

In this first section of the lab you'll install Code Composer V10.

## 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:

[https://software-dl.ti.com/ccs/esd/documents/ccs\\_downloads.html](https://software-dl.ti.com/ccs/esd/documents/ccs_downloads.html)

The selections will look like this:

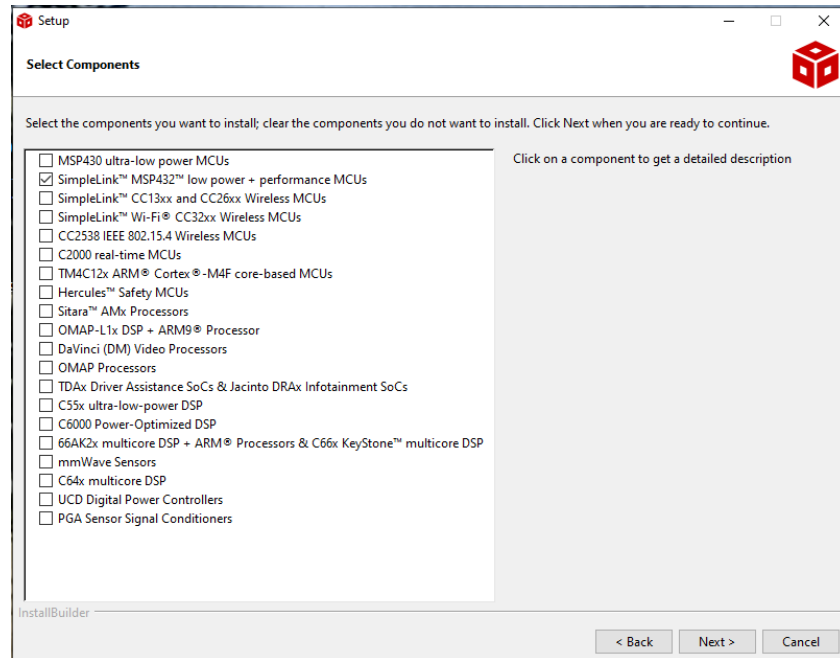
## Code Composer Studio Version 10 Downloads

There are two types of installers:

- On-demand installers allow you to download only the software components that you require. Formerly known as web installers.
- Single file installers will download a large compressed file (about 1GiB) so you may then uncompress it then select what you require to install. Formerly known as offline installers.

Release	Build #	Date	Download	Notes
10.0.0	10.0.00010	Mar 16, 2020	<b>Single file (offline) installers:</b> Windows 64-bit-only MD5 Linux MD5 - 64-bit only MacOS MD5 <b>On-demand (web) installers:</b> Windows 64-bit-only MD5 Linux MD5 - 64-bit only MacOS MD5  Manifest	<ul style="list-style-type: none"><li>• <b>New/Notable In This Release (10.0.0.00010):</b></li><li>• <a href="#">Release notes</a></li><li>• Windows support: Code Composer Studio is now supported only on 64bit Windows machines.</li><li>• Mac OS installers are distributed as signed and notarized disk image (DMG) files.</li><li>• Bugfixes for IDE, DVT, Debugger</li><li>• Energia 21 support</li><li>• CCS User's Guide</li><li>• MCU Compilers LTS 20.02.0</li><li>• C6000 Compiler 8.3.6</li><li>• MSP GCC v8.3.1.25</li><li>• SysConfig Core v1.4.0.1234</li><li>• XDCTools 3.61.0.16</li><li>• Device support updates<ul style="list-style-type: none"><li>◦ Automotive v1.1.8</li><li>◦ C2000 v6.1.0.0</li><li>◦ mmWave v1.6.6</li><li>◦ Sitara v1.5.1</li><li>◦ Shared support v1.1.6</li></ul></li><li>• TI EMU20_M02 v9.1.0.00001</li></ul>

I will generally choose the On-demand (web) installers, of the two offered here. As you install the software use the custom (default) selection, and when directed install only the MSP432 support:



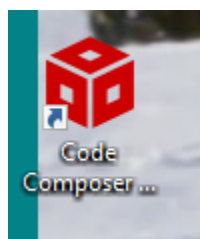
Here is a link to a detailed video on installing and initially using code composer:

<https://www.youtube.com/watch?v=qTpLYq-TPnA>

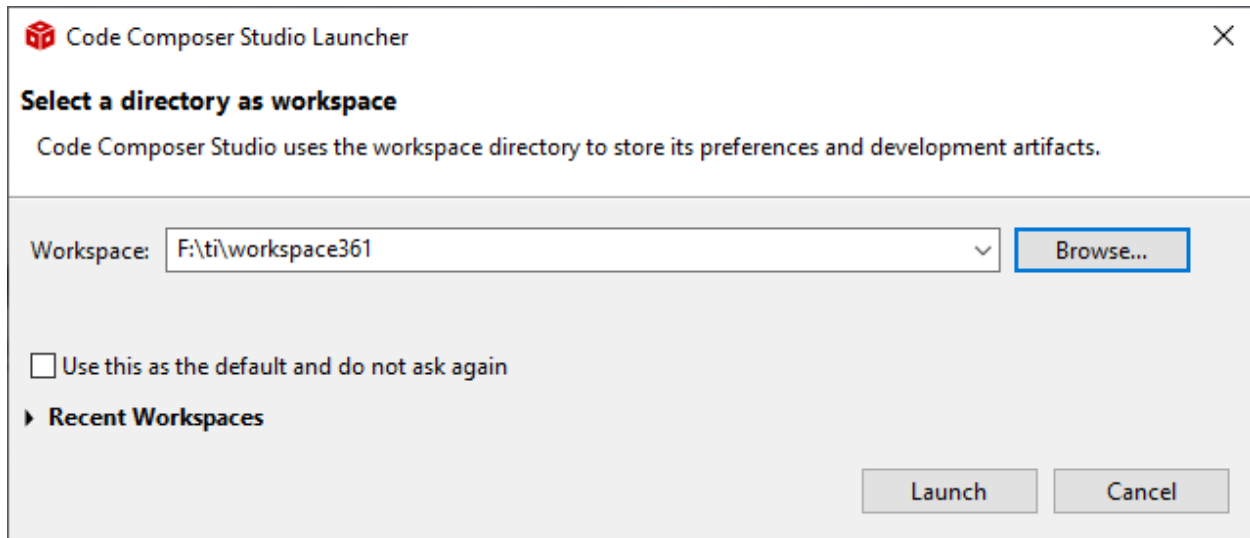
Once you've installed you can then create your first application.

