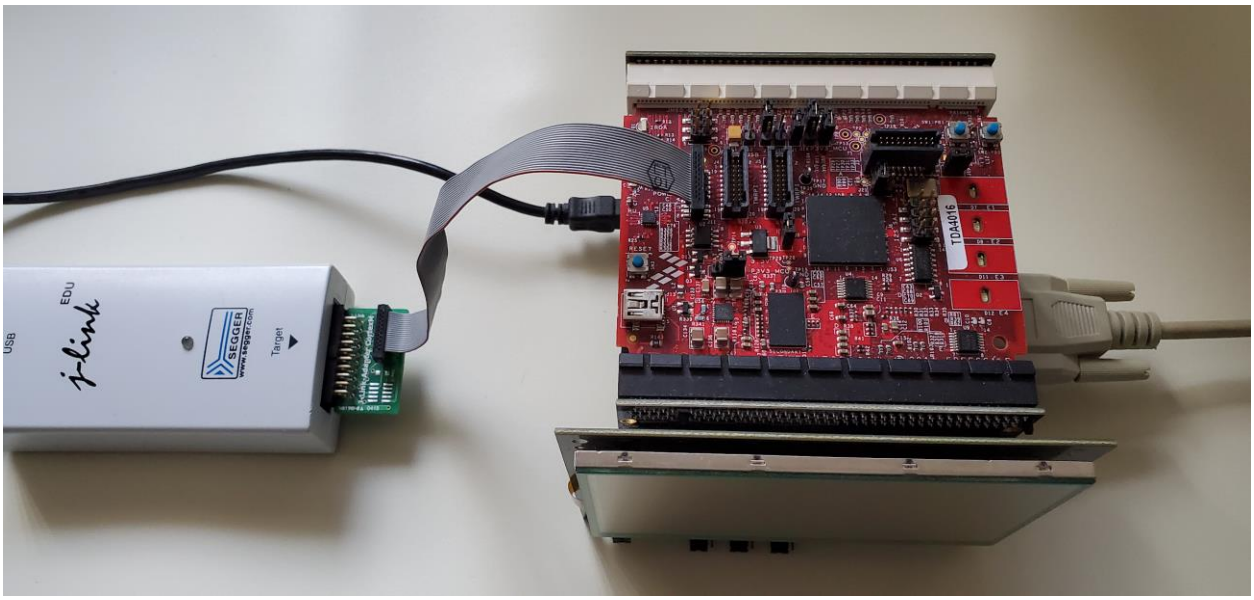
Setting up the NXP K70 with Kinetis Design Studio (KDS)

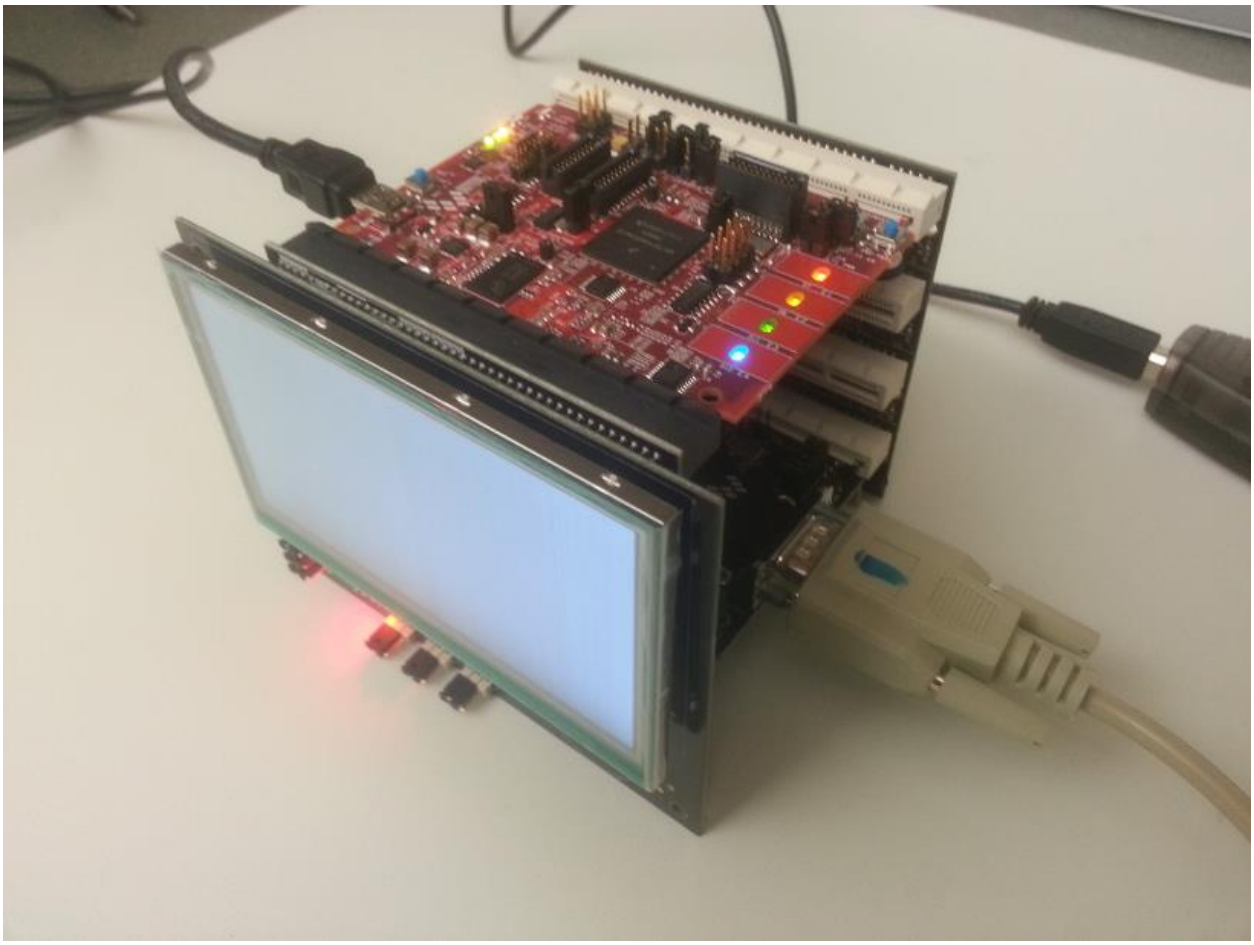
This document describes how to configure your K70 device on Windows 10, Mac OS X, and Linux. For Linux, this guide has instructions for both Fedora and Ubuntu, however much of this information should carry over to other distributions, with some exceptions such as package names and commands.

Connect the K70 to your Computer

Connect your hardware to your computer as per the directions on the course web site.



K70 using SEGGER



K70 using OSBDM

KDS Capabilities

KDS does not have all of the capabilities that were available in CodeWarrior. In particular, KDS does not support input through semihosting. This means that it is not possible to perform input under KDS using the C Programming Language standard I/O input from stdin. If you intend to install KDS under Windows, you might prefer using CodeWarrior so that you have the ability to perform standard I/O input.

Installing KDS

Download the Software

You may need to register with NXP in order to download the software.

https://www.nxp.com/design/designs/design-studio-integrated-development-environment-ide:KDS_IDE?tid=vanKDS#downloads

Download the software applicable to your operating system.

Windows 10

Installation Dialog Windows

Do you want to allow this app to make changes to your device?

Yes

Welcome to the Kinetis Design Studio Setup Wizard

Next >

Custom Setup: Click on the icons in the tree below to change the way features will be installed.

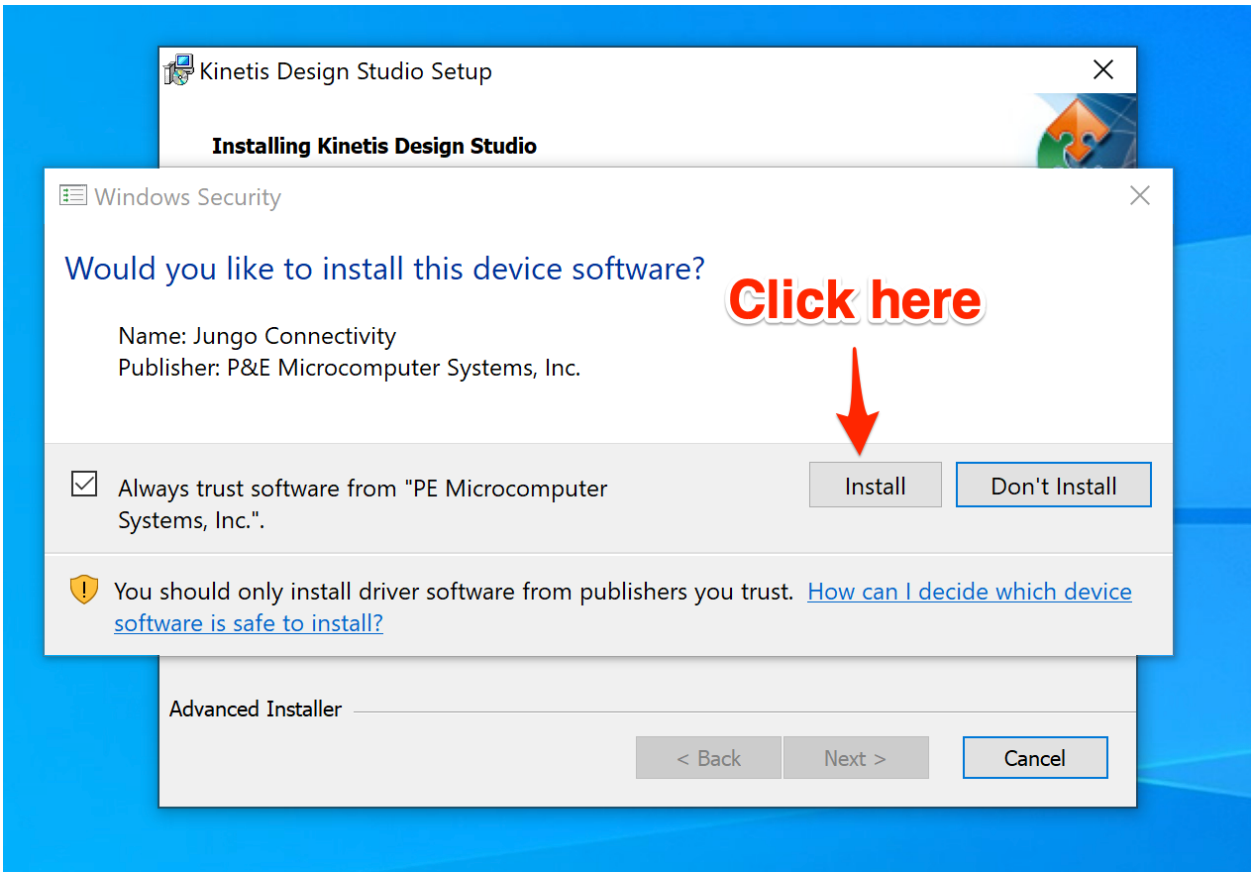
Install all features (Kinetis Design Studio: Entire feature will be installed on local hard drive). (This should be the default selection)
Next >

