

Kinetis Design Studio v2.0.0 User Guide

SOMNIUM® Document Number: SOMN-KDS-0001
Freescale Document Number: KDS200UG Rev 0, 11/14

Copyright © 2013-2014 SOMNIUM® Technologies Limited

Contents

1	Introduction	1
1.1	Main features	1
1.2	Supported host operating systems	1
1.3	Supported Freescale devices	1
1.4	License information	1
1.5	Objectives of this document	2
2	Installing the Kinetis Design Studio software development tools	2
2.1	Installing on Windows	2
2.1.1	Installing from the command line	2
2.2	Installing on Linux	2
2.2.1	Installing with Red Hat package manager (RPM)	2
2.2.2	Installing with Debian package manager (DEB)	3
2.3	Installing KSDK	3
3	Building an embedded application	5
3.1	Launching the Kinetis Design Studio IDE	5
3.2	Creating a new Kinetis Design Studio project	6
3.3	Configuring the project	7
4	Debugging an embedded application	8
4.1	Development environment	8
4.2	Debugging with the Kinetis Design Studio IDE	9
4.3	Controlling the application in the debugger	13
5	Flashing an embedded application	14
5.1	Flashing with the Kinetis Design Studio IDE	14
A	Technical notes	15
A.1	Differences with the GNU ARM Embedded toolchain	15
A.2	Semihosting	15
B	Driver installation	16
B.1	Installing on Windows	16
B.2	Installing on Linux	16
C	Notices	18

List of Figures

1	Installing KSDK	4
2	Kinetis Design Studio IDE welcome screen	5
3	New Kinetis Design Studio project wizard	6
4	Project Explorer	6
5	Project Properties, C/C++ build settings	7
6	Linking with newlib-nano	8
7	Build console and the generated binary	8
8	Supported debug adapters	9
9	Create a new debug configuration	10
10	Debugger tab for a <i>GDB OpenOCD Debugging</i> configuration	11
11	Debugger tab for a <i>GDB SEGGER J-Link Debugging</i> configuration	11
12	Debugger tab for a <i>GDB P&E Interface Debugging</i> configuration	12
13	Startup tab for a <i>GDB SEGGER J-Link Debugging</i> configuration	12
14	Kinetis Design Studio IDE debug perspective	13
15	Flash configuration	14

1 Introduction

Welcome to the Kinetis Design Studio software development tools, a GNU/Eclipse-based development environment for Freescale Kinetis devices.

1.1 Main features

The Kinetis Design Studio software development tools are provided free of charge and include the following features:

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

1.2 Supported host operating systems

The Kinetis Design Studio software development tools support the following host operating systems:

- Microsoft® Windows® 7 and Windows® 8 (all editions). Windows hosted variants of the Kinetis Design Studio 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 14.04 LTS. Linux-hosted variants of the Kinetis Design Studio software development tools are distributed as 64-bit binaries, which will not work on 32-bit systems.

Note The Kinetis Design Studio software development tools may work on older versions of these systems, but this has not been tested and is not supported.

To use the Kinetis Design Studio software development tools the following minimum 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 software development tools support devices in the Kinetis product range, see individual component documentation for complete lists of supported devices.

1.4 License information

The Kinetis Design Studio 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 software development tools:

- Install the Kinetis Design Studio software development tools on either a Windows or Linux host
- Start the Kinetis Design Studio 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 software development tools and the GNU ARM Embedded Toolchain

2 Installing the Kinetis Design Studio software development tools

2.1 Installing on Windows

The Kinetis Design Studio software development tools are installed on Windows using Windows Installer. Simply double-click the installer, `KDS-v2.0.0.exe`. The Windows Installer will be started and on-screen instructions will guide you through the installation.

2.1.1 Installing from the command line

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

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

Example 2.1 Launching Windows Installer from the command line

```
KDS-v2.0.0.exe /qb
```

The basic user interface will not ask any questions but will display a progress bar.

2.2 Installing on Linux

Package files are provided for installing the Kinetis Design Studio 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)

2.2.1 Installing with Red Hat package manager (RPM)

The Kinetis Design Studio 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

```
$ sudo rpm -Uvh kinetis-design-studio-2.0.0-1.x86_64.rpm
Preparing...                               ##### [100%]
1:Kinetis Design Studio                    ##### [100%]
```

This will install the Kinetis Design Studio software development tools to the default location (/opt/Freescale/KDS_2.0.0).

