STONY BROOK UNIVERSITY
DEPARTMENT OF COMPUTER AND ELECTRICAL ENGINEERING

ESE 381.L02

Lab 10: Air Quality System I - Basic Operation of SCD41 CO2, Humidity and Temperature Sensor

Name: Jason Tan
SBU ID #: 112319102
Due Date: April 22, 2022 by end of Lab

```c
/*
* lcd_dog_AVR128_driver.h
*
* Created: 3/19/2022 8:09:13 PM
* Author : jason
*/

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

/*
Is what is responsible for transmitting the command for the LCD
*/
void lcd_spi_transmit_CMD (unsigned char cmd);

/*
Is what is responsible for transmitting the data for the LCD into what is to be displayed
*/
void lcd_spi_transmit_DATA (unsigned char cmd);

/*
Initialize the SPI LCD to being blank
*/
void init_spi_lcd (void);

/*
Initialize the LCD Dog to being blank
*/
void init_lcd_dog (void);

/*
Function prototype for 40 ms
*/
void delay_40mS(void);

/*
Function prototype for 40 microsecond
*/
void delay_30uS(void);

/*
To update on the LCD for something to be displayed
*/
void update_lcd_dog(void);


// Display buffer for DOG LCD using sprintf()
char dsp_buff1[17];
char dsp_buff2[17];
char dsp_buff3[17];
```

```
void lcd_spi_transmit_CMD (unsigned char cmd) {
    //Poll until ready to send the command
    //while(!(SPI0_INTFLAGS & SPI_IF_bm)){}
    PORTC_OUT &= ~PIN0_bm;  //Clear PC0 = RS = 0 = command
    PORTA_OUT &= ~PIN7_bm;  //clear PA7 = /SS = selected
    SPI0_DATA = cmd;
    //Poll until ready to send the command
    while(!(SPI0_INTFLAGS & SPI_IF_bm)){}
    PORTA_OUT |= PIN7_bm;   //clear PA7 = /SS = selected

}

void lcd_spi_transmit_DATA (unsigned char cmd) {
    //Poll until ready to send the command
    while(!(SPI0_INTFLAGS & SPI_IF_bm)){}
    PORTC_OUT |= PIN0_bm;   //PC0 = RS = 1 = command
    PORTA_OUT &= ~PIN7_bm;  //clear PA7 = /SS = selected
    SPI0_DATA = cmd;
    //Poll until ready to send the command
    while(!(SPI0_INTFLAGS & SPI_IF_bm)){}
    PORTA_OUT |= PIN7_bm;   //clear PA7 = /SS = selected

}

void init_spi_lcd (void) {
    PORTA_DIR |= PIN4_bm | PIN6_bm | PIN7_bm;  //Set MOSI, SCK and //SS as output
      while MISO as input

    PORTC_DIR |= PIN0_bm;  //Set RS of LCD as output

    SPI0_CTRLA |= SPI_ENABLE_bm | SPI_MASTER_bm; //Enable the SPI and make it in the
      Master Mode

    SPI0_CTRLB |= SPI_SSD_bm | SPI_MODE1_bm | SPI_MODE0_bm;   //Put the SPI with slave
       select (/SS) to be enabled and be in SPI Mode 3 (CPOL = 1 and CPHA = 1)

    //Wait to clears the IF flag in the INTFLAG meaning there no serial data yet to be
       transferred
    //while(SPI0_INTFLAGS & SPI_IF_bm){}
    PORTC_OUT &= ~PIN0_bm;  //PC0 = RS = 0 = command



}


void init_lcd_dog (void) {

    init_spi_lcd();     //Initialize mcu for LCD SPI
```

```
    //start_dly_40ms:
    delay_40mS();    //startup delay.


    //func_set1:
    lcd_spi_transmit_CMD(0x39);   // sedn function set #1
    delay_30uS();   //delay for command to be processed


    //func_set2:
    lcd_spi_transmit_CMD(0x39); //send fuction set #2
    delay_30uS();   //delay for command to be processed


    //bias_set:
    lcd_spi_transmit_CMD(0x1E); //set bias value.
    delay_30uS();   //delay for command to be processed


    //power_ctrl:
    lcd_spi_transmit_CMD(0x55); //~ 0x50 nominal for 5V
