Lab 1: Blink an Onboard LED

Using TM4C LaunchPad™ Development Kit

School of Engineering

Electrical and Computer Engineering Department

ECEG 721-61 Embedded Systems

Lucia Rosado-Fournier

September 25th, 2020

Table of Contents

1.1 Objective

The main objective of this lab is to get an introduction to the Tiva C Series TM4C123GXL LaunchPad Evaluation Kit by learning how to configure the GPIO pins to turn on and off the onboard green LED with the use of the C language.

1.2 Introduction

The Tiva C Series TM4C123GXL LaunchPad Evaluation Kit (EK-TM4C123GXL) is a starter kit loaded with many features to better understand the onboard microcontroller, the TM4C123GH6PM. With the use of the EK-TM4C123GXL and the microcontroller together, it is possible to do several different tasks, including the lab discussed in this report.

One of the many things onboarded on the EK-TM4C123GXL is three different colored LED's, connected to the microcontrollers GPIO pins, PF1, PF2, and PF3. In this lab, the goal is to blink the green LED, connected to PF3, at a set interval of time. Sending a HIGH or LOW to the pin connected to the LED will turn it on or off respectively, achieving the blinking that is being sought after.

As mentioned earlier, the GPIO port PF3 will be used in this lab. On the TM4C123GH6PM, there are six different GPIO blocks with corresponding ports ranging from A to F. On these ports, there are the different pins that can be connected. In the case of the green LED, it is connected to pin three on port F, as seen in the figure below.

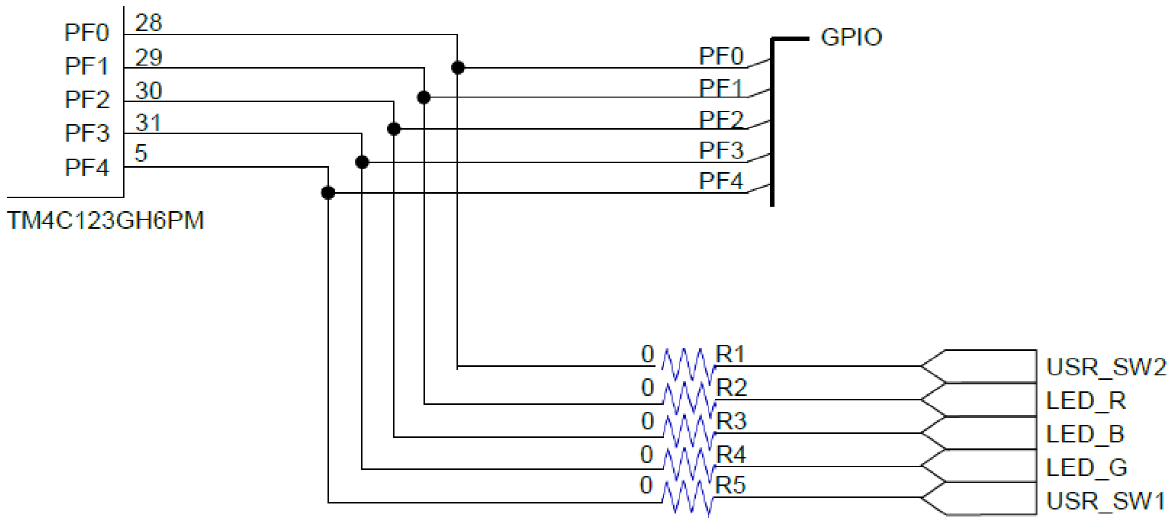

Figure 1 - Schematic of the GPIO connected to the LEDs.

The schematic also outlines the pin connections to two other LED's and the two onboard user switches which will not be used in this lab.

1.3 Component Requirements

Starting with the software components, the lab being followed states to use Code Composer Studio as an Integrated Development Environment (IDE), but it was advised to use Keil instead. This is mainly because Keil can support multiple boards, unlike Code Composer Studio. The other software component needed was the TivaWare for C Series software suite. This was added to have the libraries necessary to work with different peripherals.

Moving onto hardware, the main component needed was the EK-TM4C123GXL which has the microcontroller being used as discussed in the introduction. The only other component needed was USB cable that came with the kit so it could be plugged into a computer.

