CSE 474 Lab Report #1

Luke Jiang, 1560831

Abstract:

In this lab, I explored the GPIO interface on TM4C123G launchpad.
In Section A of this lab, by programming Port F, I was able to implement two functionalities:

1) Flashing the red, green and blue LEDs continuously.
2) Turning on red LED when switch 1 is pressed and green LED when switch 2 is pressed.

I then used Tektronix TDS 360 oscilloscope for debugging by measuring the voltage output across LEDs and pushbuttons.
In Section B, I used Port A as the driver for a FSM implementing a traffic light control system.

Introduction:

This lab involves using TI TM4C123G launchpad and IAR Workbench IDE as developing environment. The purpose of this lab contains 6 parts:

1) Getting familiar with developing tools.
2) Learning basics of GPIO.
3) Performing simple I/O interfacing.
4) Using C pointers for addressing GPIO registers.
5) Understanding debouncing.
6) Design a FSM controller.

The methods I applied while developing this Lab includes:

1) Looking up datasheet for information about the registers used in this lab.
2) Maintaining a clear code structure for better readability and performance.
3) Using basic circuit design principles to extend the hardware setup by including multiple switches and LEDs.

Procedure:

Section A:

Step 1: I tried the program in figure 4 and the red LED on the board is turned on.

Step 2: The code without macro definitions

```
// Replace macro definitions in the sample code in Figure 4

int main () {
  *((int *) 0x400FE608) = 0x20;
  *((int *) 0x40025400) = 0x02;
  *((int *) 0x4002551C) = 0x02;
  *((int *) 0x400253FC) = 0;
  *((int *) 0x400253FC) = 0x02;
  while (1) {}
  return 0;
}
```

Step 3: The code I used to implement the functionality of flashing LEDs continuously is the following:

```
1  // CSE 474
2  // Lab 1 Section A
3  // Luke Jiang
4  // 21/06/2018
5
6  // This program makes the LED on TM4C123G LaunchPad flash continnuously.
7
8
9  #include "lab1a.h"
10
11 int main() {
12   // enable port F GPIO
13   RCGCGPIO = RCGCGPIO_F_ON;
14   // set three LEDs as outputs and enable digital
15   GPIO_F_DEN = RGB;
16   GPIO_F_DIR = RGB;
17   // clear all port F
18   GPIO_F_DATA = CLEAR;
19   while (1) {
20     GPIO_F_DATA = RED;
21     for (int i = 0; i < DELAY; i++);
22     GPIO_F_DATA = BLUE;
23     for (int i = 0; i < DELAY; i++);
24     GPIO_F_DATA = GREEN;
25     for (int i = 0; i < DELAY; i++);
26   }
27   return 0;
28 }
29
```

This C code contains only the main() function. At line 13, I turn on the GPIO Port F by setting register RCGCGPIO by setting the pin 5 bit of the register to 1. This process enables the clock of RCGCGPIO. At line 15, I set GPIODEN register to 0x0E = 0b00001110, which enables analog option for P1, P2 and P3. At line 16, I set the GPIODIR register to 0x0E = 0b00001110. This number sets P1 (red), P2 (green), P3 (blue) as output (1). At line 18, I write 0x00 to clear all previous written data in GPIODATA register. Inside the while loop, I first turn on the red LED by writing 0x02 = 0b00000010 to GPIODATA, which turns on P1 (red) and turns off other LEDs. After that, in order to make the red LED stays on for a while, I use a DELAY = 200000 so that each LED stays on for about 2 seconds. When the delay is finished, I turn on the blue LED by writing 0x08 = 0b00001000 to GPIODATA, which turns on P3 (blue) and turns off other LEDs. I need the same delay to make the blue LED stays on for about 2 seconds. Finally, at line 24, I turn on the green LED by writing 0x04 = 0b00000100 to GPIODATA, which turns on P2 (green) and turns off other LEDs. Similarly, this statement is also followed by the delay.

