TI ARM Lab 3
Blinking Lights

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

Acknowledgements
Developed by Craig Kief, Brian Zufelt, and Jacy Bitsoie at the Configurable Space Microsystems Innovations & Applications Center (COSMIAC). Co-Developers are Bassam Matar from Chandler-Gilbert and Karl Henry from Drake State. *Funded by the National Science Foundation (NSF).*

Lab Summary
This lab builds upon the second lab which turned on a light. This lab will go through the process of showing the "#define" syntax as well as the looping syntax for developing a delay.

Lab Goal
The goal of this lab is to continue to build upon the skills learned from previous labs. This lab helps the student to continue to gain new skills and insight on the C code syntax and how it is used in the TI implementation of the ARM processor. Each of these labs add upon the previous labs and it is the intention of the authors that the student 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.

As each tutorial is presented, a higher level of abstraction will be introduced. This is done to allow future work to be achieved on a wide variety of different processors.

Learning Objectives
The student should begin to become familiar with the complier and understand the use and modification of a "main.c" file. In addition, the student should begin to understand the concept of a "for loop."

Grading Criteria
N/A

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 as a learning guide. Up through Lab 2, it is possible to do the labs with just what is shown in the labs themselves. However, from here forward, it is best to be able to just download the group of lab projects and install them.

Equipment and Materials
Access to Tiva TM4C123G LaunchPad software and evaluation kit (EK-RM4C123GXL). It is assumed that the student has already completed Lab 2 and the software is installed properly.

Software needed	Quantity
Have already installed the Code Composer Studio software from the instructions in Lab 1. Download and install the Lab Tutorials from this site: http://cosmiac.org/Community_Portal_Micro.html	1
Hardware needed	Quantity

The hardware required is the TI Tiva LaunchPad Kit	1

Additional References

The Evaluation Board user's manual is on this web site: http://datasheet.octopart.com/EK-TM4C123GXL-Texas-Instruments-datasheet-15542121.pdf

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". It is assumed that the evaluation board is properly installed and operating. Launch the Code Composer software. The easiest way to find the executable is to go to Start, All programs and then look in the area identified in Figure 1.

Figure 1. Launching Code Composer

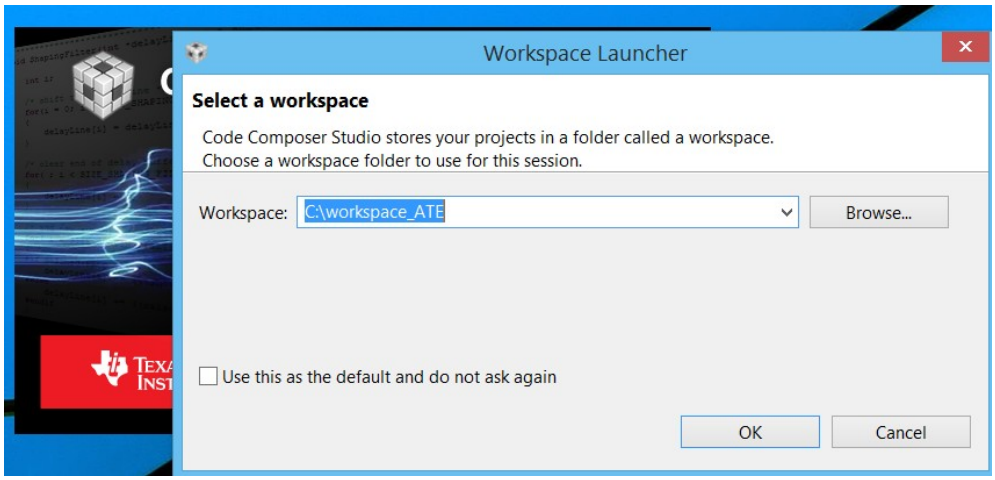

Figure 2. Workspace Launcher

The designer will be presented with the choices shown in Figure 2. The Code Composer again wants to know where the workspace is located. If you have installed the small zip file from the end of Lab 1 (Lab Procedure 3) then the installer will have loaded a lot of tutorial files to your C root directory in a directory called workspace_ATE. Use this as the location for your "select a workspace" choice shown in Figure 2. Click OK.

