**Networked Embedded Systems**

**Practicum 3: Sensors and Communication**

**Group number: 8**

|  |  |  |
| --- | --- | --- |
| **Name** | **Student ID** | **Email** |
| Bernberger Sarah | k12112018 | sarahbernberger.sb@gmail.com |
| Grundner Simon | k12136610 | simon.grundner@gmail.com |

**19.05.2024**

Map of Content

[Theory Questions 3](#_Toc168519895)

[Task A: Read Data from the LIS3MDL Magnetic Sensor 6](#_Toc168519896)

[A.1. Calculations 6](#_Toc168519897)

[A.2. Implementation 7](#_Toc168519898)

[Main Function 8](#_Toc168519899)

[Read / Write Specific Registers 8](#_Toc168519900)

[A.3. Results 8](#_Toc168519901)

[A.4. Discussion 8](#_Toc168519902)

[Task B: Read Humidity Values from the HTS221 Sensor 9](#_Toc168519903)

[B.1. Calculations 9](#_Toc168519904)

[B.2. Implementation 9](#_Toc168519905)

[B.3. Results 9](#_Toc168519906)

[B.4. Discussion 9](#_Toc168519907)

[Task C: Read Temperature Values from the HTS221 Sensor 10](#_Toc168519908)

[C.1. Calculations 10](#_Toc168519909)

[C.2. Implementation 10](#_Toc168519910)

[C.3. Results 10](#_Toc168519911)

[C.4. Discussion 10](#_Toc168519912)

[Task D: Read Temperature & Pressure Values from the LPS22HH Sensor 11](#_Toc168519913)

[D.1. Calculations 11](#_Toc168519914)

[D.2. Implementation 11](#_Toc168519915)

[D.3. Results 11](#_Toc168519916)

[D.4. Discussion 11](#_Toc168519917)

[Index 12](#_Toc168519918)

[Figures 12](#_Toc168519919)

[Code Segments 12](#_Toc168519920)

[Tables 12](#_Toc168519921)

# Theory Questions

1. **What is calibration and why do the sensors need that? How are calibration values used to calculate the measurement result? Visualize the relation between calibration curve and measured values (simplified, e.g., linear curves).**

Calibration values are needed, to ensure that the acquired sensor data corresponds to the actual physically present measurement (such as ambient temperature). Wrong calibration can lead to systematic errors, where the sensor makes consistent measurements, but is offset from the real-world value.

|  |  |  |
| --- | --- | --- |
| Zufällige und systematische Fehler  Figure 1: Good Sensor Good Calibration | Zufällige und systematische Fehler  Figure 2: Bad Sensor | Zufällige und systematische Fehler  Figure 3: Good Sensor Bad Calibration |

By linear interpolation, the acquired sensor data is shifted to the correct systematic offset. In this case (HTS221) the calibration values are stored in non-volatile memory and set from factory, so no additional calibration is required by the user.

**Ein Bild, das Text, Reihe, Diagramm, Screenshot enthält.

Automatisch generierte Beschreibung**

Figure 4: Linear Interpolation Diagram for HTS221 Humidity Sensor

Interpolation can be achieved with following formular (derived from Figure 4):

1. **Describe I2C. Where do we use it? How does communication via I2C work?**

I2C is short for Inter-Integrated Circuit, which literally means the interconnection between multiple ICs. I2C is a simple Communication protocol which works on a serial data bus consisting of two wires[[1]](#footnote-1), the Serial Data Line (SDA) and the Serial Clock Line (SCL). The I2C-bus has following aspects:

* **Serial:** The data is transmitted on one line , where the data word is sent out bit by bit.
* **Synchronous:** The bus has a clock line, controlling when a data word begins, and a bit is read.
* **Simplex:** Transmitting and receiving data takes place on the same line, meaning only one peripheral can talk at a time.

Ein Bild, das Screenshot, Diagramm, Symbol, Schrift enthält.

Automatisch generierte BeschreibungThe I2C bus requires Pull-Up resistors on each data line because the output stage only consists of a FET Pulling the bus line low, leaving the line floating during the highperiod and the idle state. This topology is called an *Open Collector* output.

I2C is usually used where the master is in short proximity to its I2C peripherals for example on printed circuit boards. Due to its bus topology, longer data lines limit the data rate due to slower rise and fall times caused by a higher bus capacitance. The edge rise time can be somewhat controlled by the pullup resistors. Lower pull ups can compensate for an increased bus capacitance () but affect the current consumption.

