Freescale Semiconductor, Inc.

Document Number: KSDKK22UG
User's Guide
Rev. 1.0.0, 07/2014

Kinetis SDK K22 User's Guide

1 Introduction

This document describes the hardware and software environment setup for the Kinetis SDK (KSDK) for first time Kinetis SDK users. It also explains how to build and run demo applications provided in the KSDK release package.

2 Overview

2.1 Kinetis SDK

Kinetis SDK (KSDK) is a Software Development Kit that provides comprehensive software support for the core and peripherals of all Freescale devices with Cortex®-M core. 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 to highlight the main features of targeted SoCs. It also contains the latest available RTOS kernels. Additionally, KSDK integrates the TCP/IP stack and File system. Therefore, it is necessary to add these integrated software solutions.

Contents

2.2 Hardware requirement

- TWR-K22F120M Tower System module or Freescale Freedom FRDM-K22F platform
- USB A to micro AB cable
- Personal Computer

2.3 Toolchain requirement

- IAR embedded Workbench version 7.20.2 or later is required.
- ARM GCC 4.8.3 2014q1
- Keil MDK 5.11
- Kinetis Design Studio (KDS) v 1.0.1

3 Hardware Configurations

This section describes how to set up the TWR-K22F120M Tower System module and the Freescale Freedom FRDM-K22F platform for the KSDK application.

3.1 TWR-K22F120M Tower System module introduction

3.1.1 TWR-K22F120M Tower System module features

- K22FN512VDC12 MCU (120 MHz, 512 KB Flash, 128 KB RAM, 144 MAPBGA package)
- Dual-role USB interface with Micro-AB USB connector
- Onboard debug circuit: K20DX128VFM5 open (OpenSDA) with virtual serial port
- Three-axis accelerometer (MMA8451Q)
- Four (4) user-controllable LEDs
- One (1) user-controllable RGB LED
- Two (2) user-pushbutton switches for GPIO interrupts (SW1/SW3)
- One Potentiometer
- Independent, battery-operated power supply for Real Time Clock (RTC) and tamper detection modules

3.1.2 TWR-K22F120M Tower System module first look

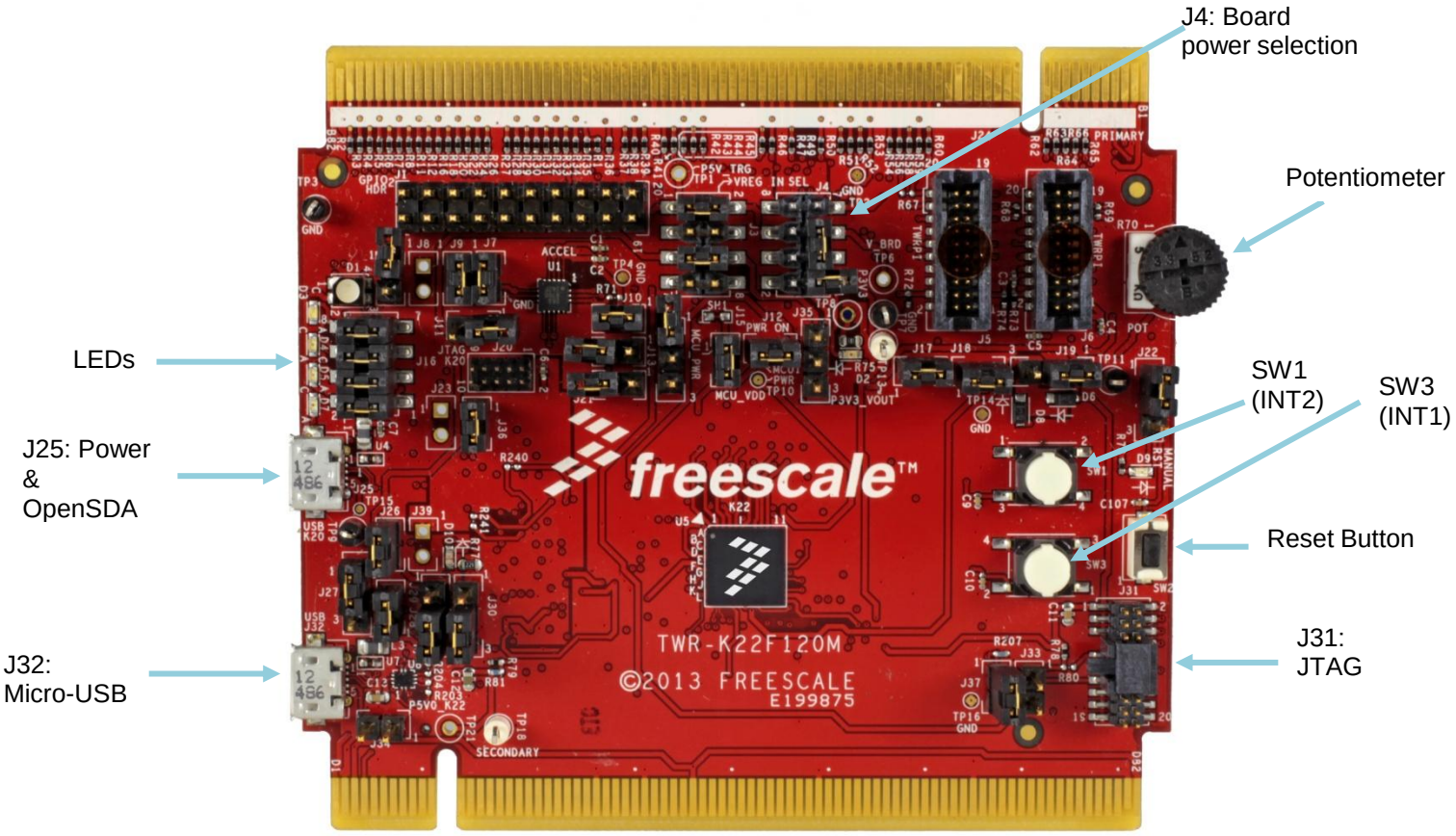

Figure 1 Front side of TWR-K22F120M Tower System module

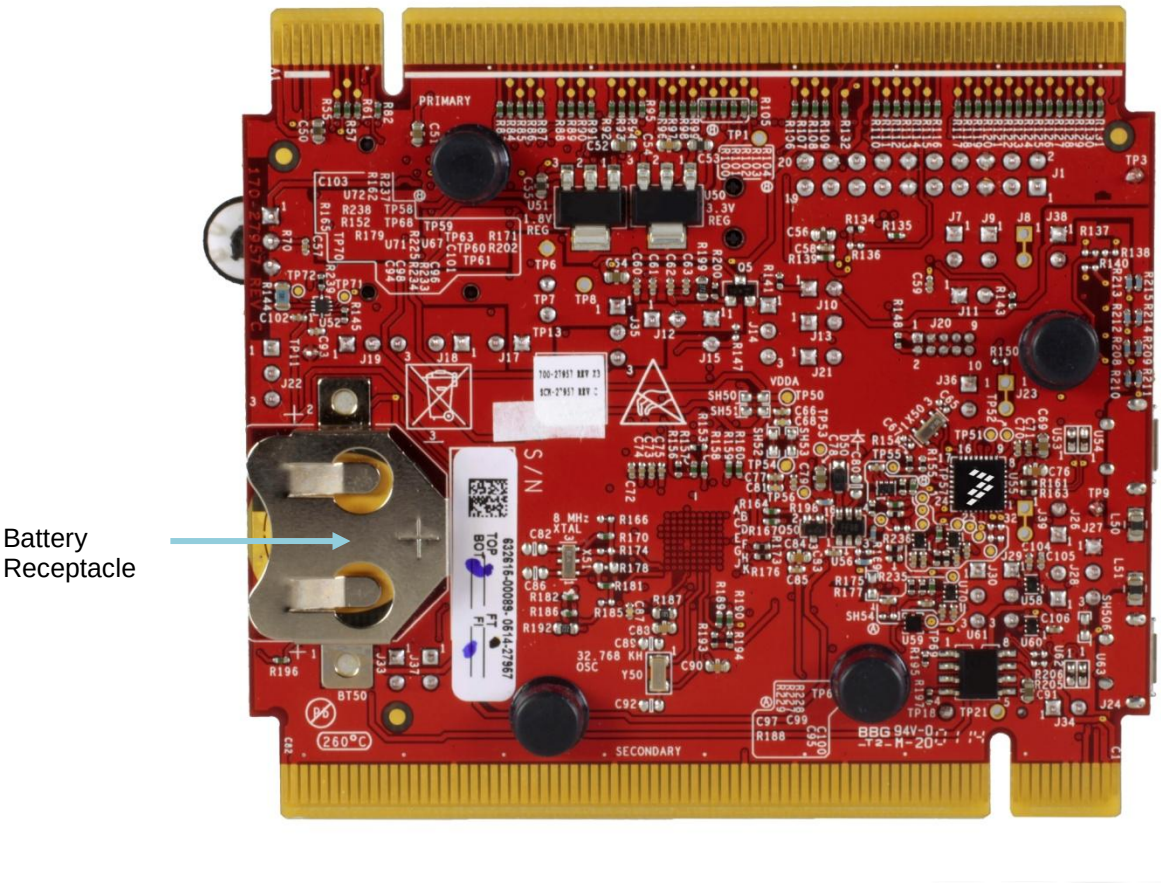

Figure 2 Back side of TWR-K22F120M Tower System module

3.1.3 TWR-K22F120M Tower System module jumper settings

Configure the jumper settings as indicated here. For detailed jumper options and headers, see the *TWR-K22F120M Tower System module User's Guide*.

Table 1 TWR-K22F120M Tower System module jumper settings

Option	Jumper	Setting	Description
3.3 V Voltage Regulator Input Selector (VREG In Selector)	J3	5-6	Selects the K22 USB (P5V0_K22_USB) as the source voltage of the VREGIN pin on the K22 device.
		1-2	Selects the open SDA USB (P5V_TRG_SDA) as the source voltage of the System Voltage Regulator.
Board Power Selector	J4	3-5	Connect 3.3 V onboard regulator output (P3V3) to onboard supply (V_BRD)
MCU Power connection	J15	ON	Connect onboard 3.3 V or 1.8 V supply (V_BRD) to MCU VDD
MCU Power VDDA for current measurement	J12	ON	Connect MCU_PWR (3.3 V or 1.8 V) to VDDA and VREFH

VBAT Power Source	J19	1-2	Connect VBAT to on-board 3.3 V or 1.8 V supply
LED connections	J2	ON	Connect V_BRD to LEDs
	J16	1-2	Connect PTD4 to Yellow/Green LED (D7)
		3-4	Connect PTD5 to Yellow LED (D5)
		5-6	Connect PTD6 to Orange LED (D4)
		7-8	Connect PTD7 to Blue LED (D3)
OpenSDA (K20) UART Selection	J29	2-3	Connect the OpenSDA RX buffer to UART1_RX
	J30	2-3	Connect the OpenSDA TX buffer to UART1_TX

3.2 Freescale Freedom FRDM-K22F platform introduction

3.2.1 Freescale Freedom FRDM-K22F platform features

- K22FN512VDC12 MCU (120 MHz, 512 KB Flash, 128 KB RAM, 144 MAPBGA package)
- Dual-role USB interface with Micro-AB USB connector
- Onboard debug circuit: K20DX128VFM5 open (OpenSDA) with virtual serial port
- Three-axis combined accelerometer and magnetometer (FXOS8700CQ)
- One (1) user-controllable RGB LED
- Two (2) user pushbutton switches for GPIO interrupts (SW2/SW3)
- One (1) audio code (SGTL5000) with line-in and audio output jacks
- One (1) visible light sensor
- One (1) micro SD card receptacle
- Connections for RF module support and Bluetooth support
- Independent, battery-operated power supply for Real Time Clock (RTC)

