LAB 2 Report

Honglei Liu

Bingxuan Ying

Intro:

In this lab, we are asked to use given SPI and I²C sample project to understand how to communicate between circuit via SPI and I²C. In addition to that, we will also learn to use Saleae logic component to measure and analyze the waveform. As for programming part, instead of building project that allows the Tiva Launchpad to communicate with either the LED screen or the other Tiva Launchpad, we are also asked to design a small accelerating ball game on the screen.

Procedure:

1. Import and build the spi\_demo on CC3200 board. In order to finish part one, we will modify our code slightly after we finish looking at the code for slave-master SPI example.
2. The first change we made in part 1 is to change the Master-Slave mode from MASTER\_MODE from 1 to 0. Then we use Ti Pinmux tools to configure some usable pins according to the document provided on Canvas. We set three customized buttons to our OLED signals-DC, RESET and OLEDCS.

Next, we import the graphical library provided by Adafruit to our project directory and implemented the WriteData() and WriteCommand() functions in Adafruit\_OLED.c. .In our own code, we turned DC to high and OLEDCS to low, send and read data and CS to high to finish sending data.

Then, in our main.c file, we wanted to realize all the functions that are listed in lab manual. So we followed the instructions and used several built functions to realize all the listed requirements.

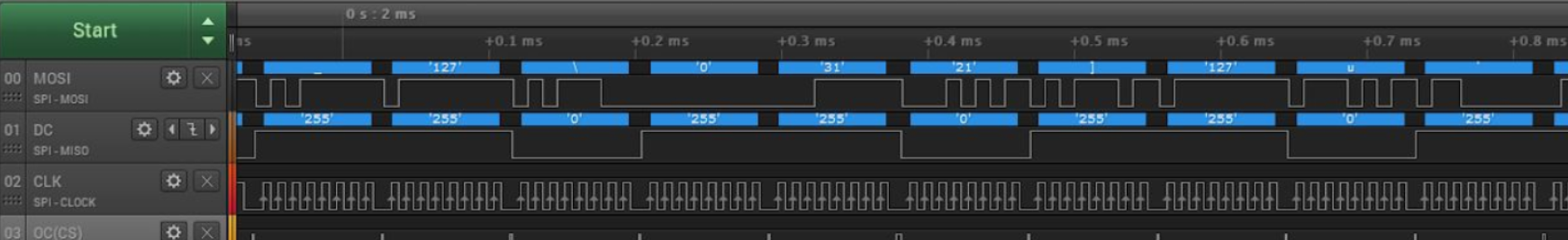
1. In additions to the code that is running on OLED, we also wanted to connect our Saleae Logic to the cable set that connecting from OLED to Tiva Launchpad. Then, we want to launch our main to see the waveform through cables.
2. For part 2, we import i2c\_demo sample project and debug it. We used the terminal to figure out the accelerator X, Y and Z direction and the initial value since we already put the OLED chip plus the Tiva Launchpad on a flat table.

Before we construct the rolling ball application, we want to first use the Saleae Logic to find out the how actually the different direction components be like in waveform.

For the application code, we first design several variables: X and Y position, speed in X and Y direction and acquired X and Y value from accelerator. We divided the X and Y value from accelerator by 6 to reduce the speed to a reasonable magnitude. Since our ball cannot exceed the boundary of the screen, we set two if-else conditions to limit the X and Y position value. And at the beginning of each loop, we erase the previous ball position and draw the new position.

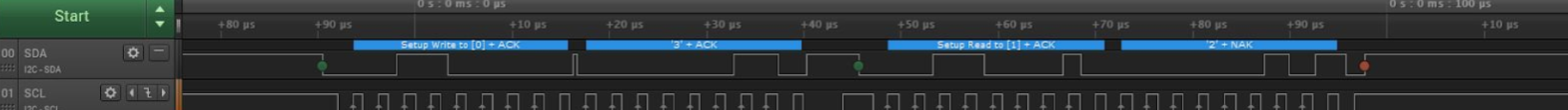
Discussion

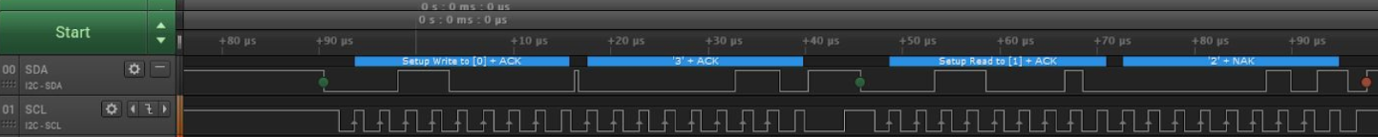
Screenshot from Part 1



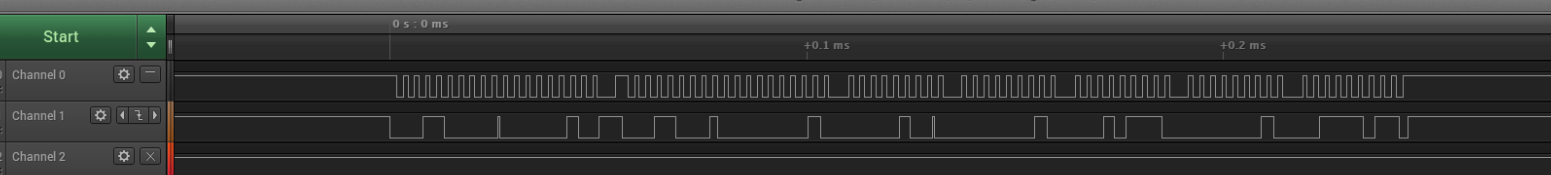
Since we consider our Tiva Launchpad as our Master and OLED as Slave. When OLEDCS and DC were low, a command is sent and when DC is high, the Slave is writing the data after receiving the command This is both the theoretical logic and what we could get from the waveform. Command length is 8 cycles so every 8 cycles CS and DC toggles. The data, on the other hand, will process the entire data command in two of the eight cycles. We could see from the screen shot that when CS and DC are low, a command which is 8 cycles long is presented and then DC toggles to high to let the data pas. As for MOSI, the waveform is simply representing the data being transmitted.

Screenshot from Part 2





From the two screen shots above, interpreting with the data we received from Tera Term, we could understand the waveform much better. As we can see from the screen shot, the second term from the labelled wave are different. The X direction I2C waveform printed 3+ACK and Y direction I2C waveform printed 5+ACK which are correctly matching that the register 0x3 is X direction and 0x5 is Y direction. As for the procedure. These two directions’ screen shots are similar since the first step is the program passing the device address through I2C to be read. Then the second step pass the value of register address corresponding to specific direction. Next is to pass the data length to be read. And the last step is to show the result of the read reg command.



We have faced two main challenge, the first one is when we worked on part 1, we spent tons of time trying to figure out the reason OLED doesn’t light up. Since at that time, we didn’t check the document on Canvas but only the MOSI, MISO etc. connections on Wiki, we used one wrong pin for our GPIO. The second one is from part 2 the last application, we spent time in figuring out what type of data should we choose to receive the data from accelerator and how to detect the sign of the speed since the speed value is always a given 2’s compliment number.