

Tutorial for NXP S32 Design Studio

Starting out - Creating a Workspace	1
Creating a project from existing project file	3
Building	5
Debugging	6
Terminal	9
Creating a new project	11
Enabling Floating Point Unit (FPU)	17

Starting out - Creating a Workspace

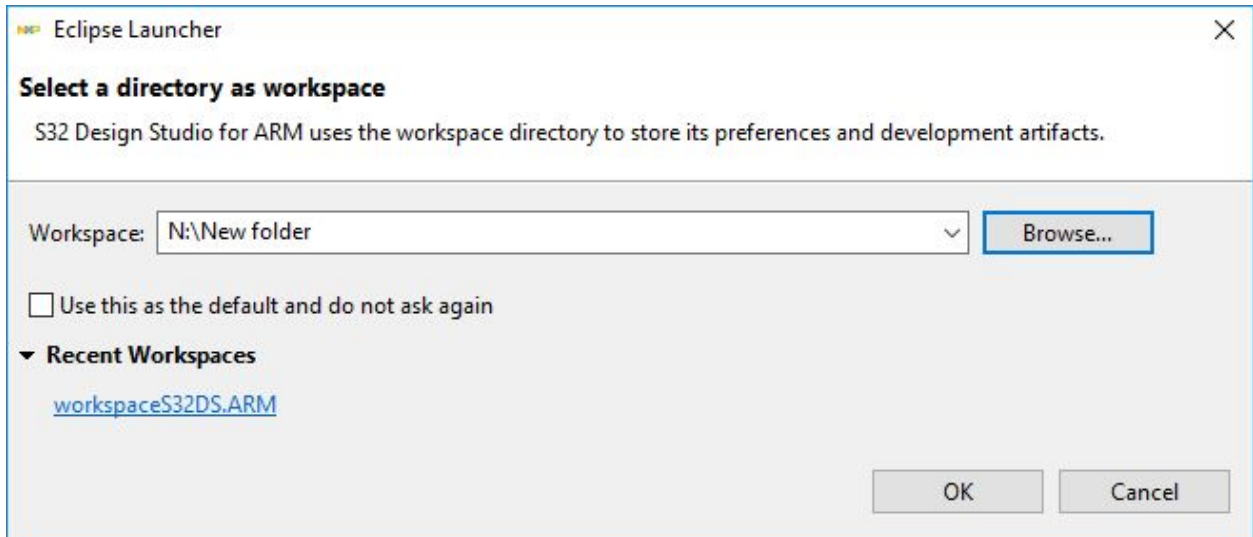
The purpose of this document is to get you familiar with the Integrated Development Environment (IDE) we will be using for lab known as NXP S32 Design Studio. To begin, click on the icon for “NXP s32 Design Studio for ARM v2.0”



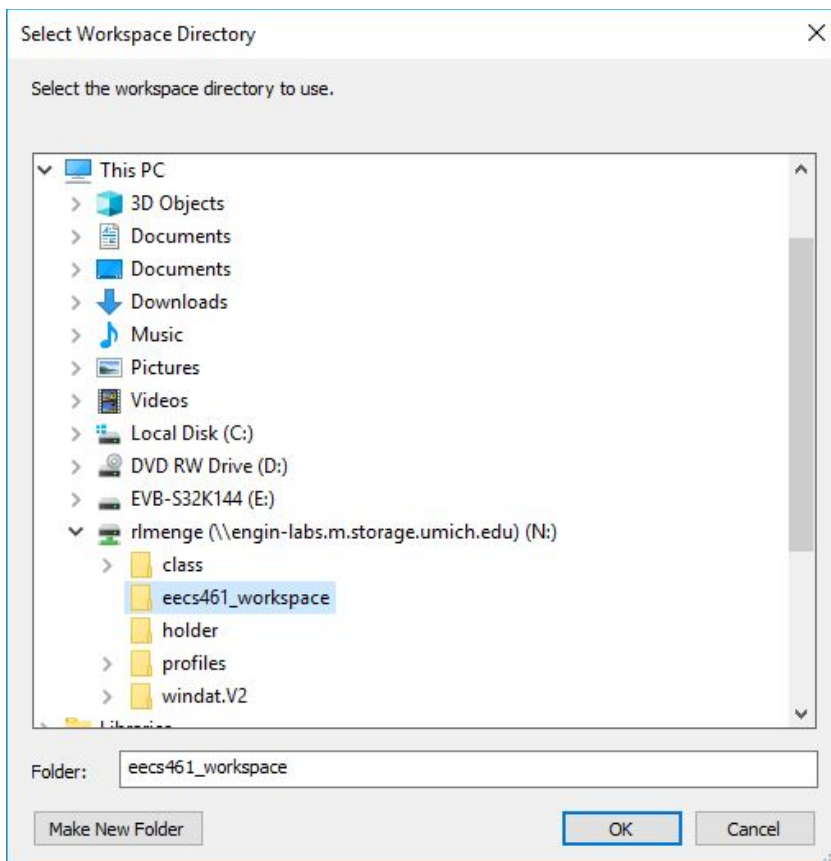
Icon for NXP s32 Design Studio for ARM v2.0

It will begin to load the eclipse environment and IDE we will be using to write and test code.

It will then ask you to choose a workspace to work in. A workspace is nothing more than a collection of projects. You should only need one workspace for this class.

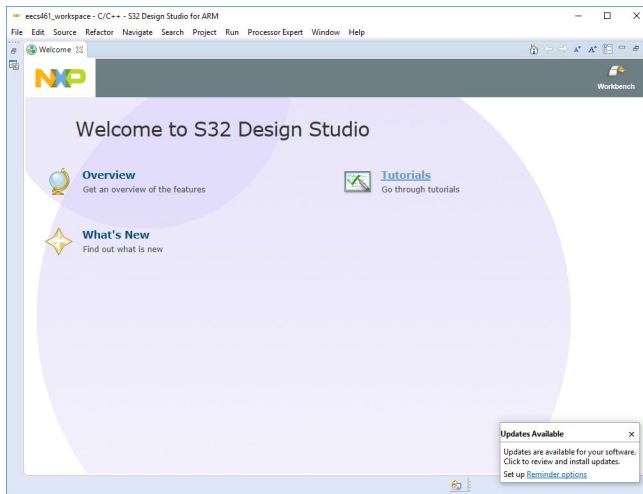


Click “Browse”. Navigate to your network drive. Your network drive (N:) is listed under “This PC”. Make a new folder in your network drive called “eecs461_workspace” using the “Make new folder” button as shown below.



Click your newly created folder and press “OK”.

It will now load the welcome page.



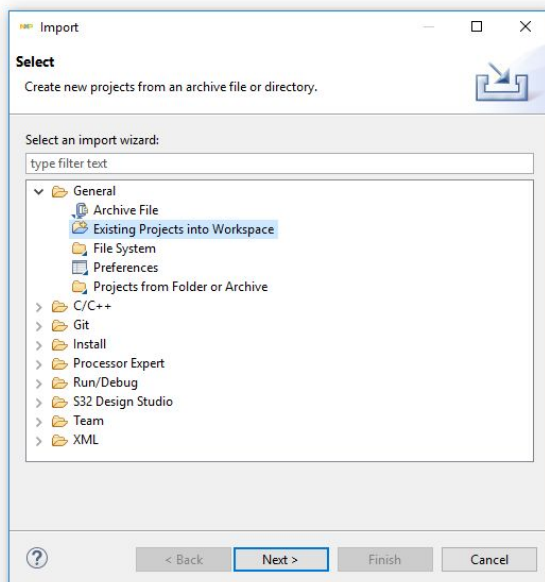
From here you can create a new project or open an existing project.