1.4 Software

As mentioned in the objective, this lab is written in the C language and being developed in the Keil IDE. After following the proper procedure, which is detailed in the procedure section, the software is programmed onto the microcontroller.

Flowchart

Following the flowchart in the figure below, the first thing that needs to be done is to configure the frequency of the system clock to 40MHz. This is done for the process of setting the delay between the green LED being turned on an off later. Afterwards, GPIO F needs to be enabled so pin three can be used later. Once it’s been enabled, PF3 is then configured to be the output. As mentioned in the introduction, a HIGH output turns on the LED and a LOW turns it off. With this in mind, a delay is set between the HIGH and LOW output and looped again using a while loop that doesn’t end.

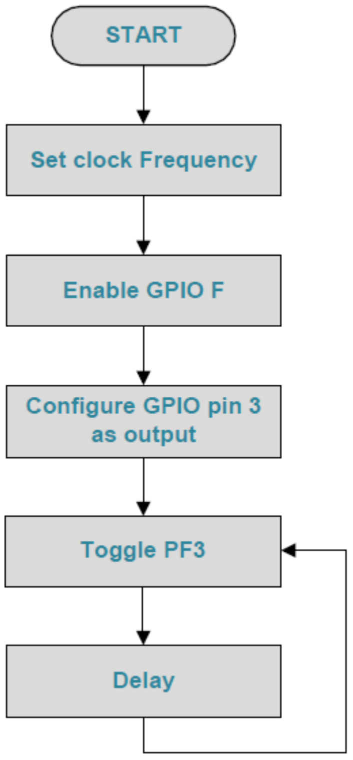

Figure 2 Flowchart of the code to blink an LED.

To expand upon the delay, it can be set by determining the number of counts to delay and using that number with the `SysCtlDelay()` function. In this program, the delay is 500ms. Using the formula below, it is possible to calculate the number of counts to be used.

$$Number\ of\ Counts = Time\ Delay\ Required * System\ Clock\ Frequency$$

$$Number\ of\ Counts = 500 * 10^{-3} * 40 * 10^{6} = 20 * \ 10^{6}$$

C code

The code below follows the logic outlined in the flowchart. Though the code was given, it is heavily commented to explain the logic going through it.

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

int main(void)
{
  /*Configures the device clock. Decides which input crystal frequency,
  oscillator to be used, PLL use
  and system clock divider.*/
  SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|
  SYSCTL_OSC_MAIN);

  //Enable a peripheral (port F in this case)
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

  //Port F GPIO Pins 1, 2, 3 are configured to be used as outputs (these
  are the LEDs)
  GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|
  GPIO_PIN_3);

  //While loop to turn on LED, wait, turn off LED, wait
  while(1){
    /*Write a value to a specific pin. 0x02 is red (pin1), 0x04 is blue
    (pin2), and 0x08 is green (pin3)
    Using pipes, can specify multiple LEDs to turn on
    GPIOPinWrite(base address [f in this case], bit packed rep of pin
    [again the LED pins], value to write to pins)
    Seems like order of the pins and values doesn't matter*/
    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x08);

    //Specifies an amount of time to delay (500ms in this case)
    SysCtlDelay(20000000);
    //Write an off value to the pin. 0x00 is off

    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00);