3.2.2 Freescale Freedom FRDM-K22F platform first look

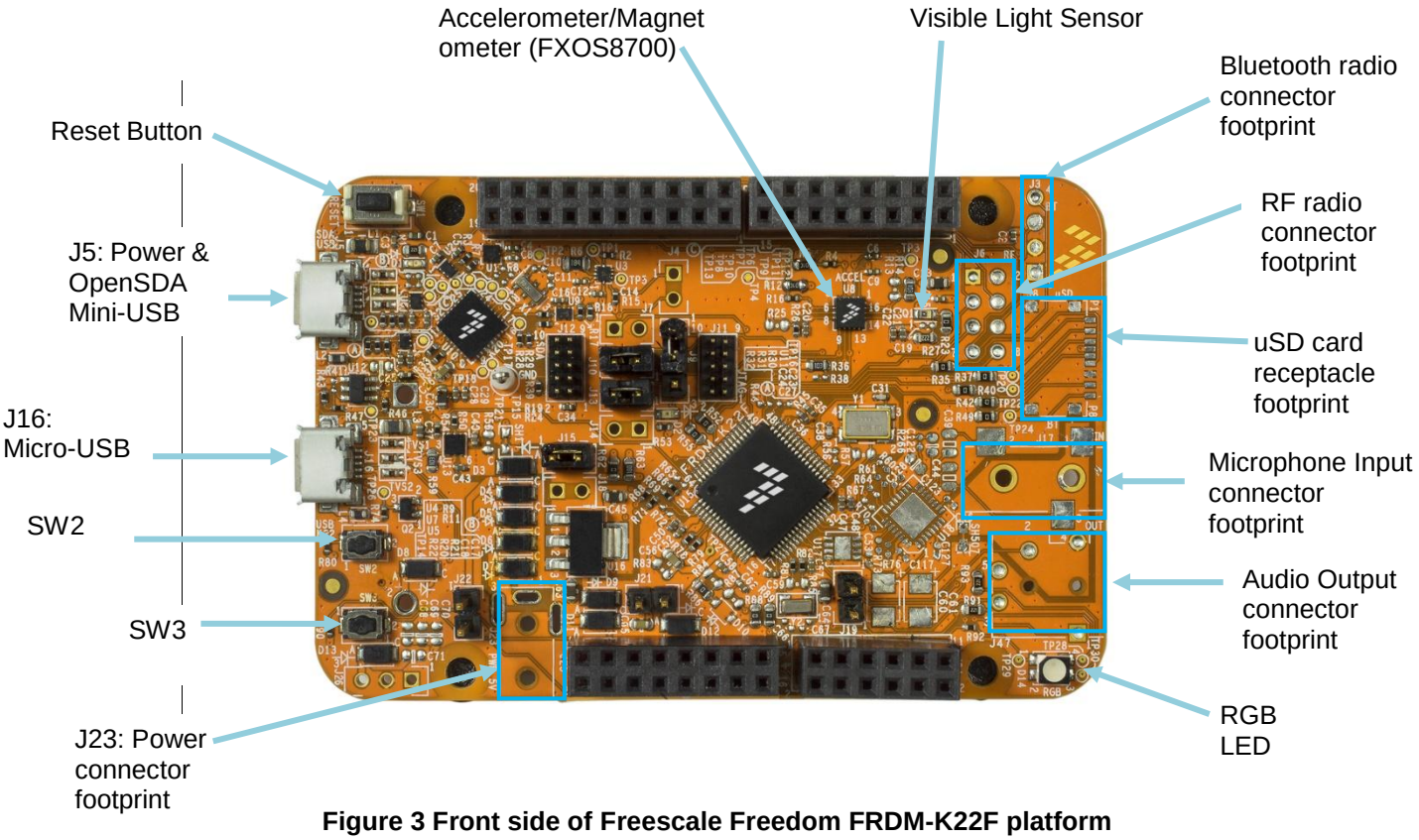

Figure 3 Front side of Freescale Freedom FRDM-K22F platform

Table 2 Freescale Freedom FRDM-K22F Default jumper settings

Option	Jumper	Setting	Description
MCU power	J15	ON	Allows the connection or disconnection of power to the MK22FN512VLH12 device for current measurements. Note that a 0 Ohm and a 10 Ohm resistor are connected in parallel with the jumper (and each other) to allow for other power measurement options.
SWD CLK Isolation Jumper	J10	ON	Allows the OpenSDA circuitry to connect to the SWD CLK pin.
SWD DIO Isolation Jumper	J13	ON	Allows the OpenSDA circuitry to connect to the SWD DIO pin.
Reset Bypass Jumper	J9	1-2	Allows the OpenSDA circuitry to control the reset pin of the target MCU. Connect 2-3 when OpenSDA is not powered.
VBAT Jumper	J21	OFF	Allows the VBAT pin to be powered by the same source as the main MCU power. (Note that there is a shorting trace on this jumper, so even though it is not populated by default, the VBAT pin is connected to the main power source of the MCU.)

4 Build and Run the KSDK Demo Applications using IAR

This section describes the steps required to configure the IAR Embedded Workbench to build, run, and debug demo applications and necessary driver libraries provided in the Freescale KSDK. The hello_world application targeted for the TWR-K22F120M Tower System hardware platform is used as an example.

4.1 Building the platform driver library in IAR

Before building and debugging demo applications in KSDK, the driver library project should be built to generate the library archive platform_lib.a. This library contains all HAL and peripheral driver functions which are device-specific. Therefore, each device has its own library (platform.a). The platform library is prebuilt. It should not be necessary to build the library after initially downloading the KSDK. However, if it is necessary, follow these directions to rebuild the platform library.

Open the workspace file in IAR. The platform driver library project is located in this folder:

<Install_dir>/lib/ksdk_platform_lib/iar/<device_name>

The workspace file is named lib.eww:

<Install_dir>/lib/ksdk_platform_lib/iar/<device_name>lib.eww

The project file is named platform_lib.ewp:

<Install_dir>/lib/ksdk_platform_lib/iar/<device_name>/platform_lib.ewp

To build the platform driver library for the K22, open this workspace file in IAR:

<Install_dir>/lib/ksdk_platform_lib/iar/K22F51212/lib.eww

In the IAR Embedded Workbench project file, two compiler/linker configurations (build "targets") are supported:

- 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.

Choose the appropriate build target: "Debug" or "Release", then click the "Make button" (highlighted by a red rectangle in this figure):

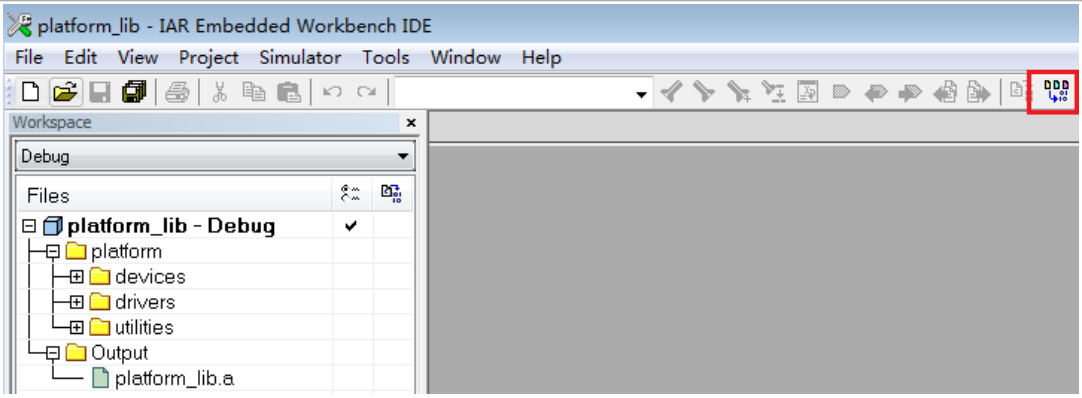

Figure 4 Platform driver library build

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

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

4.2 Build a demo application

The KSDK demo applications utilize a prebuilt linkable library to compile the necessary functions. It is required that this library be built before compiling and downloading. To check if the library has been generated, verify that the platform_lib.a file is located in <Install_dir>/lib/ksdk_platform_lib/<toolchain>/<device_name>/<build> where 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, the platform_lib.a library should be in this folder:

<Install_dir>/lib/ksd_platform_lib/iar/K22F51212/debug

Next, continue by opening the demo application project. Demo applications workspace files are located in this folder:

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

The hello_world application is used as an example. The IAR workspace file is located in this folder:

<install_dir>/demos/hello_world/iar/twrk22f120m/hello_world.eww

To build a demo application project, click the “Make button”, which is highlighted by a red square in the this figure:

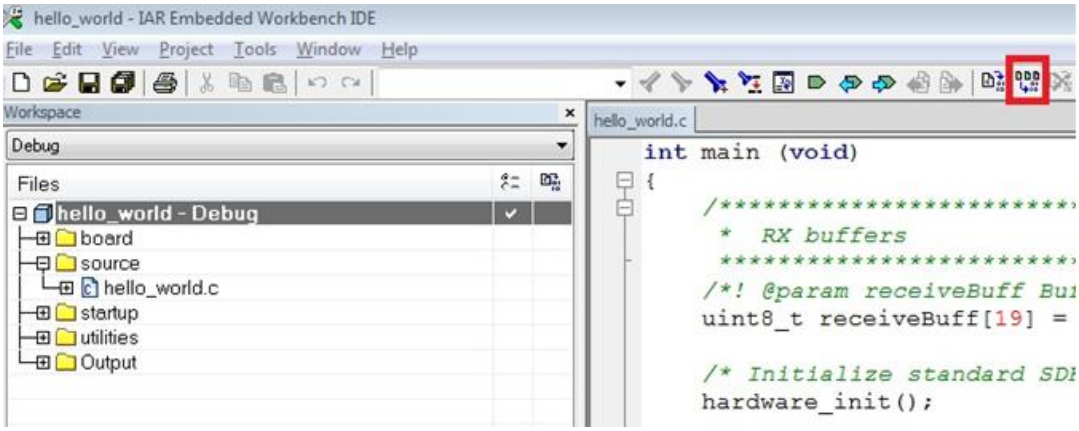

Figure 5 Build the hello_world demo application

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

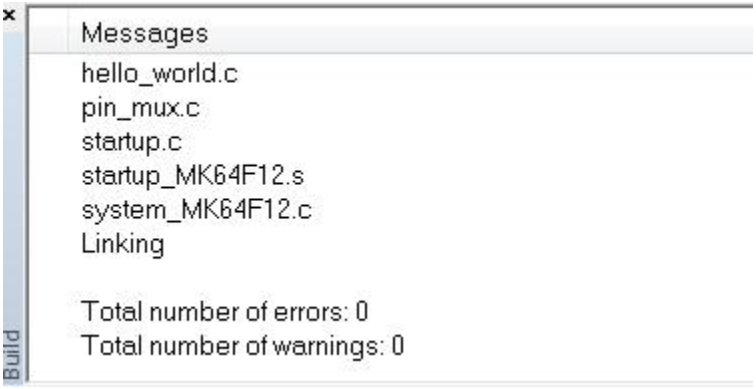

Figure 6 Build hello_world demo successfully

4.3 Run a demo application

To download and run the application, perform these actions:

1. Connect the K22 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 OpenSDA serial port number. Configure the terminal with these settings:
    a. 115200 baud rate
    b. No parity
    c. 8 data bits
    d. 1 stop bit

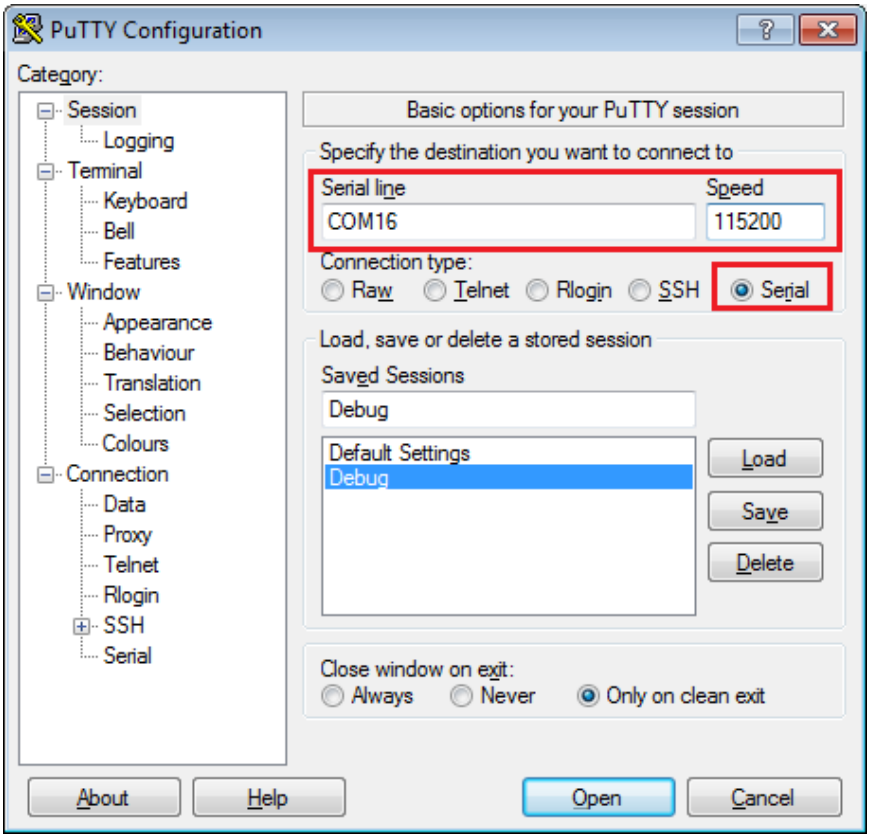

Figure 7 Terminal (PuTTY) configurations

3. Ensure that the debugger configuration is correct in the project options.
    a. The flash loader must be selected to support downloading the binary to internal Flash.

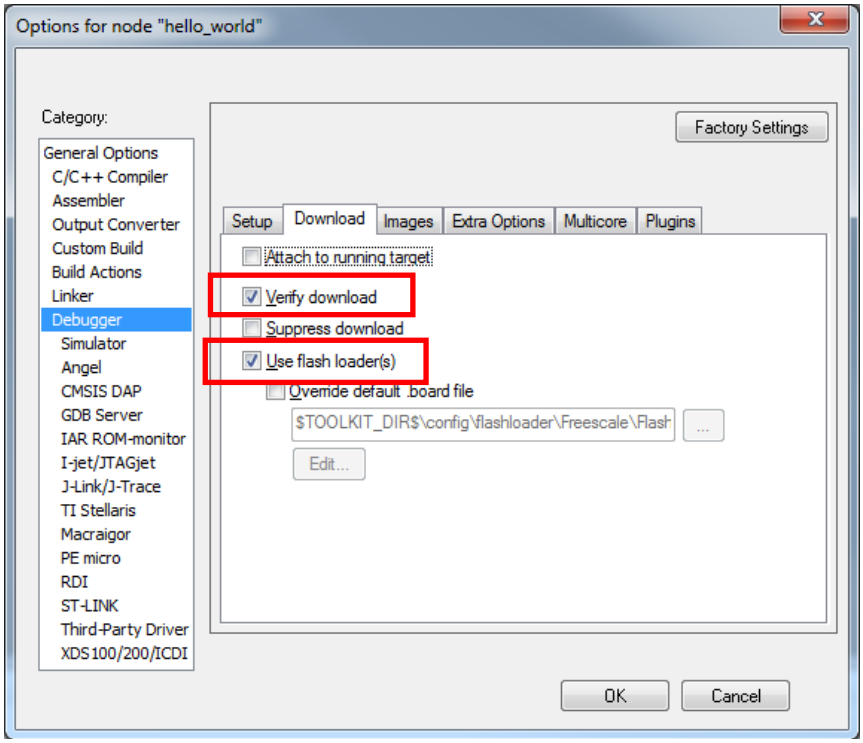

Figure 8 Flash loader configurations

b. Select the appropriate debugger in the debugger setup.

