

# Lab-0: Getting started with Tiva C Series Launchpad and Code Composer Studio IDE

ERTS Lab, CSE Department  
IIT Bombay

## Lab Objective:

1. Familiarization with Tiva C series Launchpad
2. Install Code Composer Studio (CCS) software for programming
3. Familiarization with CCS project setup and debugging options
4. Mechanism to load program and execute it on Launchpad

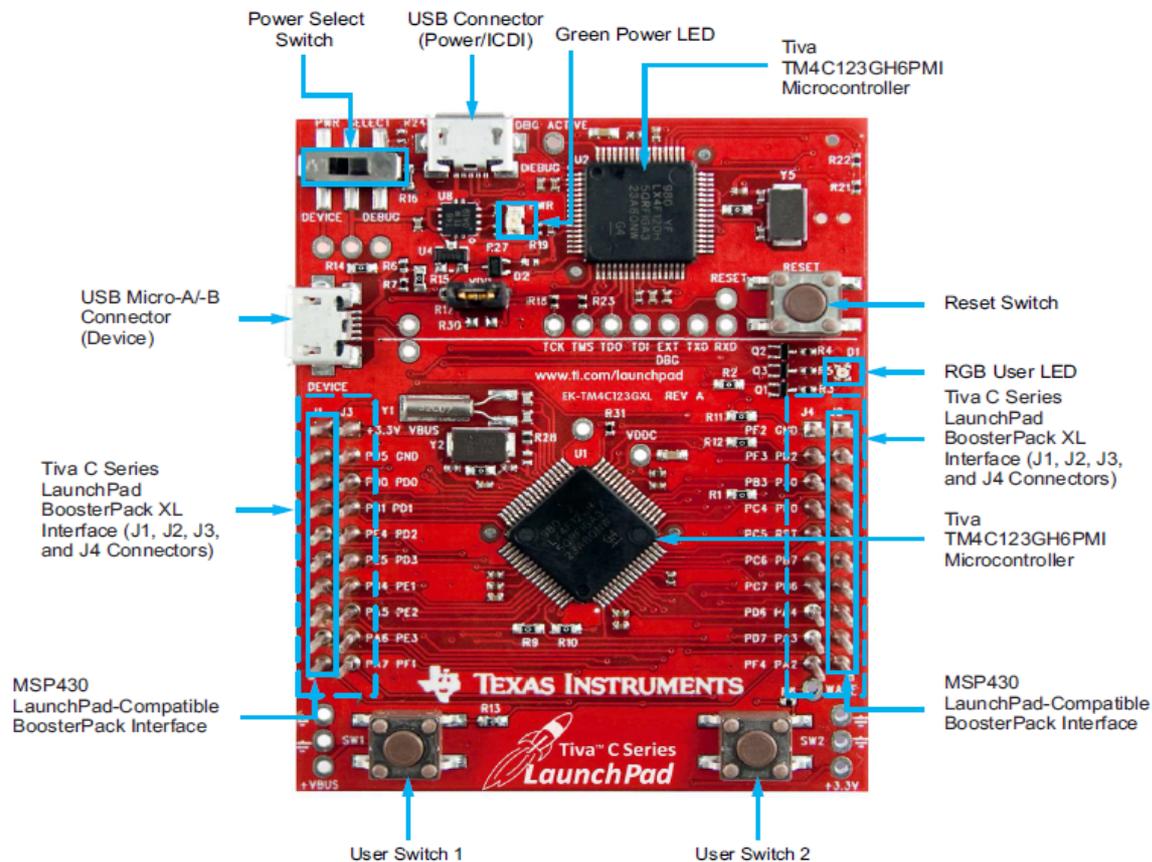
## 1. Overview of Launchpad:

### Section Objective:

- Acquaint with specifications of Launchpad
- Acquaint with location of switches, LEDs, power-on switch, powering the board.

In this course we are using Tiva C Series TM4C123G Launchpad Evaluation Kit for learning concepts of microcontroller programming and subsequently developing an embedded system applications.

Tiva C series Launchpad Evaluation Kit is developed and manufactured by Texas Instruments. Board and its various components are shown in figure-1 below.



**Tiva C Series TM4C123G LaunchPad Evaluation Board**

(Image Courtesy: TI User Guide for Tiva C series Launchpad)

This LaunchPad includes:

- Tiva C Series TM4C LaunchPad Evaluation Board (EK-TM4C123GXL)
- On-board In-Circuit Debug Interface (ICDI)
- USB Micro-B plug to USB-A plug cable
- README First document

Ensure the kit provided to you contains all the above mentioned items. In case you find any missing items please report to course TAs immediately.

The Tiva C Series TM4C123GH6PM is an ARM Cortex-M4 based microcontroller which offers:

- 80-MHz crystal oscillator
- 32-bit ARM Cortex-M4 CPU with floating point,
- 256 kBytes of 100,000 write-erase cycles of flash memory,
- 32KB RAM
- 2-KB EEPROM
- On-chip ROM with drivers and boot loaders
- 2x 12ch 12-bit ADCs (1 MSPS)
- 16x Motion PWM channels
- 24x Timer/Capture/Compare/PWMs
- 3x Analog comparators
- 4x SPI/SSI, 4x I2C, 8x UART
- USB Host/Device/OTG
- 2x CAN
- Low-power hibernation mode
- 43x GPIO pins

The Launchpad comprises of **two** general-purpose **user switches**, a **reset switch**, **power LED**, and user-programmable **RGB LED** are provided on board to ease development. A jumper is also included to allow current measurements, as well. Refer figure-1 to locate these.

Besides these board also provide Booster Pack compatibility through two dual-gender, 20-pin stackable headers which provide connection options out of the box. These 40 pins allow you to add support for displays, wireless interfaces, sensors and other capabilities to your projects.

Refer [User Guide](#) – Resource#2 on course web page to know more details about the board.

You may refer to [Datasheet to TM4C123G](#) – Resource#6 on course web page to understand

architecture details about the controller.

## 2. Software Installation and Hardware Setup:

---

Section Objective:

- Installation of CCS on Windows and Linux based machine
  - Configuring CCS for TM4C123G microcontroller
  - Understand debugging techniques
- 

For this course, we are using Code Composer Studio (CCS) IDE to write C code, generate the object file and load it on the microcontroller. We will download and install Code Composer Studio, as well as download the various other support documents and software tools that will be used throughout the course.

### .2.1 Code Composer Studio (CCS):

Code Composer Studio is an Integrated Development Environment (IDE) that supports TI's Microcontroller and Embedded Processors. Code Composer Studio comprises a suite of tools used to develop and debug embedded applications. It includes an optimizing C/C++ compiler, source code editor, project build environment, debugger, profiler, and various other features. The IDE combines the advantages of the Eclipse software framework with advanced embedded debug capabilities from TI resulting in a compelling feature-rich development environment for embedded developers.

CCS supports application development on various high-end microcontrollers and embedded processor such as MSP low power MCUs, C2000 Real-time MCUs, SimpleLink Wireless MCUs, Sitara (Cortex A & ARM9) Processors, C3x/C4x/TMS320x DSPs, etc.

For this lab course we will use CCS 6.1.2.

### 2.2 System Requirements:

A laptop running 32 or 64-bit version of Windows XP, and above or Linux (tested for Ubuntu 14.04) operating system. Laptop should have at least 3GB of free hard drive space and a minimum 1GB of RAM ... more is better.

Internet Connection is required for installation.

### 2.3 Download and Installation Instruction:

#### 2.3.1 Download and Installation Instruction - CCS

CCS 6.1.2 web Installer for Windows and Linux based systems can be [downloaded](#) from Resources#1.a. Web Installer will need internet connection to complete the installation process.