    //Specifies an amount of time to delay
    SysCtlDelay(20000000);
  }
}
```

Code Block 1 – C code to blink a green LED.

1.5 Procedure

The steps to run the code above on the microcontroller are below:

1. Connect the EK-TM4C123GXL to the computer using the USB cable and turn it on.
2. Open Keil on the computer.
3. Click “Project” → “New µVision Project”.
4. In the new window, select where the project should be saved. It is suggested to save each project in their own folder because each project makes multiple files.
5. Click “Save”.
6. Back at Keil, click the + next to “Target 1” and do the same for “Source Group 1”. This shows the files in the project.
7. Right click “Source Group 1” then click “Add New Item to Group ‘Source Group 1’…”.
8. In the new window, click “C File (.c)” and then type “main” in the “Name” box. Then click “Add”. This creates the main file where the code is run.
9. Under “Source Group 1” double click “main.c” and add the code detailed above.
10. Before things can work, the necessary libraries are missing. Right click “Source Group 1” and click “Add Existing Files to Group ‘Source Group 1’…”.
11. In the new window, navigate to where Keil is installed → ARM → Startup → TI → TM4C123 and select system_TM4C123.c and click “Add”.
12. In the same window, navigate to where ti is installed (needed to be installed beforehand) → TivaWare_C_Series-2.1.4.178 → driverlib → rvmdk and select driverlib.lib and click “Add”.
13. Now the board needs to be selected. Click “Flash” → “Configure Flash Tools…”.
14. In the new window go to the “Debug” tab and select the “Use” option. In the dropdown, select “Stellaris ICDI”.
15. In the same window, go to the “C/C++” tab and under “Include Paths” click the button with the “…”.
16. In the new window, click the “New (Insert)” button and then click the “…” button that appears.
17. In the new window, navigate to where ti is located and select the sub folder and click “ok”. This is where the library added earlier is located.

The previous steps must be done when making a new project and using a new board. This doesn’t need to be done every time the code is changed. The following steps do.

18. Next, click “Build” and ensure there are no errors on the build output. Then click “Download” to program the microcontroller.
19. At the microcontroller, click the reset button on the board and the board should start blinking green.

1.6 Observation
As expected, the LED on the board started blinking green 500ms intervals. There were no issues with this process.

1.7 Summary
Since I was on a Windows PC, the set up and implementation was relatively simple. The code was easy to follow and a good introduction to the board. Altogether, the lab process and report writing took about two hours.

1.8 Exercise
The following exercises are an extension of the lab demonstration discussed prior. After trying different values, the values for the different colors were determined, again elaborated upon in the C Code section.

Blink Two LED’s
The task was to blink any other two LED’s on the board. Coding wise, the process was very similar to the green LED blinking. The code for that is detailed below and the differences are highlighted.

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

int main(void)
{
  /*Configures the device clock. Decides which input crystal frequency,
  oscillator to be used, PLL use
  and system clock divider.*/
  SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|
  SYSCTL_OSC_MAIN);

  //Enable a peripheral (port F in this case)
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
  //Port F GPIO Pins 1, 2, 3 are configured to be used as outputs (these
  are the LEDs)
  GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|
  GPIO_PIN_3);

  //While loop to turn on LED, wait, turn off LED, wait
  while(1){
    /*Write a value to a specific pin. 0x02 is red (pin1), 0x04 is blue
    (pin2), and 0x08 is green (pin3)
    Using pipes, can specify multiple LEDs to turn on
    GPIOPinWrite(base address [f in this case], bit packed rep of pin
    [again the LED pins], value to write to pins)
    Seems like order of the pins and values doesn't matter*/
    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
    0x02|0x04);

    //Specifies an amount of time to delay (500ms in this case)
    SysCtlDelay(20000000);


    //Write an off value to the pin. 0x00 is off

    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00);

    //Specifies an amount of time to delay
    SysCtlDelay(20000000);
  }
}
```

Code Block 2 – C code to blink the blue and red LED to create purple.

Blink All LED's

The task was to blink all of the LED's on the board to blink white. Coding wise, the process was very similar to the green LED blinking. The code for that is detailed below and the differences are highlighted.

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

int main(void)
{
  /*Configures the device clock. Decides which input crystal frequency,
  oscillator to be used, PLL use
  and system clock divider.*/
  SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|
  SYSCTL_OSC_MAIN);

  //Enable a peripheral (port F in this case)
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

  //Port F GPIO Pins 1, 2, 3 are configured to be used as outputs (these
  are the LEDs)
  GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|
  GPIO_PIN_3);

  //While loop to turn on LED, wait, turn off LED, wait
  while(1){
    /*Write a value to a specific pin. 0x02 is red (pin1), 0x04 is blue
    (pin2), and 0x08 is green (pin3)
    Using pipes, can specify multiple LEDs to turn on
    GPIOPinWrite(base address [f in this case], bit packed rep of pin
    [again the LED pins], value to write to pins)
    Seems like order of the pins and values doesn't matter*/
    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
    0x02|0x04|0x08);

    //Specifies an amount of time to delay (500ms in this case)
    SysCtlDelay(20000000);


    //Write an off value to the pin. 0x00 is off

    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00);

    //Specifies an amount of time to delay
    SysCtlDelay(20000000);
  }
}
```

Code Block 3 – C code to blink all of the LED's to create white.