# Getting Started with Kinetis SDK (KSDK) v.2.0

# **1** Overview

The Kinetis Software Development Kit (KSDK) provides comprehensive software support for Kinetis Microcontrollers. The KSDK includes a flexible set of peripheral drivers designed to speed up and simplify development of embedded applications. Along with the peripheral drivers, the KSDK provides an extensive and rich set of example applications covering everything from basic peripheral use case examples to full demo applications. The KSDK also contains RTOS kernels, a USB host and device stack, and various other middleware to support rapid development on Kinetis devices.

For supported toolchain versions, see the Kinetis SDK v.2.0.0 Release Notes (document KSDK200RN).

For the latest version of this and other Kinetis SDK documents, see the Kinetis SDK homepage www.nxp.com/ ksdk

### Contents

1	Overview	1
2	KSDK Board Support Folders	2
3	Run a demo application using IAR	4
4	Run a demo using Keil® MDK/ µVision	8
5	Run a demo using Kinetis Design Studio IDE	11
6	Run a demo using Atollic® TrueSTUDIO®	25
7	Run a demo using ARM GCC	33
8	KSDK Project Generator	42
9	Appendix A - How to determine COM port	46
10	Appendix B - Default debug interfaces	
11	Appendix C - Updating OpenSDA firmware	48
12	Revision History	50



#### **KSDK Board Support Folders**



Figure 1. KSDK layers

# 2 KSDK Board Support Folders

KSDK board support provides example applications for Kinetis development and evaluation boards. Board support packages are found inside of the top level boards folder, and each supported board has its own folder (a KSDK package can support multiple boards). Within each <br/>board\_name> folder there are various sub-folders to classify the type of examples they contain. These include (but are not limited to):

- demo\_apps: Full-featured applications intended to highlight key functionality and use cases of the target MCU. These applications typically use multiple MCU peripherals and may leverage stacks and middleware.
- driver\_examples: Simple applications intended to concisely illustrate how to use the KSDK's peripheral drivers for a single use case. These applications typically only use a single peripheral, but there are cases where multiple are used (for example, ADC conversion using DMA).
- rtos\_examples: Basic FreeRTOS examples showcasing the use of various RTOS objects (semaphores, queues, and so on) and interfacing with the KSDK's RTOS drivers
- usb\_examples: Applications that use the USB host/device/OTG stack.

# 2.1 Example Application Structure

This section describes how the various types of example applications interact with the other components in the KSDK. To get a comprehensive understanding of all KSDK components and folder structure, see the *Kinetis SDK v.2.0 API Reference Manual* document (KSDK20APIRM).

Each <board\_name> folder in the boards directory contains a comprehensive set of examples that are relevant to that specific piece of hardware. We'll discuss the hello\_world example (part of the demo\_apps folder), but the same general rules apply to any type of example in the <board\_name> folder.

In the hello\_world application folder you see this:



Figure 2. Application folder structure

All files in the application folder are specific to that example, so it's very easy to copy-paste an existing example to start developing a custom application based on a project provided in the KSDK.

# 2.2 Locating Example Application Source Files

When opening an example application in any of the supported IDEs, there are a variety of source files referenced. The KSDK *devices* folder is designed to be the "golden core" of the application and is, therefore, the central component to all example applications. Because it's a core component, all of the examples reference the same source files and, if one of these files is modified, it could potentially impact the behavior of other examples.

The main areas of the KSDK tree used in all example applications are:

- devices/<device\_name>: The device's CMSIS header file, KSDK feature file and a few other things.
- devices/<device\_name>/drivers: All of the peripheral drivers for your specific MCU.
- devices/<device\_name>/<tool\_name>: Toolchain-specific startup code. Vector table definitions are here.
- devices/<device\_name>/utilities: Items such as the debug console that are used by many of the example applications.

#### Run a demo application using IAR

For examples containing middleware/stacks and/or a RTOS, there will be references to the appropriate source code. Middleware source files are located in the *middleware* folder and RTOSes are in the *rtos* folder. Again, the core files of each of these are shared, so modifying them could have potential impacts on other projects that depend on them.

# 3 Run a demo application using IAR

This section describes the steps required to build, run, and debug example applications provided in the Kinetis SDK. The hello\_world demo application targeted for the FRDM-K64F Freedom hardware platform is used as an example, although these steps can be applied to any example application in the KSDK.

# 3.1 Build an example application

The following steps guide you through opening the hello\_world example application. These steps may change slightly for other example applications as some of these applications may have additional layers of folders in their path.

1. If not already done, open the desired demo application workspace. Most example application workspace files can be located using the following path:

<install\_dir>/boards/<board\_name>/<example\_type>/<application\_name>/iar

Using the FRDM-K64F Freedom hardware platform as an example, the hello\_world workspace is located in

<install\_dir>/boards/frdmk64f/demo\_apps/hello\_world/iar/hello\_world.eww

2. Select the desired build target from the drop-down. For this example, select the "hello\_world – Debug" target.

Workspace		×
Debug		-
Debug		
Release		
🗆 🗇 hello_world - Deb	×	
📙 🛏 🗀 board		
📗 🛏 🗀 doc		
📕 🛏 🗀 drivers		
🛛 🛏 🗀 source		
📕 🛏 🗀 startup		
📕 🛏 🗀 utilities		
📔 🖵 Output		

### Figure 3. Demo build target selection

3. To build the demo application, click the "Make" button, highlighted in red below.





4. The build completes without errors.

# 3.2 Run an example application

#### Run a demo application using IAR

To download and run the application, perform these steps:

- 1. Reference the table in Appendix B to determine the debug interface that comes loaded on your specific hardware platform.
  - For boards with CMSIS-DAP/mbed/DAPLink interfaces, visit developer.mbed.org/handbook/Windows-serial-configuration and follow the instructions to install the Windows<sup>®</sup> operating system serial driver.
  - For boards with P&E Micro interfaces, visit www.pemicro.com/support/downloads\_find.cfm and download the P&E Micro Hardware Interface Drivers package.
  - For the MRB-KW01 board, visit www.nxp.com/USB2SER to download the serial driver. This board does not support OpenSDA, so an external debug probe (such as a J-Link) is required. Steps below referencing OpenSDA do not apply as there is only a single USB connector for serial output.
- 2. Connect the development platform to your PC via USB cable between the OpenSDA USB connector (may be named OSJTAG for some boards) and the PC USB connector.
- 3. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug COM port (to determine the COM port number, see Appendix A). Configure the terminal with these settings:
  - a. 115200 or 9600 baud rate, depending on your board (reference BOARD\_DEBUG\_UART\_BAUDRATE variable in board.h file)
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit

stegory:		
Session	Basic options for yo	our PuTTY session
	Specify the destination you w	ant to connect to
E reminal	Serial li <u>n</u> e	Speed
Bell	COM16	115200
Features 	Connection type:	ogin © <u>S</u> SH 💿 Serial
	Load, save or delete a stored Sav <u>e</u> d Sessions	l session
	Debug	
	Default Settings	Load
- Connection	Debug	
Proxy		Save
<ul> <li>Froxy</li> <li>Telnet</li> <li>Rlogin</li> <li>SSH</li> <li>Serial</li> </ul>		Delete
	Close window on exit: Always Never	Only on clean exit

## Figure 5. Terminal (PuTTY) configuration

4. In IAR, click the "Download and Debug" button to download the application to the target.



### Figure 6. Download and Debug button

5. The application is then downloaded to the target and automatically runs to the main() function.



### Figure 7. Stop at main() when running debugging

6. Run the code by clicking the "Go" button to start the application.



### Figure 8. Go button

7. The hello\_world application is now running and a banner is displayed on the terminal. If this is not true, check your terminal settings and connections.



Figure 9. Text display of the hello\_world demo

# 4 Run a demo using Keil<sup>®</sup> MDK/µVision

This section describes the steps required to build, run, and debug example applications provided in the Kinetis SDK. The hello\_world demo application targeted for the FRDM-K64F Freedom hardware platform is used as an example, although these steps can be applied to any demo or example application in the KSDK.