2.2.2 Installing with Debian package manager (DEB)

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

Example 2.3 Installing with DPKG

```
$ sudo dpkg -i kinetis-design-studio_2.0.0-1_amd64.deb
(Reading database ... 209462 files and directories currently installed.)
Preparing to replace kinetis-design-studio 2.0.0 (using kinetis-design-studio_2.0.0-1_amd64 ←
.deb) ...
Unpacking replacement kinetis-design-studio ...
Setting up kinetis-design-studio (2.0.0) ...
```

This will install the Kinetis Design Studio software development tools to the default location (/opt/Freescale/KDS_2.0.0).

2.3 Installing KSDK

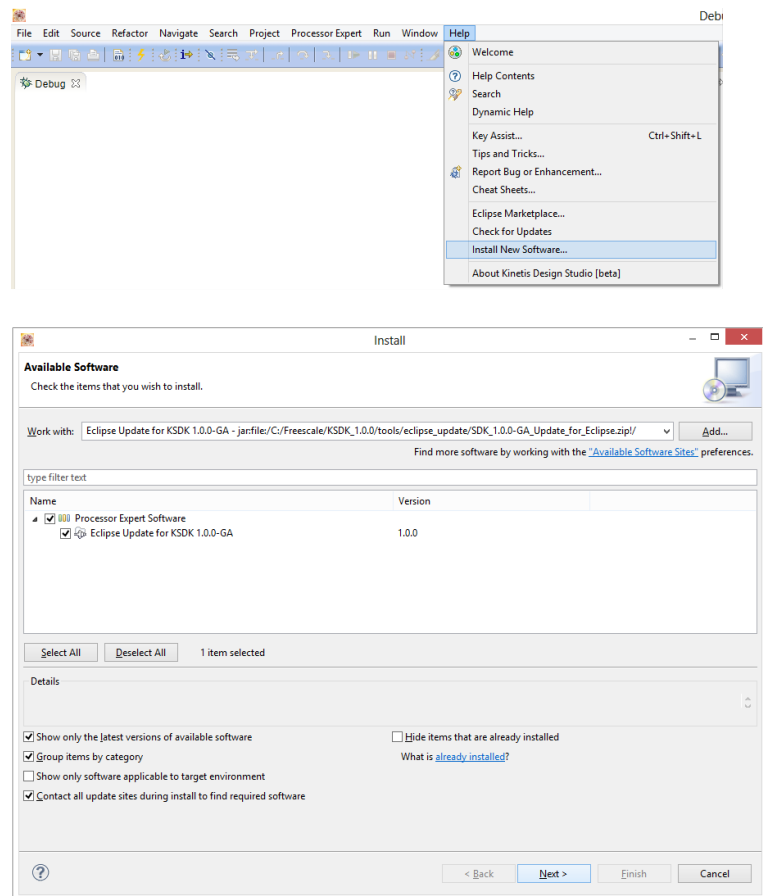
Some Kinetis Design Studio IDE functionality requires the Kinetis SDK (KSDK). To activate this functionality the KSDK must be installed both on the host machine and into the Kinetis Design Studio IDE. Installation on the host machine is achieved by downloading and running the KSDK installer (see <http://www.freescale.com/ksdk>).

The KSDK installation directory contains an Eclipse update site (a .zip file in the tools\eclipse_update directory). This needs to be installed into the Kinetis Design Studio IDE using the *Install New Software* wizard.

Note Users with the Kinetis Design Studio IDE installed in a read-only location, which is the default for Linux systems, must launch the Kinetis Design Studio IDE with administrative/root privileges to install the KSDK.

1. Start the Kinetis Design Studio IDE (section 3.1)
2. Start the *Install New Software* wizard, from the *Help* menu.
3. On the *Available Software* screen, click *Add...*, to add a new location to *Work with*.
4. Click *Archive*, and select to the .zip file provided by KSDK. Then click *OK*.
5. The *Work with* box should now be populated, and the list of available software should show *Eclipse Update for KSDK* in the *Processor Expert Software* category.
6. Check the checkbox to install *Eclipse Update for KSDK*.
7. Click *Next* and follow the on-screen instructions to complete the installation.