    //~ 0x55 for 3.3V (delicate adjustment).
    delay_30uS();   //delay for command to be processed


    //follower_ctrl:
    lcd_spi_transmit_CMD(0x6C); //follower mode on...
    delay_40mS();   //delay for command to be processed


    //contrast_set:
    lcd_spi_transmit_CMD(0x7F); //~ 77 for 5V, ~ 7F for 3.3V
    delay_30uS();   //delay for command to be processed


    //display_on:
    lcd_spi_transmit_CMD(0x0c); //display on, cursor off, blink off
    delay_30uS();   //delay for command to be processed


    //clr_display:
    lcd_spi_transmit_CMD(0x01); //clear display, cursor home
    delay_30uS();   //delay for command to be processed


    //entry_mode:
    lcd_spi_transmit_CMD(0x06); //clear display, cursor home
    delay_30uS();   //delay for command to be processed
}


void delay_40mS(void) {
```

```
    int i;
    for (int n = 40; n > 0; n--)
    for (i = 0; i < 800; i++)
    __asm("nop");
}

void delay_30uS(void) {
    int i;
    for (int n = 1; n > 0; n--)
    for (i = 0; i < 2; i++)
    __asm("nop");
}


// Updates the LCD display lines 1, 2, and 3, using the
// contents of dsp_buff_1, dsp_buff_2, and dsp_buff_3, respectively.
void update_lcd_dog(void) {

    init_spi_lcd();     //init SPI port for LCD.

    // send line 1 to the LCD module.
    lcd_spi_transmit_CMD(0x80); //init DDRAM addr-ctr
    delay_30uS();
    for (int i = 0; i < 16; i++) {
        lcd_spi_transmit_DATA(dsp_buff1[i]);
        delay_30uS();
    }

    // send line 2 to the LCD module.
    lcd_spi_transmit_CMD(0x90); //init DDRAM addr-ctr
    delay_30uS();
    for (int i = 0; i < 16; i++) {
        lcd_spi_transmit_DATA(dsp_buff2[i]);
        delay_30uS();
    }

    // send line 3 to the LCD module.
    lcd_spi_transmit_CMD(0xA0); //init DDRAM addr-ctr
    delay_30uS();
    for (int i = 0; i < 16; i++) {
        lcd_spi_transmit_DATA(dsp_buff3[i]);
        delay_30uS();
    }
}
```

```c
/*
 * SCD41_LCD_Multifile.c
 *
 * Created: 4/18/2022 12:56:05 AM
 * Author : jason
 */

#include <avr/io.h>
#include "lcd_dog_AVR128_driver.h"
#include "SCD41_AVR128_driver.h"


int main(void)
{

    I2C0_SCD41_init();   //Initializes  the AVR128DB48 I2C0 to communicate with SCD41
    //Initialize the buffer of the LCD
    init_lcd_dog();
    while (1)
    {
        while(SCD41_get_data_ready_status(I2CSLAVE_ADDR_WRITE,
          ADDRESS_GETDATAREADY_MSB, ADDRESS_GETDATAREADY_LSB)){
            //_delay_ms(1);     //Wait 1 ms to read the measurement
            SCD41_start_periodic_measurement(I2CSLAVE_ADDR_WRITE,
              ADDRESS_STARTPERIODIC_MSB, ADDRESS_STARTPERIODIC_LSB);

            SCD41_read_measurement(I2CSLAVE_ADDR_WRITE, ADDRESS_READMEASUR_MSB,
              ADDRESS_READMEASUR_LSB);

            //Print the CO2 value into LCD buffer
            sprintf(dsp_buff1, "CO2: %d", getParseCO2);

            //Print the humidity value into LCD buffer
            sprintf(dsp_buff2, "Humidity: %d", getParseRh);

            //Print temperature value into LCD buffer
            sprintf(dsp_buff3, "Temperature: %d", getParseTemp);

            //Update the 3 line messages into the LCD buffer
            update_lcd_dog();

            //_delay_ms(1);    //Wait 500 ms during measurement to stop periodic
              measurement

