{OLmax(A)} > I_{ILmax(B)}$

6mA > 1uA+0.33mA

[VOL max (I2C) = 0.33 × VDD @ 6mA]]



Calculation:

VIH(min)(A) = 2.31 V

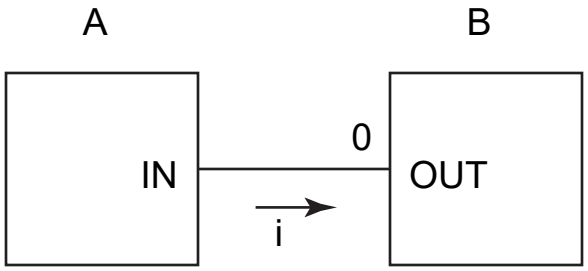
VOH(min)(B) = 3.29 V
2.31 V < 3.29 V → Satisfies logic high voltage level requirement

$V_{IHmin(A)} < V_{OHmin(B)}$

2.31V < 3.29v

$I_{IHmax(A)} < I_{OHmax(B)}$

1uA < 99uA



IIL(max)(A) = 3 mA + 5 nA ≈ 3.000005 mA

IOL(max)(B) = 3 mA

$V_{ILmax(A)} > V_{OLmax(B)}$

0.99V > 0.66V

$I_{ILmax(A)} < I_{OLmax(B)}$

0.33mA+5nA < 3mA

```c
/*
 * lab10_task2_scd41_read.c
 *
 * Task 2 – Read raw CO?, temperature, and humidity values from the SCD41 sensor
 * using TWI0 (I²C) on the AVR128DB48 and display them in Studio 7 watch window.
 *
 * Author: Naafiul Hossain
 * Date: 2025-04-21
 */

#include <avr/io.h>
#define F_CPU 4000000UL
#include <util/delay.h>
#include <string.h>
#include <stdio.h>

// I²C address and command constants from the SCD41 datasheet
#define SCD41_I2C_ADDR      0x62
#define CMD_START_MEAS      0x21B1
#define CMD_DATA_READY      0xE4B8
#define CMD_READ_MEAS       0xEC05

/**
 * @brief Initializes TWI0 (I²C) on PA2 (SDA) and PA3 (SCL).
 */
void init_twi0(void) {
    PORTMUX.TWIROUTEA = PORTMUX_TWI0_ALT1_gc;     // Route TWI0 to PA2/PA3
    TWI0.MBAUD = 0x01;                             // 400kHz I²C baud for F_CPU = 4MHz
    TWI0.MCTRLA = TWI_ENABLE_bm;                   // Enable TWI master
    TWI0.MSTATUS = TWI_BUSSTATE_IDLE_gc;           // Set bus state to idle
    PORTA.PIN2CTRL |=PORT_PULLUPEN_bm;
    PORTA.PIN3CTRL |=PORT_PULLUPEN_bm;
}

/**
 * @brief Sends a 16-bit command to the SCD41 sensor via TWI0.
 * @param cmd 16-bit command to send.
 */
void scd41_send_command(uint16_t cmd) {
    uint8_t cmd_high = (cmd >> 8);
    uint8_t cmd_low = (cmd & 0xFF);

    TWI0.MADDR = (SCD41_I2C_ADDR << 1);  // I²C write address
    while (!(TWI0.MSTATUS & TWI_WIF_bm));
    TWI0.MDATA = cmd_high;
    while (!(TWI0.MSTATUS & TWI_WIF_bm));
```

```
47      TWI0.MDATA = cmd_low;
48      while (!(TWI0.MSTATUS & TWI_WIF_bm));
49      TWI0.MCTRLB = TWI_MCMD_STOP_gc; //stop
50  }
51
52  /**
53   * @brief Polls the sensor to check if new data is ready.
54   * @return 1 if data is ready, 0 otherwise.
55   */
56  uint8_t scd41_is_data_ready(void) {
57      scd41_send_command(CMD_DATA_READY);
58      _delay_ms(2);  // Wait for sensor to process
59
60      TWI0.MADDR = (SCD41_I2C_ADDR << 1) | 0x01;  // I²C read address
61      while (!(TWI0.MSTATUS & TWI_RIF_bm));
62      uint8_t status_high = TWI0.MDATA;
63      while (!(TWI0.MSTATUS & TWI_RIF_bm));
64      uint8_t status_low = TWI0.MDATA;
65      while (!(TWI0.MSTATUS & TWI_RIF_bm));
66      uint8_t crc = TWI0.MDATA;  // Not used in this task
67
68      TWI0.MCTRLB = TWI_MCMD_STOP_gc;
69
70      uint16_t status = ((uint16_t)status_high << 8) | status_low;
71      return (status & 0x07FF);  // Check readiness flag in lower 11 bits
