

# PAC Eclipse Usage

*Power Application Controller*<sup>®</sup>



[www.active-semi.com](http://www.active-semi.com)  
Copyright © 2019 Active-Semi, Inc.

No portion of this document may be reproduced or reused in any form without Active-Semi's prior written consent

## Table of Contents

1 Introduction .....	4
2 Using Eclipse .....	4
2.1 Importing a Project .....	4
2.2 Building the Project.....	7
2.3 Debugging the Project.....	9
2.3.1 Add PAC55xx support to J-Flash.....	9
2.3.2 Setup Hardware .....	9
2.3.3 Special Note about Debugging, Downloading, and Erasing.....	9
2.3.4 Starting Debug .....	9
2.4 Erasing and Flashing.....	11
Sometimes it's desirable to Erase the device or Flash it without debugging. There are 2 options for Erasing and Flashing using J-Link. The first option uses the built in J-Link plugin, and the second option uses the external JFlash tool. Note that the JFlash tool requires the J-Link HW to be licensed for use with JFlash. ....	11
2.4.1 Erasing and Flashing with built-in J-Link Plugin .....	11
2.4.2 Erasing and Flashing with JFlash .....	14
2.5 Adding Files and Folders to the project.....	17
2.5.1 Adding Existing Files and Folders that are within the project .....	17
2.5.2 Adding Files and Folders that are outside the project folder .....	17
2.5.3 Use File Explorer to add a file that is outside the project folder .....	17
2.5.4 Use File Explorer to add folders/subfolders that are outside the project folder.....	18
2.6 Renaming a Project .....	19
3 Troubleshooting .....	19
3.1.1 make not found in PATH.....	19
3.1.2 Java Error messages .....	20
3.1.3 If sources.mk, objects.mk, or makefile are not found .....	21
3.1.4 .launch Configuration variable references empty selection .....	21
3.1.5 Debug or External Tool .launch files not available from Icon menu .....	21
3.1.6 jflash-jlink.launch Can't save jflash.ini .....	22
3.1.7 JFlash - Could not open project file .....	23
3.1.8 Debug session already started – Terminate the first one before restarting .....	24
4 Eclipse Tips .....	25
4.1 Eclipse Workspaces .....	25

4.2 Eclipse Paths .....	25
4.3 Project Build Settings .....	26
4.4 Save Modified Files Before Building.....	27
4.5 Change editor to insert spaces instead of tabs .....	27
4.6 Memory Browser vs Memory.....	28
4.7 Keep Memory or Memory Browser Window in focus .....	28
4.8 Accelerate build process .....	29
About Active-Semi <sup>®</sup> .....	29

## 1 INTRODUCTION

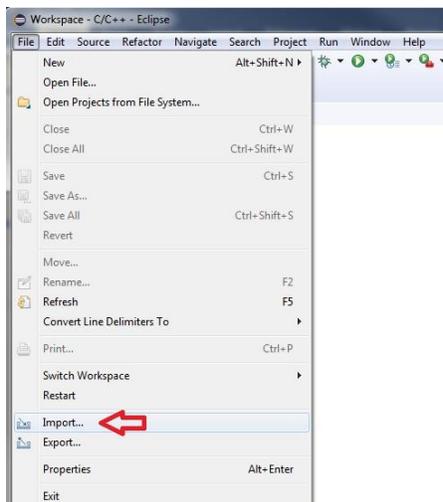
This document explains in more detail how to use Eclipse with PAC devices. Note that Eclipse and its plugins are also covered by the standard Eclipse documentation. If any issues are encountered, while using Eclipse, consult the Troubleshooting section 3 of this document for possible solutions, and if not found there, the internet is your friend, because the tools are all open source and community supported.

The rest of this document assumes that “Eclipse with PAC Support” has been installed and Eclipse has been started. The instructions will be provided for PAC55xx, and are nearly identical for PAC52xx. In most places, pac52xx can be substituted for pac55xx. Where there is a substantial difference, it will be noted.

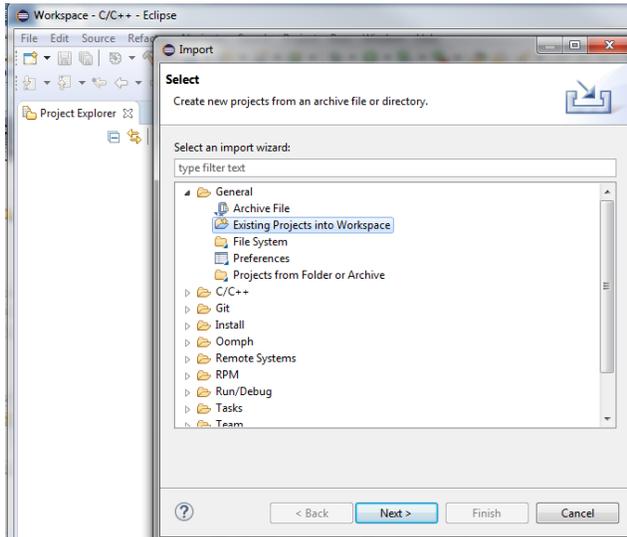
## 2 USING ECLIPSE

### 2.1 Importing a Project

Import a project into the workspace by selecting File->Import... as shown below:

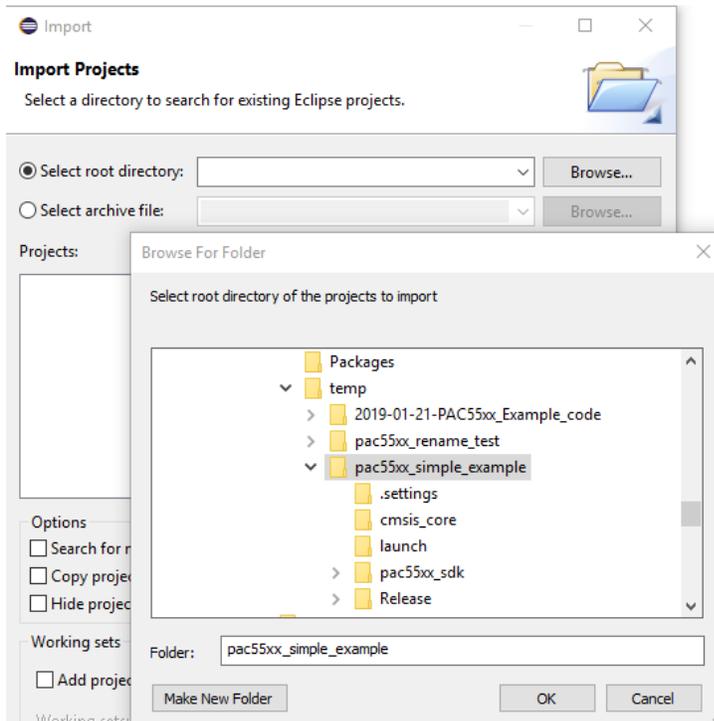


Then choose General->”Existing Projects into Workspace” and click next as shown below:

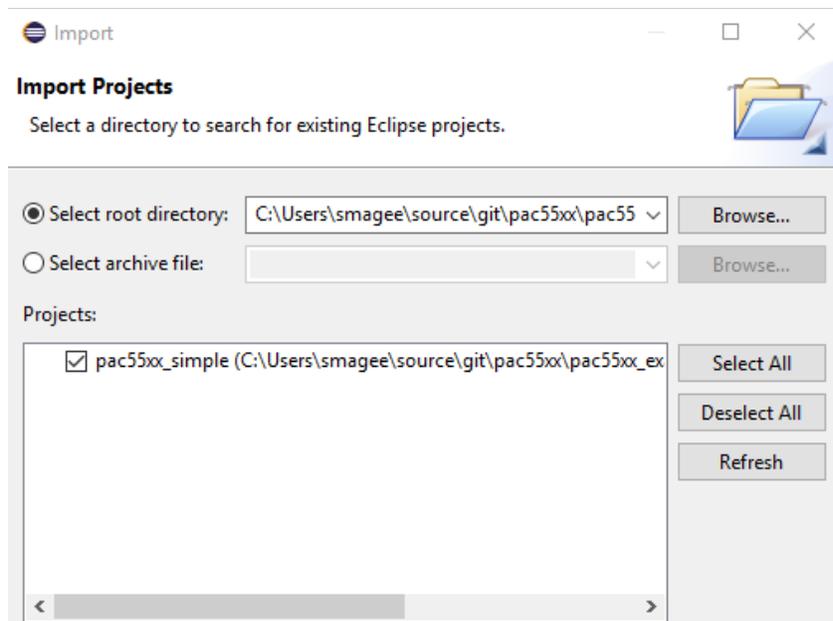


Click Browse, and then navigate to the folder that contains the eclipse project. This will be the folder that contains the files .project and .cproject.

Now select ok.



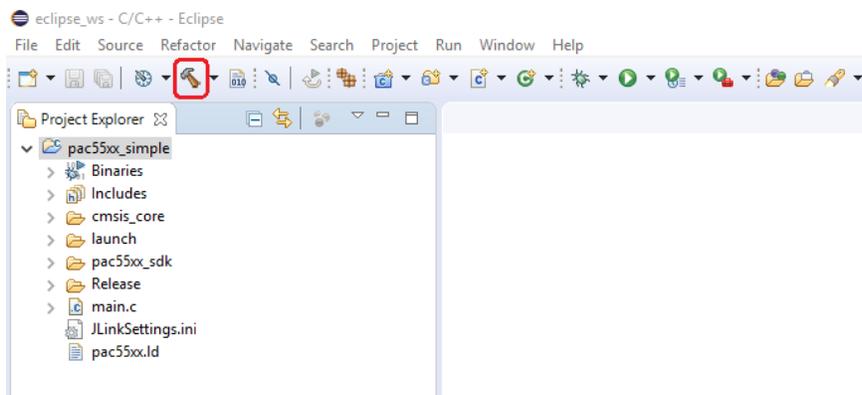
The project should show up in the list as shown below. Now click Finish.



The project should now be in the workspace and within the Project Explorer window.



Now open the project by clicking on the arrow to the left of the project (see figure above). The list of files contained in the project will be shown in the Project Explorer (see figure below).



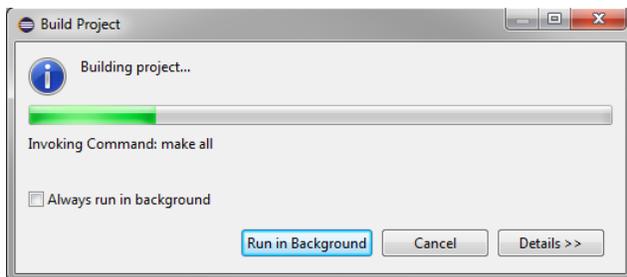
## 2.2 Building the Project

Before the project is built for the first time, click File->Refresh to regenerate the files Eclipse needs for building the project. Most PAC projects are not shipped with a Release folder since all of the files such as objects, executables, etc. can be regenerated. So, a Refresh is necessary to regenerate it.

