

Kinetis SDK KV31 User's Guide

1 Introduction

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

2 Overview

2.1 Kinetis SDK

KSDK is a software development kit that provides comprehensive software support for the core and peripherals of all Freescale devices with ARM 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 contains the latest available RTOS kernels, and USB stacks for use on supported evaluation boards.

Contents

1	Introduction.....	1
2	Overview	1
3	Hardware Configurations.....	2
4	Build and Run the KSDK Demo Applications using IAR	5
5	Build and Run the KSDK Demo Applications using ARM GCC	12
6	Build and Run the KSDK Demo Applications using Keil MDK.....	22
7	Build and Run the KSDK Demo Applications using Kinetis Design Studio	30
8	Revision history	41

2.2 Hardware requirement

- TWR-KV31F120M Tower System module
- USB A to micro B cable for debug interface and power
- Personal Computer

2.3 Toolchain requirement

- IAR embedded Workbench version 7.20.2 or later
- ARM GCC 4.8.3 2014q1 or later
- Keil MDK 5.11 or later
- Kinetis Design Studio v. 1.0.2 or later

3 Hardware Configurations

This section describes how to set up the TWR-KV31F120M Tower System module for the KSDK application.

3.1 TWR-KV31F120M Tower System module introduction

3.1.1 TWR-KV31F120M Tower System module features

- Tower-compatible microcontroller module
- KV31F512VLL12 MCU (120 MHz, 512 KB Flash, 96 KB RAM, 100 LQFP package)
- General-purpose Tower Plug-in (TWRPI) socket
- Onboard debug circuit: K20DX128VFM5 (OpenSDA) with virtual serial port
- FXOS8700CQ: 6-Axis Digital Sensor Accelerometer + Magnetometer.
- Four (4) user controllable LEDs plus RGB LED
- Four (4) user pushbutton switches for GPIO interrupts
- One (1) user pushbutton switch for MCU reset
- Potentiometer

3.1.2 TWR-KV31F120M Tower System module first look

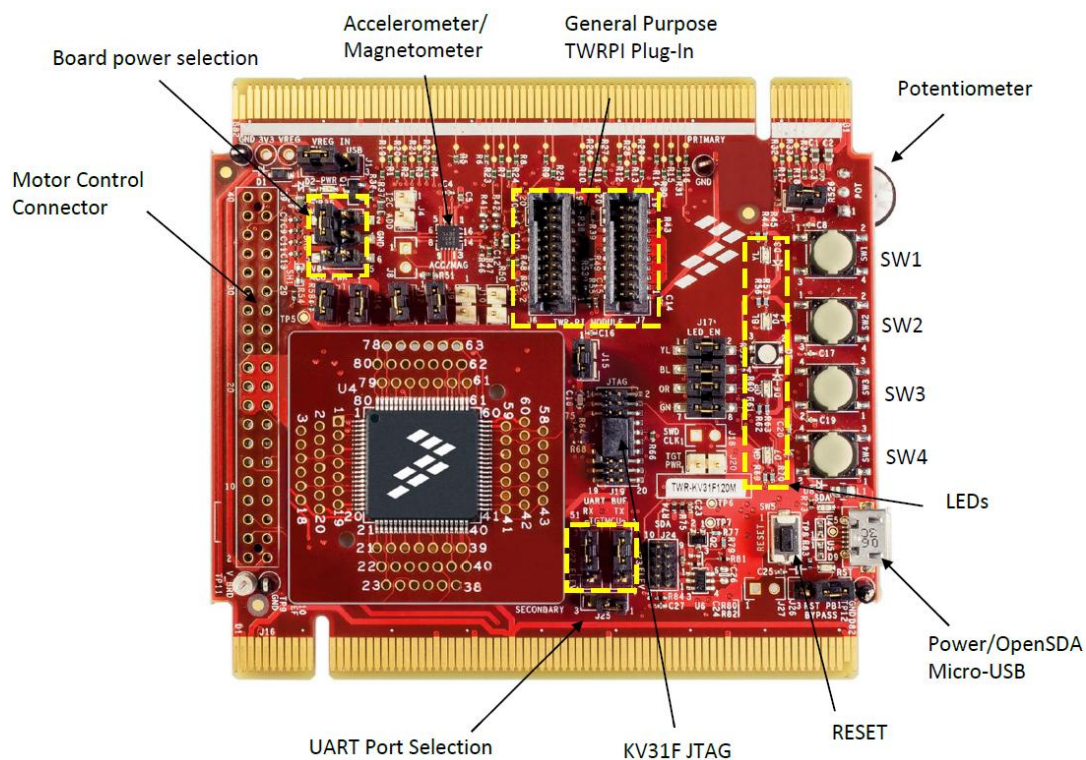


Figure 1 Front side of TWR-KV31F120M Tower System module

3.1.3 TWR-KV31F120M Tower System module jumper settings

Configure the jumper settings as indicated. For detailed jumper options and headers, see the TWR-KV31F120M Tower System Module User's Guide.

Table 1: TWR-KV31F120M Tower System module jumper settings

Option	Jumper	Setting	Description
Clock Input Source Selection	J25	1-2	Connect main EXTAL to onboard 8 MHz crystal
		2-3	Connect EXTAL to CLKIN0 signal on Primary Elevator (B24)
Tower Voltage Regulator Input Selector	J1	1-2	[Default] Connect P5V_TRG_SDA (5V from OpenSDA) or P5_Elev to VREG_IN
		2-3	[EXT]Connect USB0_VBUS from Primary Elevator (A5 7) to VREG_IN
Board Power Selector	J5	1-3	Connect onboard 3.3V regulator output (P3V3_REG) to main board power line (V_BRD)
		3-5	Connect onboard 1.8V regulator output (P1V8) to main board power line (V_BRD)
MCU VDD current measurement	J13	ON	Connect V_BRD to MCU_PWR
		OFF	Allow current measurement on MCU VDD
LED Connections	J17	1-2	Connect PTE1 to Yellow LED D3
		3-4	Connect PTE0 to Red LED D4
		5-6	Connect PTB19 to Orange LED D6
		7-8	Connect PTD7 Green LED D7
UART RX Selection	J22	1-2	Connect UART0_RX_TGTMCU to UART1_RX_ELEV_BUF (Tower Elevator)
		2-3	Connect UART0_RX_TGTMCU to UART1_RX_TGTMCU_BUF (OpenSDA)
UART TX Selection	J23	1-2	Connect UART0_TX_TGTMCU to UART1_TX_ELEV_BUF (Tower Elevator)
		2-3	Connect UART0_TX_TGTMCU to UART1_TX_TGTMCU_BUF (OpenSDA)

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 demo application, targeted for the TWR-KV31F120M Tower System module hardware platform, is used as an example.