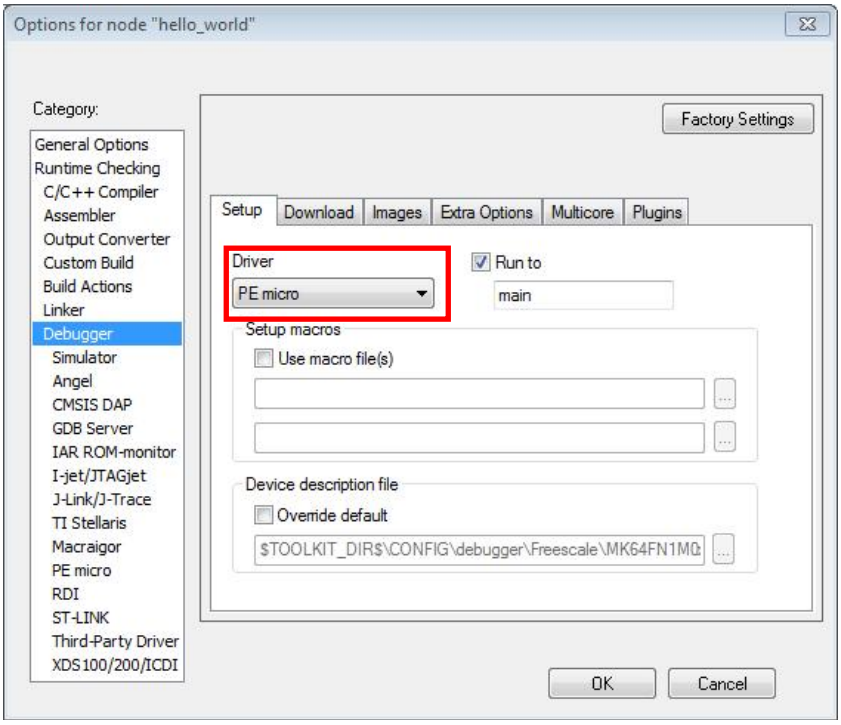

Figure 9 Debugger configurations for TWR-K22F120M Tower System module

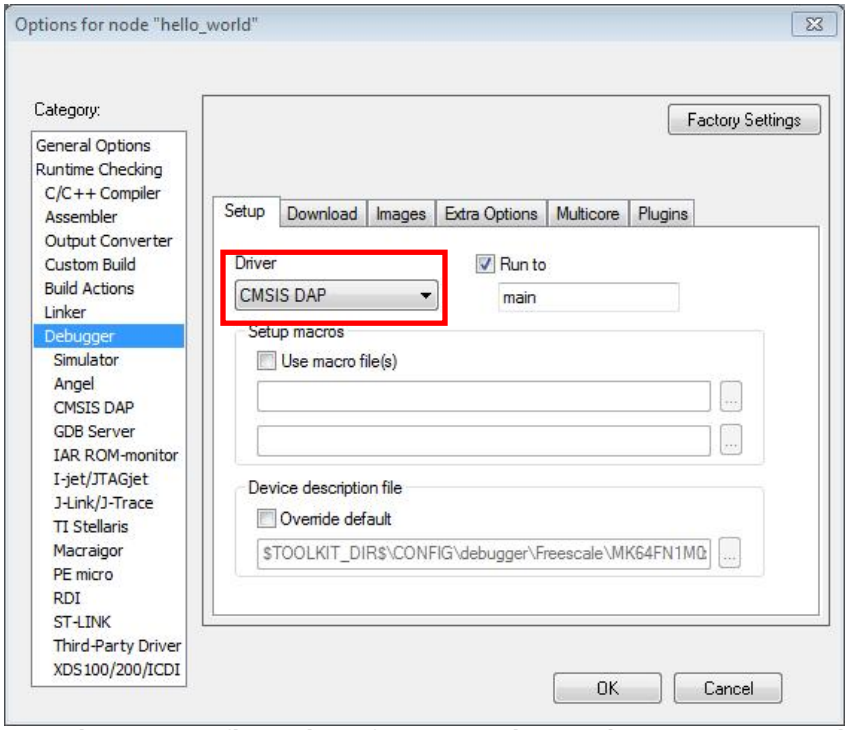

Figure 10 Debugger configurations for Freescale Freedom FRDM-K22F platform

c. SWD should be configured as the debugger interface in the debugger specific category.

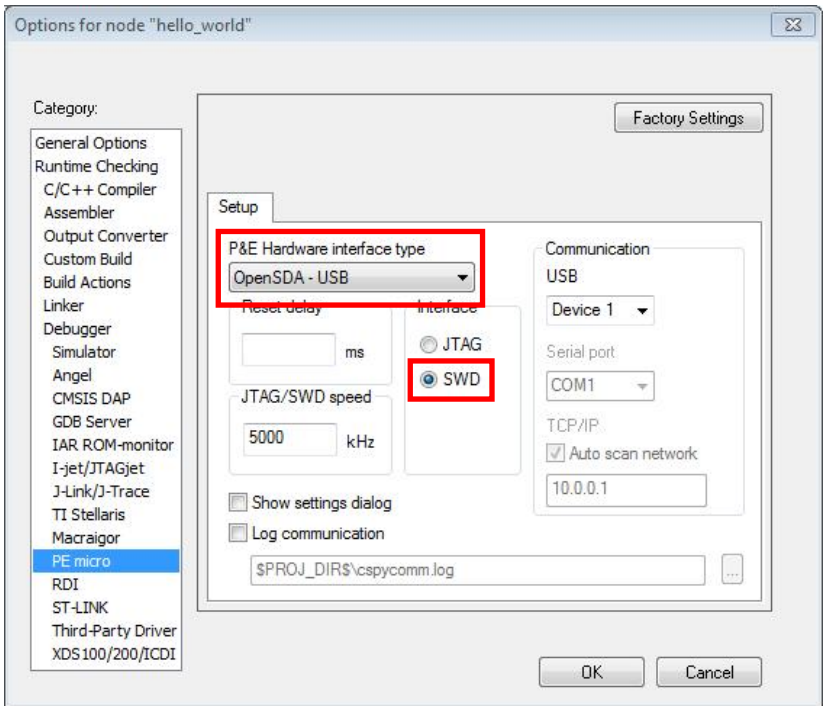

Figure 11 Debugger configurations for TWR-K22F120M Tower System module (PE micro)

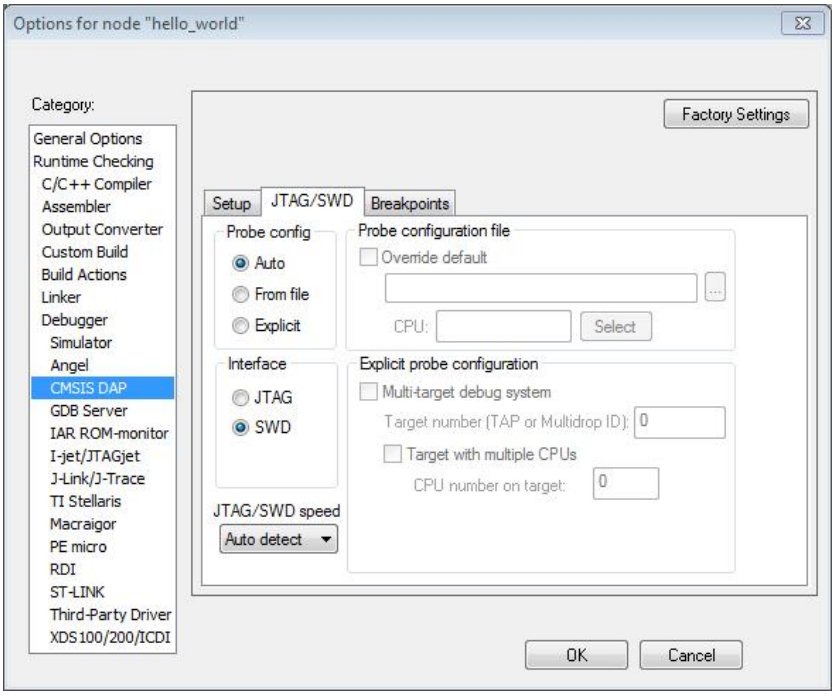

Figure 12 Debugger configurations for Freescale Freedom FRDM-K22F platform (CMSIS-DAP)

4. When the application is built, press the “Download and Debug button” to download the application to the target.

Figure 13 Download and Debug button

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

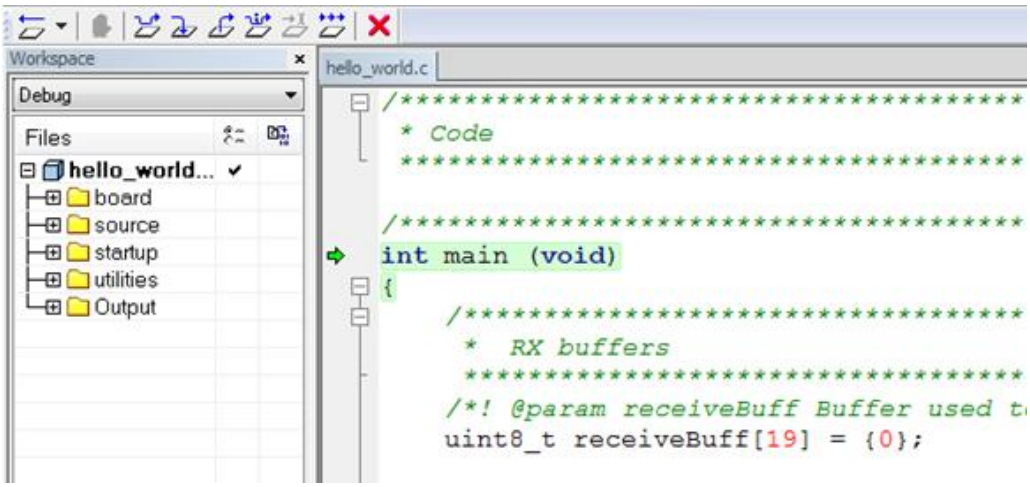

Figure 14 Stop at main() when run debugging

Now run the code by clicking on the “go button” to start the application:

Figure 15 Go button

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

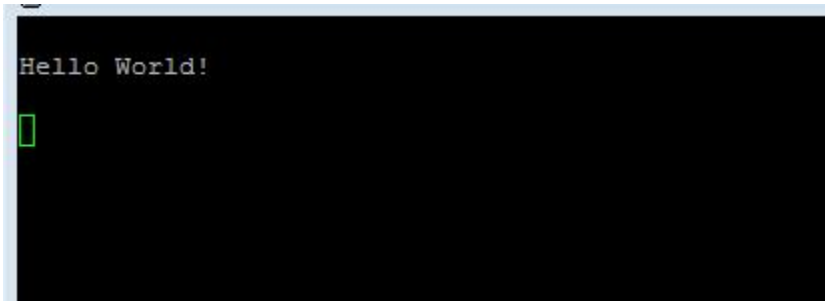

Figure 16 Main prompt of the hello_world demo

5 Build and Run the KSDK Demo Applications using ARM GCC

This section describes the steps required to configure the ARM GCC toolchain 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-K22F120M Tower System module hardware platform is used as an example.

5.1 Environment setup

5.1.1 Install GCC ARM v4.8.3 2014q1 embedded toolchain

1. Download the Windows installer.
2. Install the toolchain in the /Program Files/ location, which is usually on your "C:\" drive.

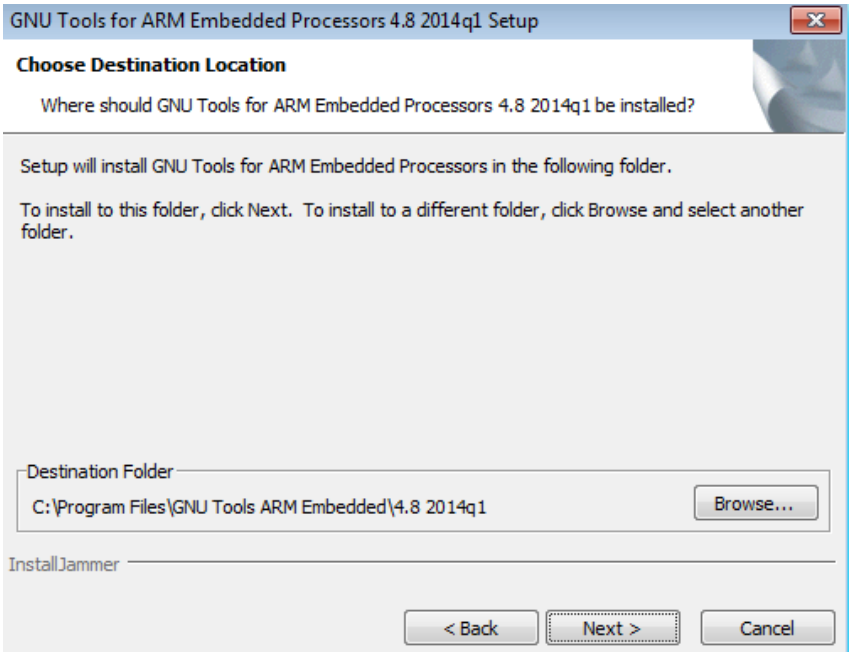

Figure 17 Install GCC ARM embedded toolchain

5.1.2 Install MinGW and MSYS

1. Download the MinGW installer, which is located here.
2. Run mingw-get-setup.exe and select the installation path, such as: C:/MINGW.
3. Select the mingw32-base and the msys-base under the Basic Setup as shown in Figure 18 MinGW Installer.

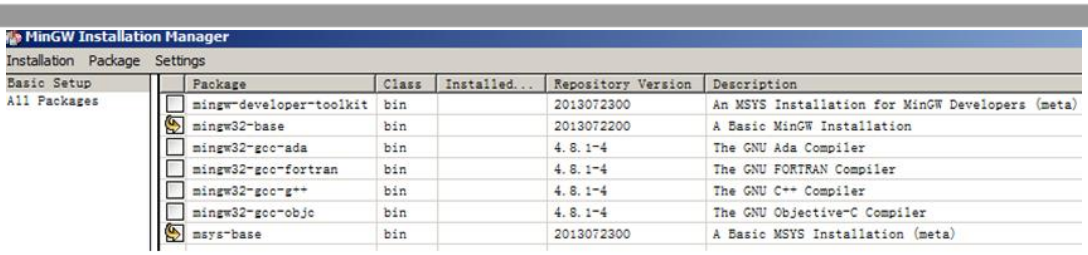

Figure 18 MinGW Installer

4. Click "Apply Changes" from the "Installation" menu to install packages, as shown here.

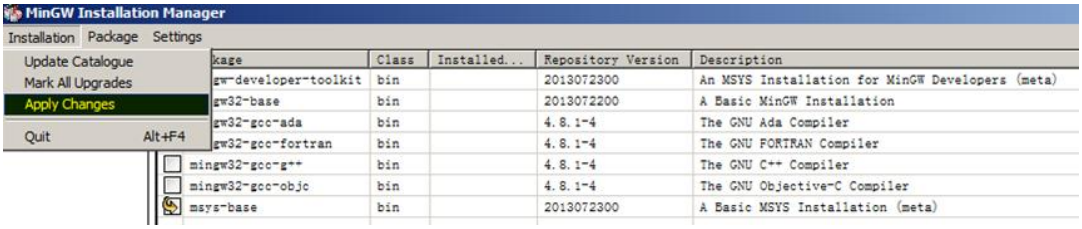

Figure 19 MinGW Installer apply change

5.1.3 Configure system environment

1. Update the system environment variable "Path" to include the MINGW installation folder, such as the <drive>\MINGW\msys\1.0\bin;<drive>:\MINGW\bin.

