Introduction

Install MCUXpresso IDE

Our development board is an LPCXpresso 1549 from NXP and we use MCUXpresso development tools for development and debugging. MCUXpresso features an Eclipse based IDE with gcc toolchain and support for integrated debug probe found on the development board.

Go to http://nxp.com/mcuxpresso/ide and download and install MCUXpresso. During installation allow all suggested NXP drivers to be installed.

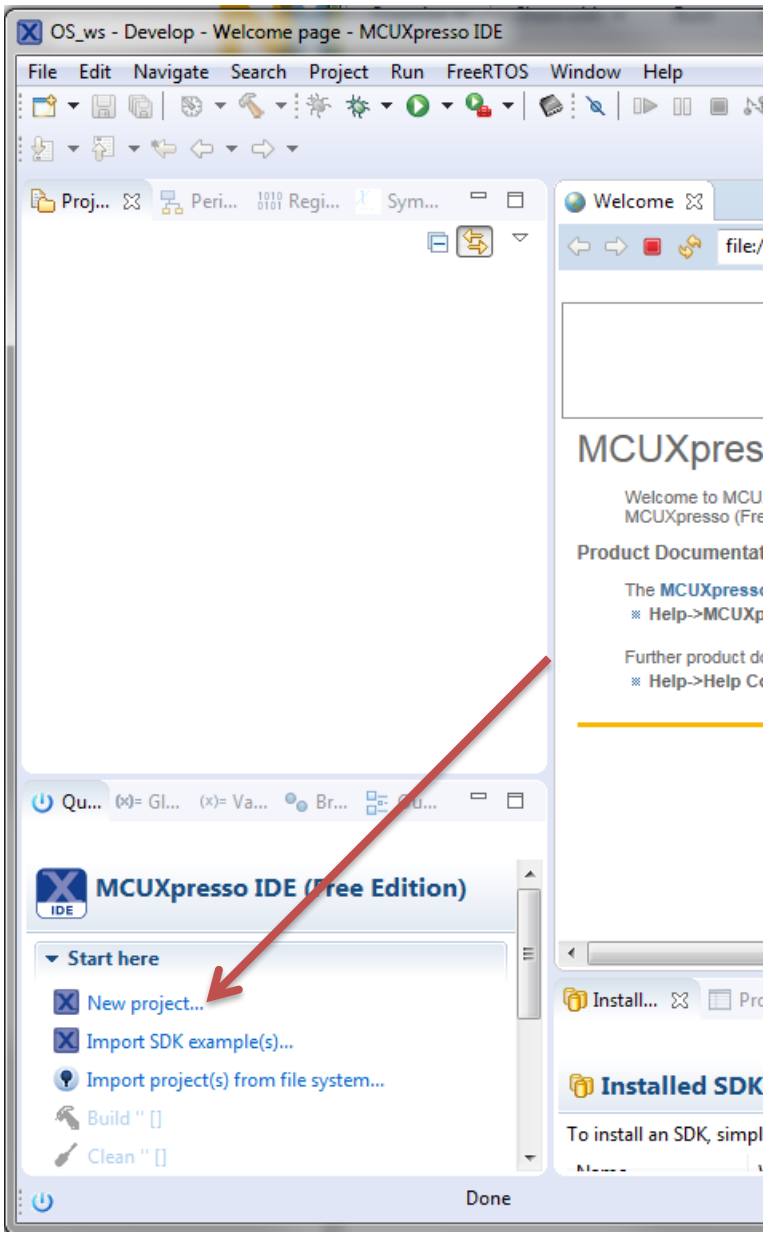
MCUXpresso comes with a vast set of libraries and example code for different platforms. The examples and libraries are installed with the IDE. The files go into
<your installation directory>\MCUXpressoIDE_10.0.2_411\ide\Examples

Create a new workspace and import driver libraries

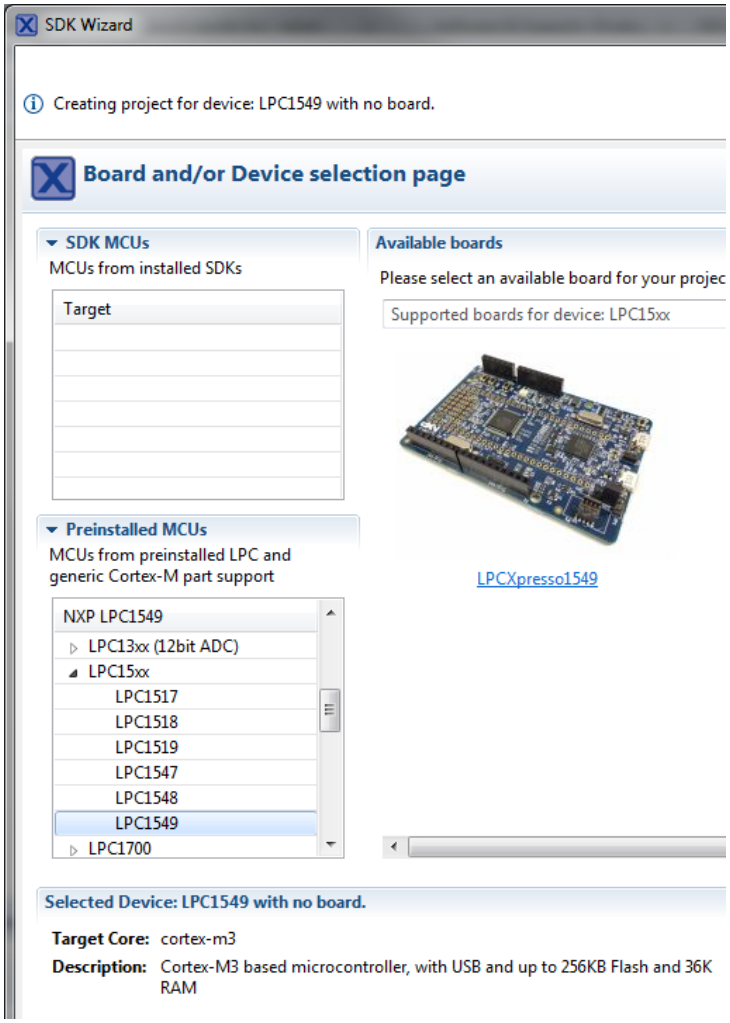
The IDE installation includes LPCOpen software platform that contains peripheral drivers and sample code. Each processor family requires different drivers and it is up to the developer to select and import driver packages to be used.

When you start the IDE you will be prompted to select a workspace. If you select a folder that doesn't have a workspace an empty workspace is created in the directory. A workspace typically contains multiple projects: peripheral driver library projects and your own project(s) that make a reference to (i.e. use) the driver projects.

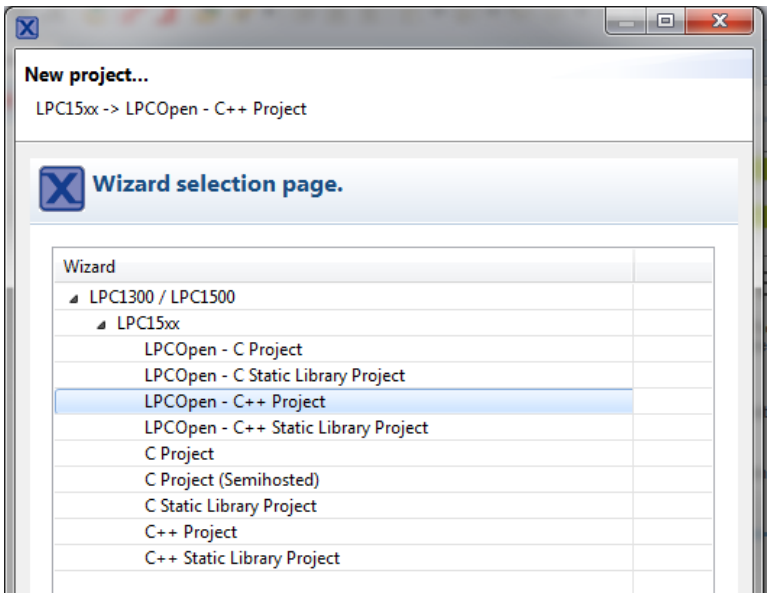
The easiest way to import drivers and to create a new project is to use the project wizard. The project wizard is launched from the quick start window. See screen shot below. The wizard will ask you to import driver libraries when you create the project. However first you need to select processor and project type.