            //SCD41_stop_periodic_measurement(I2CSLAVE_ADDR_WRITE,
              ADDRESS_STOPPERIODIC_MSB, ADDRESS_STOPPERIODIC_LSB);
            //_delay_ms(500);    //Wait 500 ms during measurement to stop periodic
              measurement
        }

    }
}
```

```
/*
 * SCD41_AVR128_driver.h
 *
 * Created: 4/18/2022 12:58:29 AM
 *  Author: jason
 */


#ifndef SCD41_AVR128_DRIVER_H_
#define SCD41_AVR128_DRIVER_H_

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

//Function Prototypes that will be used
void I2C0_SCD41_init();
void SCD41_start_periodic_measurement(uint8_t, uint8_t, uint8_t);
void SCD41_stop_periodic_measurement(uint8_t, uint8_t, uint8_t);
void SCD41_read_measurement(uint8_t, uint8_t, uint8_t);
uint8_t SCD41_get_data_ready_status(uint8_t, uint8_t, uint8_t);
uint8_t sensirion_common_generate_crc(const uint8_t*, uint16_t);

//For computing the checksum
#define CRC8_POLYNOMIAL 0x31
#define CRC8_INIT 0xFF

#define I2CSLAVE_ADDR_WRITE 0xC4      // 110 0010 0    0xC4
#define I2CSLAVE_ADDR_READ 0xC5       // 110 0010 1    0xC5

//The least significant and most significant byte address for the start periodic
  function
#define ADDRESS_STARTPERIODIC_LSB 0xB1
#define ADDRESS_STARTPERIODIC_MSB 0x21

//The least significant and most significant byte address for the stop periodic
  function
#define ADDRESS_STOPPERIODIC_LSB 0x86
#define ADDRESS_STOPPERIODIC_MSB 0x3F

//The least significant and most significant byte address for the read measurement
  periodic function
#define ADDRESS_READMEASUR_LSB 0x05
#define ADDRESS_READMEASUR_MSB 0xEC

//The least significant and most significant byte address for the get data ready
  function
#define ADDRESS_GETDATAREADY_LSB 0xB8
#define ADDRESS_GETDATAREADY_MSB 0xE4


//For the get_data_ready_status function to get the data response value
```

```
uint8_t readDataStatusMSB;
uint8_t readDataStatusLSB;
uint16_t getDataStatusReadyResponse, getDataStatusReadyResponse1;

//For the get_measurement function to get the CO2 value plus the CRC
uint8_t readCO2MSB;
uint8_t readCO2LSB;
uint8_t readCO2CRC;
uint16_t getParseCO2;

//For the get_measurement function to get the temperature value plus the CRC
uint8_t readTempMSB;
uint8_t readTempLSB;
uint8_t readTempCRC;
uint16_t getParseTemp;

//For the get_measurement function to get the relative humidity value plus the CRC
uint8_t readRhMSB;
uint8_t readRhLSB;
uint8_t readRhCRC;
uint16_t getParseRh;

uint8_t readDataStatusCRC;

int i = -1;
int j = -1;

uint8_t storedCO2[2];
uint8_t storedTemp[2];
uint8_t storedRH[2];

//uint16_t storedCO2[1000];
//uint16_t storedTemp[1000];
//uint16_t storedRH[1000];

//Store all the history of checksum in an array
//uint8_t storeCO2CheckSum[1000];
//uint8_t storeTempCheckSum[1000];
//uint8_t storeRhCheckSum[1000];
//uint8_t storeDataReadyStatusCRC[1000];


/*
int main(void)
{

    I2C0_SCD41_init();   //Initializes  the AVR128DB48 I2C0 to communicate with SCD41
    while (1)
    {
        //Check on the least significant 11 bits if they are not 0 for data to be
          ready to be read.
        while(SCD41_get_data_ready_status(I2CSLAVE_ADDR_WRITE,
```

```
          ADDRESS_GETDATAREADY_MSB, ADDRESS_GETDATAREADY_LSB)){
            //_delay_ms(1);     //Wait 1 ms to read the measurement