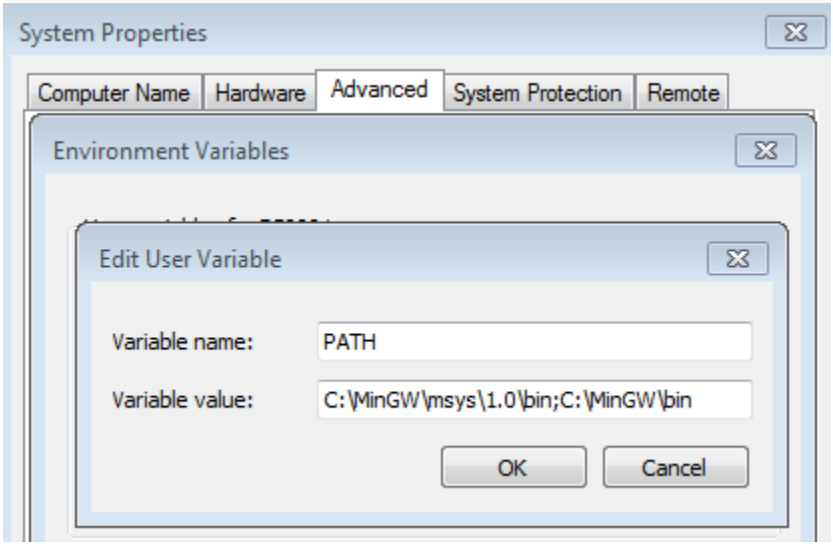

Figure 20 Environment variable update

2. Run the GCC Command prompt in Start->All Programs-> GNU Tools ARM Embedded 4.8.3 2014q1.

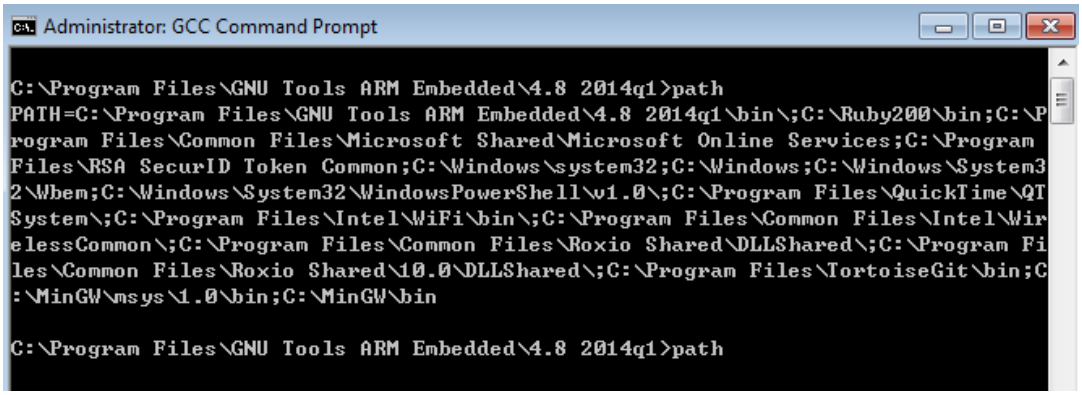

Figure 21 PATH environment

3. When using a Windows command line, add the environment variable, ARMGCC_DIR, which is also the short name of the ARM GCC installation path.

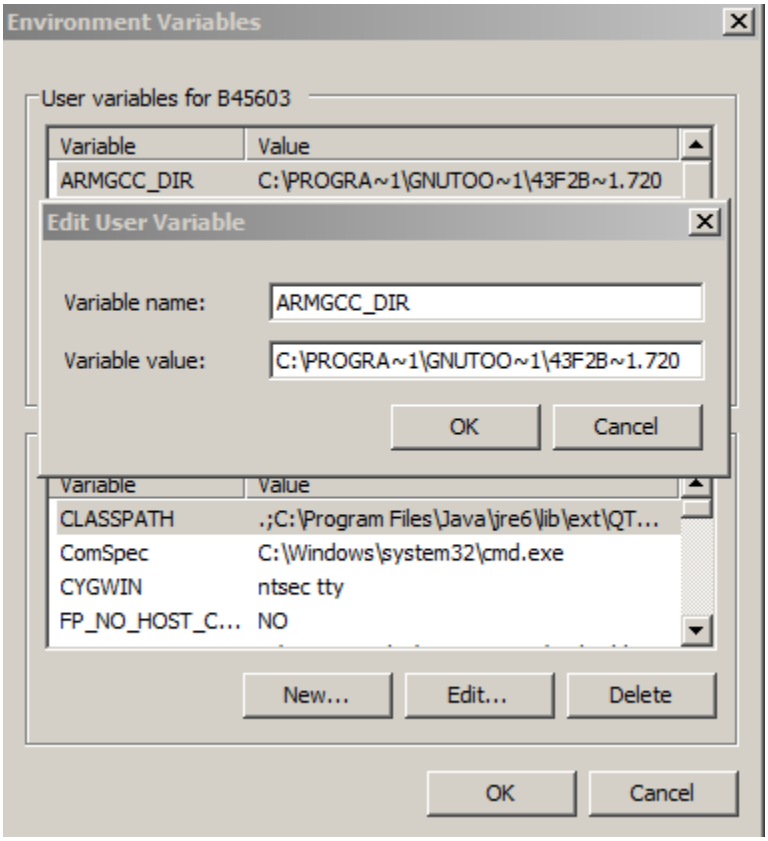

Figure 22 Add environment variable

4. When using GIT Bash or Cygwin, set the environment variable to "export ARMGCC_DIR=C:/PROGRA~/GNUTOO~1/4298B~1.820".

Note

Use a forward slash '/' as a separator.

When using the KDS GCC toolchain, add the system environment variable, KDSGCC_DIR, which is also the path of KDS GCC toolchain. Use a short name and run the “mingw32-make toolchain=kdsgcc”.

>/mk/common.mk

5.2 Building the platform driver library in ARM GCC

Before building and debugging any demo applications in the SDK, the driver library project must be built to generate the necessary library archives (platform_lib.a). This library contains all binary codes for the HAL and peripheral drivers specific to the chip and each device has its own library archive.

To build the platform library, change the current directory in the ARM GCC command prompt to:
<install_dir>/lib/ksdk_platform_lib/gcc/<device_name>/

Using K22 as an example, the correct path would be:
<install_dir>/lib/ksdk_platform_lib/gcc/K22F51212.

Once the command prompt path has been correctly set, run the command “mingw32-make build=debug” to build the debug library or “mingw32-make build=release” to build the release library.

Once the build has successfully completed, the platform_lib.a file (library archive) is located in these directories:

Debug - <install_dir>/lib/ksdk_platform_lib/gcc/<device_name>/Debug

Release - <install_dir>/lib/ksdk_platform_lib/gcc/<device_name>/Release

5.3 Build a demo application

The KSDK demo applications use a prebuilt linkable library to compile in the necessary functions. Therefore, it is required that this library be built before compiling and downloading. To check that this library has been generated, verify that the platform_lib.a file resides in <Install_dir>/lib/ksdk_platform_lib/<toolchain>/<device_name>/<build>

where the build is the desired build, which is either Debug or Release. For example, if the desired project to run is the Debug version of the hello_world demo application, it is required that the platform_lib.a library exists in this folder:

<Install_dir>/lib/ksd_platform_lib/gcc/K22F51212/debug

Continue by changing the directory of your ARM GCC compiler to the directory where the application makefile resides. This location is
<install_dir>/demos/<demo_name>/<toolchain>/<board_name>/Makefile

The hello_world application is used as an example. The GCC Makefile is located in:
<install_dir>/demos/hello_world/gcc/twrk22f120m/

To build a demo application project, open an ARM GCC command prompt and change the directory to the location where the desired Makefile is stored (listed above). Then execute the command "mingw32-make build=<target_build> target=<target_mem_location>, where build and target can be one of these:

Table 3 Build and Target Options

Build options	Target options
Debug	Flash
Release	SRAM

When the build is complete, the GCC command prompt appears as shown:

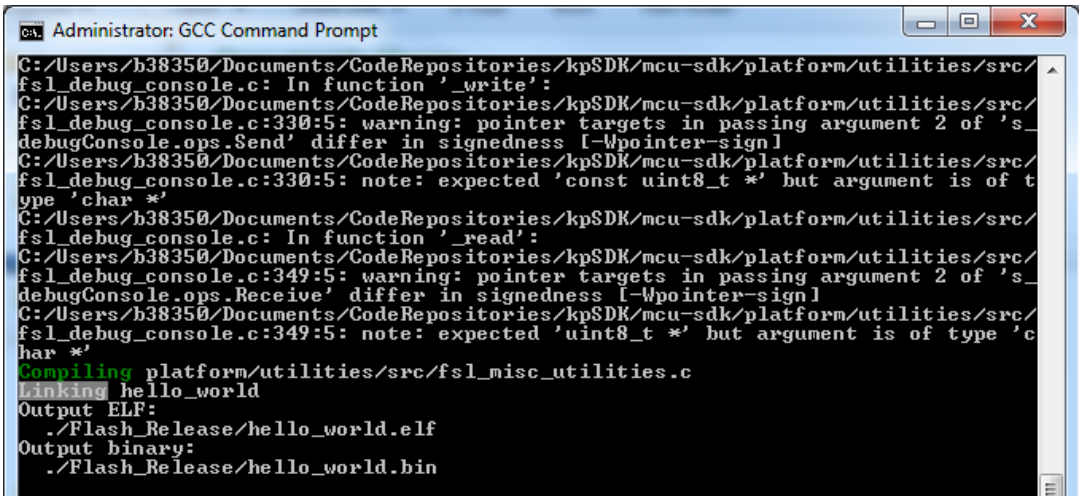

Figure 23 hello_world demo successfully built

5.4 Run a demo application

To download and run the application, perform these actions:

1. If you do not already have the Segger J-Link software and documentation installed, do this first. Go to the segger.com/jlink-software.html to download the package. Otherwise, continue to step 2.
2. For Freedom platform, remove the OpenSDA isolation jumpers (jumpers J10 and J13 for the Freedom platform, and jumpers J36 and J37 for the Tower System) if they are populated.
3. Connect your J-Link debug pod to the MK22F512 SWD connector (J31) of the Freedom platform.
4. Open the J-Link GDB Server application. After opening this application, modify your connection settings as shown in this figure:

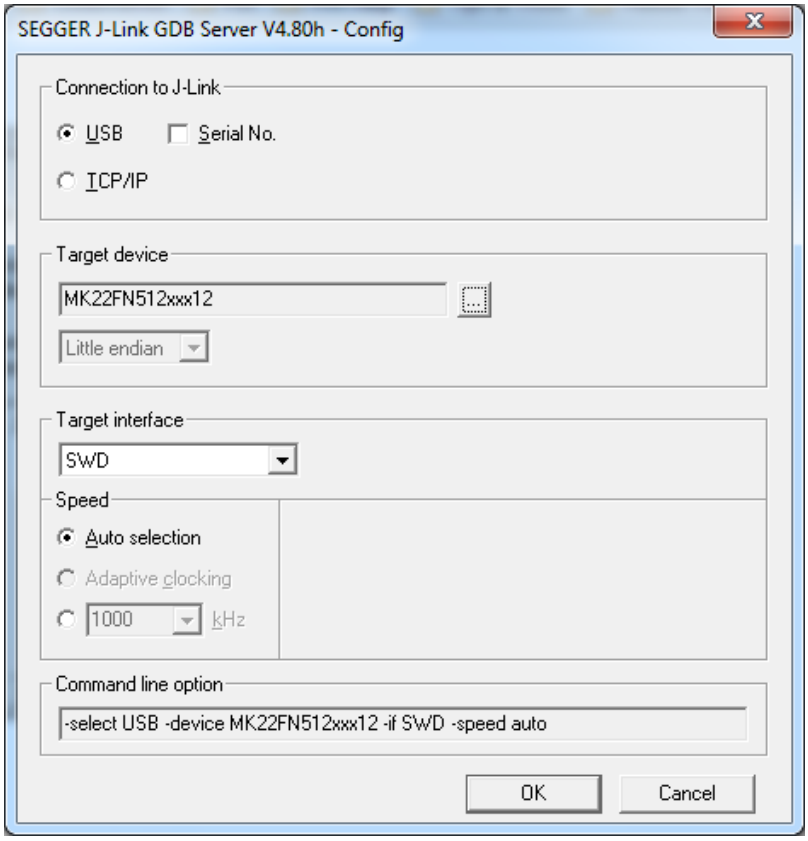

Figure 24 Segger J-Link GDB server configurations

5. Once connected, the screen appears as shown:

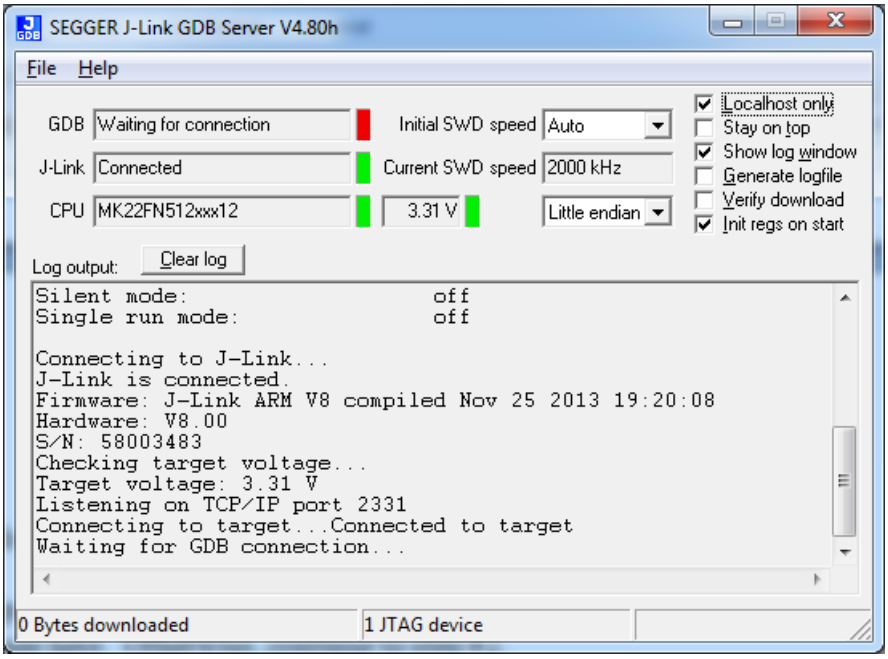

Figure 25 Segger J-Link GDB Server screen after successful connection

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

<SDK root>\demos\hello_world\gcc\twrk22f120m\Flash_Debug.

