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, 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 is a lab for connecting the hardware the first time. This lab will go through the process of turning on a single light.

Lab Goal
The goal of this lab is provide sufficient instruction and information so that individuals will be able to connect the Stellaris board for the first time and create a simple project that lights an LED.

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

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.

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 1 and the software is installed.

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

This page is for general information:
http://www.ti.com/ww/en/launchpad/stellaris_head.html?DCMP=stellaris-launchpad&HQS=stellaris-launchpad

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". The next series of steps assume this is the first time the evaluation kit has been attached to the computer. It goes through some of the possible problems that can occur the first time that connectivity is achieved.

If after plugging in the board, you receive an error like the one shown in Figure 1, it will be necessary to install the drivers so the evaluation kit can communicate to the computer.

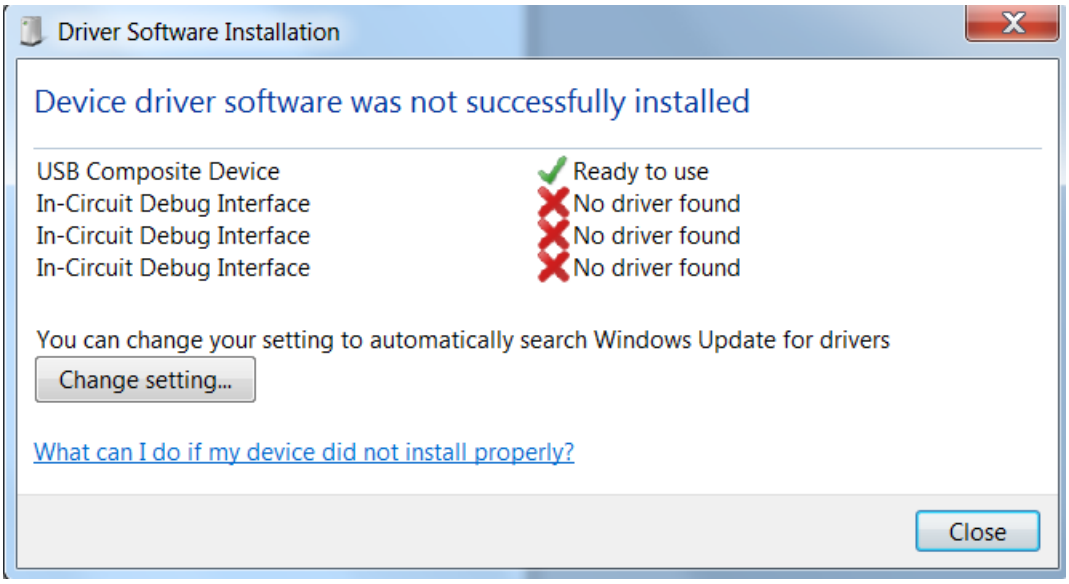

Figure 1. Drivers Not Installed

On your computer, open Device Manager. You may note three unknown "In-Circuit Debug Interface" icons

