

Getting Started with Kinetis SDK (KSDK) v.1.3

1 Overview

Kinetis SDK (KSDK) is a Software Development Kit that provides comprehensive software support for Freescale Kinetis devices. The KSDK includes a Hardware Abstraction Layer (HAL) for each peripheral and peripheral drivers built on top of the HAL. Demo and driver example applications are provided to demonstrate driver and HAL usage and to highlight the main features of supported SoCs. Also, the KSDK contains the latest available RTOS kernels, a USB stack, and other middleware to support rapid development on supported Kinetis devices. The image below highlights the layers and features of the KSDK.

For supported toolchain versions, see the *Kinetis SDK v.1.3.0 Release Notes* (document KSDK130RN).

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

Contents

1	Overview.....	1
2	KSDK demo and example applications.....	2
3	Run a demo using IAR.....	5
4	Run a demo using Keil® MDK/μVision.....	11
5	Run a demo using Kinetis Design Studio IDE.....	18
6	Run a demo using Atollic® TrueSTUDIO®.....	30
7	Run a demo using ARM® GCC.....	37
8	Appendix A - How to determine COM port.....	47
9	Appendix B - Default debug interfaces	49
10	Appendix C - Updating OpenSDA firmware.....	50
11	Revision History.....	52

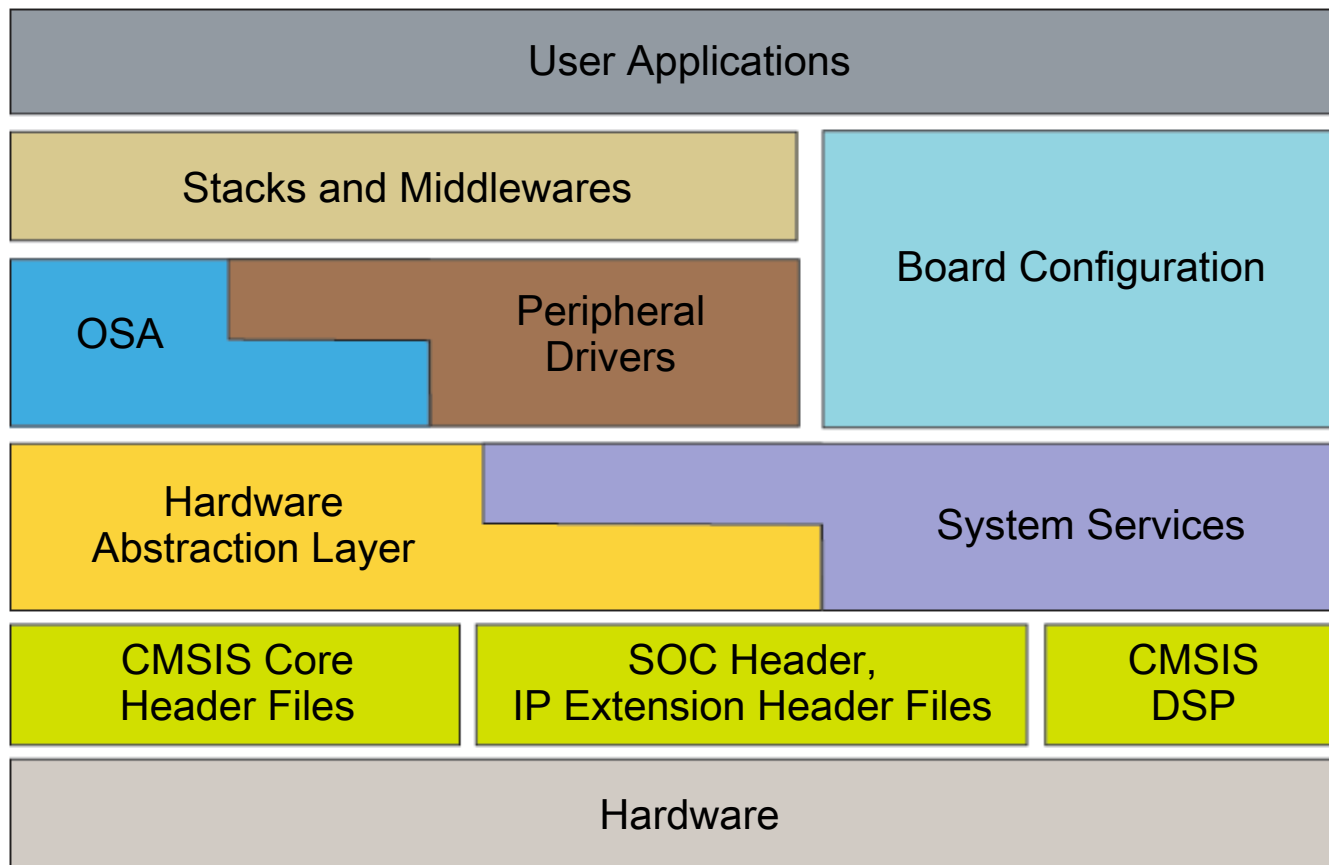


Figure 1. KSDK layers

2 KSDK demo and example applications

The KSDK provides two classes of software examples:

- **Demos Applications:** Full-featured applications intended to highlight key functionality of a MCU, focusing on a particular use case.
- **Driver Examples:** Simple applications intended to concisely illustrate how to use the KSDK's peripheral drivers.

This section describes how the demo and driver example applications interact with other components of the KSDK. To get a comprehensive understanding of all KSDK components and folder structure, see the *Kinetis SDK API Reference Manual* (Document KSDK13APIRM).

Demo and driver example applications reside in folders that correspond to a specific board (`<install_dir>/examples/<board_name>`). Within each board folder, there are folders containing sets of demo (demo_apps folder) and driver example (driver_examples folder) applications. This document focuses primarily on demo applications because they are full-featured applications. Everything that applies to driver demo applications is equally applicable to example applications.

When opening a board folder (`<install_dir>/examples/<board_name>`), this structure is observed:

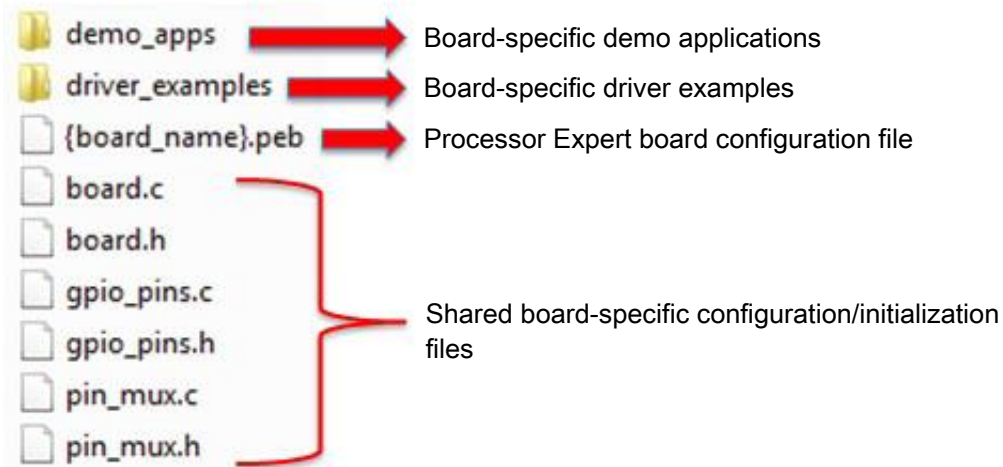


Figure 2. Board folder structure

At the top level of each board folder there is a common set of files used by the demos and driver examples. These files can be modified to do operations such as changing the pin mux configuration. All board support files provided as part of the KSDK are generated using reference *.peb file. These files are:

- **board.c/h:** The header file contains board-specific configuration macros for things such as debug terminal configuration, push buttons, LEDs, and other board-specific items. The C file contains clock and oscillator initialization functions.
- **gpio_pins.c/h:** Definitions used by the KSDK GPIO driver for the platform's GPIO pins. These include push buttons and LEDs, but can include other items such as interrupt pins for external sensors, for example.
- **pin_mux.c/h:** Contains peripheral-specific pin mux configurations. These functions can be called by the hardware_init() function or individually by the demo application.
- **Processor Expert PEB file:** Reference file for Freescale's Processor Expert tool for the specific board.

Moving down to the demo_apps folder, a typical use case looks something like this:

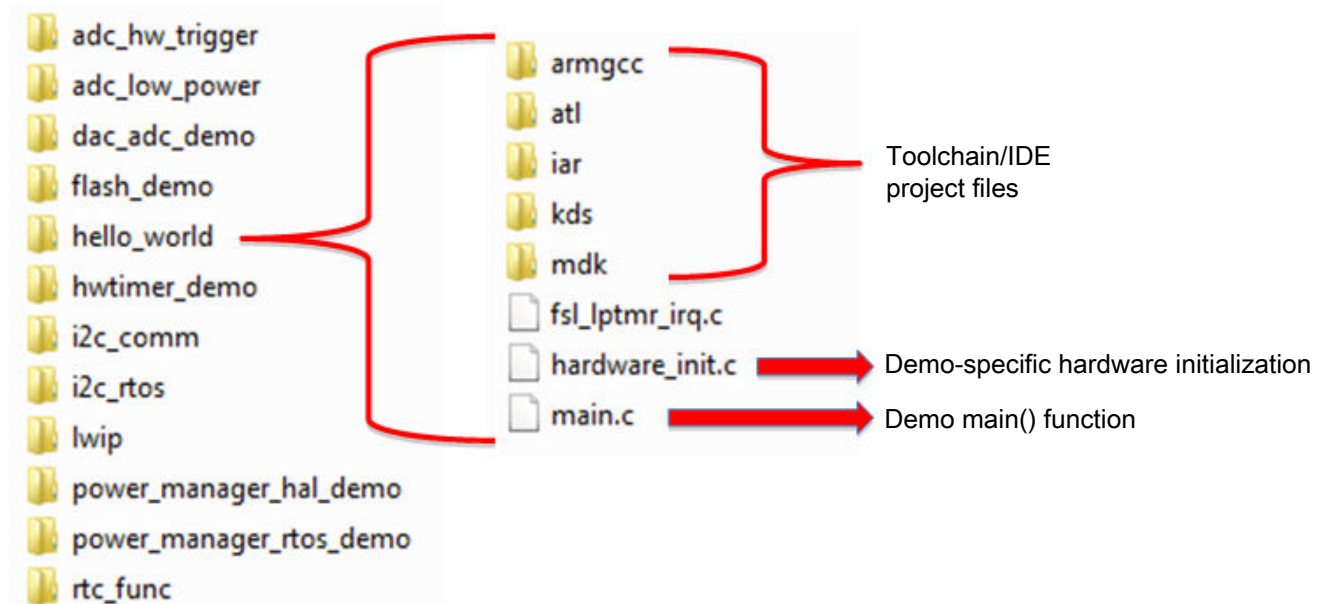


Figure 3. Demo Application folder structure

2.1 Locating demo source files

When opening a demo or driver example application in any of the supported IDEs, there are a variety of source files referenced. It is important to understand the location of these source files in the KSDK tree so that, if needed, they can be copied or modified to help develop applications for custom hardware later on. Additionally, many files are shared and, if modified, impact other demos. As a result, the user should have a full grasp of the KSDK structure to fully understand the effect of manipulating the source files.

There are two other main areas of the KSDK tree used to provide the full source code for each demo application:

- **<install_dir>/platform**: Contains shared, SoC-specific linker files, startup code and source for KSDK HAL, peripheral drivers, and system services.
- **<install_dir>/lib**: Contains the compiled library files of the KSDK platform components such as HAL, peripheral drivers, startup code, and system services.

2.2 KSDK platform folder

The platform folder is the most important folder in the KSDK. It contains the “foundation” of the KSDK, and stores the source code for the primary components including CMSIS header files, peripheral drivers, HAL, OS abstraction, startup, system services, and linker files. Building a demo application successfully requires a majority of these components.

When building a demo application that utilizes the KSDK platform components, two methods are possible: including individual source files for each required piece (startup file, driver, etc.), or link in a library that contains all or relevant components of the platform folder. All demo applications in the KSDK utilize the latter approach, choosing to provide a library that contains all source code in the platform folder. This simplifies application development because it only requires the include paths to be set correctly in the project files, as opposed to the user manually adding each file needed by the application.

2.3 KSDK lib folder

Previous sections describe how the KSDK demo applications reference a library containing all components of the platform folder. These library file projects reside in the KSDK top-level *lib* folder in the *ksdk_platform_lib* directory.

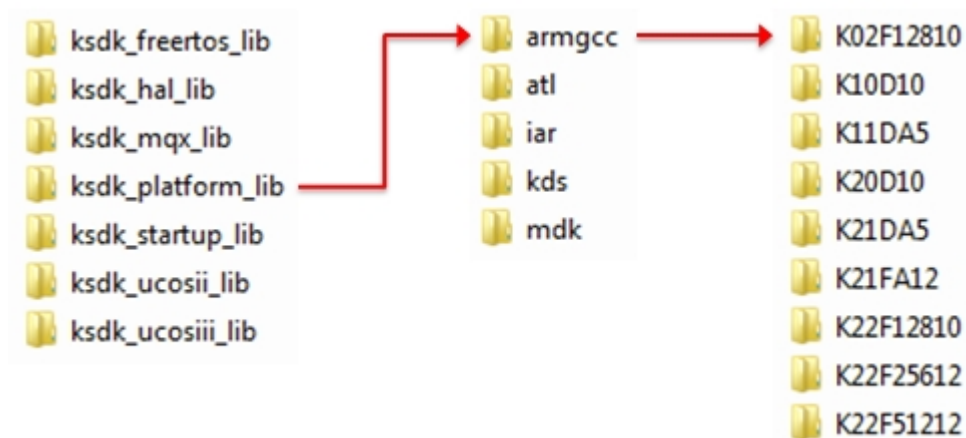


Figure 4. Lib folder

Each library configuration in the lib folder contains a folder for each supported toolchain. Each toolchain contains a folder for the supported SoC families. The *ksdk_platform_lib* must be built for the specific SoC being used in the demo. This is discussed in detail in the subsequent toolchain-specific sections.

3 Run a demo using IAR

This section describes the steps required to build, run, and debug demo applications provided in the Kinetis SDK. This section also shows how to build the necessary library that the demos use. 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.

3.1 Build the platform library

These steps show how to open the demo workspace in IAR Embedded Workbench, how to build the platform library required by the demo, and how to build the demo application.

1. Open demo workspace in: `<install_dir>/examples/<board_name>/demo_apps/<demo_name>/iar`

The workspace file is named `<demo_name>.eww`, so for this specific example, the actual path is:

```
<install_dir>/examples/frdmk64f/demo_apps/hello_world/iar/hello_world.eww
```

After the workspace is open, two projects are shown: one for the KSDK platform library and one for the demo. Also, the platform library project is bold, indicating that it is the active project. The active project can be changed at any time by right clicking on the desired project and selecting “Set as Active” or via the build target drop-down at the top of the workspace browser.

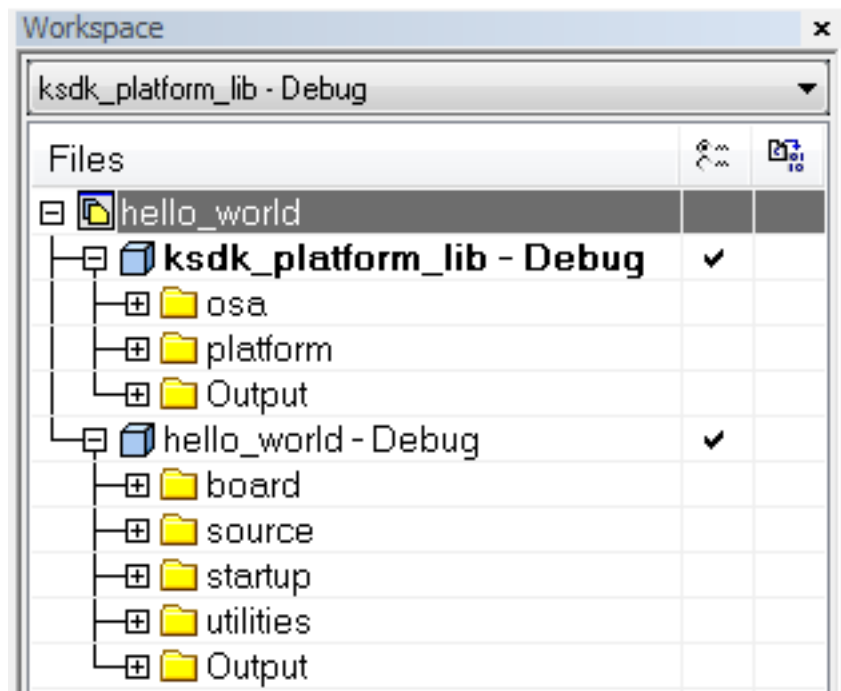


Figure 5. Workspace view

2. 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.

The tool allows you to select either the Debug or Release configuration on a per-project basis, but since the demo has a dependency on the platform library, whichever configuration is selected for the demo must also be selected for the platform library. Selecting a configuration in the drop-down also makes whichever project and configuration that is selected the active project.

For this example, select the “ksdk_platform_lib – Debug” target.