4.1 Building the platform driver library in IAR

The driver library project should be built to generate the library archive platform_lib.a before building and debugging demo applications in KSDK. This library contains all HAL and peripheral driver functions which are device-specific. Therefore, each device has its own library (platform.a). It should not be necessary to build the library after initially downloading the KSDK because the platform library is prebuilt. However, if this 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 KV31, open the workspace file in IAR:

```
<Install_dir>/lib/ksdk_platform_lib/iar/KV31F51212/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 is used for developing and debugging.
- Release - The compiler optimization is set to high. The debug information is not generated. This target is 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):

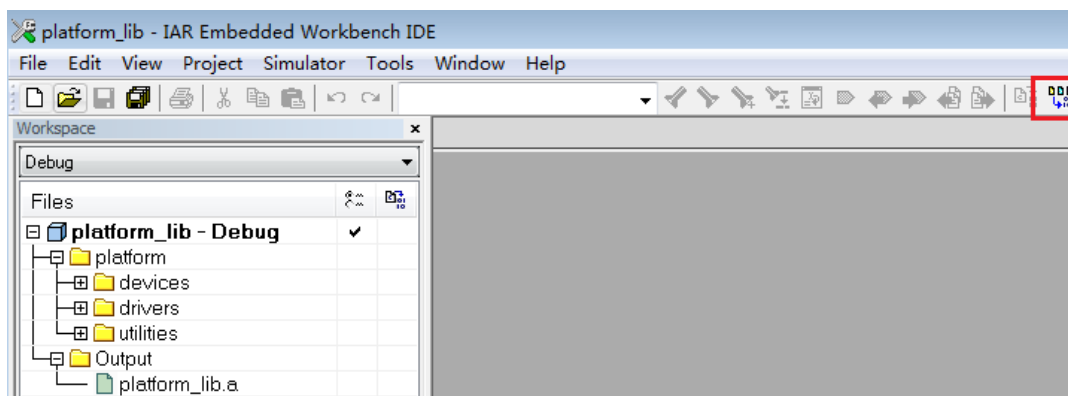


Figure 2 Platform driver library build

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. 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 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 platform_lib.a library should be in this folder:

<Install_dir>/lib/ksd_platform_lib/iar/KV31F51212/debug

If the platform driver library has not already been built, follow the directions in the appendix of this document to build the platform driver library before continuing. Otherwise, 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/twrkV31f120m/hello_world.eww

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

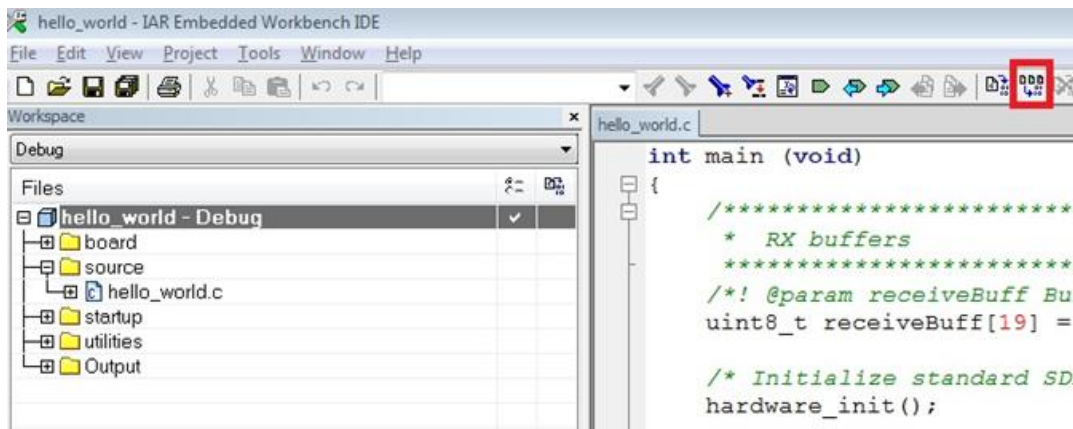


Figure 3 Build the hello_world demo application

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

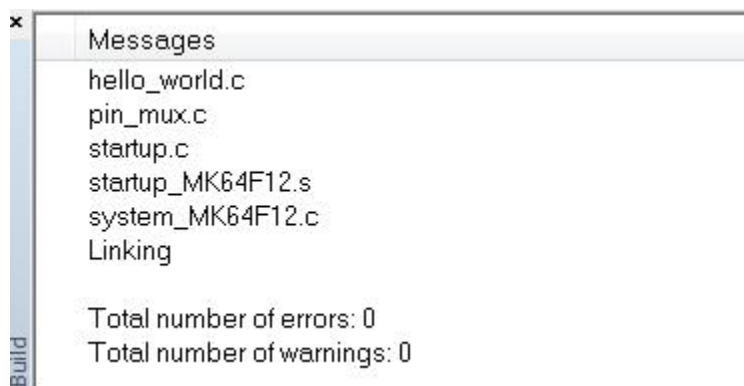


Figure 4 Build hello_world demo successfully

4.3 Run a demo application

To download and run the application, perform these actions:

1. Connect the KV31 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

