

## **Kinetis Design Studio v1.0.1 Quick Start Guide**

**Document: SOMN-KDS-0001 Revision: B**

Kinetis Design Studio is produced for Freescale by SOMNIUM™ Technologies <http://www.somniumtech.com>.

This document is Copyright ©2013-2014 SOMNIUM™ Technologies Limited. All rights reserved.

All trademarks are the properties of their respective Companies.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Main features . . . . .	1
1.2	Supported host operating systems . . . . .	1
1.3	Supported Freescale devices . . . . .	1
1.4	License information . . . . .	2
1.5	Objectives of this document . . . . .	2
<b>2</b>	<b>Installing the Kinetis Design Studio [beta] software development tools</b>	<b>2</b>
2.1	Installing on Windows . . . . .	2
2.2	Installing on Linux . . . . .	3
2.2.1	Installing with Red Hat package manager (RPM) . . . . .	3
2.2.2	Installing with Debian package manager (DEB) . . . . .	3
2.2.3	Extracting the Kinetis Design Studio [beta] software development tools from the TAR archive . . . . .	4
<b>3</b>	<b>Building an embedded application</b>	<b>5</b>
3.1	Launching the Kinetis Design Studio [beta] IDE . . . . .	5
3.2	Creating a new Kinetis Design Studio [beta] project . . . . .	5
3.3	Configuring the project . . . . .	8
<b>4</b>	<b>Debugging an embedded application</b>	<b>10</b>
4.1	Development environment . . . . .	10
4.2	Debugging with the Kinetis Design Studio [beta] IDE . . . . .	11
4.3	Controlling the application in the debugger . . . . .	14
<b>5</b>	<b>Flashing an embedded application</b>	<b>15</b>
5.1	Flashing with the Kinetis Design Studio [beta] IDE . . . . .	15
<b>A</b>	<b>Technical notes</b>	<b>16</b>
A.1	Differences with the GNU ARM Embedded toolchain . . . . .	16
A.2	Semihosting . . . . .	16
<b>B</b>	<b>Supported Freescale Kinetis MCUs</b>	<b>17</b>
<b>C</b>	<b>Driver Installation</b>	<b>18</b>
C.1	Installing on Windows . . . . .	18
C.2	Installing on Linux . . . . .	18
<b>D</b>	<b>CodeWarrior project migration</b>	<b>20</b>
D.1	Examples of warnings from an existing or imported CodeWarrior project . . . . .	20
D.2	Converting a CodeWarrior project file . . . . .	21
D.3	Building a migrated CodeWarrior project . . . . .	21
D.4	Common build errors after migration . . . . .	21

**E Disclaimer**

**23**

## List of Figures

1	Windows installer . . . . .	2
2	Kinetis Design Studio [beta] IDE welcome screen . . . . .	5
3	New Kinetis Design Studio [beta] project wizard . . . . .	6
4	Select whether to use Kinetis SDK . . . . .	6
5	Rapid application development using Processor Expert . . . . .	7
6	Project Explorer . . . . .	7
7	Project Properties, C/C++ build settings . . . . .	8
8	Linking with newlib-nano . . . . .	9
9	Build console and the generated binary . . . . .	9
10	Supported debug adapters . . . . .	10
11	Create a new debug configuration . . . . .	11
12	Debugger tab for a <i>GDB OpenOCD Debugging</i> configuration . . . . .	12
13	Debugger tab for a <i>GDB SEGGER J-Link Debugging</i> configuration . . . . .	12
14	Debugger tab for a <i>GDB P&amp;E Interface Debugging</i> configuration . . . . .	13
15	Startup tab for a <i>GDB SEGGER J-Link Debugging</i> configuration . . . . .	13
16	Kinetis Design Studio [beta] IDE debug perspective . . . . .	14
17	Flash configuration . . . . .	15

# 1 Introduction

Welcome to the Kinetis Design Studio [beta] software development tools, a GNU-based toolchain with an Eclipse Integrated Development Environment (IDE) for developing embedded applications targeted at the range of Freescale Kinetis MCUs based on ARM® Cortex™-M technology.

## 1.1 Main features

The Kinetis Design Studio [beta] software development tools are provided free of charge and includes the following features:

- A GNU-based toolchain
- An Eclipse based IDE for a robust editing, application build and debugging experience
- Integration with Processor Expert and Kinetis SDK
- Support for SEGGER J-Link/J-Trace, P&E USB Multilink Universal/USB Multilink Universal FX and OpenSDA/CMSIS-DAP debug adapters
- Optional newlib-nano C runtime library to reduce the memory footprint of an embedded application
- CodeWarrior to Kinetis Design Studio [beta] project migration assistant
- The Kinetis Design Studio [beta] software development tools are rigorously validated using commercial validation and performance suites

## 1.2 Supported host operating systems

The Kinetis Design Studio [beta] software development tools supports the following host operating systems:

- Microsoft® Windows® 7 and Windows® 8 (all variants). Windows-hosted variants of the Kinetis Design Studio [beta] software development tools are distributed as 32-bit binaries, which will run on 32-bit and 64-bit machines.
- Red Hat® Enterprise Linux (RHEL), CentOS 6.4 and Ubuntu 12.04 LTS. Linux-hosted variants of the Kinetis Design Studio [beta] software development tools are distributed as 32-bit binaries, which may require additional libraries to run on 64-bit systems.

---

**Note** The Kinetis Design Studio [beta] software development tools may work on older versions of these systems, but this has not been tested.

---

To use the Kinetis Design Studio [beta] software development tools the following minimal system requirements should be met:

- 1.8 GHz processor
- 2 GB of RAM
- Approximately 1.5 GB of free disk space (when installing the full product)

## 1.3 Supported Freescale devices

The Kinetis Design Studio [beta] software development tools supports a number of Freescale Kinetis K and L Series MCUs based on ARM® Cortex™-M technology. A complete list of supported devices is provided in Appendix B.

## 1.4 License information

The Kinetis Design Studio [beta] software development tools are licensed under the terms outlined in `license.htm`, which is found at the top of the install directory.

## 1.5 Objectives of this document

This guide demonstrates how to quickly get started using the Kinetis Design Studio [beta] software development tools. By following this guide you will be able to do the following:

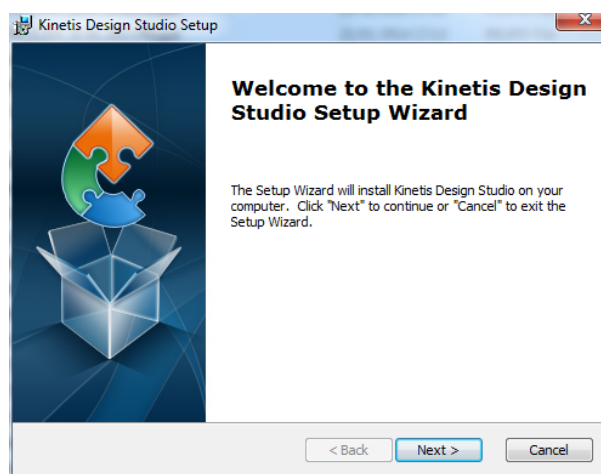
- Install the Kinetis Design Studio [beta] software development tools on either a Windows or Linux host
- Start the Kinetis Design Studio [beta] IDE and create a new C/C++ project
- Build an embedded application for a Freescale Kinetis device
- Debug an embedded application running on a Freescale Kinetis device
- Flash an embedded application directly to a Freescale Kinetis device
- Understand how file I/O is supported by semihosting
- Be aware of the differences between the Kinetis Design Studio [beta] software development tools and the GNU ARM Embedded Toolchain

## 2 Installing the Kinetis Design Studio [beta] software development tools

### 2.1 Installing on Windows

The Kinetis Design Studio [beta] software development tools are installed on Windows using the Windows Installer. Simply run `KDS-v1.0.1.exe`. The Windows installer will be started and on-screen instructions will guide you through the installation.