# 4.1 Install CMSIS device pack

After the MDK tools are installed, Cortex Microcontroller Software Interface Standard (CMSIS) device packs must be installed to fully support the device from a debug perspective. These packs include things such as memory map information, register definitions and flash programming algorithms. Follow these steps to install the appropriate CMSIS pack.

1. Open the MDK IDE, which is called µVision. In the IDE, select the "Pack Installer" icon.



#### Figure 10. Launch the Pack installer

2. After the installation finishes, close the Pack Installer window and return to the  $\mu$ Vision IDE.

# 4.2 Build an example application

• If not already done, open the desired example application workspace in: <install\_dir>/boards/<board\_name>/ <*example\_type*>/<application\_name>/mdk

The workspace file is named <demo\_name>.uvmpw, so for this specific example, the actual path is:

- <install\_dir>/boards/frdmk64f/demo\_apps/hello\_world/mdk/hello\_world.uvmpw
- To build the demo project, select the "Rebuild" button, highlighted in red.



### Figure 11. Build the demo

• The build completes without errors.

# 4.3 Run an example application

To download and run the application, perform these steps:

#### Run a demo using Keil® MDK/µVision

- 1. Reference the table in Appendix B to determine the debug interface that comes loaded on your specific hardware platform.
  - For boards with the CMSIS-DAP/mbed/DAPLink interface, visit mbed Windows serial configuration.
  - For boards with a P&E Micro interface, visit www.pemicro.com/support/downloads\_find.cfm and download and install the P&E Micro Hardware Interface Drivers package.
  - For the MRB-KW01 board, visit www.nxp.com/USB2SER to download the serial driver. This board does not support the OpenSDA. Therefore, an external debug probe (such as a J-Link) is required. Steps below referencing the OpenSDA do not apply because there is only a single USB connector for serial output.
  - For boards with the OSJTAG interface, install the driver from www.keil.com/download/docs/408.
- 2. Connect the development platform to your PC via USB cable between the OpenSDA USB connector (may be named OSJTAG on some boards) and the PC USB connector.
- 3. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug serial port number (to determine the COM port number, see Appendix A). Configure the terminal with these settings:
  - a. 115200 or 9600 baud rate, depending on your board (reference BOARD\_DEBUG\_UART\_BAUDRATE variable in board.h file)
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit

		_
Session	Basic options for yo	our PuTTY session
Logging	Specify the destination you v	vant to connect to
	Serial line	Speed
Bell	COM16	115200
- Features	Connection type:	login 🔘 <u>S</u> SH 💿 Serial
Appearance Behaviour Translation	Load, save or delete a stored Sav <u>e</u> d Sessions	d session
Selection	Debug	
Colours	Default Settings	Load
- Data	Debug	Save
Proxy Telnet Rlogin ⊕ SSH Serial		Delete
	Close window on exit: Always Never	Only on clean exit

### Figure 12. Terminal (PuTTY) configurations

4. In μVision, after the application is properly built, click the "Download" button to download the application to the target.

#### Run a demo using Keil® MDK/µVision

🧇 🕮 🕮 픯   🙀	hello_world Debug	- 🔊
Project	д 💌	
🖃 🚰 WorkSpace		
🗄 🔧 Project: hello_wor	ld	

### Figure 13. Download button

5. After clicking the "Download" button, the application downloads to the target and should be running. To debug the application, click the "Start/Stop Debug Session" button, highlighted in red.



Figure 14. Stop at main() when run debugging

6. Run the code by clicking the "Run" button to start the application.



Figure 15. Go button

The hello\_world application is now running and a banner is displayed on the terminal. If this is not true, check your terminal settings and connections.



Figure 16. Text display of the hello\_world demo

# 5 Run a demo using Kinetis Design Studio IDE

NOTE

Ensure that you selected the Kinetis Design Studio IDE toolchain when you generated the KSDK Package.

This section describes the steps required to configure Kinetis Design Studio (KDS) IDE to build, run, and debug example applications. The hello\_world demo application targeted for the FRDM-K64F Freedom hardware platform is used as an example, though these steps can be applied to any example application in the KSDK.

# 5.1 Select the workspace location

The first time that KDS IDE launches, it prompts the user to select a workspace location. KDS IDE is built on top of Eclipse, which uses workspace to store information about its current configuration, and in some use cases, source files for the projects in the workspace. The location of the workspace can be anywhere, but it is recommended that the workspace be outside of the KSDK tree.

# 5.2 Updating the KDS IDE components

The user must update the KDS IDE installation before using the Kinetis SDK with it. How the update is performed depends on the KDS IDE version.

# 5.2.1 Update KDS IDE 3.0 and KDS IDE 3.1

NOTE

If you have previously installed New Project Wizard for KSDK 2.0 to KDS IDE, update it using the instructions described in the subsequent section.

1. Select the menu Help -> Install New Software.

ert Window	Help			
☆・○・(		Welcome		(÷ -
	0	Help Contents		
	22	Search		
		Dynamic Help		
		Key Assist	Ctrl+Shift+L	
		Tips and Tricks		
	-	Report Bug or Enhancement		
		Cheat Sheets		
	~	Check for Updates		
,		Install New Software		
	10	Installation Details		
	mp	Eclipse Marketplace		
		Freescale Licenses		
	1990 - Contra - Contr	About Kinetis Design Studio		

### Figure 17. Install new software

2. Select "Freescale KDS Update Site" as the site to work with.

🛞 Install	
Available Software Check the items that you wish to install.	
Work with: <sup> </sup>	e/KDS
type filter text	
Name	Version
>       000       KSDK 1.2.0 Eclipse Update         >       000       MQX RTOS Plug-ins         >       000       New Kinetis SDK 2.x Project Wizard         >       000       PEx for Kinetis 3.0.0 KM1x/KM3x @50MHz/75MHz Eclipse Update         >       000       Processor Expert for Kinetis 3.0.2         >       000       Project of Projects	
Show only the latest versions of available software	re already installed
<ul> <li>✓ <u>G</u>roup items by category What is <u>already ins</u></li> <li>□ Show only software applicable to target environment</li> <li>✓ <u>C</u>ontact all update sites during install to find required software</li> </ul>	<u>stalled</u> ?
(?)       < <u>Back</u> <u>Next</u> >	<u>Einish</u> Cancel

### Figure 18. Select KDS IDE update site

3. Wait until the site content is displayed and select the "New Kinetis SDK 2.x Project Wizard".

🧩 Install				
Available Software Check the items that you wish to install.				
Work with: Freescale KDS Update Site - http://freescale.com/lgfiles/updates/Eclipse/KDS	▲dd			
Find more software by working with the <u>"Available Software</u>	are Sites" preferences.			
type filter text				
Name Version	~			
<ul> <li>Image: Solution of the second s</li></ul>	E			
Image: Second Seco	-			
Select All Deselect All 1 item selected				
Details	1			
Show only the latest versions of available software				
Group items by category What is <u>already installed</u> ?				
Show only software applicable to target environment				
☑ Contact all update sites during install to find required software				
(?) < <u>B</u> ack <u>N</u> ext > <u>F</u> inish	Cancel			

Figure 19. Select New Project Wizard

- 4. Confirm and complete installation.
- 5. Restart the IDE.

# 5.2.2 Update KDS IDE 3.2

1. Select the menu Help -> Check for Updates.

Window	Help	)		
\$= • O		Welcome		<b>~</b>
	⑦	Help Contents Search		
		Dynamic Help		
	ß	Key Assist Tips and Tricks Report Bug or Enhancement	Ctrl+Shift+L	
	- Apr	Cheat Sheets		
	000 C	Check for Updates		
		Install New Software		
	89	Installation Details		
	mp	Eclipse Marketplace		
		Freescale Licenses		
		About Kinetis Design Studio		

### Figure 20. Check for updates

2. Wait until the site content is displayed and select "New Kinetis SDK 2.x Project Wizard". Ensure that no other items are selected.