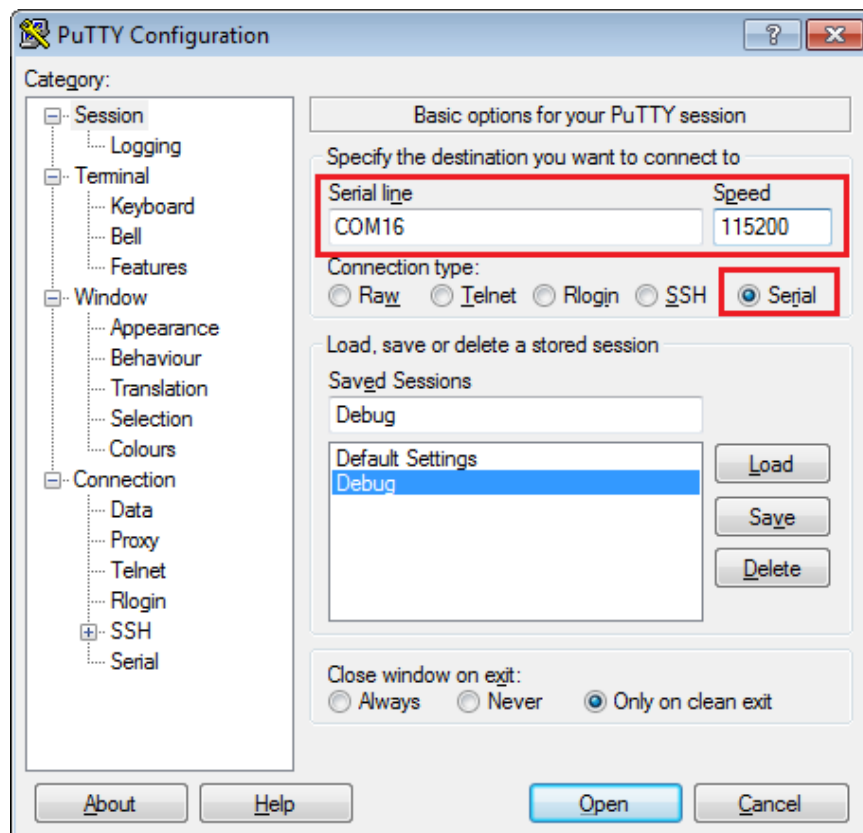


Figure 5 Terminal (PuTTY) configurations

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.

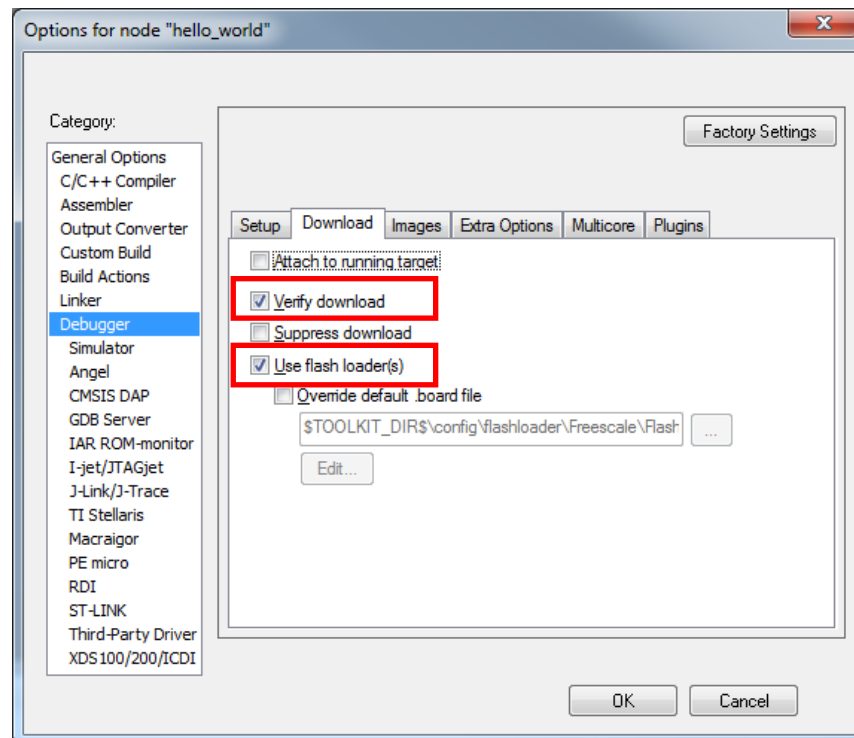


Figure 6 Flash loader configurations

- b. Select the appropriate debugger in the debugger setup.

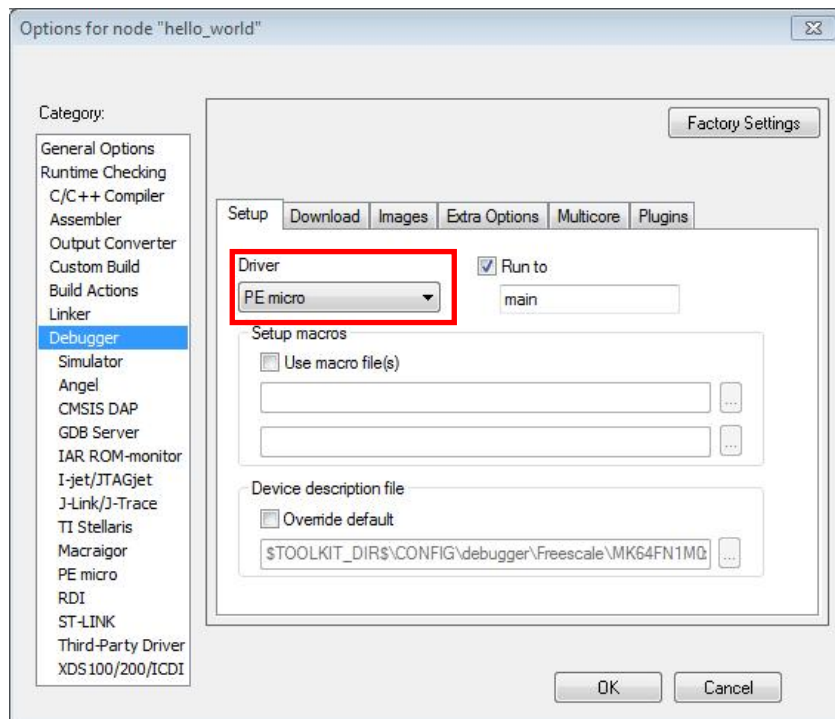


Figure 7 Debugger configurations for TWR-KV31F120M Tower System module

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

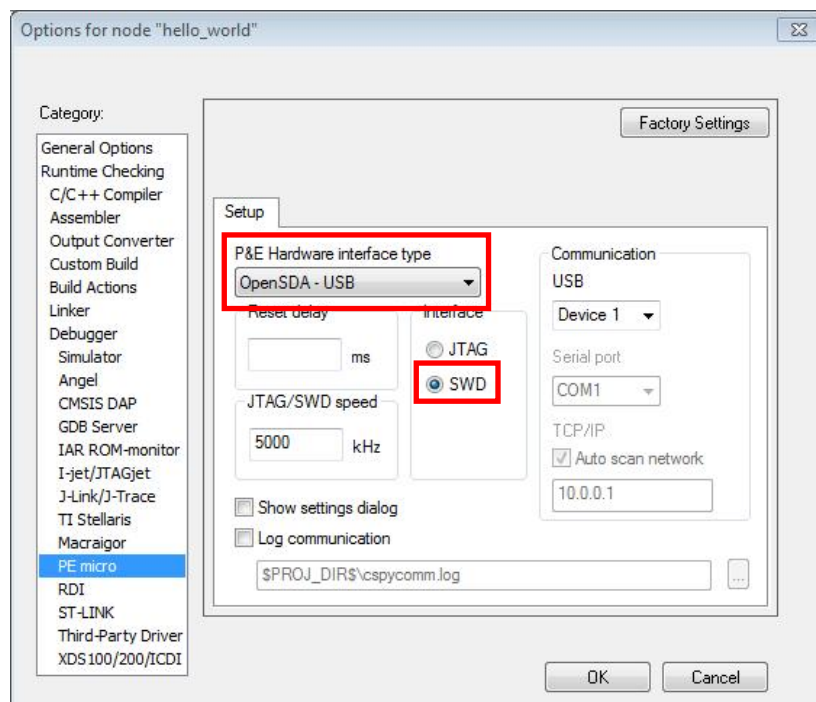


Figure 8 Debugger configurations for TWR-KV31F120M Tower System module (PE micro)

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



Figure 9 Download and Debug Button

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

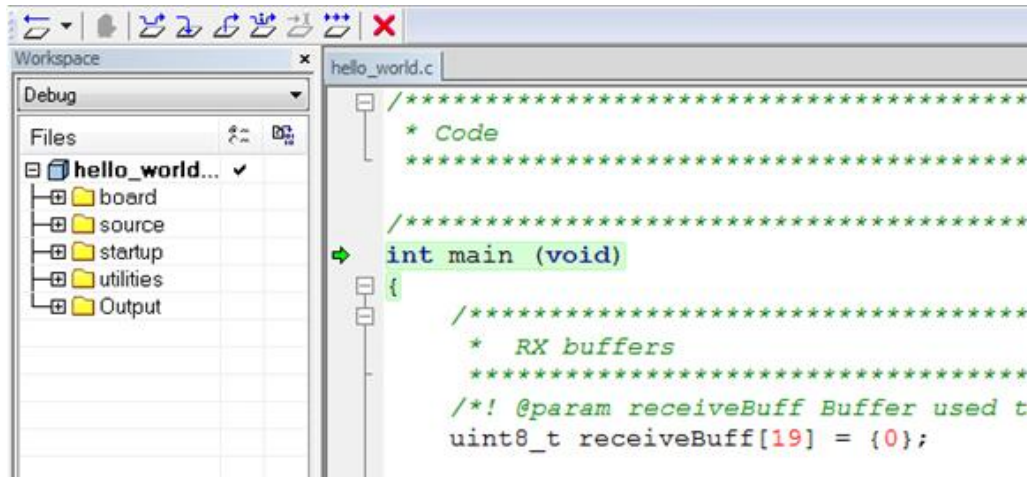


Figure 10 Stop at main() when run debugging

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



Figure 11 Go Button

- 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.

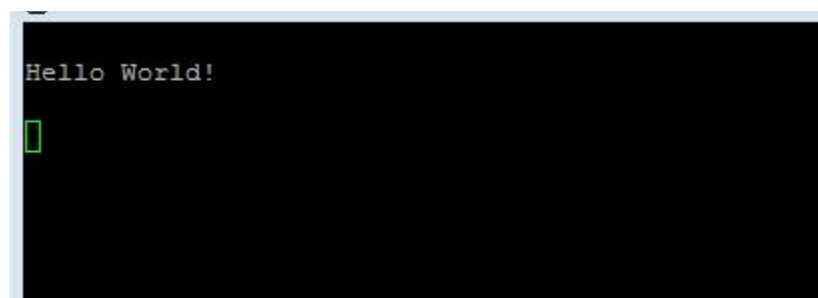


Figure 12 Main prompt of the hello_world demo

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-KV31F120M Tower System module hardware platform, is used as an example.

5.1 Environment setup

5.1.1.1 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.

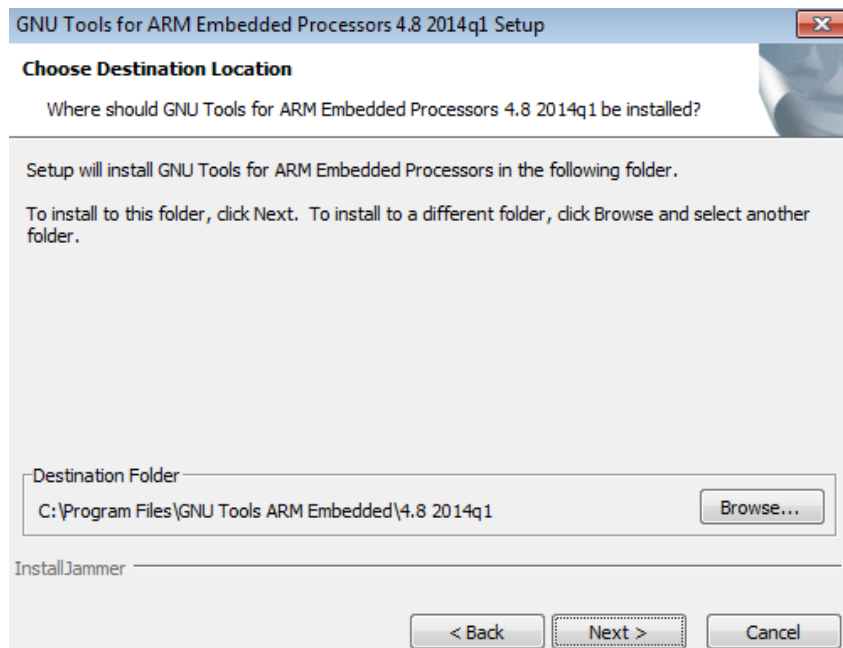


Figure-13 Install GCC ARM embedded toolchain

5.1.1.2 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-14 MinGW Installer.

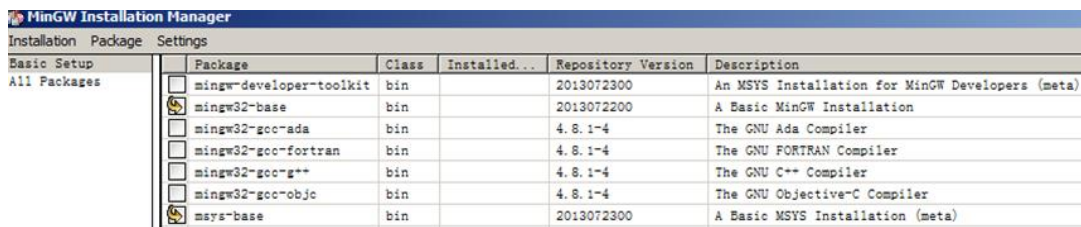


Figure-14 MinGW Installer

- Click “Apply Changes” from the “Installation” menu to install packages, as shown here.

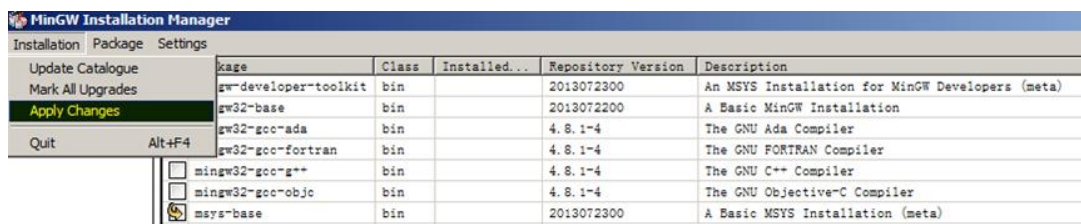


Figure-15 MinGW Installer apply change

5.1.1.3 Configure system environment

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

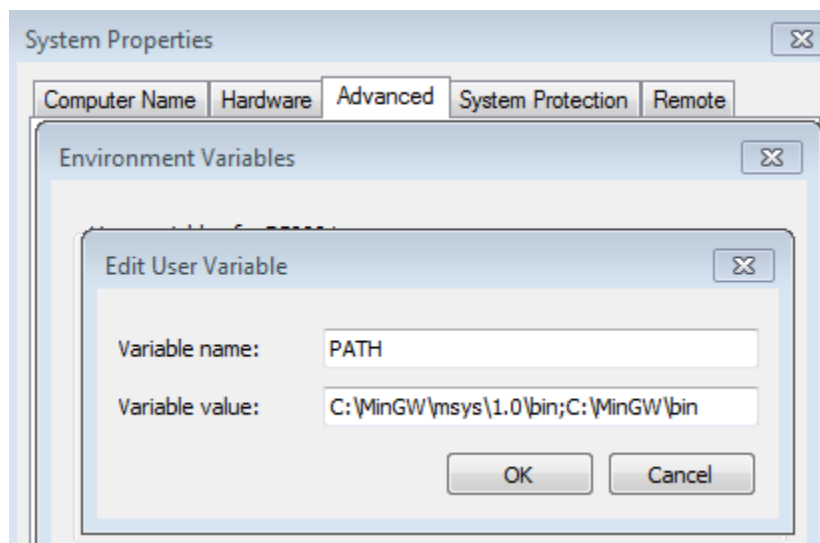
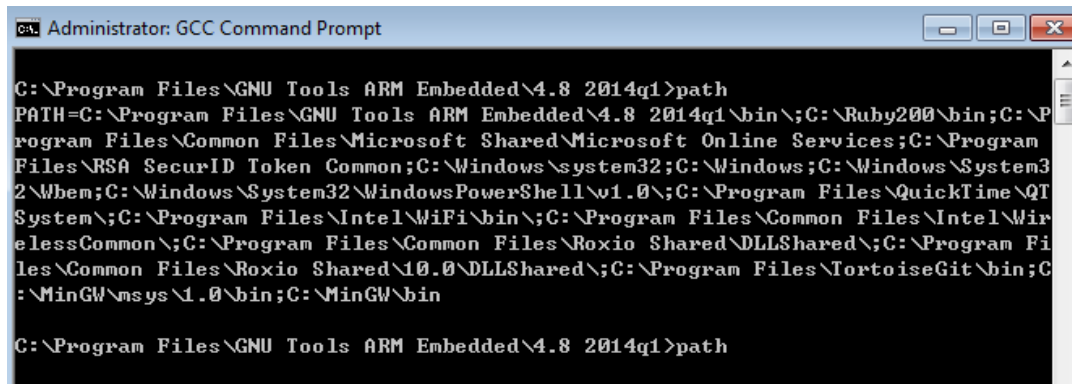


Figure-16 Environment variable update

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