Creating a project from existing project file

(If you are jumping into this point of the tutorial launch NXP and choose the `eecs461_workspace` described above)

Download the zip file containing the project files from Canvas and extract it to any location (in this tutorial it is extracted in the downloads folder)

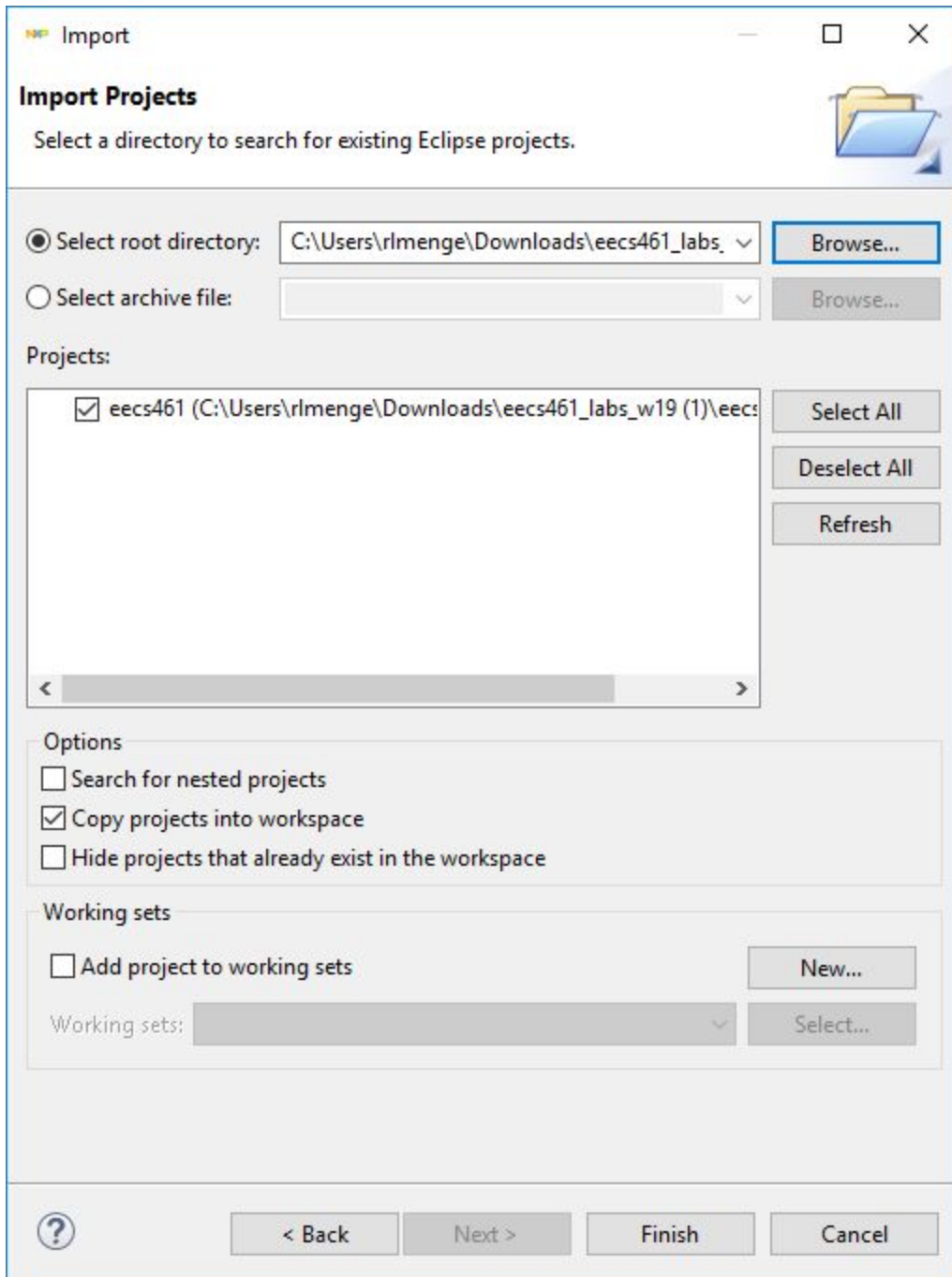
In NXP click "File" > "Import" > "Existing Project into Workspace" and double click



Choose “Select root directory” and browse to select the extracted zip file from your downloads
Select the project in the “Projects” window

Make sure the “Copy projects into workspace” box is checked

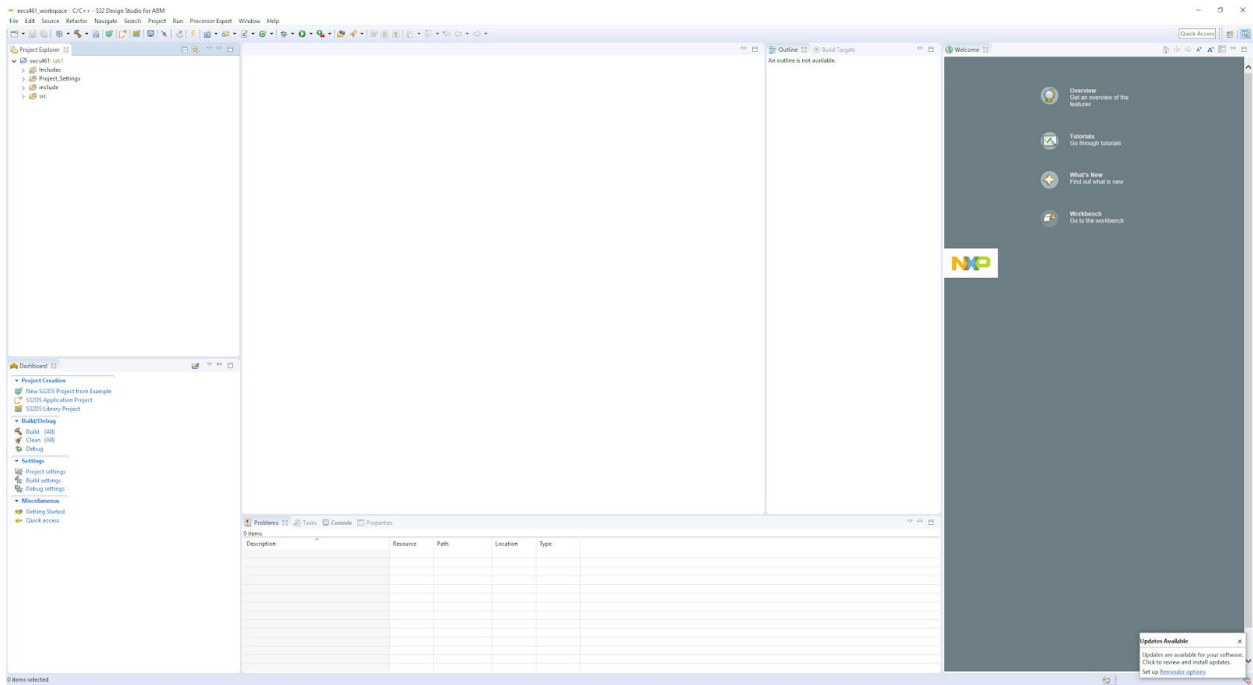
Your window should look like this:



Press “Finish”. You may need to press the windows project icon on the left of the welcome page to view the project (the top icon with two rectangles overlaid)




Your project should now look like this (you can remove the welcome page if you wish)



From here you can edit code in the src and include folders. Makefiles have been created for you to help with building as described below.

