

Getting Started with MCUXpresso SDK for TWR-KM35Z75M

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 *MCUXpresso SDK Release Notes for TWR-KM35Z75M* (document MCUXSDKKM35Z75MRN) .

For more details about MCUXpresso SDK, see the MCUXpresso SDK homepage [MCUXpresso-SDK: Software Development Kit for MCUXpresso](#).

Contents

1	Overview.....	1
2	MCUXpresso SDK board support package 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.....	16
6	Run a demo using Arm® GCC.....	19
7	MCUXpresso Config Tools.....	28
8	MCUXpresso IDE New Project Wizard	28
A	How to determine COM port.....	29
B	Default debug interfaces.....	31
C	Updating debugger firmware.....	33



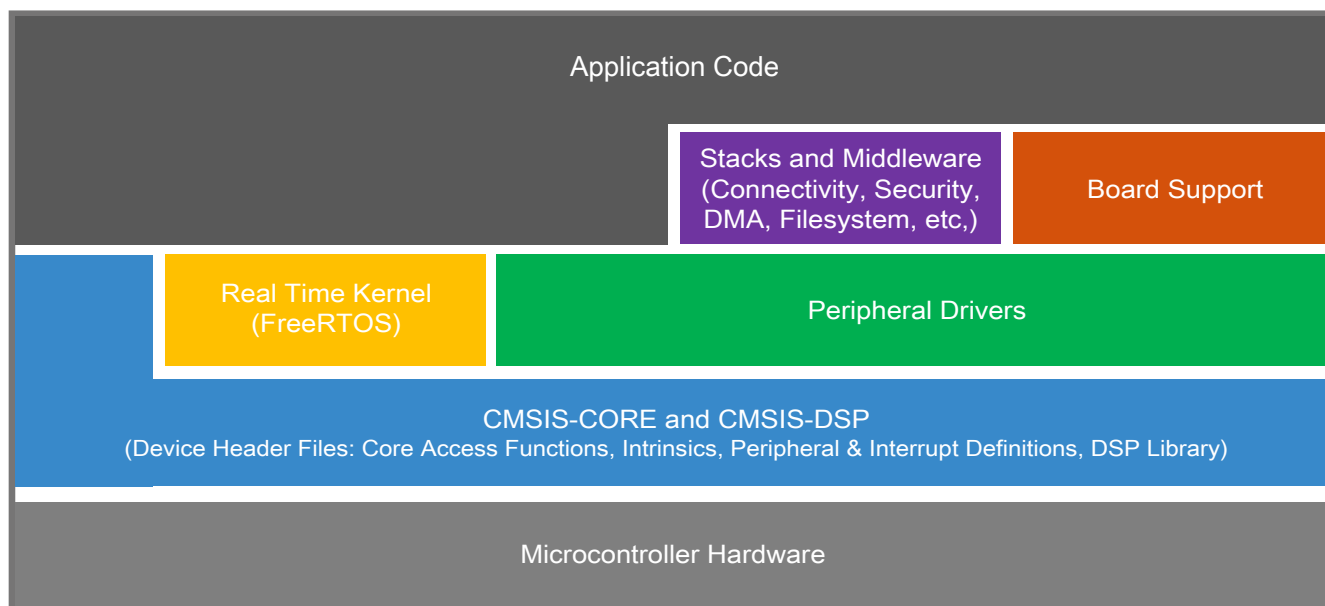


Figure 1. MCUXpresso SDK layers

2 MCUXpresso SDK board support package folders

MCUXpresso SDK board support package 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 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 it contain. These include (but are not limited to):

- **demo_apps**: Full-featured applications that 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 that show 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 peripherals are used (for example, SPI conversion using DMA).
- **rtos_examples**: Basic FreeRTOS™ OS examples that show the use of various RTOS objects (semaphores, queues, and so on) and interfaces with the MCUXpresso SDK's RTOS drivers
- **multicore_examples**: Applications for both cores showing the usage of multicore software components and the interaction between cores.
- **mmcau_examples**: Simple applications intended to concisely illustrate how to use middleware/mmcau stack.
- **wireless_examples**: Applications that use the Zigbee and OpenThread stacks.
- **usb_dongle_examples**: Simple applications to be used on the PCB2459-2 JN5189 USB DONGLE.

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 *MCUXpresso SDK API Reference Manual*.

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:

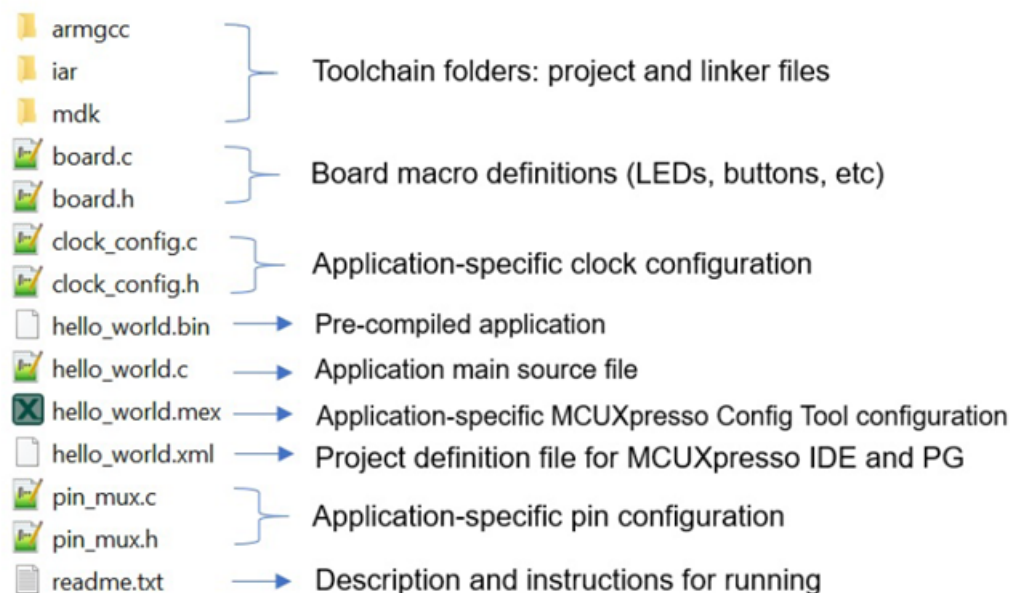


Figure 2. Application folder structure

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.
- `devices/<device_name>/project template`: Project template used in CMSIS PACK new project creation.

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 one could have potential impacts on other projects that depend on that file.

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 v11.1.1 to build, run, and debug example applications. The `hello_world` demo application targeted for the TWR-KM35Z75M 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 are in the workspace. The location of the workspace can be anywhere, but it is recommended that the workspace be located 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 **OK** and wait until the import has finished.

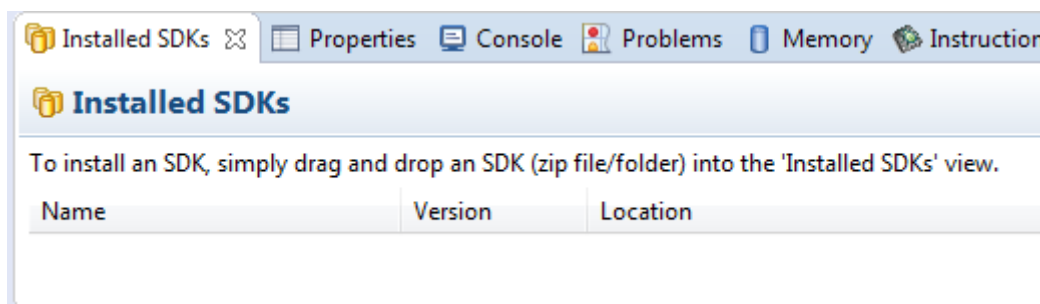


Figure 3. Installing an SDK

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

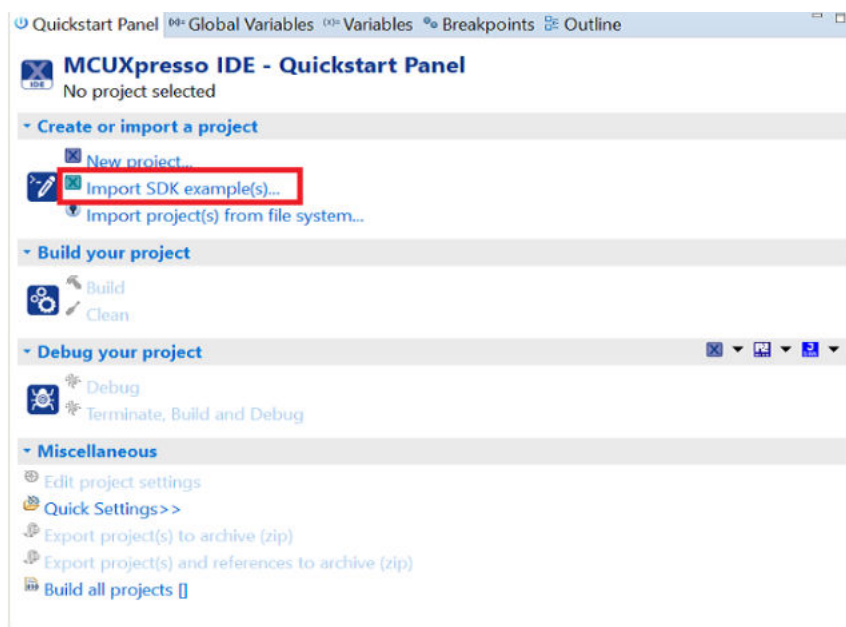


Figure 4. Importing an SDK example

3. In the window that appears, expand the **KM3x** folder and select **MKM35Z512xxx7**. Then, select **twrkm35z75m** and click **Next**.