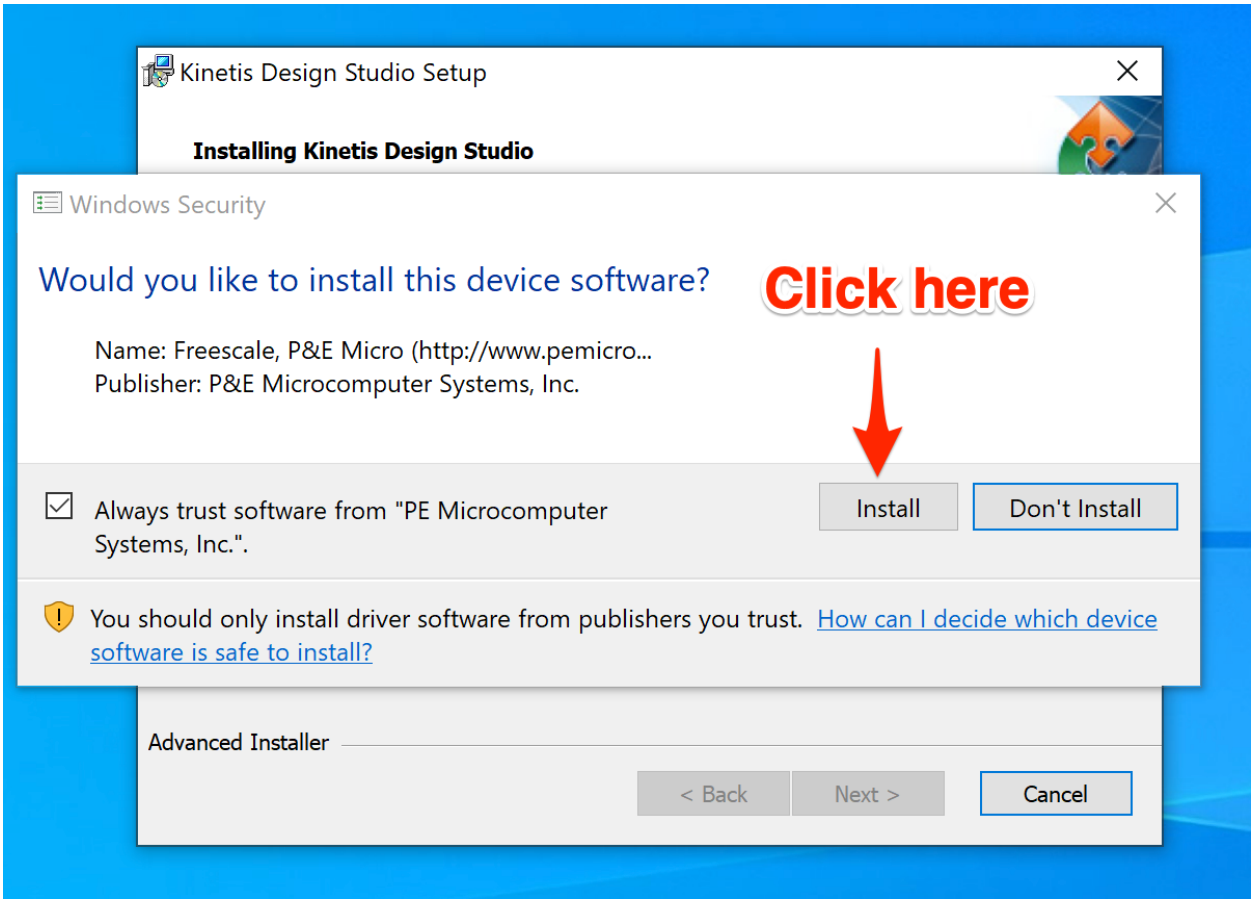
Configure Shortcuts

Create shortcuts for Kinetis Design Studio in the following locations: Desktop
Leave checked or uncheck as you wish.
Next >

Ready to Install

Install


Kinetis Design Studio Setup
Installing Kinetis Design Studio
Windows Security
Would you like to install this device software?
Name: Jungo Connectivity
Publisher: P&E Microcomputer Systems, Inc.
Click here
Always trust software from "PE Microcomputer Systems, Inc.".
Install
Don't Install
You should only install driver software from publishers you trust. How can I decide which device software is safe to install?
Advanced Installer
< Back
Next >
Cancel



Completing the Kinetis Design Studio Setup Wizard

Finish

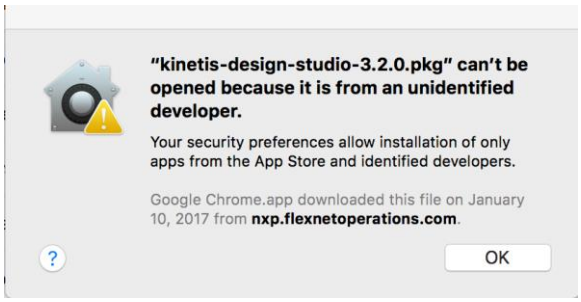
Mac

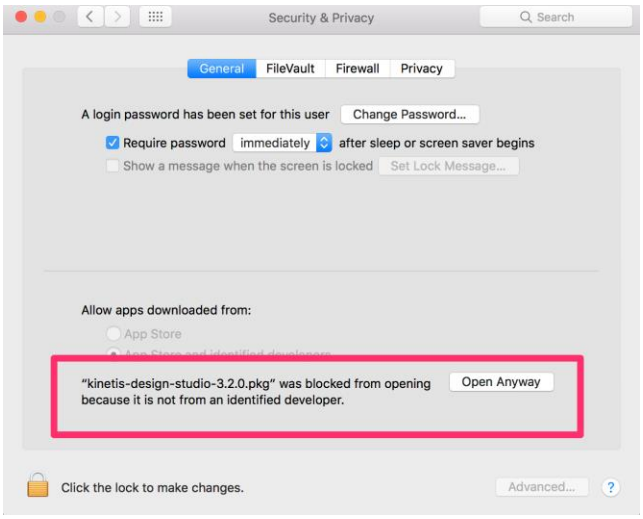
Issues with Newer MacOS Versions

We have reports that KDS may not work under new MacOS versions including (Big Sur 11.1). If you are experiencing these issues, you might consider installing KDS under a VM with another OS.

Security Exception

You may get an error in OS X that the installer cannot run because it's not digitally signed.





If you get this error, then you can allow a one time exception to install this software in the Security and Privacy System Preferences panel as shown above.

Run KDS_v3.app

The installer should have placed the program in your Applications folder. Simply double click on it.

Linux

Dependencies

Some utilities shipped with KDS are 32-bit and require additional libraries to be installed on a 64-bit operating system. On Fedora, you can run this command to install the correct package requirements:

```
sudo dnf install -y glibc.i686 ncurses-compat-libs.i686
```

On Ubuntu, you can run this command:

```
sudo apt-get install  libc6:i386 libncurses5:i386 libstdc++6:i386
```

Install the KDS Software

There are RPM and DEB packages available. Download the one that's applicable to your distribution of Linux. For Fedora, it would be the RPM. Once you've downloaded it, install it with this command:

```
sudo dnf install -y kinetis-design-studio-3.2.0-1.x86_64.rpm
```

On Ubuntu, you can run this command:

```
sudo dpkg -i kinetis-design-studio_3.2.0-1_amd64.deb
```

Finally, on Fedora, you need to have your user as a member of the 'dialout' group. This provides you access to the devices that are used for serial communication with your tower:

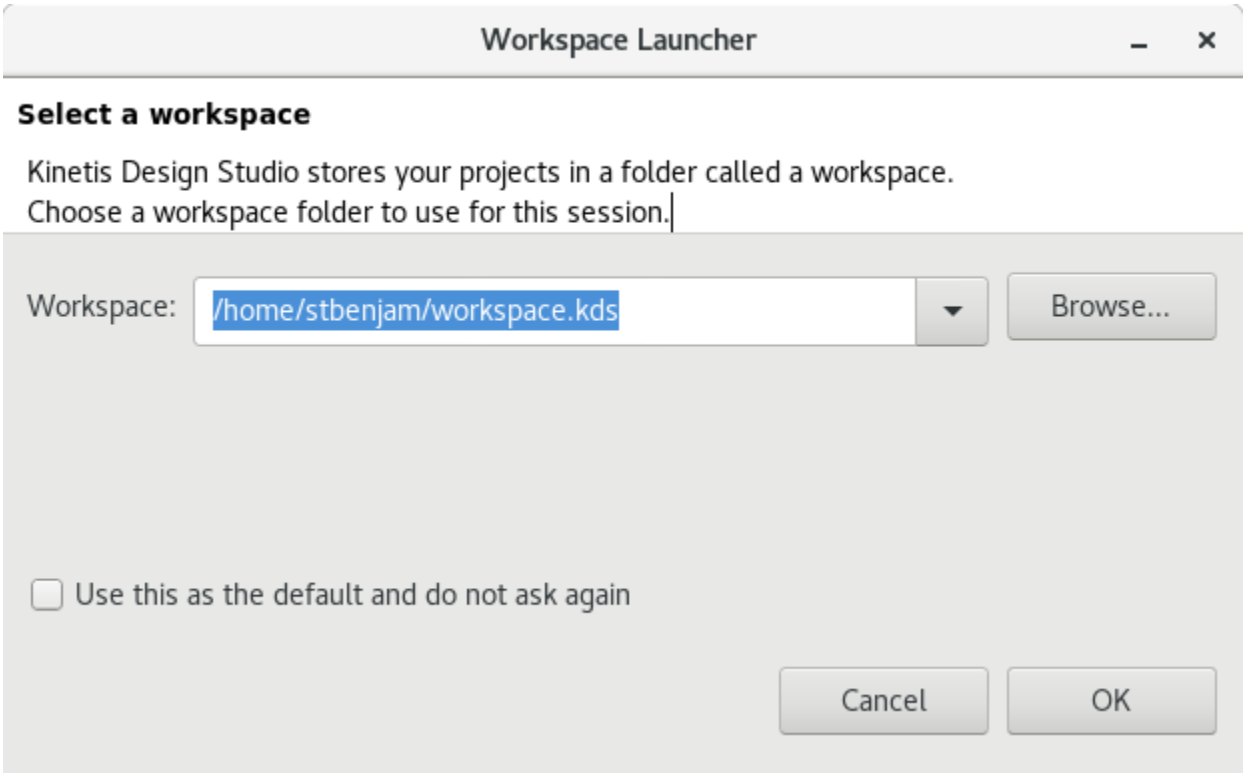
```
sudo usermod -a -G dialout $USERNAME
```

After installation, you should have a new Kinetis Design Studio available in your application launcher.

