1 Introduction

In this lab, you will write a software-based "bit banged" SPI implementation, which will read and write data to a simulated peripheral.

Part	Due Date
Code	Apr. 03

Figure 1: Table of due dates for each part.

Contents

2 How to Submit

When your code is ready to turn in, please submit only your `main.cpp` file via Moodle. Note that you should write all code that you need within `main.cpp`. As before, your code must compile to receive any credit.

3 Simulated Sensor

The peripheral devices you will be interfacing with during this lab is a simulated "sensor". We leave it up to your imagination as to what it senses. This device works very similarly to the LIS3DH, but is greatly simplified. It has only two registers, which are both read-only.

The first register is `WHO_AM_I`, which is at address `0x0F`. When read, it will always return `0x34`.

The second register is `DATA`, which is at address `0x10`. This register will return the sensor "reading". Note that the sensor readings are actually hard-coded, however keep in mind that they will be modified to different hard-coded values during grading.

Much like the LIS3DH, a register is read by transmitting it's 8-bit address via MOSI. The 8-bit register contents will then be returned via MISO.

3.1 Program Requirements

Your program should first query the `WHO_AM_I` register and verify it returns the correct value. If not, then your program should display an error message (on standard out) and exit.

If the `WHO_AM_I` register can be verified, then your program should read the `DATA` register **four** times, printing the raw value received each time to standard out, then exit.

3.2 Grading

Your code will be inspected for style and correctness by a human reader. This aspect of grading will be fairly lenient and mostly for the purpose of giving you useful feedback. However you may still lose points for egregious stylistic problems, or failing to write code that clearly attempts to solve the problem at hand.

Your code will also be run against the same simulated sensor as you are given in the project skeleton, however the hard-coded sensor "readings" will be changed to different values.

Additionally, your code will be run against a version of the simulated sensor which is defective, and reports an incorrect value when the `WHO_AM_I` register is read.

The correctness of your code will be judged by considering both your printouts, as well as the values sigrok detects on the SPI bus. Note that the error value `0xEE` or null `0x00` can appear on the bus as the "dummy value" which one side or the other "sends" while receiving data. This is not a cause for concern.

If your code does not compile and run on the CSCE linux lab computers, you will not receive credit.

4 Rubric

- 10 points - Code style
- 10 points - Correct data is transmitted
- 10 points - Correct sensor readings are received and printed
- 10 points - Correct checking of `WHO_AM_I` register

Maximum number of points possible: 40.

Keep in mind that some items not listed on the rubric may cause you to loose points, including cheating, submitting code which is inconsistent with what you have demonstrating, plagiarizing code or reflection content, etc.