Run a demo using MCUXpresso IDE

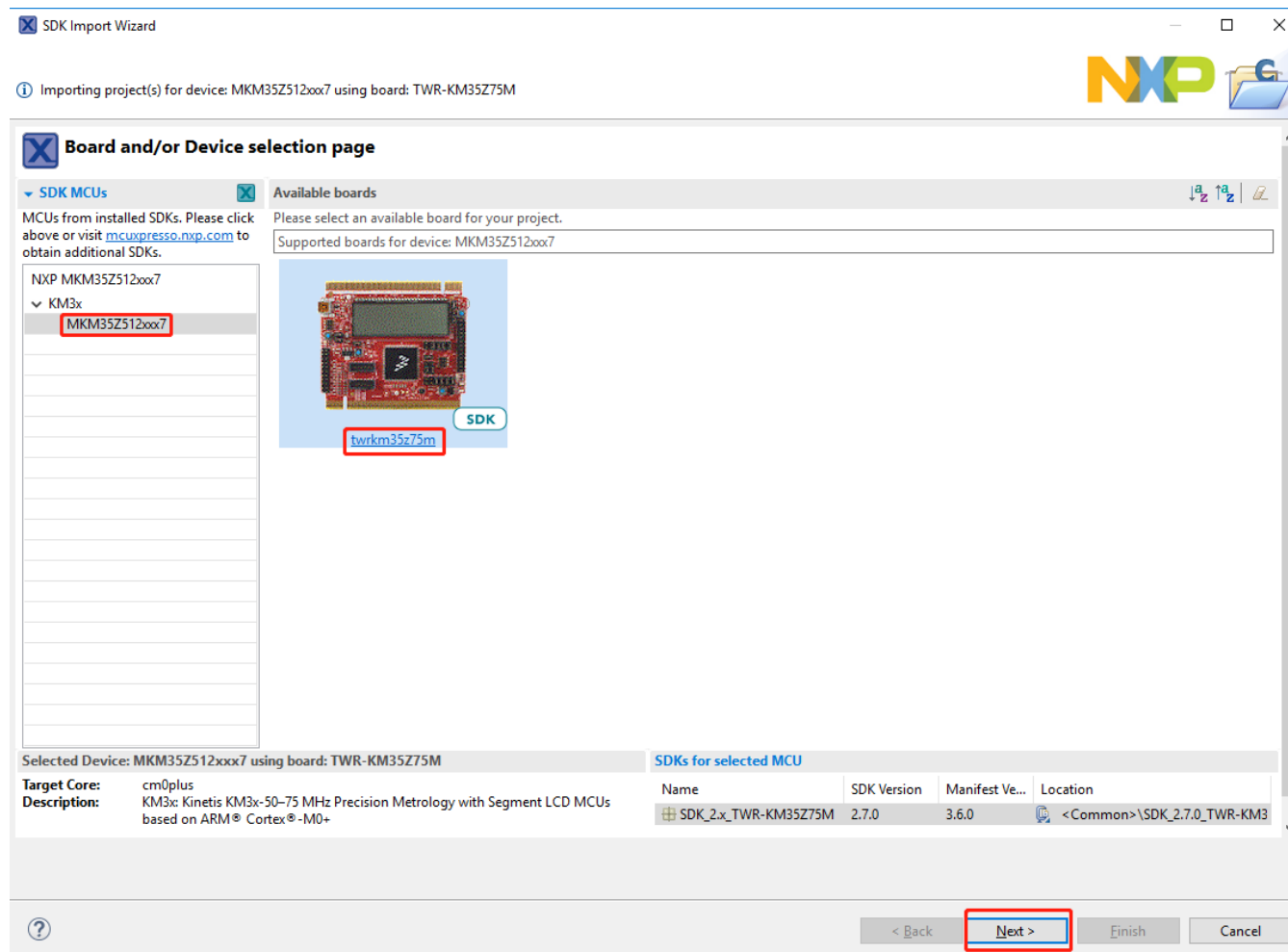


Figure 5. Selecting TWR-KM35Z75M board

- Expand the **demo_apps** folder and select **hello_world**. Then, click Next.

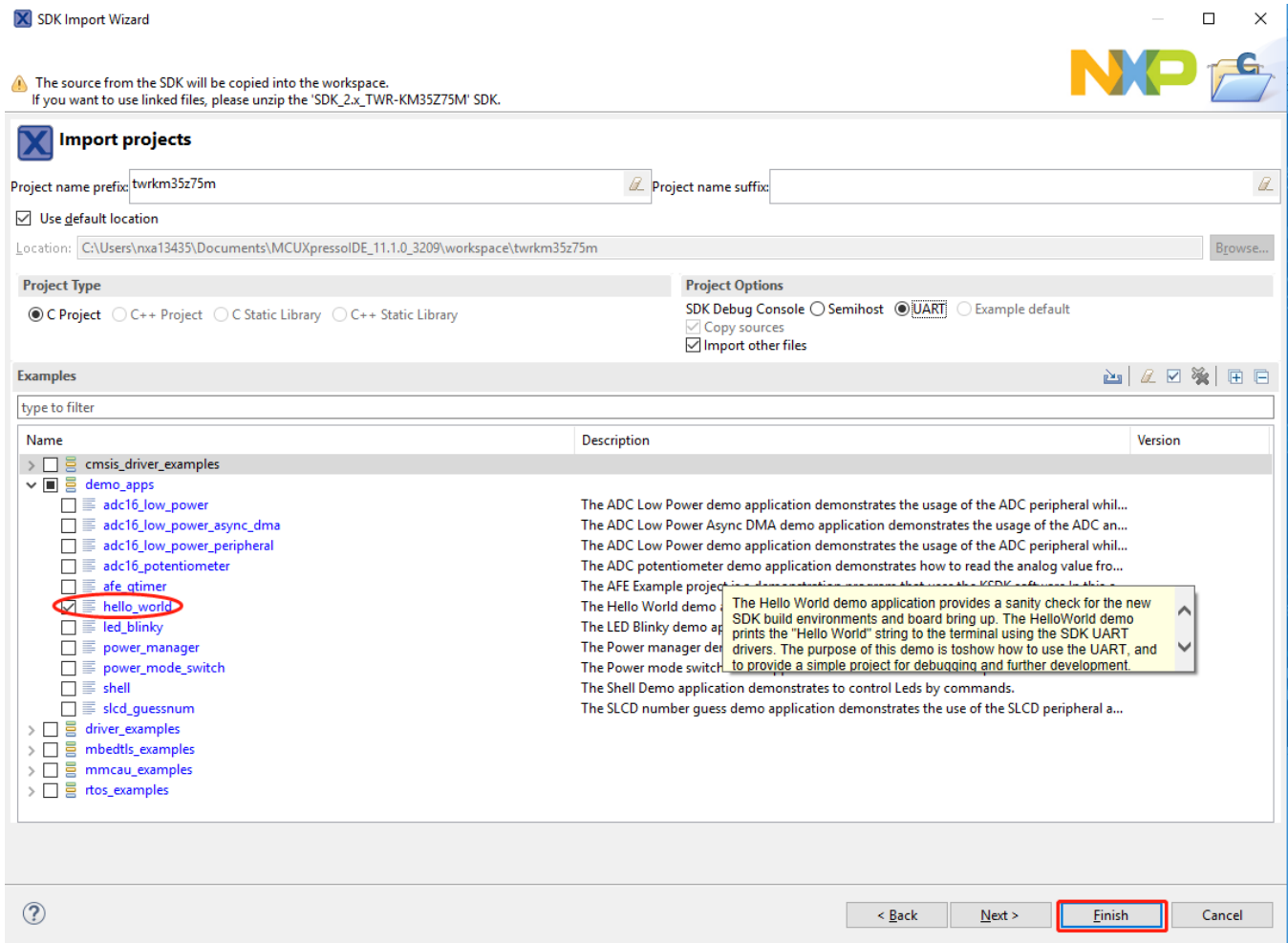


Figure 6. Selecting hello_world

5. Ensure the option of 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_dma and adc_interrupt. Otherwise, it is not necessary to select this option. Then, click **Finish**.

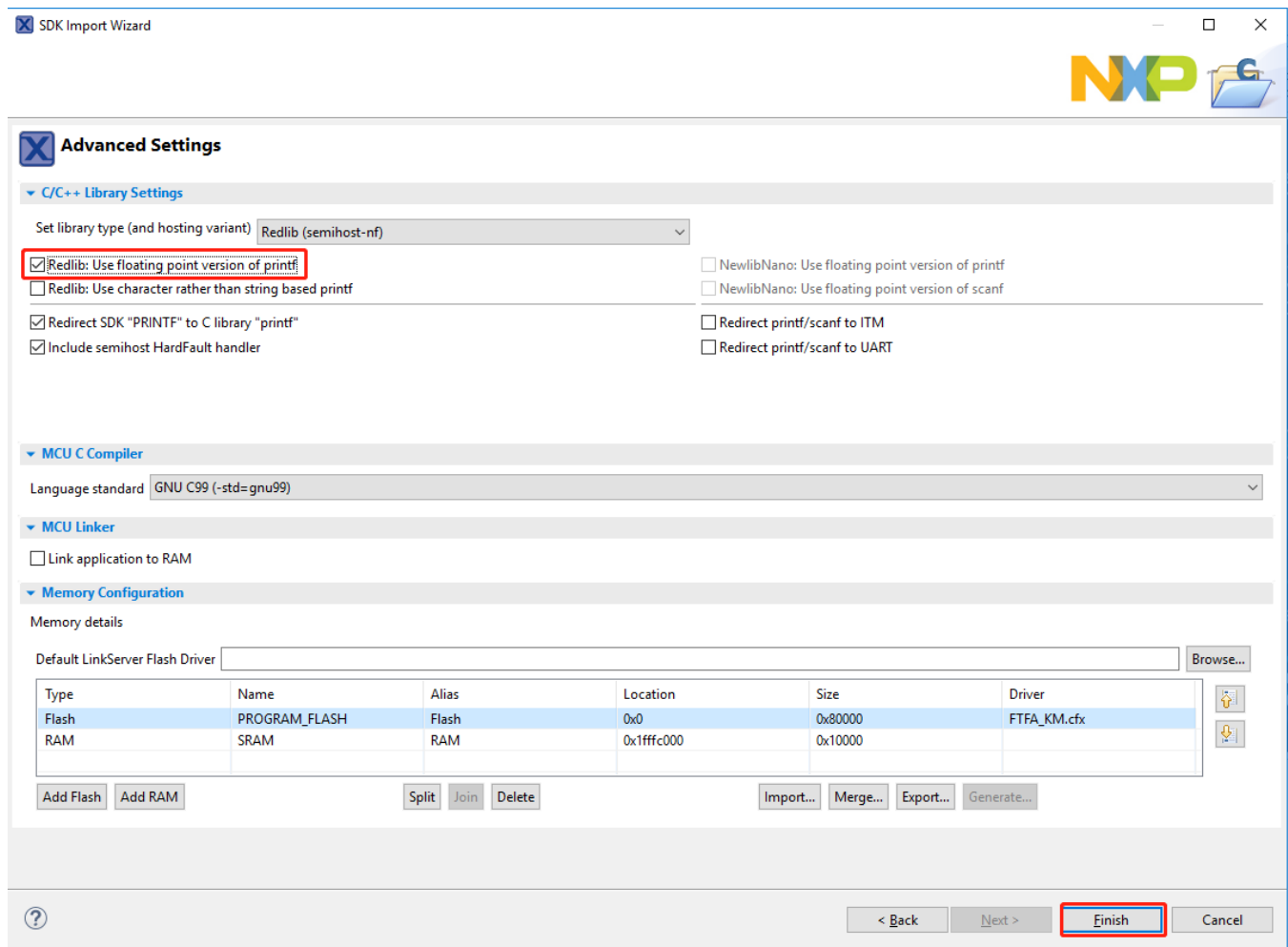


Figure 7. Selecting User floating print version of printf

3.3 Run an example application

For more information on debug probe support in the MCUXpresso IDE 11.1.1, visit 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 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.
- 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 the board.h file)
 - No parity
 - 8 data bits
 - 1 stop bit