            SCD41_start_periodic_measurement(I2CSLAVE_ADDR_WRITE,
              ADDRESS_STARTPERIODIC_MSB, ADDRESS_STARTPERIODIC_LSB);

            SCD41_read_measurement(I2CSLAVE_ADDR_WRITE, ADDRESS_READMEASUR_MSB,
              ADDRESS_READMEASUR_LSB);
            //_delay_ms(1);    //Wait 500 ms during measurement to stop periodic
              measurement

            //SCD41_stop_periodic_measurement(I2CSLAVE_ADDR_WRITE,
              ADDRESS_STOPPERIODIC_MSB, ADDRESS_STOPPERIODIC_LSB);
            //_delay_ms(500);    //Wait 500 ms during measurement to stop periodic
              measurement
        }
        //_delay_ms(1);     //Wait 1 ms to read the measurement
    }
}
*/


//Initializes the AVR128DB48's I2C to communicate with the MCP23017.
//The bit transfer rate between the AVR128DB48 and the MCP23017 must be
//as fast as possible, but less than or equal to 100 kb/s.
void I2C0_SCD41_init()
{
    //Baud rate for the I2C which set to 15 assuming that is the fastest you can get
      to
    TWI0.MBAUD = 15;

    //Enable for the I2C Master
    TWI0.MCTRLA = TWI_ENABLE_bm;

    //Force the I2C to the idle state
    TWI0.MSTATUS = TWI_BUSSTATE_IDLE_gc;
}

//Starts the periodic measurement, signal update interval is 5 seconds
void SCD41_start_periodic_measurement(uint8_t SCD41_address, uint8_t SCD41_MSB,
  uint8_t SCD41_LSB){

    //To write the address of SCD41 (0x62) except also write operation so 110 0010 0
    TWI0_MADDR = SCD41_address;
    while(!(TWI0_MSTATUS & TWI_WIF_bm));

    //To write the most significant byte
    TWI0_MDATA = SCD41_MSB;
    while(!(TWI0_MSTATUS & TWI_WIF_bm));

    //To write the least significant byte
    TWI0_MDATA = SCD41_LSB;
    while(!(TWI0_MSTATUS & TWI_WIF_bm));
```

```
    //Execute acknowledge action followed by issuing a stop condition
    TWI0_MCTRLB = TWI_MCMD_STOP_gc;
}


//This function is what stops the periodic measurement to change the sensor configuration or to save
//power. Note that the sensor will only respond to other commands after waiting 500 ms after issuing the
//stop_periodic_measurement command
void SCD41_stop_periodic_measurement(uint8_t SCD41_address, uint8_t SCD41_MSB, uint8_t SCD41_LSB){
    //To write the address of SCD41 (0x62) except also write operation so 110 0010 0
    TWI0_MADDR = SCD41_address;
    while(!(TWI0_MSTATUS & TWI_WIF_bm));

    //To write the most significant byte
    TWI0_MDATA = SCD41_MSB;
    while(!(TWI0_MSTATUS & TWI_WIF_bm));

    //To write the least significant byte
    TWI0_MDATA = SCD41_LSB;
    while(!(TWI0_MSTATUS & TWI_WIF_bm));

    //Execute acknowledge action followed by issuing a stop condition
    TWI0_MCTRLB = TWI_MCMD_STOP_gc;
}

//Function to read the measurement value for the temperature, relative humidity and CO2 of the SCD41
void SCD41_read_measurement(uint8_t SCD41_address, uint8_t SCD41_MSB, uint8_t SCD41_LSB){
    //To write the address of SCD41 (0x62) except also write operation so 110 0010 0
    TWI0_MADDR = SCD41_address;
    while(!(TWI0_MSTATUS & TWI_WIF_bm));

    //To write the most significant byte command
    TWI0_MDATA = SCD41_MSB;
    while(!(TWI0_MSTATUS & TWI_WIF_bm));

    //To write the least significant byte command
