

# Getting Started with MCUXpresso SDK Supporting FRDM-KE16Z

## 1 Overview

The MCUXpresso Software Development Kit (SDK) provides comprehensive software support for Kinetis and LPC Microcontrollers. The MCUXpresso SDK includes a flexible set of peripheral drivers designed to speed up and simplify development of embedded applications. Along with the peripheral drivers, the MCUXpresso SDK provides an extensive and rich set of example applications covering everything from basic peripheral use case examples to full demo applications. The MCUXpresso SDK contains FreeRTOS and various other middleware to support rapid development.

For supported toolchain versions, see the *MCUXpresso SDK Release Notes Supporting FRDM-KE16Z* (document MCUXSDKKE16RN).

For the latest version of this and other MCUXpresso SDK documents, see the MCUXpresso SDK homepage [MCUXpresso-SDK: Software Development Kit for MCUXpresso](#).

### Contents

1	Overview.....	1
2	MCUXpresso SDK board support folders.....	2
3	Run a demo using MCUXpresso IDE.....	4
4	Run a demo application using IAR.....	13
5	Run a demo using Keil® MDK/µVision.....	17
6	Run a demo using Arm® GCC.....	20
7	MCUXpresso Config Tools.....	29
8	MCUXpresso IDE New Project Wizard.....	30
9	Appendix A - How to determine COM port.....	31
10	Appendix B - Default debug interfaces .....	33
11	Appendix C - Updating debugger firmware.....	33





**Figure 1. MCUXpresso SDK layers**

## 2 MCUXpresso SDK board support folders

MCUXpresso SDK board support provides example applications for NXP development and evaluation boards for Arm® Cortex®-M cores, including Freedom, Tower System, and LPCXpresso boards. Board support packages are found inside of the top level boards folder, and each supported board has its own folder (an MCUXpresso SDK package can support multiple boards). Within each <board\_name> folder, there are various sub-folders to classify the type of examples they contain. These include (but are not limited to):

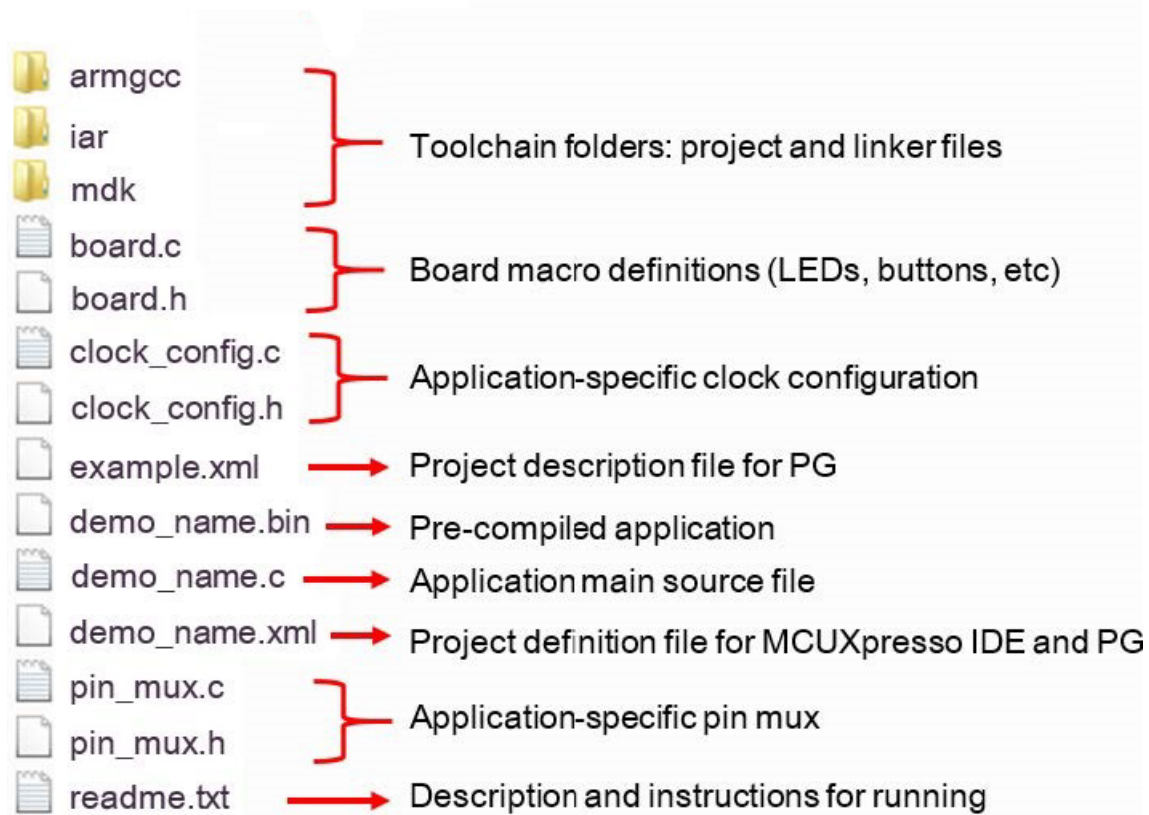
- cmsis\_driver\_examples: Simple applications intended to concisely illustrate how to use CMSIS drivers.
- demo\_apps: Full-featured applications intended to highlight key functionality and use cases of the target MCU. These applications typically use multiple MCU peripherals and may leverage stacks and middleware.
- driver\_examples: Simple applications intended to concisely illustrate how to use the MCUXpresso SDK's peripheral drivers for a single use case. These applications typically only use a single peripheral, but there are cases where multiple are used (for example, SPI conversion using DMA).
- rtos\_examples: Basic FreeRTOS™ OS examples showcasing the use of various RTOS objects (semaphores, queues, and so on) and interfacing with the MCUXpresso SDK's RTOS drivers

### 2.1 Example application structure

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

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

In the hello\_world application folder you see the following contents:



**Figure 2. Application folder structure**

All files in the application folder are specific to that example, so it is easy to copy and paste an existing example to start developing a custom application based on a project provided in the MCUXpresso SDK.

## 2.2 Locating example application source files

When opening an example application in any of the supported IDEs (except MCUXpresso IDE), a variety of source files are referenced. The MCUXpresso SDK devices folder is the central component to all example applications. It means the examples reference the same source files and, if one of these files is modified, it could potentially impact the behavior of other examples.

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

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

## Run a demo using MCUXpresso IDE

For examples containing an RTOS, there are references to the appropriate source code. RTOSes are in the *rtos* folder. Again, the core files of each of these are shared, so modifying them could have potential impacts on other projects that depend on them.

## 3 Run a demo using MCUXpresso IDE

### NOTE

Ensure that the MCUXpresso IDE toolchain is included when generating the MCUXpresso SDK Package.

This section describes the steps required to configure MCUXpresso IDE v10.3.0 to build, run, and debug example applications. The hello\_world demo application targeted for the FRDM-KE16Z hardware platform is used as an example, though these steps can be applied to any example application in the MCUXpresso SDK.

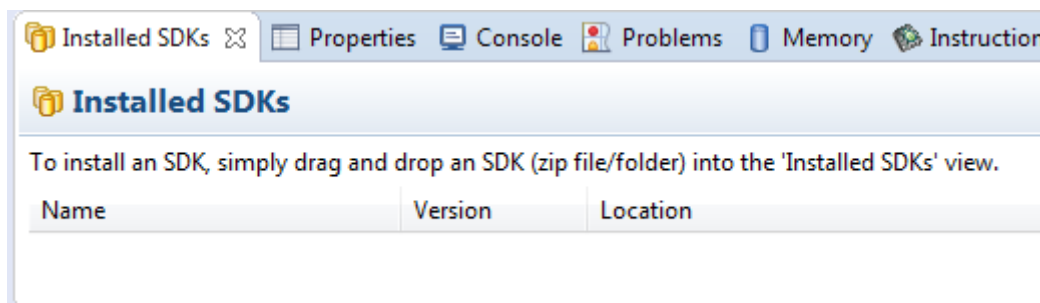
### 3.1 Select the workspace location

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

### 3.2 Build an example application

To build an example application, follow these steps.

1. Drag and drop the SDK zip file into the “Installed SDKs” view to install an SDK. In the window that appears, click the “OK” button and wait until the import has finished.



**Figure 3. Install an SDK**

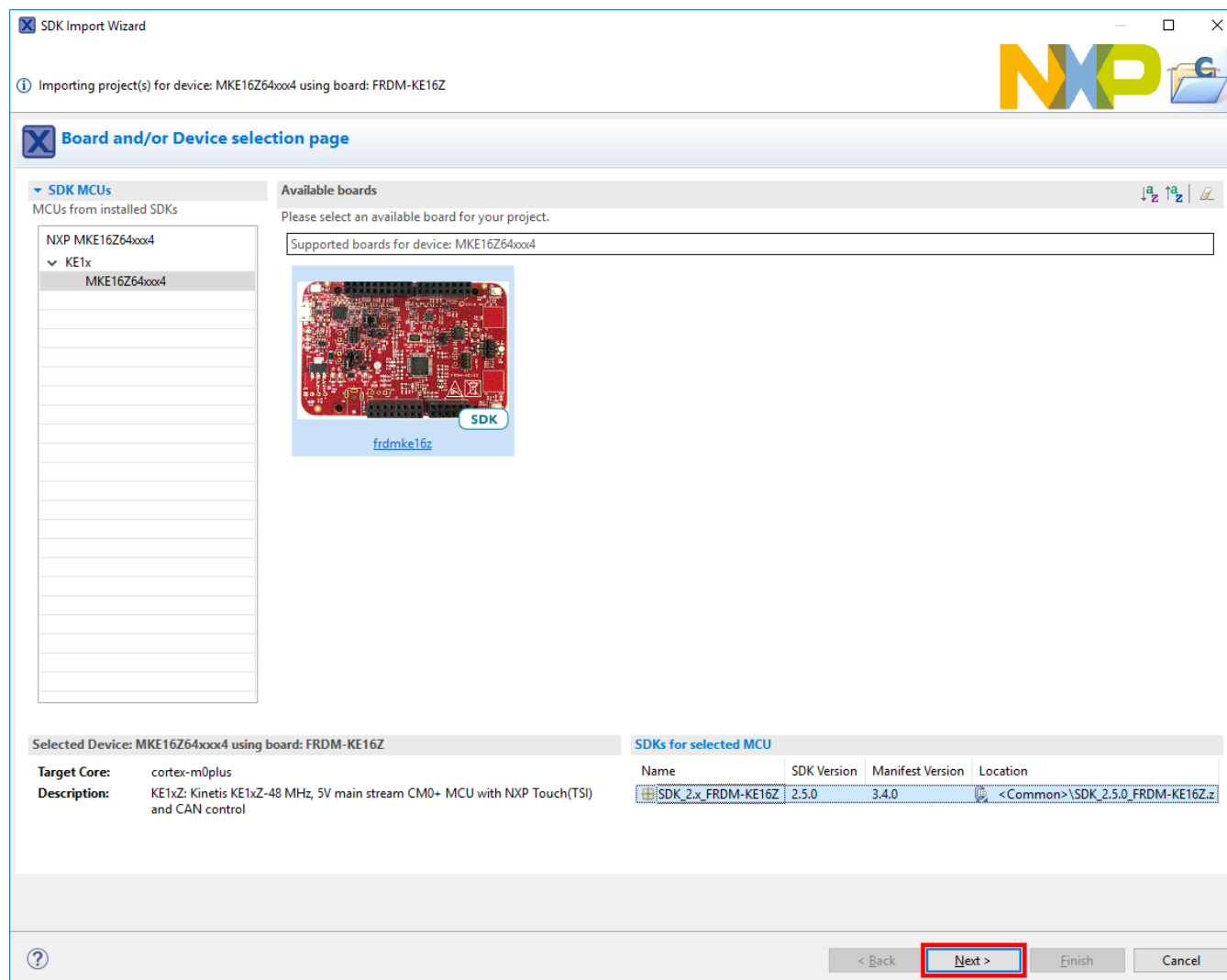
2. On the *Quickstart Panel*, click “Import SDK example(s)...”.