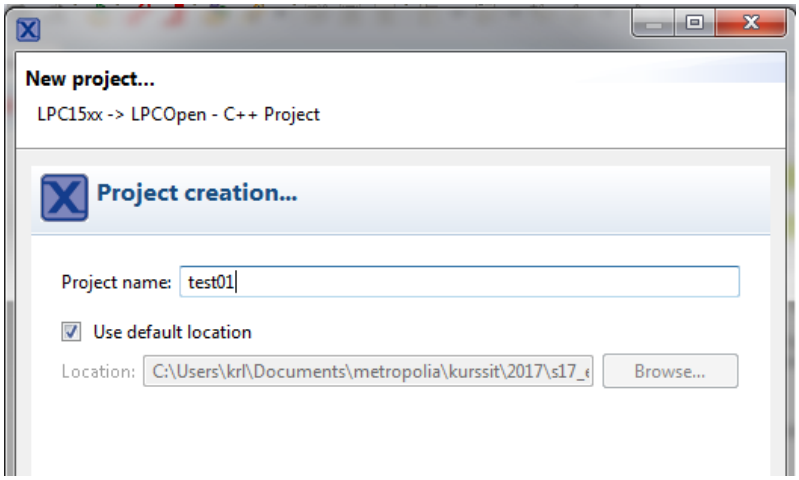
Our development board is based on LPC1549 processor so we select LPC1500-series from **Preinstalled MCUs**. Expand **LPC15xx** categories and select **1549**. Then click the picture of your board on the wizard and press next.



Then you need to select project type. Select **LPCOpen – C++ Project** and press next.

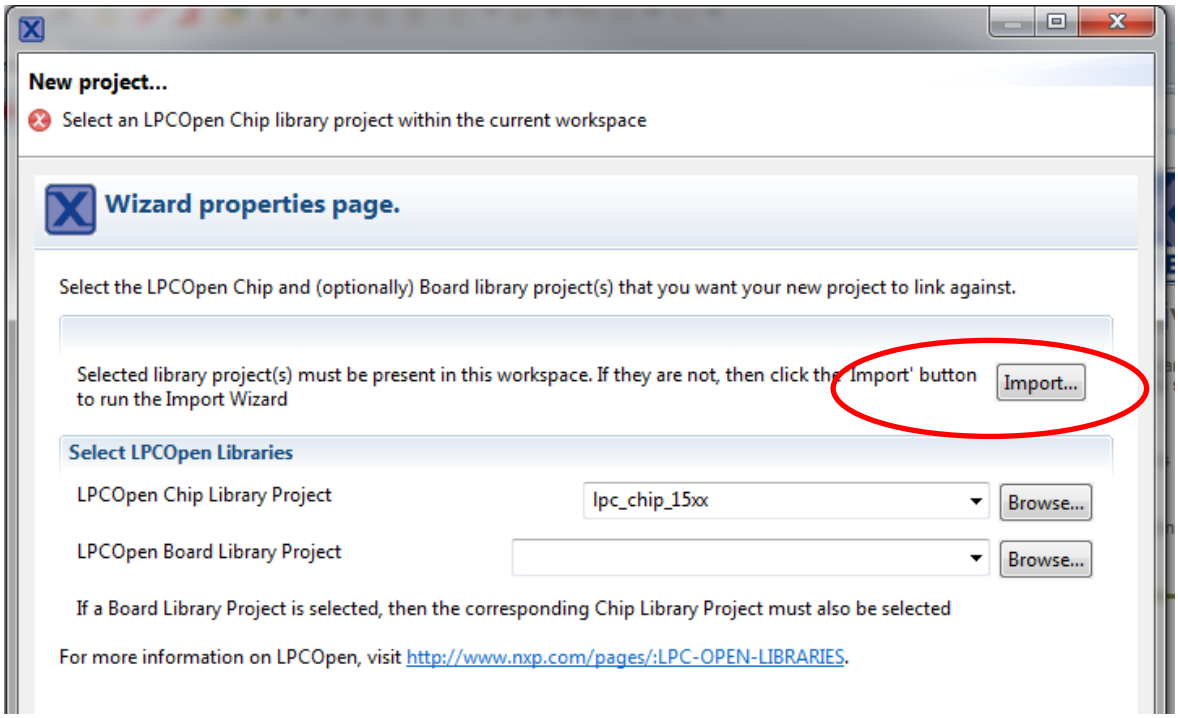


Then name your project. **Don’t use spaces in the project names** some tools may produce unexpected errors path name contains spaces. Just press next after entering the project name.



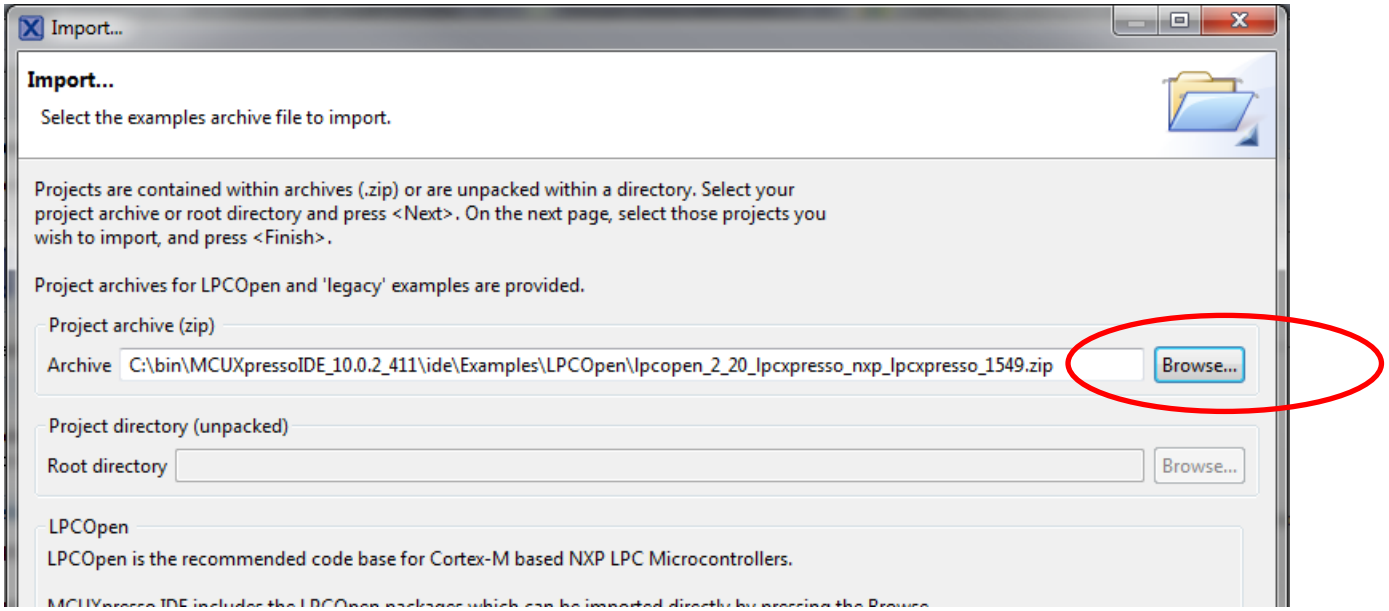
Then we add chip and board libraries to the project. This step is very important since **there is no easy way of adding chip and board libraries to a project that was created without them.**

If you are adding a project to an empty workspace you need to import chip and board libraries in to the workspace. Click import button to start the import wizard.

