CODE COMPOSER STUDIO (CCS) TUTORIAL

By Phil Bones for ENCE361

v1.1 26.2.2018

For CCS Version 7.4.0

Please note, these tutorial notes may change without notice. We suggest you check for any updates on Learn

Starting your first ENCE361 lab with CCS

During your lab session in Week-2, you will:

i. Configure Code Composer Studio (CCS) project using a simple GPIO example.

ii. Learn how to compile, link, and debug a CCS project using the TM4C123GH6PM microcontroller.

iii. Configure and use the application programming interface (API), TivaWare.

iv. Understand how general purpose input and output (GPIO) work on TM4C series microcontroller by examining and modifying C source code.

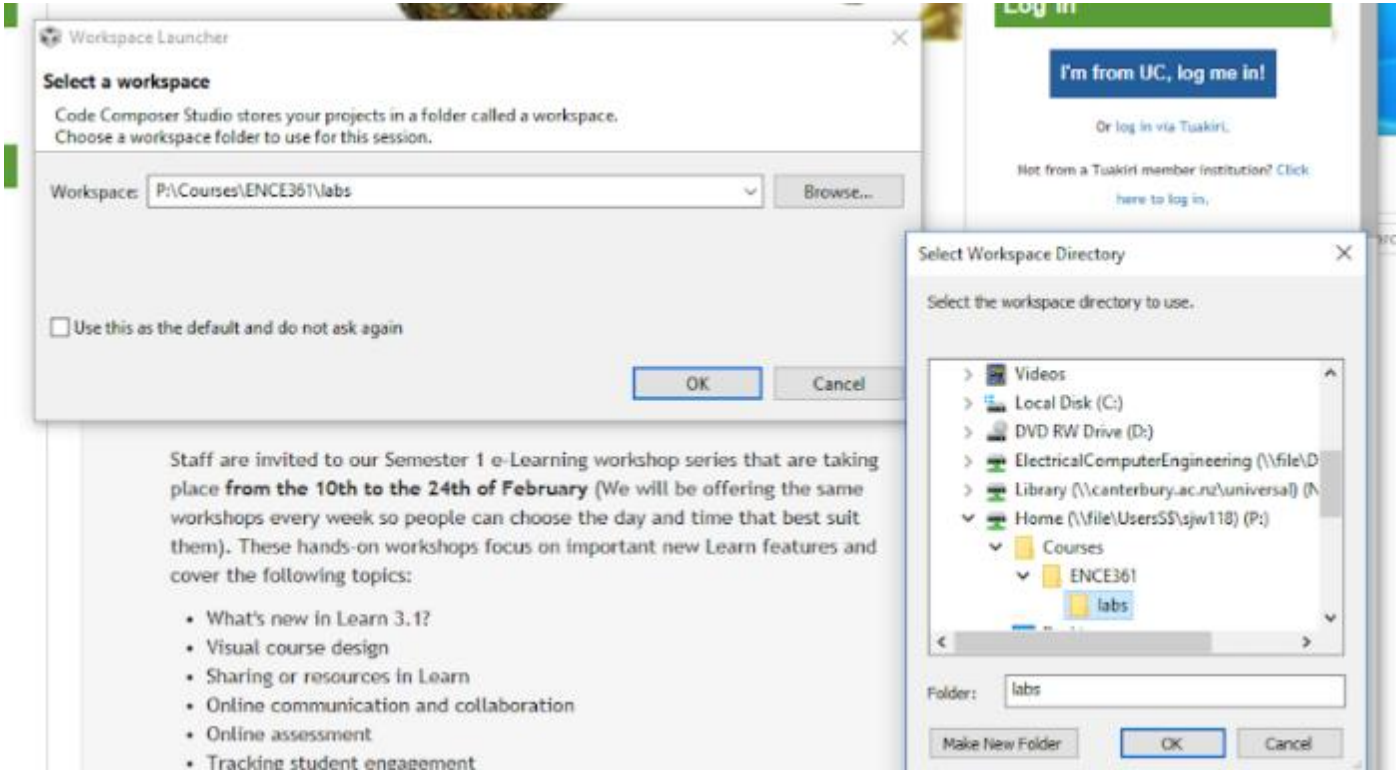
This guide is designed to get you started. *You may need to read it again for later labs.*

Note: *on all Level 2 lab PCs (in Electronics, Embedded Systems and Computer Labs), Code Composer Studio (CCS), Ver. 7.4.0, has been installed. The application programming interface (API) used for the TM4C series microcontroller is called TivaWare. It is installed in the C:\ti\TivaWare_C_Series-2.1.4.178 directory. During labs you simply use CCS and link with the TivaWare API on the local PC with your source code and projects located on your P: drive.*

If you are setting up CCS on your laptop you will need to install both CCS (first) and TivaWare (second). This tutorial does not cover private installations of CCS and/or TivaWare. Neither does it cover using CCS in conjunction with git.

Logging in and setting up a CCS workspace on your P: drive

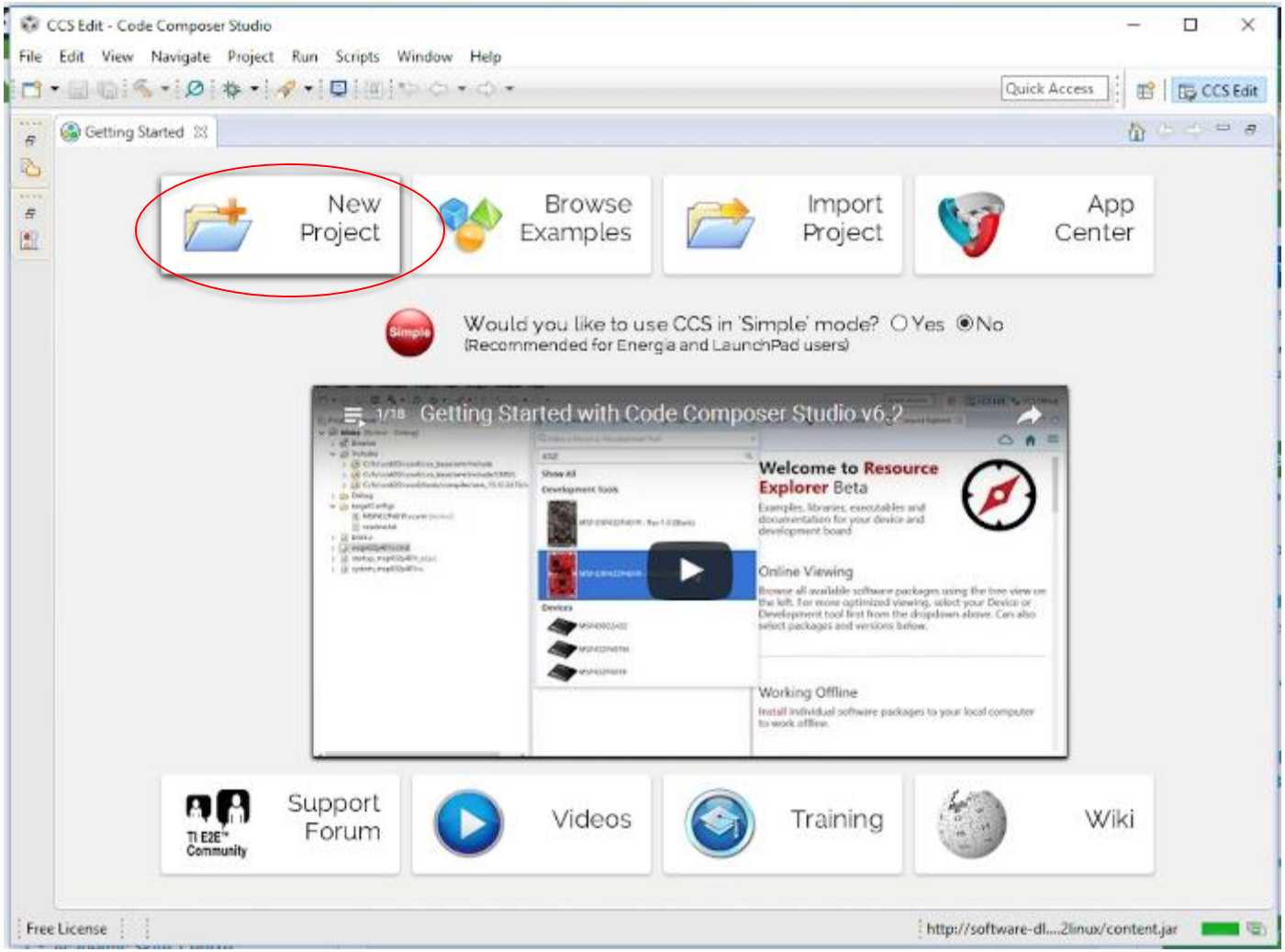
- Login to a PC in the Electronics lab with your UOCNT account; there will be 3 students per bench, but only one student can log in.
- Make sure your P: drive is mapped. Check this by opening a file explorer window, clicking on “This PC” and checking that your usercode is mapped to the “P” drive.