Figure 1: Installing KSDK



3 Building an embedded application

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

3.1 Launching the Kinetis Design Studio IDE

- On Windows double-click the Kinetis Design Studio IDE in the *Start Menu*. A *Desktop* shortcut may also have been created, if requested during installation.
- On Linux run the following command (which assumes Kinetis Design Studio was installed at the default location of `/opt/Freescale/KDS_2.0.0`):

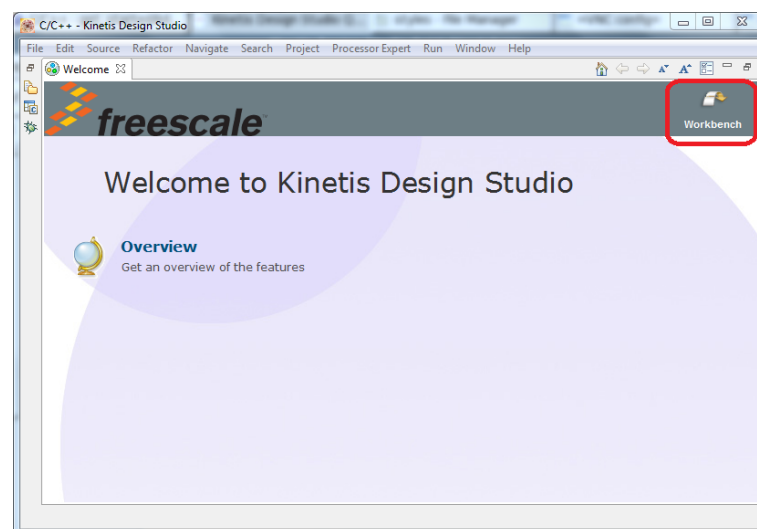
Example 3.1 Launching the Kinetis Design Studio IDE on Linux

```
$ /opt/Freescale/KDS_2.0.0/eclipse/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 opening a new workspace the Kinetis Design Studio IDE will display a *Welcome* screen. This can be closed at this point by clicking on the *Workbench* icon in the top right corner.

Figure 2: Kinetis Design Studio IDE welcome screen



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

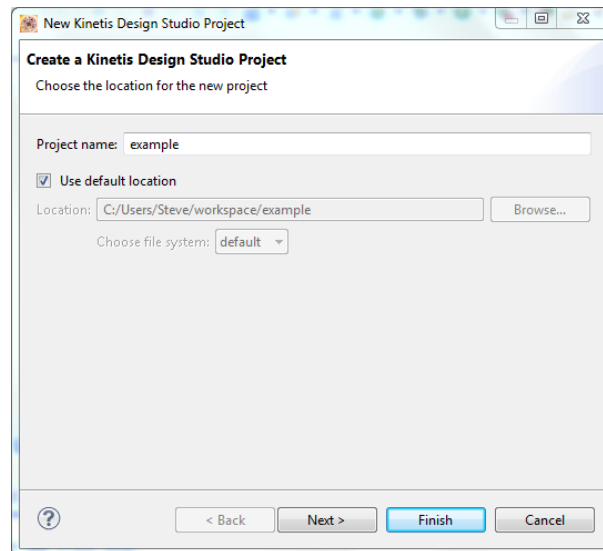
This is achieved by changing the value of the `-Xmx` argument found in the file `install-dir\eclipse\kinetis-design-studio.ini`.

The default is `-Xmx512m`, giving the IDE a maximum heap size of 512MB.

3.2 Creating a new Kinetis Design Studio project

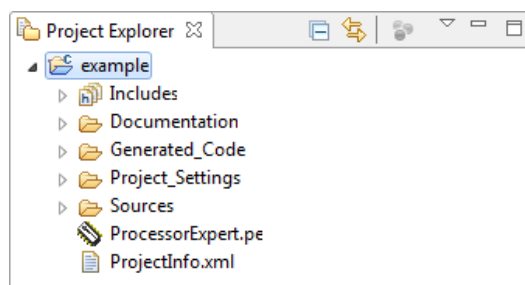
1. To create a new Kinetis Design Studio project, open the *File* menu, then select *New > Project > Kinetis Design Studio Project*. You will be presented with the *New Kinetis Design Studio Project* wizard.

Figure 3: New Kinetis Design Studio project wizard



2. Enter a *Project name*. You can specify the location of the project or check the box to use the default location. Then click *Next*.
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*. You can use the *filter text* box to search for the device name.