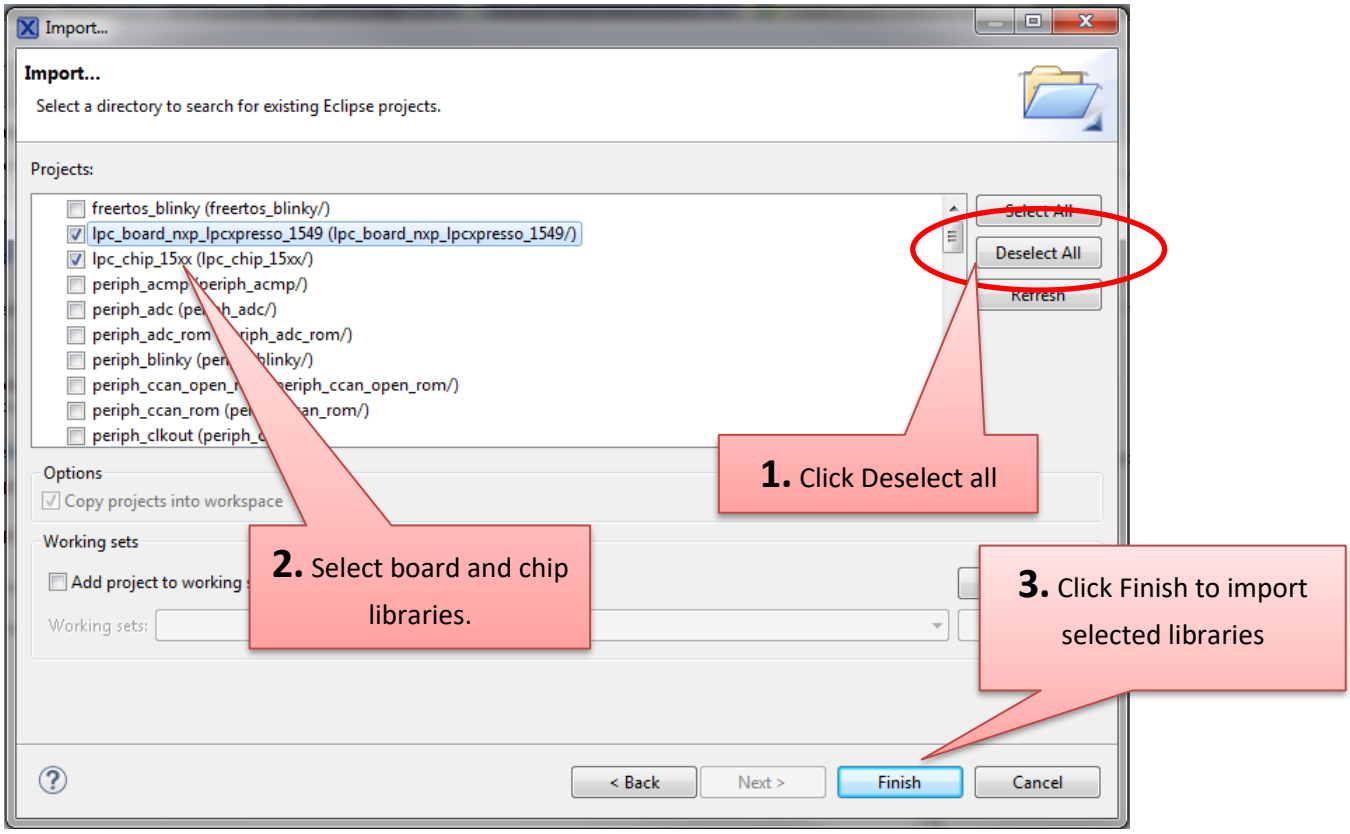


During installation the library archives are copied in to the installation directory. Click Browse and navigate into your installation directory. Then locate the library archives in:

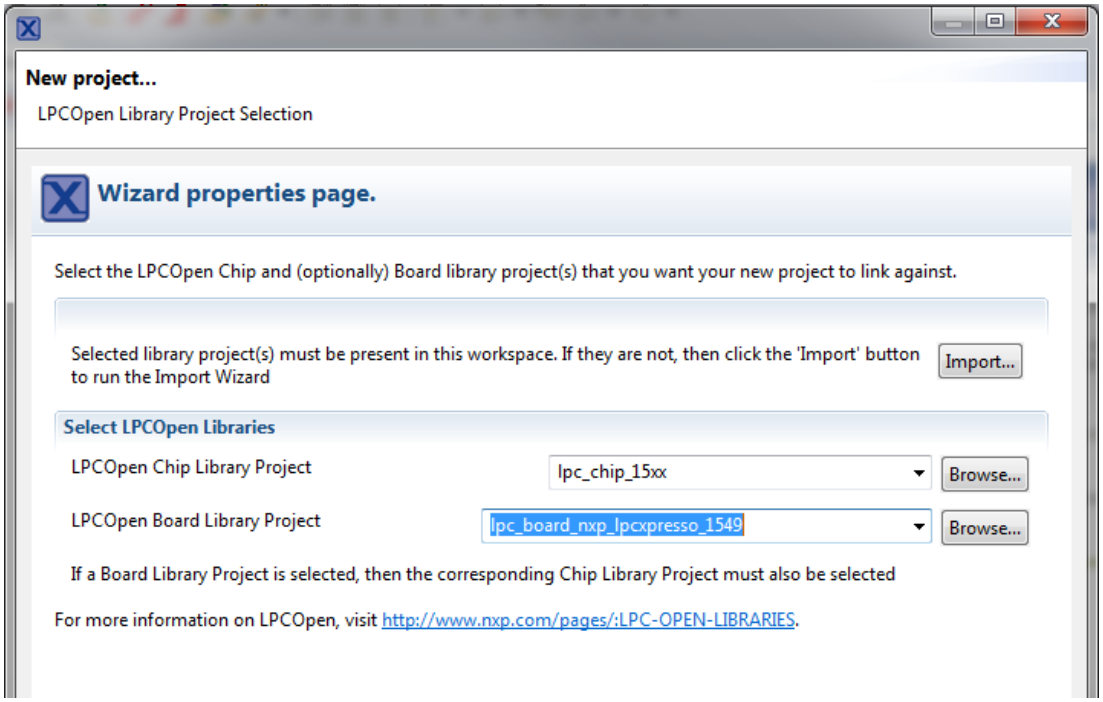
<your installation directory>\MCUXpressoIDE_10.0.2_411\ide\Examples\LPCOpen and select lpcopen_2_20_nxp_lpcxpresso1549.zip