Figure 1: Windows installer



The Kinetis Design Studio [beta] software development tools will be installed under `C:\Freescale\KDS_1.0.1` when selecting either a *Typical* or *Complete* setup in the installer. However, when selecting a *Custom* setup a different installation directory (*install-dir*) can be specified to the installer.

The installer will offer to create a *Desktop* shortcut.

Alternatively, the Windows Installer can be launched from the command line and controlled using the standard Windows Installer [command line switches](#).

The following example launches the Windows Installer using a basic user interface to install the Kinetis Design Studio [beta] software development tools.

---

### Example 2.1 Launching the Windows Installer from the command line

---

```
KDS-v1.0.1.exe /qb
```

---

The basic user interface will not ask any questions but will pop up a progress bar. Note also that a *Desktop* shortcut will not be created. This can be created manually if required.

## 2.2 Installing on Linux

Three package files are provided for the installation of the Kinetis Design Studio [beta] software development tools on a Linux system:

1. `.rpm` - for installing on systems that use the *RPM* package manager (e.g. Red Hat and CentOS)
2. `.deb` - for installing on systems that use the *Debian* package manager (e.g. Ubuntu)
3. `.tar.bz2` - *TAR* archive, for those who don't want to use a package manager

### 2.2.1 Installing with Red Hat package manager (RPM)

The Kinetis Design Studio [beta] software development tools may be installed on an LSB (Linux Standard Base) compliant system using the `.rpm` package file:

---

#### Example 2.2 Installing With RPM

---

```
$ rpm -Uvh kinetis-design-studio-1.0.1-1.rpm
Preparing... ##### [100%]
1:Kinetis Design Studio [beta] ##### [100%]
```

---

This will install the Kinetis Design Studio [beta] software development tools to the default location of `/opt/Freescale/KDS_1.0.1`.

### 2.2.2 Installing with Debian package manager (DEB)

On Debian-like systems, including Ubuntu, the Kinetis Design Studio [beta] software development tools can be installed using the `.deb` package file:

---

#### Example 2.3 Installing with DPKG

---

```
$ dpkg -i kinetis-design-studio_1.0.1-1_i386.deb
(Reading database ... 209462 files and directories currently installed.)
Preparing to replace kinetis-design-studio 1.0.1 (using kinetis-design-studio_1.0.1-1_i386. ←
deb) ...
Unpacking replacement kinetis-design-studio ...
Setting up kinetis-design-studio (1.0.1) ...
```

---

This will install the Kinetis Design Studio [beta] software development tools to the default location of `/opt/Freescale/KDS_1.0.1`.

### 2.2.3 Extracting the Kinetis Design Studio [beta] software development tools from the TAR archive

Users wishing to use the Kinetis Design Studio [beta] software development tools without using a package manager, can extract it from the TAR archive file. The Kinetis Design Studio [beta] software development tools are fully relocatable; the extracted directory can be placed anywhere on a system and will work correctly.

---

**Example 2.4** Extracting the Kinetis Design Studio [beta] software development tools from the TAR archive

---

```
$ tar -xjf kds-v1.0.1-unknown-linux-gnu.tar.bz2
```

---



### 3 Building an embedded application

This section describes how to launch the Kinetis Design Studio [beta] IDE and create a new Kinetis Design Studio [beta] project to build an embedded application for a Freescale Kinetis device.

#### 3.1 Launching the Kinetis Design Studio [beta] IDE

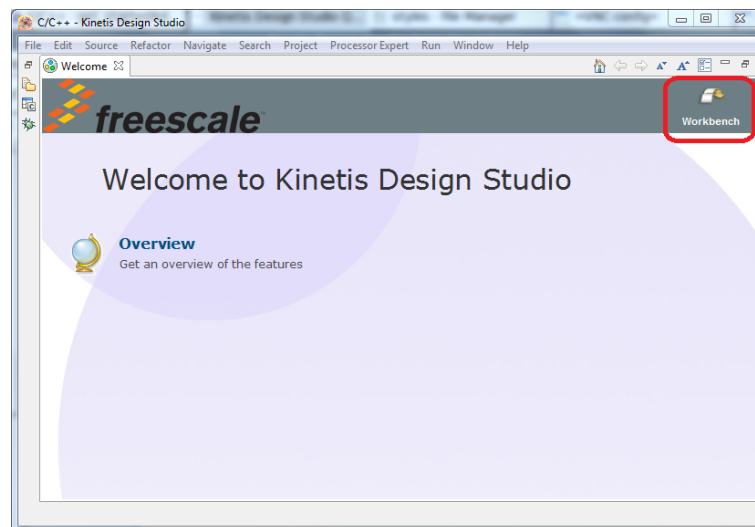
- On Windows double-click the Kinetis Design Studio [beta] IDE short cut, which may either be found on the *Desktop* or under the *Start Menu*. This assumes you selected a shortcut to be created by the Windows Installer.
- On Linux run the following command (which assumes Kinetis Design Studio [beta] was installed at the default location of `/opt/Freescale/KDS_1.0.1`):

**Example 3.1** Launching the Kinetis Design Studio [beta] IDE on Linux

```
$ /opt/Freescale/KDS_1.0.1/bin/kinetis-design-studio
```

A splash screen will appear while the IDE loads. You will be prompted to select a *Workspace*. This is the directory where your settings and projects will be stored. When selecting a new workspace, rather than an existing workspace found on your file system, you will see the Kinetis Design Studio [beta] IDE *Welcome* screen. The *Welcome* screen can be closed at this point by clicking on the *Workbench* icon in the top right corner (highlighted).

Figure 2: Kinetis Design Studio [beta] IDE welcome screen



**Note** If the Kinetis Design Studio [beta] IDE fails to launch with the error *Failed to create the Java Virtual Machine* then you may need to decrease the maximum size of the heap available to the IDE. Similarly, if the IDE reports "Out of memory" errors while running, then you may need to increase the maximum size of the heap available to the IDE.

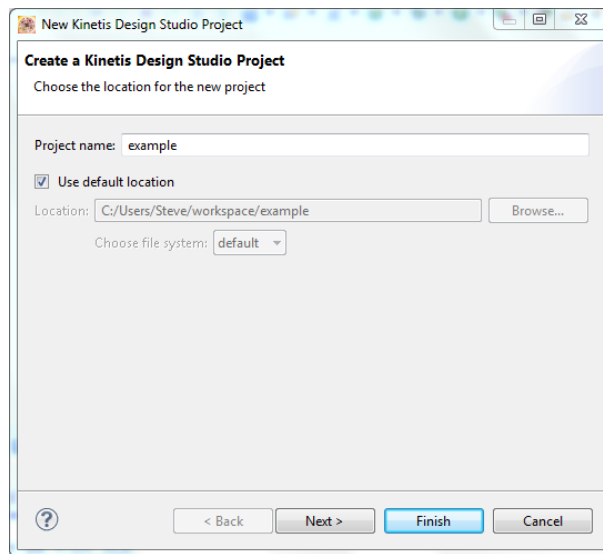
Adjusting the maximum size of the heap available to the IDE is made by changing the value of the `-Xmx` argument found in the file `install-dir\eclipse\eclipse.ini`.

It is set to `-Xmx512m` by default, giving the IDE a maximum heap size of 512MB.

#### 3.2 Creating a new Kinetis Design Studio [beta] project

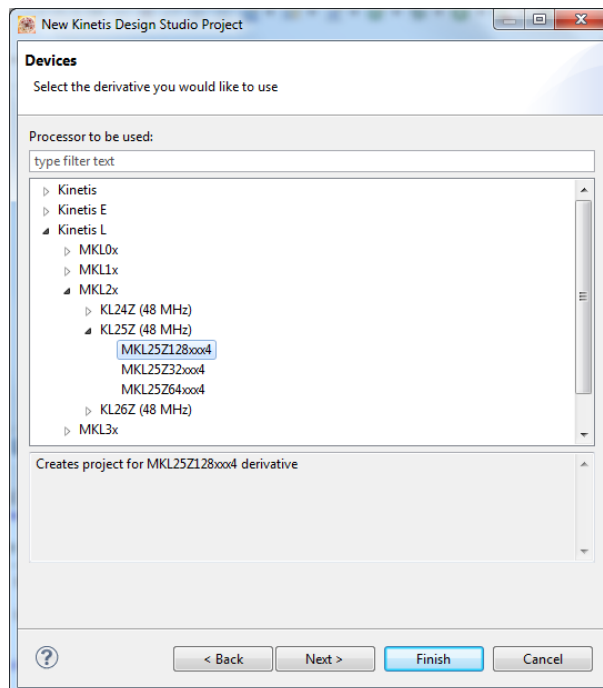
1. To create a new Kinetis Design Studio [beta] project, open the *File* menu, then select *New > Project > Kinetis Design Studio [beta] Project*. You will be presented with the *New Kinetis Design Studio [beta] Project* wizard.