- Next, for your workspace create the directory structure, `P:\Courses\ENCE361\labs`. **Note: In this course, do not use spaces when defining either directory or file names. You will have no end of problems if you use spaces!** Lastly, click OK…

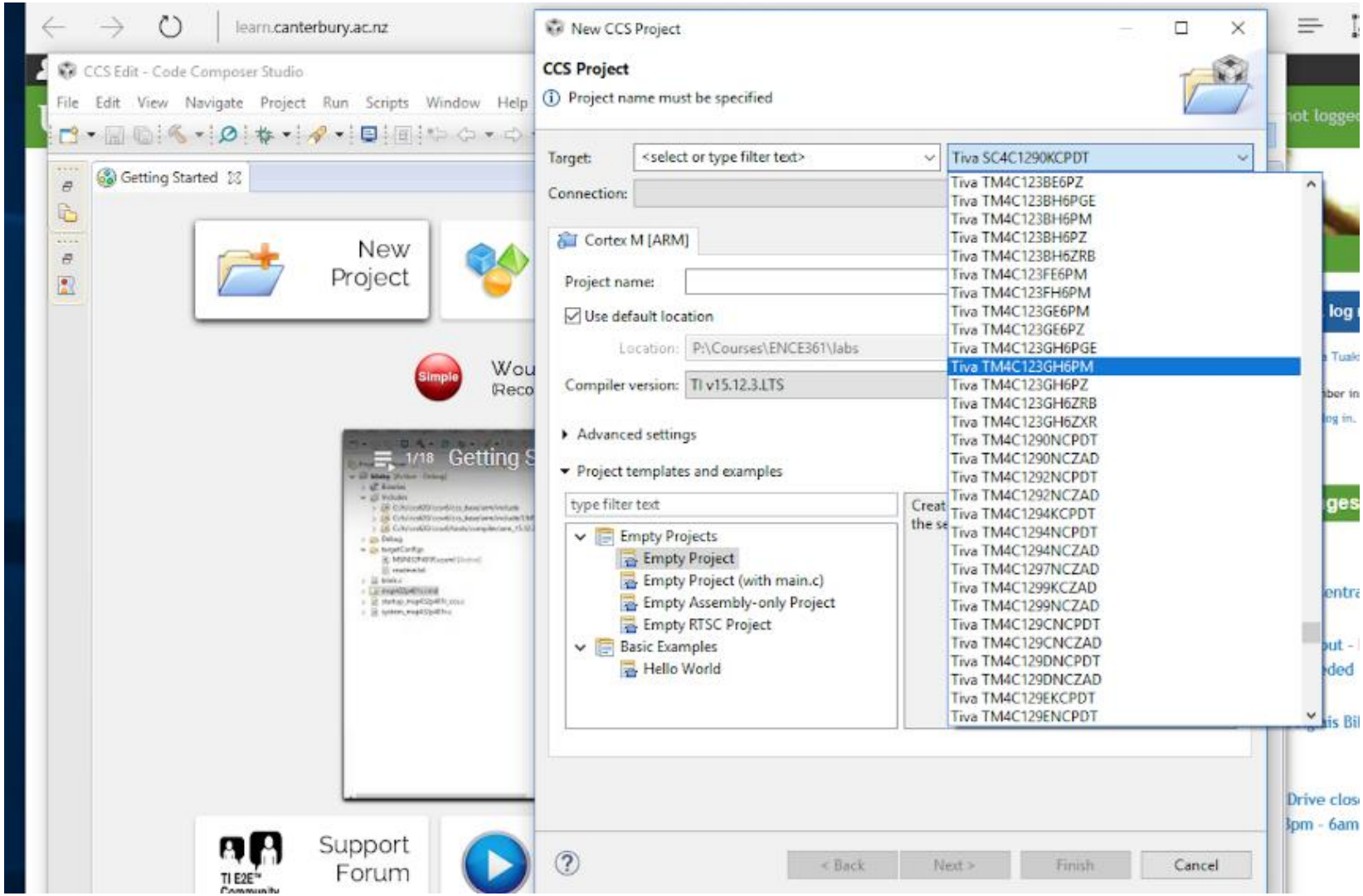
Setting up CCS in the lab, cont.

- You should now see the CCS “Getting Started Window”. Click on “New Project”.