4. The *Rapid Application Development* dialog, allows you to configure use of *Use Processor Expert for configuration* and *KSDK*. Processor Expert is included with Kinetis Design Studio software, but KSDK must be installed separately. For details see section 2.3 and the documentation provided with these components.
5. Once all your selections have been made, click *Finish*.

Figure 4: Project Explorer



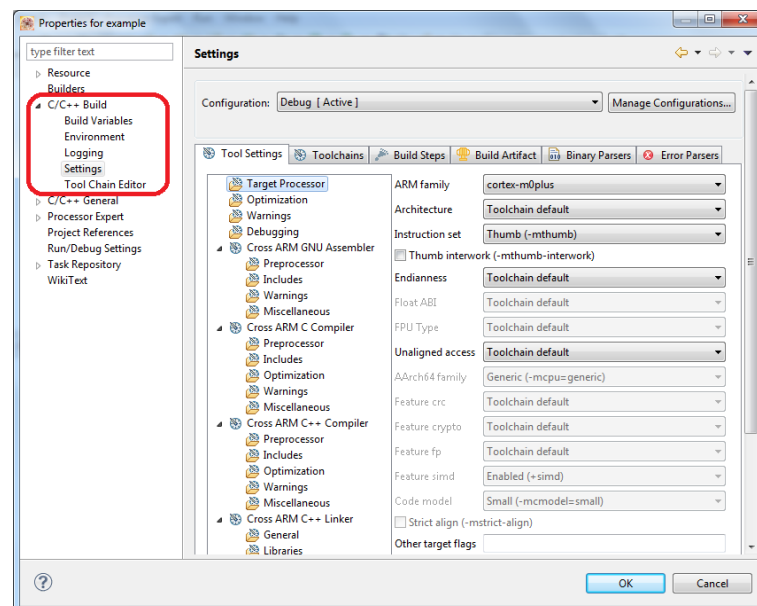
6. Your new Kinetis Design Studio project should now be shown under the *Project Explorer*.
7. To create a new source file under the project, right click on the project and select *New > Source File*. Alternatively you can drag and drop existing source files, header files, directories into the project.

3.3 Configuring the project

A new Kinetis Design Studio 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*.

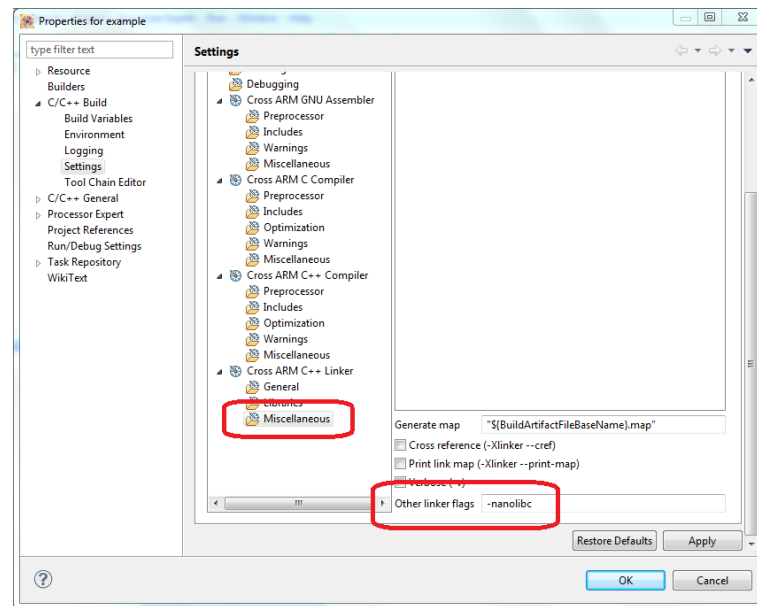
The configuration of the project can be adjusted by performing a right click on the project and selecting *Properties*. In the *Properties* wizard, the toolchain build settings can be changed by clicking on *C/C++ Build > Settings*.

Figure 5: Project Properties, C/C++ build settings



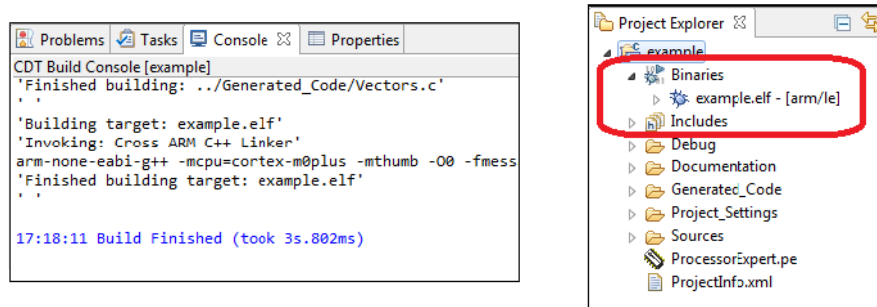
As an example, the Kinetis Design Studio software development tools supports two C runtime libraries, newlib and newlib-nano. By default a Kinetis Design Studio project will link an application with newlib-nano. If you wish to link with newlib instead remove the flag `-nanolibc` from the *Other linker flags* field under *Miscellaneous* of the *Cross ARM C++ Linker* folder.

Figure 6: Linking with newlib-nano



After changing the project's settings, 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 7: Build console and the generated binary



