**NXP Semiconductors**
User's Guide

Document identifier: MCUXSDKMIMXRT5XXGSUG
Rev. 0, January 21, 2021

# Getting Started with MCUXpresso SDK for EVK-MIMXRT595

# Contents

# Chapter 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 EVK-MIMXRT595* (document MCUSDKRT595RN).

For more details about MCUXpresso SDK, see MCUXpresso Software Development Kit (SDK).

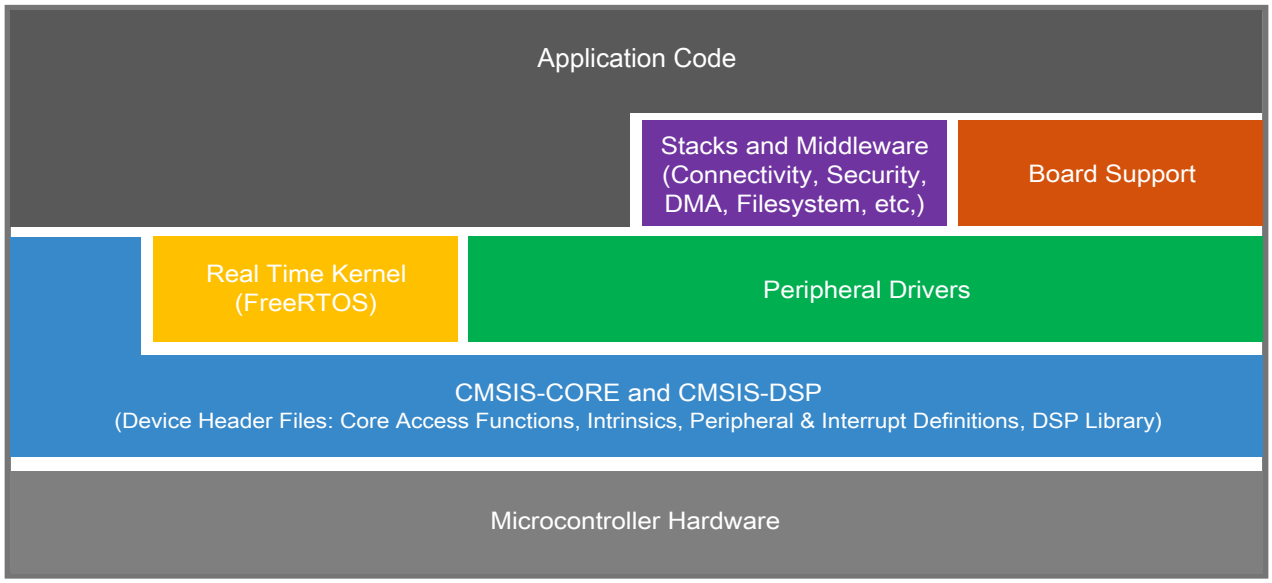


**Figure 1. MCUXpresso SDK layers**

# Chapter 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, RT-EVK, 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):

- `cmsis_driver_examples`: Simple applications intended to show how to use CMSIS drivers.
- `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).
- `emwin_examples`: Applications that use the emWin GUI widgets
- `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
- `usb_examples`: Applications that use the USB host/device/OTG stack

## 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:

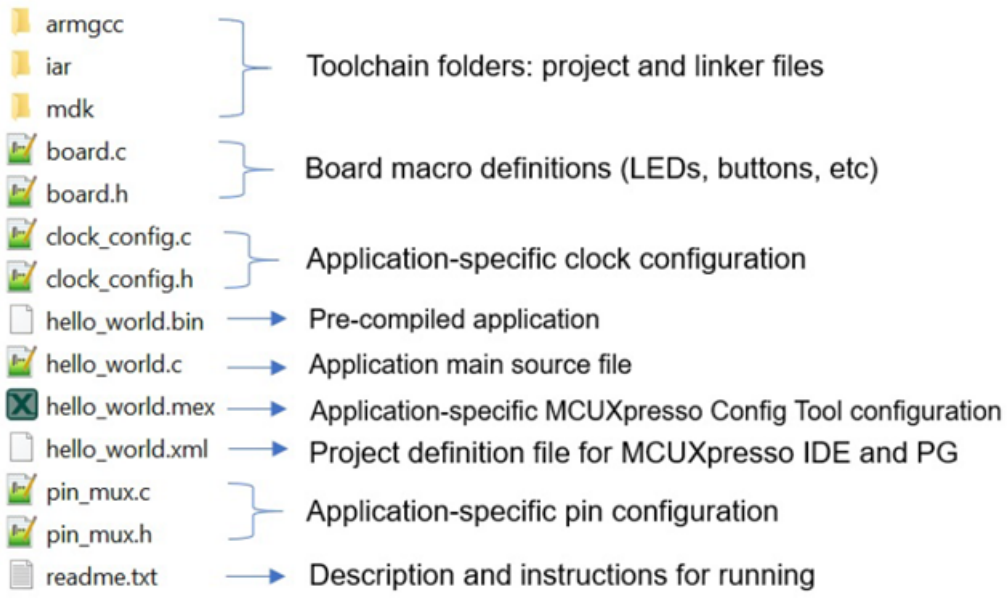


Figure 2. Application folder structure

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, 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 files
- *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, including vector table definitions.
- *devices/<device_name>/project*: Project template used in CMSIS PACK new project creation
- *devices/<device_name>/utilities*: Items such as the debug console that are used by many of the example applications

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

# Chapter 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 to build, run, and debug example applications. The `hello_world` demo application targeted for the MIMXRT595-EVK 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.

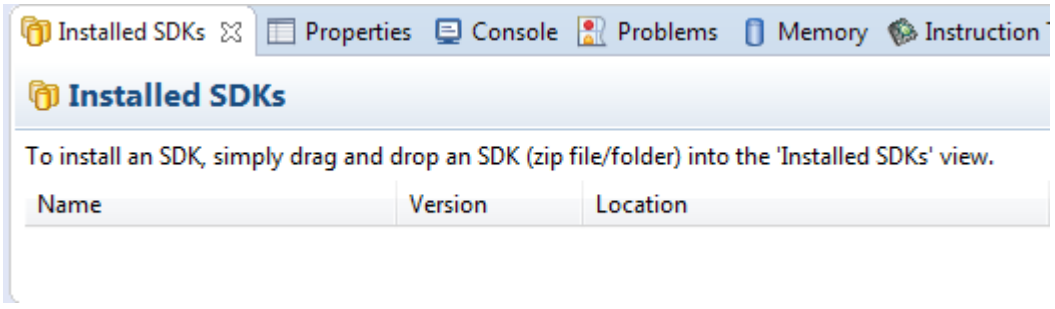


**Figure 3. Install an SDK**

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

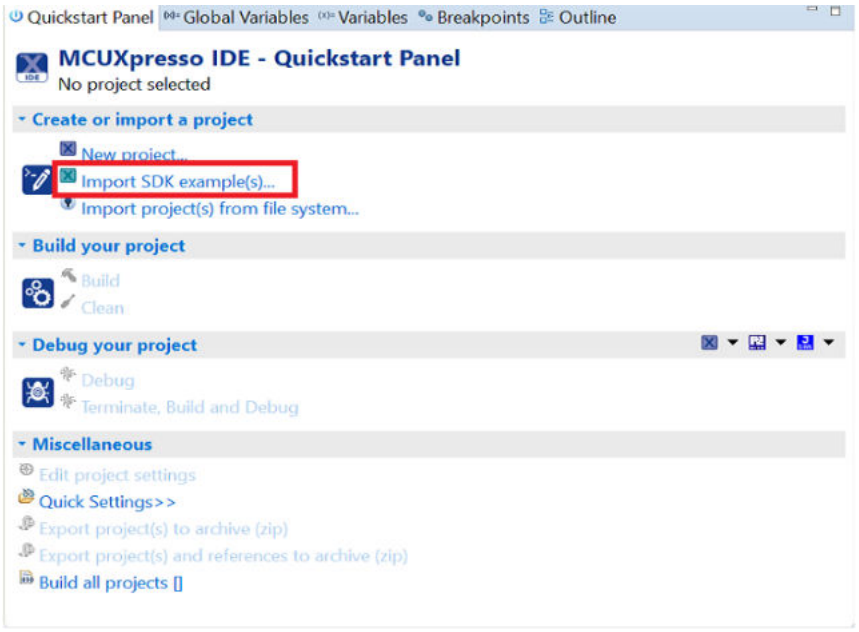


Figure 4. Import an SDK example

3. In the window that appears, expand the **MIMXRT500** folder and select **MIMXRT595S**. Then, select **evkmimxrt595** and click **Next**.

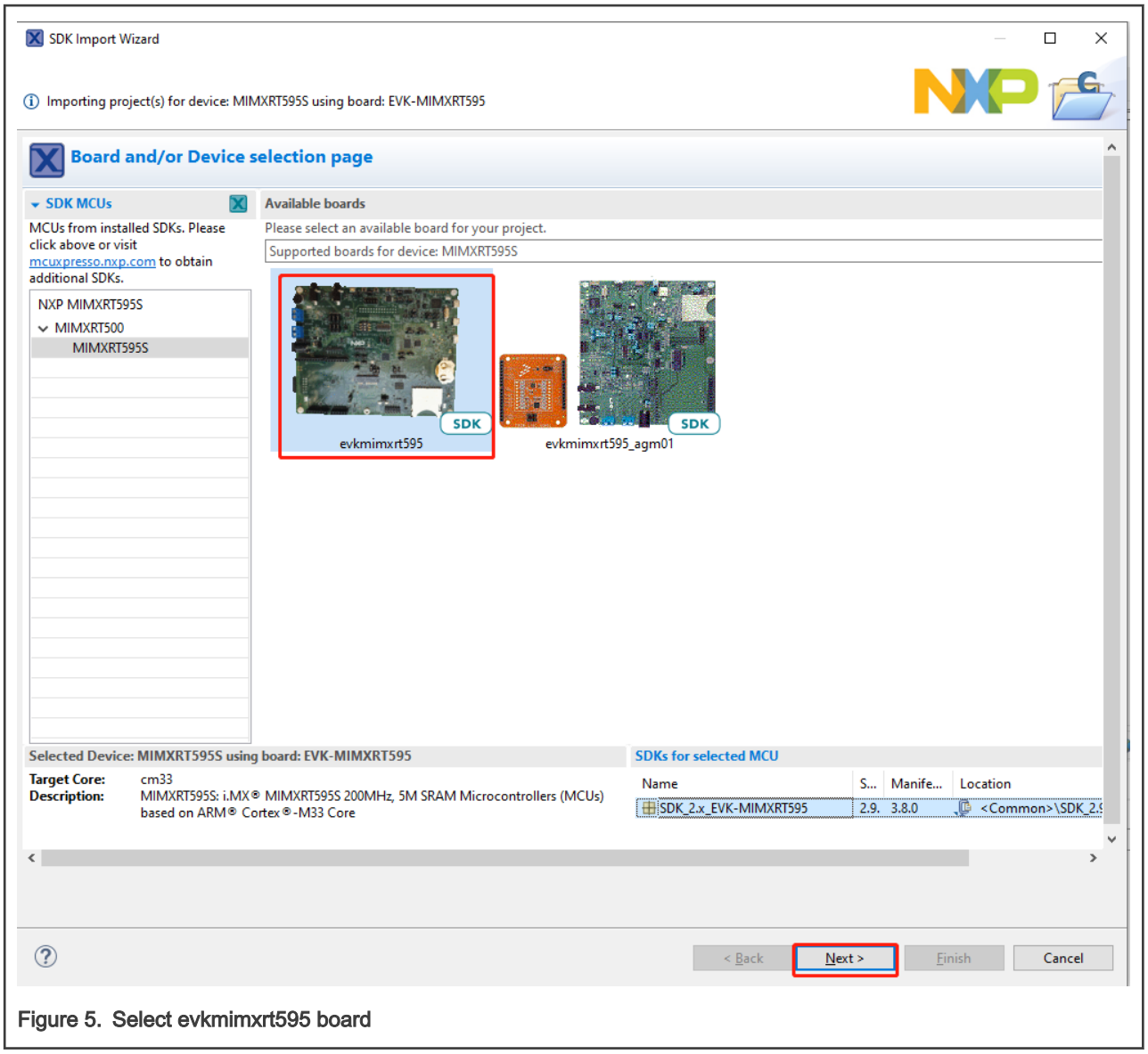


Figure 5. Select evkmimxrt595 board

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

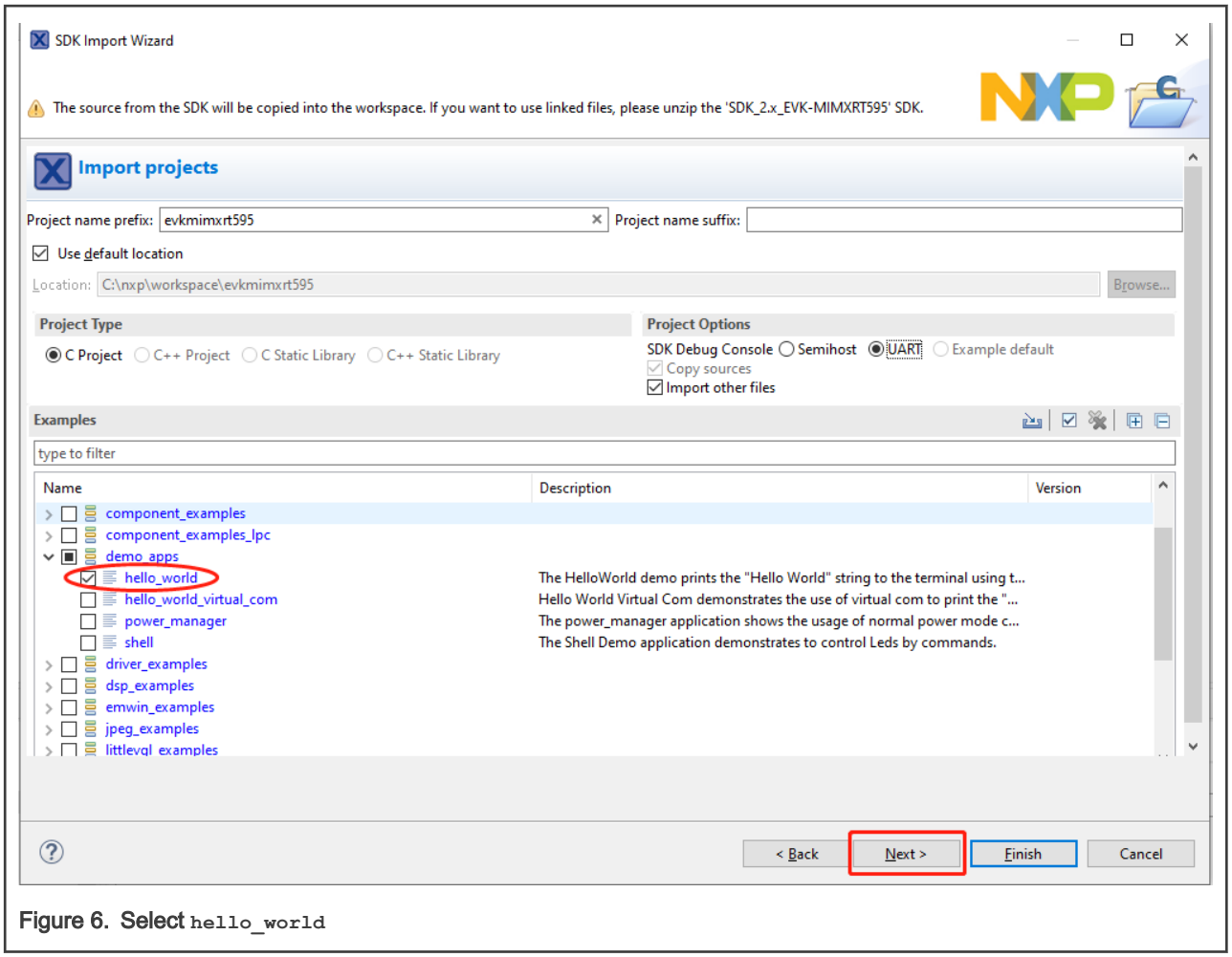


**Figure 6. Select `hello_world`**

5. Ensure **Redlib: Use floating point version of printf** is selected if the example prints floating point numbers on the terminal. Otherwise, it is not necessary to select this option. Then, click **Finish**.