### Creating your First Application:

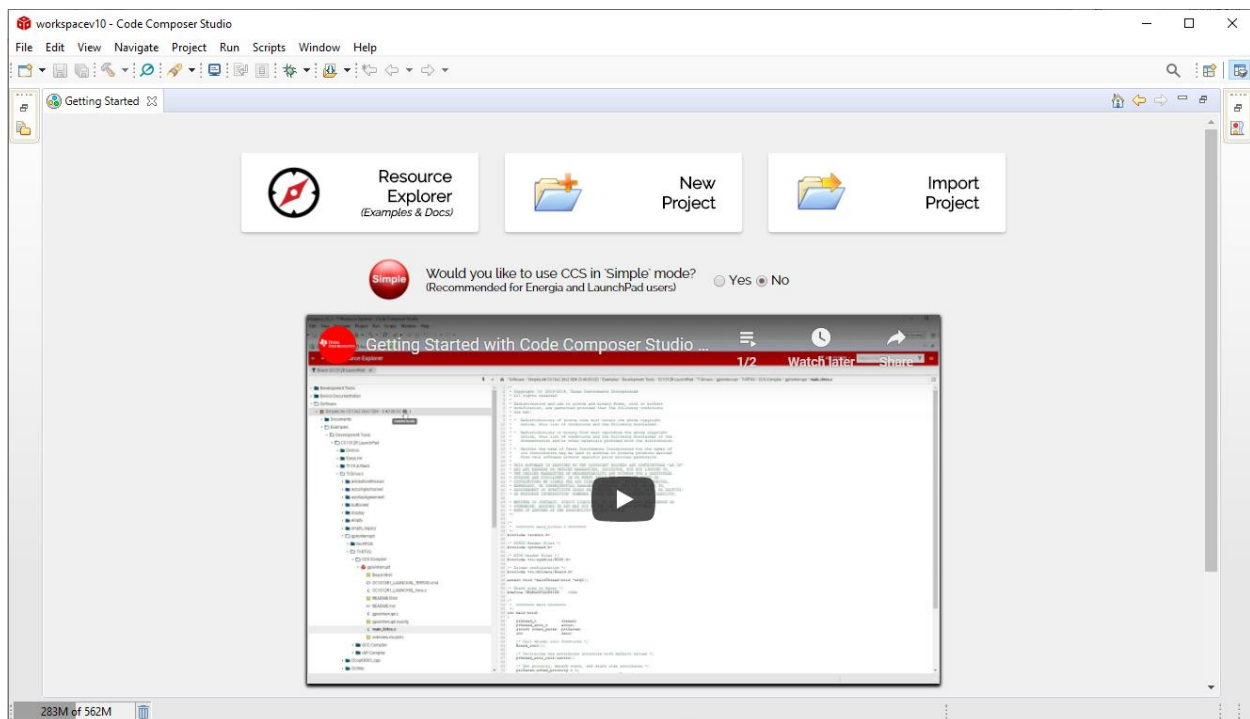
Now that you have access to the software, let's create your very first application. First, start the software by double clicking on the icon:



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



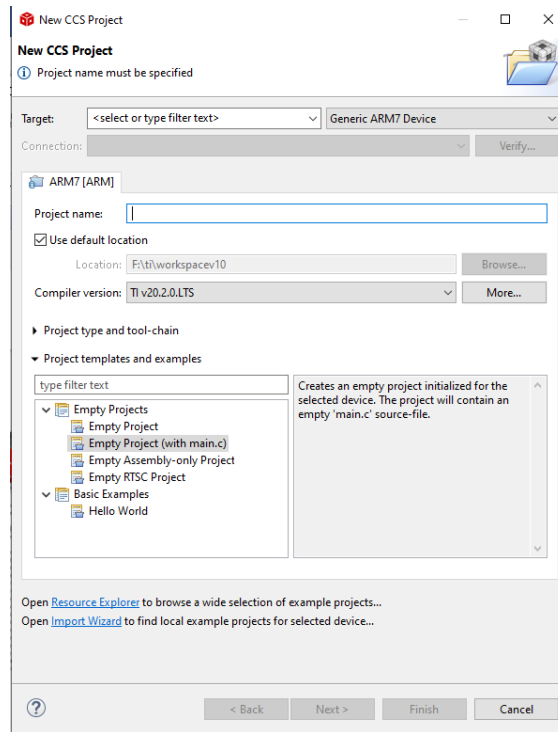
This workspace is the directory where you project files will be placed. You can use several different workspaces if you want to work on very different projects. Click Launch when you have specified the work space directory (you can just use the default.)



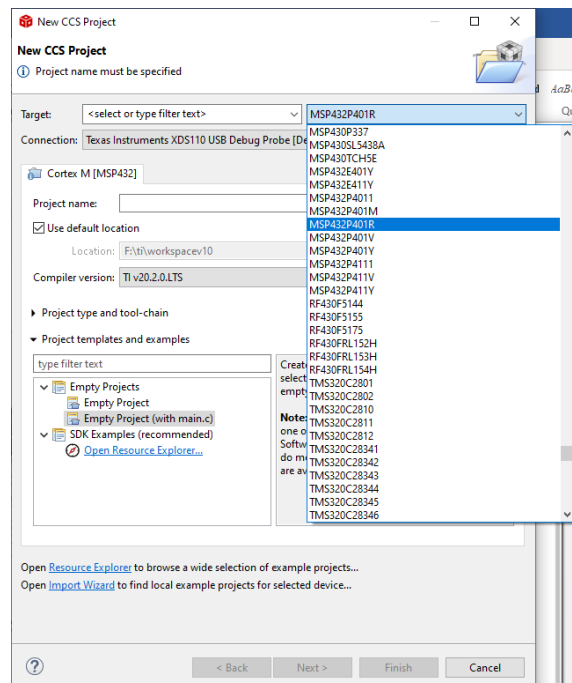
This is the introductory screen. Feel free to watch the Getting Started with Code Composer Studio v10.

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

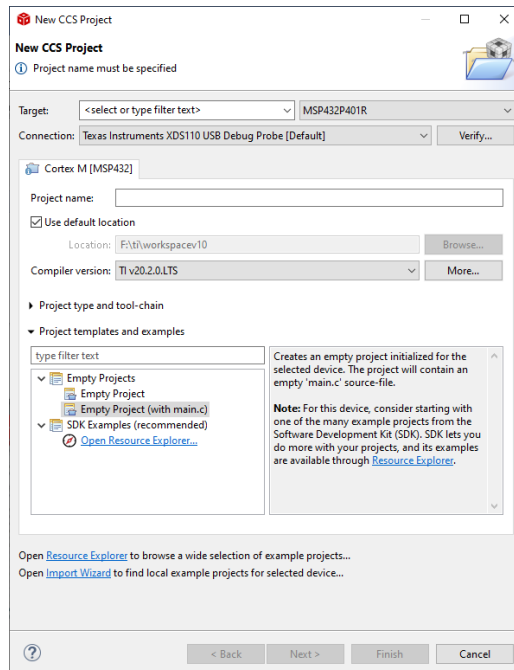
To start a new project press the new Project button at the top of the screen. You should now see this selection:



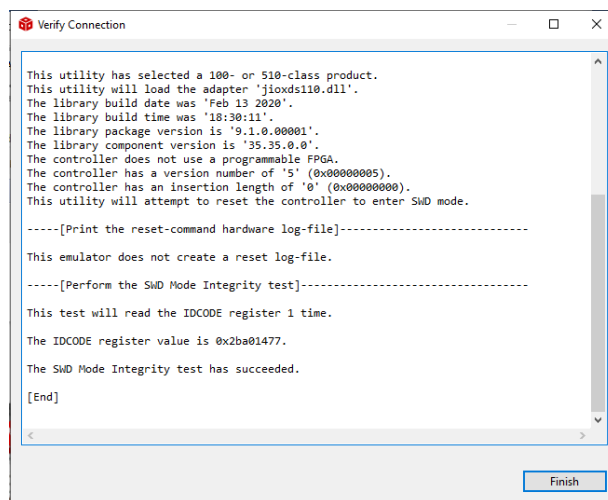
The first you'll want to do is the change the Target device. Under the Target: selection choose the MSP432P401R selection, like this:



You should now see this:



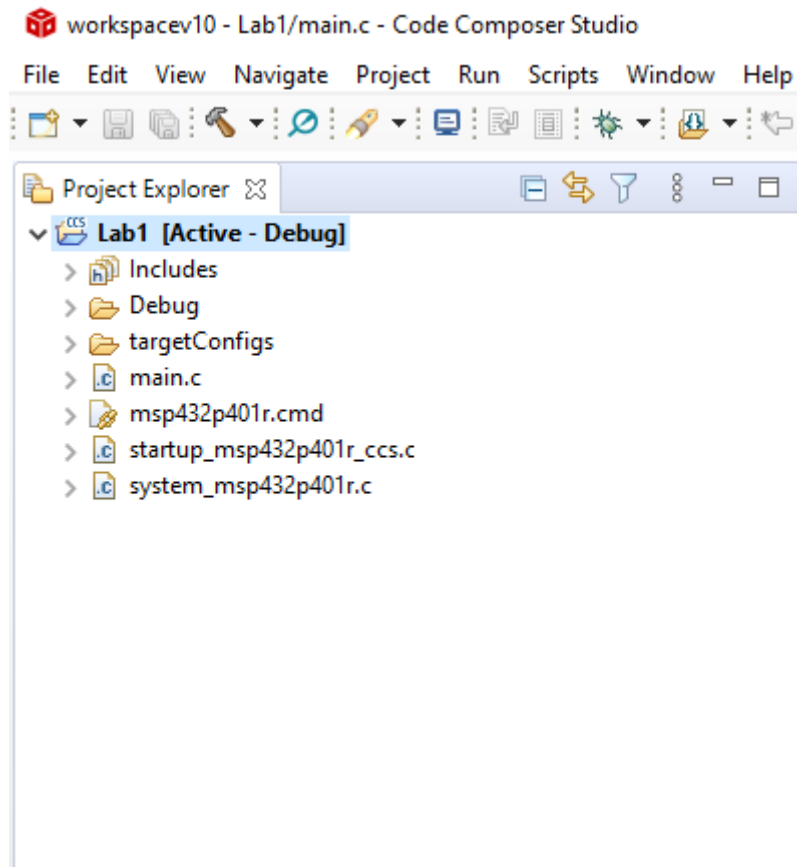
The Connection is showing the Default connection to this particular board, in this case it is a USB connection via the XDS110 interface (hardware on the board.) If you want to verify that Code Composer is connected to your board, then click Verify... and you should see this:



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.

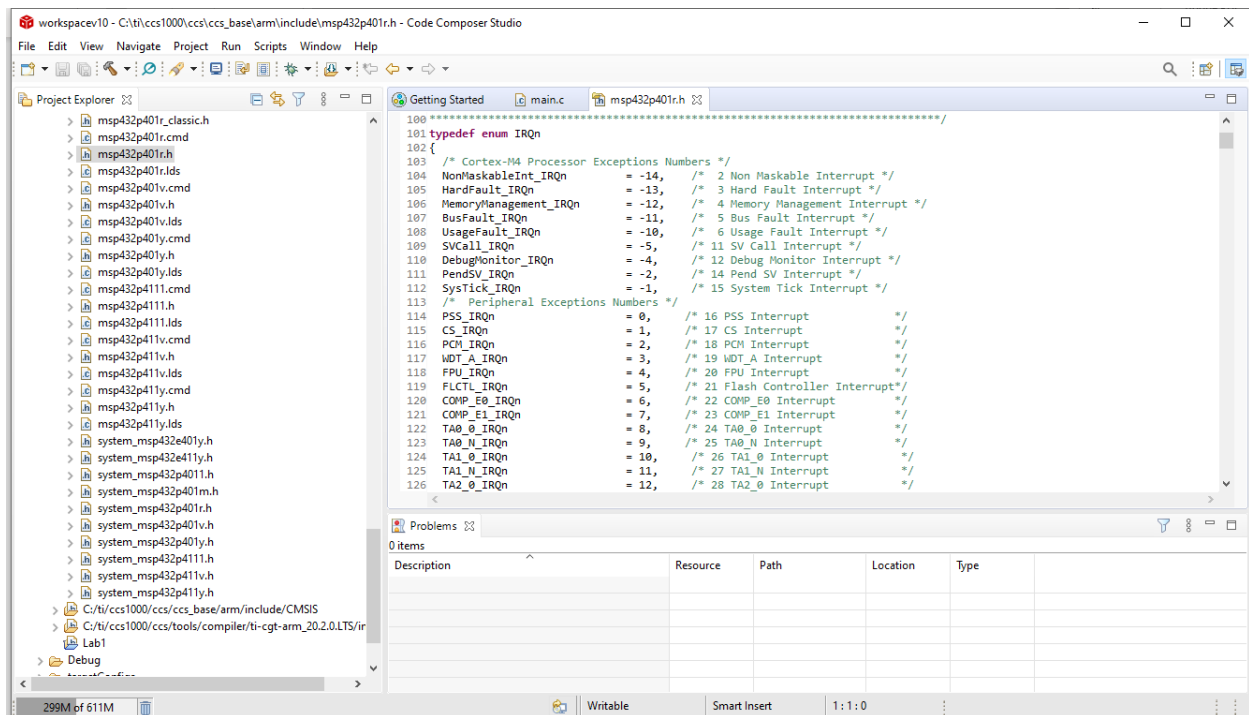




There are several directories and files that are created. Let's look at each briefly.