72  }
73
74  /**
75   * @brief Reads CO2, temperature, and humidity values from the SCD41.
76   * @param[out] co2  CO? value in ppm
77   * @param[out] temp Raw temperature value (convert later)
78   * @param[out] hum  Raw humidity value (convert later)
79   */
80  void scd41_read_raw_measurements(uint16_t *co2, uint16_t *temp, uint16_t *hum)↵
       {
81      scd41_send_command(CMD_READ_MEAS);
82      _delay_ms(1);  // Allow sensor to prepare data
83
84      TWI0.MADDR = (SCD41_I2C_ADDR << 1) | 0x01;
85
86      // Read CO2
87      while (!(TWI0.MSTATUS & TWI_RIF_bm));
88      uint8_t co2_msb = TWI0.MDATA;
89      while (!(TWI0.MSTATUS & TWI_RIF_bm));
90      uint8_t co2_lsb = TWI0.MDATA;
91      TWI0.MDATA;  // CRC (not used)
92
93      // Read temperature
94      while (!(TWI0.MSTATUS & TWI_RIF_bm));
```

```
95      uint8_t temp_msb = TWI0.MDATA;
96      while (!(TWI0.MSTATUS & TWI_RIF_bm));
97      uint8_t temp_lsb = TWI0.MDATA;
98      TWI0.MDATA;  // CRC (not used)
99
100     // Read humidity
101     while (!(TWI0.MSTATUS & TWI_RIF_bm));
102     uint8_t hum_msb = TWI0.MDATA;
103     while (!(TWI0.MSTATUS & TWI_RIF_bm));
104     uint8_t hum_lsb = TWI0.MDATA;
105     TWI0.MDATA;  // CRC (not used)
106
107     TWI0.MCTRLB = TWI_MCMD_STOP_gc;
108
109     *co2  = ((uint16_t)co2_msb << 8) | co2_lsb;
110     *temp = ((uint16_t)temp_msb << 8) | temp_lsb;
111     *hum  = ((uint16_t)hum_msb << 8) | hum_lsb;
112 }
113
114 /**
115  * @brief Main loop – initializes TWI and SCD41, then polls and reads data.
116  */
117 int main(void) {
118     uint16_t co2_ppm = 0;
119     uint16_t temp_raw = 0;
120     uint16_t hum_raw = 0;
121
122     init_twi0();
123     _delay_ms(100);  // Let everything stabilize
124
125     scd41_send_command(CMD_START_MEAS);
126     _delay_ms(5000);  // Allow sensor to warm up
127
128     while (1) {
129         if (scd41_is_data_ready()) {
130             scd41_read_raw_measurements(&co2_ppm, &temp_raw, &hum_raw);
131
132             // You can watch co2_ppm, temp_raw, and hum_raw in Studio 7's
                watch window
133         }
134         _delay_ms(500);
135     }
136 }
137
```

```c
/**
 * @file main.c
 * @brief Task 3 – Display CO2, temperature, and humidity on SPI-based LCD using SCD41 sensor
 *
 * Author: Naafiul Hossain
 */
#include "scd41.h"
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include "lcd.h"

#define F_CPU 4000000UL
#include <util/delay.h>

#include <avr/io.h>
#include "scd41.h"

int main(void) {
    uint16_t co2_ppm, temp_raw, hum_raw;

    init_twi0();
    scd41_start_periodic_measurement();

    while (1) {
        if (scd41_is_data_ready()) {
            scd41_read_raw_measurements(&co2_ppm, &temp_raw, &hum_raw);
            //  note please checks values in a watch window
        }
        _delay_ms(1000);
    }
}