Figure 3: New Kinetis Design Studio [beta] project wizard



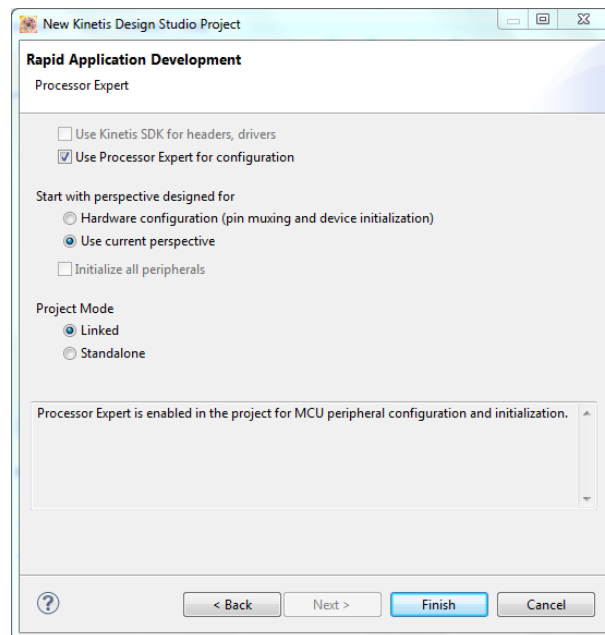
2. Enter a *Project name*. Specify the location of the project or select to use the default location. Then click *Next*.

Figure 4: Select whether to use Kinetis SDK



3. In the *Select the device derivative you would like to use* dialog, select the Freescale Kinetis device that you would like to use and click *Next*.

Figure 5: Rapid application development using Processor Expert



4. In the *Rapid Application Development* dialog, you can select *Use Processor Expert for configuration* to enable Processor Expert in the project for MCU peripheral configuration and initialization.
5. If you have selected to enable *Processor Expert* then you can choose the IDE perspective to start with and the *Project Mode*. The perspective to start with can either be the current perspective or a hardware perspective, which is designed for pin muxing and peripheral configuration. The project mode can either be *Linked* where static files (for instance cpu and peripheral init modules, I/O map and system files) are shared between all other projects or *Standalone* where all static files are copied under the project.

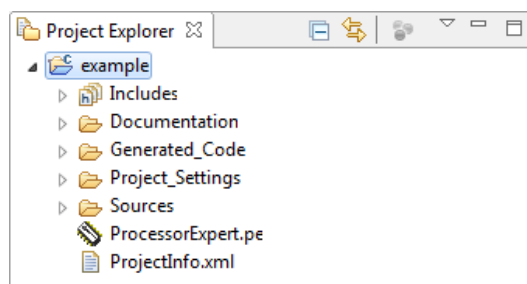
---

**Note** Only if you have selected the MK64FN1M0xxx12 device will you be able to select *Use Kinetis SDK for headers, drivers* to add Kinetis SDK source files, startup and linker files to the project.

---

6. Once all your selections have been made, click *Finish*.

Figure 6: Project Explorer



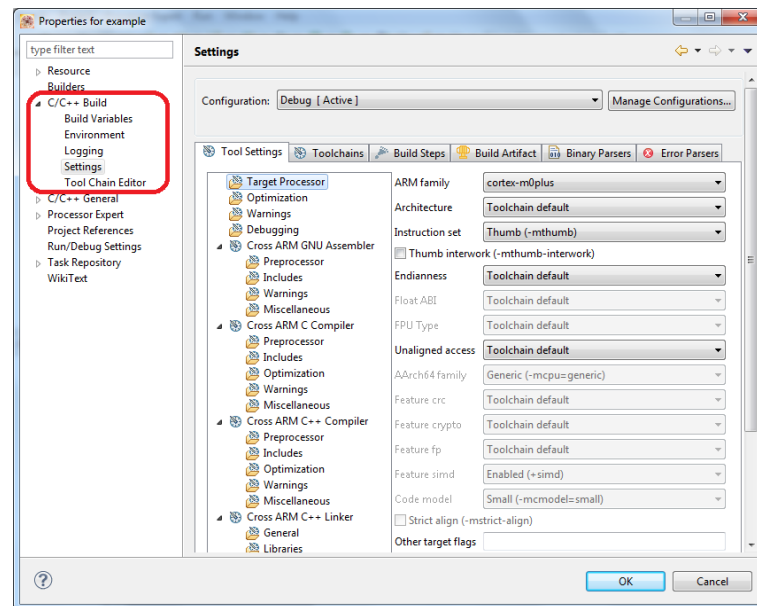
7. Your new Kinetis Design Studio [beta] project should now be shown under the *Project Explorer*.
8. To create a new source file under the project, right click on the project and select *New > Source File*. Alternatively you can drag and drop in existing source files, header files, directories under the project.

### 3.3 Configuring the project

A new Kinetis Design Studio [beta] project will be pre-configured and so you will be able to build the project for your Freescale Kinetis MCU based target board straight away. In the *Project Explorer* view, right click on the project and select *Build Project*.

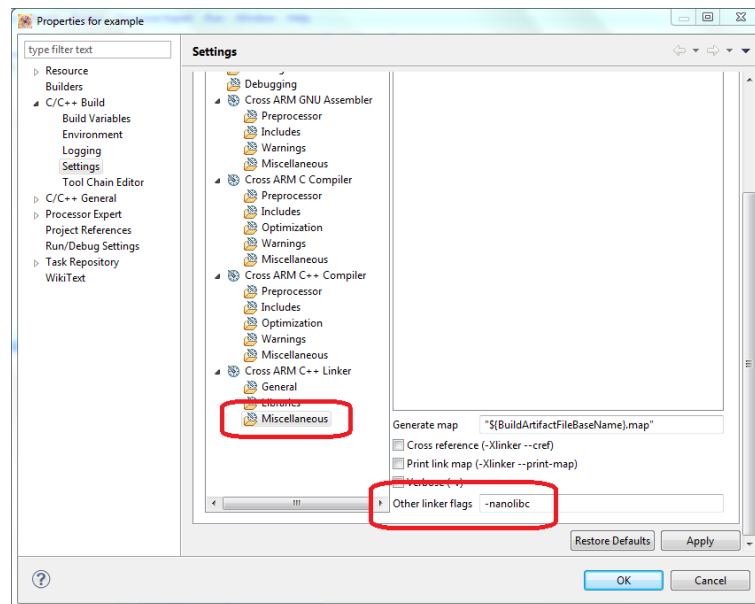
The configuration of the project can be adjusted by performing a right click on the project and selecting *Properties*. In the *Properties* wizard, the toolchain build settings can be changed by clicking on *C/C++ Build > Settings*.

