Jordan Murti

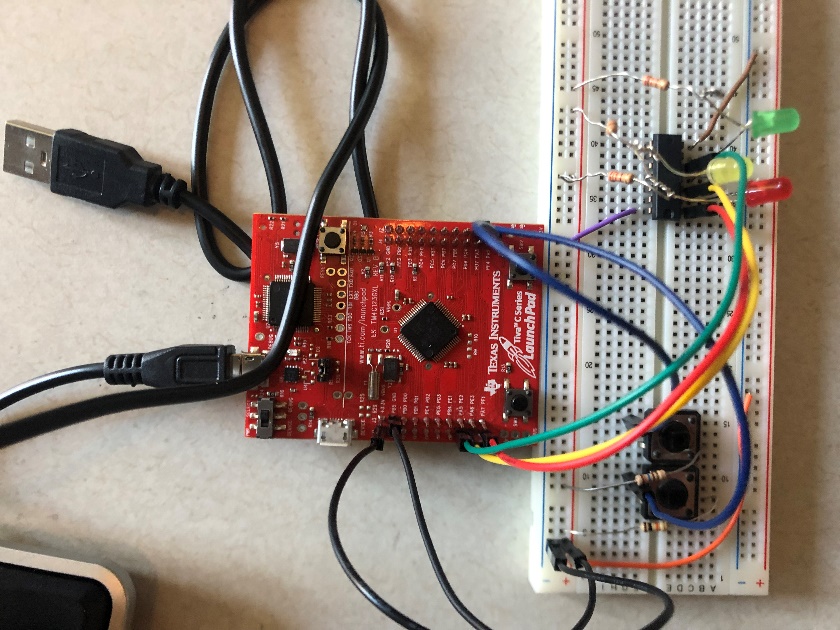
EE 474

Lab 1

**Procedure**

For part one of the lab I used mostly the notes and a little bit of the data sheet to initialize the required registers.. For the sequence of lighting all RGB, the notes told me what I needed to know for the Red LED so I just guessed the blue one was the bit to left, and green one more. It turned out to work. To iterate through each I figured counting through the bit sequence between 2 to 14 would cover all possible combinations of 000 through 111 (since the last bit didn’t matter to be set).

For part two of the lab I used more of the datasheet and also I used the lab sheet itself to examine the location of the ports connected to the RGB LED’s and the User Switches. Using a case statement seemed to yield the most consistent results.

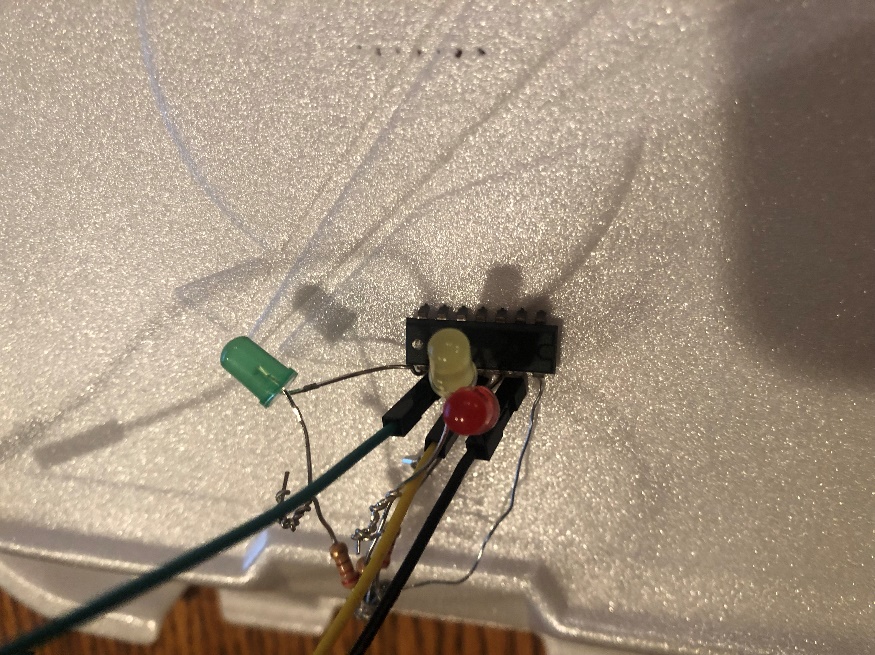
For part three was the big guns. I defined four states, OFF, STOP,GO, and WAIT. In the main while loop the program has an if statement for each possible state. At the beginning it changes the state variable to the next state then runs a for loop. This loop can be interrupted and in the interrupt will also change the next state from the default next state. For the Go state the interrupts are the Pedestrian button and the Off button. For initialization I used the datasheet to get all the variables shown in the lab example use. I modified the button output so that it inputs just a 1 or 0 instead of 0x4 or 0x8. I sticked to the proper convention for porta, because my plan was to use the lab’s given initialization until I realized it would be far more simple to just initialize everything all at once since there wasn’t any difference other than defining input and output differently. 

