Lab 2 Report

Nathan Grant & Jiss Xavier

I. INTRODUCTION

This lab focused on using SPI and I2C communication, as well as learning to use the Saleae logic probe to observe the signals being transferred through these communication protocols. In order to achieve our final goal of moving a ball on the OLED screen based on the accelerometer values of our CCS3200 Launchpad, we implemented the OLED interface using SPI communication, and then transmitted the accelerometer values to the Launchpad using I2C.

II. BACKGROUND

While a lot of the software and hardware used in this lab are similar and carried over from the first lab, there are a few new concepts and tools used in this lab that are new.

In terms of concepts, we looked at the Master-Slave relationship commonly found in embedded systems. It is a model of communication in which one device has control over all the functionalities and utilities of another device or devices. In our case, the LaunchPad acts as the Master to the OLED display controlling what gets displayed. For this lab, we looked at two different types of interface buses. The first is SPI, a serial peripheral interface, which is an interface bus used to transmit data between devices. The SPI interface bus consists of clock and data lines. I^2C, an inter-integrated circuit, is a synchronous packet-switched serial communication bus. For our lab, we will be using I^2C to communicate with the onboard BMA222 accelerometer found on the LaunchPad.

In terms of hardware, we will once again primarily be utilizing Code Composer Studio to control the functionalities of the CCS LaunchPad with the help of terminal emulation software, such as PuTTY. We will be working with the Bosch BMA222 accelerometer found on the LaunchPad to obtain data to be used for motion-based applications. Additionally, we will be using the Adafruit OLED Breakout Board which is a small 128x128 colored display that can be wired to our LaunchPad to display different objects and images. Lastly, we will be using the Saleae Logic 8 along with associate Saleae Logic software which allows us to analyze the data passed through different interface buses, whether that be SPI or I^2C.

III. GOALS

Part 1: Interfacing to the OLED using the Serial Peripheral Interface (SPI)

The objective of the first portion of the lab is to interface our CCS3200 Launchpad with the newly provided OLED display. The OLED display and the LaunchPad exhibit a Master-Slave relationship. The LaunchPad behaves as the Master controlling the functionalities of the Slave in this case the OLED display. We begin by implementing an SPI demo project that interfaces our two LaunchPads to demonstrate the Master-Slave relationship. One board, the Master, is able to send strings that are received by the other board, the Slave.

Next, we replace the LaunchPad that behaves as the Slave with the OLED display. We wire the LaunchPad to the OLED display using the breadboard closely following the pin configuration table from the provided lab manual. Once the OLED is correctly wired to the LaunchPad, we are to execute sample test programs that show the different display functionalities of the OLED display. Additionally, we are to print the full character set, the string “Hello World”, and 8 different colored bands both horizontally and vertically to the display.

Once the OLED display is successfully interfaced and the printing functionalities are implemented and tested, we are to verify the SPI waveforms using the Saleae Logic software. We do this by replacing the OLED display with the Saleae

Logic 8 device. Using the SPI logic analyzer built into Saleae Logic software we are to analyze the SPI waveforms produced by our LaunchPad. We observe the ENABLE, DC, CLOCK, and MOSI waveforms and analyze and verify that they behave as expected.

Part 2: Using I^2C to communicate with the onboard BMA222 accelerometer

The objective of the second portion of the lab is to use I^2C to communicate with the LaunchPad's built-in accelerometer. Firstly, we implement the demo I^2C example to ensure that we are able to receive accelerometer data from our LaunchPad. Then, we verify the I^2C waveforms using the Logic 8 device. We observe the SCL and SDA waveforms on the Logic software.

Then, we build a simple application utilizing both the accelerometer of the LaunchPad as well as the OLED display. The application displays a small ball on the OLED display that can be moved around the display by pivoting or moving the LaunchPad about two axes. Motion on the x-axis should cause the ball to move left and right and motion on the y-axis should cause the ball to move up and down. The steeper the roll angle, the faster the ball should move across the display and the ball must hit a wall when the edge of the display is reached.

What did we learn?

From the first part of the lab, we learned the concept of the Master-Slave relationship in embedded systems and how they work. Then we utilized this relationship to interface an OLED display to our LaunchPad and display different functionalities on the display. In this process, we learned to wire an external device to our LaunchPad using jumper cables and a breadboard as well as to write code that would display different items on the display. Then, we learned how to use the Saleae Logic 8 device and the associated software to observe, analyze, and verify waveforms, SPI and I^2C, generated by our LaunchPad when executing our different programs. Lastly, we learned how to use I^2C to communicate with our LaunchPad's accelerometer and learned to utilize that data to build an application that exhibited an object whose behavior depended on that data.

