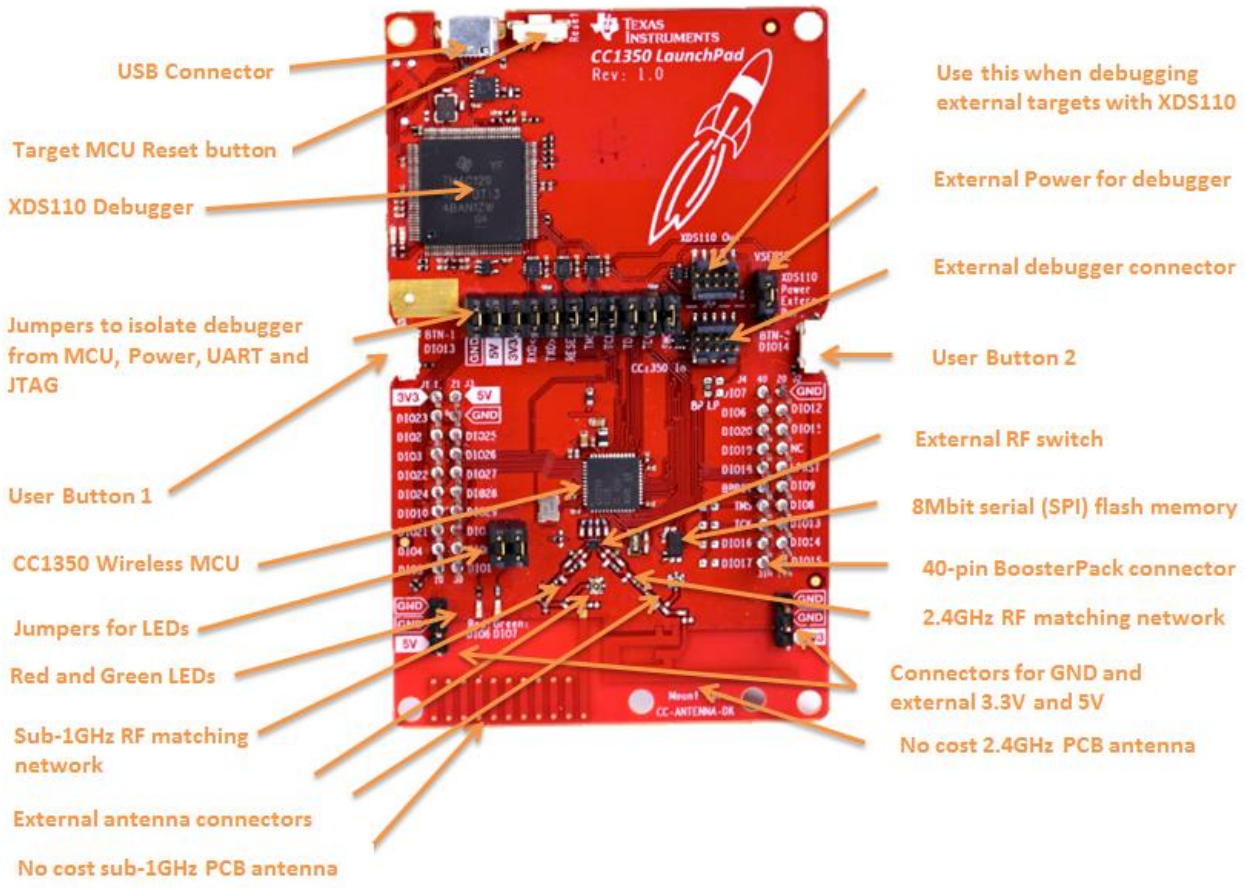
Lab: Introduction to Code Composer and Energie on the MSP432P401R

Introduction:

For this class you will be using a couple of different development environments to introduce you to concepts associated with Embedded Systems. You will be using two different kinds of environments, one that allow more access to the details of the hardware, the other which will abstract, or hide, the details. The objective of this lab is to get you up and running with the hardware and the software.

Hardware:

The hardware you'll use for this lab is the MSP432P401R. Here is a picture of this hardware, with all the key parts of the hardware noted:



You'll use a USB cable to connect the hardware to your host computer.

Code Composer

In this first section of the lab you'll install Code Composer V8.

Installing the Software:

You'll have a choice in using Code Composer to develop you applications. You can either use the Cloud version, where the code actually is run on a remote server and you access the capabilities through a web browser, or you can use the downloadable version, where the code runs on your local machine. I prefer the latter, so I don't have to worry about always having internet access when I develop.

To install the software download the appropriate installer at:

http://processors.wiki.ti.com/index.php/Category:Code_Composer_Studio_v8

The selections will look like this:

Download

Download 8.1.0.00011	Installers (Offline installer is recommended for slow and unreliable connections)	Notes
Windows	Offline Installer Online Installer	
Mac OS	Offline Installer Online Installer	
Linux 64bit	Offline Installer Online Installer	

Move to Windows 64bit in 2019 Our current plan is to migrate CCS to be a 64bit application on Windows in 2019. This will mean that it will not work on 32bit Windows systems. People with 32bit Windows Systems would need to stick with CCSv8 or earlier.

Version 8.1.0.00011 **Linux Installation Instructions** | **System Hardware Requirements** | **Training Material**

Licensing: CCSv8 is Technology Software Publicly Available (TSPA) compliant. This means that it does not require a paid license.

Update Status: This release will not be available as an update.

Mac Users Please note that only microcontroller and connectivity devices are supported on Mac. Processors devices are not support. See MacOS Host Support CCSv8 for more information. Also if you do not have adminstrative rights on your Mac then you will need to run the installer with a command that looks like this: xattr -r -d com.apple.quarantine ccs_setup_7.1.0.00016.app (replace the filename with the version you are using). If you do not do that then MacOS will copy the executable to another folder and run it from there, as a result the installer will not be able to find the offline files and will run as a web installer.

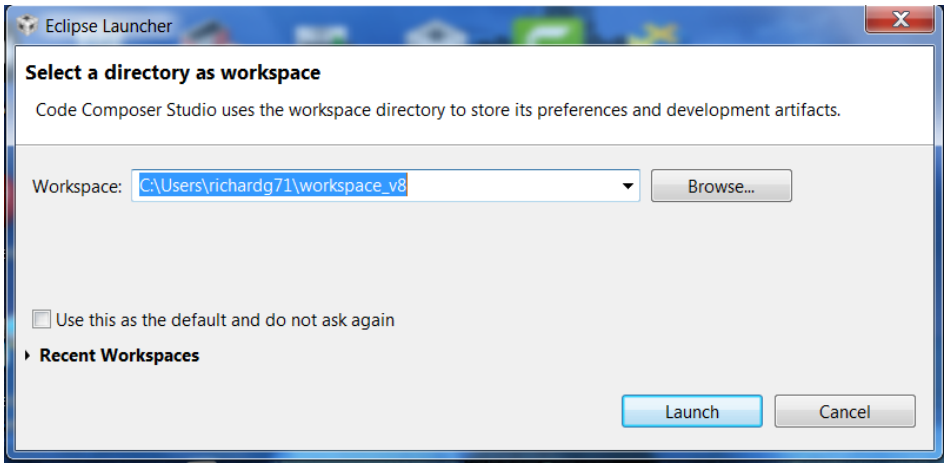
Linux Users Please note that there is an issue with certain version of the linux kernel that if installed will prevent CCS from starting and RTSC based projects from buiilding. A fix is available in the latest kernel updates. If you are running into the issue, please make sure you have the latest updates applied.

Once you have downloaded the installer, run the installer.

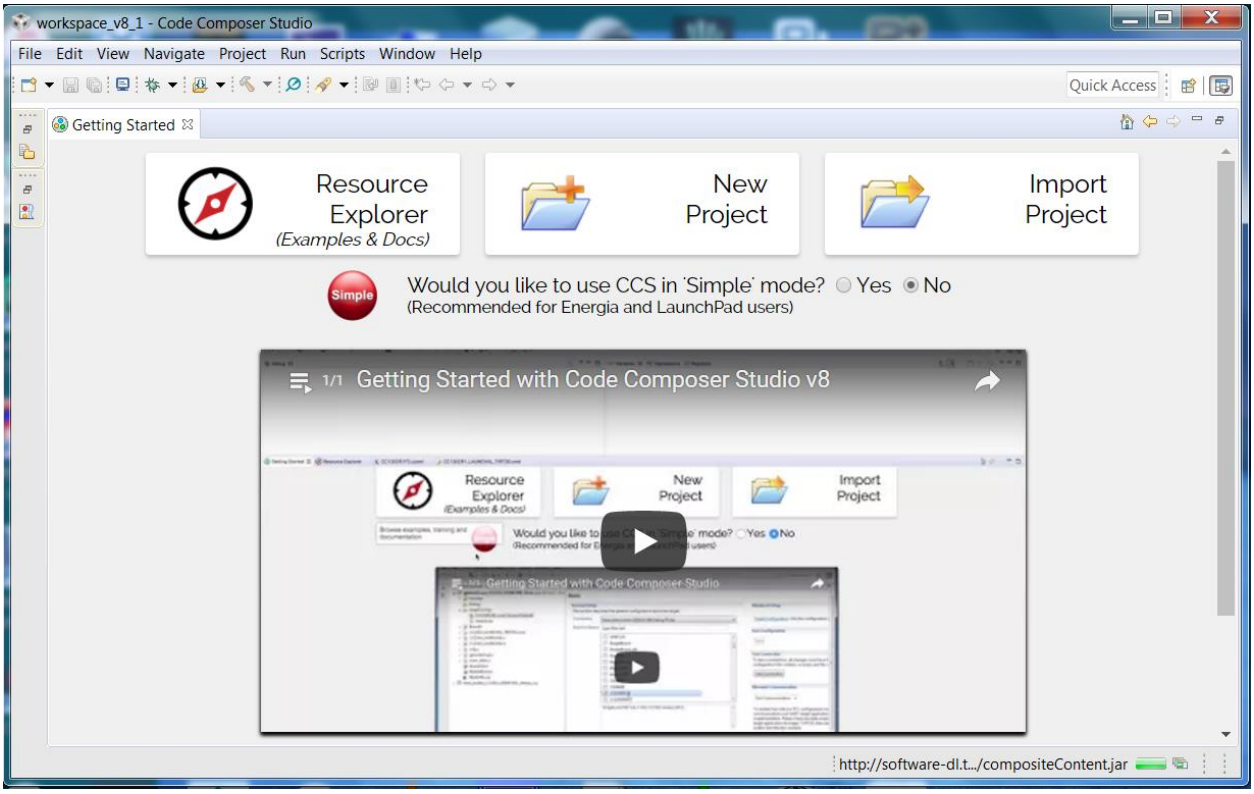
Creating your First Application:

Now that you have access to the software, let's create your very first application. First, start the software by double clicking on the icon:

As the program starts you should be prompted for the workspace you want to use.