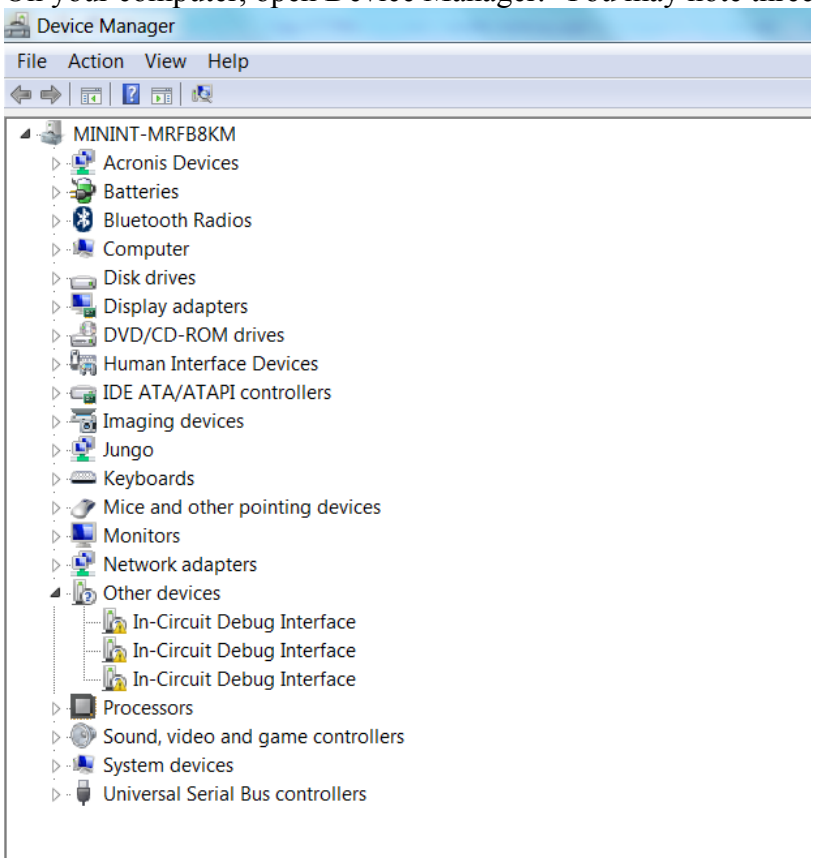

Figure 2. Device Manager

Right click on the top unknown debug interface and choose to update the driver. Scroll to the location where the drivers were unzipped in Lab 1. In my case, this was C:\temp\stellaris_icdi_drivers.

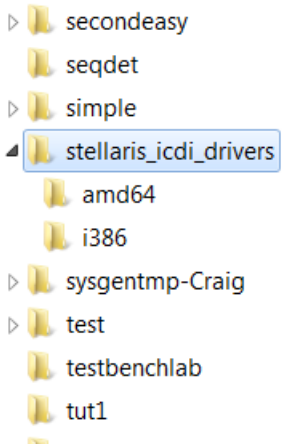

Figure 3. Driver Location

Select this directory and click next. Ensure you have selected to include subfolders as shown in Figure 4.

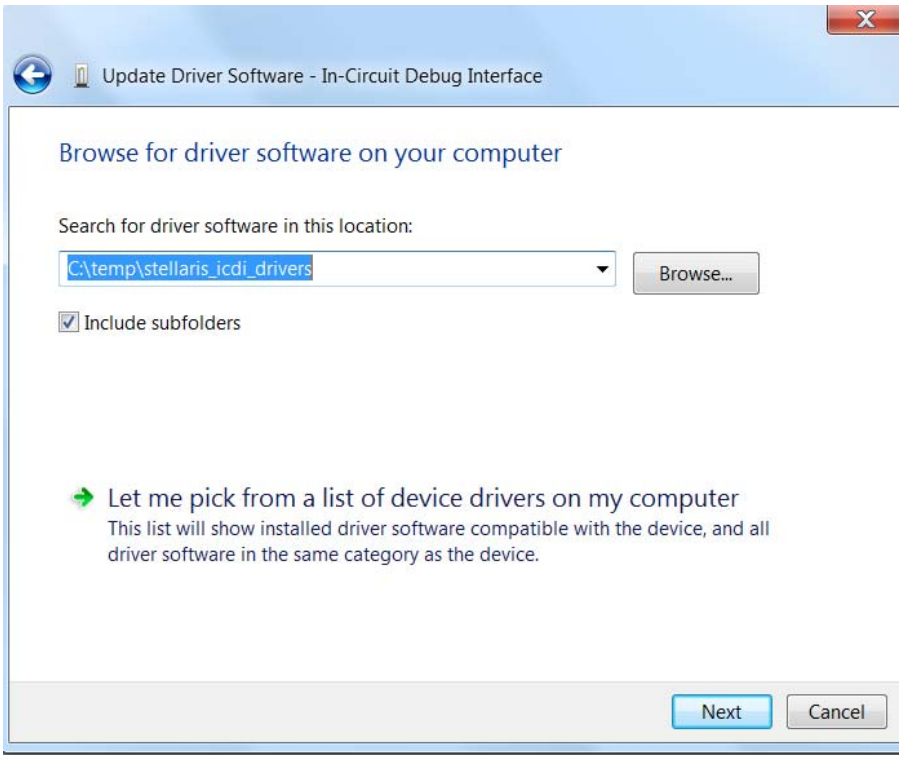

Figure 4. Updating Drivers

The driver should install successfully as shown in Figure 5. Repeat the process two more times to clear all the unresolved driver errors.

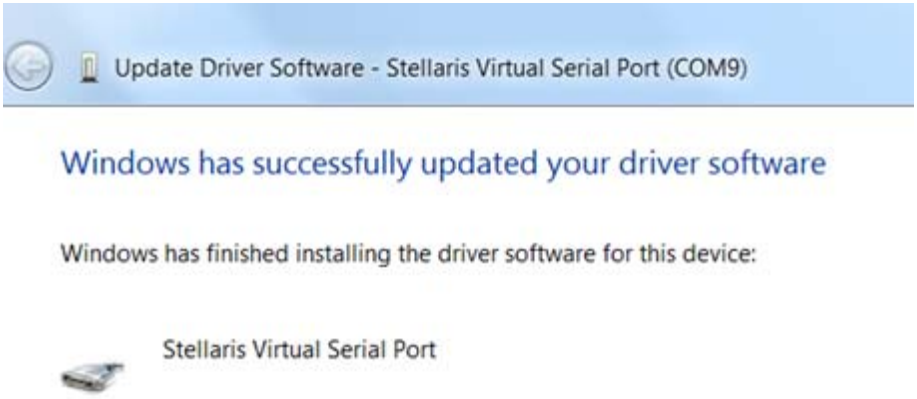

Figure 5. Successful Driver Installation

Figure 6 is how the device manager will look when all the errors (or unknown devices) have been resolved.