Create your first project

Create the workspace

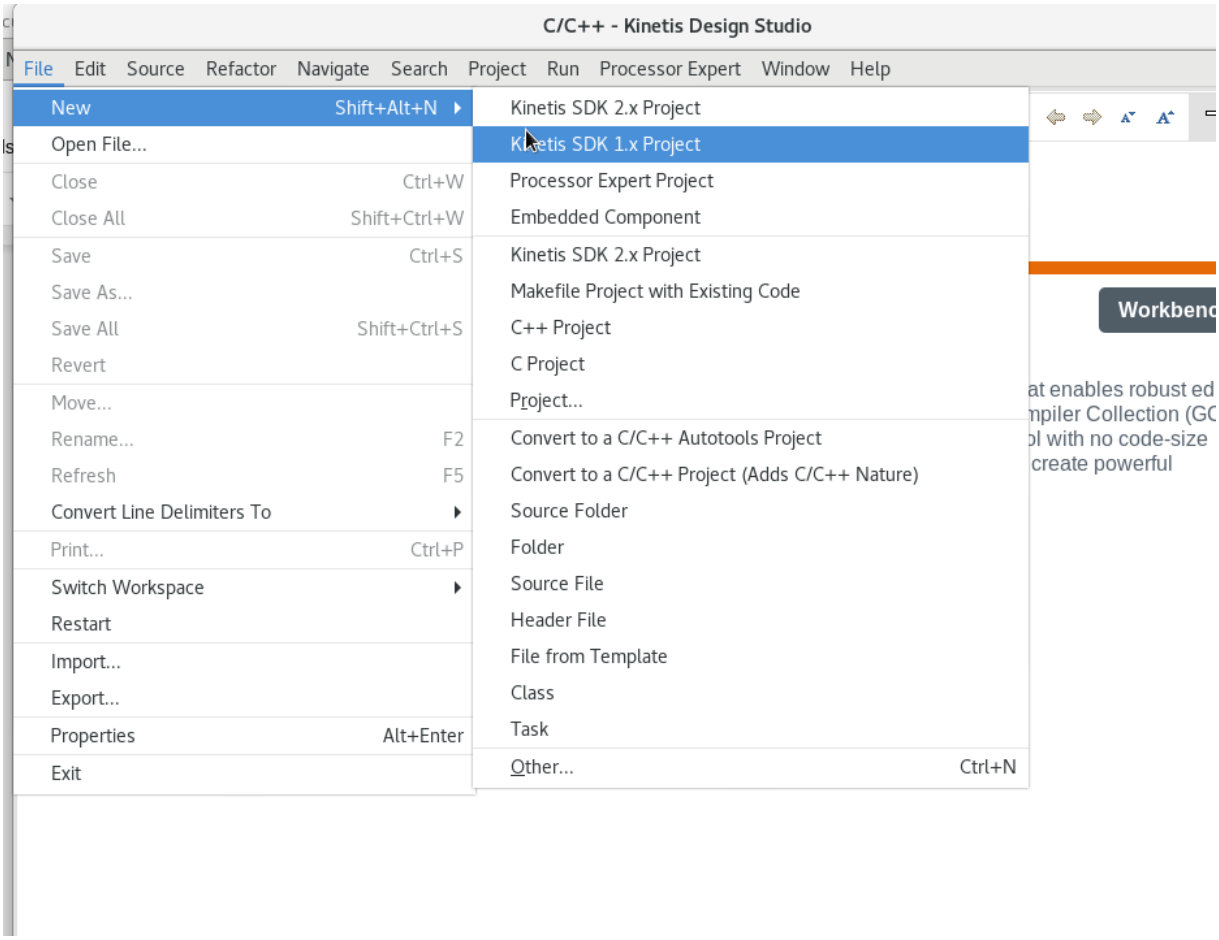
After launching KDS, you'll be prompted to select your workspace. You should choose a folder within your git repo under the src folder or plan on adding your workspace to your git repo later.



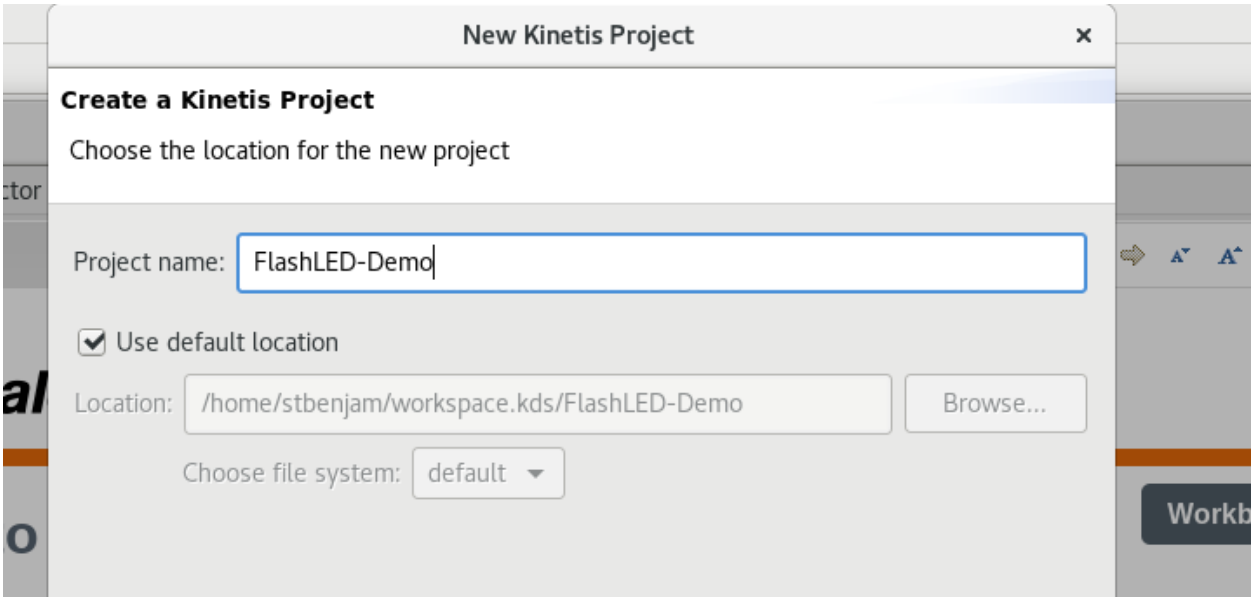
If you wish, select "Use this as the default and do not ask again."

Create a project

Select the type of project, which should be a Kinetis SDK 1.x Project:

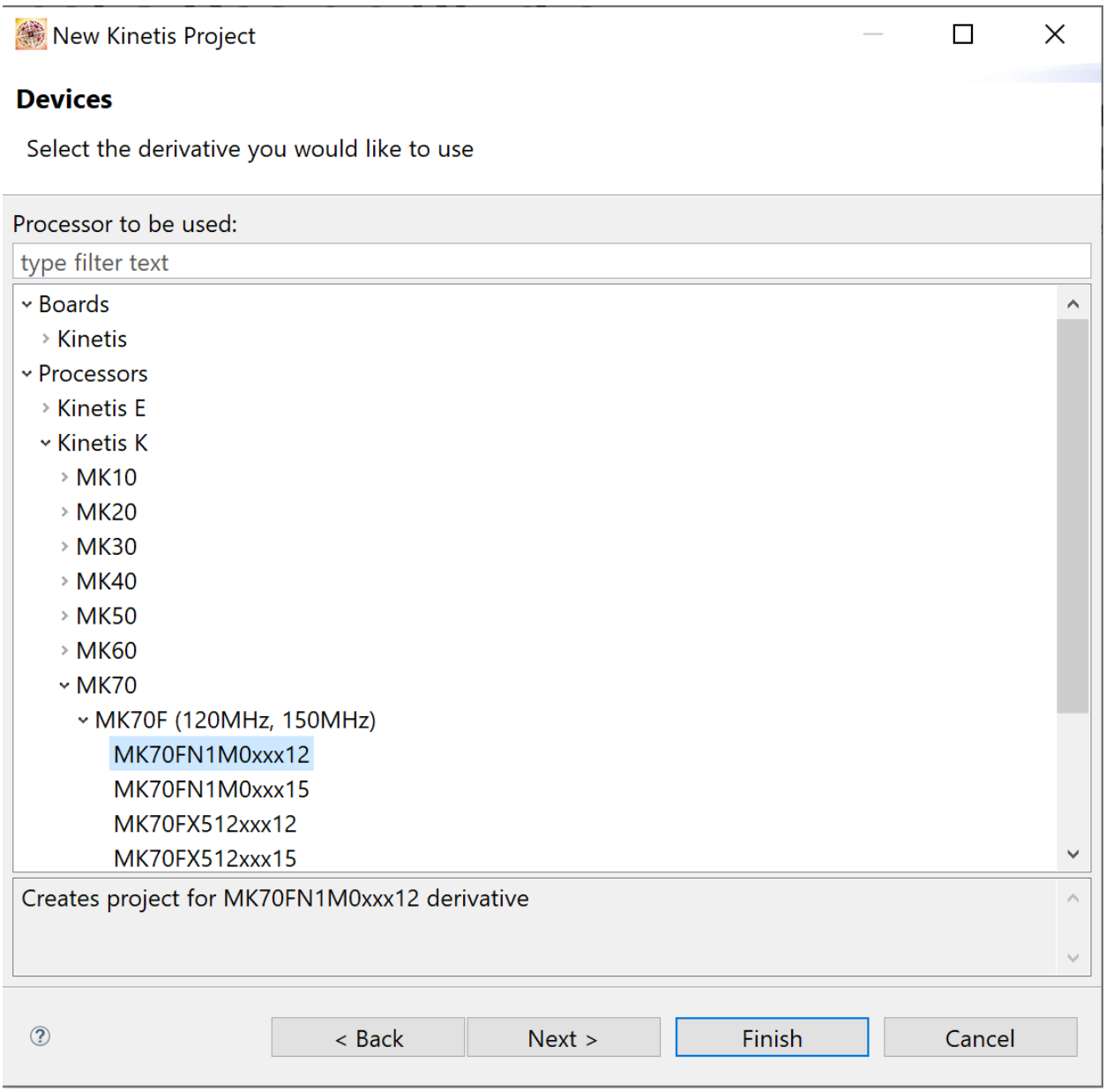


Give the project a name.