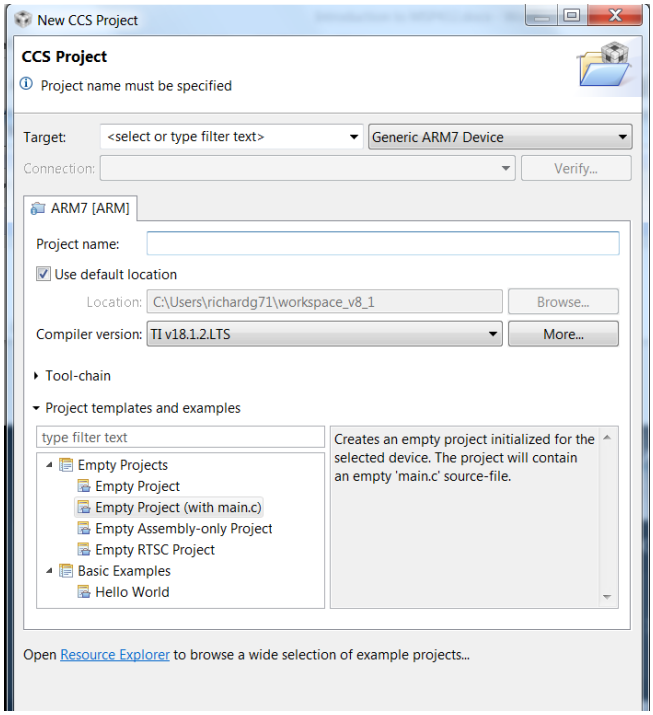
This workspace is the directory where you project files will be placed. You can use several different workspaces if you want to work on very different projects. Click Launch when you have specified the work space directory (you can just use the default.)



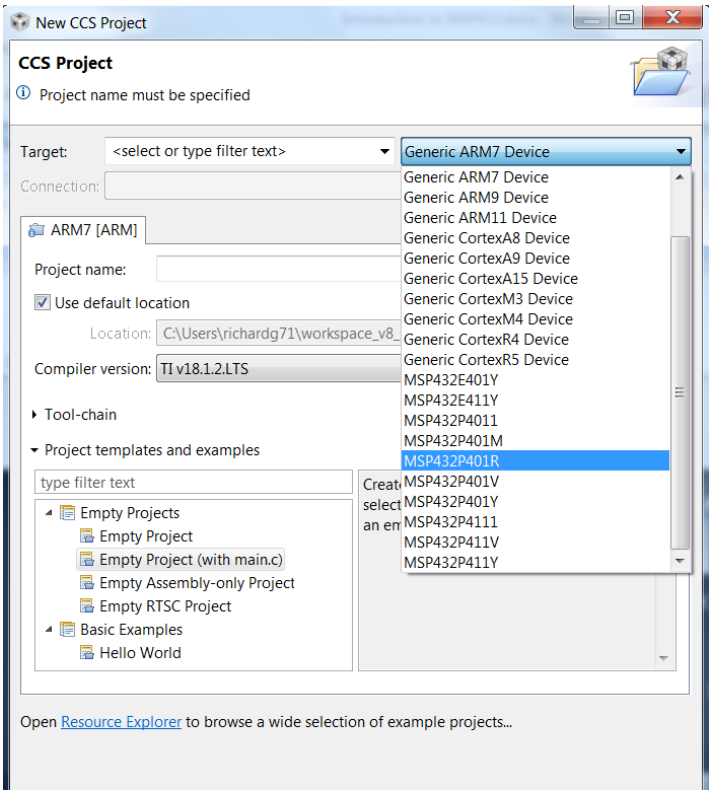
This is the introductory screen. Feel free to watch the Getting Started with Code Composer Studio v8.

If you haven't already connect your MSP432P401R to the computer via a USB cable.

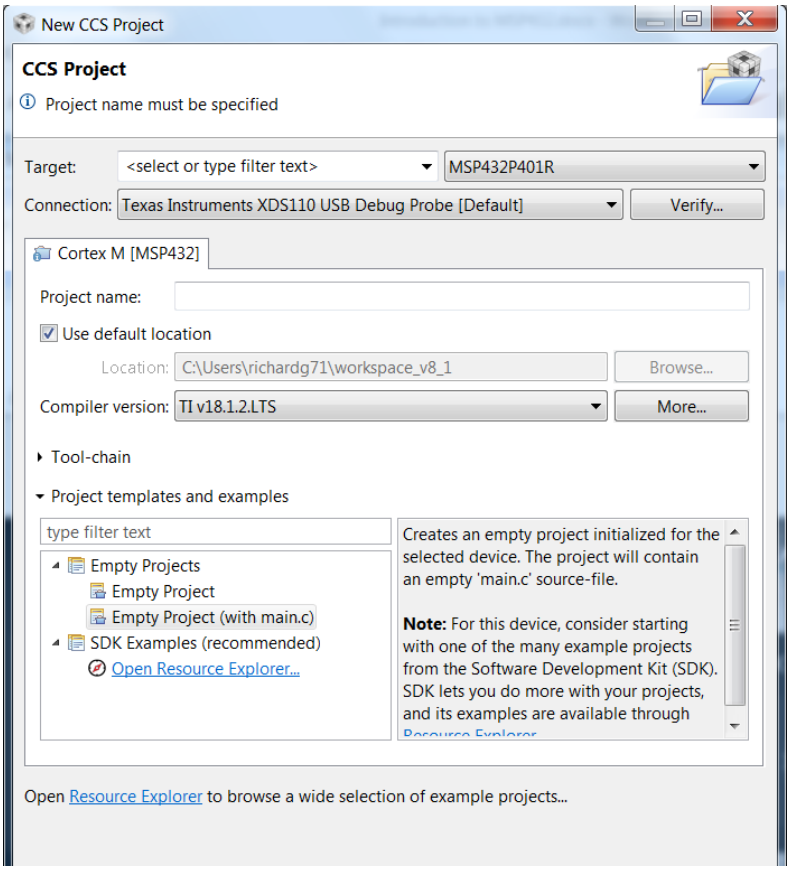
To start a new project press the new Project button at the top of the screen. You should now see this selection:



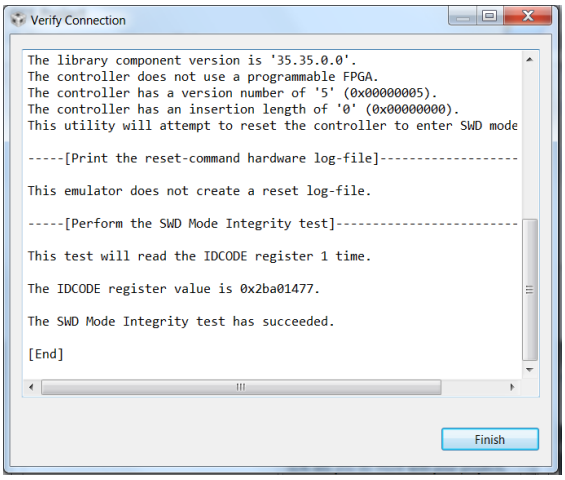
The first you'll want to do is the change the Target device. Under the Target: selection choose the MSP432P401R selection, like this:



You should now see this:

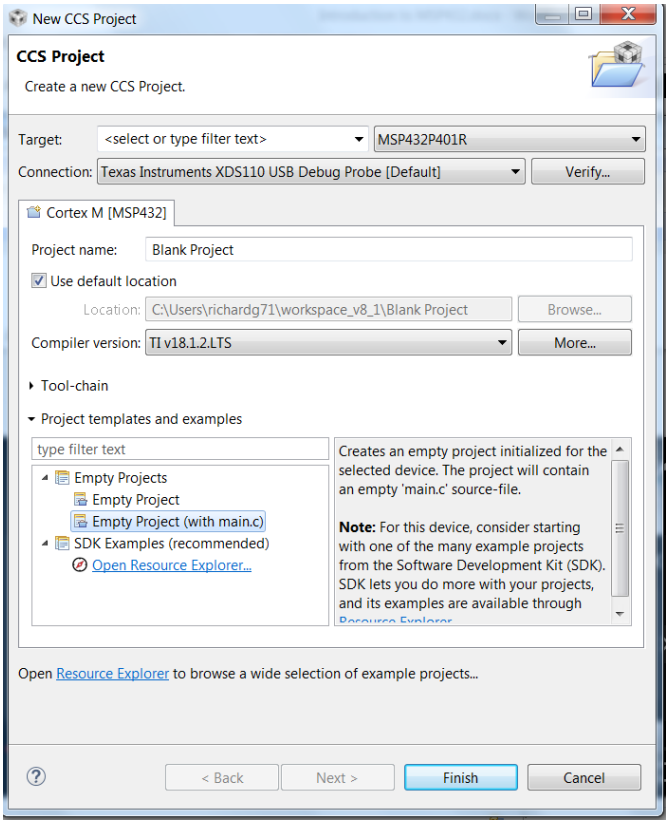


The Connection is showing the Default connection to this particular board, in this case it is a USB connection via the XDS110 interface (hardware on the board.) If you want to verify that Code Composer is connected to your board, then click Verify… and you should see this:

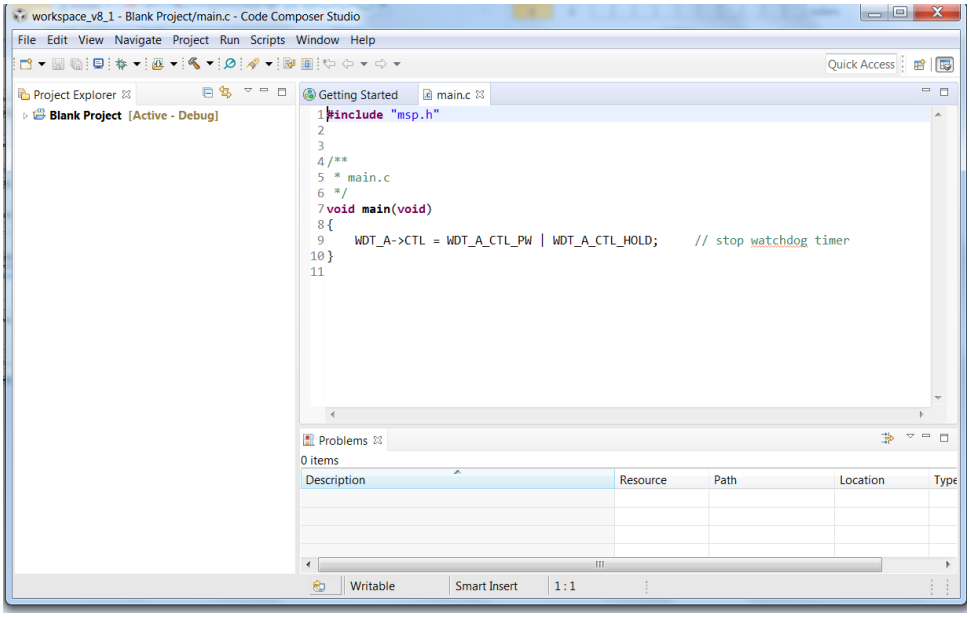


Your device is connected, so click Finish.

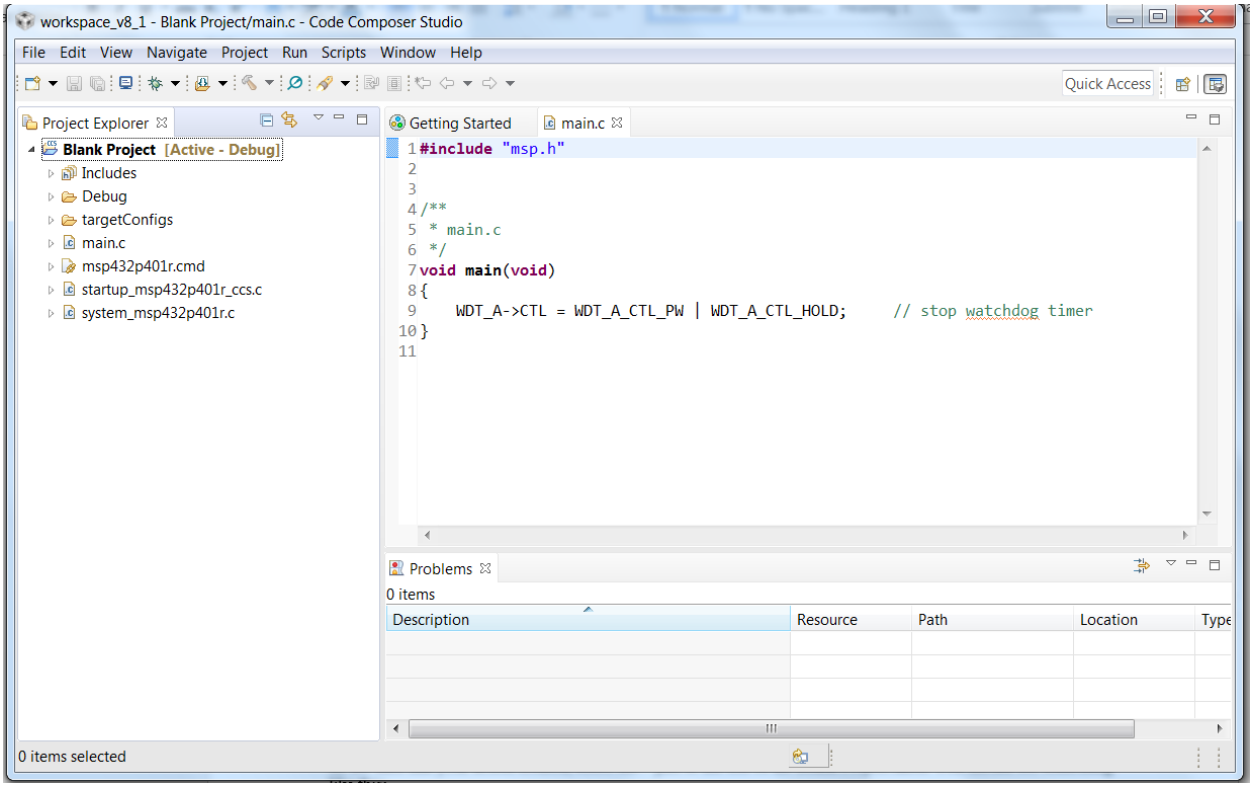
Now you have a choice. You can create a blank project, a blank project with just main.c in it, or you can use one of the example projects. Let's create a blank project with just main.c in it for a moment, just to look at the files that creates. So go ahead and type in a Project name and select Empty Project (with main.c) and select Finish.



You should see this:

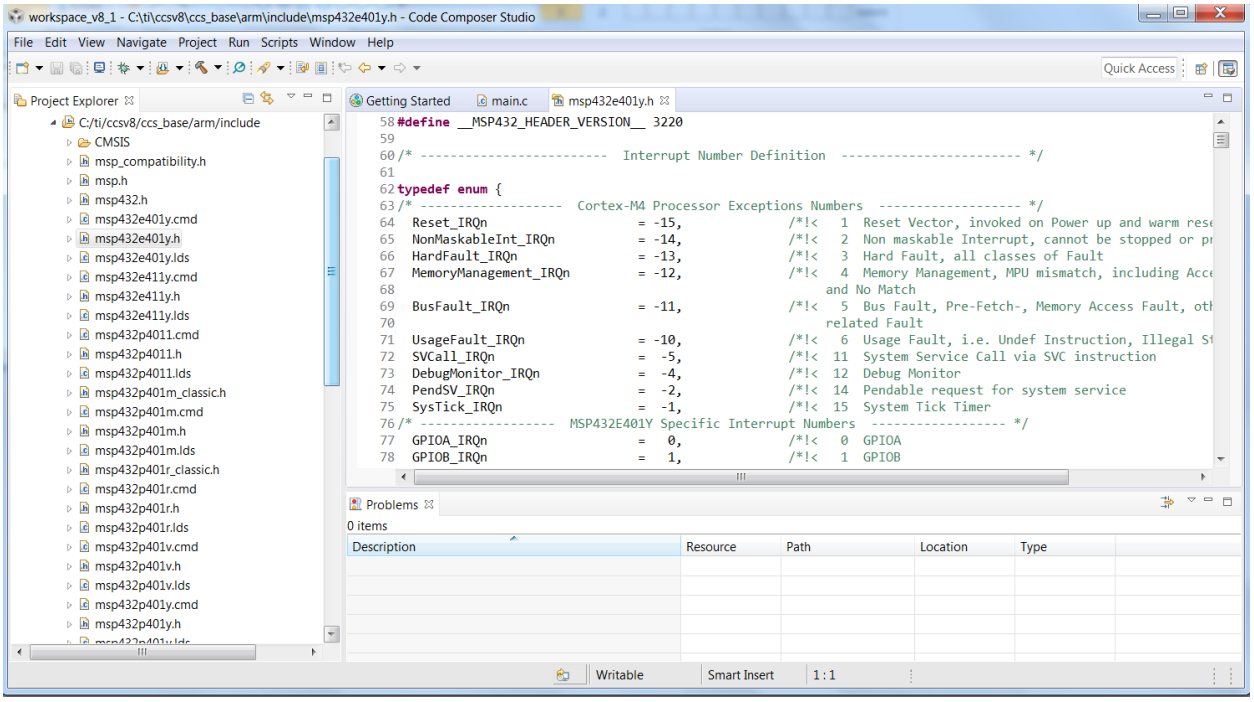


Notice in the Upper Left you now have a black project. The screen is also showing you your main.c file. To look at all the files that your action created, select the > just to the left of the Blank Project label. You should see something like this:



There are several directories and files that are created. Let's look at each briefly.

