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.

Equipment and Materials
Access to Stellaris LM4F120 LaunchPad software and evaluation kit (EK-LM4F-120XL). It is assumed that the student has already completed Lab 2 and the software is installed properly.

Software needed	Quantity
Download Stellaris LaunchPad™ software from the TI website http://www.ti.com/tool/SW-EK-LM4F120XL	1
Hardware needed	Quantity
The hardware required is the TI Stellaris LaunchPad Kit	1

Additional References

Current TI ARM manuals found on TI web site: http://www.ti.com/tool/ek-lm4f120xl.

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.

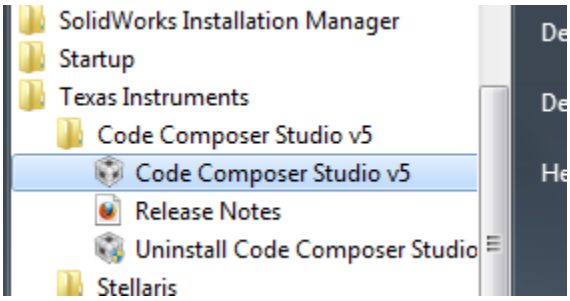

Figure 1. Code Composer

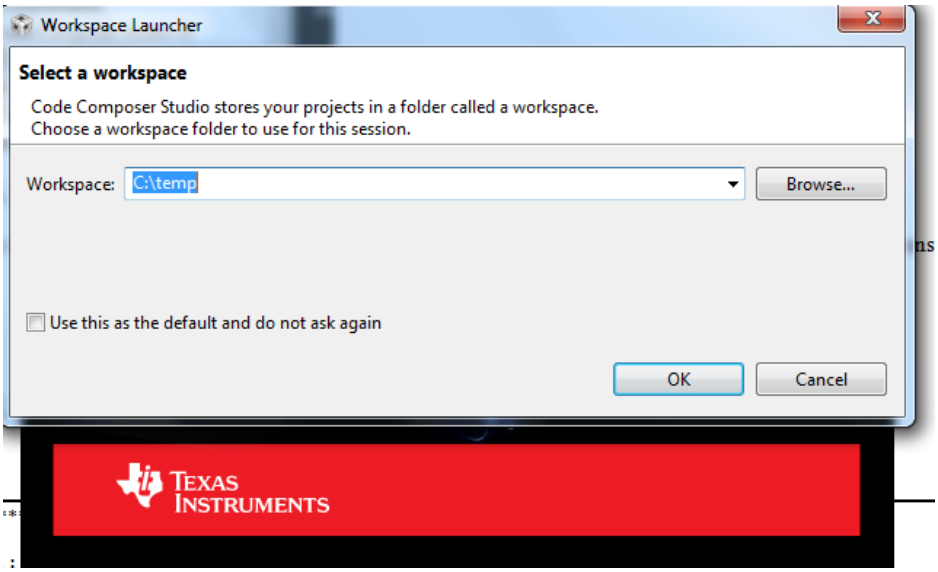

Figure 2. Workspace Launcher

The designer will be presented with the choices shown in Figure 2. The Code Composer again wants to know where the workspace is located. After installation, the authors never change this setting and just hit "OK." The authors chose their "temp" directory. What should come up is the project from lab 1. As a quick recap, the main source file in the "main.c" file. All projects for ARM processor will be completed using C code. Copy the code presented in Figure 3 and replace the old code from main.c with this new code.

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

Project : LAB 3 (which is nothing more than a modification to Lab 1 and 2) ATE
Version : 1.0
Date    : 4/20/2013
Author  : Brian Zufelt / Craig Kief
Company : COSMIAC/UNM
Comments:
This Code is intended to show how to blink different color LEDs on the Stellaris Launchpad
Board


*****************************************************
Chip type               : ARM LM44F120H5QR
Program type            : Firmware
Core Clock frequency      : 80.000000 MHz
*****************************************************/
#include <lm4f120h5qr.h>

// definitions

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

// Lab definitions for the 2 versions of the lab
//#define Lab1
#define Lab2
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 Lab1
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 Lab2

      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<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_RED;           // set LED PORT F pins high
            for(i=0;i<4000000;i++){};              //delay