Select the processor type

Our Board is not listed so you need to select the correct processor from the Processors list. Expand the family option under Kinetis K and then select the one shown below.

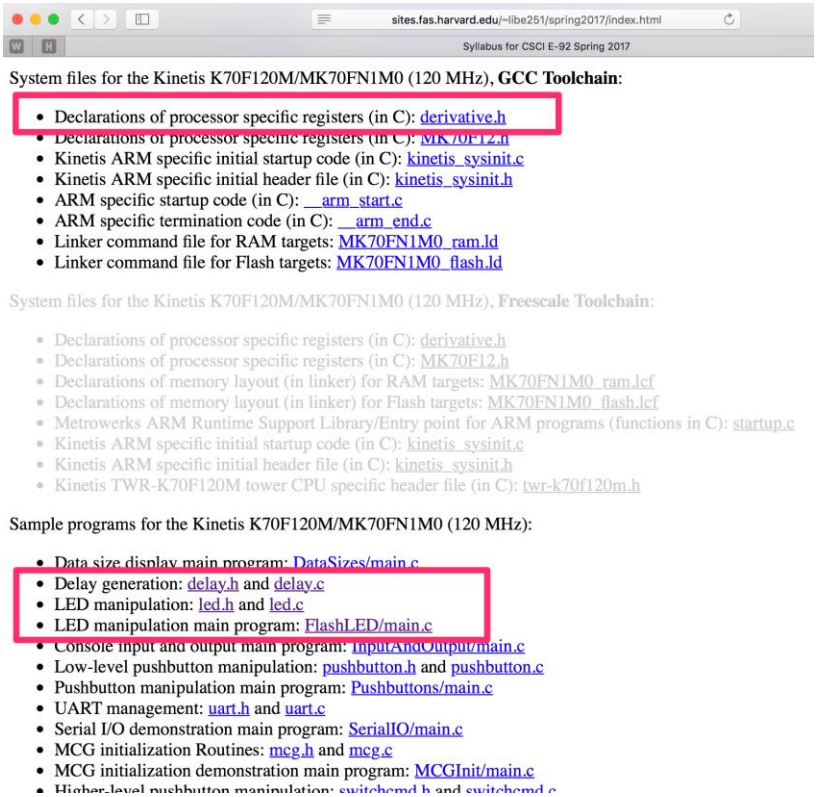


Create the .h and .c files in your project

From the course website, locate some of the sample files in order to get a simple demo of the LED's working. Download the highlighted files in the image below and add them to your project.

You need to download the following:

- derivative.h
- delay.h
- delay.c
- led.h
- led.c
- FlashLED/main.c



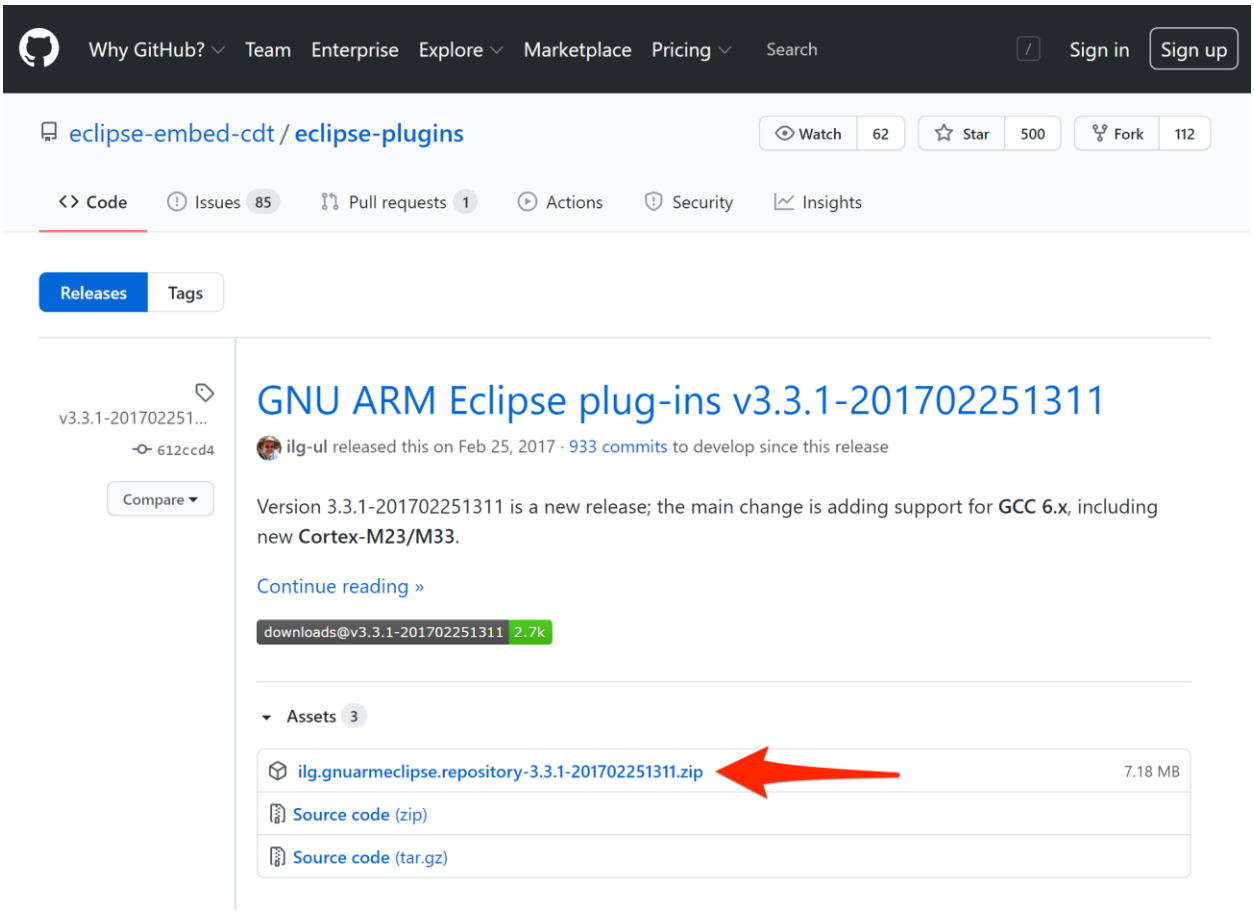
You should put all of the `.h` files in the Includes directory, and all of the `.c` files in the Sources directory.

Update ARM Build Tool Chain

You can extend the SRAM heap size (the size available to system malloc) in KDS. In order to do that, you must upgrade the toolchain to 3.3.1 and configure the heap and stack settings. You only need to update this once. **On Linux, this must be done as the root user.**

Upgrading the Tool Chain

1. Download version 3.3.1 from the GNU ARM Eclipse site on GitHub. Download the 7 MB zip file, not either of the source code links. Do not download version that is newer than 3.3.1.

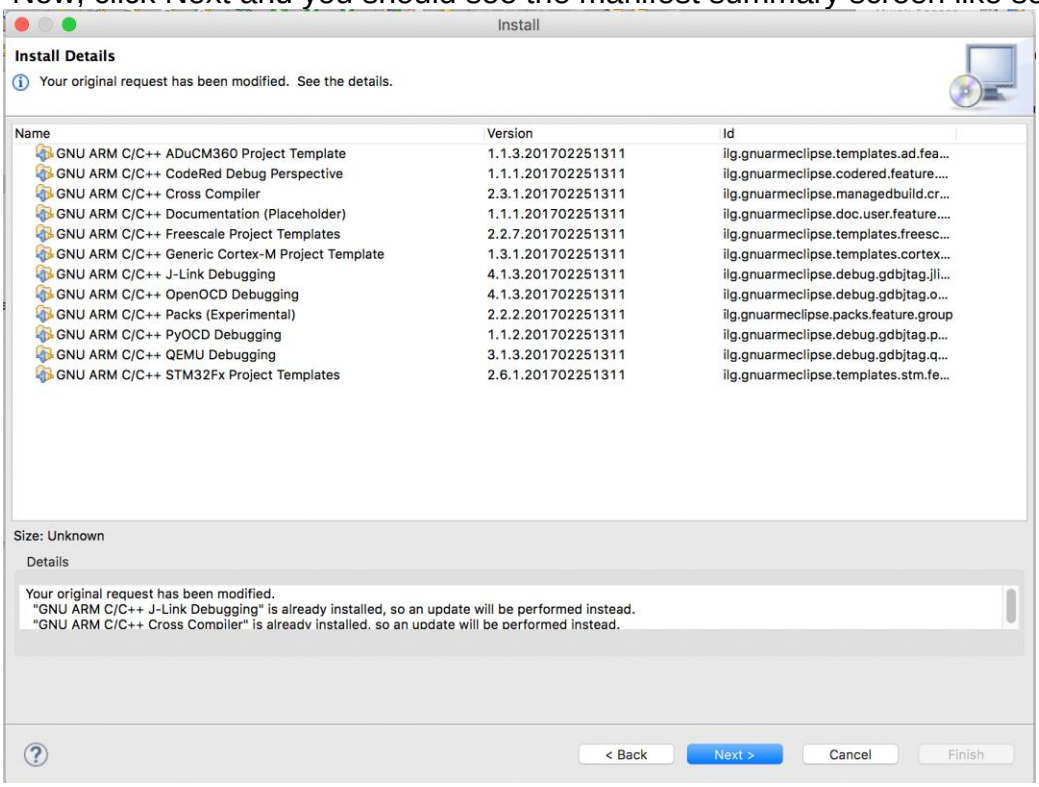


2. On Linux only, you need to log in to a graphical session using your 'root' user, or from a Terminal window run:

```
sudo /opt/Freescale/KDS_v3/eclipse/kinetis-design-studio
```