    TWI0_MDATA = SCD41_LSB;
    _delay_ms(1);    //Wait 500 ms during measurement to stop periodic measurement
    while(!(TWI0_MSTATUS & TWI_WIF_bm));

    //To write the I2C slave address which would then indicate reading from the slave to the master
    TWI0_MADDR = I2CSLAVE_ADDR_READ; //SCD41_address;

    //To start reading from the slave the Data_MSB of CO2
    while(!(TWI0_MSTATUS & TWI_RIF_bm));
```

```
    TWI0_MCTRLB = TWI_ACKACT_ACK_gc;
    readCO2MSB = TWI0_MDATA;
    storedCO2[0] = readCO2MSB;

    //Poll until there's something to read from the slave: the Data_LSB of CO2
    while(!(TWI0_MSTATUS & TWI_RIF_bm));
    TWI0_MCTRLB = TWI_ACKACT_ACK_gc;
    readCO2LSB = TWI0_MDATA;
    storedCO2[1] = readCO2LSB;

    //Read modify write for the CO2
    getParseCO2 = (getParseCO2 & ~(0b11111111 << 0)) | ((readCO2LSB & 0b11111111) <<
      0);
    getParseCO2 |= (getParseCO2 & ~(0b11111111 << 8)) | ((readCO2MSB & 0b11111111) <<
      8);
    //i++;
    //storedCO2[i] = getParseCO2;


    //Poll until there's something to read from the slave: the CRC of CO2 which isn't
      necessary to read for lab 10
    while(!(TWI0_MSTATUS & TWI_RIF_bm));
    TWI0_MCTRLB = TWI_ACKACT_ACK_gc;
    readCO2CRC = TWI0_MDATA;
    //storeCO2CheckSum[i] = readCO2CRC;

    //-------------------------------------------------------------------------------
      -----------------

    //Poll until there's something to read from the slave: the Most significant byte
      of the temperature
    while(!(TWI0_MSTATUS & TWI_RIF_bm));
    TWI0_MCTRLB = TWI_ACKACT_ACK_gc;
    readTempMSB = TWI0_MDATA;
    storedTemp[0] = readTempMSB;

    //Poll until there's something to read from the slave: the least significant byte
      of the temperature
    while(!(TWI0_MSTATUS & TWI_RIF_bm));
    TWI0_MCTRLB = TWI_ACKACT_ACK_gc;
    readTempLSB = TWI0_MDATA;
    storedTemp[1] = readTempLSB;

    //Read modify write to parse each byte of the two bytes into the 16 bit field for
      the temperature
    getParseTemp = (getParseTemp & ~(0b11111111 << 0)) | ((readTempLSB & 0b11111111)
      << 0);
    getParseTemp |= (getParseTemp & ~(0b11111111 << 8)) | ((readTempMSB & 0b11111111)
      << 8);
    //storedTemp[i] = getParseTemp;

    //Poll to read the CRC of temperature which isn't necessary to read for lab 10
```

```
    while(!(TWI0_MSTATUS & TWI_RIF_bm));
    TWI0_MCTRLB = TWI_ACKACT_ACK_gc;
    readTempCRC = TWI0_MDATA;
    //storeTempCheckSum[i] = readTempCRC;

    //-------------------------------------------------------------------------------
      -----------------

    //Poll until there's something to read from the slave: the Most significant byte
      of the relative humidity (RH)
    while(!(TWI0_MSTATUS & TWI_RIF_bm));
    TWI0_MCTRLB = TWI_ACKACT_ACK_gc;
    readRhMSB = TWI0_MDATA;
    storedRH[0] = readRhMSB;

    //Poll until there's something to read from the slave: the least significant byte
      of the relative humidity (RH)
    while(!(TWI0_MSTATUS & TWI_RIF_bm));
    TWI0_MCTRLB = TWI_ACKACT_ACK_gc;
    readRhLSB = TWI0_MDATA;
    storedTemp[1] = readRhLSB;

    //Read modify write to parse each byte of the two bytes into the 16 bit field for
      the relative humidity (RH)
