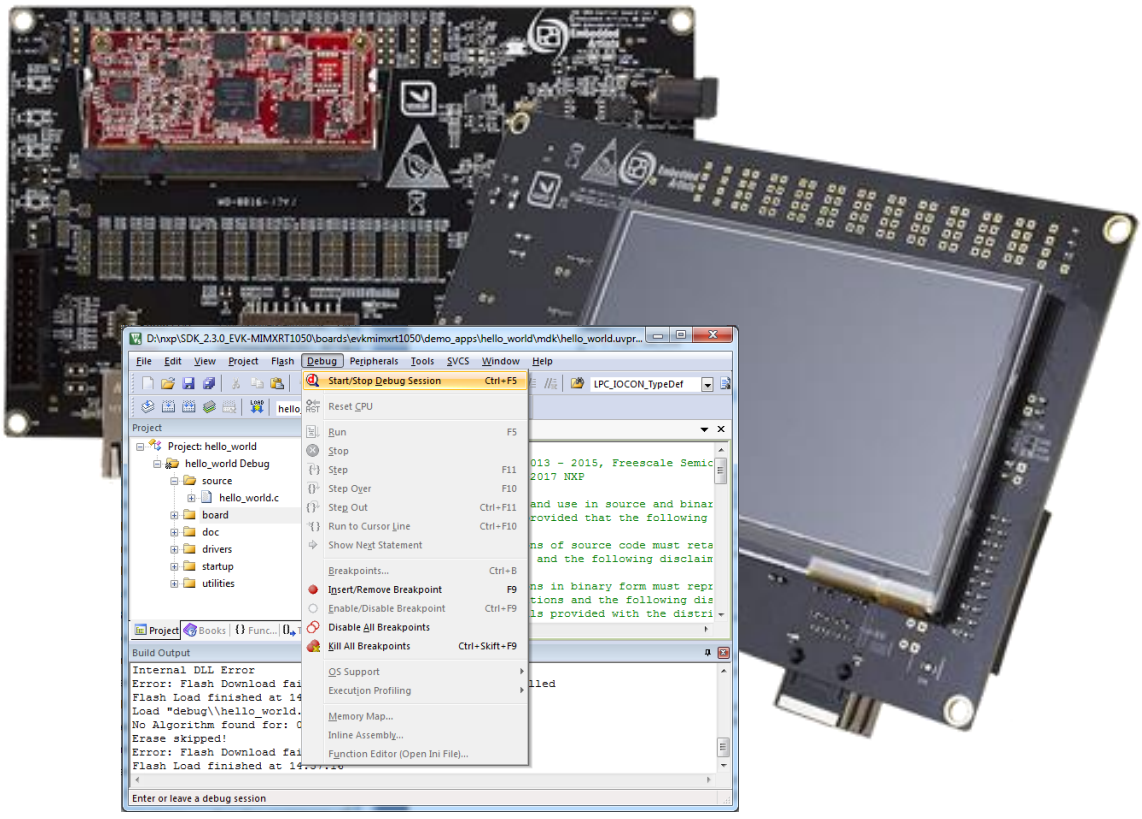
iMX RT1052/1062 Developer's Kit Program Development Guide



Get Up-and-Running Quickly and
Start Developing Your Application On Day 1!

Embedded Artists

Embedded Artists AB
Rundelsgatan 14
211 36 Malmö
Sweden

http://www.EmbeddedArtists.com

Disclaimer

Embedded Artists AB makes no representation or warranties with respect to the contents hereof and specifically disclaim any implied warranties or merchantability or fitness for any particular purpose. Information in this publication is subject to change without notice and does not represent a commitment on the part of Embedded Artists AB.

Feedback

We appreciate any feedback you may have for improvements on this document. Please send your comments to support@EmbeddedArtists.com.

Trademarks

All brand and product names mentioned herein are trademarks, services marks, registered trademarks, or registered service marks of their respective owners and should be treated as such.

