MCUXSDKMIMXRT6XXGSUG

Getting Started with MCUXpresso SDK for EVK-MIMXRT685

Rev. 2.12.0 — 30 June 2022 User guide

Document information

Information	Content
Keywords	MCUXpresso SDK, Getting Started, EVK-MIMXRT685, RT6XX
Abstract	This document describes the steps to get started with MCUXpresso SDK for EVK-MIMXRT685.

1 Overview

The NXP MCUXpresso software and tools offer comprehensive development solutions designed to optimize, ease and help accelerate embedded system development of applications based on general purpose, crossover and Bluetooth™-enabled MCUs from NXP. 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 optional RTOS integrations such as FreeRTOS and Azure RTOS, a USB host and device stack, and various other middleware to support rapid development.

For supported toolchain versions, see *MCUXpresso SDK Release Notes for EVK-MIMXRT685* (document MCUXSDKMIMXRT6XXRN).

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

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, i.mxrt EVK boards, 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

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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

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:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

- `devices/<device_name>`: The device's CMSIS header file, MCUXpresso SDK feature file and a few other files
- `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>/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.

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 MIMXRT685-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)…**.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 4. Import an SDK example

3. In the window that appears, expand the **MIMXRT600** folder and select **MIMXRT685S** . Then, select **evkmimxrt685** and click **Next**.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 5. Select MIMXRT685-EVK board

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

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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**.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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 community.nxp.com.

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

1. See the table in Section 11 to determine the debug interface that comes loaded on your specific hardware platform.
   - For boards with CMSIS-DAP/mbed/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 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.
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 Section 9). 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

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

c. 8 data bits
d. 1 stop bit

Figure 8. Terminal (PuTTY) configurations

4. On the **Quickstart Panel**, click on **Debug** `evkmimxrt685_hello_world [Debug]` to launch the debug session.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 10. Attached Probes: debug emulator selection

Note: *If the debug probe is CMSIS-DAP and the debugging application is running in flash, make sure the board is set to FlexSPI flash boot mode (ISP2: ISP1: ISP0 = ON, OFF, ON). Otherwise, you should not set to FlexSPI boot mode when debugging the application. If the debug probe is J-Link,* ***Reset before running*** *must be disabled. Double click the* `<example> J-Link Debug.launch` *file, and deselect this option under* ***JLink Debugger ->Additional Options*** *menu.*

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 11. Disabling Reset before running

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

Figure 12. Stop at `main()` when running debugging

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

Figure 13. 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.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 14. 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 MIMXRT685-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 MIMXRT685-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 **MIMXRT600** folder and select **MIMXRT685S**. Then, select **evkmimxrt685** and click **Next**.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 15. Select the MIMXRT685-EVK 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**.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 16. Select the `hello_world` TrustZone example

3. Now, two projects should be imported into the workspace. To start building the TrustZone application, highlight the `evkmimxrt685_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 17. For this example, select the **Debug** target.

Figure 17. 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

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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. Build the application for the secure project first.*

Figure 18. 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 *Section 3.3, "Run an example application"*. These steps are common for single core, and TrustZone applications, ensuring `evkmimxrt685_hello_world_s` is selected for debugging.

In the Quickstart Panel, click **Debug evkmimxrt685__hello_world_s [Debug]** to launch the second debug session.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 19. Debug `evkmimxrt685_hello_world_s` case

Figure 20. TrustZone debug sessions

Now, the TrustZone sesions 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.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 21. Run Hello World trustzone example and get the message

4 Run a demo application using IAR

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

Note: *IAR Embedded Workbench for Arm version 8.40.2 is used in the following example, and the IAR toolchain should correspond to the latest supported version, as described in the MCUXpresso SDK Release Notes (document ID: MCUXSDKRN).*

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 MIMXRT685-EVK hardware platform as an example, the `hello_world` workspace is located in:

   ```
   <install_dir>/boards/k32w061dk6/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**.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 22. Demo build target selection

3. To build the demo application, click **Make**, highlighted in red in Figure 23.

Figure 23. 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. See the table in Section 11 to determine the debug interface that comes loaded on your specific hardware platform.
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 Section 9). 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

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 24. Terminal (PuTTY) configuration

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

Figure 25. Download and Debug button

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

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 26. Stop at `main()` when running debugging

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

Figure 27. 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 28. 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
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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/evkmimxrt685/trustzone_examples/
hello_world/hello_world_s/iar/hello_world_s.eww
```

```
<install_dir>/boards/evkmimxrt685/trustzone_examples/
hello_world/hello_world_s/iar/hello_world.eww
```

This project `hello_world.eww` 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 steps 1 – 4 as described in *Section 4.2, Run an example application*. These steps are common for both 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 29. Stop at `Rest_Hander` when running debugging