**Figure 4. Import an SDK example**

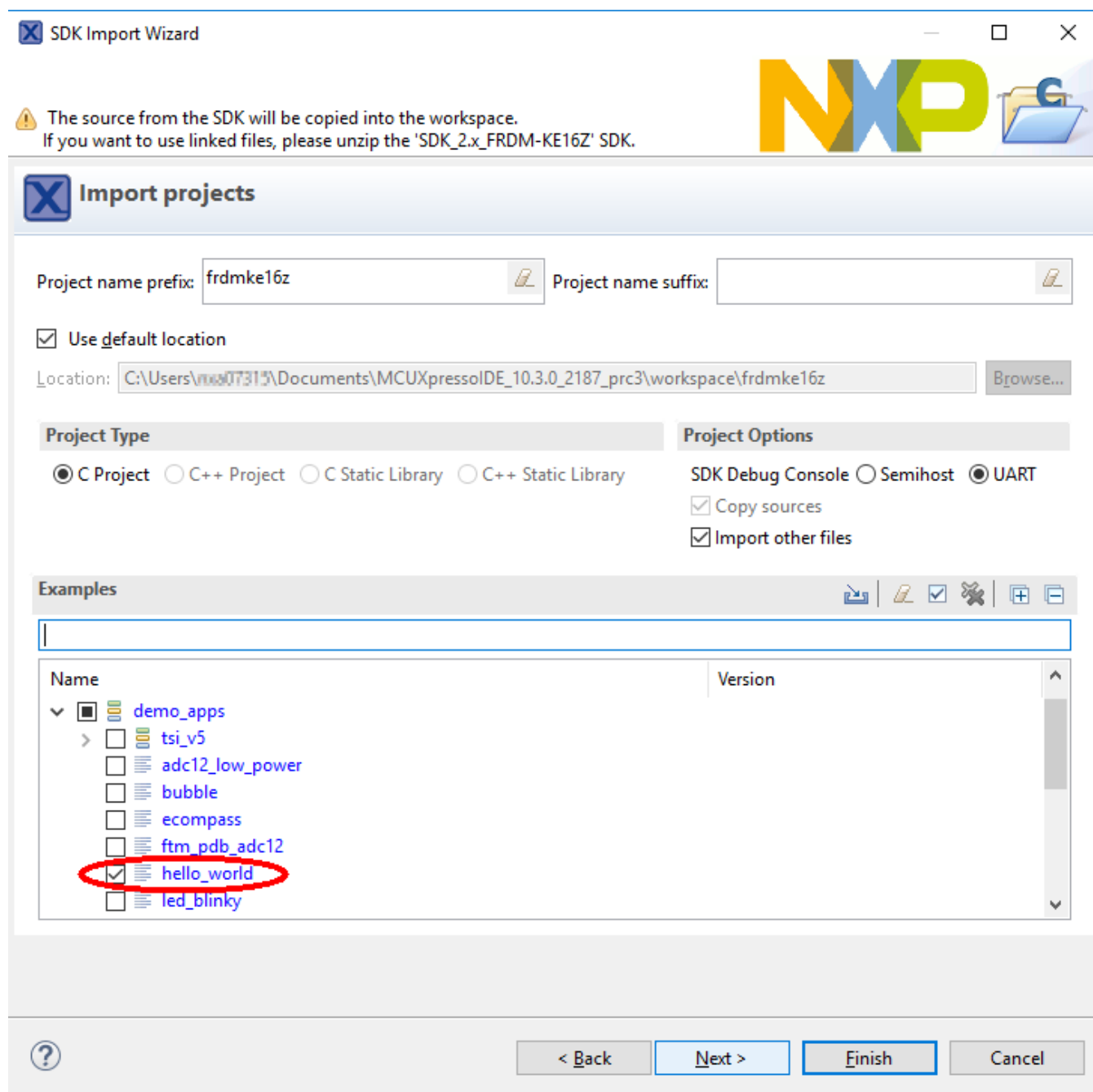
3. In the window that appears, expand the “KE1x” folder and select "MKE16Z64xxx4" . Then, select "frdmke16z" and click the “Next” button.

## Run a demo using MCUXpresso IDE



**Figure 5. Select FRDM-KE16Z board**

- Expand the “demo\_apps” folder and select “hello\_world”. Then, click the “Next” button.



**Figure 6. Select "hello\_world"**

5. Ensure the option "Redlib: Use floating point version of printf" is selected if the cases' print floating point numbers are on the terminal for demo applications such as adc\_basic, adc\_burst, adc\_dma, and adc\_interrupt. Otherwise, it is not necessary to select this option. Then, click the "Finish" button.

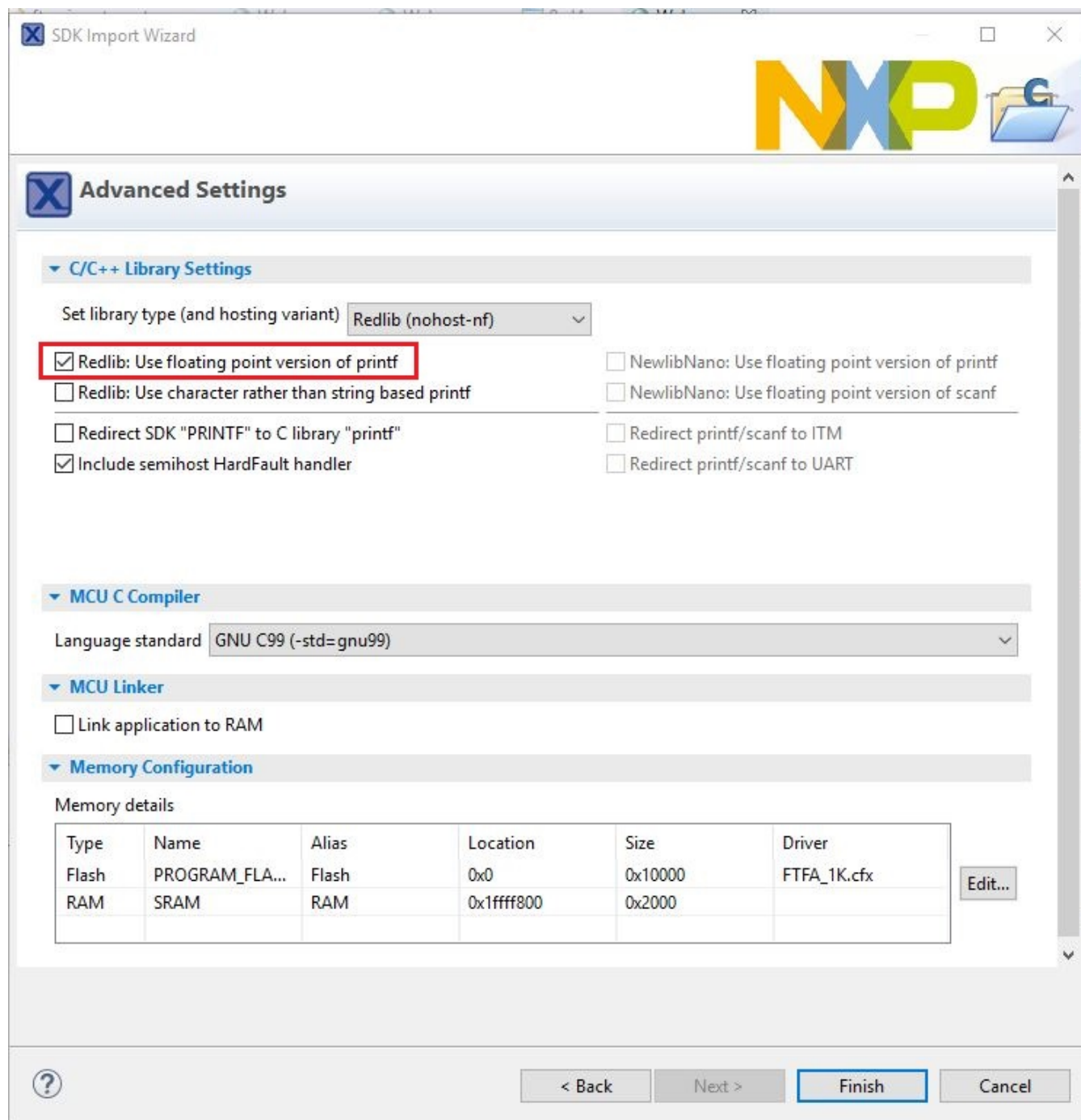


Figure 7. Select "User floating print version of printf"

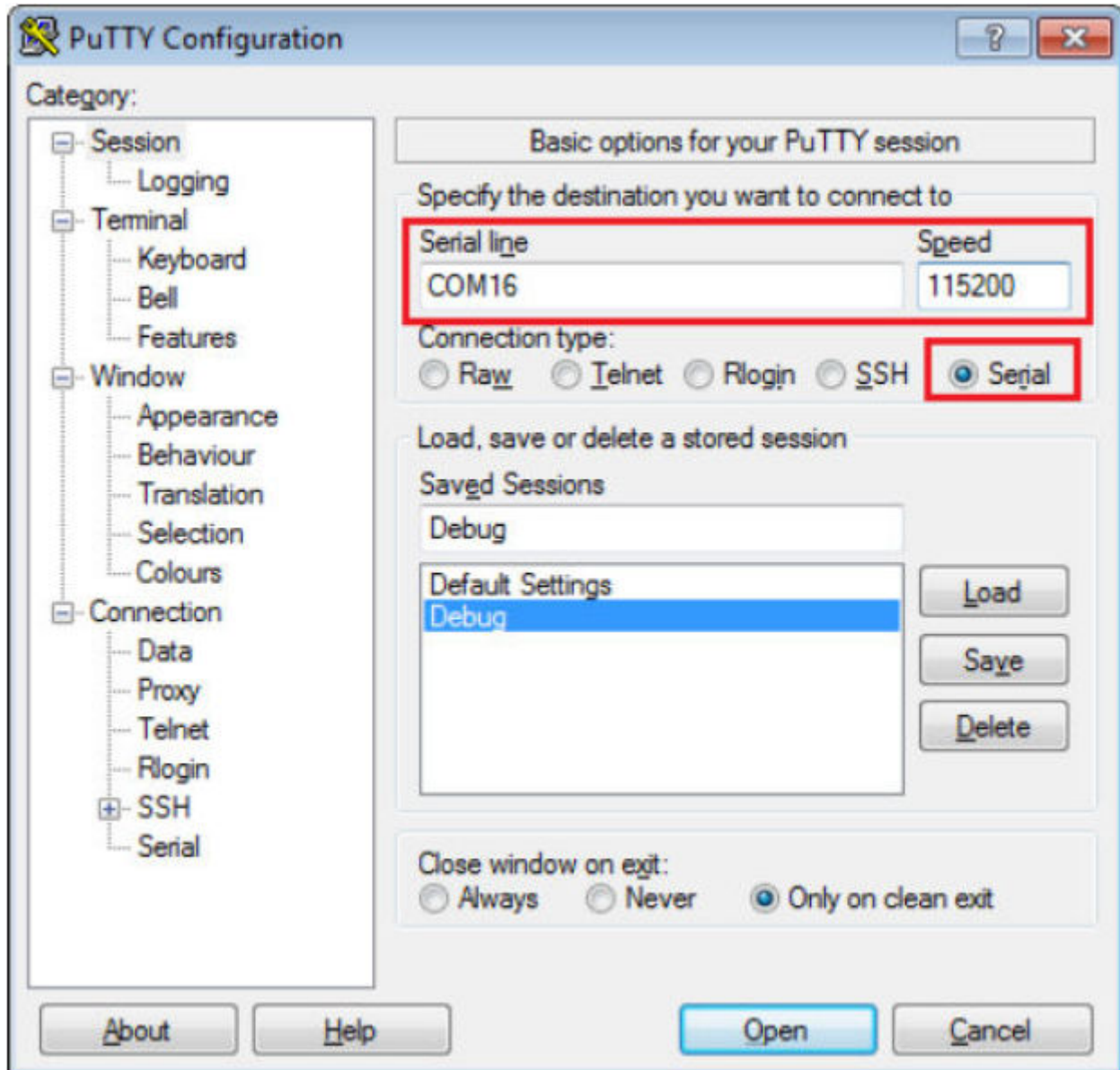
### 3.3 Run an example application

For more information on debug probe support in the MCUXpresso IDE v10.3.0, visit [community.nxp.com](https://community.nxp.com).

To download and run the application, perform these steps:



- Reference the table in Appendix B to determine the debug interface that comes loaded on your specific hardware platform.
  - For boards with a P&E Micro interface, visit [www.pemicro.com/support/downloads\\_find.cfm](http://www.pemicro.com/support/downloads_find.cfm) and download and install the P&E Micro Hardware Interface Drivers package.
  - If using J-Link with either a standalone debug pod or OpenSDA, install the J-Link software (drivers and utilities) from [www.segger.com/jlink-software.html](http://www.segger.com/jlink-software.html).
- Connect the development platform to your PC via USB cable.
- Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug serial port number (to determine the COM port number, see Appendix A). Configure the terminal with these settings:
  - 115200 or 9600 baud rate, depending on your board (reference BOARD\_DEBUG\_UART\_BAUDRATE variable in board.h file)
  - No parity
  - 8 data bits
  - 1 stop bit



**Figure 8. Terminal (PuTTY) configurations**

- On the *Quickstart Panel*, click on "Debug 'frdmke16z\_demo\_apps\_hello\_world' [Debug]".

