Garrett Tashiro

CSE 474

Summer 2021

Lab 1

Procedure 1:

I approached the assigned task by first looking at what I needed. I thought about any ports that were needed for that task, and all the registers. I wrote down any registers I would be needing, as well as the port, so I could look on the data sheet for the bases and offsets.

For task 1 a, I started by having each LED on the Tiva light up individually to check for correctness with the registers and hex values for pins. After that, I broke the task into a slightly smaller task by having two LED's light up with a delay in between and turned them off. Once I was able to turn on and off two LED's in a sequence, I wrote the code to turn all the LED's on in a sequence, and stay on during the sequence, then had them turn off all at once.

For task 1 b I brought in port J with the GPIOPUR register for the switches, so I got the base and offset for all the registers. Once I had the registers setup in the header file, I set all of the registers for the ports and pins. After that, I thought about all the cases in which the two switches could be pressed. I wrote down pseudocode for no switches pressed is no LED's on, SW1 pressed lit up the first LED, SW2 lit up the second LED, and both switches pressed lit up both LED's. From the pseudocode I introduced the correct logic needed for each of the switches and LED's.

In task 2 I started by testing the code that was given to us. I copied over all of the "External LED code" first. I built my circuit using the inverter and an LED. I looked at the data sheet for the inverter to be sure I was using the correct pins on the chip. After I had the external LED working, I tested out the "External Button Code". Once I had the external button and LED working, I change the ports to port L. I then sketched out a simple finite state machine diagram that can be seen in Figure 1 at the end of Procedure 1. With the diagram I figured out what states I was going to need, and what actions would come from each state transition. I made two states to start and tested that the transitions and actions worked as expected. From there, I added in the other state transitions and actions.

In task 1, I feel as if my code was not be the best organized, I was just trying to get everything to work so I wrote all the code in the main which could be moved into separate functions. This was my first-time coding with C, so I just wanted things to work. The organization for all of task 1 is not great, and I see that. Task 2 was better organized, and had functions for everything. The main for task 2 has initialization, a while loop, and calling the state machine in it. I know I can improve upon what I have still, but there is a big difference in organization between task 1 to task 2.

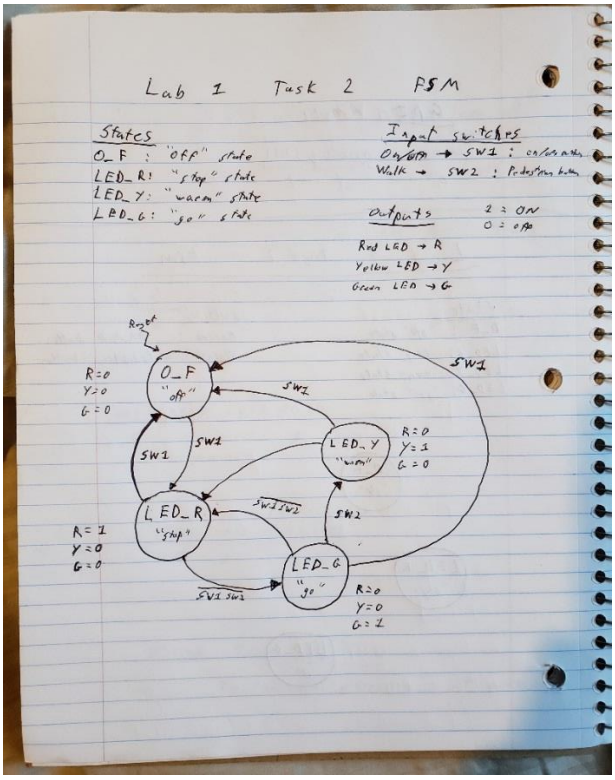

Fig. 1: Finite State Machine

Procedure 2:

For task 1 a, the obtained results were the LED's lighting up from left to right on the board. The lights lit up in a sequential order, and stayed on until all four LED's were lit up, then they turned off one at a time in the same order in which they lit up. This lighting sequence was in an infinite while loop, so it would continue. Task 1 b used two of the LED's from task 1 a that were associated with Port N. The two onboard switches were used for this task as well. If you press in SW1 the LED labeled D0 would light up. If SW2 was pressed the LED labeled D1 would light up. If you pressed SW1 and SW2 at the same time then LED's D0 and D1 would light up. If you released either SW then the corresponding LED would turn off. If no SW is pressed, no LED's will turn on. For task 2, if the button associated with PL0 is pressed after reset, the red LED will light up for about half a second. The red LED will turn off and the green LED will turn on. After about half a second the green LED will turn off and the red LED will turn back on. If no buttons are pressed, this process will continue. If the button associated with PL1 is pressed while the red LED is on nothing will happen, but if pressed with the green LED on then the green light will turn off and the yellow LED will turn on right after. The yellow LED will stay on for about half a second then turn off and the red LED will turn on. This will go back to the loop from red to green and then back again. If the button associated to PL0 is pressed when any of the LED's are on, then all LED's will turn off. Once that button is pressed again the red LED will light up and the state changes will happen again.

Procedure 3:

The feedback I have for the report is that it was fairly difficult coming into this with little to no knowledge about C. Also having no knowledge about an embedded system and how to make things work was difficult. Trying to figure out registers with their bases, offsets, and what exactly was being changed was a challenge.