Figure 5: I2C Open Collector Output

Communication via I2C works as Follows: (In this case writing one byte to a register)

* **ST**: Start-Bit - Pulling the Data line Low initiates a transmission
* **SAD + W**: 7-Bit Slave Address + Read (1) / Write (0) bit
* **SAK**: Slave Acknowledge
* **SUB**: Sub address (Register)
* **DATA**: 8-Bit Data-Word
* **SP**: Stop-Condition

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Master | ST | SAD + W |  | SUB |  | DATA |  | SP |
| Slave |  |  | SAK |  | SAK |  | SAK |  |

Table 1: I2C Transmission Example

1. **Describe UART. What does the abbreviation mean? How does communication via UART work?**

Ein Bild, das Screenshot, Diagramm, Rechteck, Design enthält.

Automatisch generierte BeschreibungUART stand for **U**niversal **A**synchronous **R**eceive and **T**ransmit and is, as the name suggests, a **serial**, **asynchronous** transmission protocol. UART can be used to interconnect **two** Peripherals, in this case the STM32H7 and some sort of USB-Bridge to communicate with the PC via USB. The UART bus has two signals RX and TX, while RX from one peripheral must connect to TX from the other. The bus participants are equal meaning that there is **no master controller**, which enables each peripheral to send and receive at the same time, making the data transmission **full duplex.**

Figure 6: UART Interconnection

A UART Transmission has following Format:

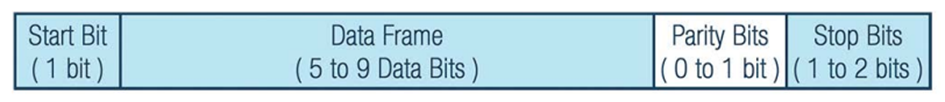


Figure 7: UART Transmission Frame

A Start Bit signalizes the receiver that a new transmission is beginning. The voltage on the transmission line is *Normally High* and is pulled down by the Start Bit. This is followed by the data frame, which is five to nine bits long depending on the configuration. A parity bit is used to validate the transmission (can be disabled). At the end of the transmission, a stop bit sets the bus back to the idle state.

# Task A: Read Data from the LIS3MDL Magnetic Sensor

## A.1. Calculations

There are several control registers (CTRL\_REGx, x: 1-5) to configure how the sensor operates.

*Describe the necessary startup sequence. Which registers are involved and what does each command do?*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter Description** | **Value** | **Unit (if any)** | **Register Address** |
|  |  |  |  |

*Registers for measurement:*

|  |  |
| --- | --- |
| **Description** | **Register Address** |
|  |  |

*How is the magnetic value of each axis calculated from the involved registers? Explain!*

## A.2. Implementation

|  |  |
| --- | --- |
| The LIS3MDL Magnetometer Interface consists of two header files and one source file, where one header *(lis2mdl.h)* declares the function prototypes for the application interface and the other *(lis3mdl\_regsiters.h)* defines values for the registers to reduce magic numbers in code and make it more readable.  A simple struct LIS3MDL\_HandleTypeDef is defined to pass I2C relevant parameters, such as the I2C-handle and the slave address, into the different functions. Additionally, data read from the sensor is stored in this struct as well. | Ein Bild, das Text, Screenshot, Schrift enthält.  Automatisch generierte Beschreibung  Figure 8: LIS3MDL Interface Structure |
|  | Code Segment 1: LIS2MDL\_HandleTypeDef |

|  |
| --- |
| LIS3MDL\_Init(LIS3MDL\_HandleTypeDef\* hlis3mdl, I2C\_HandleTypeDef\* hi2c, uint8\_t address) |
| Initializes the sensor. Assigns the I2C-Handle and the address to the LIS3MDL-Struct. Performs all initial control register configurations.  HAL\_OK: success  HAL\_ERROR: device not connected, address invalid or I2C communication failed |
| LIS3MDL\_ReadRegister(LIS3MDL\_HandleTypeDef\* hlis3mdl, uint8\_t reg, uint8\_t\* data); |
| Reads the provided register and stores the result in a data pointer. Acts as a wrapper function for the HAL\_I2C\_MemRead() function to reduce the number of redundant parameters.  HAL\_OK: success  HAL\_ERROR: I2C communication failed or timed out |
| LIS3MDL\_ReadRegisters(LIS3MDL\_HandleTypeDef\* hlis3mdl, uint8\_t reg, uint8\_t\* data, uint16\_t size) |
| Reads multiple registers starting at the one provided and auto increments the register address by one for each reading operation. Stores the acquired data in a provided data-pointer.  HAL\_OK: success  HAL\_ERROR: I2C communication failed or timed out |
| LIS3MDL\_WriteRegister(LIS3MDL\_HandleTypeDef\* hlis3mdl, uint8\_t reg, uint8\_t data) |
| Writes provided data to a specified register. Acts as a wrapper function for the HAL\_I2C\_MemWrite() function to reduce the number of redundant parameters.  HAL\_OK: success  HAL\_ERROR: I2C communication failed or timed out |
| LIS3MDL\_ReadXYZ(LIS3MDL\_HandleTypeDef\* hlis3mdl) |
| Reads the Data on all six sensor registers and converts it into a Gauss value. The data is stored in the x, y and z field of the LIS3MDL-Struct.  HAL\_OK: success  HAL\_ERROR: I2C communication failed or timed out |
| LIS3MDL\_ReadStatus(LIS3MDL\_HandleTypeDef\* hlis3mdl) |
| Reads the status register and stores it in the status field of the LIS3MDL-Struct  HAL\_OK: success  HAL\_ERROR: I2C communication failed or timed out |