**For Windows based machine**, run the web Installer, (downloaded from above link), double click to

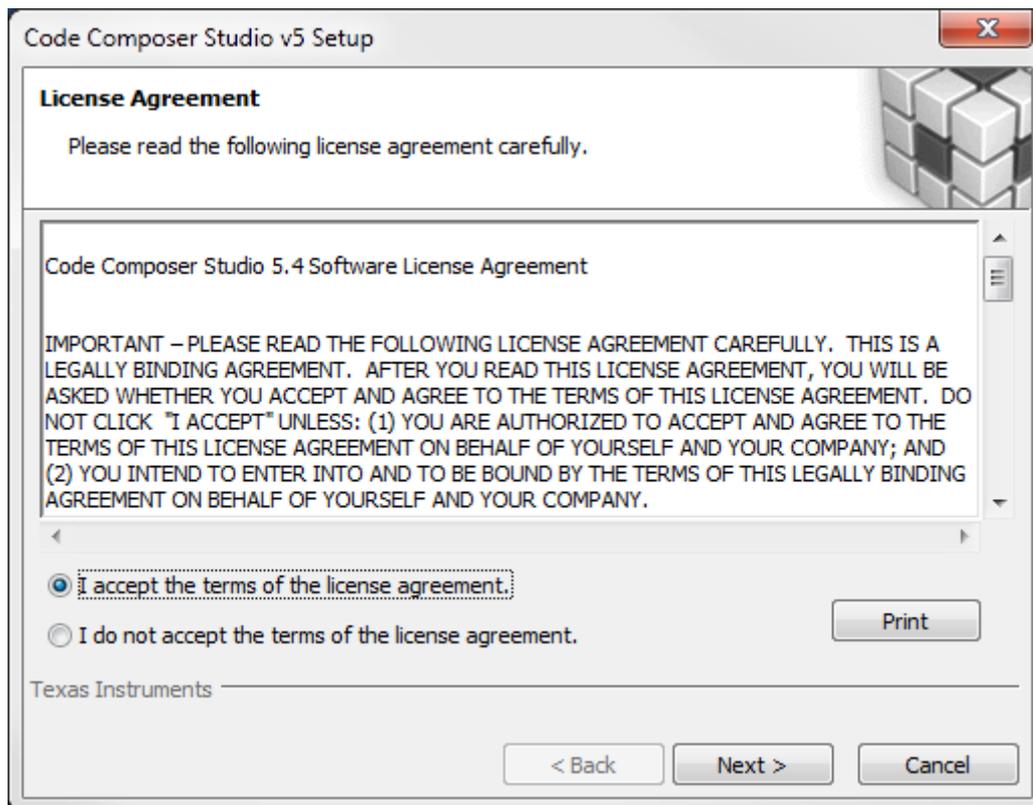
start the installation.

### For Linux based machine

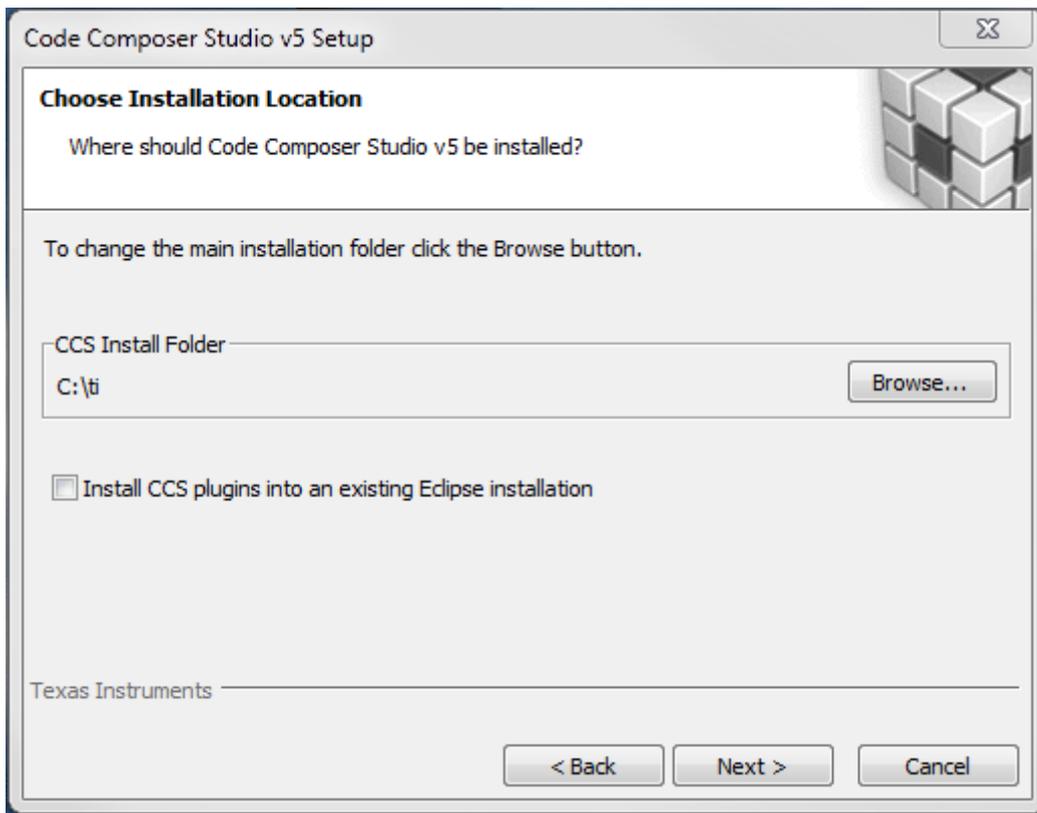
- Instructions for installation on Ubuntu 15.04 machine is [here](#)
- Instructions for installation on Ubuntu 14.04 machine is [here](#)
- Instructions for other Linux based machines is [here](#)

**Following steps are common for both Windows and Linux based machines(Appearance and display sequence may vary depending on OS and CCS version) :**

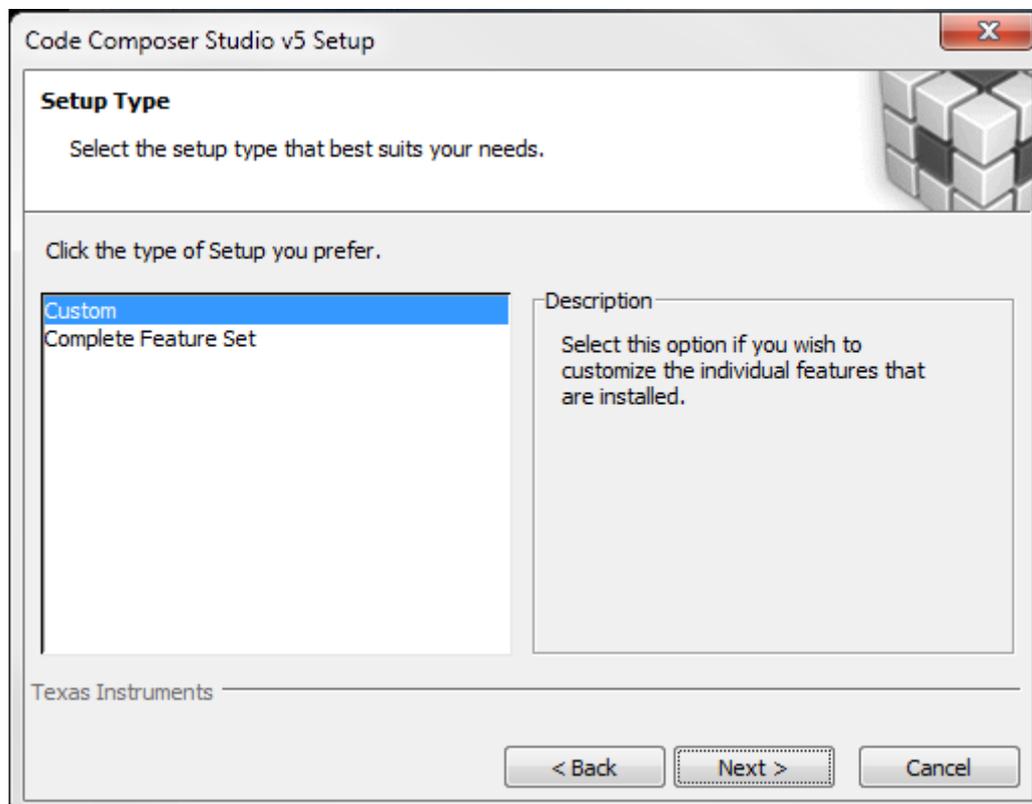
1. Accept the Software License Agreement and click *Next*.



2. For windows default install location is C:\ti and for Ubuntu location is /opt/ti. (**Note: C:\ti and /opt/ti** will henceforth refer to **default install location** in Windows and Linux based machine respectively) Unless you have a specific reason to install CCS in another location, accept the default installation folder and click next. If you have another version of CCS and you want to keep it, we recommend that you install this version into a different folder.