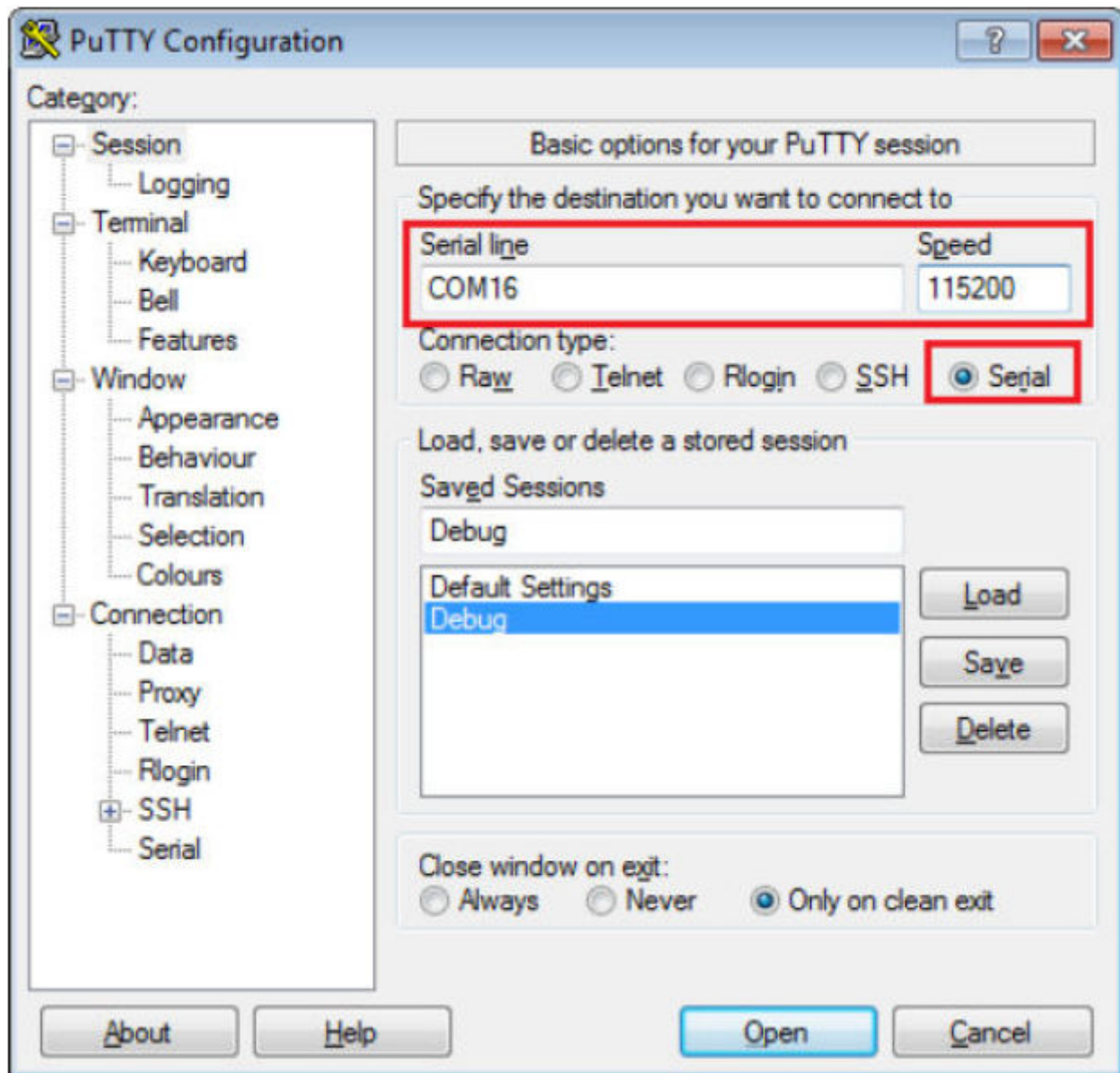


Figure 8. Terminal (PuTTY) configurations

4. On the Quickstart Panel, click on Debug 'twrkm35z75m_demo_apps_hello_world' [Debug].

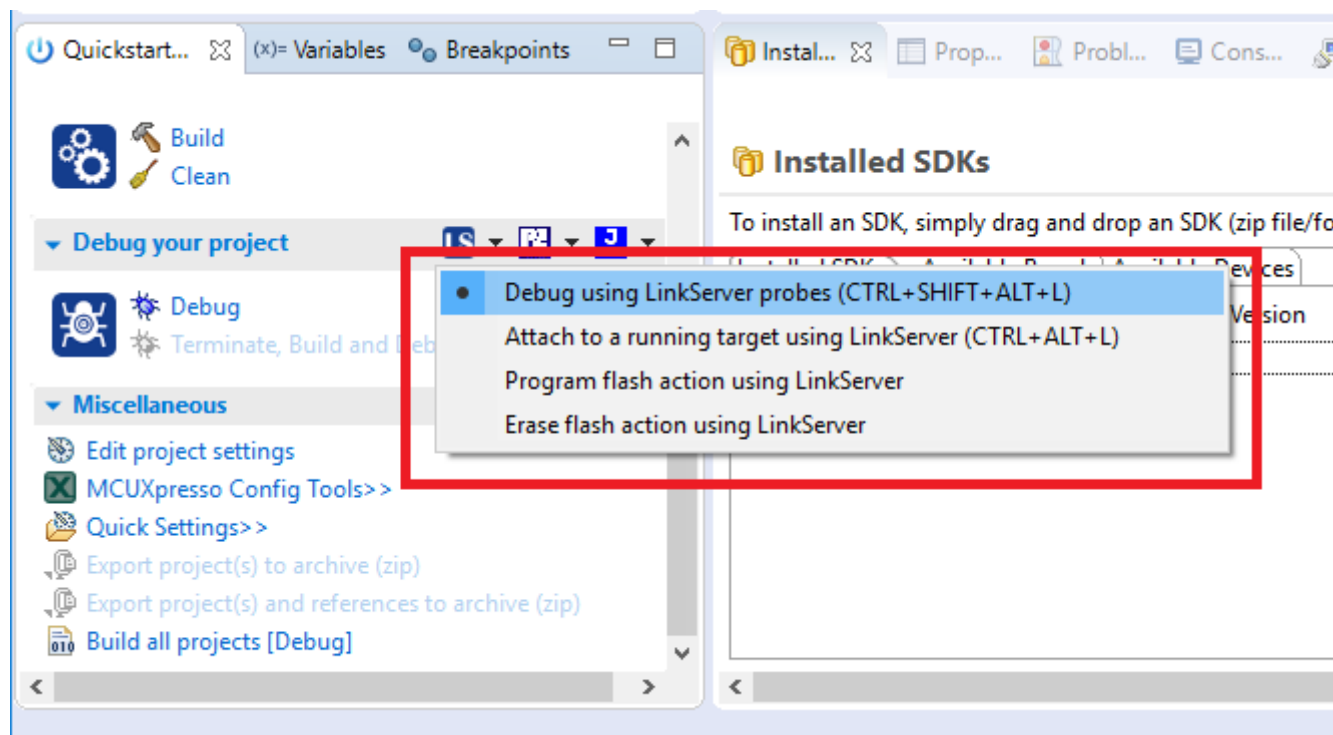


Figure 9. Debugging `hello_world` case

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 **OK**. (For any future debug sessions, the stored probe selection is automatically used, unless the probe cannot be found.)