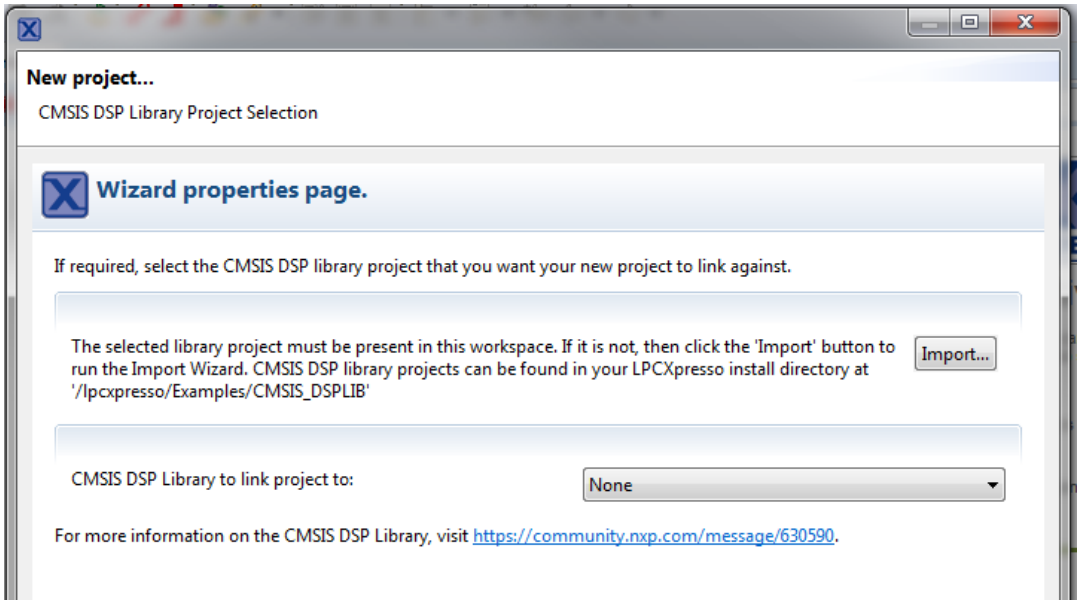
Then press next and a menu with a list of projects in the archive is shown. Click **deselect all** and manually check board and chip libraries. See the screenshot below.



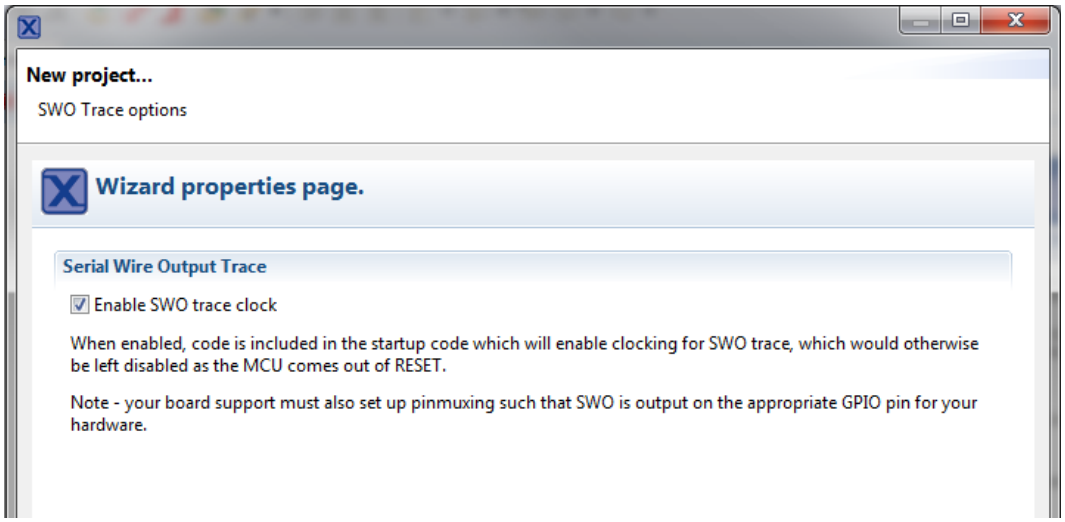
After the import you are returned to the New project wizard. Make sure that you have selected both chip and board libraries and press next.



Don’t add a DSP library.

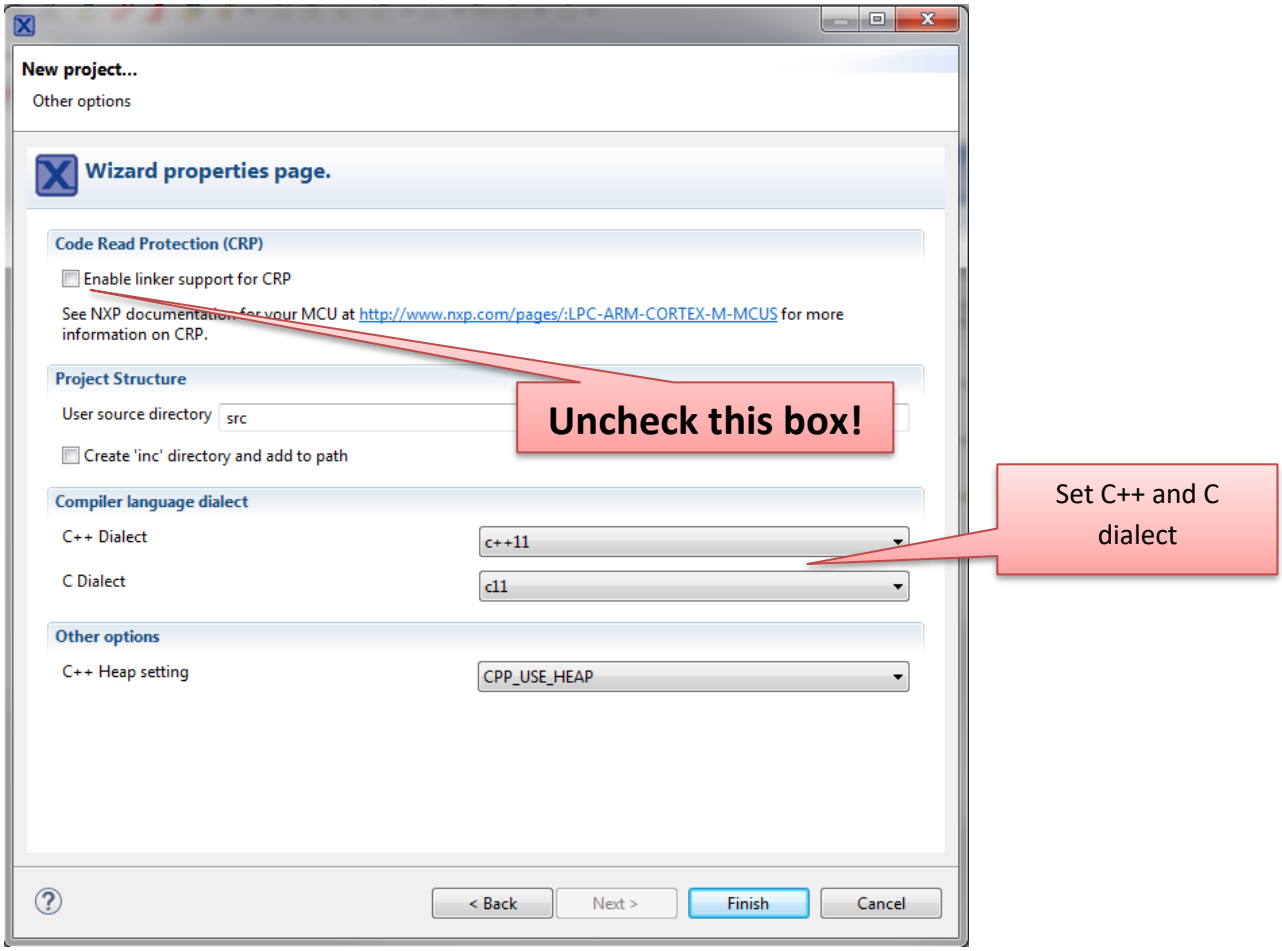


Make sure that Enable SWO trace clock is selected. Debugging will not work without SWO trace clock.