Figure 7: Project Properties, C/C++ build settings



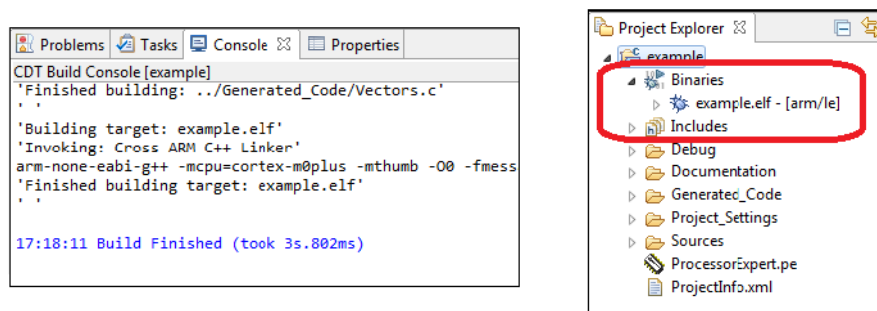
As an example, the Kinetis Design Studio [beta] software development tools supports two C runtime libraries, newlib and newlib-nano. By default a Kinetis Design Studio [beta] project will link an application with newlib. However, if the memory footprint of the embedded application is a concern, then linking with newlib-nano may help. This is done by providing the linker flag `-nanolibc` in the *Other linker flags* field under *Miscellaneous* of the *Cross ARM C++ Linker* folder.

Figure 8: Linking with newlib-nano



Once the project's settings have been adjusted, click OK. In the *Project Explorer* view, right click on the project and select *Clean Project*. Once cleaned select *Build Project*. Monitor the generated command lines used to build the embedded application in the build *Console* view. Any problems with the build will be reported under the *Problems* view. Assuming the build is successful, the generated binary will be listed under the project in the *Project Explorer*.

Figure 9: Build console and the generated binary



**Warning** When launching the debugger (discussed in the next chapter) the application binary will be programmed to the on-chip flash. Attempting to flash a binary without a properly defined flash configuration field may lock the device, rendering it unusable. Ensure all Kinetis projects explicitly set the flash configuration field and position it correctly.

Note that all Kinetis Design Studio [beta] projects created will contain auto-generated linker scripts and startup code that will explicitly set the flash configuration field to a safe value and position it correctly, so should not accidentally lock the device. For further information refer to the Freescale Application Note [AN4507 Using the Kinetis Security and Flash Protection Features](#).

## 4 Debugging an embedded application

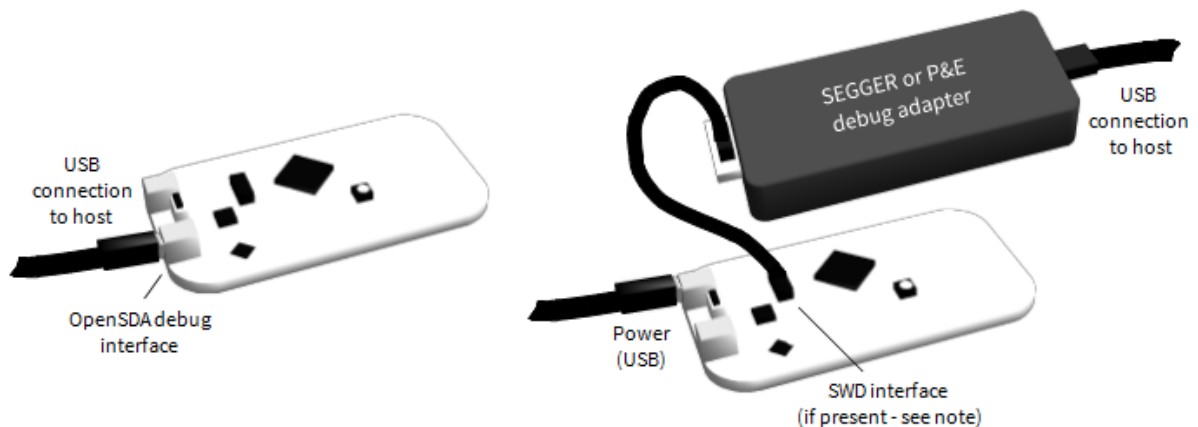
This section describes how to debug an embedded application on a Freescale Kinetis device using the Kinetis Design Studio [beta] IDE.

### 4.1 Development environment

The Kinetis Design Studio [beta] software development tools supports the following debug adapters for debugging applications on a Freescale Kinetis device:

1. On-board *OpenSDA* debug interface running the *MBED CMSIS-DAP* firmware<sup>1</sup>
2. SEGGER *J-Link* and *J-Trace* debug adapters
3. P&E *USB Multilink Universal* and *USB Multilink Universal FX* debug adapters

Figure 10: Supported debug adapters



**Note** Not all Freescale Kinetis boards have a SWD interface or SWD interface header present on the board. Refer to the Freescale user manual for the board you are using to check these details.

**Note** Both SEGGER and P&E provide firmware for the OpenSDA debug interface.

The P&E firmware is provided in the Kinetis Design Studio [beta] software development tools under `install-dir\pemicro\opensda\MSD-DEBUG-freescale_board_Pemicro_v114.SDA`, where `freescale_board` is the Freescale board that you are using.

The SEGGER firmware is not delivered with the Kinetis Design Studio [beta] software development tools, however it can be downloaded from the SEGGER web site at <http://www.segger.com/opensda.html>.

The procedure to debug an embedded application using the SEGGER OpenSDA firmware is the same as if you were using either a SEGGER J-Link or J-Trace debug adapter. Similarly the procedure to debug an embedded application using the P&E OpenSDA firmware is largely the same as if you were using either a P&E *USB Multilink Universal* or *USB Multilink Universal FX* debug adapter.

<sup>1</sup>Out-of-the-box Freescale developments boards have the on-board OpenSDA loaded with the *MBED CMSIS-DAP* firmware. See section Appendix C for further details.

## 4.2 Debugging with the Kinetis Design Studio [beta] IDE

1. In the *Project Explorer*, right click on the project containing the embedded application that you want to debug. Select *Debug As > Debug Configurations*.


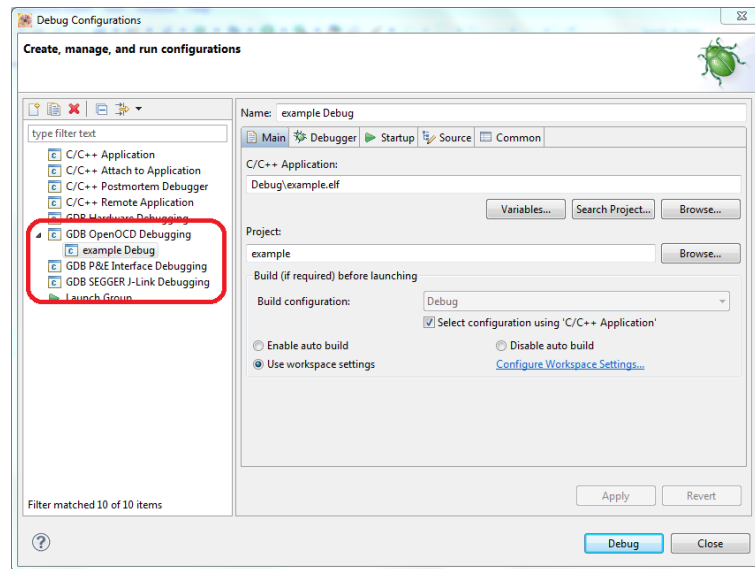
Alternatively, click on the drop down icon of the  button from the main toolbar and select *Debug Configurations*.