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

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 30. 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 31. Text display of the `trustzone hello_world` application

Note: *If the application is running in RAM (debug/release build target), in* ***Options > Debugger > Download*** *tab, disable* ***Use flash loader(s)****. This can avoid the* `_ns` *download issue on i.MXRT600.*

Figure 32. Disabling Use flash loader(s)

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 MIMXRT685-EVK hardware platform which is used as an example.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Note: *ARMGCC version 8-2019-q3 is used as an example in this document. The latest GCC version for this package is as described in the MCUXpresso SDK Release Notes Supporting MIMXRT685-EVK (document MCUXSDKMIMXRT6XXRN).*

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, and so on). The GCC toolchain should correspond to the latest supported version, as described in *MCUXpresso SDK Release Notes*.

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 MinGW.
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 33. Set up MinGW and MSYS

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

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 34. 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.*

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 35. 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:

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.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

```
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 36. Convert path to short path

Figure 37. Add ARMGCC_DIR system variable

5.1.4 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.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 38. 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 39. Launch command prompt

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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/evkmimxrt685/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 40.

```
[ 85%] Building C object CMakeFiles/hello_world.elf.dir/C_/nxp/SDK_2.5.0_EVK-MIMXRT685/devices/MIMXRT685S/drivers/fsl_fl
excomm.c.obj
[ 90%] Building C object CMakeFiles/hello_world.elf.dir/C_/nxp/SDK_2.5.0_EVK-MIMXRT685/devices/MIMXRT685S/drivers/fsl_gp
io.c.obj
[ 95%] Building C object CMakeFiles/hello_world.elf.dir/C_/nxp/SDK_2.5.0_EVK-MIMXRT685/devices/MIMXRT685S/drivers/fsl_re
set.c.obj
[100%] Linking C executable debug\hello_world.elf
[100%] Built target hello_world.elf

C:\nxp\SDK_2.5.0_EVK-MIMXRT685\boards\evkmimxrt685\demo_apps\hello_world\armgcc>IF "" == "" (pause )
Press any key to continue . . .
```

Figure 40. `hello_world` **build successful**

5.3 Run an example application

This section describes steps to run a demo application using J-Link GDB Server application. To update the on-board LPC-Link2 debugger to Jlink firmware, see Section 12.

Note: *J-Link GDB Server application is not supported for TFM examples. Use CMSIS DAP instead of J-Link for flashing and debugging TFM examples.*

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 LPC-Link2 USB connector and the PC USB connector. If using a standalone J-Link debug pod, 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 Section 9). 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

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 41. Terminal (PuTTY) configurations

Note: *Make sure the board is set to FlexSPI flash boot mode (ISP2: ISP1: ISP0 = ON, OFF, ON) before use GDB debug.*

3. Open the J-Link GDB Server application.
4. Open the J-Link GDB Server application. Go to the SEGGER install folder. For example, *C:\Program Files(x86)\SEGGER\JLink_Vxxx*. Open the command windows. Use the JLinkGDBServer.exe -device MIMXRT685S_M33 -if SWD -scriptfile: <install_dir>/boards/<board_name>/<example_type>/<application_name>/evkmimxrt685.JLinkScript command.
5. After it is connected, the screen should look like this figure:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 42. 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 43. 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/evkmimxrt685/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`.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 44. Run `arm-none-eabi-gdb`

9. Run these commands:
   a. `target remote localhost:2331`
   b. `monitor halt`
   c. `load`
   d. `monitor reset`
10. The application is now downloaded and halted at the watch point. 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 does not appear, check your terminal settings and connections.

Figure 45. 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
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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/evkmimxrt685/trustzone_examples/
hello_world/hello_world_ns/armgcc/build_debug.bat
```

```
<install_dir>/boards/evkmimxrt685/trustzone_examples/
hello_world/hello_world_s/armgcc/build_debug.bat
```

Build both applications separately, following steps for single core examples as described in *Section 6.2, "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 46. `hello_world_s` **example build successful**

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 47. `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 *Section 6.3, "Run an example application"*.