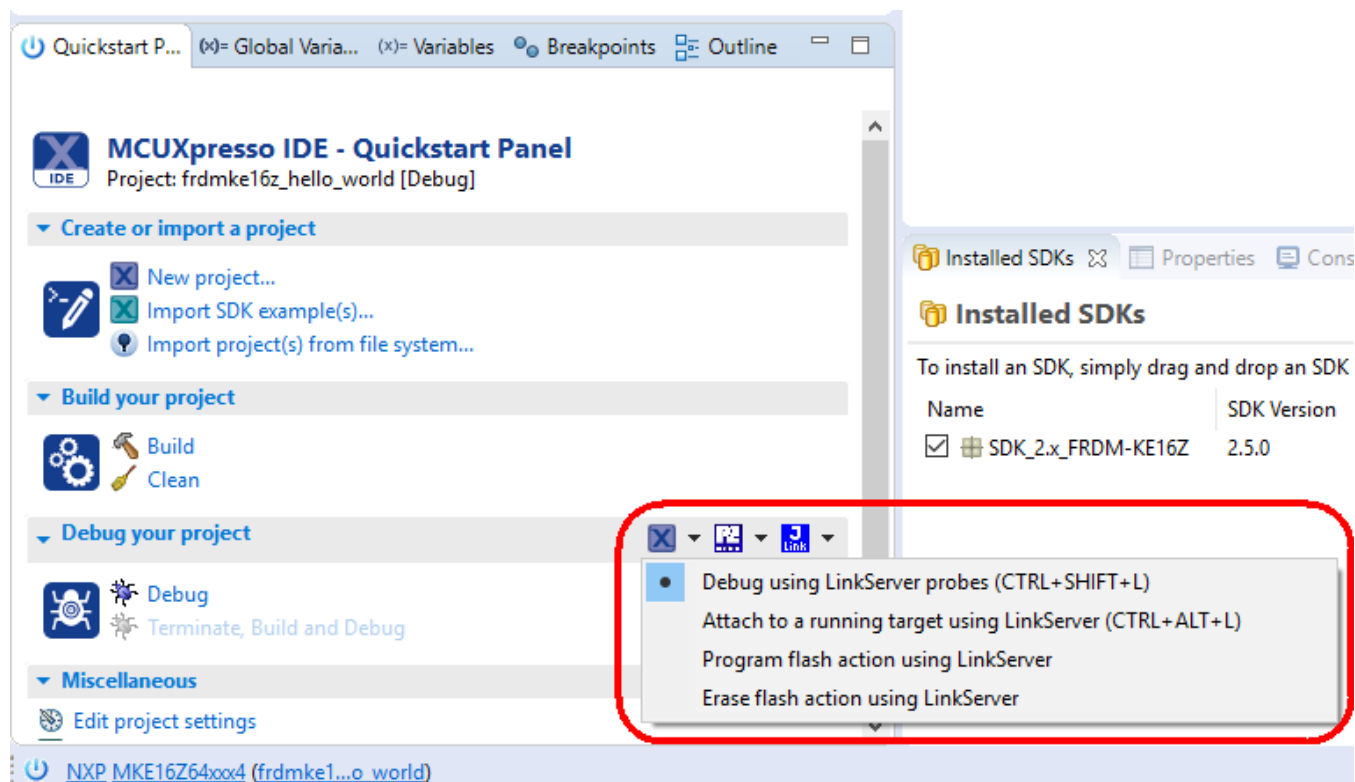
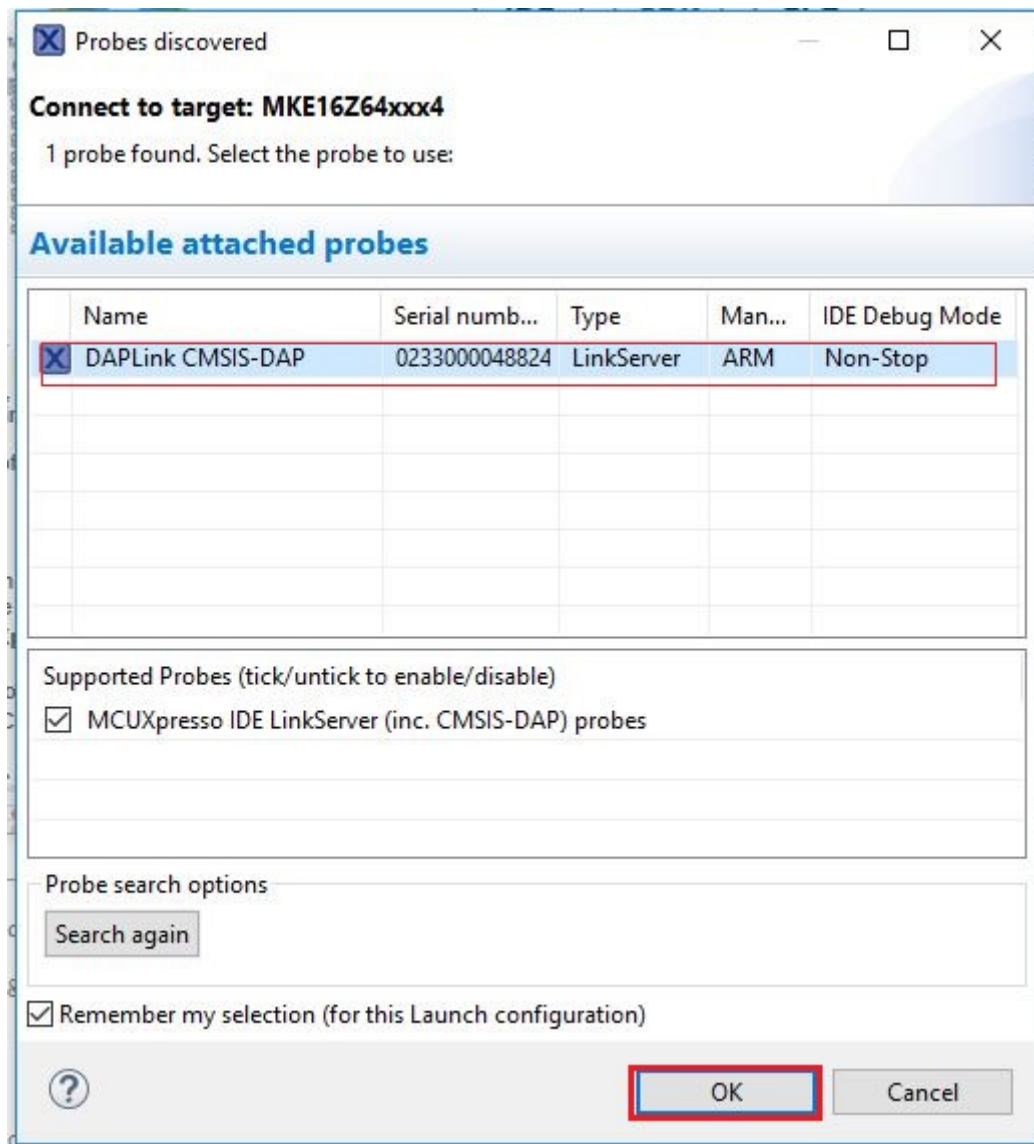


Figure 9. Debug "hello\_world" case

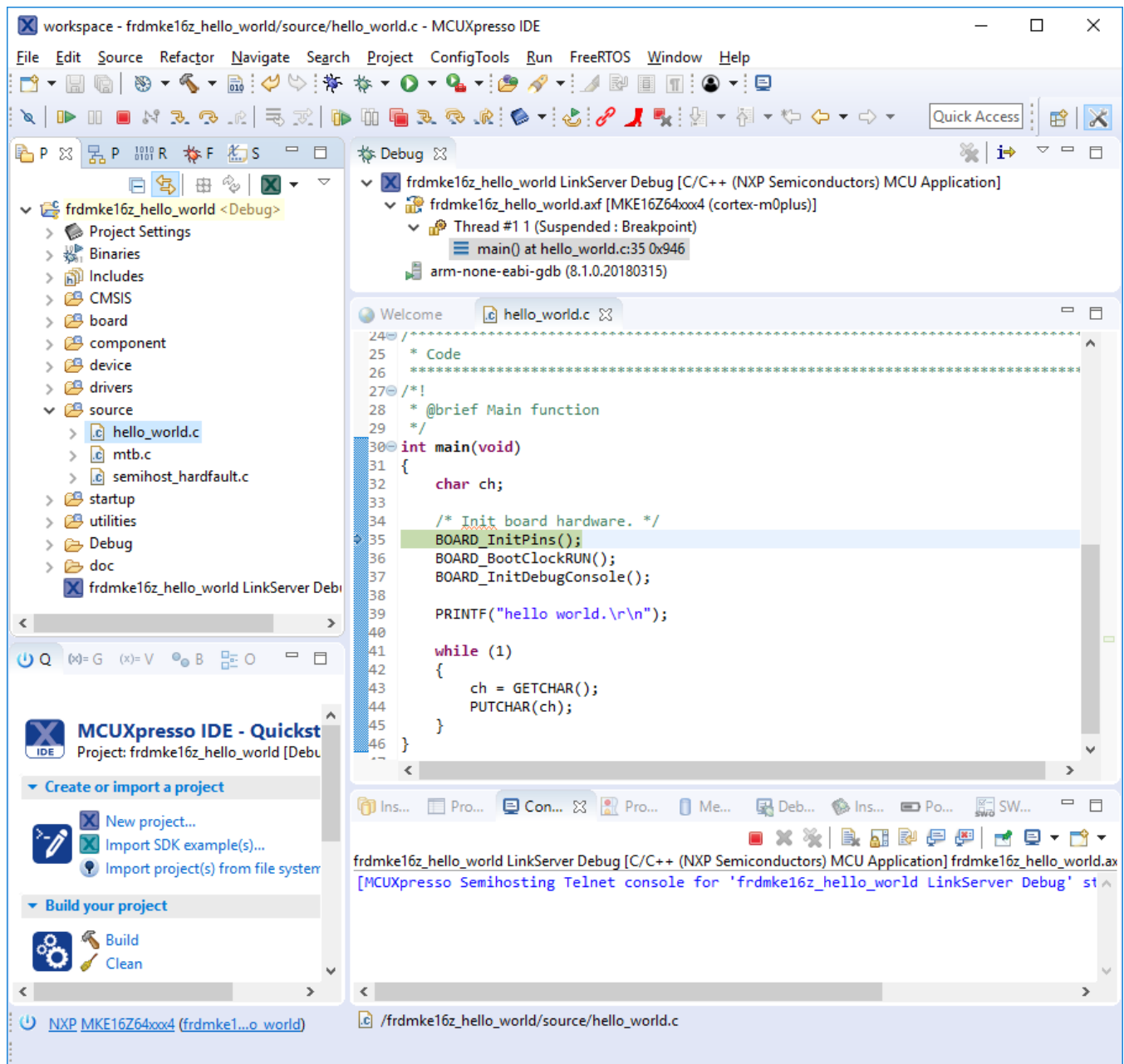
5. The first time you debug a project, the Debug Emulator Selection Dialog is displayed, showing all supported probes that are attached to your computer. Select the probe through which you want to debug and click the "OK" button. (For any future debug sessions, the stored probe selection is automatically used, unless the probe cannot be found.)



**Figure 10. Attached Probes: debug emulator selection**

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

## Run a demo using MCUXpresso IDE



**Figure 11. Stop at main() when running debugging**

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



**Figure 12. Resume button**

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

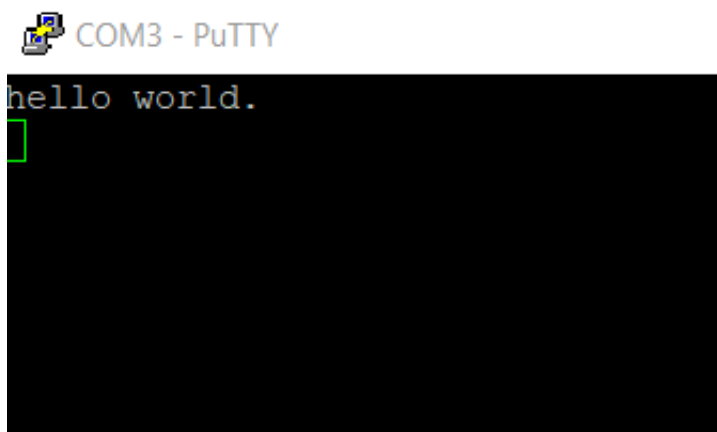


Figure 13. Text display of the hello\_world demo

## 4 Run a demo application using IAR

This section describes the steps required to build, run, and debug example applications provided in the MCUXpresso SDK.

### NOTE

IAR Embedded Workbench for Arm version 8.32.1 is used as an example to show below steps, and the IAR toolchain should correspond to the latest supported version, as described in the *MCUXpresso SDK Release Notes* (document MCUXSDKRN).

### 4.1 Build an example application

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

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

`<install_dir>/boards/<board_name>/<example_type>/<application_name>/iar`

Using the FRDM-KE16Z Freedom hardware platform as an example, the hello\_world workspace is located in