**Results**

In the end after some tweaks, all tasks were fulfilled as required by the lab. For an unknown reason the first task consistently requires to be loaded twice for mine and many others. The counting cycle through colors and the traffic light work well with both buttons.

**Problems Encountered**

My kit didn’t include a breadboard so at first for the traffic portion I tried doing it with Styrofoam, until I realized that definitely would work. Luckily a friend of mine from the class had a spare breadboard. Getting all the necessary things for button input was kind of weird, but luckily with the help of the datasheet and youtube I figured out unlocking. For some time all of the lights were on but that was because I forgot to actually run my initializer function in main. At one pointing the timing was a bi odd but arguably could have made some sweet beats, but nonetheless I discovered prints can disrupt timing a lot. All in all it probably took around 4-6 hours to complete the entire lab.

****

**Appendix**

**C file:**

1. #include "lab1p1header.h"
2. #include <stdint.h>
3. //#define RED 0x2
4. **int** j;
5. unsigned **long** COLOR = 0x2;
6. #define GO 0x0
7. #define WAIT 0x1
8. #define STOP 0x2
9. #define OFF 0x3//for the state machine
10. **int** state;
12. ///////////////////////////////////////////////////////////////////////////////////////////////////////////////
14. **void** LED\_initALLLL(**void**)
15. {
16. **volatile** unsigned **long** delay;
17. SYSCTL\_RCGC2\_R |= 0x01; // activate clock for Port A//Do I really need this?Ill assume so
18. delay = SYSCTL\_RCGC2\_R; // allow time for clock to start//Do I really need this?
19. GPIO\_PORTA\_PCTL\_R &= ~0x00000F00; // regular GPIO//no idea what this does but i moved the F to the port A spot
20. GPIO\_PORTA\_AMSEL\_R &= ~0xEC; // disable analog function of ALL PA Used - 11101100
21. GPIO\_PORTA\_DIR\_R |= 0xE0; // set PA7-5 to output 1110000
22. GPIO\_PORTA\_AFSEL\_R &= ~0xEC; // regular port function
23. GPIO\_PORTA\_DEN\_R |= 0xEC; // enable digital on ALL PA USED
24. }


28. /////////////////////////////////////////////////Initially this was the plan but instead I just did one intialization as seen above
29. //void switch\_initSS(void)
30. //{
31. // volatile unsigned long delay;
32. //
33. // SYSCTL\_RCGC2\_R |= 0x00000001; // activate clock for Port A
34. //
35. // delay = SYSCTL\_RCGC2\_R; // allow time for clock to start
36. //
37. // GPIO\_PORTA\_AMSEL\_R &= ~0x04; // disable analog on PA2
38. // GPIO\_PORTA\_PCTL\_R &= ~0x00000F00; // PCTL GPIO on PA2
39. // GPIO\_PORTA\_DIR\_R &= ~0x04; // set PA2 to input
40. // GPIO\_PORTA\_AFSEL\_R &= ~0x04; // PA2 regular port function
41. // GPIO\_PORTA\_DEN\_R |= 0x04; // enable PA2 as digital port
42. //}
43. //
44. //
45. //
46. //void switch\_initPED(void)
47. //{
48. // volatile unsigned long delay;
49. //
50. // SYSCTL\_RCGC2\_R |= 0x00000001; // activate clock for Port A
51. //
52. // delay = SYSCTL\_RCGC2\_R; // allow time for clock to start
53. //
54. // GPIO\_PORTA\_AMSEL\_R &= ~0x08; // disable analog on PA3
55. // GPIO\_PORTA\_PCTL\_R &= ~0x0000F000; // PCTL GPIO on PA3
56. // GPIO\_PORTA\_DIR\_R &= ~0x08; // set PA3 to input
57. // GPIO\_PORTA\_AFSEL\_R &= ~0x08; // PA3 regular port function
58. // GPIO\_PORTA\_DEN\_R |= 0x08; // enable PA3 as digital port
59. //}
60. //
61. //void LED\_initG(void)
62. //{
63. // volatile unsigned long delay;
64. // SYSCTL\_RCGC2\_R |= 0x01; // activate clock for Port A
65. // delay = SYSCTL\_RCGC2\_R; // allow time for clock to start
66. // GPIO\_PORTA\_PCTL\_R &= ~0x00000F00; // regular GPIO
67. // GPIO\_PORTA\_AMSEL\_R &= ~0x20; // disable analog function of PA5
68. // GPIO\_PORTA\_DIR\_R |= 0x20; // set PA5 to output
69. // GPIO\_PORTA\_AFSEL\_R &= ~0x04; // regular port function
70. // GPIO\_PORTA\_DEN\_R |= 0x20; // enable digital output on PA5
71. //}
72. //
73. //
74. //void LED\_initY(void)
75. //{
76. // volatile unsigned long delay;
77. // SYSCTL\_RCGC2\_R |= 0x01; // activate clock for Port A
78. // delay = SYSCTL\_RCGC2\_R; // allow time for clock to start
79. // GPIO\_PORTA\_PCTL\_R &= ~0x00000F00; // regular GPIO
80. // GPIO\_PORTA\_AMSEL\_R &= ~0x40; // disable analog function of PA6
81. // GPIO\_PORTA\_DIR\_R |= 0x40; // set PA6 to output
82. // GPIO\_PORTA\_AFSEL\_R &= ~0x04; // regular port function
83. // GPIO\_PORTA\_DEN\_R |= 0x40; // enable digital output on PA6
84. //}
85. //// turn on LED connected to PA6
86. //
87. //
88. //void LED\_initR(void)
89. //{
90. // volatile unsigned long delay;
91. // SYSCTL\_RCGC2\_R |= 0x01; // activate clock for Port A
92. // delay = SYSCTL\_RCGC2\_R; // allow time for clock to start
93. // GPIO\_PORTA\_PCTL\_R &= ~0x00000F00; // regular GPIO
94. // GPIO\_PORTA\_AMSEL\_R &= ~0x80; // disable analog function of PA7
95. // GPIO\_PORTA\_DIR\_R |= 0x80; // set PA7 to output
96. // GPIO\_PORTA\_AFSEL\_R &= ~0x04; // regular port function
97. // GPIO\_PORTA\_DEN\_R |= 0x80; // enable digital output on PA7
98. //}