To download and run the TrustZone application, perform steps 1 to 10, as described in *Section 5.3, "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 halt`
4. `monitor exec SetFlashDLNoRMWThreshold = 0x20000`
5. `load <install_dir>/boards/evkmimxrt685/trustzone_examples/hello_world/hello_world_ns/armgcc/debug/hello_world_ns.elf`
6. `load <install_dir>/boards/evkmimxrt685/trustzone_examples/hello_world/hello_world_s/armgcc/debug/hello_world_s.elf`
7. `monitor reset`

The application is now downloaded and halted at the watch point. Execute the `monitor go` command to start the demo application.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

```
$ arm-none-eabi-gdb.exe
GNU gdb (GNU Tools for Arm Embedded Processors 8-2019-q3-update) 8.3.0.20190703-git
Copyright (C) 2019 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".
(gdb) target remote localhost:2331
Remote debugging using localhost:2331
warning: No executable has been specified and target does not support
determining executable automatically.  Try using the "file" command.
0x0001c04a in ?? ()
(gdb) monitor halt
(gdb) load hello_world_ns/armgcc/debug/hello_world_ns.elf
Loading section .interrupts, size 0x130 lma 0x8041000
Loading section .text, size 0x1be8 lma 0x8041130
Loading section .ARM, size 0x8 lma 0x8042d18
Loading section .init_array, size 0x4 lma 0x8042d20
Loading section .fini_array, size 0x4 lma 0x8042d24
Loading section .data, size 0x60 lma 0x8042d28
Start address 0xc01e4, load size 7560
Transfer rate: 3691 KB/sec, 1260 bytes/write.
(gdb) load hello_world_s/armgcc/debug/hello_world_s.elf
Loading section .flash_config, size 0x1000 lma 0x18000000
Loading section .interrupts, size 0x130 lma 0x18001000
Loading section .text, size 0x5258 lma 0x18001130
Loading section .ARM, size 0x8 lma 0x18006388
Loading section .init_array, size 0x4 lma 0x18006390
Loading section .fini_array, size 0x4 lma 0x18006394
Loading section .data, size 0x70 lma 0x18006398
Loading section .gnu.sgstubs, size 0x20 lma 0x18040e00
Start address 0x100801e4, load size 25640
Transfer rate: 4173 KB/sec, 2848 bytes/write.
(gdb) monitor reset
Resetting target
(gdb) monitor go
(gdb)
```

Figure 48. Loading and running the trustzone example

```
COM57 - PuTTY
Hello from secure world!
Entering normal world.
Welcome in normal world!
This is a text printed from normal world!
Comparing two string as a callback to normal world
String 1: Test1
String 2: Test2
Both strings are not equal!
```

Figure 49. Text display of the trustzone `hello_world` application

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.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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 MIMXRT685S CMSIS pack.

1. Download the MIMXRT685S pack .
2. After downloading the DFP, double click to install it.

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/evkmimxrt685s/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 the table in Section 11 to determine the debug interface that comes loaded on your specific hardware platform.
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 serial port number (to determine the COM port number, see Section 9). 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

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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**.
   Note: *When debugging with jlink, it expects one jlinkscript file named JLinkSettings.JLinkScript in the folder where the uVision project files are located. For details, see Segger Wiki. For the contents in this JlinkSettings.JLinkScript, use contents in evkmimxrt1020_sdram_init.jlinkscript.*
   Note: *Make sure the board is set to FlexSPI flash boot mode (ISP2: ISP1: ISP0 = ON, OFF, ON) before using KEIL debug.*

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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_s.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.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

To download and run the TrustZone application, switch to the secure application project and perform steps as described in Section 6.5. 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.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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

7 Run a demo using the prebuilt binary

This section describes the steps to write a prebuilt binary in the SDK package to the external Flash and boot from the external Flash. The `hello_world` demo application is used as an example.

Note: *The prebuilt binaries are provided for quick evaluation of the SDK and the board. Only parts of the examples are provided with prebuilt binary such as examples under `boards/<board_name>/demo_apps` and `boards/<board_name>/usb_examples`.*

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

7.1 Identify the load address of the binary

The MCUXpresso SDK supports several toolchains. The prebuilt binaries are built from the armgcc flash_release targets using Arm GCC. If there is no flash_release target, then the release target is used.

For example, the hello_world demo application located in `<install_dir>/boards/<board_name>/<demo_apps>/hello_world/`.

The example was built from `<install_dir>/boards/<board_name>/<demo_apps>/hello_world/armgcc/build_flash_release.sh`.

The linker file in the same folder identifies the load address. For example, `<device_name>_flash.ld`.

```
MEMORY
{
  m_flash_config         (RX)   : ORIGIN = 0x08000400, LENGTH = 0x00000200
  m_interrupts           (RX)   : ORIGIN = 0x08001000, LENGTH = 0x00000130
  m_text                 (RX)   : ORIGIN = 0x08001130, LENGTH = 0x001FEED0
  m_data                 (RW)   : ORIGIN = 0x20080000, LENGTH = 0x00180000
  m_usb_sram             (RW)   : ORIGIN = 0x40140000, LENGTH = 0x00004000
```

Figure 58. Memory layout in the linker

Therefore, the load address for hello_world.bin is 0x08000400.