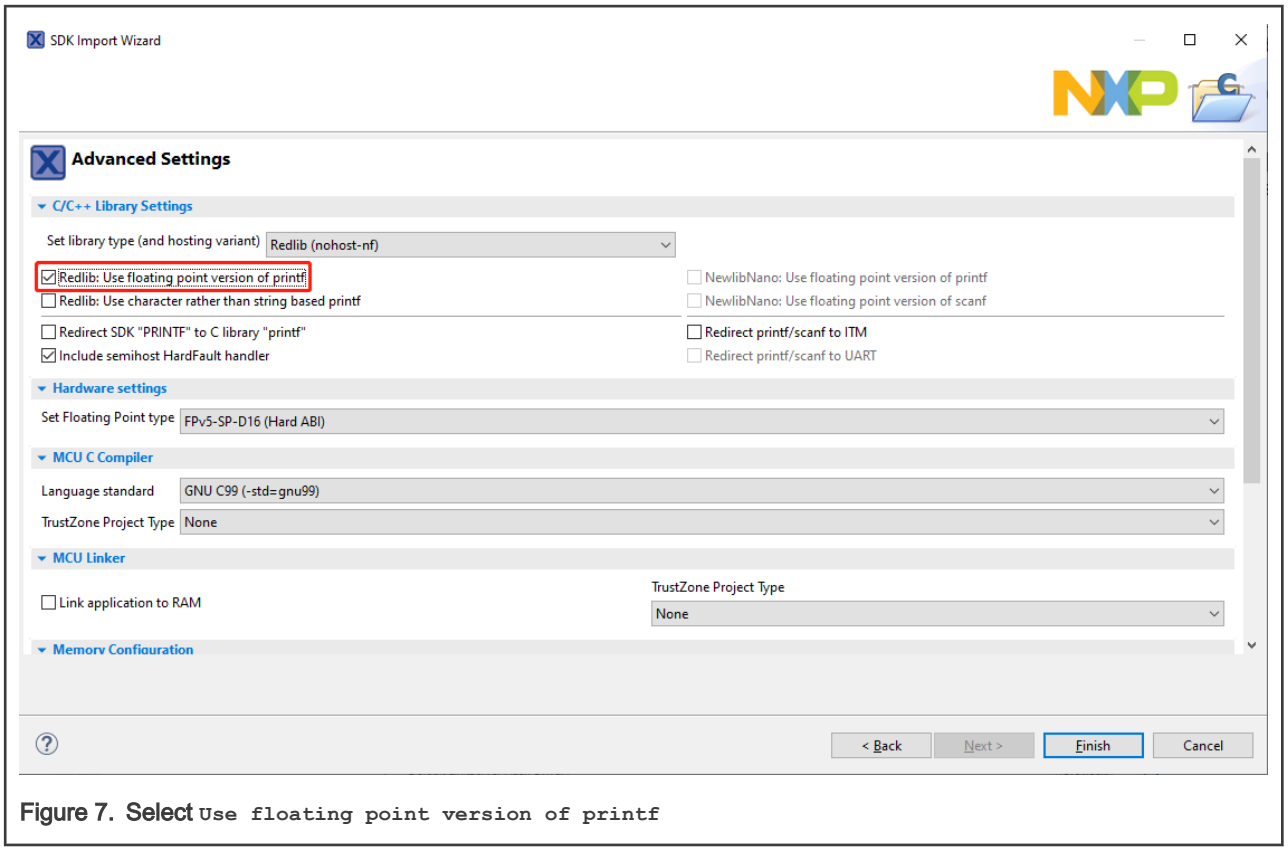


**Figure 7. Select `Use floating point version of printf`**

## 3.3 Run an example application

For more information on debug probe support in the MCUXpresso IDE, see NXP website.

To download and run the application, perform the following steps:

1. See Table 1 to determine the debug interface that comes loaded on your specific hardware platform.
   - For boards with a P&E Micro interface, see PE micro to download and install the P&E Micro Hardware Interface Drivers package.
2. Connect the development platform to your PC via a USB cable. Make sure the boot mode is FLEXSPI boot, where SW7 [1-3] is [**OFF, OFF, ON**].
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:
   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

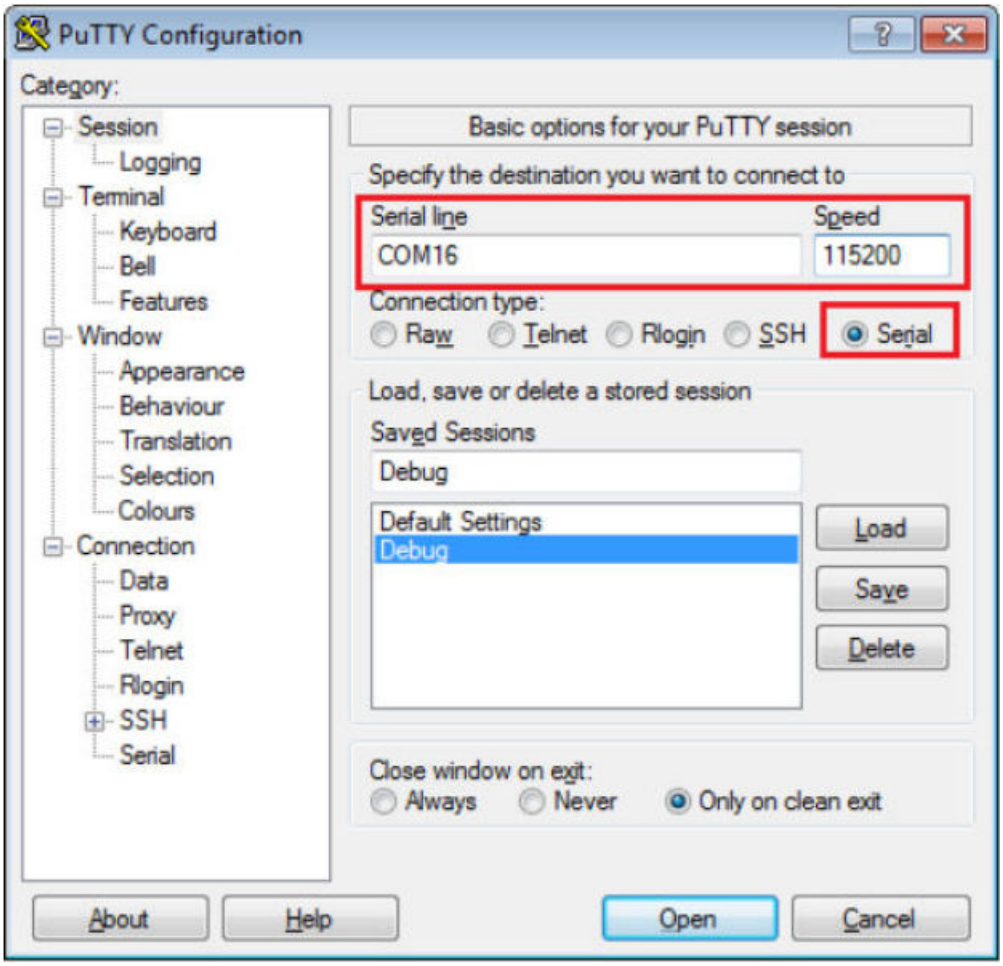


Figure 8. Terminal (PuTTY) configurations

4. On the **Quickstart Panel**, click on **Debug evkmimxrt595_hello_world [Debug]** to launch the debug session.

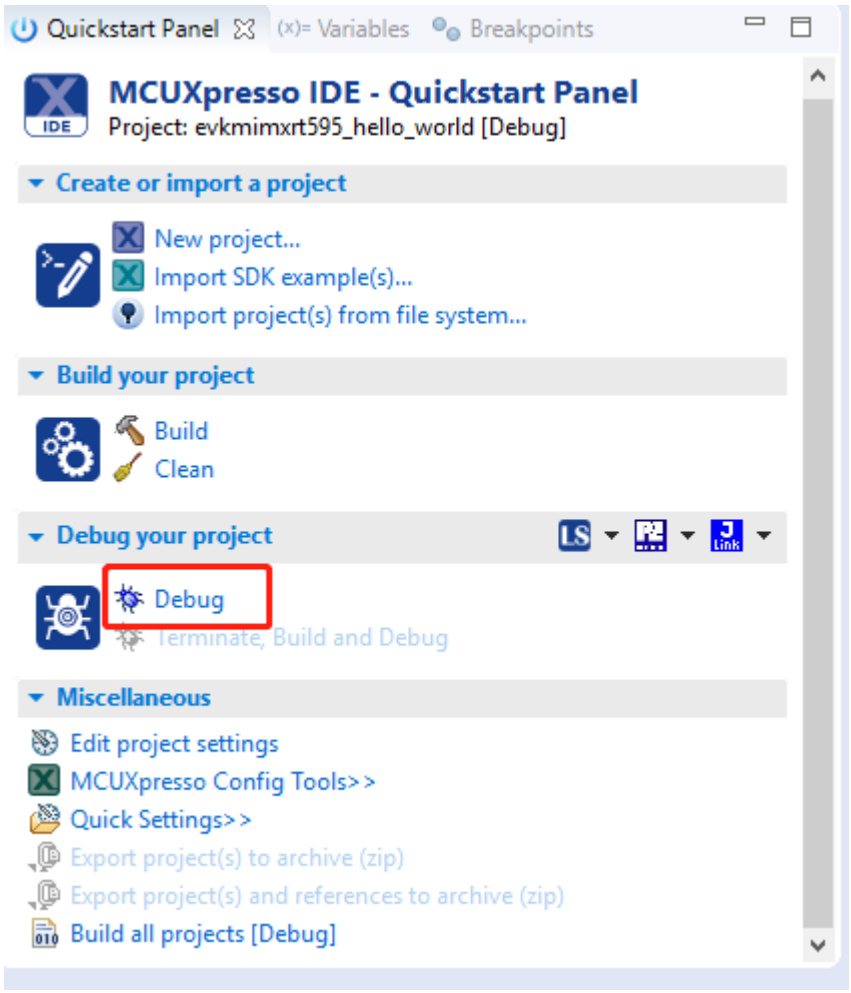


**Figure 9. Debug `hello_world` case**

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 **OK**.

**NOTE**

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

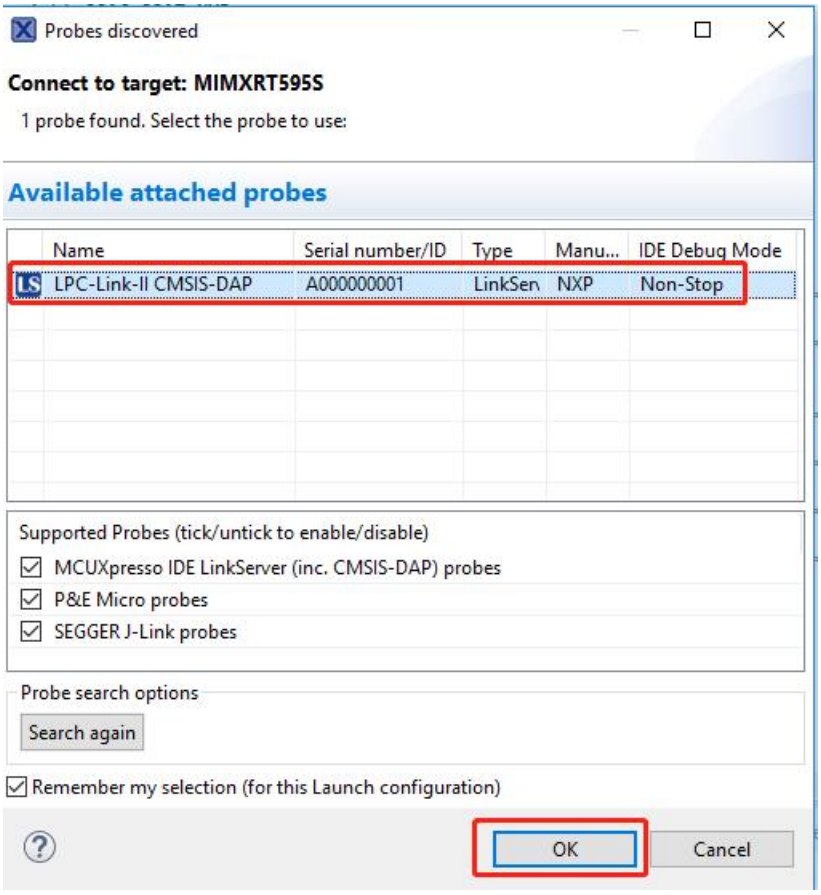


Figure 10. Attached Probes: debug emulator selection

**NOTE**

If the debugging application is running in flash, make sure the board is set to FlexSPI flash boot mode.

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

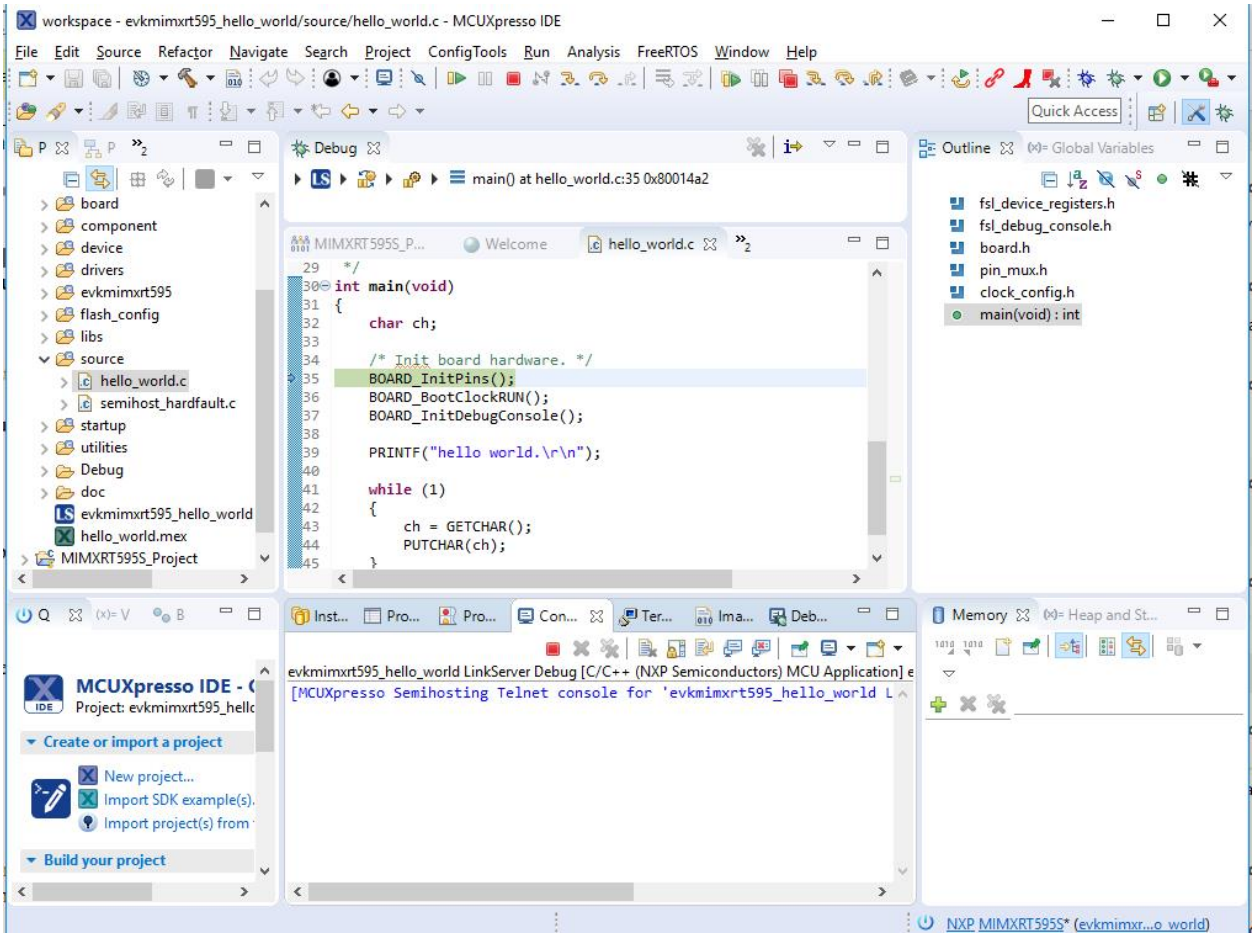


**Figure 11. Stop at `main()` when running debugging**

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.

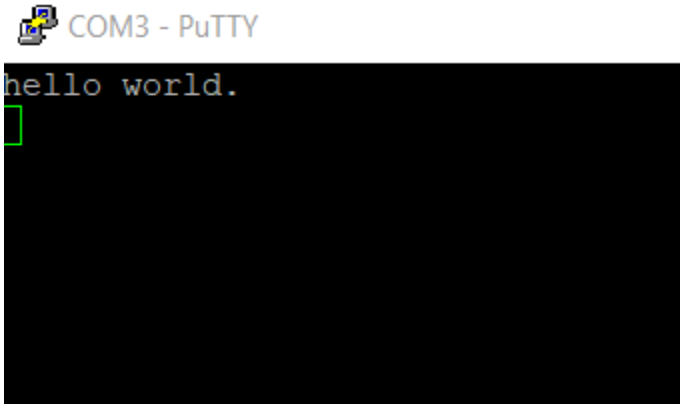


**Figure 13. Text display of the `hello_world` demo**

## 3.4 Build a TrustZone example application

This section describes the steps required to configure MCUXpresso IDE to build, run, and debug TrustZone example applications. The trustzone version of the `hello_world` example application targeted for the MIMXRT595-EVK hardware platform is used as an example, though these steps can be applied to any TrustZone example application in the MCUXpresso SDK.

1. TrustZone examples are imported into the workspace in a similar way as single core applications. When the SDK zip package for MIMXRT595-EVK is installed and available in the **Installed SDKs** view, click **Import SDK example(s)...** on the Quickstart Panel. In the window that appears, expand the **MIMXRT500** folder and select **MIMXRT595S**. Then, select **evkmimxrt595** and click **Next**.