Figure 11: Create a new debug configuration

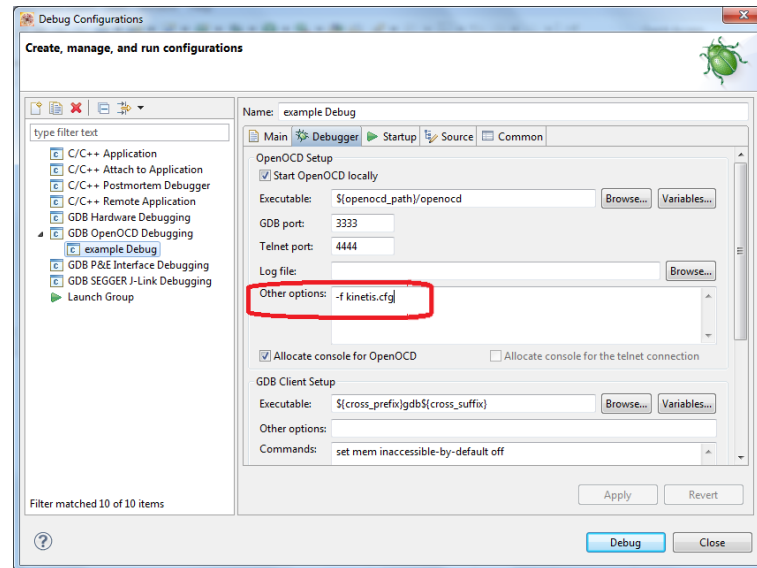


**Note** Appendix C provides details on the device drivers which are needed to work with the chosen debug interface.

- (a) If you are using the on-board *OpenSDA* debug interface running the *MBED CMSIS-DAP* firmware then right click on *GDB OpenOCD Debugging* and select *New* to create a new debug configuration. This configuration will use the *OpenOCD* (Open On-Chip debugger) GDB server to interface with the target.
  - (b) If you are using a SEGGER *J-Link* or *J-Trace* debug adapter, or if you are using the on-board *OpenSDA* debug interface running the SEGGER firmware then right click on *GDB SEGGER J-Link Debugging* and select *New* to create a new debug configuration. This configuration will use the SEGGER *J-Link* GDB server to interface with the target.
  - (c) If you are using a P&E *USB Multilink Universal* or *USB Multilink Universal FX* debug adapter, or if you are using the on-board *OpenSDA* debug interface running the P&E firmware then right click on *GDB P&E Interface Debugging* and select *New* to create a new debug configuration. This configuration will use the P&E GDB server to interface with the target.
2. Select the *Main* tab and check that the correct *Project* and *C/C++ Application* is selected. Next, select the *Debugger* tab.
    - (a) If you are using the *OpenOCD* debug interface then insert the options `-f kinetis.cfg` under the *Other options* field of the *OpenOCD* section of the *Debugger* tab.

This will connect to the remote target as a *localhost*.

Figure 12: Debugger tab for a GDB OpenOCD Debugging configuration

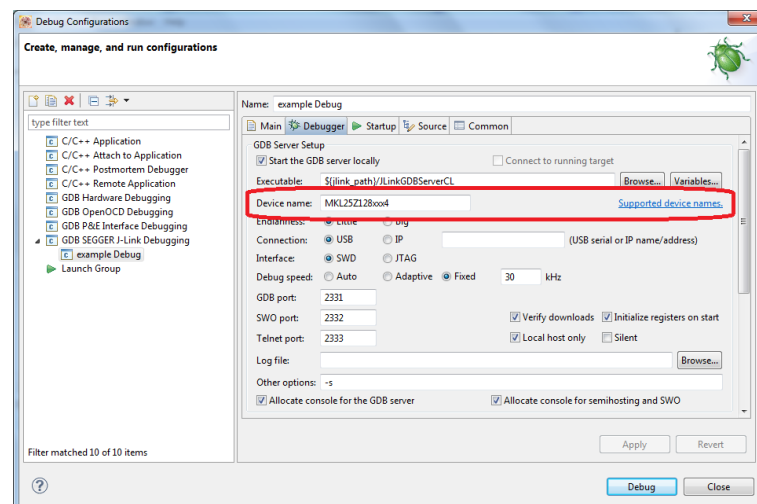


- (b) If you are using a SEGGER J-Link debug interface then enter the *Device name* for your Freescale Kinetis device. Use the link *Supported device names* to help you with your selection or refer to Appendix B.

**Note** SEGGER software tries to protect users from accidentally permanently locking their devices by providing two variants of each Freescale Kinetis device. The default will not allow disabling mass erase, while the alternate (labelled "allow security") will. As a result it is recommended to *not* use the "allow security" devices without good reason.

The remaining fields of the *Debugger* tab can be left with the default entries. This will connect to the remote target as a *localhost*.

Figure 13: Debugger tab for a GDB SEGGER J-Link Debugging configuration

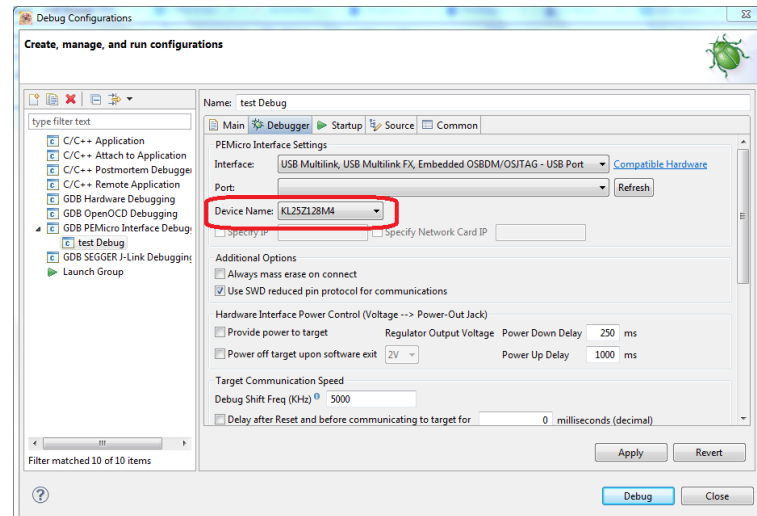


- (c) If you are using a P&E debug interface then select the *Device name* for your Freescale Kinetis device from the drop down list. The remaining fields of the *Debugger* tab can be left with the default entries<sup>2</sup>. This will connect to the remote target as a *localhost*.

<sup>2</sup> If you are using the P&E OpenSDA firmware then you need to select the *Interface* to be *OpenSDA Embedded Debug - USB Port*.



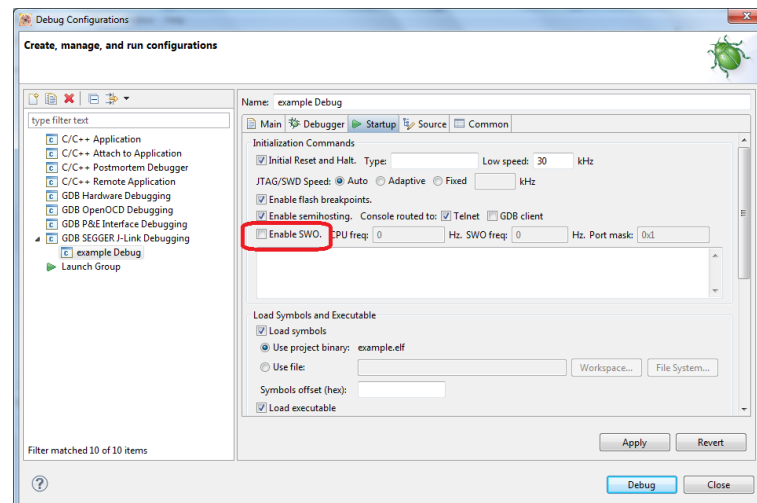
Figure 14: Debugger tab for a GDB P&amp;E Interface Debugging configuration



### 3. Next select the *Startup* tab.

- Users of the OpenOCD debug interface do not need to change any of the default settings under the *Startup* tab. Click on *Apply* and hit the *Debug* button. This will launch the debugger.
- For users of the SEGGER *J-Link* debug interface then de-select the *Enable SWO* radial button, as this is not currently supported by the Kinetis Design Studio [beta] software development tools.