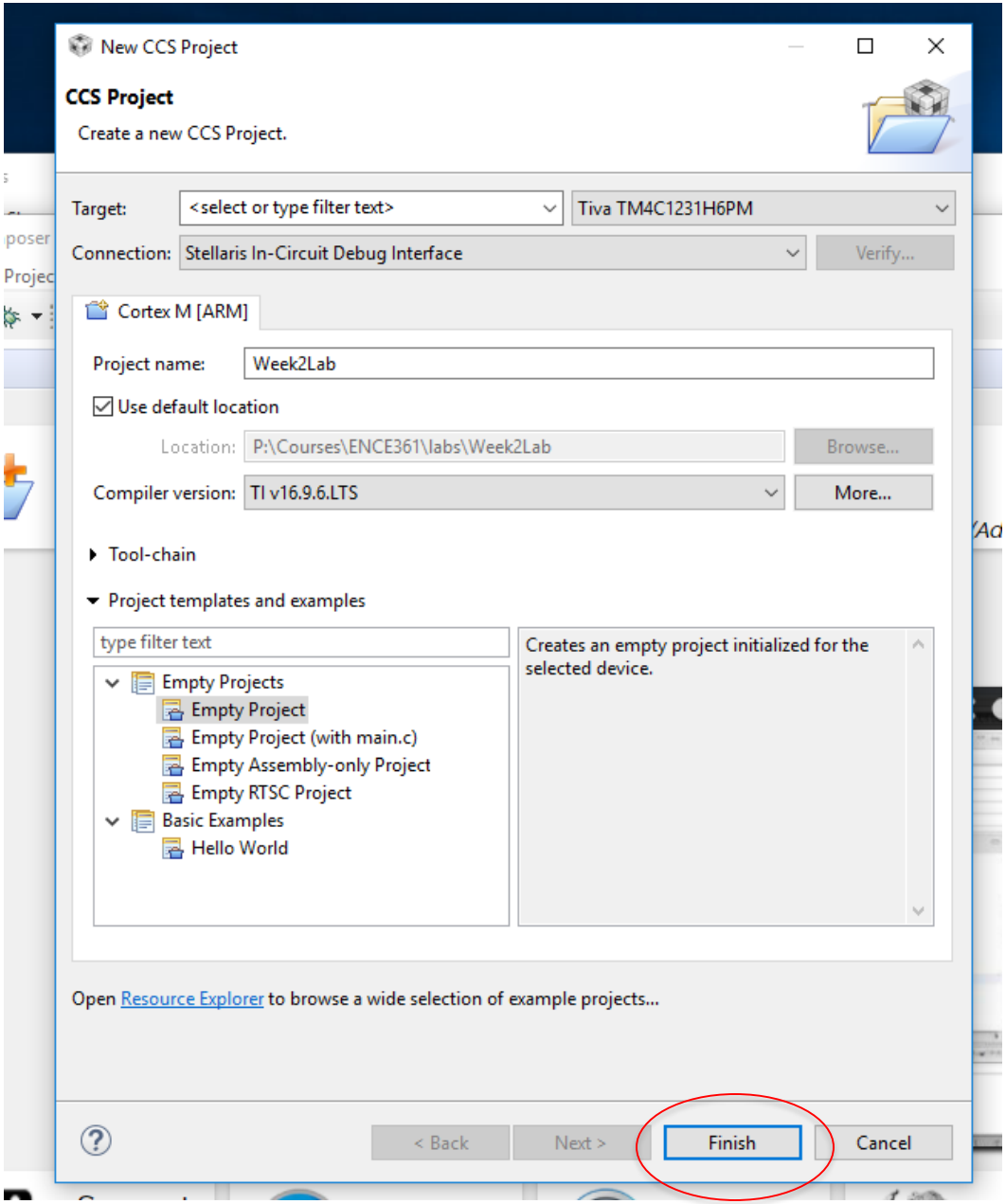
Setting up CCS, cont…

- Click on the second “Target” window shown in the dialog box and start typing… “Tiva TM4C123GH6PM”



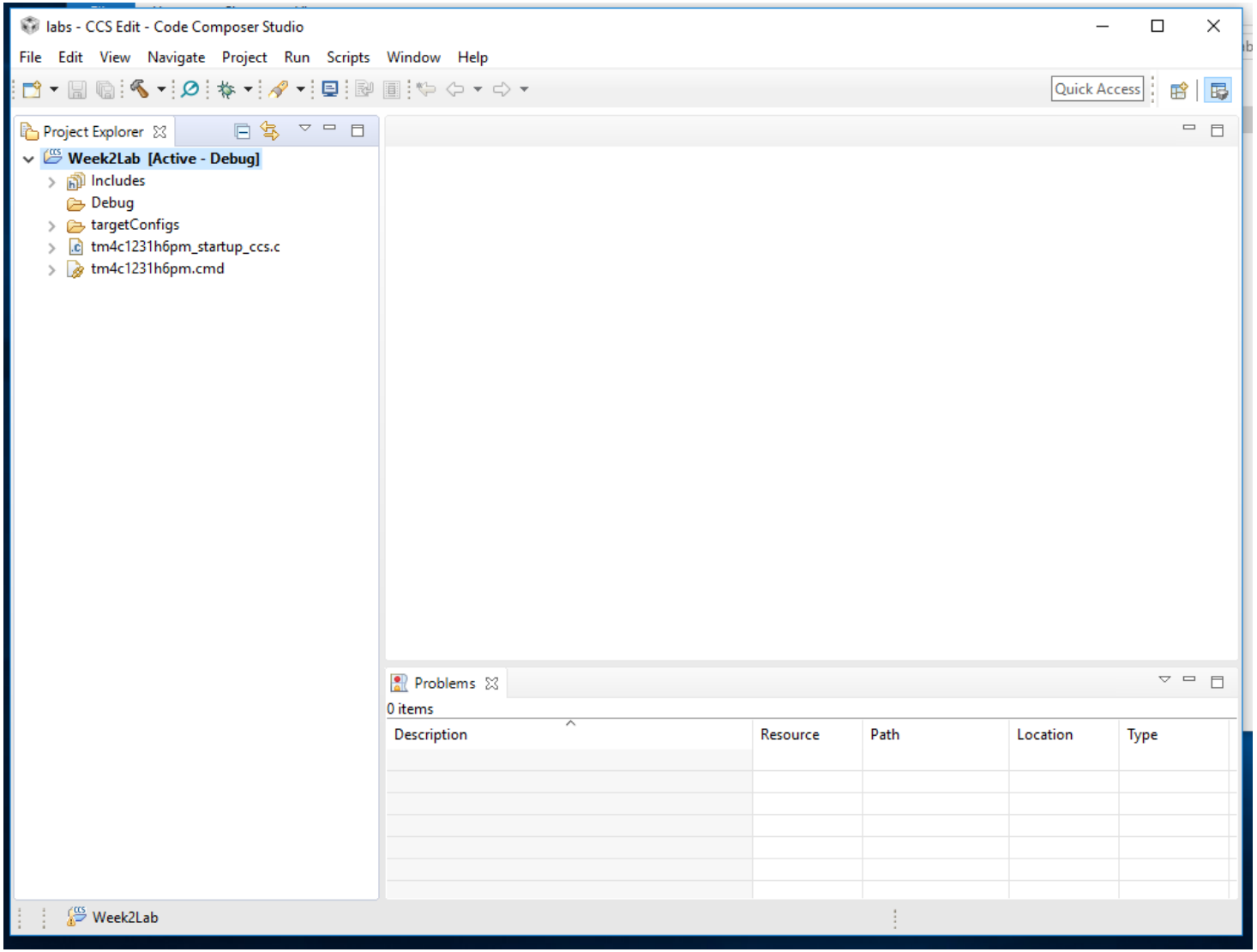
Setting up in CCS, cont.

- For “**Project name**”, type in something descriptive, like: “Week2Lab”.
- For the “**Connection**”, select “Stellaris in-circuit debugger”.
- Lastly, select “Empty project” from the Project “**Templates**” window.



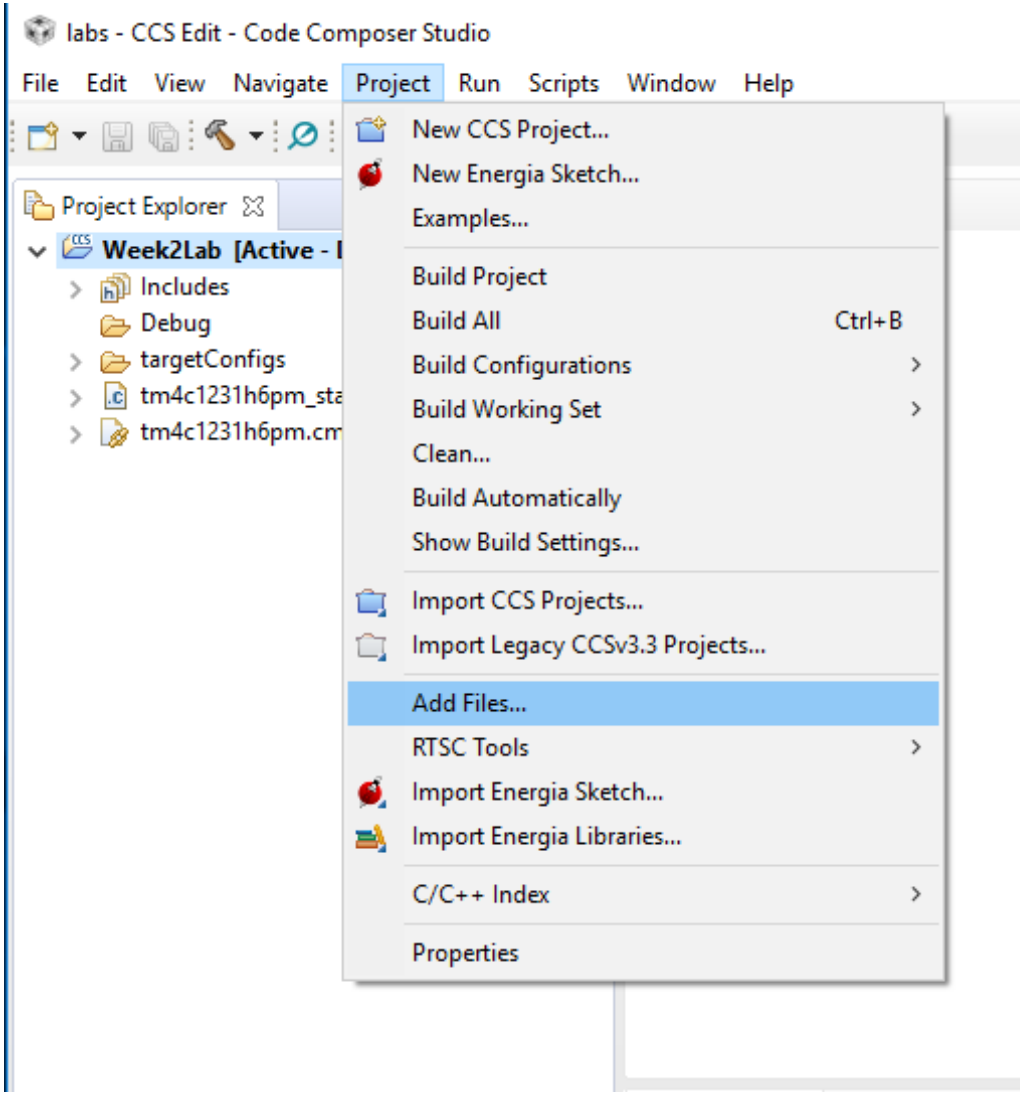
Setting up in CCS, cont.