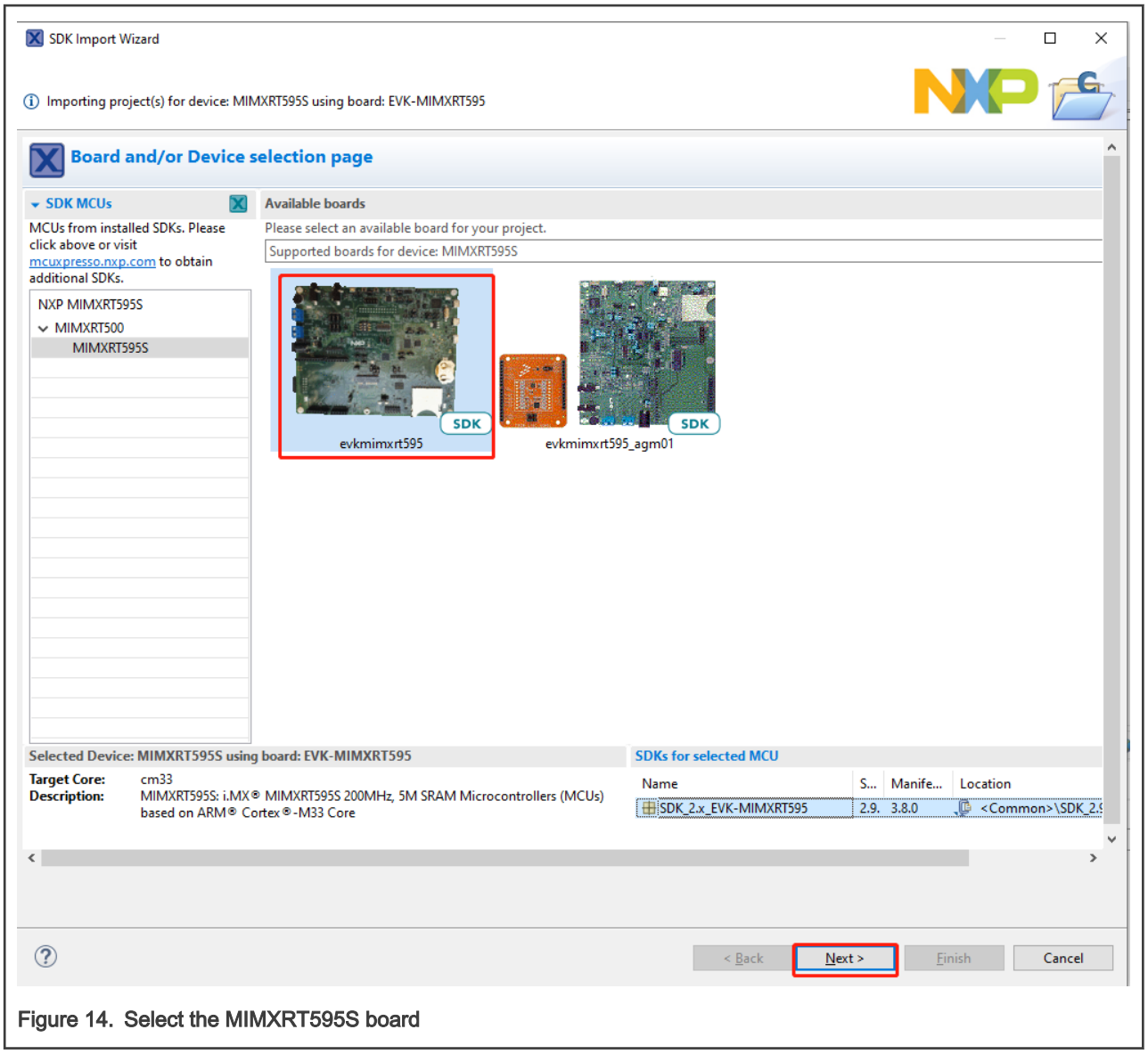


Figure 14. Select the MIMXRT595S board

2. Expand the `trustzone_examples/` folder and select `hello_world_s`. Because TrustZone examples are linked together, the non-secure project is automatically imported with the secure project, and there is no need to select it explicitly. Then, click **Finish**.

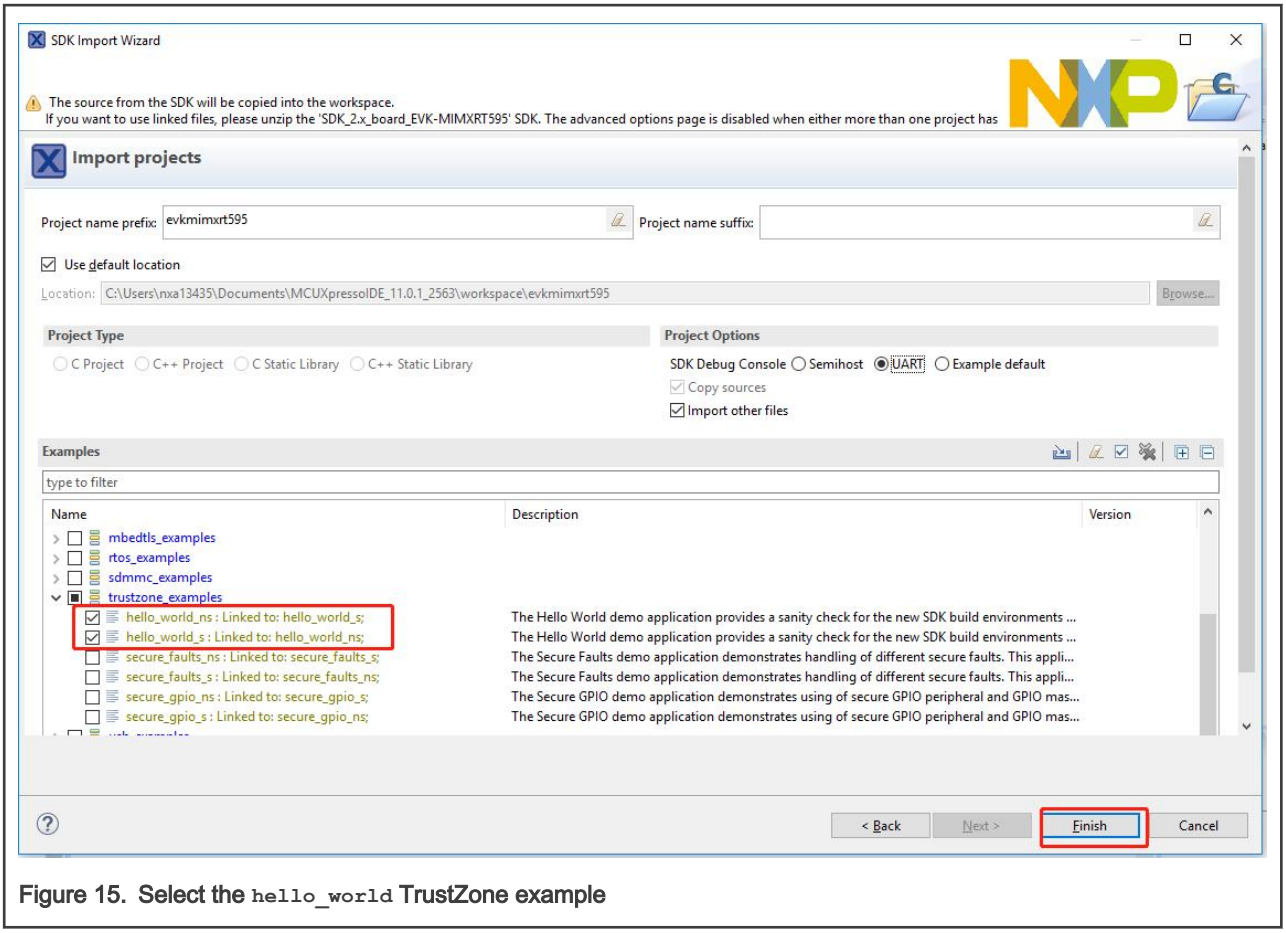


**Figure 15. Select the `hello_world` TrustZone example**

3. Now, two projects should be imported into the workspace. To start building the TrustZone application, highlight the `evkmimxrt595_hello_world_s` project (TrustZone master project) in the Project Explorer. Then, choose the appropriate build target, **Debug** or **Release**, by clicking the downward facing arrow next to the hammer icon, as shown in Figure 16. For this example, select the **Debug** target.


**Figure 16. Selection of the build target in MCUXpresso IDE**

The project starts building after the build target is selected. It is requested to build the application for the secure project first, because the non-secure project needs to know the secure project since CMSE library when running the linker. It is not possible to finish the non-secure project linker when the secure project since CMSE library is not ready.

**NOTE**

When the **Release** build is requested, it is necessary to change the build configuration of both the secure and non-secure application projects first. To do this, select both projects in the Project Explorer view by clicking to select the first project, then using shift-click or control-click to select the second project. Right click in the Project Explorer view to display the context-sensitive menu and select **Build Configurations** ->**Set Active** - >**Release**. This is also possible by using the menu item of **Project** ->**Build Configuration** - >**Set Active** ->**Release**. After switching to the **Release** build configuration, please build the application for the secure project first.

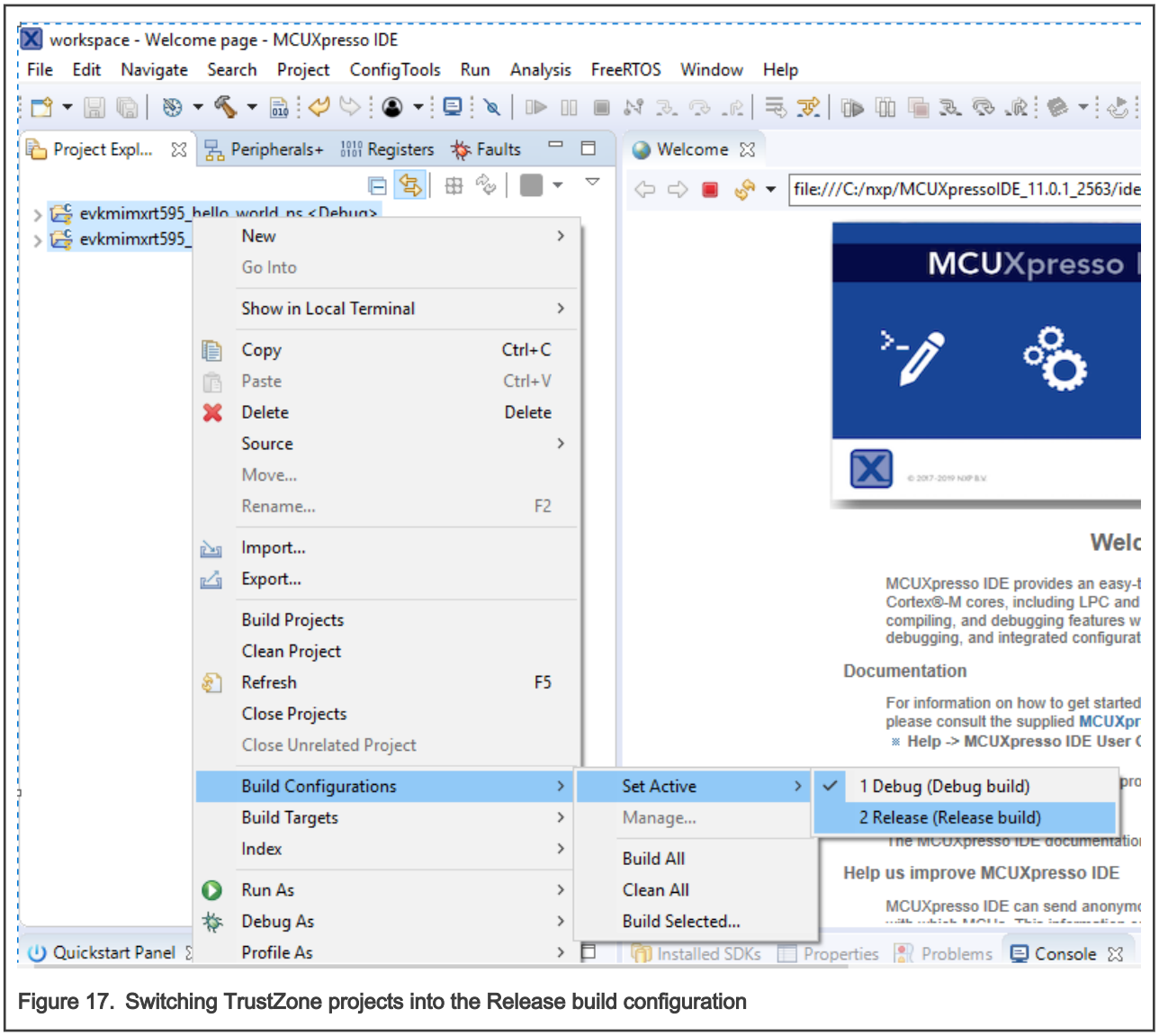


**Figure 17. Switching TrustZone projects into the Release build configuration**

## 3.5 Run a TrustZone example application

To download and run the application, perform all steps as described in Run an example application. These steps are common for single core, and TrustZone applications, ensuring `evkmimxrt595_hello_world_s` is selected for debugging.

In the **Quickstart Panel**, click **Debug** to launch the second debug session.

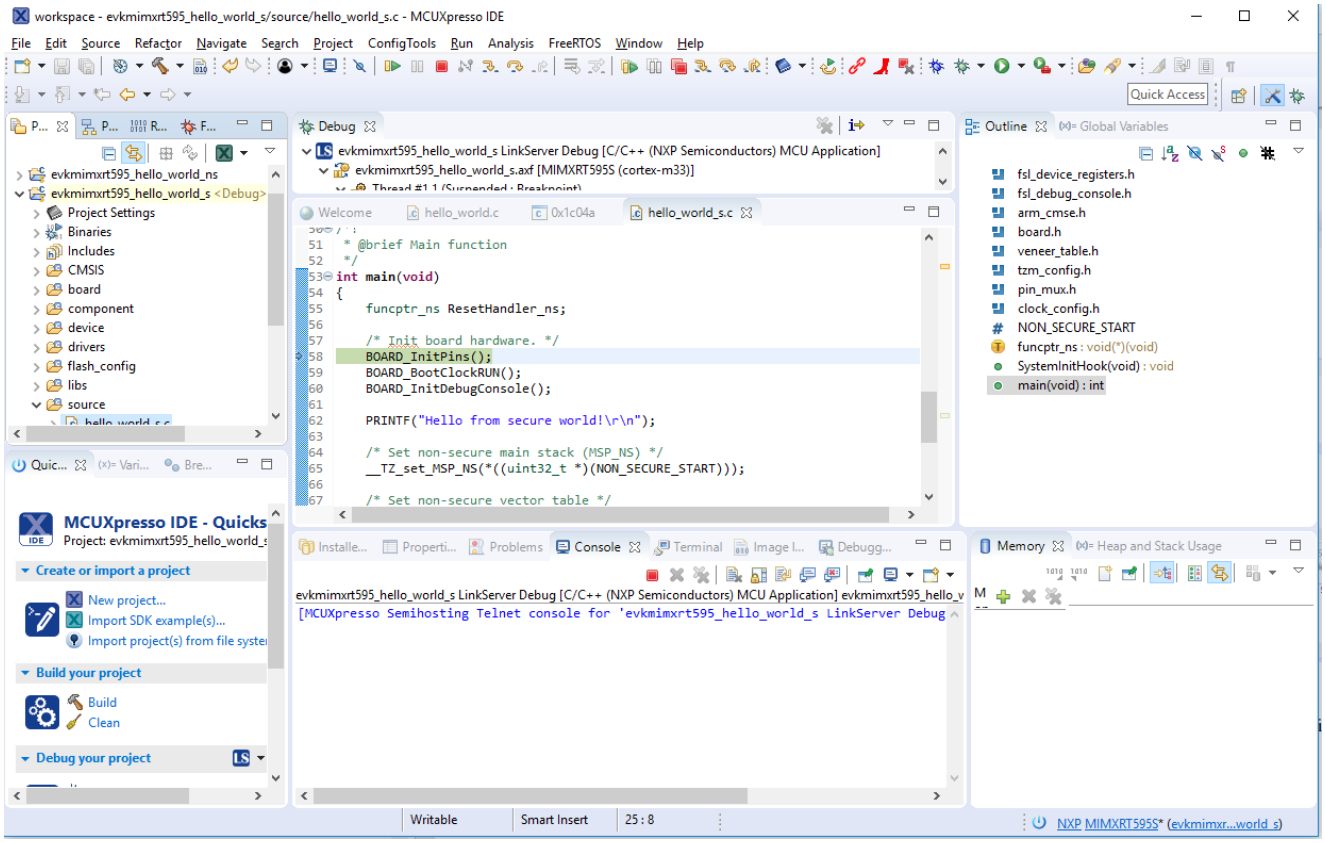


Figure 18. TrustZone debug sessions

Now, the TrustZone sessions should be opened. Click **Resume**. The `hello_world` TrustZone application then starts running, and the secure application starts the non-secure application during run time.

