TI ARM Lab 2
Bright Light

National Science Foundation
Funded in part, by a grant from the National Science Foundation
DUE 1068182

Acknowledgements
Developed by Craig Kief and Brian Zufelt from the COSMIAC Research Center at the University of New Mexico. Co-Developers are Bassam Matar from Chandler-Gilbert and Karl Henry from Drake State. Originally Funded by the National Science Foundation.

Lab Summary
This is a lab for connecting the hardware the first time. This lab will go through the process of turning on a single light. This lab was originally developed for the Stellaris LaunchPad and then modified for the Tiva LaunchPad.

Lab Goal
The goal of this lab is provide sufficient instruction and information so that individuals will be able to connect the Tiva board for the first time and create a simple project that lights an LED. It will introduce the user to the building and debugging process within the Code Composer Studio (CCS v6) environment.

Each of these labs add upon the previous labs and it is the intention of the authors that the students will build (with each lab) a better understanding of the ARM processor and basic C code. Even though these tutorials assume the student has not entered with a knowledge of C code, it is the desire that by the time the student completes the entire series of tutorials that they will have a sufficient knowledge of C code so as to be able to accomplish useful projects on the ARM processor.

It is assumed that the "student" will be installing the packages in accordance with the instructions in Lab1. That process will provide the framework and load all the support files/projects for the associated workshop. Each Lab will have two files. For example, there is a generally a Lab x project and a Lab x Solution project. The only difference between these two projects is the code in the main.c file. The exception is in the interrupt project (Lab 5) where the …startup_ccs.c file is also edited in the solution. The reason this was done this way was to provide the student with two choices. They can follow the tutorial in the Lab and type in the associated code identified in the tutorial. This will allow for errors to be made and learning to be achieved through the debugging process. The alternative is that the student has a fall back option if everything goes bad and they can just look at/copy the source file contents from the "Solution" projects.

Learning Objectives
The student should begin to become familiar with the complier and understand the use of a "main.c" file.

Time Required
Approximately one hour

Lab Preparation

It is highly recommended that the student read through this procedure once before actually using it was a tutorial. By understanding the final goal it will be easier to use this as a tutorial. This Lab will introduce the Tiva C Series TM4C123G LaunchPad Evaluation Kit which is a low-cost evaluation platform for ARM® Cortex™-M4F-based microcontrollers from Texas Instruments. The design of the TM4C123G LaunchPad highlights the TM4C123GH6PM microcontroller with a USB 2.0 device interface and hibernation module.

The EK-TM4C123GXL evaluation kit also features programmable user buttons and a Red, Green, RGB LED for custom applications. The stackable headers of the Tiva C Series TM4C123G LaunchPad BoosterPack XL Interface make it easy and simple to expand the functionality of the TM4C123G LaunchPad when interfacing to other peripherals with Texas Instruments' MCU BoosterPacks.

Equipment and Materials

Access to the four software packages: Code Composer, Tivaware, Driver file and ATE Workshop installer are required and should have been installed in accordance with Lab 1. In addition, the user should have access to the Tiva evaluation kit (EK-TM4C123GXL) and (for Labs 3-9) the Digilent Orbit board. It is assumed that the student has already completed Lab 1 and the software is installed.

Software needed	Quantity
The installation files are covered in Lab 1. The software package can be downloaded as one large zip file from this URL: http://cosmiac.org/thrust-areas/education-and-workforce-development/community-portal/	1
Hardware needed	**Quantity**
The hardware required is the Tiva LaunchPad Kit. The kit and the ORBIT daughter card can be purchased from the Digilent Corporation www.digilentinc.com	1

Additional References

This page is for general information: http://www.ti.com/tool/ek-tm4c123gxl on the Tiva LauchPad Kit. If the link above is no longer valid, just google the Tiva LaunchPad and it will pop up. The full set of tutorials is available on this website: http://cosmiac.org/thrust-areas/education-and-workforce-development/microcontrollers/ate-developed-material/

