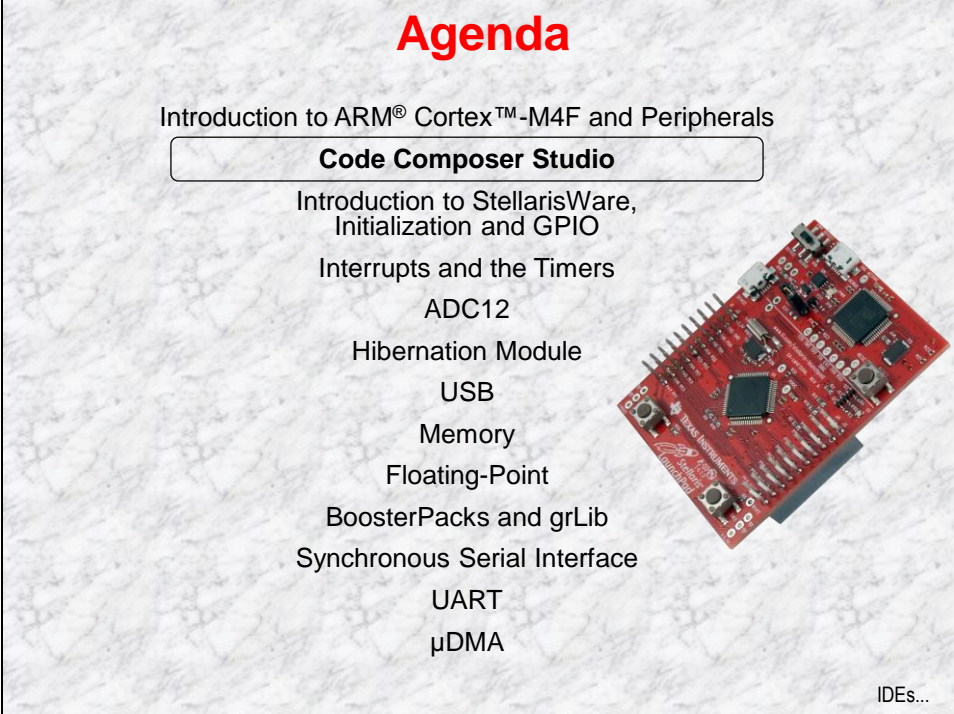


# Code Composer Studio

---

## Introduction

This chapter will introduce you to the basics of Code Composer Studio. In the lab, we will explore some Code Composer features.



**Agenda**

Introduction to ARM® Cortex™-M4F and Peripherals

**Code Composer Studio**

Introduction to StellarisWare,  
Initialization and GPIO

Interrupts and the Timers

ADC12

Hibernation Module

USB

Memory

Floating-Point


BoosterPacks and grLib

Synchronous Serial Interface

UART

μDMA

IDEs...







# Chapter Topics

<b>Code Composer Studio .....</b>	<b>2-1</b>
<i>Chapter Topics.....</i>	<i>2-2</i>
<i>Stellaris Development Tools .....</i>	<i>2-3</i>
<i>Code Composer Studio .....</i>	<i>2-4</i>
<i>Lab2: Code Composer Studio .....</i>	<i>2-7</i>
Objective.....	2-7
Load the Lab 2 Project.....	2-8
<i>LM Flash Programmer .....</i>	<i>2-15</i>
Creating a bin File for the Flash Programmer .....	2-17
<i>Hints and Tips.....</i>	<i>2-18</i>

# Stellaris Development Tools

## Development Tools for Stellaris MCUs

				
Eval Kit License	30-day full function. Upgradeable	32KB code size limited. Upgradeable	32KB code size limited. Upgradeable	Full function. Onboard emulation limited
Compiler	GNU C/C++	IAR C/C++	RealView C/C++	TI C/C++
Debugger / IDE	gdb / Eclipse	C-SPY / Embedded Workbench	µVision	CCS/Eclipse-based suite
Full Upgrade	99 USD personal edition / 2800 USD full support	2700 USD	MDK-Basic (256 KB) = €2000 (2895 USD)	445 USD
JTAG Debugger		J-Link, 299 USD	U-Link, 199 USD	XDS100, 79 USD

What is CCS?...