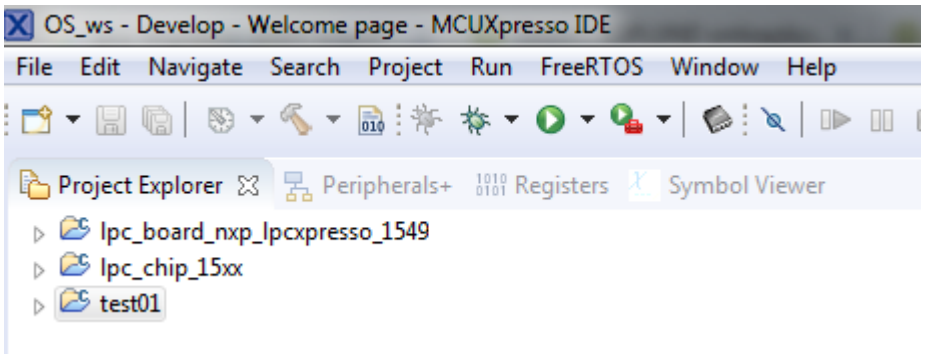
Then select dialect for C and C++. Use c++11 and c11.

Note that there is a bug in the project wizard which does not set C++ dialect properly. At some point in the future this will (hopefully) be fixed so it is a good habit pick the dialect in the wizard.



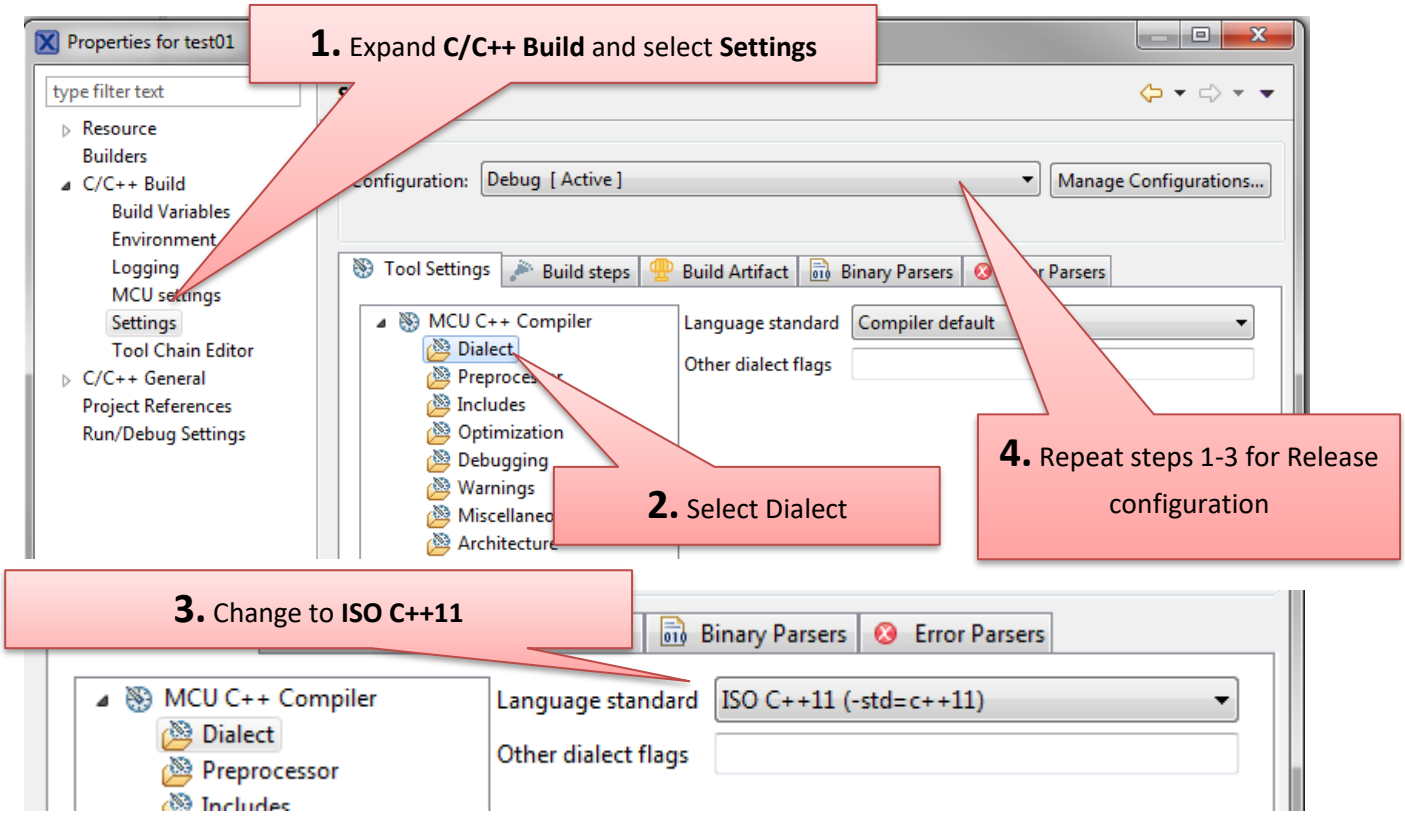
Press Finish and the project will be created. There are still a few steps that need to be taken before you are done.

Now you should have three projects in you workspace.



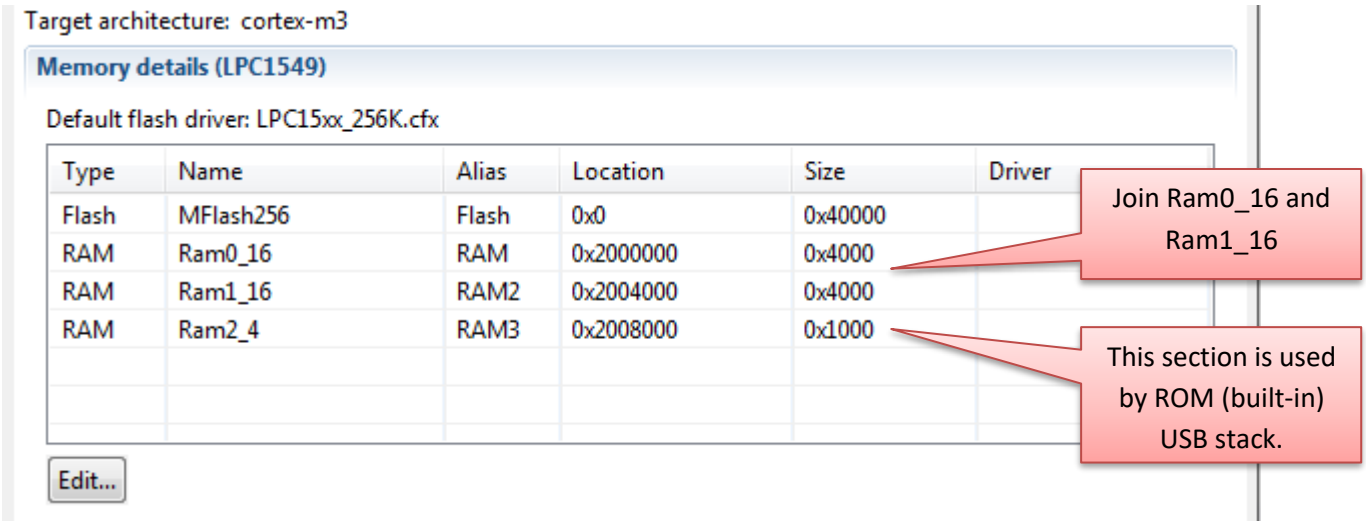
The next thing to do is to set C++ dialect of your project.

Right click the project name and select **properties** at the bottom of the popup menu.