🛞 Available Updates		
Available Updates Check the updates that you wish to install.		
Name	Version	Id
GNU ARM C/C++ Cross Compiler	2.2.1.201604190915	ilg.gnuarmeclipse.managedbuild.cross.fe
📰 🖗 GNU ARM C/C++ J-Link Debugging	3.2.1.201604190915	ilg.gnuarmeclipse.debug.gdbjtag.jlink.fe
GNU ARM C/C++ OpenOCD Debugging	3.2.1.201604190915	ilg.gnuarmeclipse.debug.gdbjtag.openo
CONTRACT SDK 2 x Project Without	2.4.5.201604291203	com.pemicro.debug.gdbjtag.pne.feature
W New Knetis SDK 2.x Project Wizard	1.1.0.201004070033	com.nxp.reature.npw+sok.reature.group
Select All Deselect All		
Details		
		-
?	< <u>B</u> ack	Next > Einish Cancel

Figure 21. Available updates for KDS IDE

- 3. Confirm and complete update.
- 4. Restart the IDE.

# 5.3 Build an example application

NOTE

The steps required for the Linux<sup>®</sup> OS and Mac<sup>®</sup> OS are identical to those for the Windows<sup>®</sup> operating system. The only difference is that the IDE looks slightly different.

1. Select "File -> Import" from the KDS IDE menu. In the window that appears, expand the "Project of Projects" folder and select "Existing Projects Sets". Then, click the "Next" button.

Minport Import	
Select	Ľ
Select an import source:	
type filter text	
<ul> <li>General</li> <li>C/C++</li> <li>Component Development Environment</li> <li>CVS</li> <li>Git</li> <li>Forcessor Expert</li> <li>Project of Projects</li> <li>Existing Projects Sets</li> <li>Remote Systems</li> <li>RPM</li> <li>Run/Debug</li> <li>Tasks</li> <li>Team</li> <li>Tracing</li> </ul>	
? < Back Next > Finish	Cancel

# **Figure 22. Selection of the correct import type in KDS IDE** 2. Click the "Browse" button next to the "Import from file:" option.

Run a demo using Kinetis Design Studio IDE Import Working Sets and Projects Import Working Sets and Projects Ø Please select a file to import Browse ... Import from file: Select All Deselect All Refresh ? Finish Cancel < Back Next >

### Figure 23. Projects directory selection window

3. Point to the example application project, which can be found using this path:

<install\_dir>/boards/<board\_name>/<example\_type>/<application\_name>/kds

For this example, the specific location is:

<install\_dir>/boards/frdmk64f/demo\_apps/hello\_world/kds

4. After pointing to the correct directory, your "Import Working Sets and Projects" window should look like the figure below. Click the "Finish" button.

Getting Started with Kinetis SDK (KSDK) v.2.0, Rev. 2, 07/2016

23

🈹 Import Working	Sets and Projects		
Import Working	Sets and Projects		
Import Working	Sets and Projects from the loo	al file system.	
Import from file:	C:\nxp\SDK_2.0_FRDM-K64F	\boards\frdmk64f\demo_a	pps\he Browse
b I hello_work	d		
Select All	select All		Refearb
Select All	SELECT AIL		Kerresh
-			
?	< Back Ne	t > Finish	Cancel

### Figure 24. Select K64F12 platform library project

- 5. There are two project configurations (build targets) supported for each KSDK project:
  - Debug Compiler optimization is set to low, and debug information is generated for the executable. This target should be selected for development and debug.
  - Release Compiler optimization is set to high, and debug information is not generated. This target should be selected for final application deployment.
- 6. Choose the appropriate build target, "Debug" or "Release", by clicking the downward facing arrow next to the hammer icon, as shown below. For this example, select the "Debug" target.



### Figure 25. Selection of the build target in KDS IDE

The library starts building after the build target is selected. To rebuild the library in the future, click the hammer icon (assuming the same build target is chosen).

# 5.4 Run an example application

#### NOTE

The steps required for the Linux OS and Mac OS are identical to those for the Windows operating system. The only difference is that the IDE looks slightly different. Any platform-specific steps are listed accordingly.

To download and run the application, perform these steps:

- 1. Reference the table in Appendix B to determine the debug interface that comes loaded on your specific hardware platform.
  - For Windows operating system and Linux OS users, download the driver that corresponds to your debug interface:

- For boards with the CMSIS-DAP/mbed/DAPLink interface, visit developer.mbed.org/handbook/Windowsserial-configuration and follow the instructions to install the Windows operating system serial driver. If running on Linux OS, this step is not required.

- For boards with a P&E Micro interface, visit www.pemicro.com/support/downloads\_find.cfm and download and install the P&E Micro Hardware Interface Drivers package.

If J-Link is used, either a standalone debug pod or OpenSDA, see www.segger.com/jlink-software.html.

For the MRB-KW01 board, see www.nxp.com/USB2SER to download the serial driver. This board does not support OpenSDA, so an external debug probe (such as a J-Link) is required. Steps below referencing OpenSDA do not apply as there is only a single USB connector for serial output.

• For Mac OS users, KDS only supports the J-Link OpenSDA interface.

Follow the instructions in Appendix C to update your board's OpenSDA interface to the J-Link OpenSDA application. Then, see www.segger.com/jlink-software.html to download the necessary software and drivers.

- For TWR-K80F150M and FRDM-K82F platforms, the J-Link OpenSDA application is required to be loaded because KDS IDE does not support CMSIS-DAP/mbed for those devices. See Appendix C for more information.
- 2. Connect the development platform to your PC via USB cable between the OpenSDA USB connector (may be named OSJTAG for some boards) and the PC USB connector.
- 3. In the Windows operating system environment, open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug serial port number (to determine the COM port number, see Appendix A). For Linux OS, open your terminal application and connect to the appropriate device.

Configure the terminal with these settings:

- a. 115200 or 9600 baud rate, depending on your board (reference BOARD\_DEBUG\_UART\_BAUDRATE variable in board.h file)
- b. No parity
- c. 8 data bits
- d. 1 stop bit

Carcion	Pasia antiana farunar	
	Basic options for your	FUTTT Session
Teminal	Specify the destination you want	t to connect to
Keyboard	Serial li <u>n</u> e	Speed
Bell	COM16	115200
- Features - Window - Appearance - Behaviour - Translation	Connection type:	n 🔘 <u>S</u> SH 💽 Serjal
	Load, save or delete a stored se Sav <u>e</u> d Sessions	ession
Selection	Debug	
- Colours - Connection - Data	Default Settings	Load
		Sa <u>v</u> e
Telnet Rlogin		Delete
⊕- SSH Serial	Close window on exit: Always Never	Only on clean exit

### Figure 26. Terminal (PuTTY) configurations

4. For Linux OS users only, run the following commands in your terminal. These install libudev onto your system, which is required by KDS IDE to launch the debugger.

user@ubuntu:~\$ sudo apt-get install libudev-dev libudev1

user@ubuntu:~\$ sudo ln -s /usr/lib/x86\_64-linux-gnu/libudev.so /usr/lib/x86\_64-linux-gnu/libudev.so.0

- 5. In KDS IDE, ensure that the debugger configuration is correct for the target you're attempting to connect to. Consult Appendix B for more information about the default debugger application on the various hardware platforms supported by the KSDK.
  - a. To check the available debugger configurations, click the small downward arrow next to the green "Debug" button and select "Debug Configurations".



### Figure 27. Debug Configurations dialog button

b. In the Debug Configurations dialog box, select the debug configuration that corresponds to the hardware platform you're using. In this example, since the FRDM-K64F is used, select is the CMSIS-DAP/DAPLink option under OpenOCD. To determine the interface to use for other hardware platforms, refer to Appendix B.