# Code Composer Studio

## What is Code Composer Studio?

### Integrated development environment for TI embedded processors

- ◆ Includes debugger, compiler, editor, simulator, OS...
- ◆ The IDE is built on the Eclipse open source software framework
- ◆ Extended by TI to support device capabilities

### CCSv5 is based on “off the shelf” Eclipse (version 3.7 in CCS 5.2)

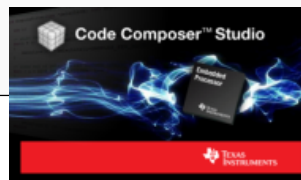
- ◆ Uses **unmodified** version of Eclipse
  - ◆ TI contributes changes directly to the open source community
- ◆ Drop in Eclipse plug-ins from other vendors or take TI tools and drop them into an existing Eclipse environment
- ◆ Users can take advantage of all the latest improvements in Eclipse

### Integrate additional tools

- ◆ OS application development tools (Linux, Android, Sys/BIOS...)
- ◆ Code analysis, source control...

### Runs under Windows and Linux

**\$445 single seat. \$99/year subscription fee**



User Interface Modes...

## User Interface Modes

### Simple Mode

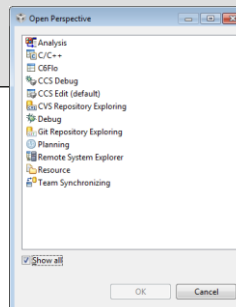
- ◆ By default CCS will open in simple/basic mode
- ◆ Simplified user interface with far fewer menu items, toolbar buttons
- ◆ TI supplied CCS Edit and CCS Debug Perspectives

### Advanced Mode

- ◆ Uses default Eclipse perspectives (similar to what existed in CCSv4)
- ◆ Recommended for users integrating other Eclipse based tools into CCS

### Switching Modes

- ◆ On the CCS menu bar, select Window → Open Perspective → Other...  
Check the “Show all” checkbox  
“C/C++” and “Debug” are the advanced perspectives



Common Tasks...

## Common Tasks

### Creating New Projects

- ◆ Very simple to create a new project for a device using a template

### Build options

- ◆ Simplified build options dialog from earlier CCS versions
- ◆ Updates to options are delivered via compiler releases and not dependent on CCS updates

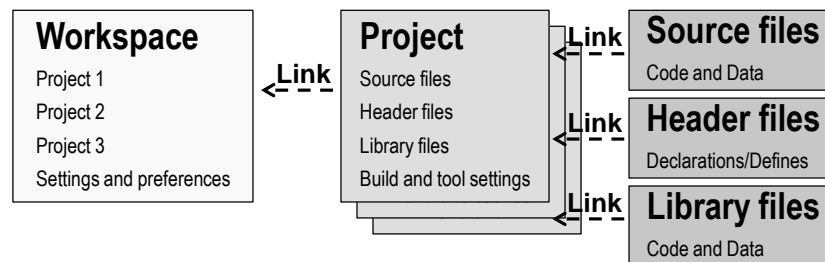
### Sharing projects

- ◆ Easy for users to share projects, including working with version control (portable projects)
- ◆ Setting up linked resources has been simplified from earlier CCS versions



Workspaces and Projects...

## Workspaces and Projects



A workspace contains your settings and preferences, as well as links to your projects. Deleting projects from the workspace deletes the links, not the files\*

A project contains your build and tool settings, as well as links to your input files. Deleting files from the workspace deletes the links, not the files\*

**\* Unless you have located or copied files into the workspace**

Project Wizard...

## Project Wizard

**Single page wizard for majority of users**

- ◆ Next button will show up if a template requires additional settings

**Debugger setup included**

- ◆ User chooses location, device and connection
- ◆ A modifiable cxxml file is created

**Simple by default**

- ◆ Compiler version, endianness... are under advanced settings

Add Files...

## Adding Files to Projects

◆ Add Files to Project allows users to control how the file is added to the project

◆ Linking Files using built-in macros allows easy creation of portable projects

Lab...


## Lab2: Code Composer Studio

### Objective

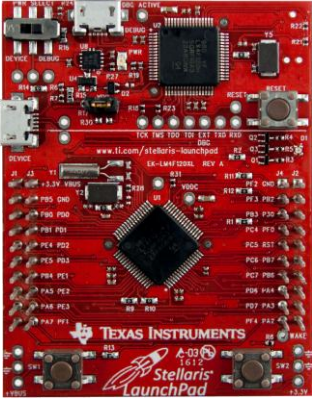
The objective of this lab exercise is to explore the basics of how to use Code Composer Studio.

This lab will not discuss the actual C code. The following chapter will explore the code.