    getParseRh = (getParseRh & ~(0b11111111 << 0)) | ((readRhLSB & 0b11111111) << 0);
    getParseRh |= (getParseRh & ~(0b11111111 << 8)) | ((readRhMSB & 0b11111111) << 8);
    //storedRH[i] = getParseRh;


    //Poll to read the CRC of relative humidity (RH) which isn't necessary to read for
       lab 10
    while(!(TWI0_MSTATUS & TWI_RIF_bm));
    readRhCRC = TWI0_MDATA;
    //  storeRhCheckSum[i] = readRhCRC;
    //Master send to slave to stop reading data by sending a NACK response
    TWI0_MCTRLB |= TWI_MCMD_STOP_gc | TWI_ACKACT_bm;

}


//Check if data is ready to be read from the SCD41
uint8_t SCD41_get_data_ready_status(uint8_t SCD41_address, uint8_t SCD41_MSB, uint8_t
  SCD41_LSB){
    //Poll to write the address of SCD41 (0x62) except also write operation so 110
      0010 0
    TWI0_MADDR = SCD41_address;
    while(!(TWI0_MSTATUS & TWI_WIF_bm));

    //Poll To write the most significant byte command
    TWI0_MDATA = SCD41_MSB;
    while(!(TWI0_MSTATUS & TWI_WIF_bm));
```

```
    //Poll To write the least significant byte command
    TWI0_MDATA = SCD41_LSB;
    while(!(TWI0_MSTATUS & TWI_WIF_bm));
    _delay_ms(1);       //Wait 1 ms to read the measurement

    //To write the I2C slave address which would then indicate reading from the slave
      to the master
    TWI0_MADDR = I2CSLAVE_ADDR_READ; //SCD41_address;

    //Poll until there's something to read from the slave: the Data_MSB
    while(!(TWI0_MSTATUS & TWI_RIF_bm));

    TWI0_MCTRLB = TWI_ACKACT_ACK_gc;
    readDataStatusMSB = TWI0_MDATA;

    //Poll until there's something to read from the slave: the Data_LSB
    while(!(TWI0_MSTATUS & TWI_RIF_bm));
    TWI0_MCTRLB = TWI_ACKACT_ACK_gc;
    readDataStatusMSB = TWI0_MDATA;

    //16 bit data status result
    getDataStatusReadyResponse = (getDataStatusReadyResponse & ~(0b11111111 << 0)) |
      ((readDataStatusLSB & 0b11111111) << 0);
    getDataStatusReadyResponse |= (getDataStatusReadyResponse & ~(0b11111111 << 8)) |
      ((readDataStatusMSB & 0b11111111) << 8);

    //Poll until there's something to read from the slave: the CRC of
      data_ready_status value which we don't need for lab 10
    while(!(TWI0_MSTATUS & TWI_RIF_bm));
    j++;
    readDataStatusCRC = TWI0_MDATA;
    //  storeDataReadyStatusCRC[j] = readDataStatusCRC;
    //Master send to slave to stop reading data by sending a NACK response
    TWI0_MCTRLB |= TWI_MCMD_STOP_gc | TWI_ACKACT_NACK_gc;  //Send a NACK meaning that
      done reading data

    //The case if the LSB 11 bits are not all 0's else go to the return value
    if(getDataStatusReadyResponse & 0b11111111111){
        return 1;
    }

    return 0;
}



//This is what is responsible for computing the checksum
uint8_t sensirion_common_generate_crc(const uint8_t* data, uint16_t count) {
    uint16_t current_byte;
    uint8_t crc = CRC8_INIT;
    uint8_t crc_bit;
```

```
    /* calculates 8-Bit checksum with given polynomial */
    for (current_byte = 0; current_byte < count; ++current_byte) {
        crc ^= (data[current_byte]);
        for (crc_bit = 8; crc_bit > 0; --crc_bit) {
            if (crc & 0x80)
            crc = (crc << 1) ^ CRC8_POLYNOMIAL;
            else
            crc = (crc << 1);
        }
    }
    return crc;
}

#endif /* SCD41_AVR128_DRIVER_H_ */
```