Building

Click the arrow next to the build icon  in order to choose which lab folder you wish to build. This lab will now be built every time you press the hammer. This can be changed by selecting a new lab.

Please remember to save files with any new changes before you build and that you need to build before trying to run any new code.

Debugging

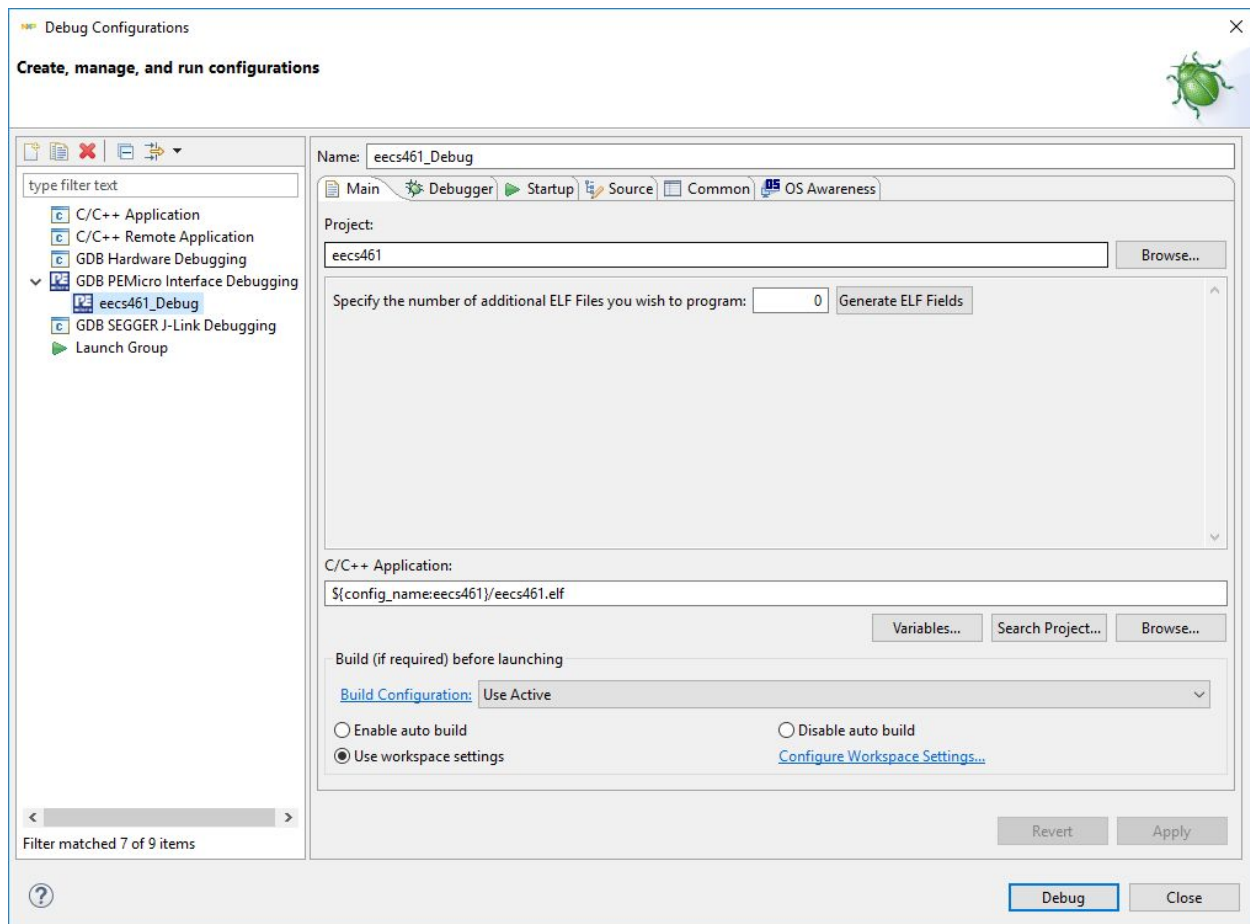
Debugging allows you to run code and choose points on where to stop. These points are called breakpoints.

To begin debugging/ running your program, press the arrow next to the bug icon



Click “Debug Configurations” > GDB PEMicro Interface Debugging > eecs461 Debug

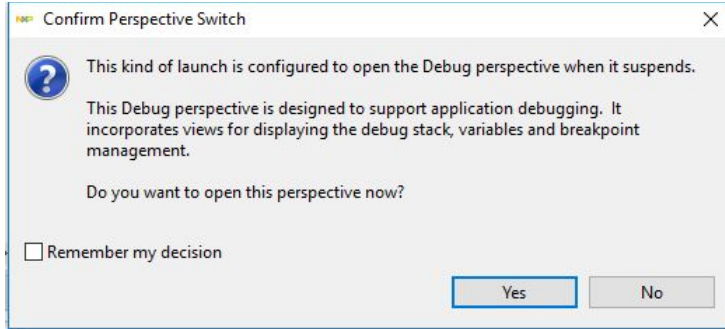
Ensure that the values match this:



And press Debug

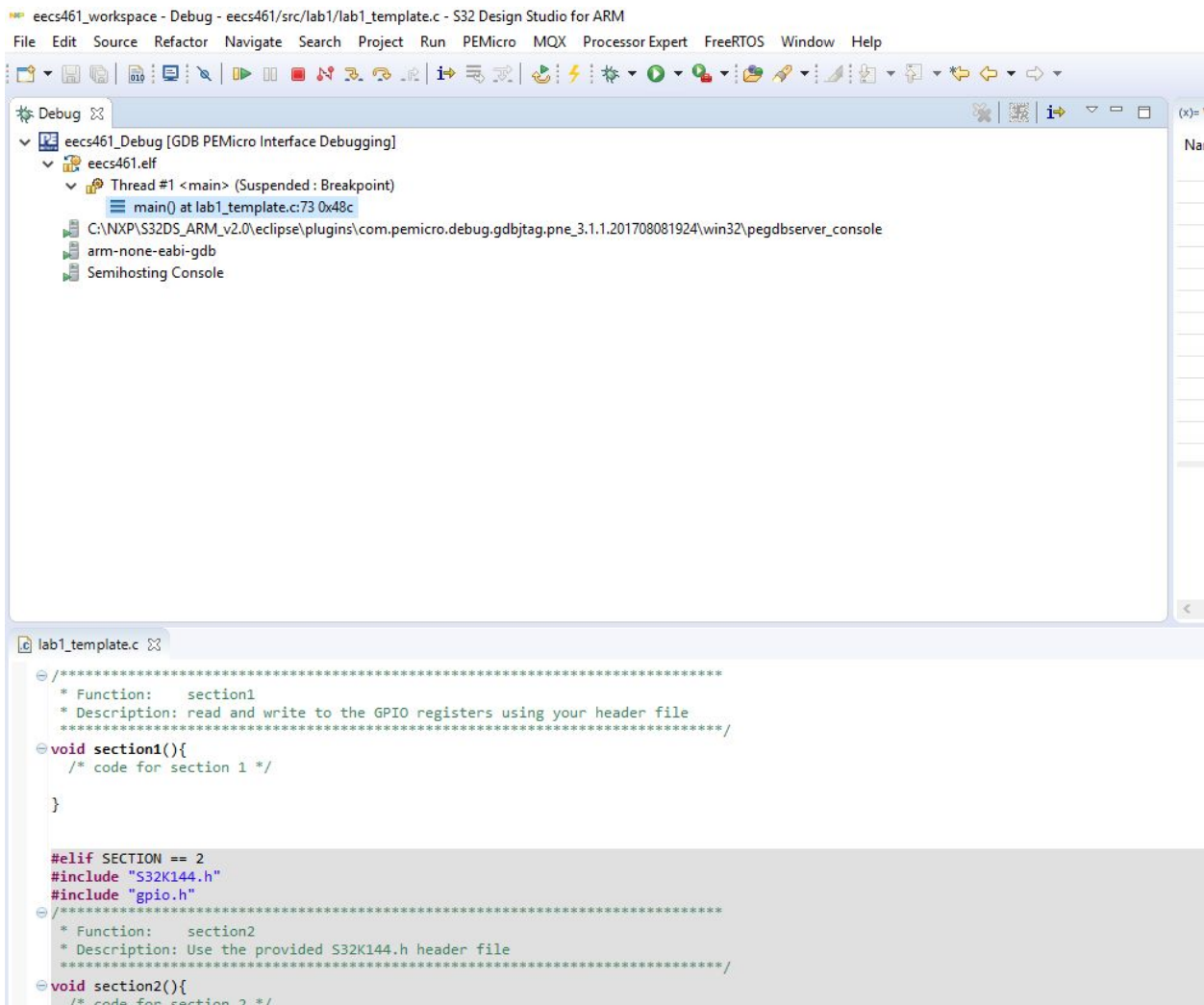
Note: in order to debug, you need to have the board properly plugged in