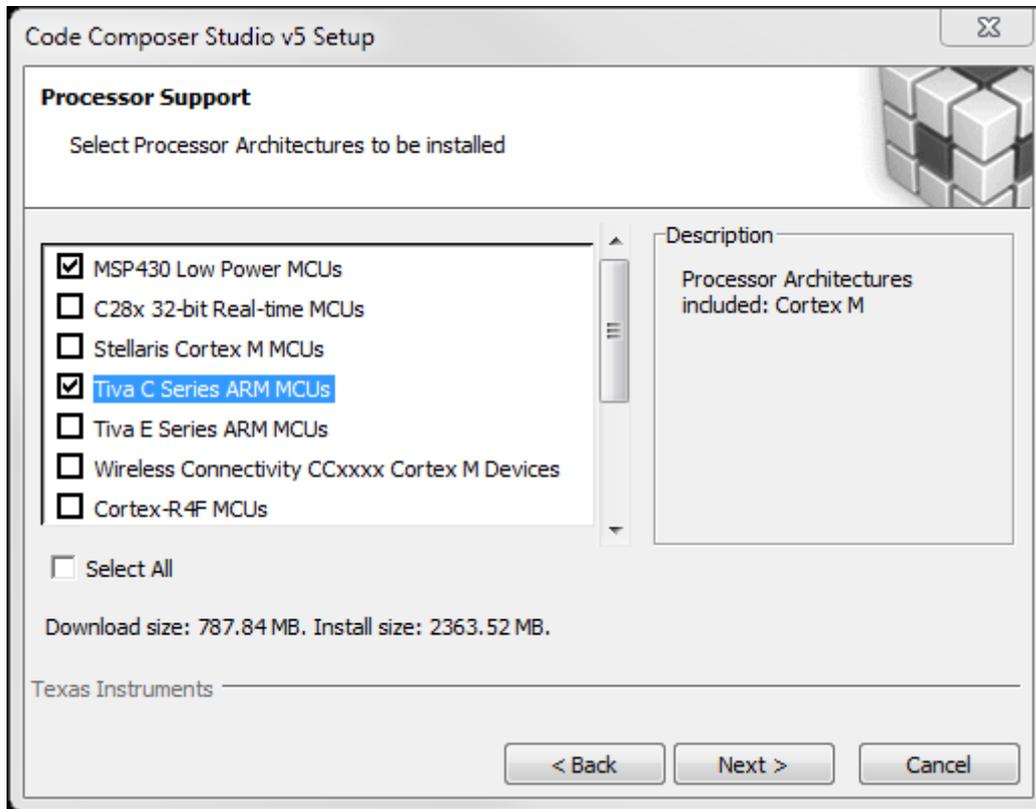


3. Select “Custom” for the Setup type and click Next.

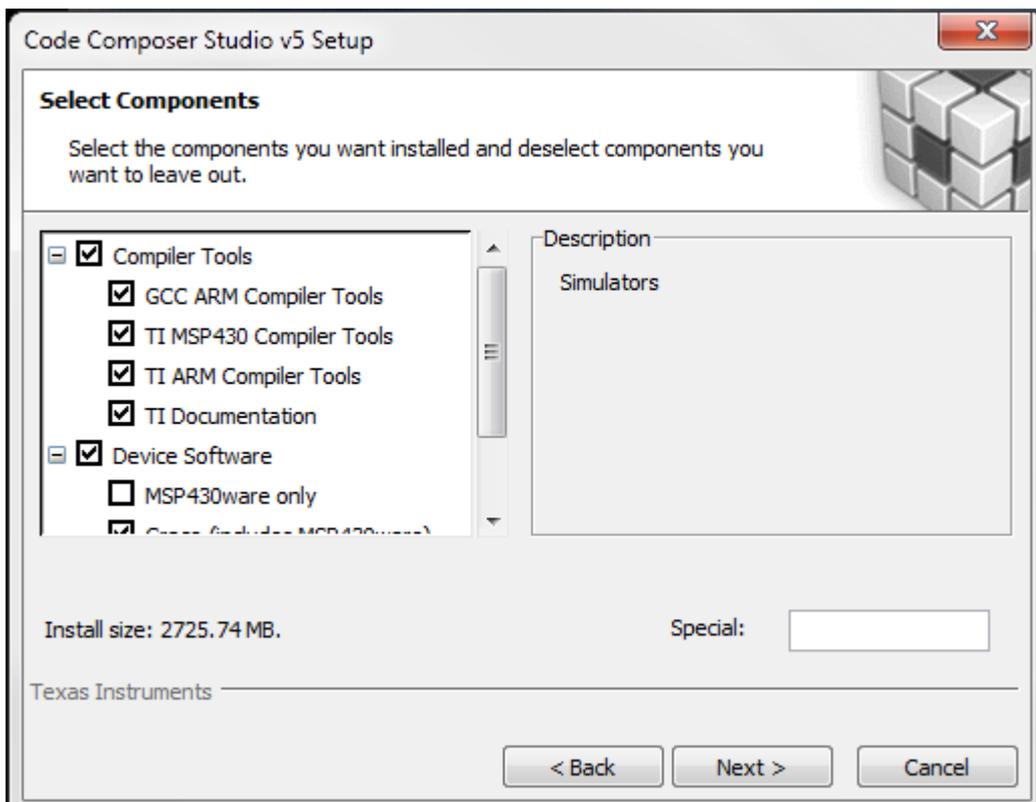


4. In the next dialog, select the processors that your CCS installation will support. You must select

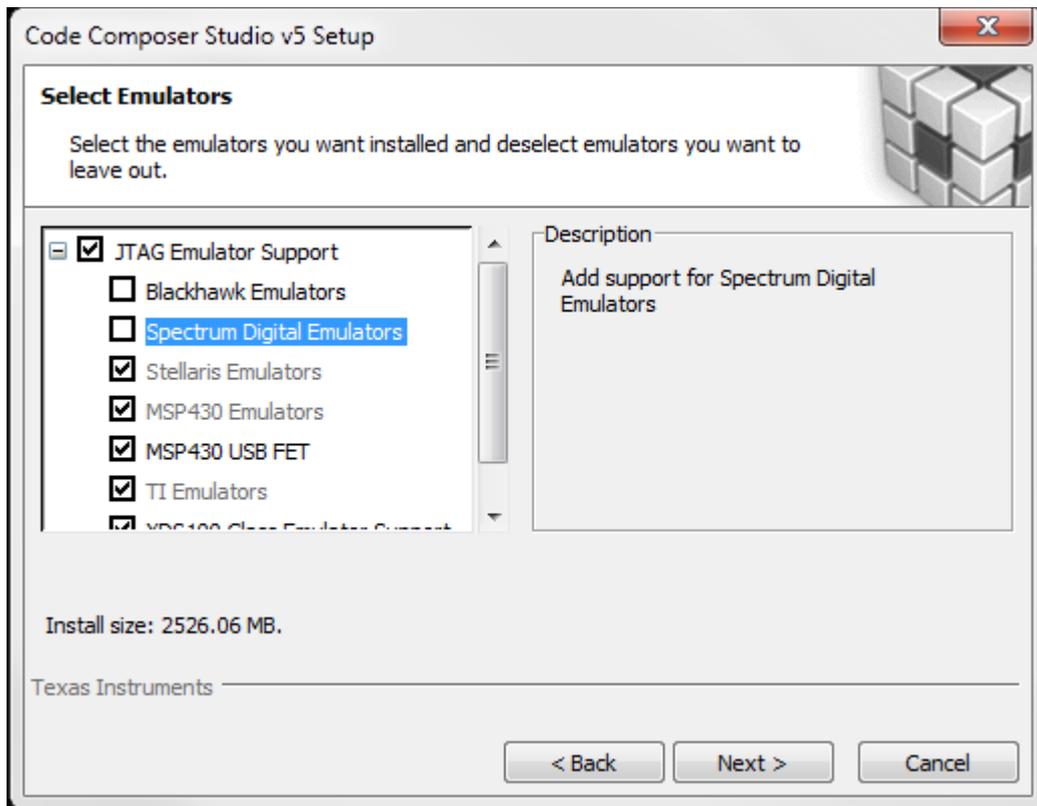
“32-bit ARM MCUs” in order to run the labs in this workshop. You can click select all, in case you are confused, the installation time and size will vary accordingly. Make sure you have selected TI and GCC compiler. Click Next.