7. Run the command “arm-none-eabi-gdb <DEMO_NAME>.elf”. For this example, “arm-none-eabi-gdb hello_world.elf”.
8. Run these commands:

“target remote localhost: 2331”

   a. “monitor reset”
   b. “monitor halt”
   c. “load”
   d. “monitor reset”

9. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the OpenSDA serial port number. Configure the terminal with these settings:
    e. 115200 baud rate
    f. No parity
    g. 8 data bits
    h. 1 stop bit

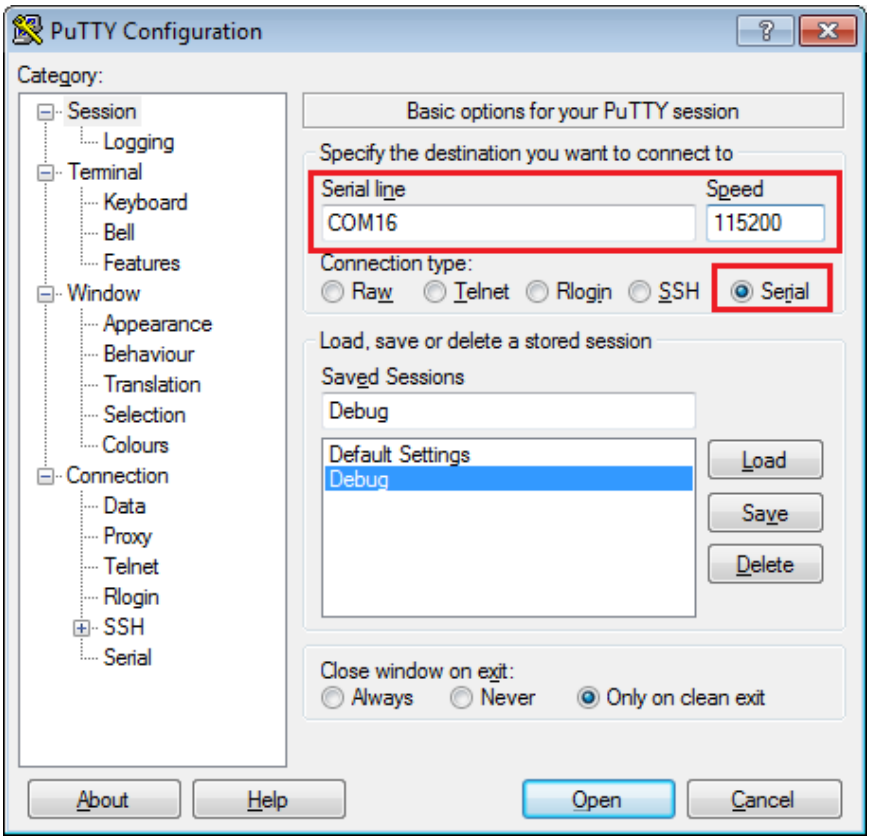

Figure 26 Terminal (PuTTY) configurations

10. The application is now downloaded and connected. You may execute the "monitor go" command to begin the demo application.

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

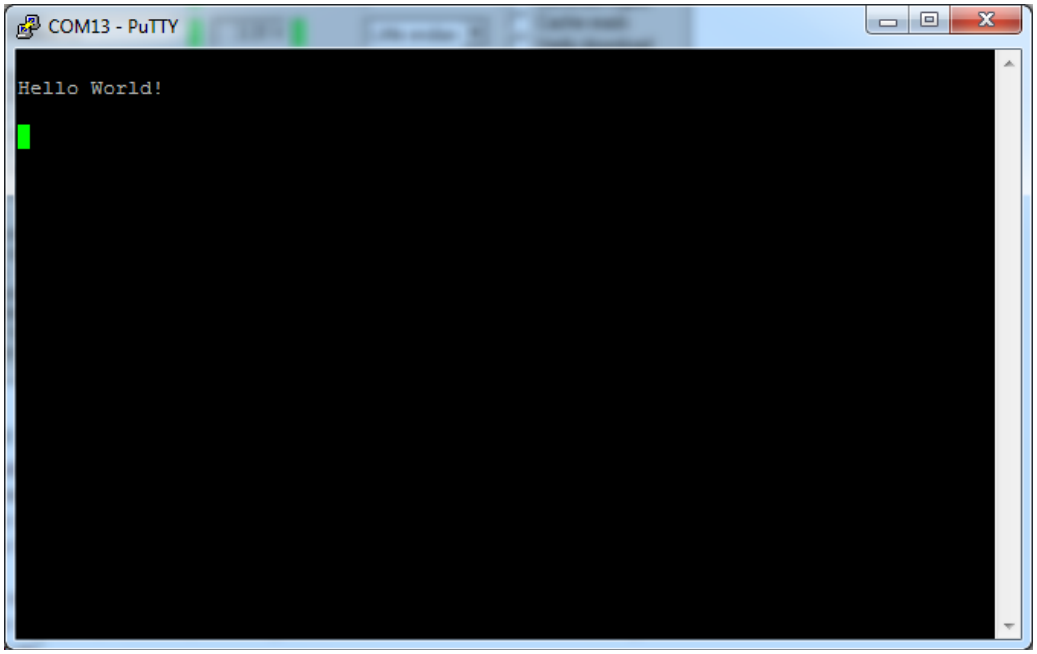

Figure 27 Main prompt of the hello_world demo

6 Build and Run the KSDK Demo Applications using Keil MDK

This section describes the steps required to build, run, and debug demo applications and necessary driver libraries provided in the Freescale KSDK using ARM MDK v5 (Keil version 5). The Kinetis K20 Series Device Support and examples DFP 1.1.0 must be updated. The hello_world demo application is used as an example (TWR-K22F120M Tower System module hardware platform).

NOTE

Before opening the K22 KSDK demos and/or the libraries, the KEIL Device Family Pack for the K22 series should be installed. This software pack can be downloaded from the Freescale MK22FN512xxx12 webpage on KEIL's website at http://www.keil.com/dd2/freescale/mk22fn512xxx12/.

6.1 Building the platform driver library in Keil MDK

Before building and debugging demo applications in the KSDK, the driver library for the target device should be built. The library archive for the Keil demo applications is named **ksdk_platform_lib.lib**. This library contains all HAL and peripheral driver functions which are device-specific. Therefore, each device has its own library (ksdk_platform_lib.lib). The platform library is not prebuilt. Therefore, it is necessary to build the library after initially downloading the KSDK.

Open the hello_world multiproject workspace file (hello_world.uvmpw) in Keil (Project->Open project…). The demo application multiproject workspace is located in this folder:

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

Using the hello_world as an example, the path should be:

<Install_dir>/demos/hello_world/uv4/twrk22f120m/hello_world.uvmpw

To build the platform driver library for the K22, first set the ksdk_platform_lib as the active project, by right-clicking on the ksdk_platform_lib and selecting “Set as Active Project”.

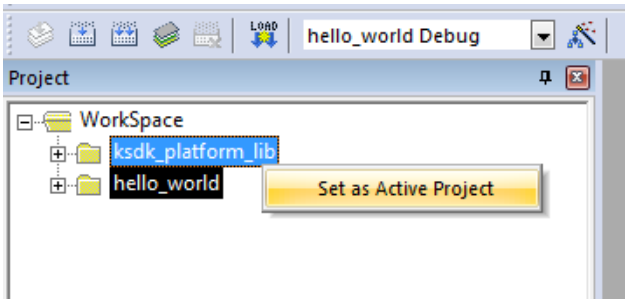

Figure 28 Selection of the ksdk_platform_lib as the active project

In the ksdk_platform_lib project, two compiler/linker configurations (build "targets") are supported:

- 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.

Choose the appropriate build target: "Debug" or "Release" from the drop-down box.

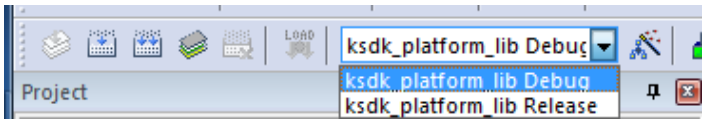

Figure 29 Drop-down selection of the Debug/Release target

Rebuild the project files by left-clicking the "Rebuild button".

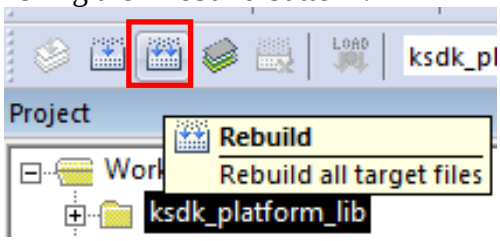

Figure 30 Rebuild all button

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

- Debug - <install_dir>/lib/ksdk_platform_lib<toolchain>/<device_name>/ Debug
- Release - <install_dir>/lib/ksdk_platform_lib/<toolchain>/<device_name>/ Release

6.2 Build a demo application

The KSDK demo applications utilize a prebuilt linkable library to compile the necessary functions. It is required that this library be built before compiling and downloading. To check that this library has been generated, verify that the ksdk_platform_lib.lib file is in <Install_dir>/lib/ksdk_platform_lib/<toolchain>/<device_name>/<build> location, where 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, the platform_lib.a library should be in this folder:

<Install_dir>/lib/ksdk_platform_lib/uv4/K22F51212/debug

Continue by opening the demo application project. Demo applications workspace files are located in this folder:

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

The hello_world application is used as an example. The Keil multi-project workspace file is located in this folder:

<install_dir>/demos/hello_world/uv4/twrk22f120m/hello_world.uvmpw

To build a demo application project, click the "Rebuild button", which is highlighted by a red square in this figure:

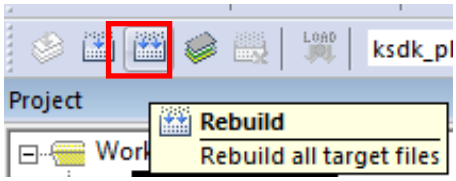

Figure 31 Build the demo application

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

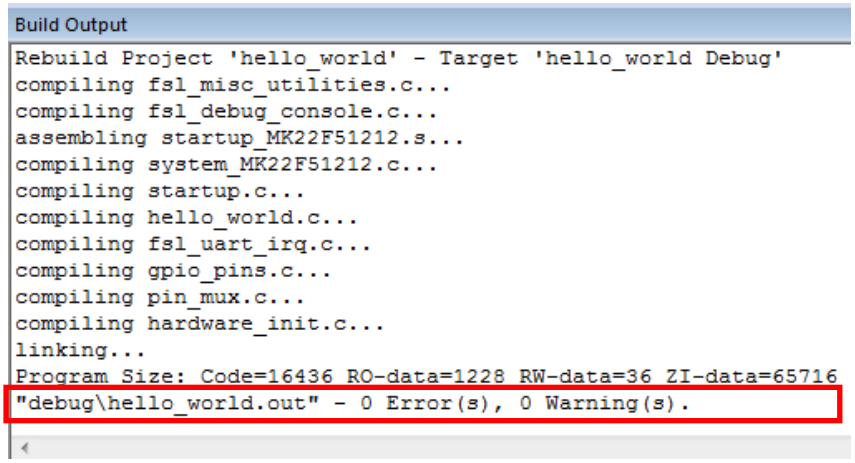

Figure 32 Build Output window upon successful hello_world build

6.3 Run a demo application

To download and run the application, perform these actions:

1. Connect the K22 development board 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 OpenSDA serial port number. Configure the terminal with these settings:
    a. 115200 baud rate
    b. No parity
    c. 8 data bits
    d. 1 stop bit

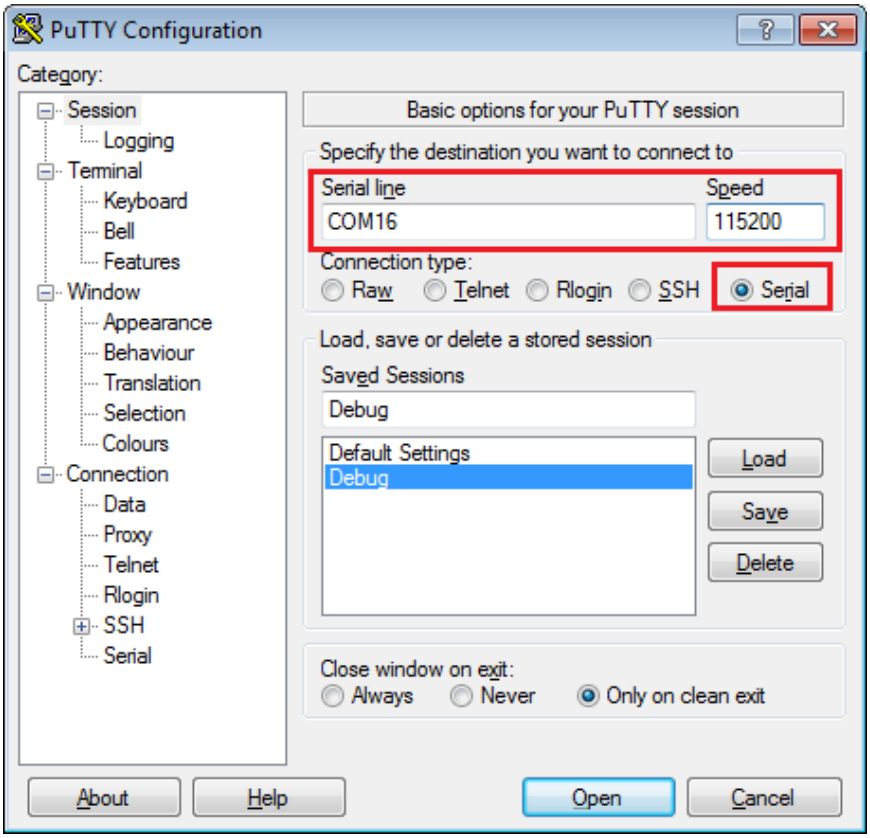

Figure 33 Terminal (PuTTY) configurations

3. Ensure that the debugger configuration is correct in the project options.
    a. To verify the debugger configurations, first open Options for the Target using the Options for Target dialog button

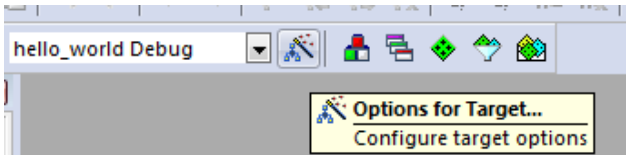

Figure 34 Options for Target dialog button

    b. In the "Options for Target …" dialog box, select the Debug tab and ensure that the simulator is not selected and the correct debug driver is selected. If the target is a TWR-K22F120M Tower System development card, the PEMicro Debugger should be selected. If the target is a Freescale Freedom FRDM-K22F platform development system, the CMSIS-DAP should be selected.

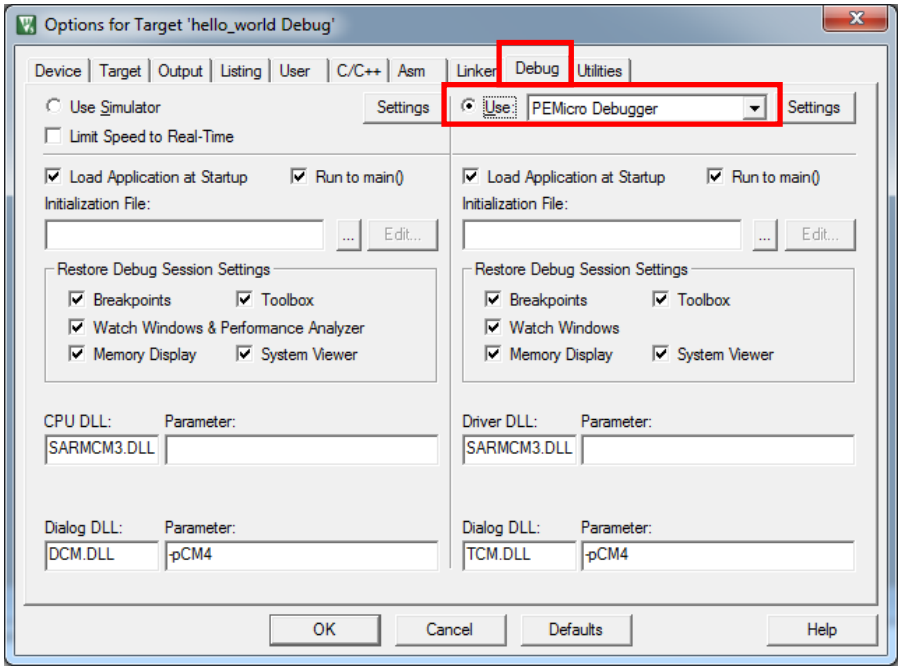

Figure 35 Debug driver selection for TWR-K22F120M Tower System module

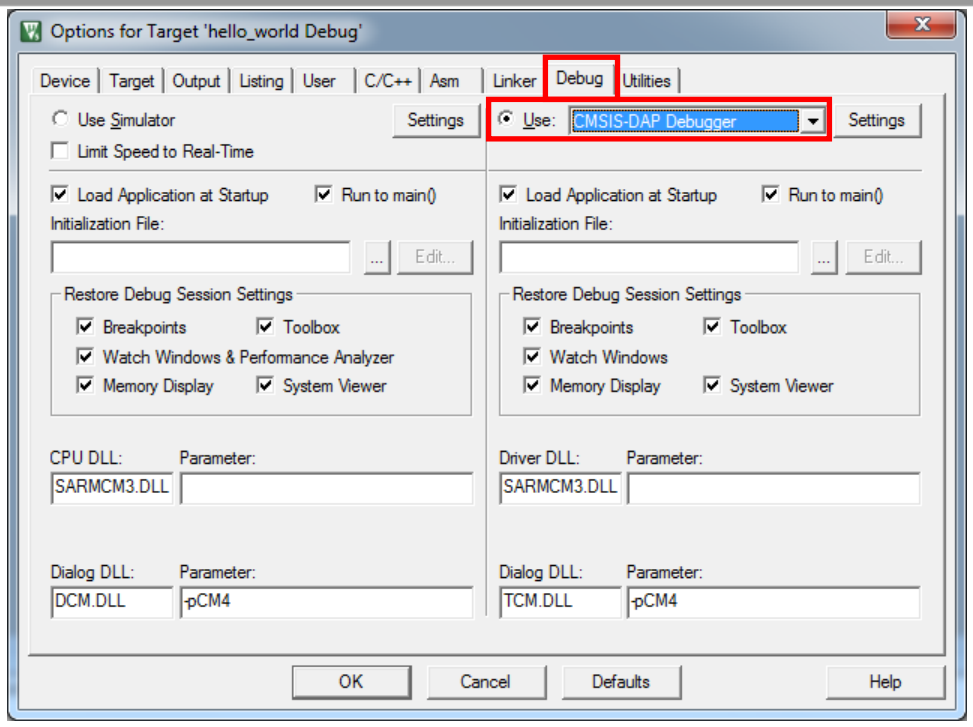

Figure 36 Debug driver selection for Freescale Freedom FRDM-K22F platform

c. Next, click the “Settings button” next to the debug driver selection drop-down box. Ensure that the settings are correct for the appropriate development platform.

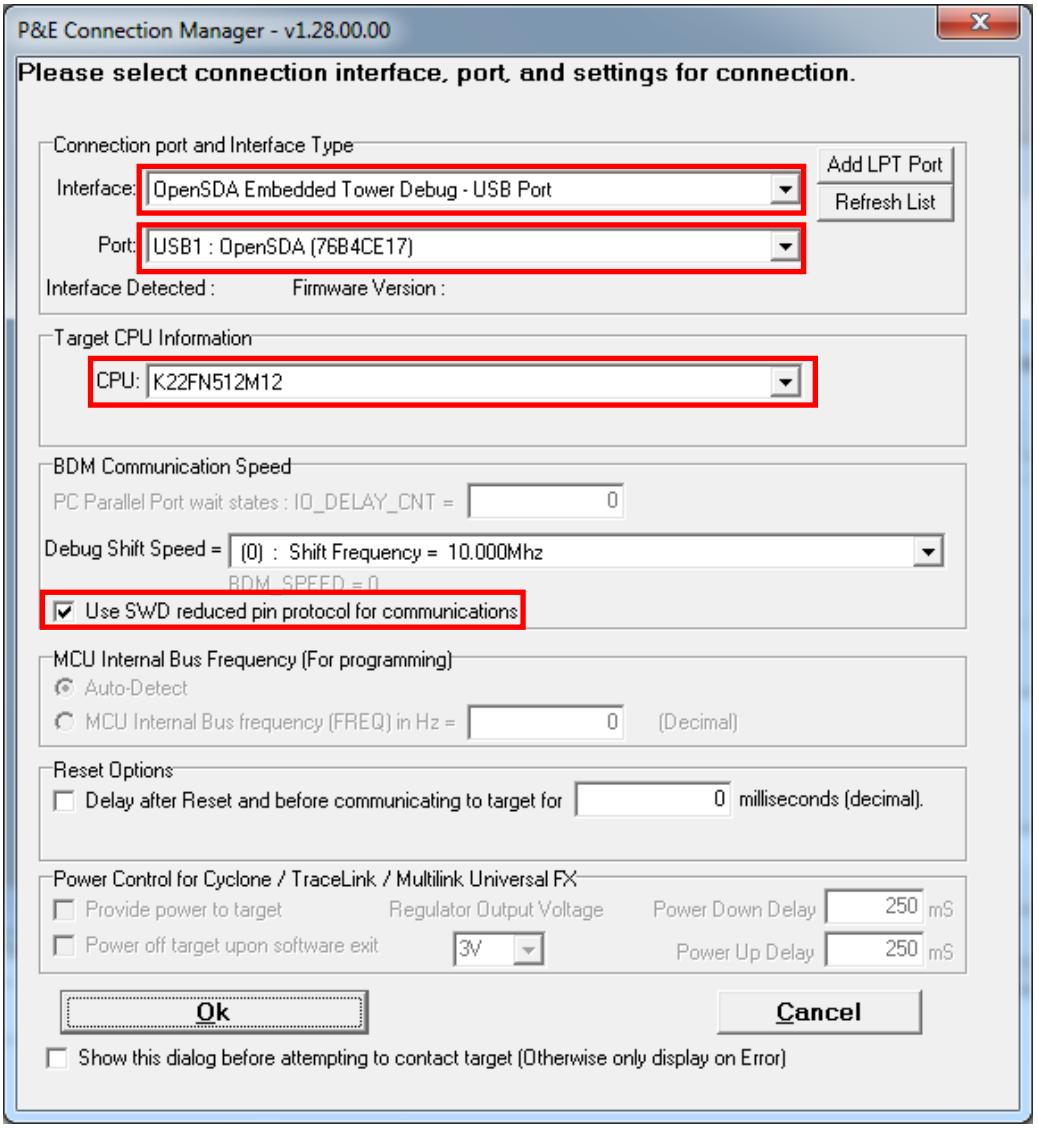

Figure 37 Debugger configurations for TWR-K22F120M Tower System module

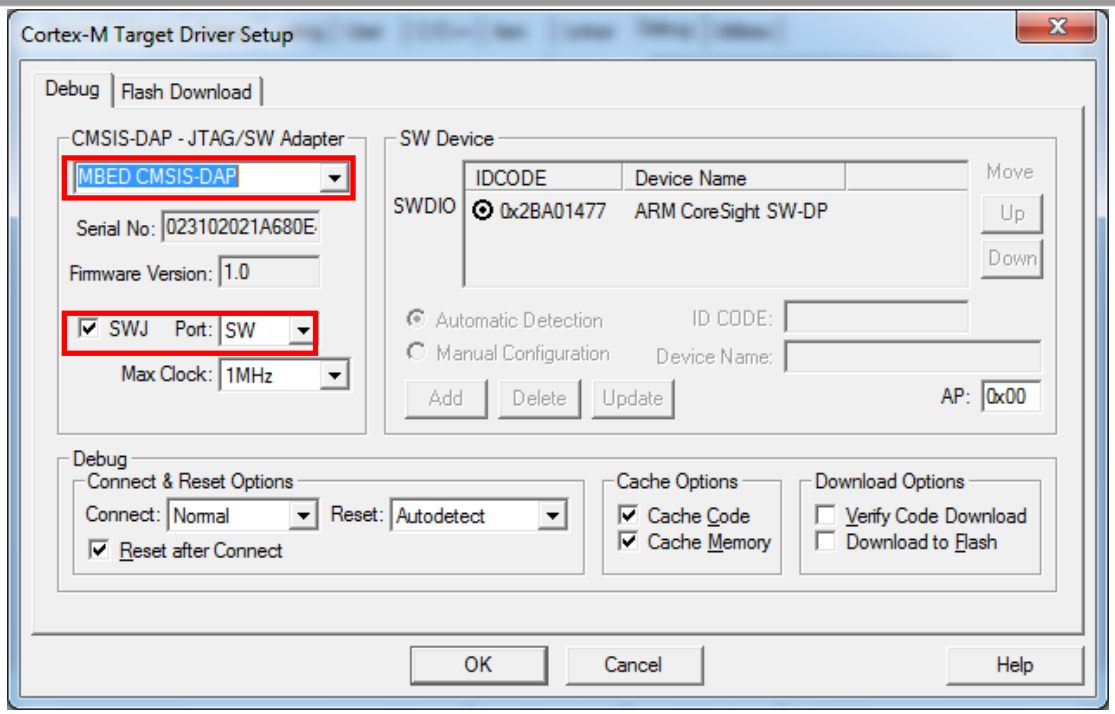

Figure 38 Debugger configurations for Freescale Freedom FRDM-K22F platform

d. If the target debugger is the CMSIS-DAP, the Flash loader must also be configured properly. To verify that the flash loader is configured properly, select the "Flash Download" in the Cortex-M Target Driver Setup dialog box and verify that the flash loader settings are configured as shown in this figure.

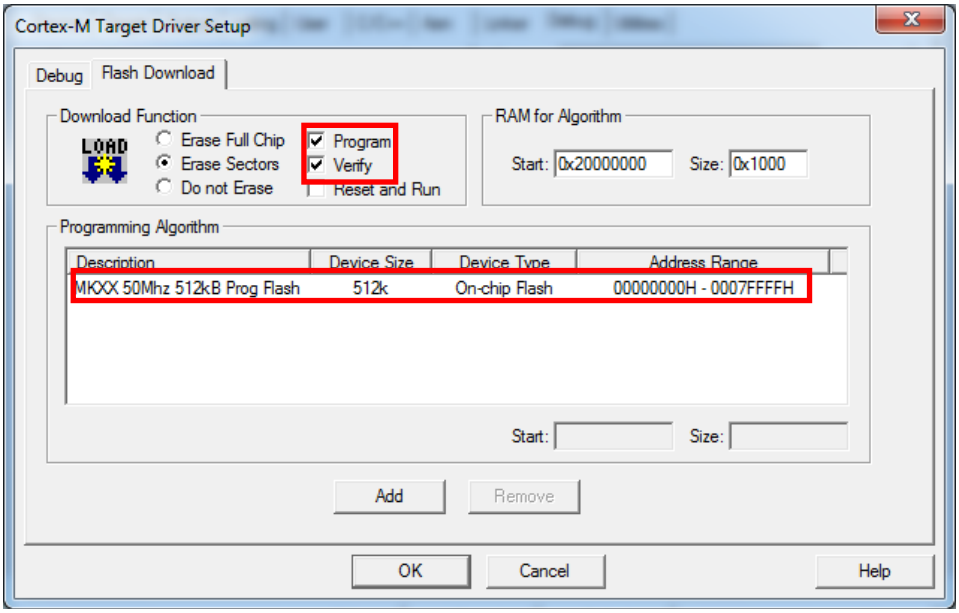

Figure 39 Flash loader configurations for Freescale Freedom FRDM-K22F platform

4. Once the application has been properly built and the debugger configurations have been verified, press the "Download button" to download the application to the target.

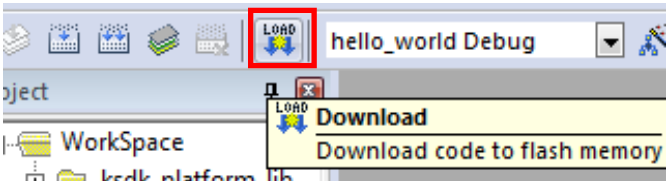

Figure 40 Download Button

5. After clicking the "Download button", the application has been downloaded to the target and should be running. To debug the application as well, click the "Start/Stop Debug Session button".

Figure 41 Start/Stop Debug Session Button

After clicking the “Start/Stop Debug Session button”, the debugger resets the target device and stop at main():

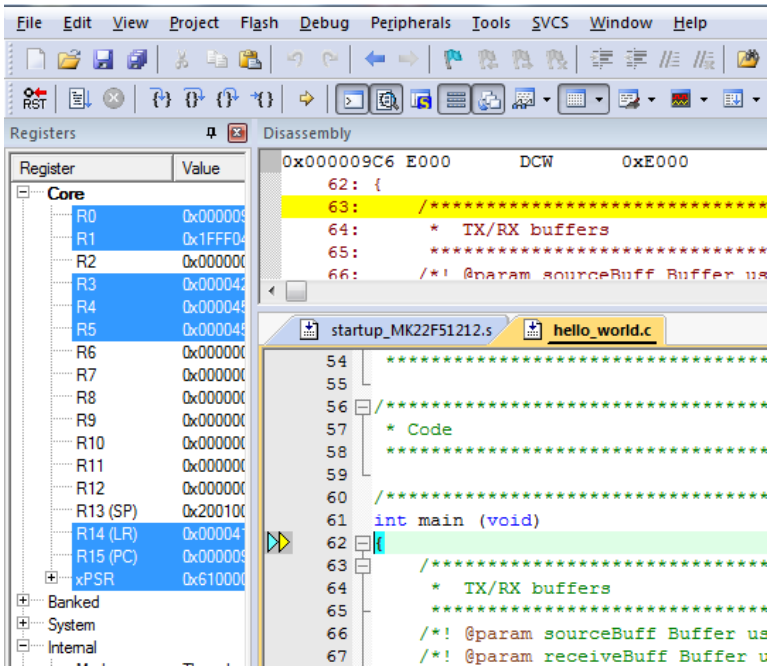

Figure 42 Stop at main() when run debugging

Now you can run the code by clicking on the “Run button” to start the application:

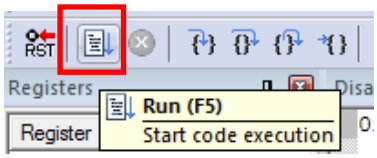

Figure 43 Go Button

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

Figure 44 Main prompt of the hello_world demo

7 Build and Run the KSDK Demo Applications using Kinetis Design Studio

This section describes the steps required to build, run, and debug demo applications and necessary driver libraries provided in the KSDK using Kinetis Design Studio. The hello_world demo application is used as an example (TWR-K22F120M Tower System module hardware platform).

7.1 Installing KSDK Eclipse update

Before using any Eclipse-based IDE with KSDK, apply the SDK Eclipse update, titled KSDK_<version>_Update_for_Eclipse. Without the update, Eclipse cannot generate KSDK-compatible projects. To install the update, follow these instructions:

1. Select Help > Install New Software.

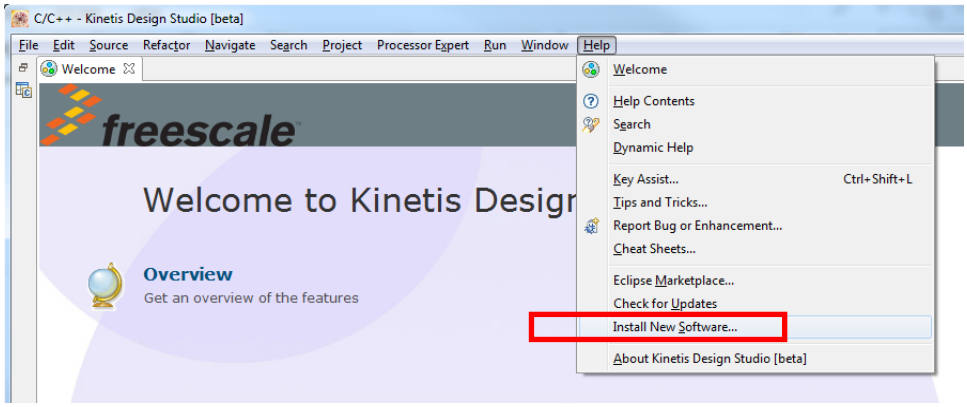

Figure 45 Install new software

2. In the "Install New Software" dialog box, select the "Add…" button in the upper right corner.

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

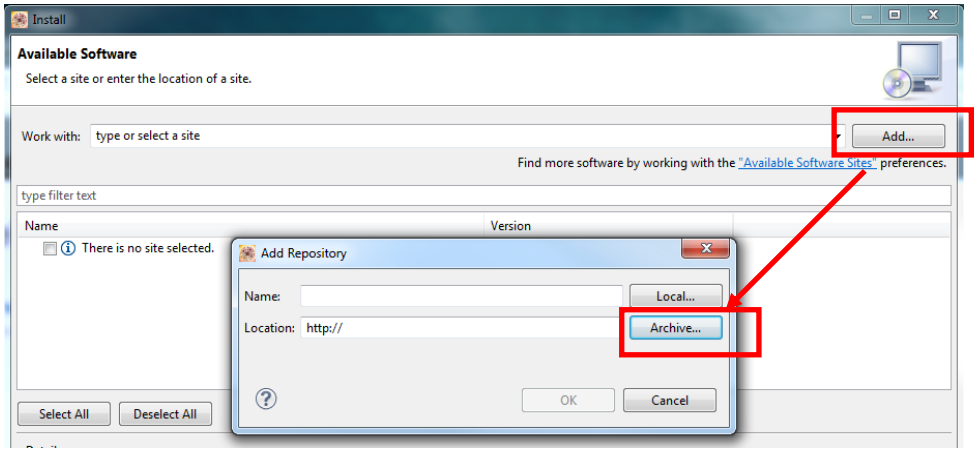

Figure 46 Add repository

4. In the dialog box that appears, browse your KSDK install directory.
5. From the top-level, enter the tools/eclipse_update folder and select the Eclipse update .zip file. The file name depends on the version of SDK that you are using. For example, the update file in the SDK 1.0.0 version is SDK_1.0.0-GA_Update_for_Eclipse.zip.
6. Click “Open”, then “OK” in the “Add Repository” dialog box.
7. The KSDK update now shows up in the list of the original Install dialog.
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.
10. After the update is applied, restart the KDS/Eclipse for the changes to take effect.

7.2 Building the platform driver library in Kinetis Design Studio

Before building and debugging demo applications in the KSDK, the driver library for the target device should be built. The library archive for the KDS demo applications is named ksdk_platform_lib.a. This library contains all HAL and peripheral driver functions which are device-specific. Therefore, each device has its own library (ksdk_platform_lib.a). The platform library is not prebuilt. Therefore, it is necessary to build the library after initially downloading the KSDK.

7.2.1 Open the library project in KDS

1) Select File->Import… from the KDS Eclipse menu.

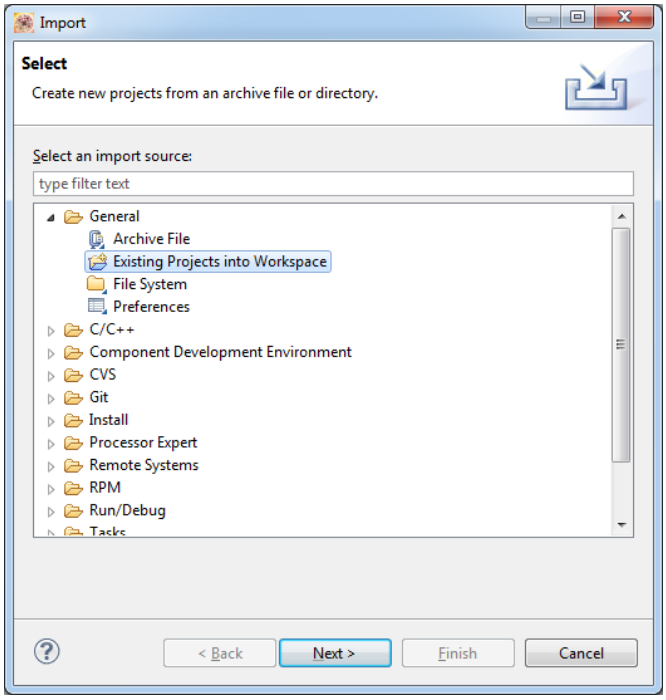

Figure 47 Selection of the correct import type in KDS

2) Select the “Select root directory:” option. Then click “Browse…” to point KDS to the correct library.

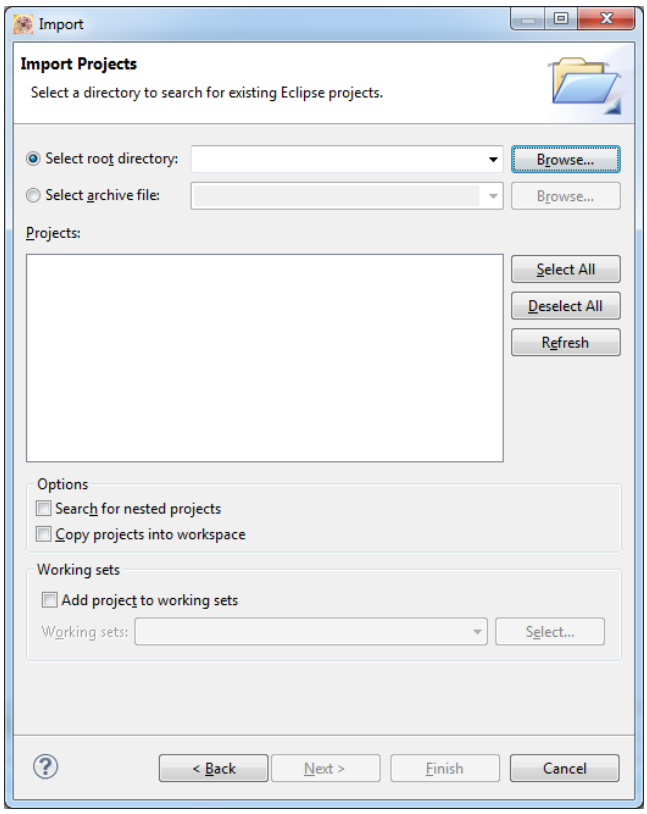

Figure 48 Import Projects directory selection window

3) Point KDS to the ksdk_platform_lib project in the K22F51212 (the library project is located at <Install_dir>/lib/ksdk_platform_lib/kds/<device_name>). Once you have done this, your Import Projects directory selection window should look like the following.

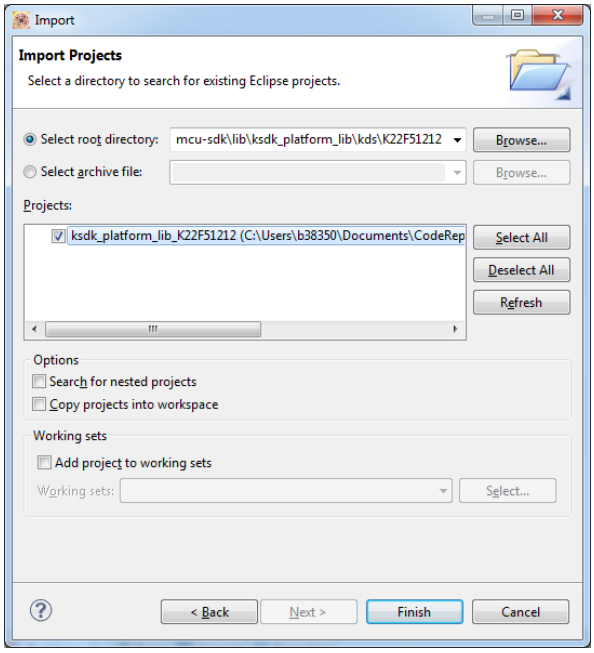

Figure 49 Import Projects directory selection window after selecting the K22F51212 ksdk_platform_lib project.

4) Click Finish.

7.2.2 Build the library project

In the Kinetis Design Studio platform library project, two compiler/linker configurations (build "targets") are supported:

- 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.

Choose the appropriate build target: "Debug" or "Release", by left-clicking the arrow next to the hammer icon as shown.

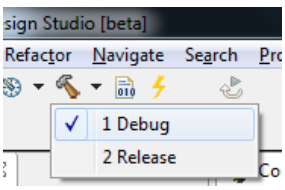

Figure 50 Selection of build target in KDS

If the library build does not begin after selecting the desired target, left-click the hammer icon to begin the build. When the build is complete, the library (ksdk_platform_lib.a) is generated in this directory according to the build target:

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

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

7.3 Build a demo application

The KSDK demo applications utilize a prebuilt linkable library to compile the necessary functions. Therefore, it is required that this library be built before compiling and downloading. To check that this library has been generated, verify that the ksdk_platform_lib.a file is in the <Install_dir>/lib/ksdk_platform_lib/<toolchain>/<device_name>/<build> location, where 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, the ksdk_platform_lib.a library should be in this folder:

<Install_dir>/lib/ksdk_platform_lib/kds/K22F51212/debug

Continue by opening the demo application project. Demo applications workspace files are located in the following folder:

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

Follow the steps in the above section 7.2.1. Open the library project to open the hello_world demo application project. The project is located in the following folder:

<install_dir>/demos/hello_world/kds/twrk22f120m/

To build the demo application project, select the target to build by left-clicking the arrow next to the hammer icon.