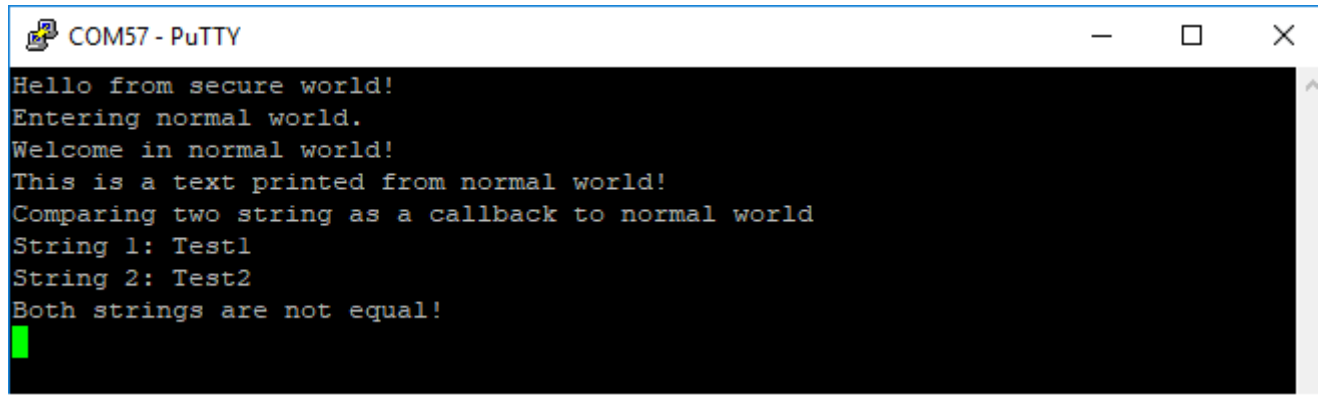


Figure 19. Run Hello World trustzone example and get the message

# Chapter 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**

The IAR toolchain should correspond to the latest supported version, as described in *MCUXpresso SDK Release Notes for EVK-MIMXRT595* (document MCUSDKRT595RN).

## 4.1 Build an example application

Do 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 MIMXRT595-EVK hardware platform as an example, the `hello_world` workspace is located in:

   *<install_dir>/boards/evkmimxrt595/demo_apps/hello_world/iar/hello_world.eww*

   Other example applications may have additional folders in their path.

2. Select the desired build target from the drop-down menu.

   For this example, select **hello_world** – **debug**.

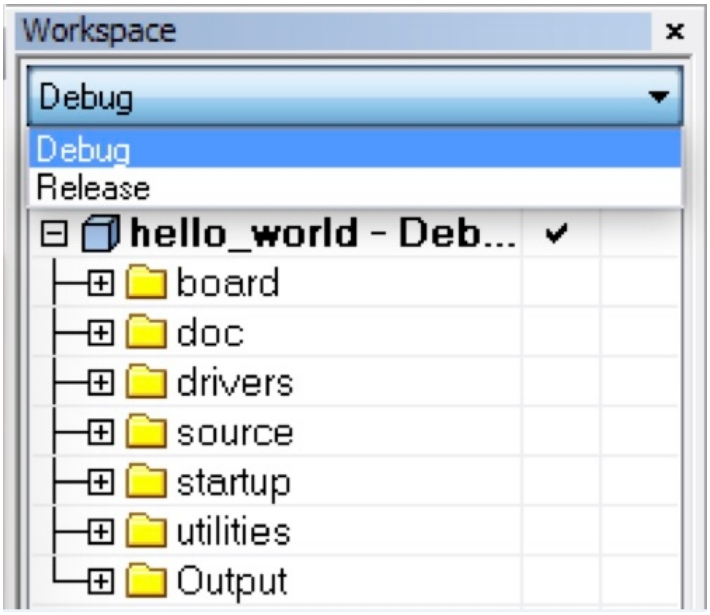


**Figure 20. Demo build target selection**

3. To build the demo application, click **Make**, highlighted in red in Figure 21.

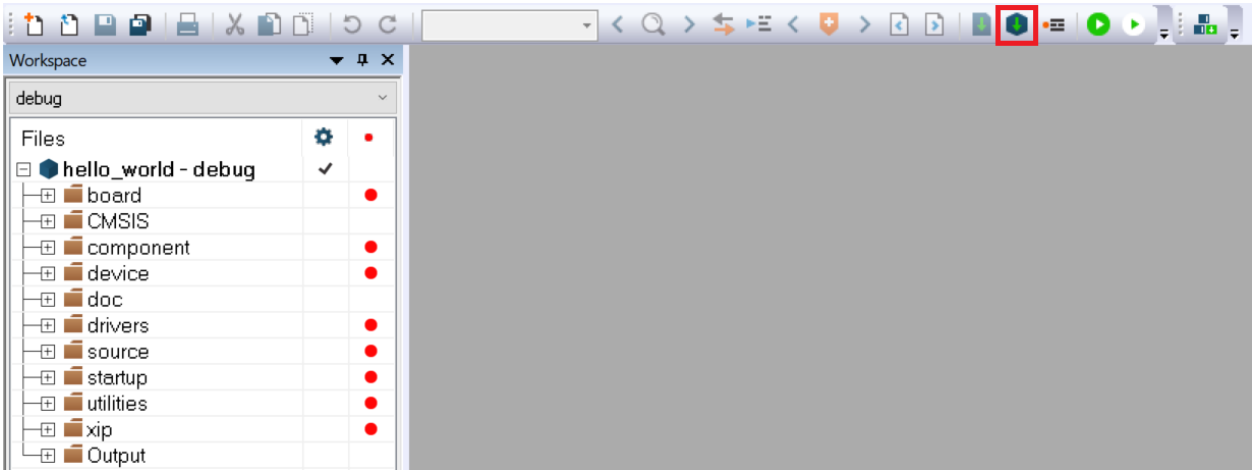


**Figure 21. Build the demo application**

4. The build completes without errors.

## 4.2 Run an example application

To download and run the application, perform these steps:

1. Connect the development platform to your PC via USB cable. Make sure the boot mode is FLEXSPI boot, where SW7 [1-3] is [**OFF, OFF, ON**] for the flash targets.
2. 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

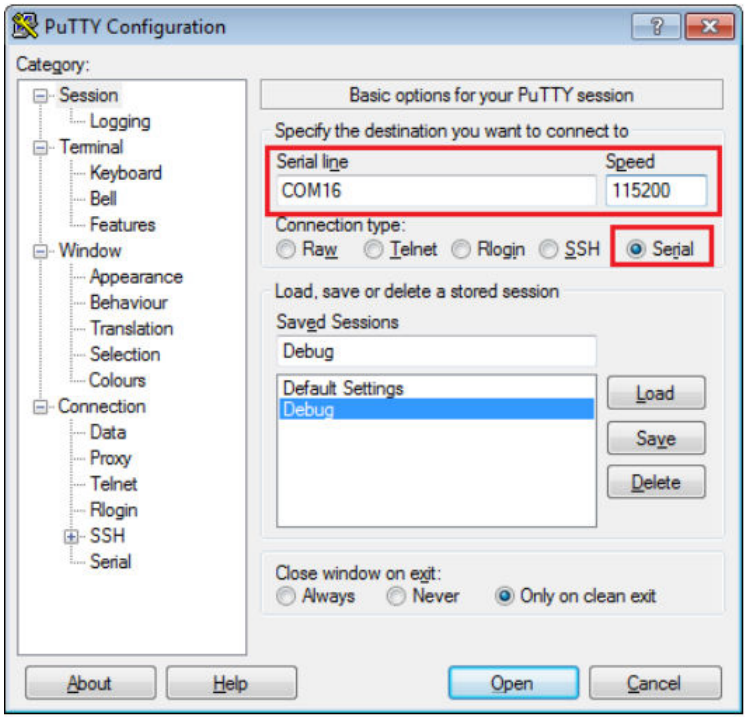


Figure 22. Terminal (PuTTY) configuration

3. In IAR, click the **Download and Debug** button to download the application to the target.

Figure 23. Download and Debug button

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

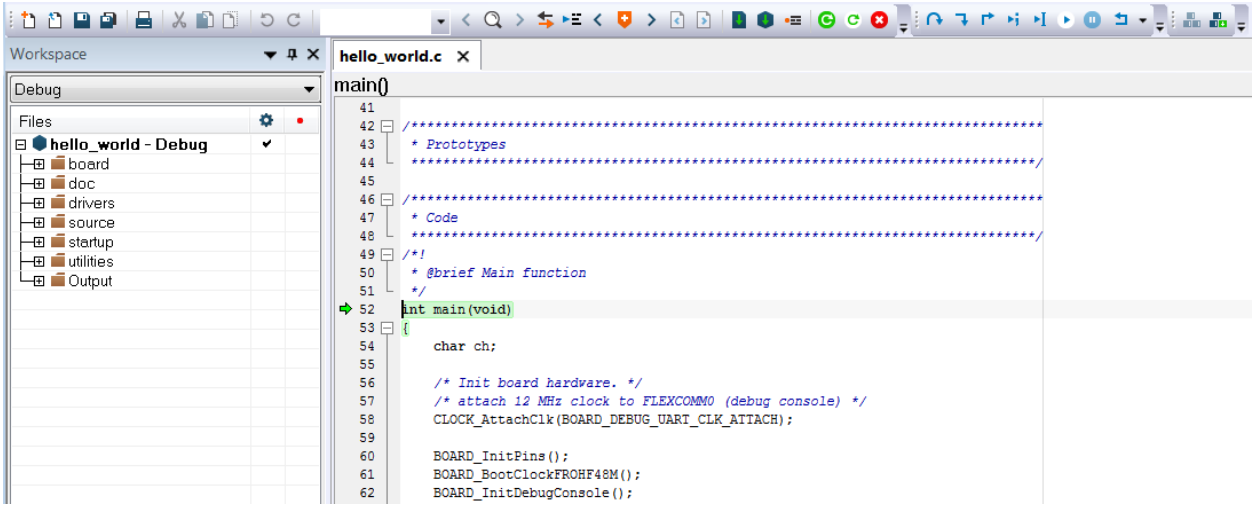


Figure 24. Stop at main() when running debugging

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

Figure 25. Go button

6. 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.

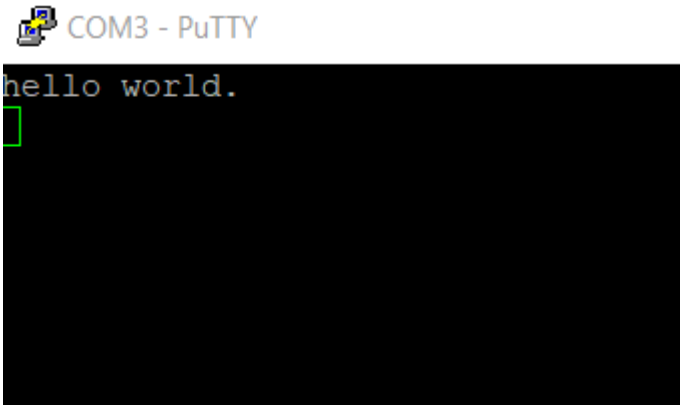


Figure 26. Text display of the hello_world demo

## 4.3 Build a TrustZone example application

This section describes the particular steps that need to be done in order to build and run a TrustZone application. The demo applications workspace files are located in this folder:

*<install_dir>/boards/<board_name>/trustzone_examples/<application_name>/[<core_type>]/iar/<application_name>_ns/iar*

*<install_dir>/boards/<board_name>/trustzone_examples/<application_name>/[<core_type>]/iar/<application_name>_s/iar*

Begin with a simple TrustZone version of the Hello World application. The TrustZone Hello World IAR workspaces are located in this folder:

*<install_dir>/boards/evkmimxrt595/trustzone_examples/hello_world/hello_world_ns/iar/hello_world_ns.eww*

*<install_dir>/boards/evkmimxrt595/trustzone_examples/hello_world/hello_world_s/iar/hello_world_s.eww*

*<install_dir>/boards/evkmimxrt595/trustzone_examples/hello_world/hello_world_s/iar/hello_world.eww*

This `hello_world.eww` project contains both secure and non-secure projects in one workspace and it allows the user to easily transition from one project to another. Build both applications separately by clicking **Make**. It is requested to build the application for the secure project first, because the non-secure project needs to know the secure project, since the CMSE library is running the linker. It is not possible to finish the non-secure project linker with the secure project since CMSE library is not ready.

## 4.4 Run a TrustZone example application

The secure project is configured to download both secure and non-secure output files, so debugging can be fully managed from the secure project. To download and run the TrustZone application, switch to the secure application project and perform Step 1 – 4 as described in Run an example application. These steps are common for single-core and TrustZone applications in IAR. After clicking **Download and Debug**, both the secure and non-secure image are loaded into the device memory, and the secure application is executed. It stops at the `Rest_Hander` function.

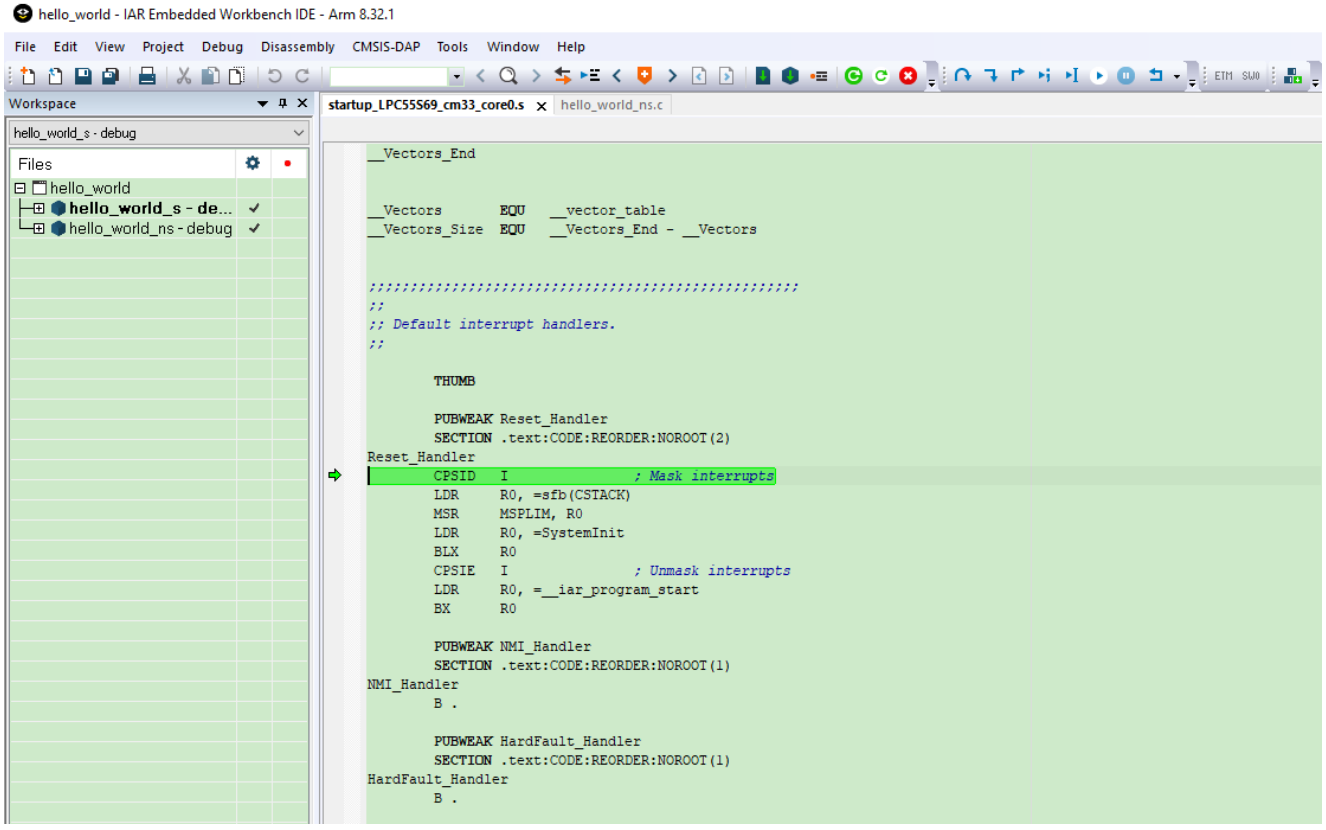


**Figure 27. Stop at `Rest_Hander` when running debugging**

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

**Figure 28. Go button**

The TrustZone `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.

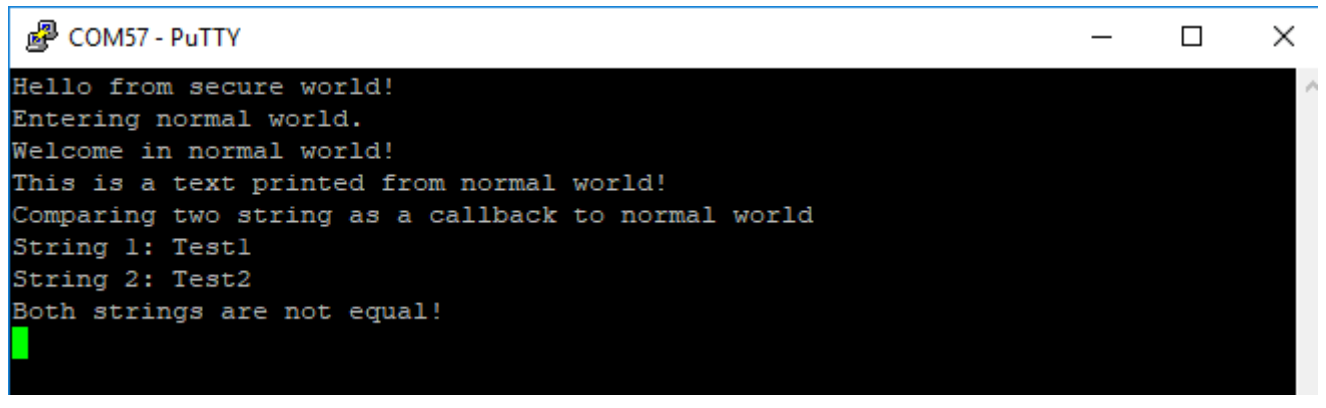


**Figure 29. Text display of the `trustzone hello_world` application**

**NOTE**

If the application is running in RAM (debug/release build target), in **Options->Debugger**, disable **Use flash loader(s)**. This can avoid the `_ns` download issue on i.MXRT500.