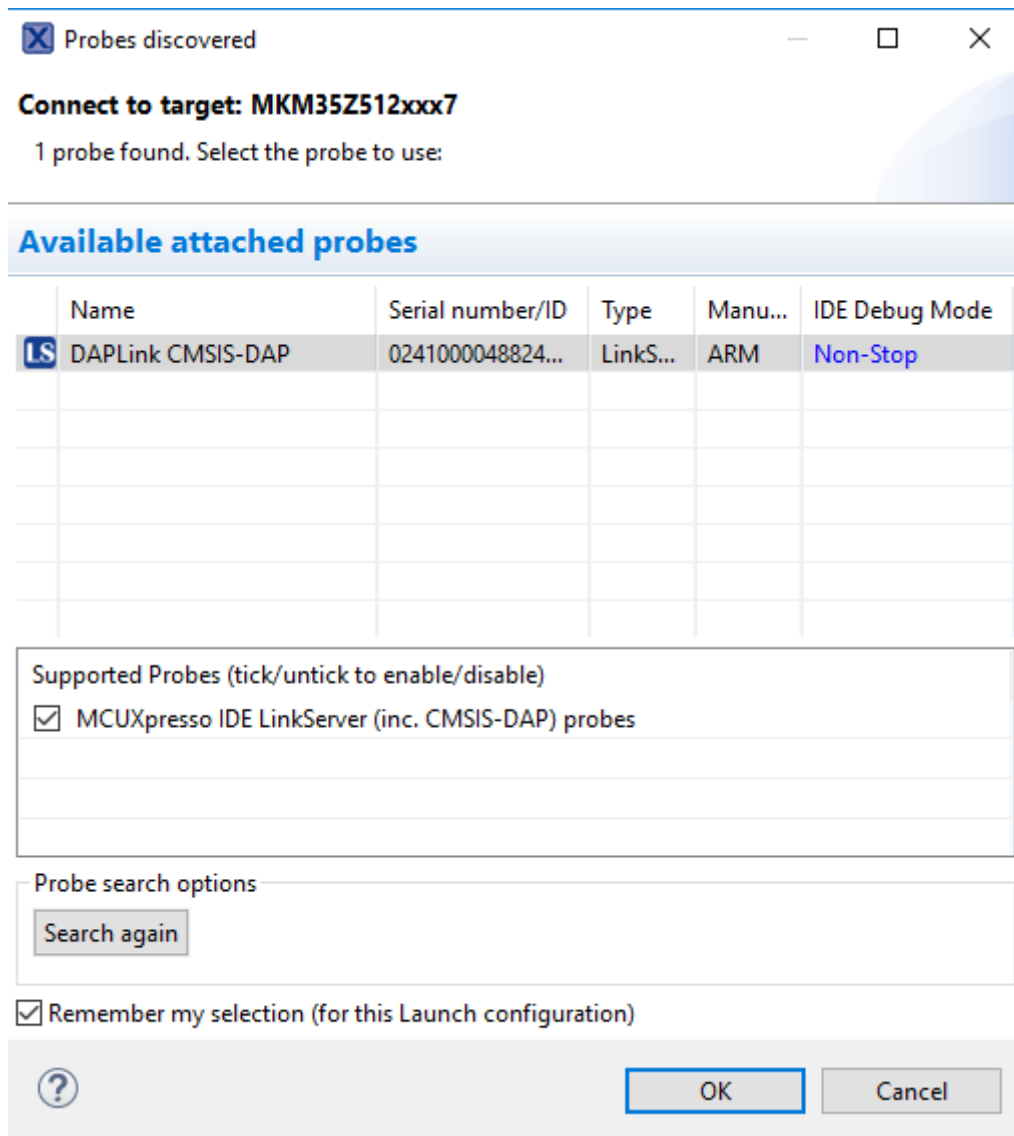


Figure 10. Attached Probes: debug emulator selection

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

Run a demo using MCUXpresso IDE

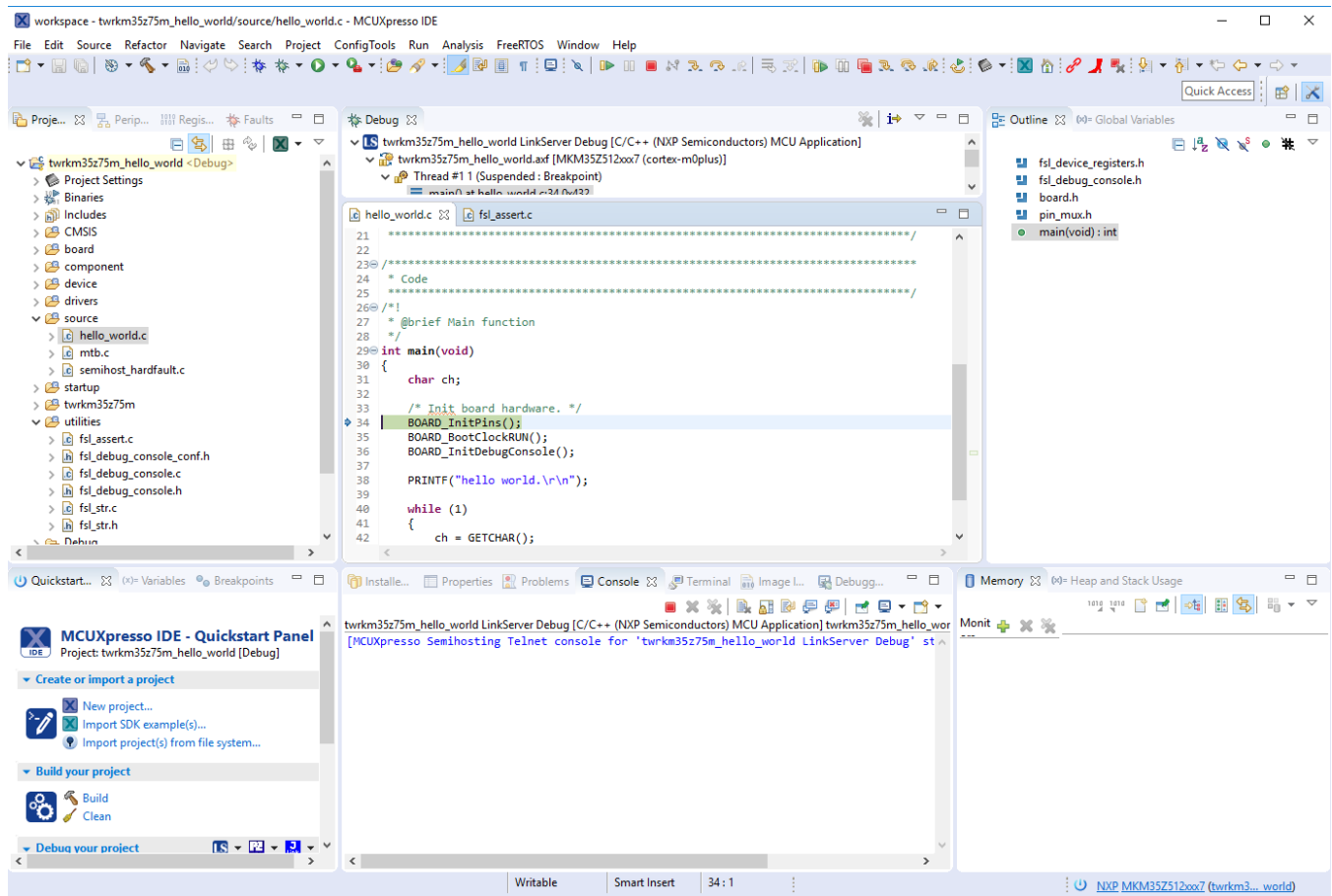


Figure 11. Stop at `main()` when running debugging

7. Start the application by clicking **Resume**.



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.

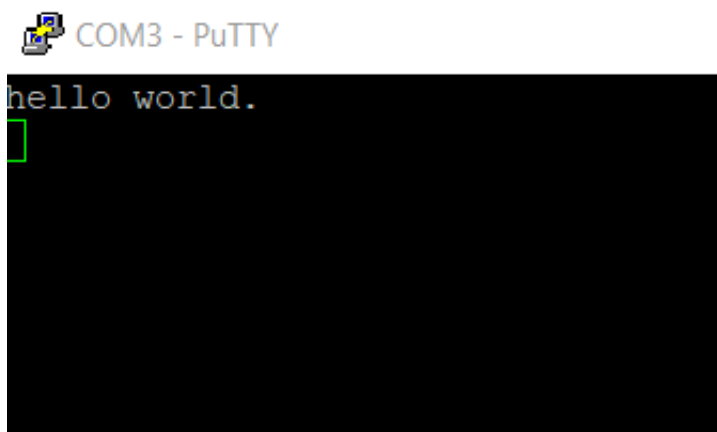


Figure 13. Text display of the `hello_world` demo

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 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 for TWR-KM35Z75M* (document MCUXSDKKM35Z75MRN).

4.1 Build an example application

Perform the following steps to build the `hello_world` example application.

1. 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 TWR-KM35Z75M hardware platform as an example, the `hello_world` workspace is located in:

```
<install_dir>/boards/twrkm35z75m/demo_apps/hello_world/iar/hello_world.eww
```

2. Select the desired build target from the drop-down menu. For this example, select the **hello_world – debug** target.

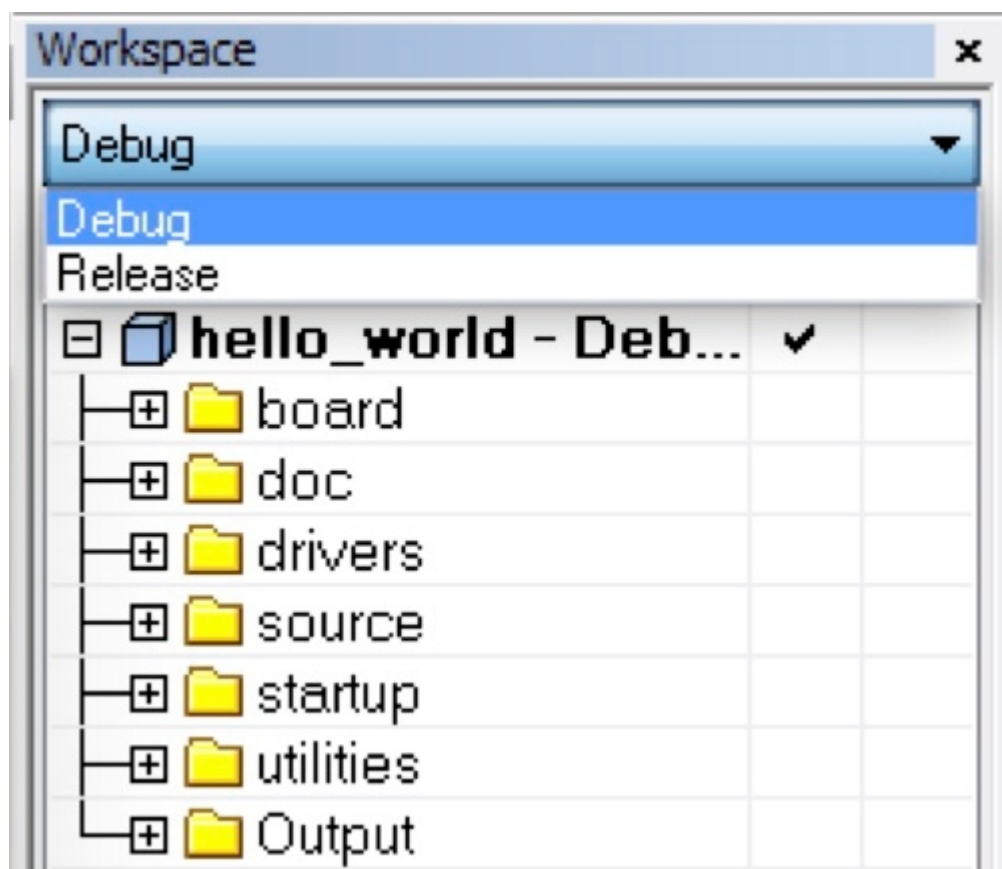


Figure 14. Demo build target selection

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

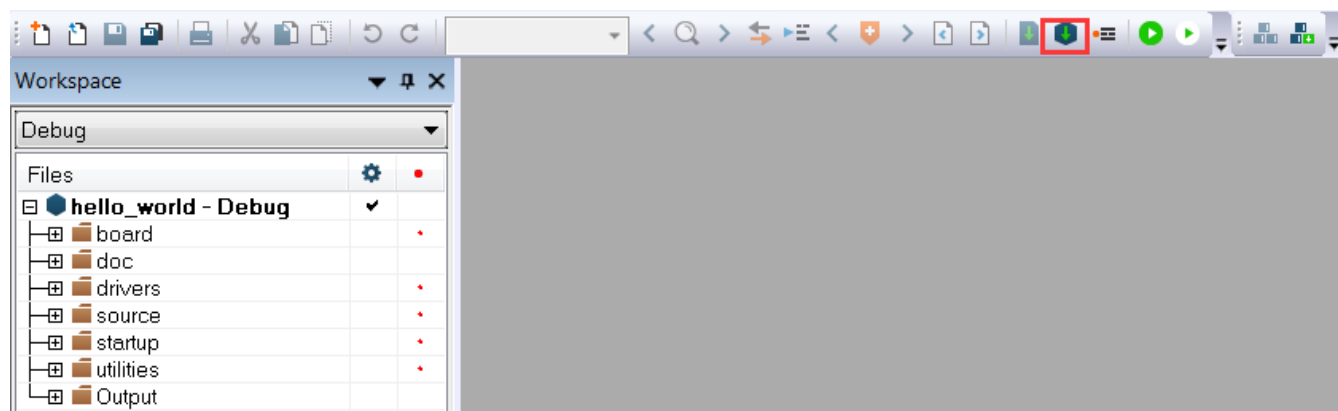


Figure 15. Building the demo application

4. The build completes without errors.

4.2 Run an example application

To download and run the application, perform these steps:

1. See the table in [Default debug interfaces](#) 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 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 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 [How to determine COM port](#)). Configure the terminal with these settings:
 - a. 115200 or 9600 baud rate, depending on your board (reference BOARD_DEBUG_UART_BAUDRATE variable in the board.h file)
 - b. No parity
 - c. 8 data bits
 - d. 1 stop bit

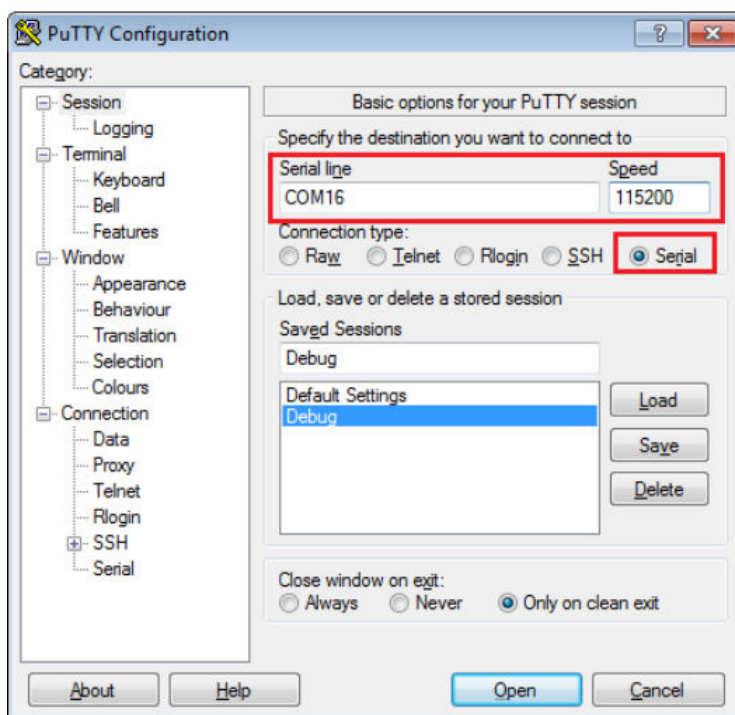


Figure 16. Terminal (PuTTY) configuration

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.