```

```c
/*
 * scd41.c
 *
 * Implements SCD41 communication over TWI0 (I²C) for Lab 10 Task 3.
 * Handles setup, data polling, and measurement retrieval.
 *
 * Author: Naafiul Hossain
 */

#include "scd41.h"
#include <util/delay.h>

// I²C address and command constants
#define SCD41_I2C_ADDR      0x62
#define CMD_START_MEAS      0x21B1
#define CMD_DATA_READY      0xE4B8
#define CMD_READ_MEAS       0xEC05

/**
 * @brief Initializes TWI0 on alternate pins PA2 (SDA) and PA3 (SCL) for I²C communication.
 *        Sets baud rate to 400kHz assuming F_CPU is 4 MHz.
 */
void init_twi0(void) {
    PORTMUX.TWIROUTEA = PORTMUX_TWI0_ALT1_gc;     // Route TWI0 to PA2/PA3
    TWI0.MBAUD = 0x01;                             // 400kHz I²C baud for F_CPU = 4MHz
    TWI0.MCTRLA = TWI_ENABLE_bm;                   // Enable TWI master
    TWI0.MSTATUS = TWI_BUSSTATE_IDLE_gc;           // Set bus state to idle
    PORTA.PIN2CTRL |=PORT_PULLUPEN_bm;
    PORTA.PIN3CTRL |=PORT_PULLUPEN_bm;
}


/**
 * @brief Sends a 16-bit command to the SCD41 sensor over I²C.
 *
 * @param cmd The 16-bit command to be sent.
 */
static void scd41_send_command(uint16_t cmd) {
    uint8_t cmd_high = (cmd >> 8);
    uint8_t cmd_low = (cmd & 0xFF);

    TWI0.MADDR = (SCD41_I2C_ADDR << 1);  // Write mode
    while (!(TWI0.MSTATUS & TWI_WIF_bm));
    TWI0.MDATA = cmd_high;
    while (!(TWI0.MSTATUS & TWI_WIF_bm));
    TWI0.MDATA = cmd_low;
    while (!(TWI0.MSTATUS & TWI_WIF_bm));
```

```
48      TWI0.MCTRLB = TWI_MCMD_STOP_gc;
49  }
50
51  /**
52   * @brief Starts periodic CO?, temperature, and humidity measurements.
53   *        Includes a 5-second delay for sensor warm-up.
54   */
55  void scd41_start_periodic_measurement(void) {
56      scd41_send_command(CMD_START_MEAS);
57      _delay_ms(5000);  // Warm-up time
58  }
59
60  /**
61   * @brief Checks if new measurement data is ready from the SCD41 sensor.
62   *
63   * @return uint8_t Returns non-zero if data is ready, 0 otherwise.
64   */
65  uint8_t scd41_is_data_ready(void) {
66      scd41_send_command(CMD_DATA_READY);
67      _delay_ms(2);
68
69      TWI0.MADDR = (SCD41_I2C_ADDR << 1) | 0x01;  // Read mode
70      while (!(TWI0.MSTATUS & TWI_RIF_bm));
71      uint8_t status_high = TWI0.MDATA;
72      while (!(TWI0.MSTATUS & TWI_RIF_bm));
73      uint8_t status_low = TWI0.MDATA;
74      while (!(TWI0.MSTATUS & TWI_RIF_bm));
75      TWI0.MDATA; // CRC – ignored for now
76
77      TWI0.MCTRLB = TWI_MCMD_STOP_gc;
78
79      uint16_t status = ((uint16_t)status_high << 8) | status_low;
80      return (status & 0x07FF);  // Check ready bit
81  }
82
83  /**
84   * @brief Reads raw CO?, temperature, and humidity values from the SCD41
      sensor.
85   *        Each measurement is 16-bit, received in MSB+LSB format.
86   *
87   * @param co2 Pointer to variable to store CO? ppm.
88   * @param temp Pointer to variable to store temperature raw value.
89   * @param hum Pointer to variable to store humidity raw value.
90   */
91  void scd41_read_raw_measurements(uint16_t *co2, uint16_t *temp, uint16_t *hum)
      {
92      scd41_send_command(CMD_READ_MEAS);
93      _delay_ms(1);
94
```

```
 95      TWI0.MADDR = (SCD41_I2C_ADDR << 1) | 0x01;
 96
 97      // CO2
 98      while (!(TWI0.MSTATUS & TWI_RIF_bm));
 99      uint8_t co2_msb = TWI0.MDATA;
100      while (!(TWI0.MSTATUS & TWI_RIF_bm));
101      uint8_t co2_lsb = TWI0.MDATA;
102      TWI0.MDATA; // CRC
103
104      // Temperature
105      while (!(TWI0.MSTATUS & TWI_RIF_bm));
106      uint8_t temp_msb = TWI0.MDATA;
107      while (!(TWI0.MSTATUS & TWI_RIF_bm));
108      uint8_t temp_lsb = TWI0.MDATA;
109      TWI0.MDATA; // CRC
110
111      // Humidity
112      while (!(TWI0.MSTATUS & TWI_RIF_bm));
113      uint8_t hum_msb = TWI0.MDATA;
114      while (!(TWI0.MSTATUS & TWI_RIF_bm));
115      uint8_t hum_lsb = TWI0.MDATA;
116      TWI0.MDATA; // CRC
117
118      TWI0.MCTRLB = TWI_MCMD_STOP_gc;
119
120      *co2 = ((uint16_t)co2_msb << 8) | co2_lsb;
121      *temp = ((uint16_t)temp_msb << 8) | temp_lsb;
122      *hum = ((uint16_t)hum_msb << 8) | hum_lsb;
123  }
124
```

```c
/*
 * scd41.h
 *
 * Interface for communicating with the SCD41 CO?, temperature, and humidity sensor via TWI0.
 * Used in Lab 10 Task 3 for a modular multifile implementation.
 *
 * Author: Naafiul Hossain
 */

#ifndef SCD41_H
#define SCD41_H

#include <avr/io.h>
#include <stdint.h>

// === Public API ===
void init_twi0(void);
void scd41_start_periodic_measurement(void);
uint8_t scd41_is_data_ready(void);
void scd41_read_raw_measurements(uint16_t *co2, uint16_t *temp, uint16_t *hum);

#endif // SCD41_H