```
C:\Program Files\GNU Tools ARM Embedded\4.8 2014q1>path
PATH=C:\Program Files\GNU Tools ARM Embedded\4.8 2014q1\bin\;C:\Ruby200\bin\;C:\P
rogram Files\Common Files\Microsoft Shared\Microsoft Online Services\;C:\Program
Files\RSA SecurID Token Common\;C:\Windows\system32\;C:\Windows\;C:\Windows\System3
2\Wbem\;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\QuickTime\QT
System\;C:\Program Files\Intel\WiFi\bin\;C:\Program Files\Common Files\Intel\Wir
elessCommon\;C:\Program Files\Common Files\Roxio Shared\DLLShared\;C:\Program Fi
les\Common Files\Roxio Shared\10.0\DLLShared\;C:\Program Files\TortoiseGit\bin\;C
:\MinGW\msys\1.0\bin\;C:\MinGW\bin

C:\Program Files\GNU Tools ARM Embedded\4.8 2014q1>path
```

Figure-17 PATH environment

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

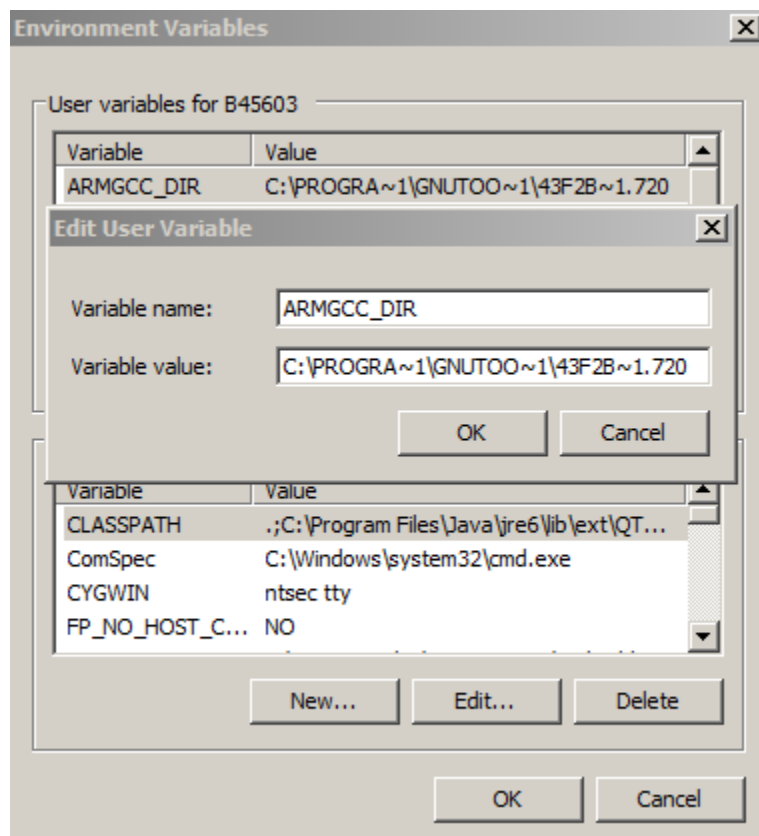


Figure-18 Add environment variable

- 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 KV31 as an example, the correct path would be:
<install_dir>/lib/ksdk_platform_lib/gcc/KV31F51212.

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>
location, where build is the desired build, and can be 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 exist in the following folder:
<Install_dir>/lib/ksd_platform_lib/gcc/KV31F51212/debug

Next, continue by opening the demo application project. Demo applications Makefile is located in:

<install_dir>/demos/<demo_name>/<toolchain>/<board_name>/Makefile

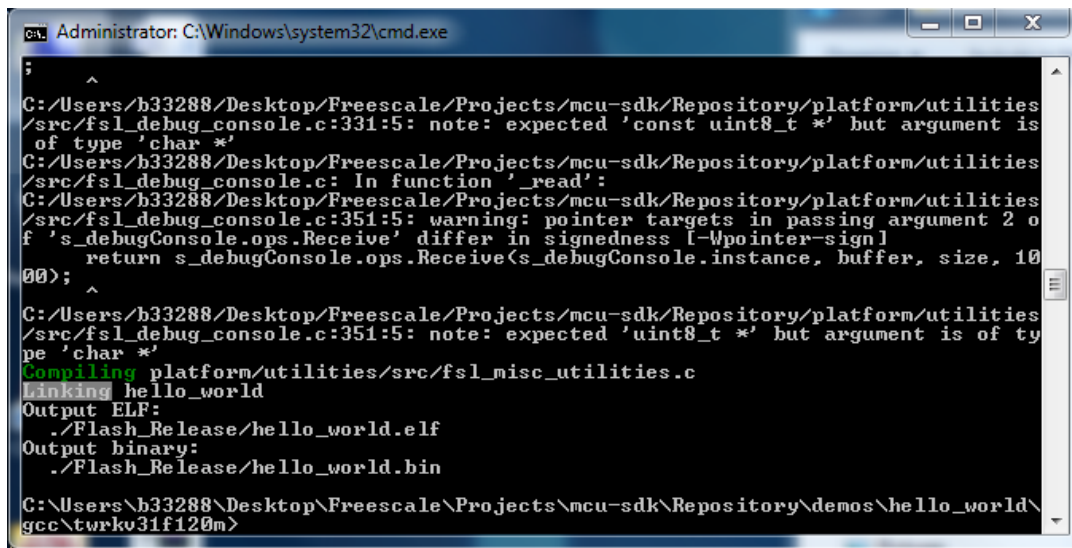
The hello_world application is used as an example. The GCC Makefile is located in:

<install_dir>/demos/hello_world/gcc/twrkv31f120m/

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 the following:

Build options	Target options
Debug	Flash
Release	SRAM

When the build is complete, the GCC command prompt shows the following:



```
Administrator: C:\Windows\system32\cmd.exe

C:/Users/h33288/Desktop/Freescale/Projects/mcu-sdk/Repository/platform/utilities
/src/fsl_debug_console.c:331:5: note: expected 'const uint8_t *' but argument is
of type 'char *'
C:/Users/h33288/Desktop/Freescale/Projects/mcu-sdk/Repository/platform/utilities
/src/fsl_debug_console.c: In function '_read':
C:/Users/h33288/Desktop/Freescale/Projects/mcu-sdk/Repository/platform/utilities
/src/fsl_debug_console.c:351:5: warning: pointer targets in passing argument 2 o
f 's_debugConsole.ops.Receive' differ in signedness [-Wpointer-sign]
    return s_debugConsole.ops.Receive(s_debugConsole.instance, buffer, size, 10
00);
C:/Users/h33288/Desktop/Freescale/Projects/mcu-sdk/Repository/platform/utilities
/src/fsl_debug_console.c:351:5: note: expected 'uint8_t *' but argument is of ty
pe 'char *'
Compiling platform/utilities/src/fsl_misc_utilities.c
Linking hello_world
Output ELF:
./Flash_Release/hello_world.elf
Output binary:
./Flash_Release/hello_world.bin