`<install_dir>/boards/frdmke16z/demo_apps/hello_world/iar/hello_world.eww`

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

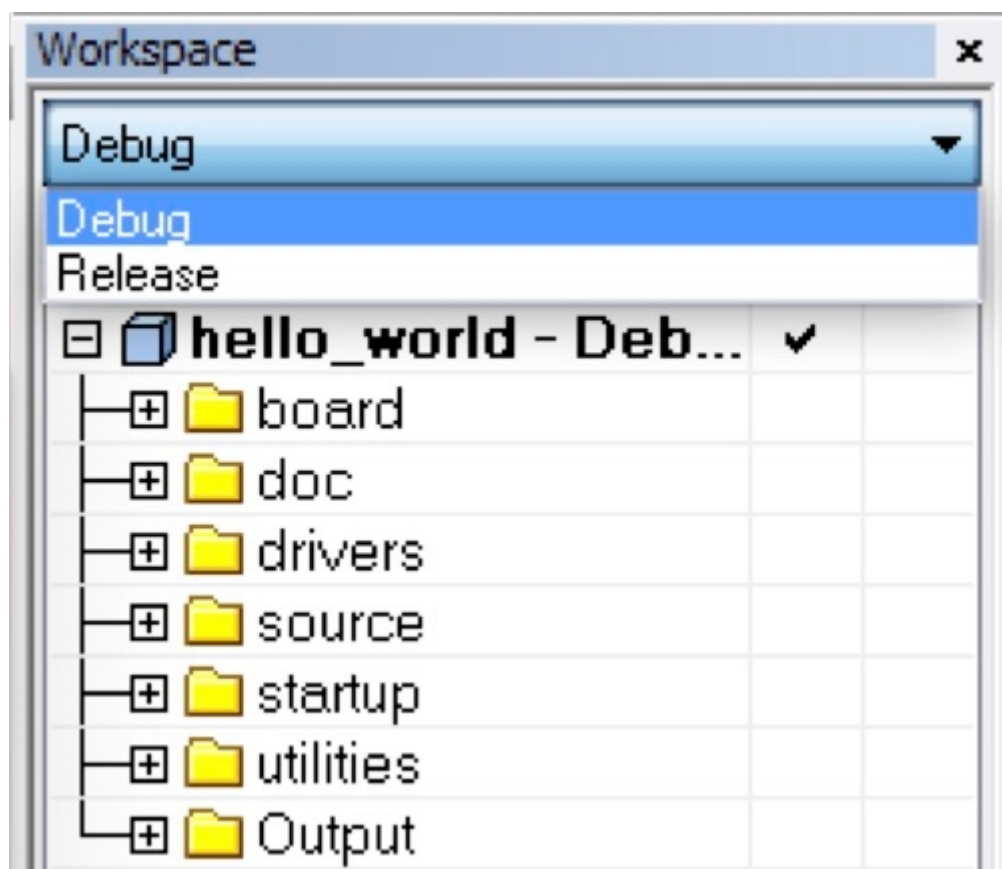


Figure 14. Demo build target selection

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

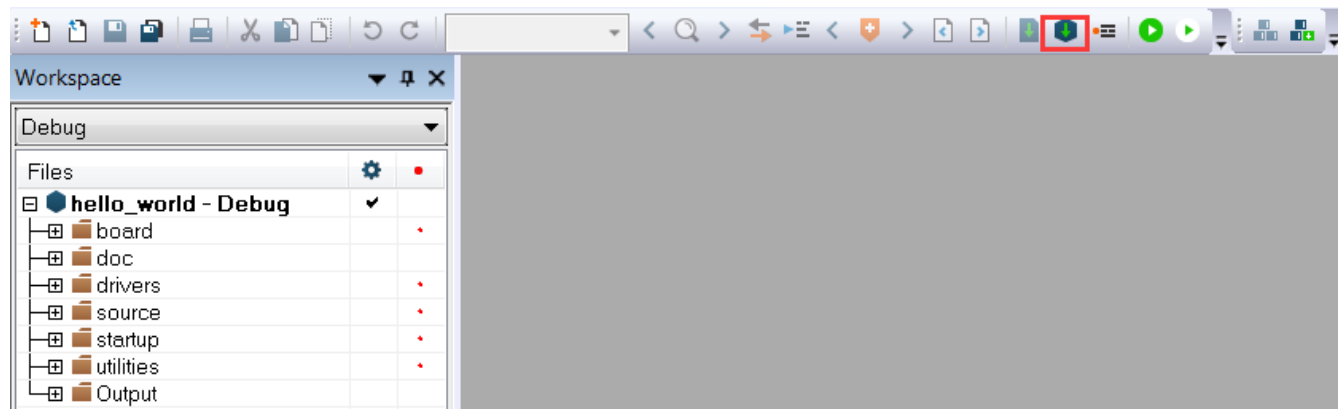


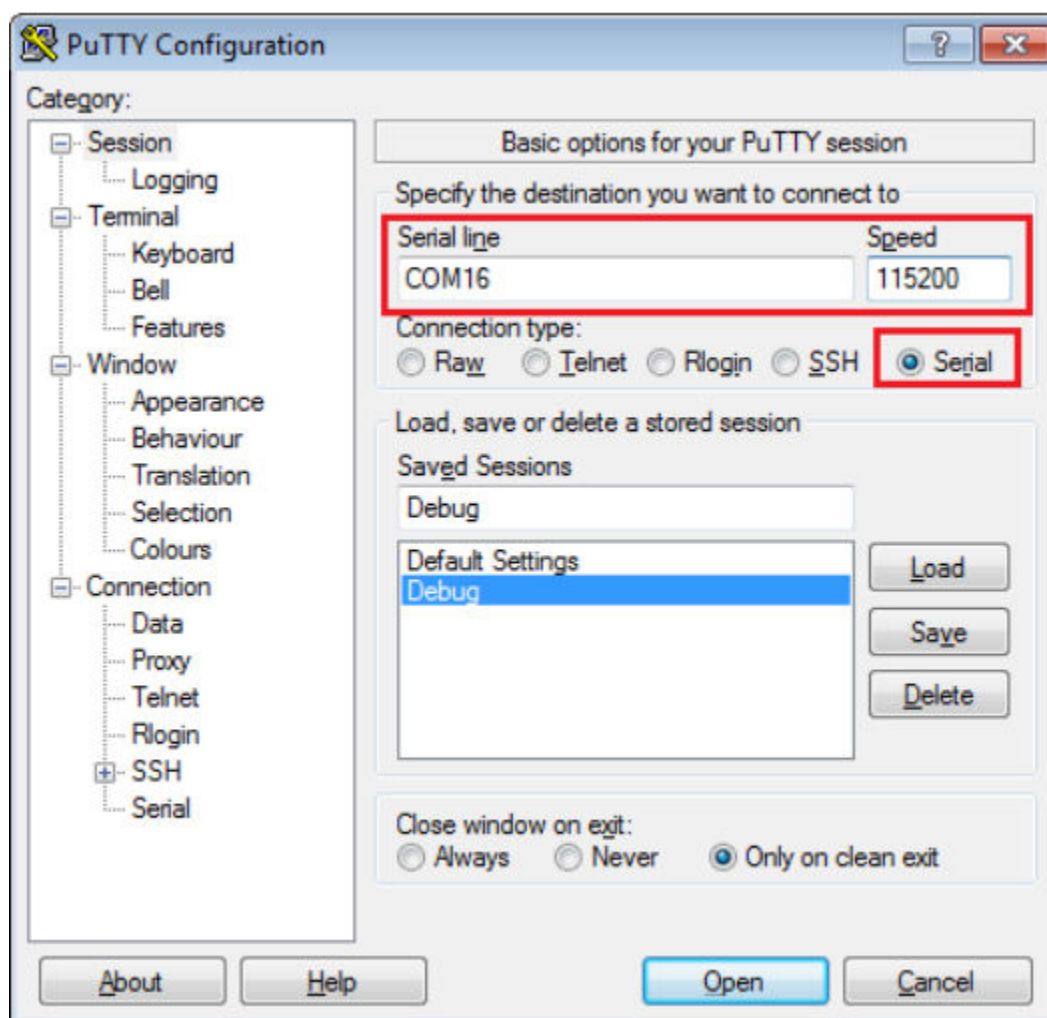
Figure 15. Build the demo application

4. The build completes without errors.

## 4.2 Run an example application

To download and run the application, perform these steps:

1. Reference the table in Appendix B to determine the debug interface that comes loaded on your specific hardware platform.
  - For boards with CMSIS-DAP/mbd/DAPLink interfaces, visit [developer.mbed.org/handbook/Windows-serial-configuration](http://developer.mbed.org/handbook/Windows-serial-configuration) and follow the instructions to install the Windows® operating system serial driver. If running on Linux® OS, this step is not required.
  - For boards with P&E Micro interfaces, visit [www.pemicro.com/support/downloads\\_find.cfm](http://www.pemicro.com/support/downloads_find.cfm) and download the P&E Micro Hardware Interface Drivers package.
2. Connect the development platform to your PC via USB cable.
3. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug COM port (to determine the COM port number, see Appendix A). Configure the terminal with these settings:
  - a. 115200 or 9600 baud rate, depending on your board (reference BOARD\_DEBUG\_UART\_BAUDRATE variable in board.h file)
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit



**Figure 16. Terminal (PuTTY) configuration**

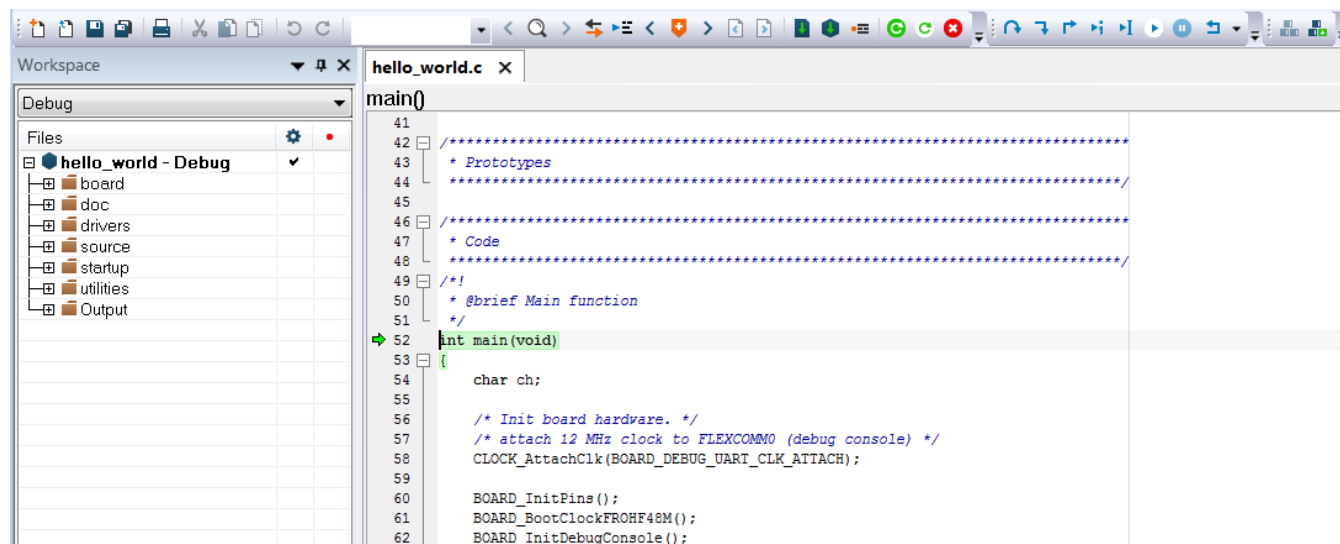
4. In IAR, click the "Download and Debug" button to download the application to the target.





**Figure 17. Download and Debug button**

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



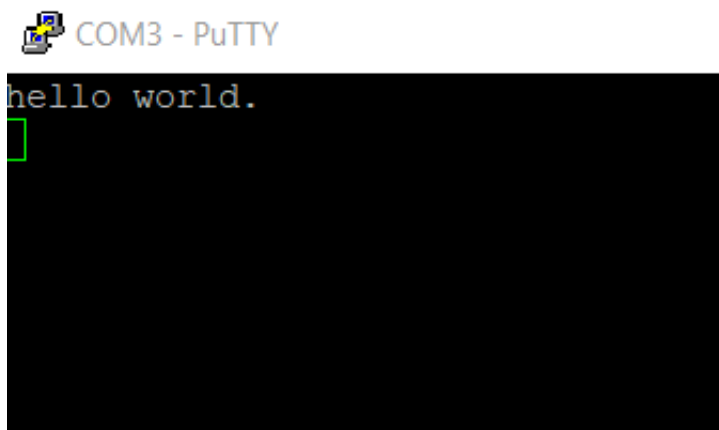
**Figure 18. Stop at main() when running debugging**

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



**Figure 19. Go button**

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



**Figure 20. Text display of the hello\_world demo**



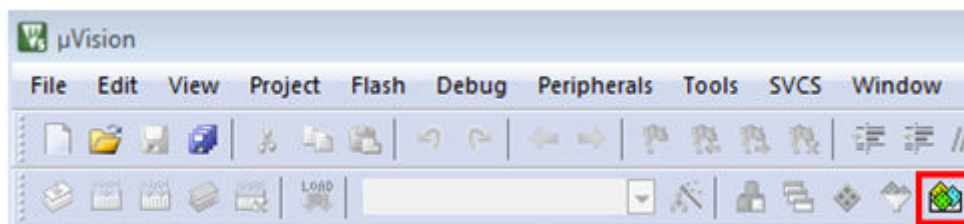
## 5 Run a demo using Keil® MDK/μVision

This section describes the steps required to build, run, and debug example applications provided in the MCUXpresso SDK. The hello\_world demo application targeted for the FRDM-KE16Z Freedom hardware platform is used as an example, although these steps can be applied to any demo or example application in the MCUXpresso SDK.