```

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

#include "lcd.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  }
```

```c
/**
 * @brief 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.
 *
 * @code
 * 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
 * @endcode
 */

void update_SerLCD(void) {
    write_spi0_SerLCD(0xFE);
    write_spi0_SerLCD(0x80); // First line
    for (uint8_t i = 0; i < 20; i++) {
        write_spi0_SerLCD(dsp_buff1[i]);
    }

    write_spi0_SerLCD(0xFE);
    write_spi0_SerLCD(0xC0); // Second line
    for (uint8_t i = 0; i < 20; i++) {
        write_spi0_SerLCD(dsp_buff2[i]);
    }

    write_spi0_SerLCD(0xFE);
    write_spi0_SerLCD(0x94); // Third line
    for (uint8_t i = 0; i < 20; i++) {
        write_spi0_SerLCD(dsp_buff3[i]);
    }

    write_spi0_SerLCD(0xFE);
    write_spi0_SerLCD(0xD4); // Fourth line
    for (uint8_t i = 0; i < 20; i++) {
        write_spi0_SerLCD(dsp_buff4[i]);
    }
}
/**
 * @brief 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.
 *
 * @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  }
```

```c
/*
 * lcd.h
 *
 * Created: 4/2/2025 1:20:53 AM
 *  Author: Naafiul Hossain
 */
#ifndef LCD_H
#define LCD_H

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


void init_spi0_SerLCD(void);
void write_spi0_SerLCD(unsigned char data);
void update_SerLCD(void);
void clear_display_buffs(void);
void select_SS(void);  // Add this line
void deselect_SS(void);  // Add this line