IV. METHODS

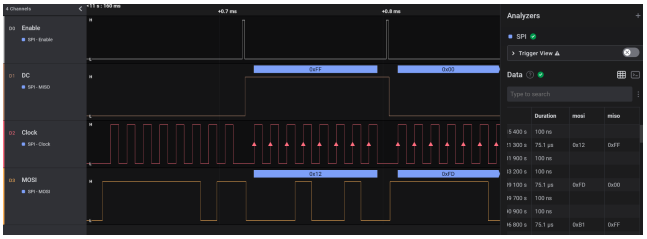
The first step of this lab was interfacing to the OLED using SPI communication protocol. We did this by running the SPI demo on two separate computers, each connected to a launchpad and with its own PuTTY window. We set the appropriate master-slave values and connected the wires in the appropriate MOSI pins, and then verified that we were able to communicate and send messages back and forth between the two launchpads and display them on the terminal. We then proceeded to configure the appropriate SPI pins on the LaunchPad using the TI PinMux Tool, and made the following connections: Launchpad MOSI to OLED SI, Launchpad SCLK to OLED CL, 3 GPIO outputs each to OLED DC, R, and OC, 3.3V to Vin on OLED, and GND to GND. Next we wrote a loop to go through and print all the characters in the font file, print the string "Hello World", and call all of the test functions to verify the OLED screen was working properly. The Last step in part 1 was connecting all these wires to the channels of the Saleae logic probe instead of the OLED and verifying that these signals displayed on the logic analyzer are what we expected based on our code. This allows us to learn how to see what our code is actually outputting for future debugging.

In part 2 we began by understanding and running the I2C demo program, and used this program to get the X and Y values of our board using readreg 0x18 0x3 1 and 0x18 0x5 1 respectively. This allowed us to turn the board and output the values from the accelerometer using I2C, so we could see how different angles of the board output different X and Y values. After this we connected our I2C SCL and SDA pins to the logic analyzer to once again verify our output values were as expected.

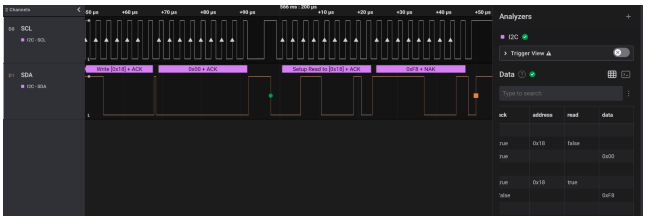
The last part of this lab was using these accelerometer values to control a small ball on the OLED screen. We read the X and Y values the same way we did in the I2C demo, and then get the new X and Y values. Before updating the position of the ball using the fillcircle() function, we fill the balls previous location with a black circle, and then calculate the new position by adding the old position

plus (change in position / 3). We chose to divide the accelerometer value by 3 to desensitize the board a little bit, allowing us to have easier control over the ball's movement. Lastly, we made sure to limit the balls X and Y positioning so that it did not exceed the values of the OLED, by checking that the origin + or - radius values were in between 0 and 128. The following are the screenshots of our waveforms from this lab.

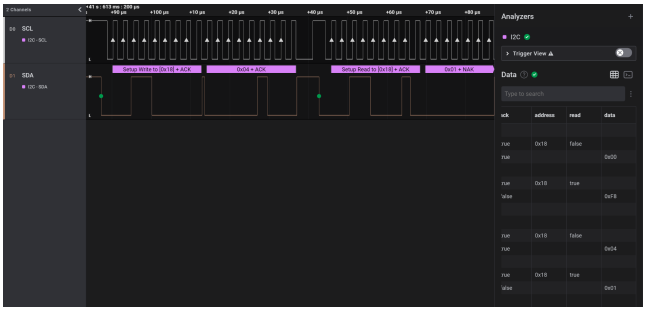
SPI:



I2C X Coordinate:



I2C Y Coordinate:



V. CONCLUSION

The second lab was significantly more challenging and time-consuming than our first lab which was to be expected. We were able to quickly interface our boards and execute the demo SPI program for Part 1.1. We then faced some unforeseen challenges that made this lab more difficult than we had initially anticipated. We found ourselves stuck on getting the OLED to first turn on and initialize in Part 1.2 of the lab. The fix to this problem turned out to be that we had to clear the receive register for SPI after we had PUT data. This small issue took hours of debugging and help from the TA. However, having gone through the code line by line, we were able to better understand the code which meant the next portion of the lab, implementing different display functionalities, turned out to be easy. We were able to then quickly verify the SPI waveforms using the Logic SPI analyzer and the waveforms helped consolidate what was happening. For Part 2, we faced some minor setbacks in getting our demo I^2C project to work. This was due to our terminal emulator which caused us to shift to PuTTY. Having resolved this we were once again able to use the Saleae Logic Analyzer to verify the I^2C waveforms and move on to the last portion of the lab. The creation of the ball application took us some time but by dedicating the time and utilizing resources, we were able to successfully complete the application. As a whole, this lab came with a large learning curve but having gone through it, we walk away with a lot of lessons learned and knowledge regarding our subject matter. Therefore, this lab was incredibly rewarding to complete.