### 5.1 Install CMSIS device pack

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

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



**Figure 21. Launch the Pack installer**

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

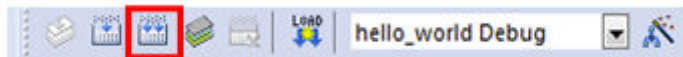
### 5.2 Build an example application

- Open the desired example application workspace in: `<install_dir>/boards/<board_name>/<example_type>/<application_name>/mdk`

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

`<install_dir>/boards/frdmke16z/demo_apps/hello_world/mdk/hello_world.uvmpw`

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



**Figure 22. Build the demo**

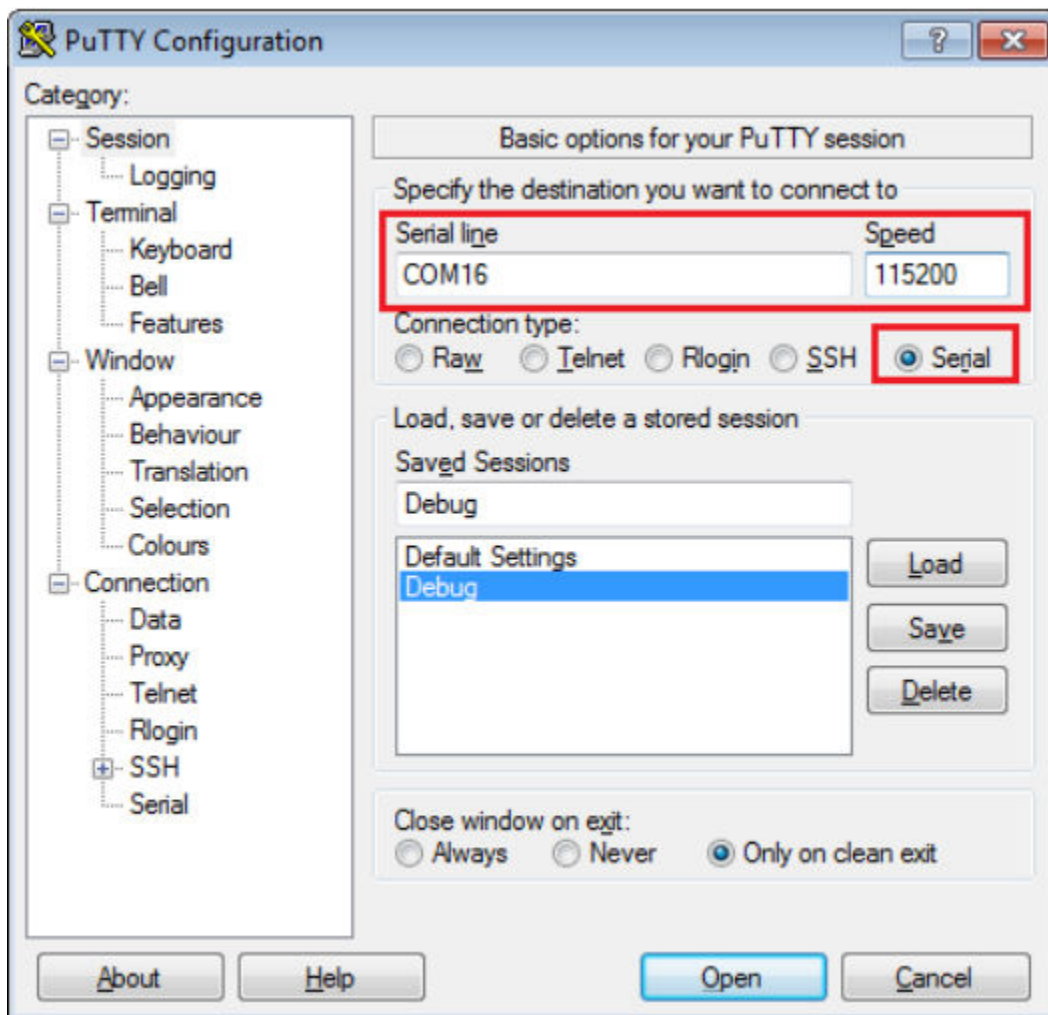
- The build completes without errors.

### 5.3 Run an example application

To download and run the application, perform these steps:

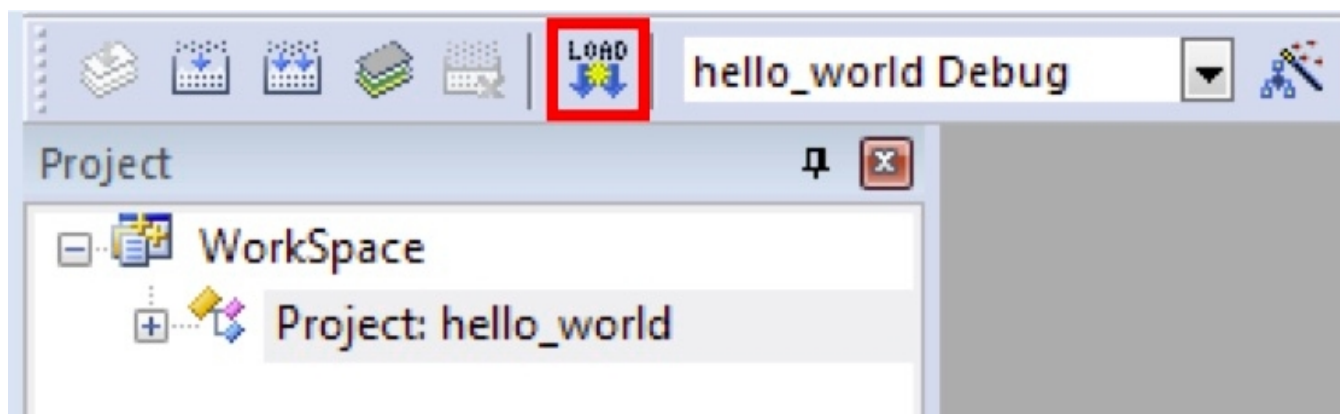
## Run a demo using Keil® MDK/μVision

1. Reference the table in Appendix B to determine the debug interface that comes loaded on your specific hardware platform.
  - For boards with the CMSIS-DAP/mbd/DAPLink interface, visit [mbed Windows serial configuration](#) and follow the instructions to install the Windows operating system serial driver. If running on Linux OS, this step is not required.
  - For boards with a P&E Micro interface, visit [www.pemicro.com/support/downloads\\_find.cfm](http://www.pemicro.com/support/downloads_find.cfm) and download and install the P&E Micro Hardware Interface Drivers package.
  - If using J-Link either a standalone debug pod or OpenSDA, install the J-Link software (drivers and utilities) from [www.segger.com/jlink-software.html](http://www.segger.com/jlink-software.html).
2. Connect the development platform to your PC via USB cable between the OpenSDA USB connector and the PC USB connector.
3. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug serial port number (to determine the COM port number, see Appendix A). Configure the terminal with these settings:
  - a. 115200 or 9600 baud rate, depending on your board (reference BOARD\_DEBUG\_UART\_BAUDRATE variable in board.h file)
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit



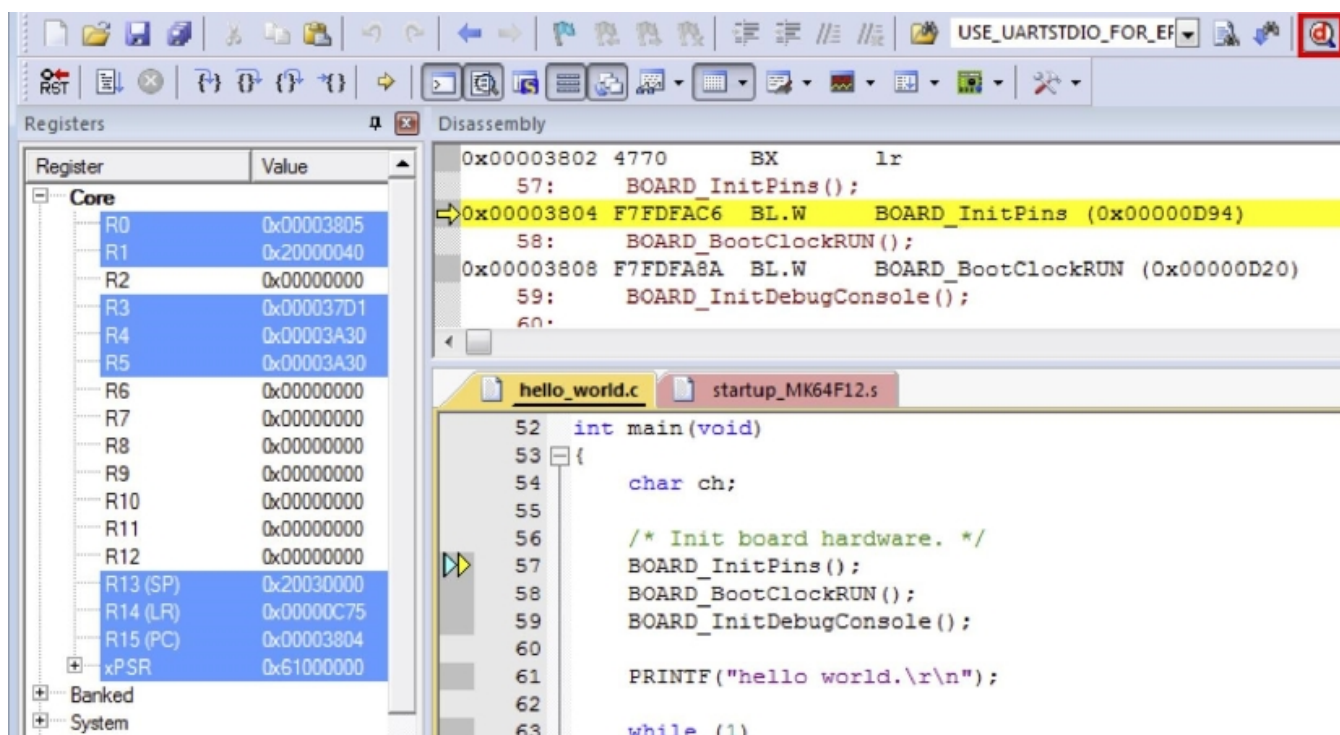
**Figure 23. Terminal (PuTTY) configurations**

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



**Figure 24. Download button**

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



**Figure 25. Stop at main() when run debugging**

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

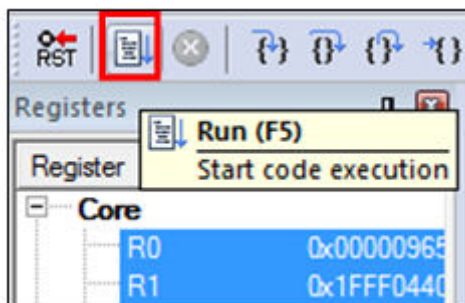
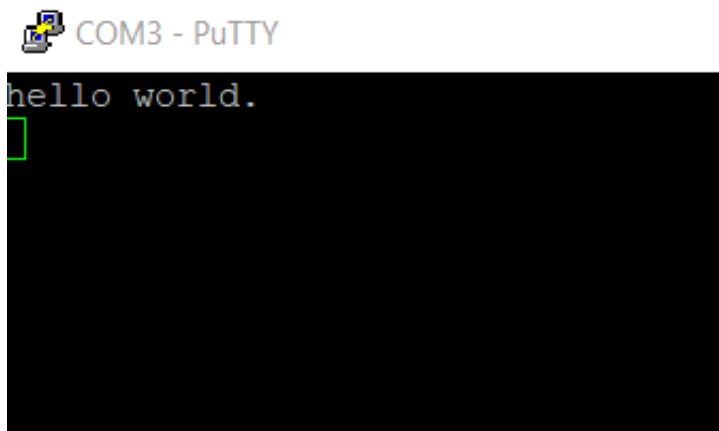


Figure 26. Go button

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

Figure 27. Text display of the `hello_world` demo

## 6 Run a demo using Arm® GCC

This section describes the steps to configure the command line Arm® GCC tools to build, run, and debug demo applications and necessary driver libraries provided in the MCUXpresso SDK. The `hello_world` demo application is targeted for the FRDM-KE16Z Freedom hardware platform is used as an example, though these steps can be applied to any board, demo or example application in the MCUXpresso SDK.

### NOTE

ARMGCC version 5.2.2015q4 is used as an example in this document, the latest GCC version for this package is as described in the *MCUXpresso SDK Release Notes* (document MCUXSDKRN).

### 6.1 Set up toolchain

This section contains the steps to install the necessary components required to build and run a MCUXpresso SDK demo application with the Arm GCC toolchain, as supported by the MCUXpresso SDK. There are many ways to use Arm GCC tools, but this example focuses on a Windows operating system environment.

### 6.1.1 Install GCC Arm Embedded tool chain

Download and run the installer from [developer.arm.com/open-source/gnu-toolchain/gnu-rm](http://developer.arm.com/open-source/gnu-toolchain/gnu-rm). This is the actual toolset (in other words, compiler, linker, and so on). The GCC toolchain should correspond to the latest supported version, as described in the *MCUXpresso SDK Release Notes*. (document MCUXSDKRN).

### 6.1.2 Install MinGW (only required on Windows OS)

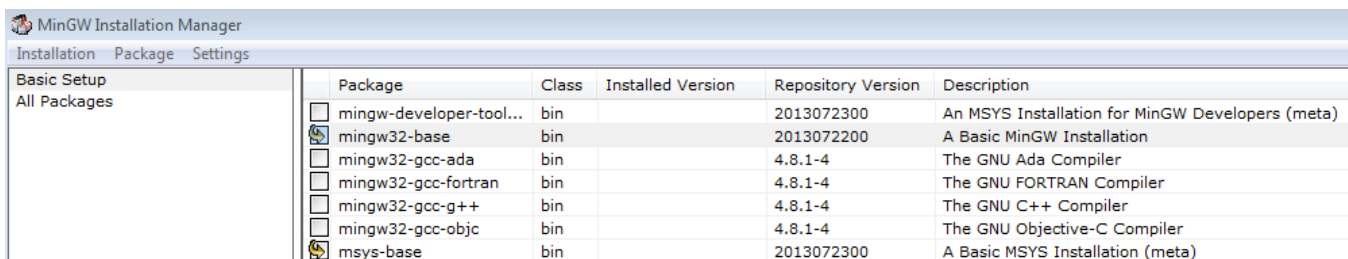
The Minimalist GNU for Windows (MinGW) development tools provide a set of tools that are not dependent on third party C-Runtime DLLs (such as Cygwin). The build environment used by the MCUXpresso SDK does not utilize the MinGW build tools, but does leverage the base install of both MinGW and MSYS. MSYS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

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

#### NOTE

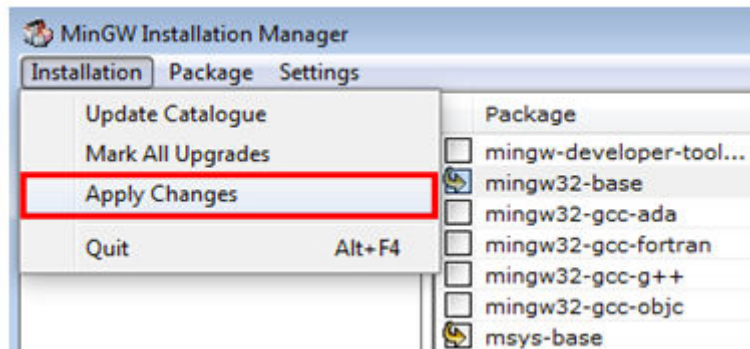
The installation path cannot contain any spaces.