#endif // LCD_H
```

lab10_task3_display_sensor_data_file

AUTHOR
Version 1.1

Table of Contents

Table of contents

File Index

File List

Here is a list of all files with brief descriptions:

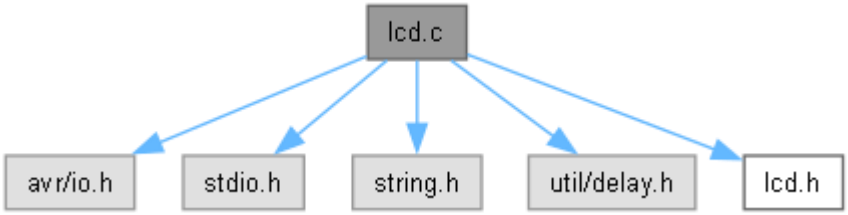
File Documentation

lcd.c File Reference

LCD display management for the temperature measurement system.

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

Include dependency graph for lcd.c:



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

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
```

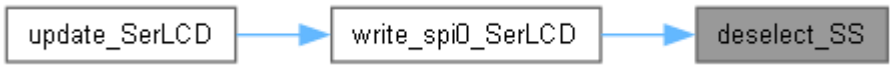
void deselect_SS (void)

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();
```

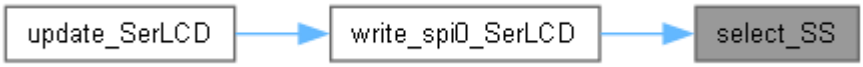
void select_SS (void)

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:



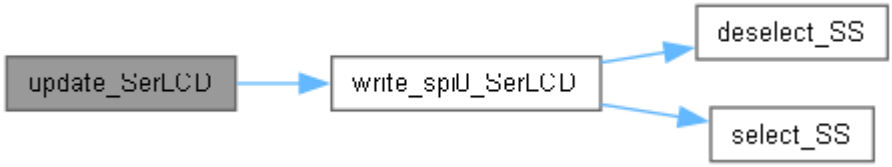
void update_SerLCD (void)

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:



void write_spi0_SerLCD (unsigned char data)

Sends a byte of data to the LCD via SPI.

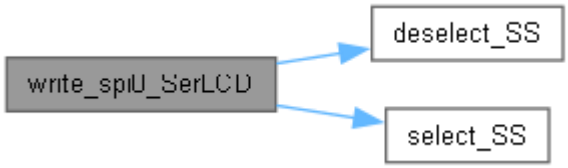
This function transmits a single byte to the LCD using SPI communication, ensuring the slave select line is appropriately managed before and after the transmission.

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]

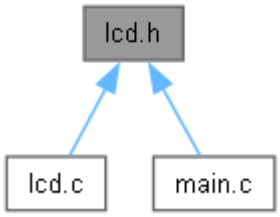
char dsp_buff2[21]

char dsp_buff3[21]

char dsp_buff4[21]

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.

Variables

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

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
```

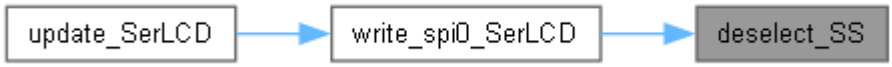
void deselect_SS (void)