### Lab 2: Code Composer Studio



USB Emulation Connection




- ◆ Create a new project
- ◆ Experiment with some CCS features
- ◆ Use the LM Flash Programmer

Agenda ...

## Load the Lab 2 Project

### Open Code Composer Studio

1. Double click on the Code Composer shortcut on your desktop to start CCS. 

When the “Select a workspace” dialog appears, browse to your My Documents folder:

(In WinXP) C:\Documents and Settings\\My Documents

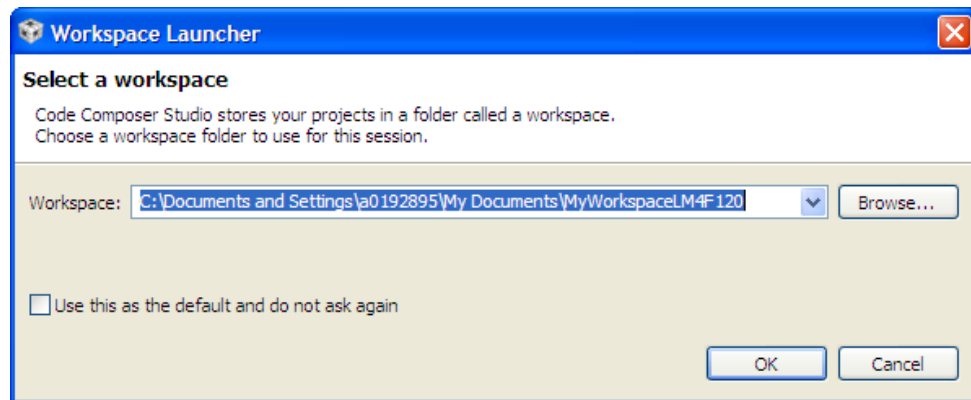
(In Win7) C:\Users\\My Documents

Obviously, replace <user> with your own username. Click OK.

The name of your workspace isn’t critical, but let’s use MyWorkspaceLM4F120.

Do not check the “Use this as the default and do not ask again” checkbox. (If at some point you accidentally check this box, it can be changed in CCS) The location of the workspace folder is not important, but to keep your projects portable, you want to locate it outside of the StellarisWare directory.

**Note: The following screen captures show the correct options for WinXP. Few, if any differences exist for Win7 and Vista.**



Click OK.

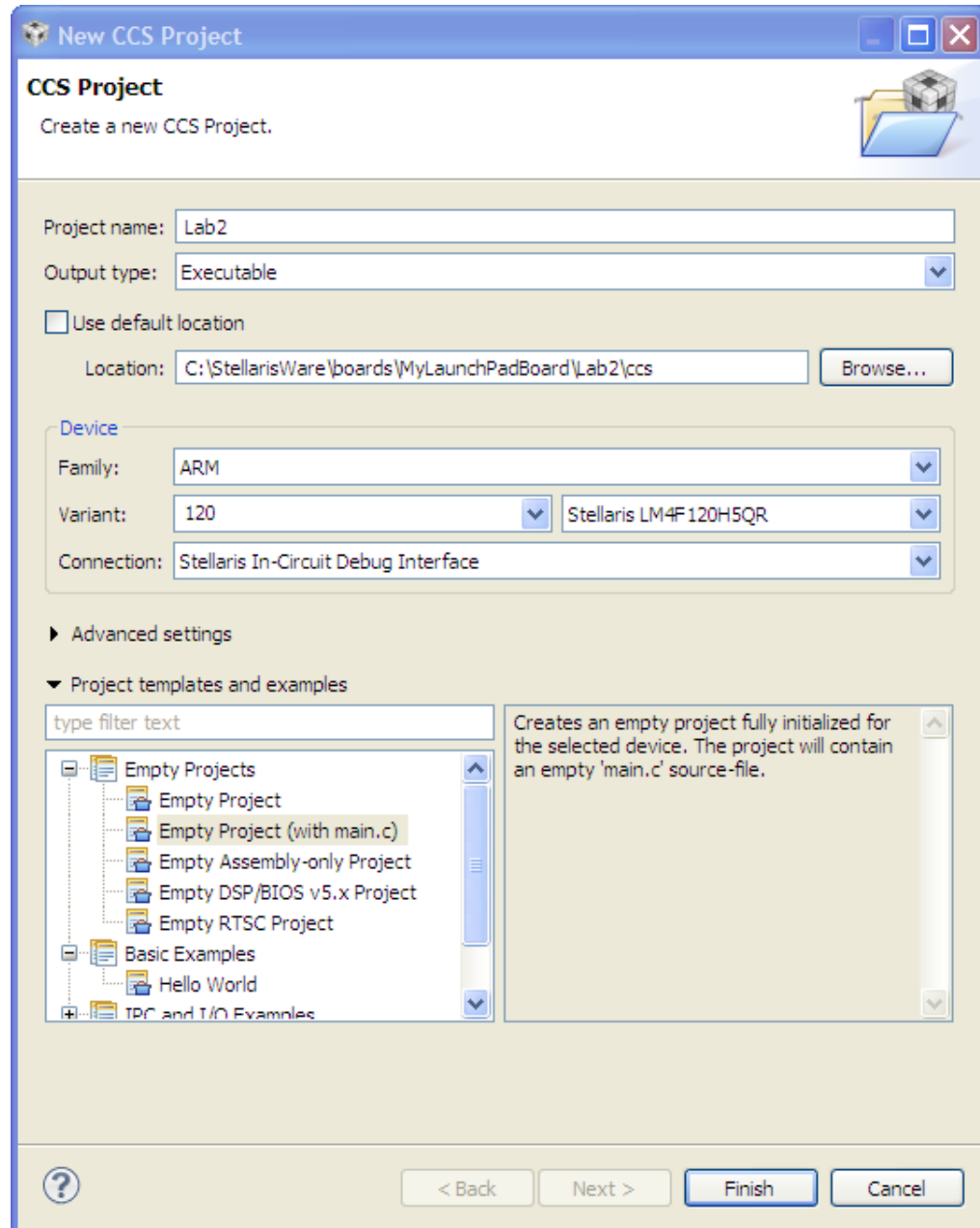
If you haven’t already licensed Code Composer, you’ll be asked to do so in the next few installation steps. When that happens, select “Evaluation”. As long as your PC is connected to the LaunchPad board, Code Composer will have full functionality, free of charge. You can go back and change the license if needed by clicking Help → Code Composer Studio Licensing Information → Upgrade tab → Launch License Setup...