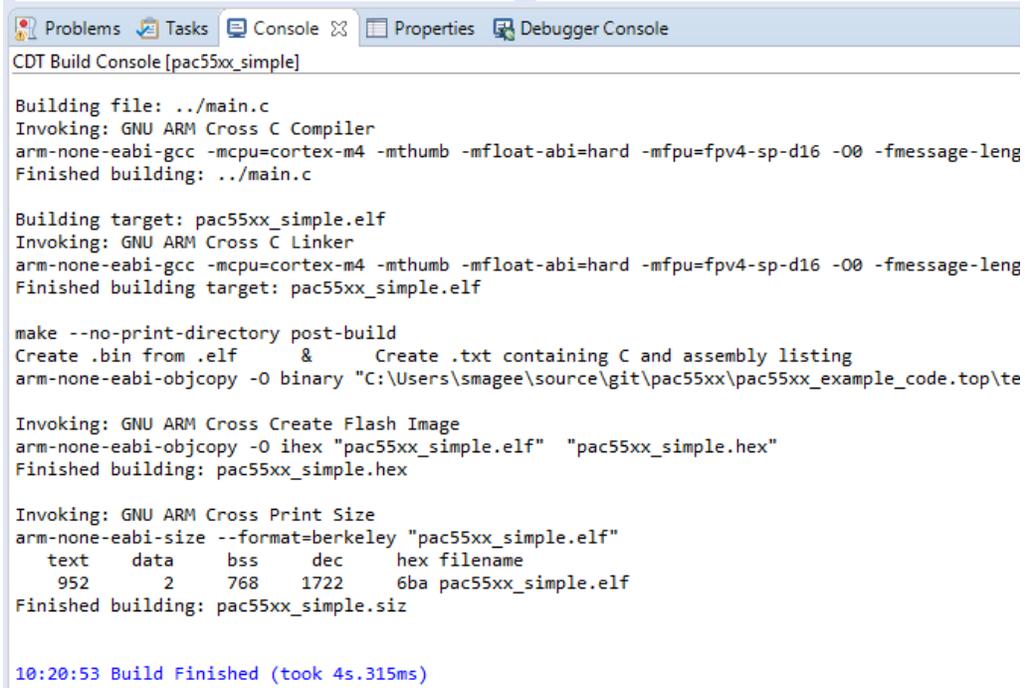
Refresh will also add any new files placed in the project folder and subfolders. Don't worry that other IDE project files get added, as these will be ignored by eclipse. If not using IAR or Keil, these can be deleted from the project; however, Eclipse will delete them from the disk drive also.

Left click on the project name so that the build icon (hammer) becomes available as shown in the figure above.

Click the build icon (hammer) to build the project. The "Build Project" dialog box will be displayed and show the progress of the build.



While it's building, you can also click on the console tab to see the build process as shown below. If all goes well, you should see no errors and the last portion of the build as shown below.



```
CDT Build Console [pac55xx_simple]

Building file: ../main.c
Invoking: GNU ARM Cross C Compiler
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -O0 -fmessage-length
Finished building: ../main.c

Building target: pac55xx_simple.elf
Invoking: GNU ARM Cross C Linker
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -O0 -fmessage-length
Finished building target: pac55xx_simple.elf

make --no-print-directory post-build
Create .bin from .elf & Create .txt containing C and assembly listing
arm-none-eabi-objcopy -O binary "C:\Users\smagee\source\git\pac55xx\pac55xx_example_code.top\te

Invoking: GNU ARM Cross Create Flash Image
arm-none-eabi-objcopy -O ihex "pac55xx_simple.elf" "pac55xx_simple.hex"
Finished building: pac55xx_simple.hex

Invoking: GNU ARM Cross Print Size
arm-none-eabi-size --format=berkeley "pac55xx_simple.elf"
  text  data  bss  dec  hex filename
  952   2   768  1722  6ba pac55xx_simple.elf
Finished building: pac55xx_simple.siz

10:20:53 Build Finished (took 4s.315ms)
```

If the build succeeds, then you are ready to move on to debug the project on a pac55xx device. If an issue is encountered, check out the Troubleshooting section 3 of this document.

## 2.3 Debugging the Project

Currently Eclipse debugging of PAC devices is only supported using J-Link. If debugging a PAC52xx device, the J-Link tools have built in support, so skip to section 1.3.2. If debugging a PAC55xx device, J-Link support must be added.

### 2.3.1 Add PAC55xx support to J-Flash

Before you can use Eclipse to debug on a PAC55xx device, J-Link/J-Flash must be updated for PAC55xx. If this has not already been performed, follow the instructions found in the folder `pac55xx_jlink_jflash_support` that is contained in the Eclipse with PAC support package. The J-Link support package can also be obtained from the [www.Active-Semi.com](http://www.Active-Semi.com) website on any of the PAC55xx device's product folder->software tab.

### 2.3.2 Setup Hardware

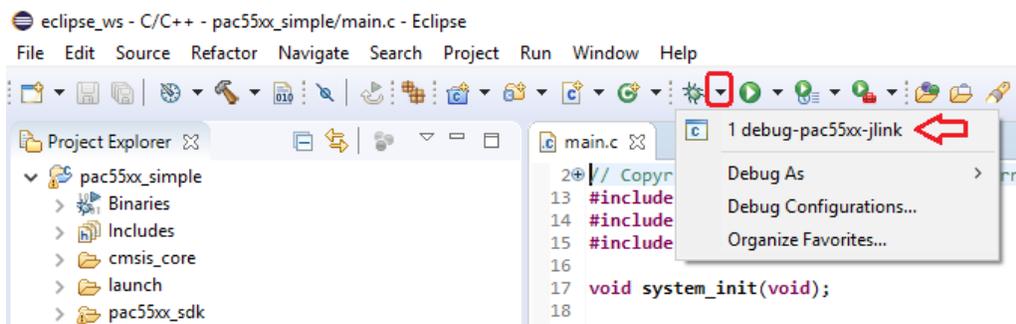
Before debugging, make sure to follow the hardware user's guide for your PAC55xx EVK for configuring, powering, and connecting a J-Link SWD debugger.

### 2.3.3 Special Note about Debugging, Downloading, and Erasing

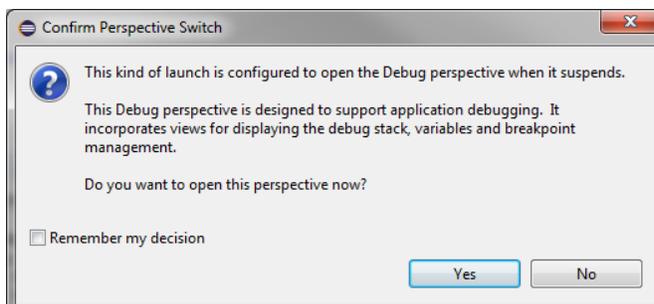
For all debug, download, erase activities, it's best to have only a single project open at a time so that .launch configurations for other projects don't show up. If problems are encountered, check the Troubleshooting Section 3

### 2.3.4 Starting Debug

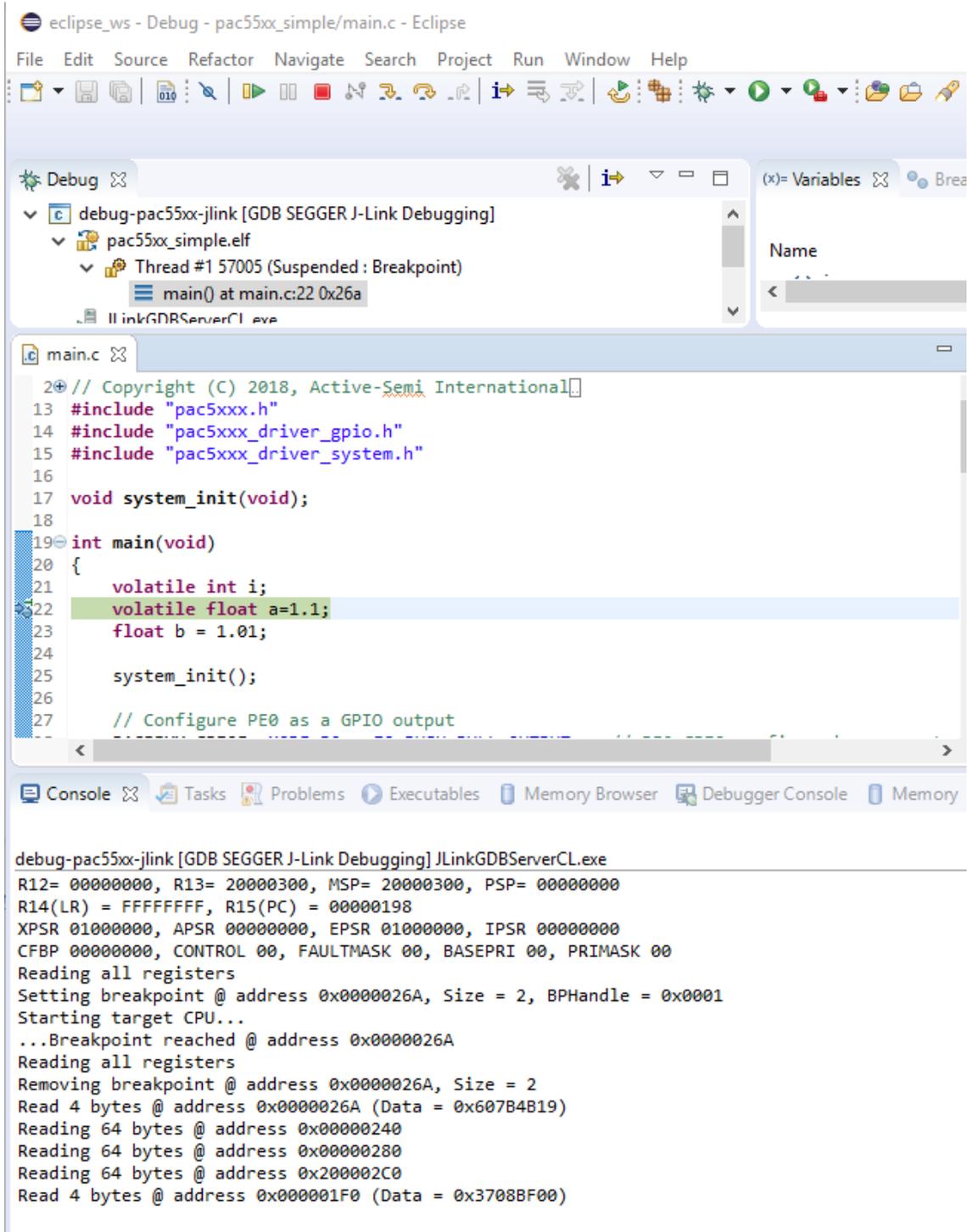
To start the debugger, click on the down arrow next to the debug icon and then click on "debug-pac5Xxx-jlink" to launch the debugger.



The perspective switch message will be shown below which lets you know that you are switching to the debug perspective. Select "Remember my decision" so you won't be prompted again.



Then, if all goes well, the project will rebuild anything that needs to be rebuilt and start the J-Link GDB server for debugging of J-Link. The debug session will run to main and stop on the first line of code as shown below. Now you can perform normal debugging operations. If an issue is encountered, check out the Troubleshooting section 3 of this document.



eclipse\_ws - Debug - pac55xx\_simple/main.c - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Debug (x)= Variables Brea

debug-pac55xx-jlink [GDB SEGGER J-Link Debugging]

- pac55xx\_simple.elf
  - Thread #1 57005 (Suspended : Breakpoint)
    - main() at main.c:22 0x26a

main.c

```

20 // Copyright (C) 2018, Active-Semi International
13 #include "pac5xxx.h"
14 #include "pac5xxx_driver_gpio.h"
15 #include "pac5xxx_driver_system.h"
16
17 void system_init(void);
18
19 int main(void)
20 {
21     volatile int i;
22     volatile float a=1.1;
23     float b = 1.01;
24
25     system_init();
26
27     // Configure PE0 as a GPIO output
  
```

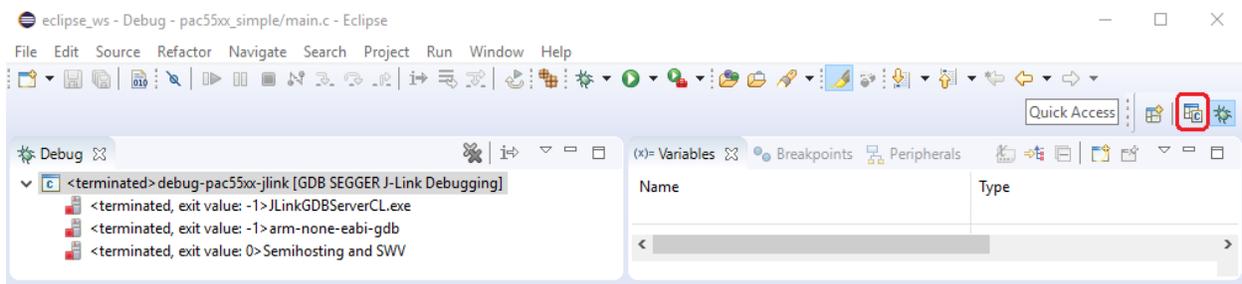
Console Tasks Problems Executables Memory Browser Debugger Console Memory

debug-pac55xx-jlink [GDB SEGGER J-Link Debugging] JLinkGDBServerCL.exe

```

R12= 00000000, R13= 20000300, MSP= 20000300, PSP= 00000000
R14(LR) = FFFFFFFF, R15(PC) = 00000198
XPSR 01000000, APSR 00000000, EPSR 01000000, IPSR 00000000
CFBP 00000000, CONTROL 00, FAULTMASK 00, BASEPRI 00, PRIMASK 00
Reading all registers
Setting breakpoint @ address 0x0000026A, Size = 2, BPHandle = 0x0001
Starting target CPU...
...Breakpoint reached @ address 0x0000026A
Reading all registers
Removing breakpoint @ address 0x0000026A, Size = 2
Read 4 bytes @ address 0x0000026A (Data = 0x607B4B19)
Reading 64 bytes @ address 0x00000240
Reading 64 bytes @ address 0x00000280
Reading 64 bytes @ address 0x200002C0
Read 4 bytes @ address 0x000001F0 (Data = 0x3708BF00)
  
```

After stopping the debugger, you can switch back to the C Perspective by clicking on the C perspective icon shown below.