4 Debugging an embedded application

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

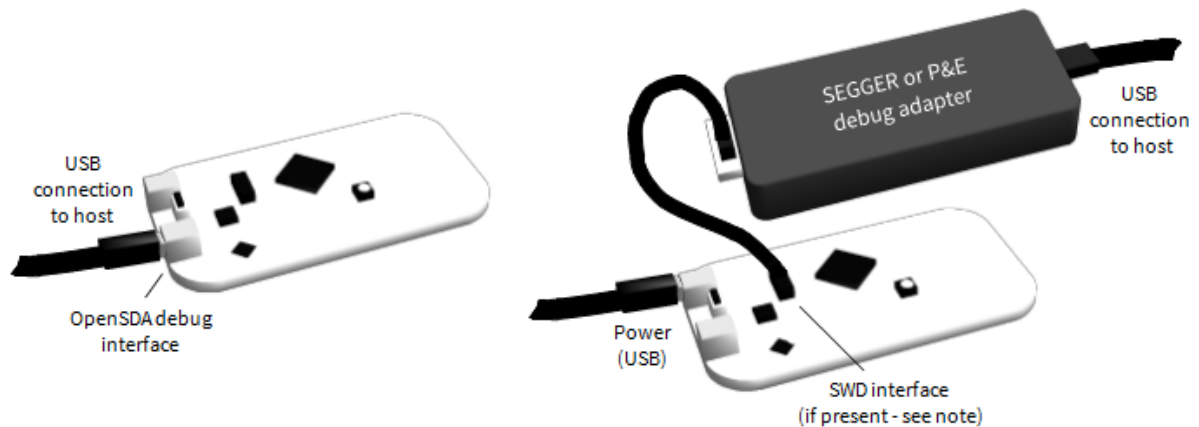
4.1 Development environment

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

1. On-board *OpenSDA* debug interface running the ARM® *mbed™* project *CMSIS-DAP* firmware

2. SEGGER J-Link and J-Trace debug adapters
3. P&E USB Multilink Universal and USB Multilink Universal FX debug adapters

Figure 8: 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 software development tools under `install-dir\pemicro\opensda`.

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

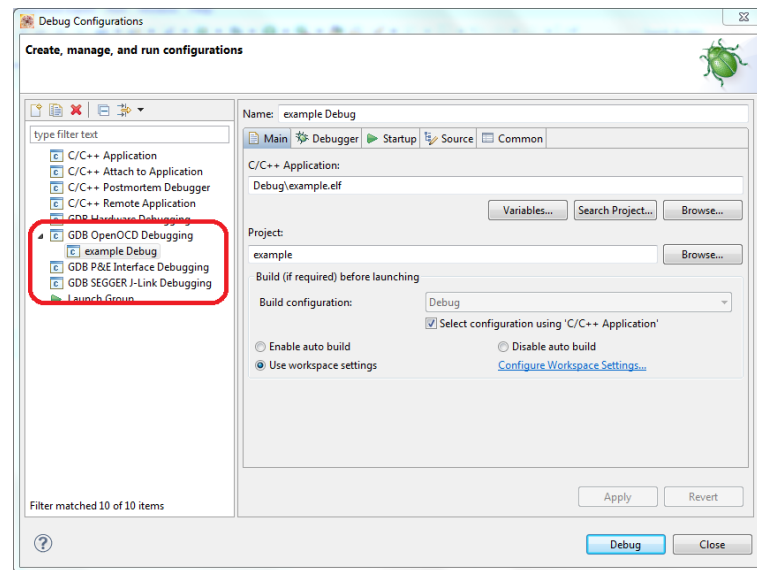
The procedure to debug an embedded application using the SEGGER OpenSDA firmware is the same as if you were using either a SEGGER J-Link or J-Trace debug adapter. Similarly the procedure to debug an embedded application using the P&E OpenSDA firmware is largely the same as if you were using either a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

4.2 Debugging with the Kinetis Design Studio 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 9: Create a new debug configuration

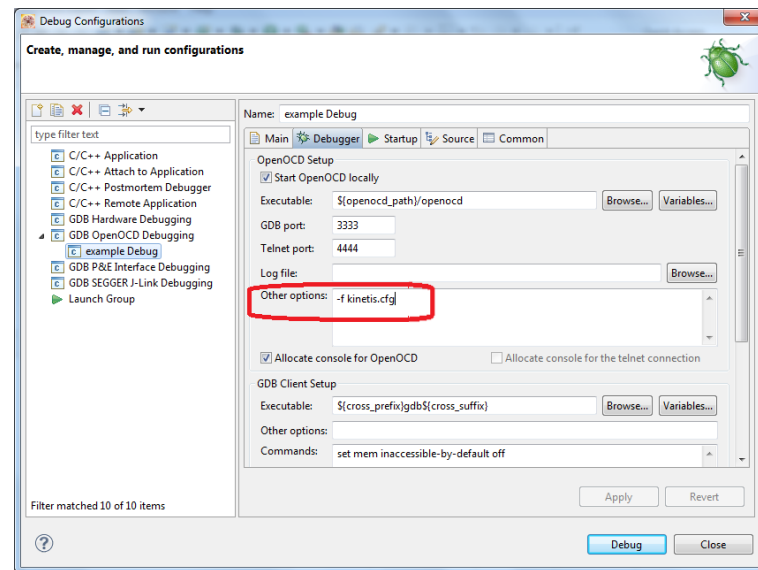


Note Appendix B provides details on the device drivers which are needed to work with the chosen debug interface.

- (a) If you are using the on-board *OpenSDA* debug interface running the *mbed projectCMSIS-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 10: Debugger tab for a GDB OpenOCD Debugging configuration

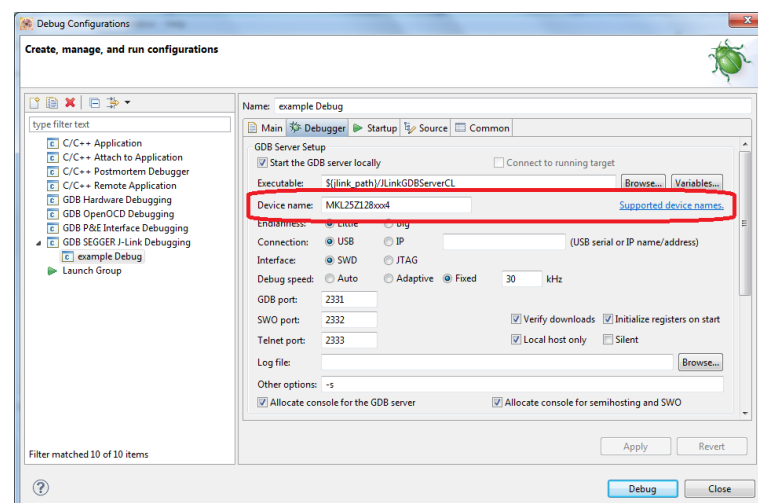


- (b) If you are using a SEGGER J-Link debug interface then enter the *Device name* for your Freescale Kinetis device. Use the link *Supported device names* to help you with your selection.