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();
```

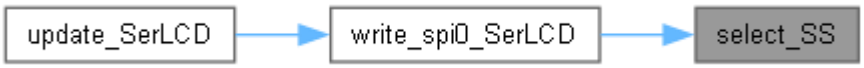
void select_SS (void)

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:



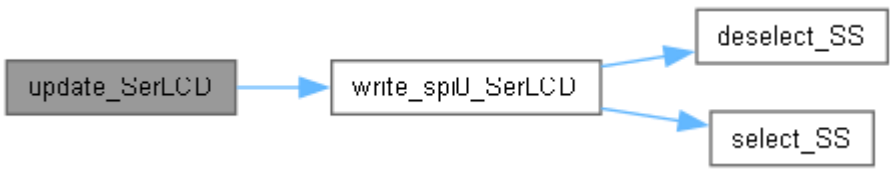
void update_SerLCD (void)

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:



void write_spi0_SerLCD (unsigned char data)

Sends a byte of data to the LCD via SPI.

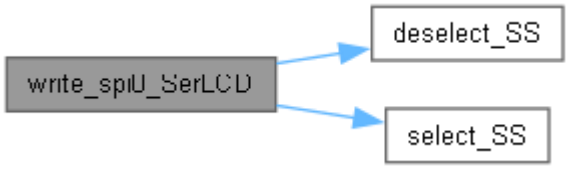
This function transmits a single byte to the LCD using SPI communication, ensuring the slave select line is appropriately managed before and after the transmission.

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] [extern]

char dsp_buff2[21] [extern]

char dsp_buff3[21] [extern]

char dsp_buff4[21] [extern]

lcd.h

Go to the documentation of this file.

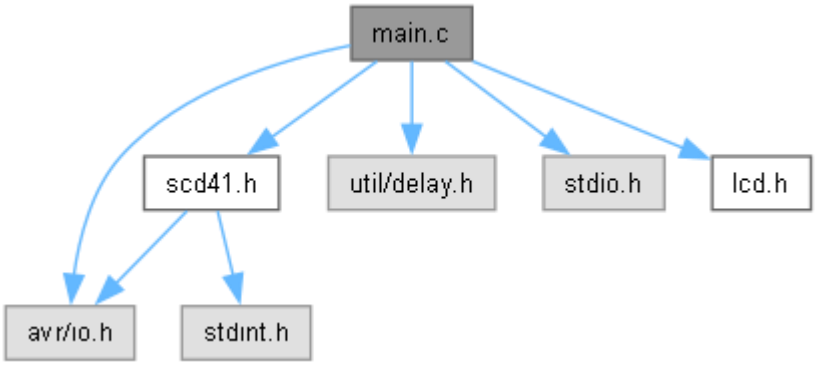
```
1 /*
2  * lcd.h
3  *
4  * Created: 4/2/2025 1:20:53 AM
5  *  Author: Naafiul Hossain
6  */
7 #ifndef LCD H
8 #define LCD_H
9
10  extern char dsp_buff1[21]; // Buffer for LCD display line 1 - Global variable,
static storage, no linkage
11  extern char dsp_buff2[21]; // Buffer for LCD display line 2 - Global variable,
static storage, no linkage
12  extern char dsp_buff3[21]; // Buffer for LCD display line 3 - Global variable,
static storage, no linkage
13  extern char dsp_buff4[21]; // Buffer for LCD display line 4 - Global variable,
static storage, no linkage
14
15
16 void init_spi0_SerLCD(void);
17 void write_spi0_SerLCD(unsigned char data);
18 void update_SerLCD(void);
19 void clear_display_buffs(void);
20 void select_SS(void);  // Add this line
21 void deselect_SS(void);  // Add this line
22
23 #endif // LCD_H
```

main.c File Reference

Task 3 – Display CO2, temperature, and humidity on SPI-based LCD using SCD41 sensor.

```
#include "scd41.h"
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include "lcd.h"
```

Include dependency graph for main.c:



Macros

- #define **F_CPU** 4000000UL

Functions

- int **main** (void)

Detailed Description

Task 3 – Display CO2, temperature, and humidity on SPI-based LCD using SCD41 sensor.

Author: Naafiul Hossain

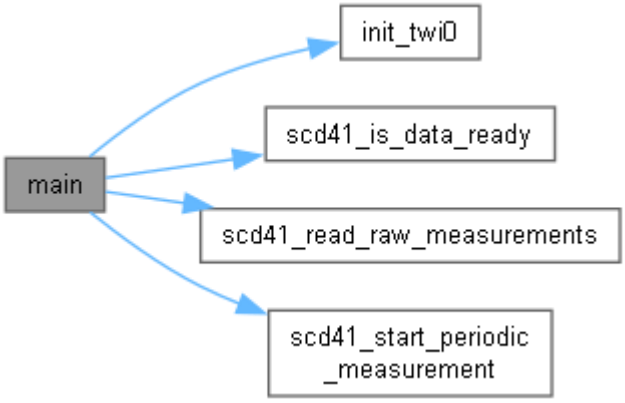
Macro Definition Documentation

#define F_CPU 4000000UL

Function Documentation

int main (void)

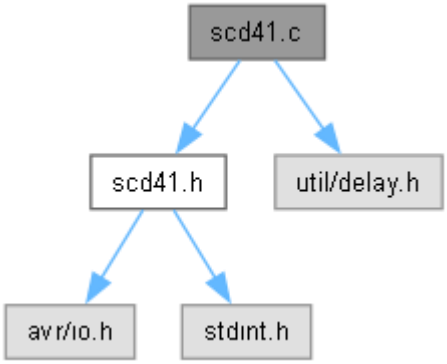
Here is the call graph for this function:


main
init_twi0
scd41_is_data_ready
scd41_read_raw_measurements
scd41_start_periodic
_measurement

scd41.c File Reference