C:\Users\h33288\Desktop\Freescale\Projects\mcu-sdk\Repository\demos\hello_world\
gcc\twrkv31f120m>
```

Figure 13 hello_world demo successfully built

5.4 Run a demo application

To download and run the application, perform these actions:

1. Install the Segger J-Link software and documentation if it is not already installed. Download the package at [http:// segger.com/downloads.html](http://segger.com/downloads.html). Otherwise, continue to step 2.
2. Connect your J-Link debug pod to the MKV31F512 SWD connector (J19) of the tower board.

3. Open the J-Link GDB Server application. After opening this application, modify the connection settings as shown in the following figure:

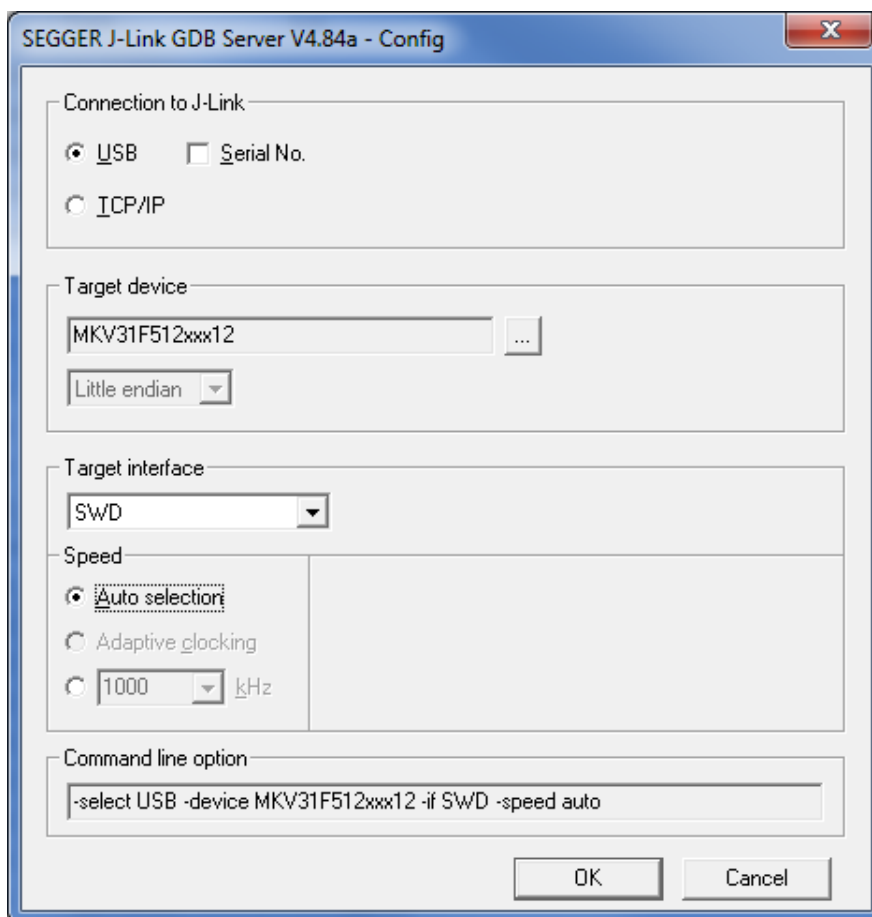


Figure 20 Segger J-Link GDB server configurations

- After it is connected, this screen should appear.

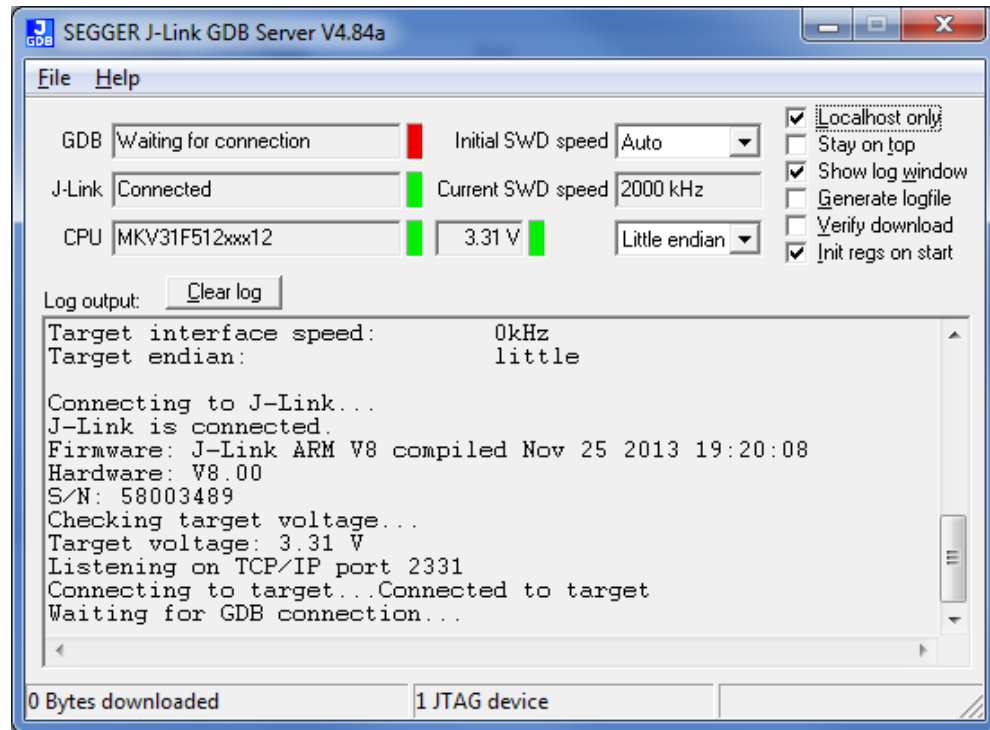


Figure 21 Segger J-Link GDB Server screen after successful connection

- 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\twrkv31f120m\Flash_Debug.
- Run the command "arm-none-eabi-gdb <DEMO_NAME>.elf". For this example, that is "arm-none-eabi-gdb hello_world.elf".
- Run the following commands
 - "target remote localhost: 2331"
 - "monitor reset"
 - "monitor halt"
 - "load"
 - "monitor reset"

8. 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

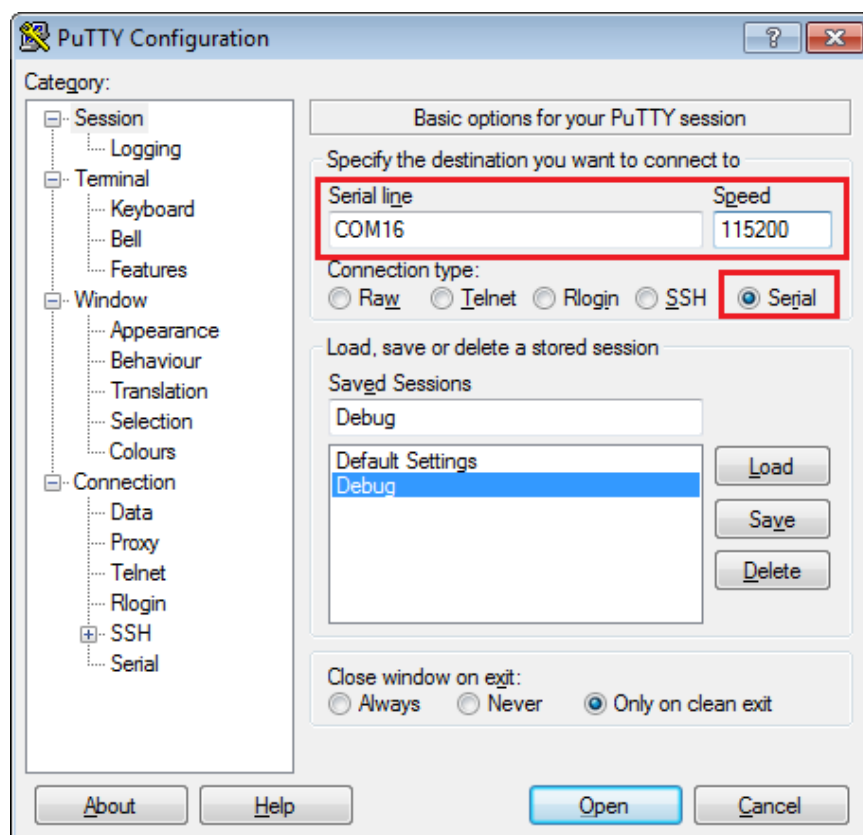


Figure 22 Terminal (PuTTY) configurations

9. The application is now downloaded and connected. Execute the following command to begin the demo application.
 - a. “monitor go”

10. The hello_world demo 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.

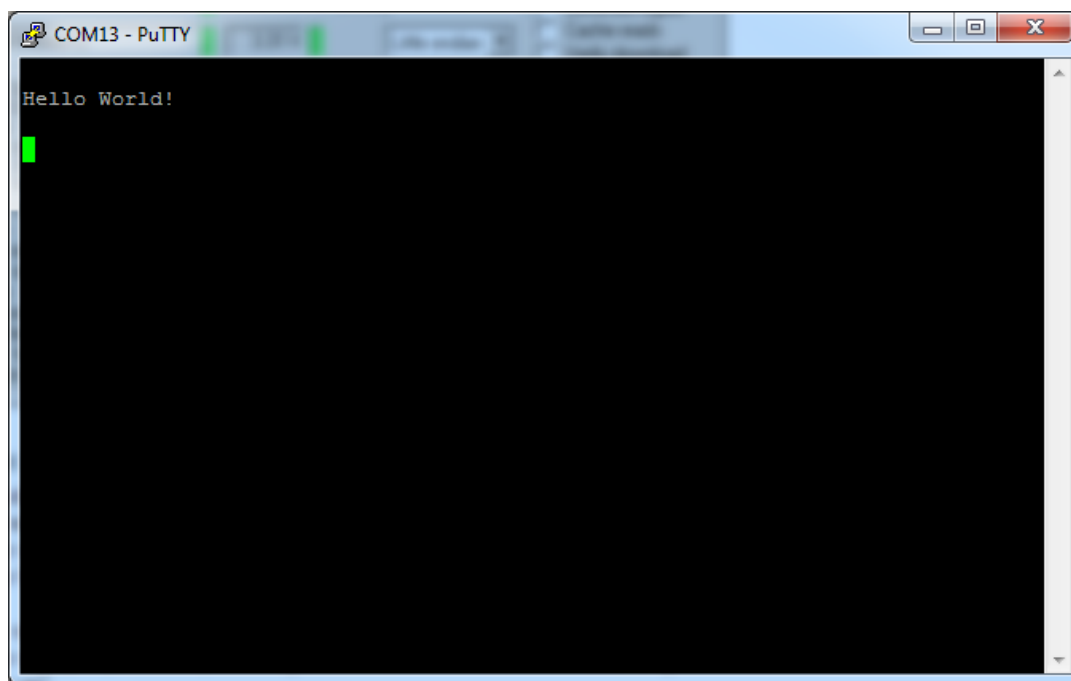


Figure 23 Main prompt of the hello_world demo

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 hello_world demo application is used as an example (TWR-KV31F120M Tower System module hardware platform).

NOTE

Kinetis KVxx Series Device Support BSP DFP must be updated using the pack installer tool.

6.1 Building the platform driver library in Keil MDK

The driver library for the target device should be built before building and debugging demo applications in the KSDK. 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, so it is necessary to build the library after initially downloading the KSDK.

Open the hello_world multi-project workspace file (`hello_world.uvmpw`) in Keil (Project->Open project...). The demo application multi-project 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/twrkv31f120m/hello_world.uvmpw`

To build the platform driver library for the KV31, first set the `ksdk_platform_lib` as the active project by right-clicking on the `ksdk_platform_lib` and selecting “Set as Active Project”.

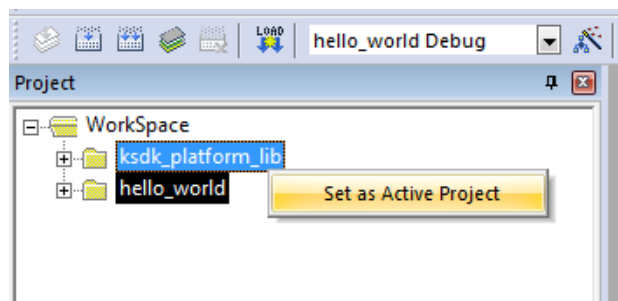


Figure 24 Selection of the `ksdk_platform_lib` as the active project

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 is used for developing and debugging.

- Release - The compiler optimization is set to high. The debug information is not generated. This target is used for final application release.

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

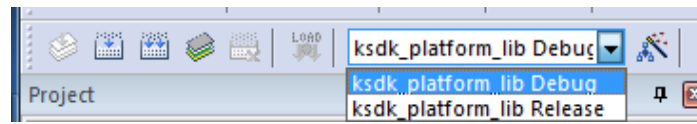


Figure 25 Drop-down selection of the Debug/Release target

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

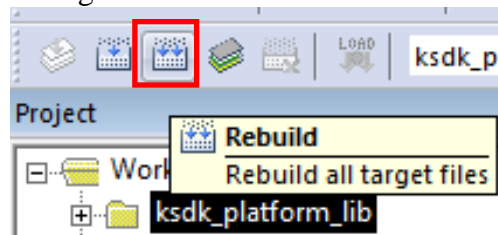


Figure 14 Rebuild all button

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. 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.lib file is in <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 platform_lib.a library should be in this folder:

<Install_dir>/lib/ksdk_platform_lib/uv4/KV31F51212/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>.uvmpw

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

<install_dir>/demos/hello_world/uv4/twrkv31f120m/hello_world.uvmpw

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

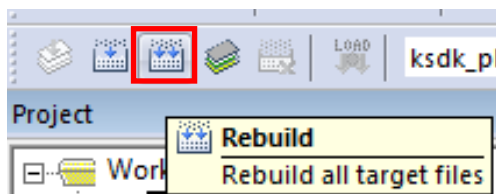


Figure 15 Build the demo application

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