Note, if you'd like to modify the “debug-pac5Xxx-jlink” launch configuration, you can do so by selecting Run->Debug Configurations... and editing the configuration.

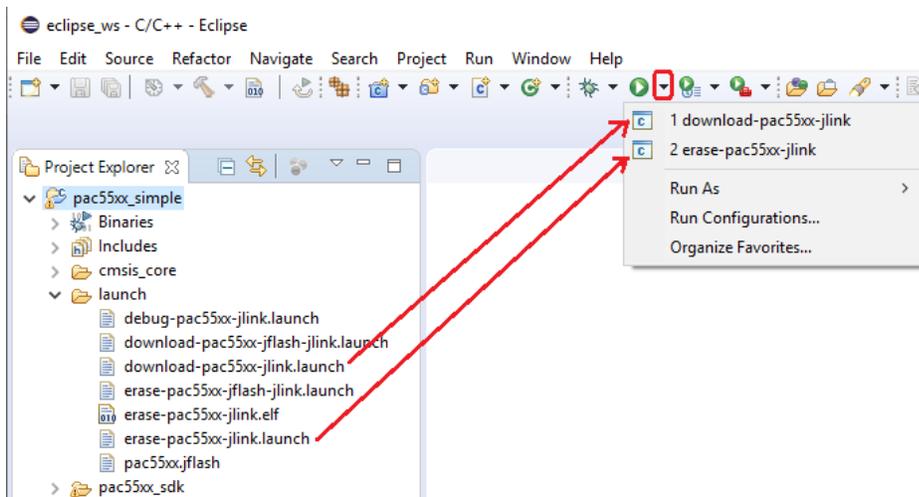
## 2.4 Erasing and Flashing

Sometimes it's desirable to Erase the device or Flash it without debugging. There are 2 options for Erasing and Flashing using J-Link. The first option uses the built in J-Link plugin, and the second option uses the external JFlash tool. Note that the JFlash tool requires the J-Link HW to be licensed for use with JFlash.

### 2.4.1 Erasing and Flashing with built-in J-Link Plugin

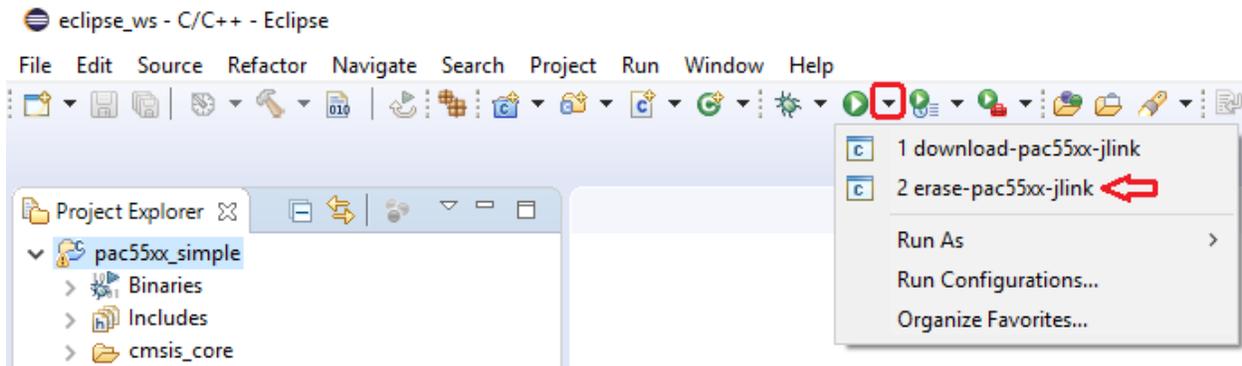
For erasing and Flashing with the built-in J-Link plugin, there are two .launch configurations that can be used and are contained in the launch folder of the project file list. These are also found in the Run controls, which can be accessed by clicking the down arrow next to the green Run icon as shown below.

- erase-pac5Xxx-jlink.launch
- download-pac5Xxx-jlink.launch



#### 2.4.1.1 Erasing with J-Link Plugin

To erase the device, click the down arrow next to the Run icon and then click on “erase-pac5Xxx-jflash-jlink” as shown below.



This J-Link erase mechanism performs the erase by downloading an ELF file that contains all FFs. The console will look similar to below as it downloads and verifies that all FFs have been written, which effectively ensures the part is erased.

```

erase-pac55xx-jlink [GDB SEGGER J-Link Debugging] JLinkGDBServerCL.exe
Resetting target
Received monitor command: halt
Halting target CPU...
...Target halted (PC = 0xFFFFFFFF)
Received monitor command: regs
R0 = 00000000, R1 = 00000000, R2 = 00000000, R3 = 00000000
R4 = 00000000, R5 = 00000000, R6 = 00000000, R7 = 00000000
R8 = 00000000, R9 = 00000000, R10= 00000000, R11= 00000000
R12= 00000000, R13= FFFFFFFC, MSP= FFFFFFFC, PSP= 00000000
R14(LR) = FFFFFFFF, R15(PC) = FFFFFFFE
XPSR 01000000, APSR 00000000, EPSR 01000000, IPSR 00000000
CFBP 00000000, CONTROL 00, FAULTMASK 00, BASEPRI 00, PRIMASK 00
Reading all registers
Received monitor command: speed 1000
Target interface speed set to 1000 kHz
Received monitor command: flash breakpoints 0
Flash breakpoints disabled
Downloading 16368 bytes @ address 0x00000000 - Verified OK
Downloading 16368 bytes @ address 0x00003FF0 - Verified OK
Downloading 16368 bytes @ address 0x00007FE0 - Verified OK
Downloading 16368 bytes @ address 0x0000BFD0 - Verified OK
Downloading 16368 bytes @ address 0x0000FFC0 - Verified OK
Downloading 16368 bytes @ address 0x00013FB0 - Verified OK
Downloading 16368 bytes @ address 0x00017FA0 - Verified OK
Downloading 16368 bytes @ address 0x0001BF90 - Verified OK
Downloading 128 bytes @ address 0x0001FF80 - Verified OK
Comparing flash [
  
```

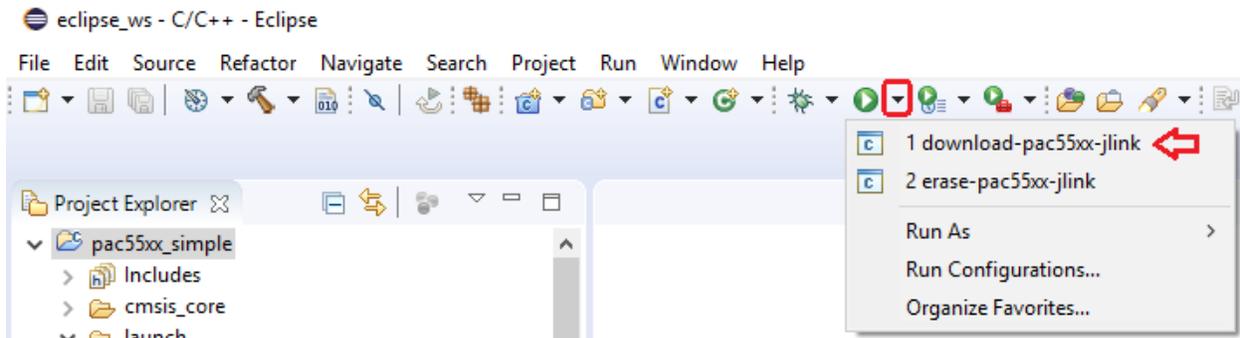
After the operation is complete, the console will display the following:

```

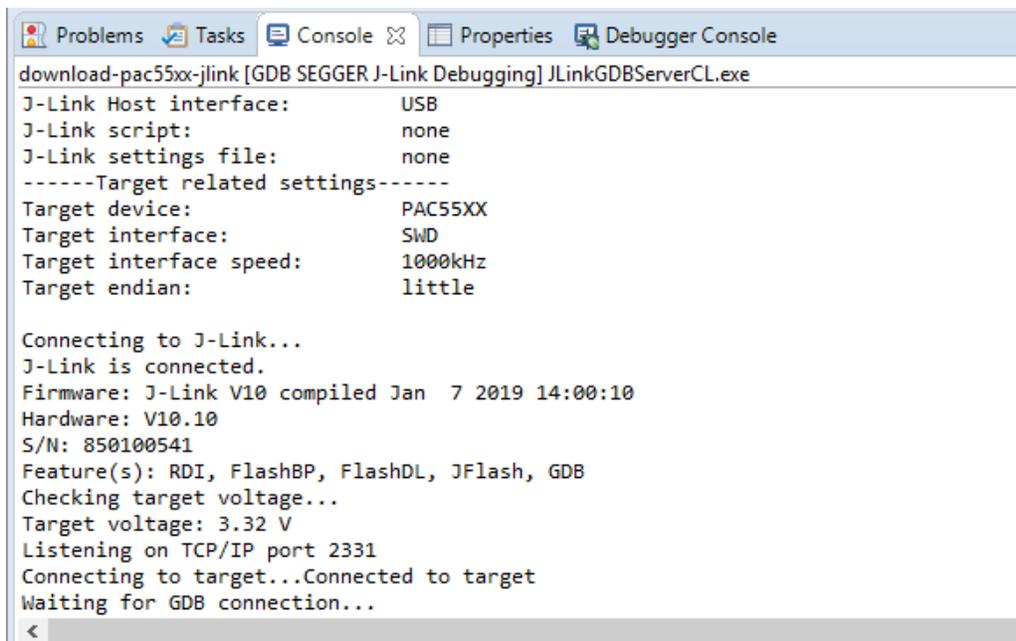
<terminated> erase-pac55xx-jlink [GDB SEGGER J-Link Debugging] Semihosting and SWV
SEGGER J-Link GDB Server V6.42d - Terminal output channel
Connection closed by the GDB server.
  
```

### 2.4.1.2 Downloading (Erase, Program, Verify) with J-Link Plugin

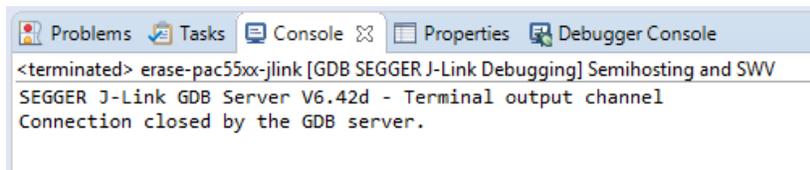
Download will erase, program and verify the device. To download using the J-Link plugin, click the down arrow next to the Run icon and then click on “download-pac5Xxx-jlink” as shown below.