Lab Procedure 1: Install/Connect board to computer

Step 1: Plug in the supplied USB cable to the top of the Evaluation Kit. Ensure the switch on the board is set to “DEBUG” and not “DEVICE”.

```
/*****************************************************

Project : LED LAB 2,3 ATE (Launchpad)
Version : 2.0
Date    : 2/20/2015
Author  : Brian Zufelt / Craig Kief
Company : COSMIAC/UNM
Comments:
This Code is intended to show how to connect, compile,
a write your first project on the Tiva-C Launchpad Board


*****************************************************
Chip type               : ARM TM4C123GH6PM
Program type            : Firmware
Core Clock frequency            : 80.000000 MHz
*****************************************************/
#include <tm4c123gh6pm.h>
#include <stdint.h>

// definitions

#define LED_RED 0x02
#define LED_BLUE 0x04
#define LED_GREEN 0x08

// Lab definitions for the 2 versions of the lab
//#define Lab2
#define Lab3

void main(void) {

long unsigned int i = 0;                    //general counter


SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;                            // enable PORT F GPIO

GPIO_PORTF_DIR_R = LED_RED|LED_BLUE|LED_GREEN;                  // set PORT F as output

GPIO_PORTF_DEN_R = LED_RED|LED_BLUE|LED_GREEN;                  // enable digital PORT F
#ifdef Lab2
GPIO_PORTF_DATA_R = 0;                                          // clear all PORT F

GPIO_PORTF_DATA_R = LED_RED|LED_BLUE|LED_GREEN;                 // set LED PORT F pins high

#endif

// loop forever
while(1){

#ifdef Lab3
```

```
            GPIO_PORTF_DATA_R = 0;                  // clear all PORT F
            GPIO_PORTF_DATA_R = LED_GREEN;                  // set LED PORT F pins high
            for(i=0;i<2500000;i++){};               //delay
            for(i=0;i<4000000;i++){};               //delay
            GPIO_PORTF_DATA_R = 0;                  // clear all PORT F
            GPIO_PORTF_DATA_R = LED_GREEN | LED_RED;    // set LED PORT F pins high
            for(i=0;i<2000000;i++){};               //delay
            GPIO_PORTF_DATA_R = 0;                  // clear all PORT F
            GPIO_PORTF_DATA_R = LED_GREEN | LED_RED;    // set LED PORT F pins high
            for(i=0;i<3000000;i++){};               //delay
            GPIO_PORTF_DATA_R = 0;                  // clear all PORT F
            GPIO_PORTF_DATA_R = LED_RED;                    // set LED PORT F pins high
            for(i=0;i<4000000;i++){};               //delay
#endif

 }
}
```

The source code that will be used in this project is shown above. This is a horrible way to code for a microcontroller but it is designed to be as simple as possible for initial understanding. Also, this is the first time the hardware is exercised so often this is the point in the workshop that takes the longest. The code is commented. For now, it is important to just read through the code and begin to get a feeling for what it does. The purpose of each line of code will be explained later. Understand that the TI part is designed to be very low power. As such, the user must turn on and activate those parts that will be used. In this case, activation of a single port (port F) and then configuring the individual port's pins to be outputs. Launch Code Composer Studio v6 by double clicking on the icon shown in figure 1 below.

Figure 1. Code Composer

For those new to CCS 6, the screen shown in Figure 2 is where you begin. This is an excellent resource and some time should be devoted watching the videos and finding out what is available there. The screen only

appears automatically the first time CCS is launched. After that, to get back to the screen shown in Figure 2, it is necessary to click on Help-> Getting Started.