```
Build Output
Rebuild Project 'hello_world' - Target 'hello_world Debug'
compiling fsl_misc_utilities.c...
compiling fsl_debug_console.c...
assembling startup_MKV31F51212.s...
compiling system_MKV31F51212.c...
compiling startup.c...
compiling hello_world.c...
compiling fsl_uart_irq.c...
compiling gpio_pins.c...
compiling pin_mux.c...
compiling hardware_init.c...
linking...
Program Size: Code=15828 RO-data=1248 RW-data=36 ZI-data=32948
"debug\hello world.out" - 0 Error(s), 0 Warning(s).
```

Figure 16 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 KV31 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

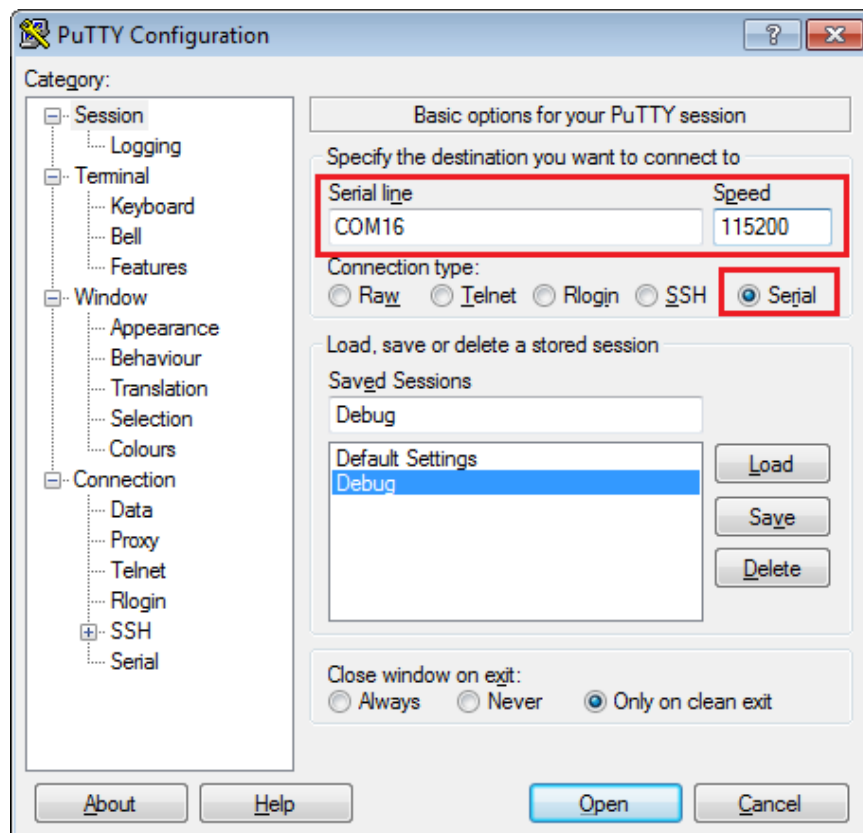


Figure 17 Terminal (PuTTY) configurations

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

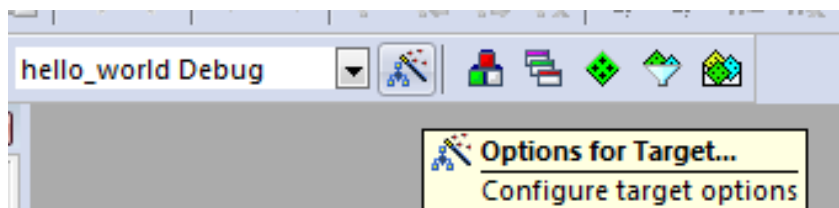


Figure 30 Options for Target dialog button

- b. In the “Options for Target ...” dialog box, select the Debug tab. Ensure that the simulator is not selected and the correct debug driver is selected. If the target is a TWR-KV31F120M Tower development card, the PEMicro Debugger should be selected.

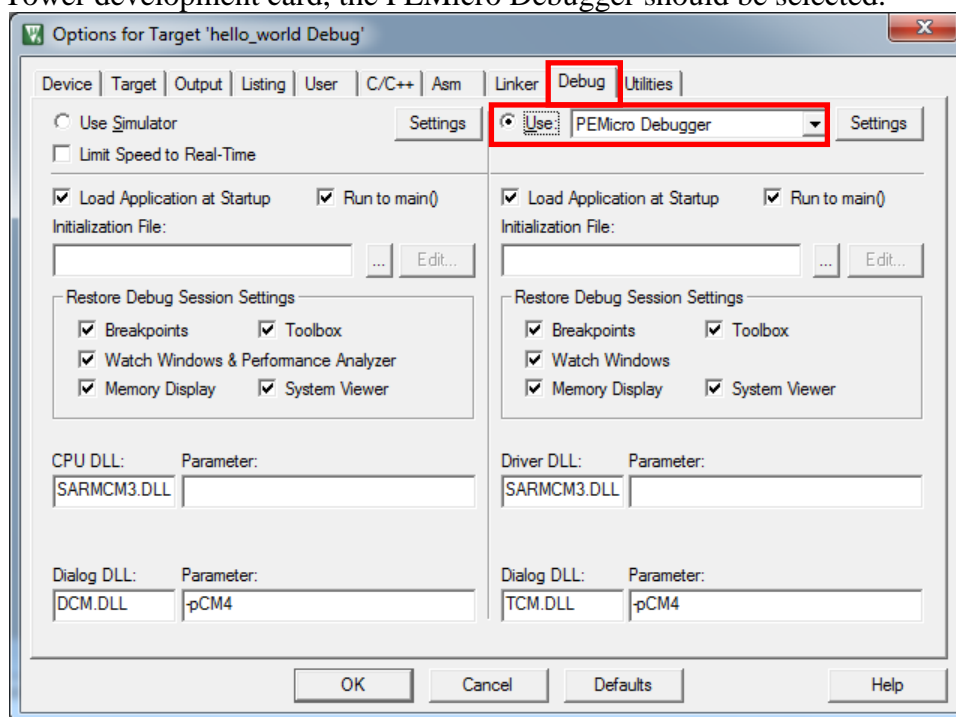


Figure 31 Debug driver selection for TWR-KV31F120M Tower System module

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

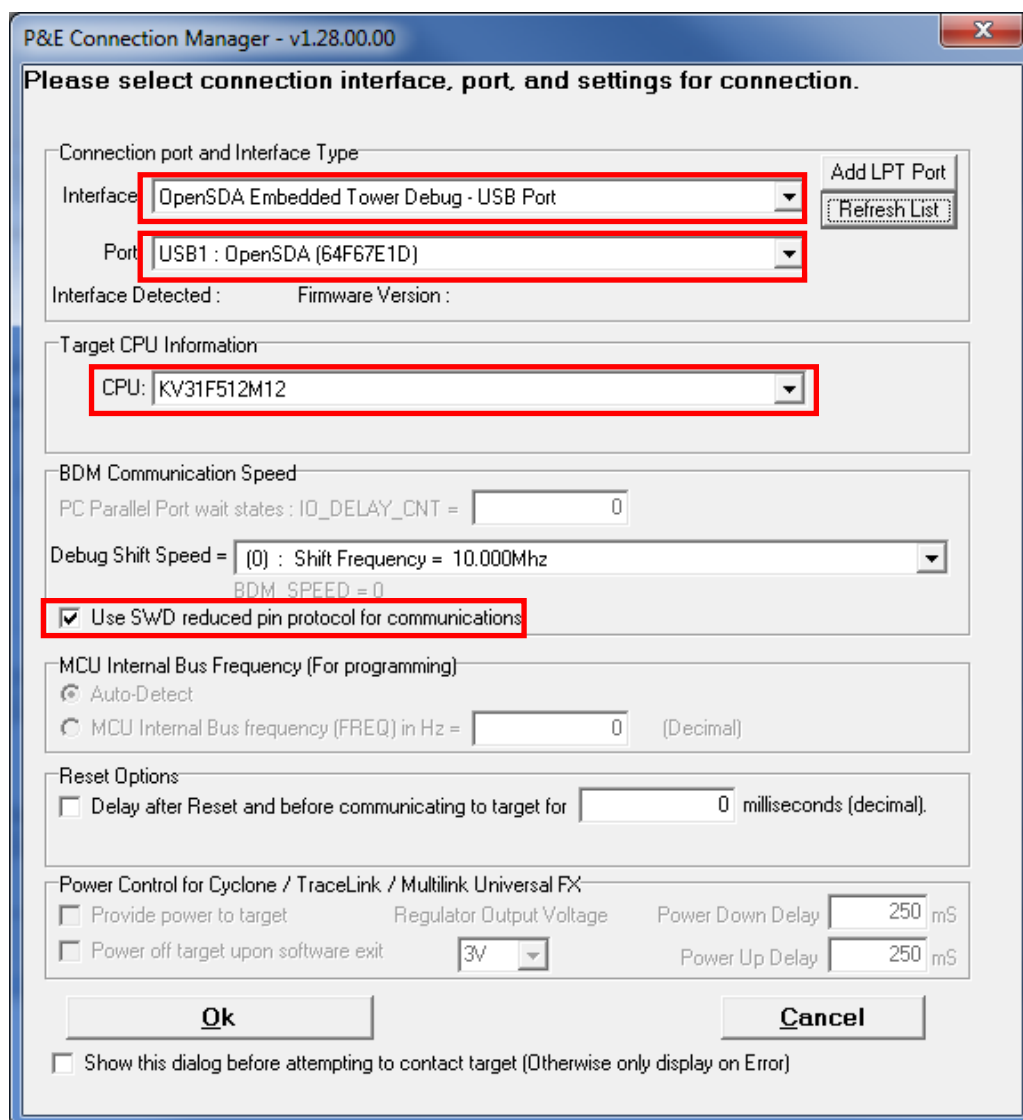


Figure 32 Debugger configurations for TWR-KV31F120M Tower System module

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

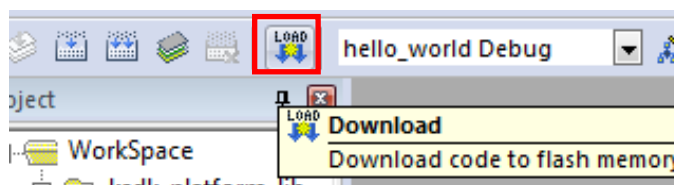


Figure 33 Download Button

- After clicking the “Download button”, the application has been downloaded to the target and should be running. To debug the application, click the “Start/Stop Debug Session button”.

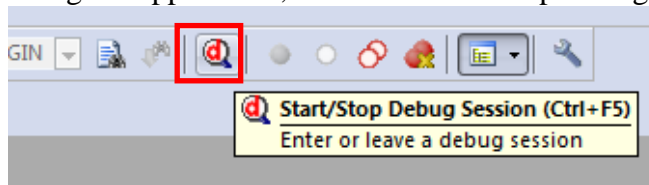


Figure 18 Start/Stop Debug Session Button

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

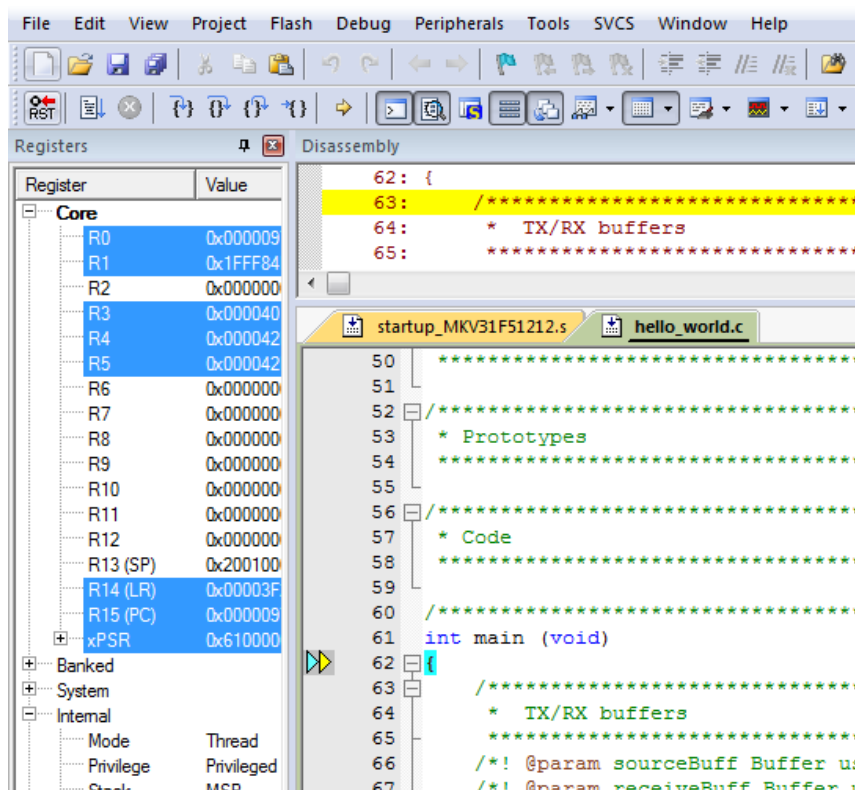


Figure 35 Stop at main() when debugging

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

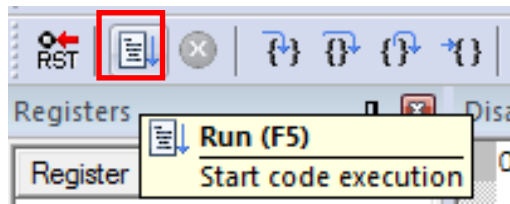


Figure 19 Go Button

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.

