Kiana Peterson and Nattapon Oonlmaom
CSE 474: Intro to Embedded Systems
October 19, 2023
Lab 1: GPIO and FSMs

Procedure

The purpose of this lab is to investigate and understand General-Purpose In/Out (GPIO) and how Finite-State Machines (FSMs) are implemented in C using IAR Embedded Workbench Integrated Development Environment (IDE) on TIVA LaunchPad.

To get started: we installed IAR and initialized a new project. Then, we configured the TIVA board and its GPIO to turn on LED4 using the code given from the Lab 1 instruction. LED4 did not show up when the code was downloaded to the board, so we troubleshooted it using the given "TIVA Board Troubleshooting" file. It turns out that we did not need to reset the board, but rather, we needed to switch the USB port. The problem was one of the USB ports on our computer was causing the error, so we resolved the issue by switching to use another port. Then, we interacted with user switches by inputting PJ0 and PJ1 using GPIODIR. After we made sure the board works properly, we carried on to Task 1.

Task 1a: This task was to change the main program and the header file to turn on and off all of the LEDs in a sequential pattern. We used a while loop to continuously turn on and off one LED at a time to display patterns, and also create approximately 0.3 seconds of a delay between changing LEDs. To approach this, we watched the lecture video by the Professor, following concisely on how to read the datasheet. Then, we analyzed the schematic of the board provided to find out how to represent the LED pins in C code. We decided to have 3 sets of code: one to set up ports N and F for LEDs as outputs (task1aSetup.c), one to perform a single cycle of LEDs from LED4 to LED1 (task1a.c), and then main.c to run the programs (setting up the ports and then running task1a.c in a loop).

Task 1b: This task was to create a new program that turns on LEDs using the user switches (buttons), turning on LED1 when SW1 is pressed and LED2 when SW2 is pressed. To approach this, we analyzed the datasheet to get the various register addresses for each port. Then, we decided to separate the code into 3 parts in a similar fashion to Task 1a: one to set port N as an output for the LEDs and port J as an input for the switches, one to light up each LED when their corresponding button is pressed (while the header file handles active low), and then the main file to run the programs (setting up the ports and then running task1b.c in a loop).

Task 2: For this task, we were to make a traffic light controller by designing a FSM for the light system with an on/off button.When on, the system always starts in the go state. Then there is a pedestrian button, where if pressed during stop, nothing changes. If pressed during go, it turns to the warn state, then changes to stop. In any state, if the power button is pressed at the same time as the pedestrian button, the result would be the same as if only the power button was pressed. The states are indicated by the LED color; red is stop, yellow is warn, green is go, and none is off. To approach this, we took the code the professor gave in the lab specs and the information datasheet to properly set up all of the LEDs and switches. Then, we created the main, which calls on the initialization functions and continuously runs the FSM. To create the FSM, we came up with the state diagram (Figure 1) by going through the requirements step by step and visualizing a real-life traffic light system. The FSM uses the helper functions given by the lab document to do the following: turn on LEDs, turn off LEDs, and get the value of the button pressed. As for the hardware, we followed the schematics from the lab

document (Figure 3 and Figure 4) combined with the inverter datasheet (Figure 2) and essentially multiplied the LEDs by 3 and the switches by 2. We also used the port assignments shown in Figure 1 and connected the whole thing together.

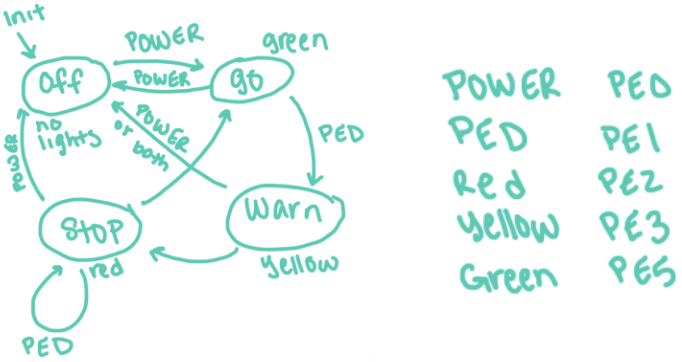

Figure 1: State Diagram of Task 2

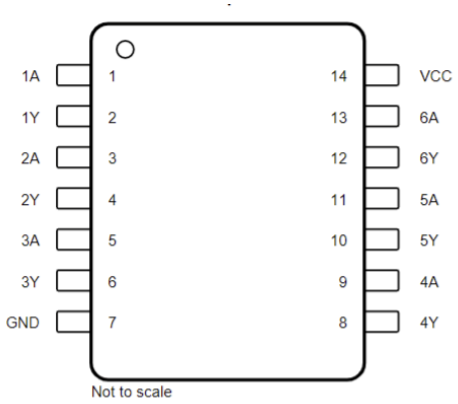

Figure 2: SNx4LS06 Inverter Pinout Diagram

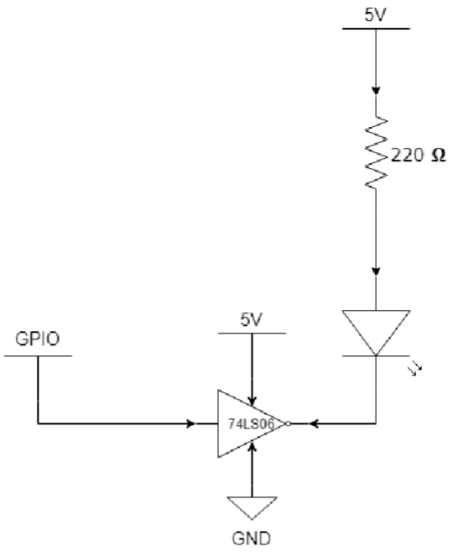

Figure 3: Single LED Circuit Schematic

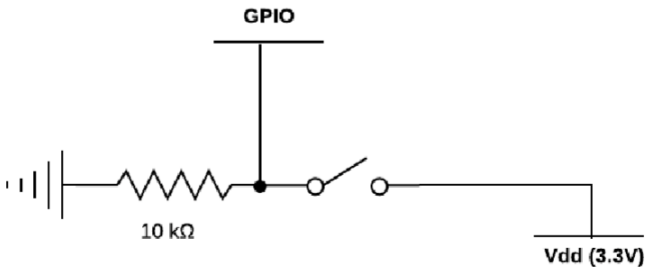

Figure 4: Single Switch Circuit Schematic

Results

Task 1a: The final product has 2 LEDS moving from LED4 to LED1, and when reset is pressed, the sequential pattern starts from the beginning again. The result accomplished what was asked as explained under the procedure.

Task 1b: The final product for task 1b shows when SW1 is pressed, LED1 lights up, and when SW2 is pressed, LED2 lights up. When both switches are pressed, both LEDs light up. Lastly, when none is pressed, nothing lights up. This successfully completed what was asked and explained under the procedure.

Task 2: As for the Task 2 final product, when the power button is pressed (the button connected to PE0), the system always starts in the Go state where the green LED (connected to PE5) lights up. Then, when the pedestrian button is pressed (the button connected to PE1), if pressed during Stop, nothing changes so the red LED (connected to PE2) stays lit. If pressed during Go when the green LED is lit, it turns to the Warn state causing the yellow LED (connected to PE3) to light up. The state automatically changes to red, transitioning to the Stop state as required and explained under the procedure. If at any point, the power button is pressed (even when pressed with the pedestrian button), the system shuts off and all LEDs turn off in the Off state.