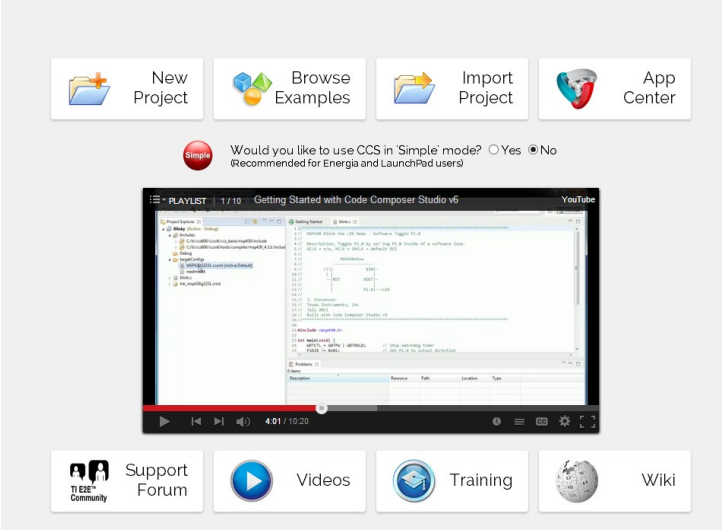

Figure 2. Start Screen

This tutorial uses Eclipse as the Development environment but uses Code Composer as the software complier. To begin, since the "student" used Lab 1 to install the associated software, there should be a Lab 2 and a Lab 2_3 Solution on the left hand side in the Project Explorer pane. It is assumed the student wants to type in the code as a learning experience. If main.c is opened in the Lab 2 Project, the screen should be the same as Figure 3 below.

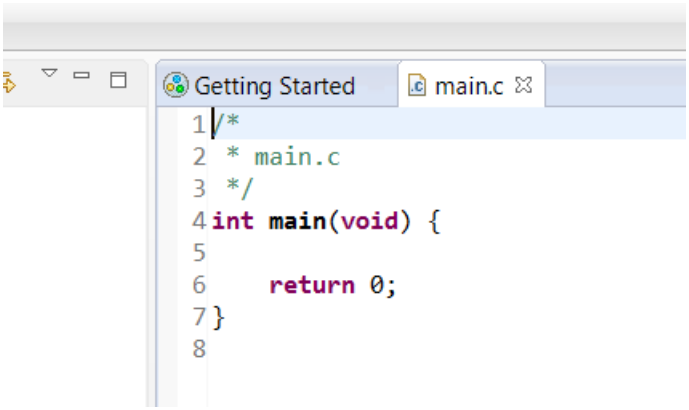

Figure 3. main.c

main.c

```
Company : COSMIAC/UNM
Comments:
This Code is intended to show how to connect, compile,
and write your first project on the Tiva Launchpad Board

************************************************
Chip type               : ARM tm4c123gh6pm
Program type            : Firmware
Core Clock frequency    : 80.000000 MHz
************************************************/
// definitions & Included Files
#include <tm4c123gh6pm.h>
#include <stdint.h>

#define LED_RED 0x02
#define LED_BLUE 0x04
#define LED_GREEN 0x08
void main(void) {
SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;               // enable PORT F GPIO
GPIO_PORTF_DIR_R = LED_RED|LED_BLUE|LED_GREEN;     // set PORT F as output
GPIO_PORTF_DEN_R = LED_RED|LED_BLUE|LED_GREEN;     // enable digital PORT F
GPIO_PORTF_DATA_R = 0;                             // clear all PORT F
GPIO_PORTF_DATA_R = GPIO_PORTF_DATA_R | LED_RED; // | LED_BLUE | LED_GREEN; // set LED PORT F pins high
// loop forever
while(1){
    }
```

Figure 4. Replace the project code

There are two choices. First, if the student desires the learning experience, then they can type in all the code that is identified in Figure 4 into the Lab 2 skeleton. The designer should now have the design as shown in in Figure 4. As can be seen in the four lines of the code that does most of the work, there is a lot of reference to port F. To understand how this works, it is important to first look at the user's manual for the evaluation kit. The URL for this is currently: http://www.ti.com/general/docs/lit/getliterature.tsp?baseLiteratureNumber=spmu296&fileType=pdf But, if this doesn't work, just google the Tiva LaunchPad Evaluation Board User's Guide.

Figure 5. The User's Manual for the Evaluation Kit

The user should be able to open the pdf shown in Figure 5. This is an excellent resource and should be read at a later time. For now, skip to the schematics shown in in Figure 6 and 7.