When the “TI Resource Explorer” and/or “Grace” windows appear, close their tabs. At this time these tools only support the MSP430.



## Create Lab2 Project

2. Maximize Code Composer. On the CCS menu bar select File → New → CCS Project. Make the selections shown below. Make sure to uncheck the “Use default location” checkbox and select the correct path. **This step is important to making your project portable and in order for the links to work correctly.** Type “120” in the variant box to bring up the four versions of the device. Select “Empty Project (with main.c)” for the project template. Click Finish.



3. The `main.c` file will be open in the editor tab. Delete the contents and type or copy/paste the following code into the file. Don't worry about the code now; we'll go over this in detail in lab 3. Note the question marks that appear to the left of the include statements. These indicate that Code Composer does not know the path to these resources. We'll fix that in a moment.

```
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"

int main(void)
{
    int LED = 2;


    SysCtlClockSet(SYSCTL_SYSDIV_4|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    while(1)
    {
        // Turn on the LED
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, LED);

        // Delay for a bit
        SysCtlDelay(2000000);

        // Cycle through Red, Green and Blue LEDs
        if (LED == 8) {LED = 2;} else {LED = LED*2;}
    }
}
```


Click the Save button  on the menu bar to save your work. If you are having problems, you can find this code in your `Lab2/ccs` folder in file `main.txt`.

If the indentation of your code doesn't look right, press `Ctrl-A` (on your keyboard) to select all the code. Then right-click on it and select `Source → Correct Indentation`.