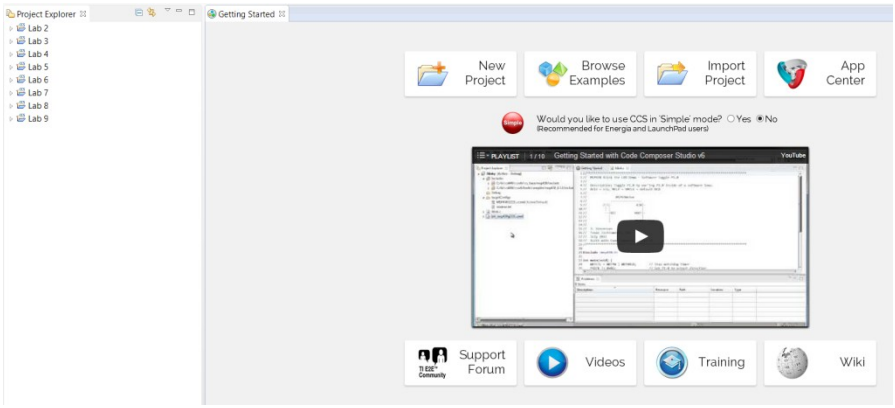

Figure 3. Project Start Screen

The user should be presented with a screen similar to that shown in Figure3. It is possible now to see all the different projects for the remaining tutorials that have been created. Open main.c as shown in Figure 4. Copy the code presented below Figure 4 below and replace the code in your main.c.

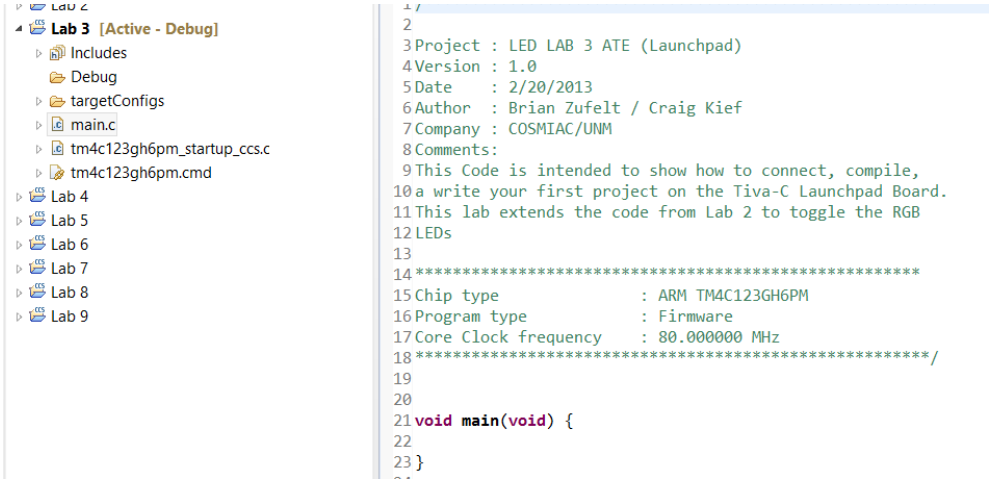

Figure 4. Main.c

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

Project : LED LAB 2,3 ATE (Launchpad)
Version : 1.0
Date    : 2/20/2013
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