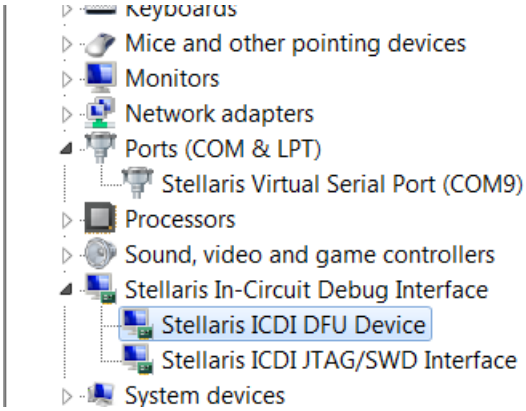

Figure 6. Device Manager

```
/*****************************************************
Project : LED LAB 1 ATE
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 Stellaris Launchpad Board

*****************************************************
Chip type               : ARM LM44F120H5QR
Program type            : Firmware
Core Clock frequency      : 80.000000 MHz
*****************************************************/
// definitions & Included Files
#include <lm4f120h5qr.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 7. The source file

Figure 7 shows the source code that will be used in this project. 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.

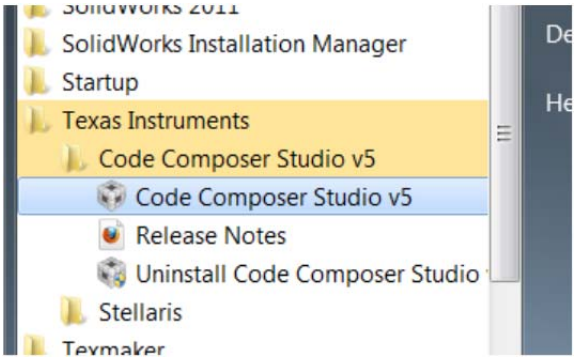

Figure 8. Code composer

This project is acomplished with the Code Composer software. Proceed to start code composer (start→ all programs →) and if you don't see the options shown in