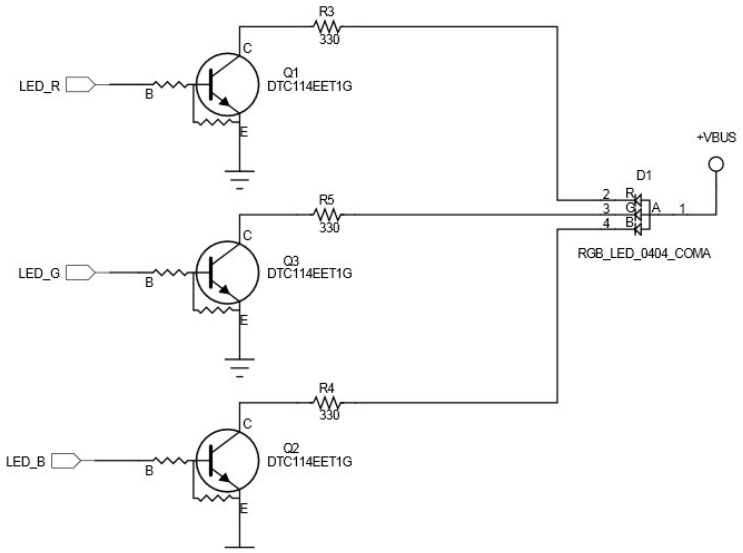

Figure 6. Three LED's in One

The Tiva board that makes up the evaluation kit has an LED on it. Although it is incredibly small, it is actually three LED in one as shown in Figure 6. This first tutorial will combine the outputs of these three LEDs to produce a white light.