It will then pop up this window:



All this is saying is that the window will go to the “debug perspective” to allow for breakpoints and monitoring

You should now see a screen like the one below (the code may be different)



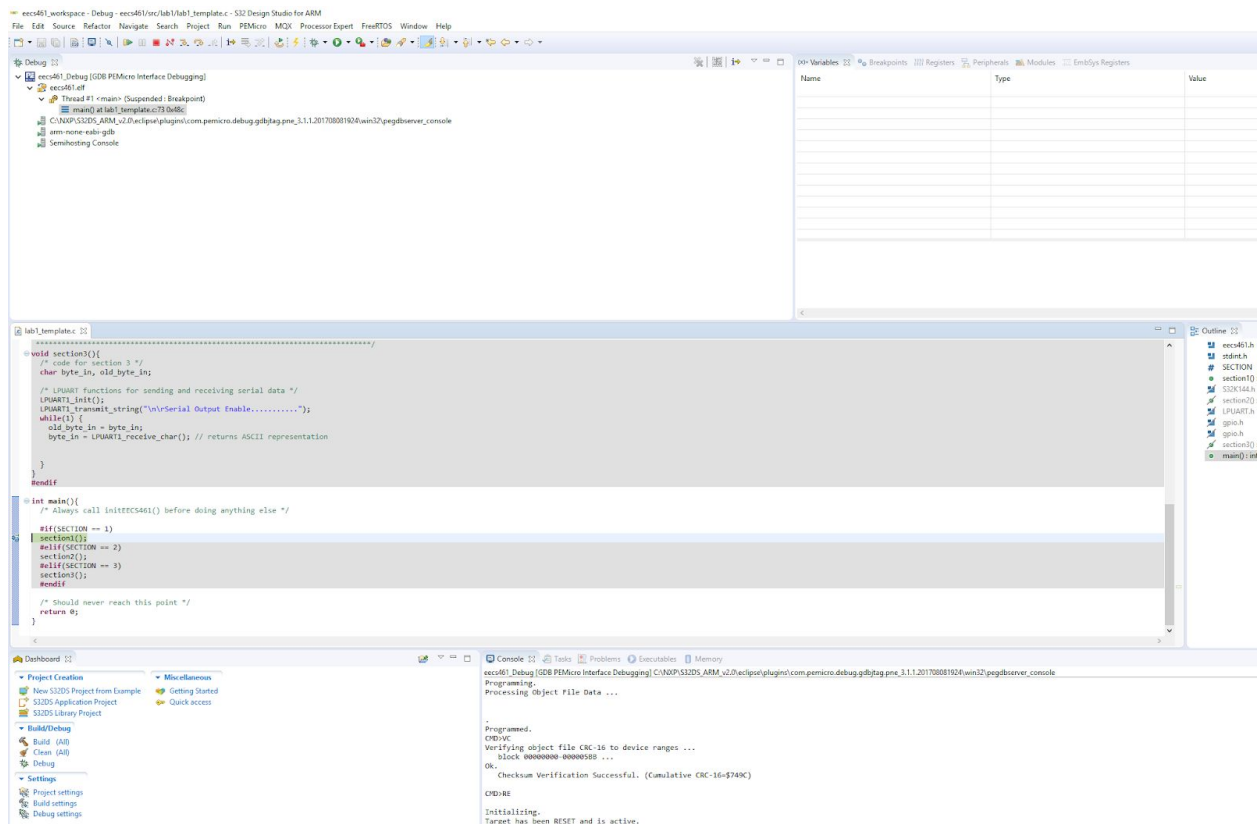
To start running your code, press the green triangle play icon. To stop the code, press the red square stop button.

Note: You can only have one instance of the program running at one time, so make sure to stop your code before trying to start a new instance



To switch back to make edits to your code, click the C file perspective icon in the top right

Breakpoints can be set by clicking to the left of the line of code you wish to stop at. Here you can see the code stopped at a breakpoint.



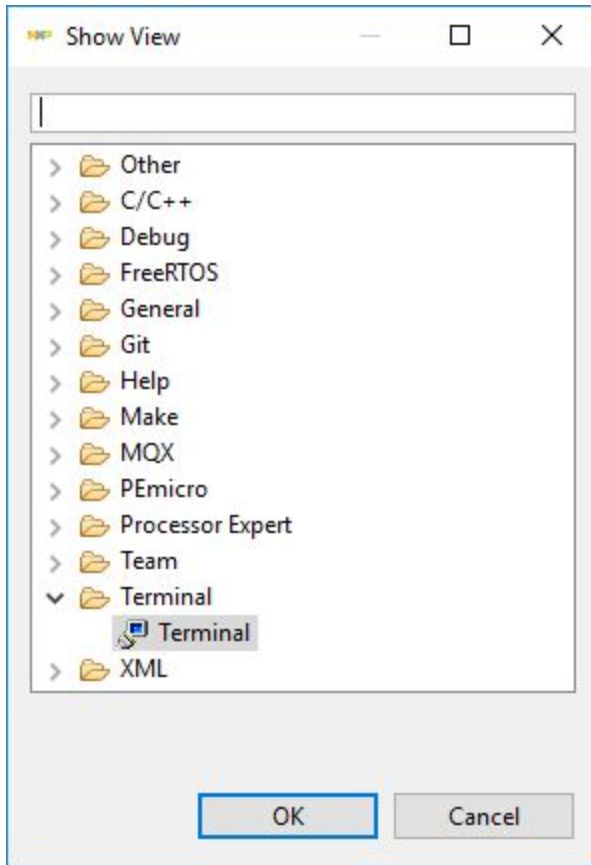
You can also see the state of variables and registers at a breakpoint