figure 8, it is likely that code composer didn't install from Lab 1. If it is necessary to install Code Composer, return to the installer. If McAfee is installed, it may be necessary to temporarily turn off the firewall. During the development of this tutorial, the author (using Windows 7) had to right click on the installer and chose to run it as in administrator. The important part of the installation (which should take approximately ten minutes) is the licensing wizard. When the license wizard comes up, chose the "free license" as shown in Figure 10.

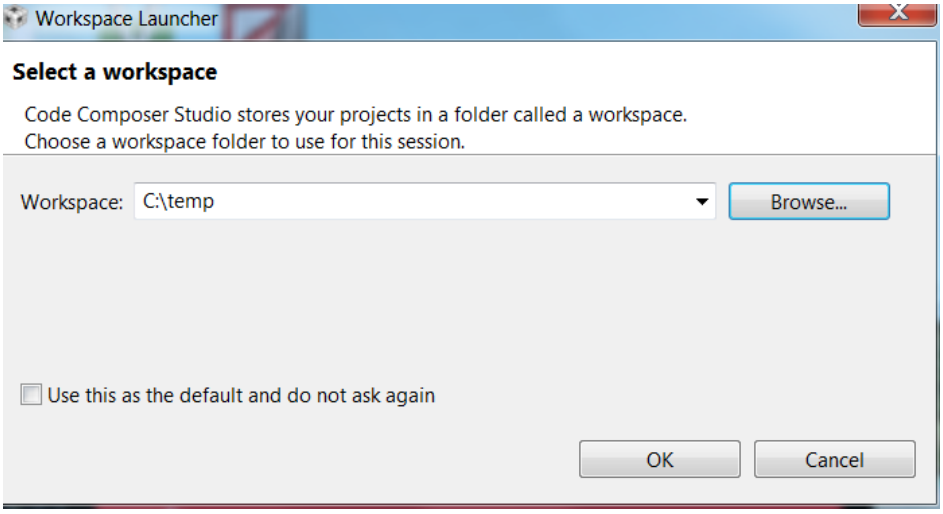

Figure 9. Workspace Location

Once you begin the Code Composer, there are several actions that will need to be accomplished once. The first is shown in Figure 9. This tutorial uses Eclipse as the Development environment but uses Code Composer as the software complier. One of the first things the complier wants to know is what directory