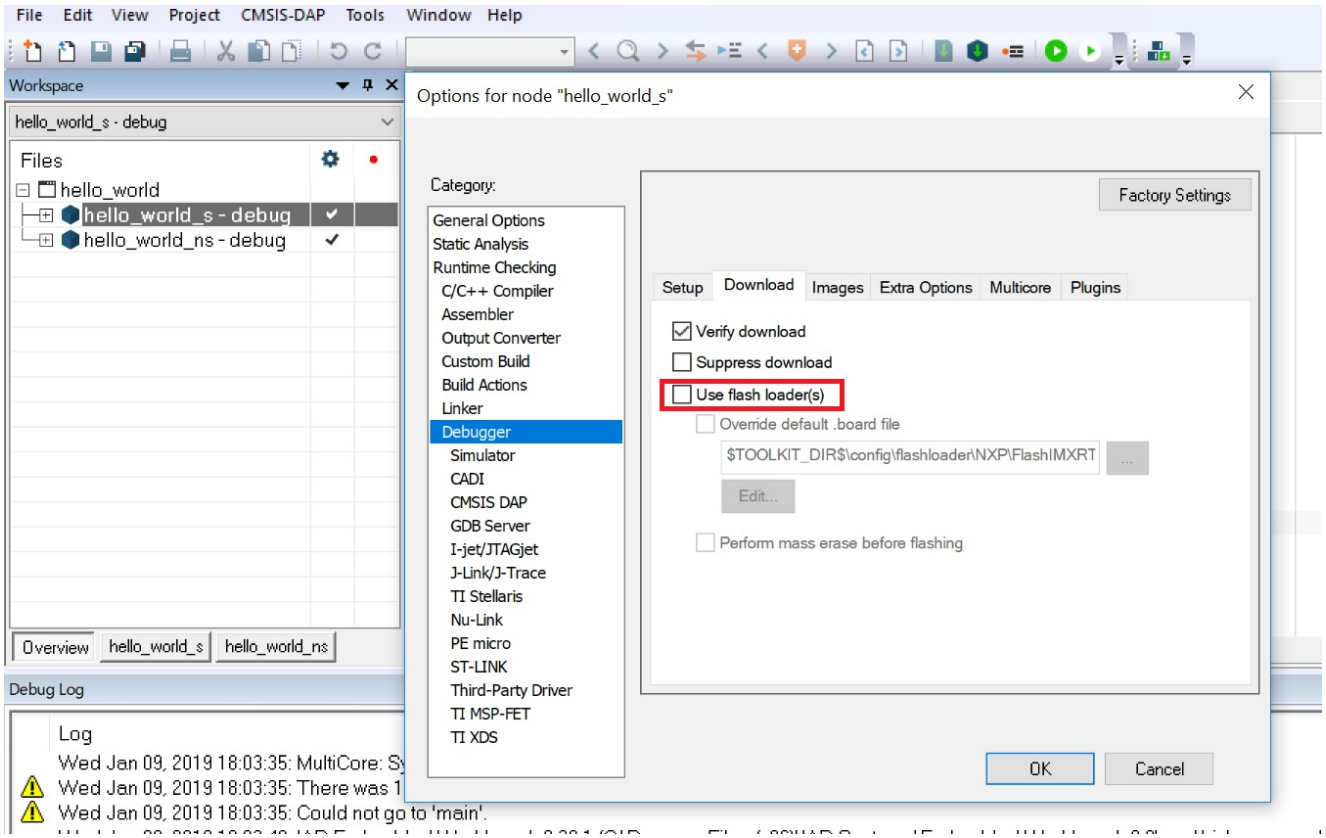


**Figure 30. Disabling Use flash loader(s)**

# Chapter 5
# 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 MIMXRT595-EVK hardware platform which is used as an example.

**NOTE**

ARMGCC version 7-2018-q2 is used as an example in this document. The latest GCC version for this package is as described in *MCUXpresso SDK Release Notes for EVK-MIMXRT595* (document MCUXSDKMIMXRT5XXRN).

## 5.1 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.

### 5.1.1 Install GCC ARM Embedded tool chain

Download and run the installer from GNU Arm Embedded Toolchain. This is the actual toolset (in other words, compiler, linker, etc.). The GCC toolchain should correspond to the latest supported version, as described in *MCUXpresso SDK Release Notes for EVK-MIMXRT595*(document MCUSDKRT5XXRN).

### 5.1.2 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 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.
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**.

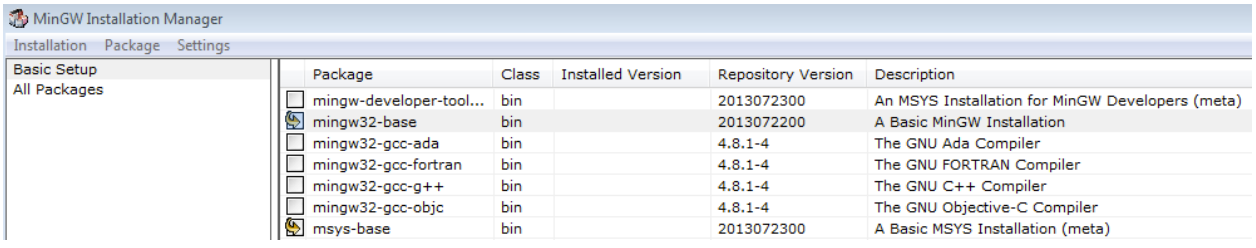


**Figure 31. Set up MinGW and MSYS**

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

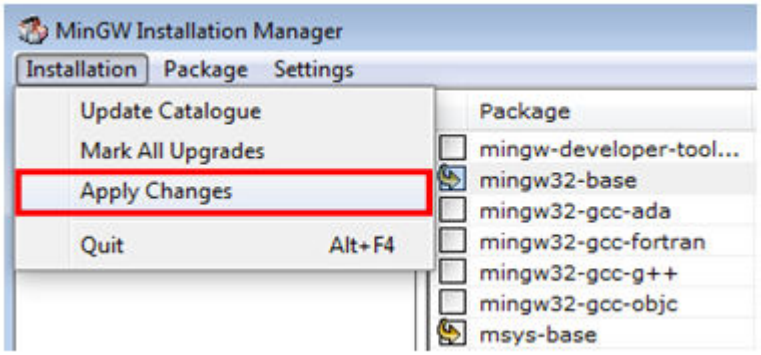


Figure 32. Complete MinGW and MSYS installation

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 is *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.

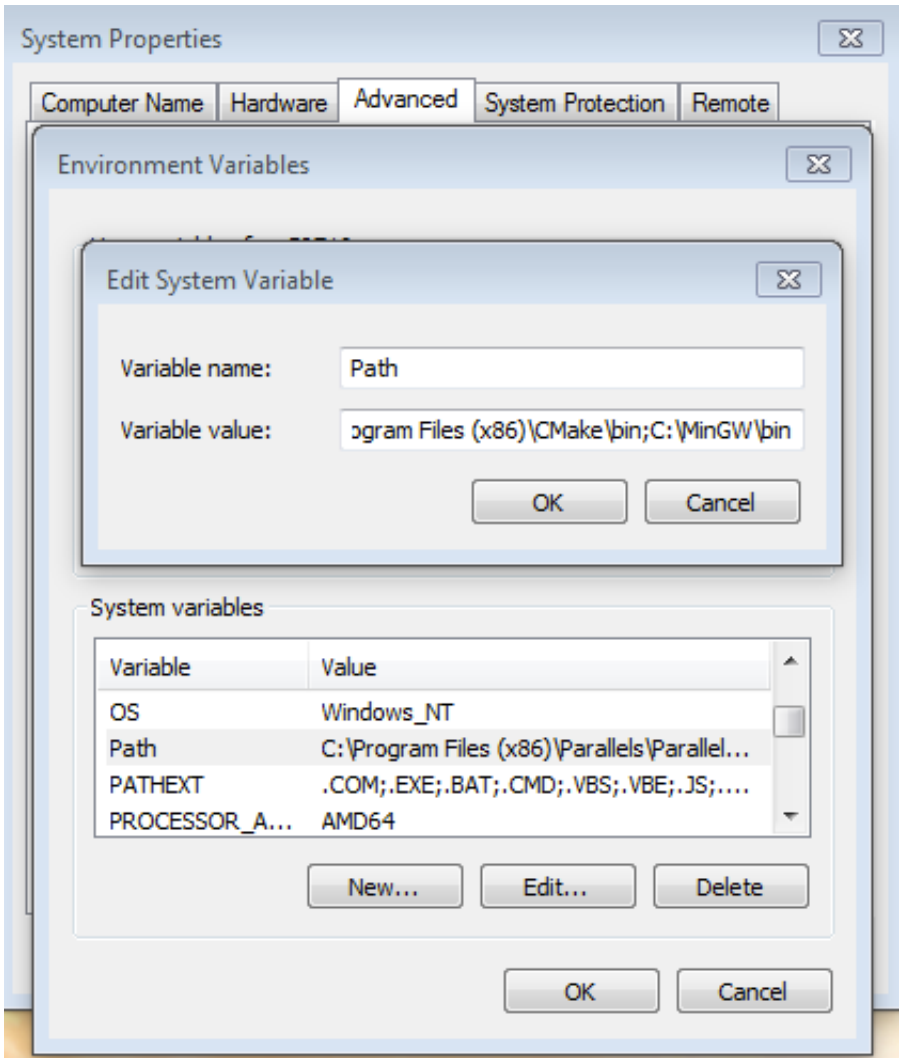


Figure 33. Add Path to systems environment

### 5.1.3 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 the `for %I in (.) do echo %~sI` command 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 34. Convert path to short path

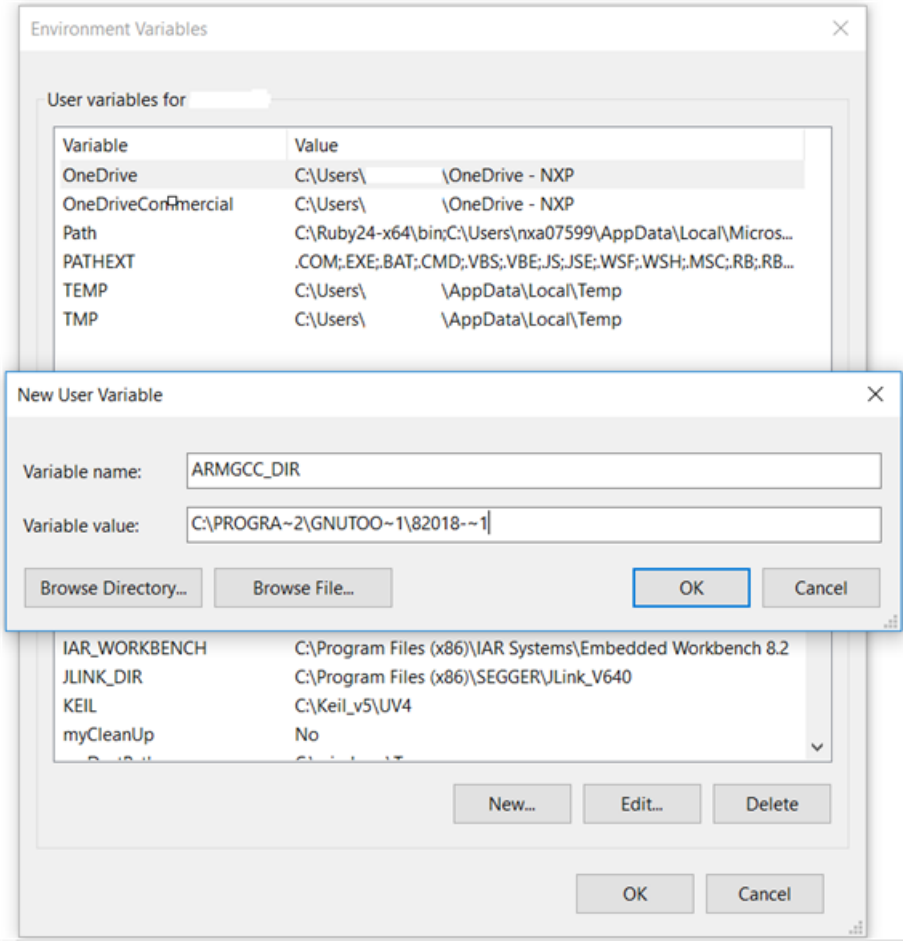


Figure 35. Add ARMGCC_DIR system variable

### 5.1.4 Install CMake

1. Download CMake 3.0.x from CMAKE.
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.

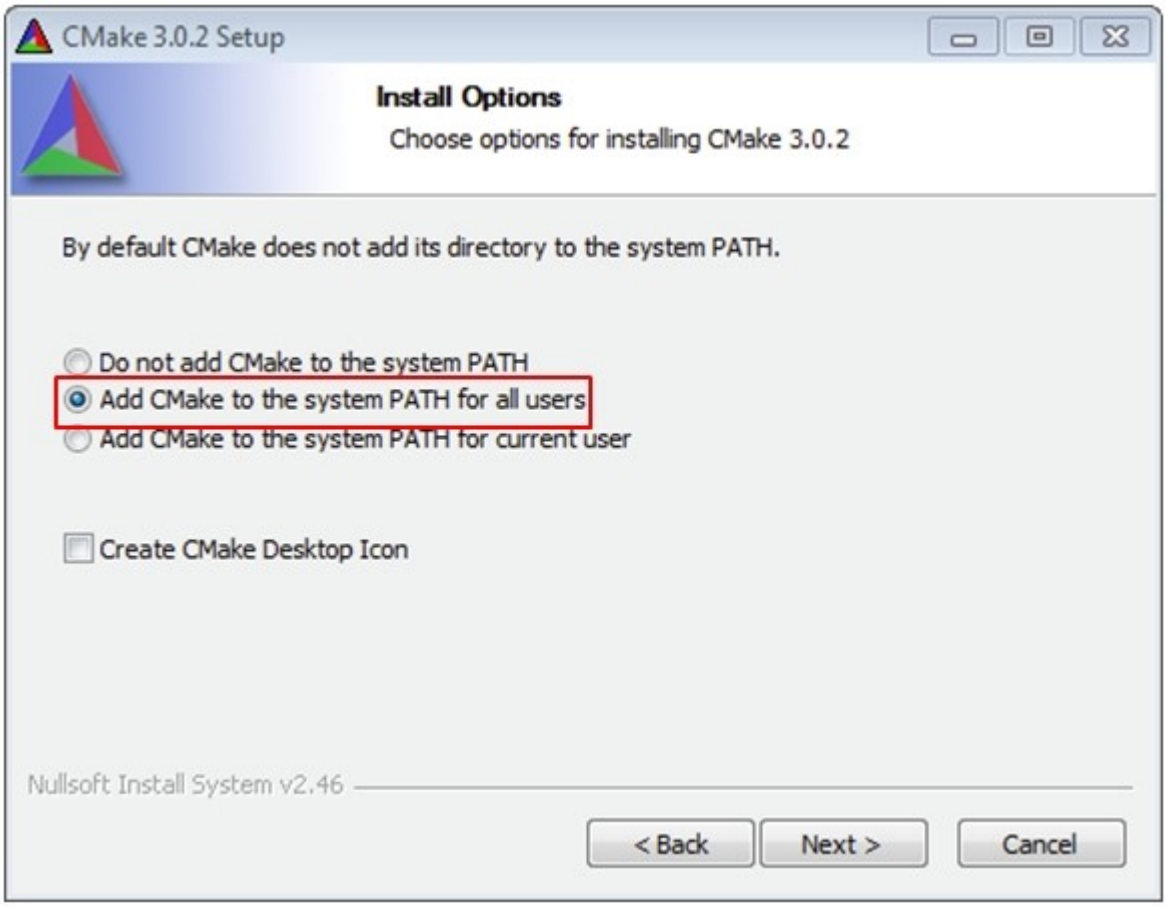


Figure 36. Install CMake

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`.