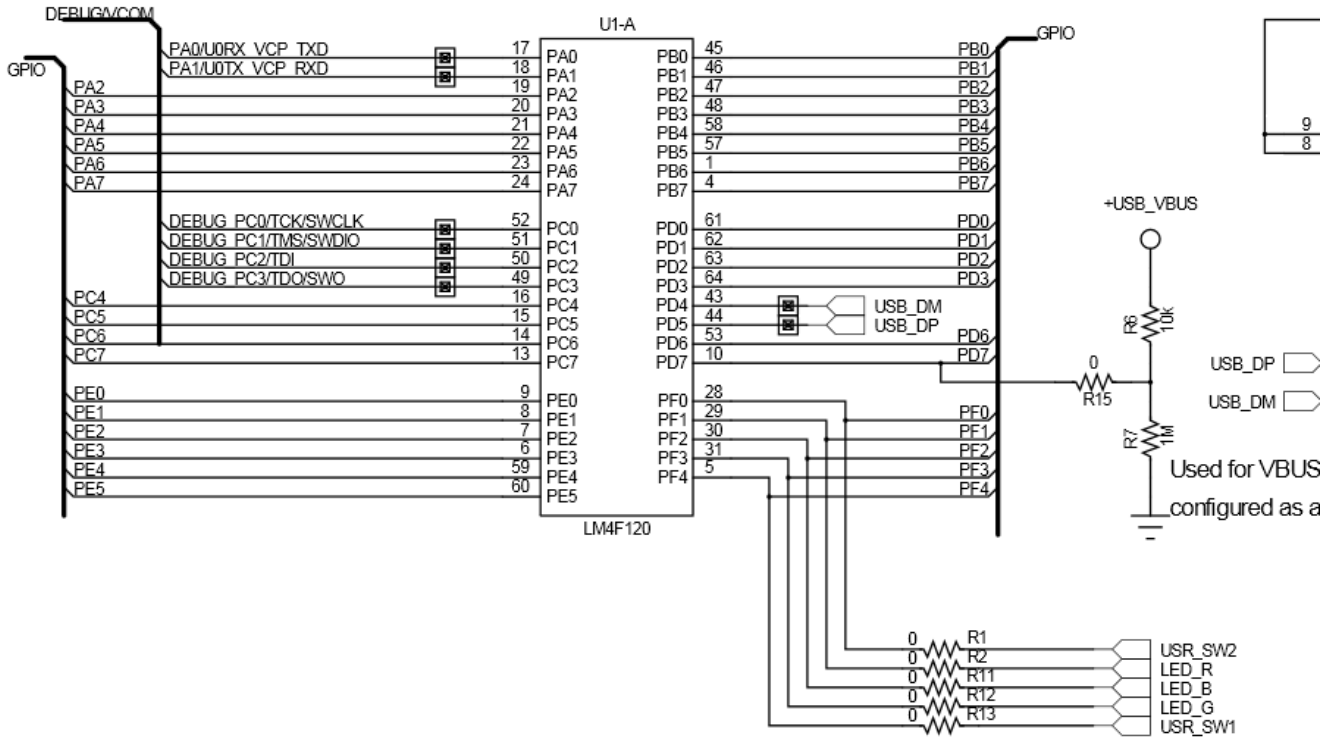

Figure 7. Port F of the Chip

Figure 7 shows how the three LEDs are connected to the processor. They are connected to a port which is one byte (8 bits) wide. In the code, we define red as "0000 0010"; blue as "0000 0100"; and green as "0000 1000". There are certain requirements for using port F that are taken care of in the first three lines of the code after void main (void). The first turns on port F. For power saving, it (like all other ports) is powered down by default. The second sends "1s" to the part which sets it as an output. The third tells port F that it is digital.

At this point it is best to explain a little more about what is going on with this line of code:

GPIO_PORTF_DATA_R = GPIO_PORTF_DATA_R | LED_RED | LED_BLUE | LED_GREEN

Remember that a "1" turns on an LED. We have three LEDs we want to turn on. First, we set the entire port F to 0 (all eight bits). Then we do our functions as shown below:

Port F =	0000 0000
Port F =	0000 0000
(Or with)	0000 0010
(Or with)	0000 0100
(Or with)	0000 1000
Port F =	0000 1110 (Turns on Red, Green, and Blue)

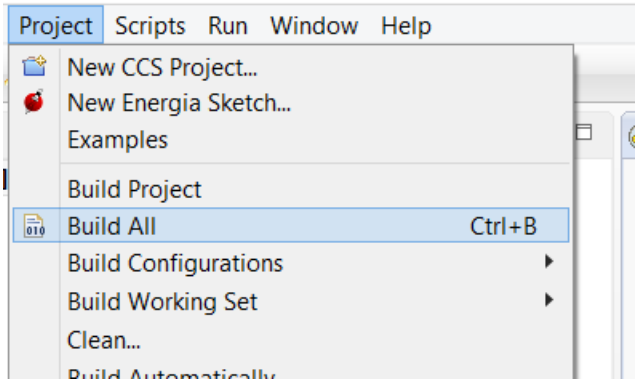

Figure 8. Building the Project

The next step is to build the project into an executable that will run on the TI chip. Click on Project→Build All. If there are errors then it might be necessary the first time to include projects into your path once. This would only occur if there were a problem with the workshop.exe file identified in Lab 1. This would be if there was a question mark next to an include statement. This include is the header file with all the important (and critical) information about the board. If when you attempt to build the project the first time, there is an error, it is because it can't find this file. The next several steps tell it where to look. Go to Project→Properties as shown in Figure 9.