to store all project subdirectories in for the rest of time. When developing this tutorial it was felt that the best location was in the "Temp" folder.

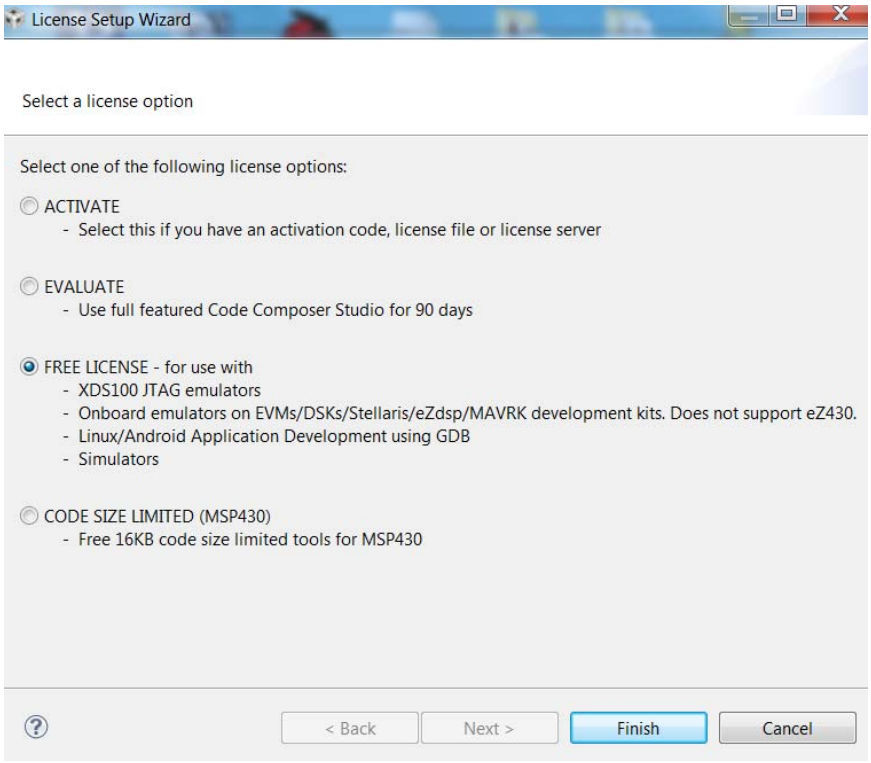

Figure 10. Composer License Wizard

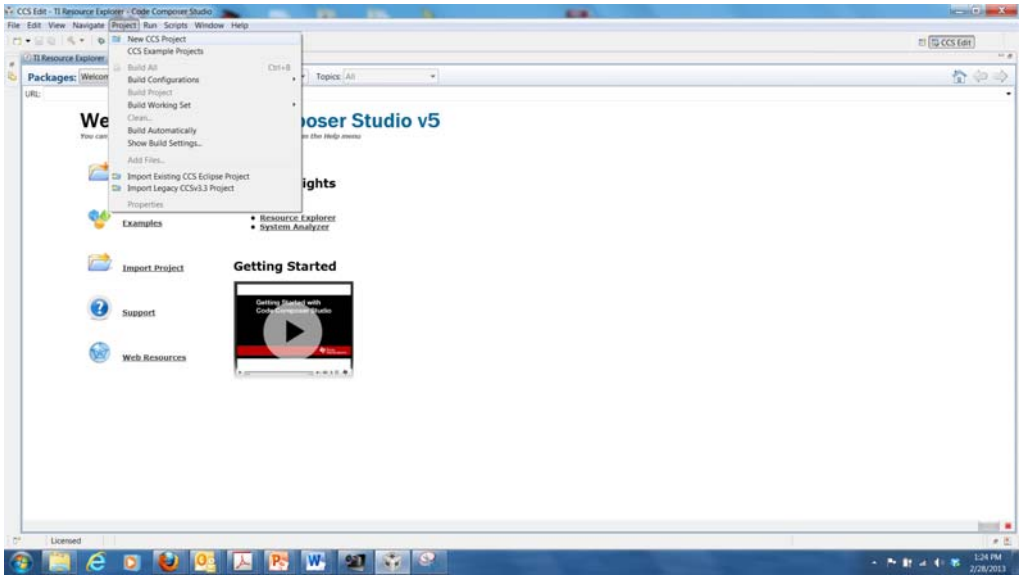

Figure 11. Starting the Project

Now, it is time to create the first project. In Code Composer, go to Project→New CCS Project (as shown in Figure 11).

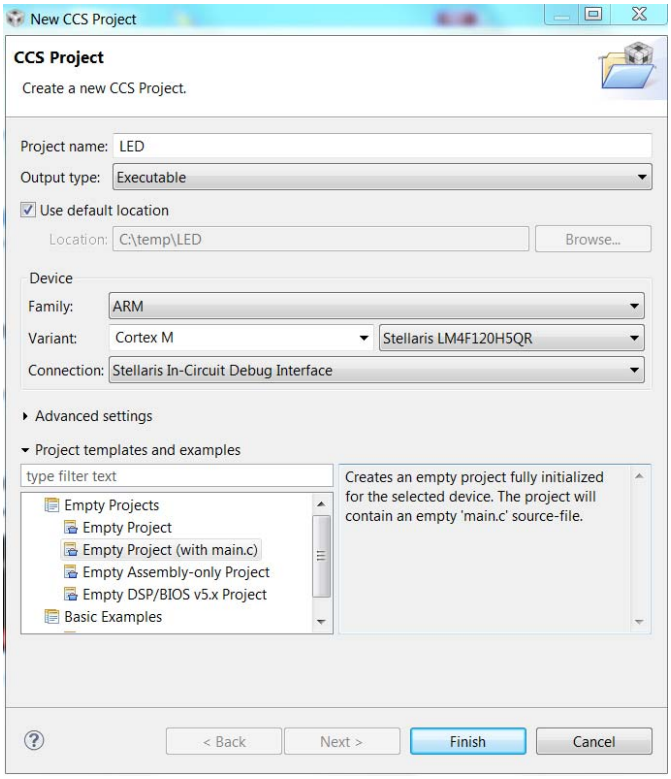

Figure 12. Project Properties

For the tutorial project, select the options as shown in Figure 12. These steps identify the project name. It declares it to be an executable project for our specific version of the ARM processor. For the purposes of this tutorial, the author chose to name the project "LED" however, any name could have been chosen.

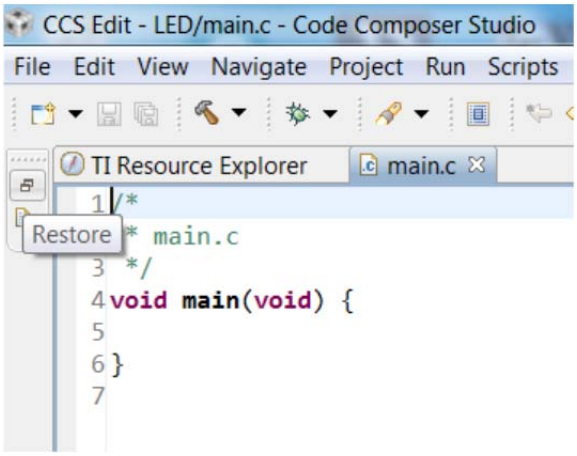

Figure 13. Main.C