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



**Figure 28. Setup MinGW and MSYS**

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



**Figure 29. Complete MinGW and MSYS installation**

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

`<mingw_install_dir>\bin`

## Run a demo using Arm® GCC

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

### NOTE

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

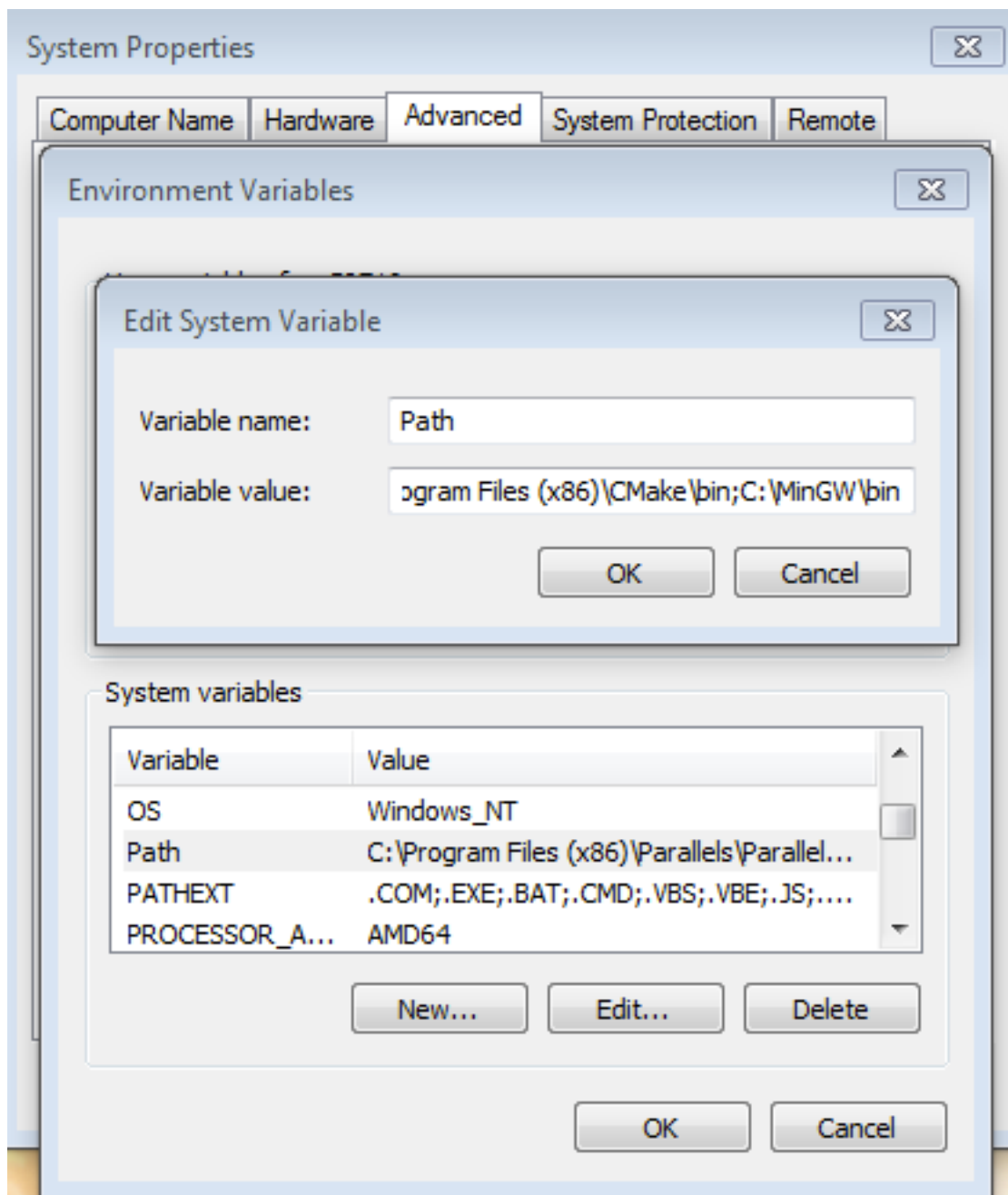


Figure 30. Add Path to systems environment

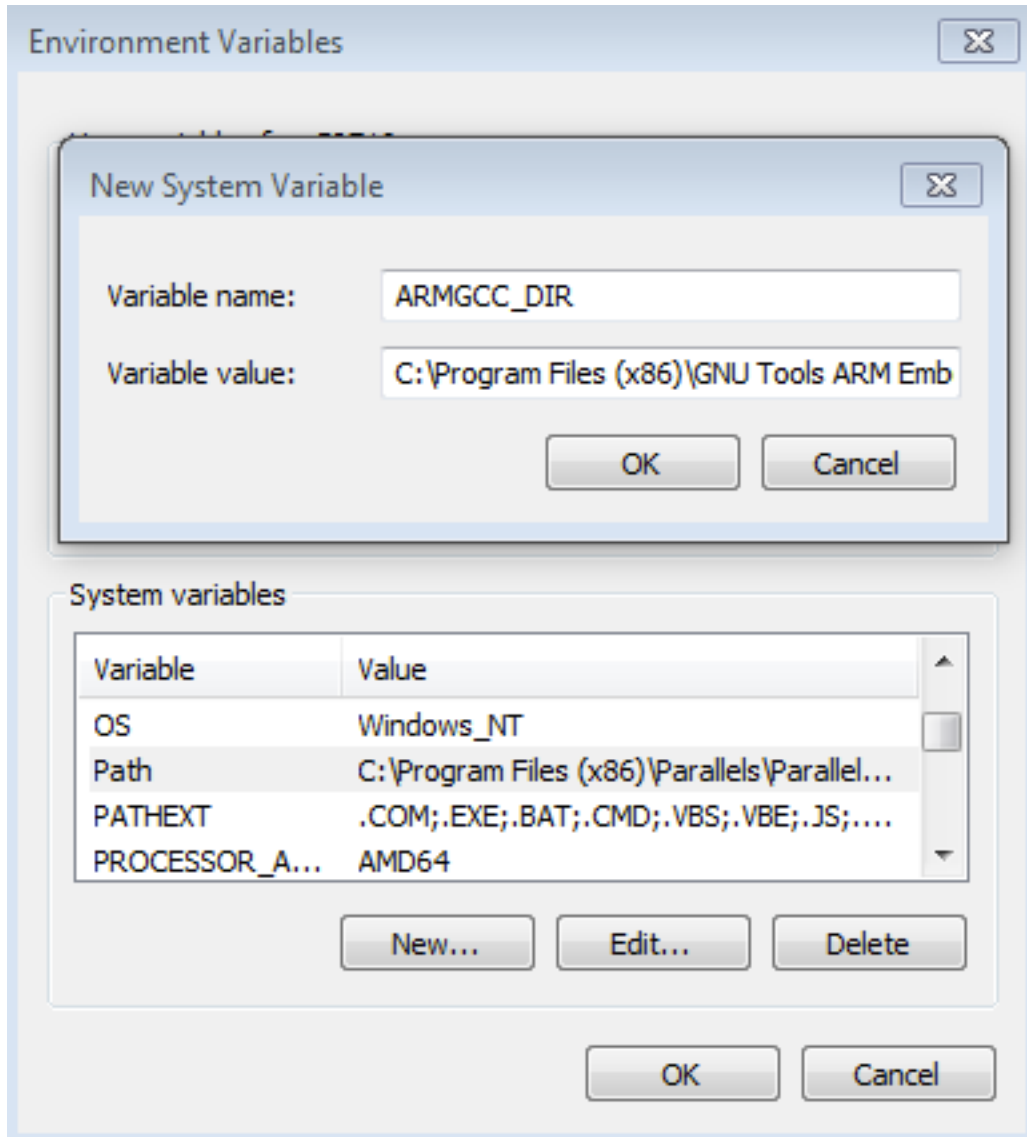
## 6.1.3 Add a new system environment variable for ARMGCC\_DIR



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

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

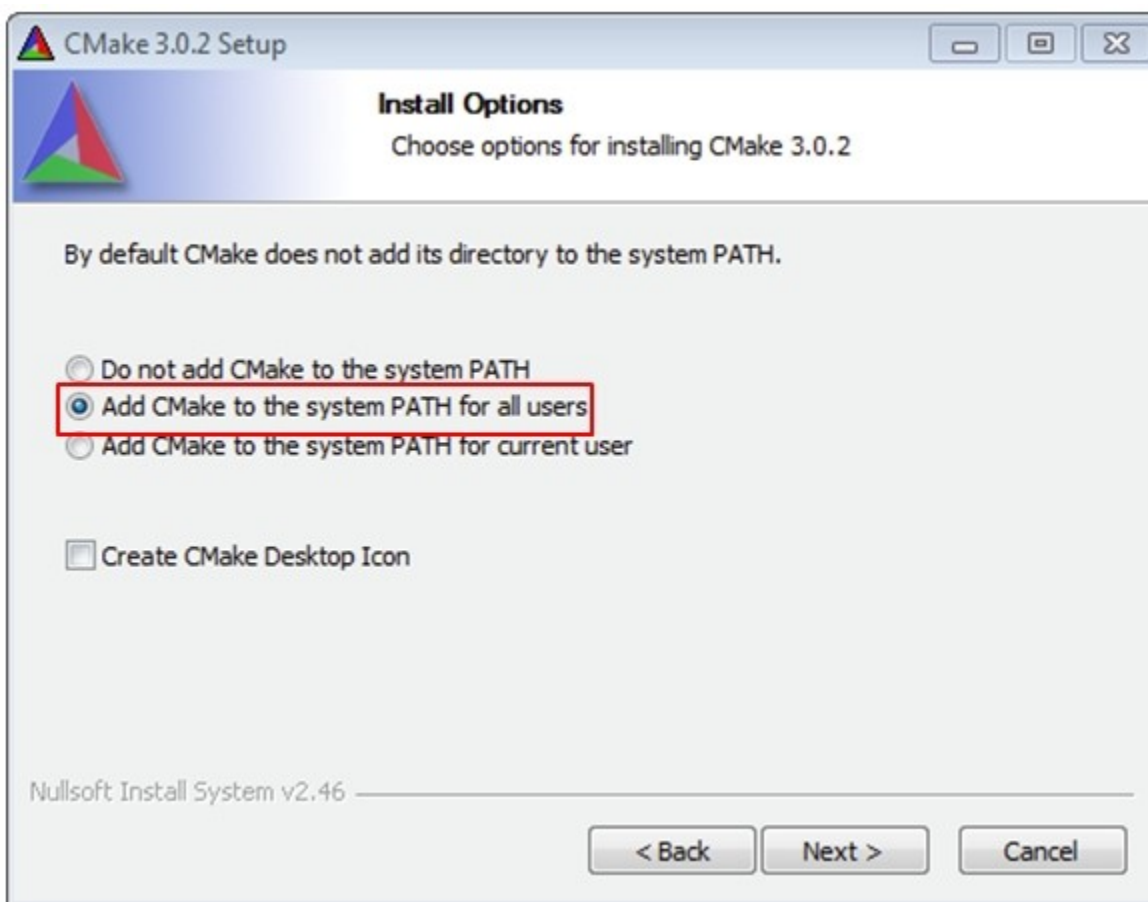
Reference the installation folder of the GNU Arm GCC Embedded tools for the exact path name of your installation.



**Figure 31. Add ARMGCC\_DIR system variable**

### 6.1.4 Install CMake

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



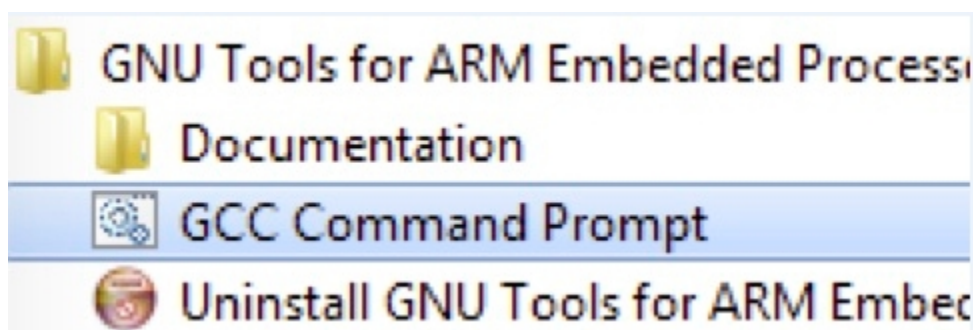
**Figure 32. Install CMake**

3. Follow the remaining instructions of the installer.
4. You may need to reboot your system for the PATH changes to take effect.
5. Make sure "sh.exe" is not in the Environment Variable PATH. This is a limitation of mingw32-make.