Includes – These files are the set of includes for all of the libraries that will be compiled together when you build your code. Let's look at just one of these files:

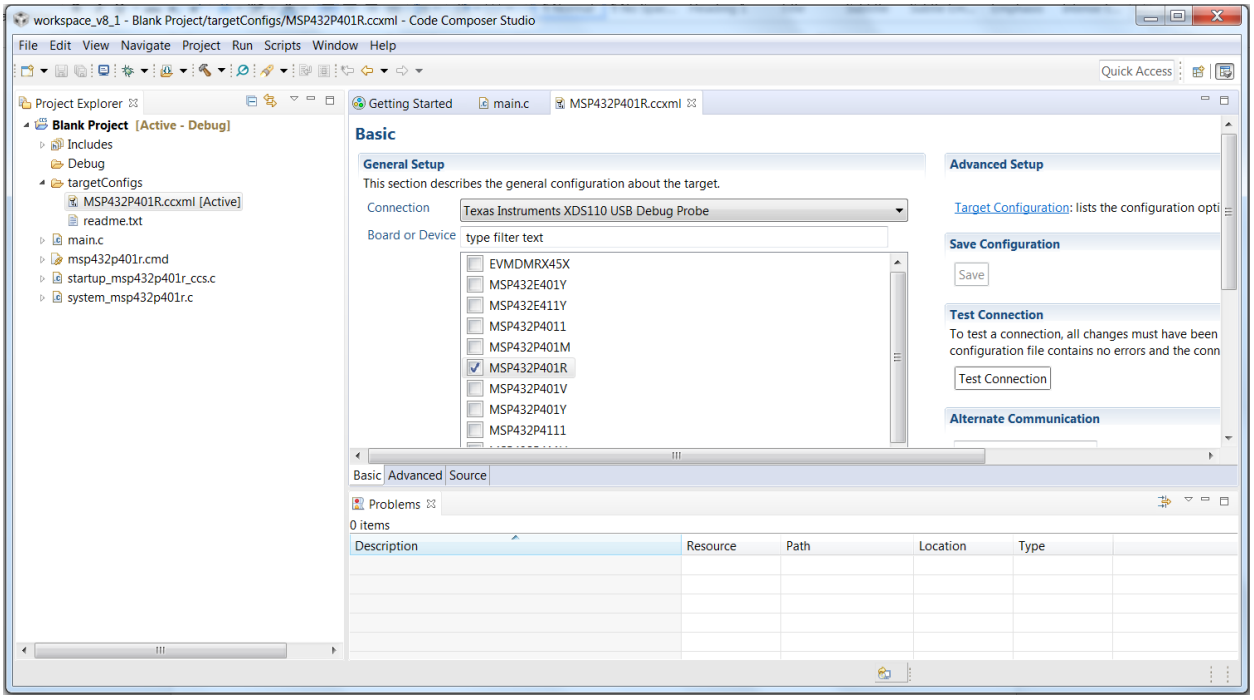


In this case I've opened the C:/ti/ccsv8/ccs_base/arm/include directory by clicking on the arrow just to the left of it, then double clicking on the file msp432e401y.h file. This has a large set of enums (definitional structures that turn numbers into names in the code) that are declared to support this hardware. I would never edit this file, but if I wanted to know what a specific number is defined as in the hardware, I could go into this file.

Get rid of the msp432e401y.h file in the editor area by clicking on the x. And you can also un-expand the directory by clicking on the arrow key just to the left of it.

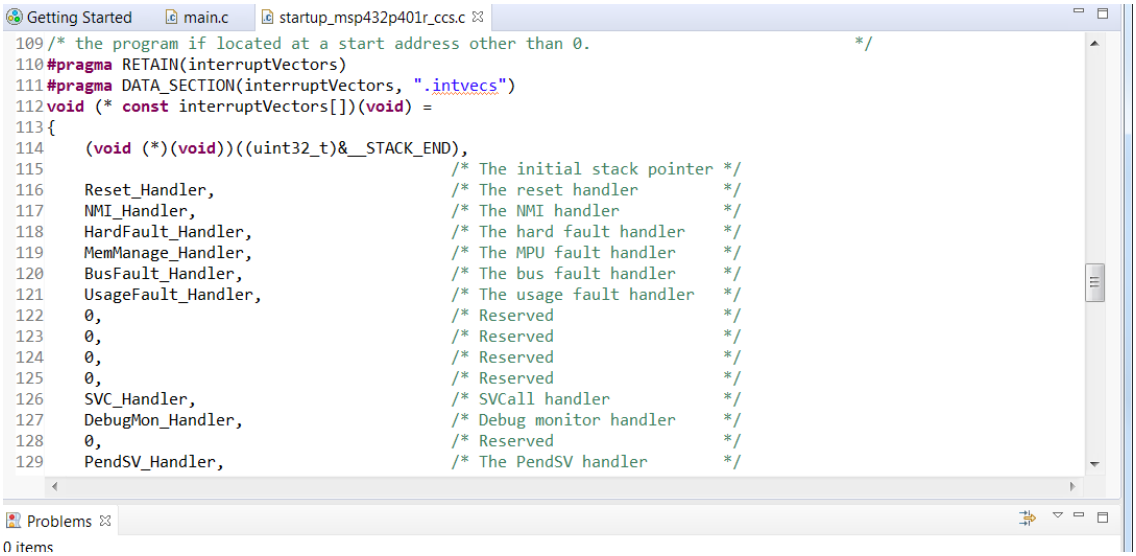
Debug – There is nothing in this file, as you haven't done any Debug. We'll cover this later.

targetConfigs – This directory specifies your hardware connection. If you select the MSP432P401R.ccxml file you will find an active page where you can set this. Don't change it.

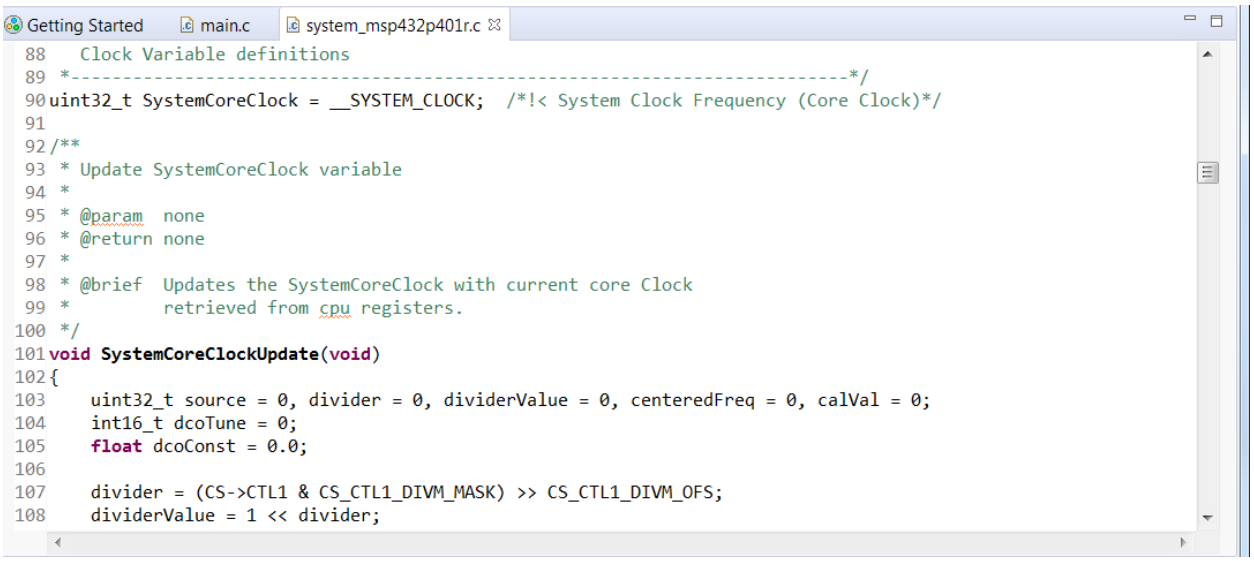


The other files in this directory are code files. The main.c file, as you have already seen, holds the main() functions. Here are details about the other two files:

startup_msp432p401r_ccs.c – This holds the interrupt vector table. We will talk about his later in the class, but is specifies the functions to run when a specific interrupt occurs. Here is what that table looks like:



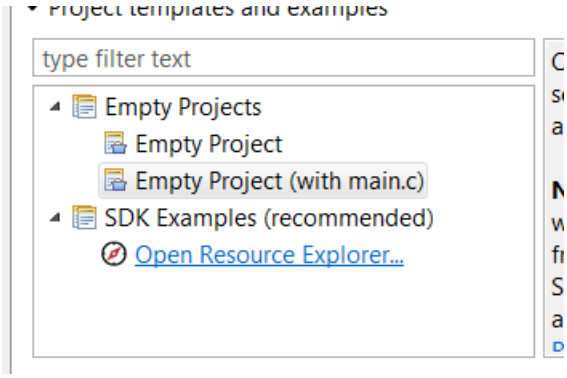
system_msp432p401r.c – This file sets up all sorts of configuration details. Here is a view of bit of the file:



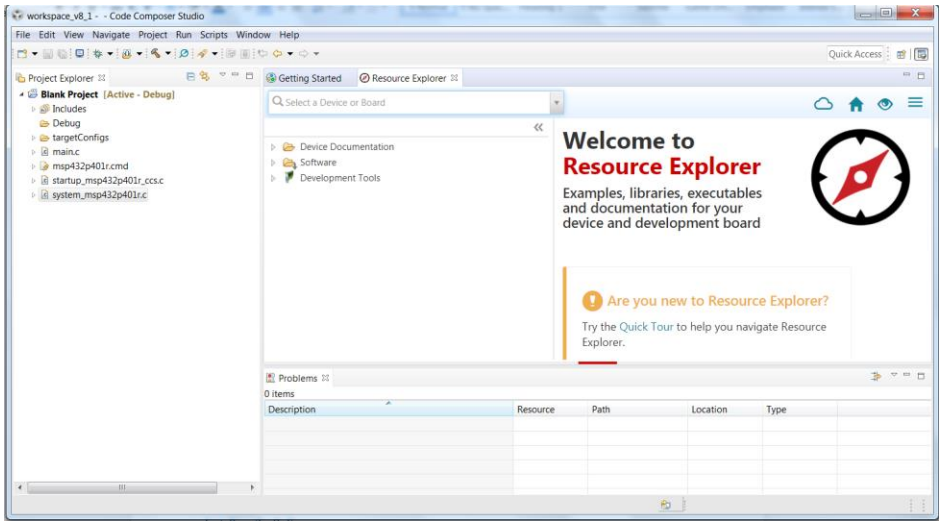
This files takes care of all the configuration details for you.

Now if you were going to add your code, you'd start by adding code to main.c in the main() function. But instead, let's start with a file that has a bit more framework to start with. So click the New Project selection from the main screen (to get back to it just close all the other files in the large main window by clicking the x next to them.)