Once the project has been created, click on "Restore" button. This will allow the designer to be able to view the main.c program file as shown in Figure 13. Main is the top module in any project in a hierarchical sense.

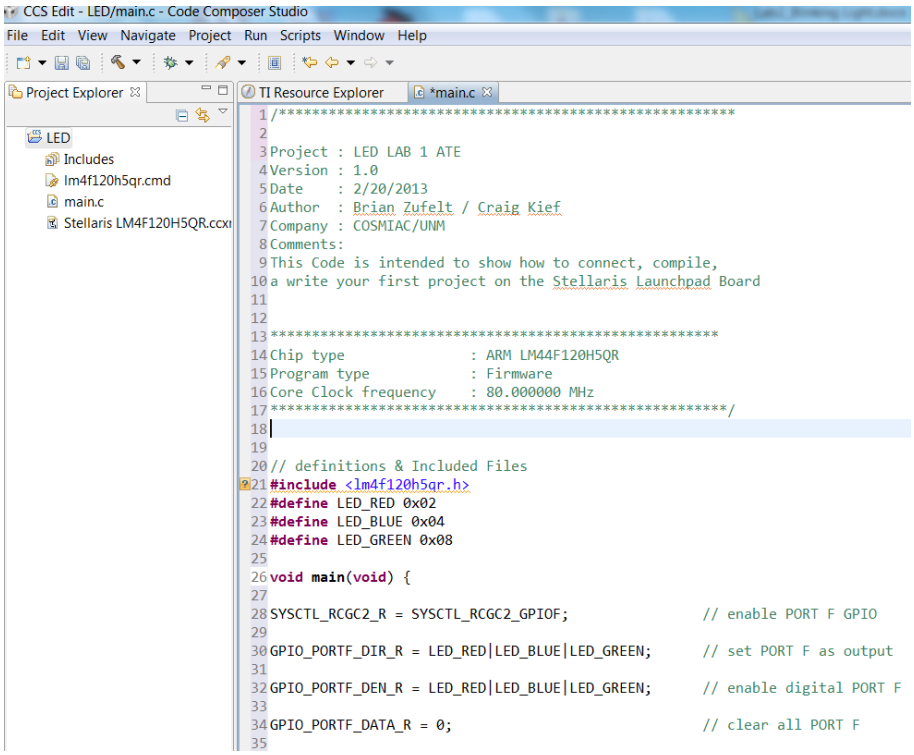

Figure 14. Replace the project code

Copy all the code from Figure 7 and replace the skeleton code that was created in Figure 13. The designer should now have the design as shown in in Figure 14. Since this is the first project completed on this installation one thing has to be corrected. The designer can see on line 21 a "?". This is because the complier doesn't understand the header file for the specific board we are using in this project. This will be fixed a few pages from now but first it is important to look at the rest of the code. 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 website for the manual is sown at the top of Figure 15. Another way to get to the manual is to go to google and search for: EK-LM4F120XL