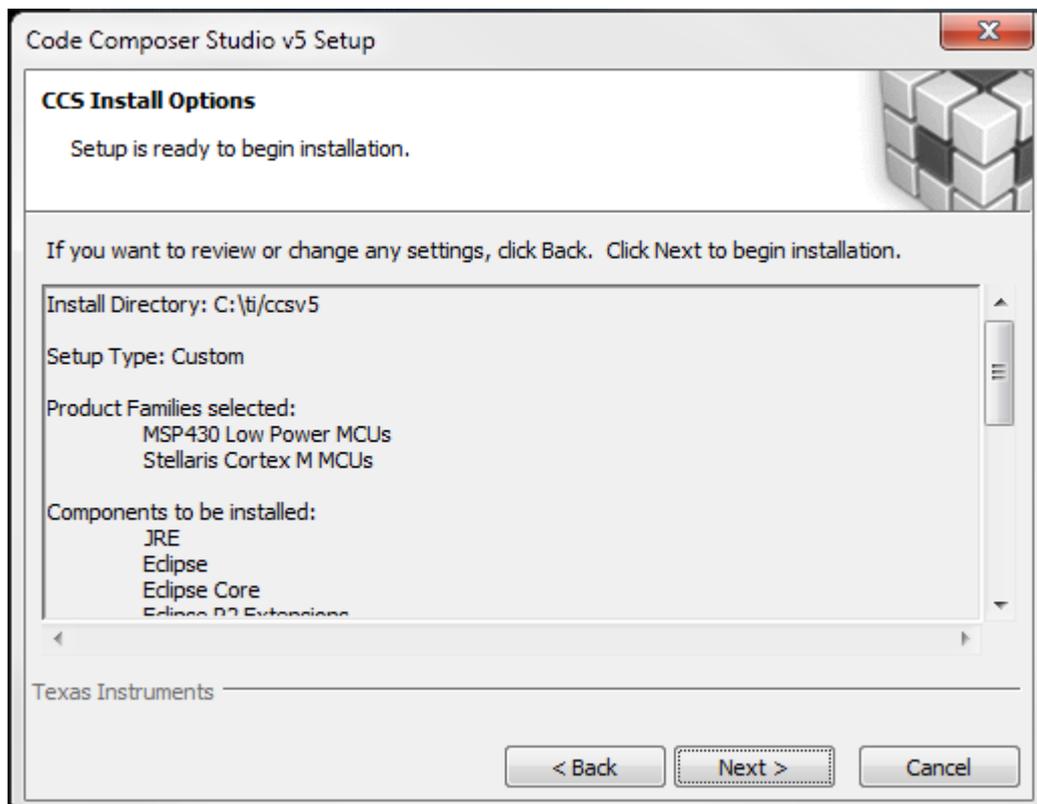
5. In the Component dialog, keep the default selections and click Next.



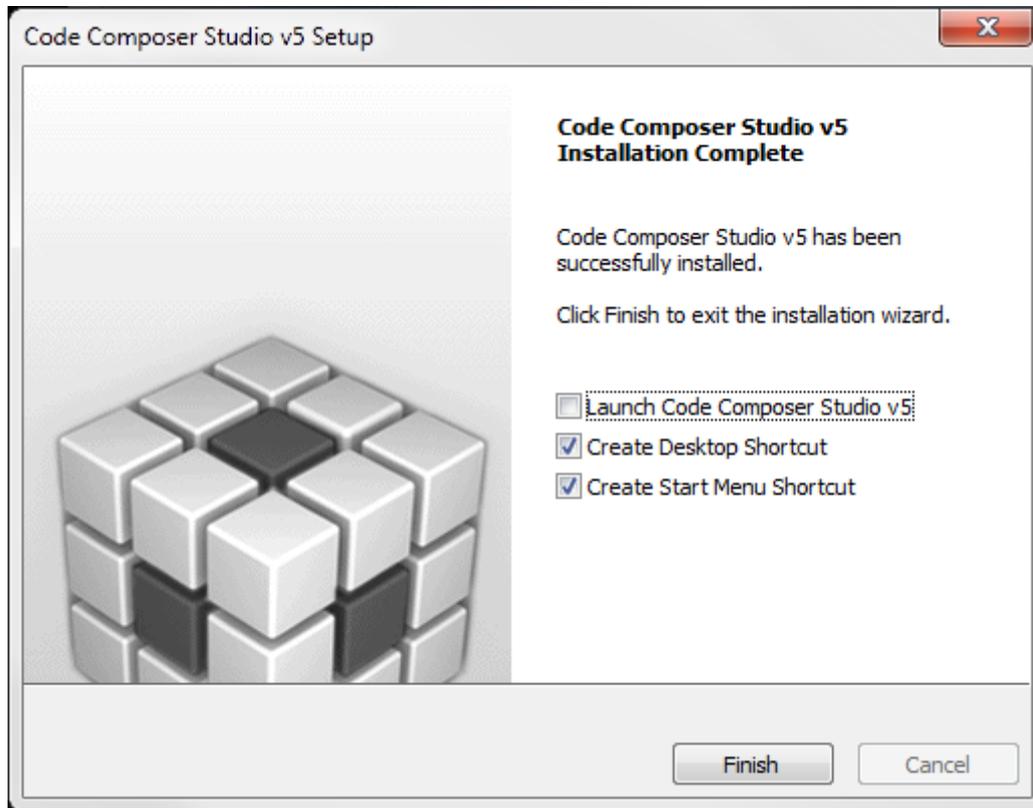
6. In the Emulators dialog, you can uncheck the Blackhawk and Spectrum Digital emulators. Click Next.



7. When you reach the final installation dialog, click Next.



8. There are several additional tools that require installation during the CCS install process. Click “Yes” or “OK” to proceed when these appear. Click Finish after CCS has been successfully installed.



### 2.3.2 Download and Installation Instruction - TivaWare:

TivaWare™ software for C Series is an extensive suite of software tools designed to simplify and speed development of Tiva C Series-based MCU applications. All TivaWare for C Series software has a free license, and allows royalty-free use so users can create and build full-function, easy-to-maintain code. TivaWare for C Series software is written entirely in C to make development and deployment efficient and easy.

#### **For windows:**

Download the latest version of [TivaWare](#) -- Resource#1.c from course webpage. Double click to install. TivaWare must be installed in CCS install directory i.e. C:\ti (if you have not changed default CCS install directory)

#### **For Linux:**

Download the latest version of [TivaWare](#) -- Resource#1.d from course webpage. Extract the zip folder into CCS install directory i.e. /opt/ti (if you have not changed default CCS install directory).

## 2.4 Configuring CCS

### 2.4.1 Creating New Project:

Following steps list process of creating a new project. These steps has to be followed every time new project is created.

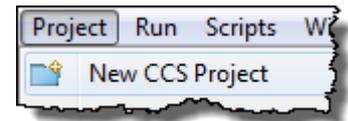
1. **Create Workspace:** Launch CCS. You will be asked to “Select a workspace”. Workspace is a folder in which all projects you create will reside. Browse to a desired location on your machine and create a folder “TIVA\_Workspace”. Click Ok. Note: Workspace has to be created only **once**. Same workspace will be used for all project.