3. Select to Help / Install New Software... in KDS
4. Select the archive you downloaded for installation by checking "GNU ARM C/C++ Cross Development Tools"

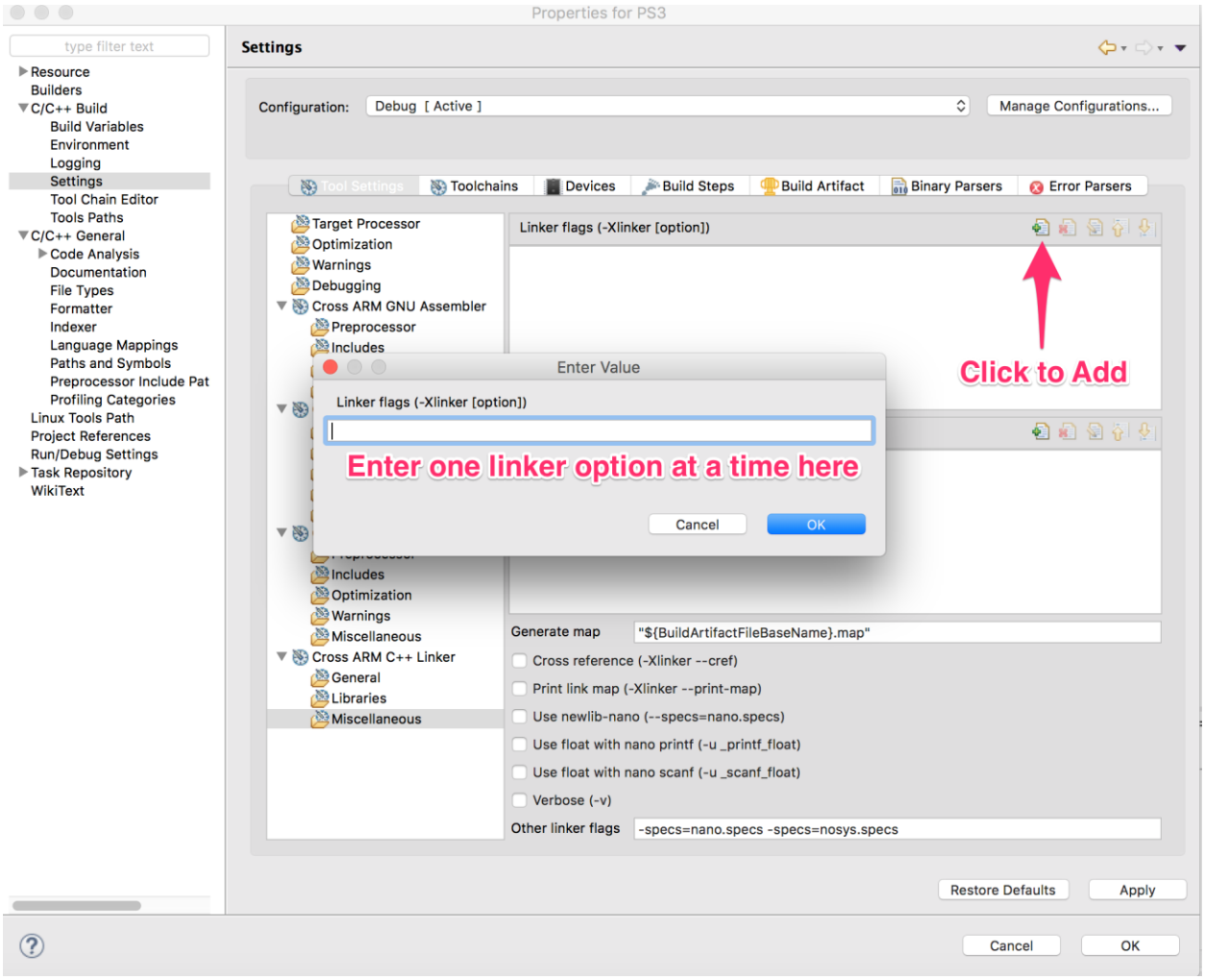
5. Now, click Next and you should see the manifest summary screen like so:

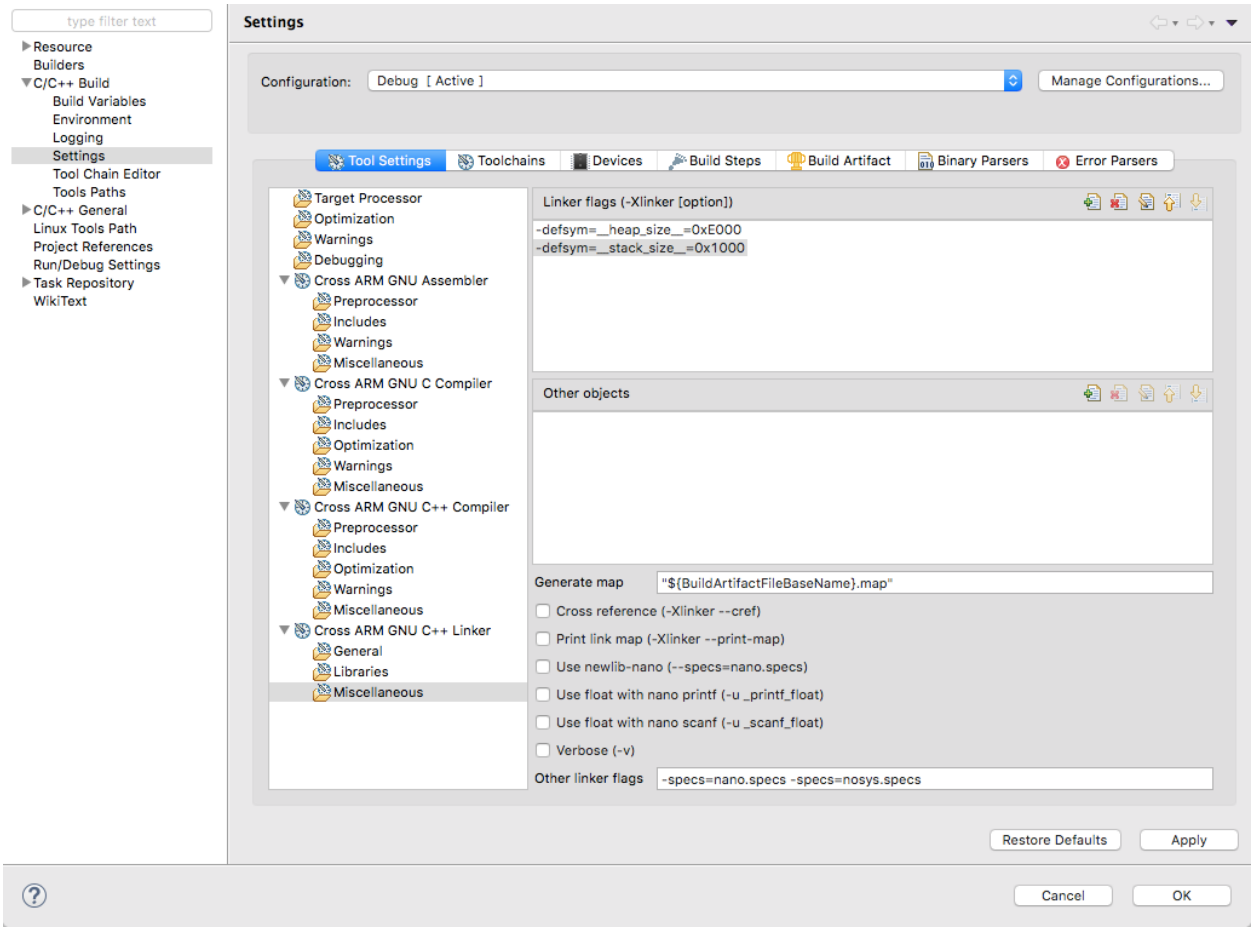


6. Click Next to proceed with the installation.
7. If you see an “Install Remediation Page,” allow your installation to be updated to be compatible with the items being installed.
8. Accept the licenses for the files, and let Eclipse update the software.
9. Restart Kinetis Design Studio if requested to do so. If you are on Linux, restart, but remember to run KDS as your normal user after upgrading the software.

Configuring the Heap, Stack, and Semihosting Options

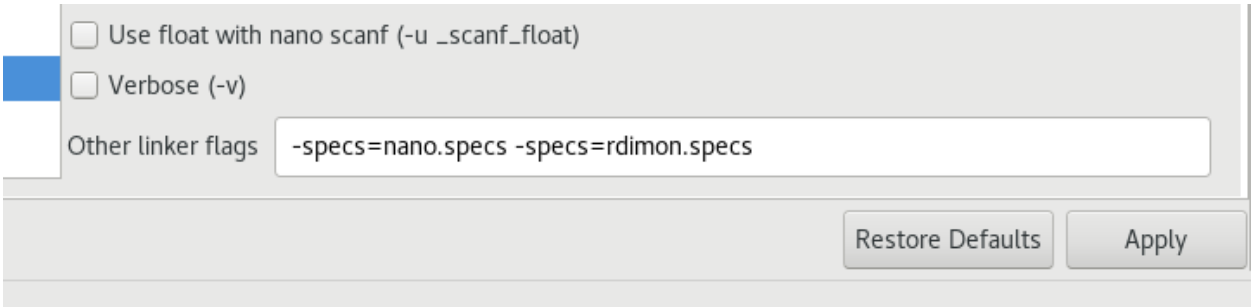
1. After restarting, select the appropriate project, ensure that the project is open, then navigate to Project / Properties, C/C++ Build, Settings and you should see a setting for linker options under the Cross ARM GNU C++ Linker’s Miscellaneous tab
2. Add two entries for linker options where you set the heap and stack size like so:
    a. -defsym=__heap_size__=0xE000
    b. -defsym=__stack_size__=0x1000


Properties for PS3
type filter text
Resource
Builders
C/C++ Build
Build Variables
Environment
Logging
Settings
Tool Chain Editor
Tools Paths
C/C++ General
Code Analysis
Documentation
File Types
Formatter
Indexer
Language Mappings
Paths and Symbols
Preprocessor Include Pat
Profiling Categories
Linux Tools Path
Project References
Run/Debug Settings
Task Repository
WikiText
Settings
Configuration: Debug [Active]
Manage Configurations...
Tool Settings
Toolchains
Devices
Build Steps
Build Artifact
Binary Parsers
Error Parsers
Target Processor
Optimization
Warnings
Debugging
Cross ARM GNU Assembler
Preprocessor
Includes
Linker flags (-Xlinker [option])
Click to Add
Enter Value
Linker flags (-Xlinker [option])
Enter one linker option at a time here
Cancel
OK
Includes
Optimization
Warnings
Miscellaneous
Cross ARM C++ Linker
General
Libraries
Miscellaneous
Generate map
"${BuildArtifactFileBaseName}.map"
Cross reference (-Xlinker --cref)
Print link map (-Xlinker --print-map)
Use newlib-nano (--specs=nano.specs)
Use float with nano printf (-u _printf_float)
Use float with nano scanf (-u _scanf_float)
Verbose (-v)
Other linker flags
-specs=nano.specs -specs=nosys.specs
Restore Defaults
Apply
Cancel
OK



3. In order to enable semihosting, scroll to the bottom where it says “Other linker flags” and REPLACE the “-specs=nosys.specs” option with “-specs=rdimon.specs” so that the entire line now reads “-specs=nano.specs -specs=rdimon.specs”

KDS does **NOT** support input while using semihosting.