Run a demo using Keil® MDK/μVision

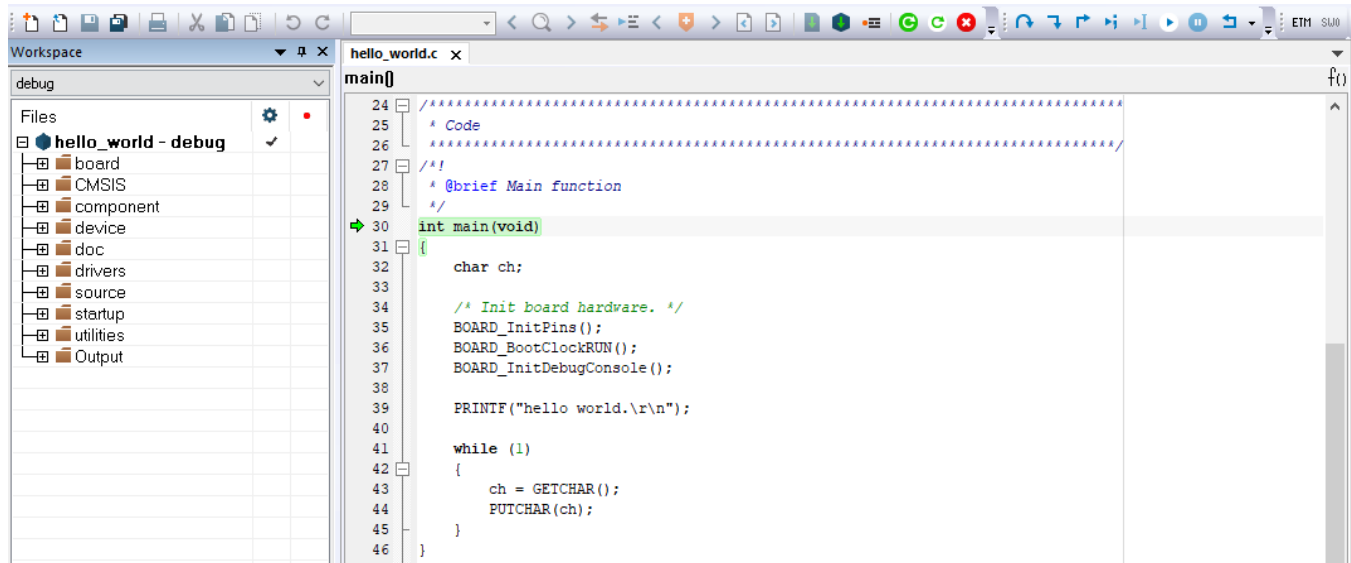


Figure 18. Stop at `main()` when running debugging

6. Run the code by clicking the **Go** button.



Figure 19. Go button

7. The `hello_world` application is now running and a banner is displayed on the terminal. If it does not appear, check your terminal settings and connections.

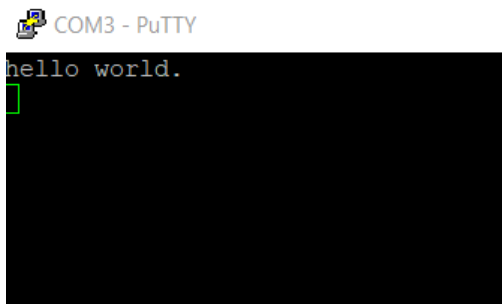


Figure 20. Text display of the `hello_world` demo

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 TWR-KM35Z75M 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.

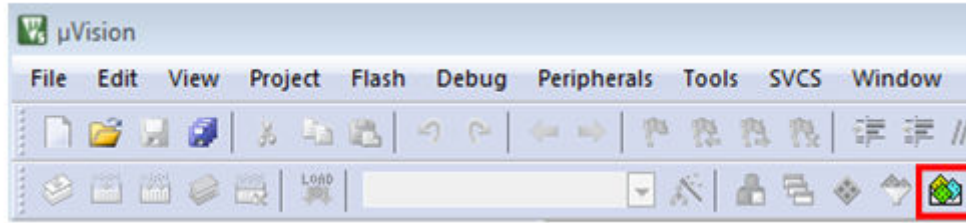


Figure 21. Launch the Pack Installer

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/twrkm35z75m/demo_apps/hello_world/mdk/hello_world.uvmpw
```

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



Figure 22. Building the demo

- The build completes without errors.

5.3 Run an example application

To download and run the application, perform these steps:

1. See the table in [Default debug interfaces](#) 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 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.
2. Connect the development platform to your PC via USB cable using OpenSDA 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 [How to determine COM port](#)). Configure the terminal with these settings:

Run a demo using Keil® MDK/μVision

- 115200 or 9600 baud rate, depending on your board (reference BOARD_DEBUG_UART_BAUDRATE variable in the board.h file)
- No parity
- 8 data bits
- 1 stop bit

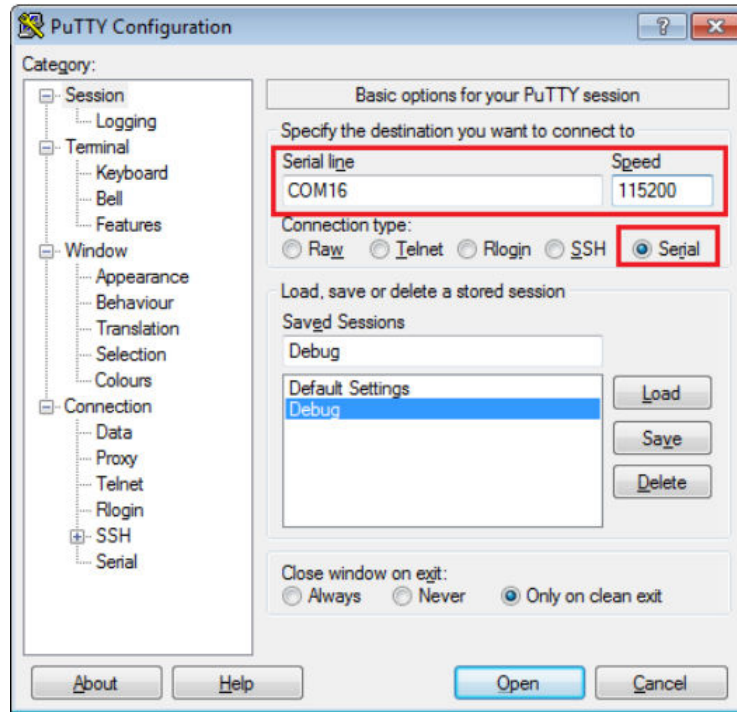


Figure 23. Terminal (PuTTY) configurations

- In μVision, after the application is built, click **Download** to download the application to the target.

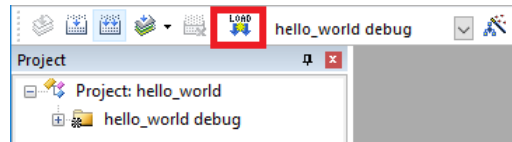


Figure 24. Download button

- After clicking **Download**, the application downloads to the target and is running. To debug the application, click **Start/Stop Debug Session**, highlighted in red.

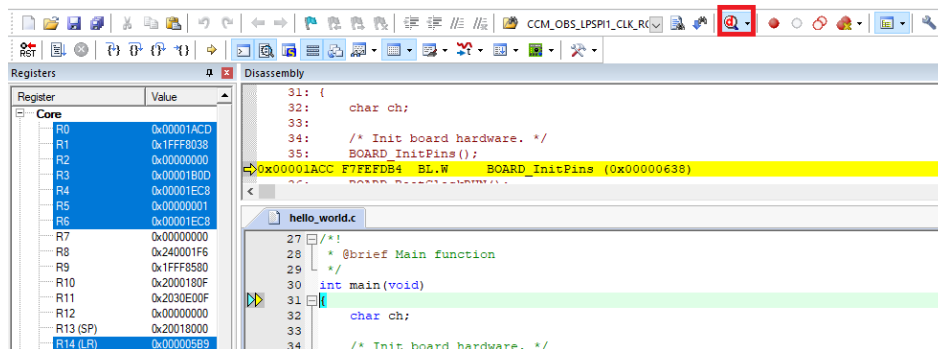


Figure 25. Stop at main() when run debugging

- Run the code by clicking **Run** to start the application.

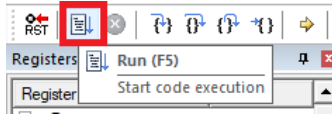
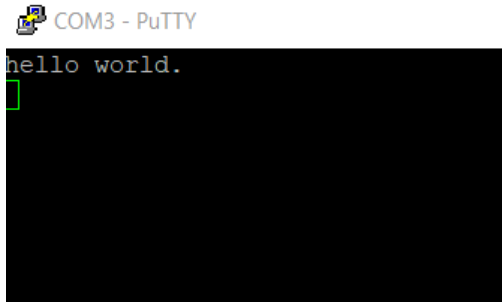


Figure 26. Go button

The `hello_world` application is now running and a banner is displayed on the terminal. If this does not appear, 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 TWR-KM35Z75M hardware platform which is used as an example.

NOTE

GCC ARM Embedded 8.2.1 is used as an example in this document. The latest GCC version for this package is as described in *MCUXpresso SDK Release Notes Supporting TWR-KM35Z75M* (document MCUXSDKKM35Z75MRN).

6.1 Set up toolchain

This section contains the steps to install the necessary components required to build and run an 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. 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 *MCUXpresso SDK Release Notes for TWR-KM35Z75M* (document MCUXSDKKM35Z75MRN).

6.1.2 Install MinGW (only required on Windows OS)

Run a demo using Arm® GCC

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 use the MinGW build tools, but does leverage the base install of both MinGW and MSYS. MSYS provides a basic shell with a Unix-like interface and tools.

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

NOTE

The installation path cannot contain any spaces.

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

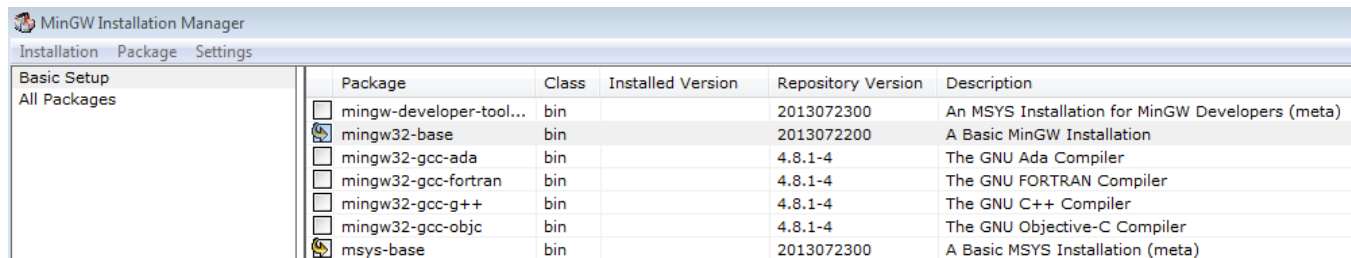


Figure 28. Set up MinGW and MSYS

4. In the **Installation** menu, click **Apply Changes** and follow the remaining instructions to complete the installation.

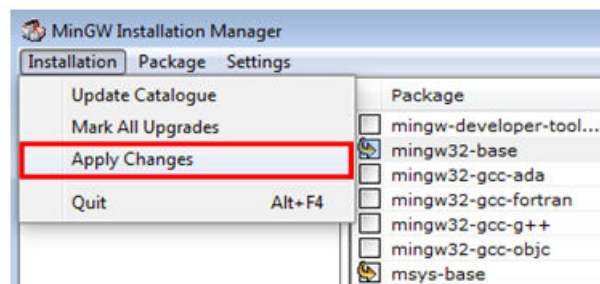


Figure 29. Complete MinGW and MSYS installation

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

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

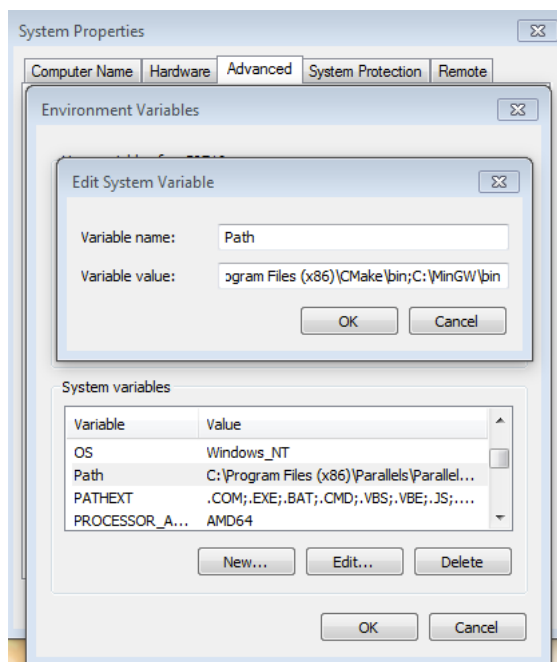


Figure 30. Add Path to systems environment

6.1.3 Add a new system environment variable for `ARMGCC_DIR`

Create a new *system* environment variable and name it as `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\8 2018-q4-major
```