2. **Select a CCS License:** To check CCS license

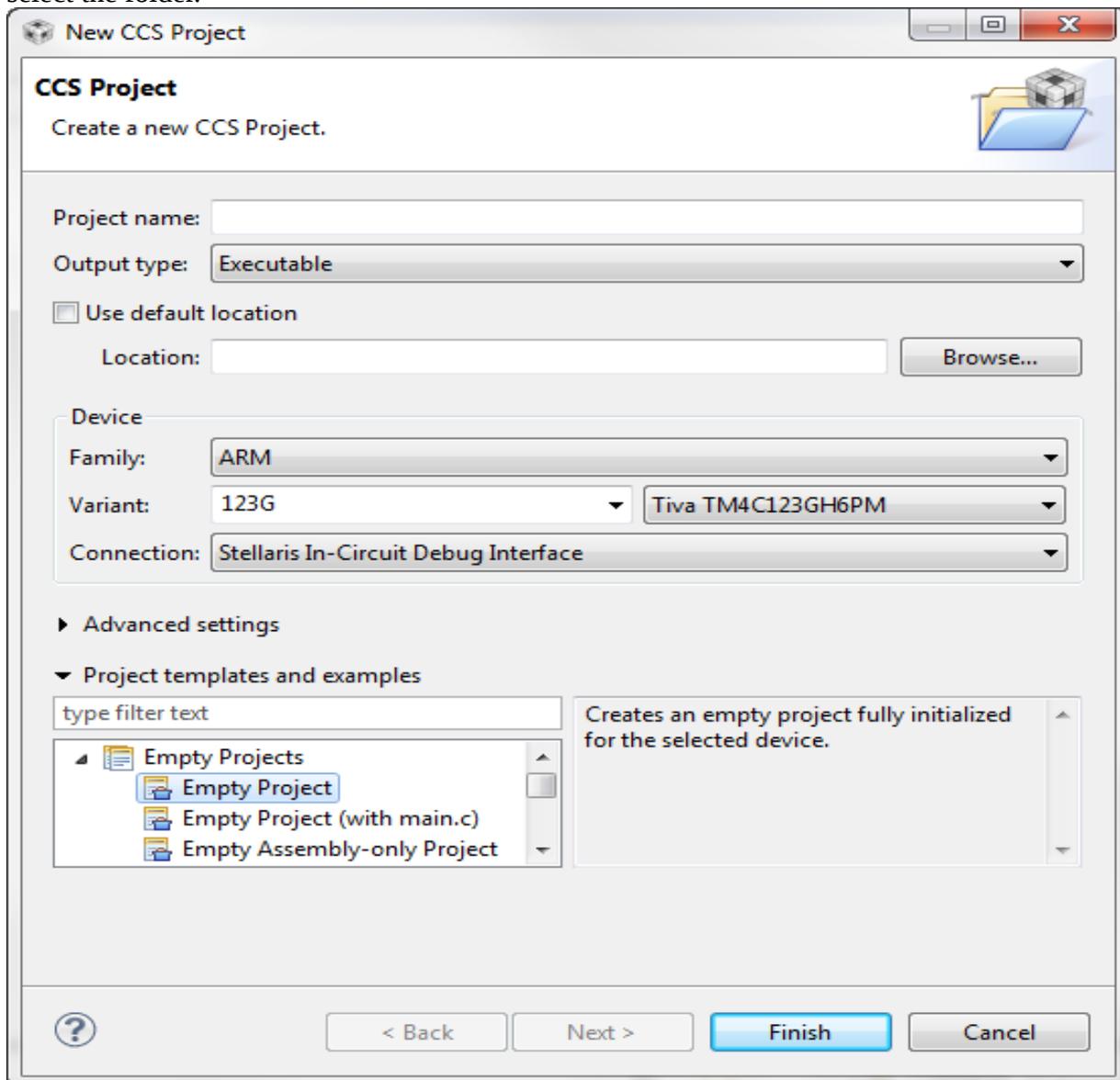
- Click on Help --> Code Composer Studio Licensing Information --> “Status” tab.
- If it shows “Not Licensed”, Select the “Upgrade” tab, and then select the “Free” license.

3. **Create a New Project:** To create a new project,

- Go to Project → New CCS Project
- Project name: Give a proper name (say ‘projectname’)
- Output type: Executable
- Location: Browse to Workspace, create a new folder with name same as “Project name”



select the folder.



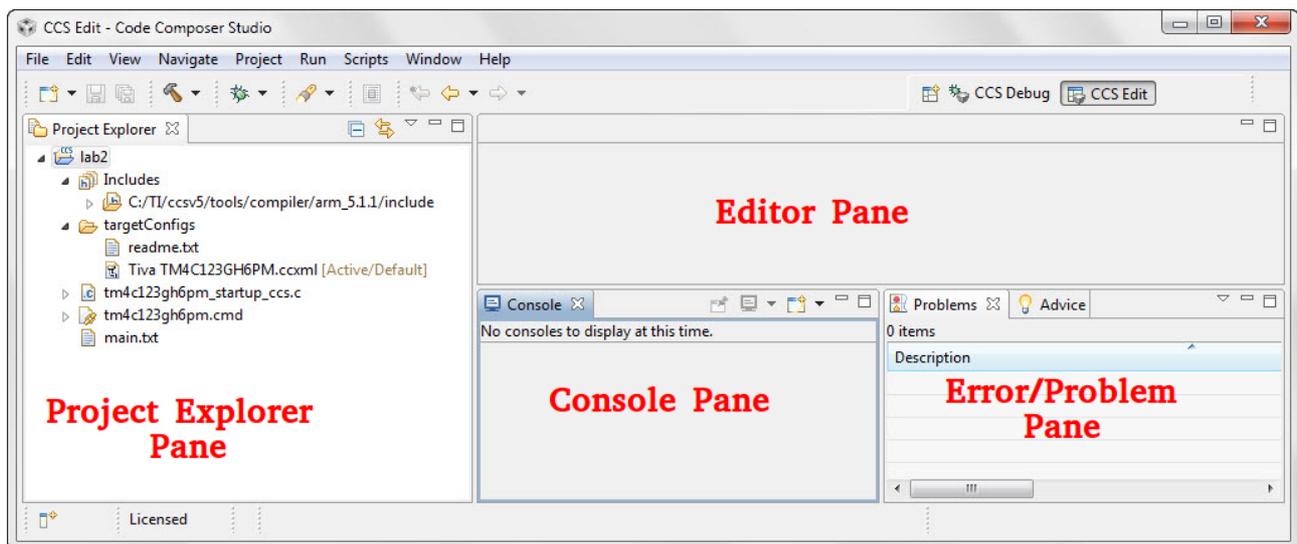
- Select Device family: ARM,
- Variant: Type 123G in the filter text field, then select Tiva **TM4C123GH6PM** in the drop-

down box (typing 123G narrows the list making it easier to find the exact part on the Tiva Launchpad board.)

- For Connection: choose Stellaris In-Circuit Debug Interface. This is the built-in emulator on the Launchpad board.
- Project templates and examples: choose Empty Project and then click Finish.

**Note:** Project window can look bit different based on OS and CCS version. You may find “Output type” specified in “Advanced settings”. Device family and variant are grouped together and is called “**Target**”, filter for Tiva C series and select **TM4C123GH6PM** as controller.

4. Review the CCS Editing GUI – Figure below shows various panes of CCS which appear after project is created. Understanding their nomenclature and functionality with help in process of application development.



### 2.4.2 Add Path and Build Variables:

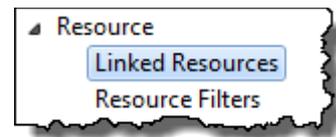
- Path variable: When you ADD (link) a file to your project, you can specify a “relative to” path. The default is PROJECT\_LOC which means that your linked resource (like a .lib file) will be linked relative to your project directory.
- Build variable: – Used for items such as the search path for include files associated with a library – i.e. it is used when you build your project.

Variables can either have a PROJECT scope (that they only work for this project) or a WORKSPACE scope (that they work across all projects in the workspace).

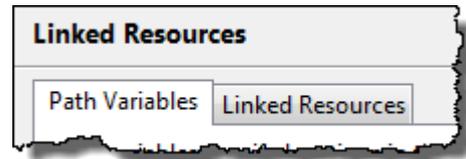
In the next step, we need to add (link) a library file and then add a search path for include files. First, we’ll add these variables MANUALLY as PROJECT variables. Later, we will show you a quick and easy way to add these variables into your WORKSPACE so that any project in your workspace can use the variables.