The J-Link plugin will perform the download (erase/program/verify) operation. The console will display a screen similar to below; however, it will be displayed very quickly before completing.



After the operation is complete, the console will display the following:



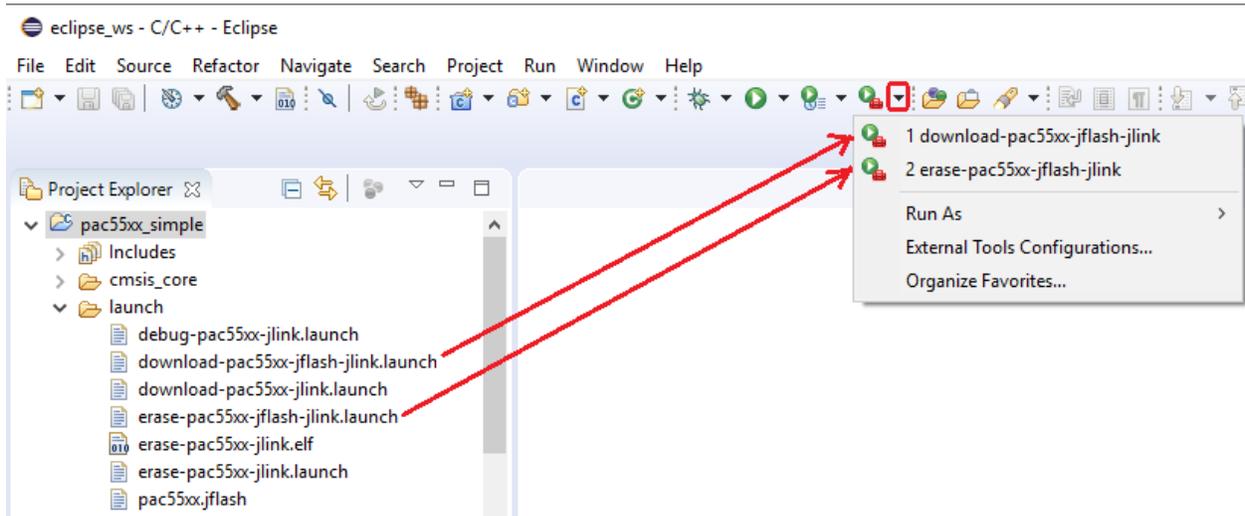
Note that these Run .launch configurations can be modified via Run->Run Configurations...

## 2.4.2 Erasing and Flashing with JFlash

To erase or flash the device without debugging, there are two .launch configurations that can be used and are contained in the project file list and also in the Run->External tools controls.

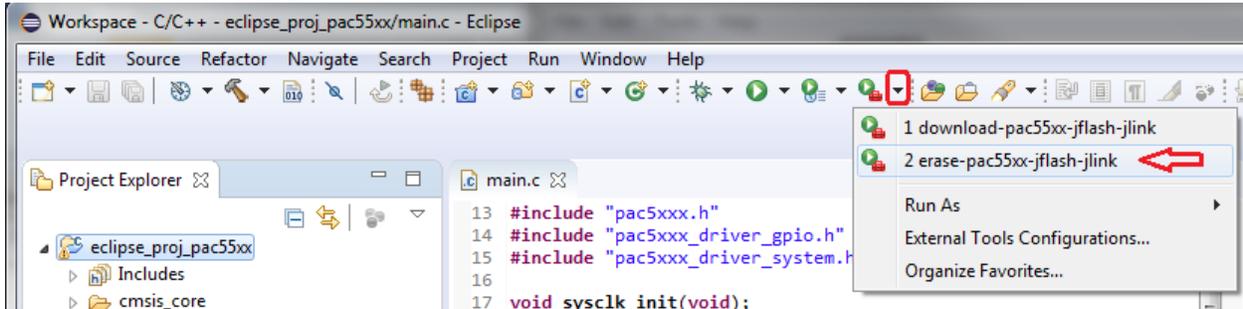
- erase-pac5Xxx-jflash-jlink.launch
- download-pac5Xxx-jflash-jlink.launch

These launch configurations make use of the Segger J-Flash application and J-Link debugger to erase and/or download the application image.

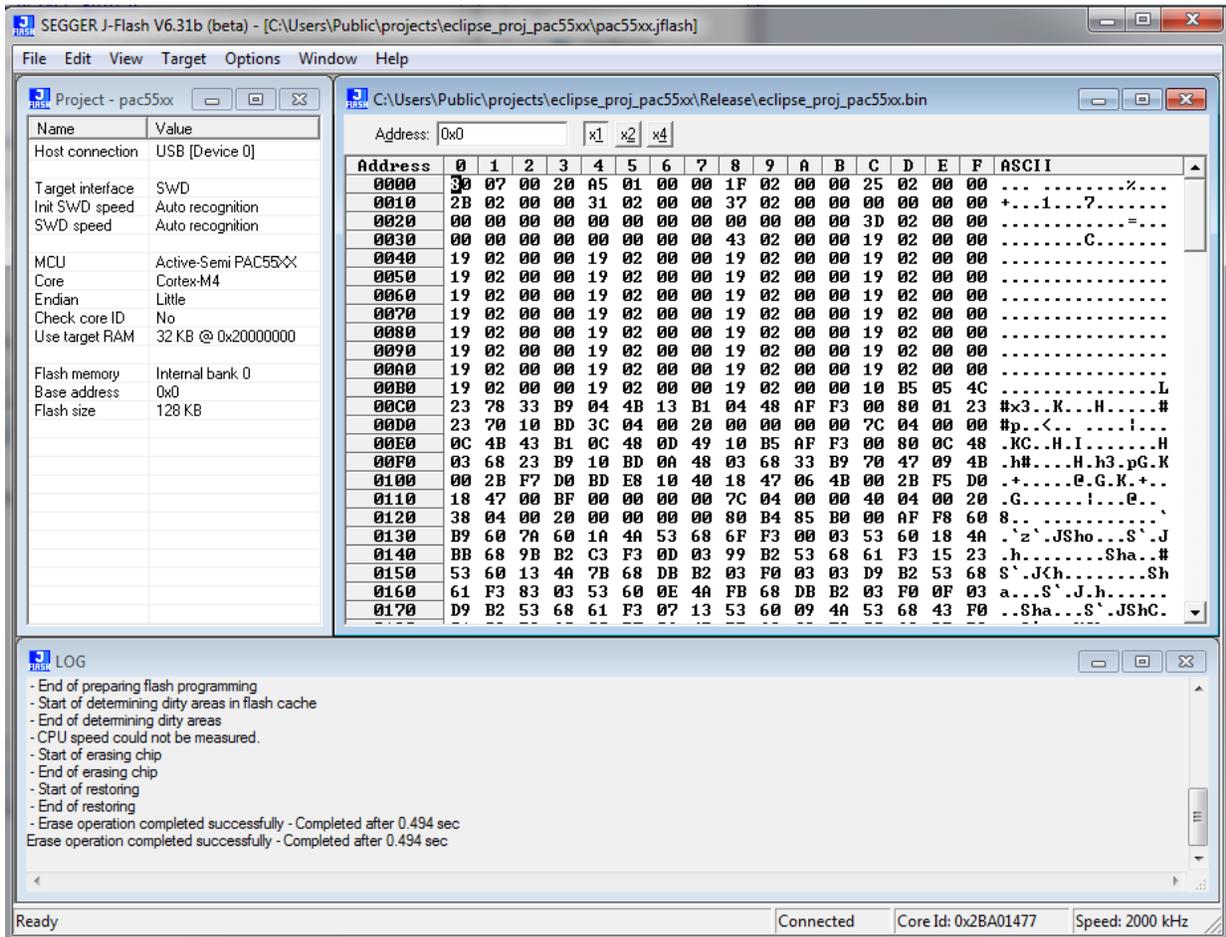


### 2.4.2.1 Erasing

To erase the device, click the down arrow next to the Run External Tools Icon and then click on “erase-pac55xx-jflash-jlink” as shown below.

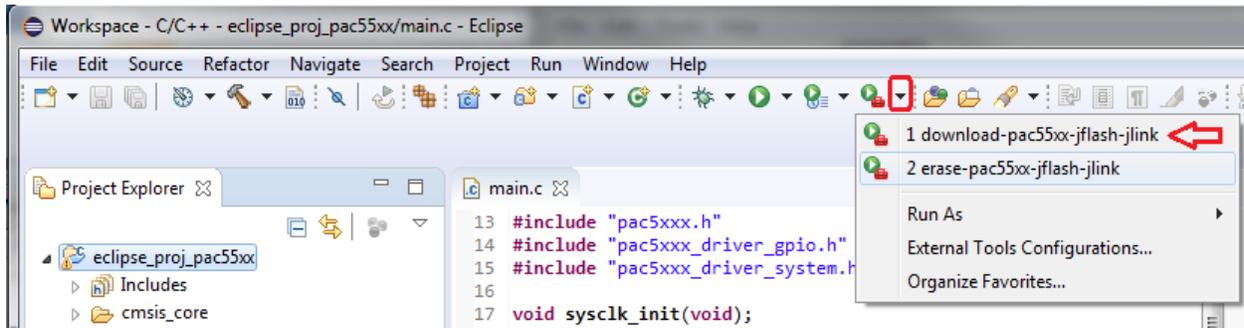


J-Flash will be started and perform the erase operation. You should see a screen as shown below. Verify the Erase operation completed successfully. Then close the window. If it asks if you want to save the jflash project, it's ok to say yes. After the first time, it won't ask again.

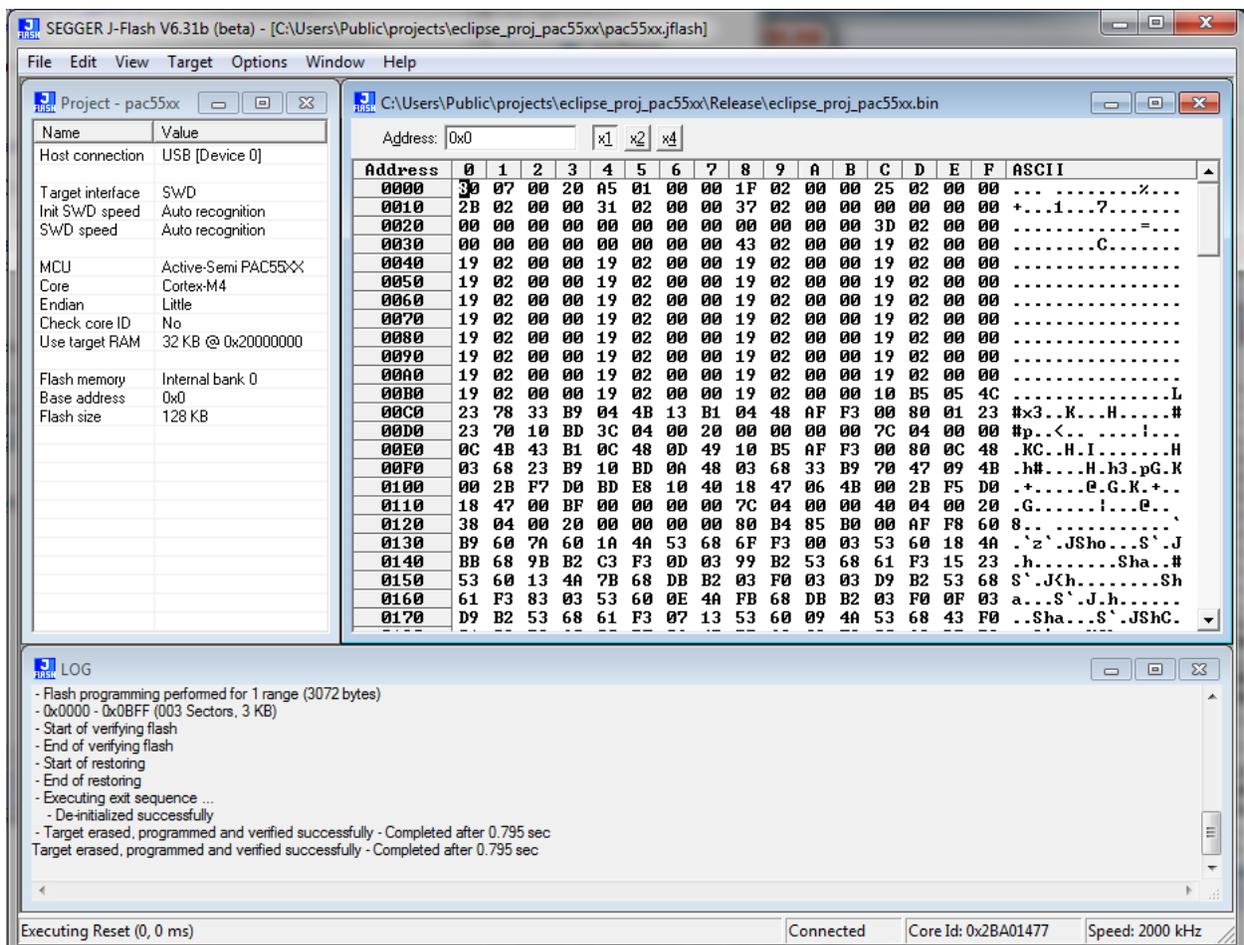


### 2.4.2.2 Downloading (Erase, Program, Verify)

Download will erase, program and verify the device. To download, click the down arrow next to the Run External Tools Icon and then click on “download-pac55xx-jflash-jlink” as shown below.