101. // turn on LED connected to PA7
102. **void** LED\_onR(**void**)
103. {
104. GPIO\_PORTA\_DATA\_R |= 0x80;
105. }
106. // turn off LED connected to PA7
107. **void** LED\_offR(**void**)
108. {
109. GPIO\_PORTA\_DATA\_R &= ~0x80;
110. }
112. **void** LED\_onY(**void**)
113. {
114. GPIO\_PORTA\_DATA\_R |= 0x40;
115. }
116. // turn off LED connected to PA6
117. **void** LED\_offY(**void**)
118. {
119. GPIO\_PORTA\_DATA\_R &= ~0x40;
120. }
121. // turn on LED connected to PA5
122. **void** LED\_onG(**void**)
123. {
124. GPIO\_PORTA\_DATA\_R |= 0x20;
125. }
126. // turn off LED connected to PA5
127. **void** LED\_offG(**void**)
128. {
129. GPIO\_PORTA\_DATA\_R &= ~0x20;
130. }

133. unsigned **long** switch\_inputPED(**void**)///slight modification of given code that outputs a 1 instead of a specific value
134. {
135. **if**((GPIO\_PORTA\_DATA\_R & 0x08) == 0x08)
136. **return** 0x1;
137. **else**
138. **return** 0x0;
140. }
142. unsigned **long** switch\_inputSS(**void**)
143. {
144. **if**((GPIO\_PORTA\_DATA\_R & 0x04) == 0x04) // 0x04 (pressed) or 0 (not pressed)
145. **return** 0x1;
146. **else**
147. **return** 0x0;
148. }

151. **void** GoLED(){
152. LED\_onG();
153. LED\_offY();
154. LED\_offR();
155. }
157. **void** WaitLED(){
158. LED\_offG();
159. LED\_onY();
160. LED\_offR();
161. }
163. **void** StopLED(){
164. LED\_offG();
165. LED\_offY();
166. LED\_onR();
167. }
169. **void** OffLED(){
170. LED\_offG();
171. LED\_offY();
172. LED\_offR();
173. }


177. ////////////////////////////////////////////////////////////////////////////////////////////////////////////
179. **int** main3(){////////////////////////////////////MAIN FOR THE 3RD LAB PART(THE NUMBER is swapped out for use )
180. LED\_initALLLL();
181. state = GO;

184. **while**(1){
185. **if**(state == GO){////////////////start
186. GoLED();
187. state = STOP;
188. **for** (j = 0; j < 500000; j++) {////only 250000 for two ifs
190. **if** (switch\_inputPED() == 0x1) {///////////////interupt if pedestrian light pressed
191. state = WAIT;
192. j = 5000000;
193. }

196. **if** (switch\_inputSS() == 0x1){////////////// interupt if OFF light pressed
197. state = OFF;
198. j = 5000000;
199. }
200. }