#### 2.4.2.1 Adding a Path Variable

To add a path variable, Right-click on your *projectname* and select *Properties*. Expand the **Resource** list in the upper left-hand corner as shown and click on **Linked Resources**:

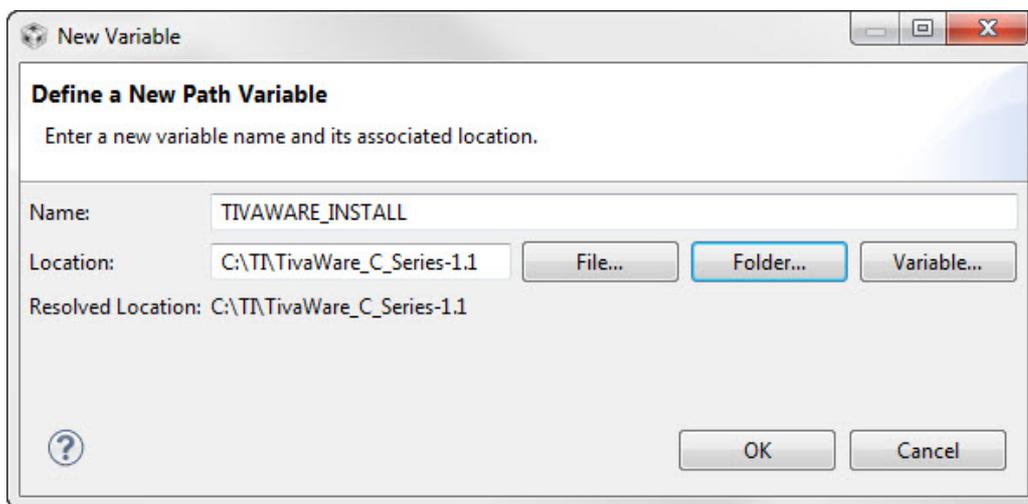


You will see two tabs on the right side – *Path Variables* and *Linked Resources*:



In the **Path Variables** tab, notice that PROJECT\_LOC is listed and will display as the default path variable for linked resources in your project. We want to add a New variable to specify exactly where you installed TivaWare.

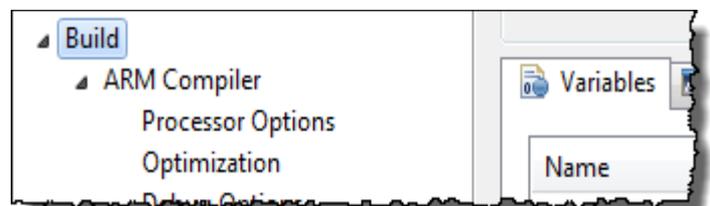
- Click *New...*,



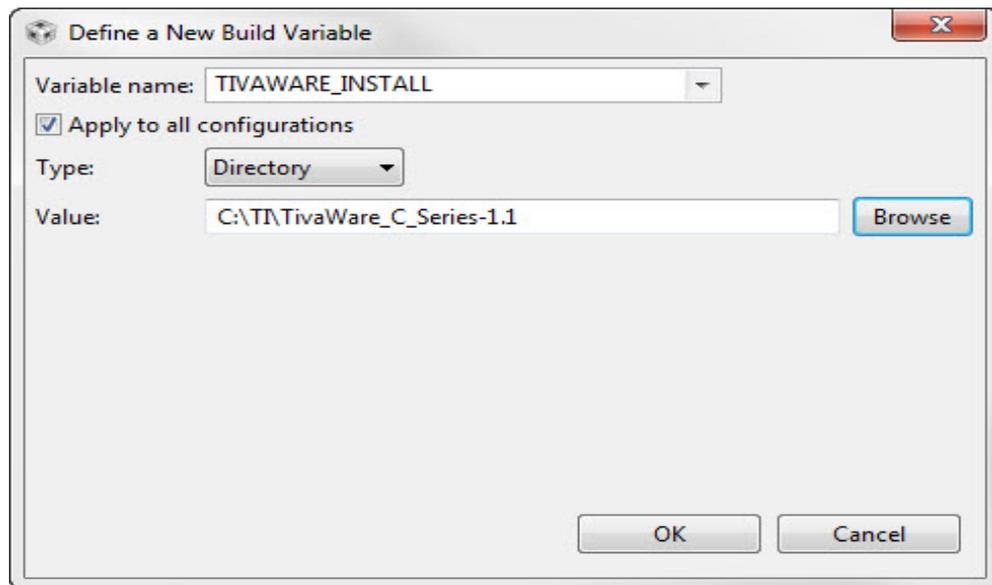
- When the New Variable dialog appears, type TIVAWARE\_INSTALL for the name, for the Location, click the Folder... button and navigate to your TivaWare\_C\_Series-2.1.2.111 installation. Click on the folder name and then click OK.
- You should see your new path variable listed in the Path Variables list with name as TIVAWARE\_INSTALL and value as *ti\_installdirectory/TivaWare\_C\_Series-2.1.2.111*. Click OK.

#### 2.4.2.2 Adding a Build Variable:

Now let's add a build variable that we will use in the include search path for the INCLUDE files associated with the TivaWare driver libraries.



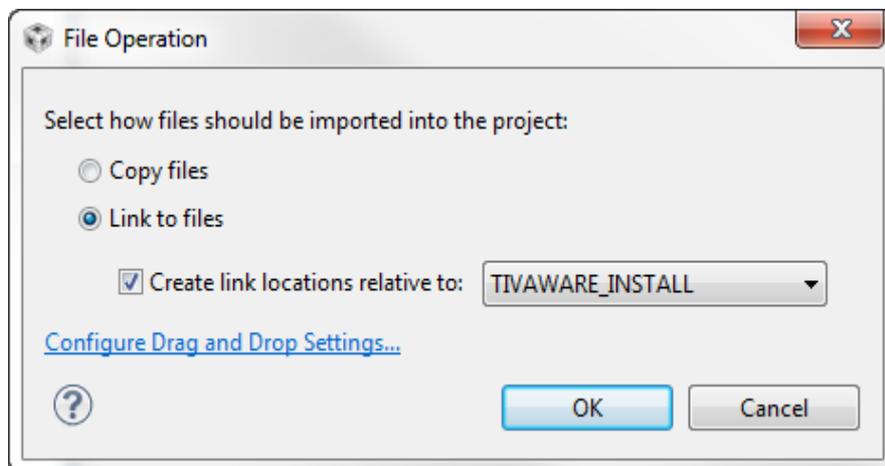
- Right-click on your *projectname* and select *Properties*.
- Click on *Build* and then the **Variables** tab:, Click the *Add...* button. When the Define a New Build Variable dialog appears, insert *TIVAWARE\_INSTALL* into the Variables name box.
- Check the “Apply to all configurations” checkbox
- Change the *Type* to **Directory** and browse to your Tivaware installation folder.
- Click OK.



- You should see a new variable with name as *TIVAWARE\_INSTALL*, type as *Directory* and value as *ti\_installdirectory/TivaWare\_C\_Series-2.1.2.111*
- Click OK again to save and close the Build Properties window.

### 2.4.3 Link the TivaWare driverlib.lib file to your project:

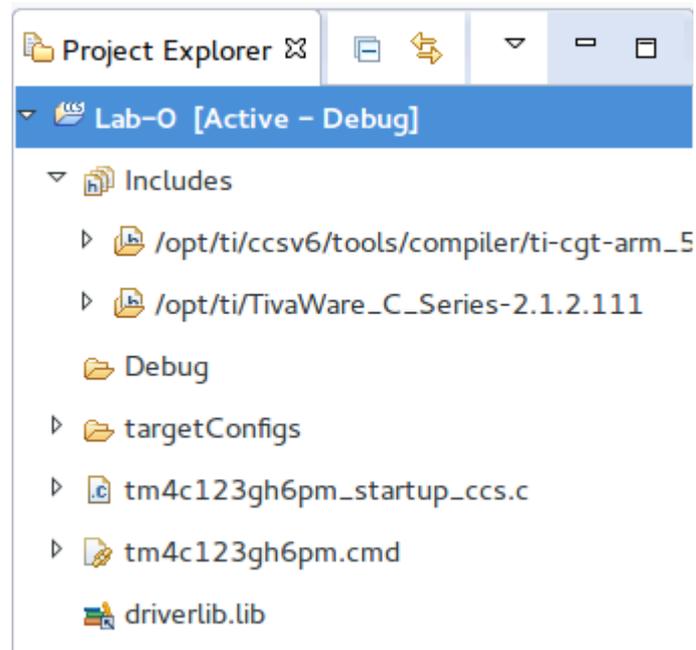
- Right Click on *projectname* -Add Files... Navigate to: *ti\_installdirectory/TivaWare\_C\_Series-2.1.2.111/driverlib/ccs/Debug/driverlib.lib*
- click Open.