J-Flash will be started and perform the download (erase/program/verify) operation. You should see a screen as shown below. Verify the download operation completed successfully. Then close the window. If it asks if you want to save the jflash project, it's ok to say yes. After the first time, it won't ask again.



Note that these External Tool .launch configurations can be modified via Run->External Tools->External Tools Configurations...

## 2.5 Adding Files and Folders to the project

### 2.5.1 Adding Existing Files and Folders that are within the project

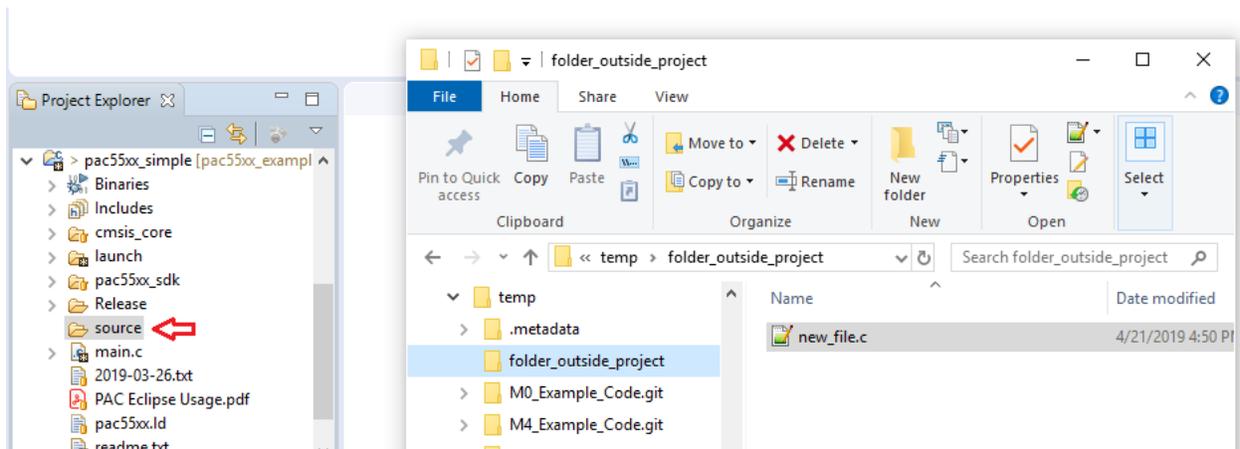
For some projects, all the source folders and files will be in the project folder or subfolders. If this is the case, then anytime an existing file is placed inside the immediate project folder or subfolders, it can be added by simply clicking F5 to refresh the project. Or, you can right click on the project name and select Refresh from the menu.

### 2.5.2 Adding Files and Folders that are outside the project folder

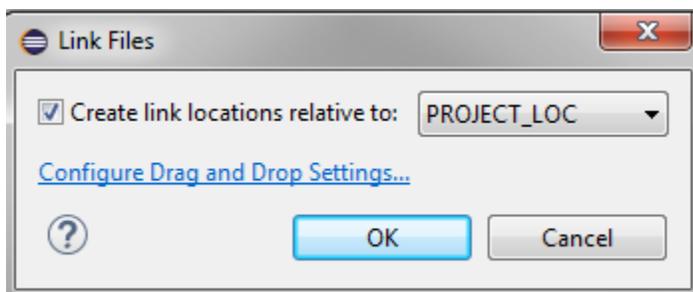
If existing files or folders are outside the project folder, then they can be added by dragging them from the Windows File Explorer application.

### 2.5.3 Use File Explorer to add a file that is outside the project folder

To add a file that is outside the project folder, open Windows File Explorer and navigate to the folder containing the file to be added.. Left click on the file and drag it to the desired location in the Eclipse Project Explorer window as shown below.

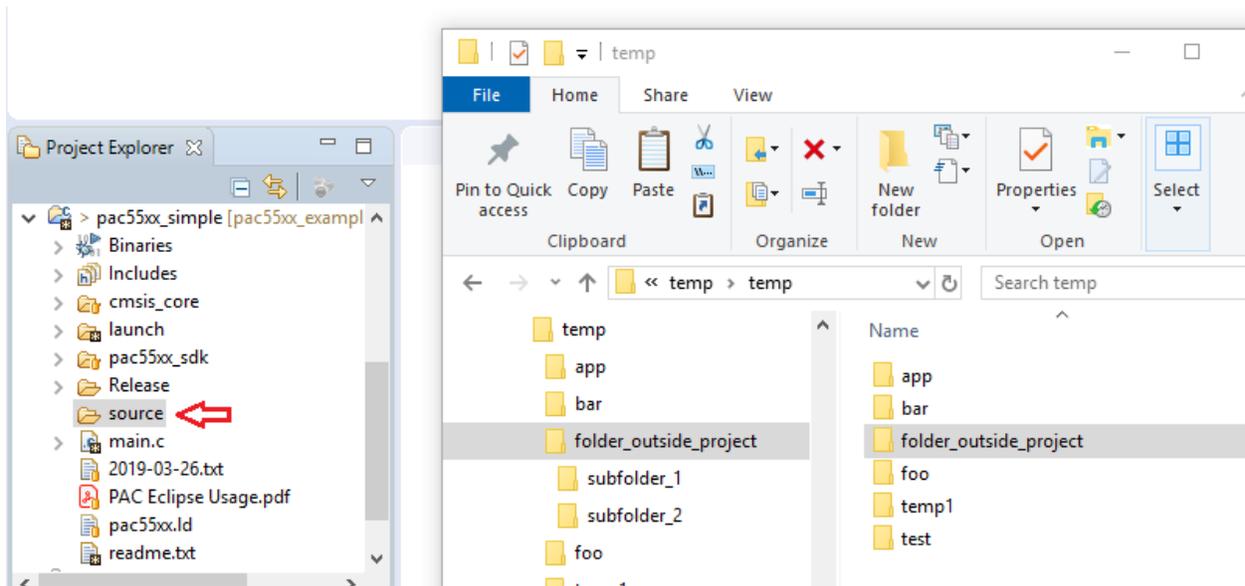


After releasing the mouse, a dialog box will appear as shown below. Click OK to allow the link to be created relative to PROJECT\_LOC. Even if the project and external files are moved, the link will continue to work as long as the file stays in the same relative location to the project folder.

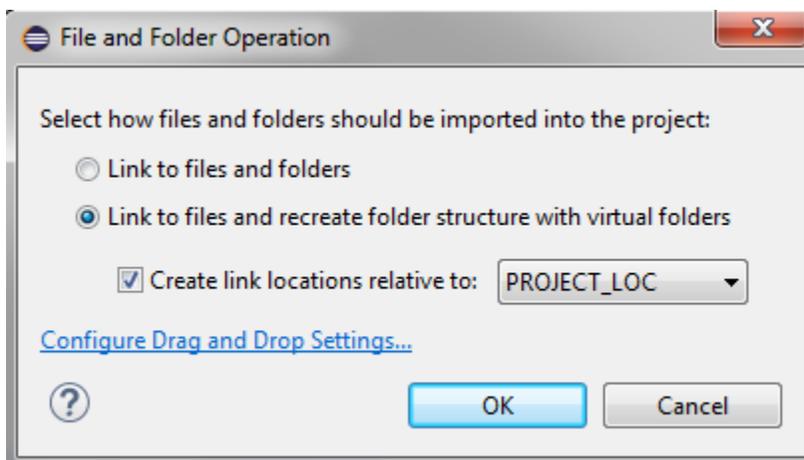


## 2.5.4 Use File Explorer to add folders/subfolders that are outside the project folder

To add a folder and subfolders that are outside the project folder, open File explorer and navigate to the folder containing the subfolders and files to be imported. Select the folder with the mouse and drag it to the desired location in the Eclipse Project Explorer window as shown below.



After releasing the mouse, a dialog box will appear as shown below. Select “Link to files and recreate folder structure with virtual folders”. Then click OK to allow the link to be created relative to PROJECT\_LOC. The folder names and structure will be recreated along with the files within Eclipse Project Explorer. Even if the project and external folders/files are moved, the link will continue to work as long as the folders/files stay in the same relative location to the project folder.



## 2.6 Renaming a Project

Renaming a project is best done by editing the project files with a text editor. First remove the project from the workspace. Then, edit the .project file and do a search and replace of the old project name for the new project name. Repeat the process for all .launch files.

## 3 TROUBLESHOOTING

### 3.1.1 make not found in PATH

When building a project an error may occur as shown below

“Program “make” not found in PATH”.

```
make -j pre-build main-build
Cannot run program "make": Launching failed

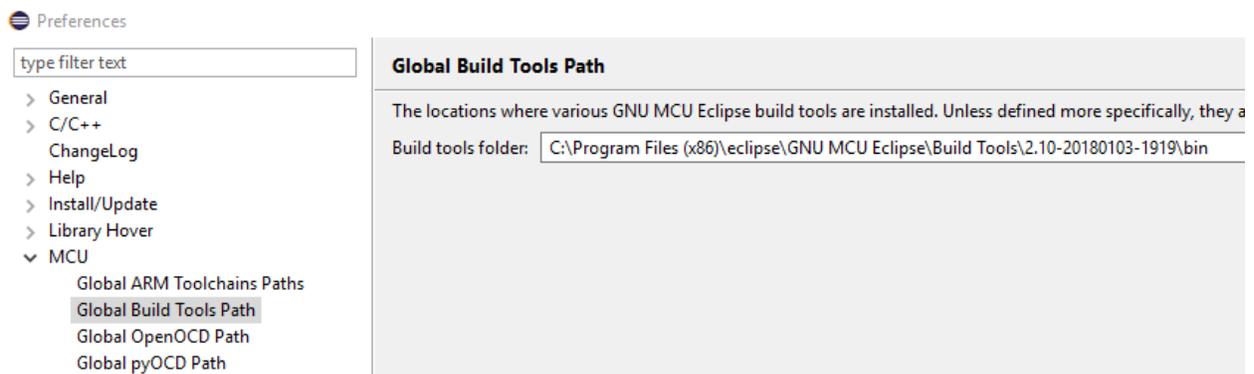
Error: Program "make" not found in PATH
PATH=[C:\Program Files (x86)\GNU Tools ARM Embedded\7 2017-q4-major\bin;C:/P

12:25:28 Build Finished (took 330ms)
```

To fix the issue, add the Build Tools PATH, by going to Window->Preferences->MCU->Global Build Tools PATH, and add the path to the GNU MCU Eclipse – Built Tools (see figure below).