Table of Contents

1 Document Revision History

Revision	*Date*	*Description*
A	2020-01-31	Updated to match SDK 2.7.0. Added information about MCUXpresso Secure Provisioning Tools. Updated example descriptions and patches.
A1	2020-05-13	Updated information in section 2.3.1 regarding clocking and the new patched file.
B	2020-12-18	Document updated to match the new pre-patched SDK (2.8.6) that is provided.
B1	2022-11-18	Updated to include information about iMX RT1062 OEM board rev C1. Updated Secure Provisioning to handle v5 of the tool.

2 Get Started with Program Development

This chapter contains information about how to get started with program development on the *iMX RT1052 Developer's Kit* and the *iMX RT1062 Developer's Kit*. Since the kits are almost identical from a program development perspective, for the rest of this document the name *iMX RT Developer's Kit* will be used. Only when there is a difference will the individual *iMX RT Developer's Kit* be referenced.

The document also contains information about program download, i.e., how to flash the OctalSPI/QuadSPI flash memory.

To start program development, you need the following things, all of them:

1. **Patched version of MCUXpresso SDK** - this is a package of sample software from NXP that has been patched by Embedded Artists to work with the *iMX RT Developer's Kit*. The zip-file can be downloaded from http://imx.embeddedartists.com.
2. **Integrated Development Environment (IDE)**
    a. **NXP MCUXpresso, Keil uVision/MDK and IAR Embedded Workbench** supports direct flash programming of the OctalSPI/QuadSPI. Update to latest version of respective IDE.
    b. Programming of the OctalSPI/QuadSPI flash is also supported via NXP's *MCUXpresso Secure Provisioning Tools* standalone application but it is not a suitable tool to use during program development.
3. **JTAG probe** to be able to download the application to SRAM, SDRAM or the OctalSPI/QuadSPI flash memory and in general to be able to debug - set breakpoints, inspect memory, etc.
    a. The low-cost MCU-Link or LPC-Link2 are excellent choices. Keil ULINK2 and ULINKplus, as well as Segger JLINK, are also excellent debug probes.
    b. Technically it is possible to program/flash the OctalSPI/QuadSPI without a JTAG probe (via NXP's *MCUXpresso Secure Provisioning Tools* application), but it is strongly recommended to use the proper tool for debugging - i.e., use a JTAG probe!
4. And of course, the ***iMX RT Developer's Kit*!**

2.1 Downloading the SDK

Starting with SDK version 2.8.6 and going forward, Embedded Artists has published a version of the SDK that has already been patched to work with the *iMX RT Developer's Kit*. The file can be downloaded from http://imx.embeddedartists.com and will have a filename like

`eaimxrt1052_sdk_<version>_freertos_<date>.zip`, or

`eaimxrt1062_4mb_sdk_<version>_<date>.zip`

`eaimxrt1062_16mb_sdk_<version>_<date>.zip`

2.2 About the SDK

NXP's SDK builder (https://mcuxpresso.nxp.com/en/) can be used to generate SDKs for:

- Windows, MAC or Linux
- FreeRTOS, AzureRTOS or bare metal (i.e., no RTOS)
- With all examples or a subset

- For all compilers/IDEs or for one of (MCUXpresso, ARM GCC, Keil uVision/MDK, IAR Embedded Workbench)

Depending on which options are selected it limits the number of examples that can be downloaded. The SDK with the broadest set of examples is the **Windows/All IDEs/All Examples/FreeRTOS** combination and that is the combination that was selected for the patched SDK.

This is what has been patched:

- Set CPU speed according to Commercial/Industrial CPU
- Correction of the VDD_SOC_IN voltage.
- LWIP projects – added reading of the MAC address from the onboard I2C EEPROM
- Changed flash algorithm, the size of the flash and the header file for the flash
- Wi-Fi and Bluetooth projects
    - added an I2C driver for the gpio expander and code to use it
    - modified pin muxing
    - patched the WICED driver
- Adjusted the USB interface number (it is different for host and device examples)
- Changed reset pin for SD card examples
- Changed the mflash driver to work with the OctalSPI/QuadSPI flash

2.3 Getting Started with a Specific IDE

The following chapters will describe how to get started with several different IDEs.

Chapter 3 describes Keil uVision and the SDK.

Chapter 4 describes Keil uVision and RTE.