After selecting the debugger interface, click the "Debug" button to launch the debugger.

reate, manage, and run configurations			Ś
) 🗈 🗙   🖻 🐎 •	Name: hello_world_frdmk6	4f debug cmsisdap	
type filter text	🗋 Main 🏇 Debugger 👂	Startup 🤤 Source 🔲 Common	
C/C++ Application	C/C++ Application:		
<ul> <li>C/C++ Attach to Application</li> <li>C/C++ Postmortem Debugger</li> </ul>	Debug/hello_world_frdmk	64f.elf	
C C/C++ Remote Application GDB Hardware Debugging GDB OpenOCD Debugging	Project:	Variables Search Project.	Browse
c hello_world_frdmk64f debug cmsisdap	hello_world_frdmk64f		Browse
hello_world_frdmk64f release cmsisdap     GDB PEMicro Interface Debugging	Build (if required) before I	aunching	
C GDB SEGGER J-Link Debugging	Build configuration:	Debug	*
c hello_world_frdmk64f debug jlink		Select configuration using 'C/C++ Appl	ication'
Launch Group Iter matched 15 of 15 items		Apply	Revert

Figure 28. Selection of the debug configuration and debugger launch

6. The application is downloaded to the target and automatically run to main():

Run a demo using Kinetis Design Studio IDE

💑 Debug - hello_world_trdmkb4t/source/hello_world.c - Kinetis Design Studio - C:\KDS Workspaces\build_test
File Edit Source Refactor Navigate Search Project Run ProcessorExpert Window Help
📸 • 🗟 🐁 🖕 🖓 • 🔍 • 🎭 • 🔪 🕪 🗉 🖷 🕸 32 🖘 22 🖬 23 🍎 🖉 🤌 🌮 🖉 🖓
🎄 Debug 💥 🙀 📝 🖻 🗆
<ul> <li>c hello_world_frdmk64f debug cmsisdap [GDB OpenOCD Debugging]</li> <li>c hello_world_frdmk64f.elf</li> <li>c Thread #1 (Suspended : Breakpoint)</li> <li>c main() at hello_world.c:57 0x3486</li> <li>c openocd</li> <li>arm-none-eabi-gdb</li> </ul>
<pre>     hello_world.c      hello_world.c      hello_world.c      '*!     * @brief Main function     */     e int main(void)     {         char ch;         /* Init board hardware. */     } } </pre>
BOARD_INITPINS(); BOARD_RestClockPUN();
BOARD_DOTCLOCKNON(); BOARD_InitDebugConsole():

Figure 29. Stop at main() when running debugging

7. Start the application by clicking the "Resume" button:

IL N	un	VV1	naov
	88		84
Res	um	e (F8	3)
	Res	Resum	Resume (F8

### Figure 30. Resume button

The hello\_world application is now running and a banner is displayed on the terminal. If this is not true, check your terminal settings and connections.



Figure 31. Text display of the hello\_world demo

# 5.5 Create a new project

1. Select the menu File -> New -> Kinetis SDK 2.x Project.



Figure 32. Select the menu File -> New -> Kinetis SDK 2.x Project

2. Enter the project name and use the default project location.

🥵 Kinetis SDK 2.x Project Wizard	
Select Kinetis SDK 2.x	
Select project name and SDK 2.x Path (root folder)	
Project name:	
MyPrj	
✓ Use <u>d</u> efault location	
Location: C:\Users\b23204\EclipseWorkspaces\runtime-KDS32\MyPrj	B <u>r</u> owse
Choose file system: default 💌	
Select Kinetis SDK folder:	
C:\nxp\SDK_2.0_FRDM-K64F	Browse
If you do not have Kinetisk SDK installed yet, visit Kinetis Expert Web Site: h	ttp://kex.nxp.com
? < <u>Back</u> Next > <u>Finish</u>	Cancel

### Figure 33. Enter the project name

- 3. The wizard supports three kinds of projects:
  - Empty project for a board see Board-><board>->New <board> project

• Example project - see Board -> <board> Examples -> <category> -> <example>

#### NOTE

This item allows a clone example project from boards/<board>/ folder in the KDS work space. It is available only if the KSDK package contains information about the project cloning.

• Empty project for a processor -> see Processor -> <processor> -> New <processor> project

#### NOTE

An empty project means that there is no significant code in the main function.

🛞 Kinetis SDK 2.x Project Wizard	- • ×
Select Kinetis Processor, Board, or Example	
Select the Kinetis Processor, Board or Example from which you would like to create this new project.	
type filter text	]
<ul> <li>Boards</li> <li>FRDM-K64F board New FRDM-K64F project</li> <li>Examples</li> <li>demo_apps</li> <li>driver_examples</li> <li>mmcau</li> <li>mmcau_api</li> <li>rtos_examples</li> <li>usb_examples</li> <li>Processors</li> <li>New MK64FN1M0xxx12 project</li> </ul>	
Clone selected example project into workspace.	
? < <u>Back</u> <u>Next</u> > <u>Finish</u>	Cancel

### Figure 34. Select board/processor

- 4. For empty projects, you can select an RTOS and drivers:
  - All drivers to have all KSDK drivers and utilities copied into the project
  - Minimum set to have only a basic set of drivers
  - Empty to create a bare metal project
- 5. Finish.

You can now build and debug the project.

# 6 Run a demo using Atollic<sup>®</sup> TrueSTUDIO<sup>®</sup>

#### Run a demo using Atollic® TrueSTUDIO®

This section describes the steps to configure Atollic TrueSTUDIO to build, run, and debug example applications provided in the KSDK. The hello\_world example application targeted for the FRDM-K64F Freedom hardware platform used as an example, though these steps can be applied to any demo or example application in the KSDK.

# 6.1 Select the workspace location

The first time that TrueSTUDIO launches, it prompts the user to select a workspace location. TrueSTUDIO uses Eclipse, which uses workspace to store information about its current configuration, and in some use cases, source files for the projects in the workspace. The location of the workspace can be anywhere, but it is recommended that the workspace be outside of the KSDK tree.

# 6.2 Build an example application

1. Select "File -> Import" from the TrueSTUDIO menu. Expand the "General" folder and select "Existing Projects into Workspace". Then, click the "Next" button.

Select Create new projects from an archive file or directory.	2
Select an import source:	
type filter text	1
<ul> <li>Ceneral</li> <li>Archive File</li> <li>Existing Projects into Workspace</li> <li>File System</li> <li>Preferences</li> <li>C/C++</li> <li>C/C++</li> <li>C/S</li> <li>Example projects</li> <li>Git</li> <li>Finstall</li> <li>Run/Debug</li> <li>SVN</li> <li>Tasks</li> </ul>	

### Figure 35. Selection of the correct import type in TrueSTUDIO

2. Click the "Browse" button next to the "Select root directory:" option.

	-
Select a directory to search for existing Eclipse pro	viects.
Select root directory:	- Browse
Select archive file:	- Browse
Projects:	
	Select All
	Deselect All
	Refresh
Options Search for nested projects Copy projects into workspace	
Options Search for nested projects Copy projects into workspace Working sets	
Options Search for nested projects Copy projects into workspace Working sets Add project to working sets	
Options Search for nested projects Copy projects into workspace Working sets Add project to working sets Working sets	* Select
Options Search for nested projects Copy projects into workspace Working sets Add project to working sets Working sets	y Select
Options Search for nested projects Copy projects into workspace Working sets Add project to working sets Working sets	* Select
Options Search for nested projects Copy projects into workspace Working sets Add project to working sets Working sets	v Select

### Figure 36. Projects directory selection window

3. Point to the example application project for the appropriate device, which can be found using this path:

<install\_dir>/boards/<board\_name>/<example\_type>/<application\_name>/atl

For this example, the specific location is:

<install\_dir>/boards/frdmk64f/demo\_apps/hello\_world/atl

4. After pointing to the correct directory, your "Import Projects" window should look like this figure. Click the "Finish" button.

