Praveen Ravishankar **Fall 2019**
ME 4405 Section B

Lab Assignment Three

Writing your First Program

Questions:

1. How should the code be modified if the requirement was to switch off the LED after a small arbitrary delay? (Hint: No timers necessary)

   In order for the LED to switch off after a small arbitrary delay without using a timer, then a 'for' loop could be implemented. Specifically, this 'for' loop will perform nothing but will iterate a large number of times (say 100,000 times) in order to cause a brief delay in the progression of the program. However, since the 'for' loop should run only immediately after the switch is released, it's implied that the program needs to detect if the switch had just been released as well. This can be accomplished by including an additional variable that tracks the switch's previous state and by incorporating an additional 'if' statement (nested within the 'if' statement that checks if the switch is NOT pressed) that checks if the previous reading of the switch corresponds to the switch being pressed in order to determine if the switch had just been released. Specifically, the 'for' loop and the additional necessary code can be implemented as shown:

```
void main(void)
{
    WDT_A_holdTimer(); // Stops the Watchdog Timer

    GPIO_setAsInputPinWithPullUpResistor(GPIO_PORT_P1, GPIO_PIN1); // Sets Switch S1 as an input button

    // Configures LED1 and sets its initial state to off
    GPIO_setAsOutputPin(GPIO_PORT_P1, GPIO_PIN0);
    GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN0);

    unsigned int usiButton1 = 0; // Initializes the input value of Switch S1
    unsigned int usiButton1_prev = 0; // Initializes the previous input value of Switch S1

    while(1) // Runs an infinite loop for this program
    {
        usiButton1 = GPIO_getInputPinValue(GPIO_PORT_P1, GPIO_PIN1); // Reads Switch S1

        if(usiButton1 == GPIO_INPUT_PIN_LOW) // Determines if Switch S1 is pressed
        {
            GPIO_setOutputHighOnPin(GPIO_PORT_P1, GPIO_PIN0); // Turns LED1 on
        }
        else if(usiButton1 == GPIO_INPUT_PIN_HIGH) // Determines if Switch S1 is not pressed
        {
            if(usiButton1 != usiButton1_prev) // Determines if the switch was previously pressed
            {
                int i; // Initializes the integer used in the upcoming 'for' loop
                for(i = 1; i < 100000; i++) // Runs an empty 'for' loop to induce a small time delay before the LED turns off
                {

                }
            }
            GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN0); // Turns LED1 off
        }

        usiButton1_prev = usiButton1; // Assigns the previous input value of Switch S1
    }
}
```

2. Suppose you were to write a program that toggles LED1 whenever S1 is pressed. With proper operation of the program you would expect to see the light turn on when you press and release S1, and then turn off when you press and release S1 again. Suppose you write this program as follows:

Assume that P1.1 has been set up with a pull-up resistor. Please do not worry about whether variables are defined or not. Your only task for this problem is to find faults in the pseudo code.

```
while(1){

        // Read button
        usiButton1 = GPIO_getInputPinValue ( GPIO_PORT_P1, GPIO_PIN1 );

        //If button is pressed...
        if ( usiButton1 == GPIO_INPUT_PIN_LOW ) {

                GPIO_toggleOutputOnPin(GPIO_PORT_P1, GPIO_PIN0) ;

        }

}
```

What are two problems with this code? What would be some potential software modifications that would fix these problems?

Firstly, with the way the code is originally written, the LED would repeatedly switch on and off at an extremely high speed while the switch is continuously pressed. This would result in the switch appearing to always be on at half the brightness of its typical brightness (since the rapid oscillation between the LED turning on and off would cause the LED to display the average brightness value of the two states, which would be half of its normal brightness level); additionally, once the switch is released, this code would cause the LED to either remain on or remain off in a random manner.

Since the objective of the program is for the LED to toggle once the switch is pressed and *released*, assigning an action to the LED while the switch is being *pressed* is not only irrelevant, but it's incorrect as well; in other words, an action should be assigned to the LED immediately after the switch is released. Hence, **the first problem with the code is the incorrect boolean of the 'if' statement: by checking if the switch is pressed**. A simple solution to this problem would be to change the 'GPIO_INPUT_PIN_LOW' value to 'GPIO_INPUT_PIN_HIGH', causing the 'if' statement to check if the switch is NOT being pressed. Now, **the second problem with the code is that the program still requires a method to detect if the switch had just been released in order to determine whether or not to toggle the LED**. This can easily be resolved by first implementing an additional variable that tracks the switch's previous state (being pressed or not being pressed). Next, an additional 'if' statement nested within the first 'if' statement (the 'if' statement that determines if the switch is not pressed) could be included that determines if the switch had just been released by checking if the previous state of the switch is equal to a 'pressed' state (if the previous state of the switch is equal to 'GPIO_INPUT_PIN_LOW'). After resolving these two problems, the code will now be fixed and function as desired:

```c
#include "msp.h"
#include "driverlib.h"
#include "driver.h"

/**
 * main.c
 */
void main(void)
{
    WDT_A_holdTimer(); // Stops the Watchdog Timer

    GPIO_setAsInputPinWithPullUpResistor(GPIO_PORT_P1, GPIO_PIN1); // Sets Switch S1 as an input button

    // Configures LED1 and sets its initial state to off
    GPIO_setAsOutputPin(GPIO_PORT_P1, GPIO_PIN0);
    GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN0);

    unsigned int usiButton1 = 0; // Initializes the input value of Switch S1
    unsigned int usiButton1_prev = 1; // Initializes the previous input value of Switch S1

    while(1) // Runs an infinite loop for this program
    {
        usiButton1 = GPIO_getInputPinValue(GPIO_PORT_P1, GPIO_PIN1); // Reads Switch S1

        if(usiButton1 == GPIO_INPUT_PIN_HIGH) // Determines if Switch S1 is not being pressed
        {
            if(usiButton1_prev == GPIO_INPUT_PIN_LOW) // Determines if Switch S1 has just been released
            {
                GPIO_toggleOutputOnPin(GPIO_PORT_P1, GPIO_PIN0); // Toggles LED1
            }
        }

        usiButton1_prev = usiButton1; // Assigns the previous state of Switch S1
    }
}
```