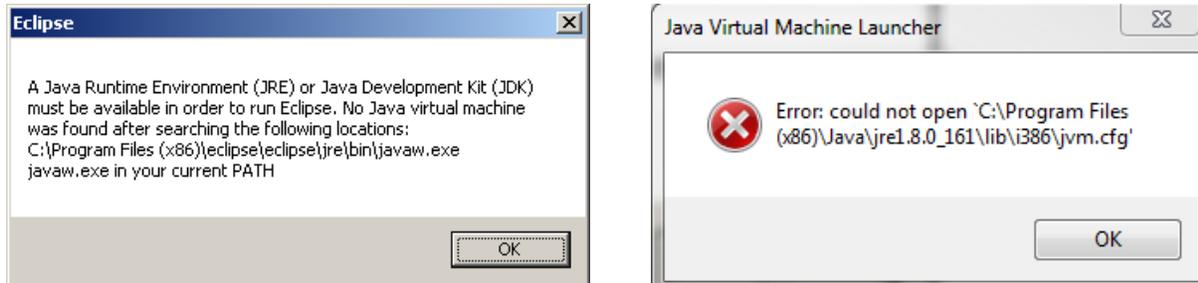
If using Eclipse with PAC support v1.1.x, the default path will be:

C:\Program Files (x86)\eclipse\GNU MCU Eclipse\Build Tools\2.10-20180103-1919\bin

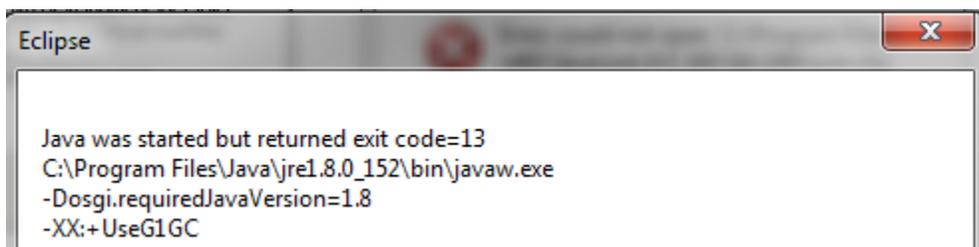


### 3.1.2 Java Error messages

If the installer finds an existing Java Development Kit (JDK) or Java Runtime Environment (JRE), it won't install the latest JDK/JRE and error messages like below may be displayed.



Another issue encountered is exit code=13 as shown below. This message is most likely because the 32-bit Eclipse is trying to run with a 64-bit Java.



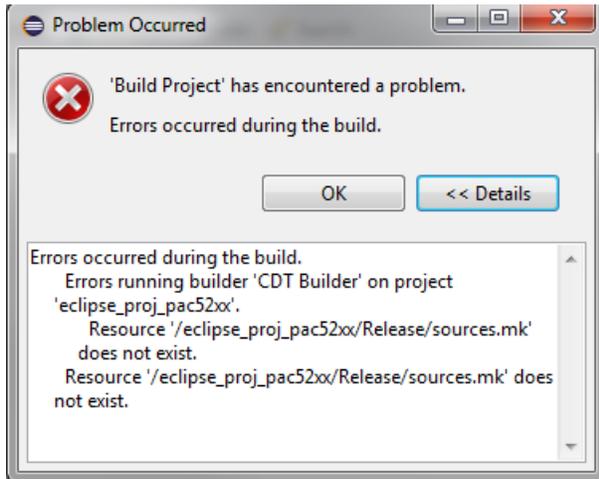
A Java 8 JRE/JDK is required to run all Neon packages based on Eclipse 4.6. For this eclipse installation, it also needs to be the 32-bit version.

To correct these issues, verify you have a 32-bit Java 8 installed, or install a 32-bit Java 8. Then modify the eclipse.ini file located in the eclipse program folder and add the -vm option to point to either the JRE or JDK as shown below. Make sure that -vm is on one line and the path to javaw.exe is on a second line. Also make sure the -vm option is just before the -vmargs option. Refer to this link if issues are encountered <https://wiki.eclipse.org/Eclipse.ini>.

```
org.eclipse.platform
--launcher.defaultAction
openFile
--launcher.appendVmargs
-vm
C:\Program Files (x86)\Java\jre1.8.0_161\bin\javaw.exe
-vmargs
-Dosgi.requiredJavaVersion=1.8
-XX:+UseG1GC
-XX:+UseStringDeduplication
-Dosgi.requiredJavaVersion=1.8
-Xms256m
-Xmx1024m
```

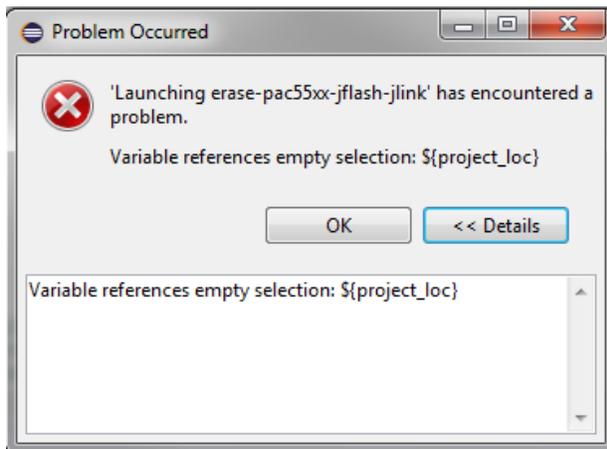
### 3.1.3 If sources.mk, objects.mk, or makefile are not found

If during the initial build, the sources.mk, objects.mk or makefile are not found (as shown in the figure below), click File->Refresh. Eclipse will regenerate the files it needs. Most PAC projects are not shipped with a Release folder, so this is necessary to regenerate it.



### 3.1.4 .launch Configuration variable references empty selection

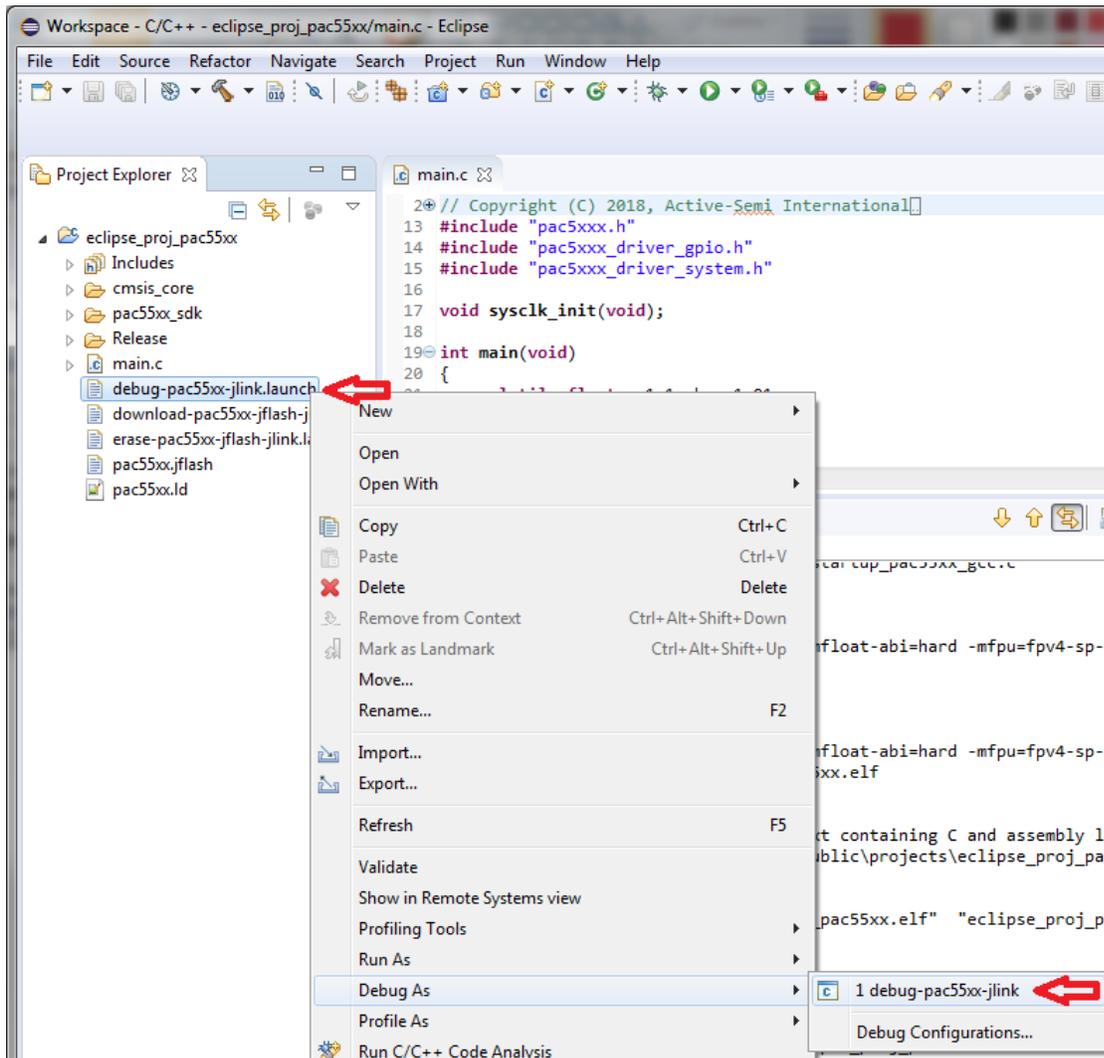
When executing a .launch, sometimes the tool complains that a variable reference is empty as shown below. To resolve the issue, first try clicking on the project name in Project Explorer, and then executing the .launch again. If that doesn't work, close the project and reopen it to re-establish the \${project\_loc} variable. You do not need to exit eclipse.



### 3.1.5 Debug or External Tool .launch files not available from Icon menu

If the debug, erase, or download launch configurations don't show up in the Icon menu, you may have to execute them from the project file list.

For Debug .launch configurations like "debug-pac55xx-jlink.launch", right click on the desired .launch file and choose "Debug As", then click on the file to run it (see figure below).



For Run .launch configurations or External tool .launch configurations like the following:

- erase-pac55xx-jlink.launch
- download-pac55xx-jlink.launch
- erase-pac55xx-jflash-jlink.launch
- download-pac55xx-jflash-jlink.launch

right click on the desired .launch file and choose “Run As”, then click on the file to run it.

### 3.1.6 jflash-jlink.launch Can't save jflash.ini

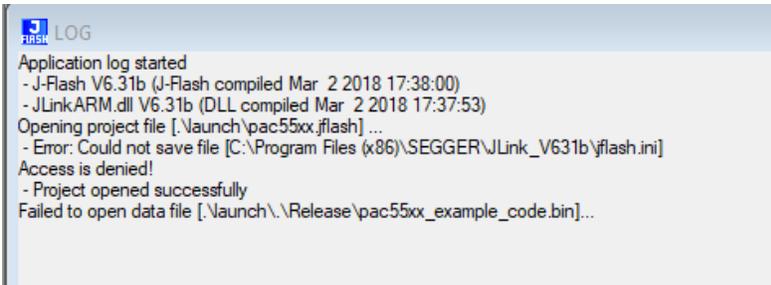
In some instances executing a download or erase using the xxxxx.jflash-jlink.launch files will result in JFlash giving an error

“Error: Could not save file [C:\Program Files (x86)\SEGGER\JLink\_V631b)\jflash.ini]”

© 2019 Copyright, Active-Semi<sup>®</sup> International, Inc. - 22 - 2019-04-21

No portion of this document may be reproduced or reused in any form without Active-Semi's prior written consent

as shown in the figure below.



```
JFLASH LOG
Application log started
- J-Flash V6.31b (J-Flash compiled Mar  2 2018 17:38:00)
- JLinkARM.dll V6.31b (DLL compiled Mar  2 2018 17:37:53)
Opening project file [.\naunch\pac55xx.jflash] ...
- Error: Could not save file [C:\Program Files (x86)\SEGGER\JLink_V631b\jflash.ini]
Access is denied!
- Project opened successfully
Failed to open data file [.\naunch\.\Release\pac55xx_example_code.bin]...
```

This error appears on Windows 10, but does not always occur. It seems that if another JFlash process hasn't completely closed down, then a second JFlash process can't write the jflash.ini file properly.