## 5.2 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**.

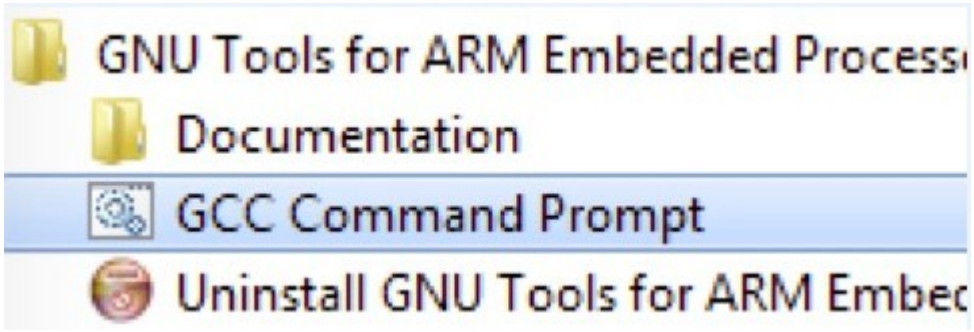


Figure 37. Launch command prompt

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/evkmimxrt595/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 **build_debug.bat** file in Windows Explorer to build it. The output is as shown in Figure 38.

```
[ 80%] Building C object CMakeFiles/hello_world.elf.dir/C_/nxp/SDK_2.6.0_EVK-MIMXRT595/components/lists/generic_list.c.o
bj
[ 84%] Building ASM object CMakeFiles/hello_world.elf.dir/C_/nxp/SDK_2.6.0_EVK-MIMXRT595/devices/MIMXRT595S/gcc/startup_
MIMXRT595S_cm33.S.obj
[ 88%] Building C object CMakeFiles/hello_world.elf.dir/C_/nxp/SDK_2.6.0_EVK-MIMXRT595/devices/MIMXRT595S/drivers/fsl_fl
excomm.c.obj
[ 92%] Building C object CMakeFiles/hello_world.elf.dir/C_/nxp/SDK_2.6.0_EVK-MIMXRT595/devices/MIMXRT595S/drivers/fsl_gp
io.c.obj
[ 96%] Building C object CMakeFiles/hello_world.elf.dir/C_/nxp/SDK_2.6.0_EVK-MIMXRT595/devices/MIMXRT595S/drivers/fsl_ia
p.c.obj
[100%] Linking C executable debug\hello_world.elf
[100%] Built target hello_world.elf

C:\nxp\SDK_2.6.0_EVK-MIMXRT595\boards\evkmimxrt595\demo_apps\hello_world\armgcc>IF "" == "" (pause )
Press any key to continue . . .
```

Figure 38. hello_world demo build successful

## 5.3 Run an example application

This section describes steps to run a demo application using J-Link GDB Server application. To perform this exercise, 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 Table 1. 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 and the PC USB connector. If using a standalone J-Link debug pod, also connect it to the SWD/JTAG connector of the board. Make sure the boot mode is FLEXSPI boot, where SW7 [1-3] is [**OFF, OFF, ON**].

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

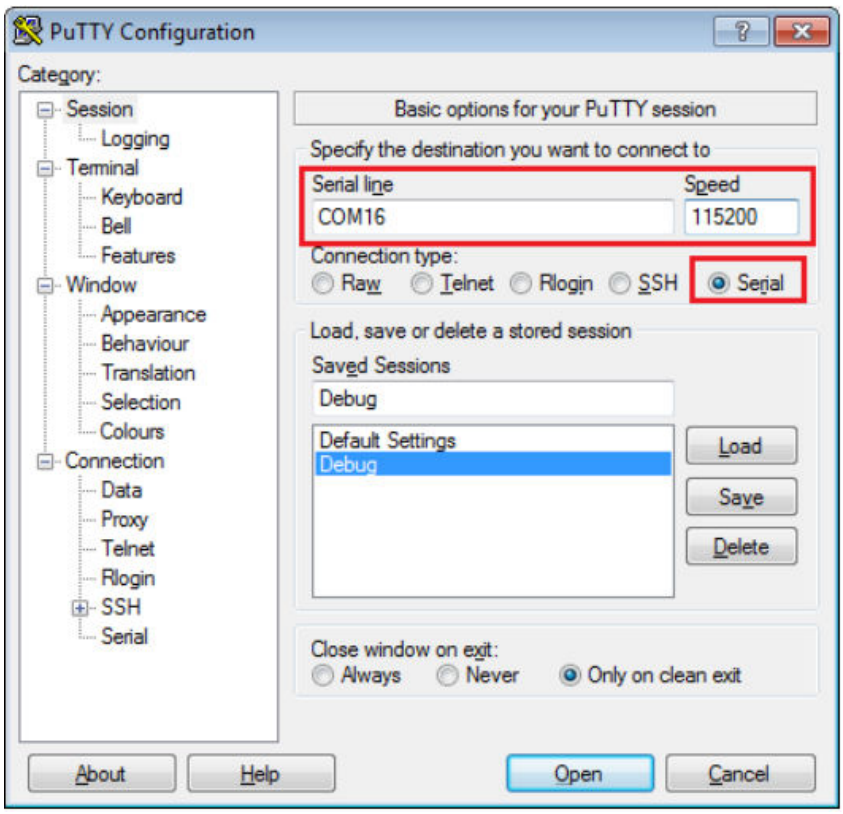


**Figure 39. Terminal (PuTTY) configurations**

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 in Figure 40. The target device selection chosen for this example is **MIMXRT595_M33**.

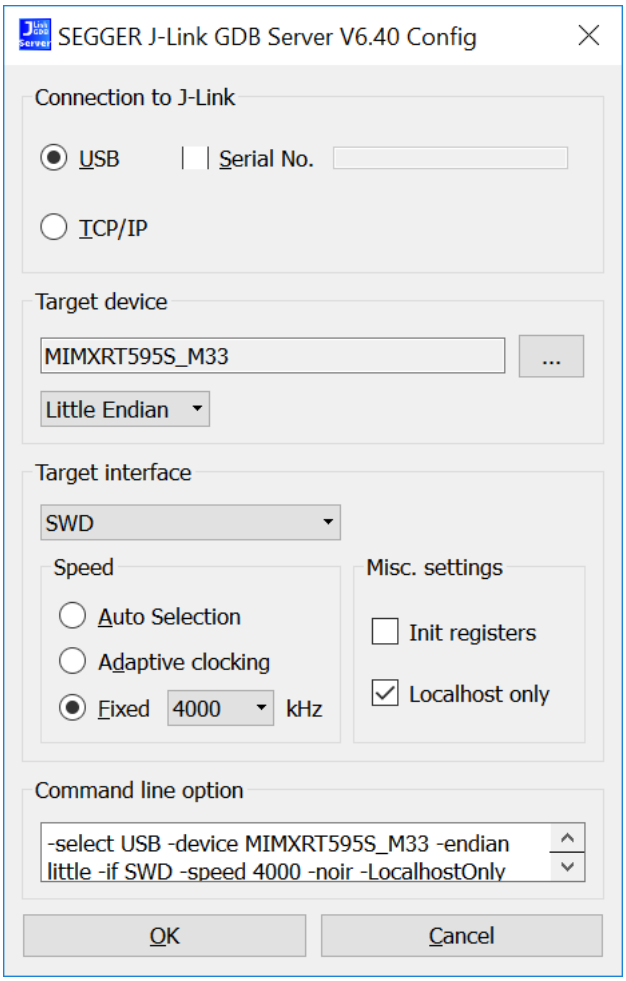


**Figure 40. SEGGER J-Link GDB Server configuration**

5. After it is connected, the screen should be as shown in Figure 41.

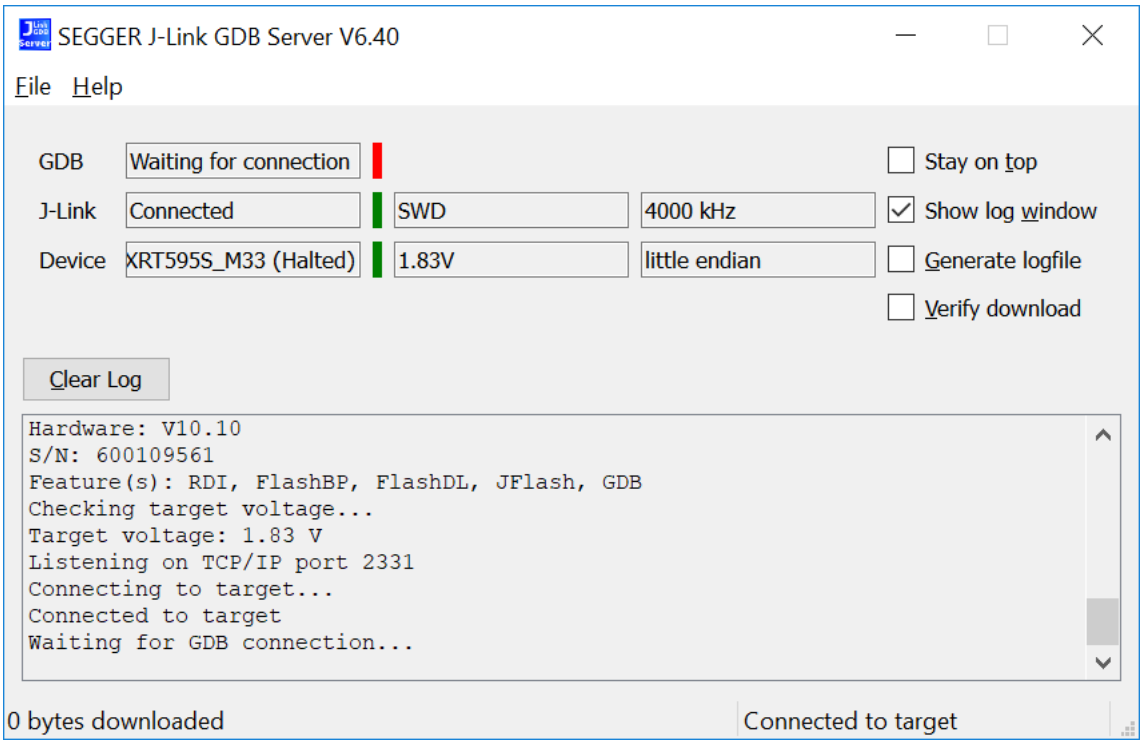


**Figure 41. SEGGER J-Link GDB Server screen after successful connection**

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

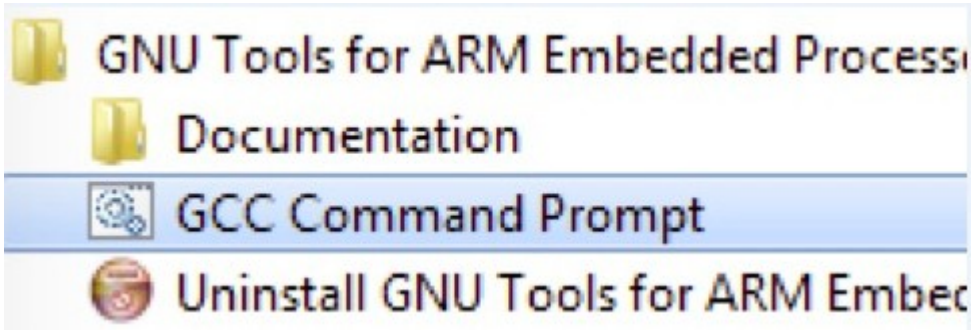


**Figure 42. Launch command prompt**

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/evkmimxrt595/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`.

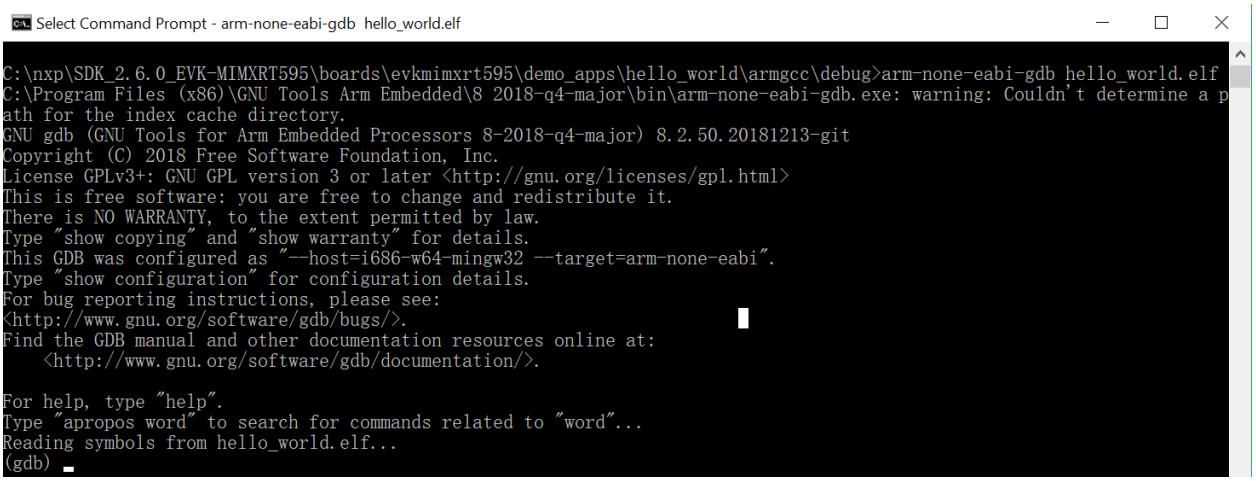


**Figure 43. Run `arm-none-eabi-gdb`**

**NOTE**

Make sure the board is set to FlexSPI flash boot mode before debugging.

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. Execute the `c` 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.

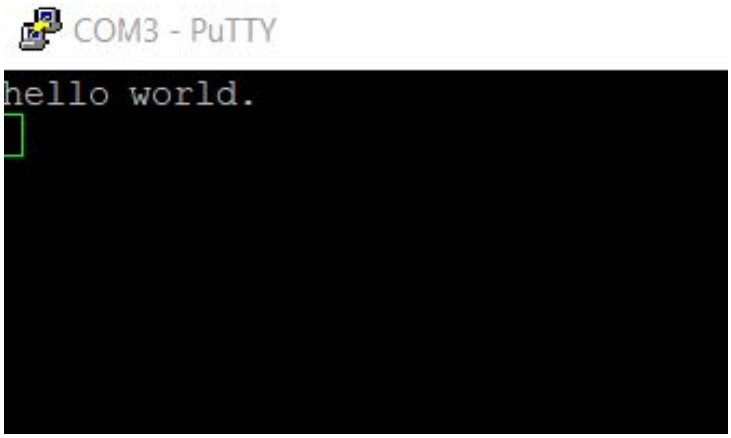


**Figure 44. Text display of the `hello_world` demo**

## 5.4 Build a TrustZone example application

This section describes the steps to build and run a TrustZone application. The demo application build scripts are located in this folder:

*<install_dir>/boards/<board_name>/trustzone_examples/<application_name>/[<core_type>]/<application_name>_ns/armgcc*

*<install_dir>/boards/<board_name>/trustzone_examples/<application_name>/[<core_type>]/<application_name>_s/armgcc*

Begin with a simple TrustZone version of the Hello World application. The TrustZone Hello World GCC build scripts are located in this folder:

*<install_dir>/boards/evkmimxrt595/trustzone_examples/hello_world/hello_world_ns/armgcc/build_debug.bat*

*<install_dir>/boards/evkmimxrt595/trustzone_examples/hello_world/hello_world_s/armgcc/build_debug.bat*

Build both applications separately, by following steps for single core examples as described in Build an example application. It is requested to build the application for the secure project first, because the non-secure project needs to know the secure project, since CMSE library is running the linker. It is not possible to finish the non-secure project linker with the secure project because the CMSE library is not ready.