Step 4:
The header file I used for this section in place of the library is the following:

```c
2   // Lab 1 Section A
3   // Luke Jiang
4   // 21/06/2018
5
6   // This header file includes useful register addresses and data
7
8   #ifndef _LAB1A_H_
9   #define _LAB1A_H_
10
11  #include <stdint.h>
12
13  #define DELAY   2000000     // time interval of flashing
14
15  // Inputs to GPIO_F_DATA for configuring LEDs and switches.
16  #define RED     0x02        // enable red LED
17  #define BLUE    0x04        // enable green LED
18  #define GREEN   0x08        // enable blue LED
19  #define RGB     0x0E        // enable all three LEDs
20  #define SWITCH  0x11        // enable switch1 and switch2
21  #define CLEAR   0x00        // disable all
22
23  // Registers configuring GPIO Port F.
24  #define RCGCGPIO        *((volatile unsigned long *) 0x400FE608)
25  #define GPIO_F_DIR      *((volatile unsigned long *) 0x40025400)
26  #define GPIO_F_DEN      *((volatile unsigned long *) 0x4002551C)
27  #define GPIO_F_DATA     *((volatile unsigned long *) 0x400253FC)
28
29  // Input to RCGCGPIO for enabling Port F.
30  #define RCGCGPIO_F_ON  0x20
31
32  // Registers unlocking GPIO PF0.
33  #define GPIO_LOCK       *((volatile unsigned long *) 0x40025520)
34  #define GPIO_CR         *((volatile unsigned long *) 0x40025524)
35  #define GPIO_UR         *((volatile unsigned long *) 0x40025510)
36
37  // Input to GPIO_LOCK for unlocking PF0.
38  #define GPIO_CR_UNLOCK 0x4C4F434B
39
40  #endif // lab1.h
```

The header file first defines DELAY as 2000000 at line 13. This value is used for making the LEDs stays lit for a while. From line 15 to 21 I define inputs to GPIODATA to turn on corresponding peripherals. From line 24 to 27 I define addresses to RCGCGPIO, GPIODIR, GPIODEN and GPIODATA. At line 30 I define 0x20 as the input value to RCGCGPIO in order to enable Port F. From line 33 to line 35 I define addresses to the registers for unlocking PF0 and at line 38 is the input to GPIOLOCK for unlocking PF0.

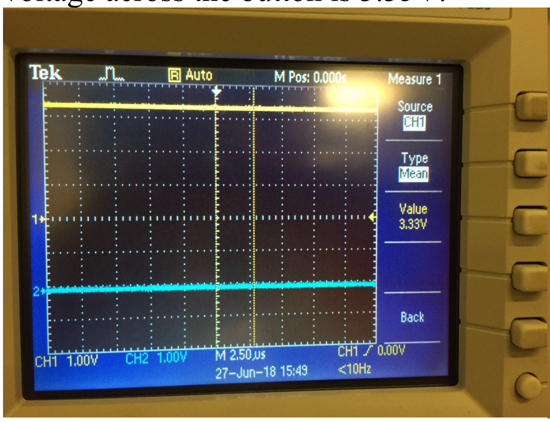
Step 5:

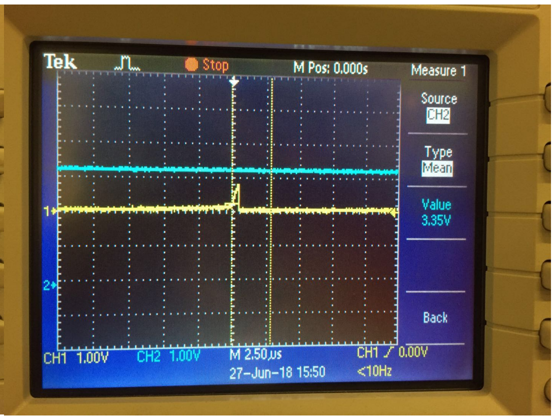
The code I used to implement the functionality of incorporating user input from the switches is the following:

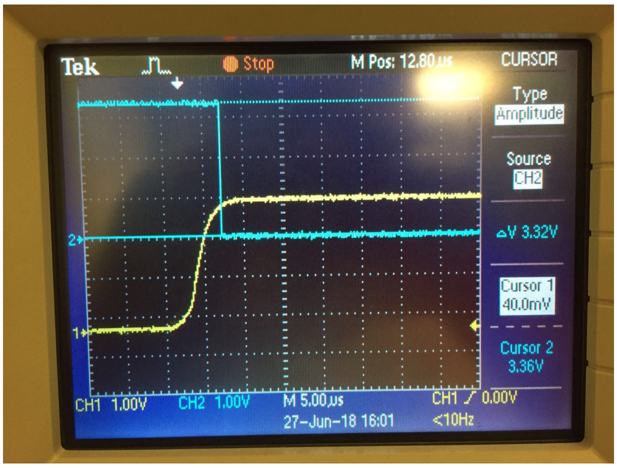
```c
1  // CSE 474
2  // Lab 1 Section A
3  // Luke Jiang
4  // 21/06/2018
5
6  // This program turns red LED on if switch1 is pressed and green LED on
7  // if switch2 is pressed.
8
9  #include "lab1a.h"
10
11 int main() {
12   // enable port F GPIO
13   RCGCGPIO = RCGCGPIO_F_ON;
14   // unlock PF0
15   GPIO_LOCK = GPIO_CR_UNLOCK;
16   GPIO_CR = SWITCH;
17   GPIO_UR = SWITCH;
18   // set two switches as inputs and three LEDs as outputs and enable digital
19   GPIO_F_DEN = RGB | SWITCH;
20   GPIO_F_DIR = RGB & ~SWITCH;
21   // clear all port F
22   GPIO_F_DATA = CLEAR;
23   while (1) {
24     int switch_1 = GPIO_F_DATA & 0x10;
25     int switch_2 = GPIO_F_DATA & 0x01;
26     if (!switch_1) {
27       GPIO_F_DATA = RED;
28     } else if (!switch_2) {
29       GPIO_F_DATA = GREEN;
30     }
31   }
32   return 0;
33 }
```

At line 9 I included the header file I habe written in step 3. At line 13, I turn on the GPIO Port F by setting register RCGCGPIO by setting the pin 5 bit of the register to 1. From line 15 to 17, I unlock PF0 before reading from it, since the pin is locked by default. At line 19 to 20, I write 0b00001110 to GPIODIR to set two switches (P0, P4) as input and three LEDs (P1, P2, P3) as output. At line 20 I write 0b00011111 to GPIODEN to enable digital for P0 to P4. At line 22 I chear all previous data stored in GPIODATA. Inside the while loop, I put the value stored at PF0 in switch_1 and value at PF4 in switch_2. If switch_1 == 0, then switch 1 is pressed, and we set GPIODATA to 0x02 which turns on only red LED. If switch_2 == 0, then switch 2 is pressed, and we set GPIODATA to 0x08 which turns on only green LED.

Step 6:
From the pictures we can see that the latency is 2.70µs; the voltage across LED is 3.33V, the voltage across the button is 3.35V.

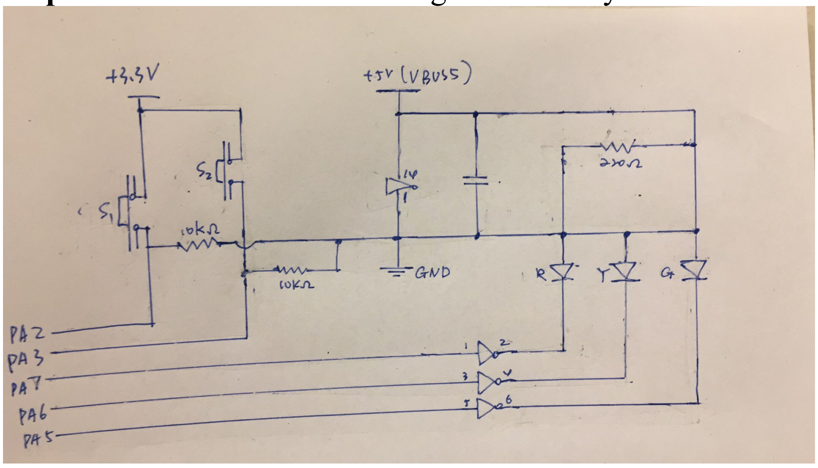






Section B:

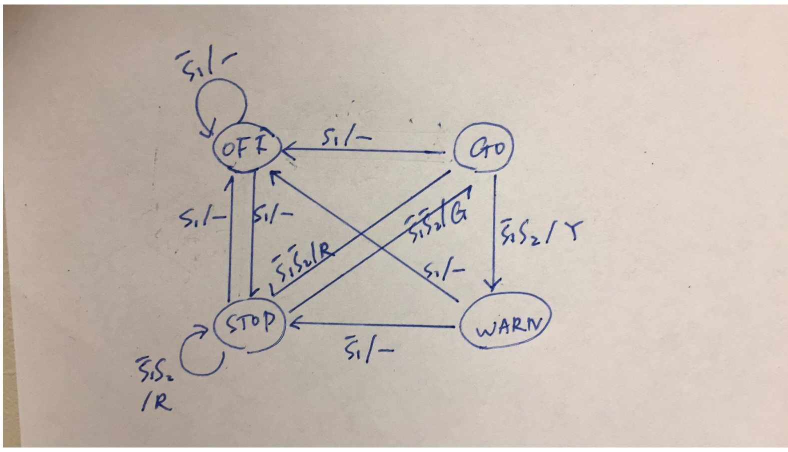
Step 1: Schematics for Traffic Light Control System



I choose Possibility 1 for controlling the three LEDs and two buttons. I modified Figure 6 to implement the traffic light control system in the following aspects:

1) I changed the gates controlling LEDs and buttons to gates specified in Possibility 1.
2) I added the other switch (S2) in parallel to S1 in series with another 10kΩ resistor controlled by PA3.
3) I added yellow (Y) and green (G) LEDs in parallel to red (R) LED in series with their own inverters, each controlled by gates specified in Possibility 1.

Step 2: Finite State Machine Diagram



Notes:

1) S1 represents stop/start button, S2 represents passenger button.
2) In OFF state, all LEDs are off. In GO state, the green LED is on. In WARN state, the yellow LED is on. In STOP state, the red LED is on.

Step 3: Driver program (header file) for interfacing buttons and LEDs

```
// CSE 474
// Lab 1 Section B
// Luke Jiang
// 27/06/2018

// This header file includes useful register addresses and data for interfacing
// GPIO Port A and controlling the traffic light system.

#ifndef _DRIVER_H_
#define _DRIVER_H_

#include <stdint.h>

// time delay used in FSM and button debouncing.
#define DELAY          2000000

// Input to RCGCGPIO for enabling Port A.
#define RCGCGPIO_A_ON  0x01

// Useful GPIO Port A registers.
#define RCGCGPIO       *((volatile unsigned long *) 0x400FE608)
#define GPIO_A_AMSEL   *((volatile unsigned long *) 0x40004528)
#define GPIO_A_PCTL    *((volatile unsigned long *) 0x4000452C)
#define GPIO_A_DIR     *((volatile unsigned long *) 0x40004400)
#define GPIO_A_AFSEL   *((volatile unsigned long *) 0x40004420)
#define GPIO_A_DEN     *((volatile unsigned long *) 0x4000451C)
#define GPIO_A_DATA    *((volatile unsigned long *) 0x400043FC)

// Pins needed for the traffic light system.
#define ON_OFF         0x04     // P2 (start_stop)
#define PASSNGR        0x08     // P3 (passenger)
#define GREEN          0x20     // P5
#define YELLOW         0x40     // P6
#define RED            0x80     // P7
```

```c
// initialize the switch (input == 1) or LED (input == 0) represented by pin.
void init(int pin, int input);
// returns 0x20 if button represented by pin is pressed, else return 0.
unsigned long switch_input(int pin);
// turns on the LED represented by pin.
void led_on(int pin);
// turns off the LED represented by pin.
void led_off(int pin);
// helper function that takes a pin as input, calculates and returns the
// corresponding value to be written to GPIOPCTL.
unsigned long int get_PCTL(int pin);
// time delay for about 2 seconds.
void delay();

#endif

```

Step 4: Driver program (implementation) for interfacing buttons and LEDs

```c
// CSE 474
// Lab 1 Section B
// Luke Jiang
// 27/06/2018

// This program implements functionalities declared in driver.h

#include "driver.h"

void init(int pin, int input) {
  RCGCGPIO |= RCGCGPIO_A_ON;          // activate clock for Port A
  GPIO_A_AMSEL &= ~pin;               // disable analog on pin
  GPIO_A_PCTL &= ~get_PCTL(pin);      // PCTL GPIO on pin
  if (input) GPIO_A_DIR &= ~pin;      // direction pin input
  else GPIO_A_DIR |= pin;             // direction pin output
  GPIO_A_AFSEL &= ~pin;               // pin regular port function
  GPIO_A_DEN |= pin;                  // enable pin digital port
}

unsigned long switch_input(int pin) {
  delay();     // debouncing
  return GPIO_A_DATA & pin;
}

void led_on(int pin) {
  GPIO_A_DATA |= pin;
}

void led_off(int pin) {
  GPIO_A_DATA &= ~pin;
}
```

```c
unsigned long int get_PCTL(int pin) {
  unsigned long int res = 0x0000000F;
  while (pin > 1) {
    res = res << 4;
    pin = pin >> 1;
  }
  return res;
}

void delay() {
  for (int i = 0; i < DELAY; i++);
}
```