 }
}
```

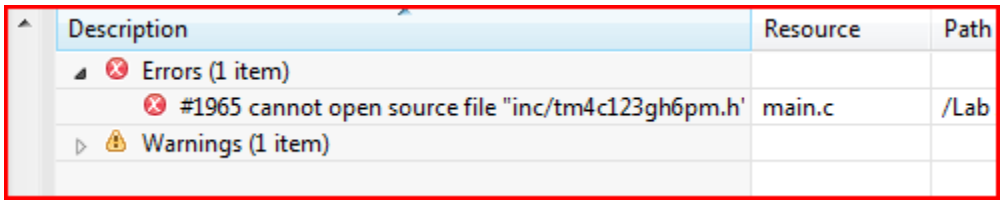

Figure 5. 1965 Error

When you attempt to do Project-> Build all, you might get an error similar to the one shown in Figure 5. This error would say something similar to #1965 cannot open source file “inc/tm4c123gh6pm.h”. This means that the includes are not set properly for the main project. Here is an easy set of steps to fix this. First, go back to the main installer as shown in Figure 6. Double click on the shown installer. This will install TivaWare_C_Series-2.1.0.12573. The complicated part is that there is a large quantity of different versions of TivaWare floating around. This is often like hitting an object floating on the ocean. To make sure everything works well in all the tutorials, install this version of TivaWare.

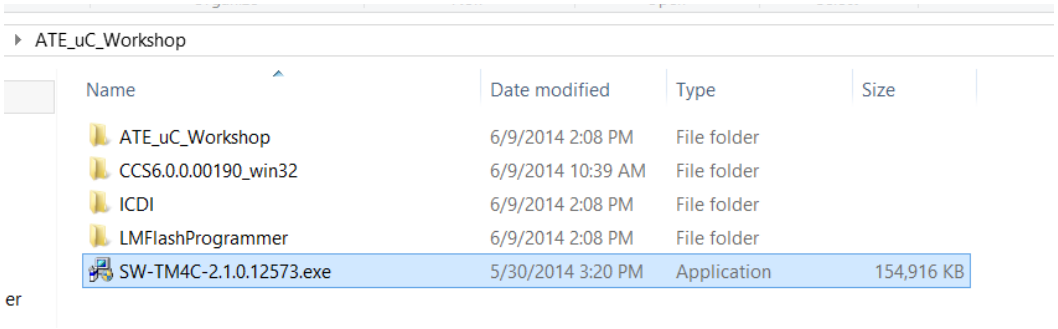

Figure 6. TivaWare Installer

Once it is complete, it is possible to go into the root TI directory as shown in Figure 7. There may be multiple versions of TivaWare. For the remainder of these tutorials, we will use Series-2.1…..

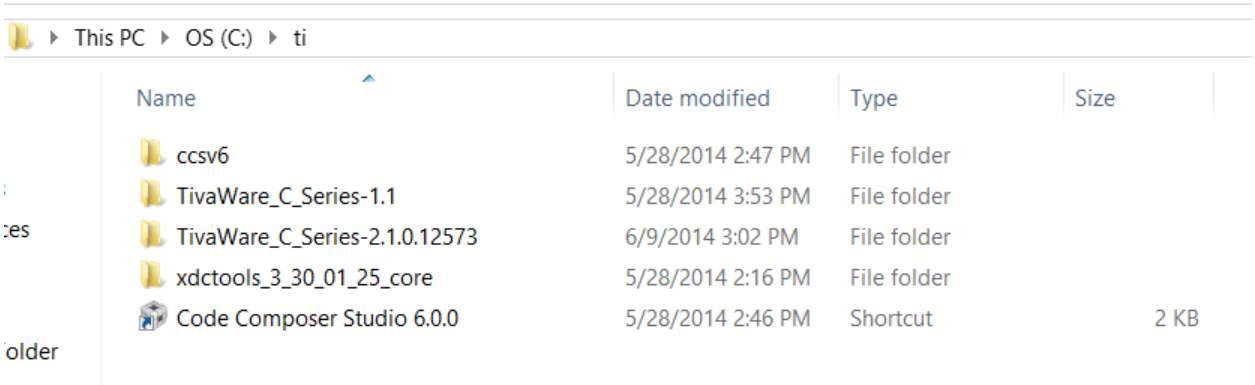

Figure 7. TI Root Directory

The way to clear the error for this and all other projects is to include the path into the project for the version of TivaWare shown above. Go into Project and then to Properties in CCS. Under the add directory in the center of the GUI, click on the green plus symbol as shown in Figure 8.

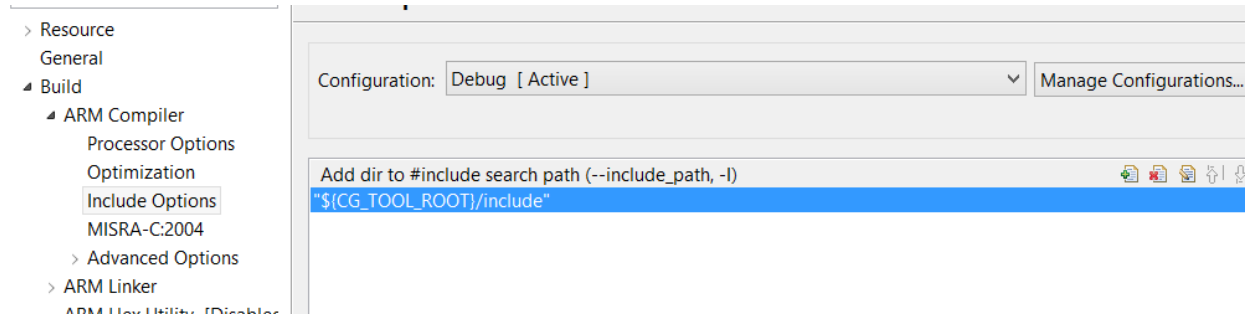

Figure 8. Project Properties

Once the green plus symbol is checked, go to the file system option as shown in Figure 9.

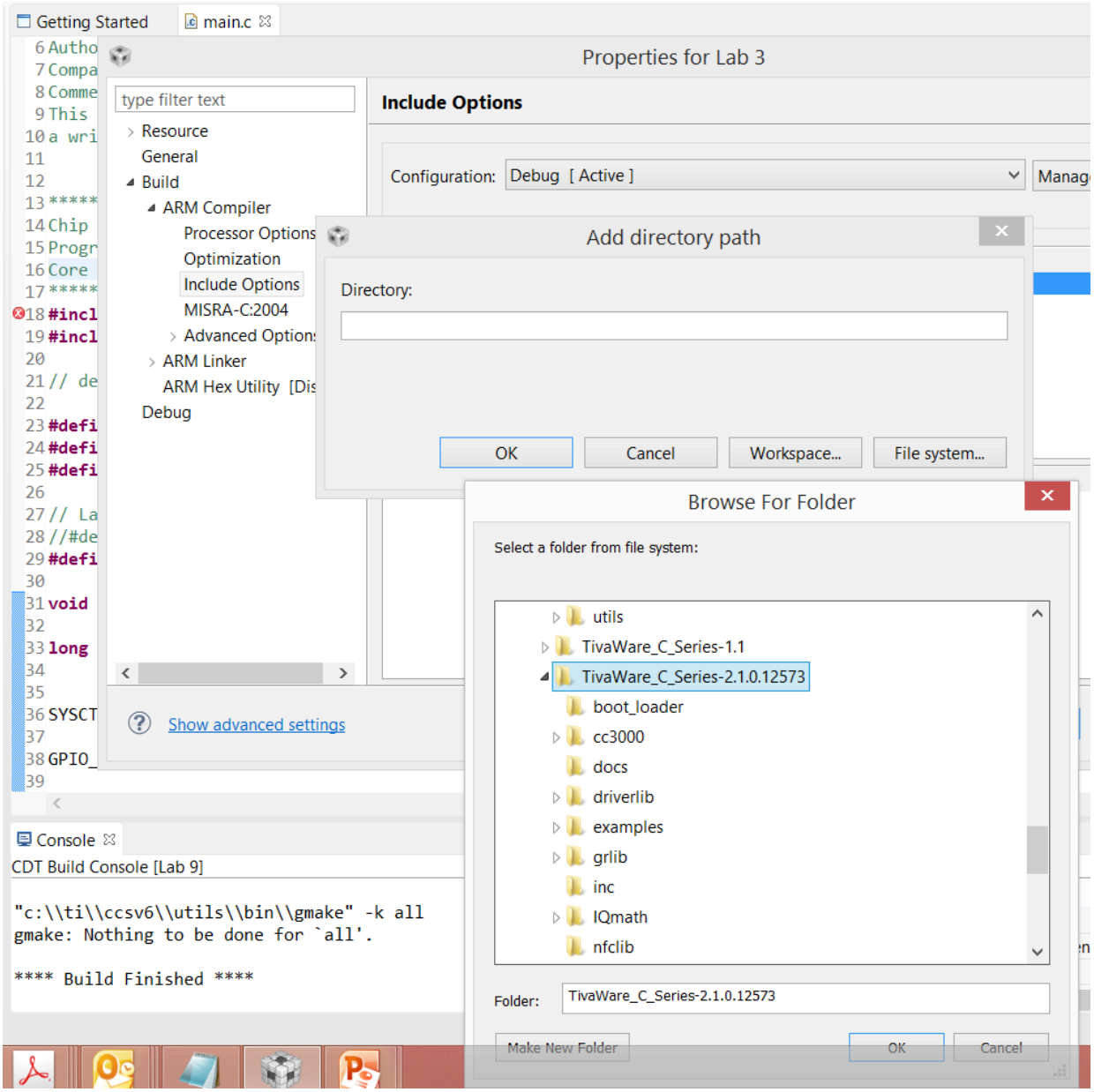

Figure 9. Adding the Include Files from TivaWare

Include the upper level of the Tivaware directory and when complete, go back to CCS and then do Project, then Build All. This should complete with no errors (warnings about different versions for creating the project are fine). Before proceeding to debugging, it is important to investigate several new aspects in the code that were not in previous labs.

There are several new constructs in this source code that should be explained. The first is the "#define". This is a preprocessor directive inherited from C that takes the form:

```
#define identifier value
```

In general, it is used to tell the preprocessor to replace all instances of “identifier” in the code with the given text before passing it on to the compiler.

```
23 #define LED_RED 0x02
24 #define LED_BLUE 0x04
25 #define LED_GREEN 0x08
26
27 // Lab definitions for the 2 versions of the lab
28 //#define Lab2
29 #define Lab3
30
31 void main(void) {
32
33 long unsigned int i = 0;        //general counter
34
35
36 SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;              // enable PORT F GPIO
37
38 GPIO_PORTF_DIR_R = LED_RED|LED_BLUE|LED_GREEN;    // set PORT F as output
39
40 GPIO_PORTF_DEN_R = LED_RED|LED_BLUE|LED_GREEN;    // enable digital PORT F
41 #ifdef Lab2
42 GPIO_PORTF_DATA_R = 0;                            // clear all PORT F
43
44 GPIO_PORTF_DATA_R |= LED_RED|LED_BLUE|LED_GREEN;  // set LED PORT F pins high
45
46 #endif
```

Figure 10. #define Example

In the case of line 23 in Figure 10, it is used to say, everywhere that LED_RED appears, replace it with 0000 0010. Another example is line 29 from Figure 10. It is now possible to have multiple projects within a single file and by commenting (or uncommenting) the various define statements, it is possible to include or exclude various sections. This is shown as follows.

If the designer has a statement: //#define Lab2 (as in line 28), this “//” means that the line is commented out and should be ignored. It is not part of the project. It allows the designer to be able to have both Lab1 and Lab2 in a single file. By commenting (or uncommenting) the #define Lab “x”, it brings the first or second lab online. The benefit of this is that it allows the designer to be able to define global constants (such as the LED_RED) assignment) once and use it for multiple projects. These files are operated upon sequentially from top to bottom.

Another new statement is the “while(1)”. This means, do all statement below this forever. A final new syntax is the for loop. The statement means that there is an integer called “i”. As long as i is less than a really big number chosen at random (in this lab, we did two or four million), keep adding “1” to it. Once i gets to the really big number, bounce out of the loop and keep going to the next line. This allows the lights to remain a certain color for a pleasing amount of time.

The next step is to build the project into an executable. Click on Project→Build All. Another way to accomplish this is by clicking the control and B keys at the same time.

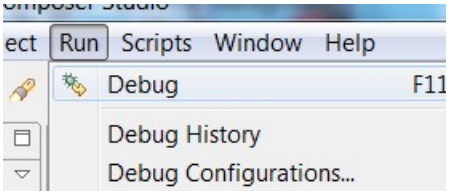

Figure 11. Debugger

Next, it is time to program the board. Click on Run and then Debug as shown in Figure 11. Another shortcut is to hit the F11 key.

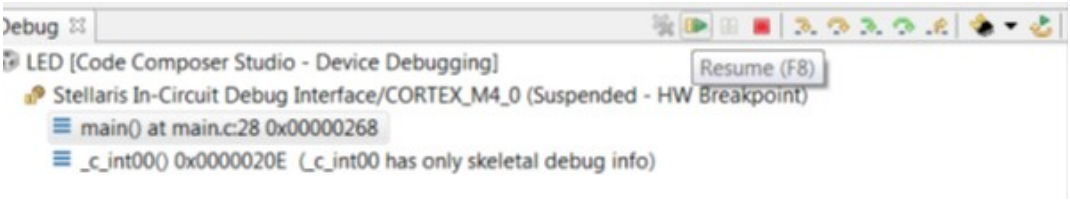

Figure 12. Stopping at main

Highlight the main file as shown in Figure 12 and then click “Resume” – green triangle. The red, green and yellow lights should begin flashing.

The final step is to modify the code to change the colors, sequence of colors and the amount of delay time.

Attachment 1: main.c solution file

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

Project : LED LAB 2,3 ATE (Launchpad)
Version : 1.0
Date    : 2/20/2013
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 1: Block Diagram of the Pins Used in Projects