User's Manual. The user should be able to open the pdf shown in Figure 15. This is an excellent resource and should be read at a later time. For now, skip to the schematics shown in in Figure 16 and 17.

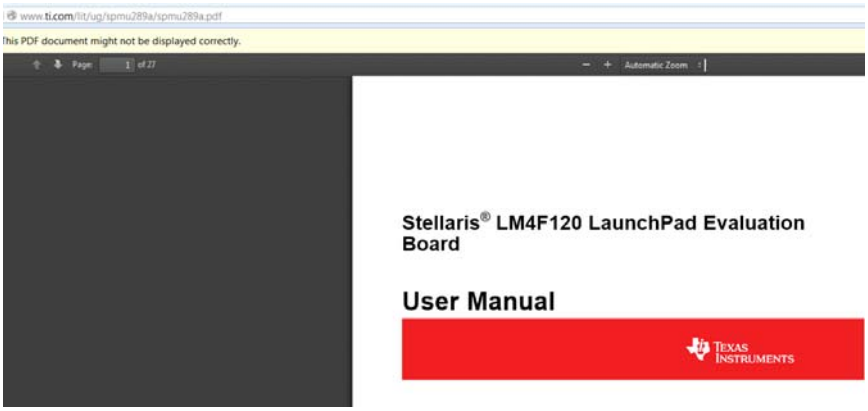

Figure 15. The User's Manual for the Evaluation Kit

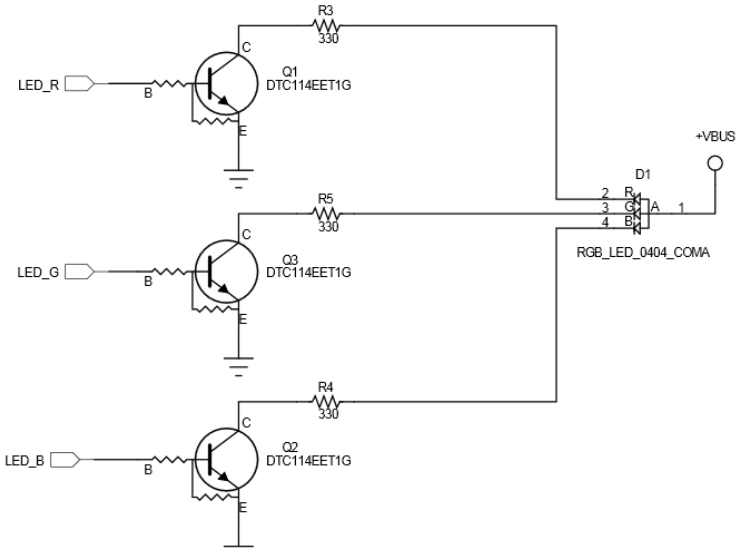

Figure 16. Three LED's in One

The Stellaris 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 16. This first tutorial will combine the outputs of these three LEDs to produce a white light.

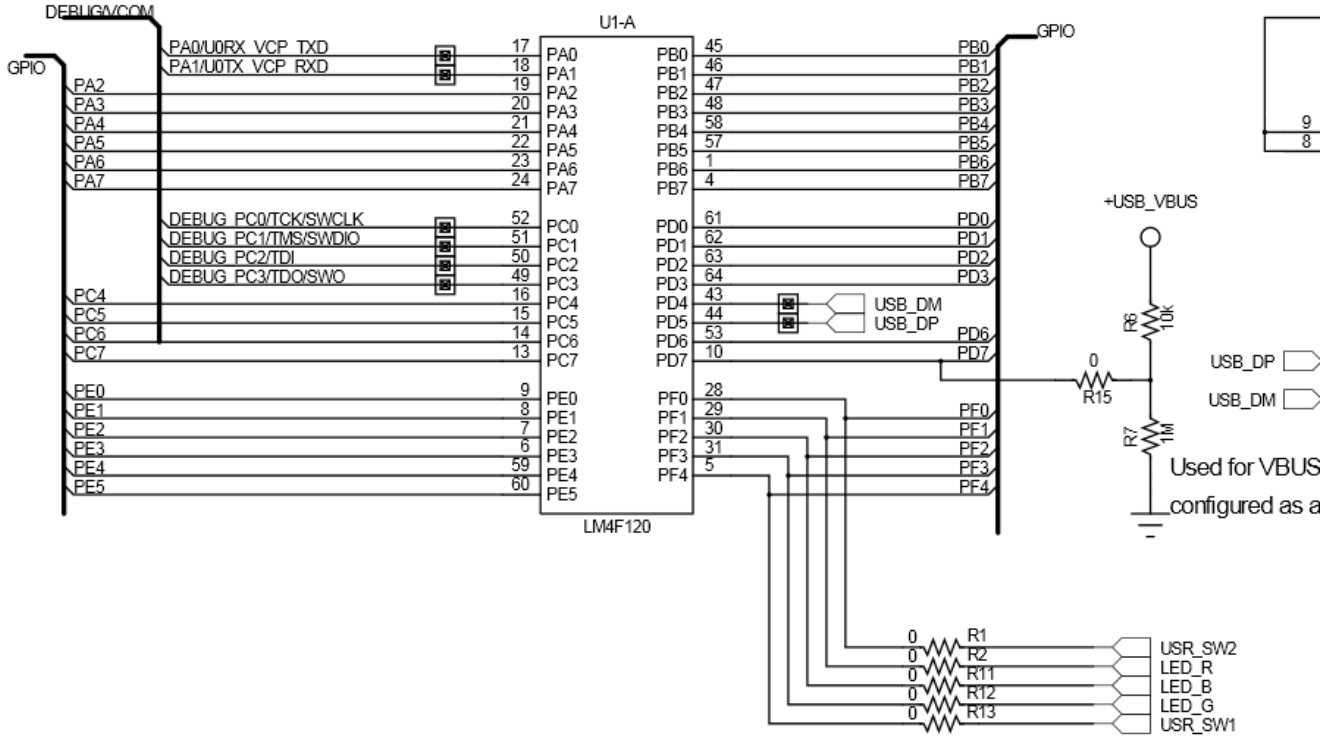

Figure 17. Port F of the Chip

Figure 17 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 "000 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. The second sends "1s" to the part which sets it as an output. The third tells port F that it is digital. The next two lines do the following:

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)
```