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

Short path should be used for path setting, you could convert the path to short path by running command "*for %I in (.) do echo %~sI*" in above path.

```
C:\Program Files (x86)\GNU Tools Arm Embedded\8 2018-q4-major>for %I in (.) do echo %~sI
C:\Program Files (x86)\GNU Tools Arm Embedded\8 2018-q4-major>echo C:\PROGRA~2\GNUTOO~1\82018~1
C:\PROGRA~2\GNUTOO~1\82018~1
```

Figure 31. Convert path to short path

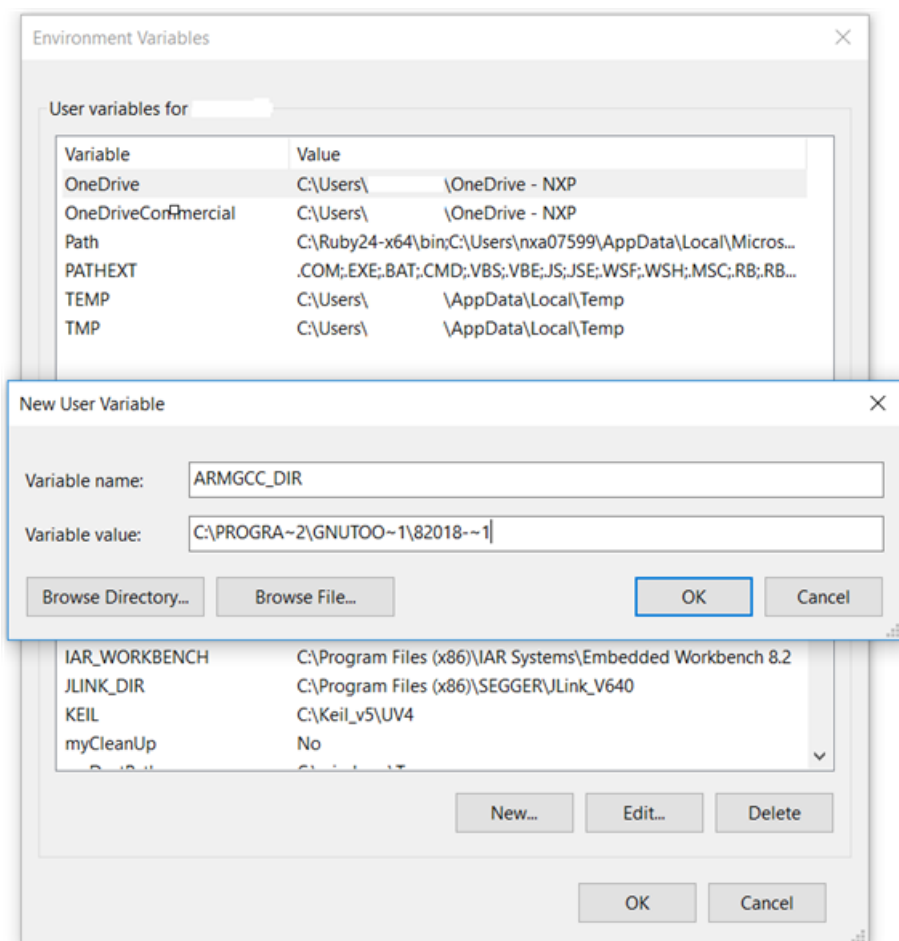


Figure 32. Add ARMGCC_DIR system variable

6.1.4 Install CMake

1. Download CMake 3.0.x from www.cmake.org/cmake/resources/software.html.
2. Install CMake, ensuring that the option **Add CMake to system PATH** is selected when installing. 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.

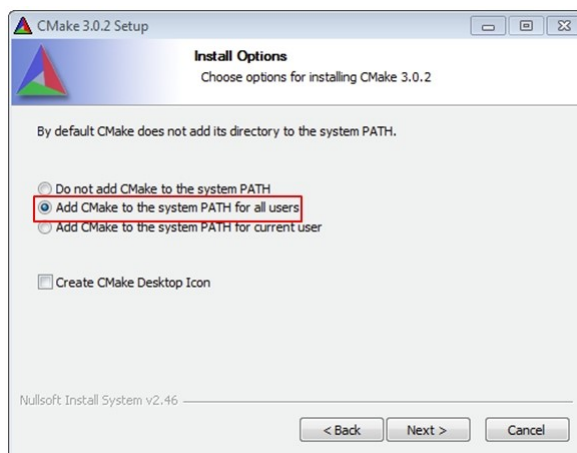


Figure 33. Install CMake

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**.

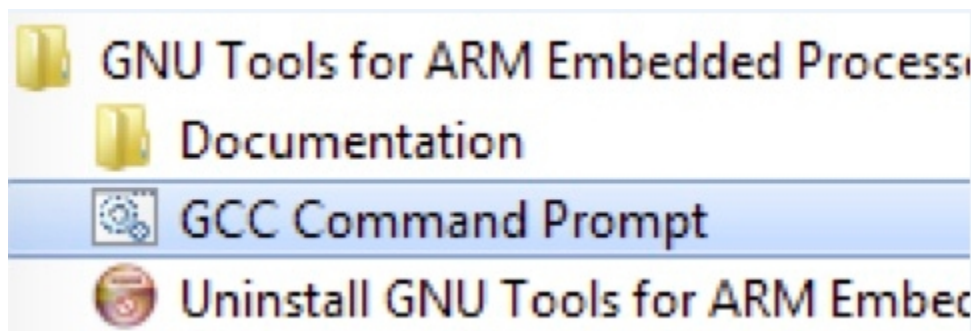


Figure 34. Launch command prompt

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>/examples/twrkm35z75m/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:

```
[ 86%] Building C object CMakeFiles/hello_world.elf.dir/C:/nxp/SDK_2.7.0_TWR-KM35Z75M/devices/MKM35Z7/drivers/fsl_smc.c.obj
[ 90%] Building C object CMakeFiles/hello_world.elf.dir/C:/nxp/SDK_2.7.0_TWR-KM35Z75M/devices/MKM35Z7/utilities/fsl_assembler.c.obj
[ 95%] Building C object CMakeFiles/hello_world.elf.dir/C:/nxp/SDK_2.7.0_TWR-KM35Z75M/devices/MKM35Z7/utilities/fsl_sbrk.c.obj
[100%] Linking C executable debug\hello_world.elf
[100%] Built target hello_world.elf
```

Figure 35. 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 [Default debug interfaces](#). For instructions on reprogramming the OpenSDA interface, see [Updating debugger firmware](#). 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 (may be named OSJTAG for some boards) and the PC USB connector. If using a standalone J-Link debug pod, also connect it to the SWD/JTAG connector of the board.
2. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug serial port number (to determine the COM port number, see [How to determine COM port](#). Configure the terminal with these settings:
 - a. 115200 or 9600 baud rate, depending on your board (reference BOARD_DEBUG_UART_BAUDRATE variable in the board.h file)
 - b. No parity
 - c. 8 data bits
 - d. 1 stop bit

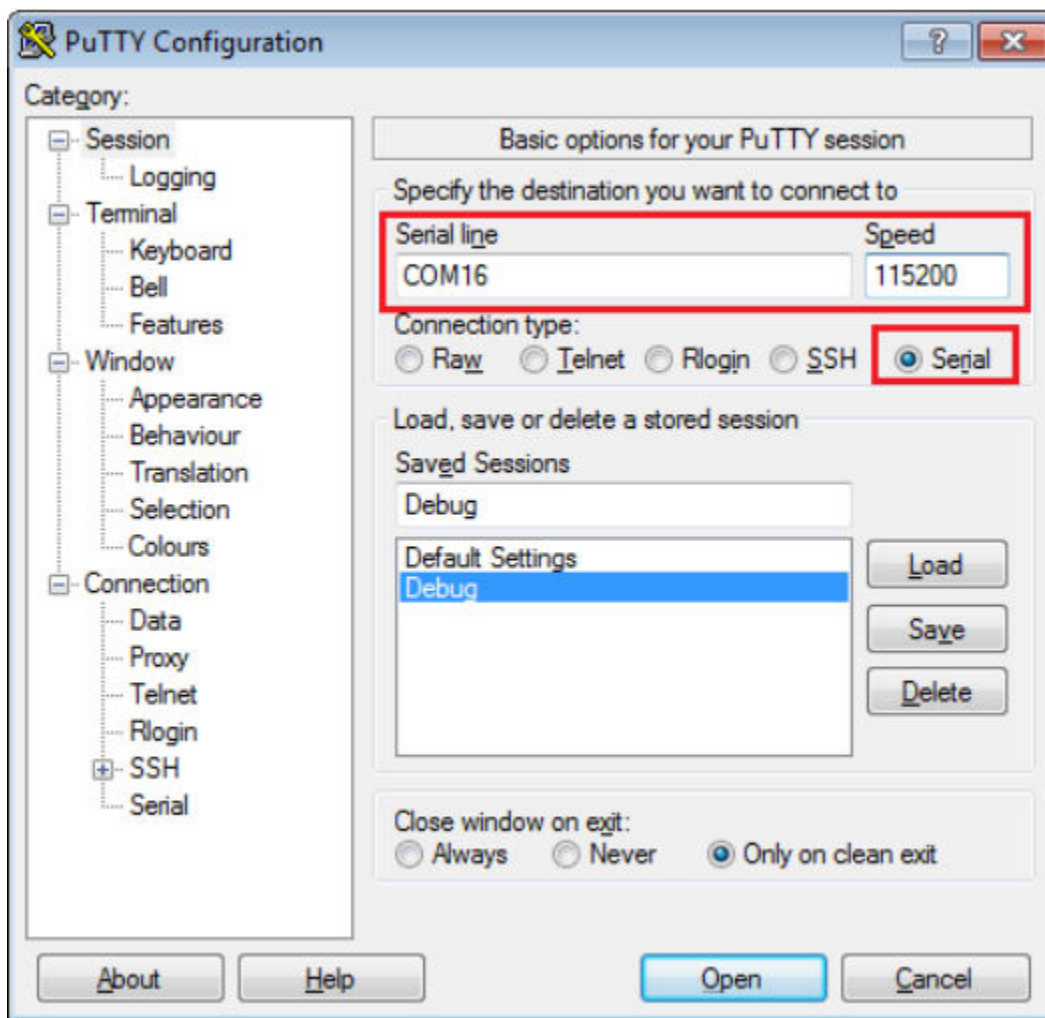


Figure 36. Terminal (PuTTY) configurations

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 MKM35Z512xxx7.
5. After it is connected, the screen should resemble [Figure 37](#) :