Lab 10 Questions

1. Fill out the logic level compatibility check list (in the laboratory folder) for the AVR128DB48 and the SCD41. Make the AVR128DB48 device A. Both devices are operated with a supply voltage of 3.3V.

AVR128DB48 TWI characteristics

V_{OL}	Output low voltage	—	—	$0.2xV_{DD}$	V	I_{load} = 20 mA, Fast mode+
		—	—	0.4V		I_{load} = 3 mA, Normal mode, V_{DD} > 2V
		—	—	$0.2×V_{DD}$		I_{load} = 3 mA, Normal mode, V_{DD} ≤ 2V

V_OL and I_OLMax for the AVR128DB48 for the TWI

V_{IH}	Input high voltage	$0.7×V_{DD}$	—	—	V	
V_{IL}	Input low voltage	—	—	$0.3×V_{DD}$	V	

V_IH and V_IL for the AVR128DB48 for the TWI

Input Leakage Current(2)						
I_{IL}	I/O PORTS(3)	—	<5	—	nA	GND ≤ V_{PIN} ≤ V_{DD}, pin at high-impedance, 85°C
		—	<5	—	nA	GND ≤ V_{PIN}(5) ≤ V_{DD}, pin at high-impedance, 125°C

AVR128 For finding the I_IHmax and the I_ILmax

SCD41 TWI characteristics

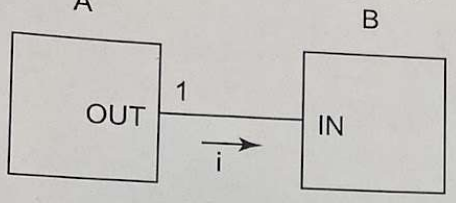
Input high level voltage	V_{IH}		0.7 x V_{DD}		1 x V_{DD}	-
Input low level voltage	V_{IL}				0.3 x V_{DD}	-
Output low level voltage	V_{OL}	3 mA sink current			0.66	V

SCD41 for finding V_IH, V_IL and V_OL and also for I_OL

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

Lab 10 Q1

ATMEGA128 0040 SC041

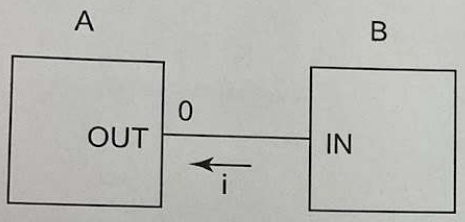


$V_{OHmin(A)} > V_{IHmin(B)}$

2.6 V > 0.7 × 3.3V = 2.31V

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

[illegible] >

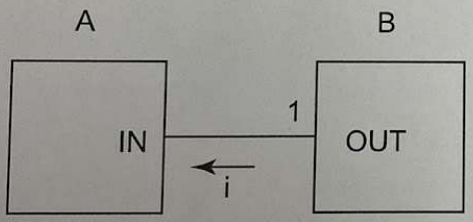


$V_{OLmax(A)} < V_{ILmax(B)}$

0.4 V < 0.3 × 3.3V = .99 V

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

3 mA > [illegible]

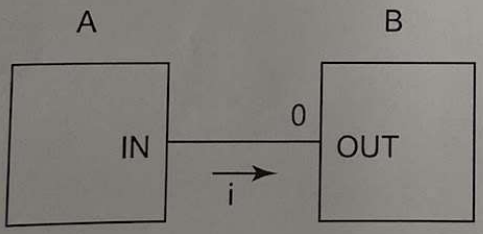


$V_{IHmin(A)} < V_{OHmin(B)}$

0.7 × 3.3V = 2.31V < [illegible]

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

5nA < [illegible]



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

0.3 × 3.3 = 0.99V > 0.66 V

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

5nA < 3mA

2. What is the maximum specified serial clock speed at which the SCD41 can be operated? What is the maximum speed and which the SCD41 can be operated in this design? Show your calculations for both.

The maximum specified serial clock speed at which the SCD41 can be operated is 100 kHz. The maximum speed in which the SCD41 can be operated in this design is the baud rate of 15.

$$BAUD = \frac{f_{CLK\_PER}}{2 \times f_{SCL}} - \left(5 + \frac{f_{CLK\_PER} \times T_R}{2}\right) \qquad (2)$$

For part 2 of question 2

f_CLK_PER = 4000000, f_SCL = 100000, T_R = 0.000001

BAUD = 4000000/2 * 100000 - (5 + 4000000 * .000001/ 2) = 13

2.4 Timing Specifications

Table 7 list the timings of the ASIC part and does not reflect the availability or usefulness of the sensor readings. The SCD4x supports the I²C "standard-mode" as is described elsewhere (see footnote [12]).

Parameter	Condition	Min.	Max.	Unit
Power-up time	After hard reset, $V_{DD} \geq 2.25$ V	-	1000	ms
Soft reset time	After re-initialization (i.e. reinit)	-	1000	ms
SCL clock frequency	-	0	100	kHz

SCD41 SCL clock freq.

Table 39-22. TWI - Timing Characteristics

Symbol	Description	Min.	Typ.	Max.	Unit	Condition
f_{SCL}	SCL clock frequency	0	—	1000	kHz	Max. frequency requires system clock at 10 MHz

AVR128DB48 SCL clock freq.

Reading measurement Salae Screenshot: