Freescale Semiconductor, Inc.
User's Guide

Document Number: KSDKGSUG
Rev. 0, 10/2014

Getting Started with Kinetis Software Development Kit (KSDK)

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. 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, USB stack and other middleware to support rapid development on supported Kinetis devices. The image below highlights the layers and features of the KSDK.

Contents

Confidentiality statement, as appropriate to document/part status.

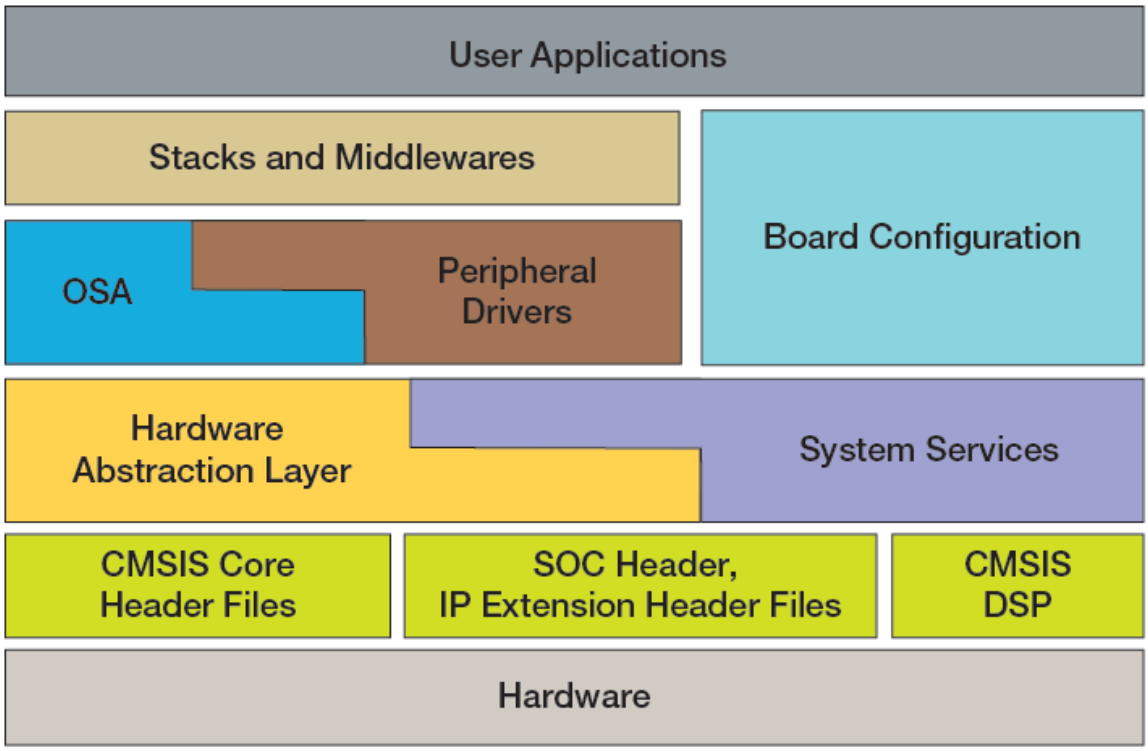

Figure 1: KSDK layers

2 KSDK Demo Applications

Kinetis SDK (KSDK) is a Software Development Kit that provides comprehensive soThis section describes how the demo applications belong to and interact with other components of the KSDK. To get a comprehensive understanding of the KSDK components and folder structure, see the *Kinetis SDK API Reference Manual* (document KSDKAPIRM).

Freescale Confidential Proprietary, Nondisclosure Agreement Required

Demo applications are in the top-level demos folder of the KSDK tree. Within the demos folder is a list of all demo applications provided as part of the KSDK. Each demo supports multiple target platforms and shares demo application source code. When opening a demo folder (the hello_world demo is used as an example), this structure is observed:

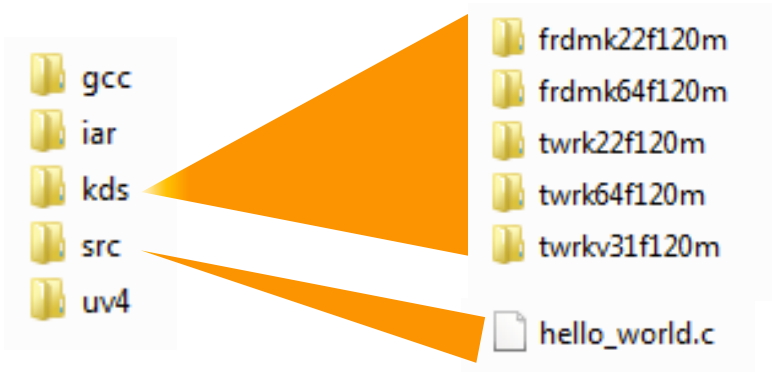

Figure 2: Demo folder structure

The toolchain folders contain directories for each supported hardware platform. Each hardware platform folder contains the necessary project and/or makefiles to build the demo using the corresponding toolchain. The src folder contains the source code for the demo application only and does not include startup, pin mux, or KSDK HAL/driver source code.

2.1 Locating demo application source files

When opening a demo 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 four main areas of the KSDK tree used to provide the full source code for each demo application:

- **Demo src folder: Contains demo-specific source files.**
- **Top-level KSDK boards folder: Contains board-specific pin mux configuration, GPIO pin definitions for KSDK drivers, and a reference configuration file for Freescale's Processor Expert tool.**
- **Top-level KSDK platform folder: Contains shared, SoC-specific linker files, startup code and source for KSDK HAL, peripheral drivers, and system services.**
- **Top-level lib folder: Contains** the compiled library file of the KSDK *platform* components such as HAL, peripheral drivers, startup code and system services.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

2.2 KSDK board folder

The KSDK includes board support files for each supported hardware platform. These files are located in *<install_dir>/boards*. The directory structure is as follows:

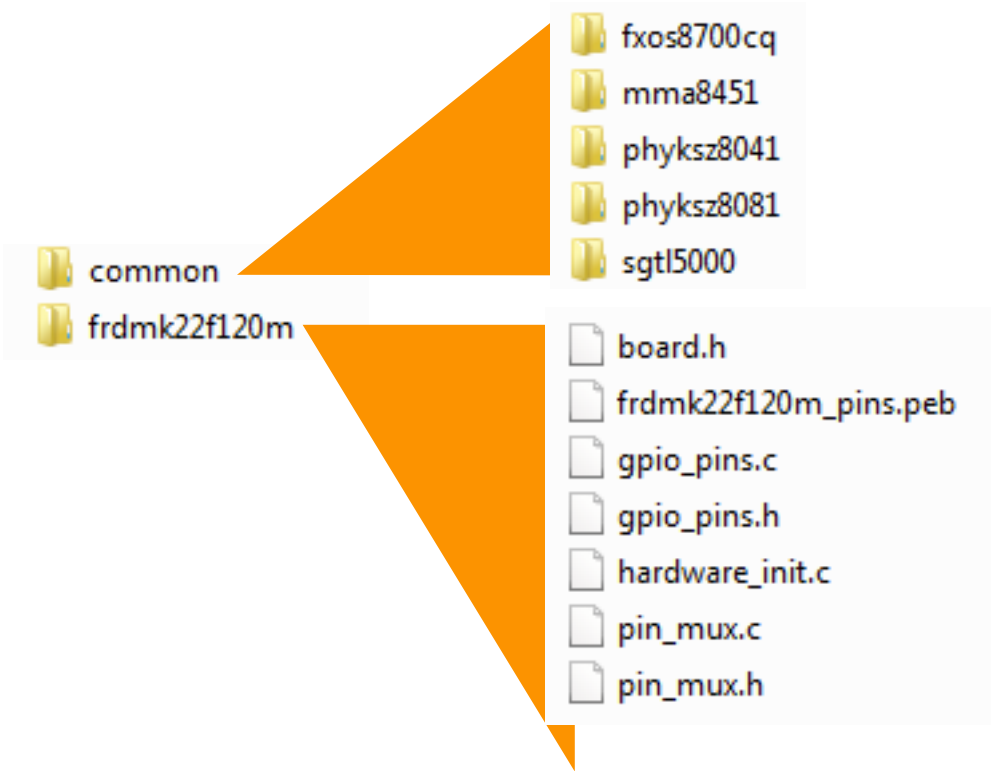

Figure 3: Boards folder

The common folder contains drivers for external peripheral devices that may be present on the KSDK-supported hardware platforms. Examples of such devices include audio codecs, accelerometers, etc.

Within each board/platform folder there is a common set of files used by the demos. All of these files can be generated automatically using Freescale's Processor Expert tool or modified by hand. All board support files provided as part of the KSDK are generated using Processor Expert and the reference *.peb file. These files are:

- **board.h**: Contains platform-specific configuration macros for thing such as debug terminal configuration, push buttons, LEDs and other board-specific items.
- **Processor Expert PEB file**: Reference file for Freescale's Processor Expert tool for the specific hardware platform.
- **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 INT pins for external sensors, for example.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

- **hardware_init.c**: Contains the hardware_init() function called by the main() loop of the demo application.
- **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.

2.3 KSDK platform folder

The platform folder is arguably 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.

2.4 KSDK lib folder

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

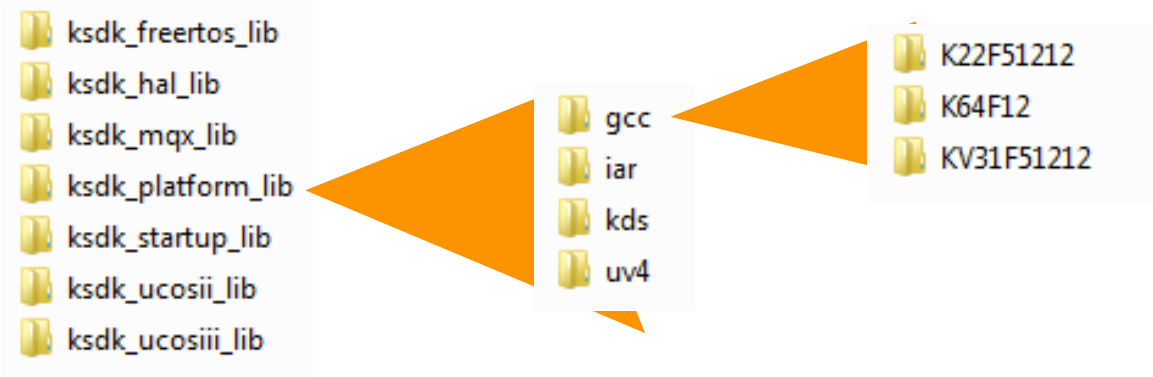

Figure 4: Lib folder

As with the demo applications, each library configuration in the lib folder contains a folder for each supported toolchain. Each toolchain contains a folder for a specific SoC. 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.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

3 Kinetis SDK

Kinetis SDK is a Software Development Kit that provides comprehensive software support for the core and peripherals of Freescale Kinetis devices. The KSDK includes a Hardware Abstraction Layer (HAL) for each peripheral and peripheral drivers built on the HAL. Example applications are provided to demonstrate driver and HAL usage and highlight the main features of targeted SoCs. Kinetis SDK also includes the latest available RTOS kernels and USB stacks for use on supported evaluation boards.

3.1 Toolchain requirement

- IAR embedded Workbenchversion7.20.1or later is required.
- ARM® GCC 4.8.3 2014q1 or later
- Keil MDK 5.11 or later
- Kinetis Design Studio (KDS) IDE v 1.0.2 or later
- TrueSTUDIO_for_ARM_Pro_win32_v5.1.1_20140820-1543

4 Build and run the KSDK demo applications using IAR

This section describes the steps required to configure IAR Embedded Workbench to build, run, and debug demo applications and necessary driver libraries provided in the Freescale KSDK. The Hello World demo application targeted for the TWR-K60D100M Tower System hardware platform is used as an example.

4.1 Building the platform driver library

Before building and debugging demo applications in KSDK, the driver library archive, platform_lib.a, must be built. This library contains all HAL and peripheral driver functions. Since the library archive is device specific, each device has its own library. To build the library for a device, follow these instructions:

1. Open the platform driver library workspace located in this folder:

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

   The workspace file is named ksdk_platform_lib.eww:

   For the K60D10, the correct folder location is as follows:

   <install_dir>/lib/ksdk_platform_lib/iar/K60D10/ksdk_platform_lib.eww

In the IAR Embedded Workbench project file and other tool chains, two compiler/linker configurations (build “targets”) are supported:

Freescale Confidential Proprietary, Nondisclosure Agreement Required

- Debug - The compiler optimization is set to low. The debug information is generated for the binary. This target should be used for developing and debugging.
- Release - The complier optimization is set to high. The debug information is not generated. This target should be used for final application release.

2. Choose the appropriate build target: “Debug” or “Release”.

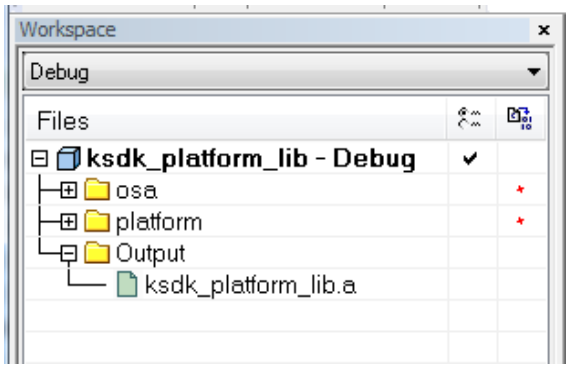

Figure 5: Debug/release

3. Click the “Make” button (highlighted by a red rectangle in this figure):

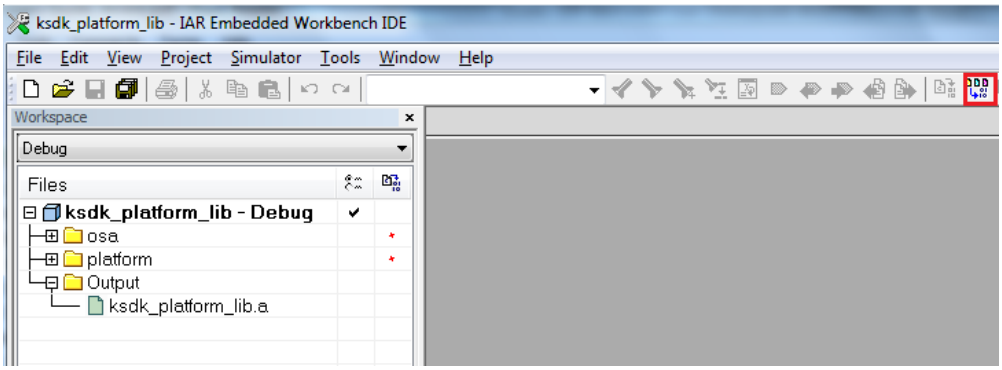

Figure 6: Build platform driver library

When the build is complete, the library (ksdk_platform_lib.a) is generated in this directory according to the build target:

<install_dir>/lib/ksdk_platform_lib<toolchain>/<device_name>/output/Debug

<install_dir>/lib/ksdk_platform_lib/<toolchain>/<device_name>/output/Release

Freescale Confidential Proprietary, Nondisclosure Agreement Required

4.2 Build a demo application

The KSDK demo applications utilize a library archive to compile in the necessary functions. Therefore, this library must be built before compiling and downloading. To check whether the library exists, verify that the ksdk_platform_lib.a file is located in:

<install_dir>/lib/ksdk_platform_lib/<toolchain>/<device_name>/<build>

The <build> is the desired build and can be either Debug or Release. For example, if the desired project is the Debug version of the hello_world demo application, there must be a ksdk_platform_lib.a library in this folder:

<install_dir>/lib/ksd_platform_lib/iar/K60D10/debug

If the appropriate library archive is not present, follow the steps in Section 3.1 to generate the necessary library archive. Otherwise, continue with the steps to build the desired demo application.

1. Open the desired demo application. Demo applications workspace files are located in this folder:

   <install_dir>/demos/<demo_name>/<toolchain>/<board_name>/<demo_name>.eww

   Using the TWR-K60D100M Tower System board as an example, the hello_world IAR workspace is located in this folder:

   <install_dir>/demos/hello_world/iar/twrk60d100m/hello_world.eww

2. To build a demo application project, click the “Make” button (highlighted by a red square in this figure)

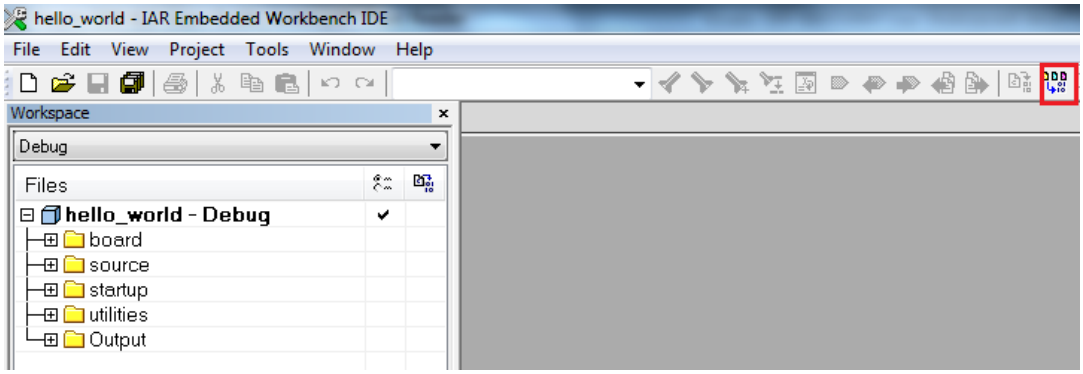

Figure 7: Build hello_world demo application

Freescale Confidential Proprietary, Nondisclosure Agreement Required

3. When the build is complete, IAR shows this information in the Build window:

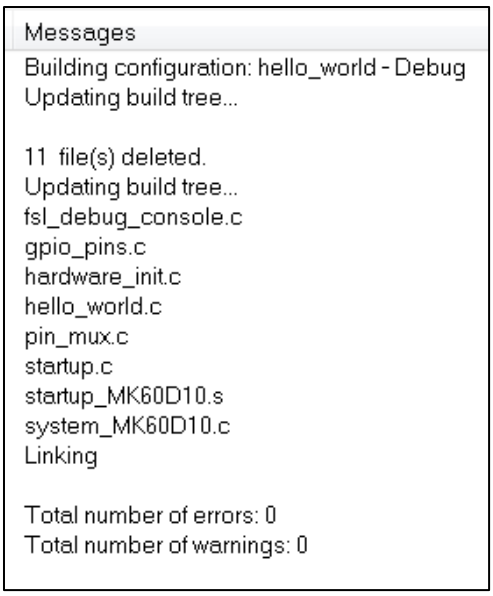

Figure 8: Build hello_world demo successfully

4.3 Run a demo application

To download and run the application, perform these steps:

1. Connect the development platform to your PC via USB cable between the OpenSDA USB connector and the PC USB connector.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

2. Open the terminal application on the PC, such as PuTTY or Teraterm, and connect to the OSJTAG serial port number. Configure the terminal with these settings:
    a. 115200 baud rate
    b. No parity
    c. 8 data bits
    d. 1 stop bit

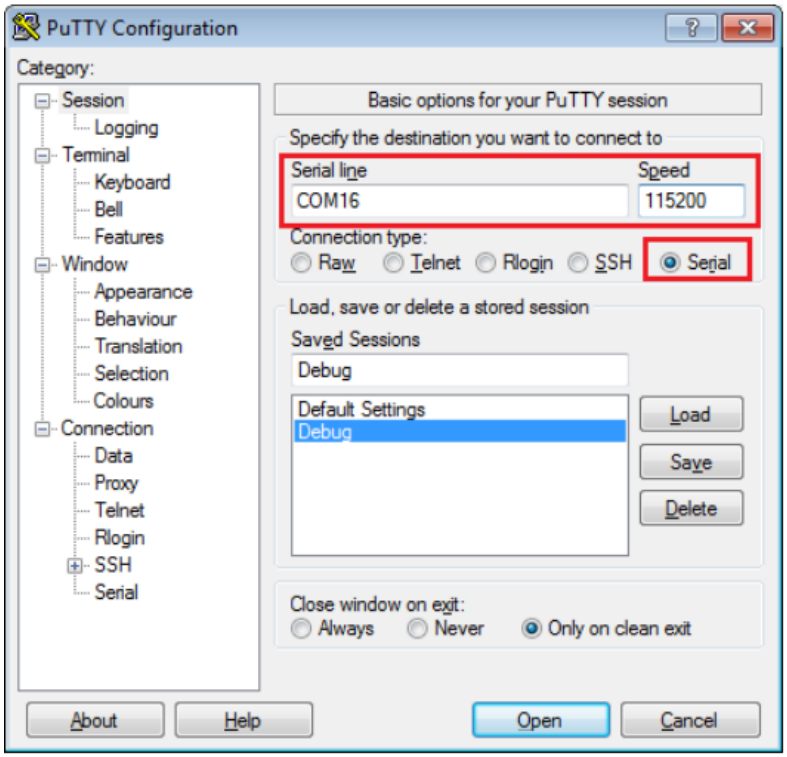

Figure 9: Terminal (PuTTy) configurations

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

Figure 10: Download and Debug button

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

Freescale Confidential Proprietary, Nondisclosure Agreement Required

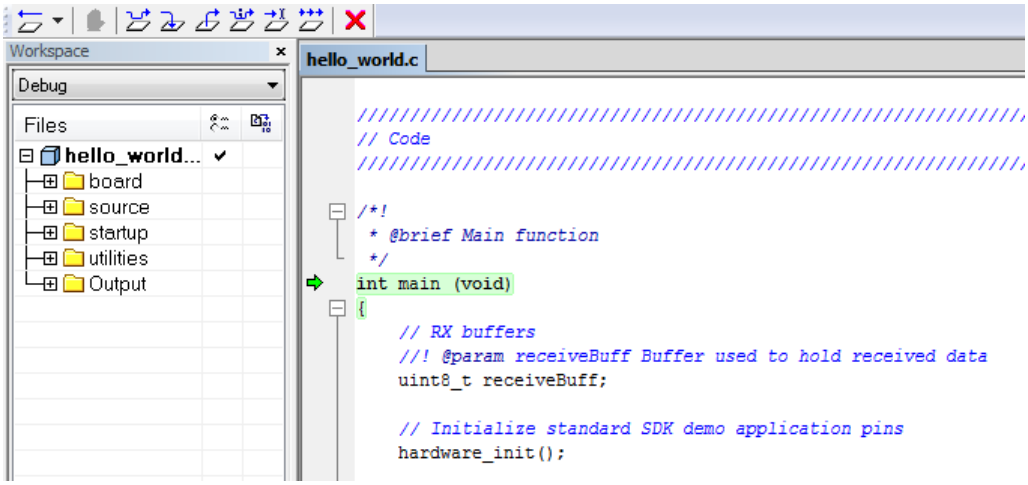

Figure 11: Stop at main() when running debugging

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

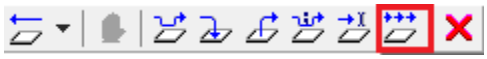

Figure 12: Go button

1. The hello_world application should now be running and this banner should be displayed on the terminal. If this is not the case, check your terminal settings and terminal connections.

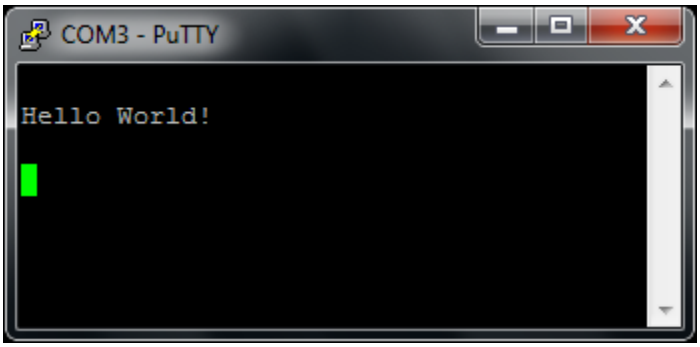

Figure 13: Main prompt of the hello_world demo

5 Build and run the KSDK demo applications using ARM GCC and KDS IDE GCC

5.1 Setup tool chains

5.1.1 Install GCC ARM Embedded tool chain

Text Download and install the installer from www.launchpad.net/gcc-arm-embedded.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

5.1.2 Install MinGW

1. Download the latest mingw-get-setup.exe.
2. Install the GCC ARM Embedded toolchain. The recommended path is C:\MINGW, however, you may install to any location. Note that the installation path may not contain a space.
3. Ensure that the mingw32-base and msys-base are selected under Basic Setup.
4. Finally, click “Installation” and “Apply changes”.

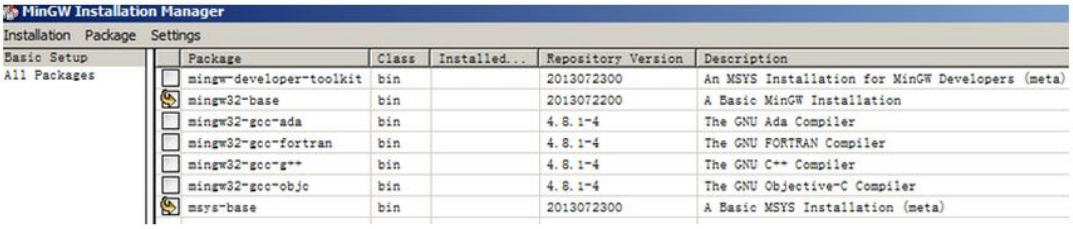

Figure 14: Setup MinGW and MSYS

Freescale Confidential Proprietary, Nondisclosure Agreement Required

5. Add paths C:\MINGW\msys\1.0\bin;C:\MINGW\bin to the system environment. Note that if the GCC aRM Embedded tool chain was installed somewhere other than the recommended location, the system paths added should reflect this change. An example using the recommended installation locations are shown below.

NOTE

There is a high chance that, if the paths are not set correctly, the tool chain will not work properly.

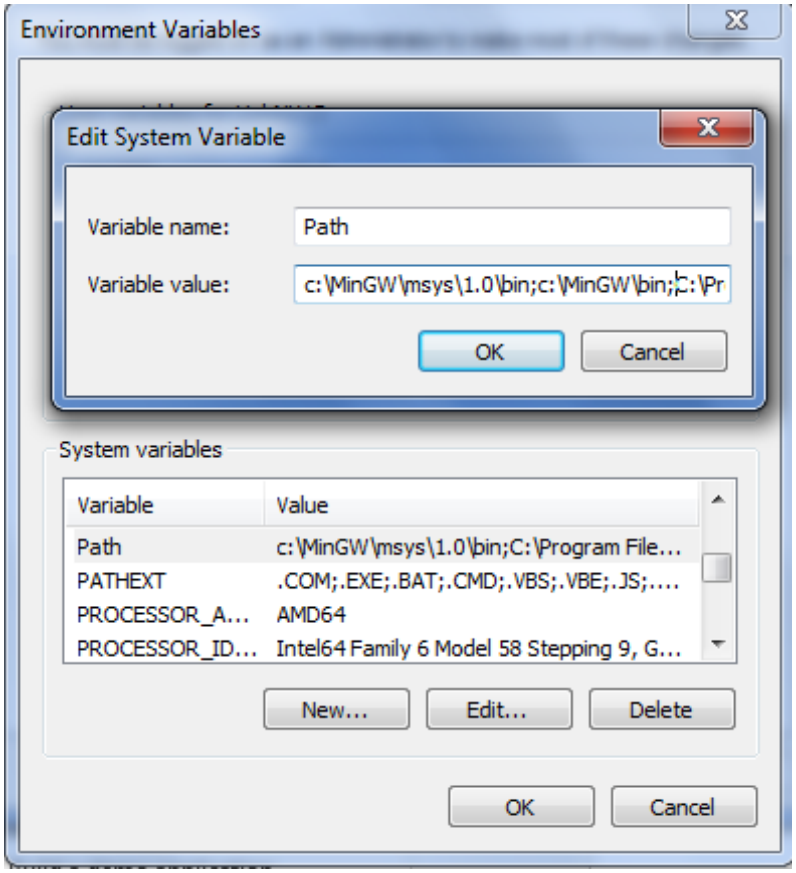

Figure 15: Add Path to systems environment

Freescale Confidential Proprietary, Nondisclosure Agreement Required

5.1.3 Add new system environment variable ARMGCC_DIR

Create a new system environment variable ARMGCC_DIR. The value of this variable should be the short name of the ARM GCC Embedded tool chain installation path.

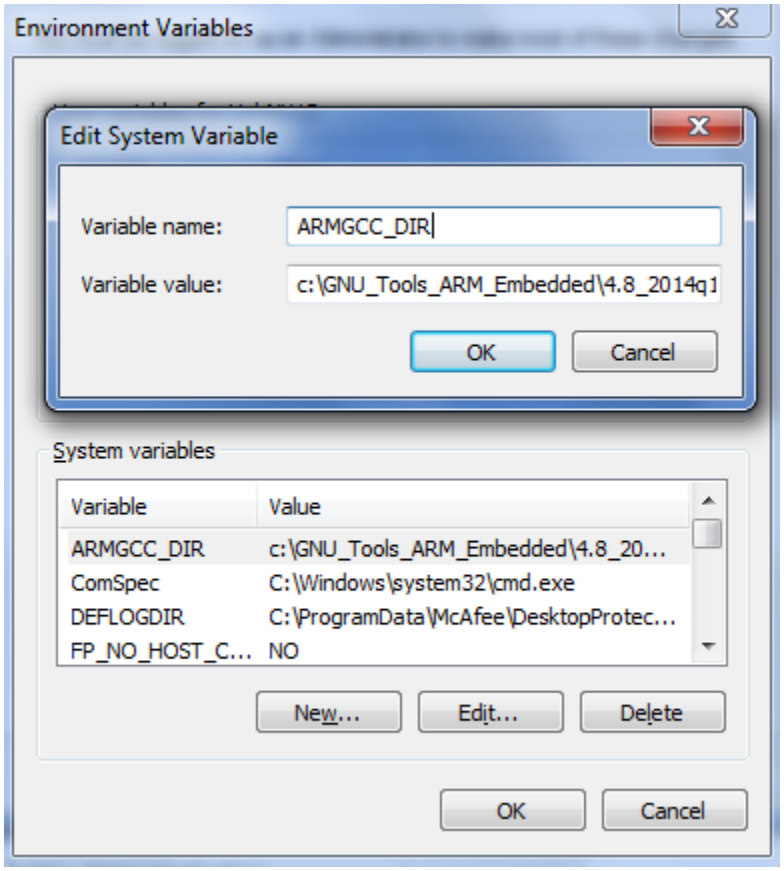

Figure 16: Add ARMGCC_DIR system variable

Freescale Confidential Proprietary, Nondisclosure Agreement Required

5.1.4 Add new system environment variable KDSGCC_DIR

Create a new system environment variable KDSGCC_DIR. The value for the variable is the name of the Kinetis Design Studio (KDS) IDE ARM GCC installation path. By default, KDS IDE is installed to the C:/Freescale/KDS_1.1.0/toolchain location.

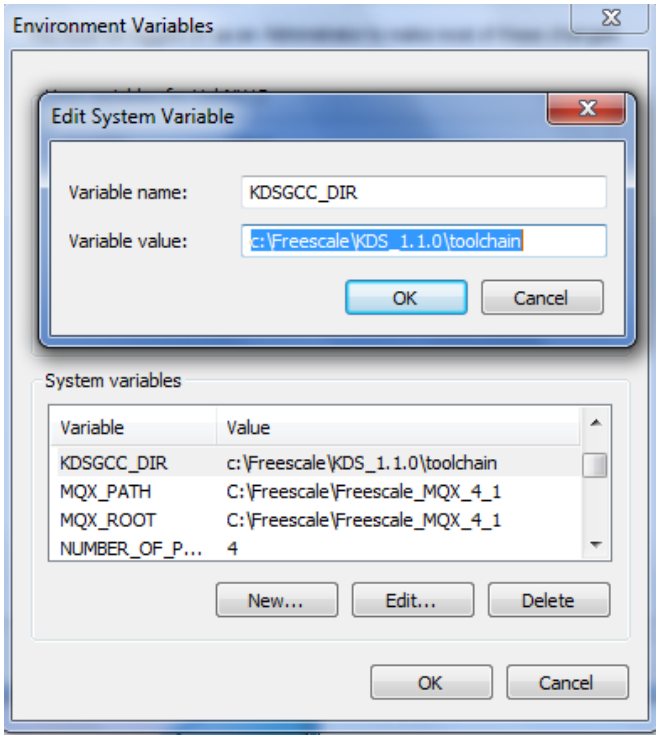

Figure 17: Add KDSGCC_DIR in system variable

5.1.5 Install CMake

1. Install Download CMake 3.0.0 from www.cmake.org/cmake/resources/software.html.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

2. Install Cmake 3.0.0 (ensure that the option "Add CMake to system PATH" is selected when installing).

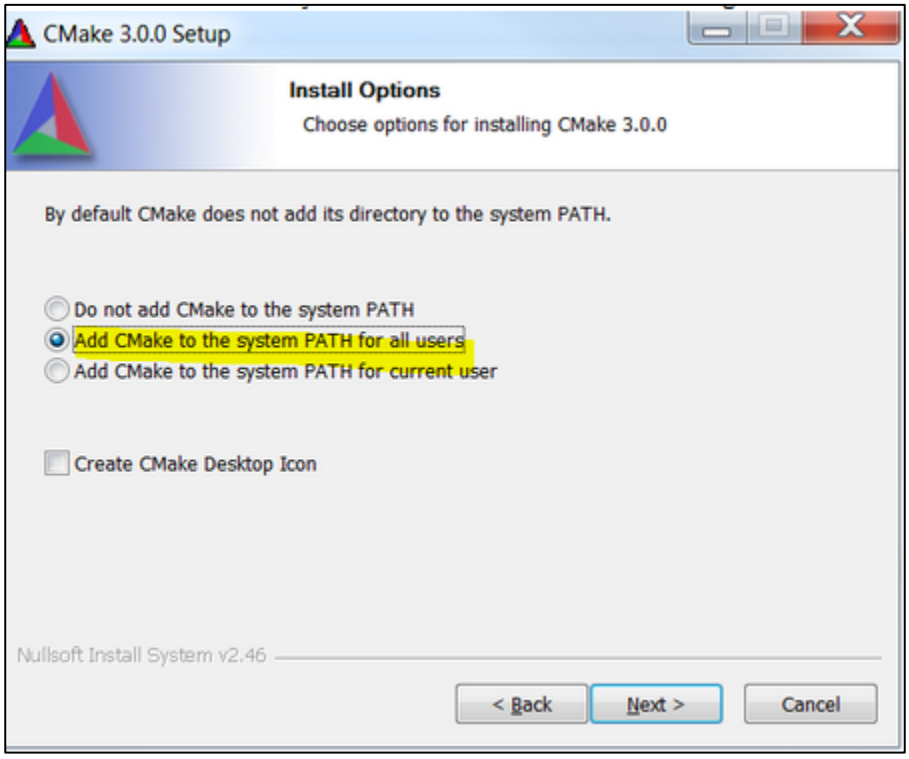

Figure 18: Install CMake

5.2 Build the platform driver library

To build the platform library, follow these instructions:

a. Open a GCC ARM Embedded tool chain command window.

b. Change the directory of the command window to the platform lib directory in the KSDK (one of these):

<install_dir>/lib/ksdk_platform_lib/kdsgcc/<device_name>/

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

c. Type “build_all”.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

d. Type “build_all”. CMake builds both the “debug” and the “release” target.

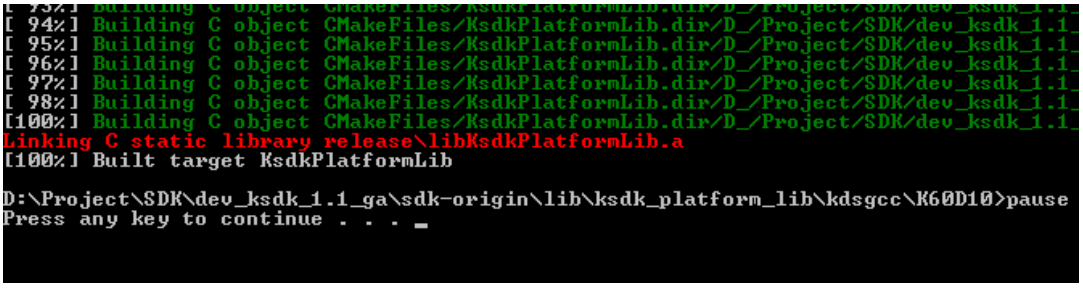

Figure 19: ksdk_platform_lib build successfully

e. The library (ksdk_platform_lib.a) is generated in these directories according to the build target:

<install_dir>/lib/ksdk_platform_lib/kdsgcc/<device_name>/Debug

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

<install_dir>/lib/ksdk_platform_lib/kdsgcc/<device_name>/Release

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

5.3 Build a demo application

1. Change the directory to the project directory:

   <install_dir>/demos/<demo_name>/kdsgcc/<board_name>

   <install_dir>/demos/<demo_name>/armgc/<board_name>

2. Run the build_all.bat two times as described in steps 3 and 4 of section 4.2. The build output is shown in this figure:

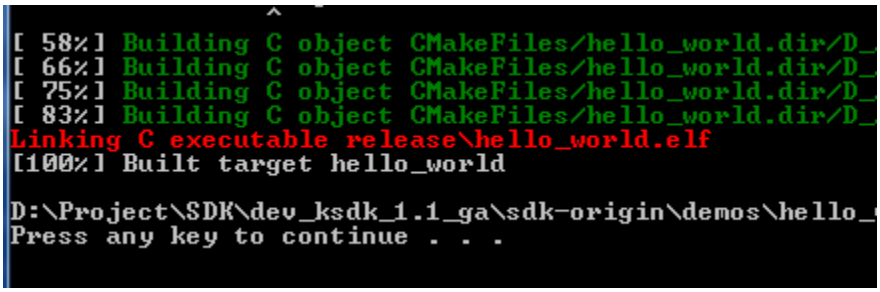

Figure 20: Hello_world demo build successfully

5.4 Run a demo application

This section describes steps to run a demo application using J-Link GDB Server application.

1. Follow steps 1 and 2 in Section 3.3 to set up the hardware and configure the terminal program.
2. Connect the J-Link debug pod to the SWD/JTAG connector of the board.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

3. Open the J-Link GDB Server application and modify your connection settings as shown in this figure.

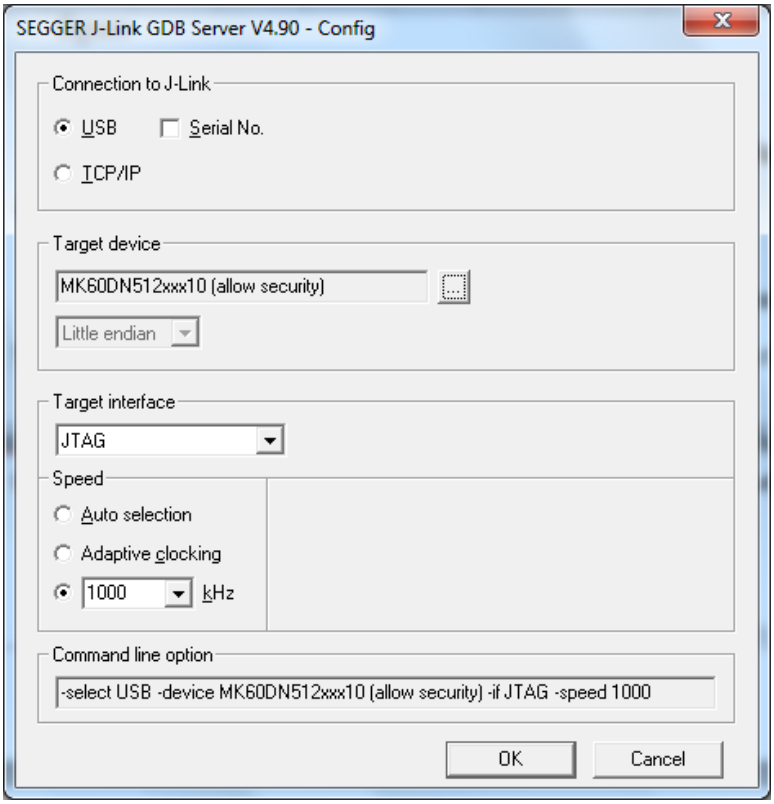

Figure 21: Segger J-Link GDB Server configuration

Freescale Confidential Proprietary, Nondisclosure Agreement Required

4. Once connected, the screen should resemble this figure:

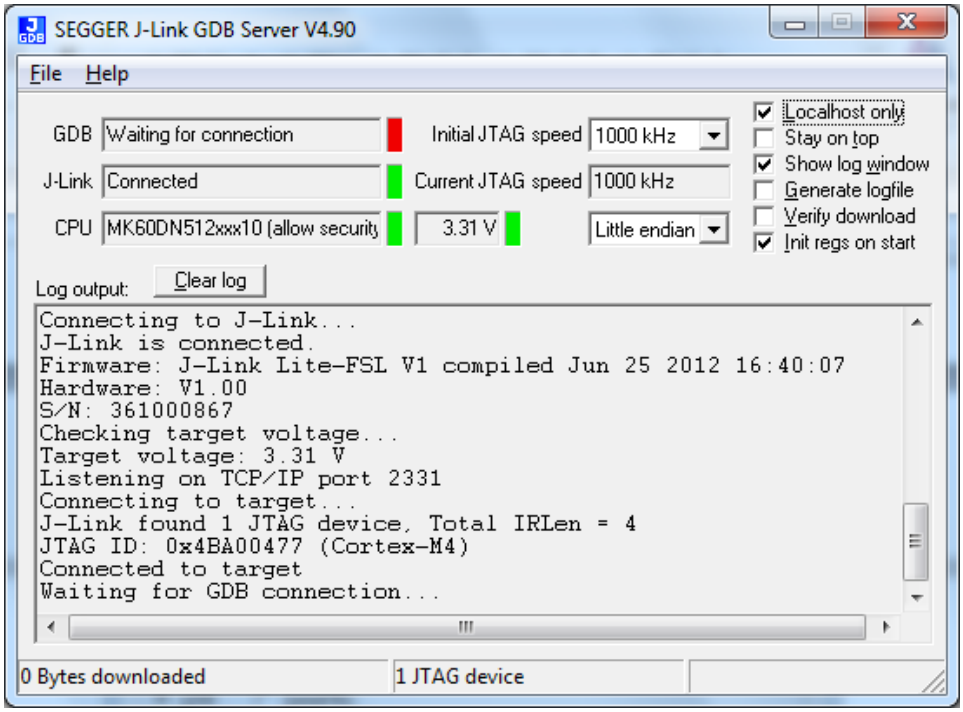

Figure 22: Segger J-Link GDB Server screen after successful connection

5. Open the ARM GCC command prompt and change the directory to the output directory of the desired demo. For this example, the directory is:

   <install_dir>/demos/<demo_name>/ armgcc /<board_name>/ Debug

Freescale Confidential Proprietary, Nondisclosure Agreement Required

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

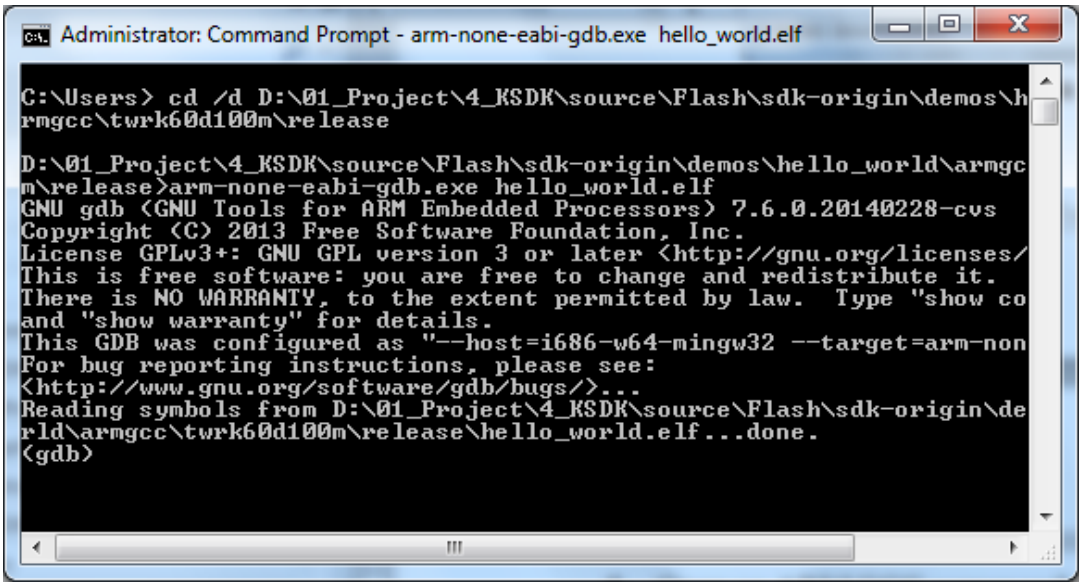

Figure 23: Run arm-none-eabi-gdb

7. Run these commands:
   a. “target remote localhost: 2331”
   b. “monitor reset”
   c. “monitor halt”
   d. “load”
   e. “monitor reset”
8. The application is downloaded and connected. Execute the “monitor go” command to start the demo application.
9. The hello_world application should now be running and the terminal should resemble Figure 9.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

6 Build and run the KSDK demo applications using Keil MDK

This section describes the steps required to configure Keil MDK to build, run, and debug demo applications and necessary driver libraries provided in the KSDK. The Hello World demo application targeted for the TWR-K60D100M Tower System hardware platform is used as an example.

6.1 Building the platform driver library

Before building and debugging KSDK demo applications, the driver library archive, platform_lib.lib, must be built. This library contains all HAL and peripheral driver functions. Since the library archive is device specific, each device has its own library. To build the library for a device, follow these instructions:

1. Open the platform driver library workspace file in Keil MDK. The platform driver library project (ksdk_platform_lib.uvproj) is located in this folder:

   <install_dir>/lib/ksdk_platform_lib/uv4/<device_name>

   Using the K60D100M as an example, open this file:

   <install_dir>/lib/ksdk_platform_lib/uv4/K60D10/ksdk_platform_lib.uvproj

2. Choose the appropriate build target: “Debug” or “Release” from the drop-down menu.

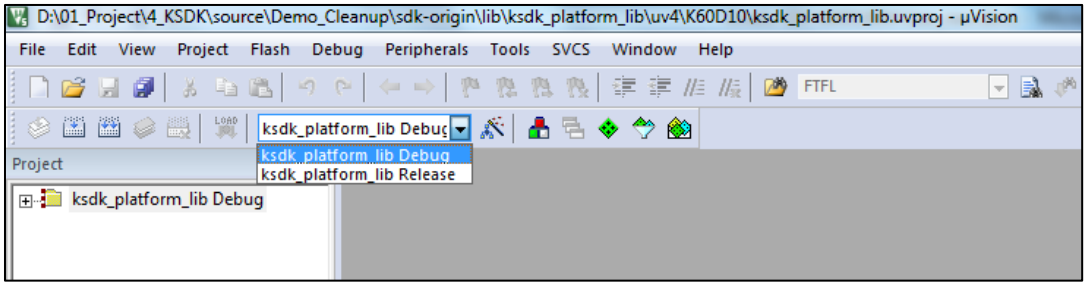

Figure 24: Drop-down selection of the debug/release target

3. Rebuild the project files by left-clicking the “Rebuild” button.

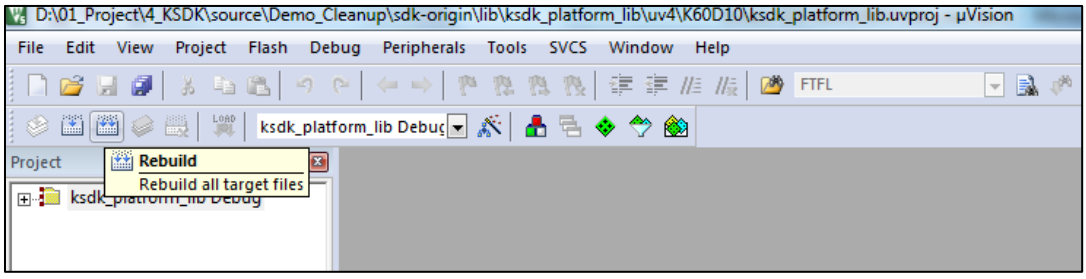

Figure 25: “Rebuild all” button

Freescale Confidential Proprietary, Nondisclosure Agreement Required

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

   <install_dir>/lib/ksdk_platform_lib<toolchain>/<device_name>/output/Debug

   <install_dir>/lib/ksdk_platform_lib/<toolchain>/<device_name>/output/Release

6.2 Build a demo application

The Kinetis SDK development platform provides both Keil MDK project and multi-project files for demo applications. The multi_project workspace, <demo_name>.uvmpw, includes the demo project and also the ksdk_platform_lib project for a specific platform. The ksdk_platform_lib can be built from the multi-project file.

The KSDK demo applications utilize a library archive to compile in the necessary functions. Therefore, this library must be built before compiling and downloading. To check if this library exists, verify that the ksdk_platform_lib.lib file is located in:

<install_dir>/lib/ksdk_platform_lib/<toolchain>/<device_name>/<build>

The <build> is the desired build, which can be either Debug or Release. For example, if the desired project is the Debug version of the hello_world demo application, there must be a ksdk_platform_lib.lib library in this folder:

<install_dir>/lib/ksd_platform_lib/uv4/K60D10/debug

If the appropriate library archive is not present, follow the steps in section 3.1 to generate the necessary library archive. Otherwise, continue with these steps to build the desired demo application.

1. Open the hello_world multi-project workspace file (hello_world.uvmpw) in Keil using the "Project -> Open project…" or double click the file:

   <install_dir>/demos/hello_world/uv4/twrk60d100m/hello_world.uvmpw

2. Set the hello_world project as the active project by right-clicking on hello_world and selecting "Set as Active Project".

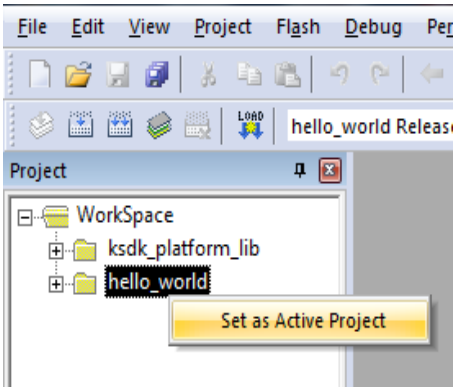

Figure 26: Selection of the hello_world project as the active project

Freescale Confidential Proprietary, Nondisclosure Agreement Required

3. When the build is complete, Keil shows this information in the Build Output window:

```
compiling startup.c...
compiling hello_world.c...
compiling gpio_pins.c...
compiling pin_mux.c...
compiling hardware_init.c...
linking...
Program Size: Code=14244 RO-data=1232 RW-data=48 ZI-data=16400
"debug\hello_world.out" - 0 Error(s), 0 Warning(s).
```

Figure 27: Build Output window upon successful hello_world build

6.3 Run a demo application

To download and run the application, perform these steps:

1. Connect the development platform to your PC via USB cable between the OpenSDA USB connector and the PC USB connector.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

2. Open the terminal application on the PC, such as PuTTY or Teraterm, and connect to the OSJTAG serial port number. Configure the terminal with these settings:
    a. 115200 baud rate
    b. No parity
    c. 8 data bits
    d. 1 stop bit

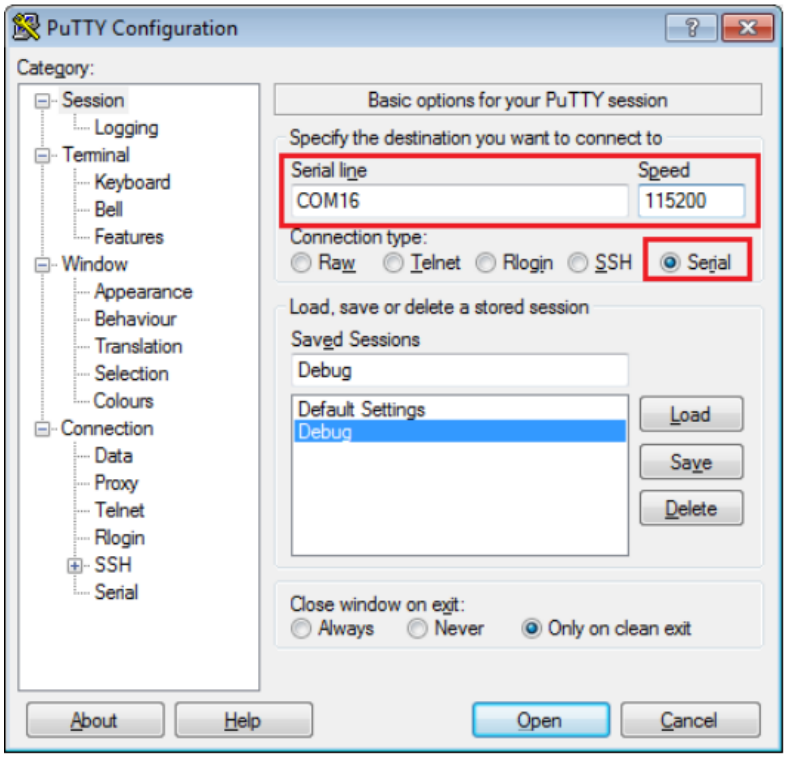

Figure 28: Terminal (PuTTY) configurations

3. After the application is properly build, click the "Download" button to download the application to the target.

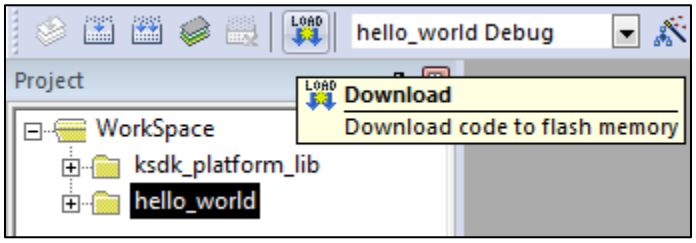

Figure 29: Download button

Freescale Confidential Proprietary, Nondisclosure Agreement Required

4. 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.

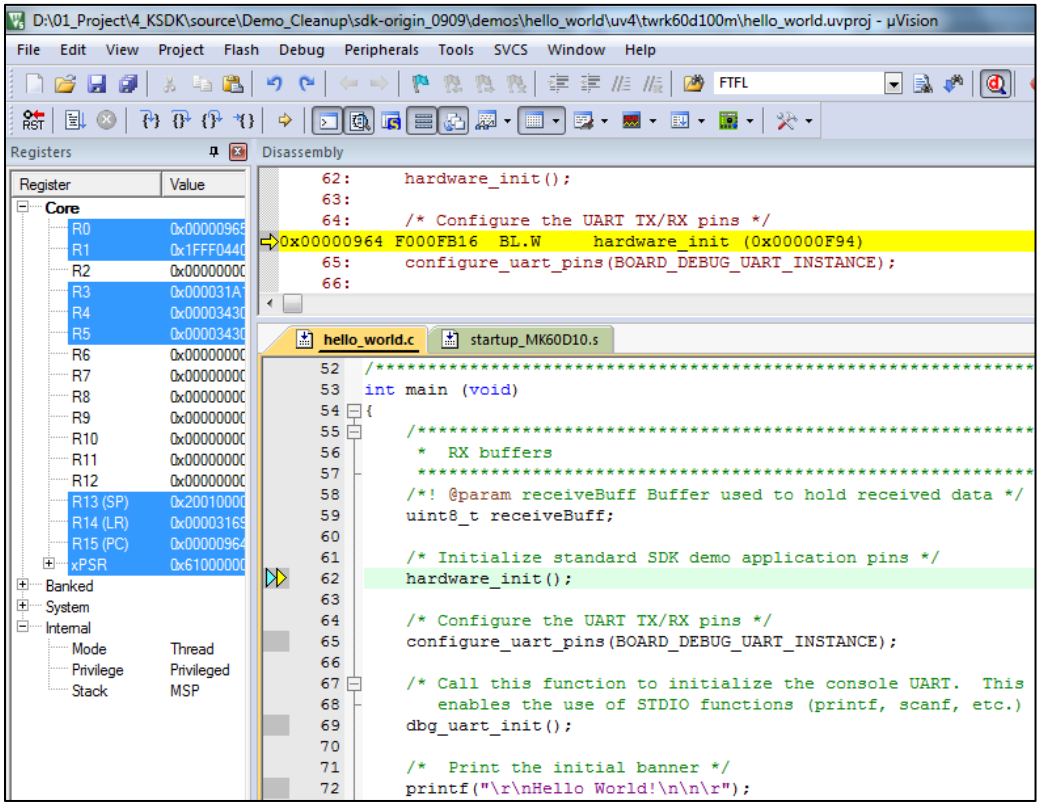

Figure 30: Stop at main() when run debugging

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

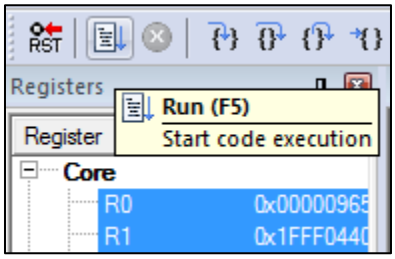

Figure 31: Go button

Freescale Confidential Proprietary, Nondisclosure Agreement Required

The hello_world application should now be running and the banner, shown in the figure, should be displayed on the terminal.

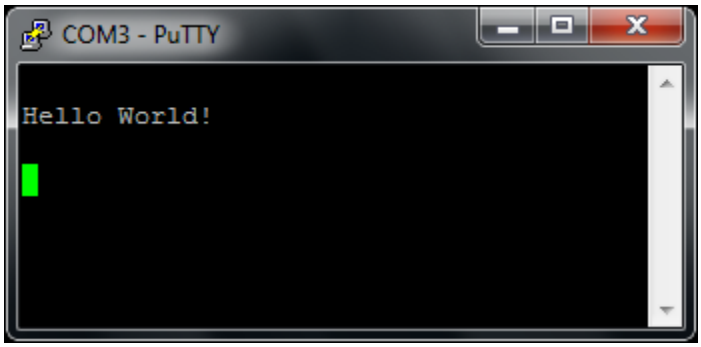

Figure 32: Main prompt of the hello_world demo

Freescale Confidential Proprietary, Nondisclosure Agreement Required

7 Build and run the KSDK demo applications 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 necessary driver libraries provided in the KSDK. The Hello World demo application targeted for the TWR-K60D100M Tower System hardware platform is used as an example.

7.1 Installing KSDK Eclipse update

Before using any Eclipse-based IDE with KSDK, the KSDK Eclipse Update must be applied. Without this update, Eclipse cannot generate KSDK-compatible projects. To install the update, follow these instructions:

1. Select “Help -> Install New Software”.

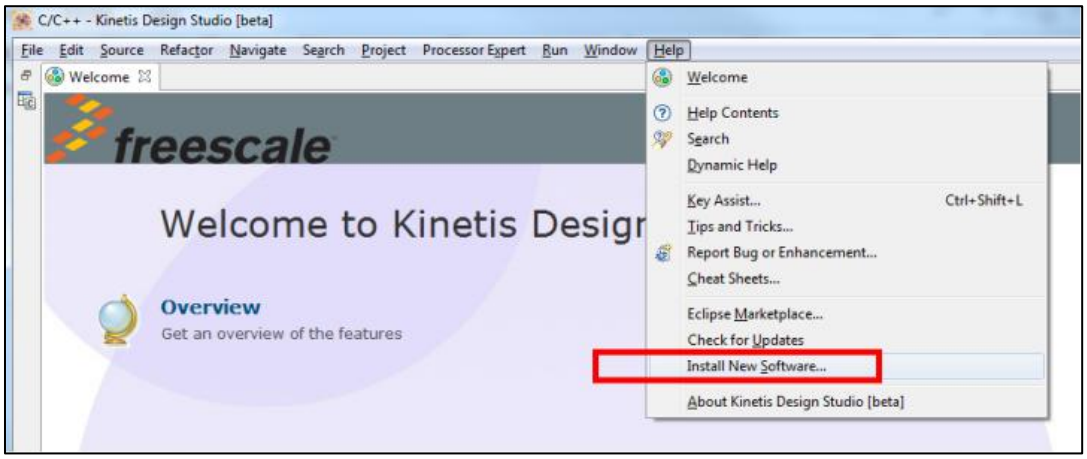

Figure 33: Install new software

2. In the “Install New Software” dialog box, click the “Add” button in the upper right corner.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

3. In the “Add Repository” dialog, select “Archive”.

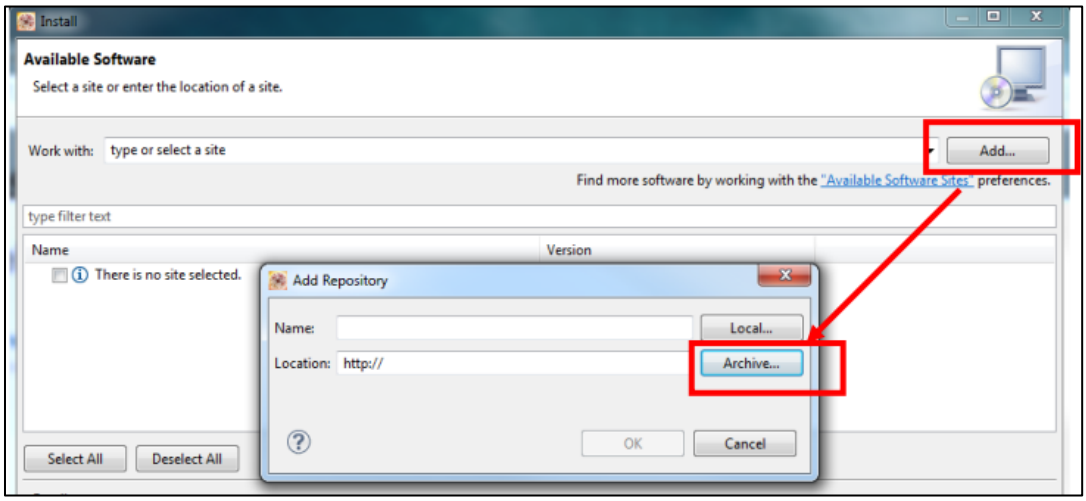

Figure 34: Add Repository for new software

4. In the Repository archive dialog box, browse the KSDK install directory.
5. From the top-level, enter the tools/eclipse_update folder and select the SDK_version_Update_for_Eclipse.zip file.
6. Click “Open”, and “OK” in the “Add Repository” dialog box.
7. The KSDK update shows up in the list of the original Install dialogs.
8. Check the box to the left of the KSDK Eclipse update and click “Next” in the lower right corner.
9. Follow the remaining instructions to finish the installation of the update.

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

7.2 Open the platform driver library

1. Select “File” and “Import” from the KDS IDE Eclipse menu.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

2. Expand the General folder and select “Existing Projects into Workspace”. Then, click “Next”.

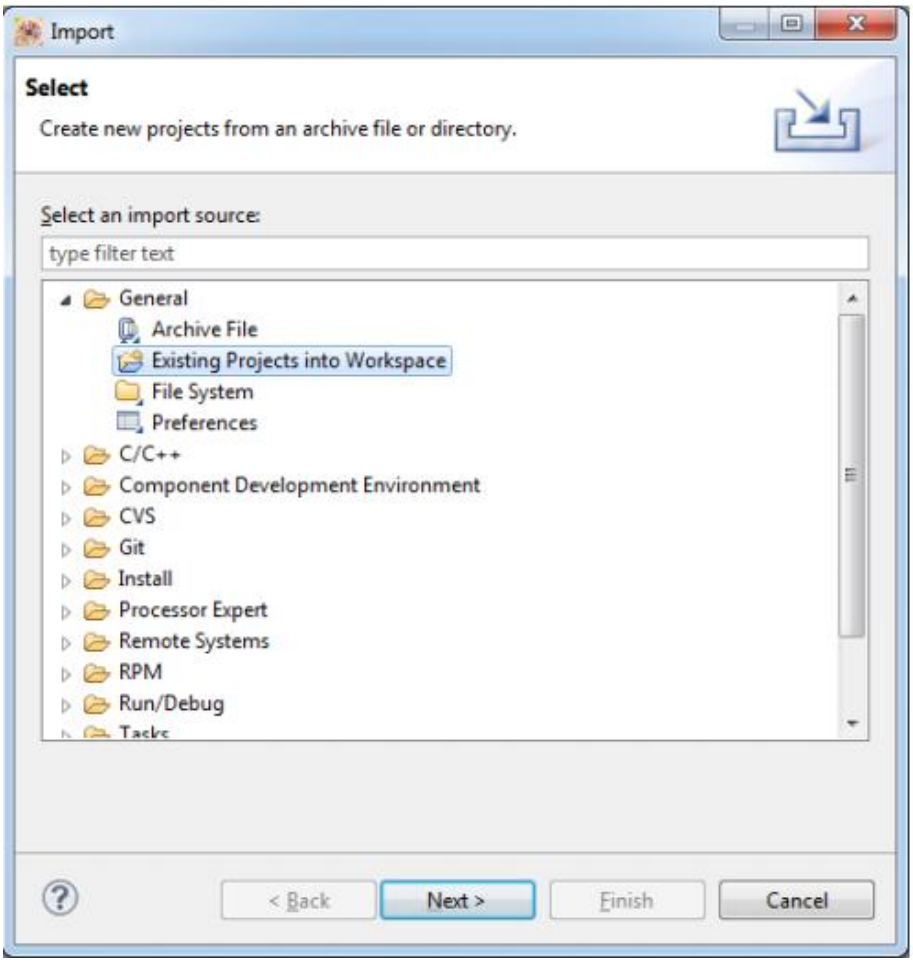

Figure 35: Selection of the correct import type in KDS IDE

3. Select the “Select root directory:” option. Click “Browse…” to point the KDS IDE to the correct library.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

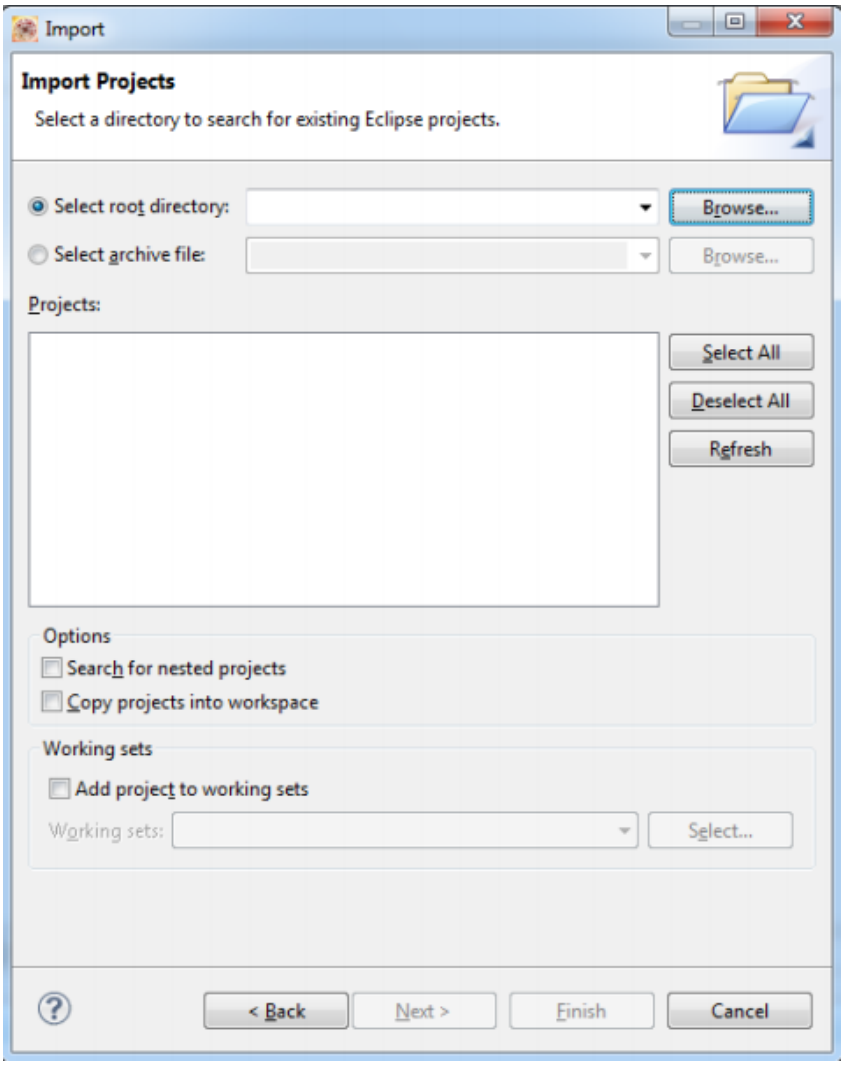

Figure 36: Projects directory selection window

4. Point the KDS IDE to the ksdk_platform_lib project in the K60D10. The library project is located at this location:

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

Freescale Confidential Proprietary, Nondisclosure Agreement Required

The import projects directory selection window should resemble this figure.

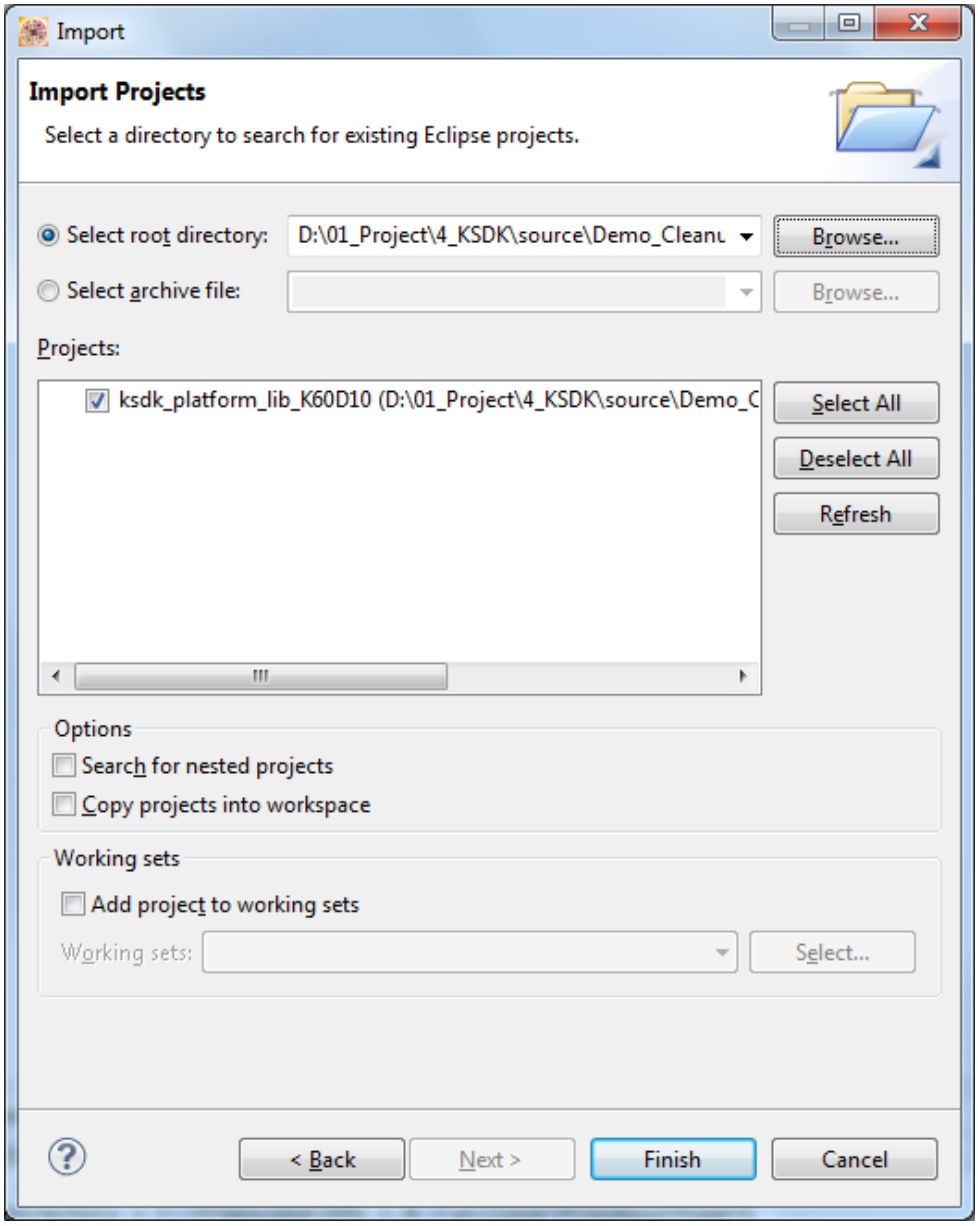

Figure 37: Select K60D100M ksdk_platform_lib project

5. Click the “Finish” button.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

7.3 Build the platform driver library

1. Choose the appropriate build target: “Debug” or “Release” by left-clicking the arrow next to the hammer icon as shown here.

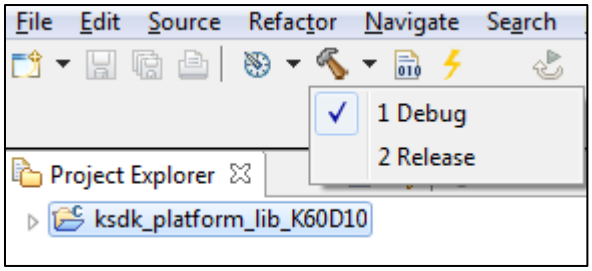

Figure 38: Selection of the build target in KDS IDE

2. If the library build does not begin after selecting the desired target, left-click the hammer icon to start the build.

7.4 Build a demo application

To build a demo application, follow these steps.

1. Repeat the steps provided in Section 6.2 and 6.3 for the hello_world demo application. The demo application project is located in this folder:

   _dir> / demos / <demo name> / kds / <device_name>

   For the hello_world demo example, this path is as follows:

   <install_dir> / demos / hello_world / kds /K60D100M

7.5 Run a demo application

To download and run the application, perform these steps:

1. Connect the development platform to your PC via USB cable between the OpenSDA USB connector and the PC USB connector.
2. Open the terminal application on the PC, such as PuTTY or Teraterm, and connect to the OSJTAG serial port number. Configure the terminal with these settings:
   a. 115200 baud rate
   b. No parity
   c. 8 data bits
   d. 1 stop bit

Freescale Confidential Proprietary, Nondisclosure Agreement Required

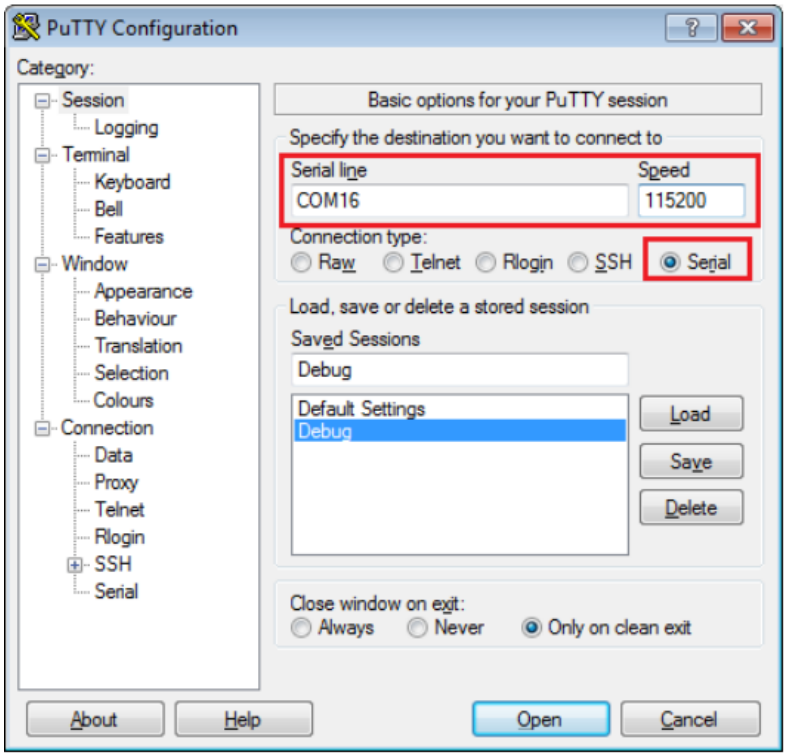

Figure 39: Terminal (PuTTY) configurations

3. Ensure that the debugger configuration is correct in the project options. Consult the device-specific User's Guide for more information about the default debugger application for the target. Note that changing the OpenSDA debug application may be necessary.

    a. To check the debugger configurations, click the down arrow next to the green "Debug" button and select "Debug Configurations".

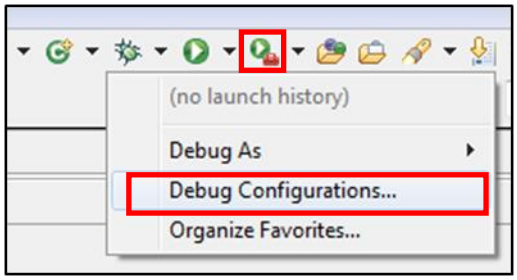

Figure 40: Debug Configurations dialog button

    b. In the "Debug Configurations" dialog box, select debug configuration from the GDB PEMicro Interface and chose the hello_world project in the left hand side of the dialog box.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

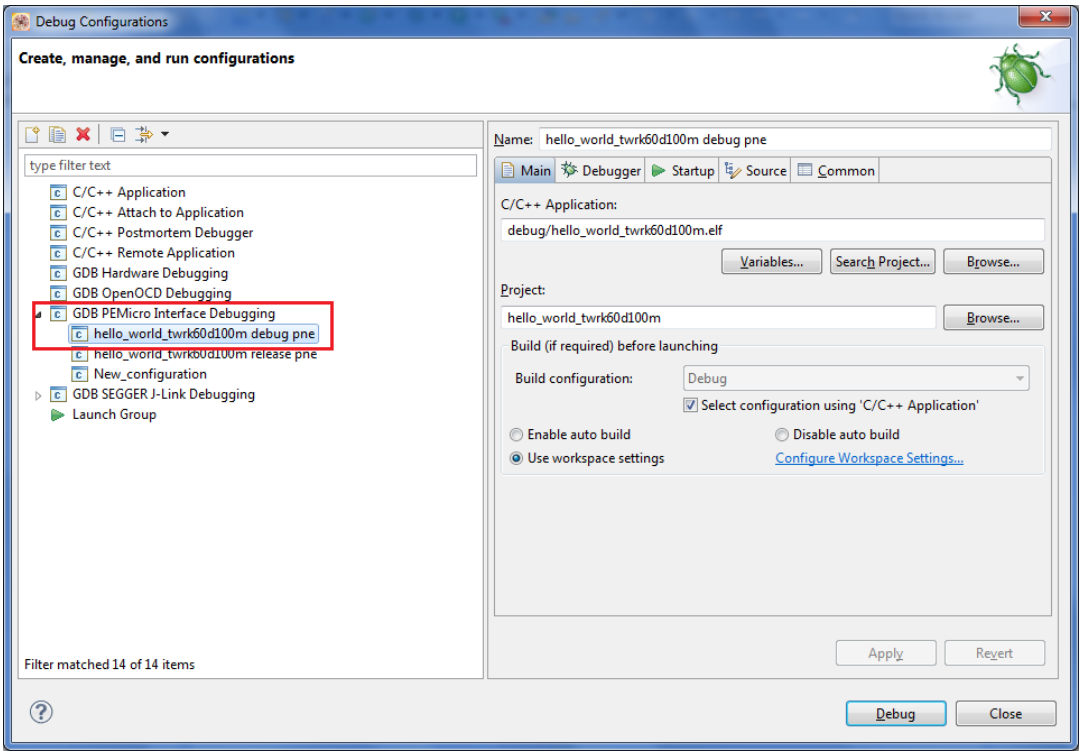

Figure 41: Selection of the Debug configuration in the Debug Configuration dialog box

Freescale Confidential Proprietary, Nondisclosure Agreement Required

c. In the debugger step, verify that the C/C++ Application and Project are set to the correct path. The C/C++ Application path should be debug/<application name>.elf, and the Project should be the target project name.

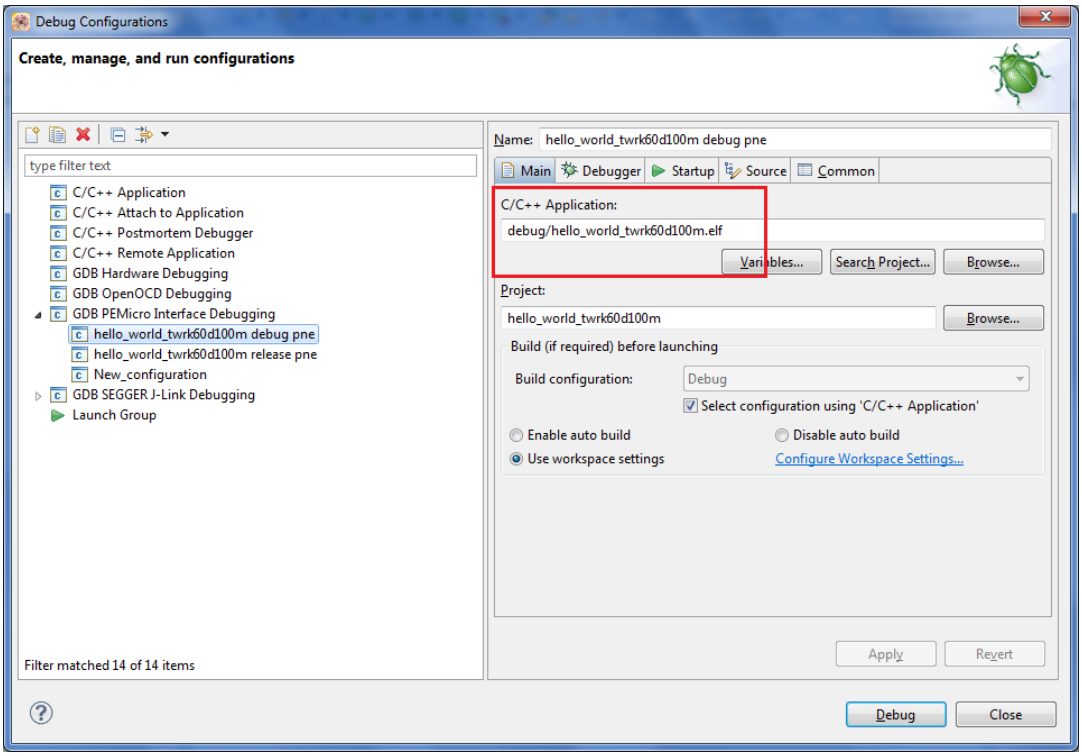

Figure 42: KDS IDE debugger configurations

Freescale Confidential Proprietary, Nondisclosure Agreement Required

d. After verifying the debugger configurations are correct, click the “Debug” button.

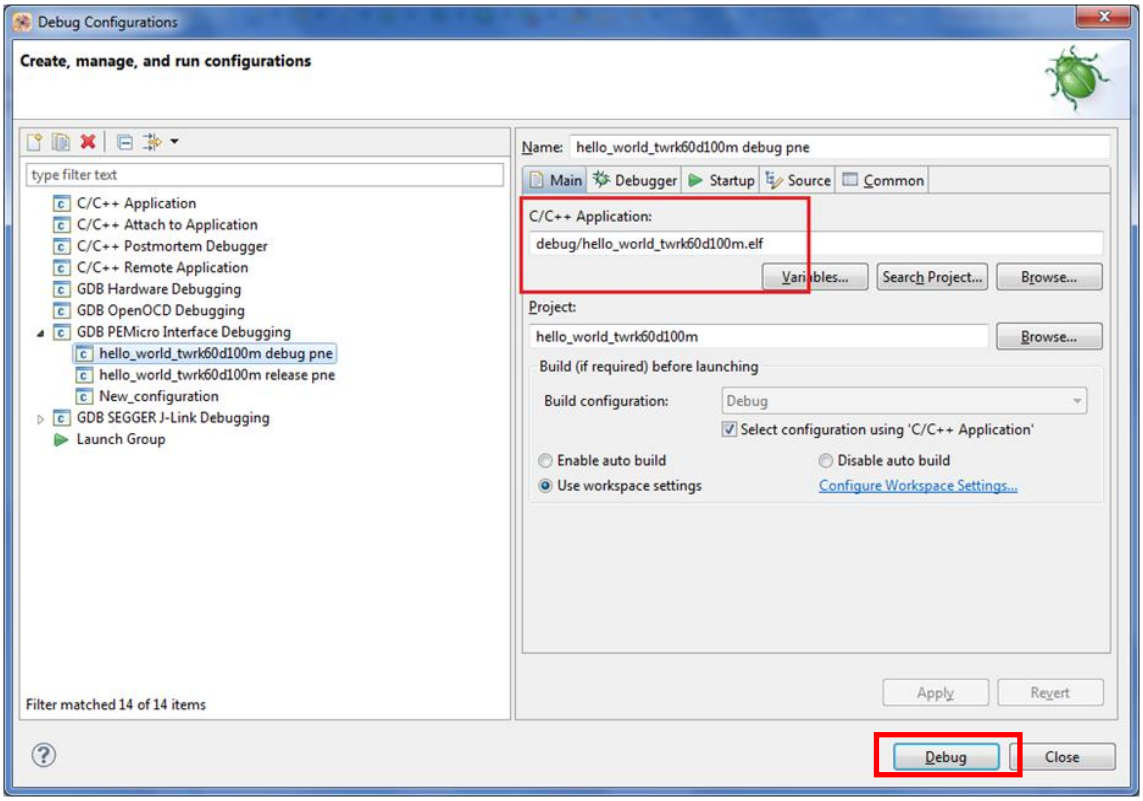

Figure 43: Debug button in Debug Configurations dialog box

Freescale Confidential Proprietary, Nondisclosure Agreement Required

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

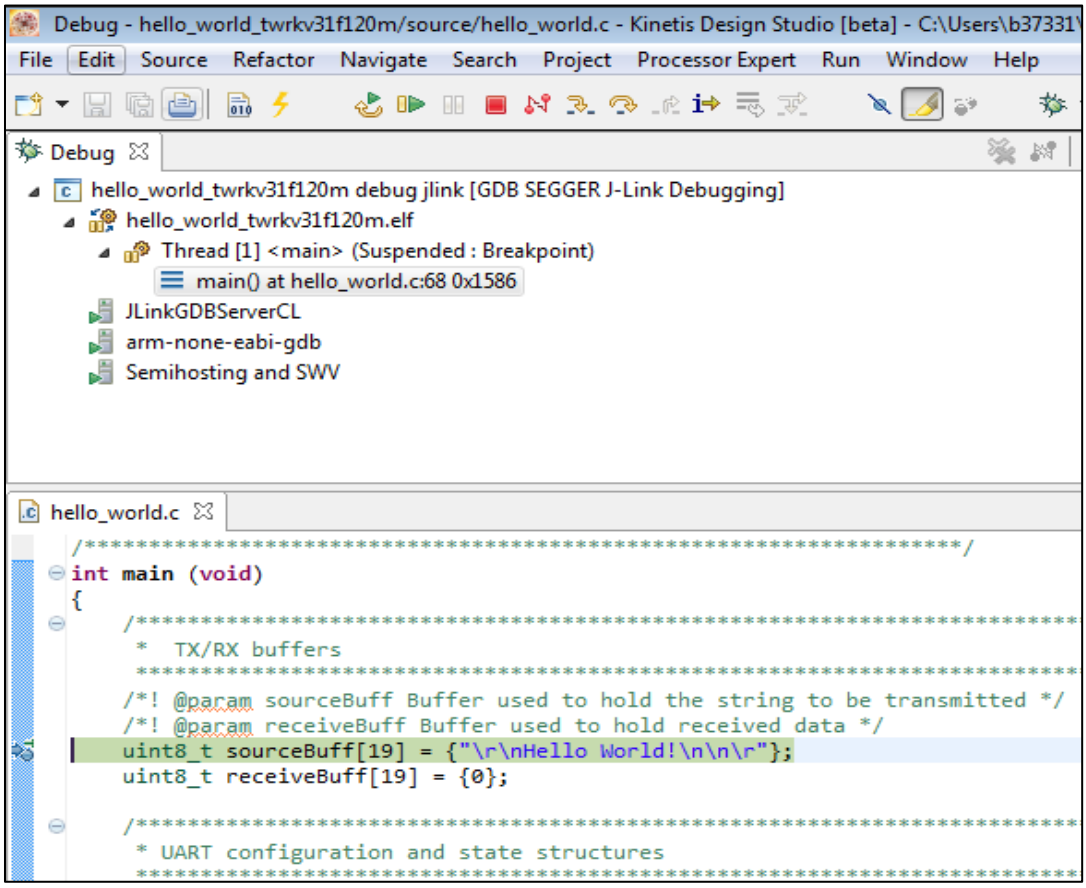

Figure 44: Stop at main() when run debugging

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

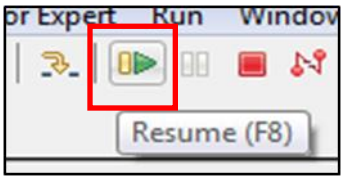

Figure 45: Resume button

The hello_world application should now be running and the banner shown in Figure 9 should be displayed in the terminal.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

8 Build and run the KSDK demo applications 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 TWR-K60D100M Tower System hardware platform is used as an example.

8.1 Installing KSDK Eclipse update

Before using any Eclipse-based IDE with KSDK, the KSDK Eclipse update must be applied. Without this update, Eclipse cannot generate the KSDK-compatible projects. To install the update, follow these instructions:

1. Select “Help” then “Install New Software”.

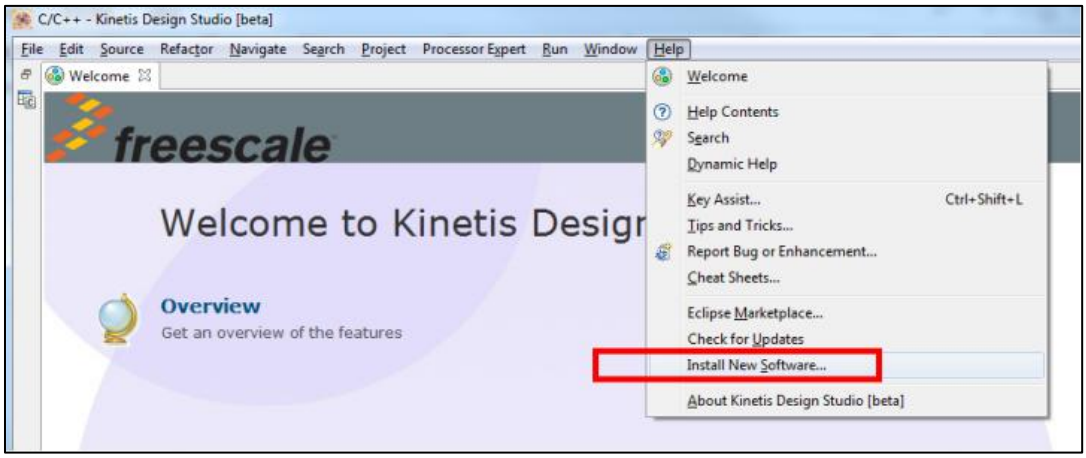

Figure 46: Install new software

2. In the “Install New Software” dialog box, click the “Add” button in the upper right corner.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

3. In the “Add Repository” dialog, select “Archive”.

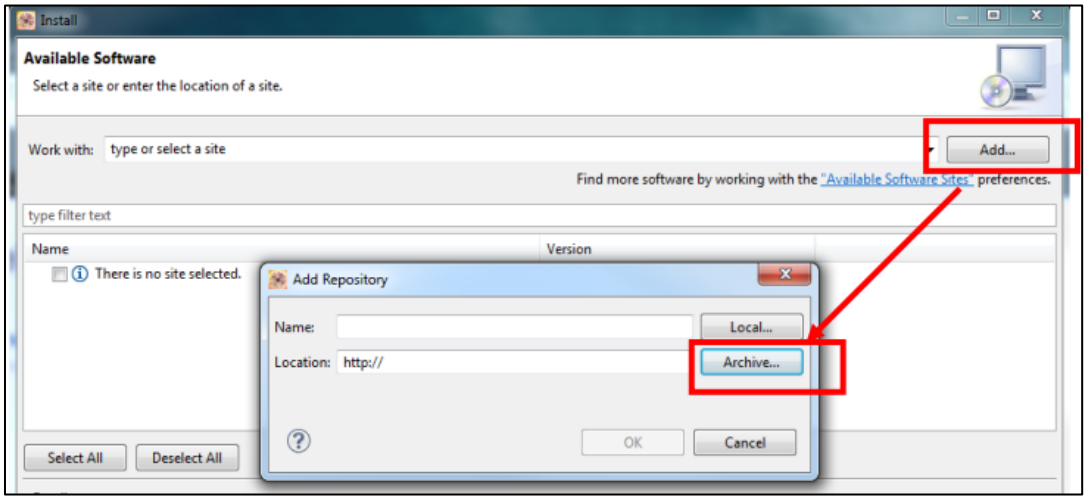

Figure 47: Add Repository for new software

4. In the Repository archive dialog box that appears, browse the KSDK install directory.
5. From the top-level, enter the tools/eclipse_update folder and select the SDK_version_Update_for_Eclipse.zip file.
6. Click “Open”, and then “OK” in the “Add Repository” dialog box.
   The KSDK update shows up in the list of the original Install dialogs.
7. Check the box to the left of the KSDK Eclipse update and click “Next” 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/Eclipse IDE for the changes to take effect.

8.2 Open the platform driver library

1. Select “File” then “Import” from the KDS Eclipse IDE menu.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

2. Expand the General folder and select “Existing Projects into Workspace”. Then, click “Next”.

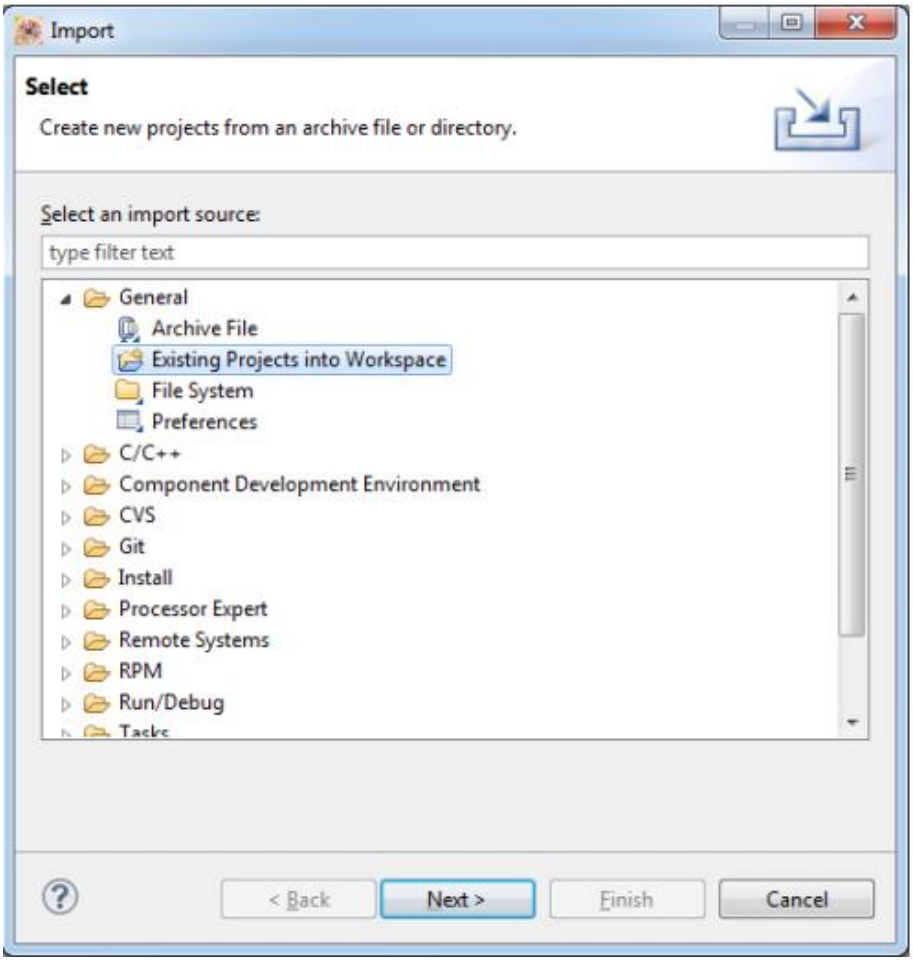

Figure 48: Selection of the correct import type in KDS IDE

Freescale Confidential Proprietary, Nondisclosure Agreement Required

3. Select the “Select root directory:” option. Then, click “Browse” to point TrueSTUDIO to the correct library.

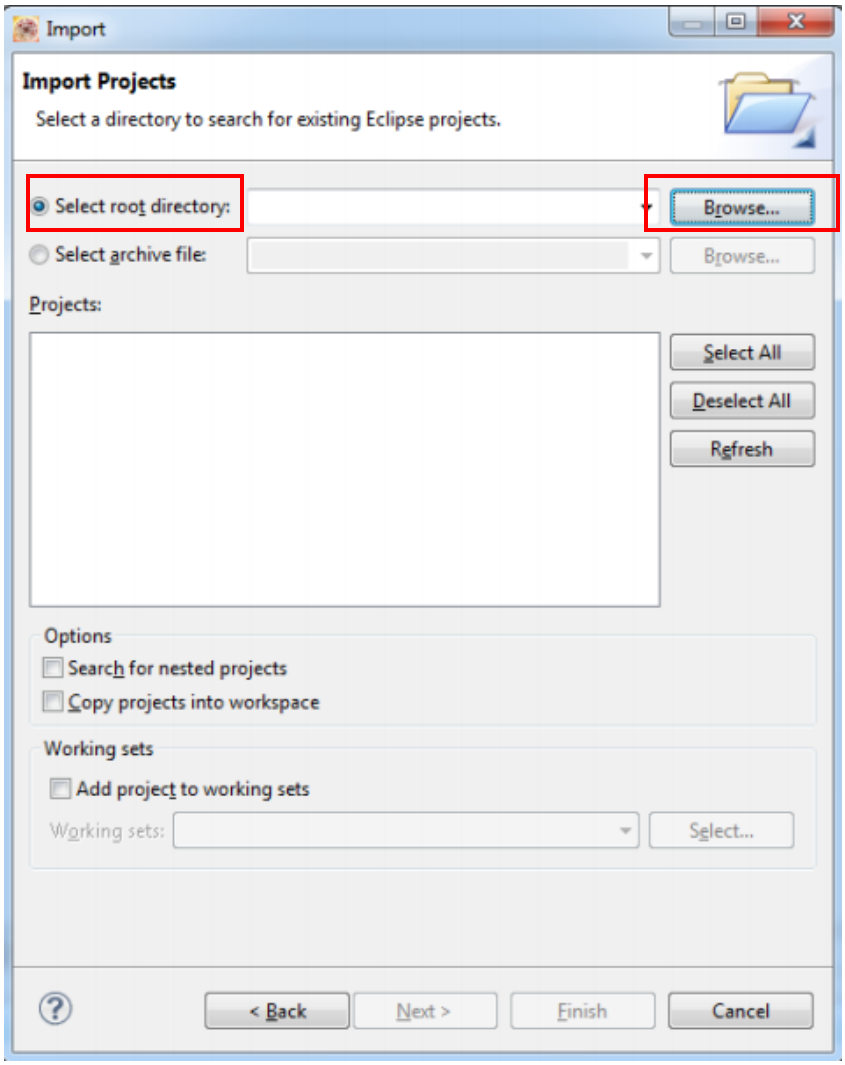

Figure 49: Projects directory selection window

Freescale Confidential Proprietary, Nondisclosure Agreement Required

4. Point KDS IDE to the kdsk_platform_lib project in the K60D10. The library project is at this location:

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

   The Import Projects directory selection window should look as shown here.

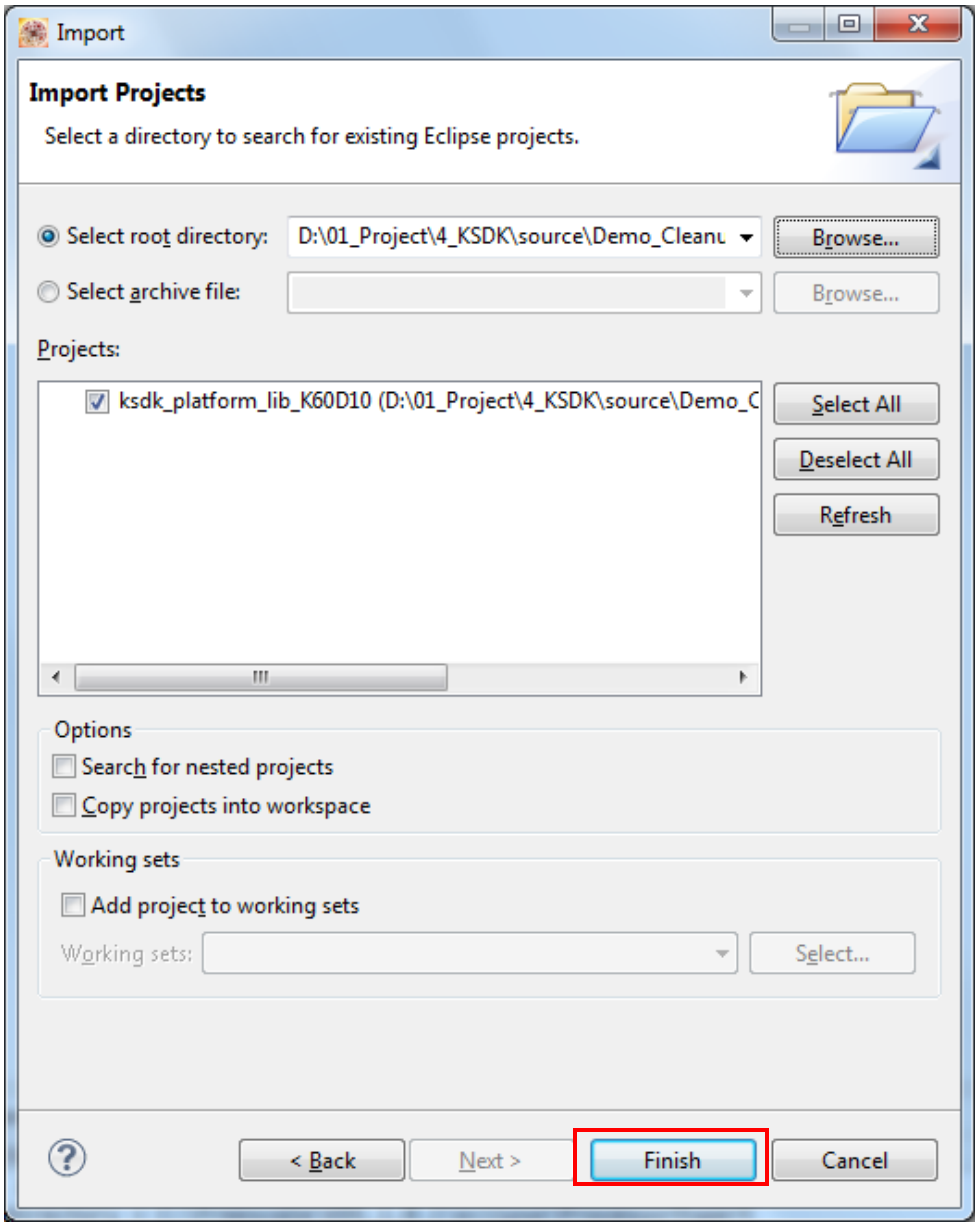

Figure 50: Select K60D100M ksdk_platform_lib project

5. Click the “Finish” button.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

8.3 Build the platform driver library

1. Choose the appropriate build target: “Debug” or “Release”, by left-clicking the arrow next to the hammer icon.

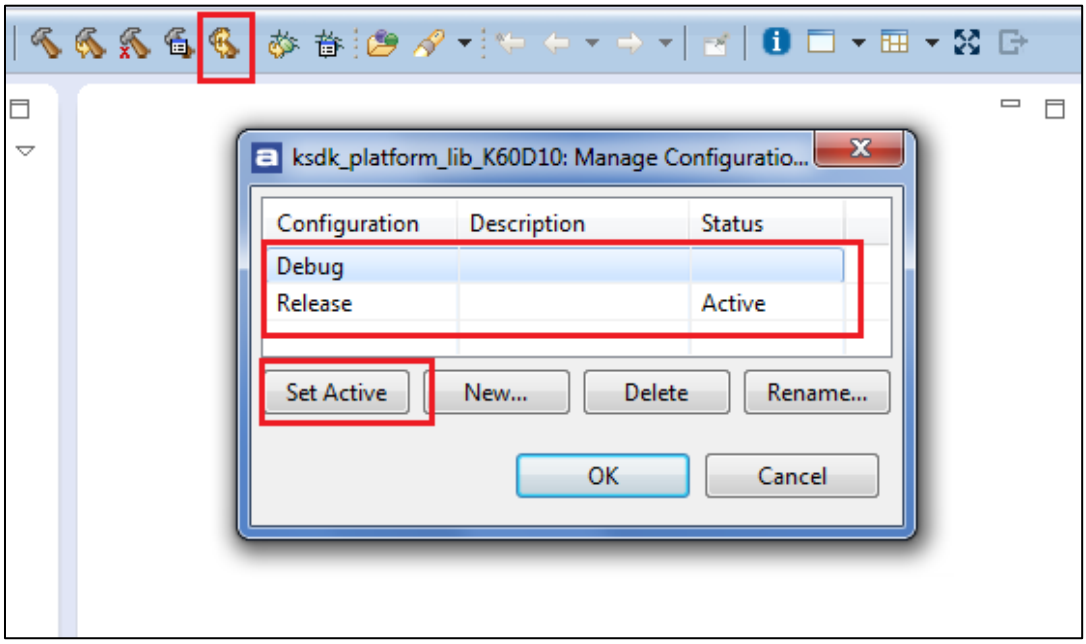

Figure 51: Selection of build target in TrueSTUDIO

2. If the library build does not begin after selecting the desired target, left-click the hammer icon to start the build.

8.4 Build a demo application

To build a demo application, follow these steps:

1. Repeat the steps provided in Section 6.2 and 6.3 for the hello_world demo application. The demo application project is located in the following folder:

   <install_dir> / demos / <demo name> /atl/ <device_name>

   Using the hello_world demo as an example, this path would be as follows:

   <install_dir> / demos / hello_world /atl /K60D100M

8.5 Run a demo application

To download and run the application, perform these steps:

1. Connect the development platform to your PC via USB cable between the OpenSDA USB connector and the PC USB connector.
2. Open the terminal application on the PC, such as PuTTY or Teraterm, and connect to the OSJTAG serial port number. Configure the terminal with these settings:

Freescale Confidential Proprietary, Nondisclosure Agreement Required

a. 115200 baud rate

b. No parity

c. 8 data bits

d. 1 stop bit

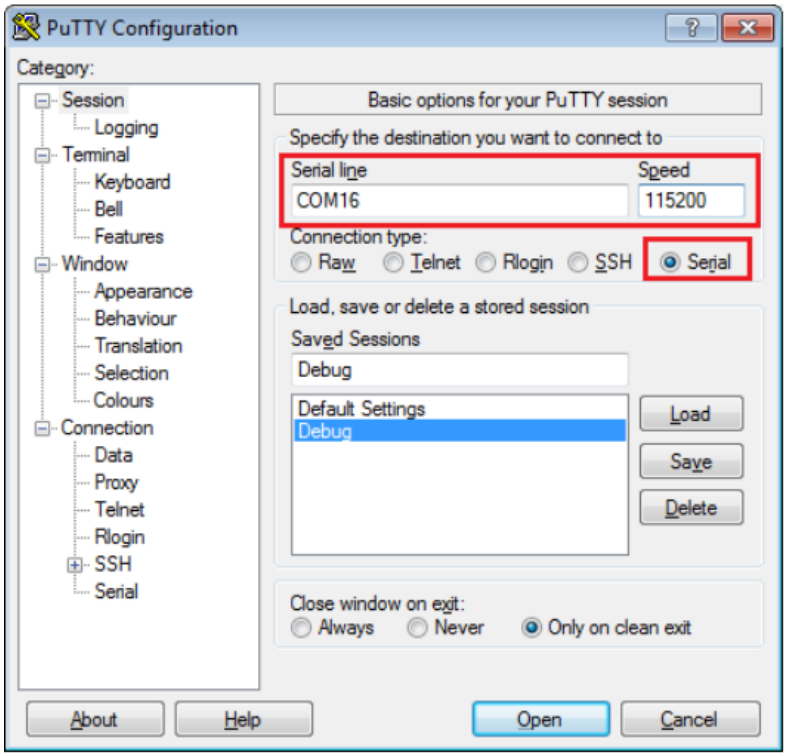

Figure 52: Terminal (PuTTY) configurations

3. Ensure that the debugger configuration is correct in the project options. Consult the device-specific User's Guide for more information about the default debugger application for the target. Note that changing the OpenSDA debug application may be required.

    a. To check the debugger configurations, click the down arrow next to the green "Debug" button and select "Debug Configurations".

Figure 53: Debug configurations dialog button

Freescale Confidential Proprietary, Nondisclosure Agreement Required

b. In the “Debug Configurations” dialog box, select debug configuration from the GDB PEMicro Interface debugging and chose hello_world project in the left hand side of the dialog box.

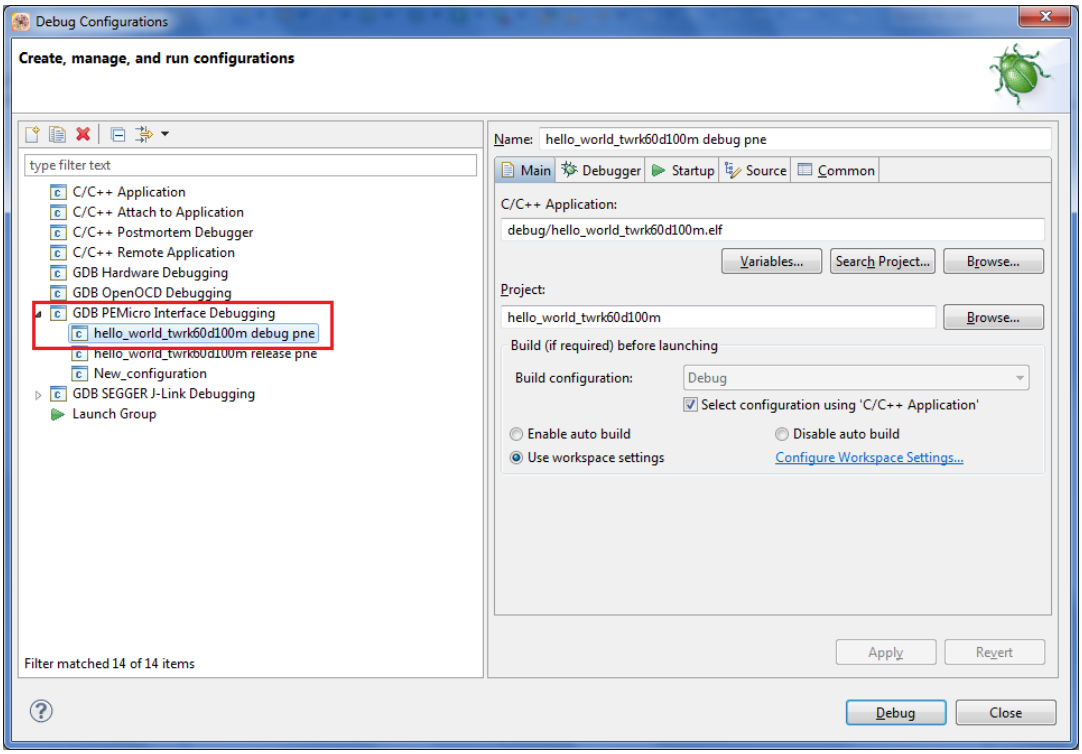

Figure 54: Selection of debug configuration in Debug Configuration dialog box

Freescale Confidential Proprietary, Nondisclosure Agreement Required

c. In the debugger setup, verify that C/C++ Application and Project are set to the correct path. The C/C++ Application path should be debug/<application name>.elf and the Project should be the target project name.

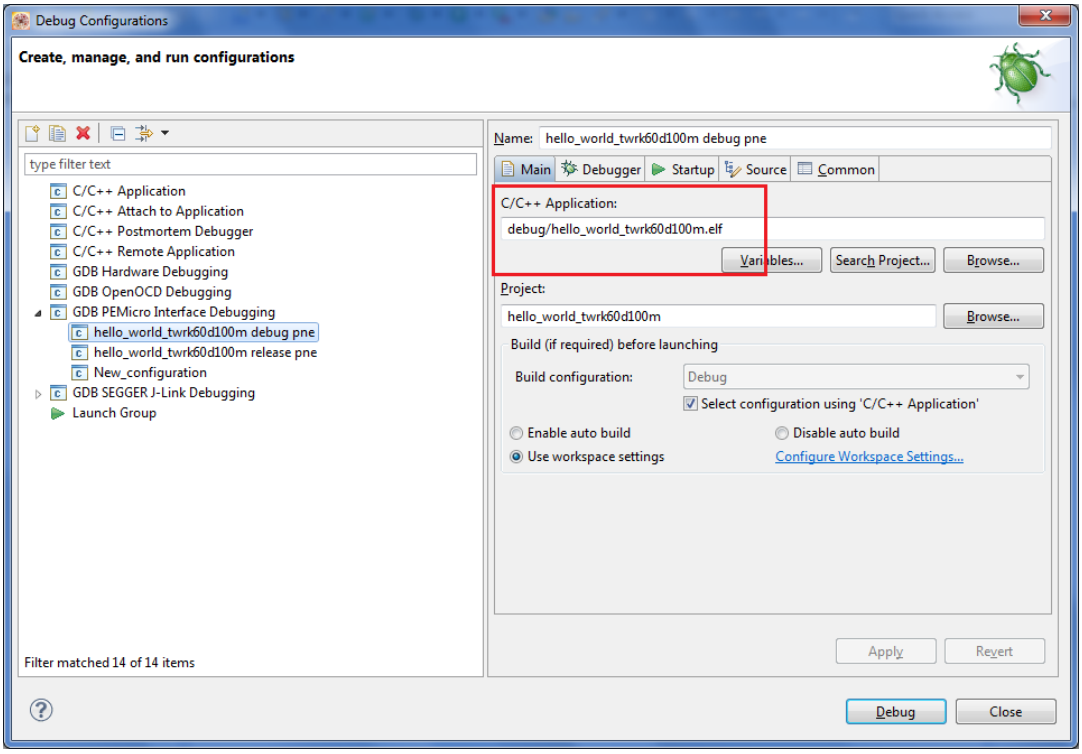

Figure 55: TrueSTUDIO debugger configuration

Freescale Confidential Proprietary, Nondisclosure Agreement Required

d. After verifying the debugger configurations are correct, click the “Debug” button.

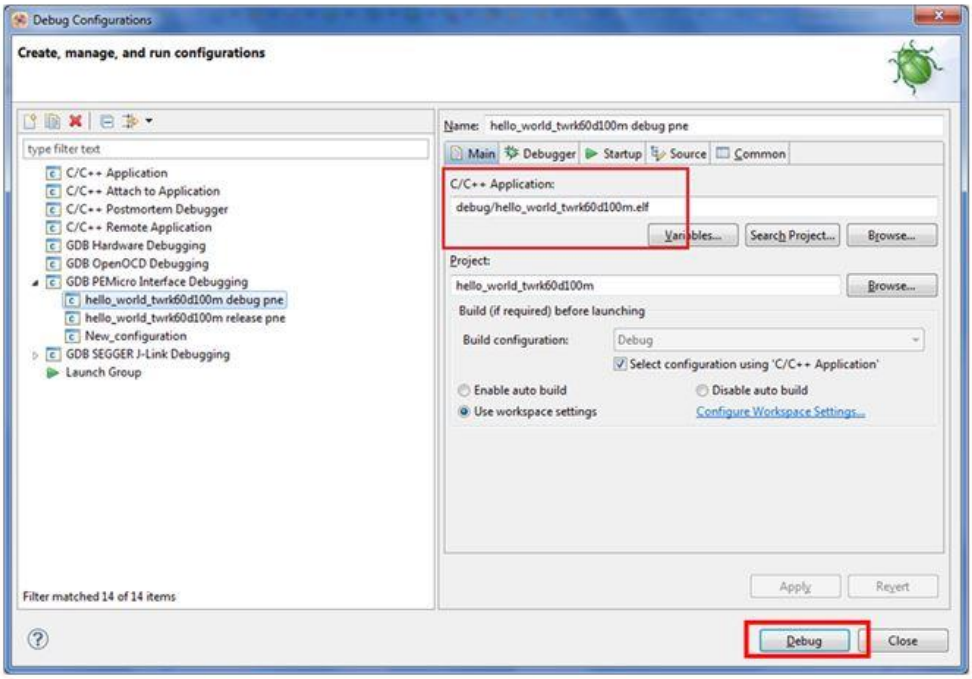

Figure 56: Debug button in Debug Configurations dialog box

Freescale Confidential Proprietary, Nondisclosure Agreement Required

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

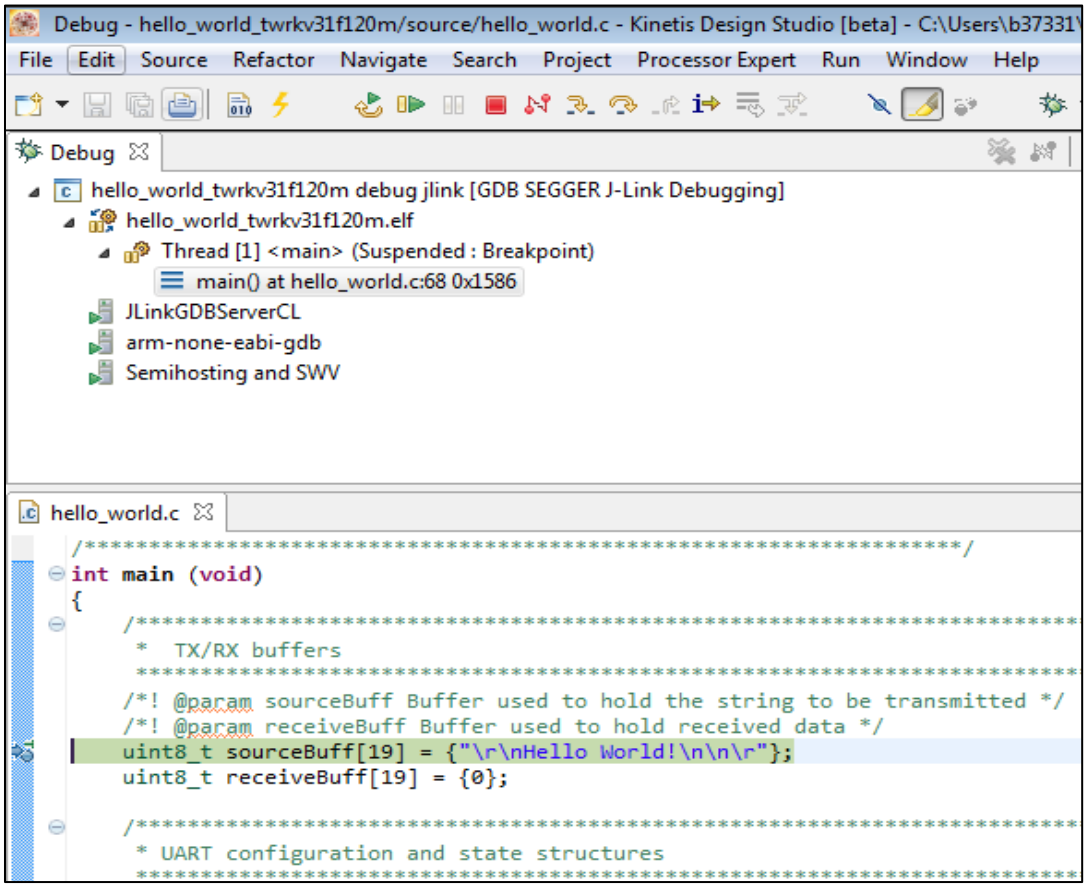

Figure 57: Stop at main() when run debugging

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

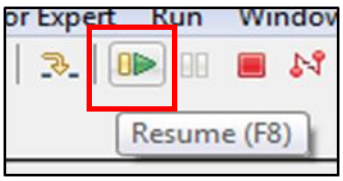

Figure 58: Resume button

6. The hello_world application should now be running and the banner shown in Figure 9 should be displayed on the terminal.

Freescale Confidential Proprietary, Nondisclosure Agreement Required

9 Revision history

This table summarizes the document.

Revision history	
Location	**Change description**
Entire document	Initial release

Freescale Confidential Proprietary, Nondisclosure Agreement Required

Freescale Confidential Proprietary, Nondisclosure Agreement Required

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. The ARM Powered Logo is a trademark of ARM Limited. ARM and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

© 2014 Freescale Semiconductor, Inc.

Document Number: KSDKGSUG
Rev. 0
10/2014