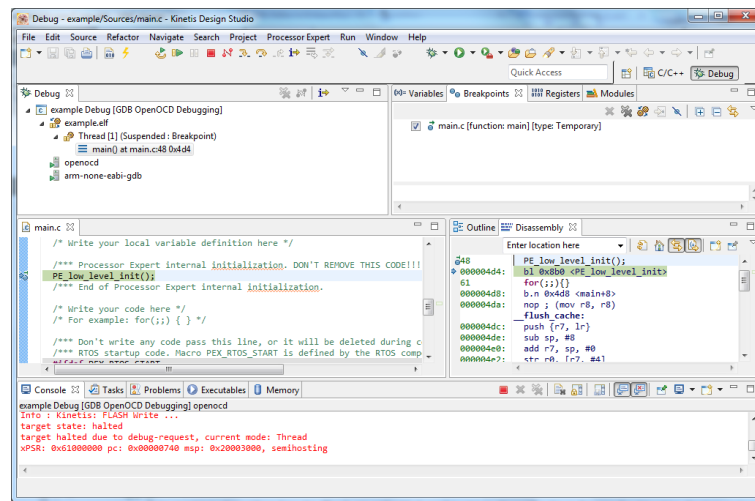
Figure 15: Startup tab for a GDB SEGGER J-Link Debugging configuration



The remaining fields of the *Startup* tab should be correctly set up ready for you to *Apply* the debug configuration and hit the *Debug* button. This will launch the debugger.

- Users of the P&E debug interface do not need to change any of the default settings under the *Startup* tab. Click on *Apply* and hit the *Debug* button. This will launch the debugger.
- ### 4. When you have launched a debug of your embedded application, you will be prompted to open the Kinetis Design Studio [beta] IDE's *Debug Perspective*. This assumes you were not already in the *Debug Perspective* at that moment. Select *Yes* to switch perspective. The *Debug Perspective* will open and the embedded application will break on the breakpoint set on **main**.











Figure 16: Kinetis Design Studio [beta] IDE debug perspective



### 4.3 Controlling the application in the debugger

Once the breakpoint on **main** has been reached, you will be able to step through lines of code, double click on the left of a code statement to set a breakpoint, resume execution, pause execution, restart or terminate the debug session.

The table below lists the main buttons which are used to control the execution of an application in the debugger.

Button	Action	Description
	Resume	Resume/continue execution of the suspended application.
	Suspend	Suspend/pause the execution of the application.
	Instruction Stepping Mode	Activate instruction step mode when selected, otherwise stepping is performed on lines of C/C++ code. The Debugger switches to the instruction stepping mode automatically when the <i>Disassembly</i> view has focus.
	Step Over	Execute the current line, following execution inside a routine.
	Step Into	Execute the current line, including any routines, and proceed to the next statement.
	Step Return	Continue execution to the end of the current routine, then follow execution to the routine's caller.
	Restart	Restart the selected debug session.
	Disconnect	Detaches the debugger from the selected debug session.
	Terminate	Terminate the selected debug session.
	Debug	Launch the last debug configuration or use the drop down list to select a debug configuration. Currently using the <i>Debug</i> button to launch the last debug configuration does not work as expected when used under the IDE's C/C++ perspective. Refer to the Kinetis Design Studio [beta] <i>Release Notes</i> for further details.

When stopping on a breakpoint or with the execution paused, the state of the target can be examined by viewing the current

callstack in the *Debug* view, ARM processor core registers, Disassembly, Memory, Breakpoints which have been set, local variables in *Variables* view, global variables in *Expressions* view and more.

Select *Window > Show View* to open a specific view if the view is not currently open in the Kinetis Design Studio [beta] IDE's *Debug Perspective*.

## 5 Flashing an embedded application

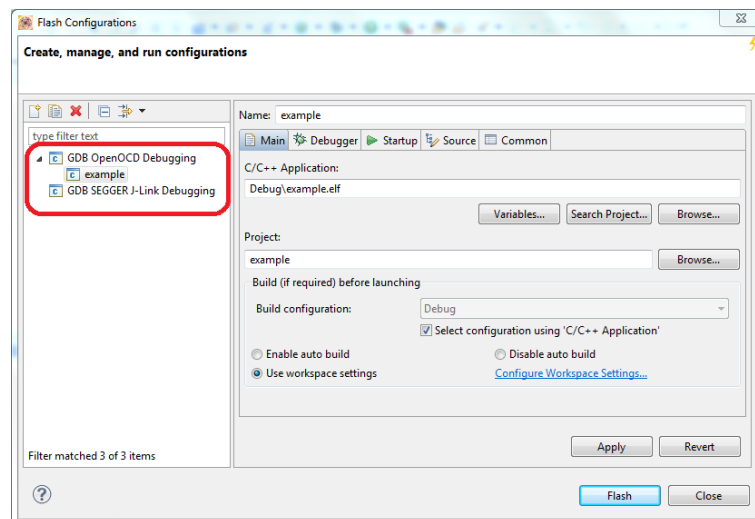
This section describes how to flash an embedded application directly to a Freescale Kinetis device using the Kinetis Design Studio [beta] IDE.

### 5.1 Flashing with the Kinetis Design Studio [beta] IDE

1. In the *Project Explorer*, under the project which was used to build the embedded application, right click on the `elf` executable file (found under *Binaries*) and select *Flash from file*.

Alternatively, click on the  button from the main toolbar.

Figure 17: Flash configuration



- (a) If you are using the *OpenOCD* debug interface then under *GDB OpenOCD Debugging* you will see your existing debug configuration.
- (b) If you are using the *SEGGER J-Link* debug interface then under *GDB SEGGER J-Link Debugging* you will see your existing debug configuration.

---

**Note** Currently this feature is not available for the P&E debug interface.

---

2. Regardless of the debug adapter being used, select the *Main* tab and confirm that the correct *Project* and *C/C++ Application* is selected.
3. Next, select the *Debug* tab. This should already be set up correctly. The *Startup* tab is not relevant for flashing and so does not need to be reviewed.
4. Finally, click *Flash*. It should take a few moments to flash the embedded application to your Freescale Kinetis device. Once completed power cycle the the board. The embedded application should boot and run.

## A Technical notes

This section describes some of the terms and concepts used in this document which relate to the Kinetis Design Studio [beta] software development tools.

### A.1 Differences with the GNU ARM Embedded toolchain

The GNU ARM Embedded toolchain is a GNU toolchain targeted at embedded ARM® processors, namely Cortex™-R/Cortex™-M processor families. The toolchain is maintained by ARM®.

Both the GNU ARM Embedded toolchain and the Kinetis Design Studio [beta] software development tools are derived from the GNU tools. Hence, they are both GNU-compatible. This means any GNU specific features/behaviours used in an application will be correctly handled by the Kinetis Design Studio [beta] software development tools. However there are several differences between the Kinetis Design Studio [beta] software development tools and the GNU ARM Embedded Toolchain to note:

- The Kinetis Design Studio [beta] software development tools only supports 32-bit Cortex™-M based Kinetis devices
- The GNU ARM Embedded toolchain requires additional libraries and linker options that aren't provided or required by the Kinetis Design Studio [beta] software development tools (for instance references to "rdimon" and "nosys"). Attempting to build applications with such references using the Kinetis Design Studio [beta] software development tools will result in build errors.
- The Kinetis Design Studio [beta] software development tools supports two C runtime libraries, newlib and newlib-nano. By default an application is linked with newlib. However, newlib-nano can be selected using the `-nanolibc` linker option. The GNU ARM Embedded toolchain also provides newlib-nano, however the process to build applications using it is different.

### A.2 Semihosting

Semihosting is a mechanism allowing ARM applications running under the control of a debug agent to access resources on the host machine. For example, it is often useful during development to have a semihosted application read to, and write from, files stored on the host machine using standard C library functions. Semihosting is built-in to the C libraries included with the Kinetis Design Studio [beta] software development tools.

Semihosting must be supported and enabled in the debug agent. During execution the debug agent watches for semihosting requests. When such a request is encountered the target application is stopped and the debug agent performs the requested operation.