Name	Type	Value
> portD_PCR	volatile uint32_t * const	0x4004c000
> portE_PCR	volatile uint32_t * const	0x0 < _isr_vector>
> gpioD	GPIO_mem * const	0xf < _isr_vector+15>
> gpioE	GPIO_mem * const	0x974
(x)= sum	uint16_t	8192
(x)= value1	uint16_t	8192
(x)= value2	uint16_t	28656
(x)= regReadVal	uint32_t	0
(x)= index	int	0

Updated values will turn yellow. If you wish to monitor the state of variable through multiple iterations, you may need to remove and add again the breakpoint (no continuous breakpoints at this time)

Terminal

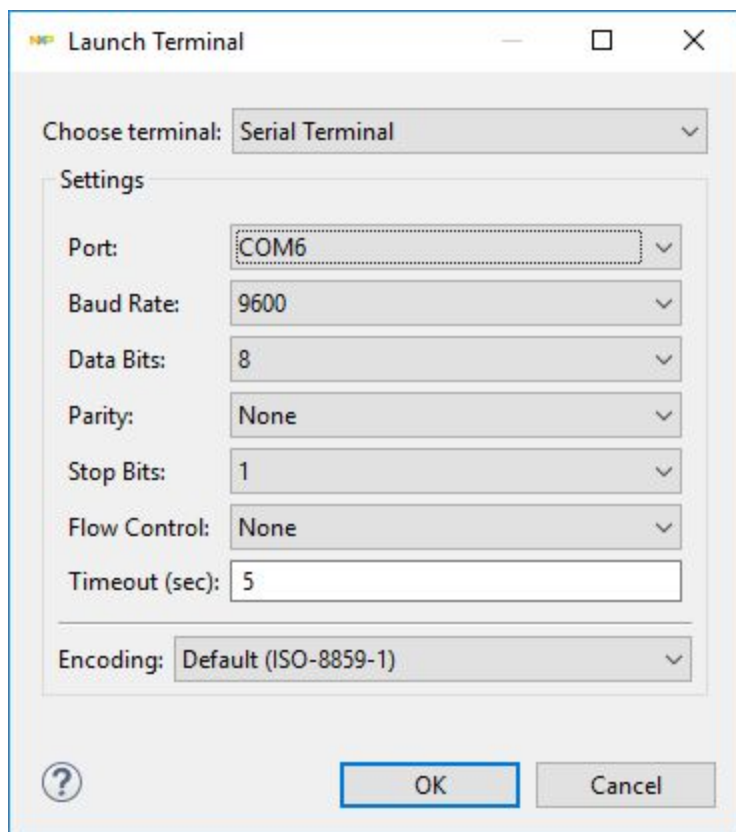
In some labs, we will be using a serial terminal to communicate with the board. To get to the terminal click “Window” > “Show View” > “Other” > “Terminal” as shown below



Click “OK”



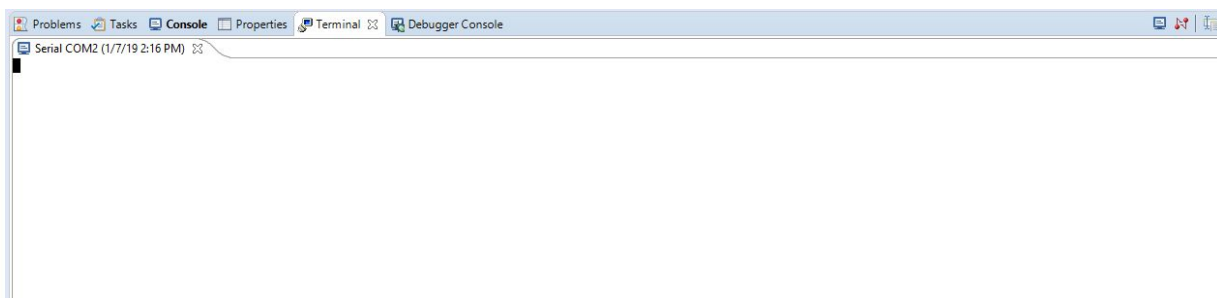
The terminal should open in a tab. Click the icon which will open the “Launch terminal” window. Under “Choose Terminal” choose Serial terminal



and set the correct configurations (as shown above)

You can also get to this menu by pressing **Ctrl+Alt+Shift+T** from any spot. (Which is a lot quicker than the steps listed above)

You should now have a terminal window!



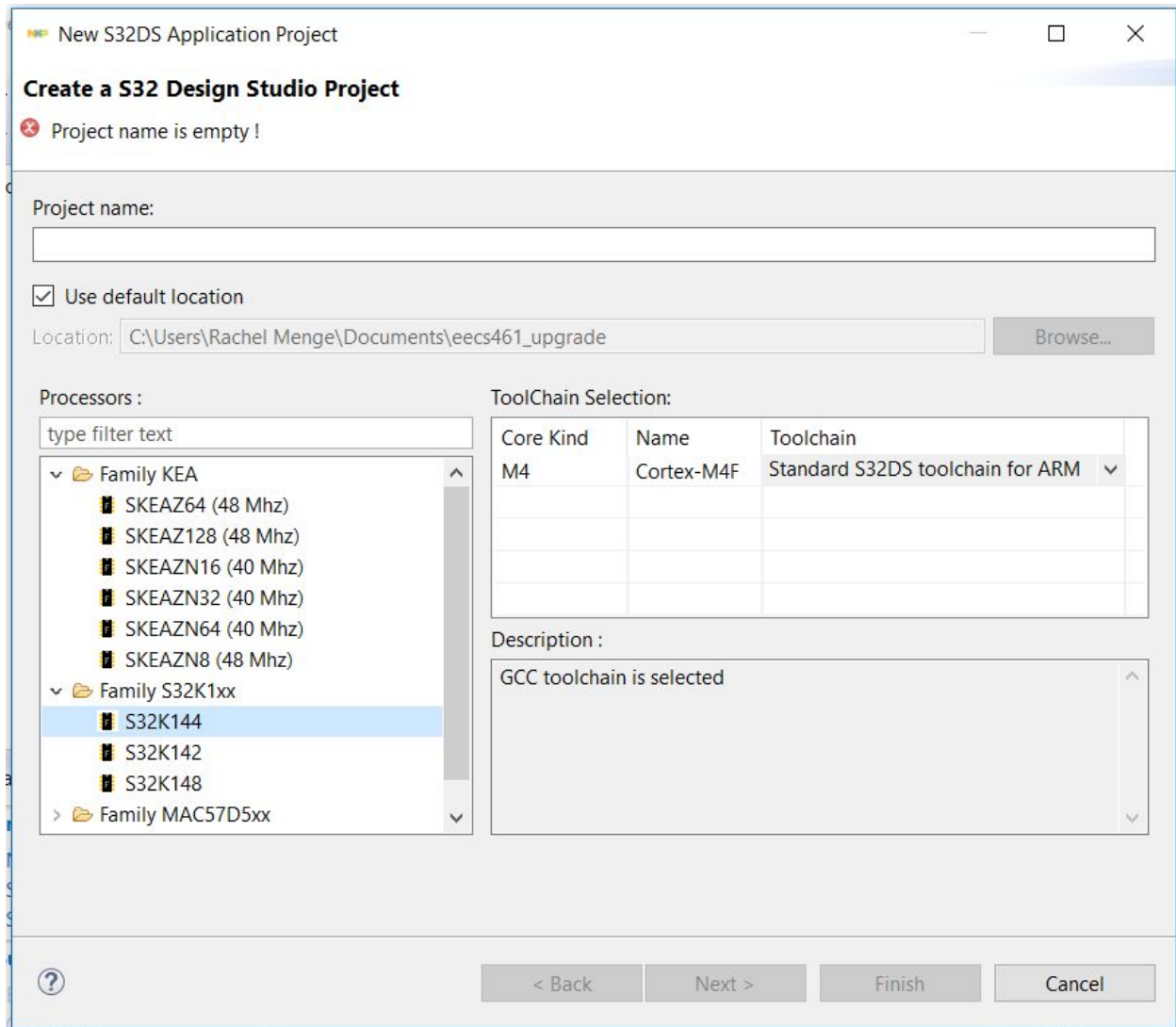
Creating a new project

You should not need to do this but just in case you want to know....