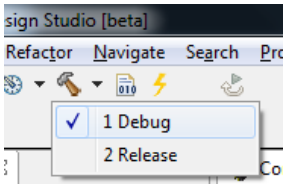

Figure 51 Selection of build target in KDS for the demo application

If your target application does not begin building immediately, left-click the hammer icon. When the build is complete, the KDS console window appears with the following information:

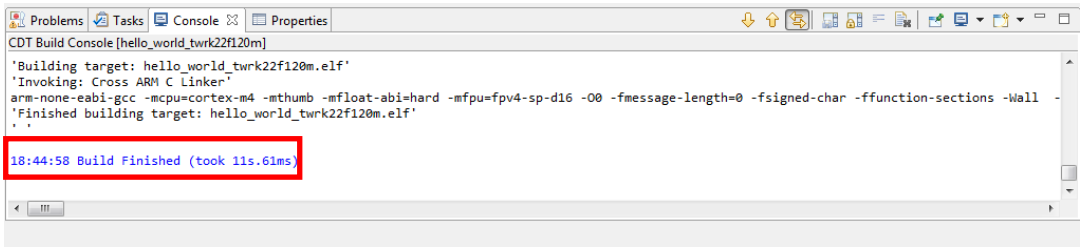

Figure 52 Console window output upon successful build

7.4 Run a demo application

To download and run the application, perform these actions:

NOTE

Before continuing with the following steps, ensure you have copied the Segger J-Link OpenSDA debug firmware to your OpenSDA hardware. If you are using the TWR-K22F120M, the original OpenSDA J-Link firmware is needed. If you are using the FRDM-K22F platform, the OpenSDA v2 J-Link firmware is needed. Both versions of the firmware can be found at https://segger.com/opensda.html.

For more details on changing your OpenSDA firmware, consult the Quick Start documentation for your Freescale Development Platform.

1. Connect the K22 development board 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 OpenSDA serial port number. Configure the terminal with the following settings:
    a. 115200 baud rate
    b. No parity
    c. 8 data bits
    d. 1 stop bit

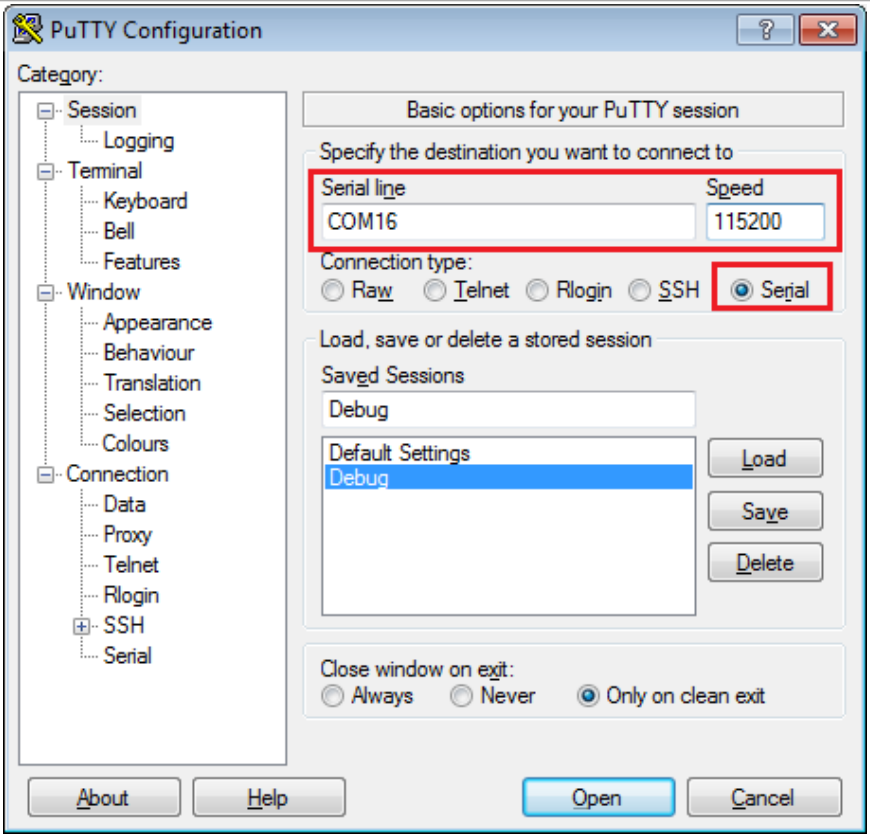

Figure 53 Terminal (PuTTY) configurations

3. Ensure that the debugger configuration is correct in the project options.
    a. To check the debugger configurations, click the down arrow next to the green debug button and select “Debug Configurations”.

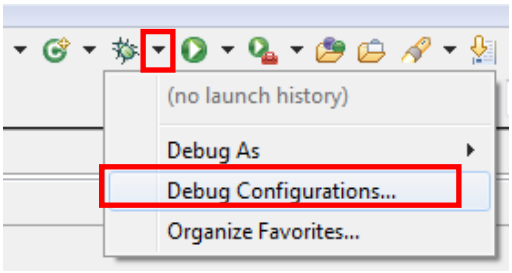

Figure 54 Debug Configurations dialog button

b. In the Debug Configurations dialog box, select debug configuration from the GDB SEGGER J-Link Debugger in the groups on the left-hand side of the dialog box.

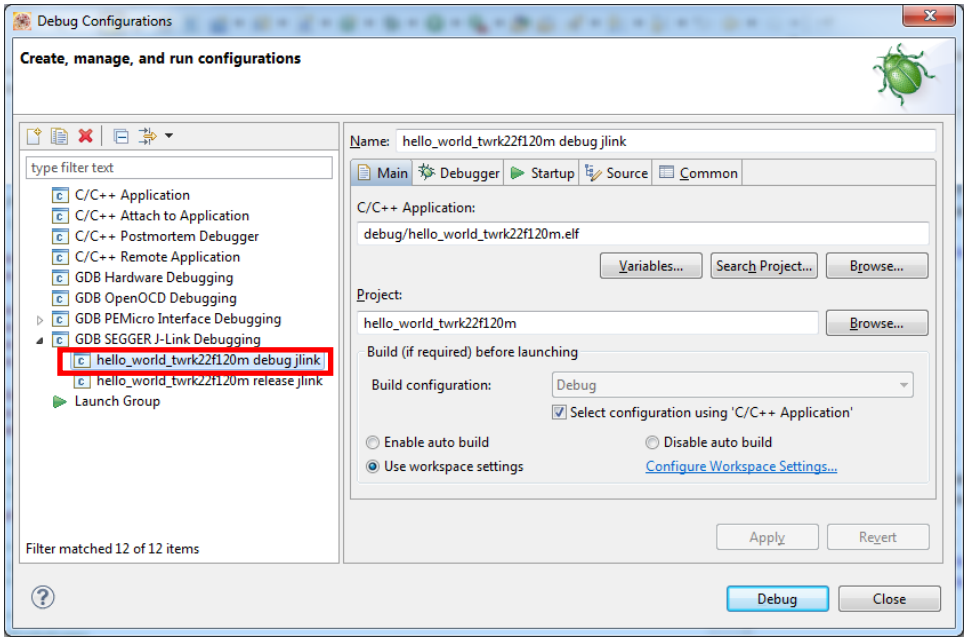

Figure 55 Selection of the debug configuration in the Debug Configurations dialog box.

c. In the debugger setup, 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).

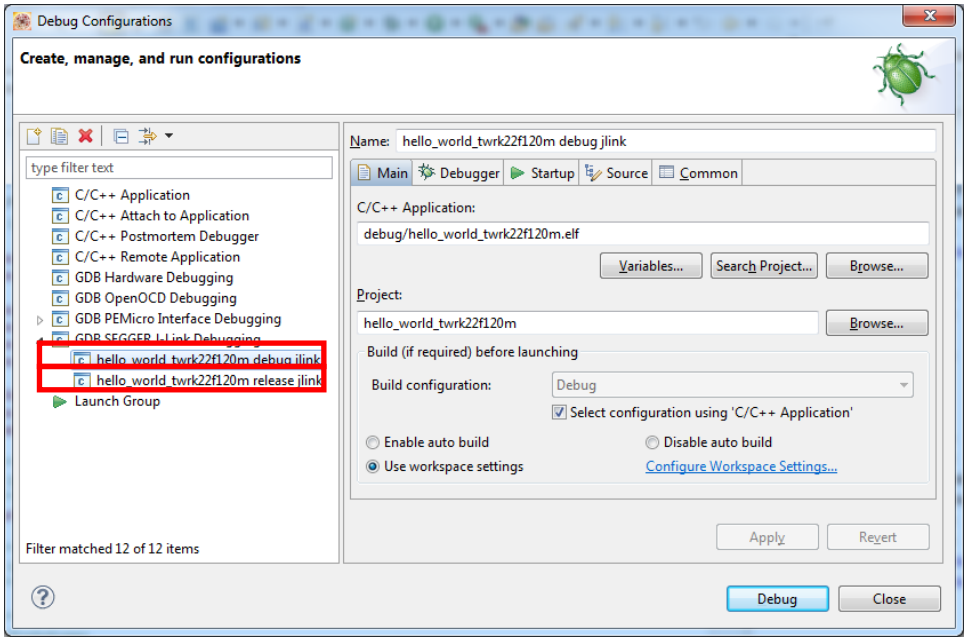

Figure 56 KDS debugger configurations for TWR-K22F120M Tower System module

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

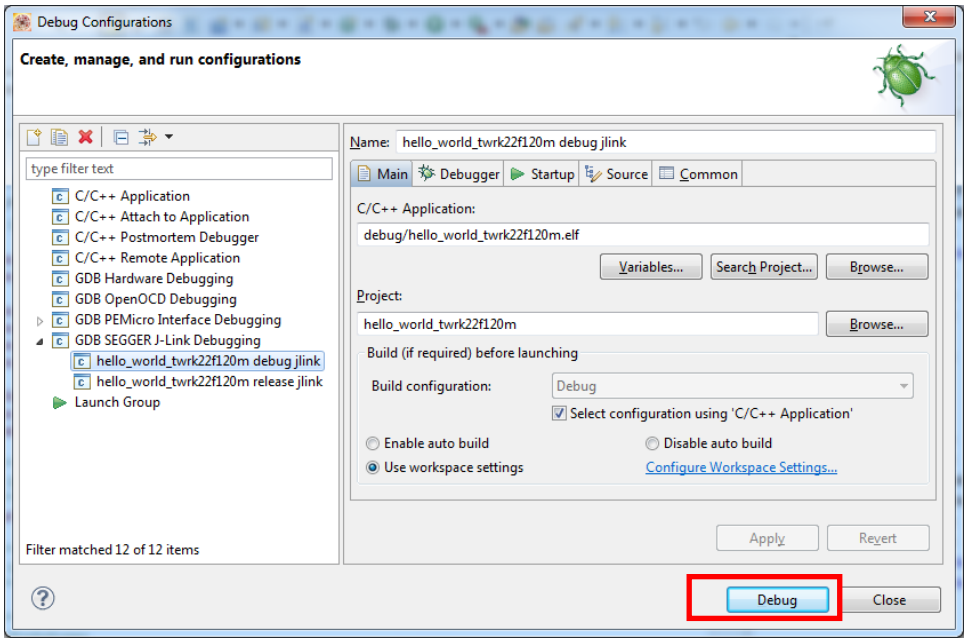

Figure 57 Debug button in Debug Configurations dialog box

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

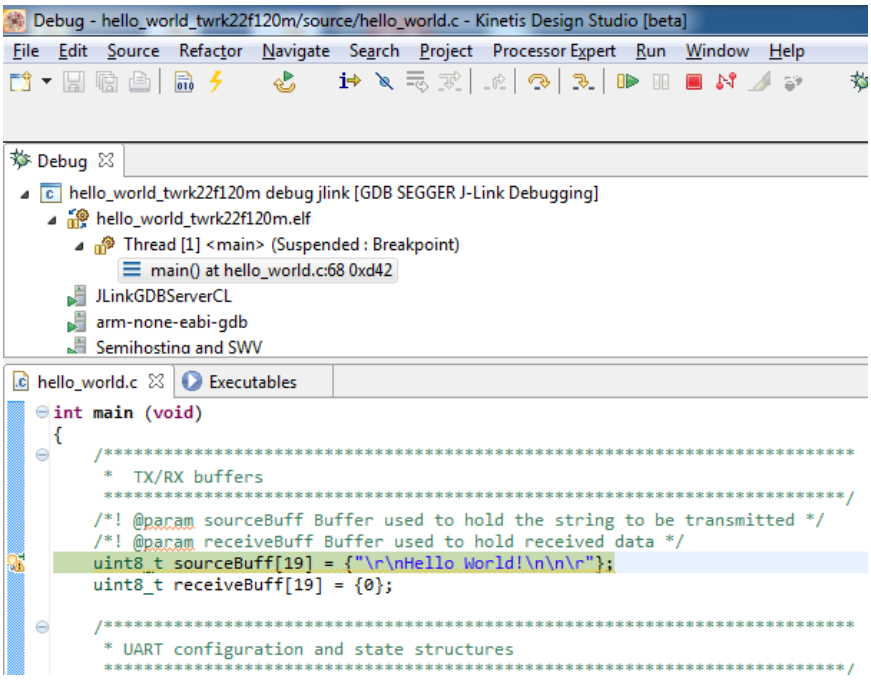

Figure 58 Stop at main() when run debugging

Now you can run the code by clicking on the "Resume button" to start the application:

Figure 59 Resume Button

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

Figure 60 Main prompt of the hello_world demo

8 Revision history

This table summarizes revisions to this document.

Revision History		
Revision number	**Date**	**Substantial changes**
1.0.0	7/2014	Initial release

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: KSDKK22UG
Rev. #1.0.0
07/2014