#### Run a demo using Atollic® TrueSTUDIO®

Import		
mport Projects Select a directory to sear	ch for existing Eclipse projects.	
<ul> <li>Select root directory:</li> <li>Select archive file:</li> <li>Projects:</li> </ul>	C:\rxp\SDK_2.0_FRDM-K64F\boards\frdmk6 +	Browse Browse
hello_world_frdm	k64f (C:\nxp\SDK_2.0_FRDM-K64F\boards\frdmk	Select All
		Deselect All Refresh
Options     Search for nested pro     Copy projects into w	ini +	
Working sets		
Add project to work	ing sets	
Working sets:		Select

Figure 37. Select the K64F12 platform library project

#### NOTE

Do not select the "Copy projects..." option.

- 5. There are two project configurations (build targets) supported for each KSDK project:
  - Debug Compiler optimization is set to low, and debug information is generated for the executable. This target should be selected for development and debug.
  - Release Compiler optimization is set to high, and debug information is not generated. This target should be selected for final application deployment.
- 6. Choose the appropriate build target, "Debug" or "Release", by clicking the "Manage build configurations" icon, as shown below. For this example, select the "Debug" target and click "Set Active". Since the default configuration is to use the Debug target, there should not be a change required.

		Run a demo usin	ig Atollic® TrueSTUDIO®
C/C++ - Atollic TrueSTUDIO® for ARM®			
File Edit Source Refactor View Navigat	e Search Project	Run Window He	lp
	. & 6 8 * ·	ii (29 - A? ▼ ) *= ·	← ▼ ⇒ ▼   ≝   0
Project Explorer	a hello_world_fro	lmk64f: Manage Config	gurations 🛛 🔀
⊢ 🤤 🍯	Configuration	Description	Status
	debug release		Active
	Set Active	New Delet	e Rename
		ОК	Cancel

Figure 38. Selection of build target in TrueSTUDIO

7. Click the "Build" icon to build the application.

# 6.3 Run an example application

The Atollic tools require either a J-Link or P&E Micro debug interface. As a result, some hardware platforms require an update to the OpenSDA debug firmware found on the board. To determine the default debug interface of your board, see Appendix B. If the default interface is not J-Link or P&E Micro, see Appendix C for instructions on how to install one of these debug interfaces.

This section describes steps to run a demo application using a J-Link debugger, although the P&E Micro interface is also supported.

In order to perform this exercise with the J-Link interface, two things must be done:

- Install the J-Link software (drivers and utilities), which can be downloaded from segger.com/downloads.html.
- Make sure that either:
  - The OpenSDA interface on your board is programmed with the J-Link OpenSDA firmware. To determine if your board supports OpenSDA, see Appendix B. For instructions on reprogramming the OpenSDA interface, see Appendix C. If your board does not support OpenSDA, then a standalone J-Link pod is required.
  - A standalone J-Link pod is connected to the debug interface of your board. Note that some hardware platforms require hardware modification in order to function correctly with an external debug interface.

The P&E Micro interface can also be used. To use this interface:

- Install the P&E Micro Hardware Interface Drivers, which can be downloaded from www.pemicro.com/support/ downloads\_find.cfm.
- If your board does not come loaded with a P&E Micro interface, if supported, reprogram the OpenSDA interface with P&E Micro OpenSDA firmware. To determine if your board supports OpenSDA, see Appendix B. For instructions on reprogramming the OpenSDA interface, see Appendix C.

#### Run a demo using Atollic® TrueSTUDIO®

For the MRB-KW01 board, visit www.nxp.com/USB2SER to download the serial driver. This board does not support OpenSDA, so an external J-Link is required.

After the debug interface is configured and ready to use to download and run the application:

- 1. Connect the development platform to your PC via USB cable between the OpenSDA USB connector (may be named OSJTAG for some boards) and the PC USB connector.
- 2. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug serial port number (to determine the COM port number, see Appendix A). Configure the terminal with these settings:
  - a. 115200 or 9600 baud rate, depending on your board (reference BOARD\_DEBUG\_UART\_BAUDRATE variable in board.h file)
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit

	Pasio ontione for w	PUTTY engelon
	Basic options for yo	
- Terminal	Specify the destination you w	vant to connect to
- Keyboard	Serial li <u>n</u> e	Speed
Bell	COM16	115200
Features	Connection type:	
Window	© Ra <u>w</u> © <u>T</u> elnet © Ri	ogin 🔘 <u>S</u> SH 🛛 () Serial
Appearance	Load, save or delete a stored	d session
Translation	Saved Sessions	
Selection	Debug	
Colours	Default Cettings	
- Connection	Debug	Load
- Data		Save
Proxy		
Telnet		Delete
- Riogin		
Serial	F	
	Close window on exit:	Only on clean avit
	O Aways O Nevel	Only on Gean exit

### Figure 39. Terminal (PuTTY) configurations

3. In Atollic IDE, ensure that the debugger configuration is correct for the target you are attempting to connect to. a. To check the debugger configurations, click the "Configure Debug" icon.



### Figure 40. Debug configurations dialog button

- b. In the Debug Configurations window, select debug configuration that corresponds to the hardware platform you're using. The Atollic tools require either a J-Link or P&E Micro debug interface, so some hardware platforms require an update to the OpenSDA debug firmware. To determine the default debug interface of your board, see Appendix B. If the default interface is not J-Link or P&E Micro, see Appendix C for instructions on how to install one of these debug interfaces. **Important:** This example assumes the J-Link interface has been installed on the FRDM-K64F Freedom hardware platform.
- c. Select the J-Link "Debug" interface and click the "Debug" button.

cate, manage, and ran comigations	
3 🕼 🗙   🖻 🐎 •	Name: hello_world_frdmk64f debug jlink
type filter text	📄 Main 🕸 Debugger 🕨 Startup Scripts 🦆 Source 🔲 Common
<ul> <li>C/C++ Application</li> <li>C/C++ Attach to Application</li> <li>C/C++ Postmortem Debugger</li> <li>C/C++ Remote Application</li> <li>Embedded C/C++ Application</li> <li>hello_world_frdmk64f debug jlink</li> </ul>	C/C++ Application:
	debug/hello_world_frdmk64f.elf Variables Search Project Browse Project:
<ul> <li>hello_world_frdmk64f release jlink</li> </ul>	hello_world_frdmk64f Browse
C hello_world_frdmk64f release pne	Build (if required) before launching
Launch Group	Build configuration: Debug *
ilter matched 10 of 10 items	Apply Revert

Figure 41. Selection of debug configuration in Debug Configuration dialog box

4. The application is downloaded to the target and automatically runs to main():

#### Run a demo using Atollic® TrueSTUDIO®

Debug - hello_world_example_frdmk64f/source/he	llo_world.c - Atollic TrueSTUDIO® for ARM®
File Edit View Run Window Help	
日 16 14 14 14 14 14 14 14 14 14 14 14 14 14	
🕸 Debug ⊠	🙀 🏹 🗖 🗖 🚺 🗱 Variables 🖾 💁 Br
<ul> <li>c hello_world_example_frdmk64f debug jlink [4</li> <li>hello_world_example_frdmk64f.elf</li> <li>Thread [1] &lt; main&gt; (Suspended : Brea</li> <li>main() at hello_world.c:57 0x258e</li> <li>gdb</li> <li>SEGGER J-LINK</li> </ul>	Embedded C/C++ Application] (x)= ch
le hello_world.c 🕴	
54 char ch; 55	
56 /* Init board hardware. */	
<pre>57 BOARD_InitPins();</pre>	
58 BOARD_BootClockRUN();	
59 BOARD_InitDebugConsole();	

Figure 42. Stop at main() when running debugging

5. Run the code by clicking the "Resume" button to start the application.



### Figure 43. Resume button

The hello\_world application is now running and a banner is displayed on the terminal. If this is not true, check your terminal settings and connections.



Figure 44. Text display of the hello\_world demo

# 7 Run a demo using ARM GCC

This section describes the steps to configure the command line ARM GCC tools to build, run, and debug demo applications and necessary driver libraries provided in the KSDK. The hello\_world demo application targeted for the FRDM-K64F Freedom hardware platform is used as an example, though these steps can be applied to any board, demo or example application in the KSDK.