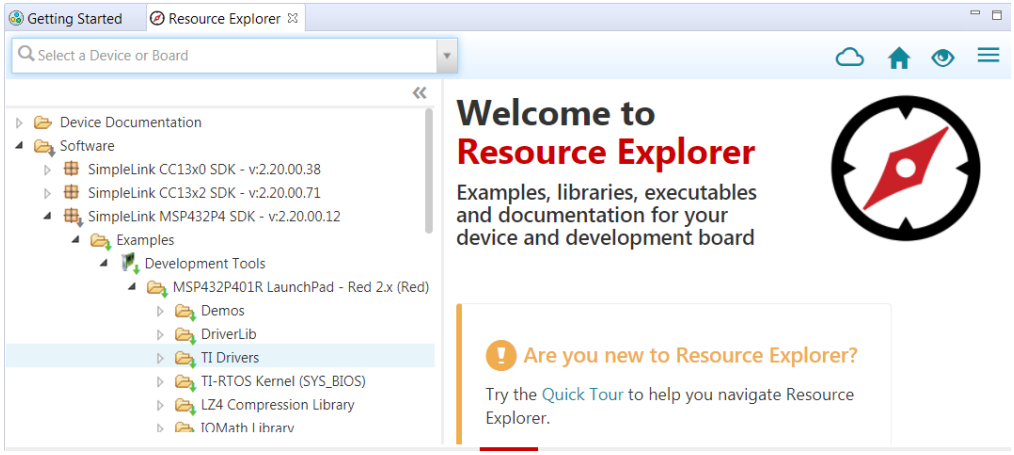
This time under the SDK Examples (recommended) selection select the Open Resource Explorer…



The following should pop up:

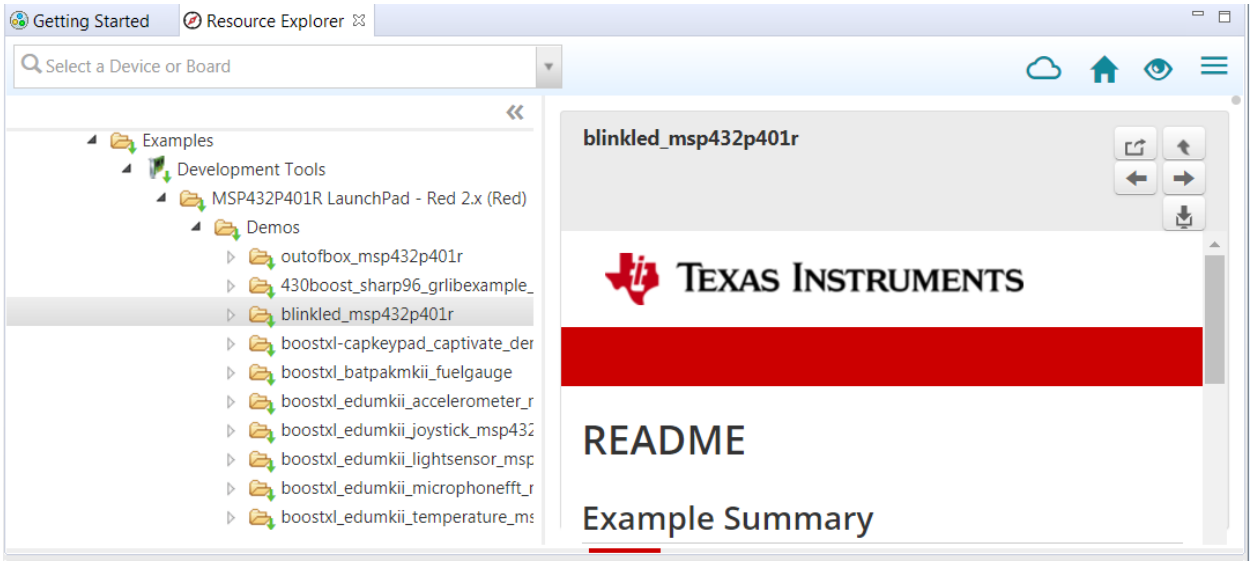


You will need to be connected to the Internet to see this. Now the only challenge with using the Resource Explorer is the large choice of possible examples. Let's get to those that apply to your board:

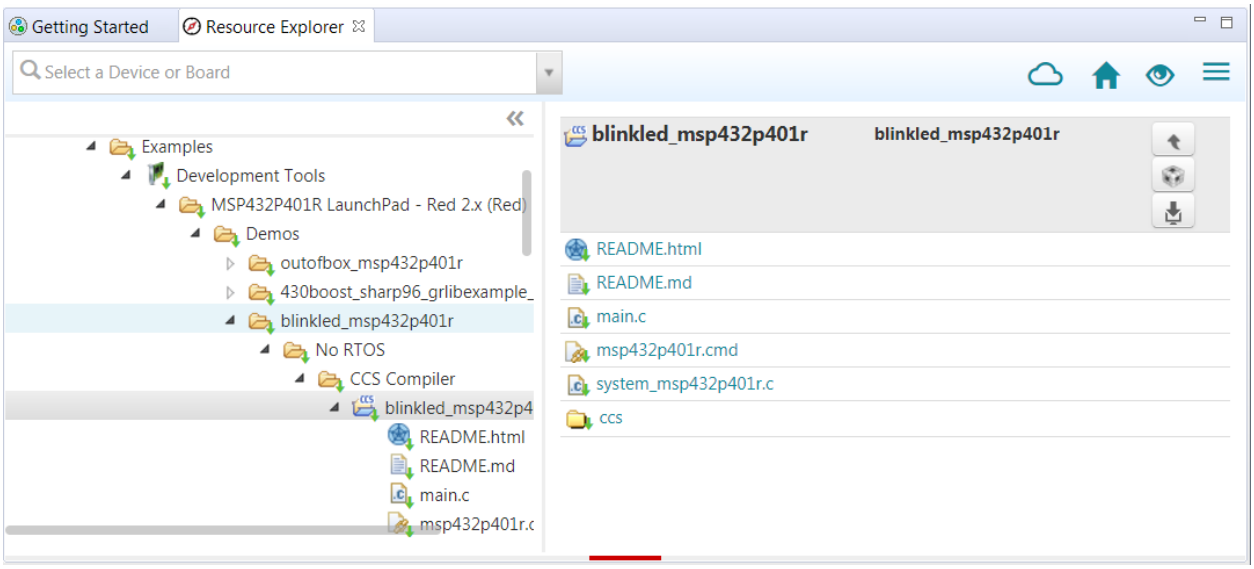


Expand the directories by clicking on the arrow at the left of each directory. You are now at the MSP432P401R LaunchPad – Red set. There are still many examples here, so it will take some understanding to figure out where to select your files from.

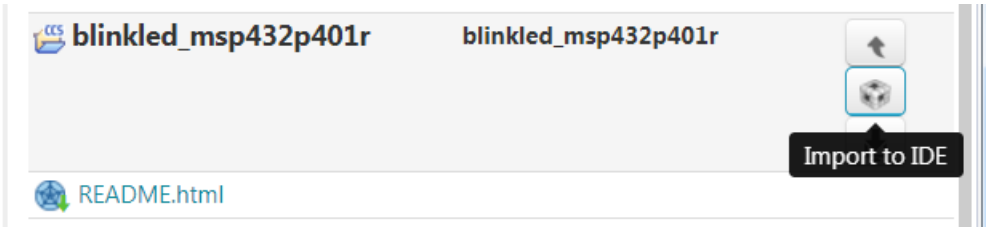
Each of these directories represent not only a set of example files, but also driver files that abstract, or hide, some of the low level hardware access via functions. In this example you'll focus on the Demos directory, so select that and you should see these choices:



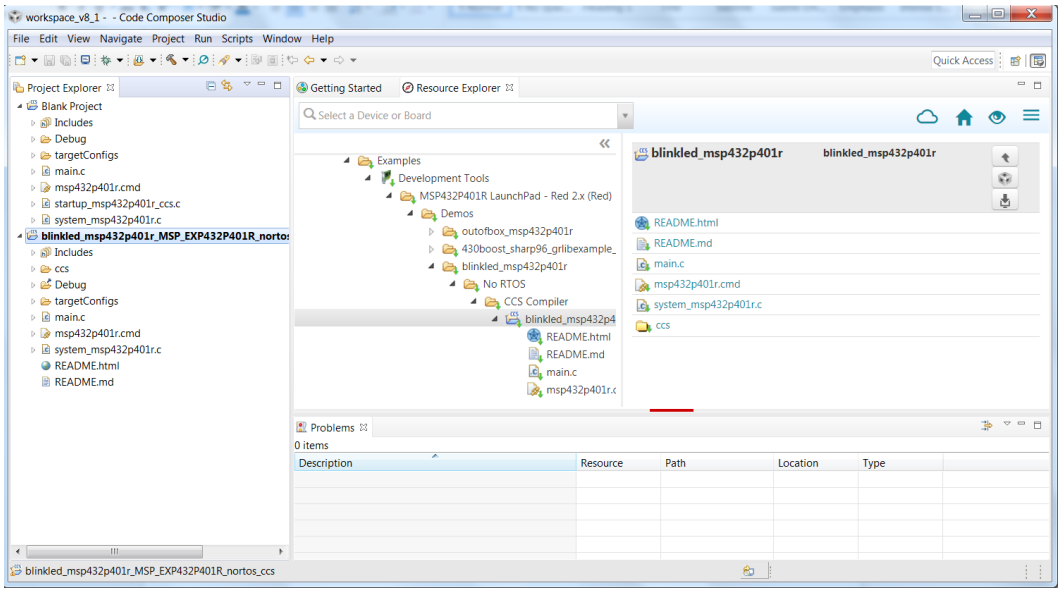
Each of these are a project, although some/many will need additional booster packs (hardware shields) to provide their functionality. Now select blinkled_msp432p401r->No RTOS->CCS Compiler->blinkled_msp432p401r from the list.