- You no longer need the “Getting Started” window, so just close this child window on the right of the main CCS window.
- On the left project explorer window, click on the *greater than* symbol to expand the file window and see the initial associated files for the Week2Lab project.

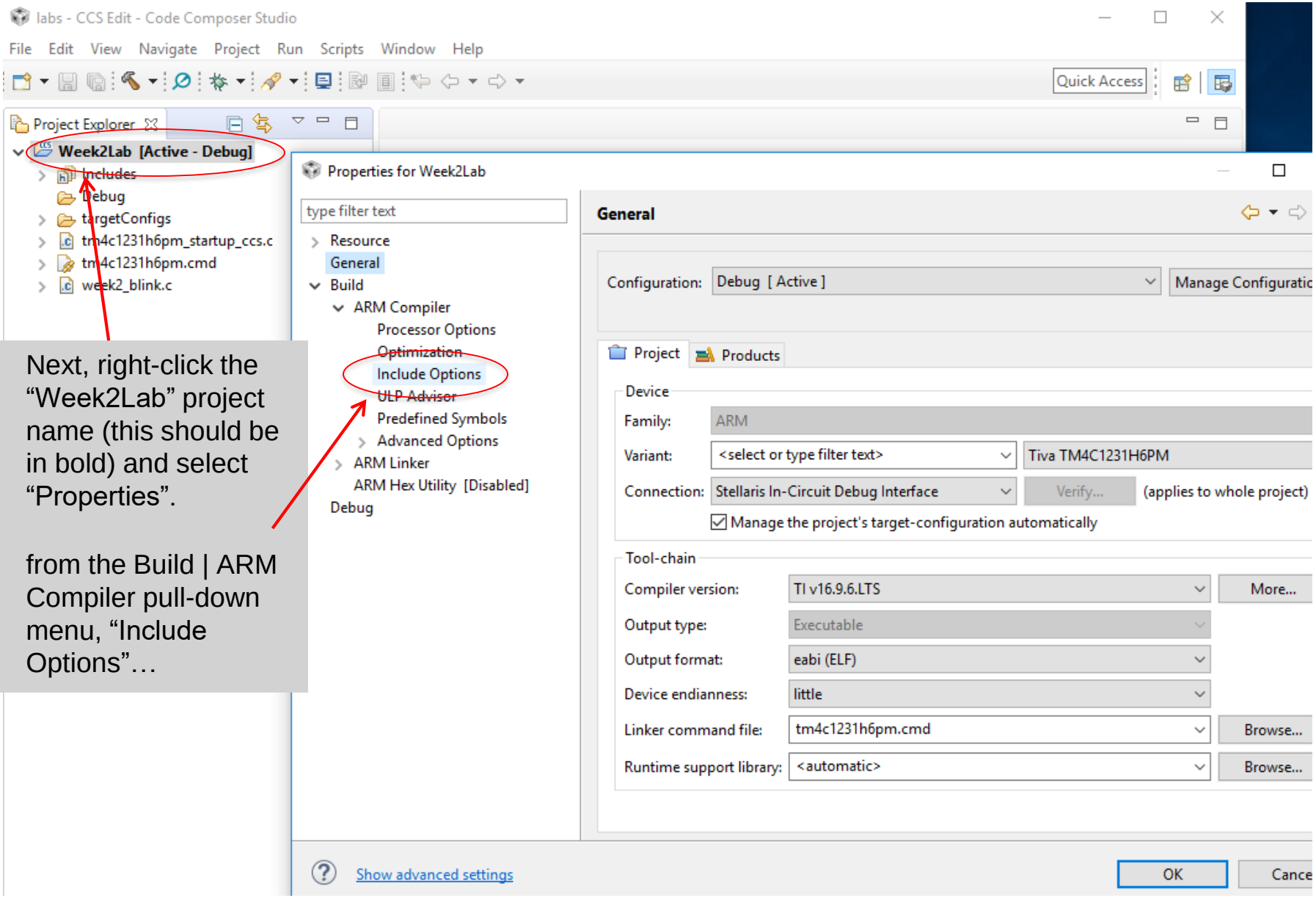


Setting up in CCS, cont.

- Download the source file for the lab from: **Embedded Systems | Laboratories | Laboratory Source Files | Week 2**
- For Week 2 there is only a single file, **week2_blink.c**, but there will be several for the following labs.
- Copy the source file into your **P:\courses\labs\Week2Lab** folder. **Note**: from Learn you can right-click on the file you want to download and select “save as” to place this directly into your P:\courses\...\Week2Lab subdirectory.
- Get back to your CCS application and look at your project explorer again. Notice something different? The **week2_blink.c** file has been automatically loaded into your project. **Note**: you can exclude files that are in your project directory from compiling, but more about this later...
- Note, you can also use “Add Files” from the menu, as shown in the right.



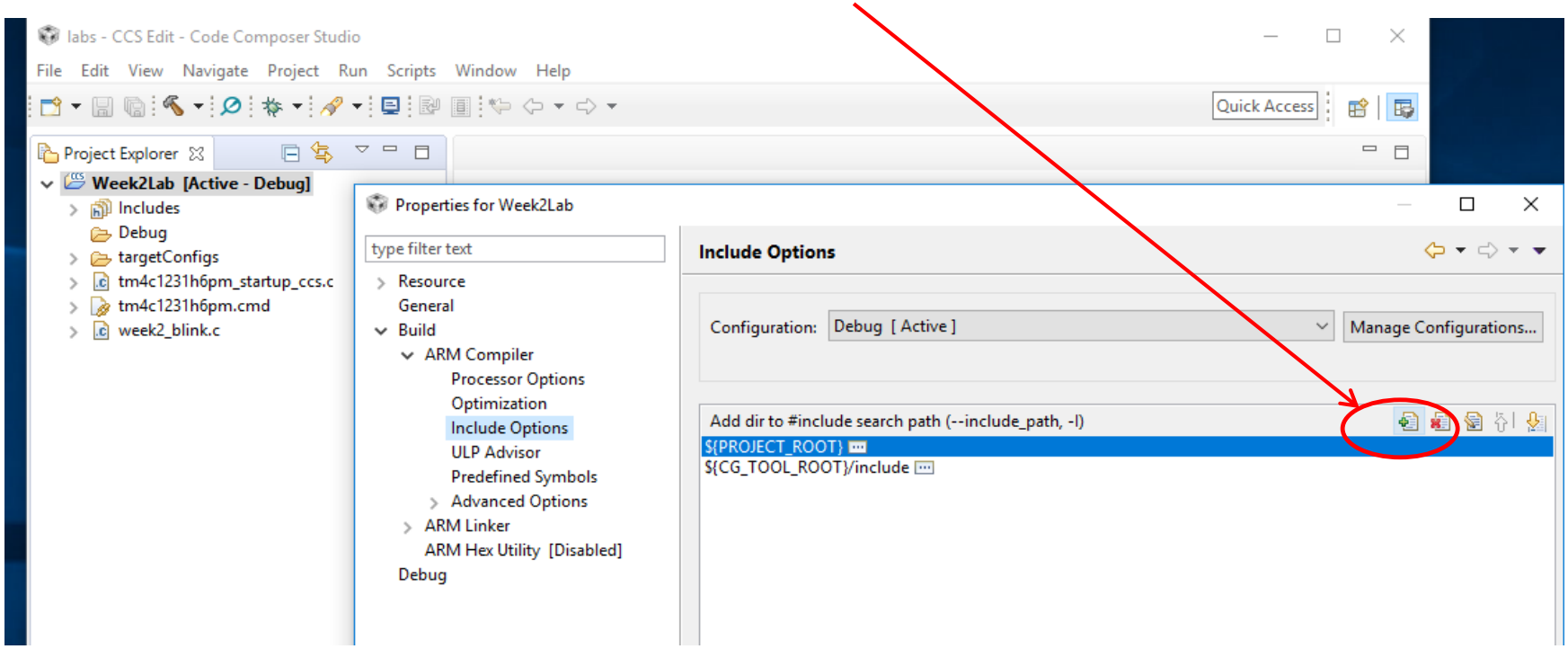
Setting up in CCS, cont.