# 7.1 Set up toolchain

This section contains the steps to install the necessary components required to build and run a KSDK demo application with the ARM GCC toolchain, as supported by the KSDK. There are many ways to use ARM GCC tools, but this example focuses on a Windows operating system environment. Though not discussed here, ARM GCC tools can also be used with both Linux OS and Mac OSX.

# 7.1.1 Install GCC ARM Embedded tool chain

Download and run the installer from launchpad.net/gcc-arm-embedded. This is the actual toolset (i.e., compiler, linker, etc.). The GCC toolchain should correspond to the latest supported version, as described in the *Kinetis SDK v.2.0.0 Release Notes*. (document KSDK200RN).

# 7.1.2 Install MinGW

The Minimalist GNU for Windows (MinGW) development tools provide a set of tools that are not dependent on third party C-Runtime DLLs (such as Cygwin). The build environment used by the KSDK does not utilize the MinGW build tools, but does leverage the base install of both MinGW and MSYS. MSYS provides a basic shell with a Unix-like interface and tools.

- 1. Download the latest MinGW mingw-get-setup installer from sourceforge.net/projects/mingw/files/Installer/.
- 2. Run the installer. The recommended installation path is C:\MinGW, however, you may install to any location.

#### NOTE

The installation path cannot contain any spaces.

3. Ensure that the "mingw32-base" and "msys-base" are selected under Basic Setup.

🏇 MinGW Installation Manager					
Installation Package Settings					
Basic Setup	Package	Class	Installed Version	Repository Version	Description
All Packages	mingw-developer-tool	bin		2013072300	An MSYS Installation for MinGW Developers (meta)
	🐑 mingw32-base	bin		2013072200	A Basic MinGW Installation
	mingw32-gcc-ada	bin		4.8.1-4	The GNU Ada Compiler
	mingw32-gcc-fortran	bin		4.8.1-4	The GNU FORTRAN Compiler
	mingw32-gcc-g++	bin		4.8.1-4	The GNU C++ Compiler
	mingw32-gcc-objc	bin		4.8.1-4	The GNU Objective-C Compiler
	🐑 msys-base	bin		2013072300	A Basic MSYS Installation (meta)

### Figure 45. Setup MinGW and MSYS

4. Click "Apply Changes" in the "Installation" menu and follow the remaining instructions to complete the installation.

tallation Package Settings	
Update Catalogue Mark All Upgrades	Package
Apply Changes	mingw32-base mingw32-occ-ada
Quit Alt+F4	mingw32-gcc-fortran mingw32-gcc-g++
	mingw32-gcc-objc

### Figure 46. Complete MinGW and MSYS installation

5. Add the appropriate item to the Windows operating system path environment variable. It can be found under *Control Panel -> System and Security -> System -> Advanced System Settings* in the "Environment Variables..." section. The path is:

<mingw\_install\_dir>\bin

Assuming the default installation path, C:\MinGW, an example is shown below. If the path is not set correctly, the toolchain does not work.

#### NOTE

If you have "C:\MinGW\msys\x.x\bin" in your PATH variable (as required by KSDK 1.0.0), remove it to ensure that the new GCC build system works correctly.

	X
Computer Name Hardw	ware Advanced System Protection Remote
Environment Variable	es 🛛 🕅
Edit System Varia	able 🛛 🕅
Variable name:	Path
Variable value:	ogram Files (x86)\CMake\bin;C:\MinGW\bin
	OK Cancel
System variables	
System variables Variable	Value
System variables Variable OS	Value  Windows_NT
System variables Variable OS Path	Value Windows_NT C: \Program Files (x86) \Parallels\Parallel
System variables Variable OS Path PATHEXT	Value Windows_NT C:\Program Files (x86)\Parallels\Parallel .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;
System variables Variable OS Path PATHEXT PROCESSOR_A	Value Windows_NT C:\Program Files (x86)\Parallels\ParallelCOM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS; AMD64
System variables Variable OS Path PATHEXT PROCESSOR_A	Value         Windows_NT         C:\Program Files (x86)\Parallels\Parallel         .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;         . AMD64         New       Edit         Delete

Figure 47. Add Path to systems environment

# 7.1.3 Add a new system environment variable for ARMGCC\_DIR

Create a new *system* environment variable and name it ARMGCC\_DIR. The value of this variable should point to the ARM GCC Embedded tool chain installation path, which, for this example, is:

C:\Program Files (x86)\GNU Tools ARM Embedded\5.2 2015q4

Reference the installation folder of the GNU ARM GCC Embedded tools for the exact path name of your installation.

Environment Variables	
New System Varia	ble 🛛 🕅
Variable name:	ARMGCC_DIR
Variable value:	C:\Program Files (x86)\GNU Tools ARM Emb
	OK Cancel
System variables	
Variable	Value
OS	Windows_NT
Path	C:\Program Files (x86)\Parallels\Parallel
PATHEXT PROCESSOR_A	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS; AMD64
	New Edit Delete
	OK Cancel

Figure 48. Add ARMGCC\_DIR system variable

# 7.1.4 Install CMake

- 1. Download CMake 3.0.x from www.cmake.org/cmake/resources/software.html.
- 2. Install CMake, ensuring that the option "Add CMake to system PATH" is selected when installing. The user chooses to select whether it is installed into the PATH for all users or just the current user. In this example, it is installed for all users.

#### Run a demo using ARM GCC

Install Options		
Choose options		
choose options i	for installing CMake 3.0.2	
s not add its directory to	the system PATH.	
to the system PATH	-	
system PATH for all user system PATH for current	rs t user	
ktop Icon		
.46	< Back Next >	Cancel
	s not add its directory to to the system PATH system PATH for all user system PATH for current ktop Icon	s not add its directory to the system PATH. to the system PATH system PATH for all users system PATH for current user ktop Icon <.46 <.46 <.46

### Figure 49. Install CMake

- 3. Follow the remaining instructions of the installer.
- 4. You may need to reboot your system for the PATH changes to take effect.

# 7.2 Build an example application

To build an example application, follow these steps.

1. Open a GCC ARM Embedded tool chain command window. To launch the window, from the Windows operating system Start menu, go to "Programs -> GNU Tools ARM Embedded <version>" and select "GCC Command Prompt".

GNU Tools for ARM Embedded Process			
Documentation			
GCC Command Prompt			
Uninstall GNU Tools for ARM Embed			

### Figure 50. Launch command prompt

2. Change the directory to the example application project directory, which has a path like this:

#### Run a demo using ARM GCC

<install\_dir>/boards/<board\_name>/<example\_type>/<application\_name>/armgcc

For this example, the exact path is: <*install\_dir>/examples/frdmk64f/demo\_apps/hello\_world/armgcc* 

NOTE

To change directories, use the 'cd' command.

3. Type "build\_debug.bat" on the command line or double click on the "build\_debug.bat" file in Windows Explorer to perform the build. The output is shown in this figure:



Figure 51. hello\_world demo build successful

# 7.3 Run an example application

This section describes steps to run a demo application using J-Link GDB Server application. To perform this exercise, two things must be done:

- Make sure that either:
  - The OpenSDA interface on your board is programmed with the J-Link OpenSDA firmware. To determine if your board supports OpenSDA, see Appendix B. For instructions on reprogramming the OpenSDA interface, see Appendix C. If your board does not support OpenSDA, then a standalone J-Link pod is required.
  - You have a standalone J-Link pod that is connected to the debug interface of your board. Note that some hardware platforms require hardware modification in order to function correctly with an external debug interface.