## B Supported Freescale Kinetis MCUs

Family	SEGGER device name	Core
K11D	MK11DX128xxx5	Cortex-M4
	MK11DX256xxx5	Cortex-M4
	MK11DN512xxx5	Cortex-M4
K12D	MK12DX128xxx5	Cortex-M4
	MK12DX256xxx5	Cortex-M4
	MK12DN512xxx5	Cortex-M4
K21D	MK21DX128xxx5	Cortex-M4
	MK21DX256xxx5	Cortex-M4
	MK21DN512xxx5	Cortex-M4
K22D	MK22DX128xxx5	Cortex-M4
	MK22DX256xxx5	Cortex-M4
	MK22DN512xxx5	Cortex-M4
K21F	MK21FN512xxx12	Cortex-M4
	MK21FN1M0xxx12	Cortex-M4
K22F	MK22FN256xxx12	Cortex-M4
	MK22FN512xxx12	Cortex-M4
K24F	MK24FN1M0xxx12	Cortex-M4
K63F	MK63FN1M0xxx12	Cortex-M4
K64F	MK64FN1M0xxx12	Cortex-M4
KL14Z	MKL14Z32xxx4	Cortex-M0+
	MKL14Z64xxx4	Cortex-M0+
KL15Z	MKL15Z32xxx4	Cortex-M0+
	MKL15Z64xxx4	Cortex-M0+
	MKL15Z128xxx4	Cortex-M0+
KL16Z	MKL16Z32xxx4	Cortex-M0+
	MKL16Z64xxx4	Cortex-M0+
	MKL16Z128xxx4	Cortex-M0+
	MKL16Z256xxx4	Cortex-M0+
KL24Z	MKL24Z32xxx4	Cortex-M0+
	MKL24Z64xxx4	Cortex-M0+
KL25Z	MKL25Z32xxx4	Cortex-M0+
	MKL25Z64xxx4	Cortex-M0+
	MKL25Z128xxx4	Cortex-M0+
KL26Z	MKL26Z32xxx4	Cortex-M0+
	MKL26Z64xxx4	Cortex-M0+
	MKL26Z128xxx4	Cortex-M0+
	MKL26Z256xxx4	Cortex-M0+
KL34Z	MKL34Z64xxx4	Cortex-M0+
KL36Z	MKL36Z64xxx4	Cortex-M0+
	MKL36Z128xxx4	Cortex-M0+
	MKL36Z256xxx4	Cortex-M0+
KL46Z	MKL46Z128xxx4	Cortex-M0+
	MKL46Z256xxx4	Cortex-M0+

## C Driver Installation

When using either the OpenOCD, SEGGER J-Link or P&E Multilink debug interface, the relevant device drivers need to have been installed.

### C.1 Installing on Windows

The Kinetis Design Studio [beta] Windows Installer will by default install the SEGGER J-Link and the P&E Multilink device drivers and so these do not need to be separately installed.

Out-of-the-box Freescale boards will have the *MBED CMSIS-DAP* firmware loaded on the on-board OpenSDA debug interface. This firmware requires the MBED serial port driver is installed. The Kinetis Design Studio [beta] Windows Installer does not do this and so it will need to be installed separately. The table below provides a link which explains how to install the MBED serial port driver. The table also describes how to manually install the SEGGER J-Link and P&E Multilink drivers if this is required.

Driver	Installation
MBED Windows serial port driver	Follow the instructions at <a href="http://mbed.org/handbook/Windows-serial-configuration">http://mbed.org/handbook/Windows-serial-configuration</a>
SEGGER J-Link	Run the J-Link driver installers found under: <code>install-dir\segger\USBDriver\InstDrivers.exe</code> <code>install-dir\segger\USBDriver\CDC\InstDriversCDC.exe</code>
P&E Multilink driver	Before connecting the P&E Multilink to the PC, download and run the P&E Multilink installer. The installer program can be downloaded from <a href="http://www.pemicro.com/support/downloads_find.cfm">http://www.pemicro.com/support/downloads_find.cfm</a>

### C.2 Installing on Linux

When the Kinetis Design Studio [beta] software development tools are installed on a Linux system, it will contain a *udev* rules file for each of the OpenOCD, SEGGER J-Link the P&E Multilink debug interfaces. The table below describes how these can be installed.

Driver	udev rules file	Notes
OpenOCD	<code>install-dir/openocd/openocd.udev</code>	<ul style="list-style-type: none"> <li>• Copy the <i>udev</i> file into the configuration directory (for example under <code>/etc/udev/rules.d/</code>)</li> <li>• Rename the file to <code>99-openocd.rules</code> (for example)</li> <li>• Optionally the permissions allocated by the rules file can be adjusted. By default this requires users to be in the <i>plugdev</i> group.</li> <li>• Run the command <b>udevadm control --reload-rules</b> to instruct <i>udev</i> to reload its rules</li> </ul>

Driver	udev rules file	Notes
Segger J-Link	<code>install-dir/segger/99-jlink.rules</code>	<ul style="list-style-type: none"><li>• Copy the <i>udev</i> file into the configuration directory (for example under <code>/etc/udev/rules.d/</code>)</li><li>• Run the command <b>udevadm control --reload-rules</b> to instruct <i>udev</i> to reload its rules</li></ul>
P&E Multilink	<code>install-dir/pemicro/drivers/libusb_64_32/28-pemicro.rules</code>	Run the <code>setup.sh</code> script found under the same directory

## D CodeWarrior project migration

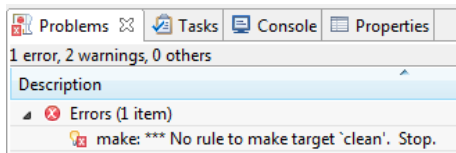
The *CodeWarrior to Kinetis Design Studio [beta] project migration assistant* assists in the migration of CodeWarrior projects into a form that is configurable by the Kinetis Design Studio [beta] IDE. It is necessary to manually alter certain project content to achieve a successful build in the Kinetis Design Studio [beta] IDE.

Existing CodeWarrior projects may be migrated into a form suitable for use with the Kinetis Design Studio [beta] IDE. To do this, first launch the Kinetis Design Studio [beta] IDE using a workspace which contains an existing CodeWarrior project or import an existing CodeWarrior project (File > Import > General > Existing Projects into Workspace) into a new workspace.

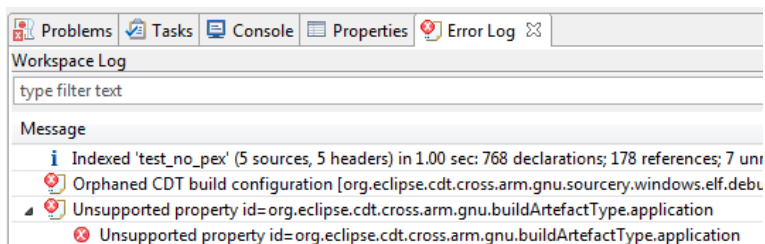
### D.1 Examples of warnings from an existing or imported CodeWarrior project

An existing CodeWarrior project or one that has just been imported may exhibit the following warnings under the Kinetis Design Studio [beta] IDE:

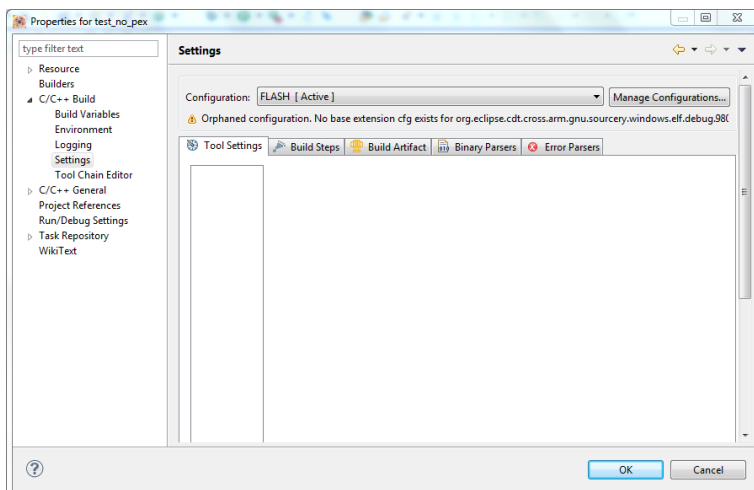
- *Problems view* (Windows > Show View > Problems)



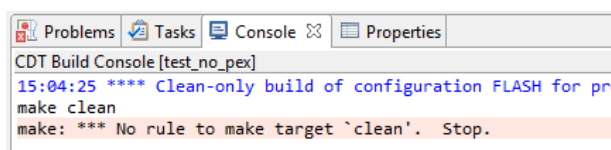
- *Error Log view* (Windows > Show View > Other > General > Error Log)



- The *Project Properties* (Project > Properties > C/C++ Build > Settings > Tool Settings) will appear unpopulated.