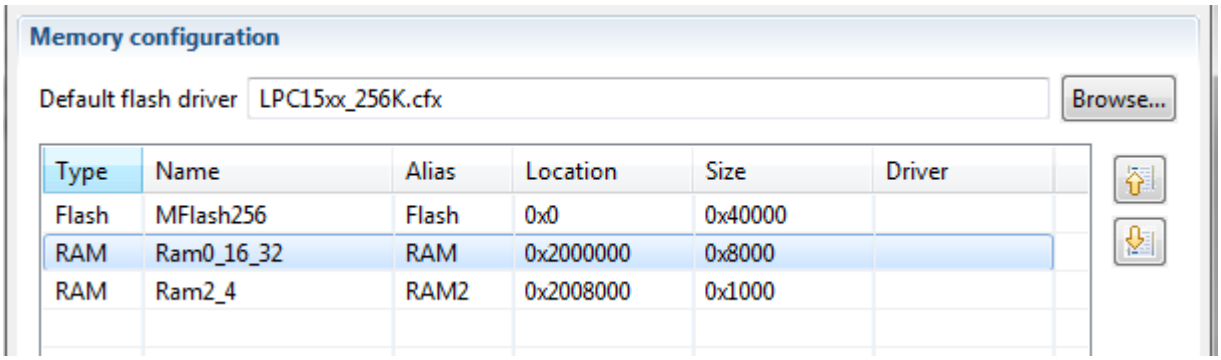


Then we need to increase the size of RAM available for linker. The onchip RAM is divided into three sections that can be enabled or disabled individually. All three sections are enabled at boot so there is no need to enable them. The default linker script divides RAM into three separate sections and by default linker uses only the section at the lowest address. Since the sections are adjacent in the RAM we can join them into a bigger section thus allowing linker to use more RAM.

Go to project properties →C/C++ Build → MCU settings. The default memory MAP looks like this:

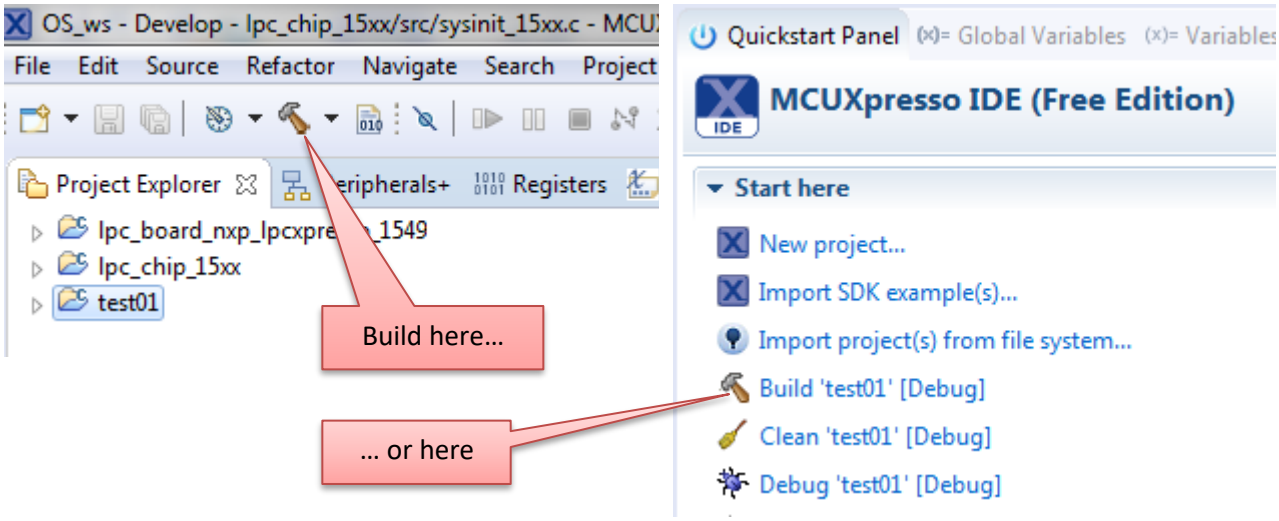


We are going to join the first two RAM sections into a bigger section. Click edit and click on Ram0_16 and then click join. You should get the following memory configuration.



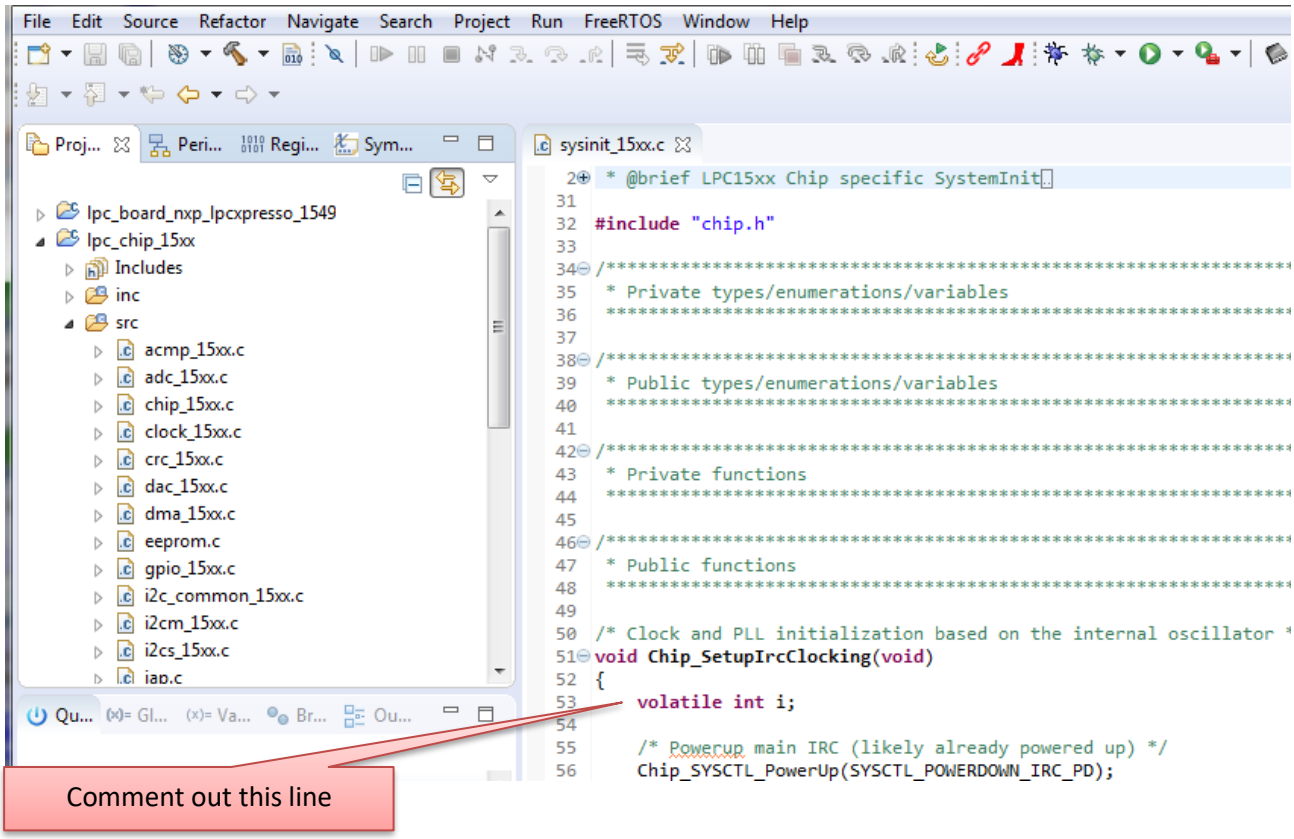
Press OK to close the editor and OK again to apply the changes you made.

The next thing on the list is to test that your project compiles. Click your project in the project explorer and click build.