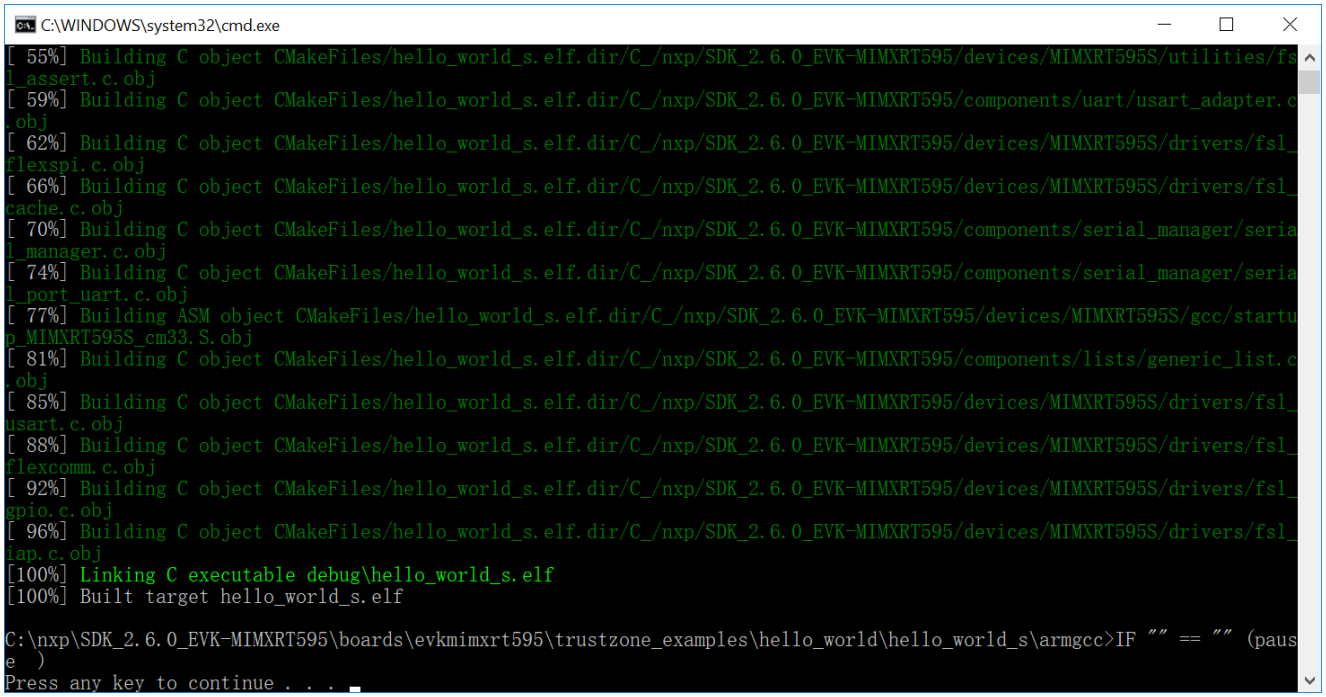


Figure 45. `hello_world_s` example build successful

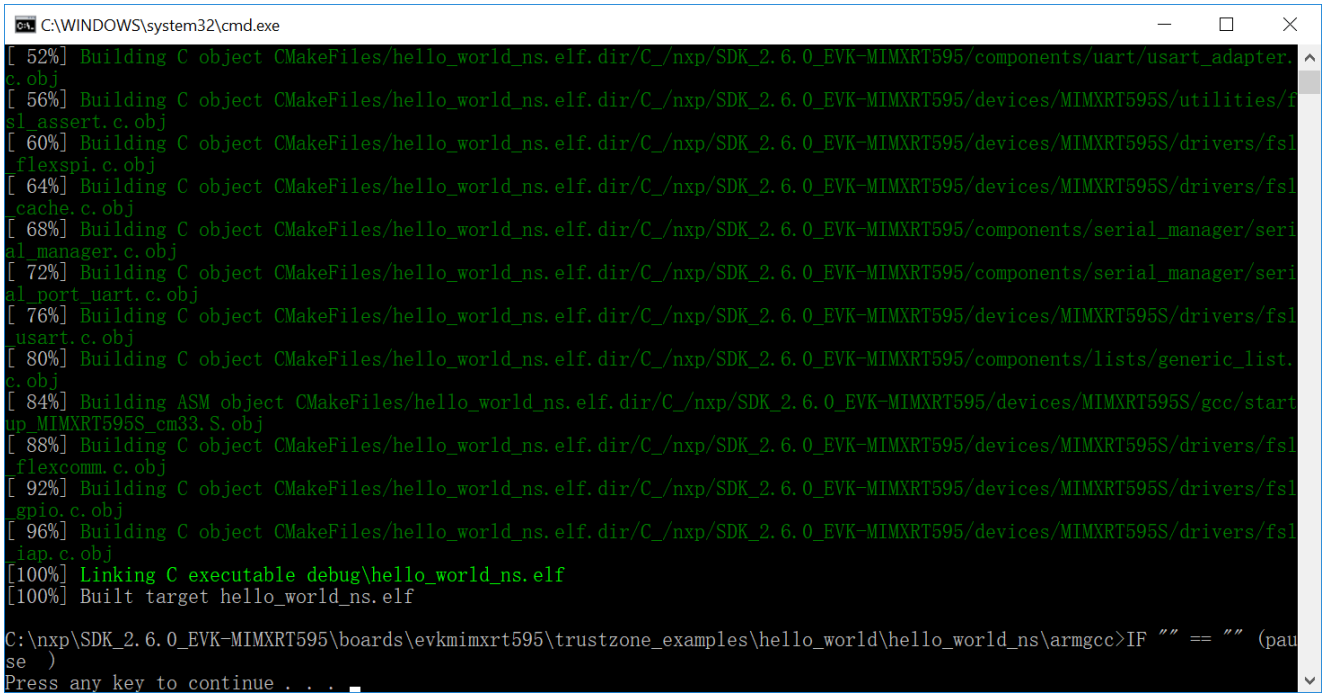


**Figure 46. `hello_world_ns` example build successful**

## 5.5 Run a TrustZone example application

When running a TrustZone application, the same prerequisites for J-Link/J-Link OpenSDA firmware, and the serial console as for the single core application, apply, as described in Run an example application.

To download and run the TrustZone application, perform steps 1 to 10, as described in Run an example application. These steps are common for both single core and trustzone applications in Arm GCC.

Then, run these commands:

1. `arm-none-eabi-gdb.exe`
2. `target remote localhost:2331`
3. `monitor reset`
4. `monitor halt`
5. `monitor exec SetFlashDLNoRMWThreshold = 0x20000`
6. `load <install_dir>/boards/evkmimxrt595/trustzone_examples/hello_world/hello_world_ns/armgcc/debug/hello_world_ns.elf`
7. `load <install_dir>/boards/evkmimxrt595/trustzone_examples/hello_world/hello_world_s/armgcc/debug/hello_world_s.elf`
8. `monitor reset`

The application is now downloaded and halted. Execute the `c` command to start the demo application.

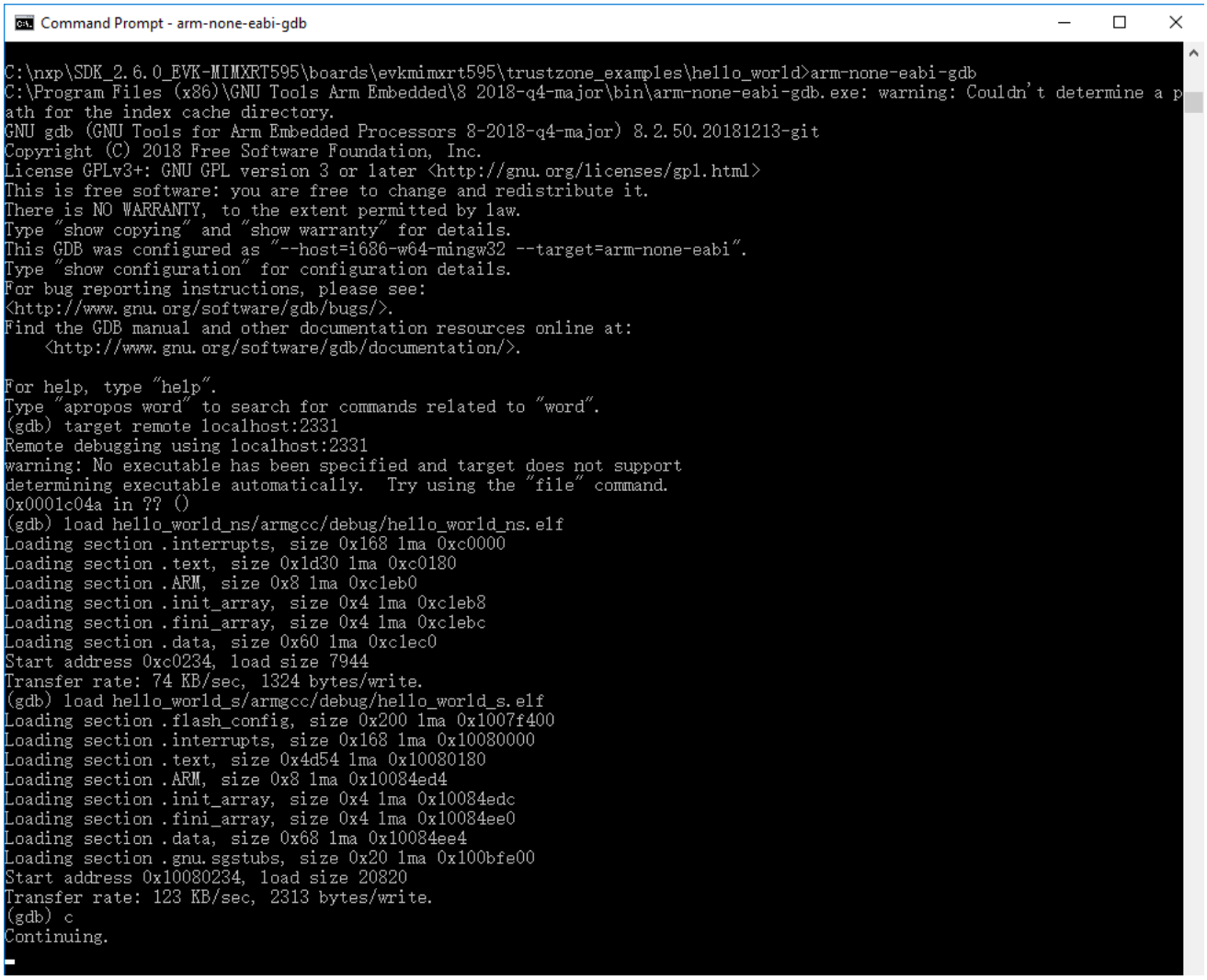


Figure 47. Loading and running the trustzone example

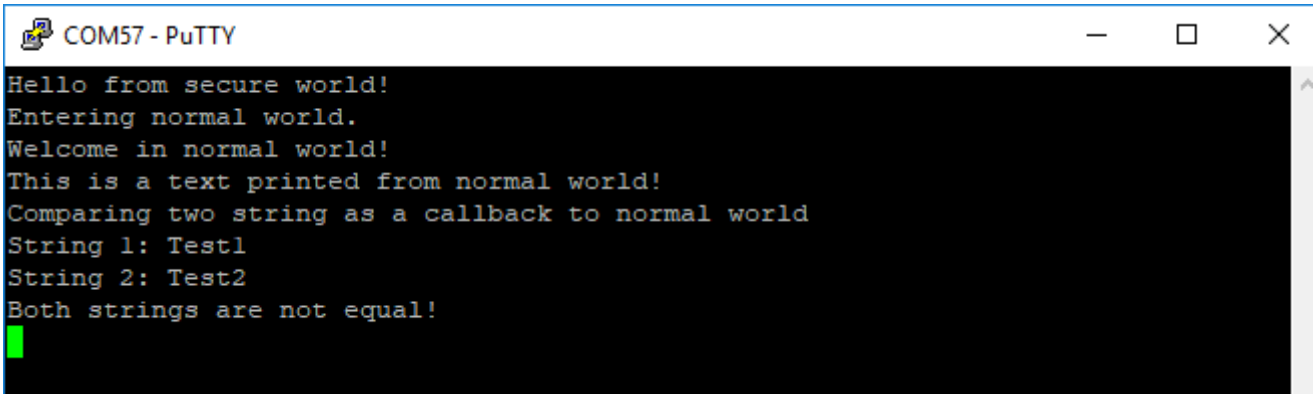


Figure 48. Text display of the trustzone hello_world application

# Chapter 6
# 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.

## 6.1 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 MIMXRT595S CMSIS pack.

1. Open the MDK IDE, which is called µVision. In the IDE, select the Pack Installer icon.
2. After the installation finishes, close the Pack Installer window and return to the µVision IDE.

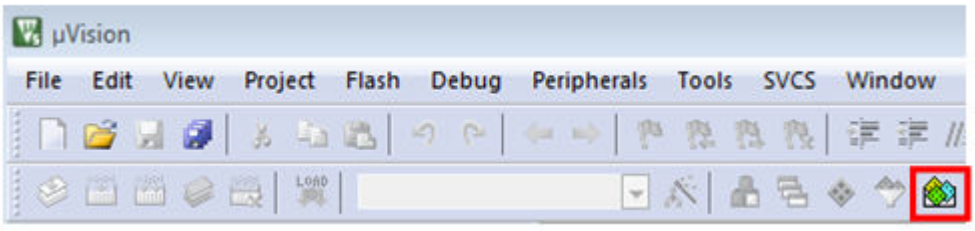


**Figure 49. Launch the pack installer**

## 6.2 Build an example application

1. Open the desired example application workspace in:

   *<install_dir>/boards/<board_name>/<example_type>/<application_name>/mdk*

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

   *<install_dir>/boards/evkmimxrt595s/demo_apps/hello_world/mdk/hello_world.uvmpw*

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


**Figure 50. Build the demo**

3. The build completes without errors.

## 6.3 Run an example application

To download and run the application, perform these steps:

1. Reference Table 1 to determine the debug interface that comes loaded on your specific hardware platform.
2. Connect the development platform to your PC via USB cable. Make sure the boot mode is FLEXSPI boot, where SW7 [1-3] is [**OFF, OFF, ON**].
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:

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

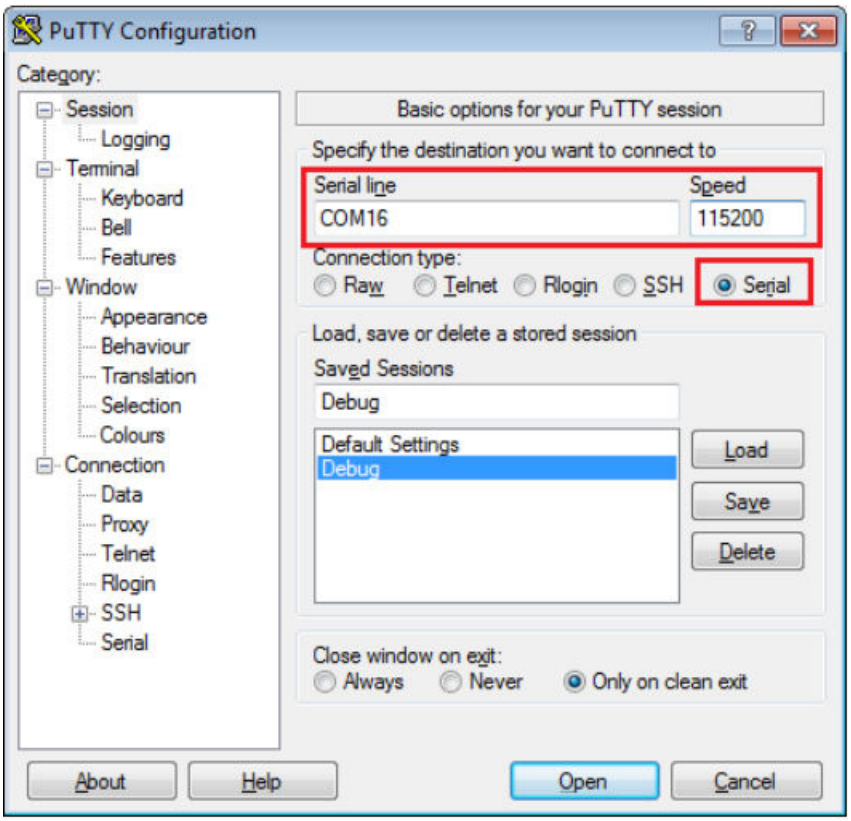


**Figure 51. Terminal (PuTTY) configurations**

4. To debug the application, click **load** (or press the F8 key). Then, click the **Start/Stop Debug Session** button, highlighted in red in Figure 52. If using **J-Link** as the debugger, click **Project option >Debug >Settings >Debug >Port**, and select **SW**.

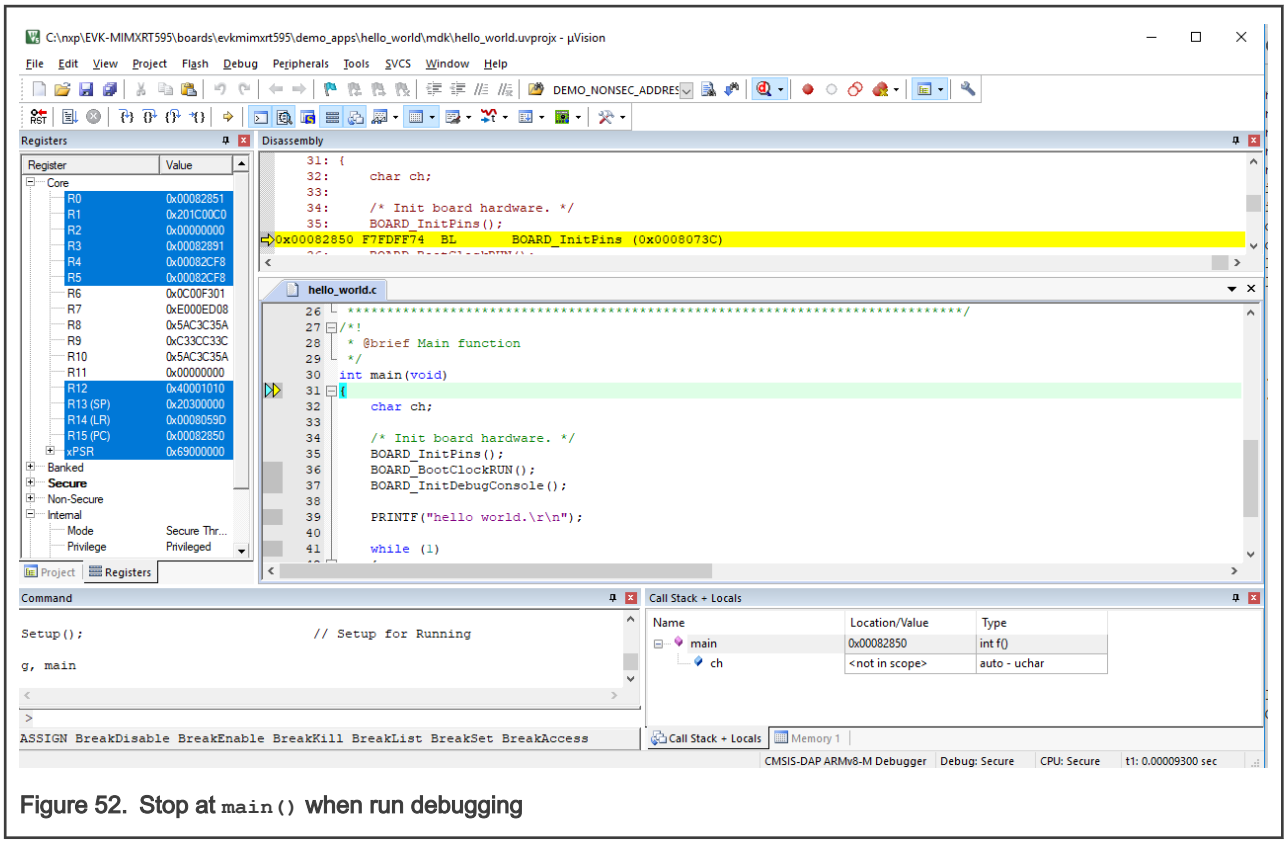


**Figure 52. Stop at `main()` when run debugging**

**NOTE**

Make sure the board is set to FlexSPI flash boot mode before debugging.