- Next, right-click the “Week2Lab” project name (this should be in bold) and select “Properties”.
- from the Build | ARM Compiler pull-down menu, “Include Options”…

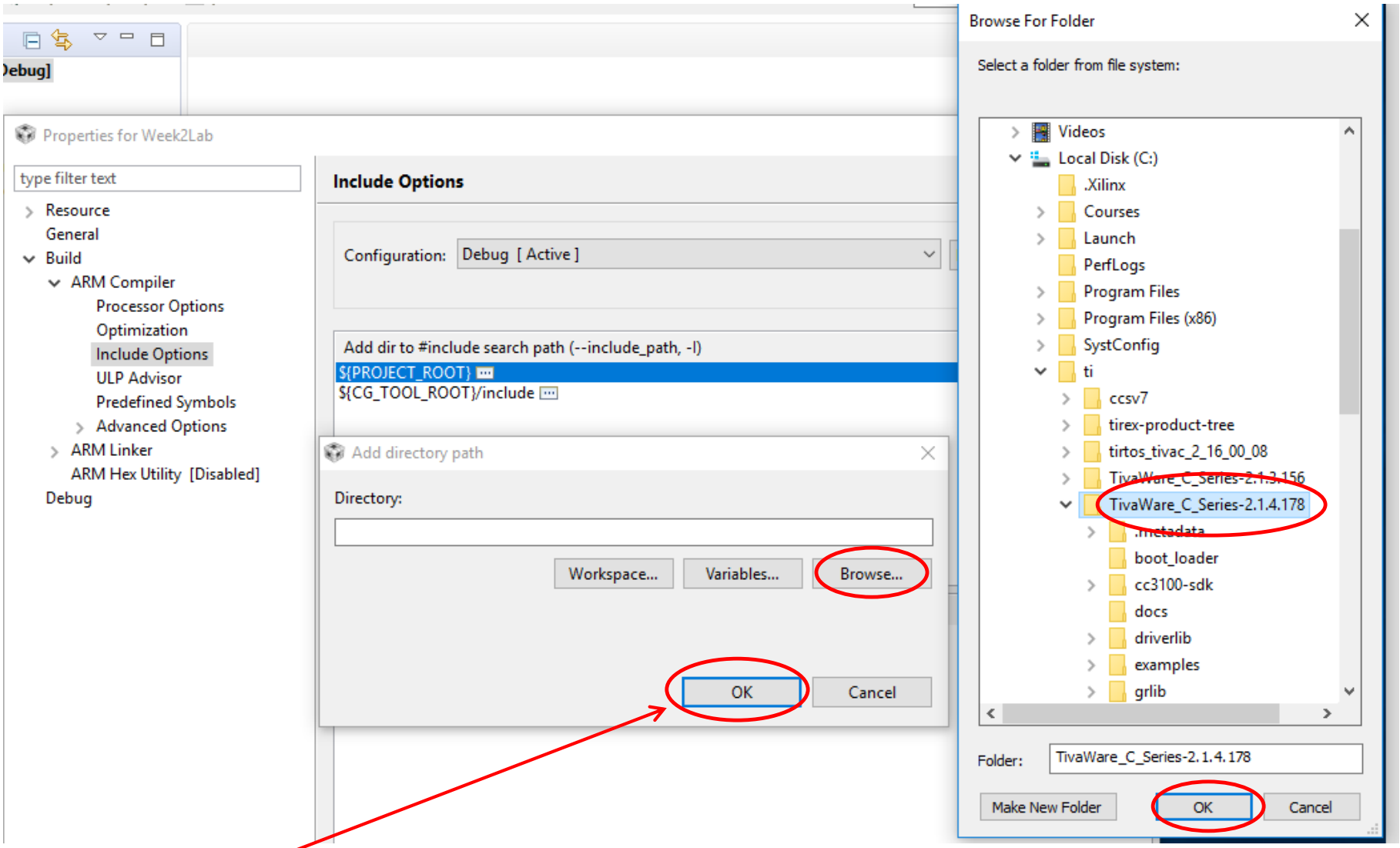
Setting up in CCS, cont.

- In the top window, click on the green '+' button



Use the browse button to find the "TivaWare_C_Series_2.1.4.178" directory in C:\ti. This will allow your CCS compiler to find the include files that CCS requires in order to build your project. See the next slide for a screenshot of this…

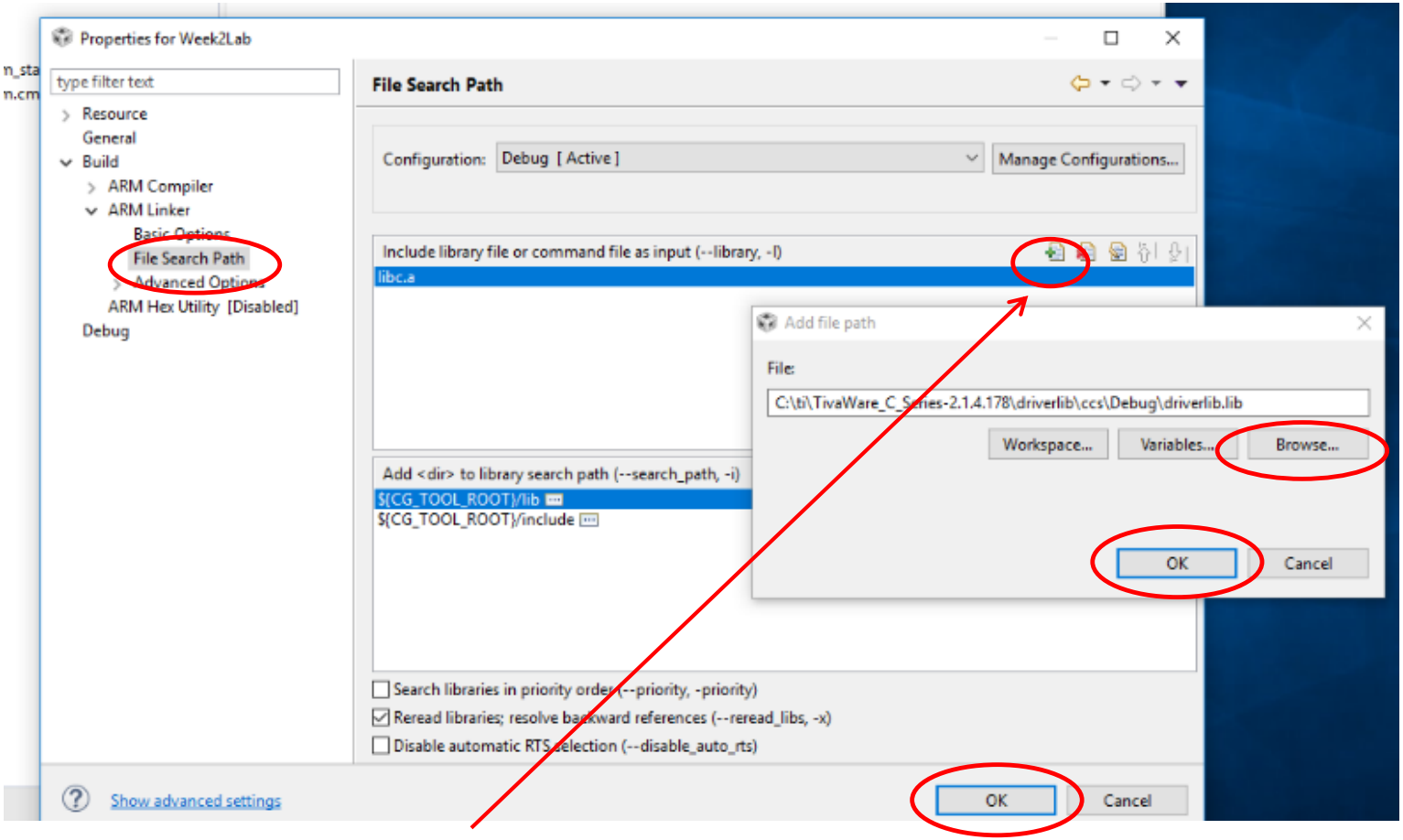
Setting up in CCS, cont.