After the J-Link interface is configured and connected, follow these steps to download and run the demo application:

- 1. Connect the development platform to your PC via USB cable between the OpenSDA USB connector (may be named OSJTAG for some boards) and the PC USB connector. If using a standalone J-Link debug pod, also connect it to the SWD/JTAG connector of the board.
- 2. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug serial port number (to determine the COM port number, see Appendix A). Configure the terminal with these settings:
  - a. 115200 or 9600 baud rate, depending on your board (reference BOARD\_DEBUG\_UART\_BAUD variable in board.h file)
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit

tegory:			
Session	Basic options for your PuTTY session		
Logging	Specify the destination you w	vant to connect to	
- Terminal	Serial li <u>n</u> e	Speed	
- Bell	COM16	115200	
- Features	Connection type:	ogin 💿 <u>S</u> SH 💿 Serial	
Appearance Behaviour Translation	Load, save or delete a stored Sav <u>e</u> d Sessions	d session	
- Selection	Debug		
Colours Connection Data Proxy Telnet Rlogin SSH Serial	Default Settings	Load	
	Debug	Sa <u>v</u> e	
		Delete	
	Close window on e <u>x</u> it: Always ONever	Only on clean exit	

### Figure 52. Terminal (PuTTY) configurations

- 3. Open the J-Link GDB Server application. Assuming the J-Link software is installed, the application can be launched by going to the Windows operating system Start menu and selecting "Programs -> SEGGER -> J-Link <version> J-Link GDB Server".
- 4. Modify the settings as shown below. The target device selection chosen for this example is the MK64FN1M0xxx12.

Run a demo using ARM GCC

SEGGER J-Link GDB Server V5.001 - Config
Connection to J-Link
💿 USB 🔲 Serial No.
C TCP/IP
Target device
MK64FN1M0xxx12
Little endian
Target interface
SWD
- Speed Misc. settings
C Auto selection 🔲 Init registers
C Adaptive clocking
● 1000        ▼ kHz
Command line option
-select USB -device MK64FN1M0xxx12 -if SWD -speed 1000 -noir
OK Cancel Exit

### Figure 53. SEGGER J-Link GDB Server configuration

5. After it is connected, the screen should resemble this figure:

#### Run a demo using ARM GCC

SEGGER J-Link GDB Server V5.001		
File Help		
GDB Waiting for connection J-Link Connected CPU MK64FN1M0xxx12	Initial SWD speed 1000 kHz  Current SWD speed 1000 kHz 3.30 V Little endian	<ul> <li>✓ Localhost only</li> <li>✓ Stay on top</li> <li>✓ Show log window</li> <li>✓ Generate logfile</li> <li>✓ Verify download</li> <li>✓ Init regs on start</li> </ul>
Log output: Target interface speed: Target endian:	1000kHz little	*
J-Link is connected. Firmware: J-Link OpenSDA : Hardware: V1.00 S/N: 621000000	2 compiled Apr 24 2014 14:	44:11
Checking target voltage Target voltage: 3.30 V Listening on TCP/IP port 3 Connecting to targetCon Waiting for GDB connection	2331 nnected to target n	
<		+
0 Bytes downloaded	1 JTAG device	1.

### Figure 54. SEGGER J-Link GDB Server screen after successful connection

6. If not already running, open a GCC ARM Embedded tool chain command window. To launch the window, from the Windows operating system Start menu, go to "Programs -> GNU Tools ARM Embedded <version>" and select "GCC Command Prompt".



### Figure 55. Launch command prompt

7. Change to the directory that contains the example application output. The output can be found in using one of these paths, depending on the build target selected:

<install\_dir>/boards/<board\_name>/<example\_type>/<application\_name>/armgcc/debug

<install\_dir>/boards/<board\_name>/<example\_type>/<application\_name>/armgcc/release

For this example, the path is:

<install\_dir>/boards/frdmk64f/demo\_apps/hello\_world/armgcc/debug

#### **KSDK Project Generator**

8. Run the command "arm-none-eabi-gdb.exe <application\_name>.elf". For this example, it is "arm-none-eabi-gdb.exe hello\_world.elf".



Figure 56. Run arm-none-eabi-gdb

- 9. Run these commands:
  - a. "target remote localhost:2331"
  - b. "monitor reset"
  - c. "monitor halt"
  - d. "load"
  - e. "monitor reset"
- 10. The application is now downloaded and halted at the reset vector. Execute the "monitor go" command to start the demo application.

The hello\_world application is now running and a banner is displayed on the terminal. If this is not true, check your terminal settings and connections.



Figure 57. Text display of the hello\_world demo

# 8 KSDK Project Generator

The KSDK Project Generator tool can generate new projects for IAR, Keil MDK, Kinetis Design Studio and Atollic TrueSTUDIO IDEs containing all KSDK drivers and utilities which are supported for the selected device and also generate new standalone projects based on KSDK examples.

The KSDK Project Generator can be found on www.nxp.com. For more information about the installation process, see the KSDK Project Generator Release Notes.

# 8.1 Create projects using KSDK project generator

Projects can be generated in these modes:

- A quick generation, which creates only linked projects for a board on all supported toolchains
- An advanced generation, which creates linked/standalone projects for a board/device or cloned projects for selected toolchains with a selected RTOS.

# 8.2 Create a linked project

Create a linked project for a board in the quick generation mode.



### Figure 58. Quick generation mode

- 1. Run the tool and set the correct KSDK path using the Browse button.
- 2. Type the project name into the corresponding field and select a board in the list of boards.
- 3. Click on the *Quick Generate!* button.
- 4. A dialog with the information about the project location is displayed.

Create a linked project for a board or a device in the advanced generation mode (Generator 2 image).

**KSDK Project Generator** 

KSDK Project Generator 2.1		
Project Name: myProject	Device: MK64FN1M0xxx12	Help
Project Type:	Device or Board:	
• New C Clone	Device C Board     Choose Supported Toolchain(s):     Kinetis Design Studio	select device or board
., po p. 0,000	IAR Embedded Workbench for	ARM
RTOS Configuration:	🗆 Keil MDK-ARM	choose toolchains
• None	Atollic TrueSTUDIO	for which project
C FreeRTOS		should be
C uC/OS-II		generated
© uC/OS-Ⅲ		
Project Parent Directory:		
C:\Freescale\clon7\SDK_2.0_FRDM-K6	4F-4-test\boards\frdmk64f\user_apps	Browse
🗌 Generate standalone project	start proj	ect generation
Return Click here to return to previous menu.	$\rightarrow$	Advanced Generate!

### Figure 59. Advanced generation mode

- 1. Run the tool and set te correct KSDK path using the *Browse* button.
- 2. Click on the *Advanced* button.
- 3. Type the project name into the corresponding field.
- 4. In the *Device or Board*: section select the *Device* option and choose a device name or select a *Board* option and choose a board name.
- 5. Select toolchains in the corresponding list.
- 6. Optionally, choose an RTOS in the *RTOS Configuration*.
- 7. Click on the Advanced Generate! button.
- 8. A dialog with the information about the project location is displayed.

# 8.3 Create a standalone project

Create a standalone project for a board or a device (Generator 2):

#### **KSDK Project Generator**

KSDK Project Generator 2.1		
Project Name: myProject	Device: MK64FN1M0xxx12	Help
Project Type:	Device or Board:	
type project name	Choose Supported Toolchain(s):	select device or board
	IAR Embedded Workbench for A	ARM
RTOS Configuration:	🗆 Keil MDK-ARM	choose toolchains
None	Atollic TrueSTUDIO	for which project
C FreeRTOS		should be
⊂ uC/OS-II		generated
⊂ uC/OS-Ⅲ		
Project Parent Directory:		
C:\Freescale\clon7\SDK_2.0_FRDM-K64F-	4-test\boards\frdmk64f\user_apps	Browse
Generate standalone project	start proje	ect generation
Return Click here to return to previous menu.	$\rightarrow$	Advanced Generate!
click here to return to previous menu.		