Now notice the selections at the upper right hand corner of the window:



You are going to select the middle one to import this project into the IDE. You should then see the project in your project window on the left (I've expanded the directory so that you can see all the files.)



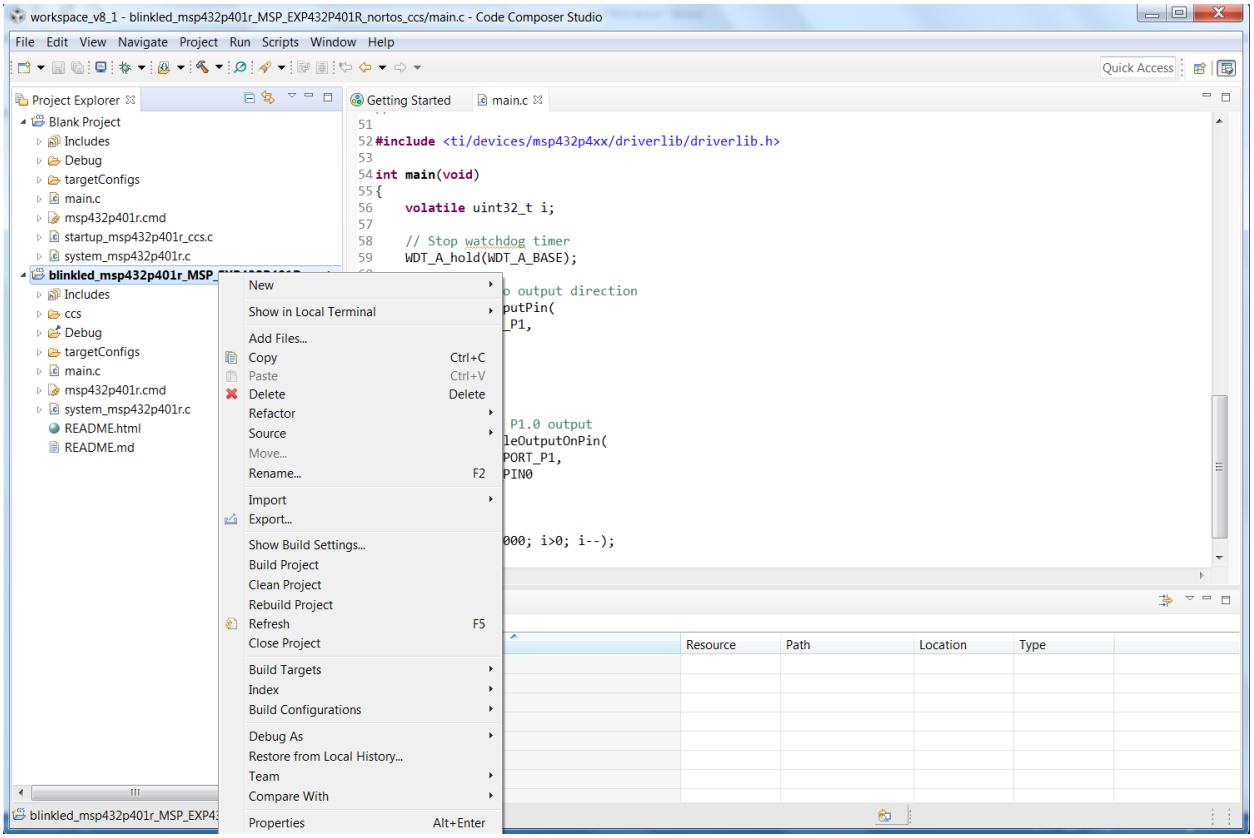
Now close the Resource tab in the main window, and double click on the main.c. You should see the code:

Getting Started | main.c

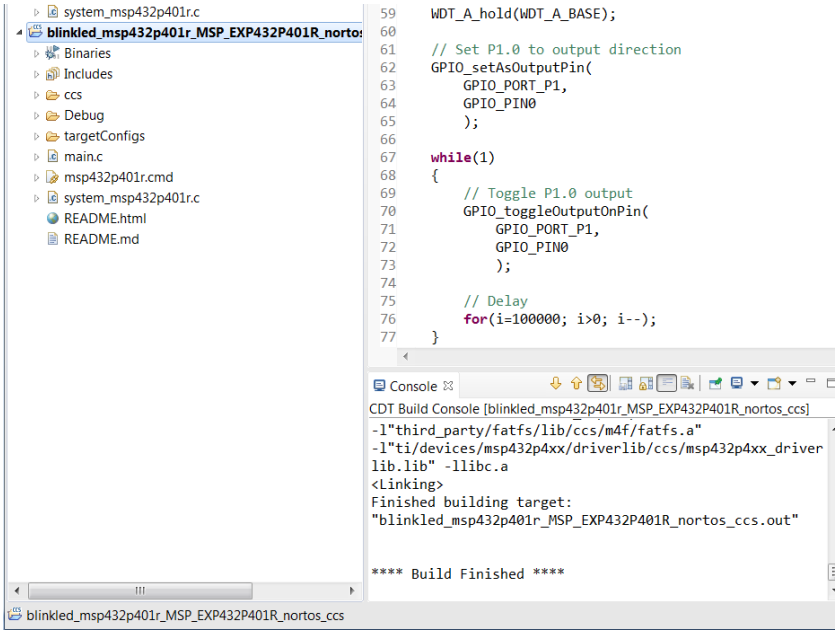
```
51
52 #include <ti/devices/msp432p4xx/driverlib/driverlib.h>
53
54 int main(void)
55 {
56     volatile uint32_t i;
57
58     // Stop watchdog timer
59     WDT_A_hold(WDT_A_BASE);
60
61     // Set P1.0 to output direction
62     GPIO_setAsOutputPin(
63         GPIO_PORT_P1,
64         GPIO_PIN0
65         );
66
67     while(1)
68     {
69         // Toggle P1.0 output
70         GPIO_toggleOutputOnPin(
71             GPIO_PORT_P1,
72             GPIO_PIN0
73             );
74
75         // Delay
76         for(i=100000; i>0; i--);
77     }
```

Now important to note is the inclusion of the library <ti/devices/msp432p4xx/driverlib/driverlib.h> . This provides some hardware abstraction (hiding of the low level hardware) by providing functions like GPIO-setAsOutputPin and GPIO_toggleOutputOnPin. You can call these functions without knowing the details of what is going on at the lower levels.

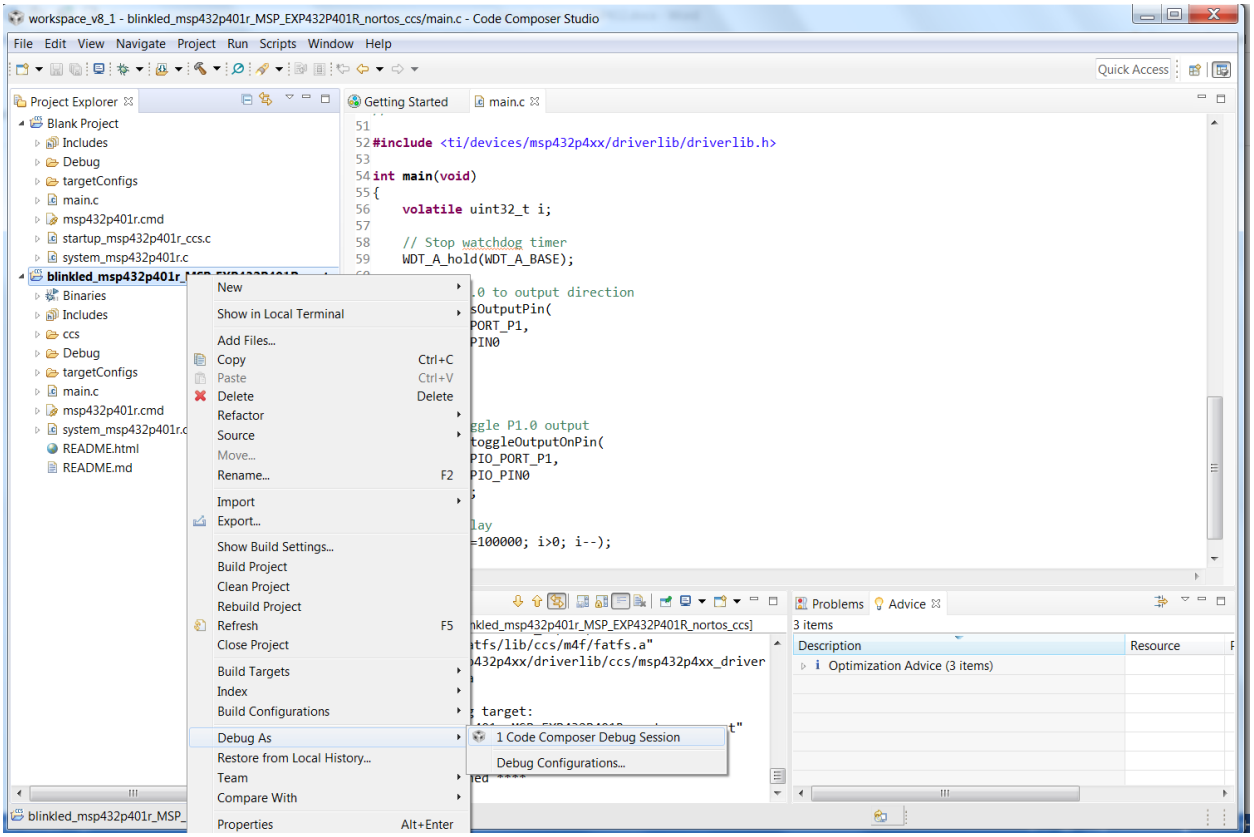
Now that you have this project, if you want to run it on your board you need to first compile it. To do this right click on the project name. You should see this selection:



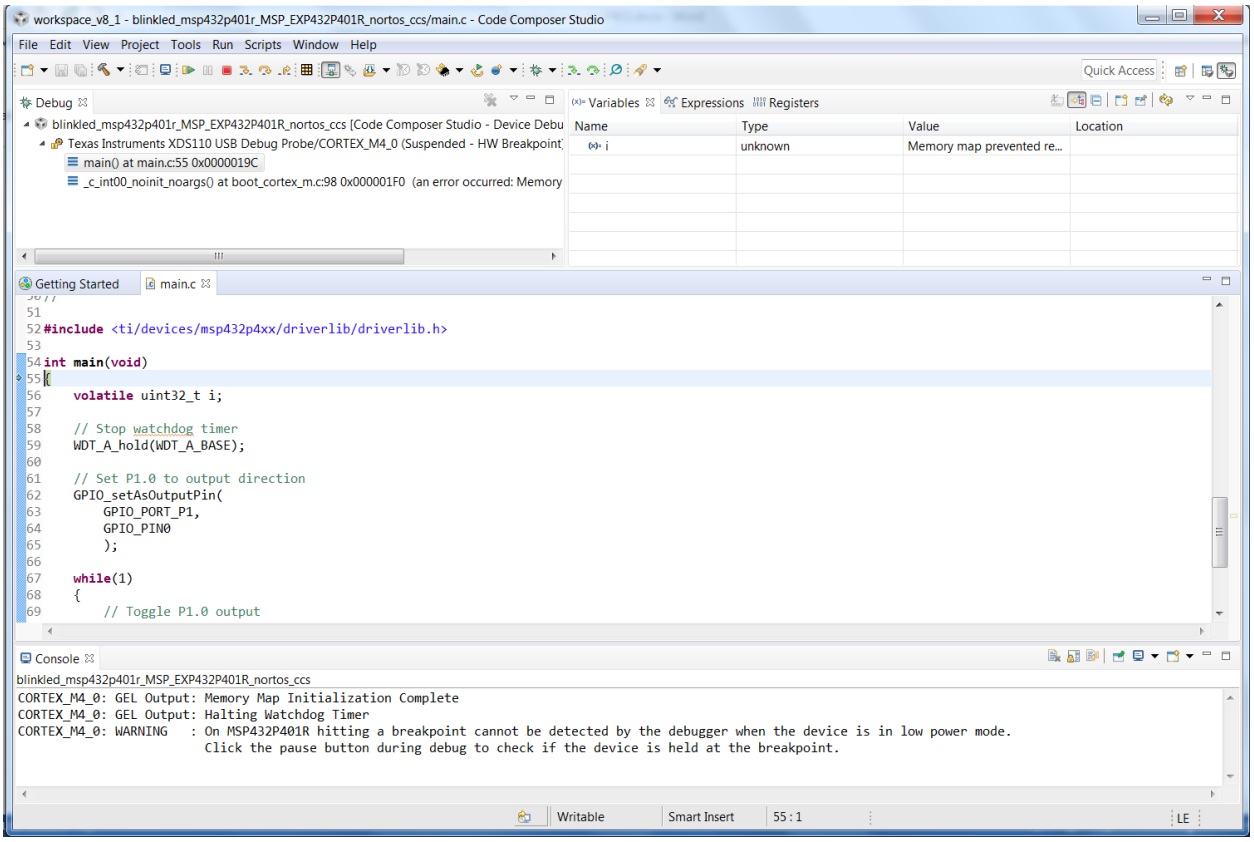
Now select the Build Project from this list. If everything went well you should see this:



In the CDT Build Console window you should see the output of the Build process. Now that the code is ready let's send it over to our board. To do this right click on the Project name again. This time choose the Debug As->Code Composer Debug Session like this:



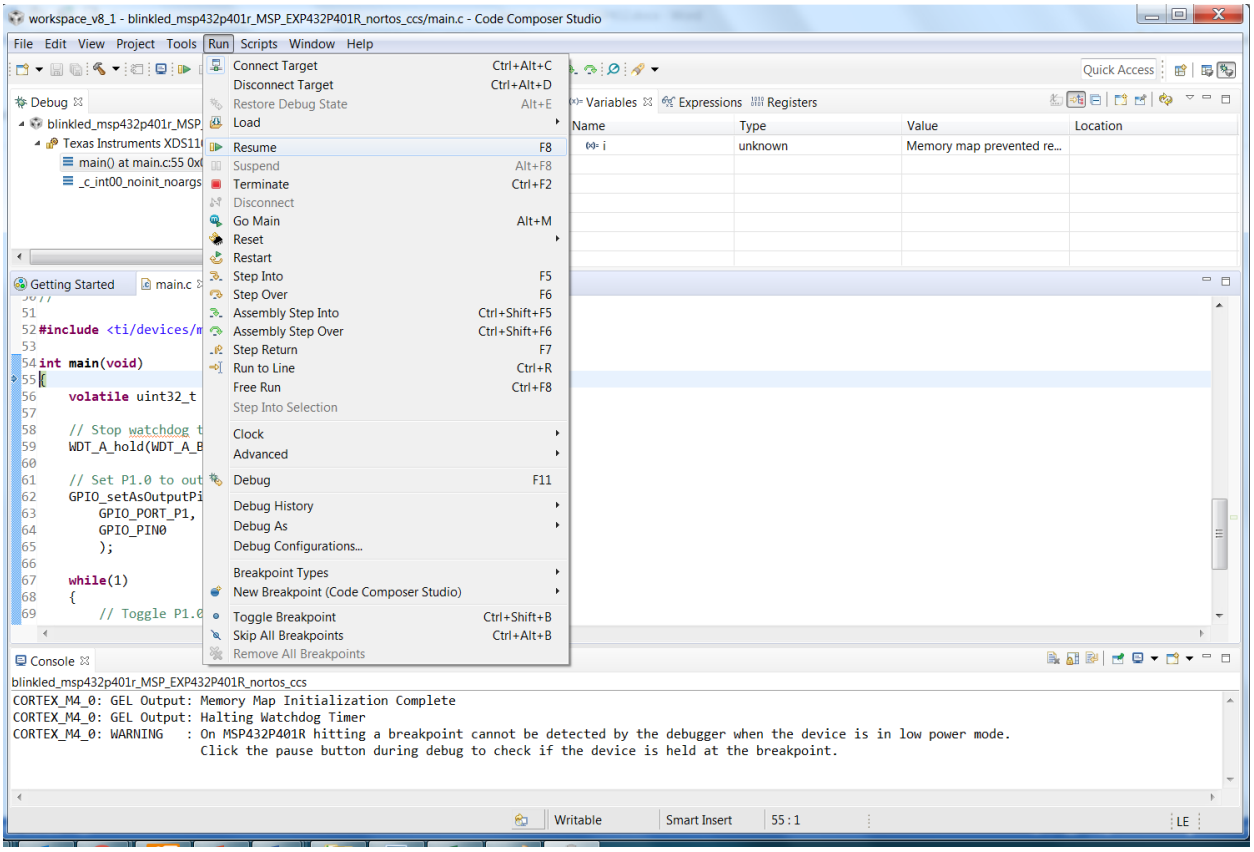
Then you should see this:



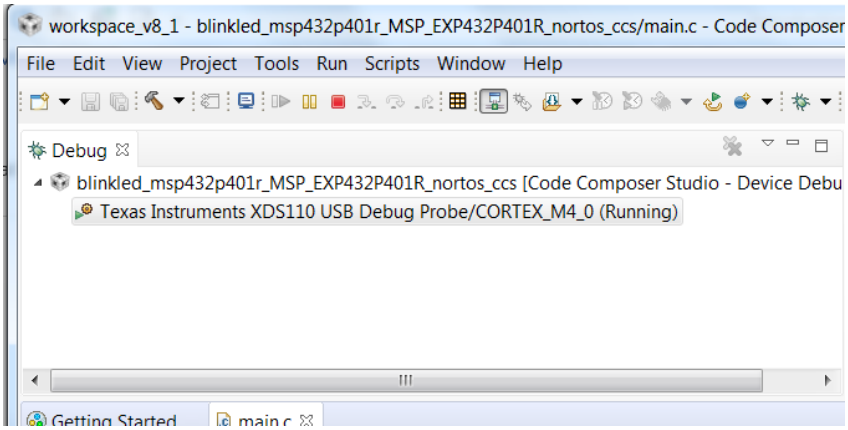
Notice that the entire view has changed. This is the debug Perspective. In the very top right hand of the IDE you can change between this and the CCS Edit perspective, which is where the IDE started.

Now let's look at the information provided by this perspective. In the middle is the main.c file, but notice that the line right after the int main(void) line is highlighted. Also note the Console output at the bottom. It simply tells us that the connection has been made and that the file is downloaded. At the top of the screen is the Debug Output, this tells you that the code is running on the MSP432Rp401R, and is in the main function.

The code is not running. It turns out that the default mode downloads the code, starts running, and then stop right after the main(void) function is hit. To continue running Run->Resume (there are shortcuts for all of this you can learn.)



Now the code should be running. The Debug window should now show:



And an LED should be blinking on the MSP432P401R. Hopefully you are now at least familiar with the Code Composer IDE.

There is another way to develop on the MSP432P401R that hides even more of the details. This one is inspired by the Arduino IDE and is called Energia.

Energia

In this section of the lab you'll install the Energia extension for Code Composer V8.