```
Hello World!
```



Figure 20 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 Freescale KSDK using Kinetis Design Studio. The hello_world demo application is used as an example (TWR-KV31F120M Tower System module hardware platform).

7.1 Installing KSDK Eclipse update

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

1. Select Help > Install New Software.

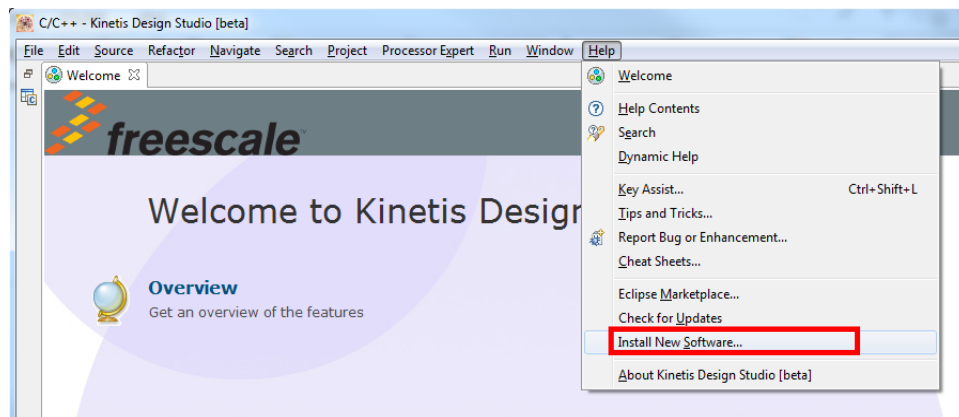


Figure 38 Install new software

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...”

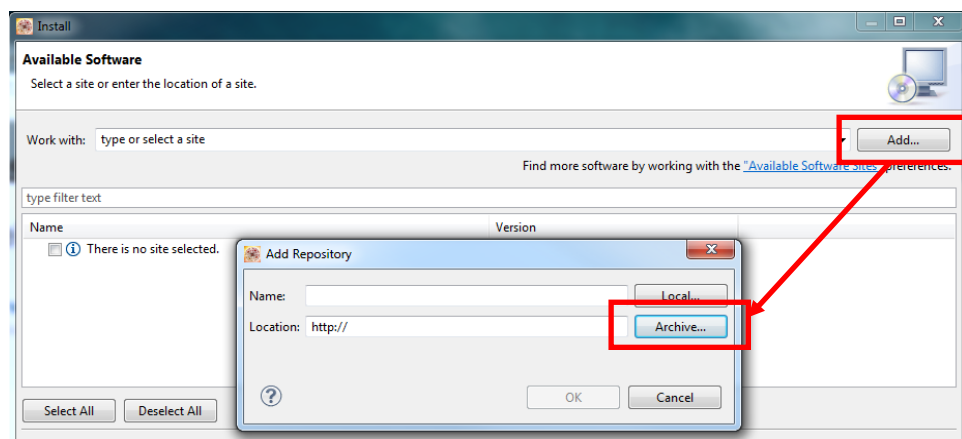


Figure 39 Add repository

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 SDK_version_Update_for_Eclipse.zip file.
6. Click “Open”, and then “OK” in the “Add Repository” dialog box. The KSDK update now shows up in the list of the original Install dialog.
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.

Once the update has been applied, restart the KDS/Eclipse for the changes to take effect.

7.2 Building the platform driver library in Kinetis Design Studio

The driver library for the target device should be built before building and debugging demo applications in the KSDK. The library archive for the KDS demo applications is named `ksdk_platform_lib.lib`. This library contains all HAL and peripheral driver functions which are device-specific. Each device has its own library (`ksdk_platform_lib.lib`). The platform library is not prebuilt, so 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.
2. Then select “Existing Projects into Workspace” from the General category in the Import window.

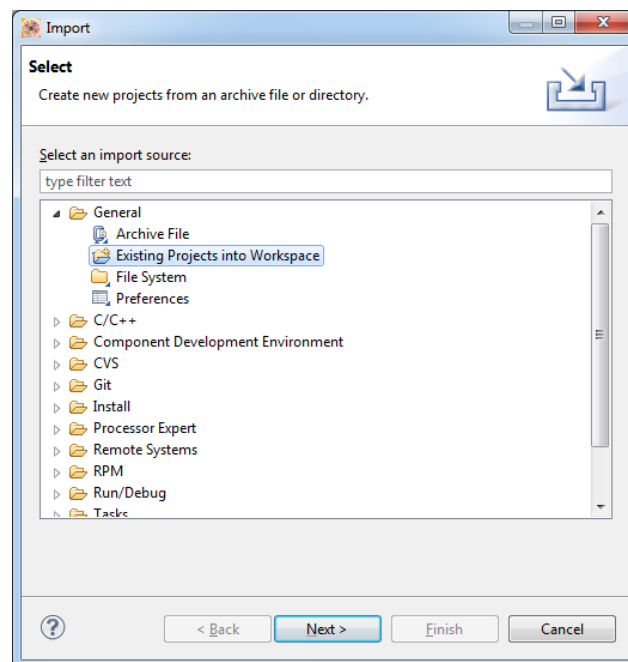


Figure 40 Selection of the correct import type in KDS

3. Ensure that “Select root directory:” option is selected and, then, click “Browse...” to point KDS to the correct library.

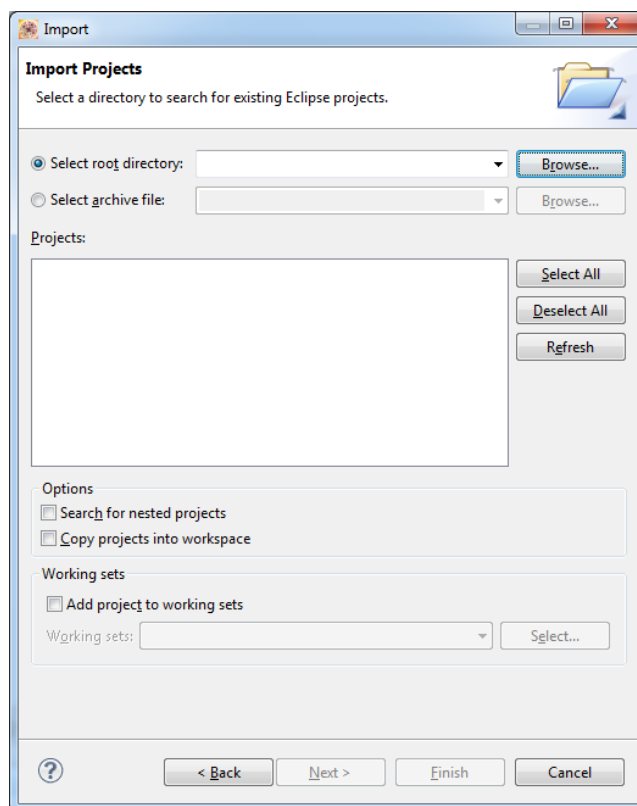


Figure 41 Import Projects directory selection window

4. Point KDS to the ksdk_platform_lib project in the KV31F51212 .The library project is located at <Install_dir>/lib/ksdk_platform_lib/kds/<device_name>. The Import Projects directory selection window looks as shown in the image.

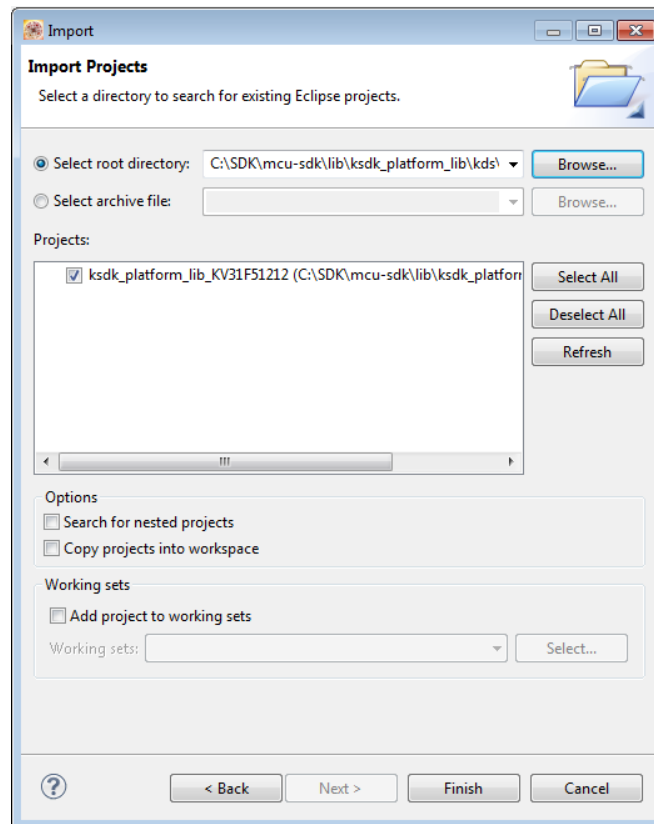


Figure 42 Import Projects directory selection window after selecting the KV31F51212 ksdk_platform_lib project.

5. Click Finish.

7.2.2 Build the library project

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

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

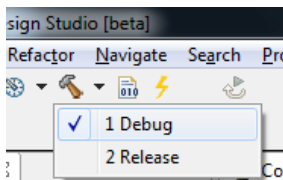


Figure 43 Selection of build target in KDS