File > New > S32DS Application Project

Give a project name > "eecs461"

Choose the processor



Select Family S32K1xx > S32K144

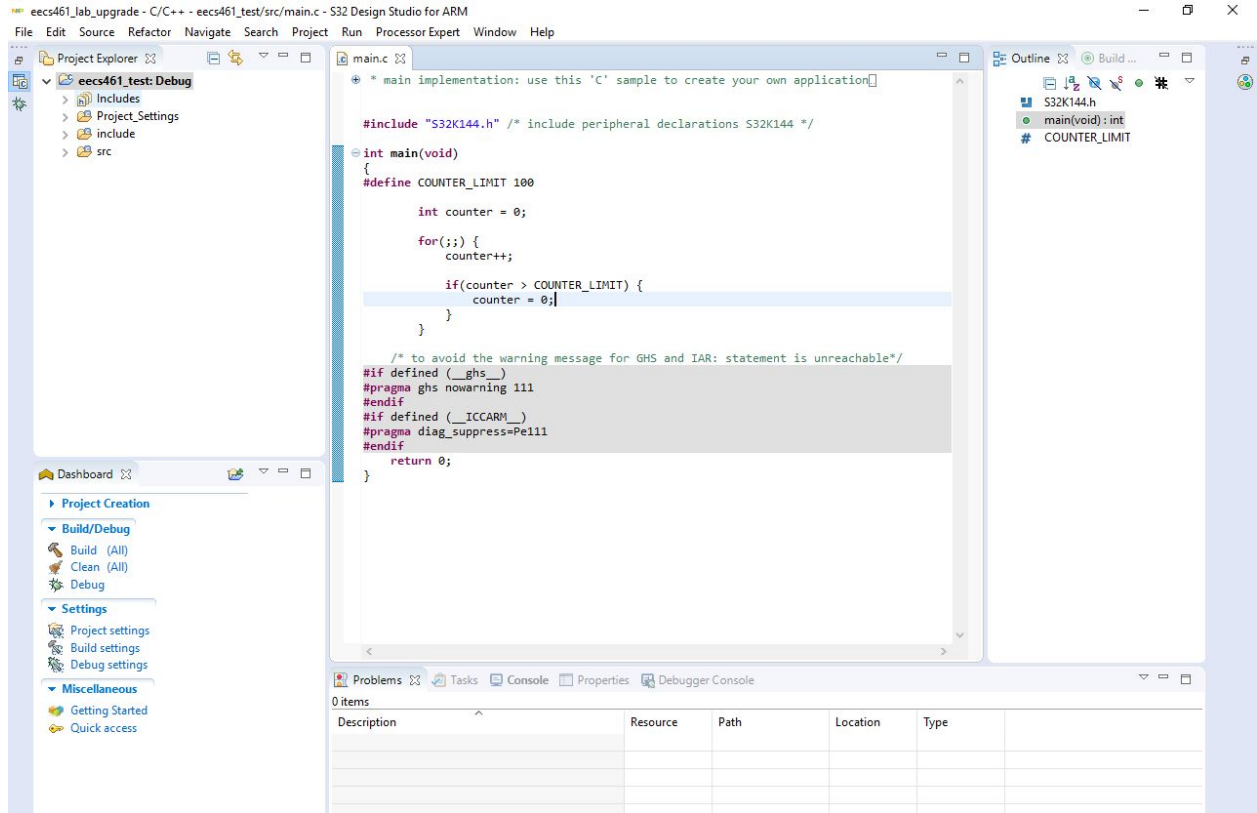
New S32DS Application Project

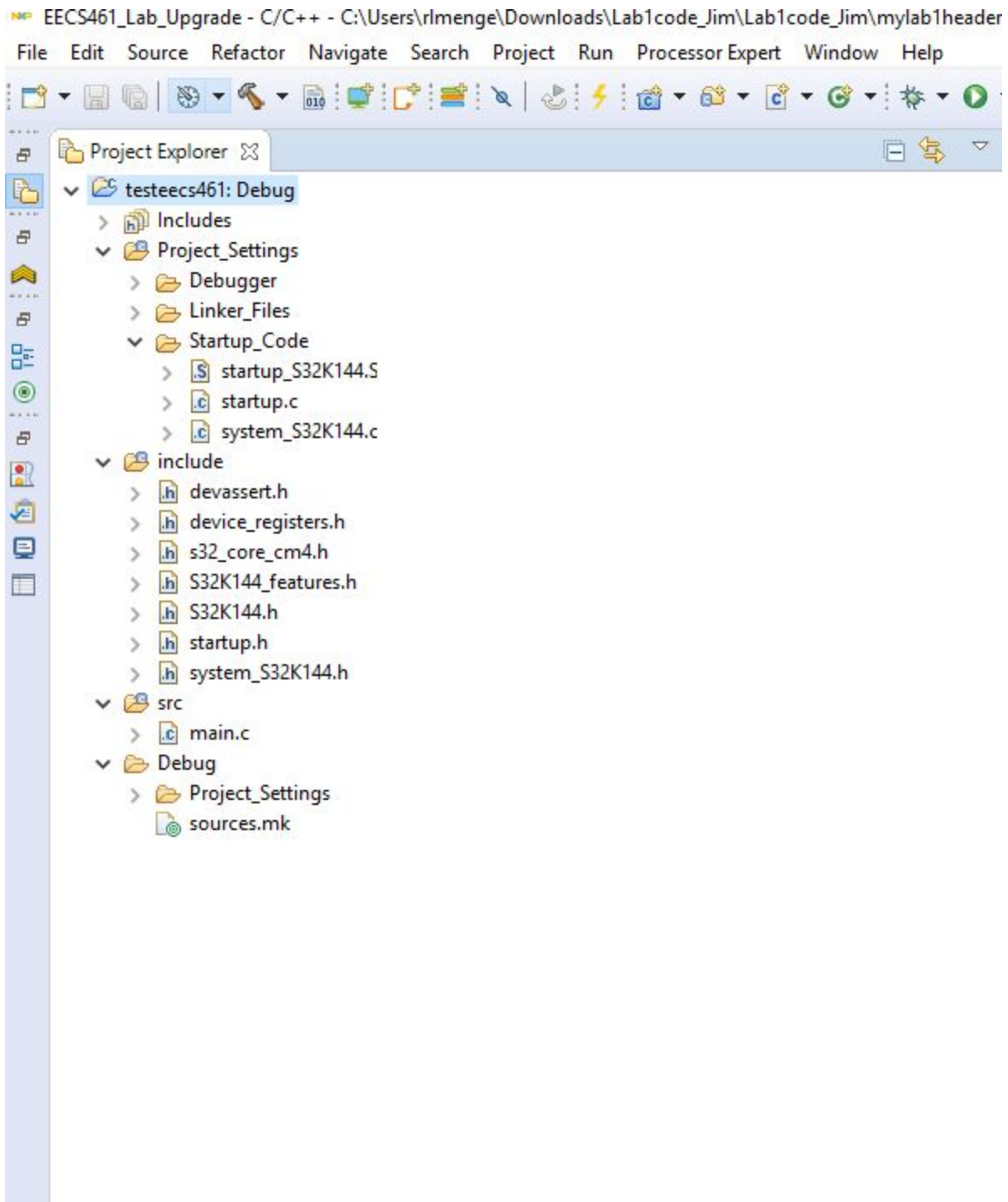
New S32DS Project for S32K144

Select required cores and parameters for them.