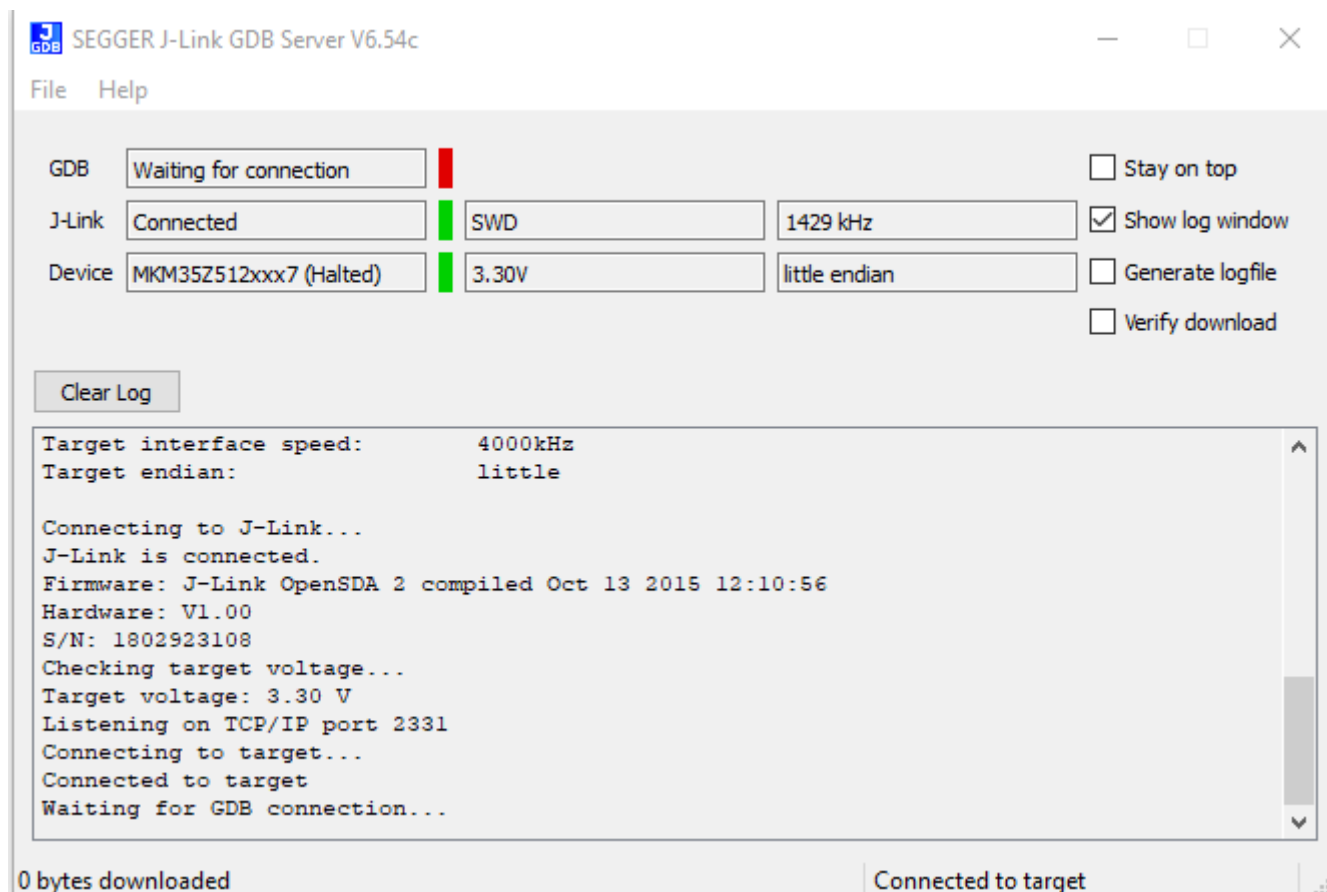


Figure 37. SEGGER J-Link GDB Server screen after successful connection

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**.

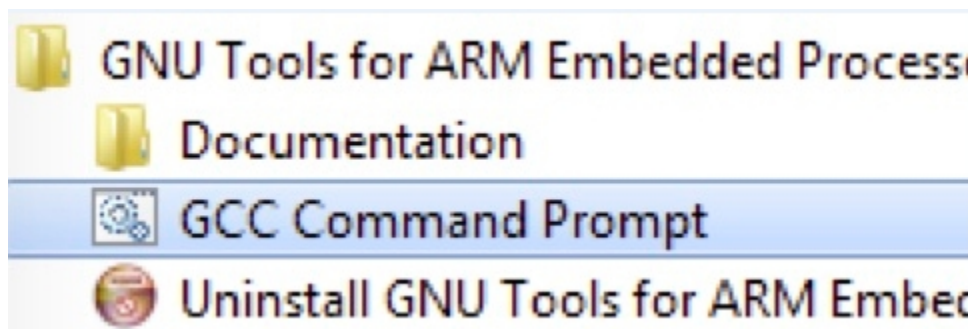


Figure 38. Launch command prompt

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:

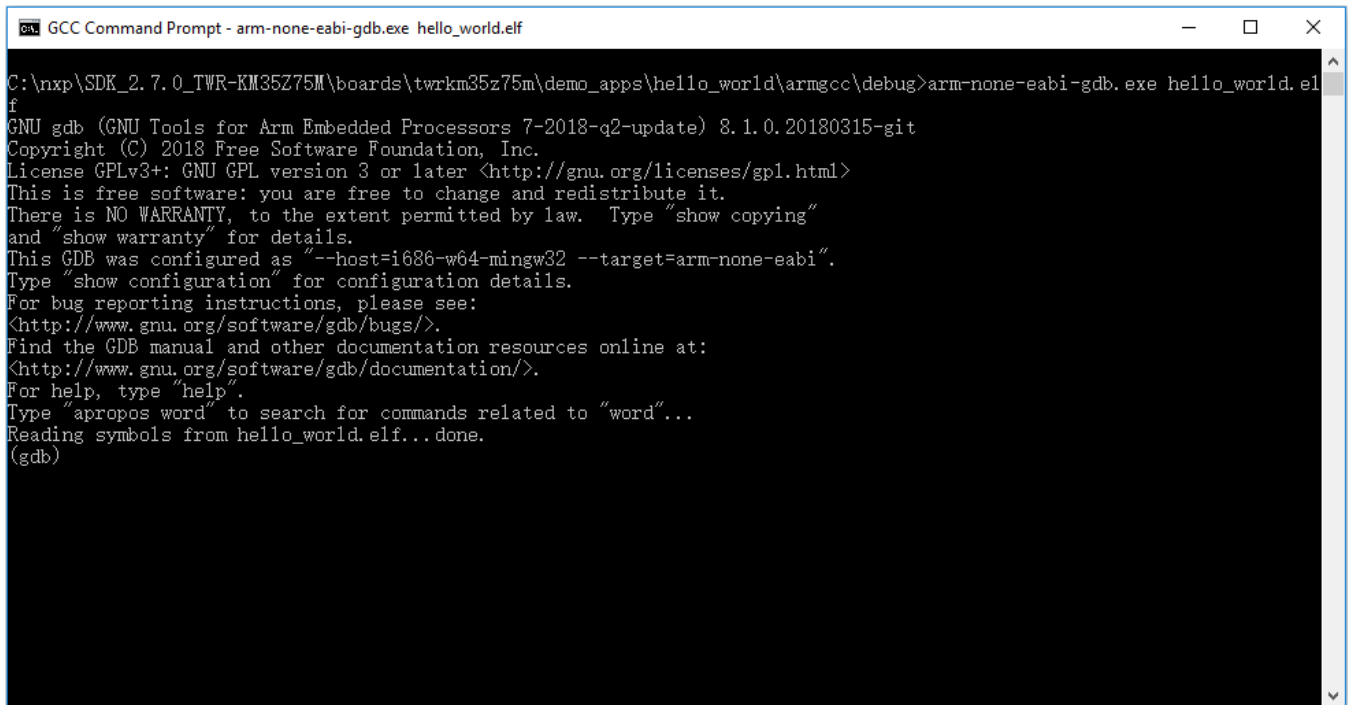
`<install_dir>/boards/<board_name>/<example_type>/<application_name>/armgcc/debug`

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

For this example, the path is:

`<install_dir>/boards/twrkm35z75m/demo_apps/hello_world/armgcc/debug`

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



```

C:\nxp\SDK_2.7.0_TWR-KM35Z75M\boards\twrkm35z75m\demo_apps\hello_world\armgcc\debug>arm-none-eabi-gdb.exe hello_world.elf
GNU gdb (GNU Tools for Arm Embedded Processors 7-2018-q2-update) 8.1.0.20180315-git
Copyright (C) 2018 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. Running `arm-none-eabi-gdb`

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

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

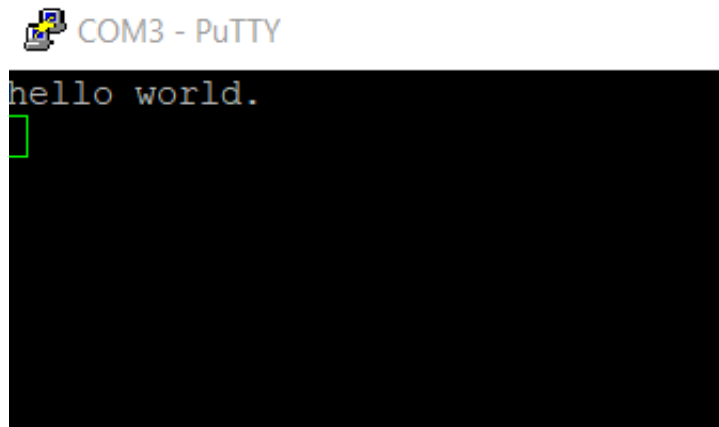







Figure 40. Text display of the `hello_world` demo

7 MCUXpresso Config Tools

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.

Table 1 describes the tools included in the MCUXpresso Config Tools.

Table 1. MCUXpresso Config Tools

Config Tool	Description	Image
Pins tool	For configuration of pin routing and pin electrical properties.	
Clock tool	For system clock configuration	
Peripherals tools	For configuration of other peripherals	
TEE tool	Configures access policies for memory area and peripherals helping to protect and isolate sensitive parts of the application.	
Device Configuration tool	Configures Device Configuration Data (DCD) contained in the program image that the Boot ROM code interprets to setup various on-chip peripherals prior the program launch.	

MCUXpresso Config Tools can be accessed in the following products:

- **Integrated** in the MCUXpresso IDE. Config tools are integrated with both compiler and debugger which makes it the easiest way to begin the development.
- **Standalone version** available for download from www.nxp.com. Recommended for customers using IAR Embedded Workbench, Keil MDK μ Vision, or Arm GCC.
- **Online version** available on 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 MCUXpresso IDE Config Tools installation folder 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). It offers user the flexibility to select and change multiple builds. The wizard also includes a library and provides source code options. The source code is organized as software components, categorized as drivers, 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 **New project**, as shown in [Figure 41](#).



Figure 41. MCUXpresso IDE Quickstart Panel

For more details and usage of new project wizard, see the *MCUXpresso_IDE_User_Guide.pdf* in the MCUXpresso IDE installation folder.

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. All NXP boards ship with a factory programmed, on-board debug interface, whether it's based on OpenSDA or the legacy P&E Micro OSJTAG interface. To determine what your specific board ships with, see [Default debug interfaces](#).

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 and the other is for Cortex M4.

2. **Windows:** To determine the COM port open Device Manager in the Windows operating system. Click on the **Start** menu and type **Device Manager** in the search bar.

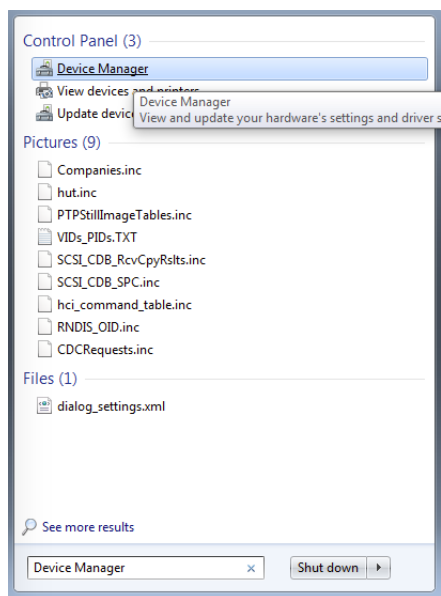


Figure A-1. Device Manager

3. In the Device Manager, expand the **Ports (COM & LPT)** section to view the available ports. The COM port names will be different for all the NXP boards.
 - a. **OpenSDA – CMSIS-DAP/mbed/DAPLink** interface:

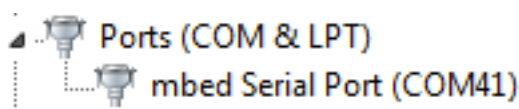


Figure A-2. OpenSDA – CMSIS-DAP/mbed/DAPLink interface

- b. **OpenSDA – P&E Micro:**

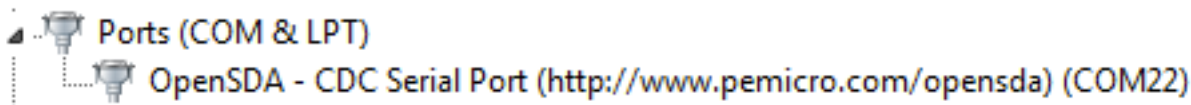


Figure A-3. OpenSDA – P&E Micro

- c. **OpenSDA – J-Link:**

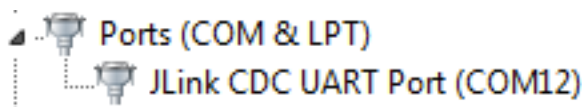


Figure A-4. OpenSDA – J-Link

- d. **P&E Micro OSJTAG:**

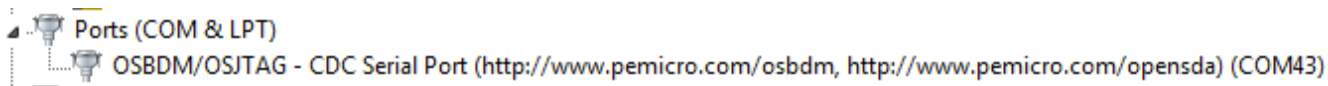


Figure A-5. P&E Micro OSJTAG

- e. **MRB-KW01:**

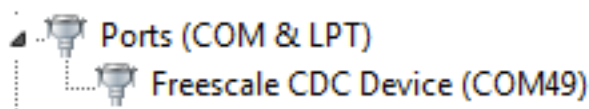


Figure A-6. MRB-KW01

Appendix B Default debug interfaces

The MCUXpresso SDK supports various hardware platforms that come loaded with a variety of factory programmed debug interface configurations. [Table B-1](#) 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 in [Table B-1](#) is not applicable to LPC.

Table B-1. Hardware platforms supported by MCUXpresso SDK

Hardware platform	Default interface	OpenSDA details
EVK-MC56F83000	P&E Micro OSJTAG	N/A
EVK-MIMXRT595	CMSIS-DAP	N/A
EVK-MIMXRT685	CMSIS-DAP	N/A
FRDM-K22F	CMSIS-DAP/mbd/DAPLink	OpenSDA v2.1
FRDM-K28F	DAPLink	OpenSDA v2.1
FRDM-K32L2A4S	CMSIS-DAP	OpenSDA v2.1
FRDM-K32L2B	CMSIS-DAP	OpenSDA v2.1
FRDM-K32W042	CMSIS-DAP	N/A
FRDM-K64F	CMSIS-DAP/mbd/DAPLink	OpenSDA v2.0
FRDM-K66F	J-Link OpenSDA	OpenSDA v2.1
FRDM-K82F	CMSIS-DAP	OpenSDA v2.1
FRDM-KE15Z	DAPLink	OpenSDA v2.1
FRDM-KE16Z	CMSIS-DAP/mbd/DAPLink	OpenSDA v2.2
FRDM-KL02Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL03Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL25Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL26Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL27Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL28Z	P&E Micro OpenSDA	OpenSDA v2.1
FRDM-KL43Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL46Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL81Z	CMSIS-DAP	OpenSDA v2.0
FRDM-KL82Z	CMSIS-DAP	OpenSDA v2.0
FRDM-KV10Z	CMSIS-DAP	OpenSDA v2.1
FRDM-KV11Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KV31F	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KW24	CMSIS-DAP/mbd/DAPLink	OpenSDA v2.1
FRDM-KW36	DAPLink	OpenSDA v2.2
FRDM-KW41Z	CMSIS-DAP/DAPLink	OpenSDA v2.1 or greater
Hexiwear	CMSIS-DAP/mbd/DAPLink	OpenSDA v2.0
HVP-KE18F	DAPLink	OpenSDA v2.2
HVP-KV46F150M	P&E Micro OpenSDA	OpenSDA v1

Table continues on the next page...

Table B-1. Hardware platforms supported by MCUXpresso SDK (continued)

Hardware platform	Default interface	OpenSDA details
HVP-KV11Z75M	CMSIS-DAP	OpenSDA v2.1
HVP-KV58F	CMSIS-DAP	OpenSDA v2.1
HVP-KV31F120M	P&E Micro OpenSDA	OpenSDA v1
JN5189DK6	CMSIS-DAP	N/A
LPC54018 IoT Module	N/A	N/A
LPCXpresso54018	CMSIS-DAP	N/A
LPCXpresso54102	CMSIS-DAP	N/A
LPCXpresso54114	CMSIS-DAP	N/A
LPCXpresso51U68	CMSIS-DAP	N/A
LPCXpresso54608	CMSIS-DAP	N/A
LPCXpresso54618	CMSIS-DAP	N/A
LPCXpresso54628	CMSIS-DAP	N/A
LPCXpresso54S018M	CMSIS-DAP	N/A
LPCXpresso55s16	CMSIS-DAP	N/A
LPCXpresso55s28	CMSIS-DAP	N/A
LPCXpresso55s69	CMSIS-DAP	N/A
MAPS-KS22	J-Link OpenSDA	OpenSDA v2.0
MIMXRT1170-EVK	CMSIS-DAP	N/A
TWR-K21D50M	P&E Micro OSJTAG	N/AOpenSDA v2.0
TWR-K21F120M	P&E Micro OSJTAG	N/A
TWR-K22F120M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-K24F120M	CMSIS-DAP/mbd	OpenSDA v2.1
TWR-K60D100M	P&E Micro OSJTAG	N/A
TWR-K64D120M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-K64F120M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-K65D180M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-K65D180M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KV10Z32	P&E Micro OpenSDA	OpenSDA v1.0
TWR-K80F150M	CMSIS-DAP	OpenSDA v2.1
TWR-K81F150M	CMSIS-DAP	OpenSDA v2.1
TWR-KE18F	DAPLink	OpenSDA v2.1
TWR-KL28Z72M	P&E Micro OpenSDA	OpenSDA v2.1
TWR-KL43Z48M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KL81Z72M	CMSIS-DAP	OpenSDA v2.0
TWR-KL82Z72M	CMSIS-DAP	OpenSDA v2.0
TWR-KM34Z75M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KM35Z75M	DAPLink	OpenSDA v2.2
TWR-KV10Z32	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KV11Z75M	P&E Micro OpenSDA	OpenSDA v1.0

Table continues on the next page...

Table B-1. Hardware platforms supported by MCUXpresso SDK (continued)

Hardware platform	Default interface	OpenSDA details
TWR-KV31F120M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KV46F150M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KV58F220M	CMSIS-DAP	OpenSDA v2.1
TWR-KW24D512	P&E Micro OpenSDA	OpenSDA v1.0
USB-KW24D512	N/A External probe	N/A
USB-KW41Z	CMSIS-DAP/DAPLink	OpenSDA v2.1 or greater

Appendix C Updating debugger firmware

C.1 Updating OpenSDA 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/mbd/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 restoring the original image also. For reference, OpenSDA firmware files can be found at the links below:

- **J-Link:** Download appropriate image from www.segger.com/opensda.html. Choose 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/mbd/DAPLink:** DAPLink OpenSDA firmware is available at www.nxp.com/opensda.
- **P&E Micro:** Downloading P&E Micro OpenSDA firmware images requires registration with P&E Micro (www.pemicro.com).

Perform the following steps to update the OpenSDA firmware on your board for Windows and Linux OS users:

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

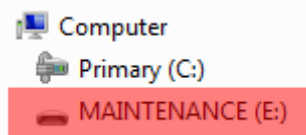


Figure C-1. MAINTENANCE drive

4. Drag and drop the new firmware image onto the MAINTENANCE drive.

NOTE

If for any reason the firmware update fails, the board can always re-enter maintenance mode by holding down **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 **Reset** button of the board. While still holding the button, plug the USB cable back into the board.

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**.
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**Web Support:**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.

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, 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, μ 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.

© 2019-2020 NXP B.V.

Document Number MCUXSDKKM35Z75MGSUG
Revision 1, 26 May 2020