Installing the Software:

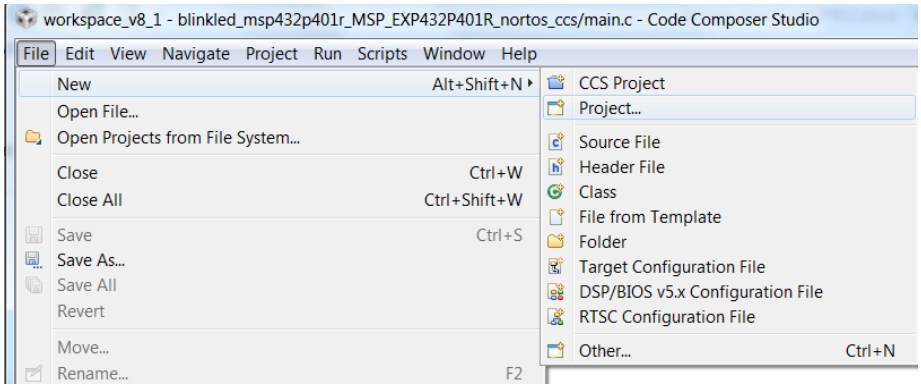
To Install the software go to the Energia web site at:

http://energia.nu/download/

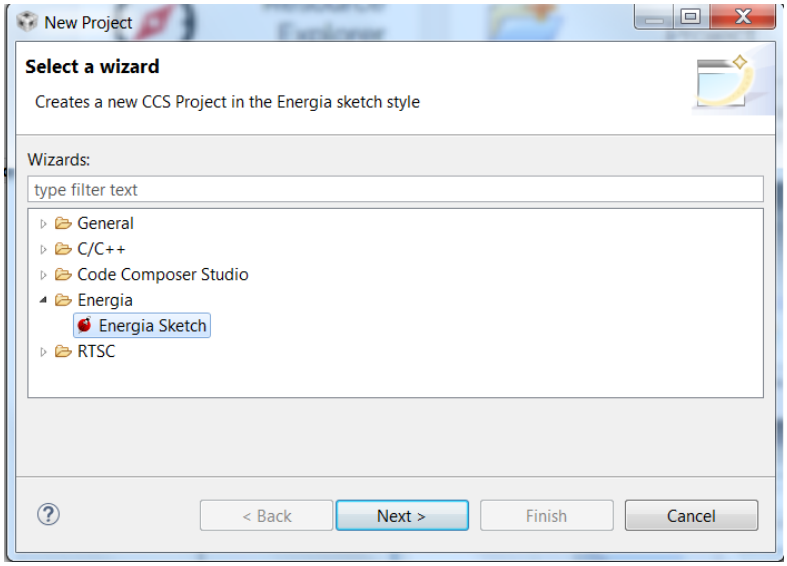
Follow the instructions for your operating system.

Creating your First Application:

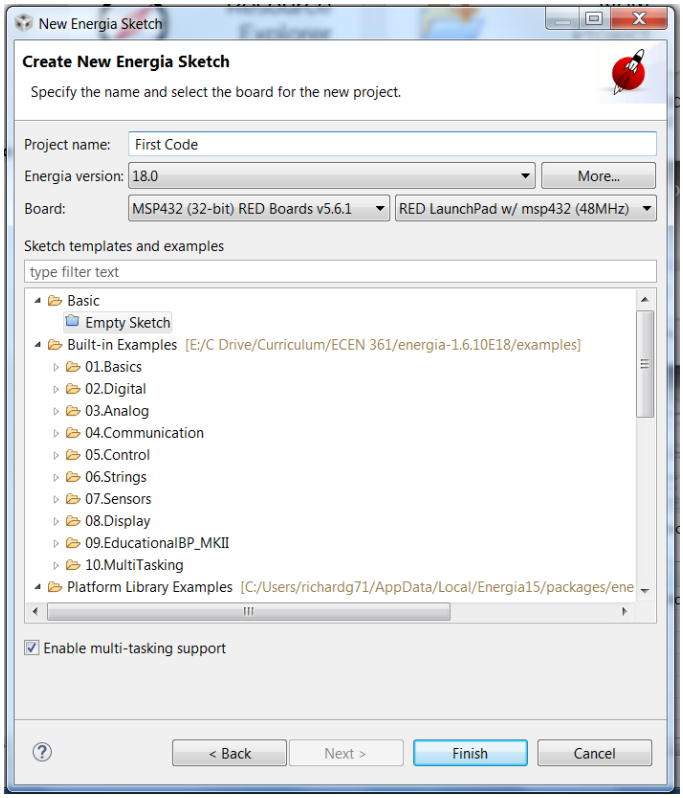
Now that you have Energia installed, let's create a project. To do this select New->Project in the CCS Edit perspective:



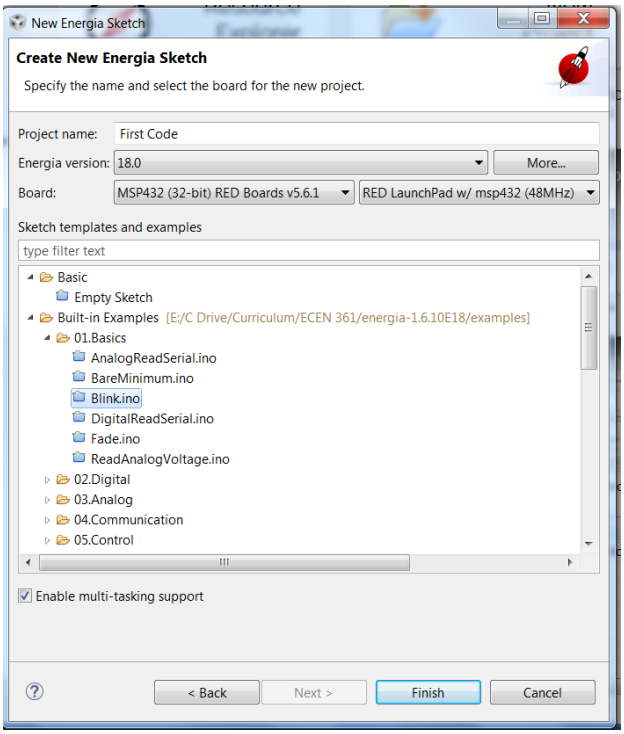
The following dialog box should open up:



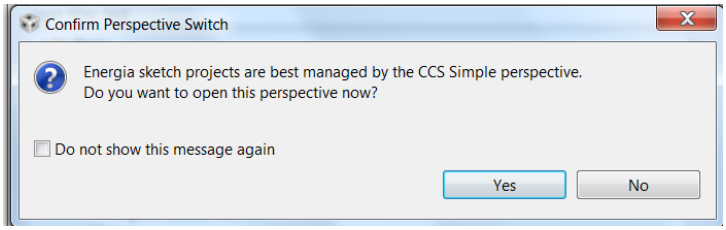
Right click on the arrow at the left of the Energia Directory, then Select the Energia Sketch, then select Next> and you should see this dialogue box:



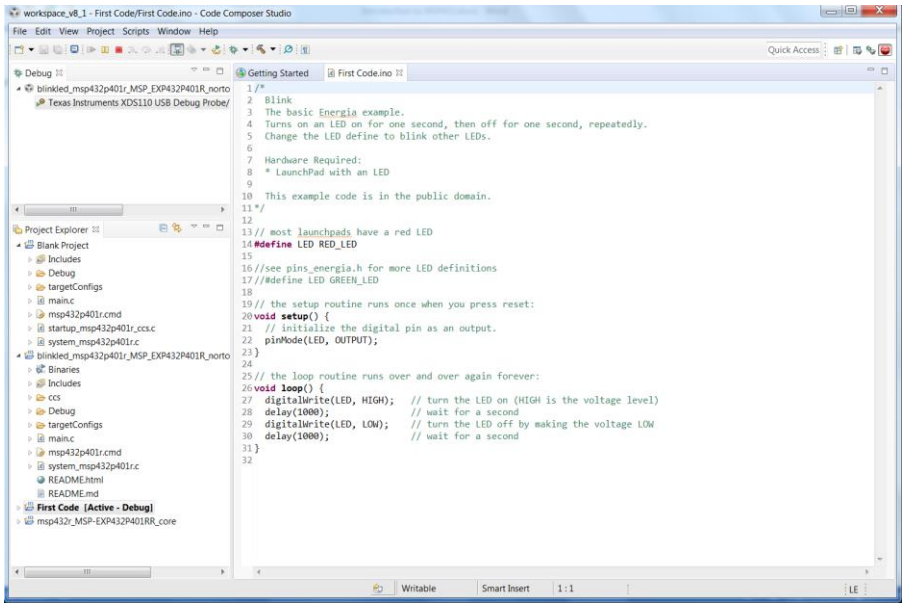
Enter a project name in the Project name: box. Now you'll notice that there is an Empty Sketch, but there are also many Built-in Examples. Let's assume you want to just blink an LED, so select the 01.Basics directory and then chose the Blink.ino example:



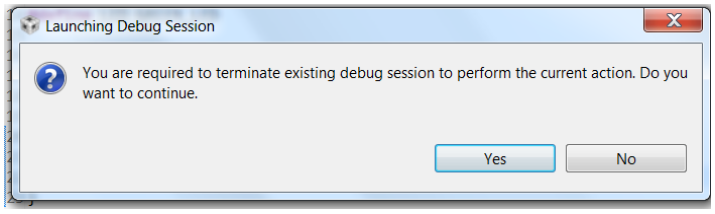
Then click Finish. You will see this dialogue box:



Click Yes. You'll now see your code in this perspective:



This looks a great deal like Arduino code. Now compile the code by the code by right clicking on the Project and choosing Build Project. Now download the code by right clicking on the project, selecting Debug As -> Code Composer Debug Session. You will see the following dialogue box:



This simply asks you if you want to stop the currently running program to download yours. Click Yes. Now you should see the RED LED blinking.

To change the file, go ahead and change the RED_LED in line 14 to GREEN_LED. Then rebuild the code by right clicking on the Project and choosing Build Project. Now download the code by right clicking on the project, selecting Debug As -> Code Composer Debug Session.

Now you should see the Green LED blinking.

Lab Results:

Submit the following on the project assignment on i-learn.

1) Submit a screen shot of your Blinking LED Code on Code Composer.

2) Submit a screen shot of your Energia Blinking LED Code.

3) Look in the Debug folder of your blinkled_msp432p401r_MSP_EXP432P401R_nortos_ccs project. Thinking back to CS 124, when you compiled your program you ended up with an a.out file that you could execute. Is there a file that is an .out file? How do you execute this file?