5. Run the code by clicking **Run** to start the application, as shown in Figure 53.

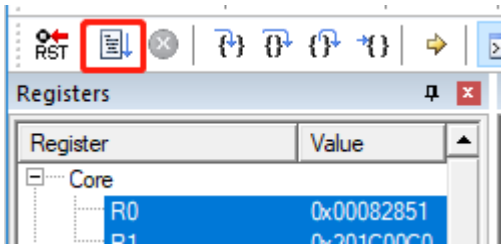


**Figure 53. Run button**

The `hello_world` application is now running and a banner is displayed on the terminal, as shown in Figure 54. If this is not true, check your terminal settings and connections.

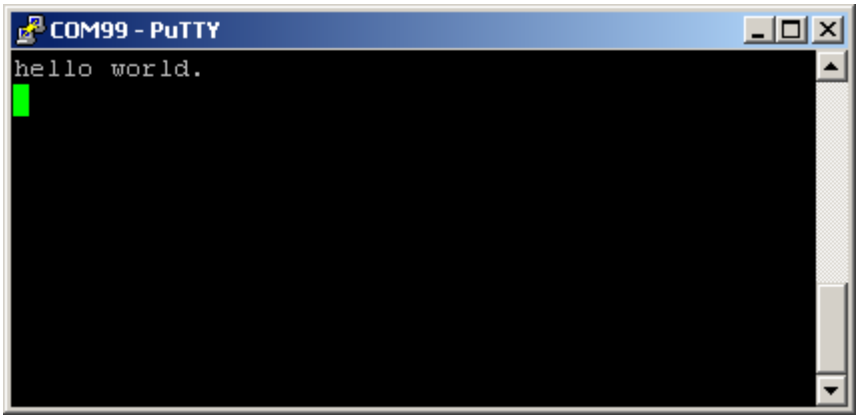


**Figure 54. Text display of the `hello_world` demo**

## 6.4 Build a TrustZone example application

This section describes the particular steps that need to be done in order to build and run a TrustZone application. The demo applications workspace files are located in this folder:

*<install_dir>/boards/<board_name>/trustzone_examples/<application_name>/<application_name>_ns/mdk*

*<install_dir>/boards/<board_name>/trustzone_examples/<application_name>/<application_name>_s/mdk*

Begin with a simple TrustZone version of the Hello World application. The TrustZone Hello World Keil MSDK/µVision ® workspaces are located in this folder:

*<install_dir>/boards/evkmimxrt595/trustzone_examples/hello_world/hello_world_ns/mdk/hello_world_ns.uvmpw*

*<install_dir>/boards/evkmimxrt595/trustzone_examples/hello_world/hello_world_s/mdk/hello_world.uvmpw*

This project `hello_world.uvmpw` contains both secure and non-secure projects in one workspace and it allows the user to easily transition from one project to another.

Build both applications separately by clicking **Rebuild**. It is requested to build the application for the secure project first, because the non-secure project needs to know the secure project since CMSE library is running the linker. It is not possible to finish the non-secure project linker with the secure project because CMSE library is not ready.

## 6.5 Run a TrustZone example application

The secure project is configured to download both secure and non-secure output files so debugging can be fully managed from the secure project.

To download and run the TrustZone application, switch to the secure application project and perform steps as described in Run an example application. These steps are common for single core, dual-core, and TrustZone applications in µVision. After clicking **Download and Debug**, both the secure and non-secure image are loaded into the device flash memory, and the secure application is executed. It stops at the `main()` function.

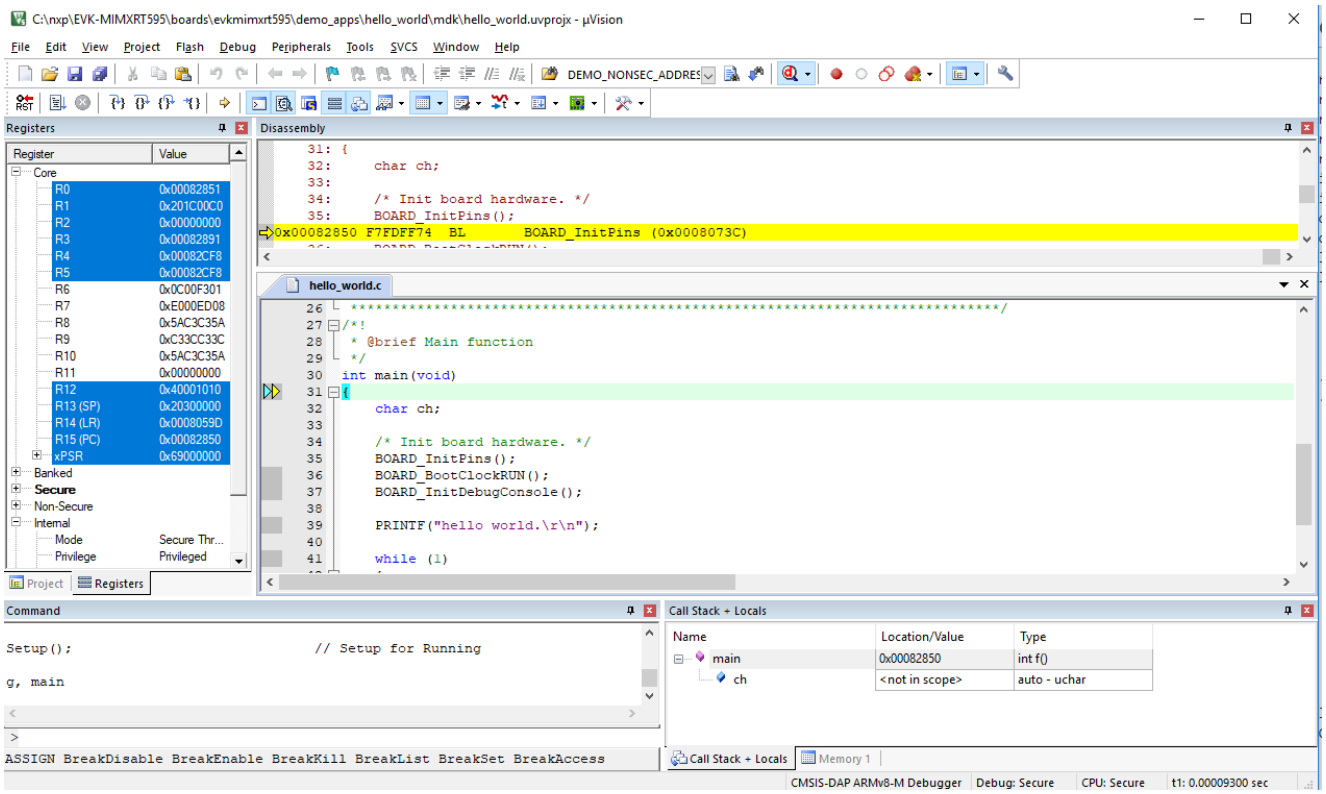


**Figure 55. Stop at `main()` when running debugging**

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

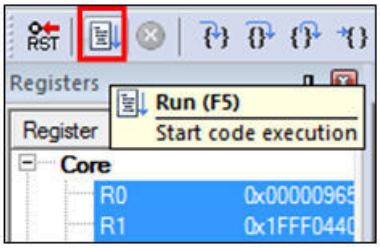


**Figure 56. Run button**

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.

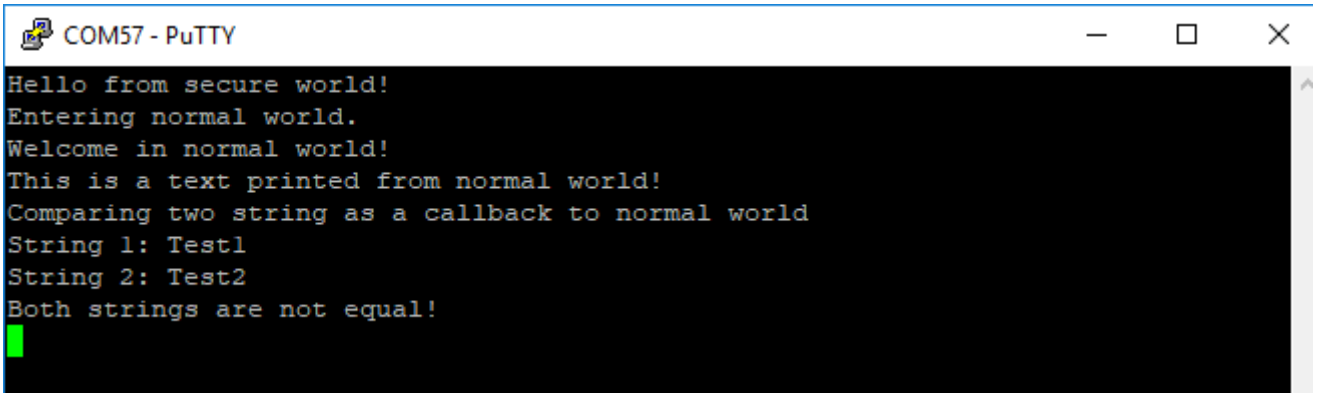


Figure 57. Text display of the trustzone `hello_world` application

# Chapter 7
# 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 58.


Figure 58. 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.

- **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
```

- **Windows**

  To determine the COM port, open **Device Manager** in the Windows operating system.

  1. Click on the **Start** menu and type **Device Manager** in the search bar.

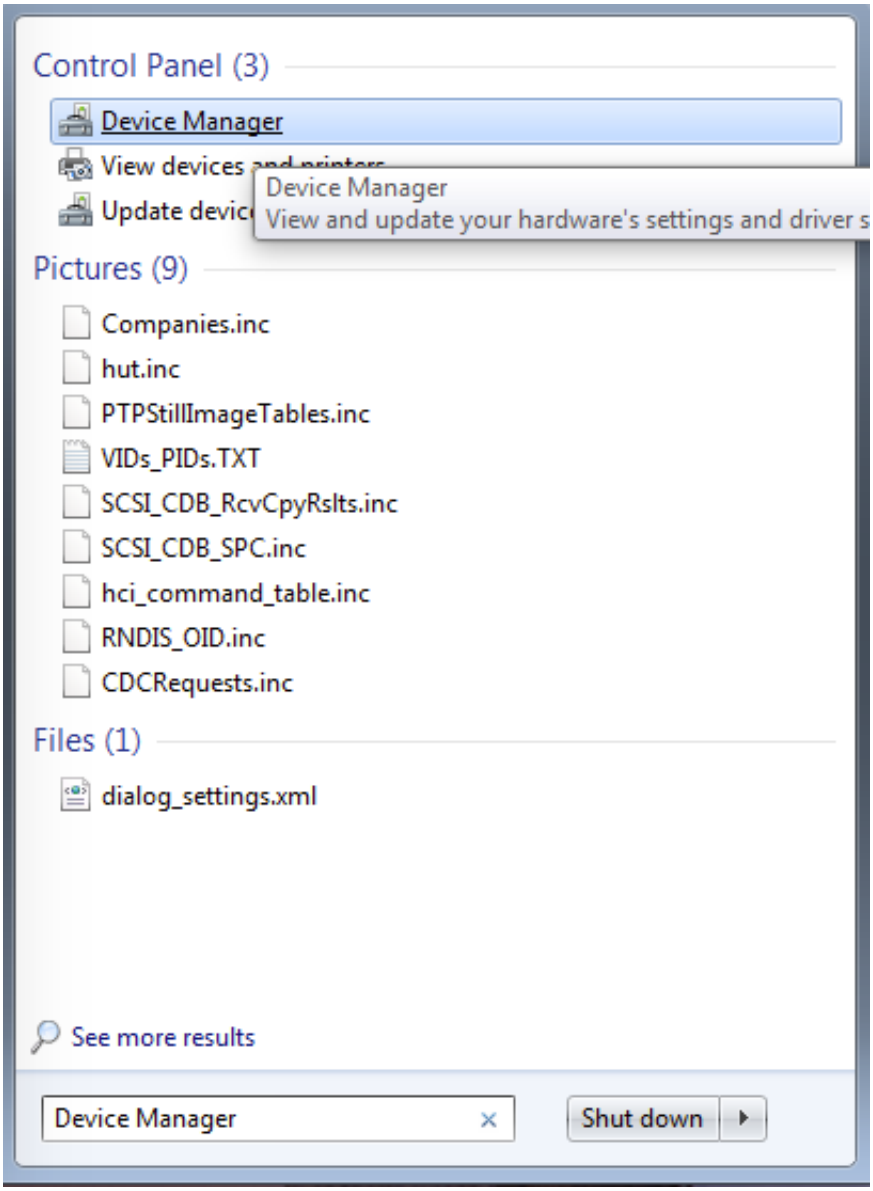


Figure 59. Device manager

2. 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

# 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 1lists the hardware platforms supported by the MCUXpresso SDK, their default debug interface, and any version information that helps differentiate a specific interface configuration.

**Table 1. Hardware platforms supported by SDK**

| Hardware platform | Default interface | OpenSDA details[1] |
|---|---|---|
| EVK-MC56F83000 | P&E Micro OSJTAG | N/A |
| EVK-MIMXRT595 | CMSIS-DAP | N/A |
| EVK-MIMXRT685 | CMSIS-DAP | N/A |
| FRDM-K22F | CMSIS-DAP/mbed/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/mbed/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/mbed/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/mbed/DAPLink | OpenSDA v2.1 |

*Table continues on the next page...*

Table 1. Hardware platforms supported by SDK (continued)

| Hardware platform | Default interface | OpenSDA details[1] |
|---|---|---|
| FRDM-KW36 | DAPLink | OpenSDA v2.2 |
| FRDM-KW41Z | CMSIS-DAP/DAPLink | OpenSDA v2.1 or greater |
| Hexiwear | CMSIS-DAP/mbed/DAPLink | OpenSDA v2.0 |
| HVP-KE18F | DAPLink | OpenSDA v2.2 |
| HVP-KV46F150M | P&E Micro OpenSDA | OpenSDA v1 |
| 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/mbed | 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 |

*Table continues on the next page...*

Table 1. Hardware platforms supported by SDK (continued)

| Hardware platform | Default interface | OpenSDA details[1] |
|---|---|---|
| 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 |
| 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 |

1. The OpenSDA details is not applicable to LPC.

# Appendix C
# Updating debugger firmware

## C.1 Updating EVK board firmware

The EVK hardware platform comes with a CMSIS-DAP-compatible debug interface (known as LPC-Link2). This firmware in this debug interface may be updated using the host computer utility called LPCScrypt. This typically used when switching between the default debugger protocol (CMSIS-DAP) to SEGGER J-Link, or for updating this firmware with new releases of these.

This section contains the steps to re-program the debug probe firmware.

**NOTE**

If MCUXpresso IDE is used and the jumper making DFUlink is installed on the board (JP1 on some boards, but consult the board user manual or schematic for specific jumper number), LPC-Link2 debug probe boots to DFU mode, and MCUXpresso IDE automatically downloads the CMSIS-DAP firmware to the probe before flash memory programming (after clicking **Debug**). Using DFU mode ensures most up-to-date/compatible firmware is used with MCUXpresso IDE.

NXP provides the LPCScrypt utility, which is the recommended tool for programming the latest versions of CMSIS-DAP and J-Link firmware onto LPC-Link2 or LPCXpresso boards. The utility can be downloaded from LPC Microcontroller Utilities.

These steps show how to update the debugger firmware on your board for Windows operating system. For Linux OS, follow the instructions described in LPCScrypt user guide (on LPC Microcontroller Utilities, select **LPCScrypt** and then select the **Documentation** tab).

1. Install the LPCScript utility And Updated firmware image from Segger.
2. Unplug the board's USB cable.
3. Make the DFU link (install the jumper labelled DFUlink).
4. Connect the probe to the host via USB (use Link USB connector).
5. Open a command shell and call the appropriate script located in the LPCScrypt installation directory, *<LPCScrypt install dir>*.
    a. To program CMSIS-DAP debug firmware: *<LPCScrypt install dir>/scripts/program_CMSIS*.
    b. To program J-Link debug firmware: *<LPCScrypt install dir>/scripts/program_JLINK*.
6. Remove DFU link (remove the jumper installed in Step 3).
7. Re-power the board by removing the USB cable and plugging it in again.

*How To Reach Us*

**Home Page:**

nxp.com

**Web Support:**

nxp.com/support



For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

**Date of release: January 21, 2021**

**Document identifier: MCUXSDKMIMXRT5XXGSUG**

arm