Project Name	eecs461_test
Core	<input checked="" type="checkbox"/> Cortex-M4F
Library	EWL
I/O Support	Debugger Console
FPU Support	Toolchain Default
Language	C
SDKs	...
Debugger	PE Micro GDB server

? < Back Next > Finish Cancel





we will be organizing files in 2 different locations

src : will contain all your lab#.c files as well as the lib folder

>lib : will contain all your .c files that do not have a "main" function

include : will contain all your header files

to add files, right click on the folder you wish to add it to
click “New” > “File”

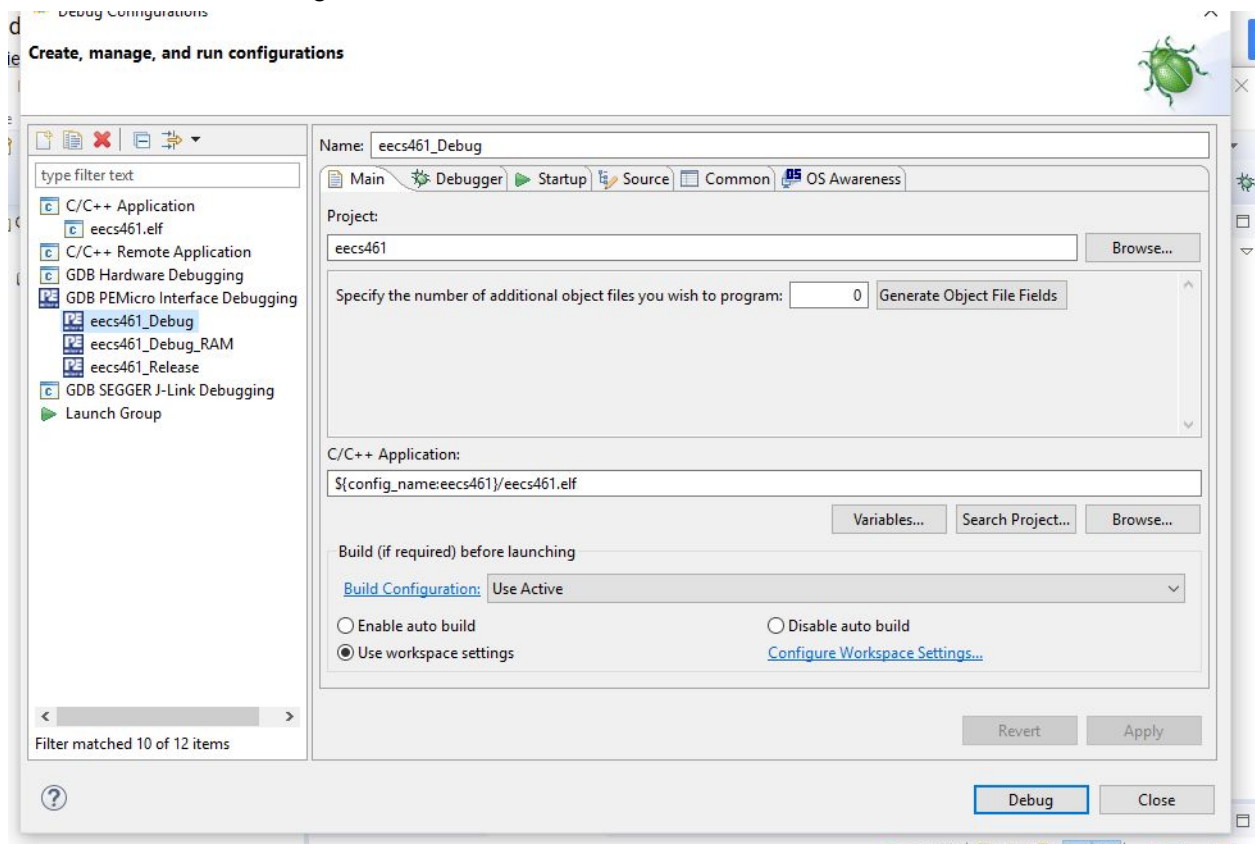
give it the proper name and press “Finish”

Once your code is written, you can now begin to debug.

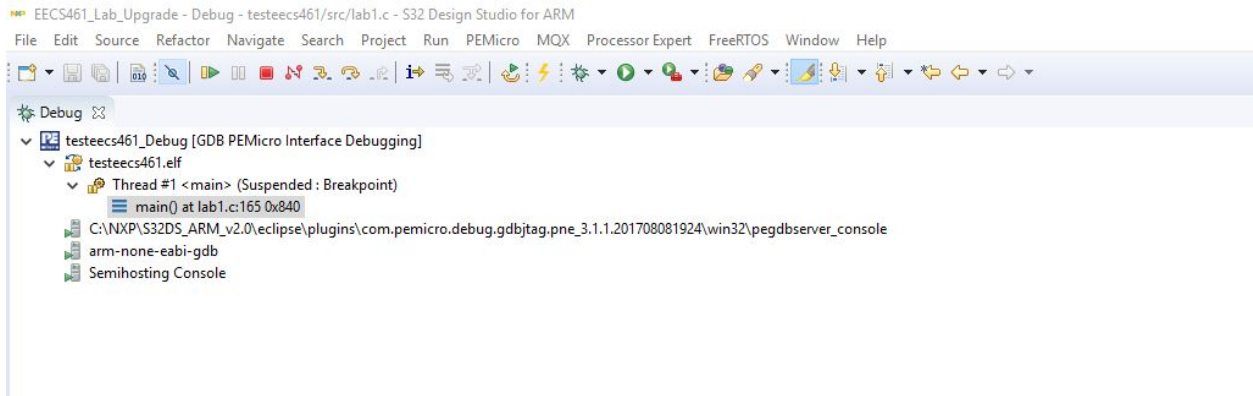
Click the bug icon and then “Debug Configurations...”