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 <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/KV31F51212/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

Follow the steps in 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/twrkv31f120m/

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

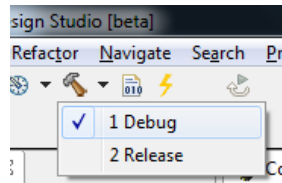


Figure 44 Selection of build target in KDS for the demo application

Left click the hammer icon if the target application does not begin building immediately. When the build is complete, the KDS console window shows this information:

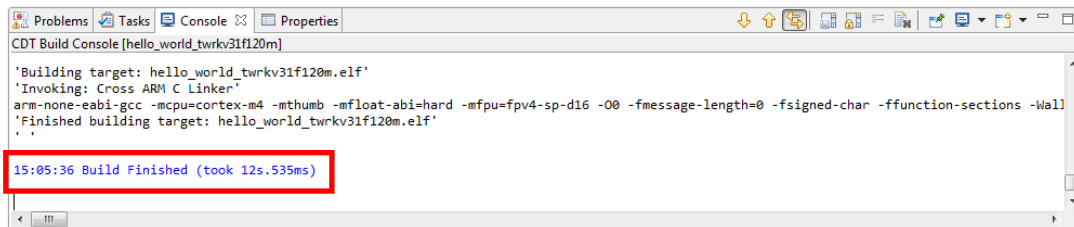


Figure 45 Console window output upon successful build

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-KV31F120M, the original OpenSDA J-Link firmware is needed. 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 KV31 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

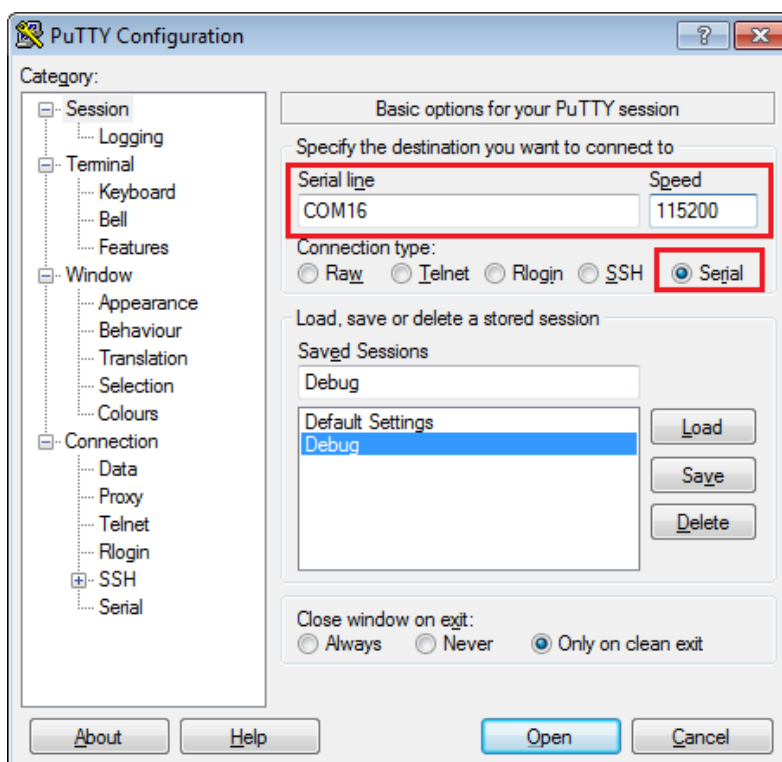


Figure 46 Terminal (PuTTY) configurations

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”.

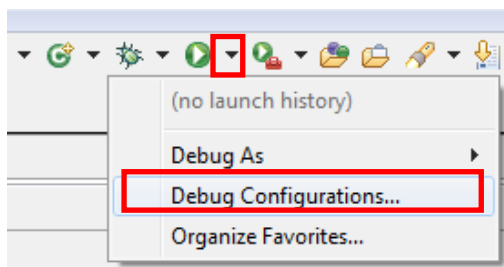


Figure 47 Debug Configurations dialog button

- 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.

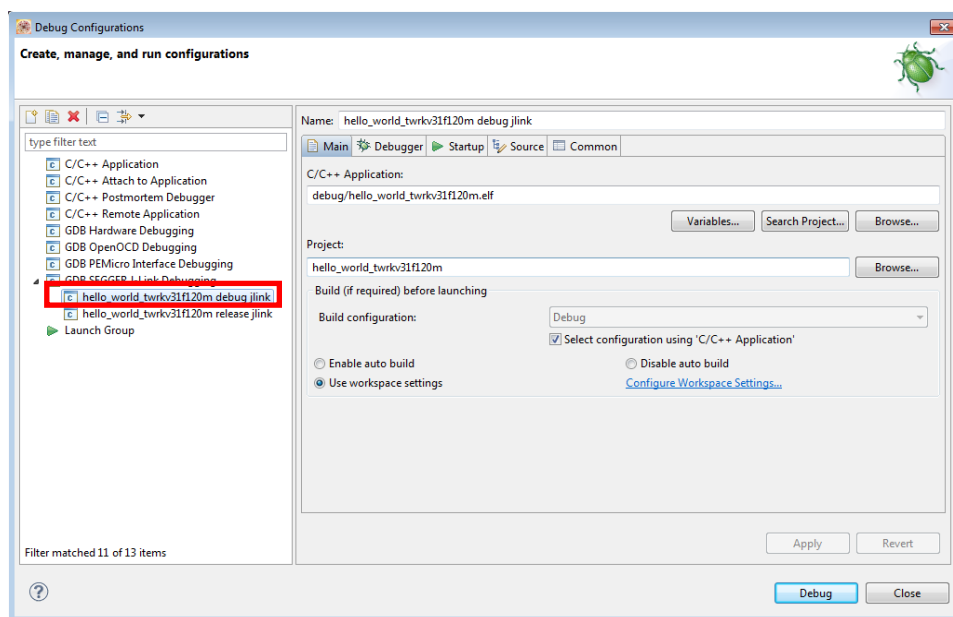


Figure 48 Selection of the debug configuration in the Debug Configurations dialog box.

- 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).

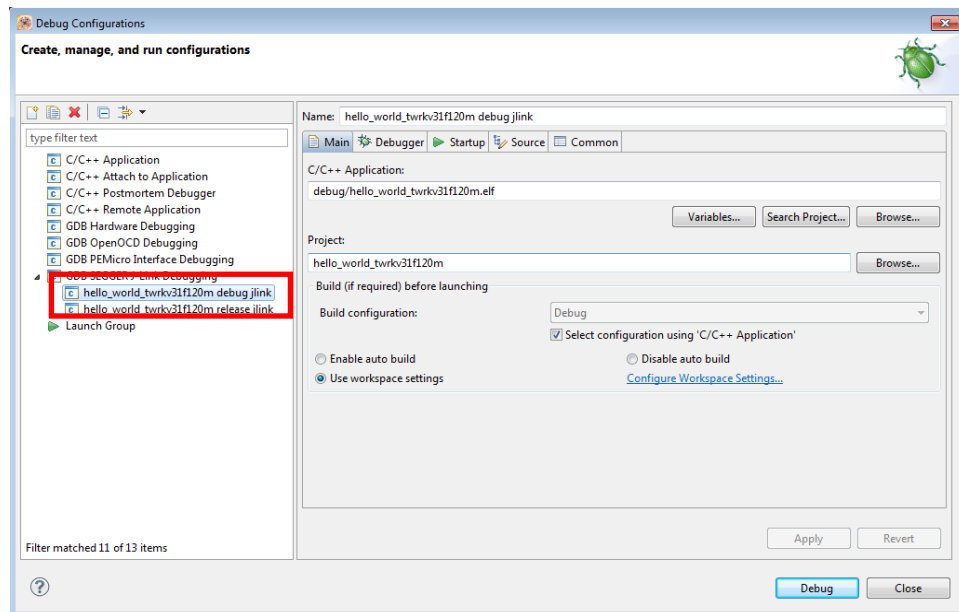


Figure 49 KDS debugger configurations for TWR-KV31F120M Tower System module

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

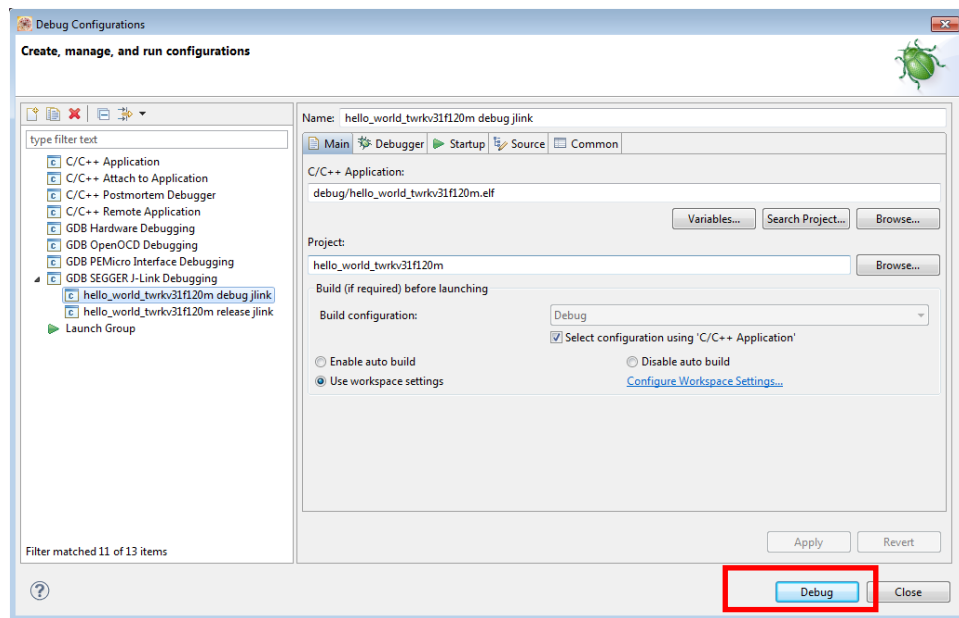


Figure 50 Debug button in Debug Configurations dialog box

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

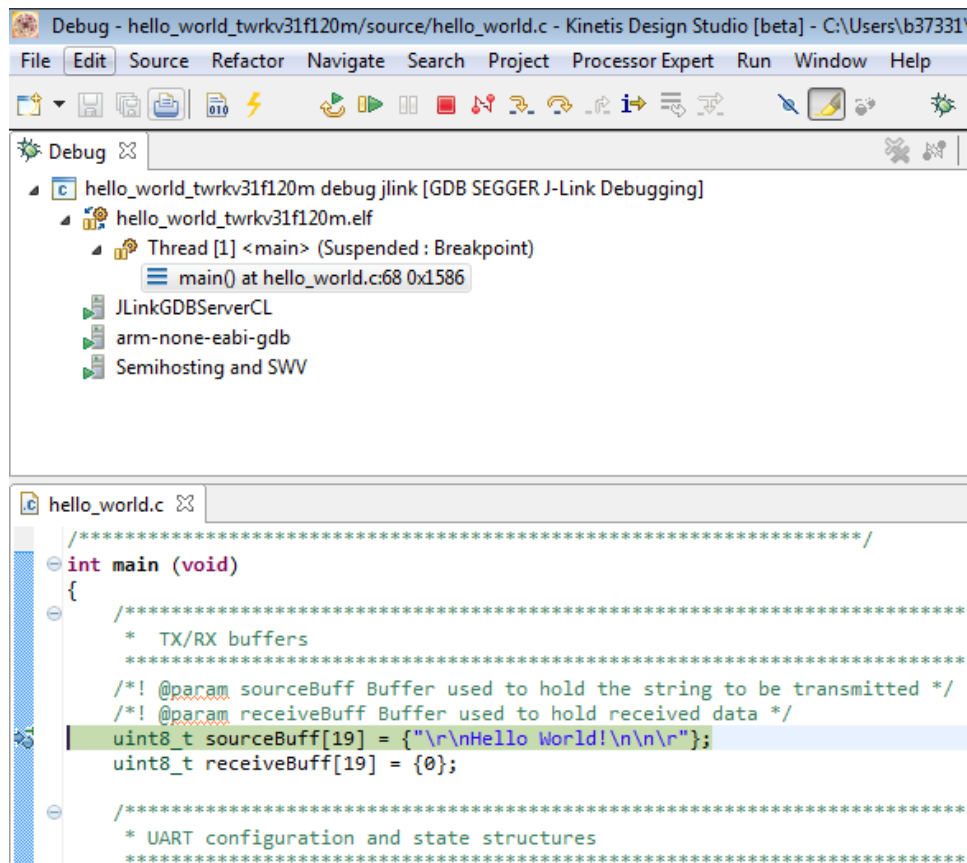


Figure 51 Stop at main() when run debugging

Run the code by clicking on the “resume button” to start the application:



Figure 52 Resume Button

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

A screenshot of a terminal window with a yellow background. The text "Hello World!" is displayed in a monospaced font. Below the text, a small black square represents the cursor, indicating the prompt for user input.

Figure 53 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

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. Tower is a trademark of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. ARM and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

© 2014 Freescale Semiconductor, Inc.