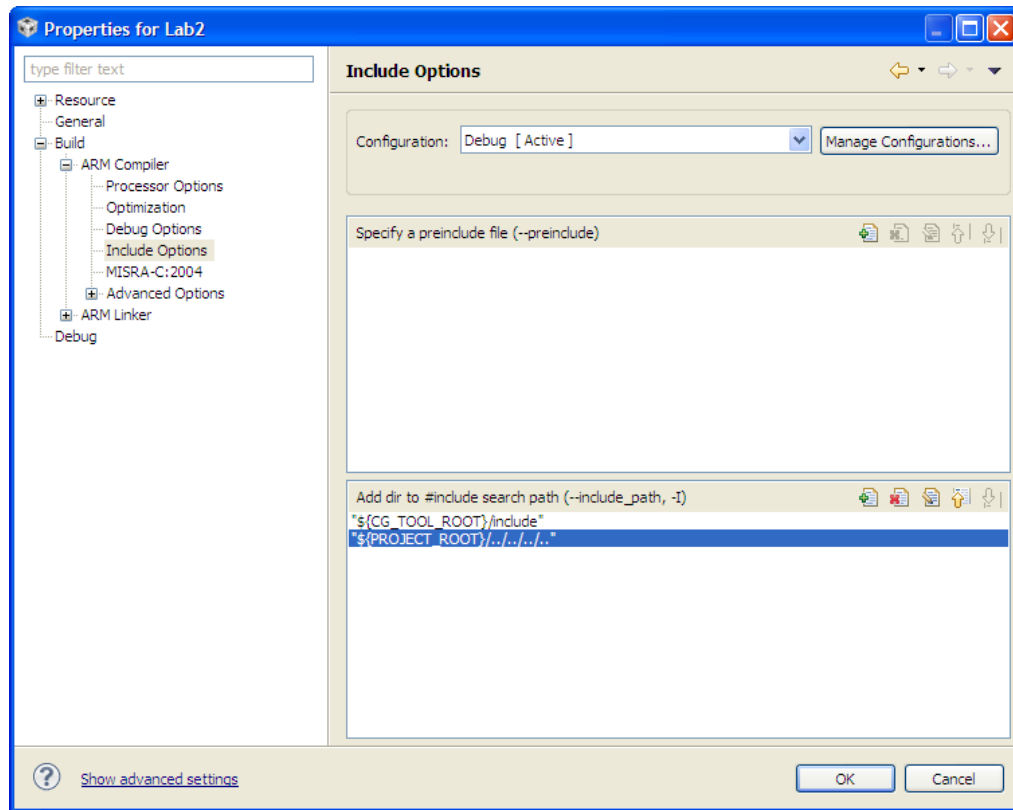
4. Expand the project in the Project Explorer pane (on the left side of your screen) by clicking the `+` or `▾` next to `Lab2`. This list shows all the files that are used to build the project. One of those files is `startup_ccs.c` that we included in your lab folder (this file is available in every StellarisWare example). Double-click on the file to open it for editing. This file defines the stack and the interrupt vector table structure among other things. These settings are essential for Code Composer to build your project. Close the editor window by clicking on the **X** in the tab at the top of the editor window. Do not save any changes if you accidentally made any.

## Set the Build Options

- Remember those question marks in the code? The next two steps will tell Code Composer where to find the resources we need to compile the code.

Right-click on Lab2 in the Project Explorer pane and select Properties (the leftmost pane in CCS). Click **Include Options** under **ARM Compiler**. In the bottom, **include search path** pane, click the Add button  and add the following include search path. You may want to copy/paste from the workbook pdf for the next two steps.