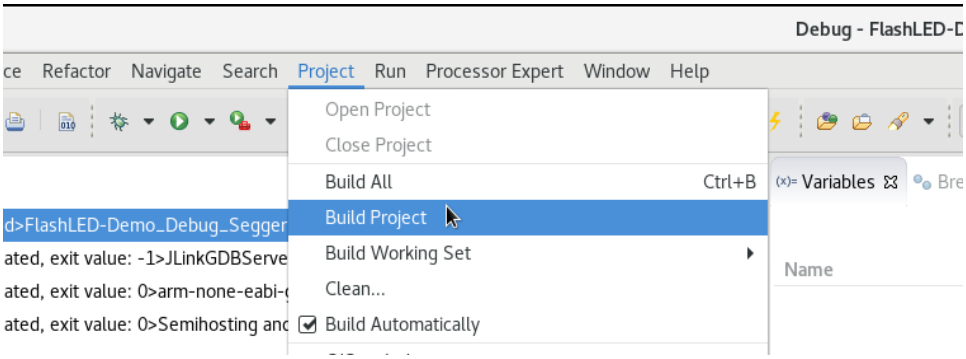


4. Keep in mind that the **linker flags and semihosting flags** need to be set as appropriate **for each project**.

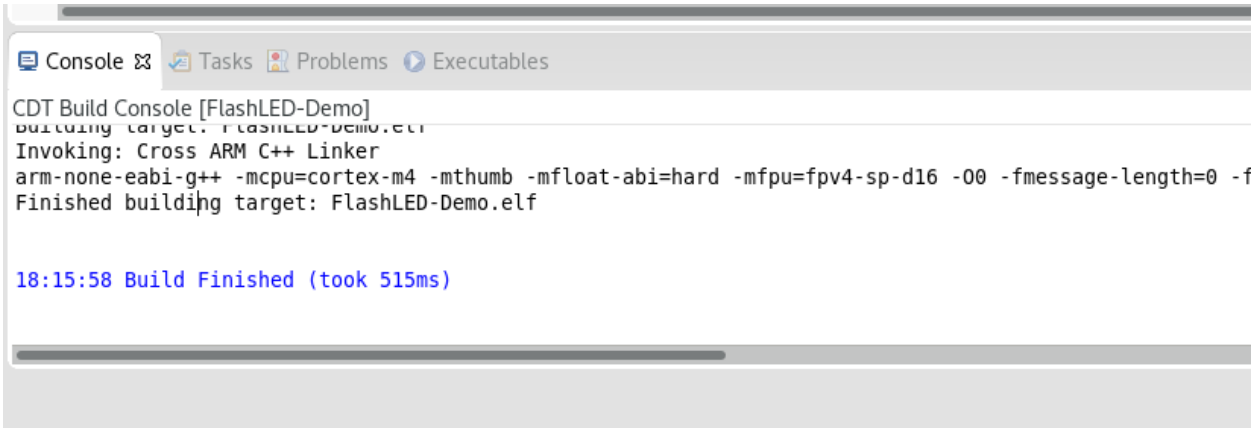
Running the Project

Build Your Project

First you need to build your program. Select Project, Build Project.



It should build successfully without any errors, and a message such as “Build Finished” should appear in the console.



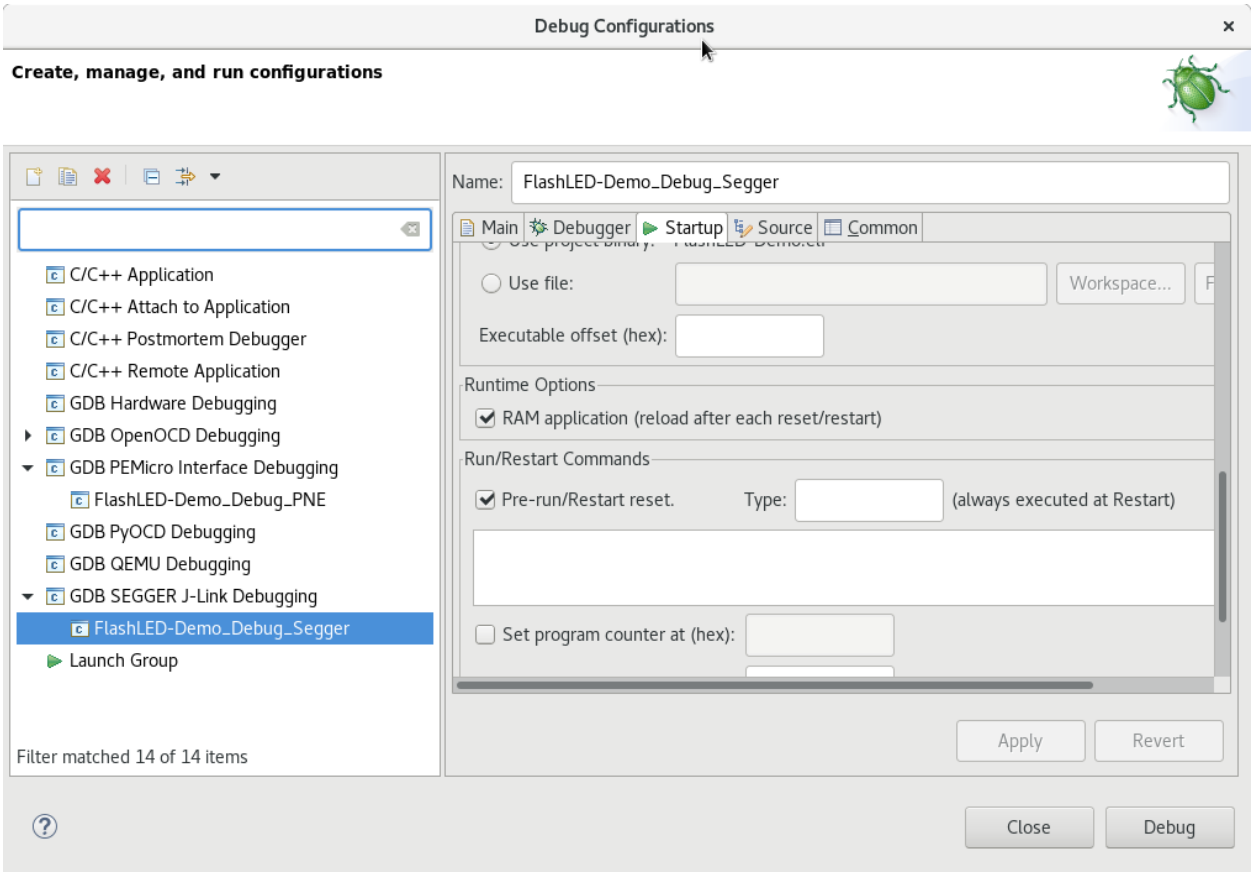
Running the project

For your first launch, you’ll need to create a launch configuration. Which configuration you use depends on if you are using the SEGGER, or OSBDM (listed as PEMicro).

SEGGER

If you are using the SEGGER, click on Run -> Debug Configurations..., then expand “GDB SEGGER” and click the first entry.

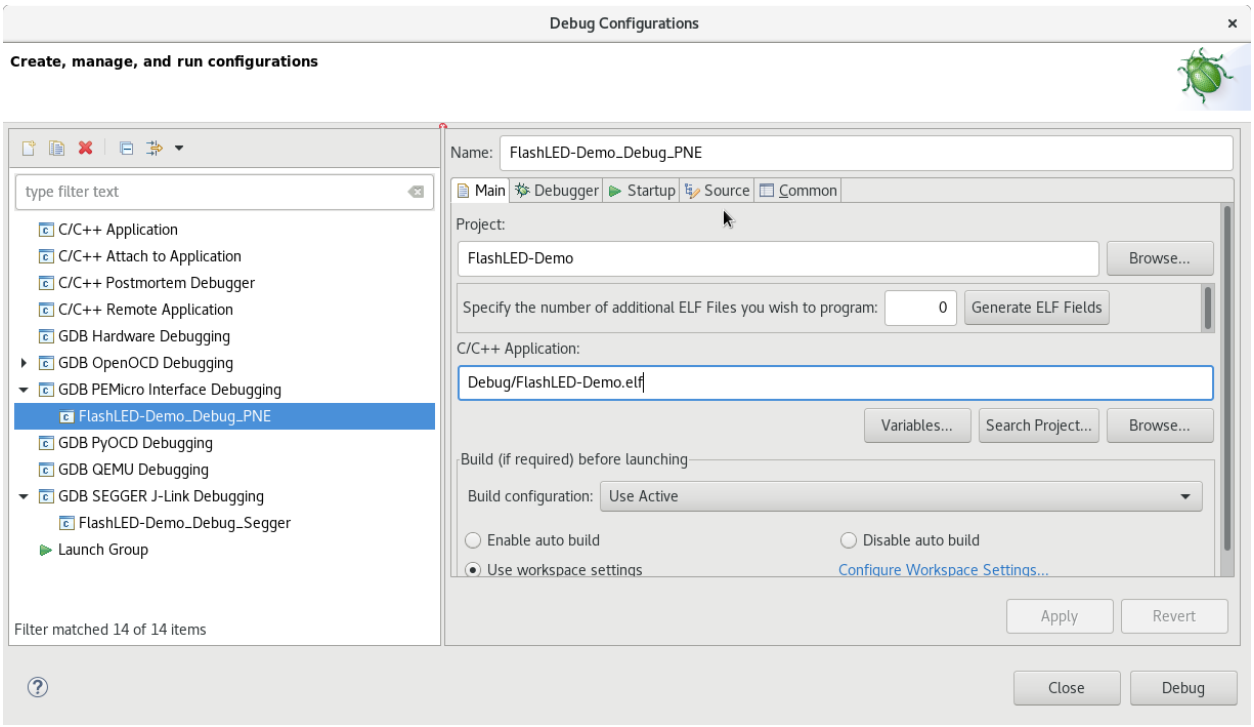
On the Startup tab, under Runtime Options, click “RAM Application.” Then click the debug button. For future launches, you can directly click the SEGGER option under the debug menu.