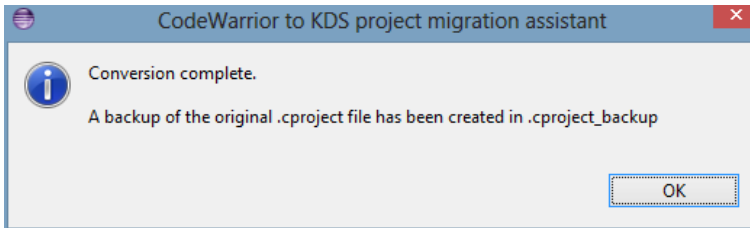
- Attempting to build the project results in failure with a message in the *CDT Build Console view* (Windows > Show View > Console):





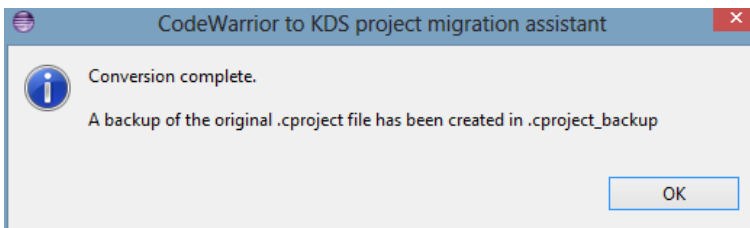
## D.2 Converting a CodeWarrior project file

Next, open the *Project Explorer* view (Windows > Show View > Project Explorer) and open the project's context menu (right click) and select *Convert CodeWarrior project file....* The *CodeWarrior to KDS project migration assistant* dialog appears displaying the following question:



Click Yes to start the conversion.

Once completed, the *CodeWarrior to Kinetis Design Studio [beta] project migration assistant* dialog will then display the following information:



A log of the conversion process is stored in the file `KDSConverter.log` in the project's root.

---

**Note** Once a CodeWarrior project file has been converted, running the converter subsequent times has no effect.

---

## D.3 Building a migrated CodeWarrior project

The project can be built and cleaned in the normal way (Project > Build Project, Clean...). Depending on the content of the CodeWarrior project it may be necessary to switch certain settings to use Kinetis Design Studio [beta] headers and libraries, rather than CodeWarrior headers and libraries.

## D.4 Common build errors after migration

When attempting to build a migrated project the following problems maybe encountered. Solutions to these problems are offered as a temporary fix to demonstrate the build working. For complete migration it may be necessary to manually convert some of the project content to point to Kinetis Design Studio [beta] headers and libraries.

- `ansi_parms.h`: No such file or directory:

```

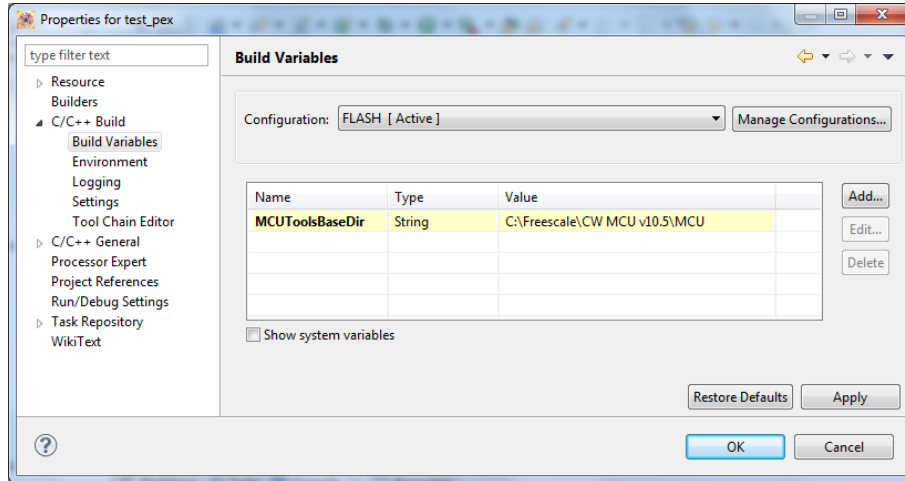
CDT Build Console [test_pex]
'Building file: ../Project_Settings/Startup_Code/__arm_end.c'
'Invoking: Cross ARM C Compiler'
arm-none-eabi-gcc -mcpu=cortex-m0plus -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -f
../Project_Settings/Startup_Code/__arm_end.c:41:24: fatal error: ansi_parms.h: No such file or directory
}
^
compilation terminated.
make: *** [Project_Settings/Startup_Code/__arm_end.o] Error 1

```

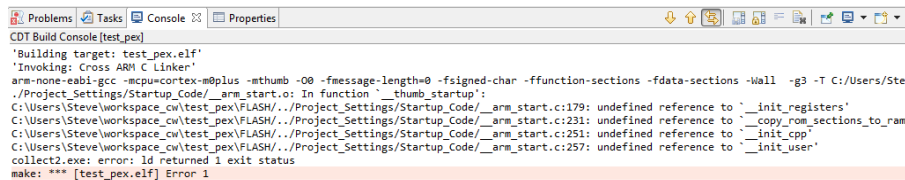
This problem may be caused by the use of the environment variable `${MCUToolsBaseDir}` in the compiler include paths (Project > Properties > C/C++ Build > Settings > Cross ARM C Compiler > Includes).

To fix this problem either define the variable so it is accessible from within the Eclipse environment or replace the include path directives to use absolute paths.

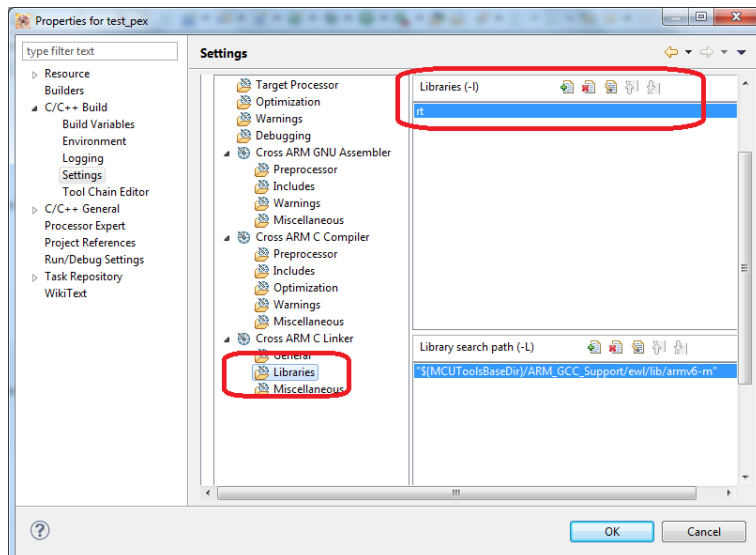
A convenient method to define an environment variable for use in the build is to define it in Project > Properties > C/C++ Build > Build Variables. For example press Add... and in *Variable name* add `MCUToolsBaseDir` and in *Value* add the directory to the CodeWarrior support libraries, for example `C:\Freescale\CW MCU v10.5\MCU`.



- collect2.exe: error: ld returned 1 exit status



This problem may be caused by the missing library `librt`. To fix this problem add the `librt` library to the project by selecting Project > Properties > C/C++ Build > Settings > Cross ARM C Linker > Libraries > Libraries (-l) and add the library "rt" (without quotes).



## E Disclaimer

Freescale, the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Kinetis and Processor Expert are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. ARM is the registered trademark of ARM Limited.

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

### How to contact Freescale:

Corporate Headquarters  
Freescale Semiconductor, Inc.  
6501 William Cannon Drive West  
Austin, Texas 78735  
U.S.A.

World Wide Web <http://www.freescale.com/kds>  
Technical Support <http://www.freescale.com/support>