**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:



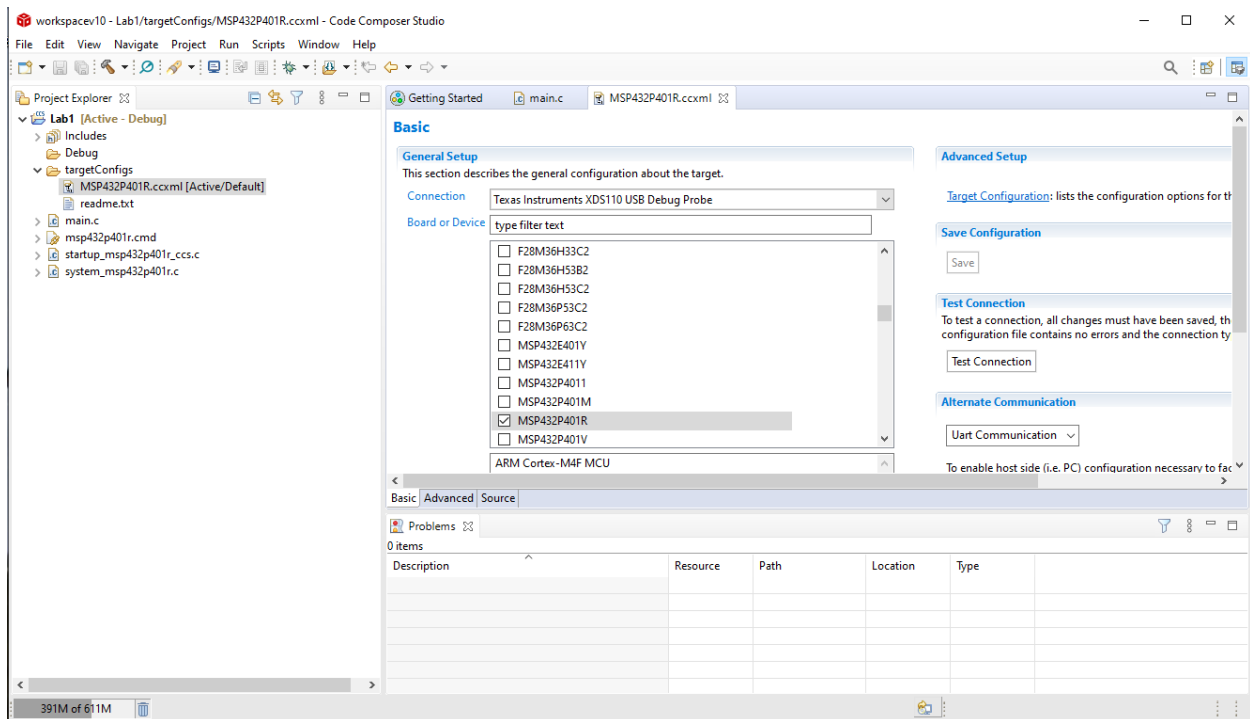


In this case I've opened the C:/ti/ccsv8/ccs\_base/arm/include directory by clicking on the arrow just to the left of it, then double clicking on the file msp432p401r.h file. This has a large set of enums (definitional structures that turn numbers into names in the code) that are declared to support 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 msp432p401r.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.

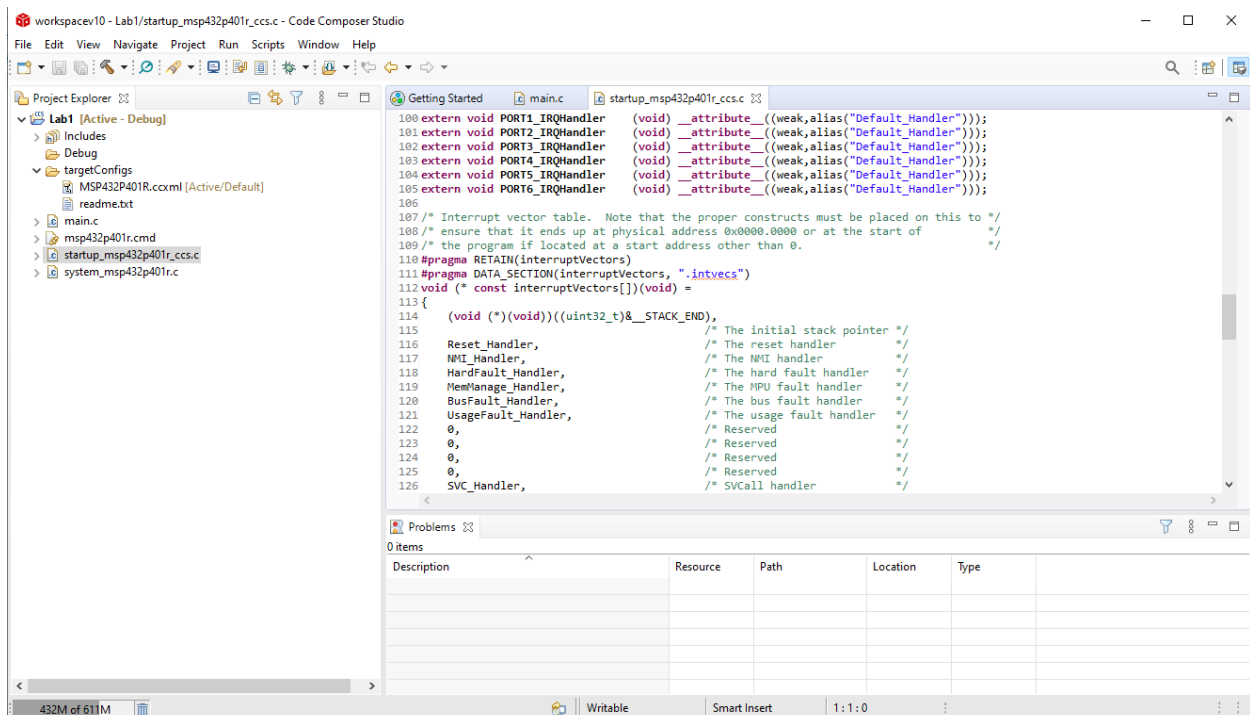
**Debug** – There is nothing in this file, as you haven't done any Debug. We'll cover this later.

**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.

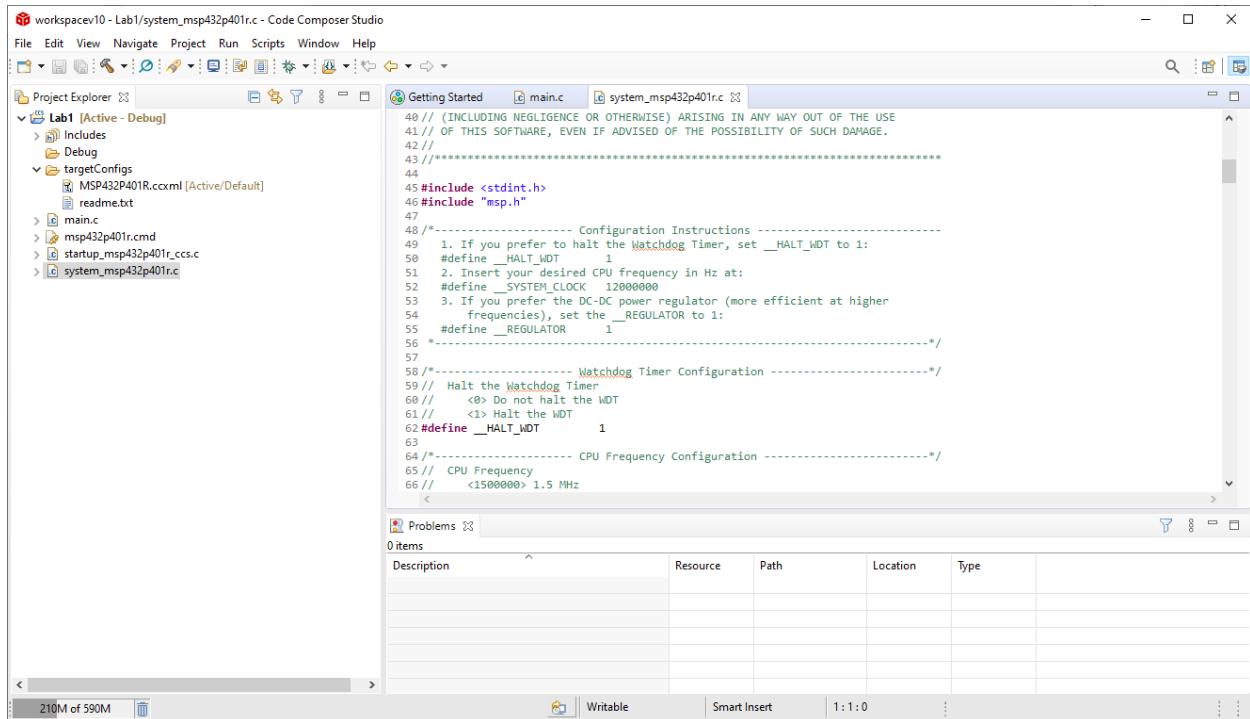


There are two other files in this directory that are code files. The main.c file, as you have already seen, holds the main() functions. Here are details about the other two files:

**startup\_msp432p401r\_ccs.c** – This holds the interrupt vector table. We will talk about this later in the class, but it specifies the functions to run when a specific interrupt occurs. Here is what that table looks like:



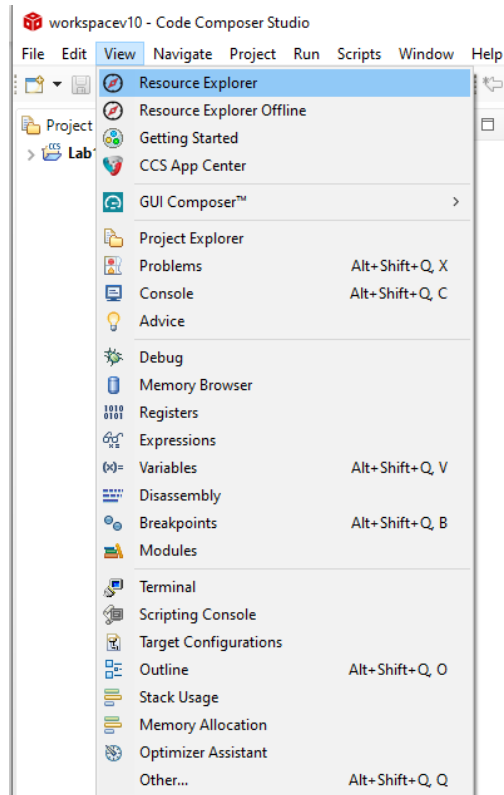
**system\_msp432p401r.c** – This file sets up all sorts of configuration details. Here is a view of bit of the file:



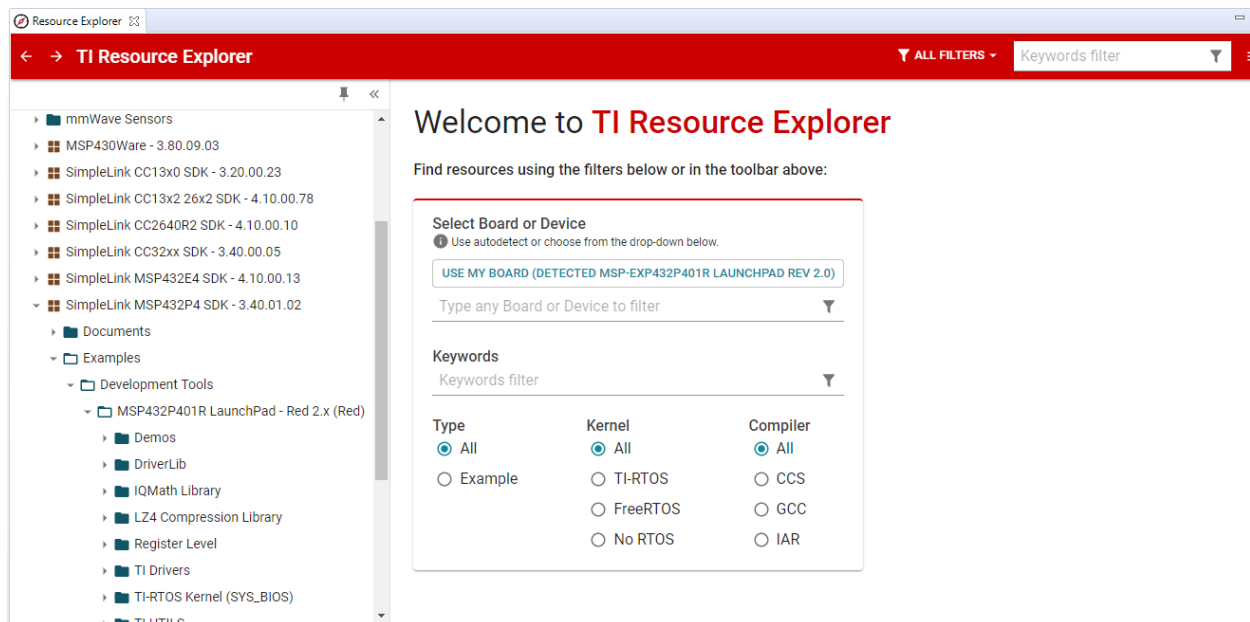
This files takes care of all the configuration details for you.

This is the start of a bare-bones project. But this time you'll start with a bit more framework. 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.)

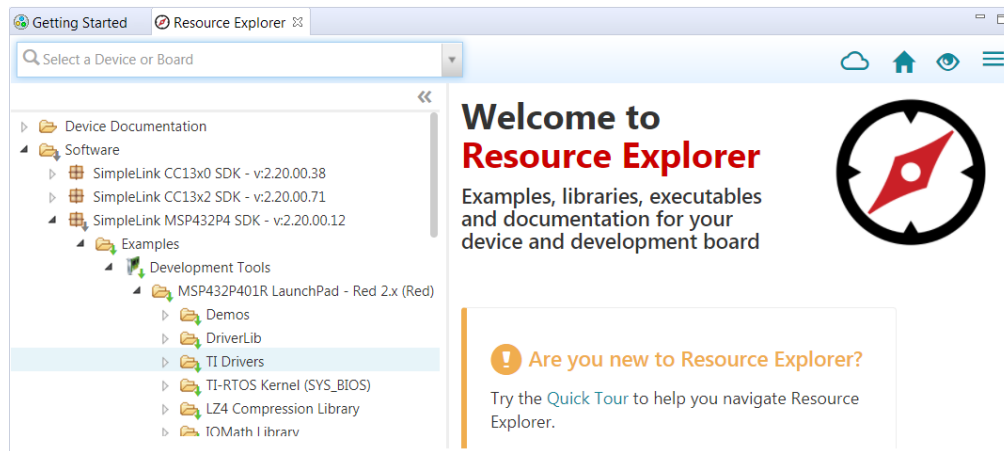
This time go to the View selection and chose the Resource Explorer...



The following should pop up:

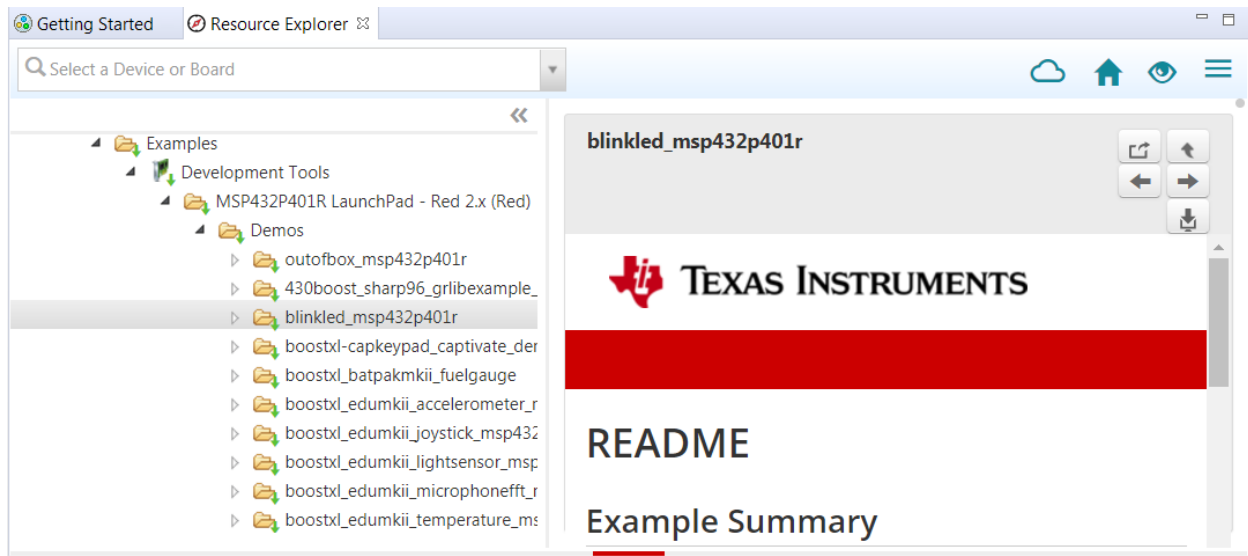


You will need to be connected to the Internet to see this. Now the only challenge with using the Resource Explorer is the large choice of possible examples. Let's get to those that apply to your board:

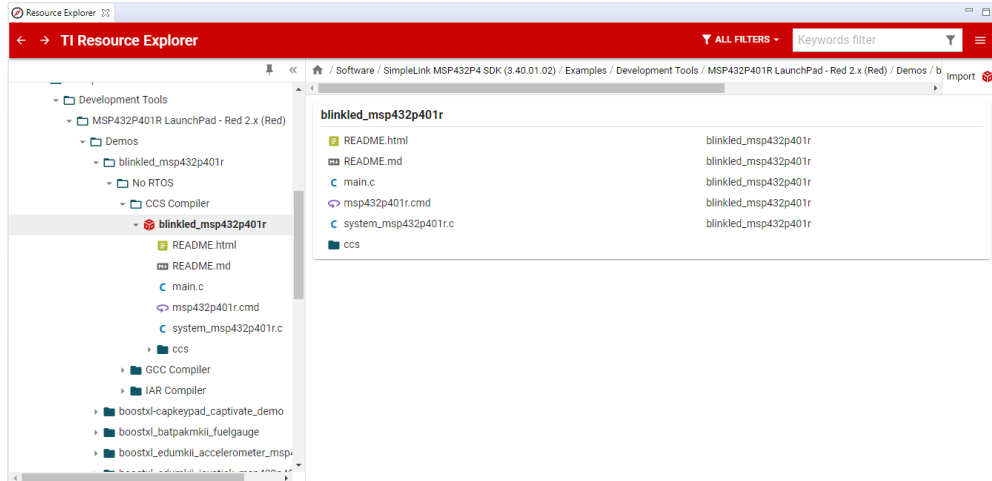


Expand the directories by clicking on the arrow at the left of each directory. You are now at the MSP432P401R LaunchPad – Red set. There are still many examples here, so it will take some understanding to figure out where to select your files from.

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:



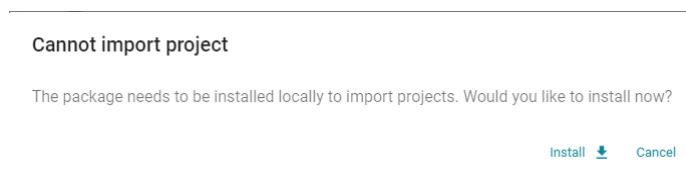
Each of these are a project, although some/many will need additional booster packs (hardware shields) to provide their functionality. Now select blinkled\_msp432p401r->No RTOS->CCS Compiler->blinkled\_msp432p401r from the list.



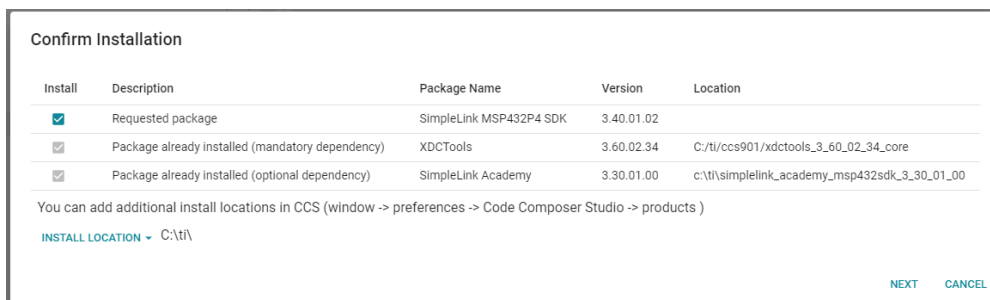
Now notice the selections at the upper right hand corner of the window:



You are going to select the import selection. You should see this selection:

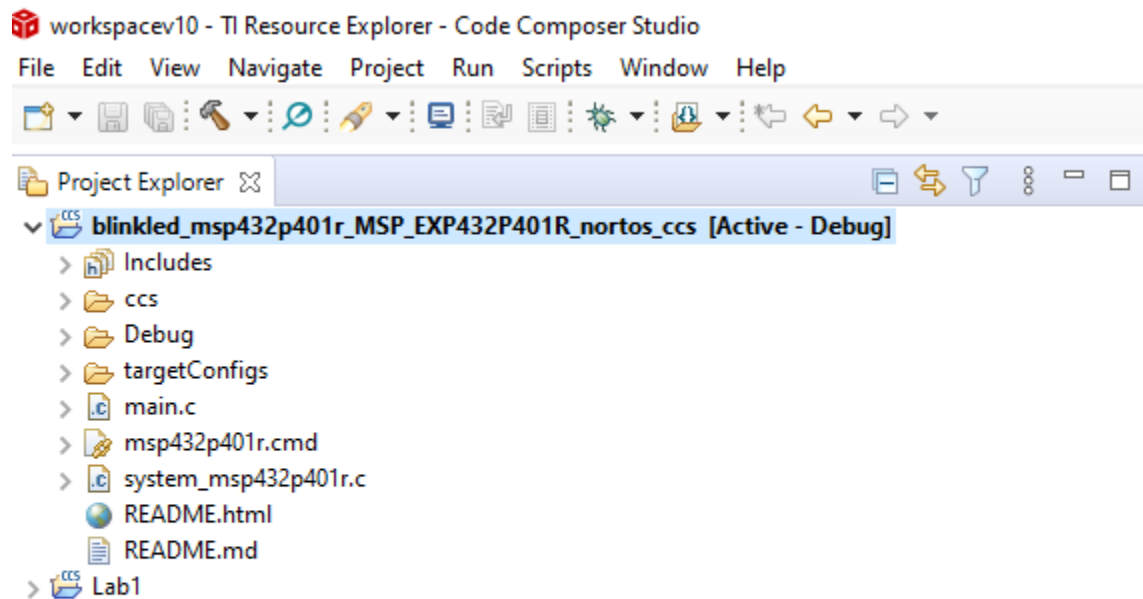


Now click Install. Then you should see this:



Click NEXT. Then OK on the next dialogue box. It might take some time, but you'll see the final dialogue box to confirm the import of the project.

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.)