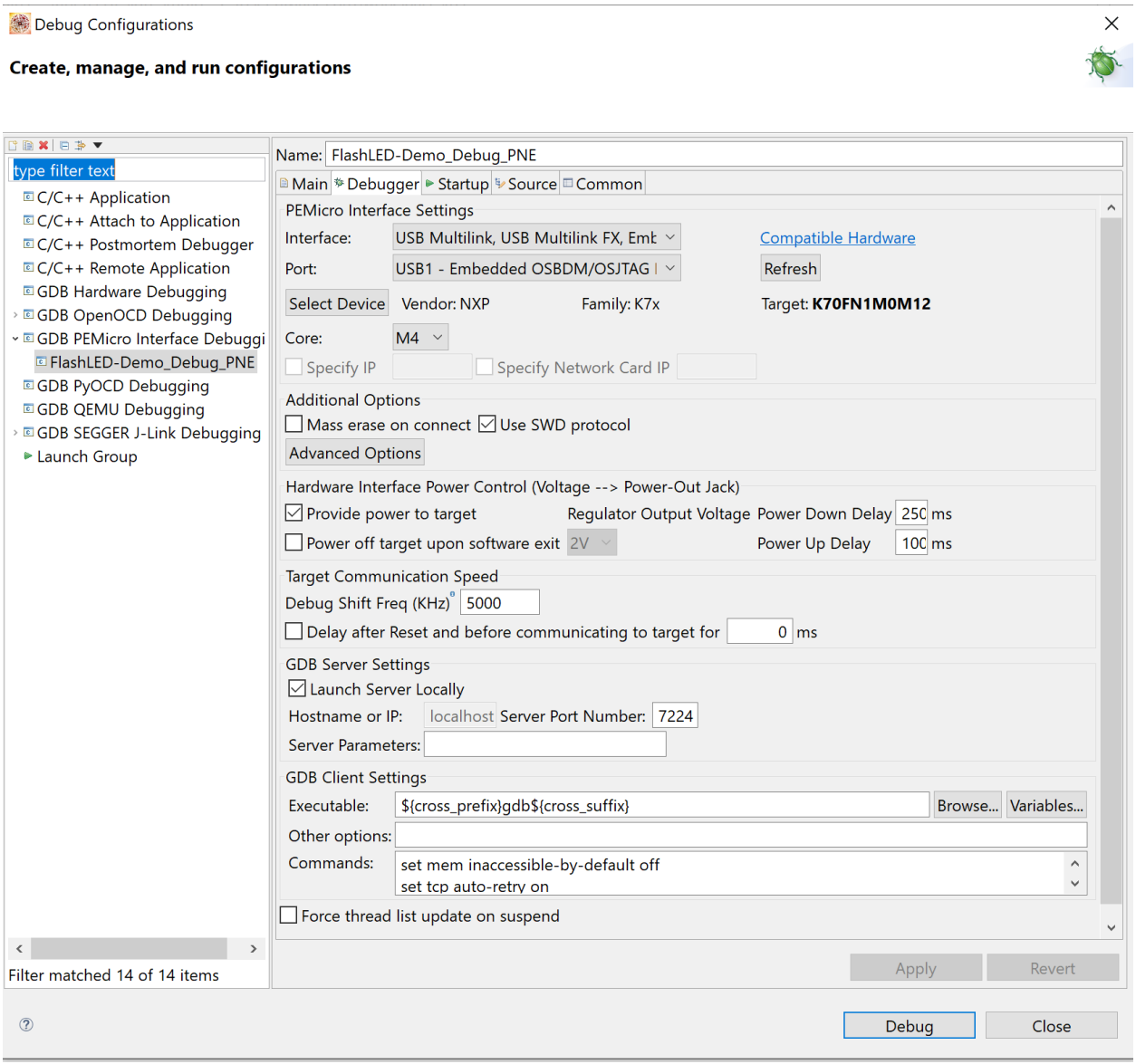
OSBDM

If you are using OSBDM, click on Run -> Debug Configurations..., then expand “GDB PEMicro Interface Debugging” menu and click on FlashLED-Demo_Debug_PNE.

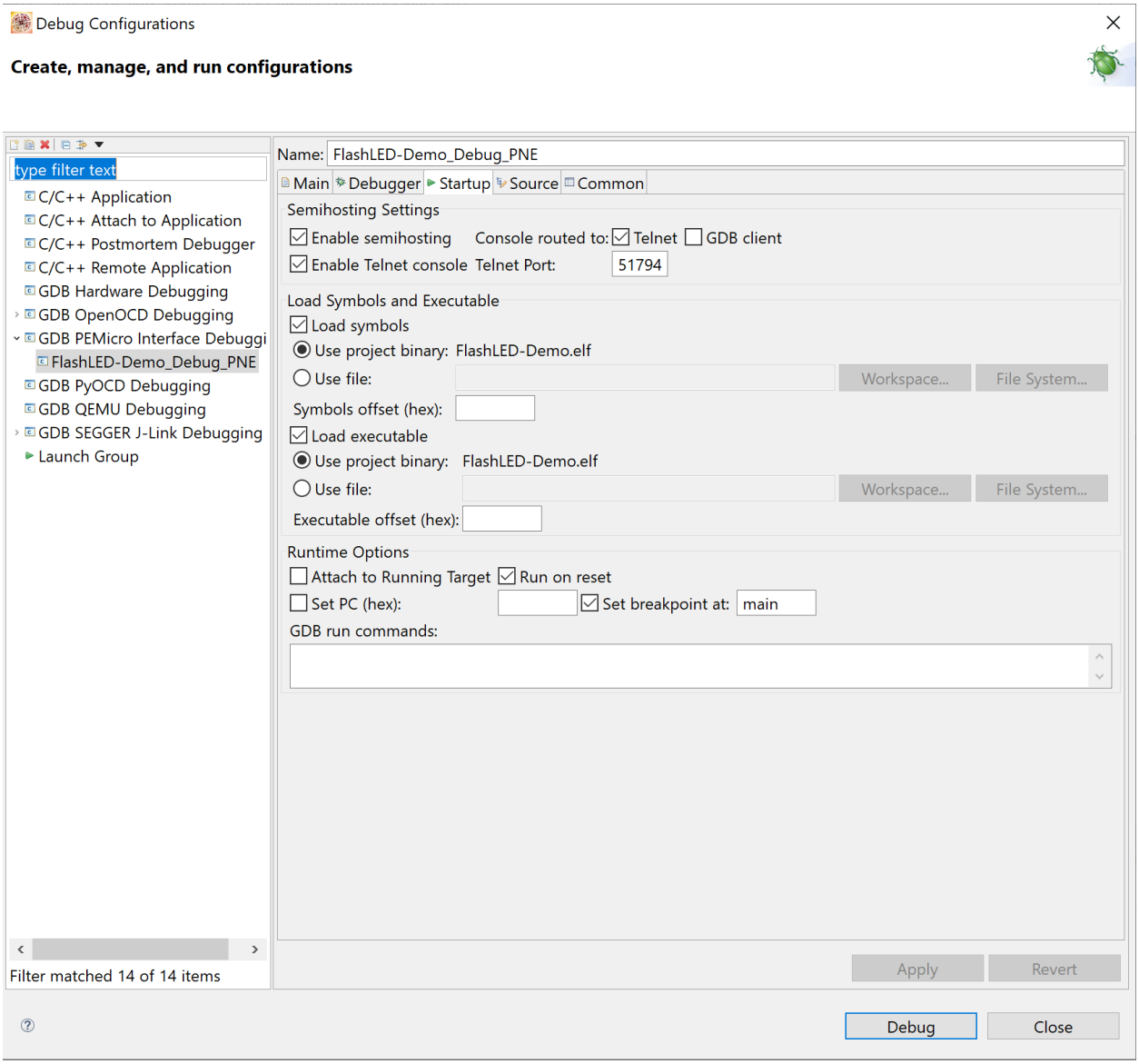
Verify that the Main, Debugger, and Startup tabs match



Debugger Tab


Debug Configurations
Create, manage, and run configurations
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
GDB Hardware Debugging
GDB OpenOCD Debugging
GDB PEMicro Interface Debuggi
FlashLED-Demo_Debug_PNE
GDB PyOCD Debugging
GDB QEMU Debugging
GDB SEGGER J-Link Debugging
Launch Group
Filter matched 14 of 14 items
Name: FlashLED-Demo_Debug_PNE
Main
Debugger
Startup
Source
Common
PEMicro Interface Settings
Interface:
USB Multilink, USB Multilink FX, Emb
Compatible Hardware
Port:
USB1 - Embedded OSBDM/OSJTAG
Refresh
Select Device
Vendor: NXP
Family: K7x
Target: K70FN1M0M12
Core:
M4
Specify IP
Specify Network Card IP
Additional Options
Mass erase on connect
Use SWD protocol
Advanced Options
Hardware Interface Power Control (Voltage --> Power-Out Jack)
Provide power to target
Regulator Output Voltage
Power Down Delay
250
ms
Power off target upon software exit
2V
Power Up Delay
100
ms
Target Communication Speed
Debug Shift Freq (KHz)
5000
Delay after Reset and before communicating to target for
0
ms
GDB Server Settings
Launch Server Locally
Hostname or IP:
localhost
Server Port Number:
7224
Server Parameters:
GDB Client Settings
Executable:
${cross_prefix}gdb${cross_suffix}
Browse...
Variables...
Other options:
Commands:
set mem inaccessible-by-default off
set tcp auto-retry on
Force thread list update on suspend
Apply
Revert
Debug
Close