Proceed to the GDB PEMicro Interface Debugging and select the eecs461_Debug option

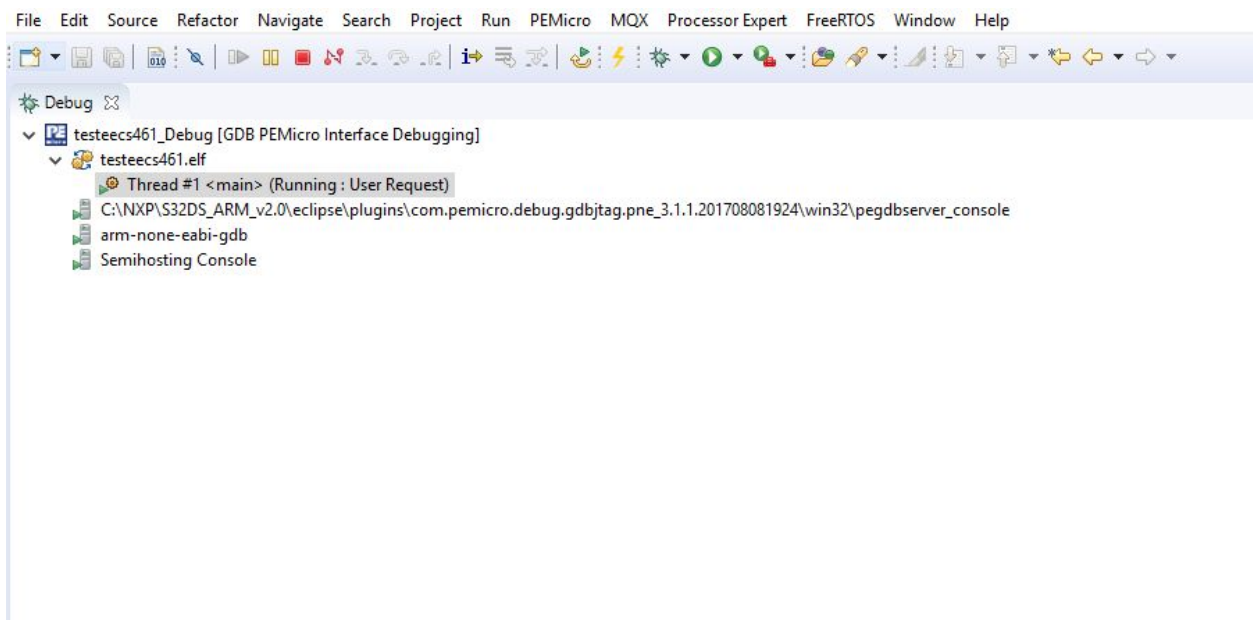
Note: You can not debug if the board is not connected / turned on



Once you click “debug” the console should show the editor beginning the program
To have it start, you need to press the green “play” button



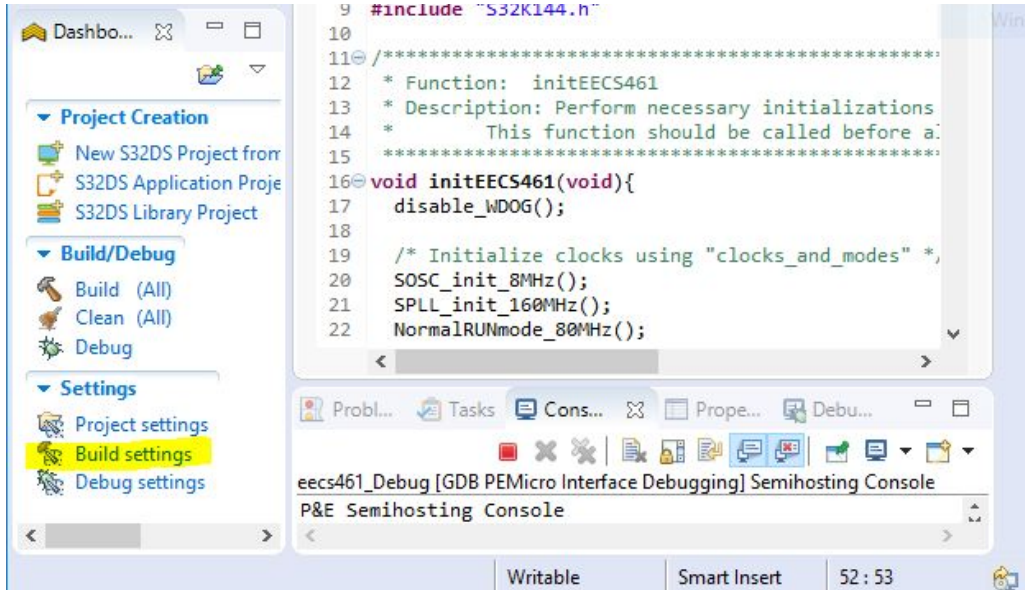
Once you press play your tool bar should look like this



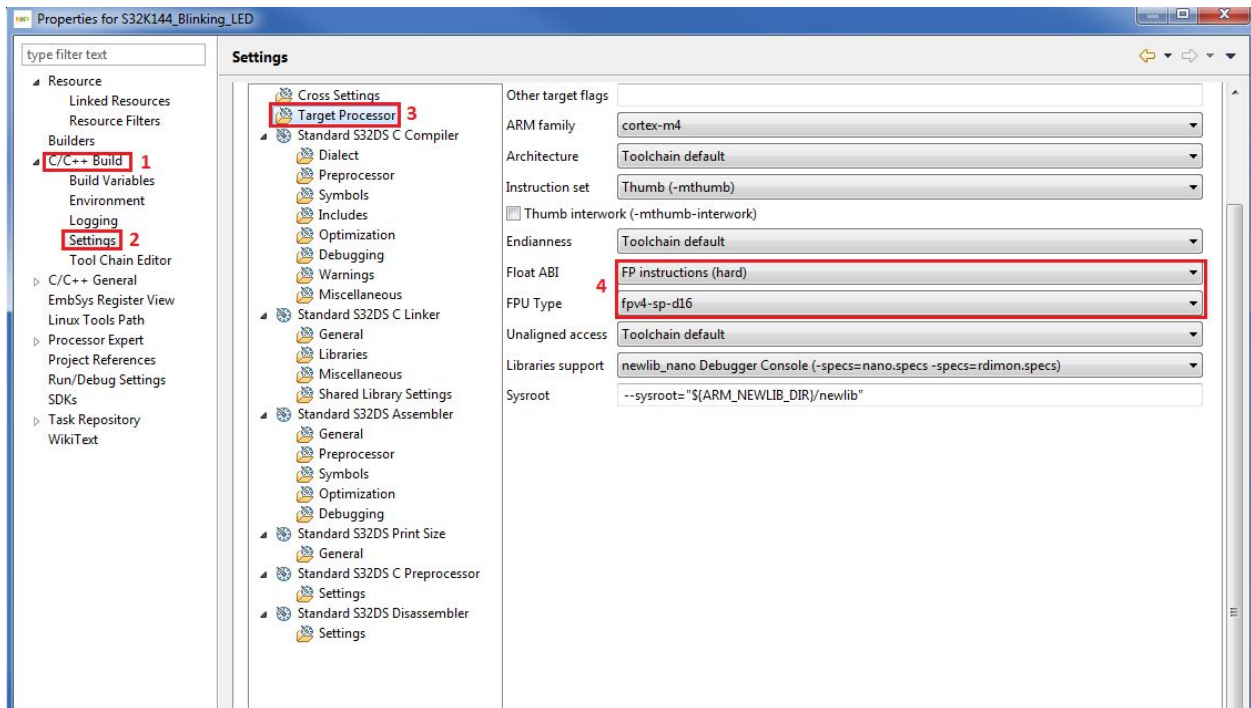
In order to stop the program (or to reset the program) press the red square "stop" button

Enabling Floating Point Unit (FPU)

Go to the build settings



In the build settings, go to (1) C/C++ Build, (2) Settings, (3) Target Processor. Change Float ABI to **FP instructions (hard)** and FPU Type to **fpv4-sp-d16**.



When prompted to rebuild the index, select **yes**.