#endif

}
}
```

Figure 3. The Source File

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. In the case of line 22 in Figure 4, it is used to say, everywhere that LED_RED appears, replace it with 0000 0010. Another example is line 27 and 28 from Figure 4. 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 Lab1, 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 Labx, 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. You will never get to line 15 until line 14 is complete

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.

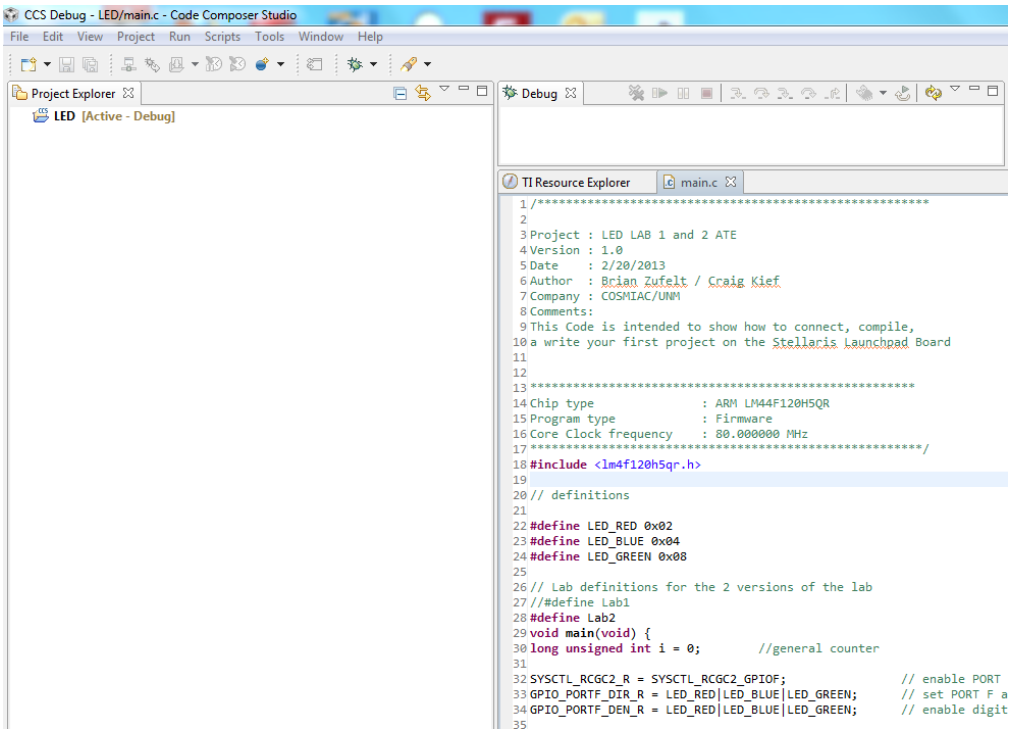

Figure 4. Pasted Code

Paste the code into the main.c project as shown above.

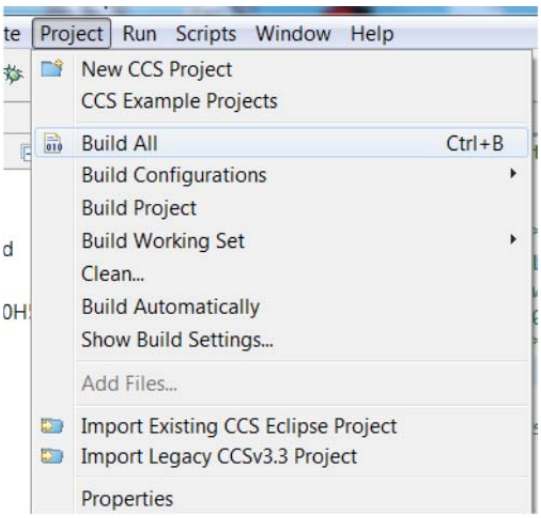

Figure 5. Building the Project

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.

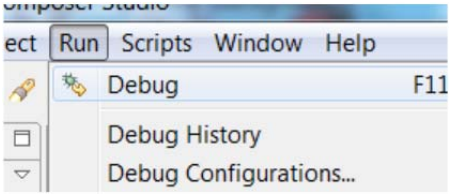

Figure 6. Debugger

Next, it is time to program the board. Click on Run and then Debug as shown in Figure 6. Another shortcut is to hit the F11 key.

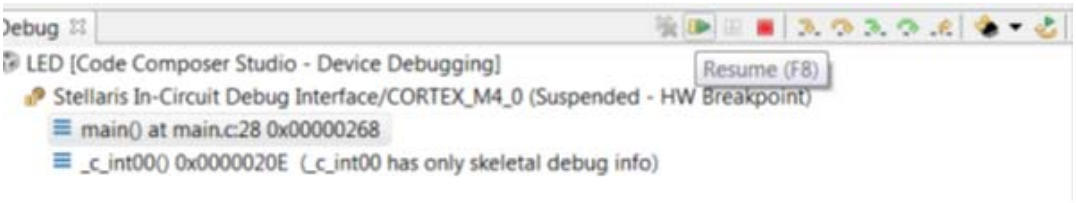

Figure 7. Stopping At Main

Highlight the main file as shown in Figure 7 and then click "Resume." 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: Block Diagram of the Pins Used in Projects