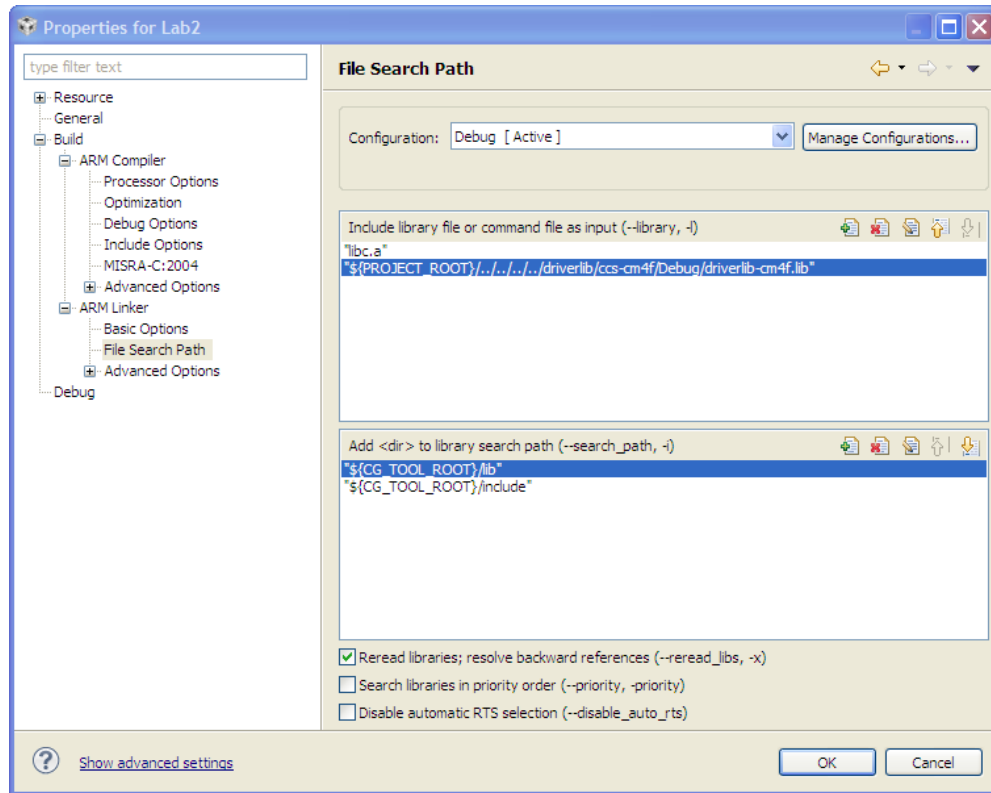
`${PROJECT_ROOT}/../../../../..`



This path allows the compiler to correctly find the `driverlib` folder, which is four levels up from your project folder. Note that if you did not place your project in the correct location, this link will not work.

- Under **ARM Linker** click **File Search Path**. Add the following include library file to the top window:



`${PROJECT_ROOT}/../../../../driverlib/ccs-cm4f/Debug/driverlib-cm4f.lib`




This step allows the linker to correctly find the lib file. Note that if you did not place your project in the correct location, this link will not work either.

Click OK to save your changes.



## Run the Code

7. Make sure that your LaunchPad board is plugged in. Check that Lab2 is the Active Project by clicking on the project in the Project Explorer pane Click the Debug  button on the CCS menu bar to build and download the Lab2 project. When the process completes, CCS will be in the Debug perspective. (Note the two tabs in the upper right of your screen ... drag them to the left a little so you can see both of them completely) You can create as many additional perspectives as you like for your specific needs. Only the Debug and Edit perspectives are pre-defined.
8. Click the Run  button on the CCS menu bar to run the code. Observe the tri-color LED blinking red, green, and blue on your LaunchPad board.

## Some CCS Features

9. Click the Suspend  button on the CCS menu bar. If the code stops with a “No source available ...” indication, click on the main.c tab. Most of the time in the `while()` loop is spent inside the delay function and that source file is not linked into this project.

### 10. Breakpoints

In the code window in the middle of your screen, double-click in the gray area to the left of the line number of the `GPIOPinWrite()` instruction to set a breakpoint (it will look like this: ). Click the Resume  button to restart the code. The program will stop at the breakpoint and you will see an arrow on the left of the line number, indicating that the program counter has stopped on this line of code. **Note that the current ICDI driver does not support adding/removing breakpoints while the processor is running.** Click the Resume button a few times or press the F8 key to run the code. Observe the LED on the LaunchPad board as you do this.

### 11. Register View

Click on View → Registers to see the core and peripheral register values. Resize the window if necessary. Click on the plus sign on the left to view the registers. Note that non-system peripherals that have not been enabled cannot be read. In this project you can view Core Registers, GPIO\_PORTA (where the UART pins are), GPIO\_PORTF (where the LEDs and pushbuttons are located), HIB, FLASH\_CTRL, SYSCTL and NVIC.

### 12. Memory View

Click on View → Memory Browser to examine processor memory. Type 0x00 in the entry box and press Enter. You can page through memory and you can click on a location to directly change the value in that memory location.


### 13. Expressions View

Make sure that you can see the Expression pane in the upper right hand corner of your screen. You may have to click on the Expressions tab. Right-click in the pane and select Remove All to make sure there are no existing watch expressions.

In your code window, double-click on the variable **LED** around line 18. Right click on the selected variable and select Add Watch Expression and then click OK. The window on the upper right will switch to the Expression view and you should see the variable listed. Run the code several times. Each time your code execution reaches the breakpoint, the watch will update. Updated values are highlighted with a yellow background.

Expression	Type	Value	Address
(x) LED	int	2	0x200000F8
+ Add new expression			

Remove all the breakpoints you have set at once by clicking Run → Remove All Breakpoints from the menu bar. Again, breakpoints can only be removed when the processor is not running.

- Click on Terminate  to return to the editor perspective. Right-click on Lab2 in the Project Explorer pane and select Close Project to close the project. Minimize CCS.

## LM Flash Programmer

15. LM Flash Programmer is a standalone programming GUI that allows you to program the flash of a Stellaris device through multiple ports. Creating the files required for this is a separate build step in Code Composer Studio that is shown on the next page.