## 6.2 Build an example application

To build an example application, follow these steps.

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



**Figure 33. Launch command prompt**



2. Change the directory to the example application project directory, which has a path similar to the following:

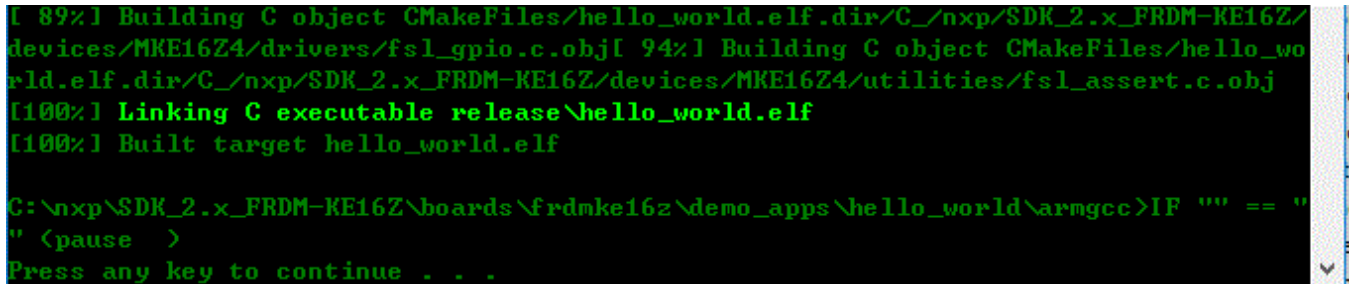
`<install_dir>/boards/<board_name>/<example_type>/<application_name>/armgcc`

For this example, the exact path is: `<install_dir>/boards/frdmke16z/demo_apps/hello_world/armgcc`

#### NOTE

To change directories, use the 'cd' command.

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



```
[ 89%] Building C object CMakeFiles/hello_world.elf.dir/C:/nxp/SDK_2.x_FRDM-KE16Z/
devices/MKE16Z4/drivers/fsl_gpio.c.obj[ 94%] Building C object CMakeFiles/hello_wo
rld.elf.dir/C:/nxp/SDK_2.x_FRDM-KE16Z/devices/MKE16Z4/utilities/fsl_assert.c.obj
[100%] Linking C executable release\hello_world.elf
[100%] Built target hello_world.elf

C:\nxp\SDK_2.x_FRDM-KE16Z\boards\frdmke16z\demo_apps\hello_world\armgcc>IF "" == "
" (pause )
Press any key to continue . . .
```

Figure 34. hello\_world demo build successful

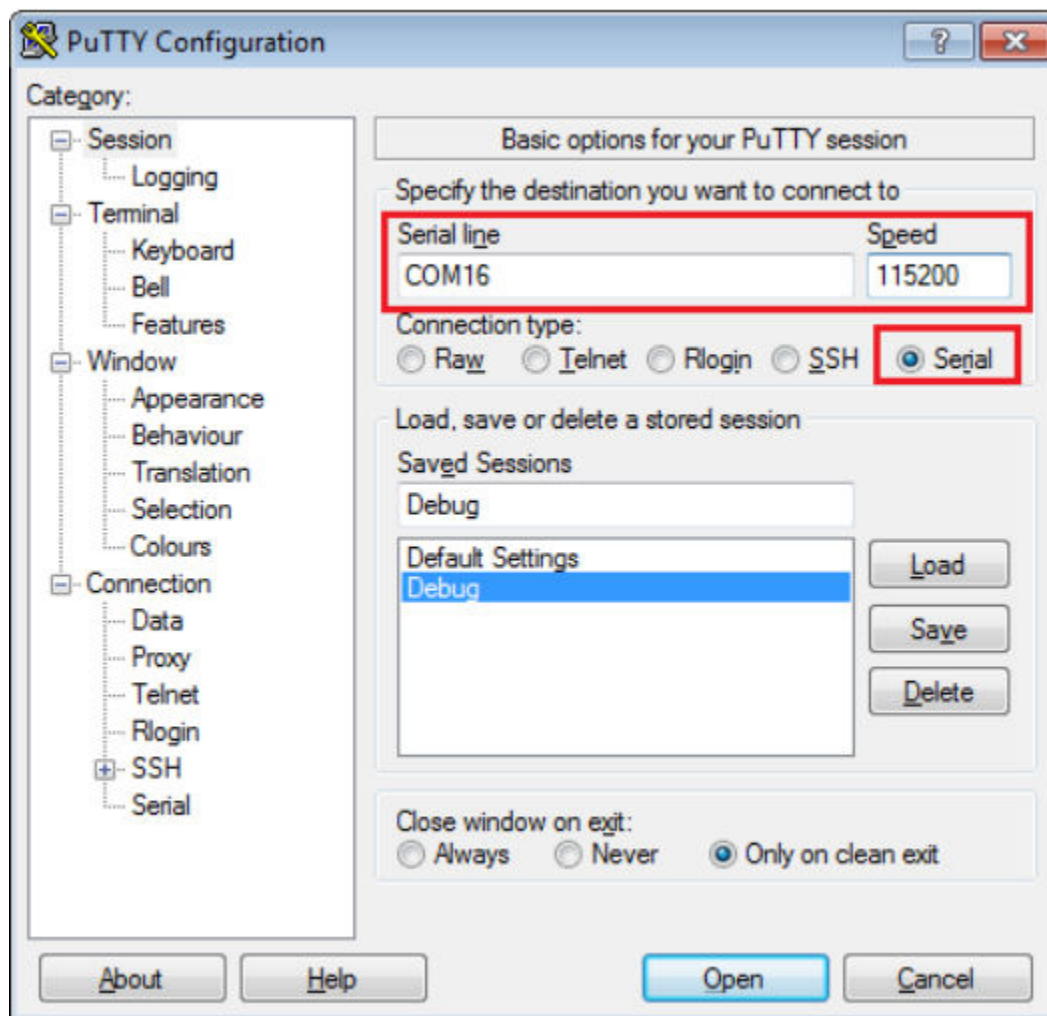
## 6.3 Run an example application

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

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

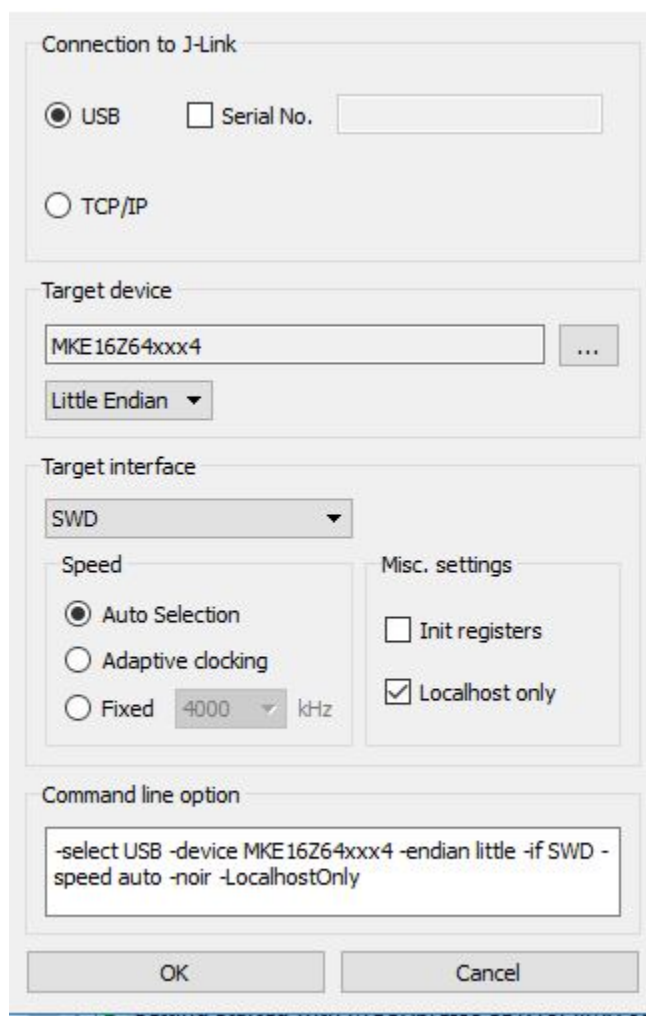
After the J-Link interface is configured and connected, follow these steps to download and run the demo applications:

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



**Figure 35. Terminal (PuTTY) configurations**

3. Open the J-Link GDB Server application. Assuming the J-Link software is installed, the application can be launched by going to the Windows operating system Start menu and selecting “Programs -> SEGGER -> J-Link <version> J-Link GDB Server”.
4. Modify the settings as shown below. The target device selection chosen for this example is K32W042S1M2xxxxx\_M4.



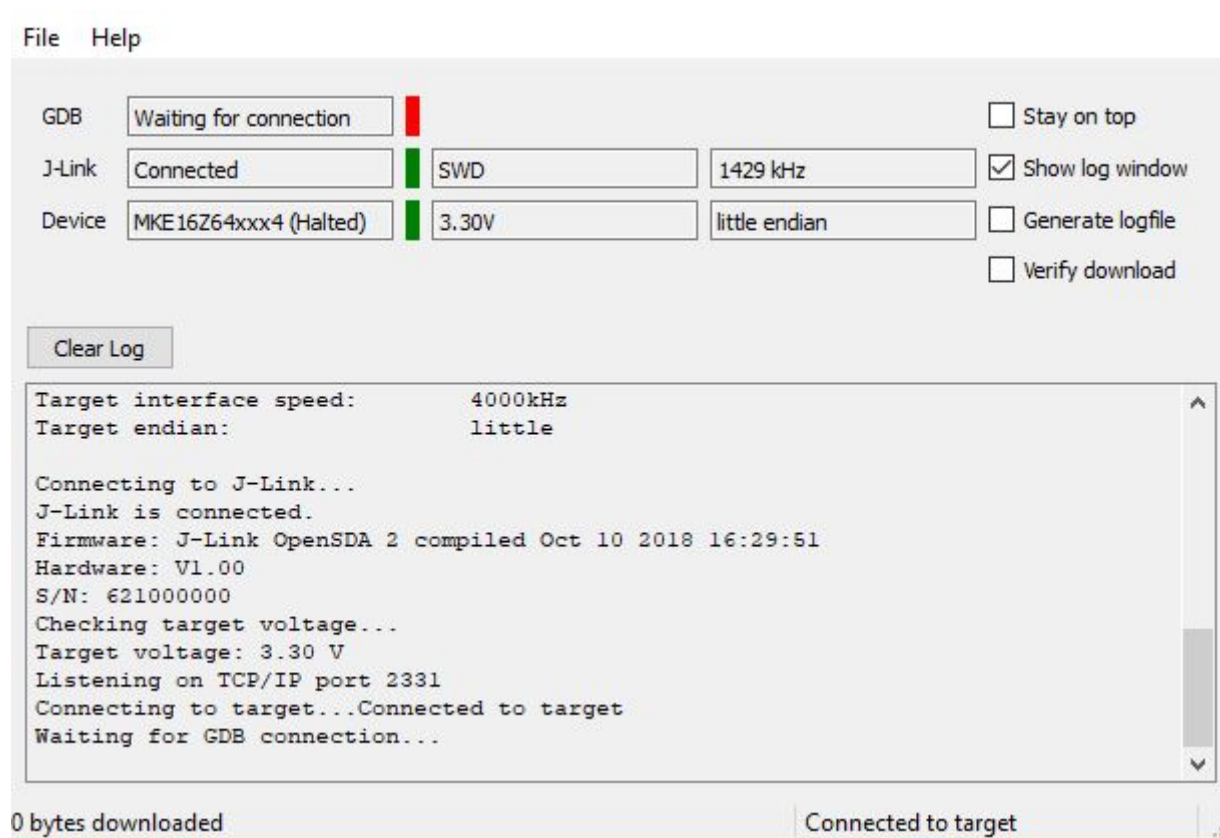
The image shows the SEGGER J-Link GDB Server configuration dialog box. It is divided into several sections:

- Connection to J-Link:**
  - ☒ USB ☐ Serial No.
  - ☐ TCP/IP
- Target device:**
  - Text box: MKE16Z64xxx4
  - Dropdown: Little Endian
- Target interface:**
  - Dropdown: SWD
  - Speed:**
    - ☒ Auto Selection
    - ☐ Adaptive docking
    - ☐ Fixed 4000 kHz
  - Misc. settings:**
    - ☐ Init registers
    - ☒ Localhost only
- Command line option:**
  - Text box: -select USB -device MKE16Z64xxx4 -endian little -if SWD -speed auto -noir -LocalhostOnly

At the bottom are 'OK' and 'Cancel' buttons.

