**Matthew Anderson** **ECGR 3101 Lab #0 Report** **Date 09/25/23**
**Mande137@uncc.edu**
**Ali Altabtabae**
**saltabta@uncc.edu**

**Lab Objective:**
**This lab aimed to teach the activation and control of GPIO blocks, using switches and LEDs with a TM4C123GXL microcontroller via the TivaWare Peripheral Driver Library API.**

**Lab Questions or Figures:**
*Before you begin, make sure to freshen your basic physics/electric circuit skills. Also, you will want to make sure that you have some basic familiarity with the board, microcontroller, and TivaWare library. You can start by answering these questions.*

1. If you draw two resistors (R1=R2) in series and there is a +5V difference from one end of the circuit to the other, use Kirchoff's to calculate the voltage drop across the first resistor.
    1. **Using Kirchoffs Voltage law we can assume 5 = VR1 + VR2, if R1 = R2 then 5 = 2R1, therefore VR1=2.5V**
2. Find the circuit diagram for the EK-TM4C123GXL development board and locate where LEDs are connected to the MCU. What components are used (besides the R/G/B LED)? What role does the transistor(s) have?
    1. **The transistors (Q1-Q3, specifically the DTC114EET1G NPN transistors) on the EK-TM4C123GXL development board are used as electronic switches. The MCU uses these to control the state of the RGB LEDs without directly sourcing or sinking the current required by the LEDs. Transistor acts as a gate, allowing the MCU to turn the LEDs on and off based on the signals it provides. The schematic also indicates that several resistors are used, one is connected from the MCU pins to the transistor of each diode, and one from the transistor to the diode itself.**
3. The TMC4 family of MCUs allows you to use a built-in, internal resistance ... to use this feature, what TivaWare function call allows you to enable it? What are the range of options to this call? Here you need to pick an unused Port with a GPIO block that is connected to one of the headers (perhaps Port E on J2 or J3).
    1. **Using the port provided the TivaWare function should look something like this: GPIOPadConfigSet(GPIO_PORTE_BASE, GPIO_PIN_3, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);**

**Commentary and Conclusion:**
**During the lab, we faced challenges with configuring GPIO pins and comprehending the SysCtlDelay function for time delays. After consulting with the TA and referring to provided resources, we successfully resolved the GPIO pin setup issues. Additionally, we encountered difficulties in determining the correct clock frequency on the TM4C123GXL board for SysCtlDelay, but we managed to solve it by manually setting the clock frequency and performing calculations.**

**In conclusion, while the TivaWare API offers somewhat intuitive but occasionally confusing macros for assigning functions to GPIO ports, we found that reading the provided documentation and user guide enabled us to complete the lab successfully.**

**Matthew Anderson**
**Mande137@uncc.edu**
**Ali Altabtabae**
**saltabta@uncc.edu**

**ECGR 3101 Lab #0 Report**

**Date 09/25/23**

**Lab Code:**

Format your code and paste it here. Do NOT just copy and paste your code straight from CCS into your word document. It will not be formatted, and you will lose points.

```
//*****************************************************************************
**
//
// Author: Matthew Anderson
// Author: Ali Altabtabae
// Date:   9/21/2023
// Due Date: 9/25/2023
//
// Description:
// This Program simply turns an LED off for a duration if both switch 1 and
switch 2 are logical high
//
//*****************************************************************************
**

#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/sysctl.h"

int clkFreq = 16000000;


#define SYSTEM_CLOCK_FREQUENCY clkFreq // set clock frequency for timer
calculation


int main(void) {
    uint8_t switch1Val, pin4Val;  // Variables to store the values of the
input pins

    // Enable Port F and E which contains the pins we are interested in
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);

    // Check if the peripheral access is enabled
    while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOF))
    {
    }

    // Enable the GPIO pin for the blue LED (PF2). Set the direction as
output.
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_2);
```

**Matthew Anderson**
**Mande137@uncc.edu**
**Ali Altabtabae**
**saltabta@uncc.edu**

```
    // Enable the GPIO pin for switch 1 (PF4/SW1). Set the direction as
input.
    GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_4);
    GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);

    // Enable the GPIO pin for the new pin 4 (PE4). Set the direction as
input.
    GPIOPinTypeGPIOInput(GPIO_PORTE_BASE, GPIO_PIN_4);
    GPIOPadConfigSet(GPIO_PORTE_BASE, GPIO_PIN_4, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);

    uint32_t delayValue = 5 * clkFreq/3;  // assuming a 16 MHz clock

    //// Loop forever.
    while (1) {

        // Read the values of switch 1 (PF4) and pin 4 (PE4).
        switch1Val = GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_4);
        pin4Val = GPIOPinRead(GPIO_PORTE_BASE, GPIO_PIN_4);

        // Check the state of switch 1 and pin 4
        if ((switch1Val != GPIO_PIN_4) && (pin4Val != GPIO_PIN_4)) {
            // Turn off the blue LED (PF2).
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0x0);
            SysCtlDelay(delayValue);

        }
        else {
            // Turn on the blue LED (PF2)
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, GPIO_PIN_2);
        }
    }
}
```