EEC 172 Lab 2 Report

Alejandro Torres
almtorres@ucdavis.edu

ABSTRACT

In this paper, I go into detail on how I interfaced the SPI with the CC3200 LaunchPad to display specific patterns on an OLED screen. To improve on the design I used an I2C to communicate with the Bosch BMA222 acceleration sensor on the CC3200 LaunchPad.

General Terms

Serial Peripheral Interface means SPI. Color Organic Light-emitting Diode means OLED. Bosch BMA222 acceleration sensor means BMA.

1. INTRODUCTION

This lab was a great opportunity to get familiar with reading datasheets to interface to the OLED screen using the SPI. After properly connecting the OLED onto the CC3200 LaunchPad I was able to complete part 1 of the lab. Part 1 consisted of calling functions that were provided to display a set of specific patterns on the OLED screen.

I was then able to add onto the existing design by using an I2C to communicate with the BMA . The BMA would return values in respect to the x and y axis of the sensor. I then took the readings returned from the BMA to move a ball on the OLED screen to its respective position.

2. Procedures for Part 1

This section will discuss the steps I took in each part of the lab and the outcome.

2.1

The first part of the lab is to run an example SPI program that connects two CC3200 LaunchPads. I connected the SPI master which would communicate with the peripheral (slave) device (CC3200 LaunchPad) to one computer. I then connected the SPI slave to another computer. To ensure proper communication I ensured to properly connect the two CC3200 LaunchPads. I was then able to open the putty on my computer to visually see the SPI master write to the SPI slave.

2.2

For the next part of the lab, I configured GPIO outputs to act as signals to control the Data/Command, Chip Select, and reset the OLED. I then properly connected the OLED to the CC3200 LaunchPad using the respective configured pins. Afterward, I studied the SPI demo to properly write the low-level functions WriteData() and WriteCommand() that used the SPI port to write bytes to the OLED.

Once the OLED screen was initialized, I used the API's provided by the ADAFruit to properly display a specific set of patterns. I started by printing each character from a provided file and made sure to stay within the 128x128 bounds of the OLED. After each pattern was displayed, I added a delay and reset the OLED screen before displaying the next pattern. I used fillScreen(0) provided by the ADAFruit API to fill the entire 128x128 OLED screen with a pixel value of 0, corresponding to the color black.

2.3

In the next part of the lab, I was able to verify the command/data being sent to the OLED using a Saleae logic (logic analyzer). I configured the Saleae logic to capture SPI waveforms and connected the probes to the DC, OC, CLK, and MOSI pins I had configured on the CC3200 Launchpad. I was then able to capture the first initial lines of code in the AdafruitInit().

I also verified the I2C waveforms that initiated the x and y axis readings from the register. In Figure 1 and 2, I could see the base address of the BMA written to I2C. Then, in Figure 1, the register value 0x03 was read to return the readings for the x axis, and in Figure 2, the register value 0x05 was read.

3. Procedures for Part 2

In the following section of the lab, I interfaced with the BMA accelerometer on the CC3200 LaunchPad to retrieve y and x axis values. My first step was to retrieve the values from the register on the BMA device, requiring the base address of the BMA, the address of the register, and the length of bytes to be read from the register. Finding these values yielded various tilt angles from the BMA device.

I then printed these values to determine the respective values when the board was turned up/down or turned left/right. I was then able to write code that would initially fill the OLED screen black before placing the initial position of the ball on the screen (center). The ball on the OLED would move, and the speed of the ball was dependent on how fast the tilt angle was changing.

4. Problems and Solutions

4.1 Part 1

In Part 1, the main issue I encountered was getting the data printed on the OLED. Initially, I could see the data in VS Code, but nothing would happen on the display. After troubleshooting, I realized that I had not added the .mux file to my project directory. However, the problem still persisted. I then realized that I had not configured the SPI pins properly. After correcting this, the OLED finally displayed the data.

4.2 Part 2

In Part 2, the main issue I encountered was setting boundaries so the ball would not roll off the screen. The glitch that was occurring was once the ball reached the boundary, it would get stuck there and not move elsewhere. I attempted to create bounds by defining the left/top regions to be bounded by a value of 4 and right/bottom regions to be bounded by 125. However, my BMA readings weren't consistent with these bounds. I had a difficult time properly bounding both axes. Additionally, when displaying the waveform, I was only getting a brief vertical line. After troubleshooting, the issue was determined to be that I needed to be zoomed in a lot to be able to see the waveform.