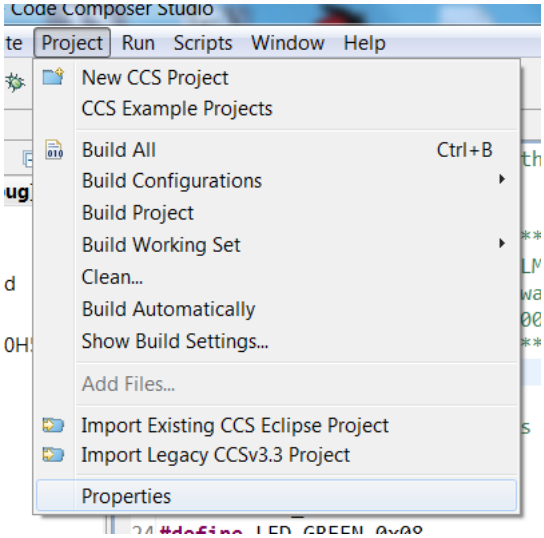

Figure 3. Project Properties

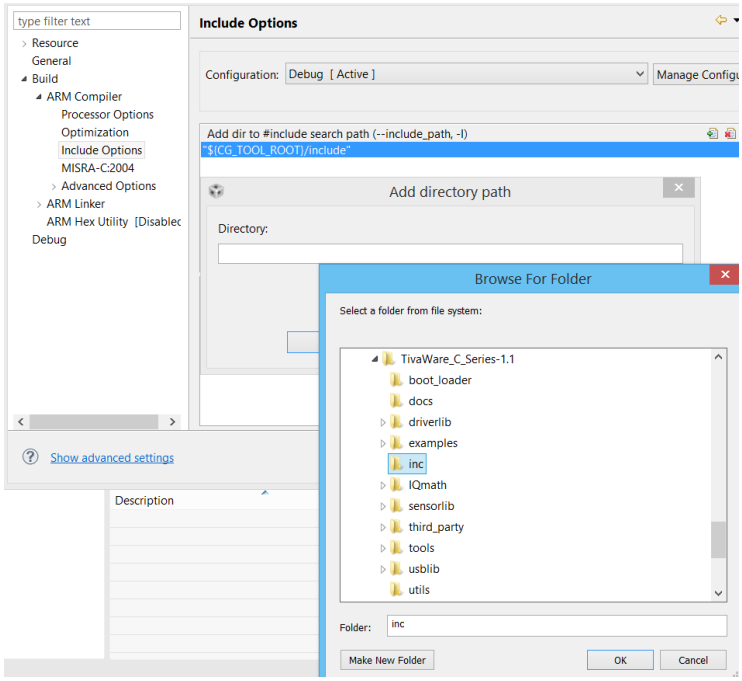

Figure 40. Adding the path for the .h File

As shown in Figure 10, click on the “Include Options” selection. Click on the “+” symbol to add a directory path scan to the “inc” directory shown in Figure 10 and click OK. This now adds the included directory to the path where the board header file is stored so the compiler can find the needed information about the specific board that is being used.

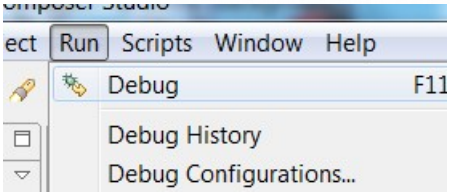

Figure 5, Debugger

Next, it is time to program the board. Click on Run and then Debug as shown in Figure 11.

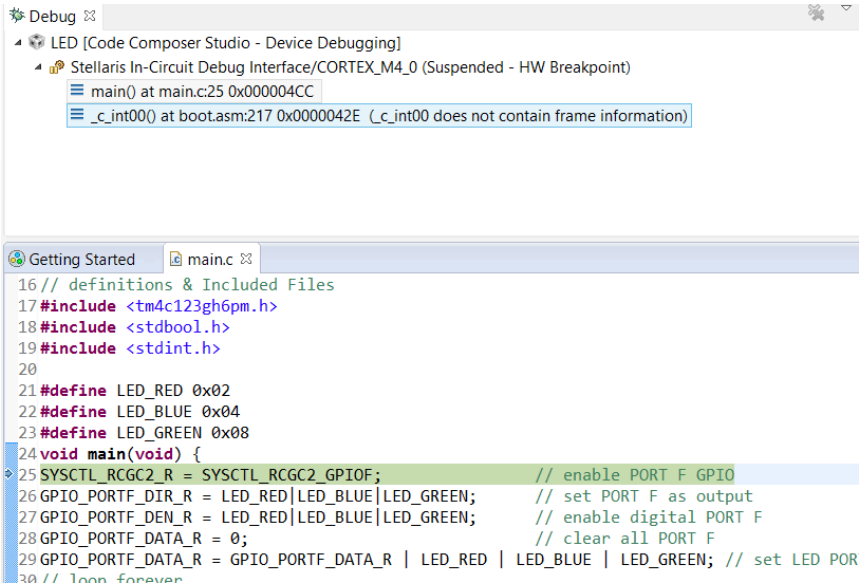

Figure 6. Stopping at Main

Highlight the main file as shown in Figure 12 and then click “Resume” which looks like a green triangle. The white light should be on.