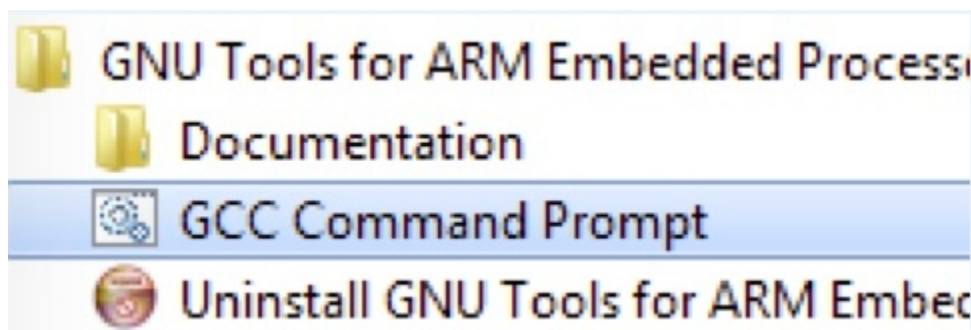
**Figure 36. SEGGER J-Link GDB Server configuration**

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



**Figure 37. SEGGER J-Link GDB Server screen after successful connection**

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



**Figure 38. Launch command prompt**

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

`<install_dir>/boards/<board_name>/<example_type>/<application_name>/armgcc/debug`

`<install_dir>/boards/<board_name>/<example_type>/<application_name>/armgcc/release`

For this example, the path is:

`<install_dir>/boards/frdmke16z/demo_apps/hello_world/armgcc/debug`

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

```

C:\Program Files (x86)\GNU Tools ARM Embedded\6 2017-q2-update>cd C:\nxp\SDK_2.x_FRDM-KE16Z\boards\frdmke16z\demo_apps\h
ello_world\armgcc\debug

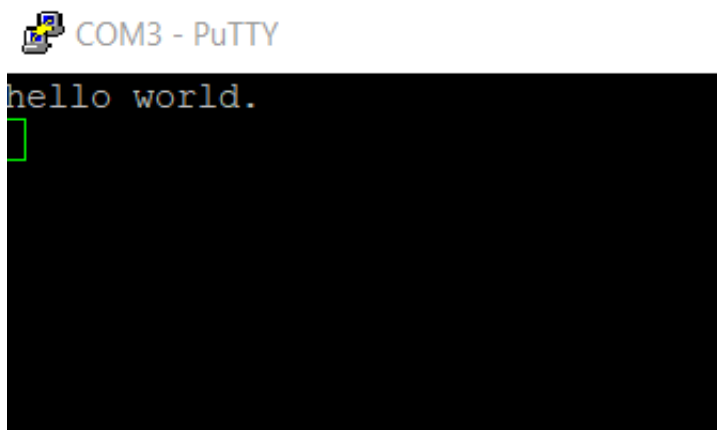
C:\nxp\SDK_2.x_FRDM-KE16Z\boards\frdmke16z\demo_apps\hello_world\armgcc\debug>arm-none-eabi-gdb hello_world.elf
GNU gdb (GNU Tools for ARM Embedded Processors 6-2017-q2-update) 7.12.1.20170417-git
Copyright (C) 2017 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --target=arm-none-eabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from hello_world.elf...done.
(gdb)

```

**Figure 39. Run arm-none-eabi-gdb**

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

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



**Figure 40. Text display of the hello\_world demo**





## 7 MCUXpresso Config Tools

## MCUXpresso IDE New Project Wizard

MCUXpresso Config Tools can help configure the processor and generate initialization code for the on chip peripherals. The tools are able to modify any existing example project, or create a new configuration for the selected board or processor. The generated code is designed to be used with MCUXpresso SDK version 2.x.

The MCUXpresso Config Tools consist of the following:

**Table 1. MCUXpresso Config Tools**

Config Tool	Description	Image
<b>Pins tool</b>	For configuration of pin routing and pin electrical properties.	
<b>Clock tool</b>	For system clock configuration	
<b>Peripherals tools</b>	For configuration of other peripherals	
<b>Project Cloner</b>	Allows creation of the standalone projects from MCUXpresso SDK examples.	

MCUXpresso Config Tools can be accessed in the following products:

- **Integrated** in the MCUXpresso IDE. Config tools are integrated with the compiler and debugger, so this represents the easiest way to begin that development.
- **Standalone version** available for download from [www.nxp.com](http://www.nxp.com). Recommended for customers using IAR Embedded Workbench, Keil MDK  $\mu$ Vision, or Arm GCC.
- **Online version** available on [mcuxpresso.nxp.com](http://mcuxpresso.nxp.com). Recommended to do a quick evaluation of the processor or use the tool without installation.

Each version of the product contains a specific “Quick Start Guide” document that can help start your work.

## 8 MCUXpresso IDE New Project Wizard

MCUXpresso IDE features a new project wizard. The wizard provides functionality for the user to create new projects from the installed SDKs (and from pre-installed part support), offers the flexibility to select/change many builds, includes a library, and provides source code options. The source code is organized as software components, categorized as driver, utilities, and middleware.

To use the wizard, start the MCUXpresso IDE. This is located in the *QuickStart Panel* at the bottom left of the MCUXpresso IDE window. Select the “New project” option, shown in the below figure.



**Figure 41. MCUXpresso IDE Quickstart Panel**

For more details of the usage of new project wizard, see the “MCUXpresso\_IDE\_User\_Guide.pdf” in the MCUXpresso IDE installation folder.

## 9 Appendix A - How to determine COM port

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

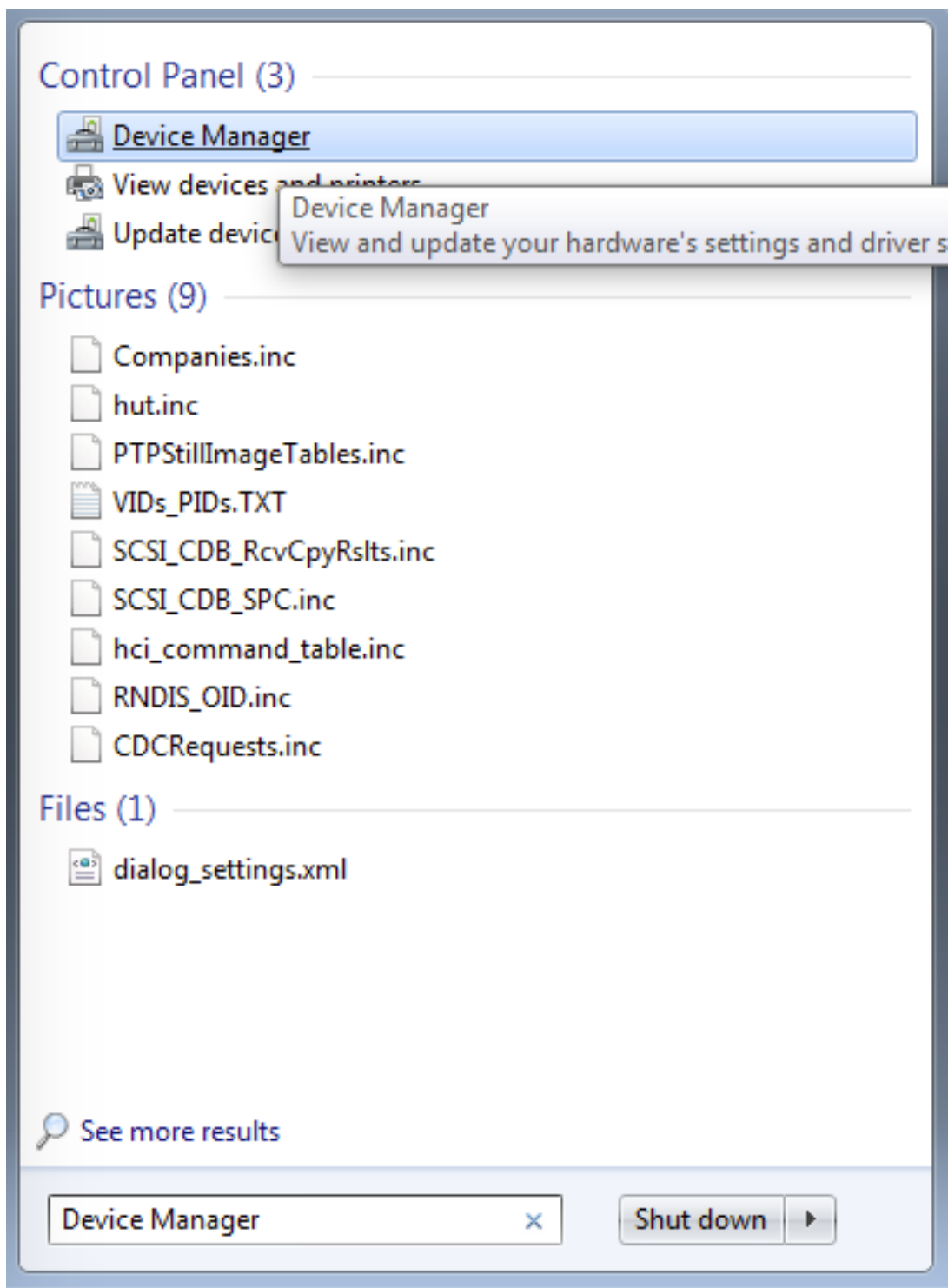
1. **Linux:** The serial port can be determined by running the following command after the USB Serial is connected to the host:

```
$ dmesg | grep "ttyUSB"
[503175.307873] usb 3-12: cp210x converter now attached to ttyUSB0
[503175.309372] usb 3-12: cp210x converter now attached to ttyUSB1
```

There are two ports, one is Cortex-A core debug console, another is for Cortex M4.

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

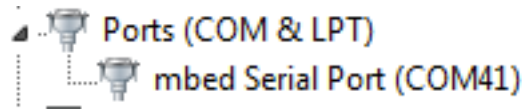




**Figure 42. Device manager**

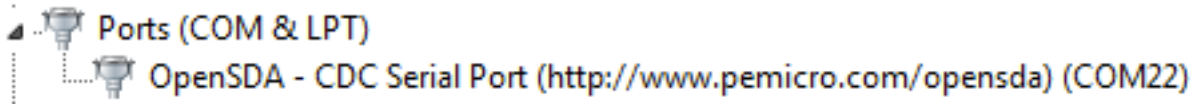
3. In the Device Manager, expand the “Ports (COM & LPT)” section to view the available ports. Depending on the NXP board you’re using, the COM port can be named differently:
  - a. OpenSDA – CMSIS-DAP/mbed/DAPLink interface:





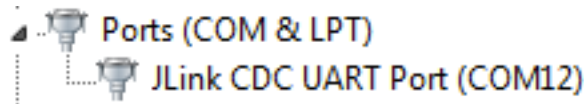
**Figure 43. OpenSDA – CMSIS-DAP/mbed/DAPLink interface**

b. OpenSDA – P&E Micro:



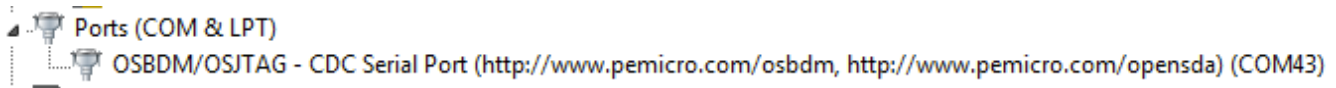
**Figure 44. OpenSDA – P&E Micro**

c. OpenSDA – J-Link:



**Figure 45. OpenSDA – J-Link**

d. P&E Micro OSJTAG:



**Figure 46. P&E Micro OSJTAG**

## 10 Appendix B - Default debug interfaces

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

### NOTE

The 'OpenSDA details' column of the following table is not applicable to LPC.

**Table 2. Hardware platforms supported by SDK**

Hardware platform	Default interface	OpenSDA details
FRDM-KE16Z	CMSIS-DAP/mbed/DAPLink	OpenSDA v2.2

## 11 Appendix C - Updating debugger firmware

### 11.1 Updating OpenSDA firmware

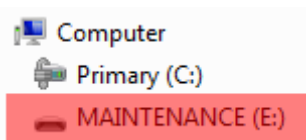
## Appendix C - Updating debugger firmware

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

- **J-Link:** Download appropriate image from [www.segger.com/opensda.html](http://www.segger.com/opensda.html). Chose the appropriate J-Link binary based on the table in Appendix B. Any OpenSDA v1.0 interface should use the standard OpenSDA download (in other words, the one with no version). For OpenSDA 2.0 or 2.1, select the corresponding binary.
- **CMSIS-DAP/mbed/DAPLink:** DAPLink OpenSDA firmware is available at [www.nxp.com/opensda](http://www.nxp.com/opensda).
- **P&E Micro:** Downloading P&E Micro OpenSDA firmware images requires registration with P&E Micro ([www.pemicro.com](http://www.pemicro.com)).

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

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



**Figure 47. MAINTENANCE drive**

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

### NOTE

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

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

1. Unplug the board's USB cable.
2. Press the board's "Reset" button. While still holding the button, plug the board back in to the USB cable.
3. For boards with OpenSDA v2.0 or v2.1, it shows up as a disk drive called "BOOTLOADER" in Finder. Boards with OpenSDA v1.0 may or may not show up depending on the bootloader version. If you see the drive in Finder, proceed to the next step. If you do not see the drive in Finder, use a PC with Windows OS 7 or an earlier version to either update the OpenSDA firmware, or update the OpenSDA bootloader to version 1.11 or later. The bootloader update instructions and image can be obtained from P&E Microcomputer website.
4. For OpenSDA v2.1 and OpenSDA v1.0 (with bootloader 1.11 or later) users, drag the new firmware image onto the BOOTLOADER drive in Finder, similar to how you would drag and drop the file onto a normal USB Flash drive.
5. For OpenSDA v2.0 users, type these commands in a Terminal window:

```
> sudo mount -u -w -o sync /Volumes/BOOTLOADER  
> cp -X <path to update file> /Volumes/BOOTLOADER
```

### NOTE

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

**How to Reach Us:****Home Page:**[nxp.com](http://nxp.com)**Web Support:**[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro,  $\mu$ Vision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2018 NXP B.V.

Document Number MCUXSDKKE16GSUG  
Revision 0, 12/2018