The File Operation dialog will open ... Use the TIVAWARE\_INSTALL path variable you created earlier. This means that the LINK (or reference to the library) file will be RELATIVE to the location of the TivaWare installation. If you hand this project to someone else, they can install the project anywhere in the file system and this link will still work. If you choose PROJECT\_LOC, you would get a path that is relative to the location of your project and it would require the project to be installed at the same “level” in the directory structure. Another advantage of this approach is that if you wanted to link to a new version, say TivaWare\_C\_Series-1.2, all you have to do is modify the variable to the new folder name.

- Make the selections shown and click OK.

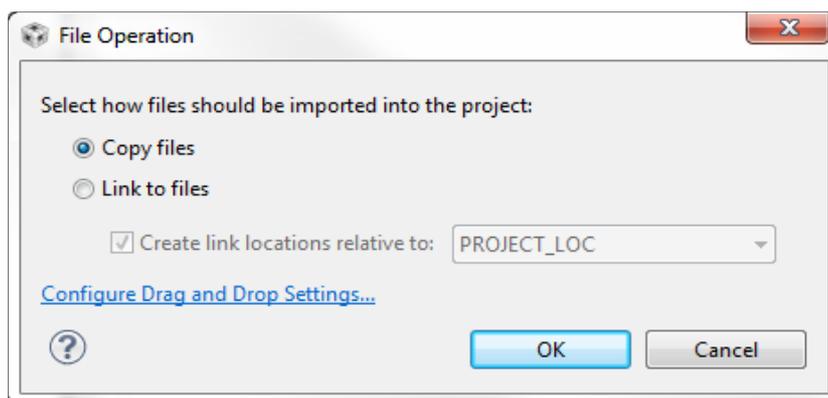
Your project should now look something like the screen capture below. Note the symbol for driverlib.lib denotes a linked file.



#### 2.4.4 Add files to your project

Now we will add "lab-0.c" to the project. This file is the C code that we have written. lab-0.c file is provided to you in Lab-0 project folder. This file is provided as test file to software and hardware. This C file should be copied to the project.

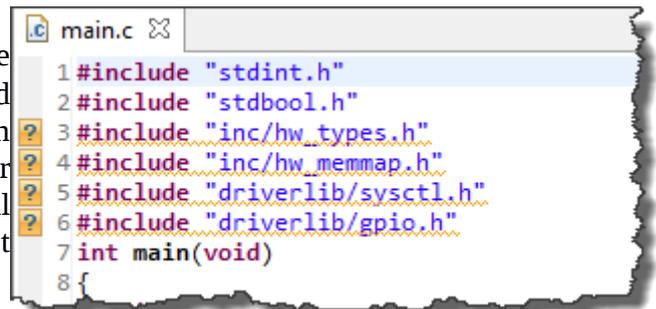
- Select Project → Add Files... Navigate to the folder Lab-0 provided on course webpage.
- Select Lab-0.c and click Open.



- Then select Copy files and click OK.

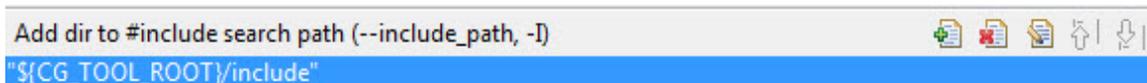
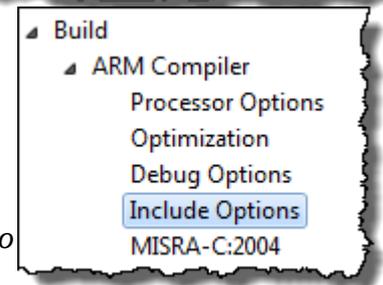
## 2.4.5 Add the INCLUDE search paths for the header files:

- Open lab-0.c by double-clicking on the Filename in the Project Explorer pane of CCS. You should see “?” warnings in the left margin which indicate “unresolved inclusion”. Hover your cursor over the question mark to see the helpful message. Until now, you haven’t told the project where to find these header files.



```
1 #include "stdint.h"
2 #include "stdbool.h"
3 #include "inc/hw_types.h"
4 #include "inc/hw_memmap.h"
5 #include "driverlib/sysctl.h"
6 #include "driverlib/gpio.h"
7 int main(void)
8 {
```

- Right-click on ‘projectname’ in the Project Explorer pane and select Properties.
- Click on Build → ARM Compiler → Include Options (as shown):
- In the lower-right panel, click the “+” sign next to *Add dir to #include search path*



```
Add dir to #include search path (--include_path, -I)
"${CG_TOOL_ROOT}/include"
```

Add the following path using the build variable you created earlier. Place the variable name inside braces, after the \$ as shown:

```
${TIVAWARE_INSTALL}
```

- Click OK.
- Click OK again, and now you should see those “?” in lab-0.c disappear after a moment.

We are almost done and next step is to build the program and then load it on the Launchpad and check the output. But before doing so, we do a final check to ensure all settings have been done properly.

## 2.4.6 Examine your Project files using Windows Explorer

- Using Windows Explorer, locate your projectname folder:
  - Do you see *lab-0.c*? It should be there because you copied it there.
  - Do you see the *driverlib.lib* file? This file should be there because it’s linked in your project. Notice the other folders in the \project folder – these contain your CCS project-specific settings. Close Windows Explorer.
  - Two files named “tm4c123gh6pm\_startup\_ccs.c” and “tm4c123gh6pm.cmd” must be present. These files are created during project creation.
  - Includes shows *ti\_installdirectory/TivaWare\_C\_Series-2.1.2.111*
  - Folders named “Debug” and “targetConfigs” are present

## 2.5 Build your Project

Build your project and fix any errors.

- You can build the project by clicking on the HAMMER (Build) button. Building the code reports issues with the code in *Error/problem pane*. If code has no error, an output



file(.out) file will be created which can be loaded on the flash memory of microcontroller.

**Build the project** with all settings you have done. Ideally it should build with no error and warning if all settings have been done as instructed. Your Error/problem pane should show 0 items and console pane should show Finished building target: **projectname.out**. “Debug” folder should show a new file named **projectname.out**. If any pane is not visible by default

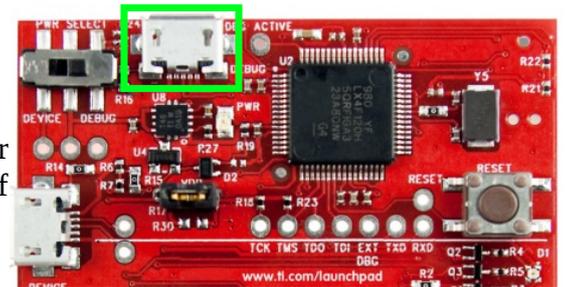
In case you find an error, find source of error by reading the console pane and Error/problem pane. Revisit steps involved in creating new project, adding path and build variables and check you have followed each instruction carefully.

## 2.6 Load and Run Project on Launchpad

### 2.6.1 Setting up the board:

Ok. Let us connect our board now

- Connect Laptop and board using USB cable. Connect USB micro-b end to USB Connector (Power/ICDI) on board, and USB-A end to USB of laptop.
- Switch “Power Select Switch” in debug mode. This should turn-on “Green Power LED”.