203. }/////////
205. **if**(state == WAIT){////////////////wait
206. WaitLED();
207. state = STOP;
208. **for** (j = 0; j < 500000; j++) {/////////500000 for 1 if
209. **if** (switch\_inputSS() == 0x1){
210. state = OFF;
211. j = 50000000;
212. }
213. }
215. }/////////
216. **if**(state == STOP){////////////////STOP
218. StopLED();
219. state = GO;
220. **for** (j = 0; j < 500000; j++) {/////////500000 for 1 if
221. **if** (switch\_inputSS() == 0x1){
222. state = OFF;
223. j = 50000000;
224. }
225. }
227. }/////////


231. **if**(state == OFF){////////////////OFF no for loop, infinit until switch is pressed
232. OffLED();
233. **if** (switch\_inputSS() == 0x0){
234. state = GO;
235. }
236. }

239. }

242. **return** 0;
243. }
245. //////////////////////////////////////////////////////////////////////////////////////////////////////////////////




251. **int** main()///////////////////////////////////////////////////LAB PART ONE
252. {
254. RCGCGPIO = PORT\_F; // enable Port F GPIO
255. PortF\_DIR = 0xE; // set ALL OF PF as output
256. PortF\_DEN = 0xE; // enable digital pin ALL PF
257. GPIO\_PORTF\_DATA\_R = COLOR; // set PF1 to 1 and the other port F pins to 0
259. **while**(1)
260. {
261. **if**(COLOR<0xE)//I figured the pins next to it were the other lights, turns out my lucky guess of left shifting was right,
262. COLOR++;//counting from 0010 to 1110 should cover all the values
263. **else**
264. COLOR=0x2;//then reset after 111
266. GPIO\_PORTF\_DATA\_R = COLOR;
268. **for** (j = 0; j < 1000000; j++) {}//delay
270. }
272. **return** 0;
273. }




279. ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////




285. **int** main2()////////////////////////////////////////////////////LAB PART TWO
286. {
288. RCGCGPIO = PORT\_F; // enable Port F GPIO
289. PortF\_DIR = 0xE; // set PF1 as output
290. PortF\_DEN = 0xFF; // enable digital pin PF1-F5
291. GPIO\_PORTF\_DATA\_R = 0x0; // set PF1 to 1 and the other port F pins to 0
292. GPIO\_LOCK = 0x4C4F434B;
293. GPIO\_CMT=0xFF;
294. GPIO\_PUR=0x11;
296. **while**(1)
297. {
298. GPIO\_PORTF\_DATA\_R = 0x0;
299. **switch**(GPIO\_PORTF\_DATA\_R & 0x11){//////Look at the last bits
300. **case** 0x01:
301. GPIO\_PORTF\_DATA\_R |= 0x2;///if button1 go green...or red... whichever it is
302. **break**;
303. **case** 0x10:
304. GPIO\_PORTF\_DATA\_R |= 0x8;///other button do other color
305. **break**;
306. **default**:
307. **break**;


311. }
312. }
314. **return** 0;
315. }

**Header File:**

1. #define GPIO\_PORTF\_DATA\_R (\*((volatile uint32\_t \*)0x400253FC))//from the lab handout
2. #define RCGCGPIO (\*((unsigned long \*)0x400fE608))
3. #define PORT\_F ((unsigned long)0x20)
4. #define PortF\_DIR (\*((unsigned long \*)0x40025400))
5. #define PortF\_DEN (\*((unsigned long \*)0x4002551C))
6. #define PortF\_DATA (\*((unsigned long \*)0x40025000))
7. #define GPIO\_LOCK (\*((unsigned long \*)0x40025520))
8. #define UNLOCKED ((unsigned long)0x4C4F434B))
9. #define GPIO\_CMT (\*((unsigned long \*)0x40025524))
10. #define GPIO\_PUR (\*((unsigned long \*)0x40025510))
11. #define GPIO\_PORTA\_AMSEL\_R (\*((volatile uint32\_t \*)0x40004528))/////FOR port A, I decided to stick to convention since I used the code copied from the lab... but then i unused it... so... well the convention is probably good.
12. #define SYSCTL\_RCGC2\_R (\*((volatile uint32\_t \*)0x400FE108))
13. #define GPIO\_PORTA\_PCTL\_R (\*((volatile uint32\_t \*)0x4000452C))
14. #define GPIO\_PORTA\_DIR\_R (\*((volatile uint32\_t \*)0x40004400))
15. #define GPIO\_PORTA\_AFSEL\_R (\*((volatile uint32\_t \*)0x40004420))
16. #define GPIO\_PORTA\_DEN\_R (\*((volatile uint32\_t \*)0x4000451C))///that was easy, I already know the bias so.....
17. #define GPIO\_PORTA\_DATA\_R (\*((volatile uint32\_t \*)0x400043FC))
19. //
20. //GPIO Port F (APB) base: 0x4002.5000 offset 51C [Digital enable]
21. //GPIO Port F (APB) base: 0x4002.5000 offset 400 [Control direction - input/output]
22. //General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO) Base 0x400F.E000 Offset 0x608