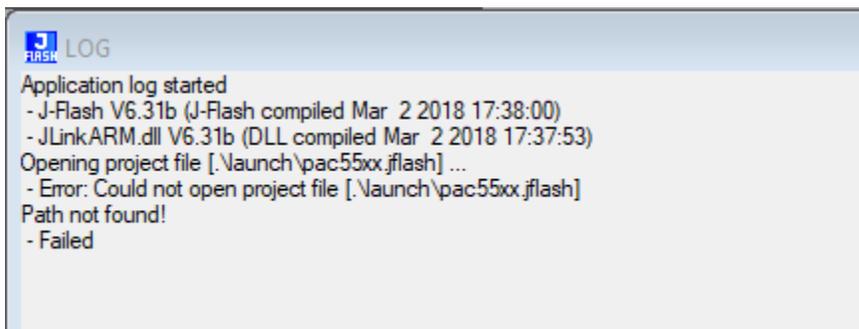
Waiting a 10 seconds and repeating the launch usually allows it to work. If available, try the equivalent xxxxxx.jlink.launch, which doesn't use Jflash.

### 3.1.7 JFlash - Could not open project file

If an Error occurs where JFlash can't open the project file (see figure below), then make sure to highlight the project name in Project Explorer before executing an External Tool from the External tool .launch configurations like the following:

- erase-pac55xx-jflash-jlink.launch
- download-pac55xx-jflash-jlink.launch

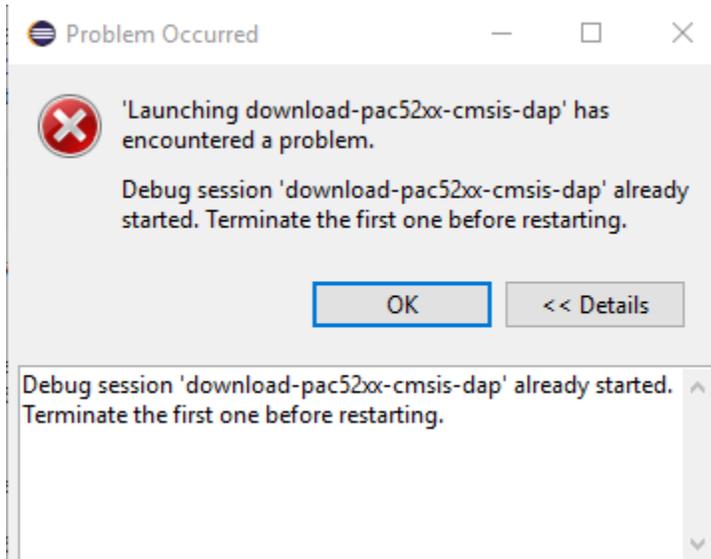
If it still fails, make sure all other projects are closed. Also, try closing and reopening the project.



```
JFLASH LOG
Application log started
- J-Flash V6.31b (J-Flash compiled Mar  2 2018 17:38:00)
- JLinkARM.dll V6.31b (DLL compiled Mar  2 2018 17:37:53)
Opening project file [.\naunch\pac55xx.jflash] ...
- Error: Could not open project file [.\naunch\pac55xx.jflash]
Path not found!
- Failed
```

### 3.1.8 Debug session already started – Terminate the first one before restarting

Sometimes an may error occur as shown below that states “Debug session ... already started. Terminate the first one before restarting. When this happens, close the project, reopen, and try the launch again.



## 4 ECLIPSE TIPS

### 4.1 Eclipse Workspaces

There are many different opinions on how to use Eclipse Workspaces. But in general, they are used to group together a collection of related projects. Most users typically do not store the projects directly in the workspace so that the projects can be accessed from multiple workspaces, and also because workspaces are Eclipse version dependent.

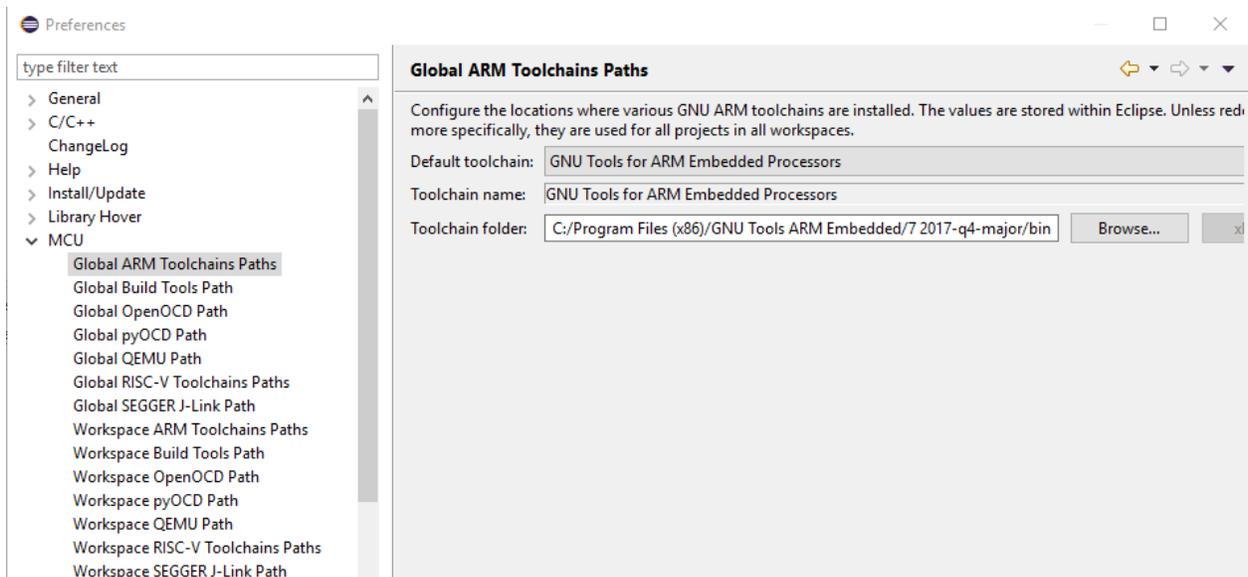
Workspaces are not guaranteed to be portable across user's and may contain hardcoded paths for projects or other metadata. So, sharing of workspaces and version controlling the workspace is not typically done.

Some settings and preferences only get stored with the workspace, but can be exported for reuse in another workspace. See this link for additional information on migrating workspace settings: <https://stackoverflow.com/questions/4848767/how-to-clone-an-eclipse-workspace>.

To find out more about workspaces, do an internet search for Eclipse Workspace and peruse the many different links on the subject.

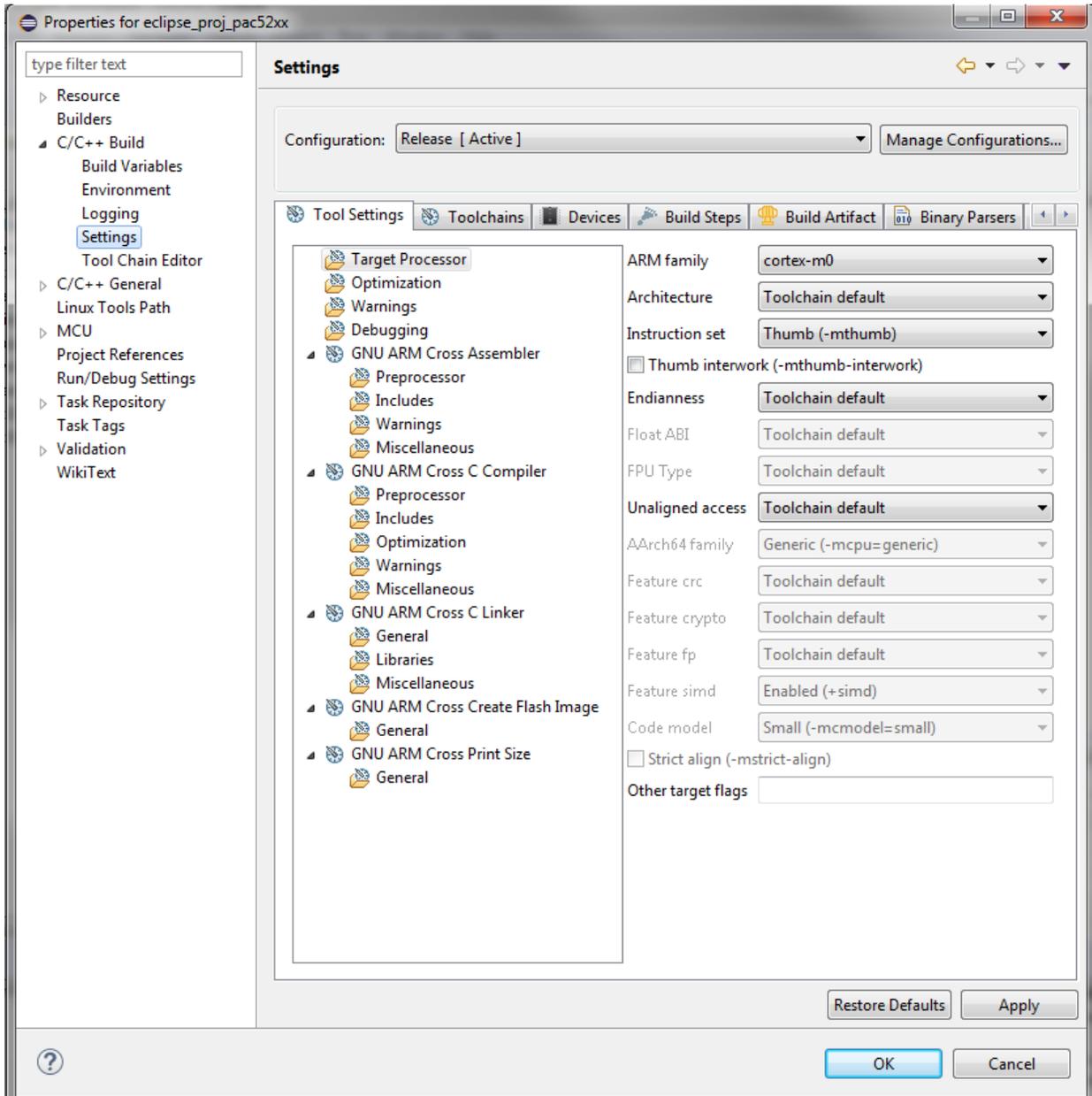
### 4.2 Eclipse Paths

Many of the paths that Eclipse uses to access programs are set in the Windows->Preferences->MCU menu. For example the ARM Toolchains Path as shown below. It's usually a good idea to set the Global path and then the Workspace path will inherit it.