```
#include "scd41.h"
#include <util/delay.h>
```

Include dependency graph for scd41.c:



Macros

- #define **SCD41_I2C_ADDR** 0x62
- #define **CMD_START_MEAS** 0x21B1
- #define **CMD_DATA_READY** 0xE4B8
- #define **CMD_READ_MEAS** 0xEC05

Functions

- void **init_twi0** (void)
- void **scd41_start_periodic_measurement** (void)
- uint8_t **scd41_is_data_ready** (void)
- void **scd41_read_raw_measurements** (uint16_t *co2, uint16_t *temp, uint16_t *hum)

Macro Definition Documentation

#define CMD_DATA_READY 0xE4B8

#define CMD_READ_MEAS 0xEC05

#define CMD_START_MEAS 0x21B1

#define SCD41_I2C_ADDR 0x62

Function Documentation

void init_twi0 (void)

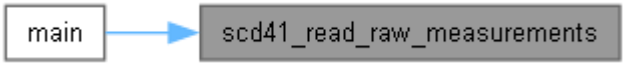
Here is the caller graph for this function:

uint8_t scd41_is_data_ready (void)

Here is the caller graph for this function:

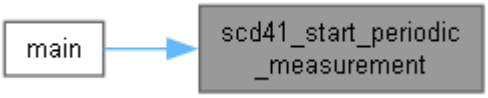
void scd41_read_raw_measurements (uint16_t * co2, uint16_t * temp, uint16_t * hum)

Here is the caller graph for this function:



void scd41_start_periodic_measurement (void)

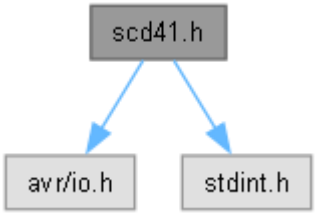
Here is the caller graph for this function:



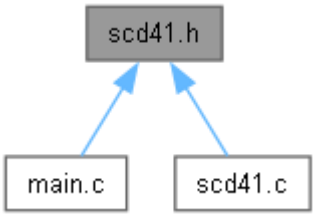
scd41.h File Reference

```
#include <avr/io.h>
#include <stdint.h>
```

Include dependency graph for scd41.h:



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



Functions

- void **init_twi0** (void)
- void **scd41_start_periodic_measurement** (void)
- uint8_t **scd41_is_data_ready** (void)
- void **scd41_read_raw_measurements** (uint16_t *co2, uint16_t *temp, uint16_t *hum)

Function Documentation

void init_twi0 (void)

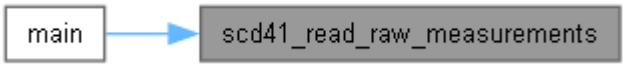
Here is the caller graph for this function:

uint8_t scd41_is_data_ready (void)

Here is the caller graph for this function:

void scd41_read_raw_measurements (uint16_t * co2, uint16_t * temp, uint16_t * hum)

Here is the caller graph for this function:



void scd41_start_periodic_measurement (void)

Here is the caller graph for this function:

scd41.h

Go to the documentation of this file.

```
1 /*
2  * scd41.h
3  *
4  * Interface for communicating with the SCD41 CO?, temperature, and humidity sensor
via TWI0.
5  * Used in Lab 10 Task 3 for a modular multifile implementation.
6  *
7  * Author: Naafiul Hossain
8  */
9
10 #ifndef SCD41_H
11 #define SCD41_H
12
13 #include <avr/io.h>
14 #include <stdint.h>
15
16 // === Public API ===
17 void init_twi0(void);
18 void scd41_start_periodic_measurement(void);
19 uint8_t scd41_is_data_ready(void);
20 void scd41_read_raw_measurements(uint16_t *co2, uint16_t *temp, uint16_t *hum);
21
22 #endif // SCD41_H
```

Index

INDEX