Notes:

1) Function led_on(), led_off() are the same to those given in the spec, except that they take a pin number as input to specify which bit in the registers is to be read or written.
2) Function init() merges the functionality of Switch_Init() and LED_Init() by taking an extra parameter input that decides to initialize pin as input or output.
3) Function switch_input() calls delay() to achieve debouncing.
4) Function get_pctl() is a helper function that takes a pin as input, and calculates and returns the corresponding value to be written to GPIOPCTL.

Step 5: FSM implementation

```c
// CSE 474
// Lab 1 Section B
// Luke Jiang
// 27/06/2018

// This program implements a FSM for the traffic light controller.

#include "driver.h"

// define possible states
typedef enum {
  OFF_STATE,
  GO_STATE,
  WARN_STATE,
  STOP_STATE
} State;

// initialize all LEDs and switches and turn off all LEDs.
void init_all();
// if on == 1, turn on the LED represented by pin. Else, turn off that LED.
void led_ctrl(int pin, int on);

int main() {
  State curr_state = OFF_STATE;
  State next_state;
  init_all();  // initialize switches and leds
  while (1) {
    switch (curr_state) {
      case OFF_STATE:
        delay();
        if (switch_input(ON_OFF)) {
          led_ctrl(GREEN, 1);
          next_state = STOP_STATE;
        } else {
          next_state = curr_state;
        }
        break;
      case GO_STATE:
        delay();
        if (switch_input(ON_OFF)) {
          led_ctrl(GREEN, 0);
          next_state = OFF_STATE;
        } else {
          int i;
          for (i = 0; i < DELAY && !switch_input(PASSNGR); i++);
          if (i < DELAY) {   // passenger
            led_ctrl(GREEN, 0);
            led_ctrl(YELLOW, 1);
            next_state = WARN_STATE;
          } else {
            led_ctrl(GREEN, 0);
            led_ctrl(RED, 1);
            next_state = STOP_STATE;
          }
        }
        break;
```

```
      case WARN_STATE:
        if (switch_input(ON_OFF)) {
          led_ctrl(YELLOW, 0);
          next_state = OFF_STATE;
        } else {
          delay();
          led_ctrl(YELLOW, 0);
          led_ctrl(RED, 1);
          next_state = STOP_STATE;
        }
        break;
      case STOP_STATE:
         delay();
         if (switch_input(ON_OFF)) {
           led_ctrl(RED, 0);
           next_state = OFF_STATE;
         } else if (switch_input(PASSNGR)) {
           next_state = STOP_STATE;
         } else {
           led_ctrl(RED, 0);
           led_ctrl(GREEN, 1);
           next_state = GO_STATE;
         }
         break;
    }
    curr_state = next_state;
  }
  return 0;
}
void init_all() {
  switch_init(ON_OFF);
  switch_init(PASSNGR);
  led_init(RED);
  led_off(RED);
  led_init(YELLOW);
  led_off(YELLOW);
  led_init(GREEN);
  led_off(GREEN);
}

void led_ctrl(int pin, int on) {
  if (on) led_on(pin);
  else led_off(pin);
}
```

Results

My implementation of functionalities specified in Section 1 works exactly as I intended. However, the time intervals between two turn-on events of each LED are not all equal to the value I specified. I also spent a decent amount of time trying to wire up the correct schematic and learning how to use the oscilloscope.

Conclusion

I learned the following skills in this lab:

1) Looking up datasheet.
2) Use Tektronix oscilloscope to find voltages and latency.
3) Interfacing GPIO ports.
4) Using IAR Workbench.
5) Building LED and button driver circuit with GPIO interface.

As an introduction to embedded system, this lab gives me a overview of how embedded programming looks like. I realized the importance of finding data and solving problems on my own in the field of embedded programming.

Reference

1. TM4C123 data sheet
2. IDE installation guide