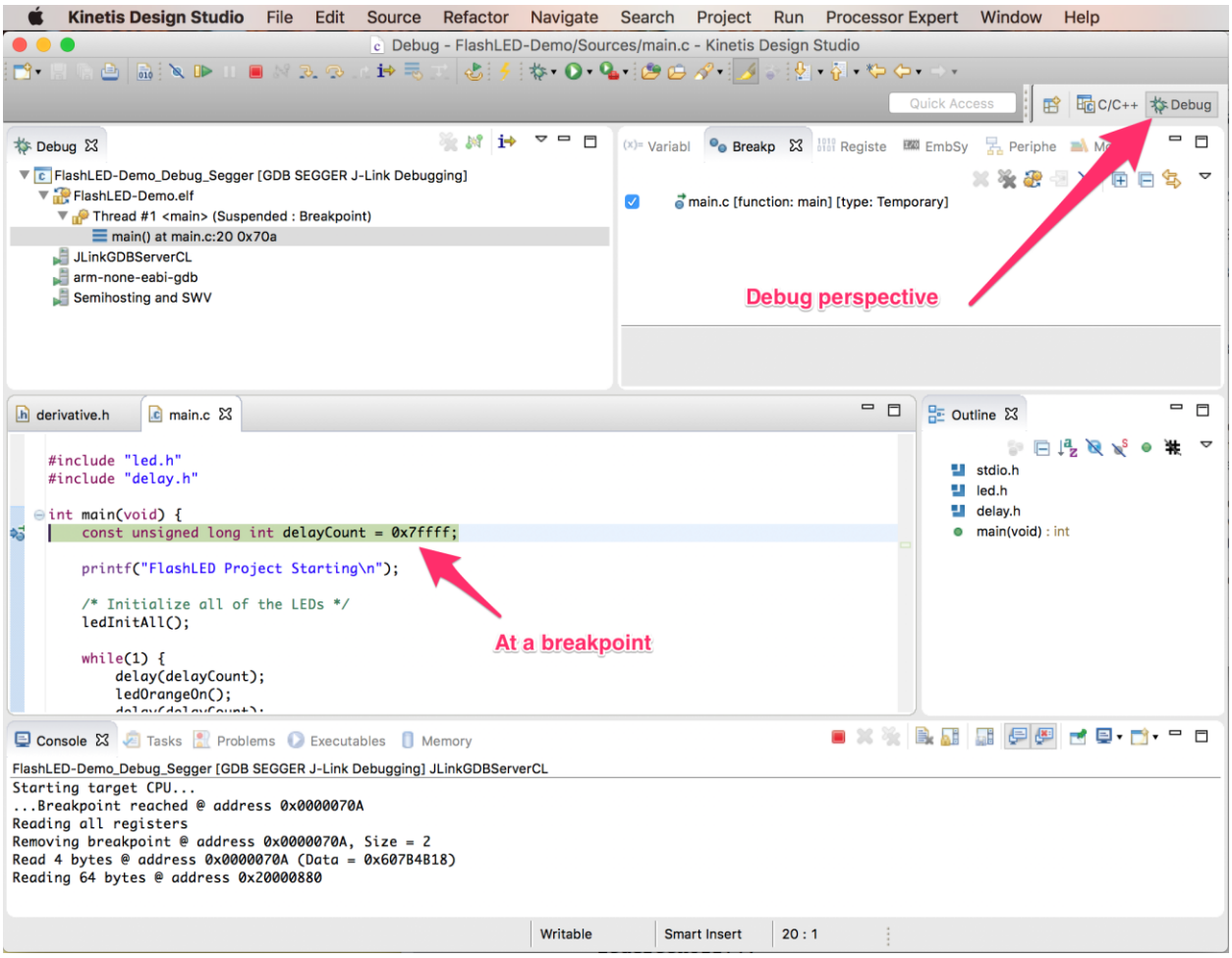
Startup Tab



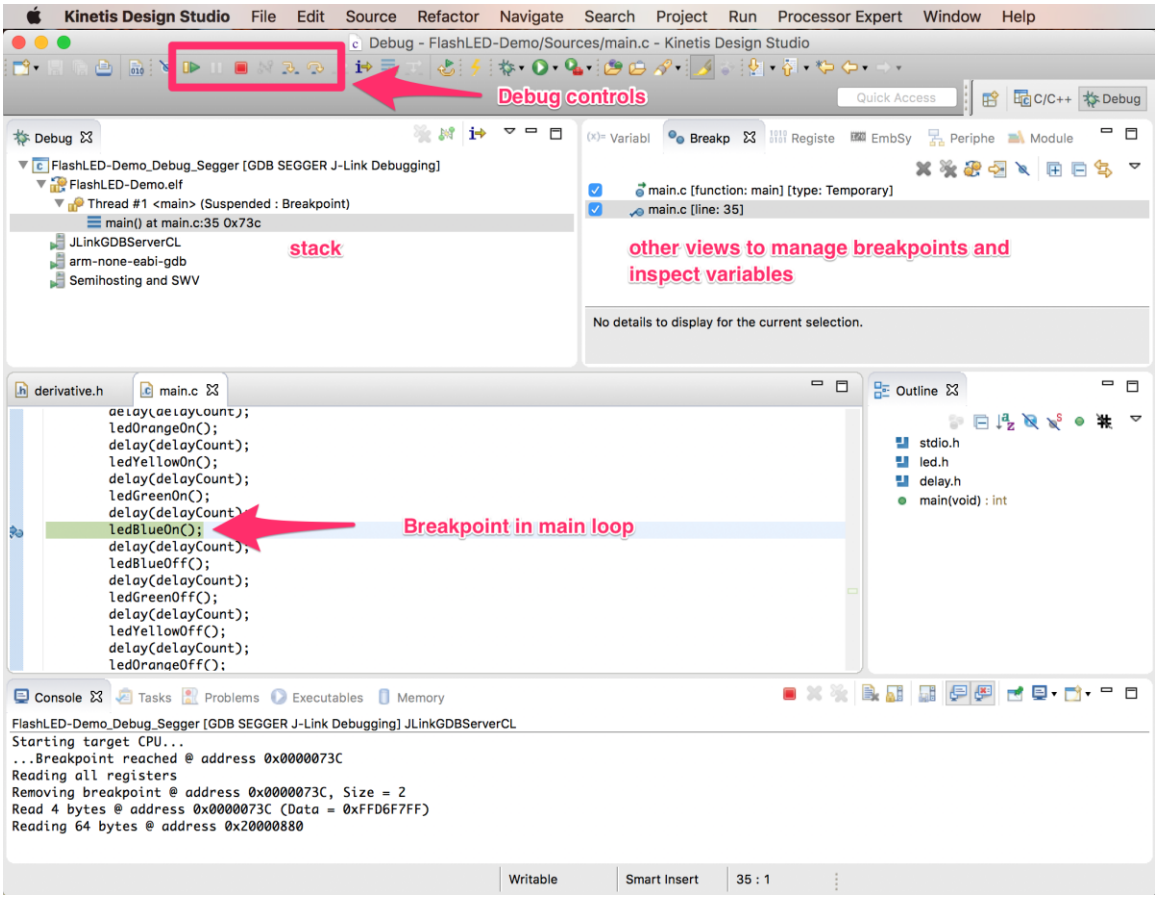
If attempting to run a program through the Run -> Debug menu selection (or with the F11 key shortcut), you may see a “Launch Configuration Selection” window. If this window appears, select the “FlashLED-Demo_Debug_PNE” entry and click on OK.

Using the debug perspective

You should see a Launch Dialog while the program is loaded onto the hardware. You may also get a prompt for switching into the Debug perspective. Perspectives in KDS (and Eclipse, which is also used by CodeWarrior) are simply a preconfigured arrangement of toolbars and panels.



Every time you launch the debug configuration, you'll need to click the green "play" button to start the program in the debug controls.



Possible Pop-Up Window About Old OSJTAG/OSBDM firmware

You may see a pop-up window labelled “Confirm” that says “Old OSJTAG/OSBDM firmware has been detected. The embedded firmware needs to be in bootloader mode to update. Please unplug the USB cable, insert a jumper on the 2-pin bootloader header (connecting JM60 IRQ to ground), and reconnect the USB cable.”

If you see this message, please follow the directions. First, UNPLUG THE USB CABLE FROM YOUR COMPUTER. A header is a component that has pins that stick up from it to allow connections. This 2-pin bootloader header is labelled J10 and is found on the upper side of the red, uppermost K70 processor PCB (printed circuit board) near the primary elevator (the primary elevator has white connectors -- rather than black connectors) in a cluster with other headers. This cluster of 2-pin headers is comprised of headers labelled J12, J2, J10, J1, J16, and J20. The jumper referred to in the directions is a small black plastic assembly with a metal connector inside. There is a currently-unused jumper installed to one pin of header J2. Remove the jumper from J2 and install it on header J10 so that it slides onto BOTH pins of header J10.

After installing the jumper on J10, reconnect the USB connector to your computer. Click OK in the “Confirm” window and wait for the “Updating firmware of P&E Interface…” to complete. You

will now see a pop-up window labelled "Confirm" that says “The embedded OSJTAG/OSBDM needs to enter run mode to start the debug/programming session. Please unplug the USB cable, remove the jumper from the 2-pin bootloader header, and reconnect the USB cable.”

Now, once again UNPLUG THE USB CABLE FROM YOUR COMPUTER. Remove the jumper from J10 and install it on only the upper pin of header J2 -- in the same position where it was originally.

After removing the jumper from J10, reconnect the USB connector to your computer. Click OK in the “Confirm” window and continue.

A note on .gitignore

There are a lot of metadata files and settings that you shouldn’t add to your git repo. These should be added to your .gitignore file so you don’t wind up polluting your merge requests with a ton of non-source code files.

For example, the workspace.kds directory at the end of this exercise will contain the following:

.metadata/
FlashLED-Demo/
RemoteSystemTempFiles/

You should add the following to your .gitignore file.

.metadata
RemoteSystemTempFiles
FlashLED-Demo/Debug
FlashLED-Demo/Project_Settings

This will ensure that only source code that you write and place in the Includes and Sources folder will make it into your repo.