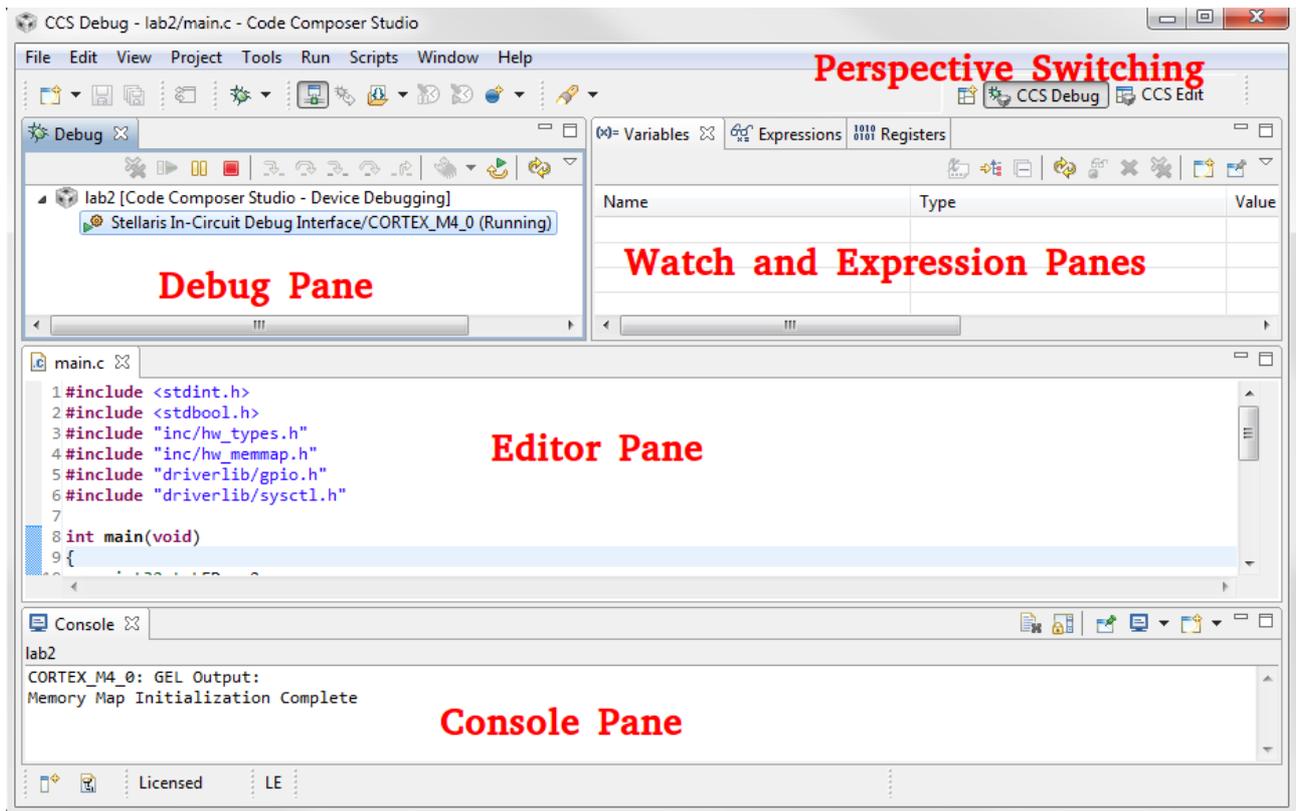


In windows based machine you may get problem while connecting device to machine. One probable cause would be missing device driver. Open device manager and check for Stellaris Virtual Serial Port. If device driver is missing, you can download it from [http://www.ti.com/tool/stellaris\\_icdi\\_drivers](http://www.ti.com/tool/stellaris_icdi_drivers). Update the driver.

### 2.6.2 Load your project:

Assure that your Launchpad is connected to your laptop. Build  and load your project to the TM4C123GH6PM flash memory by clicking the Debug button.

If connection to board is proper, window similar to one shown below will appear. This is called CCS Debug Perspective.



Note the names of the Code Composer panes above. There are two pre-defined perspectives in Code Composer; CCS Edit and CCS Debug. You can switch between perspectives by clicking the on respective tabs. Perspectives are only a “view” of the available data, you can edit your code here without changing perspectives. And you can modify these or create as many additional perspectives as you like.

### 2.6.3 Run your project:

- Click the **Resume** button  or press the F8 key on your keyboard to run the program.

The tri-color LED on your target board should blink showing the three colours in sequence.

- To stop your program running, click the **Suspend** button 

If the code stops with a “No source available ...” indication, click on the lab-0.c tab. Most of the time in the while () loop is spent inside the delay function. That source file is not linked into this project.

## 2.7 Using Various Debug Feature of CCS

### 2.7.1 Set a Breakpoint

In the code window in the middle of your screen, double-click in the blue area to the left of the line number of the GPIOPinWrite() instruction. This will 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 or 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.

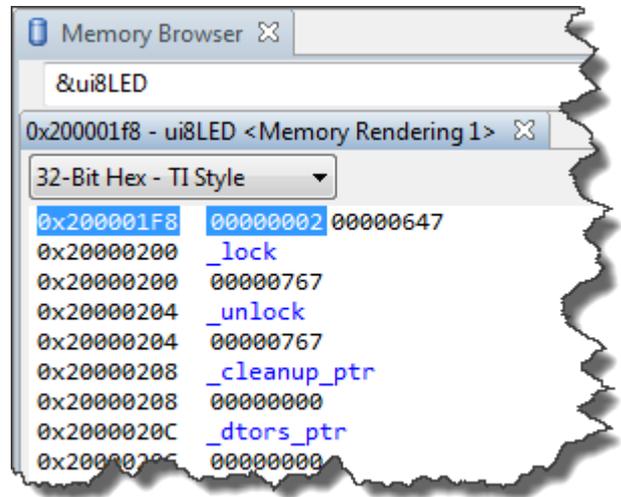
### 2.7.2 View/Watch memory and variables

- Click on the Expressions tab in the Watch and Expressions pane.
- Double-click on the ui8LED variable anywhere in main().
- Right-click on ui8LED and select Add Watch Expression...:
- Click OK. Right-click on ui8LED in the Expressions pane, and select Number Format as Hex or decimal. Note the value of ui8LED.

Of course, the ui8LED variable is located in SRAM. You can see the address in the expressions view. But let's go see it in memory.

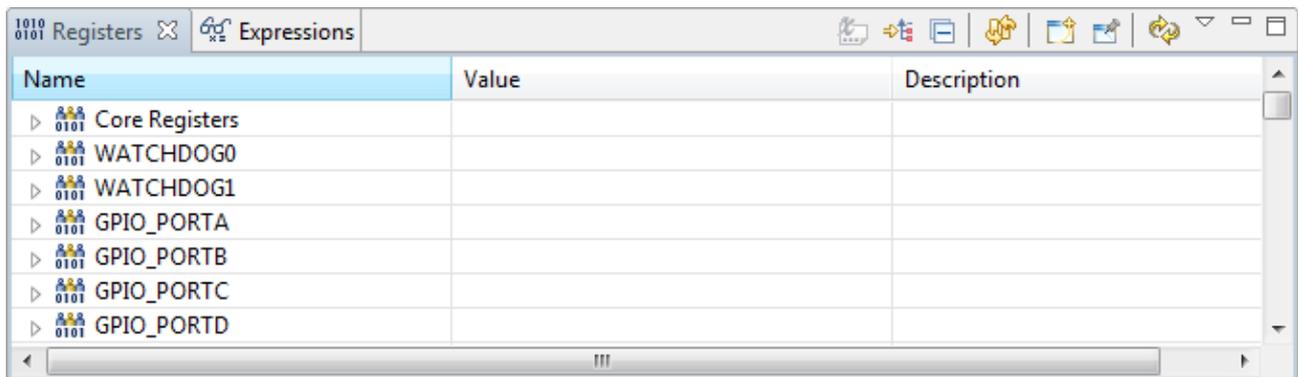


- Select View -- Memory Browser:
- Type &ui8LED into the memory window to display ui8LED in memory:



### 2.7.3 View Registers

- Select View- Registers and notice that you can see the contents of all of the registers in your target's architecture. This is very handy for debugging purposes.



Click on the arrow on the left to expand the register view. Note that non-system peripherals that have not been enabled cannot be read. In this project you can view Core Registers, GPIO\_PORTF (where the LEDs and pushbuttons are located). Check how values of GPIO\_DATA and GPIO\_DIR changes with every step.

### 2.7.4 Step Debugging:



Step Into (F5) -- Execute each statement of code one-by-one. If the statement is a call to a function, the execution jumps to the first statement in the called function and executes each statement in that function one at a time.



Step Over (F6) -- Similar to Step Into with only difference being that if the statement is a call to a function, execution does not enter the called function, instead executes the called function as a single unit and execution jumps to next statement in calling function.

You can use combination of Breakpoints, Step Into and Step Over to debug the program in very efficient way.

### 2.7.5 Terminate:



Click the red Terminate button to terminate the debug session and return to the CCS Edit perspective.

Congratulations - You are ready to dive into the world of embedded systems. See you in Lab!!