If you have not done so already, install the LM Flash Programmer onto your PC.

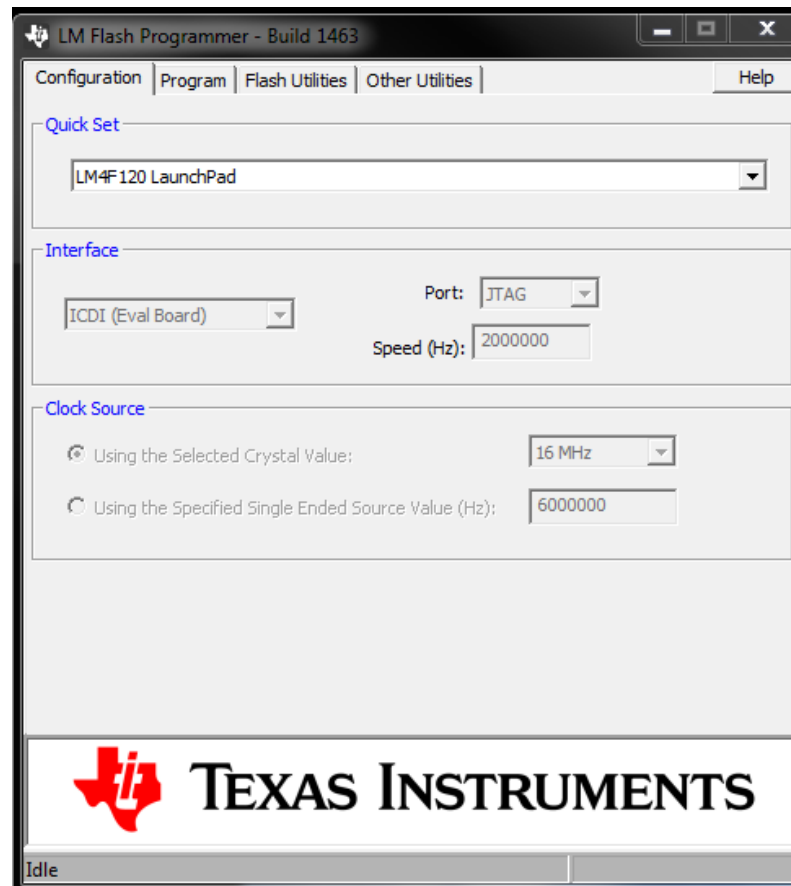
16. Make sure that Code Composer Studio is not actively running code in the CCS Debug perspective ... otherwise CCS and the Flash Programmer may conflict for control of the USB port.

There should be a shortcut to the **LM Flash Programmer** on your desktop, double-click it to open the tool. If the shortcut does not appear, go to Start → All Programs → Texas Instruments → Stellaris → LM Flash Programmer and click on LM Flash Programmer.



17. Your evaluation board should currently be running the Lab2 application. If the User LED isn't blinking, press the RESET button on the board. We're going to program the original application back into the LM4F120H5QR.

Click the Configuration tab. Select the **LM4F120 LaunchPad** from the **Quick Set** pull-down menu under the **Configuration tab**. See the user's guide for information on how to manually configure the tool for targets that are not evaluation boards.

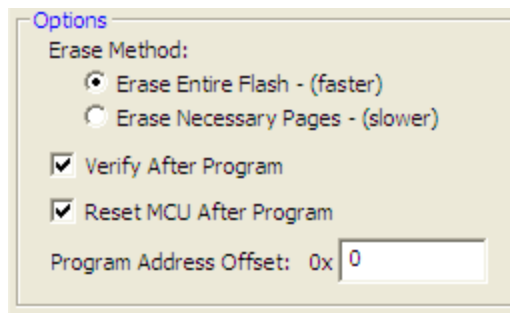


18. Click on the Program tab. Then click the Browse button and navigate to:

C:\StellarisWare\boards\ek-lm4f120XL\qs-rgb\ccs\Debug\qs-rgb.bin

This is the application that was programmed into the flash memory of the LM4F120XL during the evaluation board assembly process.

Note that there are applications here which have been built with each supported IDE. Make sure that the following checkboxes are selected:



19. Click the Program button.

You should see the programming and verification status at the bottom of the window. After these steps are complete, the quickstart application should be running on your evaluation kit.