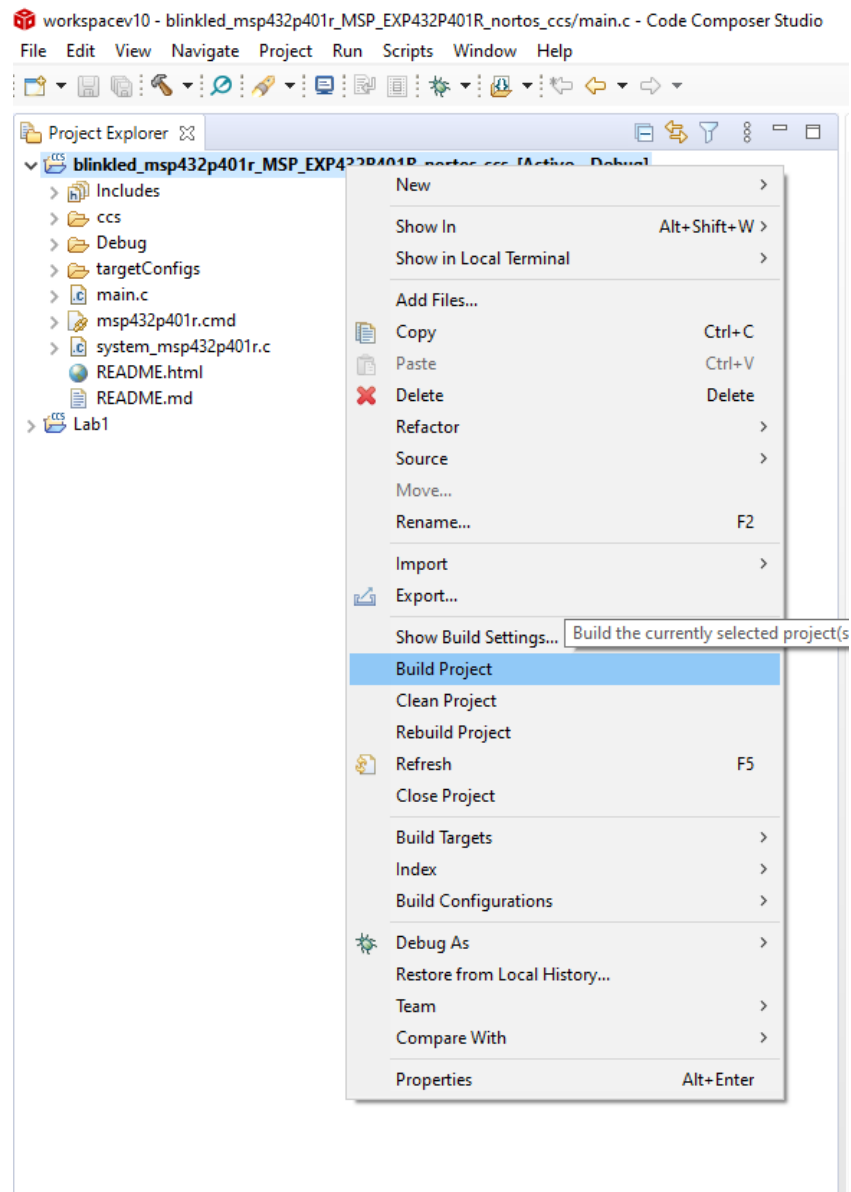


Now close the Resource Explorer tab in the main window, and double click on the main.c. You should see the code:

```
39 //
40 //      | | | XIN |
41 //      | | | RST |
42 //      | | | XOUT |
43 //      | | | P1.0 | LED
44 //
45 //
46 // E. Chen
47 // Texas Instruments, Inc
48 // March 2015
49 // Built with Code Composer Studio v6
50 //*****
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=1000000; i>0; i--);
77     }
78 }
79
```

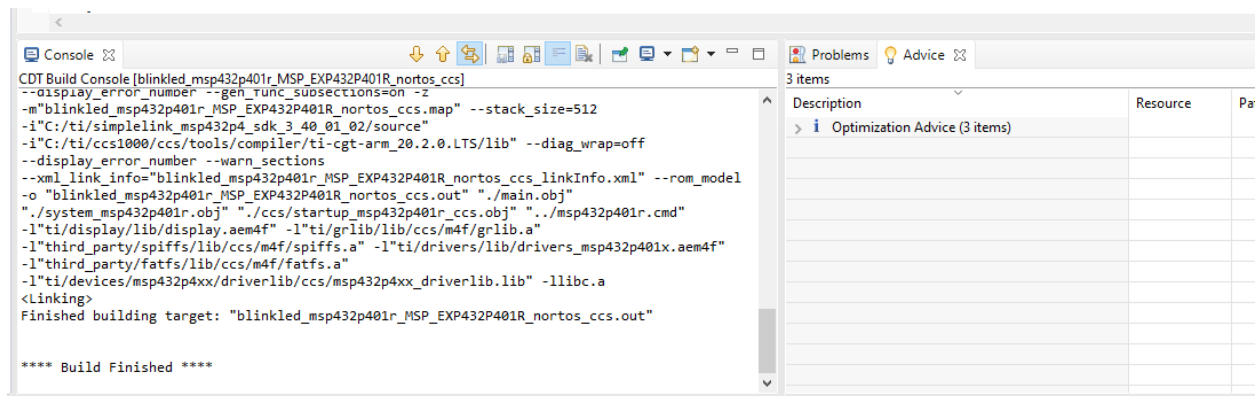
Now important to note is the inclusion of the library `<ti/devices/msp432p4xx/driverlib/driverlib.h>` . This provides some hardware abstraction (hiding of the low level hardware) by providing functions like `GPIO-setAsOutputPin` and `GPIO_toggleOutputOnPin`. You can call these functions without knowing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

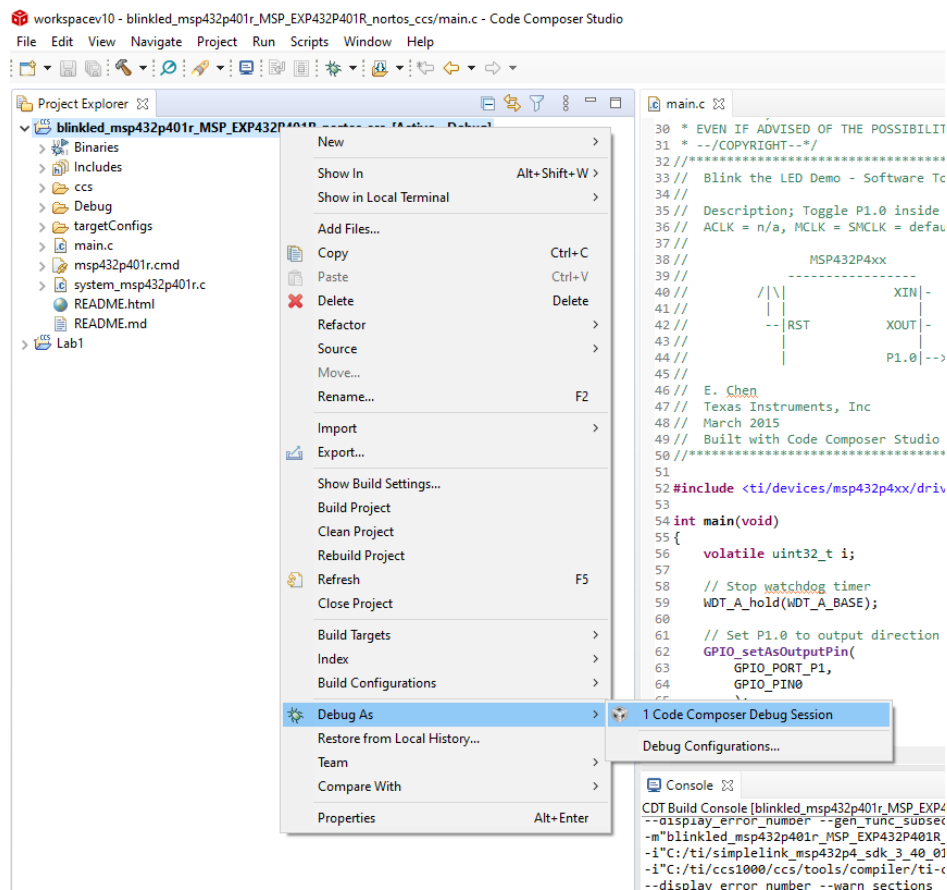


Now select the Build Project from this list. If everything went well you should see this:

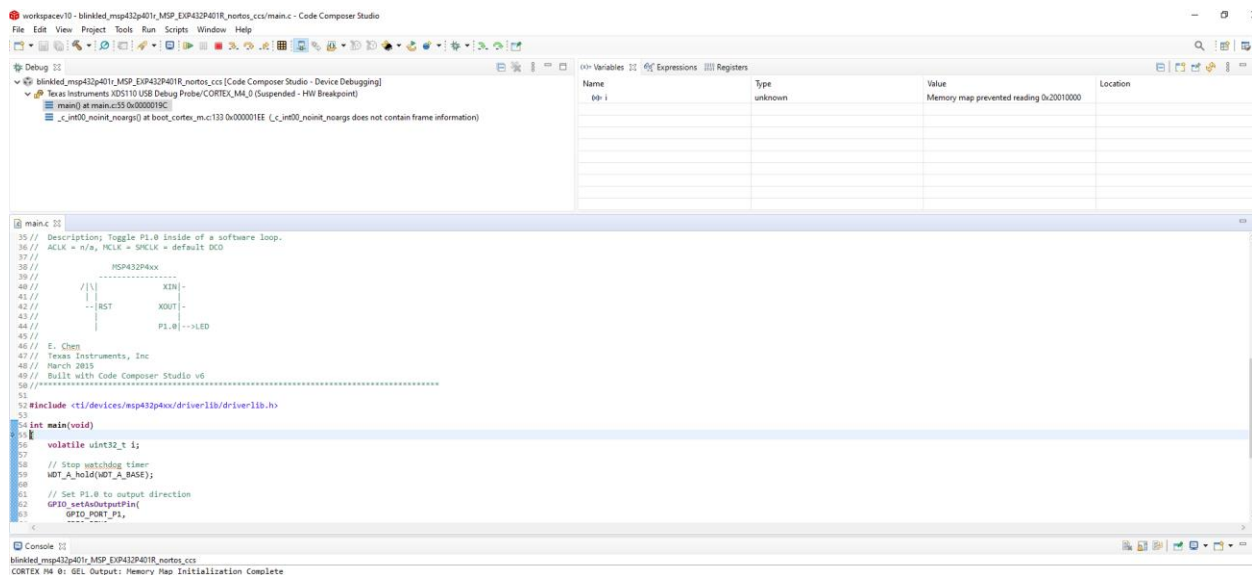




In the CDT Build Console window you should see the output of the Build process. Now that the code is ready let's send it over to our board. To do this right click on the Project name again. This time choose the Debug As->Code Composer Debug Session like this:



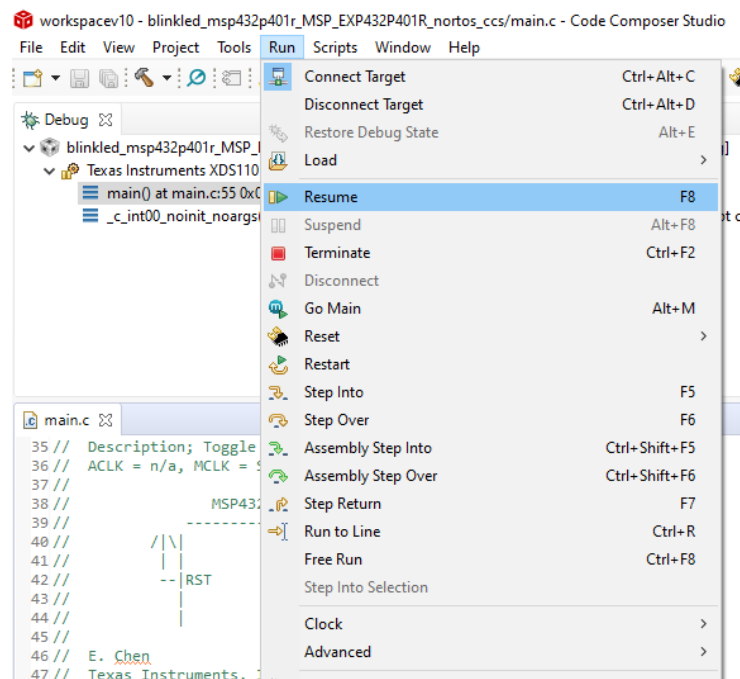
Then you should see this:



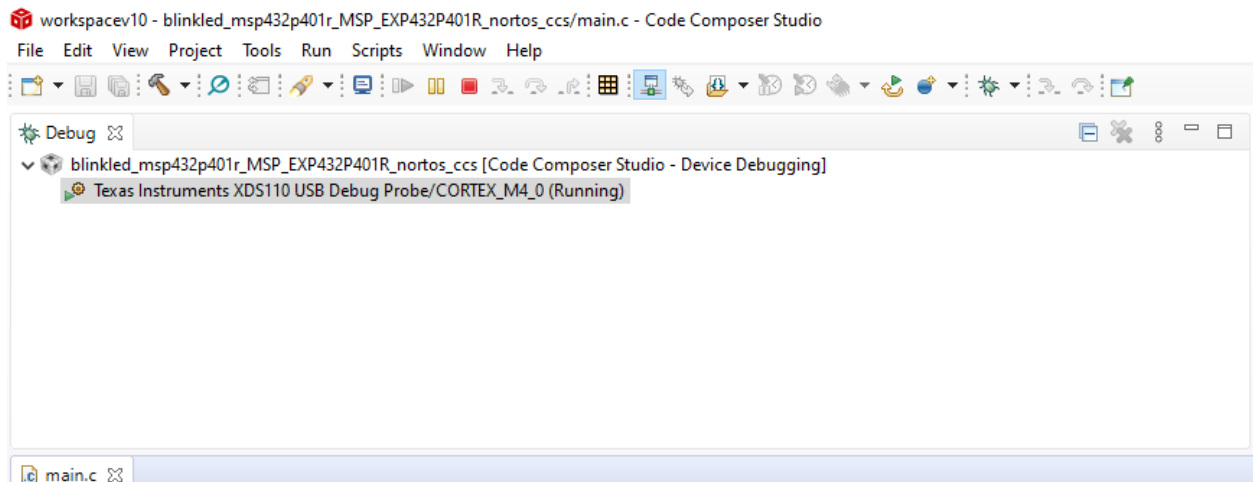
Notice that the entire view has changed. This is the debug Perspective. In the very top right hand of the IDE you can change between this and the CCS Edit perspective, which is where the IDE started.

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.