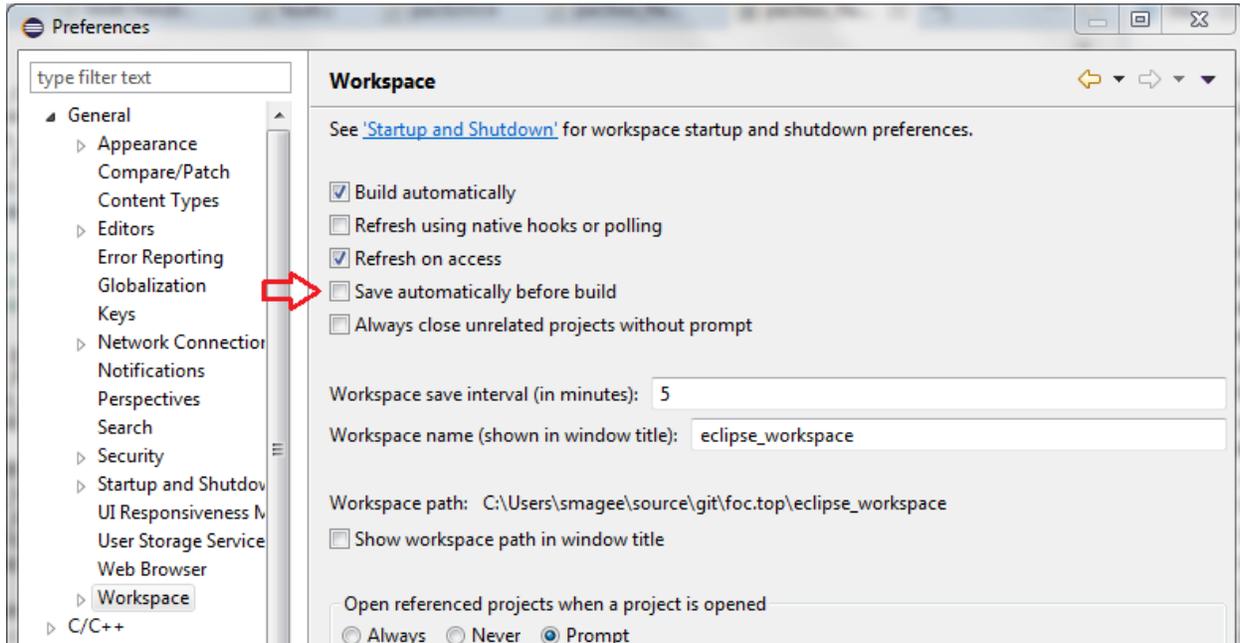
### 4.3 Project Build Settings

Many of the project build settings related to compiling, linking, etc., are stored under Project->Preferences->C/C++ Build->Settings as shown in the figure below:



#### 4.4 Save Modified Files Before Building

To save modified files before building the project, go to Window->Preferences->General->Workspace and check the box next to “Save automatically before build”. Note that you will have to do this for each Workspace or use the export/import methods discussed above.

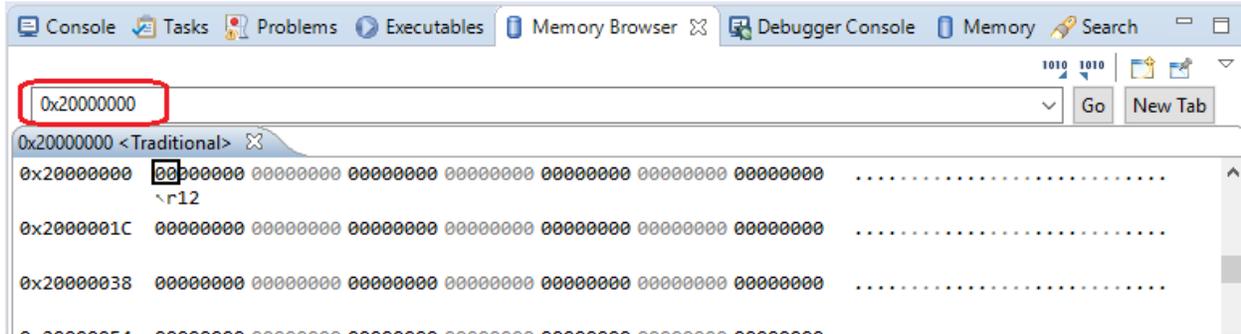


#### 4.5 Change editor to insert spaces instead of tabs

1. Click **Window » Preferences**
2. Expand **C/C++ » Code Style**
3. Click **Formatter**
4. Click the **New** button to create a new profile
5. Give a new name such as “K&R Tabs as Spaces”, then **OK** to continue
6. Click the **Indentation** tab
7. Under **General Settings**, set **Tab policy** to: Spaces only
8. Click **OK** multiple times to apply the changes.

## 4.6 Memory Browser vs Memory

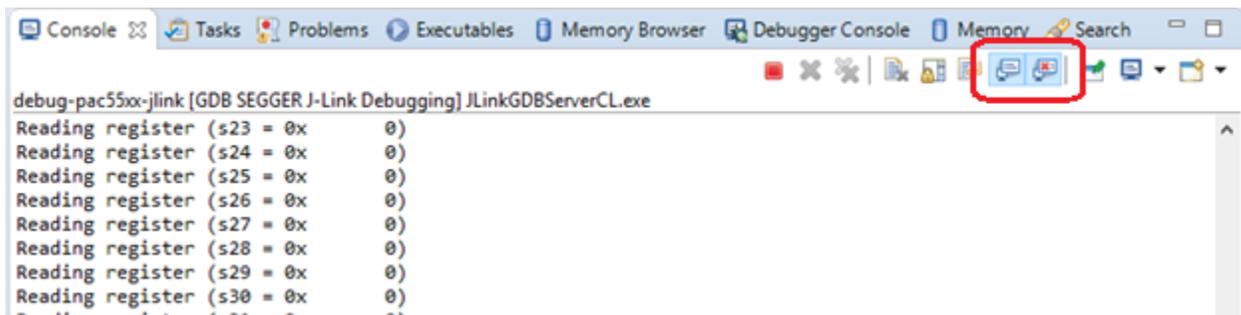
The Memory Browser window has an advantage over the Memory window in that you can change the address on the fly by entering a new address in the field at the top. To bring up the Memory Browser, just go to Window->Show View->Memory Browser.



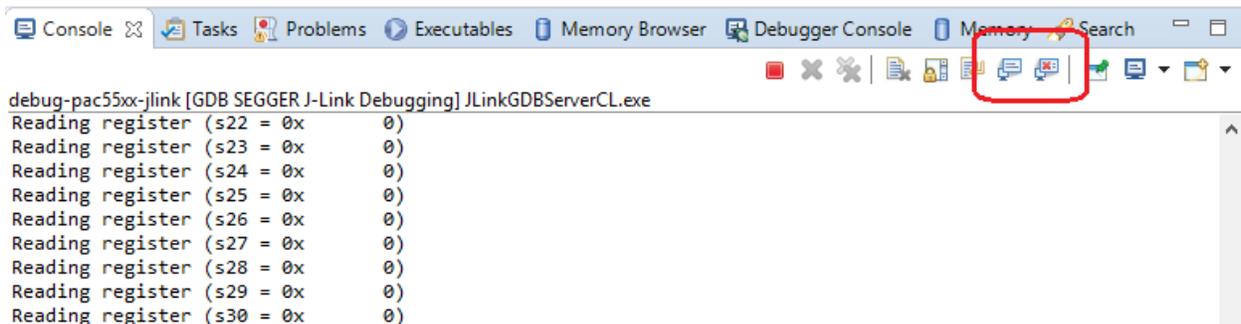
## 4.7 Keep Memory or Memory Browser Window in focus

While debugging, the default settings will cause the Console to be brought into focus every time it has something to report. This is very annoying when you want the Memory or Memory Browser window to stay visible. To solve the problem, go to the Console tab while debugging and click on the two icons shown below so that they are no longer shaded in blue. These two boxes are:

- Show Console When Standard Out Changes
- Show Console When Standard Error Changes



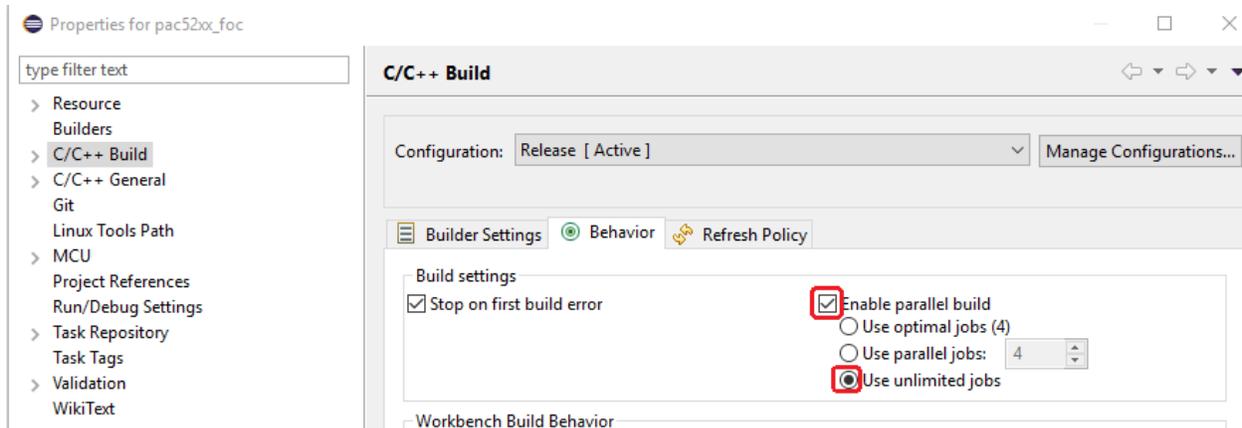
After you've clicked on each of them, they should now have the shading removed as shown below:



Now, select the Memory or Memory Browser tab and then single step the code. The window should stay in focus without jumping back to the Console.

## 4.8 Accelerate build process

To accelerate the build process, go to the project Preferences->C/C++ Build->Behavior tab and select “Enable parallel build” and “Use unlimited jobs” as shown below. This allows the PC to use multiple cores and compile the various source code in parallel.



## ABOUT ACTIVE-SEMI<sup>®</sup>

Founded in 2004 in Silicon Valley and headquartered in Allen, Texas, Active-Semi<sup>®</sup> is a rapidly emerging leader in the multi-billion dollar power management IC and intelligent digital motor drive IC markets. The company's portfolio of analog and mixed signal SoCs (systems-on-chips) are scalable core platforms used in charging, powering and embedded digital control systems for end applications such as industrial, commercial and consumer equipment. The company offers power application microcontrollers, DC/DC, AC/DC, PMU and LED drivers that significantly reduce solution size and cost while improving system-level reliability. Active-Semi's turnkey solutions deliver energy-saving power conversion architectures that minimize energy usage and compress system development cycle-time by greater than 50 percent. Active-Semi ships 50 million power ICs per quarter and reached the "one billion units shipped" milestone in May 2012. The multi-national company focuses on commercializing industry leading power management IC solution platforms and has developed broad intellectual property with over 150 patents granted and pending. For more information visit: <http://active-semi.com/>

---

**LEGAL INFORMATION & DISCLAIMER**

Copyright © 2019 Active-Semi, Inc. All rights reserved.  
All information provided in this document is subject to legal disclaimers.

Active-Semi reserves the right to modify its products, circuitry or product specifications without notice. Active-Semi products are not intended, designed, warranted or authorized for use as critical components in life-support, life-critical or safety-critical devices, systems, or equipment, nor in applications where failure or malfunction of any Active-Semi product can reasonably be expected to result in personal injury, death or severe property or environmental damage. Active-Semi accepts no liability for inclusion and/or use of its products in such equipment or applications. Active-Semi does not assume any liability arising out of the use of any product, circuit, or any information described in this document. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of Active-Semi or others. Active-Semi assumes no liability for any infringement of the intellectual property rights or other rights of third parties which would result from the use of information contained herein. Customers should evaluate each product to make sure that it is suitable for their applications. Customers are responsible for the design, testing, and operation of their applications and products using Active-Semi products. Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. All products are sold subject to Active-Semi's terms and conditions of sale supplied at the time of order acknowledgment. Exportation of any Active-Semi product may be subject to export control laws.

Active-Semi<sup>®</sup> and Power Application Controller<sup>®</sup> are registered trademarks of Active-Semi, Inc. Solutions for Sustainability<sup>™</sup>, Micro Application Controller<sup>™</sup>, Multi-Mode Power Manager<sup>™</sup>, Configurable Analog Front End<sup>™</sup>, and Application Specific Power Drivers<sup>™</sup> are trademarks of Active-Semi, Inc.

ARM<sup>®</sup> and Cortex are registered trademarks of ARM Limited. All referenced brands and trademarks are the property of their respective owners.