20. Close the LM Flash Programmer.



## Optional: Creating a bin File for the Flash Programmer

If you want to create a bin file for use by the stand-alone programmer in any of the labs in this workshop or in your own project, use these steps below. Remember that the project will have to be open before you can change its properties.

### In Code Composer 5.2 and Earlier:

In Code Composer, in the Project Explorer, right-click on your project and select Properties. On the left, click Build and then the Steps tab. Paste the following commands into the Post-build steps Command box:

```
"${CCS_INSTALL_ROOT}/utils/tiobj2bin/tiobj2bin"  
"${BuildArtifactFileName}" "${BuildArtifactFileName}.bin"  
"${CG_TOOL_ROOT}/bin/ofd470" "${CG_TOOL_ROOT}/bin/hex470"  
"${CCS_INSTALL_ROOT}/utils/tiobj2bin/mkhex4bin"
```

### In Code Composer 5.3 and Later:

In Code Composer, in the Project Explorer, right-click on your project and select Properties. On the left, click Build and then the Steps tab. Paste the following commands into the Post-build steps Command box:

```
"${CCS_INSTALL_ROOT}/utils/tiobj2bin/tiobj2bin"  
"${BuildArtifactFileName}" "${BuildArtifactFileName}.bin"  
"${CG_TOOL_ROOT}/bin/armofd" "${CG_TOOL_ROOT}/bin/armhex"  
"${CCS_INSTALL_ROOT}/utils/tiobj2bin/mkhex4bin"
```

Each command is enclosed by quotation marks and there is a space between each one. These steps will run after your project builds and the bin file will be in the ...Labx/ccs/debug folder. You can access this in the CCS Project Explorer in your project by clicking the Debug folder.



You're done.

## Hints and Tips

There are several issues and errors that users commonly run into during the class. Here are a few and their solutions:

### 1. Header files can't be found

When you create the main.c file and include the header files, CCS doesn't know the path to those files and will tell you so by placing a question mark left of those lines. After you change the Compiler and Linker options, these question marks should go away and CCS should find the files during the build. If CCS reports that your header files can't be found, check the following:

- a. Under the Project Properties click Resource on the left. Make sure that your project is located in ...\\MyLaunchPadBoard\\Labx\\ccs. If you located it in the Lab9 folder you can adjust the Include and File Search paths. If you located the project in the workspace, your best bet is to remake the project.
- b. Under the Project Properties, click on Include Options. Make sure that you added the correct search paths to the bottom window.
- c. Under the Project Properties, click on File Search Path. Make sure that you placed the path to the include library file(s) in the top window.

### 2. Unresolved symbols

This is usually the result of step 1c above or you are using a copy of the startup\_ccs.c file that includes the ISR name used in the Interrupts lab. You'll have to remove the extern declaration and change the timer ISR link back to the default.

### 3. Frequency out of range

This usually means that CCS tried to connect to the evaluation board and couldn't. This can be the result of the USB drivers or a hardware issue:

- a. Unplug and re-plug the board from your USB port to refresh the drivers.
- b. Open your Device Manager and verify that the drivers are correctly installed.
- c. Assure that your emulator cable is connected to the DEBUG microUSB port, not the DEVICE port, and make sure the PWR SELECT switch is set to the rightmost DEBUG position.
- d. Your board should be connected by its orange emulator connector, not the user connector. Also, check your USB cable. It may be faulty.

### 4. Error loading dll file

This can happen in Windows7 when attempting to connect to the evaluation board. This is a Win7 driver installation issue and can be resolved by copying the files: FTCJTAG.dll and ftd2xx.dll to:

```
C:\CCS5.x\ccsv5\ccs_base\DebugServer\drivers
```

and

```
C:\Windows\System32
```

Download these files from [http://www.ti.com/tool/lm\\_ftdi\\_driver](http://www.ti.com/tool/lm_ftdi_driver).

**5. Program run tools disappear in the Debug perspective**

The tools aren't part of the perspective, but part of the Debug window. Somehow you closed the window. Click View → Debug from the menu bar.

**6. CCS doesn't prompt for a workspace on startup**

You checked the "don't ask anymore" checkbox. You can switch workspaces by clicking File → Switch workspace ... or you can do the following: In CCS, click Window → Preferences. Now click the + next to General, Startup and Shutdown, and then click Workspaces. Check the "Prompt for workspace on startup" checkbox and click OK.

**7. The windows have changed in the CCS Edit or Debug perspective from the default and you want them back**

On the CCS menu bar, click Window → Reset Perspective ... and then Yes.

