18-847 Lab 1: PowerDue MicroController Introduction

Jawahar Chigurupati
Carnegie Mellon University, Silicon Valley
November 6th, 2019

Abstract— **This report details our introduction to the I2C and SPI protocols for serial communication in embedded systems. We interfaced with a magnetometer and accelerometer and tested the connection with both protocols. We also were required to read relevant datasheets and implement empty functions for device drivers to complete this lab. In part 3, the power consumption of these embedded devices in parts 1 and 2 were measured.**

I. INTRODUCTION

The I2C and SPI protocols were used to gather sensor data from the magnetometer. The transactions were observed on an oscilloscope and the energy consumption of the sensor was also measured via the PowerDue's instrumentation port. We also had to parse sensor data sheets and install required device libraries for the LSM303C breakout board. All code for this lab was written in Arduino.

I2C was a Philips created standard to transmit data between circuits on the same board. Just two wires – SDA (data) and SCL (clock) lines. These lines are shared between the processor (”master”) and multiple other devices (”slaves”). Each device will be in standby until the master device sends a 7-bit address which corresponds to a particular device. Different sequences of the clock and data lines indicates when data is being read or written.

The SPI protocol consists of three lines – SCK (clock signal), MOSI (Master out, slave in), and MISO (Master in, slave out). The master will send clock readings along SCK and data along MOSI. SPI uses a read/write bit and then a 7 bit address to choose the correct device. The last byte would be the data from the instrument. A sample SPI Write is shown in Figure 1.

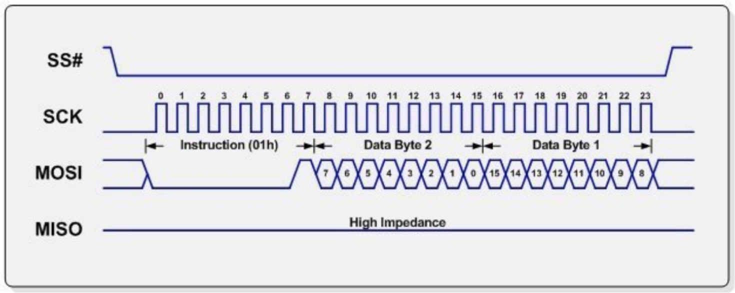

Fig. 1: A sample SPI Write transmission.

II. APPROACH

For this lab, there were 3 main parts.

1) Getting sensor data with the I2C protocol.

We used an I2C and SPI enabled breakout board that housed a 3 axis magnetometer and accelerometer. An additional Arduino library for the LSM303C breakout board was downloaded. The PowerDue was using the Wire interface – this was changed to Wire1. In this section, we modified the template file to utilize FreeRTOS threads to read the magnetometer values and output them to the Serial Monitor. A secondary task involved implementing a WhoAmI function that returned the device's ID number. There was an existing function in the drive files – we simply needed to read from the correct register and then compare those values.

2) Getting sensor data with the SPI Protocol

The second part switched to the SPI protocol. The PowerDue was integrated with the FXOS8700CQ magnetometer. There was a PowerDue SPI library but no existing Arduino library was available for the FXOS8700CQ magenetometer. We had to implement some of these methods ourselves within the supplied header and cpp files. These methods involved reading and writing to a register, reading magnetometer data, setting standby and active modes, and the init and whoAmI functions.

3) Compare processor energy draw against both protocols.

In the final section, we utilized the PowerDue's instrumentation port to measure the energy draw from the processor between the two protocols. Each protocol read a register 10,000 times in a loop and watched the readings on the Powerscope. The efficiency of each protocol was calculated based on the total power draw over the start and end times.

To convert the voltage to current, we use the formula shown in Figure 2.

$$I_{meas} = \frac{v_{meas}}{A * R_{sense}}$$

Fig. 2: A is 25, and R_sense is 1.33 ohms for the processor.

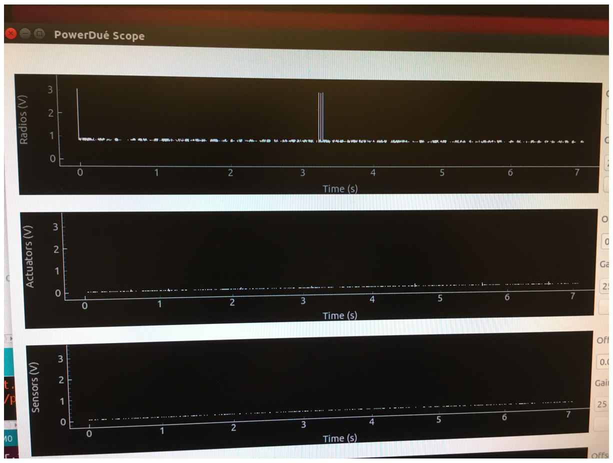

Fig. 3: Sample Pyscope output for I2C.

III. Results

Answers to Report Questions - Part 1 and 2

The default I2C clock speed was 400kHz (this is also the maximum I2C speed). The read frequency for the data was set to 800Hz. Each read contained 4 bytes - the first byte was for the device address, 1 byte containing the device register, and 2 bytes to represent the data.

The SPI protocol operates in mode 0 and transfers data such that the most significant bit, (MSB) is first. There are three steps. The first step contains a 7-bit register address + 1 Read/Write bit. The second step will have 1 byte of data with the 8th address bit.

IV. Conclusions

SPI was the superior protocol in terms of power consumption – this is partly due to not needing pull-up resistors like I2C needs. Power consumption would be limited to the switching of transistors and the minimum 4 wires required for the protocol. SPI in general will require more wires for each ”master-slave” relationship along with an additional line for the chip-select.

The link to the github repository is: https://github.com/cjawahar/wssaLab3