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*.

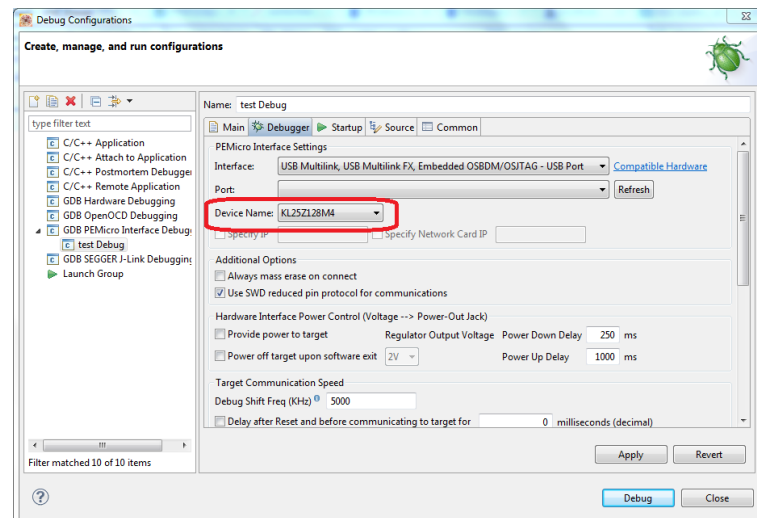
Figure 11: Debugger tab for a GDB SEGGER J-Link Debugging configuration



- (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 entries¹. This will connect to the remote target as a *localhost*.

¹ If you are using the P&E OpenSDA firmware then you need to select the *Interface* to be *OpenSDA Embedded Debug - USB Port*.

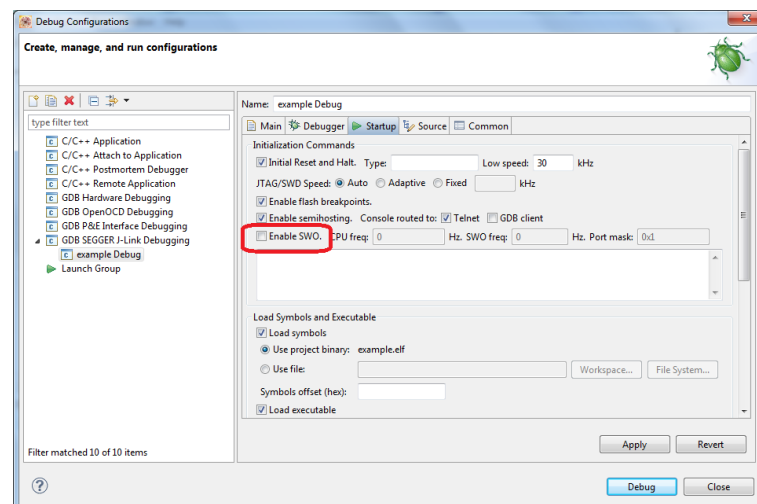
Figure 12: Debugger tab for a GDB P&E Interface Debugging configuration



3. Next select the *Startup* tab.

- (a) 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.
- (b) For users of the SEGGER *OpenSDA* application, then de-select the *Enable SWO* radial button, as this is not currently supported.

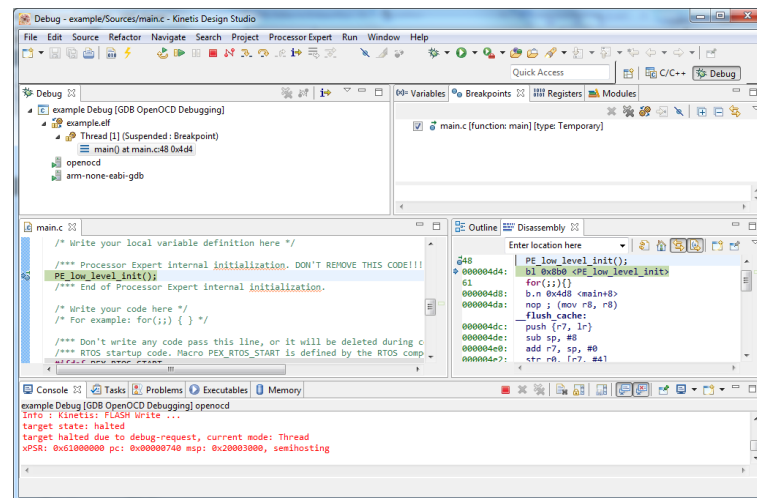
Figure 13: Startup tab for a GDB SEGGER J-Link Debugging configuration



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.

- (c) 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 *Debug Perspective* if are not already in the *Debug Perspective*. Select *Yes* to switch perspective. The *Debug Perspective* will open and the embedded application will break on the breakpoint set on **main**.











Figure 14: Kinetis Design Studio 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 <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

call stack 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 IDE's *Debug Perspective*.

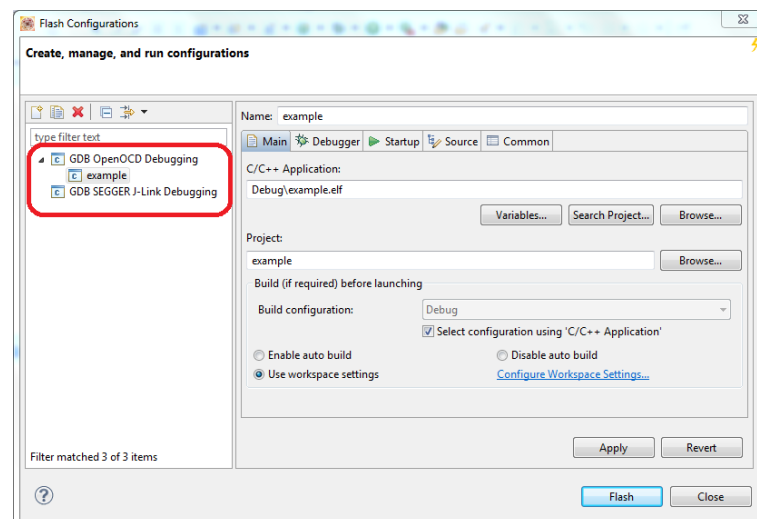
5 Flashing an embedded application

This section describes how to flash an application without an Eclipse project using the Kinetis Design Studio IDE.

5.1 Flashing with the Kinetis Design Studio IDE

1. Click on the  button from the main toolbar.

Figure 15: Flash configuration