Press this last to update the include options. **Don't** press the Properties dialog box "OK" just yet.

Setting up in CCS, cont.

- From the same “Properties” dialog box look down the list on the left side and select, “Build | ARM linker. Select, File search path”.

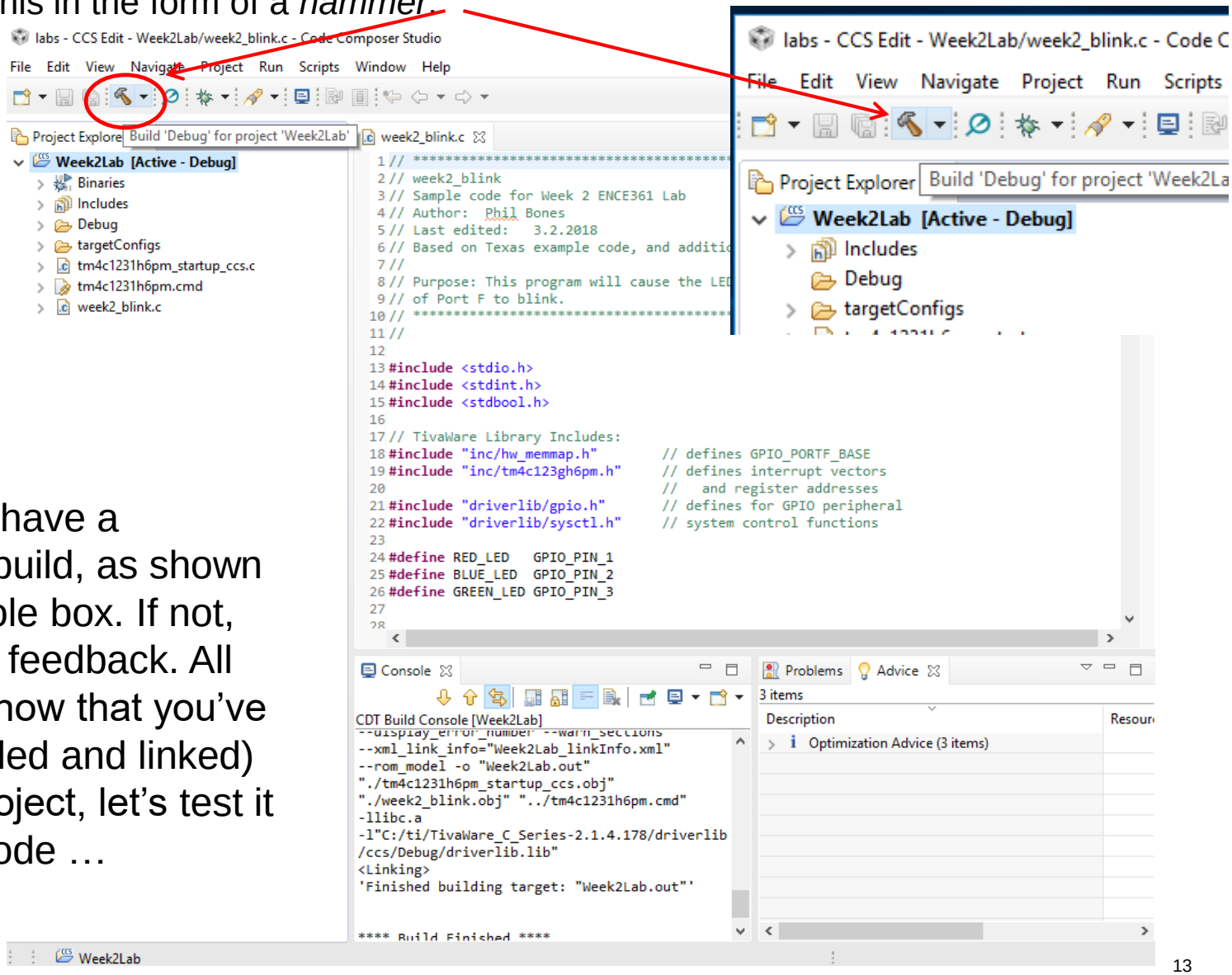


- Like before, click the green ‘+’ button in the include library dialog box. Also as before, select the browse button. Find “C:\ti\TivaWare … \driverlib\ccs\Debug” and select “driverlib”.Press OK to close the “Add File path” window. Lastly, press OK to close the “Properties” window*.

* It may be necessary for later programs to adjust the heap and/or stack size. See Slide 21.

Compiling & linking

- Lastly, under the menu item “Project” select “Build project”. Note, there is an icon that represents this in the form of a *hammer*.



- You should have a successful build, as shown in the console box. If not, analyse the feedback. All being well, now that you’ve built (compiled and linked) your first project, let’s test it in debug mode …

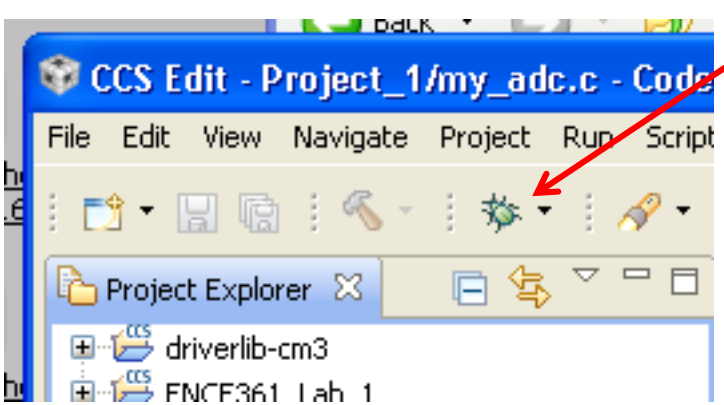
Debugging…

- After unwrapping your Tiva board and checking the contents, use the USB cord to connect your board to one of the two front USB sockets on your bench PC.
- The device driver should already find your board. If not, consult a TA.
- Go to the “Run” menu item and click “Debug”. Alternatively, click on the bug icon (see next slide).
- If you get asked for a “New Target Configuration”, choose “Yes”, “Stellaris In-Circuit Emulation” and “Tiva TM4C123GH6PM” → the file **NewTargetConfiguration.ccxml** will appear in your project directory.