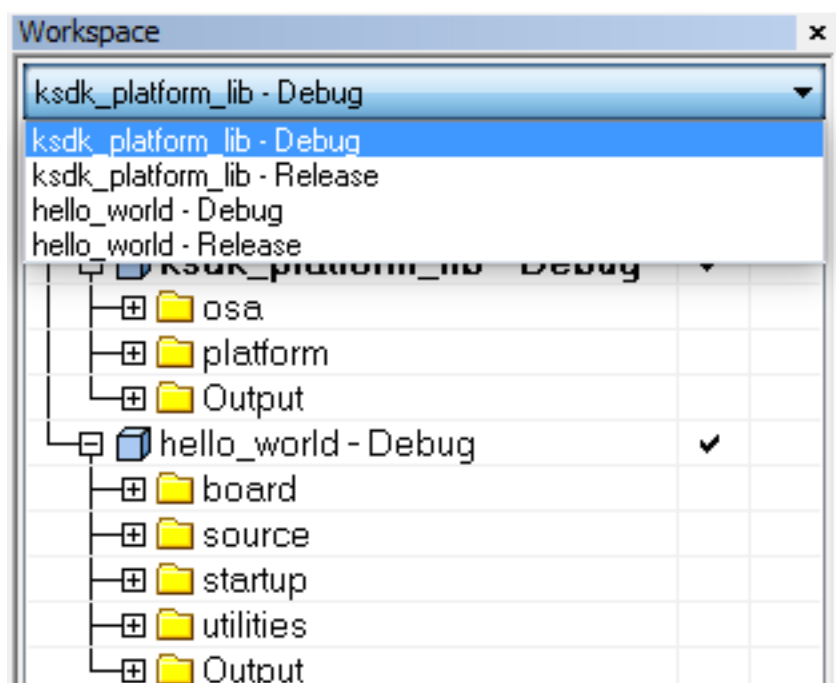


Figure 6. Platform library build target selection

3. Click the "Make" button, highlighted in red below.

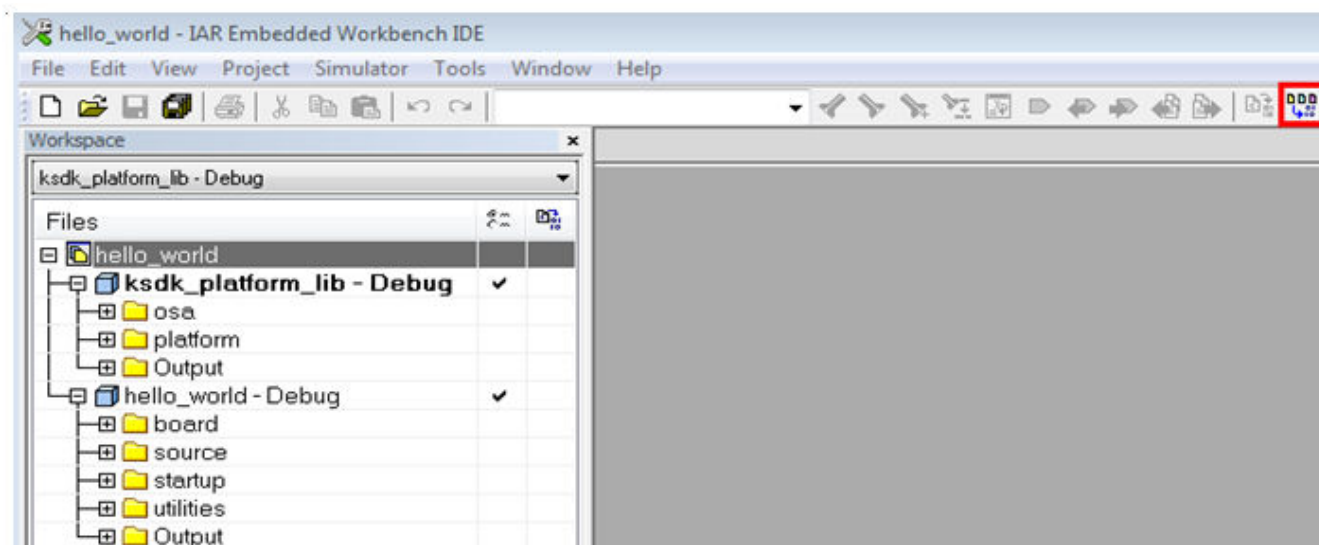


Figure 7. Build the platform library

4. When the build is complete, the library (libksdk_platform.a) is generated in one of the following directories, according to the chosen build target:

<install_dir>/lib/ksdk_platform_lib/iar/<device_name>/debug

<install_dir>/lib/ksdk_platform_lib/iar/<device_name>/release

3.2 Build a demo application

Run a demo using IAR

The KSDK demo applications are built upon the software building blocks provided in the Kinetis SDK platform library, built in the previous section. If the platform library is not present, the linker displays an error indicating that it cannot find the library. An easy way to check whether the library is present is to expand the Output folder in the ksdk_platform_lib project. If the platform library binary is not built and present, follow the steps in Section 3.1 to build it. Otherwise, continue with the following steps to build the desired demo application.

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

```
<install_dir>/examples/<board_name>/demo_apps/<demo_name>/iar
```

Using the FRDM-K64F Freedom board as an example, The hello_world workspace is located in this folder:

```
<install_dir>/examples/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.

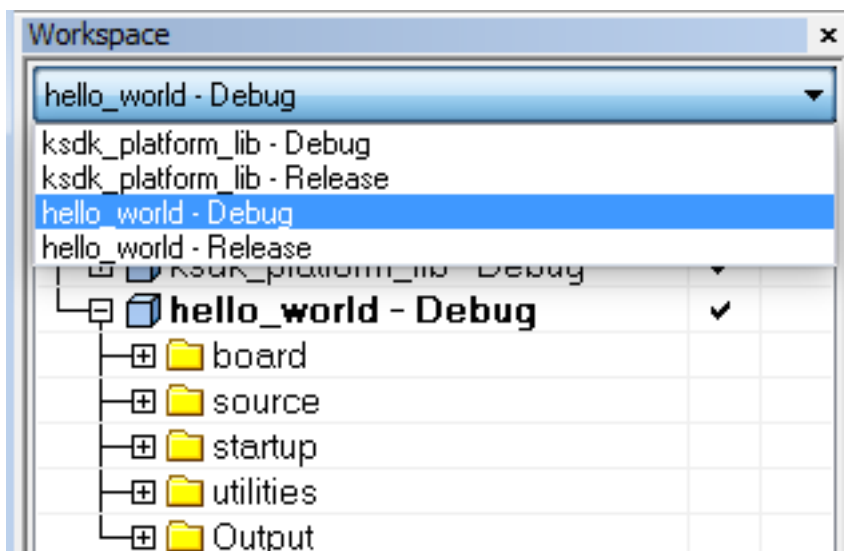


Figure 8. Demo build target selection

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

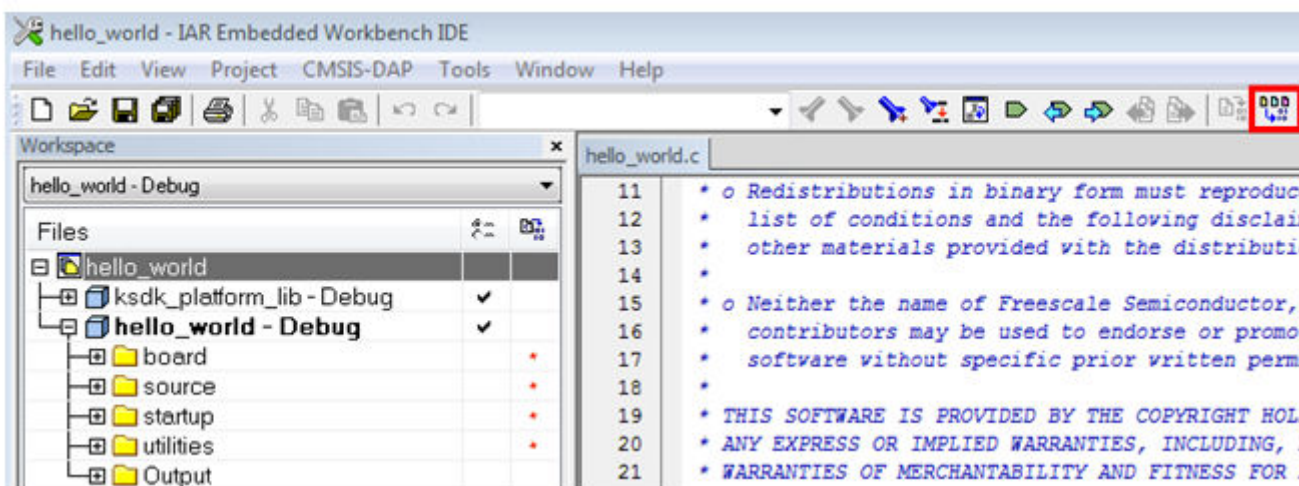


Figure 9. Build the demo application

4. The build will complete without errors.

3.3 Run a demo application

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® 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.freescale.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_BAUD variable in board.h file)
 - b. No parity
 - c. 8 data bits
 - d. 1 stop bit

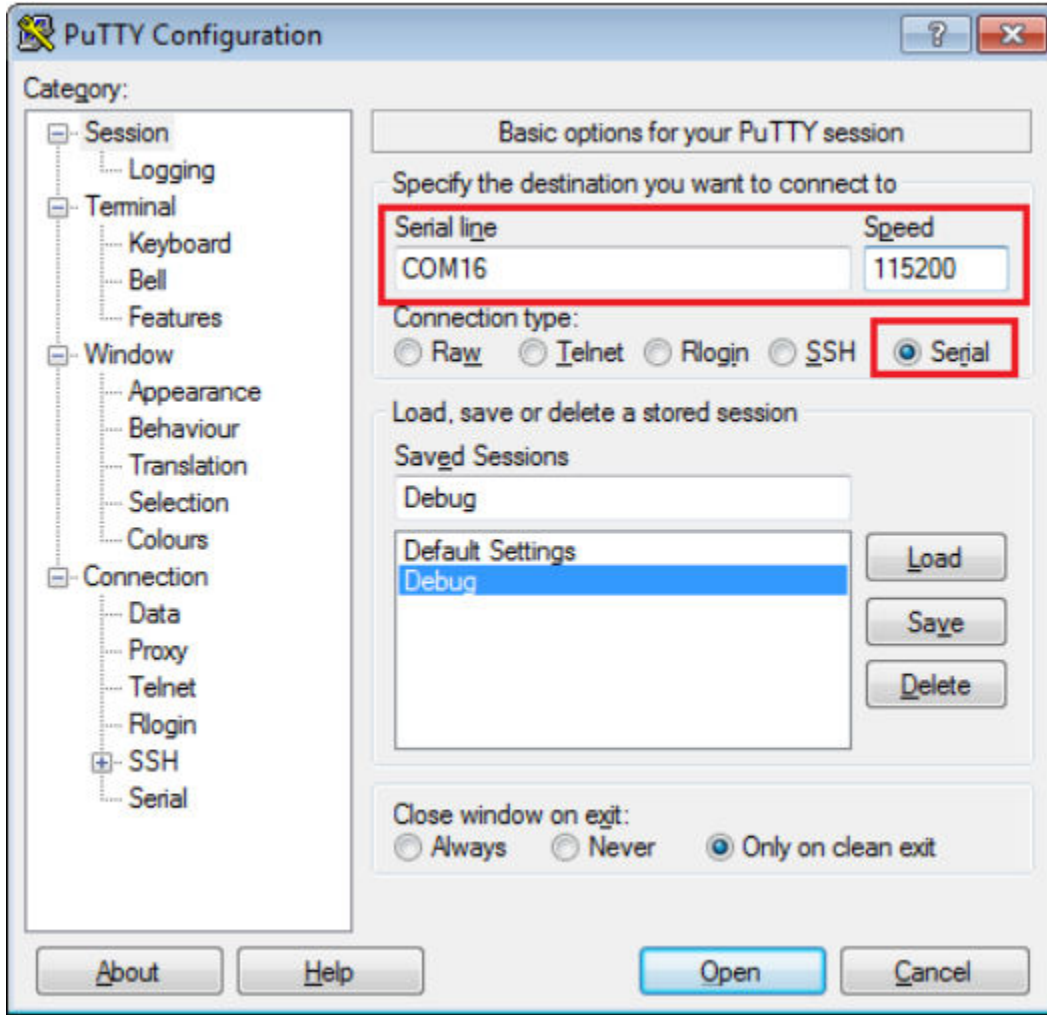


Figure 10. Terminal (PuTTY) configuration

- 4. Click the "Download and Debug" button to download the application to the target.



Figure 11. Download and Debug button

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

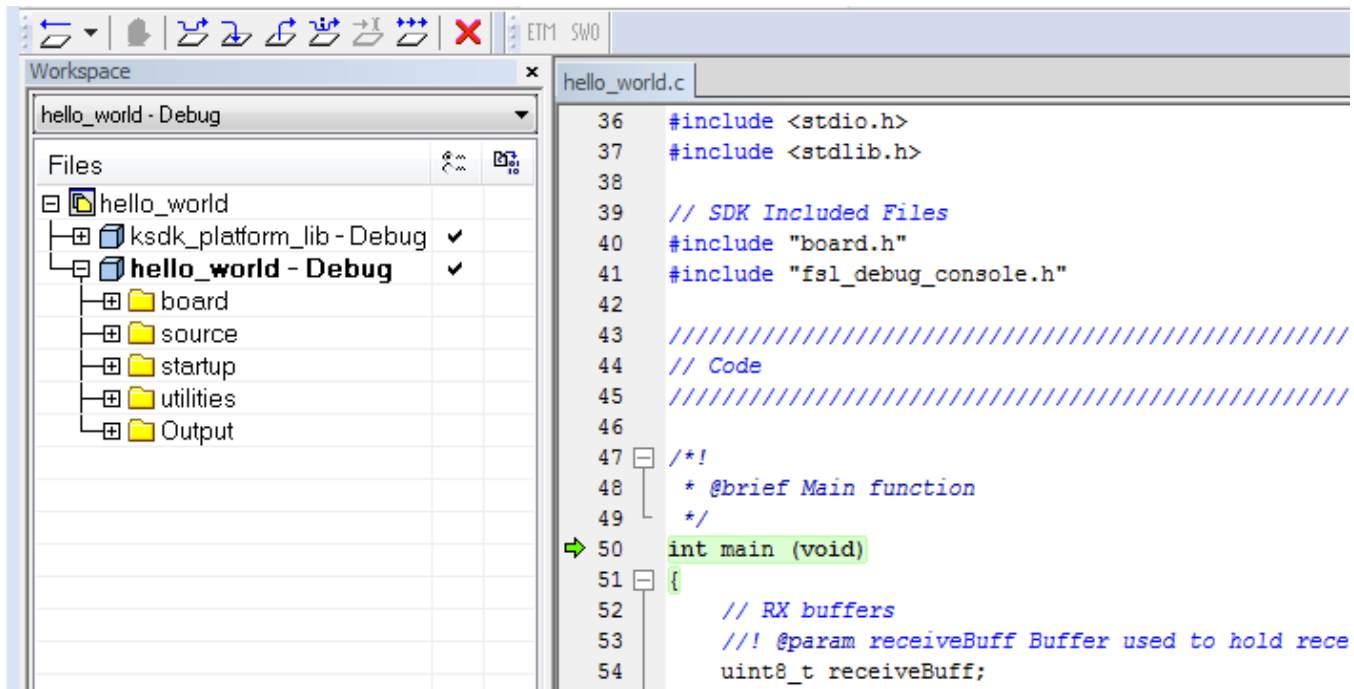


Figure 12. Stop at main() when running debugging

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

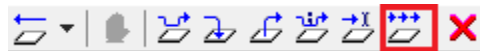


Figure 13. Go button

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.

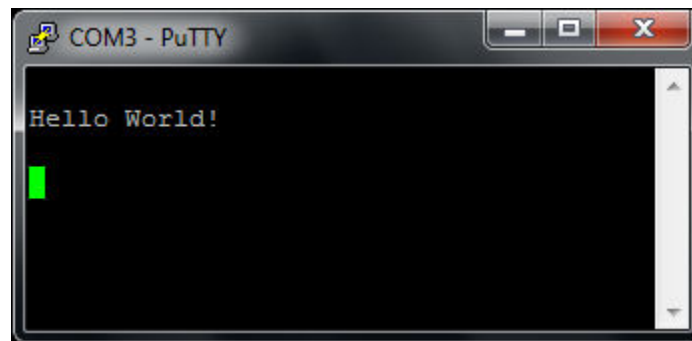


Figure 14. Text display of the hello_world demo

4 Run a demo using Keil® MDK/μVision

This section describes the steps required to build, run, and debug demo applications provided in the Kinetis SDK. This section also shows how to build the necessary library that the demos use. 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.

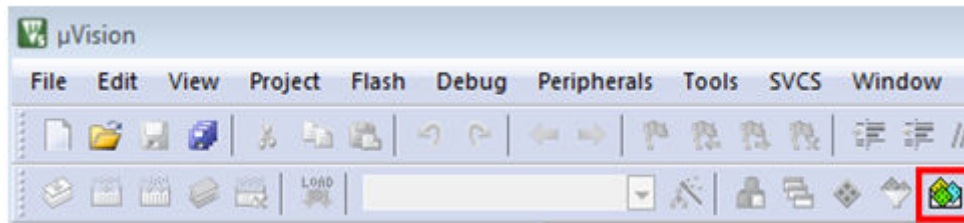


Figure 15. Launch the Pack installer

2. In the Pack Installer window, navigate to the section with the Kinetis packs (they are in alphabetical order). The Kinetis packs start with “Keil::Kinetis” and are followed by the MCU family name, for example “Keil::Kinetis_K60_DFP”. Because this example uses the FRDM-K64F platform, the K60 family pack is selected. Click on the “Install” button next to the pack. This process requires an Internet connection to successfully complete.

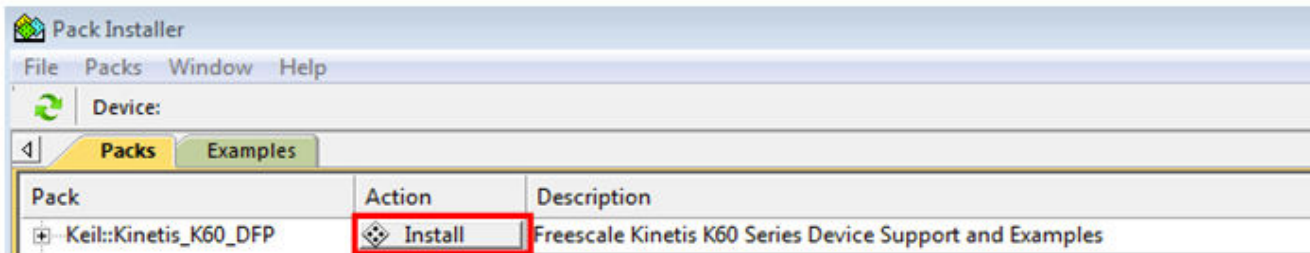


Figure 16. Install Kinetis pack

3. After the installation finishes, close the Pack Installer window and return to the μVision IDE.

4.2 Build the platform library

These steps show how to open the demo workspace in μVision, how to build the platform library required by the demo, and how to build the demo application.

1. Demo workspace files can be found using this path:

```
<install_dir>/examples/<board_name>/demo_apps/<demo_name>/<toolchain>
```

The workspace file is named <demo_name>.uvmpw. For this specific example, the actual path is:

```
<install_dir>/examples/frdmk64f/demo_apps/hello_world/mdk/hello_world.uvmpw
```

After the workspace is open, two projects show up: one for the KSDK platform library, and one for the demo. By default, the demo project is selected as the active project.

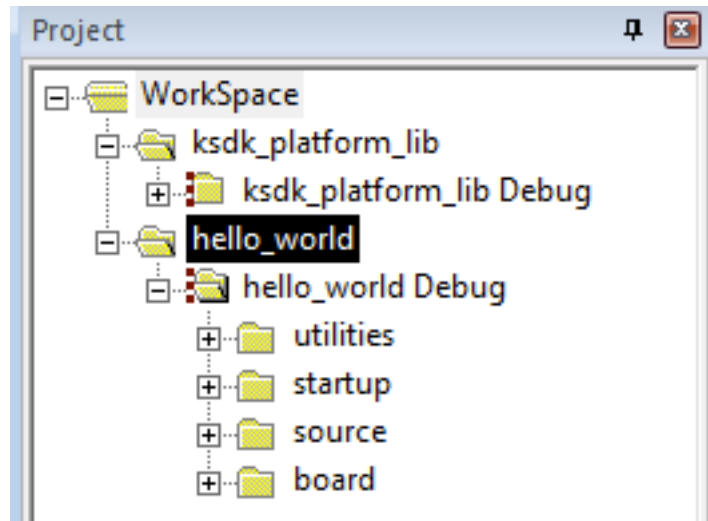


Figure 17. Workspace view

2. Make the platform library project the active project since the library is required by the demo application to build. To make the platform library project active, right click on it and select “Set as Active Project”. The active project has a black box around the project name. After it is active, the platform library project is highlighted.

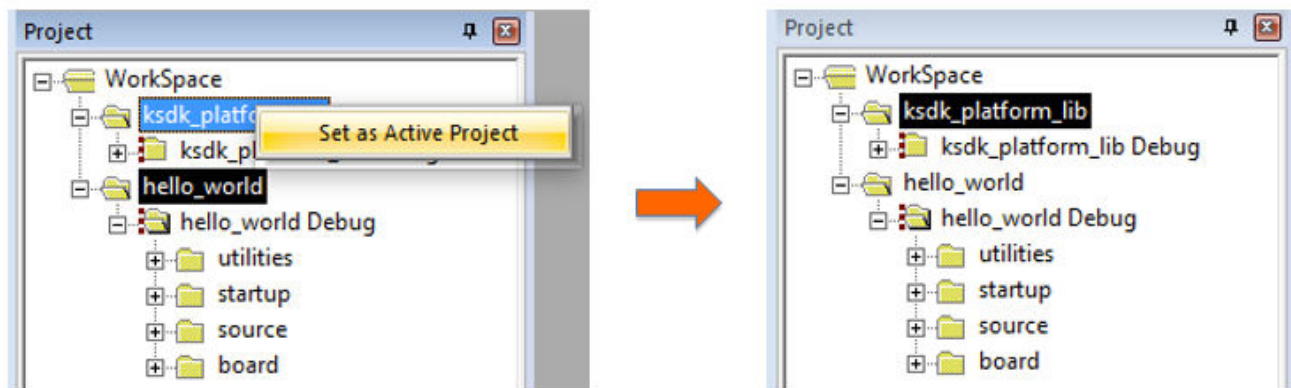


Figure 18. Make the platform library the active project

3. 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.

The tool allows selection of the build target based on the active project, so in order to change the configuration for the platform library it must be the active project. Choose the appropriate build target: “Debug” or “Release” from the drop-down menu.

For this example, select the “ksdk_platform_lib Debug” configuration.

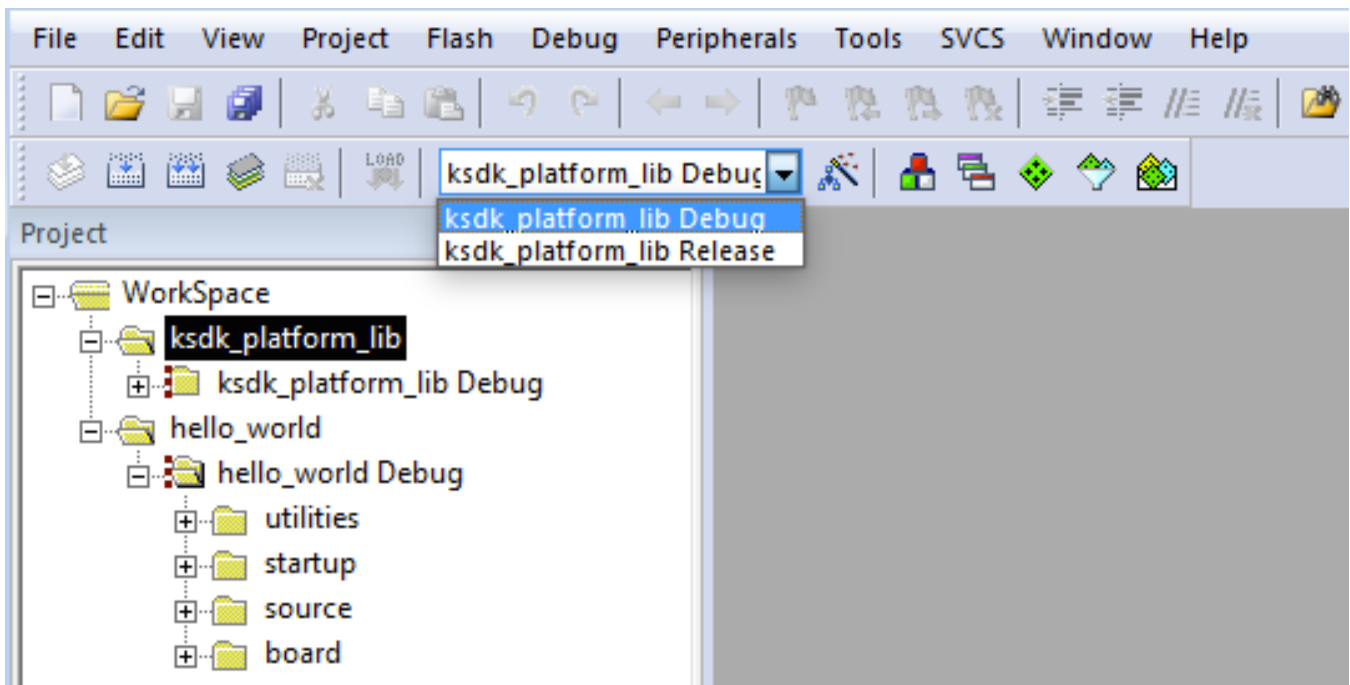


Figure 19. Platform library build target selection

4. Rebuild the project files by left-clicking the “Rebuild” button, highlighted in red.

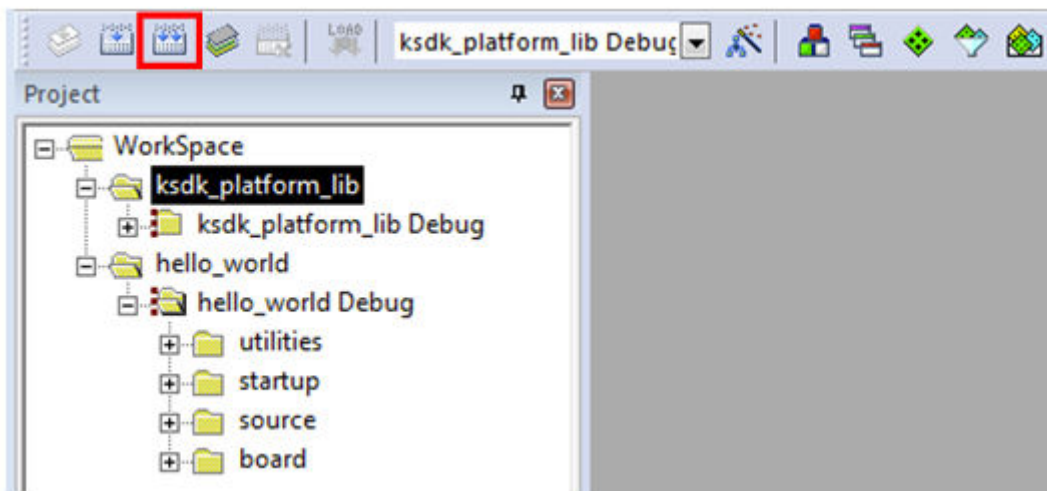


Figure 20. Build the platform library

5. When the build is complete, the library (libksdk_platform.lib) is generated in this directory according to the build target:

<install_dir>/lib/ksdk_platform_lib/mdk/<device_name>/debug

<install_dir>/lib/ksdk_platform_lib/mdk/<device_name>/release

4.3 Build a demo application

The KSDK demo applications are built upon the software building blocks provided in the Kinetis SDK platform library, built in the previous section. If the platform library is not present, the linker displays an error indicating that it cannot find the library. If the platform library binary is not built and present, follow the steps in Section 4.2 to build it. Otherwise, continue with the following steps to build the desired demo application.

1. If not already done, open the desired demo application workspace in:

```
<install_dir>/examples/<board_name>/demo_apps/<demo_name>/mdk
```

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

```
<install_dir>/examples/frdmk64f/demo_apps/hello_world/iar/hello_world.uvmpw
```

2. Make the demo the active project

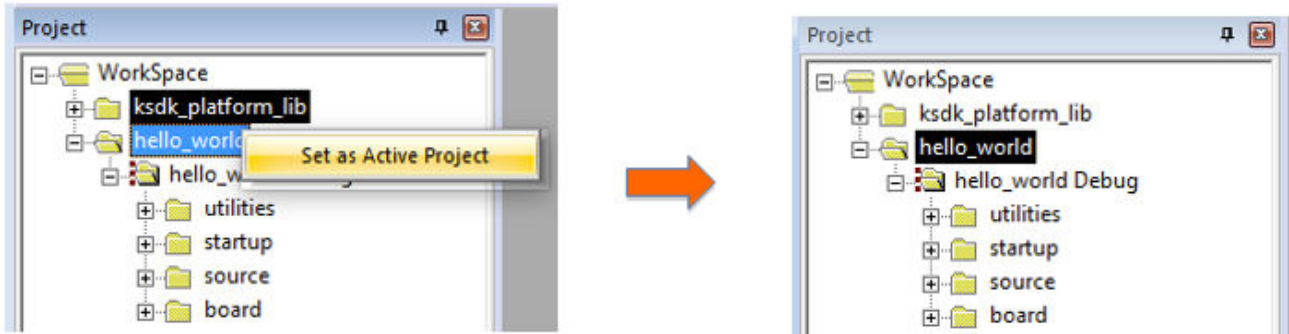


Figure 21. Make the demo project the active project

3. To build the demo project, select the "Rebuild" button, highlighted in red.

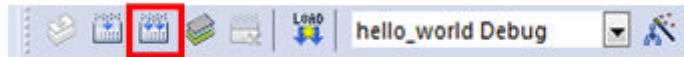


Figure 22. Build the demo

4. The build will complete without errors.

4.4 Run a demo application

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 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.freescale.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_BAUD variable in board.h file)

Run a demo using Keil® MDK/μVision

- b. No parity
- c. 8 data bits
- d. 1 stop bit

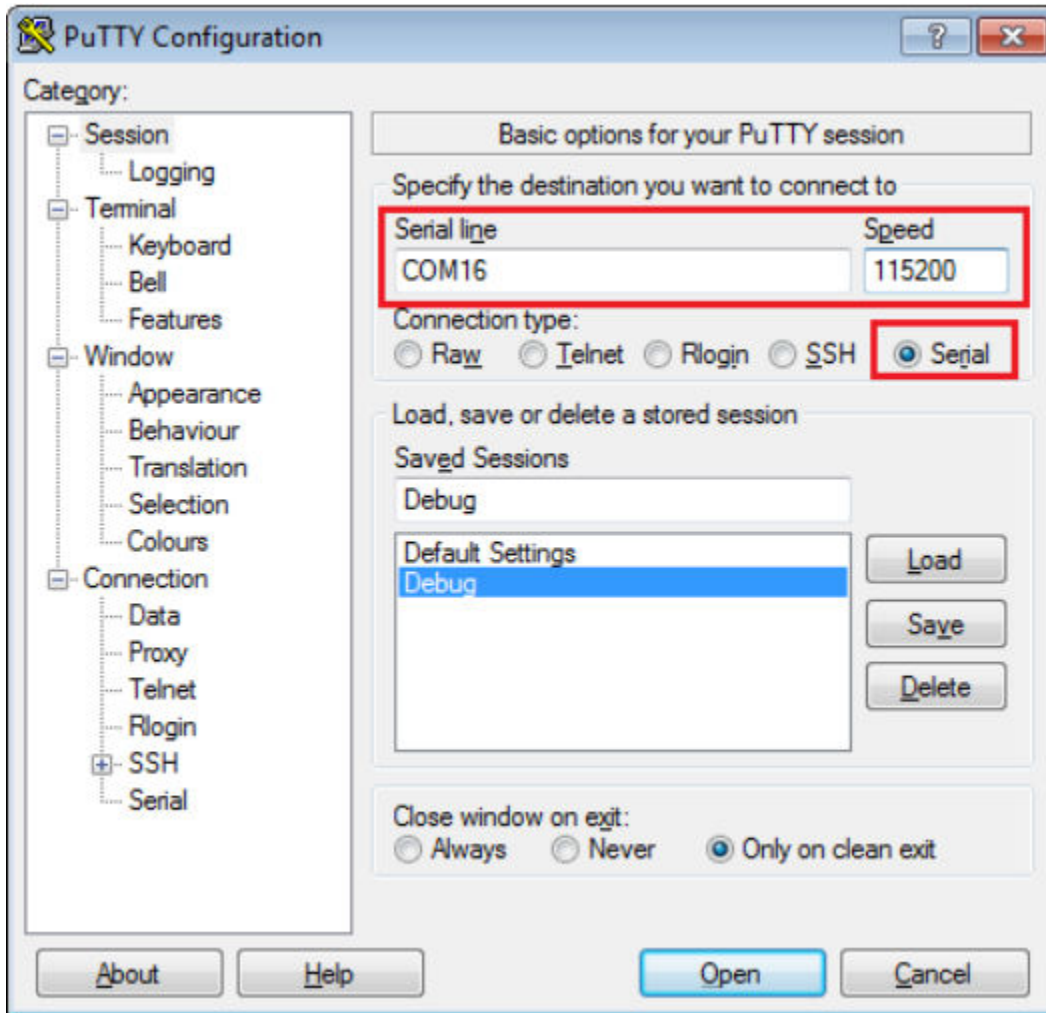


Figure 23. Terminal (PuTTY) configurations

4. After the application is properly built, click the "Download" button to download the application to the target.

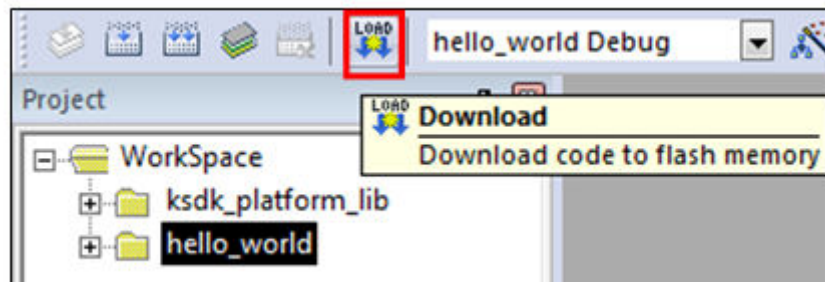


Figure 24. Download button

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.

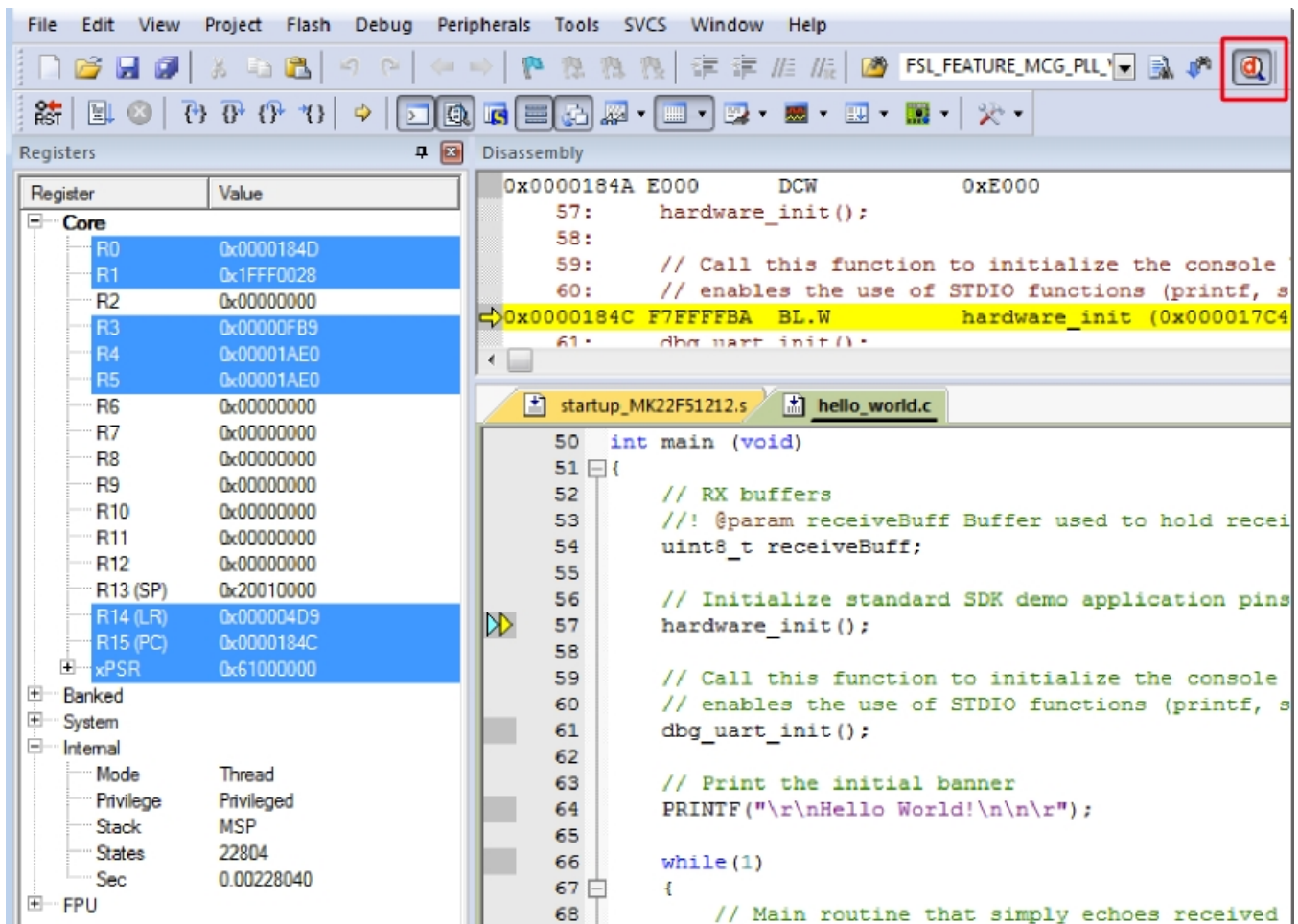


Figure 25. Stop at main() when run debugging

- Run the code by clicking the “Run” button to start the application.

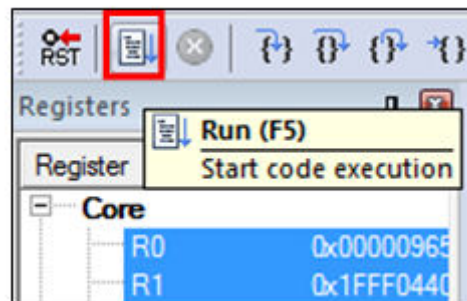


Figure 26. Go button

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.

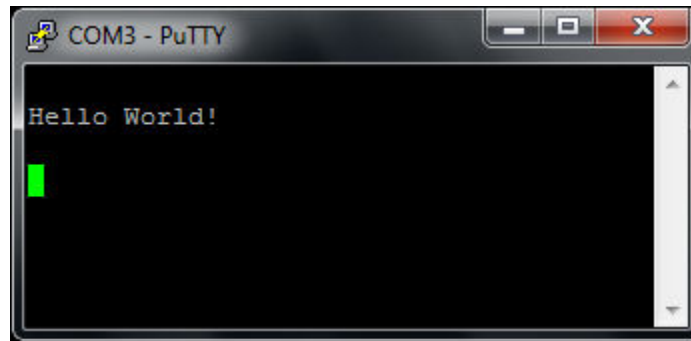


Figure 27. Text display of the hello_world demo

5 Run a demo using Kinetis Design Studio IDE

This section describes the steps required to configure Kinetis Design Studio (KDS) IDE to build, run, and debug demo applications and the necessary 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 demo or 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 Install Eclipse update

Before using KDS IDE with KSDK, the KSDK Eclipse Update must be applied. Without this update, Eclipse cannot generate KSDK-compatible projects.

5.2.1 Windows[®] operating system instructions and Mac[®] OS instructions

NOTE

The steps required for Mac OS are identical to those for the Windows operating system. The only difference is that the IDE looks slightly different.

To install the update, follow these instructions:

1. Select "Help" -> "Install New Software".

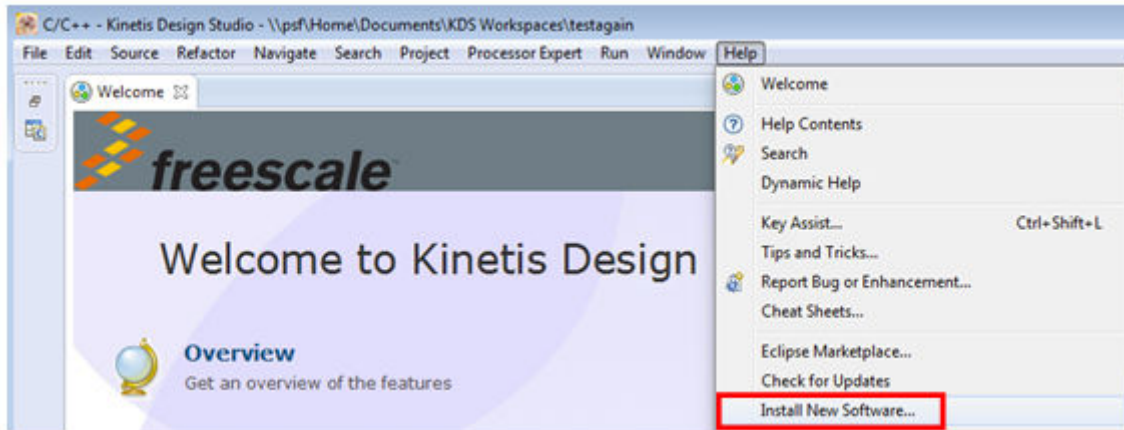


Figure 28. Install new software

2. In the Install New Software dialog box, click the "Add" button in the upper right corner. Then, in the Add Repository dialog, select the "Archive" button.

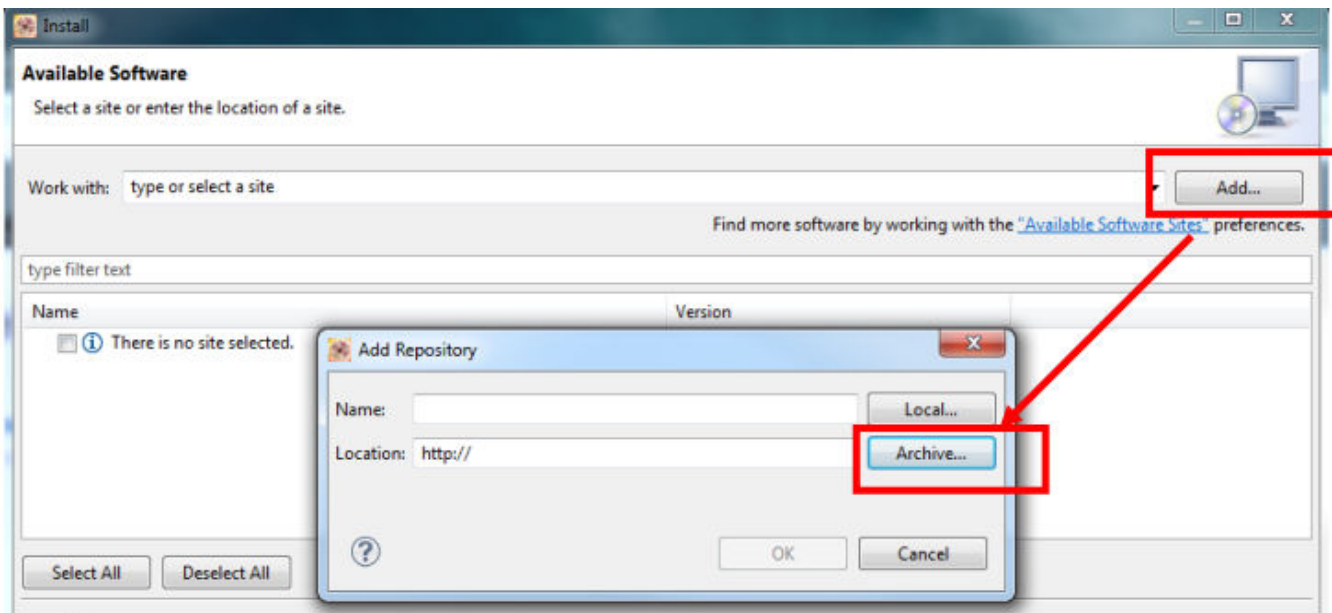


Figure 29. Add repository for new software

3. In the Repository archive dialog box, browse the KSDK install directory.
4. folder and select the *KSDK_<version>_Eclipse_Update.zip* file.
5. Click "Open", and the "OK" button in the Add Repository dialog box.
6. The KSDK update shows up in the list of the original Install dialogs.

Run a demo using Kinetis Design Studio IDE

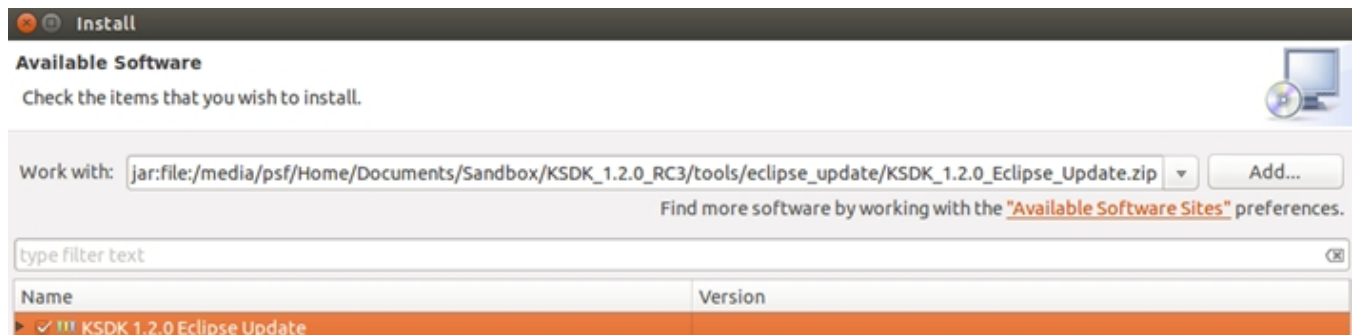


Figure 30. Select the Eclipse update

7. Check the box to the left of the KSDK Eclipse update and click the "Next" button in the lower right corner.
8. Follow the remaining instructions to finish the installation of the update.

After the update is applied, restart the KDS IDE for the changes to take effect.

5.2.2 Linux[®] OS instructions

The following instructions were performed using Ubuntu 14.04. These steps may be slightly different for other Linux OS distributions.

To install the update, follow these instructions:

1. Launch KDS IDE from the command line as the root user. On the command line, use this command, assuming the default KDS IDE install path: `user@ubuntu:~$ sudo /opt/Freescale/KDS_x.x.x/eclipse/kinetis-design-studio`

The KDS IDE version (shown above as x.x.x) should reflect the version installed on your machine, for example, 3.0.0.

2. Enter the root password.
3. Select "Help" -> "Install New Software".

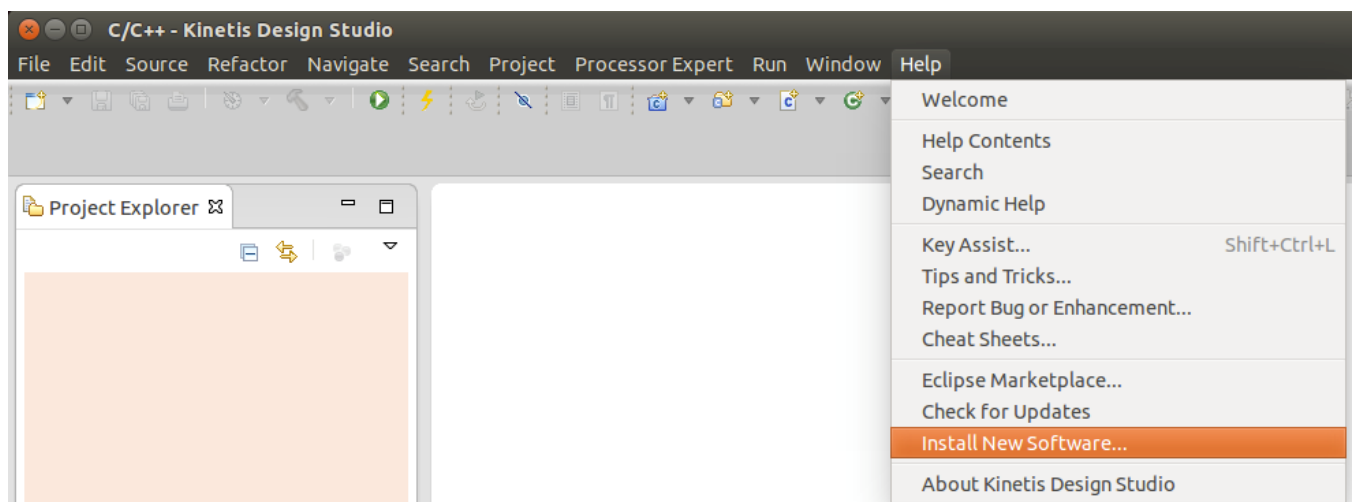


Figure 31. Install new software

4. In the "Install New Software" dialog box, click the "Add" button in the upper right corner. Then, in the "Add Repository" dialog, select "Archive".

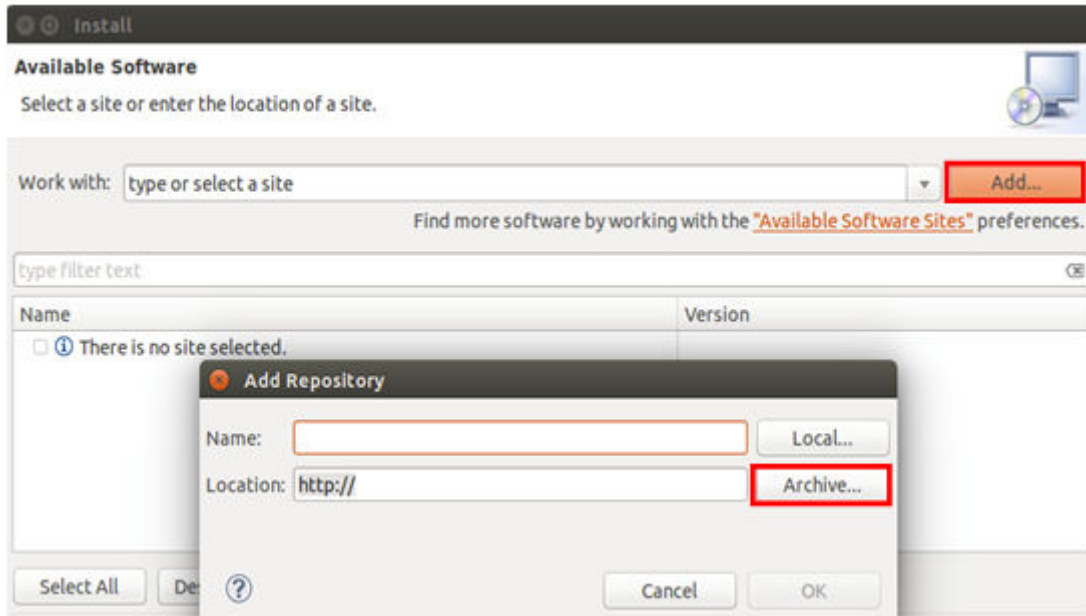


Figure 32. Add repository for new software

5. In the Repository archive dialog box, browse the KSDK install directory.
6. Enter the `<install_dir>/tools/eclipse_update` folder and select the `KSDK_<version>_Eclipse_Update.zip` file.
7. Click "Open", and "OK" in the "Add Repository" dialog box.
8. The KSDK update shows up in the list of the original Install dialogs.

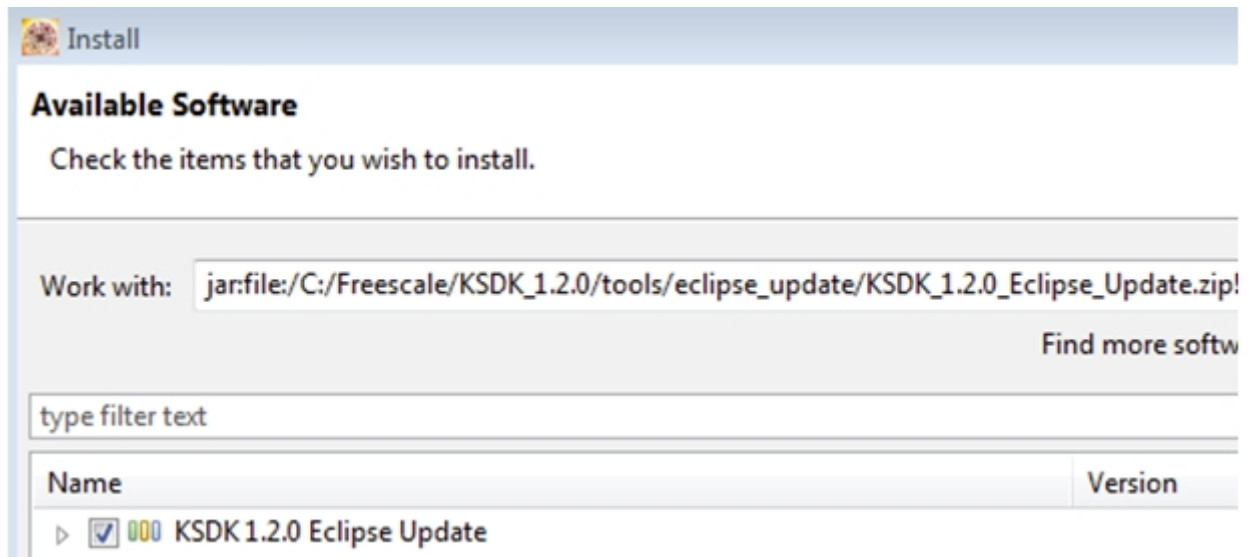


Figure 33. Select the Eclipse update

9. Check the box to the left of the KSDK Eclipse update and click the "Next" button in the lower right corner.
10. Follow the remaining instructions to finish the installation of the update.
11. After the update is applied, restart the KDS IDE for the changes to take effect.
12. After KDS IDE restarts, shut down the IDE and restart by launching KDS IDE as the *non-root user*. To do this, follow the command in step 1, only without the "sudo" command.

5.3 Build the platform library

These steps show how to open and build the platform library project in KDS IDE. The platform library is required by the demo and does not build without it.

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.

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

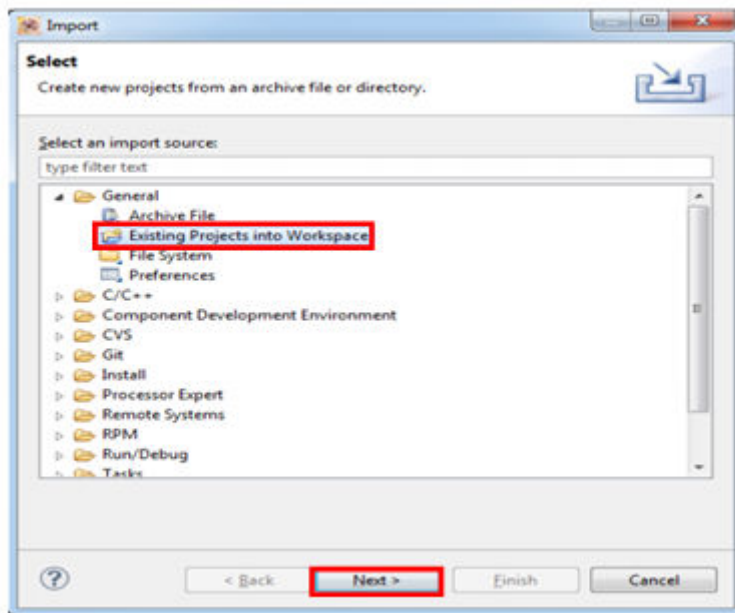


Figure 34. Selection of the correct import type in KDS IDE

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

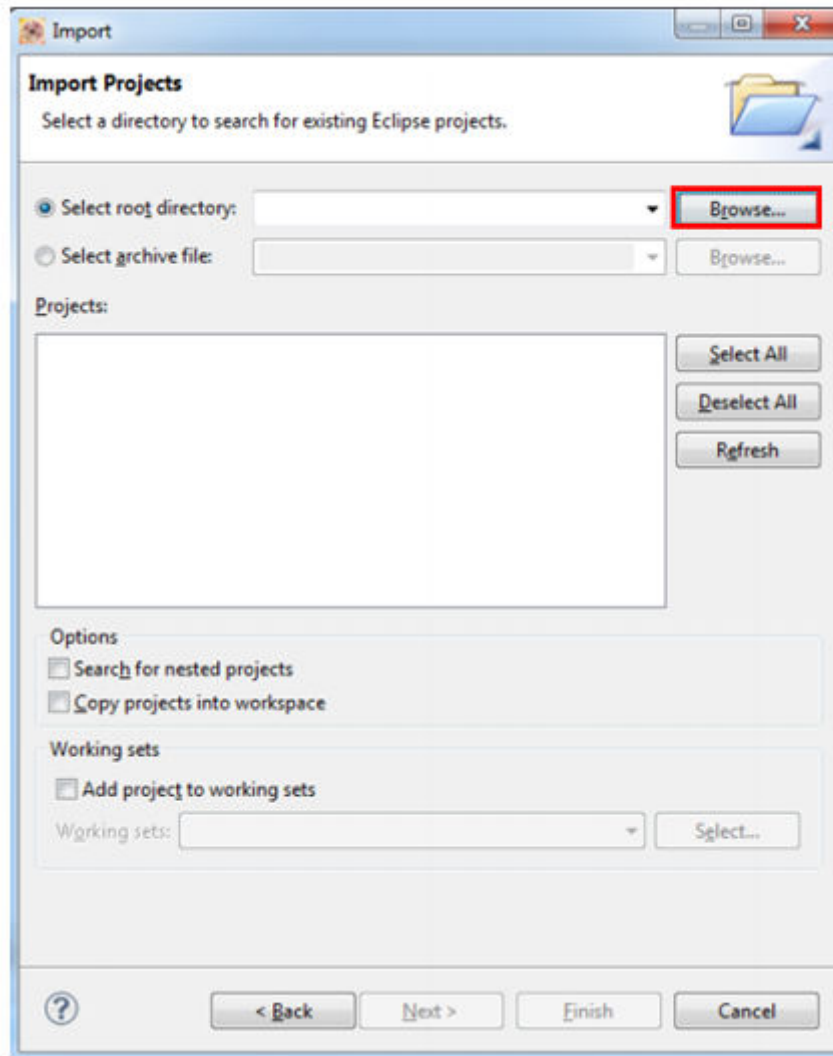


Figure 35. Projects directory selection window

- Point to the platform library project for the appropriate device, which can be found using this path:

<install_dir>/lib/ksdk_platform_lib/kds/<device_name>

For this example, the specific location is:

<install_dir>/lib/ksdk_platform_lib/kds/K64F12

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

NOTE

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

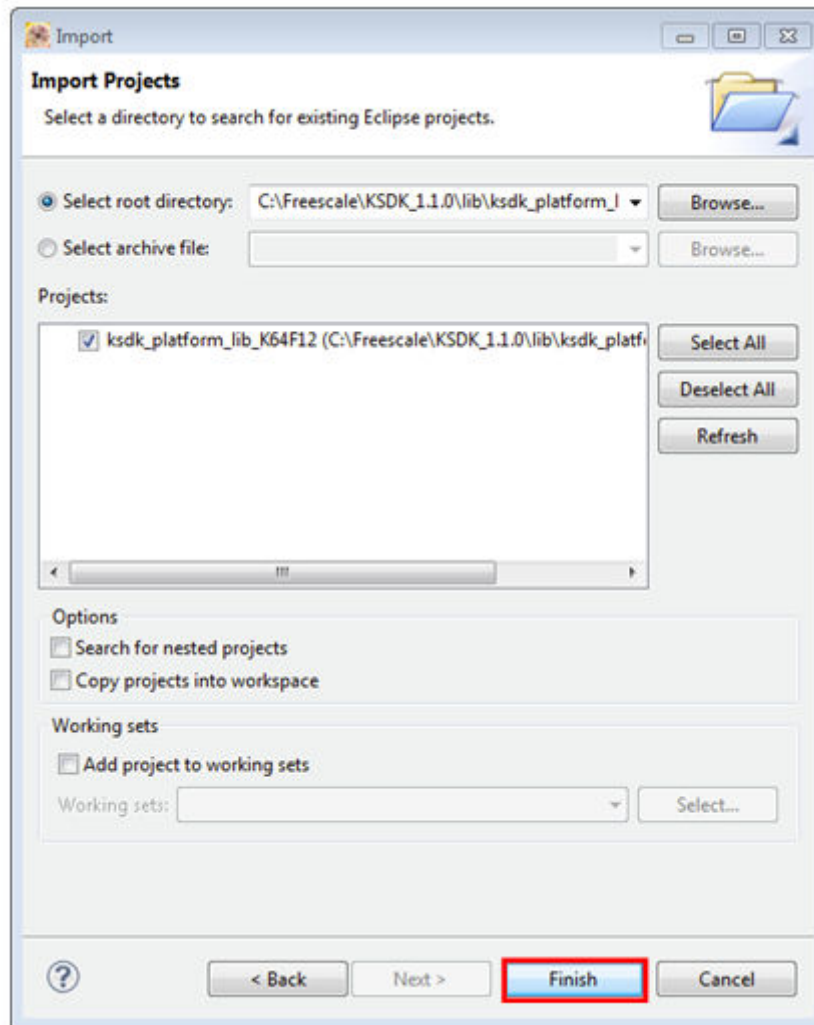


Figure 36. Select K64F12 platform library project

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.

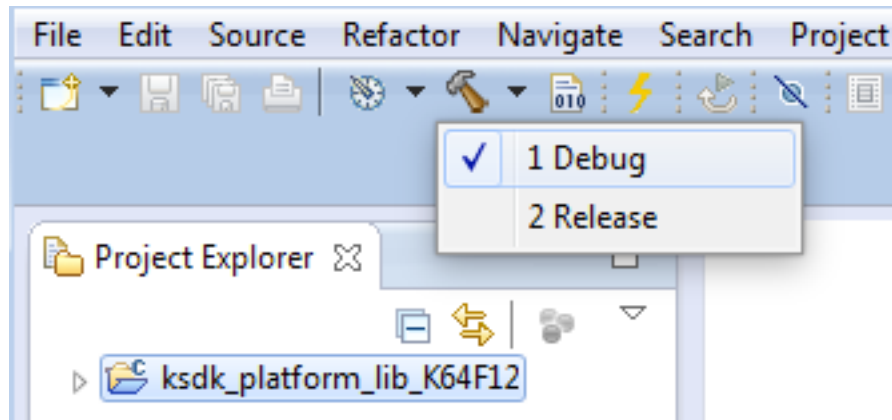


Figure 37. Selection of the build target in KDS IDE

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 Build a demo application

To build a demo application, repeat the steps listed in Section 5.3, using a demo application project instead of the platform library project. Demo application projects are located in this folder:

```
<install_dir>/examples/<board_name>/demo_apps/<demo_name>/kds
```

For this example, the path is:

```
<install_dir>/examples/frdmk64f/demo_apps/hello_world/kds
```

5.5 Run a demo 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/Windows-serial-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.

Run a demo using Kinetis Design Studio IDE

For the MRB-KW01 board, see www.freescale.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_BAUD variable in board.h file)
- b. No parity
- c. 8 data bits
- d. 1 stop bit

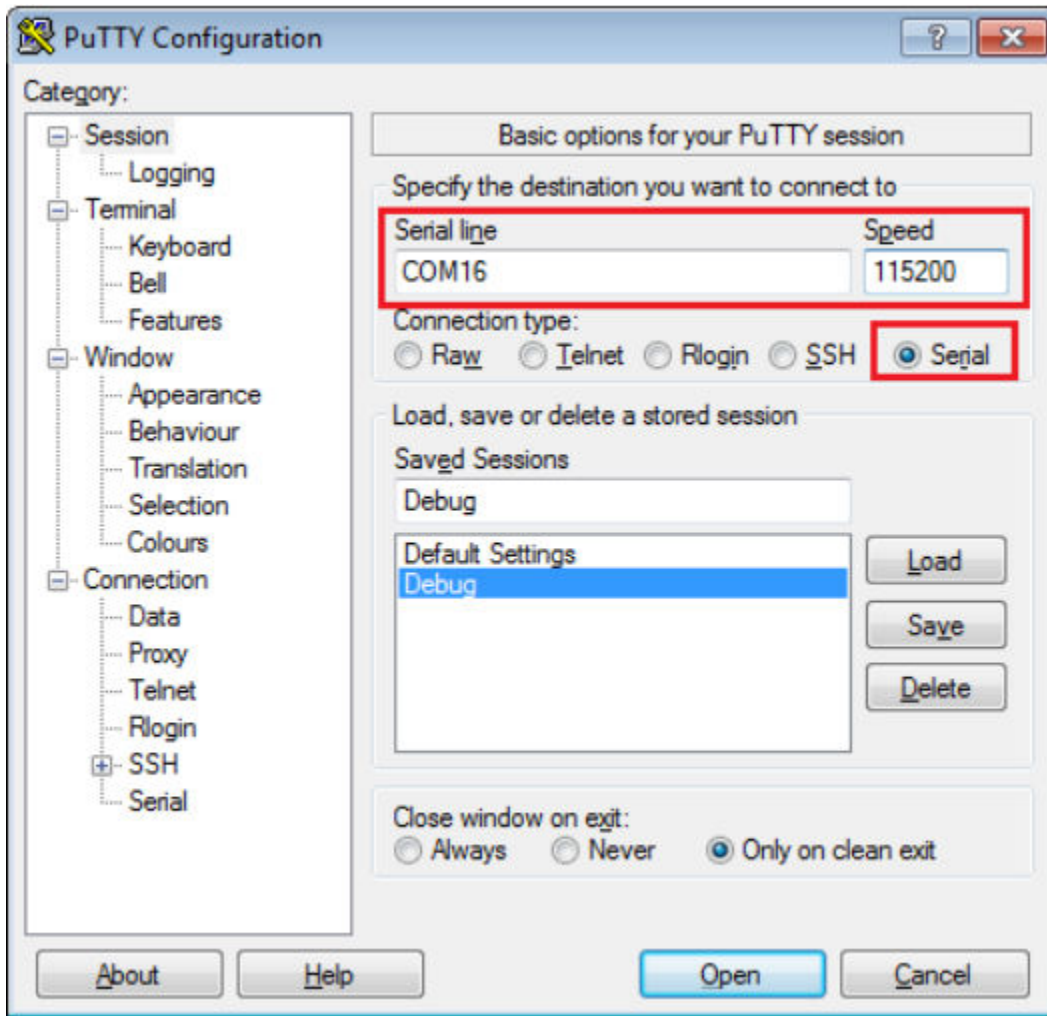


Figure 38. Terminal (PuTTY) configurations

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. 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".

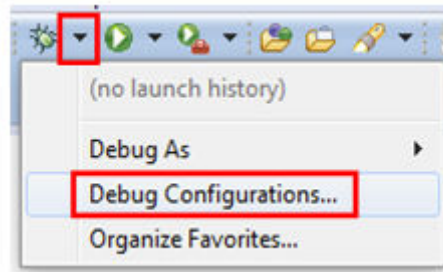


Figure 39. Debug Configurations dialog button

- 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.

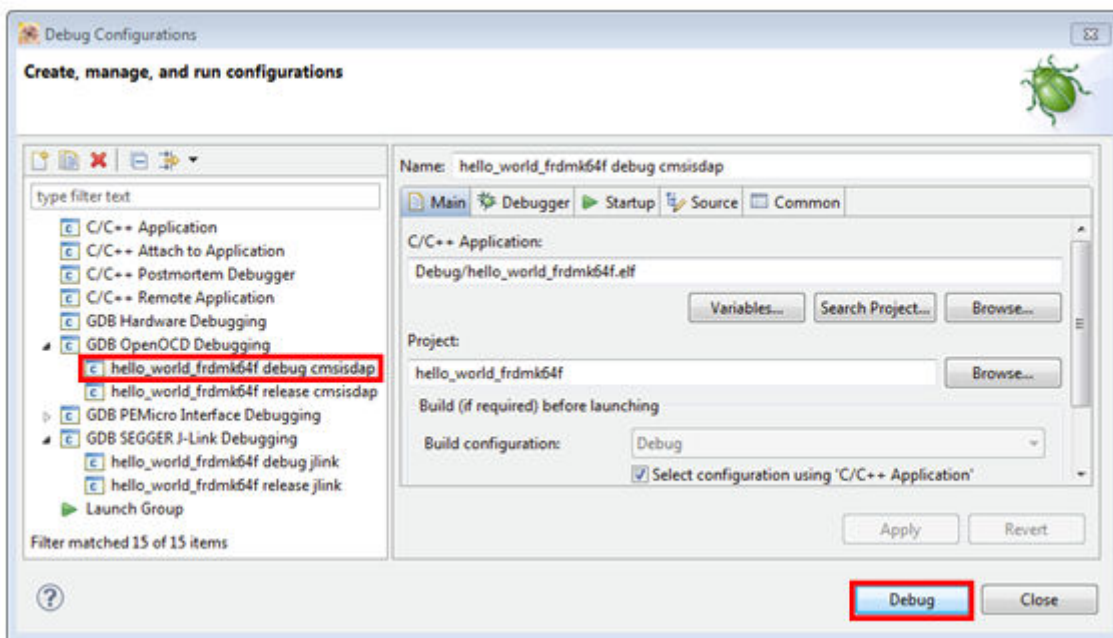


Figure 40. Selection of the debug configuration and debugger launch

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

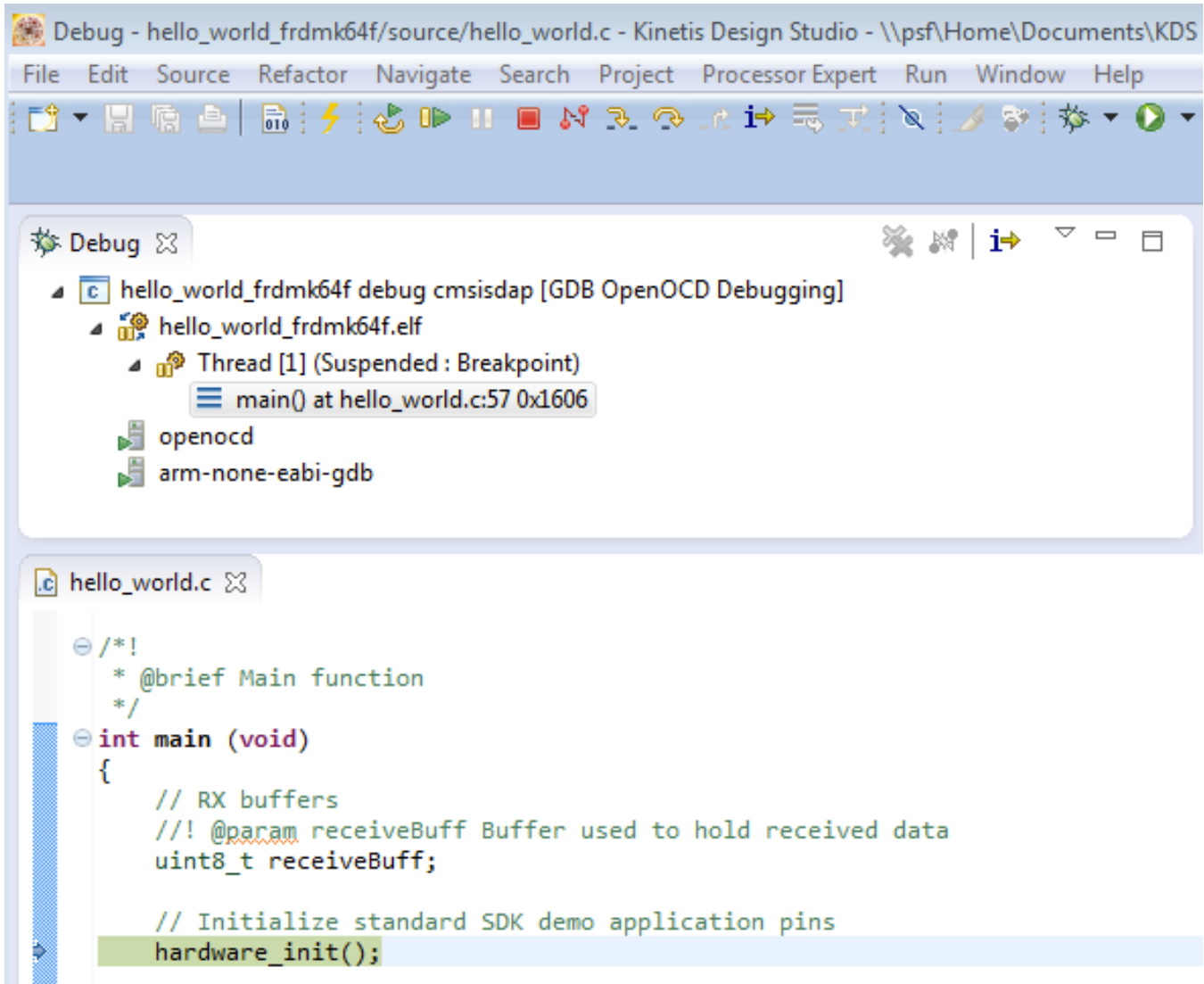


Figure 41. Stop at main() when running debugging

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

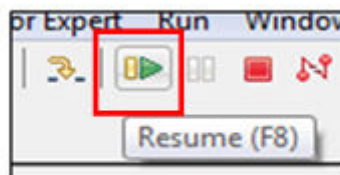


Figure 42. Resume button

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.

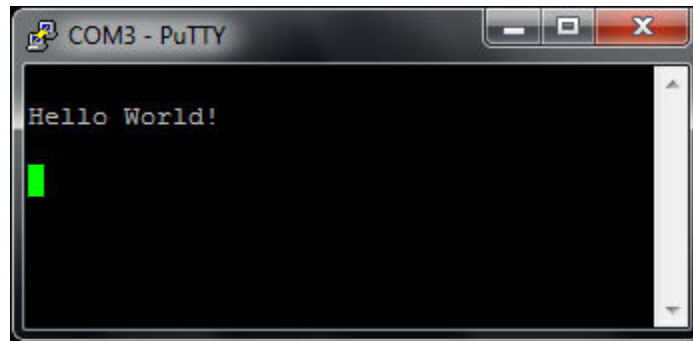


Figure 43. Text display of the hello_world demo

6 Run a demo using Atollic® TrueSTUDIO®

This section describes the steps to configure Atollic TrueSTUDIO 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 demo or example application in the KSDK.

NOTE

The Eclipse update required for the KDS IDE toolchain is not required for Atollic.

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 the platform library

These steps guide you through the process of opening and building the platform library project in TrueSTUDIO. The platform library is required by the demo and does not build without it.

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

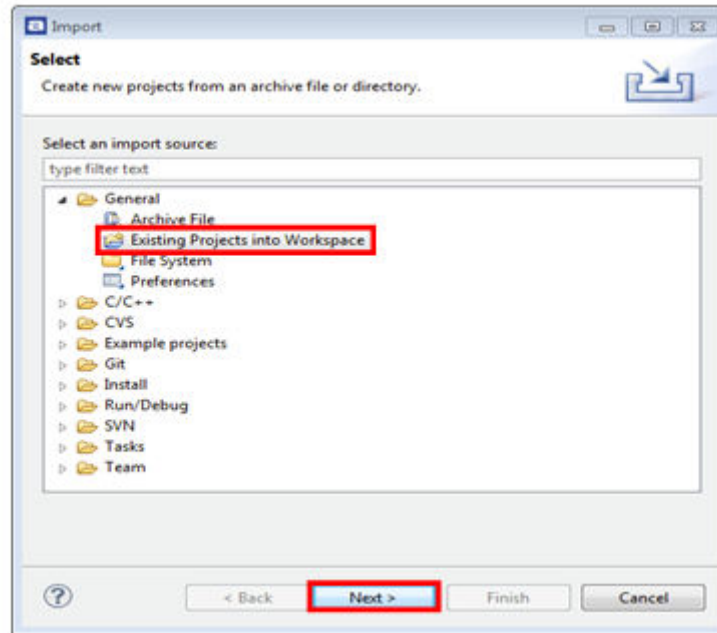


Figure 44. Selection of the correct import type in TrueSTUDIO

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

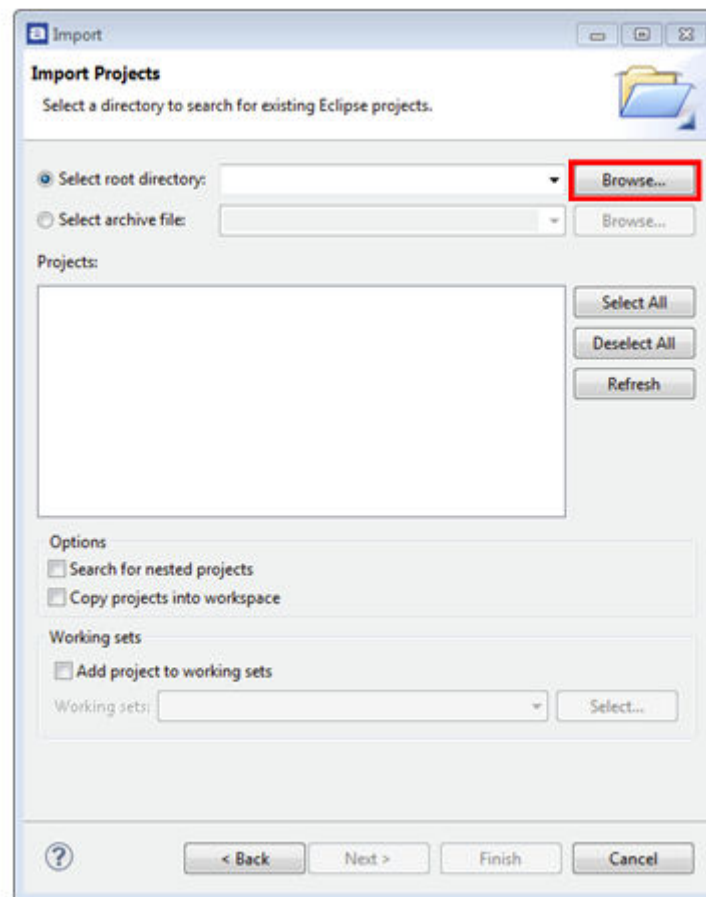


Figure 45. Projects directory selection window

Run a demo using Atollic® TrueSTUDIO®

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

`<install_dir>/lib/ksdk_platform_lib/atl/<device_name>`

For this example, the specific location is:

`<install_dir>/lib/ksdk_platform_lib/atl/K64F12`

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

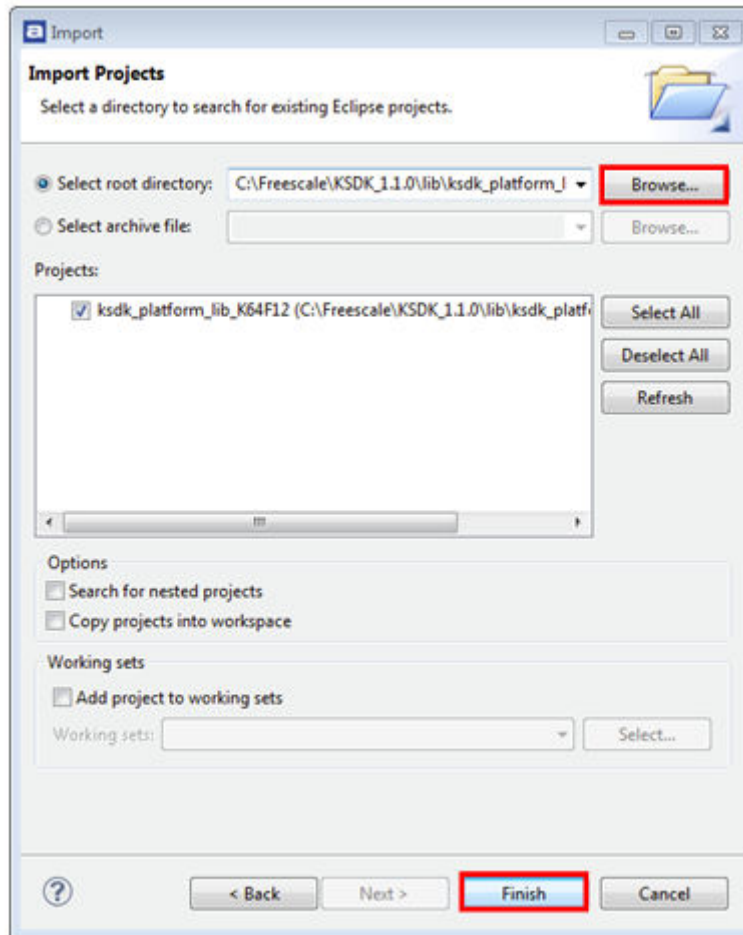


Figure 46. Select the K64F12 platform library project

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.

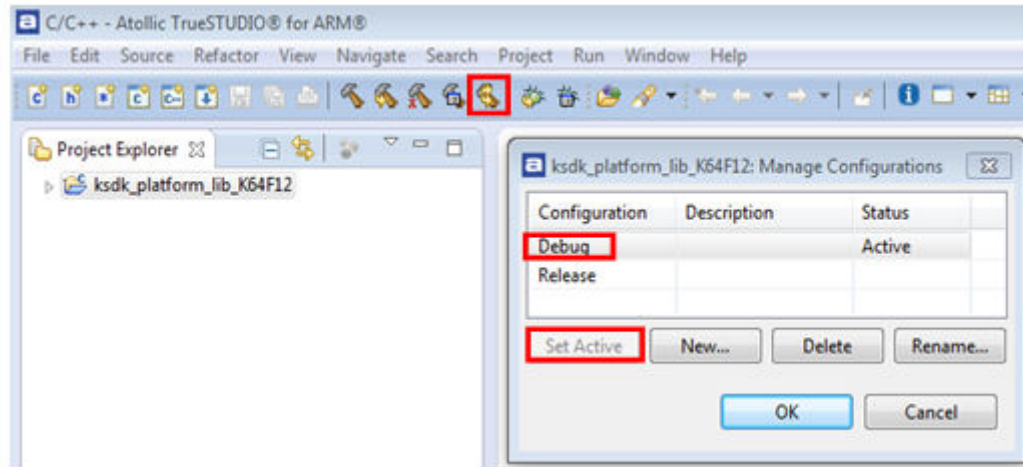


Figure 47. Selection of build target in TrueSTUDIO

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

6.3 Build a demo application

To build a demo application, repeat the steps listed in section 6.2, using a demo application project instead of the platform library project. Demo application projects are located in this folder:

`<install_dir>/examples/<board_name>/demo_apps/<demo name>/atl`

For this example, the path is:

`<install_dir>/examples/frdmk64f/demo_apps/hello_world/atl`

6.4 Run a demo 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:

Run a demo using Atollic® TrueSTUDIO®

- 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.

For the MRB-KW01 board, visit www.freescale.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 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

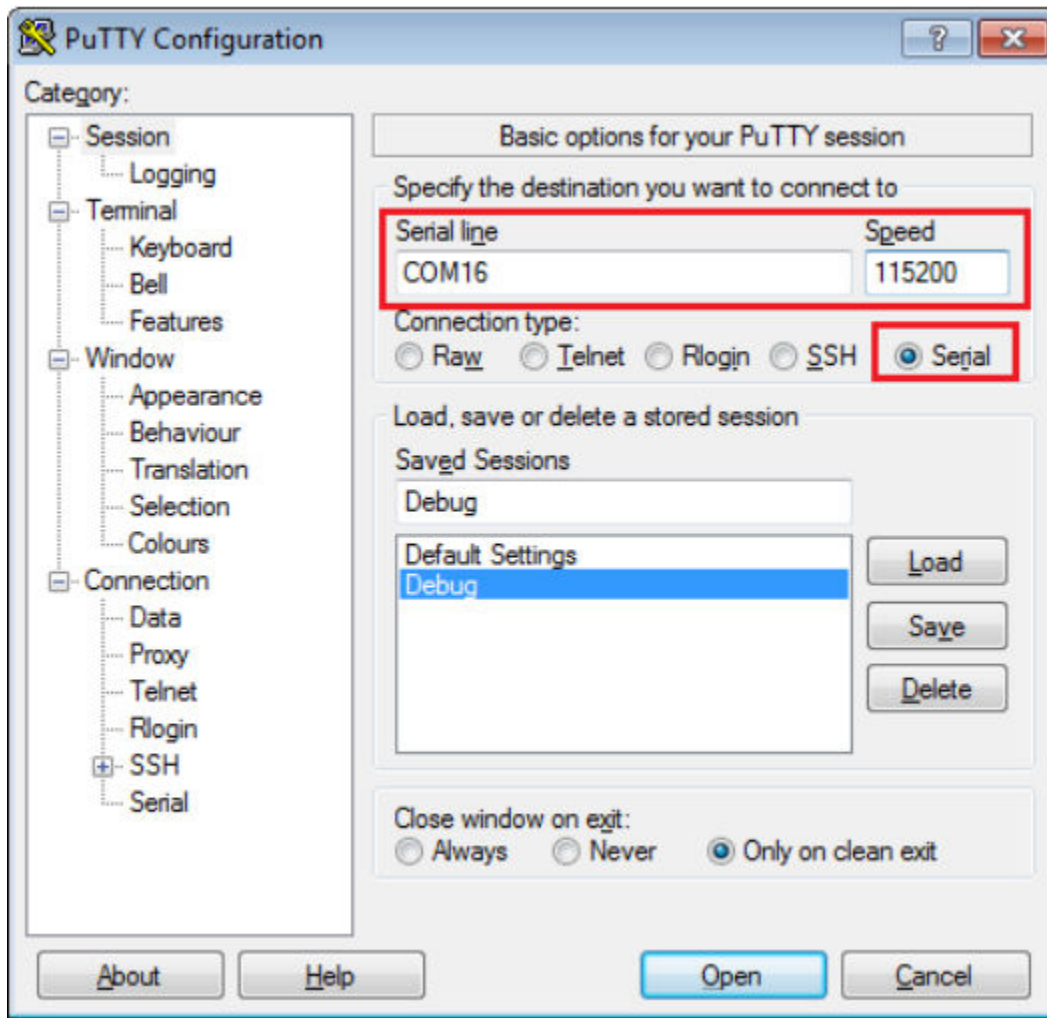


Figure 48. Terminal (PuTTY) configurations

3. 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 49. 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 Freescale Freedom development board platform .

- c. Select the J-Link “Debug” interface and click the “Debug” button.

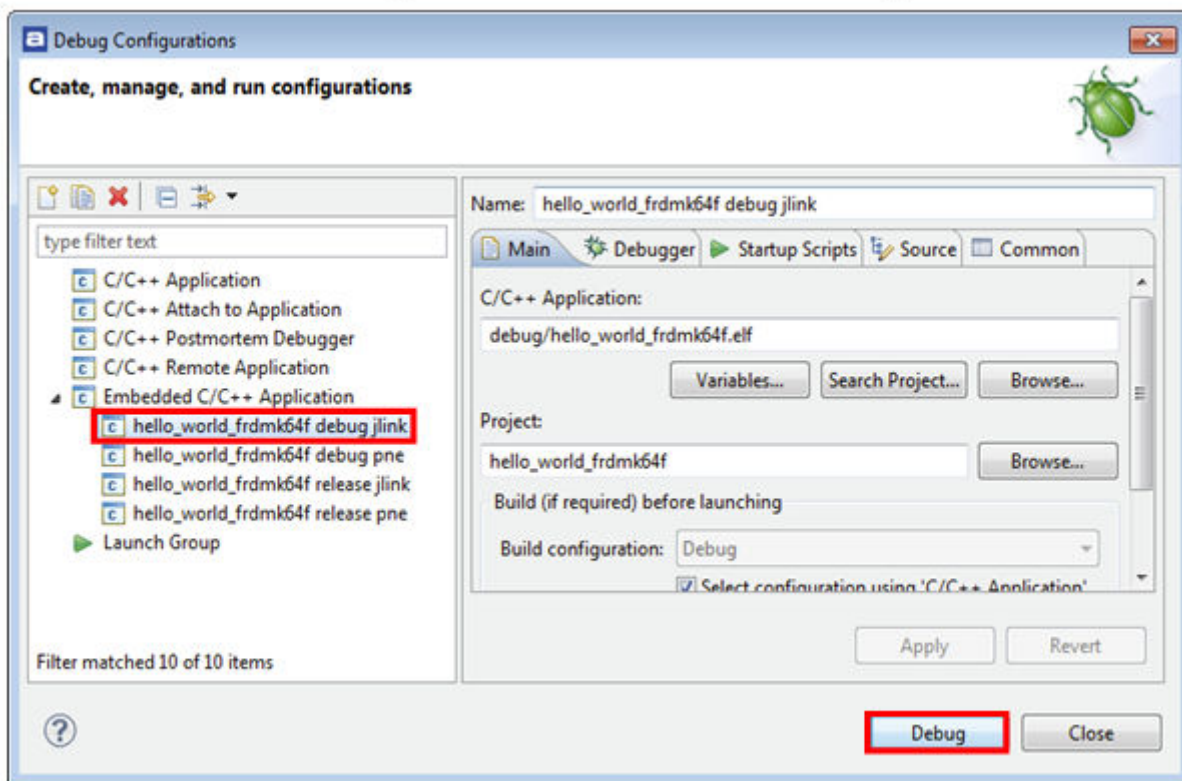


Figure 50. Selection of debug configuration in Debug Configuration dialog box

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

Run a demo using Atollic® TrueSTUDIO®

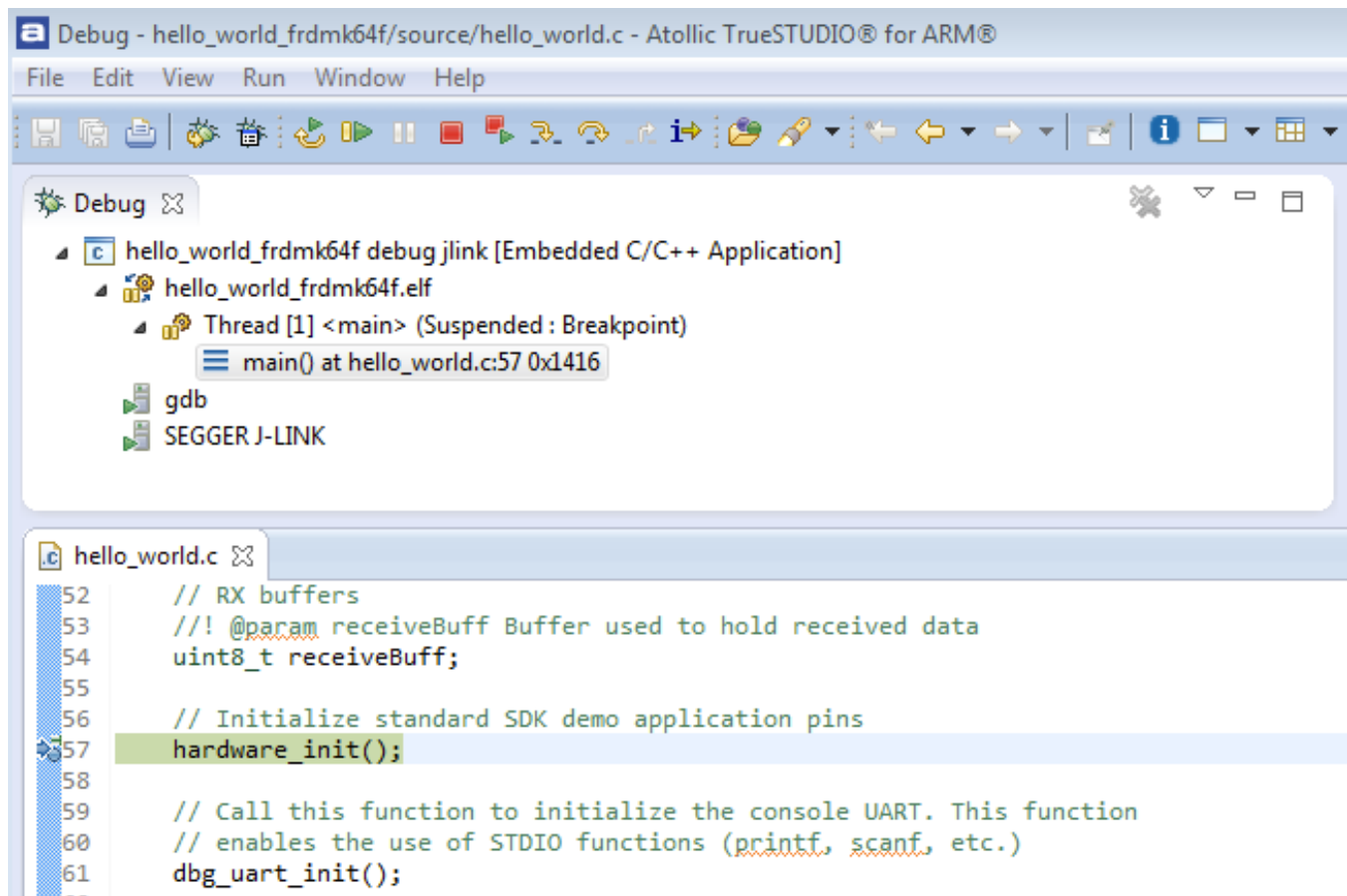


Figure 51. Stop at main() when running debugging

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



Figure 52. 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.

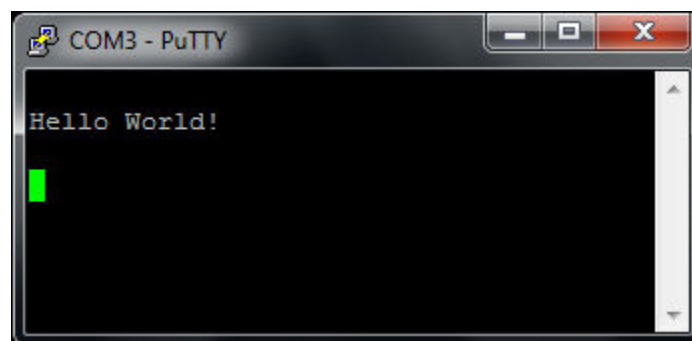


Figure 53. Text display of the hello_world demo

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.1.3.0 Release Notes*. (document KSDK130RN)

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.

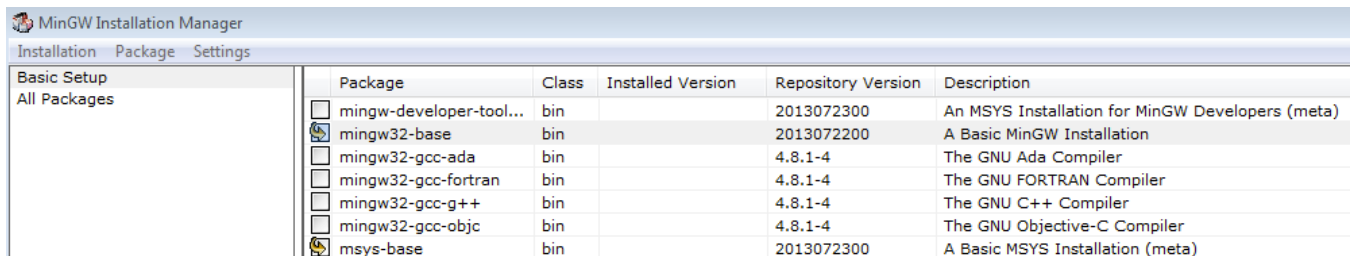


Figure 54. Setup MinGW and MSYS

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

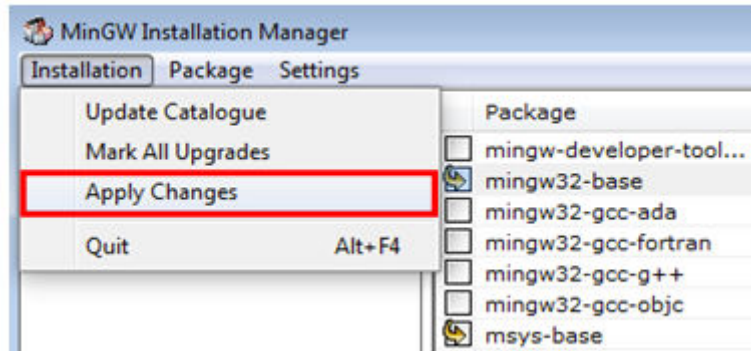


Figure 55. Complete MinGW and MSYS installation

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.

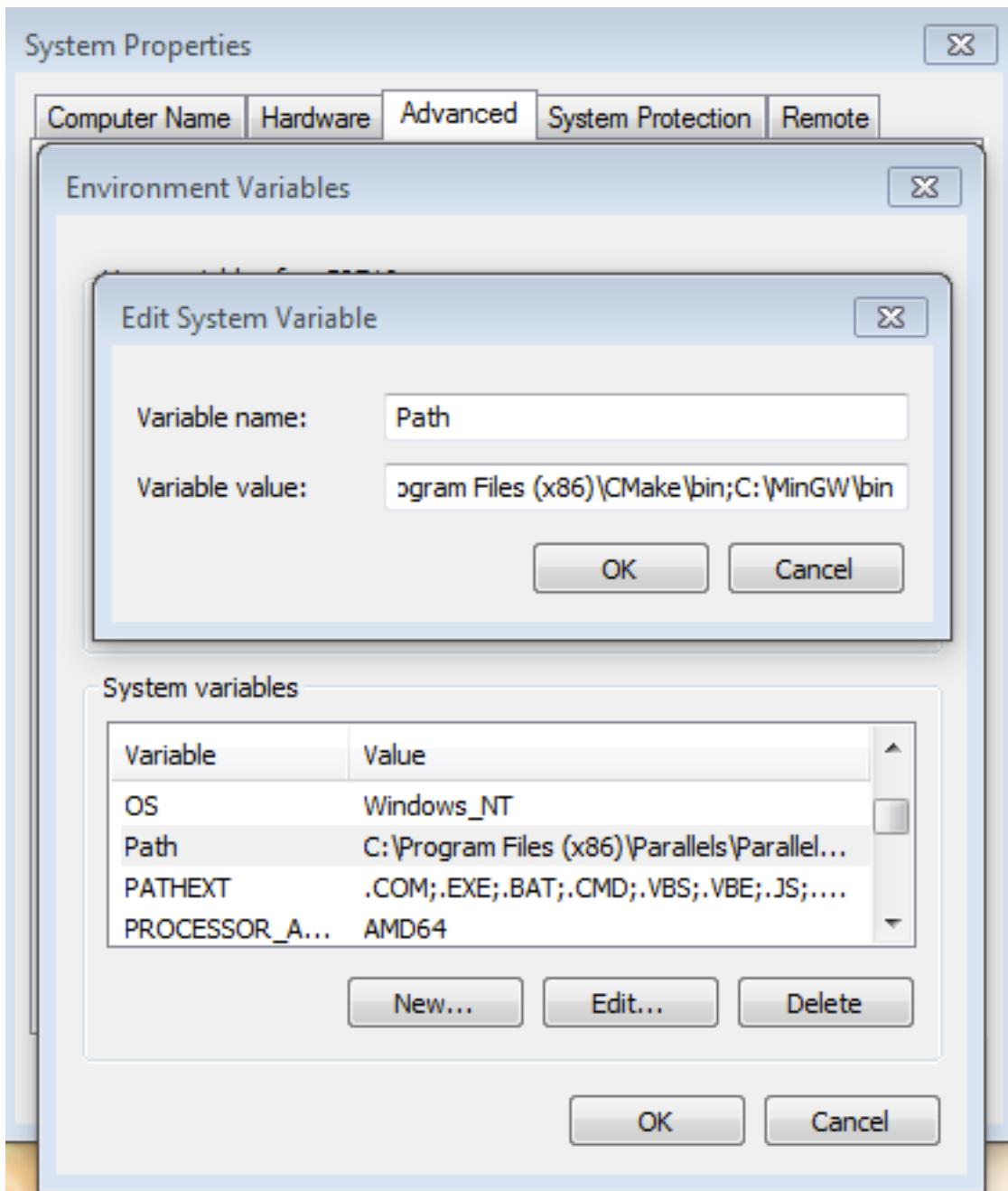


Figure 56. Add Path to systems environment

7.1.3 Add a new system environment 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\4.9 2015q1

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

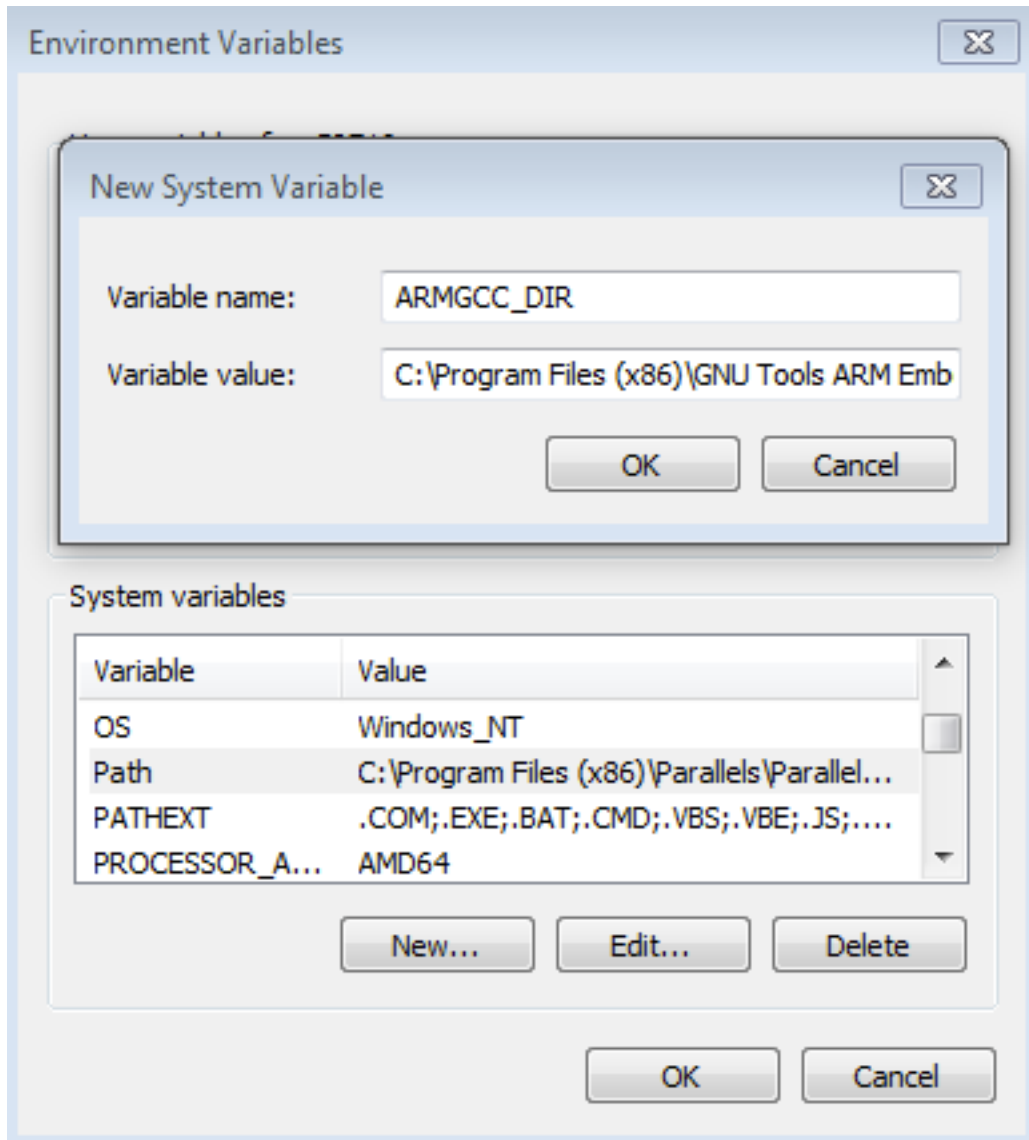


Figure 57. Add ARMGCC_DIR system variable

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. It's up to the user to select whether it's installed into the PATH for all users or just the current user. In this example, the assumption is that it's installed for all users.

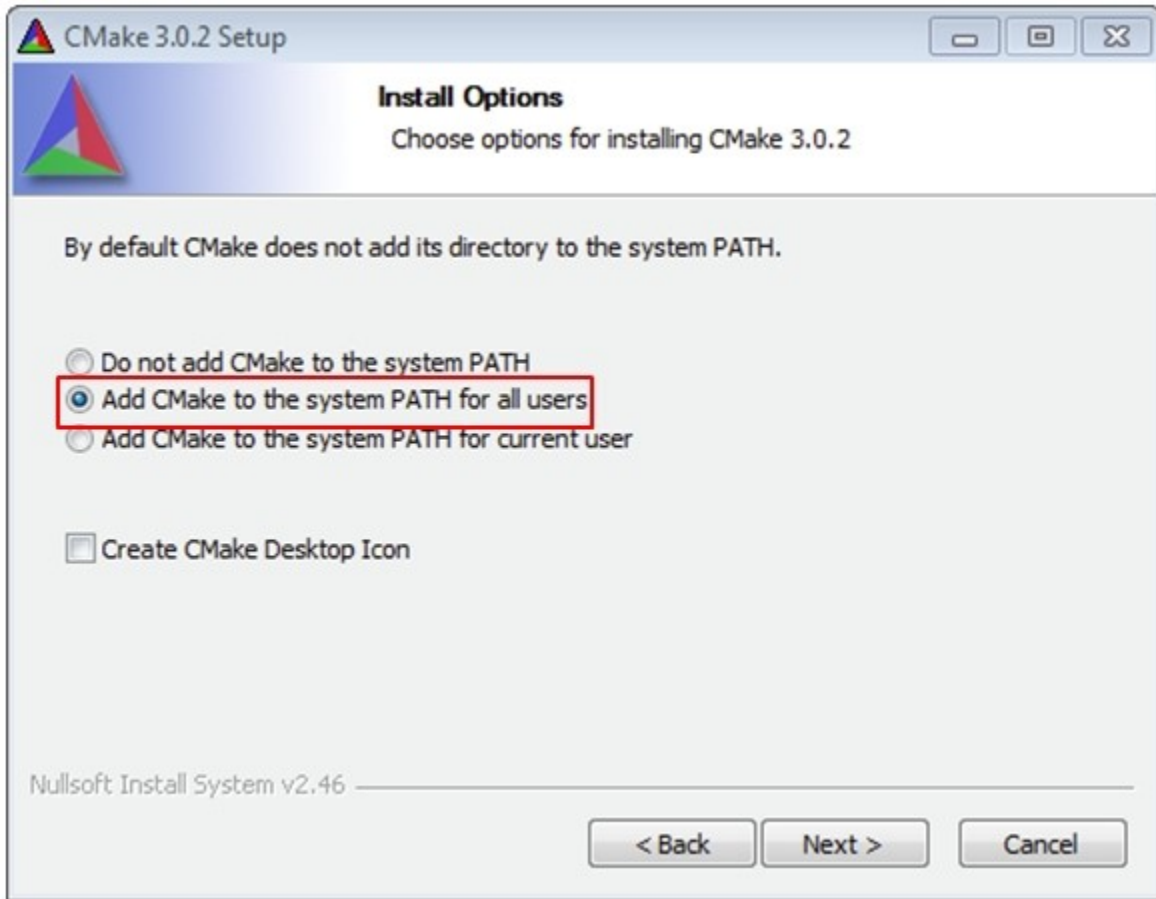


Figure 58. Install CMake

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 the platform library

To build the platform library, follow these instructions:

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”.
2. Change the directory of the command window to the platform library directory in the KSDK:

```
<install_dir>/lib/ksdk_platform_lib/armgcc/<device_name>/
```

3. 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.

There are batch files provided to build both configurations. For this example, the “Debug” target is built and “build_debug.bat” is typed on the command line. If the “Release” target is desired, type the “build_release.bat” instead.

Alternatively, if using the command line is not desired, double click on the batch files from Windows operating system Explorer.

```
Administrator: GCC Command Prompt
C:\Freescale\KSDK_1.2.0\lib\ksdk_platform_lib\armgcc\K64F12>build_debug.bat
```

Figure 59. Build debug version of platform library

- When the build finishes, the output looks like the image below.

```
[ 98%] [100%] Building C object CMakeFiles/KsdkPlatformLib.dir/C:/Freescale/KSDK_1.2.0/platform/drivers/src/uref/fsl_uref_common.c.obj
Building C object CMakeFiles/KsdkPlatformLib.dir/C:/Freescale/KSDK_1.2.0/platform/drivers/src/uref/fsl_uref_driver.c.obj
Linking C static library debug\libksdk_platform.a
[100%] Built target KsdkPlatformLib
C:\Freescale\KSDK_1.2.0\lib\ksdk_platform_lib\armgcc\K64F12>pause
Press any key to continue . . .
```

Figure 60. KSDK platform library build successful

- The library (libksdk_platform.a) is generated in one of these directories, according to the build target:

<install_dir>/lib/ksdk_platform_lib/armgcc/<device_name>/debug

<install_dir>/lib/ksdk_platform_lib/armgcc/<device_name>/release

7.3 Build a demo application

KSDK demo applications require that the platform library for the same build target (Debug or Release) is present. Follow the steps in Section 7.2 prior to attempting to build a demo application.

To build a demo application, follow these steps.

- 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”.
- Change the directory to the demo application project directory, which has a path like this:

<install_dir>/examples/<board_name>/demo_apps/<demo_name>/armgcc

For this example, the exact path is:

<install_dir>/examples/frdmk64f/demo_apps/hello_world/armgcc

- Type “build_debug.bat” on the command line or double click on the “build_debug.bat” file in Windows operating system Explorer to perform the build. The output is shown in this figure:

```

[100%] Building C object CMakeFiles/hello_world.dir/C:/Freescale/KSDK_1.2.0/exam
ples/frdmk64f/demo_apps/hello_world/fsl_lptmr_irq.c.obj
Linking C executable debug\hello_world.elf
[100%] Built target hello_world

C:\Freescale\KSDK_1.2.0\examples\frdmk64f\demo_apps\hello_world\armgcc>pause
Press any key to continue . . .

```

Figure 61. hello_world demo build successful

7.4 Run a demo 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.
 - You have a standalone J-Link pod that is connected to the debug interface of your board.

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

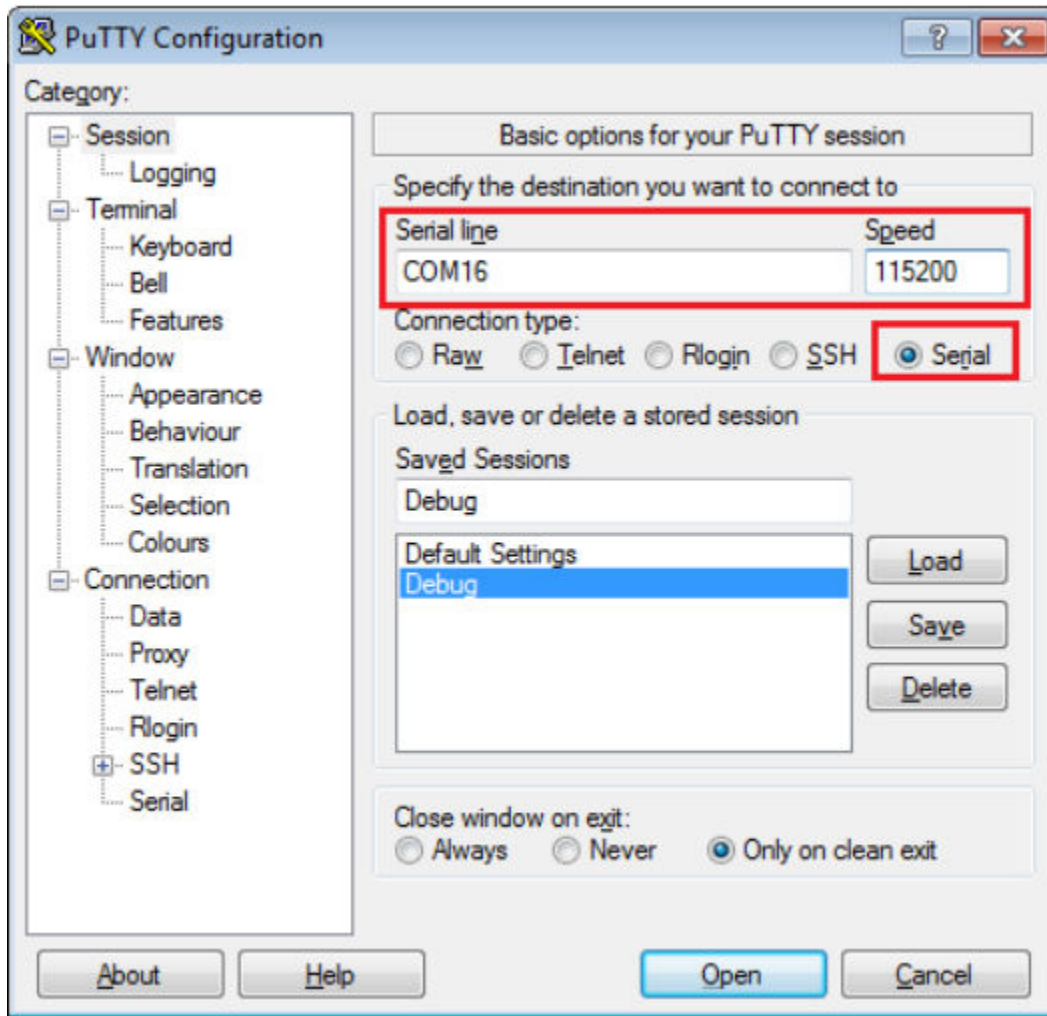


Figure 62. Terminal (PuTTY) configurations

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. MK64FN1M0xxx12 .

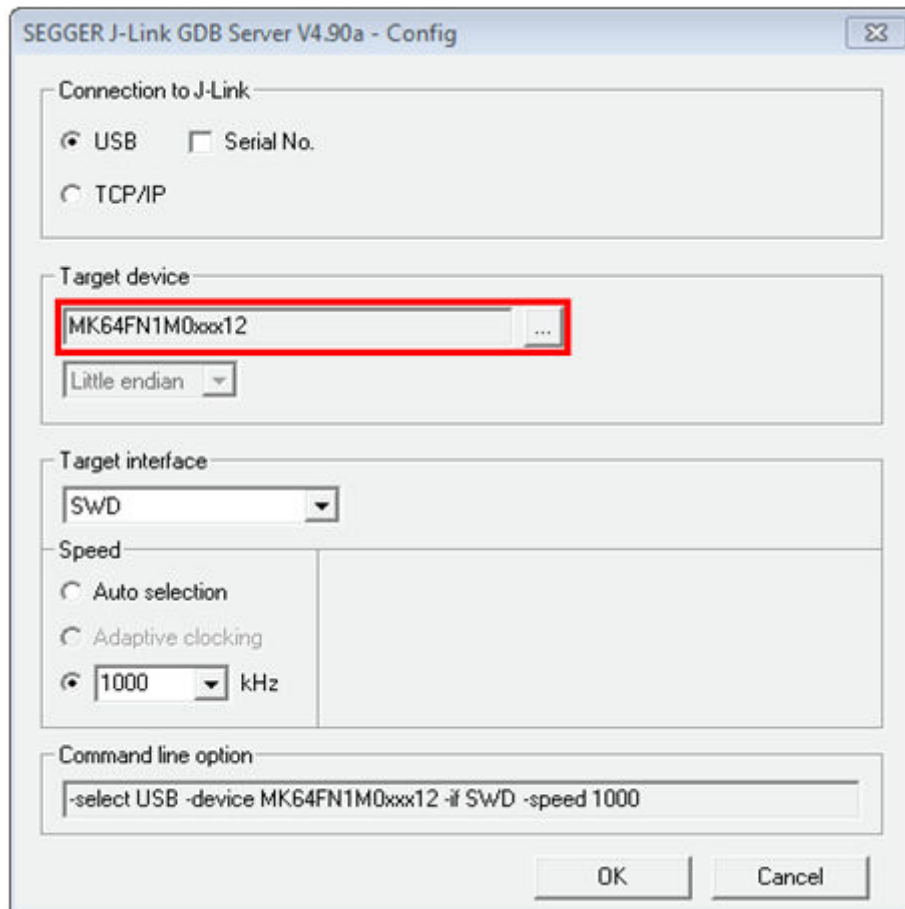


Figure 63. SEGGER J-Link GDB Server configuration

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

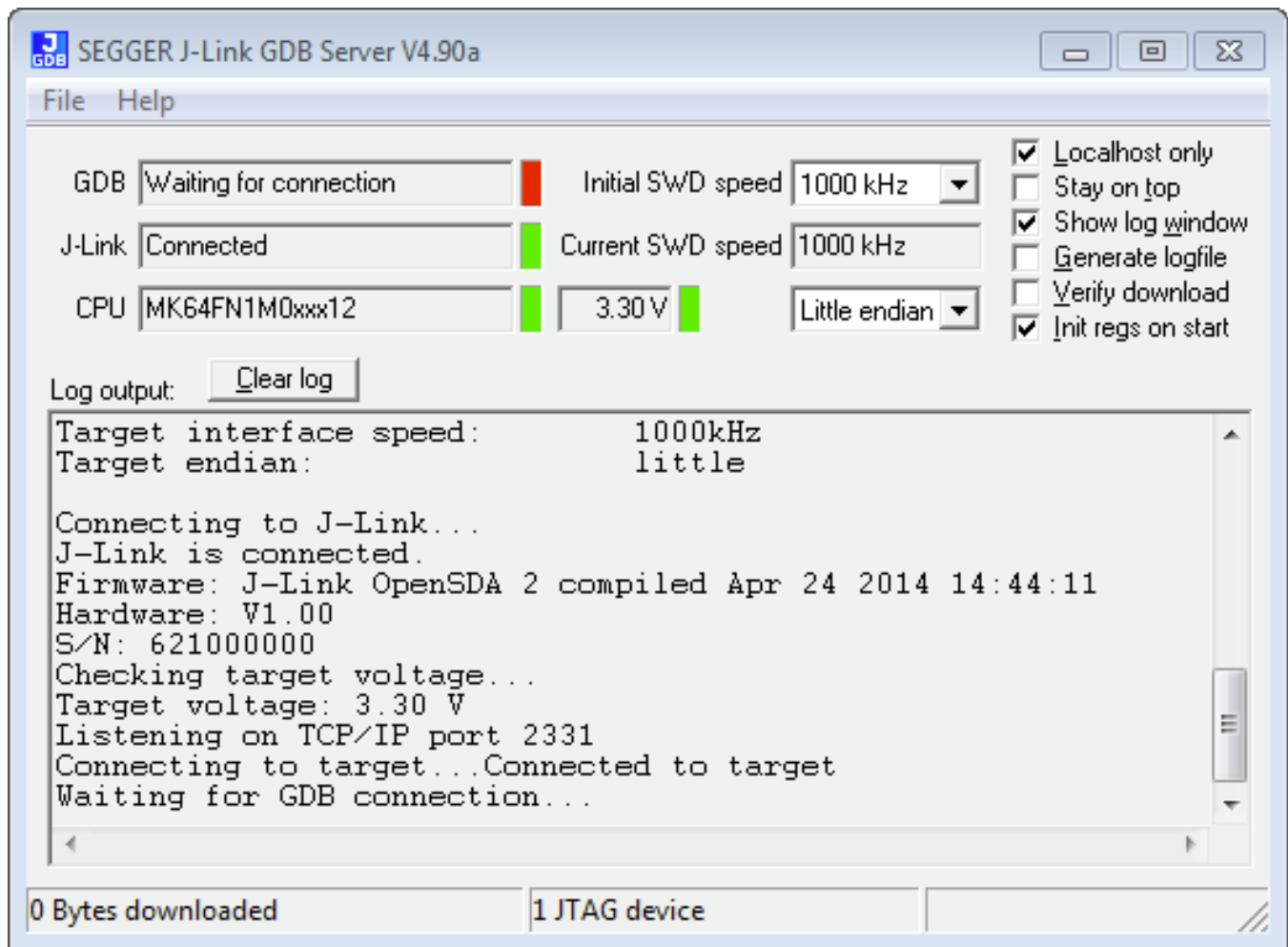


Figure 64. SEGGER J-Link GDB Server screen after successful connection

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

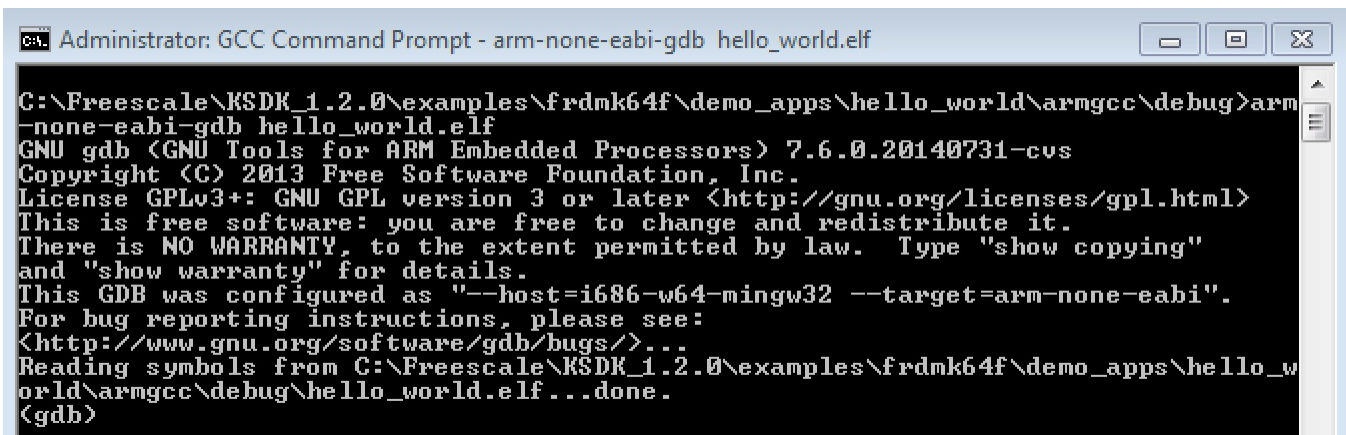
<install_dir>/examples/board_name/demo_apps/<demo_name>/armgcc/debug

<install_dir>/examples/board_name/demo_apps/<demo_name>/armgcc/frdmk64f/release

For this example, the path is:

<install_dir>/examples/frdmk64f/demo_apps/hello_world/armgcc/debug

8. Run the command “arm-none-eabi-gdb.exe <demo_name>.elf”. For this example, it is “arm-none-eabi-gdb.exe hello_world.elf”.



```

Administrator: GCC Command Prompt - arm-none-eabi-gdb hello_world.elf
C:\Freescall\KSDK_1.2.0\examples\frdmk64f\demo_apps\hello_world\armgcc\debug>arm-
none-eabi-gdb hello_world.elf
GNU gdb (GNU Tools for ARM Embedded Processors) 7.6.0.20140731-cvs
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from C:\Freescall\KSDK_1.2.0\examples\frdmk64f\demo_apps\hello_w
orld\armgcc\debug\hello_world.elf...done.
(gdb)

```

Figure 65. 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.

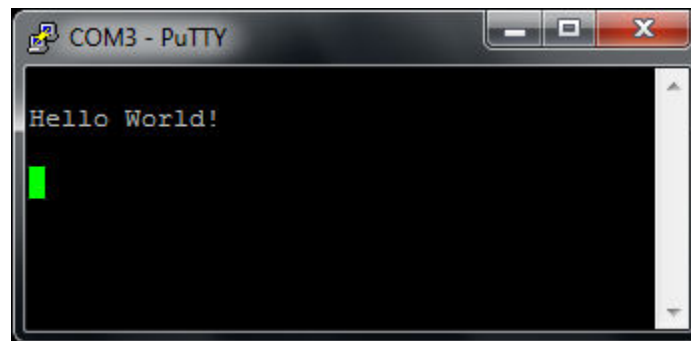


Figure 66. Text display of the hello_world demo

8 Appendix A - How to determine COM port

This section describes the steps necessary to determine the debug COM port number of your Freescale hardware development platform. All Freescale 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:

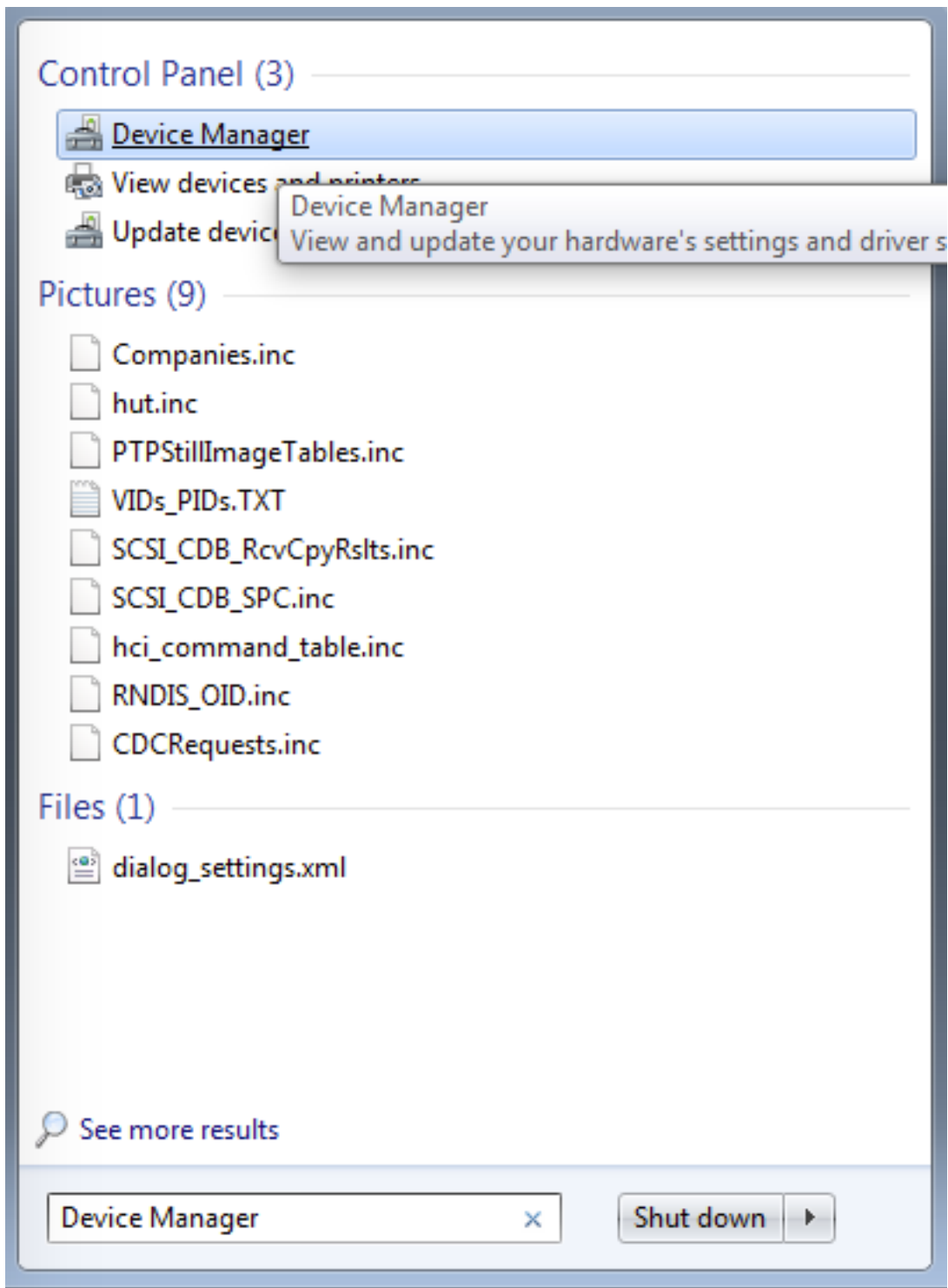


Figure 67. Device manager

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

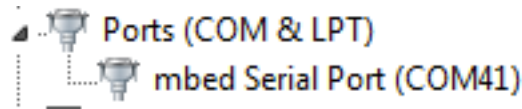


Figure 68. OpenSDA – CMSIS-DAP/mbed/DAPLink interface

b. OpenSDA – P&E Micro:

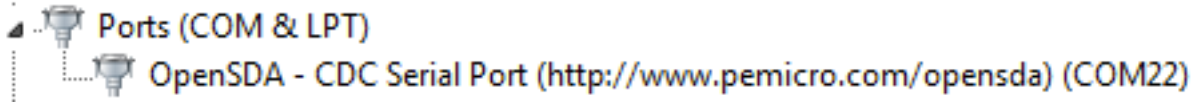


Figure 69. OpenSDA – P&E Micro

c. OpenSDA – J-Link:

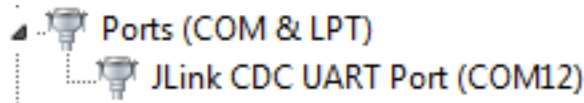


Figure 70. OpenSDA – J-Link

d. P&E Micro OSJTAG:

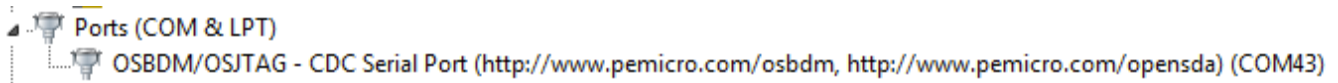


Figure 71. P&E Micro OSJTAG

e. MRB-KW01:

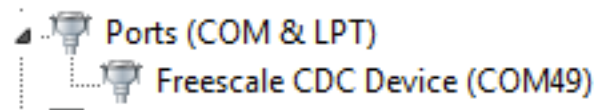


Figure 72. MRB-KW01

9 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 Freescale hardware platforms support the configurable OpenSDA standard.

Table 1. Hardware platforms supported by KSDK

Hardware platform	Default interface	OpenSDA details
FRDM-K22F	CMSIS-DAP\mbed\DAPLink	OpenSDA v2.1
FRDM-K64F	CMSIS-DAP\mbed\DAPLink	OpenSDA v2.0
FRDM-K82F	CMSIS-DAP	OpenSDA v2.1
FRDM-KL02Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL03Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL25Z	P&E Micro OpenSDA	OpenSDA v1.0

Table continues on the next page...

Table 1. Hardware platforms supported by KSDK (continued)

FRDM-KL26Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL27Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL43Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL46Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KV10Z	CMSIS-DAP	OpenSDA v2.1
FRDM-KV31F	CMSIS-DAP	OpenSDA v2.1
FRDM-KW24	CMSIS-DAP\mbed\DAPLink	OpenSDA v2.1
FRDM-KW40Z	CMSIS-DAP	OpenSDA v2.1
MRB-KW01	N/A External Probe Required	N/A
TWR-K21D50M	P&E Micro OSJTAG	N/A
TWR-K21F120M	P&E Micro OSJTAG	N/A
TWR-K22F120M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-K24F120M	CMSIS-DAP\mbed	OpenSDA v2.1
TWR-K60D100M	P&E Micro OSJTAG	N/A
TWR-K64F120M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-K65F180M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-K80F150M	CMSIS-DAP	OpenSDA v2.1
TWR-KL43Z48M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KM34Z75M	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-KW24D512	P&E Micro OpenSDA	OpenSDA v1.0
USB-KW24D512	N/A External Probe	N/A
USB-KW40Z	N/A External Probe	N/A

10 Appendix C - Updating OpenSDA firmware

Any Freescale 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:

- **J-Link:** 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.

- CMSIS-DAP/mbed/DAPLink: 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.
- [P&E Micro](http://www.pemicro.com): 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".

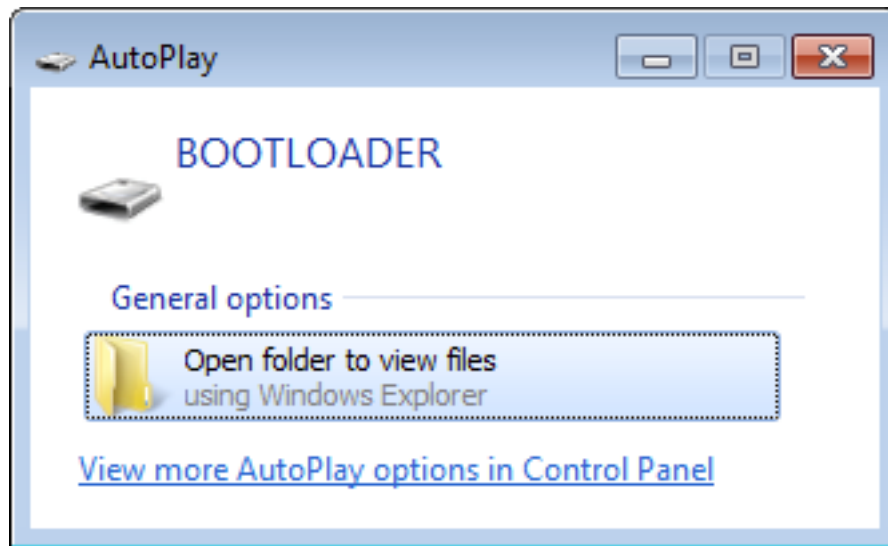


Figure 73. BOOTLOADER drive

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 OS 7 or an earlier version to 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.

11 Revision History

This table summarizes revisions to this document.

Table 2. Revision History

Revision number	Date	Substantive changes
0	09/2015	Initial release

How to Reach Us:

Home Page:

freescale.com

Web Support:

freescale.com/support

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.

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: freescale.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, μ Vision, 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.

© 2015 Freescale Semiconductor, Inc.

Document Number KSDK13GSUG
Revision 0, 09/2015