7.2 Write the binary to external flash

To write the binary to external flash:

1. To write the hello_world.bin to external flash, take J-Link Commander as an example.
2. Run the J-Link Commander and to connect the core run the following command.

```
J-Link>connect
J-Link>?
```

3. Select the correct device from the pop-up window **Target device settings**.
4. Specify targets interface as SWD and specify target interface speed per the guide in the command window.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

The “Cortex-M33 identified” appears when it is connected successfully.

Figure 59. Connect successfully

5. Use below command to write the hello_world.bin to the load address 0x08000400.

```
J-Link>loadbin hello_world.bin 0x08000400
```

7.3 Run the application

Either press the reset button on the board or use the following command in J-Link Commander to run the demo.

```
J-Link>rnh
```

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 60.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Figure 60. 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.

9 How to determine COM port

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

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

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

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

10 How to define IRQ handler in CPP files

With MCUXpresso SDK, users could define their own IRQ handler in application level to

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

override the default IRQ handler. For example, to override the default `PIT_IRQHandler` define in `startup_DEVICE.s`, application code like app.c can be implement like:

```
c
void PIT_IRQHandler(void)
{
    // Your code
}
```

When application file is CPP file, like app.cpp, then `extern "C"` should be used to ensure the function prototype alignment.

```
cpp
extern "C" {
    void PIT_IRQHandler(void);
}
void PIT_IRQHandler(void)
{
    // Your code
}
```

11 Default debug interfaces

The MCUXpresso SDK supports various hardware platforms that come loaded with a variety of factory programmed debug interface configurations. Table 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 1 is not applicable to LPC.*

Table 1. Hardware platforms supported by 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/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

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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

Hardware platform	Default interface	OpenSDA details
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
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
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

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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

Hardware platform	Default interface	OpenSDA details
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

12 Updating debugger firmware

12.1 Updating OpenSDA firmware

Any NXP hardware platform that comes with an OpenSDA-compatible debug interface has the ability to update the OpenSDA firmware. 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 Section 11. 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.
- 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 61. 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.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

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.*

12.2 Updating LPCXpresso board firmware

The LPCXpresso 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 (JP5 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 www.nxp.com/lpcutilities.

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 (www.nxp.com/lpcutilities, select **LPCScrypt**, and then the documentation tab).

1. Install the LPCScript utility.
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).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

7. Re-power the board by removing the USB cable and plugging it in again.

13 Revision history

This table summarizes revisions to this document.

Table 2. Revision history

Revision number	Date	Substantive changes
0	February 2018	Initial Release
1	15 January 2021	Updated for MCUXpresso SDK v2.9.0
2.10.0	10 July 2021	Updated for MCUXpresso SDK v2.10.0
2.11.0	11 November 2021	Updated for MCUXpresso SDK v2.11.0
2.11.1	11 March 2022	Updated for MCUXpresso SDK v2.11.1
2.12.0	30 June 2022	Updated for MCUXpresso SDK v2.12.0

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

14 Legal information

14.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

14.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

14.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Contents

1 Overview 2
2 MCUXpresso SDK board support package folders 2
2.1 Example application structure 3
2.2 Locating example application source files 3
3 Run a demo using MCUXpresso IDE 4
3.1 Select the workspace location 4
3.2 Build an example application 4
3.3 Run an example application 8
3.4 Build a TrustZone example application 13
3.5 Run a TrustZone example application 16
4 Run a demo application using IAR 18
4.1 Build an example application 18
4.2 Run an example application 19
4.3 Build a TrustZone example application 21
4.4 Run a TrustZone example application 22
5 Run a demo using Arm® GCC 23
5.1 Set up toolchain 24
5.1.1 Install GCC Arm Embedded tool chain 24
5.1.2 Install MinGW (only required on Windows OS) 24
5.1.3 Add a new system environment variable for ARMGCC_DIR 26
5.1.4 Install CMake 27
5.2 Build an example application 28
5.3 Run an example application 29
5.4 Build a TrustZone example application 32
5.5 Run a TrustZone example application 34
6 Run a demo using Keil® MDK/µVision 35
6.1 Install CMSIS device pack 36
6.2 Build an example application 36
6.3 Run an example application 36
6.4 Build a TrustZone example application 39
6.5 Run a TrustZone example application 39
7 Run a demo using the prebuilt binary 41
7.1 Identify the load address of the binary 42
7.2 Write the binary to external flash 42
7.3 Run the application 43
8 MCUXpresso IDE New Project Wizard 43
9 How to determine COM port 44
10 How to define IRQ handler in CPP files 44
11 Default debug interfaces 45
12 Updating debugger firmware 47
12.1 Updating OpenSDA firmware 47
12.2 Updating LPCXpresso board firmware 48
13 Revision history 49
14 Legal information 50

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© NXP B.V. 2022. All rights reserved.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 30 June 2022
Document identifier: MCUXSDKMIMXRT6XXGSUG