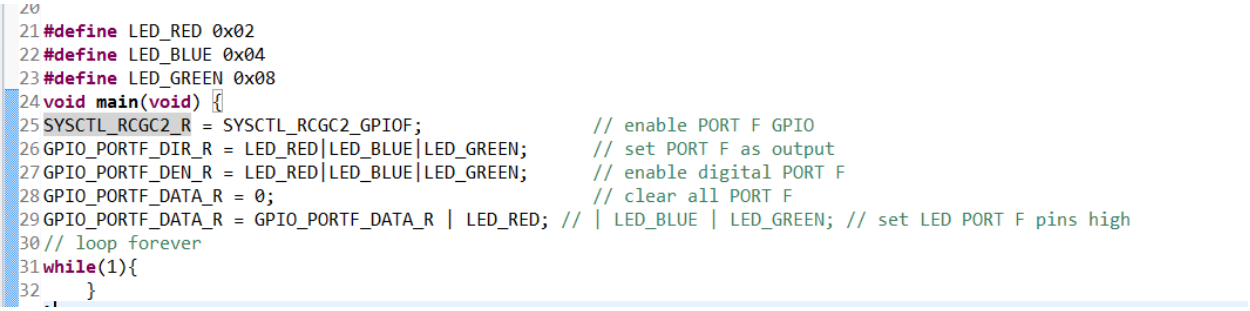

Figure 7. Going RED

The final step is to have the light be red instead of white. At this point it is best to explain a little more about what is going on with this line of code:

GPIO_PORTF_DATA_R = GPIO_PORTF_DATA_R | LED_RED | LED_BLUE | LED_GREEN

Remember that a “1” turns on an LED. We have three LEDs we want to turn on. First, we set the entire port F to 0 (all eight bits). Then we do our functions as shown below:

```
        0000 0000
or      0000 0010
or      0000 0100
or      0000 1000
        0000 1110
```

So, to turn on only Red, comment out (remove) the other two colors as shown in Figure 13 and rerun. To rerun, hit the red square, change code, save, debug (little bug symbol) then green triangle again.

Attachment 1: main.c solution file

```
/************************************************************

Project : LED LAB 2,3 ATE (Launchpad)
Version : 2.0
Date    : 2/20/2015
Author  : Brian Zufelt / Craig Kief
Company : COSMIAC/UNM
Comments:
This Code is intended to show how to connect, compile,
a write your first project on the Tiva-C Launchpad Board


************************************************************
Chip type               : ARM TM4C123GH6PM
Program type            : Firmware
Core Clock frequency : 80.000000 MHz
************************************************************/
#include <tm4c123gh6pm.h>
#include <stdint.h>

// definitions

#define LED_RED 0x02
#define LED_BLUE 0x04
#define LED_GREEN 0x08

// Lab definitions for the 2 versions of the lab
//#define Lab2
#define Lab3

void main(void) {

long unsigned int i = 0;                    //general counter


SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;                        // enable PORT F GPIO

GPIO_PORTF_DIR_R = LED_RED|LED_BLUE|LED_GREEN;          // set PORT F as output

GPIO_PORTF_DEN_R = LED_RED|LED_BLUE|LED_GREEN;          // enable digital PORT F
#ifdef Lab2
GPIO_PORTF_DATA_R = 0;                                                  // clear all PORT F

GPIO_PORTF_DATA_R |= LED_RED|LED_BLUE|LED_GREEN;        // set LED PORT F pins high

#endif

// loop forever
while(1){

#ifdef Lab3

        GPIO_PORTF_DATA_R = 0;                                                  // clear all PORT F

        GPIO_PORTF_DATA_R = LED_GREEN;                                  // set LED PORT F pins high

        for(i=0;i<2500000;i++){};                                       //delay

        for(i=0;i<4000000;i++){};                                       //delay

        GPIO_PORTF_DATA_R = 0;                                                  // clear all PORT F

        GPIO_PORTF_DATA_R = LED_GREEN | LED_RED;                // set LED PORT F pins high

        for(i=0;i<2000000;i++){};                                       //delay
```

```
        GPIO_PORTF_DATA_R = 0;                                          // clear all PORT F
        GPIO_PORTF_DATA_R = LED_GREEN | LED_RED;          // set LED PORT F pins high
        for(i=0;i<3000000;i++){};                                       //delay
        GPIO_PORTF_DATA_R = 0;                                          // clear all PORT F
        GPIO_PORTF_DATA_R = LED_RED;                                    // set LED PORT F pins high
        for(i=0;i<4000000;i++){};                                       //delay
#endif

 }
}
```

Attachment 2: Block Diagram of the Pins Used in Projects