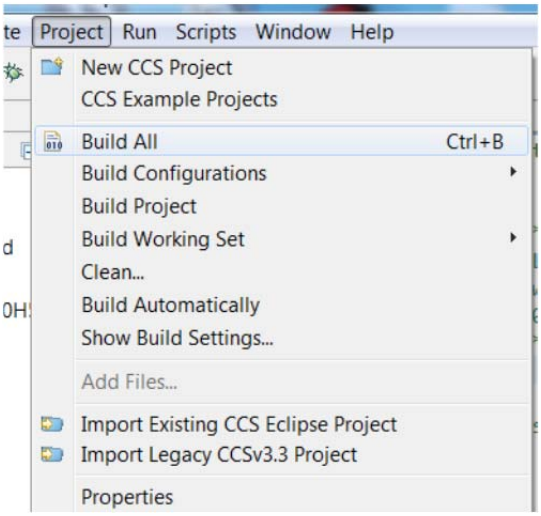

Figure 18. Building the project

The next step is to build the project into an executable. Click on Project→Build All. Back in Figure 14 there was a question mark next to an include (line 21). 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 19.

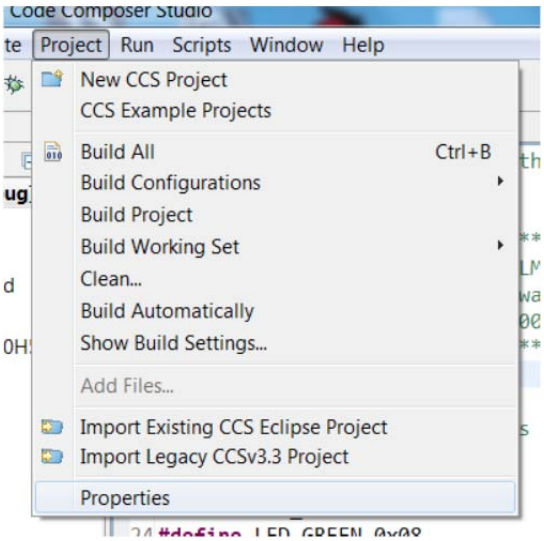

Figure 19. Properties

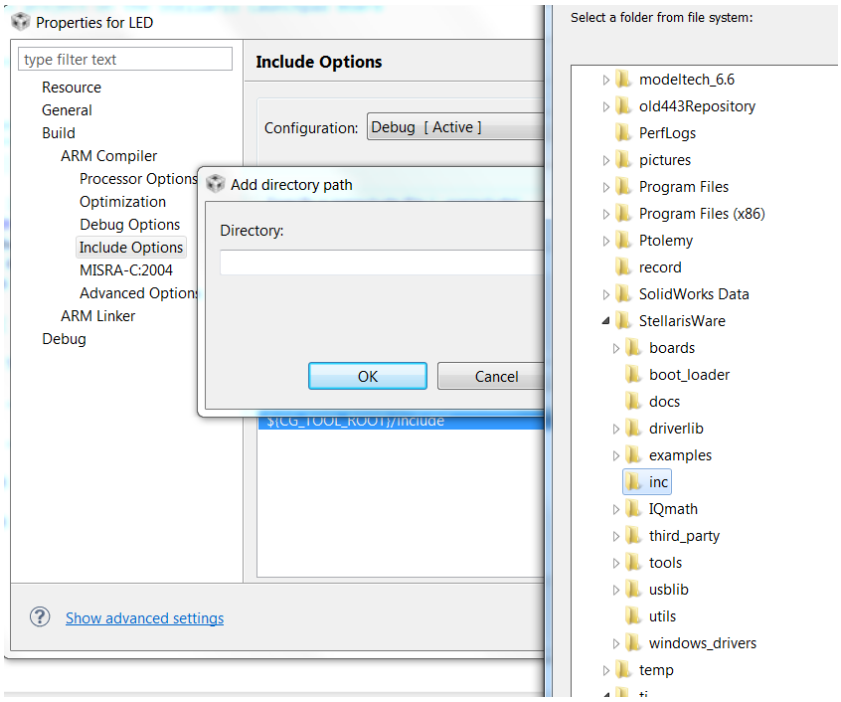

Figure 20. Adding the path for the header file

As shown in Figure 20, click on the "Include Options" selection. Click to add a directory path scan to the "inc" directory shown in Figure 20 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.

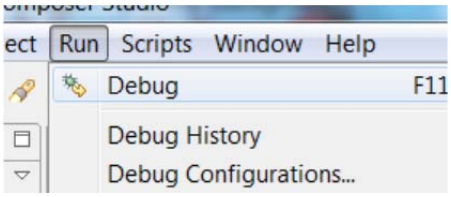

Figure 21. Debugger

Next, it is time to program the board. Click on Run and then Debug as shown in Figure 21.

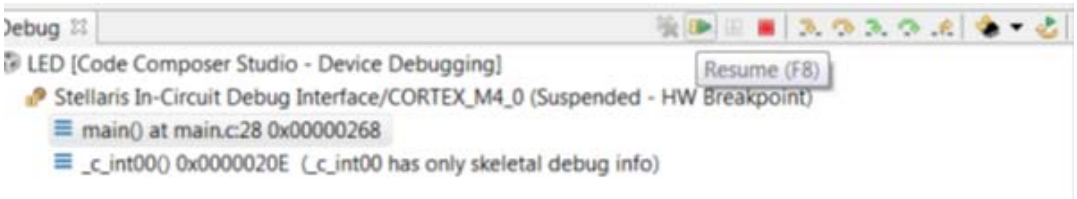

Figure 22. Stopping at main

Highlight the main file as shown in Figure 22 and then click "Resume." The white light should be on.

```
23 #define LED_BLUE 0x04
24 #define LED_GREEN 0x08
25
26 void main(void) {
27
28 SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;                // er
29
30 GPIO_PORTF_DIR_R = LED_RED|LED_BLUE|LED_GREEN;      // se
31
32 GPIO_PORTF_DEN_R = LED_RED;//|LED_BLUE|LED_GREEN;       /
33
```

Figure 23. 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, I comment out (remove) the other two colors as shown in Figure 23 and rerun.

Attachment 1: Block Diagram of the Pins Used in Projects