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*, configure it as described in section 4
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 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 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 software development tools. However there are several differences between the Kinetis Design Studio software development tools and the GNU ARM Embedded toolchain to note:

- The Kinetis Design Studio 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 software development tools (for instance references to "rdimon" and "nosys"). Attempting to build applications with such references using the Kinetis Design Studio software development tools will result in build errors.
- The Kinetis Design Studio 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 programs 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 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 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.

B.1 Installing on Windows

The Kinetis Design Studio 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.

On Windows systems the mbed project CMSIS-DAP firmware requires a serial port driver to be installed. The Kinetis Design Studio 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 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
ARM mbed project Windows serial port driver	Follow the instructions at http://mbed.org/handbook/Windows-serial-configuration
SEGGER J-Link	Run the following installers: <ul style="list-style-type: none"> <code>install-dir\segger\USBDriver\InstDrivers.exe</code> <code>install-dir\segger\USBDriver\CDC\InstDriversCDC.exe</code>
P&E Multilink driver	Run the P&E driver installer: <code>install-dir\pemicro\PEDrivers_install.exe</code>

B.2 Installing on Linux

When the Kinetis Design Studio 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"> Copy the <code>udev</code> file into the configuration directory (for example under <code>/etc/udev/rules.d/</code>) Rename the file to <code>99-openocd.rules</code> (for example) Optionally the permissions allocated by the rules file can be adjusted. By default this requires users to be in the <code>plugdev</code> group. Run the command <code>udevadm control --reload-rules</code> to instruct <code>udev</code> to reload its rules

Driver	udev rules file	Notes
Segger J-Link	<code>install-dir/segger/99-jlink.rules</code>	<ul style="list-style-type: none">• Copy the <i>udev</i> file into the configuration directory (for example under <code>/etc/udev/rules.d/</code>)• Run the command udevadm control --reload-rules to instruct <i>udev</i> to reload its rules
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

C Notices

Copyright © 2013-2014 SOMNIUM® Technologies Limited

Kinetis Design Studio is produced for Freescale by SOMNIUM® Technologies <http://www.somniumtech.com>. All rights reserved.

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

SOMNIUM and the SOMNIUM logo are registered trademarks of SOMNIUM® Technologies Limited.

Freescale, the Freescale logo, CodeWarrior, and Kinetis are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners. ARM and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. mbed is a trademark of ARM Limited (or its subsidiaries) in the EU and/or elsewhere.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale 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 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 does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: <http://www.freescale.com/SalesTermsandConditions>.