Loading and debugging your project…

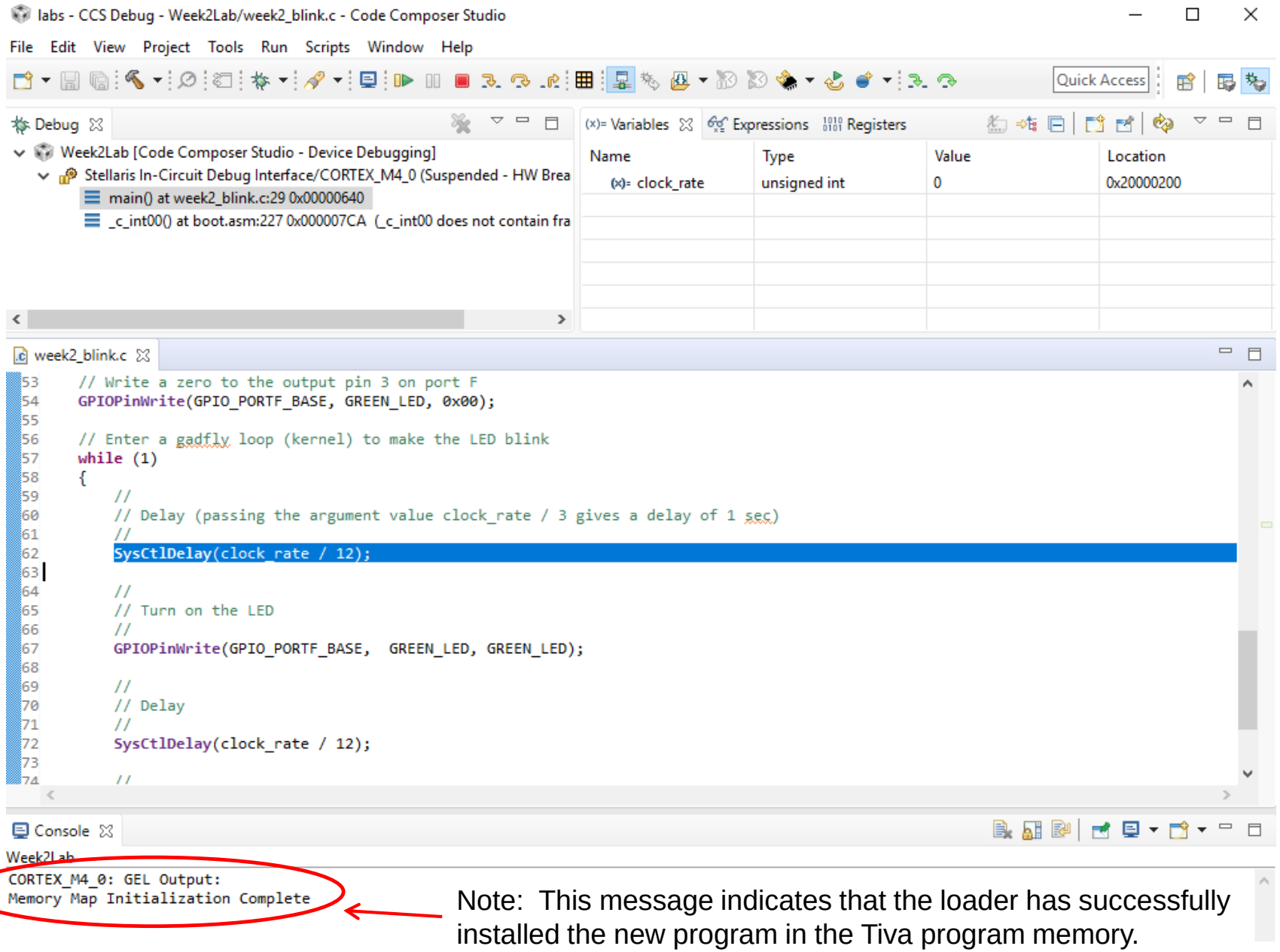
To debug your project, make sure your Tiva board is plugged into a PC via one of the front USB ports. Give the software time to install...

- With your project successfully built, a “Bug” icon will be visible just to the right of the hammer. Click on this to load and debug your program…



- The next few stages are fully automated, but it does take a few seconds before the debugger is fully operational.
- Eventually, you will see ….

THE DEBUGGER SCREEN!



Note: This message indicates that the loader has successfully installed the new program in the Tiva program memory.

Using the debugger…

- Run your program in the debugger by either selecting “Run” and select “Resume” or click on the green left pointer arrow from the menu icon bar. The green LED should flash.
- To stop your program in the debugger, click on the yellow parallel lines icon. Typically, the edit screen will either show an assembly listing or a “no source code available” message in red on a white screen. There is a good reason for this, which will be discussed in your tutorial.
- Click back on **week2_blink.c** and find the TivaWare function, GPIOPinWrite().
- Set a breakpoint on the GPIOPinWrite function. Do this by clicking on the line associated with this function at the left of the window (in the blue hash region). A dark blue diamond shape will be inserted in this blue region, associated with the function line number.
- Resume your program execution but pushing F8 or clicking the green resume arrow. Did your program stop? What is the status of the green LED? What happens if you press F6 or the “Step over” button?

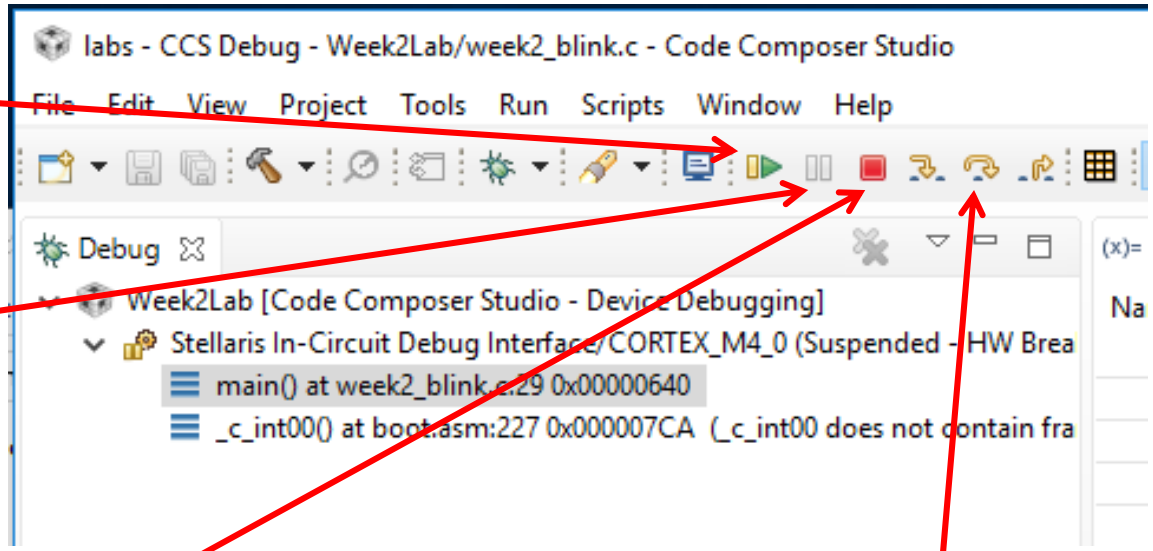
Run & debug your program on your Tiva…

Also see:
http://processors.wiki.ti.com/index.php/GSG:CCSv5.0_Debugging_projects

- To run your program that you have already loaded into the Tiva memory, click-on the green arrow…

- To stop the program, click on the two yellow parallel lines…

- Close the debugger by clicking on the red square… or go to the top right utility and switch between debug and edit mode (much faster!).