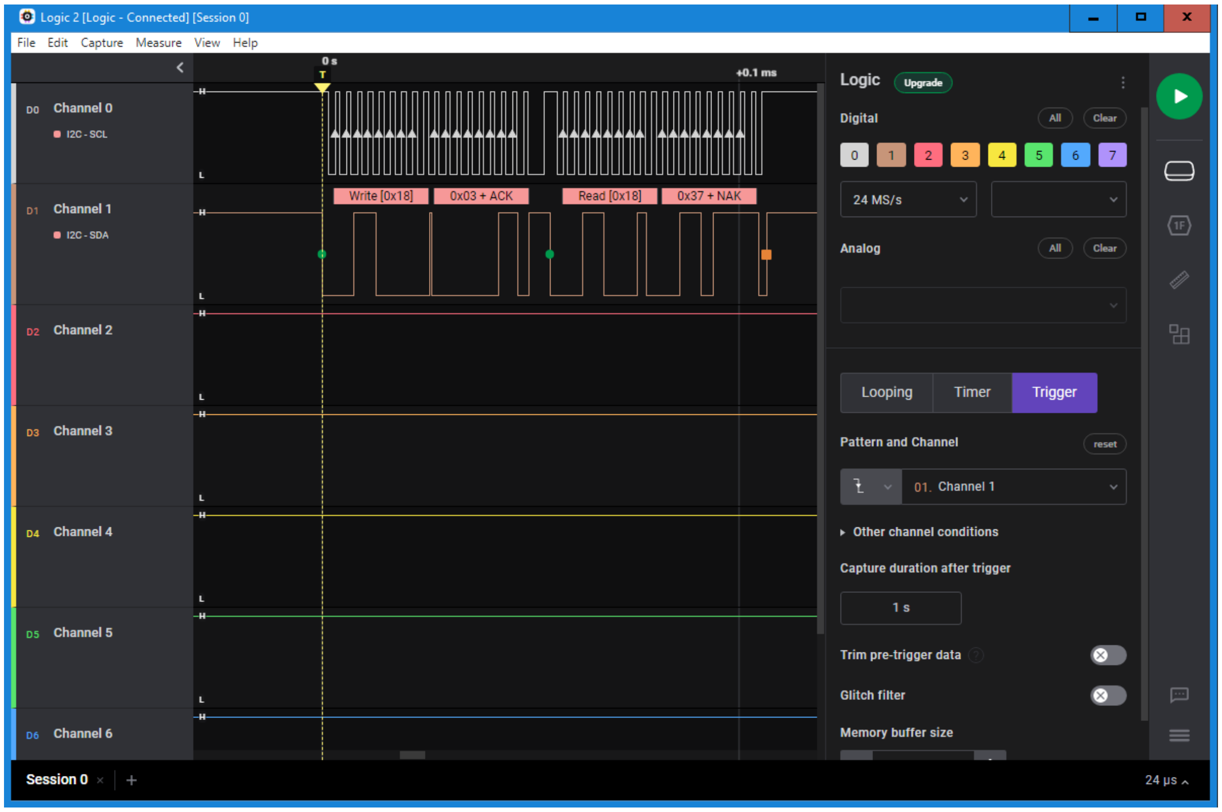

Figure 1: I2C Waveform of initializing x-axis

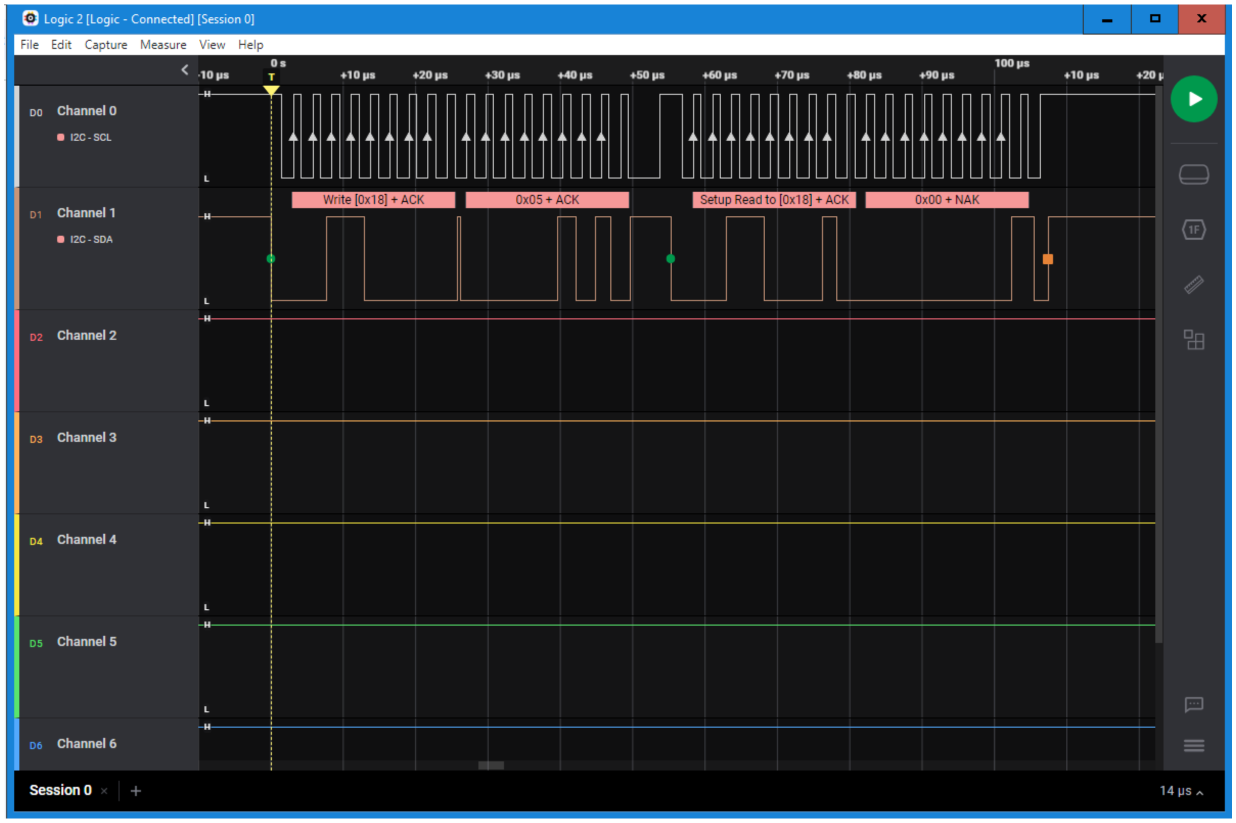

Figure 2: I2C Waveform of initializing y-axis.