### Figure 60. Advanced generation mode

- 1. Run the tool and set the correct KSDK path, using the Browse button.
- 2. Click on the *Advanced* button.
- 3. Type the project name in the corresponding field.
- 4. In the *Device or Board:* section select the *Device* option and choose the device name or select the *Board* option and choose a board name.
- 5. Select toolchains in the corresponding list.
- 6. Optionally, choose an RTOS in the *RTOS Configuration*.
- 7. Choose the *Generate standalone project* option.
- 8. Click on the Advanced Generate! button.
- 9. A dialog with information about the project location is displayed.

# 8.4 Clone KSDK examples using the KSDK Project Generator

Clone SDK examples - create new linked or standalone project based on KSDK examples.

Appendix A - How to determine COM port

NP KSDK Project Generator 2.1				
Project Name:	Board:			
myProject	FRDM-K64F  Help			
Project Type:	Project:			
🔿 New 🥿 💿 Clone	demo_apps/adc_low_power			
	Choose Supported Toolchain(s):			
select Clone option	🗌 Kinetis Design Studio			
	IAR Embedded Workbench for ARM			
RTOS Configuration:	Keil MDK-ARM select project which			
@ None	Atollic TrueSTUDIO should be cloned			
C FreeRTOS				
C uC/OS-II				
C uC/OS-III				
Project Parent Directory:				
C:\Freescale\clon7\SDK_2.0_FRDM-K64F-4-test\boards\frdmk64f\user_apps Browse				
Generate standalone project				
Return				
Click here to return to previous menu.	Advanced Generate!			

### Figure 61. Clone KSDK examples

- 1. Run the tool and set the correct KSDK path using the *Browse* button.
- 2. Click on the Advanced button.
- 3. Select the *Clone* option in the *Project Type* section.
- 4. Select the source project for cloning in the *Project* section.
- 5. Choose the *Generate standalone project* option to generate a standalone project. Otherwise, a generated project is linked (option *Generate standalone project* is supported only for some boards).
- 6. Click on the Advanced Generate! button.
- 7. A dialog with the information about the project location is displayed.

# 9 Appendix A - How to determine COM port

This section describes the steps necessary to determine the debug COM port number of your NXP hardware development platform. All NXP boards ship with a factory programmed, on-board debug interface, whether it's based on OpenSDA or the legacy P&E Micro OSJTAG interface. To determine what your specific board ships with, see Appendix B.

1. To determine the COM port, open the Windows operating system Device Manager. This can be achieved by going to the Windows operating system Start menu and typing "Device Manager" in the search bar, as shown below:

Control Panel (3)
🚔 Device Manager
View devices Device Manager Update device View and update your hardware's settings and driver s
Pictures (9)
Companies.inc
PTPStillImageTables.inc
SCSI_CDB_RcvCpyRslts.inc
hci_command_table.inc
CDCRequests.inc
Files (1)
dialog_settings.xml
₽ See more results
Device Manager × Shut down ►

### Figure 62. Device manager

- 2. In the Device Manager, expand the "Ports (COM & LPT)" section to view the available ports. Depending on the NXP board you're using (see Appendix B), the COM port can be named differently:
  - a. OpenSDA CMSIS-DAP/mbed/DAPLink interface:

Ports (COM & LPT)
 mbed Serial Port (COM41)



# **10** Appendix B - Default debug interfaces

The Kinetis SDK supports various Kinetis hardware platforms that come loaded with a variety of factory programmed debug interface configurations. The following table lists the hardware platforms supported by the KSDK, their default debug interface, and any version information that helps differentiate a specific interface configuration.

All recent and future NXP hardware platforms support the configurable OpenSDA standard.

Hardware platform	Default interface	OpenSDA details
FRDM-KV10Z	CMSIS-DAP	OpenSDA v2.1
FRDM-KV31F	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KV10Z32	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KV11Z75M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KV31F120M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KV46F150M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KV58F220M	CMSIS-DAP	OpenSDA v2.1

Table 1. Hardware platforms supported by KSDK

# 11 Appendix C - Updating OpenSDA firmware

#### Appendix C - Updating OpenSDA firmware

Any NXP hardware platform that comes with an OpenSDA-compatible debug interface has the ability to update the OpenSDA firmware. This typically means switching from the default application (either CMSIS-DAP/mbed/DAPLink or P&E Micro) to a SEGGER J-Link. This section contains the steps to switch the OpenSDA firmware to a J-Link interface. However, the steps can be applied to also restoring the original image.

For reference, OpenSDA firmware files can be found at the links below:

- <u>J-Link</u>: Download appropriate image from www.segger.com/opensda.html. Chose the appropriate J-Link binary based on the table in Appendix B. Any OpenSDA v1.0 interface should use the standard OpenSDA download (i.e., the one with no version). For OpenSDA 2.0 or 2.1, select the corresponding binary.
- <u>CMSIS-DAP/mbed/DAPLink</u>: This interface is provided to support the ARM mbed initiative. Navigate to developer.mbed.org/platforms and select your hardware platform. On the specific platform/board page, there is a link to the firmware image and instructions on how to load it, though the instructions are the same as below.
- <u>P&E Micro</u>: Downloading P&E Micro OpenSDA firmware images requires registration with P&E Micro (www.pemicro.com ).

These steps show how to update the OpenSDA firmware on your board for Windows operating system and Linux OS users:.

- 1. Unplug the board's USB cable.
- 2. Press the board's "Reset" button. While still holding the button, plug the board back in to the USB cable.
- 3. When the board re-enumerates, it shows up as a disk drive called "BOOTLOADER".

🥪 AutoPlay		
BOOTLOADER		
General options		
Open folder to view files using Windows Explorer		
View more AutoPlay options in Co	ontrol Panel	

### Figure 67. BOOTLOADER drive

4. Drag the new firmware image onto the BOOTLOADER drive in Windows operating system Explorer, similar to how you would drag and drop a file onto a normal USB flash drive.

#### NOTE

If for any reason the firmware update fails, the board can always re-enter bootloader mode by holding down the "Reset" button and power cycling.

These steps show how to update the OpenSDA firmware on your board for Mac OS users.

#### NOTE

The USB-KW019032 board has a specific OpenSDA interface, which is not compatible with the J-Link and P&E Micro OpenSDA firmware image.

- 1. Unplug the board's USB cable.
- 2. Press the board's "Reset" button. While still holding the button, plug the board back in to the USB cable.
- 3. For boards with OpenSDA v2.0 or v2.1, it shows up as a disk drive called "BOOTLOADER" in Finder. Boards with OpenSDA v1.0 may or may not show up depending on the bootloader version. If you see the drive in Finder, you may proceed to the next step. If you do not see the drive in Finder, use a PC with Windows<sup>®</sup> OS 7 or an earlier version to

#### **Revision History**

either update the OpenSDA firmware or update the OpenSDA bootloader to version 1.11 or later. The bootloader update instructions and image can be obtained from P&E Microcomputer website.

- 4. For OpenSDA v2.1 and OpenSDA v1.0 (with bootloader 1.11 or later) users, drag the new firmware image onto the BOOTLOADER drive in Finder, similar to how you would drag and drop the file onto a normal USB Flash drive.
- 5. For OpenSDA v2.0 users, type these commands in a Terminal window:
  - > sudo mount -u -w -o sync /Volumes/BOOTLOADER
  - > cp -X <path to update file> /Volumes/BOOTLOADER

#### NOTE

If for any reason the firmware update fails, the board can always re-enter bootloader mode by holding down the "Reset" button and power cycling.

# **12 Revision History**

This table summarizes revisions to this document.

### Table 2. Revision History

Revision number	Date	Substantive changes
2	07/2016	Added Chapter 8 and updated Section 5.5
1	06/2016	Added Section 5.5 related to the New Project Wizard for KSDK 2.0.0
0	01/2016	Initial release

#### How to Reach Us:

Home Page: nxp.com

Web Support: nxp.com/support Information in this document is provided solely to enable system and software implementers to use NXP 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. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP 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 NXP 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. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Freescale, the Freescale logo, 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, ARM powered logo, Keil, µVision, 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. All rights reserved.

© 2016 NXP B.V.

Document Number KSDK20GSUG Revision 2, 07/2016