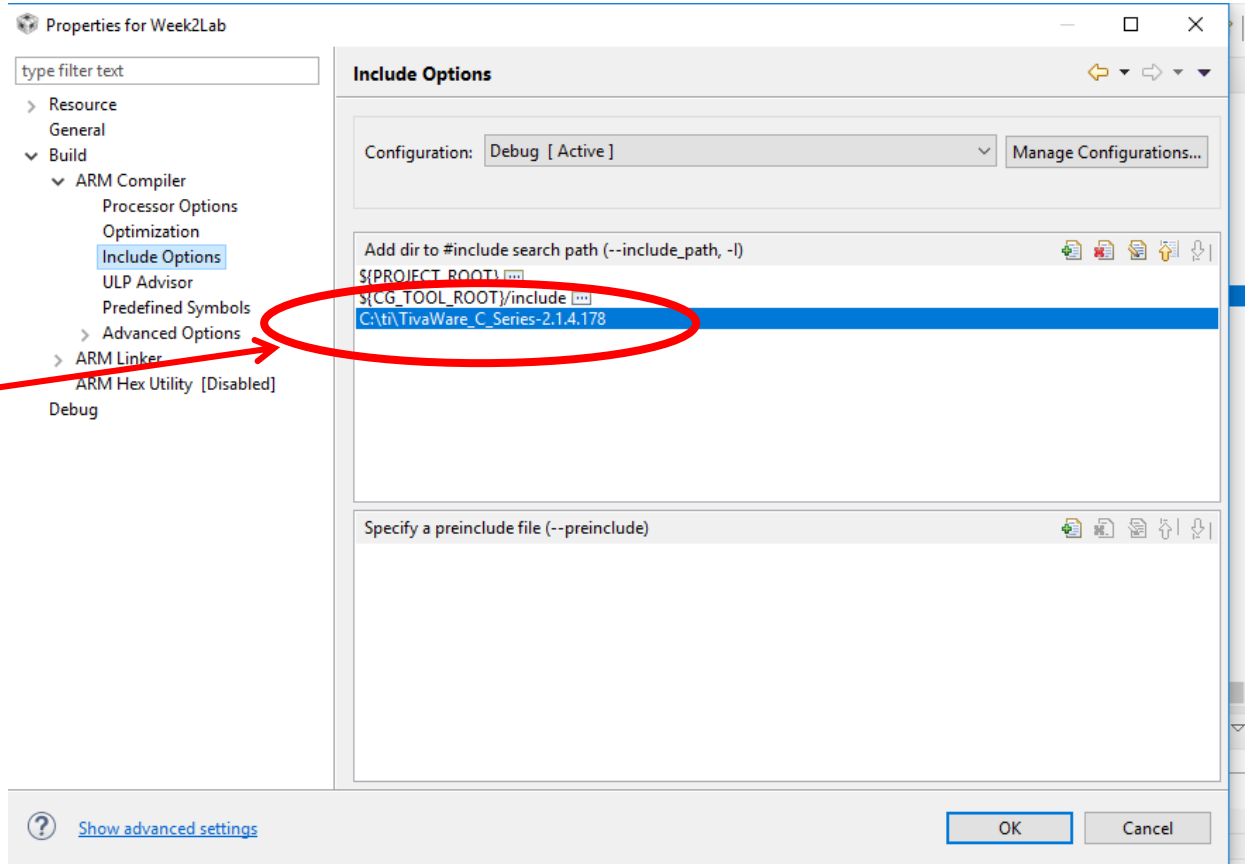
In the Help screen for CCS, select “stepping” in the index for more information.

- To single-step (“step over”), click on the right yellow arrow…

Trouble-Shooting (Slide 1 of 3)

- In your Project Explorer window, right mouse-click on your “Lab_GPIO” project, shown as “[Active – Debug]”, and select `Properties`.
- In the project properties window on the left side of the dialog box, select: “**Build | ARM Compiler | Include Options**”. You should see...

Make sure your 2 paths, especially your drive letter, are correct in the top window.



And while you’re in your project properties window…

Trouble-shooting (Slide 2 of 3)

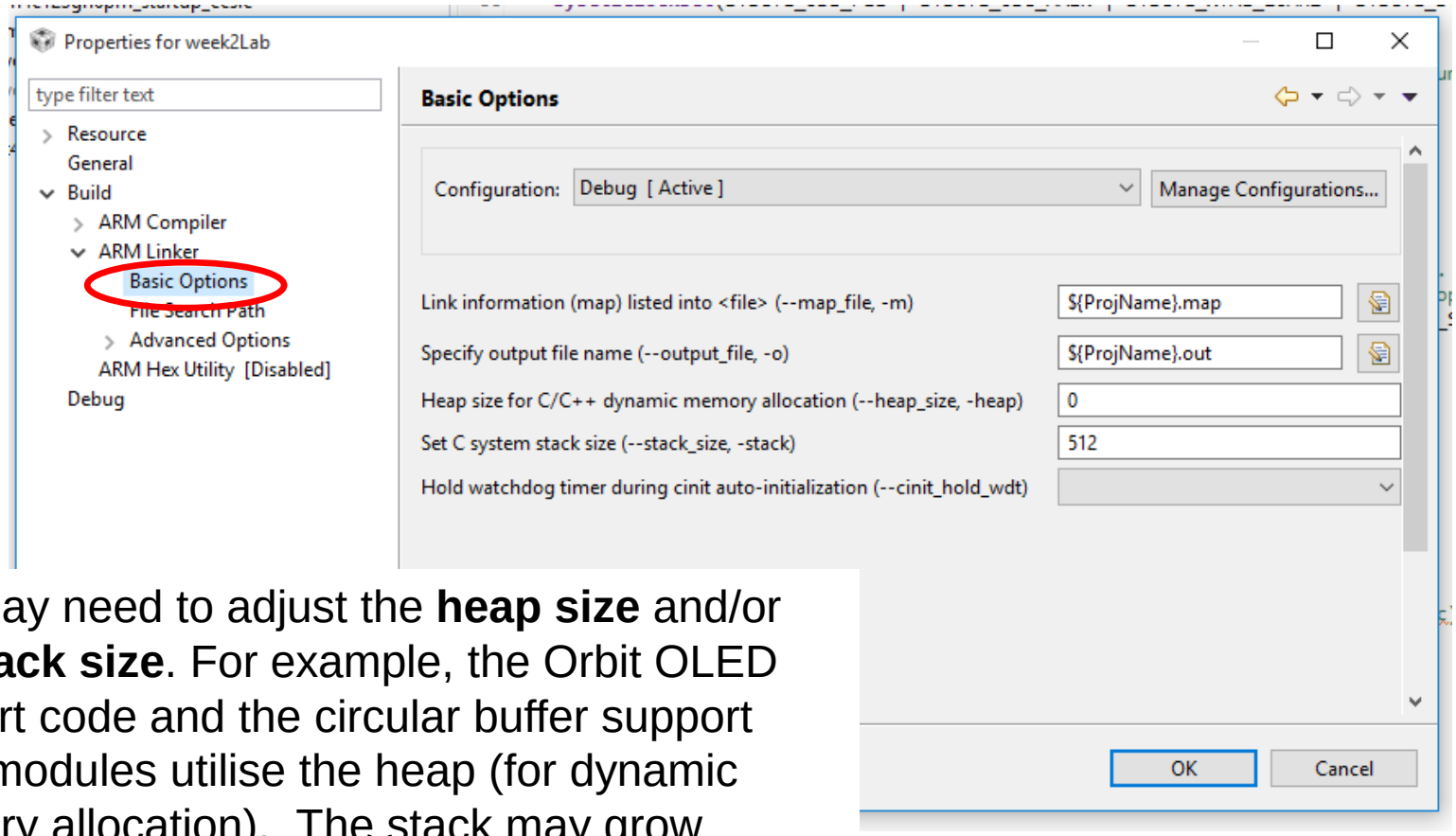
- Still in the project properties window, on the left side of the dialog box, select: "**Build | ARM Linker | File Search Path**".

Again, make sure the drive letter and path match the location of the Tivaware API.

Make sure the location of the **driverlib.lib** file is correct, c.f. Slide 12.

Trouble-Shooting (Slide 3 of 3)

- Still in the project properties window, on the left side of the dialog box, select: "**Build | ARM Linker | Basic Options**".



- You may need to adjust the **heap size** and/or the **stack size**. For example, the Orbit OLED support code and the circular buffer support code modules utilise the heap (for dynamic memory allocation). The stack may grow beyond 512 bytes, if your code has lots of nested loops. Both sizes are in bytes.