### Main Function



### Read / Write Specific Registers

*Describe your solution. Add all relevant code snippets into a listing or as screenshots and describe their purpose. Remember to use generic methods with well-defined parameters!!*

## A.3. Results

*Add a screenshot of the UART output.*

## A.4. Discussion

*Describe your experiences (e.g., design decisions, problems, lesson learned). Which part of the code will be reusable?*

# Task B: Read Humidity Values from the HTS221 Sensor

## B.1. Calculations

1. *Describe the necessary startup sequence (e.g., set CTRL\_REG1 (20h) to 0x81). What is the default value of CTRL\_REG1?*
2. *Describe how the relative humidity is calculated (incl formula) and describe each parameter! Which registers are involved? Which parameters are constants? Which parameters change? Where do you find the values that are needed for your calculation?*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Description** | **Value or source** | **Unit (if any)** | **Register address** |
|  |  |  |  |  |

## B.2. Implementation

*Describe your solution. Add all relevant code snippets into a listing or as screenshots and describe their purpose.**Remember to use generic methods with well-defined parameters!!*

## B.3. Results

*Add a screenshot of the UART output.*

## B.4. Discussion

*Describe your experiences (e.g., design decisions, problems, lesson learned). Which part of the code will be reusable?*

# Task C: Read Temperature Values from the HTS221 Sensor

## C.1. Calculations

*Describe how the temperature is calculated (incl formula) and describe each parameter! Which registers are involved? Which parameters are constants? Which parameters change? Where do you find the values that are needed for your calculation?*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Description** | **Value or source** | **Unit (if any)** | **Register Address** |
|  |  |  |  |  |

## C.2. Implementation

*Describe your solution. Add all relevant code snippets into a listing or as screenshots and describe their purpose.**Remember to use generic methods with well-defined parameters!!*

## C.3. Results

*Add a screenshot of the UART output.*

## C.4. Discussion

*Describe your experiences (e.g., design decisions, problems, lesson learned). Which part of the code will be reusable?*

# Task D: Read Temperature & Pressure Values from the LPS22HH Sensor

## D.1. Calculations

*For the startup, you have to set the registers CTRL\_REG1 and CTRL\_REG2. Based on the application notes, choose a fitting operation mode and justify your choice*

|  |  |  |
| --- | --- | --- |
|  | **Register** | **Value** |
| CTRL\_REG1 |  |  |
| CTRL\_REG2 |  |  |

*Describe how the pressure is calculated (formula) and fill in the following table:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Description** | **Value or source** | **Unit (if any)** | **Register Address** |
|  |  |  |  |  |

*Describe how the temperature is calculated (formula) and fill in the following table:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Description** | **Value or Source** | **Unit (if any)** | **Register Address** |
|  |  |  |  |  |

## D.2. Implementation

*Describe your solution. Add all relevant code snippets into a listing or as screenshots and describe their purpose.**Remember to use generic methods with well-defined parameters!!*

## D.3. Results

*Add a screenshot of the UART output.*

## D.4. Discussion

*Describe your experiences. Does your solution have any limitations? How would an ideal solution behave in your opinion?*

# Index

**Keine Indexeinträge gefunden.**

## Figures

[Figure 1: Good Sensor Good Calibration 3](#_Toc168519885)

[Figure 2: Bad Sensor 3](#_Toc168519886)

[Figure 3: Good Sensor Bad Calibration 3](#_Toc168519887)

[Figure 4: Linear Interpolation Diagram for HTS221 Humidity Sensor 3](#_Toc168519888)

[Figure 5: I2C Open Collector Output 4](#_Toc168519889)

[Figure 6: UART Interconnection 5](#_Toc168519890)

[Figure 7: UART Transmission Frame 5](#_Toc168519891)

[Figure 8: LIS3MDL Interface Structure 7](#_Toc168519892)

## Code Segments

[Code Segment 1: LIS2MDL\_HandleTypeDef 7](#_Toc168519922)

## Tables

[Table 1: I2C Transmission Example 4](#_Toc168519923)

1. I2C is also often called Two Wire Interface (TWI) due to its former patent copyright [↑](#footnote-ref-1)