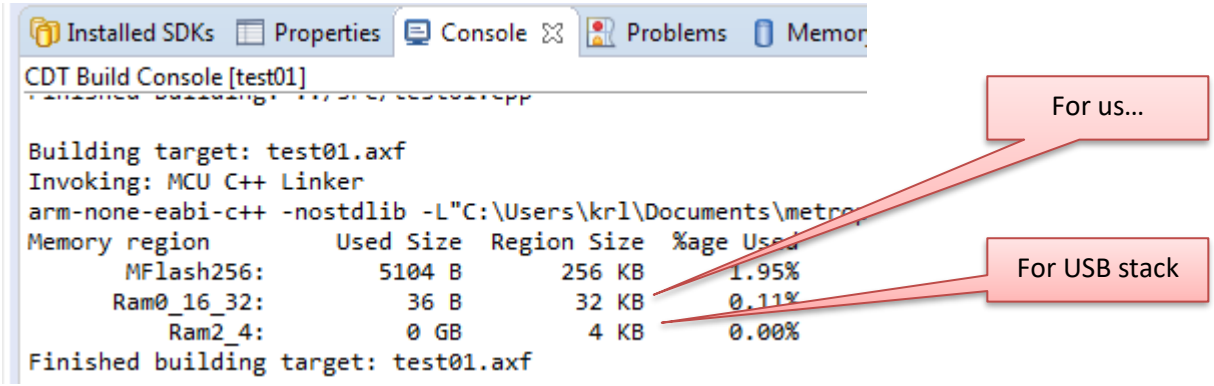
There will be warning about unused variable in the chip library.

Then we need to get rid of the warning in the chip library. Open sysinit_15xx.c in the editor



Then build your project again. If the build finishes with no errors or warnings you are ready to start coding.

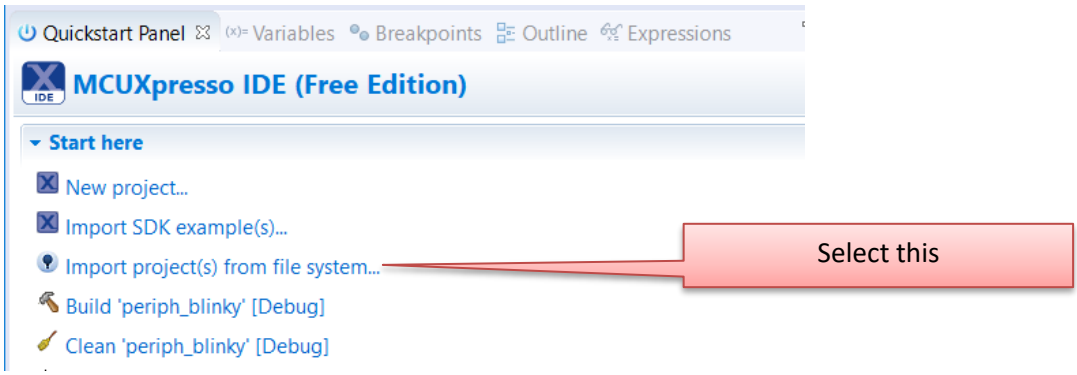
Remember to check the flash and RAM usage from the console window to be sure that your memory configuration is OK.



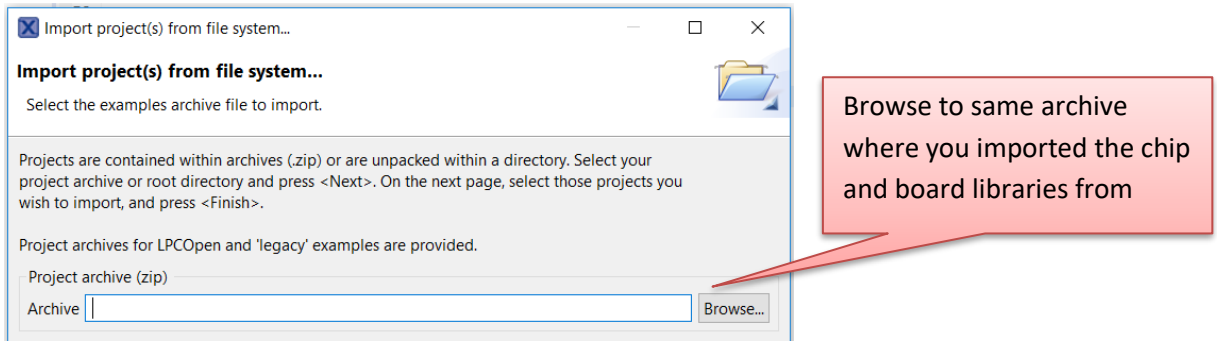
Development board and debugging

You have just created a workspace with driver libraries, an empty C++ project.

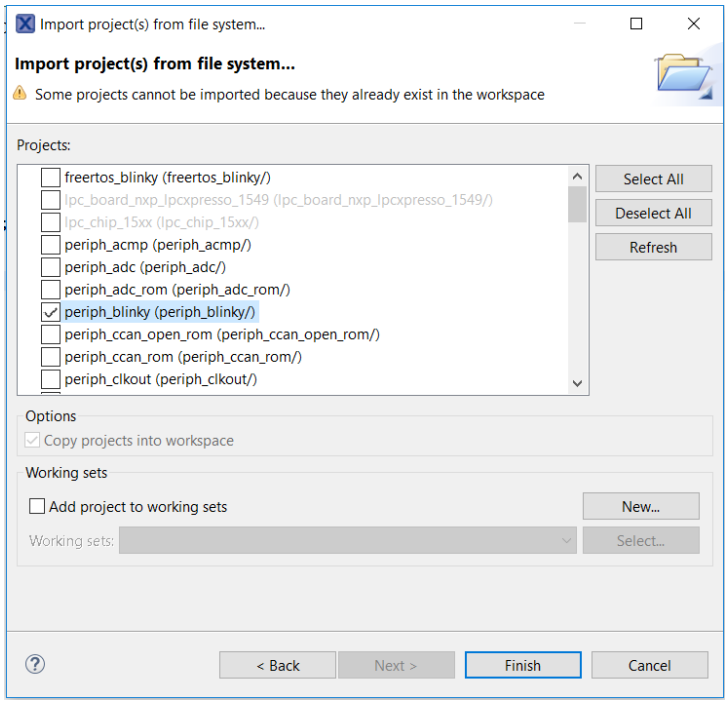
To test debugger, we are going to import a test project to see that we get the board to do something that we can observe. The test project is called periph_blinky and it uses ARM systick timer to make a led blink.



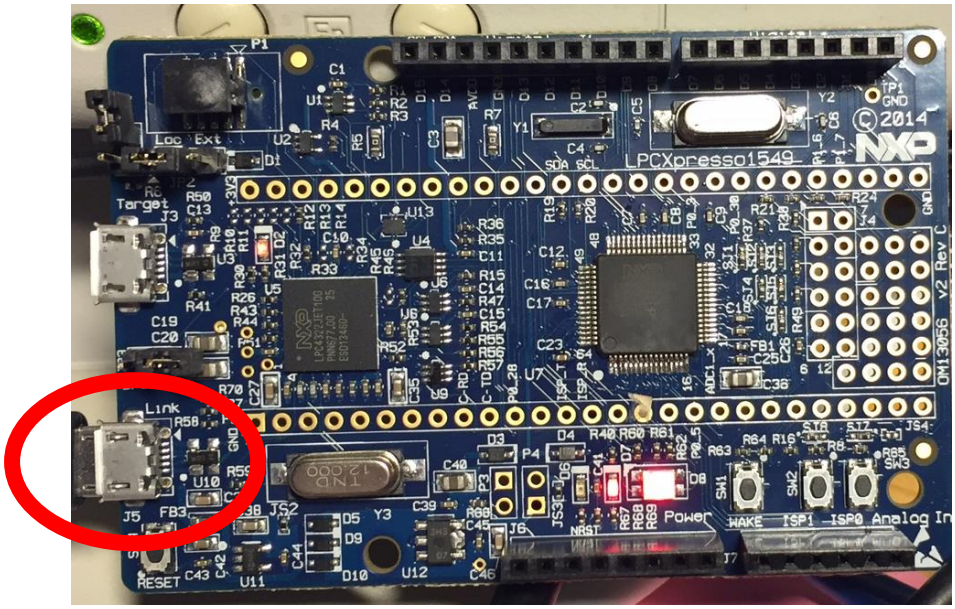
Import wizard will start:



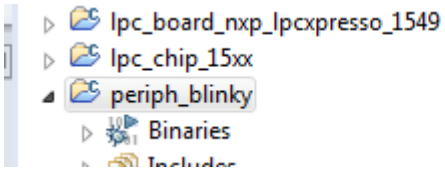
Deselect all except periph_blinky



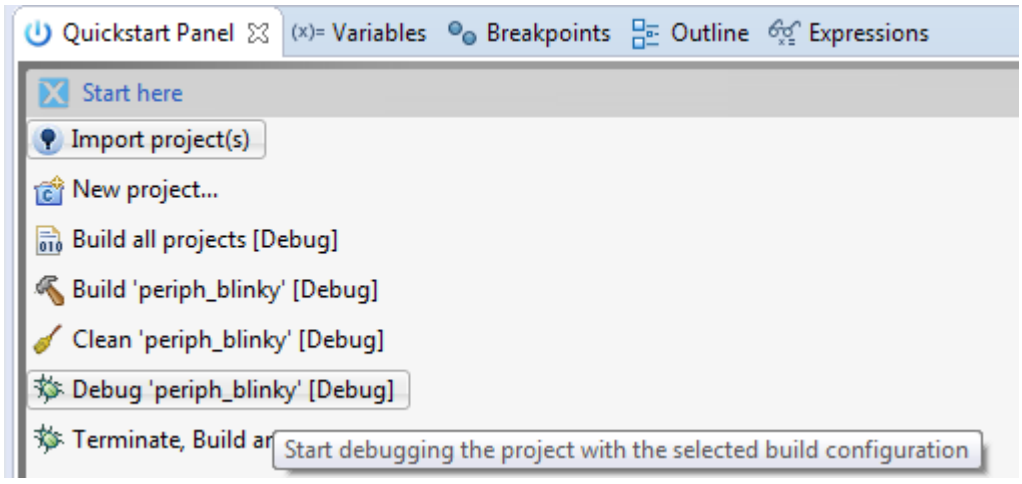
Connect your development board to your computer. Note that development board has two micro-USB connectors. Make connection using the **Link** connector (next to the reset button).



Then click on periph_blink project in Project explorer.



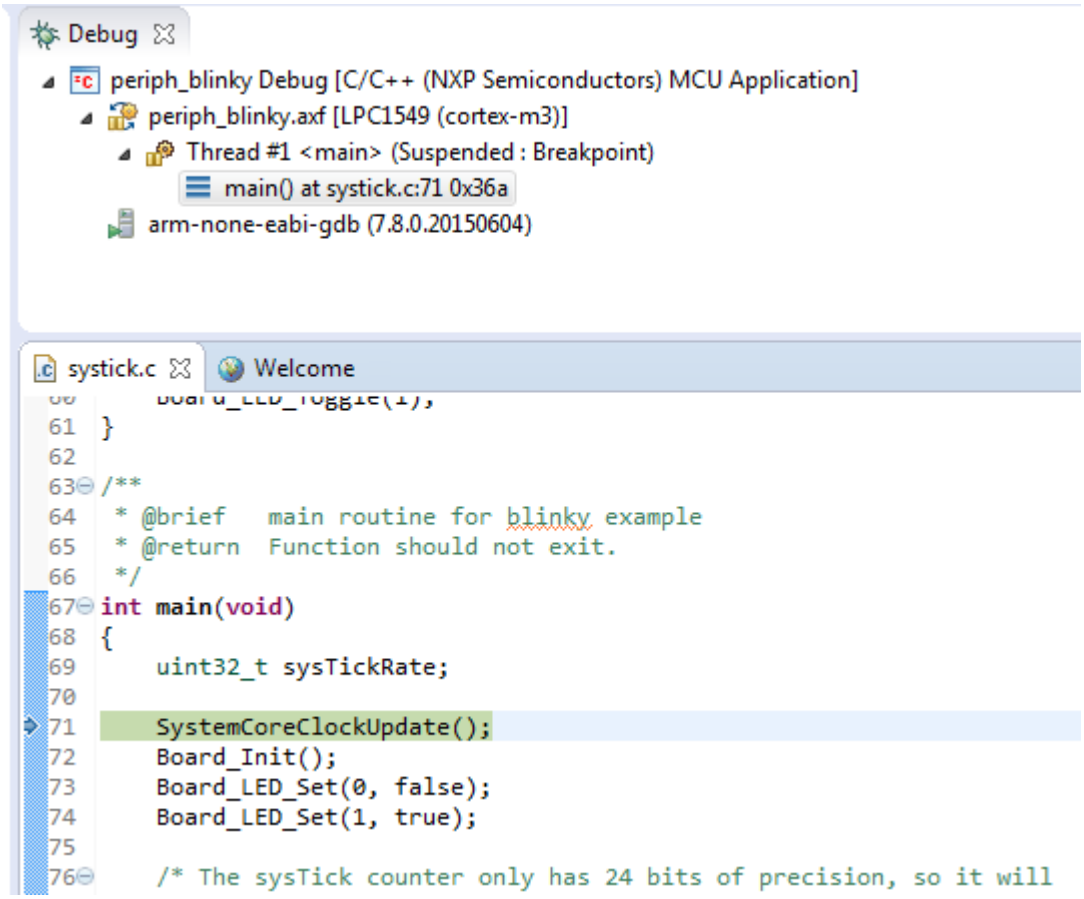
Then click Debug in the Quick start Panel.



IDE then compiles the program and starts a debugging session. On the first run you will be prompted to select a debugger. You should see CMSIS DAP in the list and it should be only one in the list. If you don't see it then click Refresh. If that doesn't help, try using a different USB port.

When your debugging session starts the program execution stops at the first executable line of main function. You can step into or over source lines, set break points and run your program. When you let the

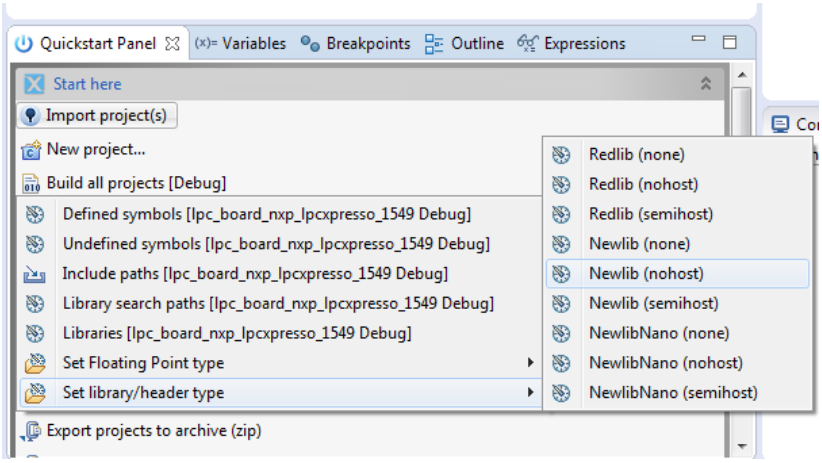
program continue you should see then on board led blinking vigorously. When you suspend the running program the cursor will typically be in the infinite loop at the end of main function.



Common linker errors

If you get a linker error about __isatty , _exit or other functions with an underscore at the beginning of the name try resetting library/header type in Quickstart panel.

Click on lpc_board_nxp_lpcxpresso_1549 library. Go to Quickstart Panel / Quick Settings and set library/header type to **Newlib (nohost)**



Repeat this step for lpc_chip_15xx and your project. Then activate your project and do **Clean** and then **Build**.

If the debugger can’t connect to the board check that the jumpers are set correctly.