Chapter 5 describes IAR Embedded Workbench.

Chapter 6 describes NXP MCUXpresso.

3 Getting Started with Keil uVision/MDK and SDK

This section is a guide to open, build and debug an example application using Keil uVision/MDK with the SDK that was downloaded in 2.1 . It is assumed that you have this development environment installed on your computer.

Following the instructions in this section will make it possible to run all examples in the downloaded SDK. Note that none of the examples use the RTE (Run-Time Environment) feature of the Keil uVision/MDK. The use of RTE is described in chapter 4 .

Make sure the version of uVision is v5.25.2, or later. Otherwise, there is no support for flash download/programming to the OctalSPI flash memory directly.

3.1 Install the SDK

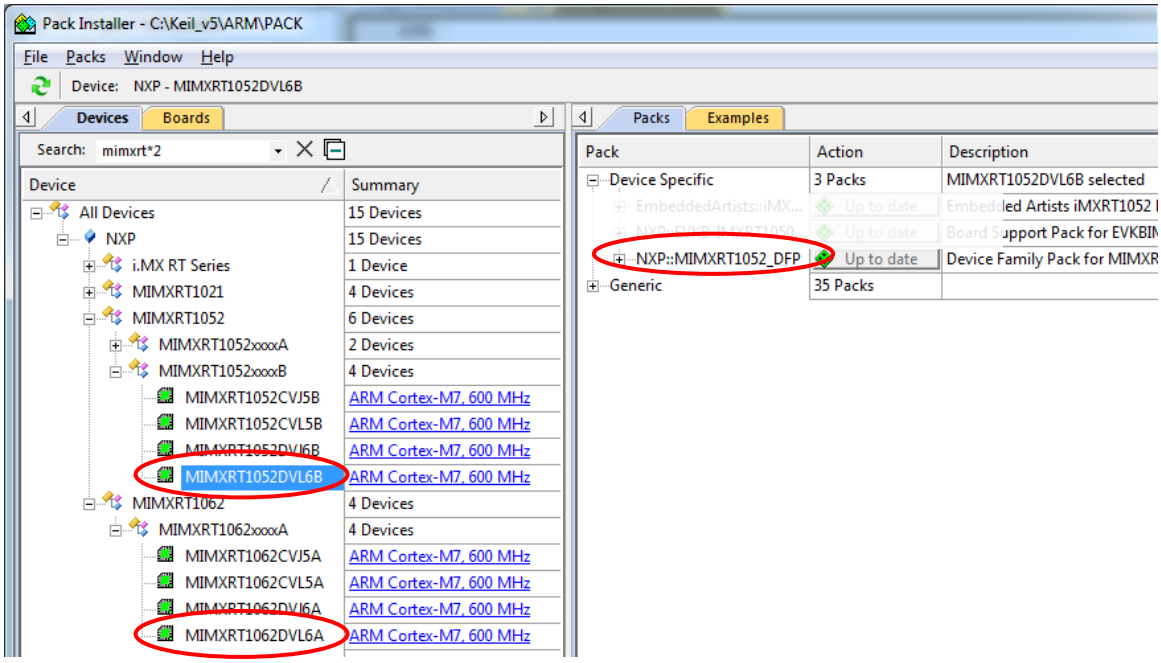
Unpack the SDK zip file that was downloaded in 2.1 somewhere on the local file system. It is suggested to use a very short path as the SDK has a deep folder structure and a path that is too long can cause problems in Windows.

The folder that the SDK is unpacked into will be referred to as **`<install_dir>`** in the following sections.

3.2 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 IMXRT105x CMSIS pack or IMXRT106x CMSIS pack.

1. Start Keil uVision
2. Start the Pack Installer tool with this button on the toolbar: 
3. Browse to the MIMXRT1052 in the device tree (NXP→MIMXRT1052→MIMXRT1052xxxxB) or MIMXRT1062 in the device tree (NXP→MIMXRT1062→MIMXRT1062xxxxA) and then click the Install button next to the package with a "_DFP" in the name. Note that the button in the image below has "Up to date" as the package has already been installed.



3.3 Build an Example Application

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

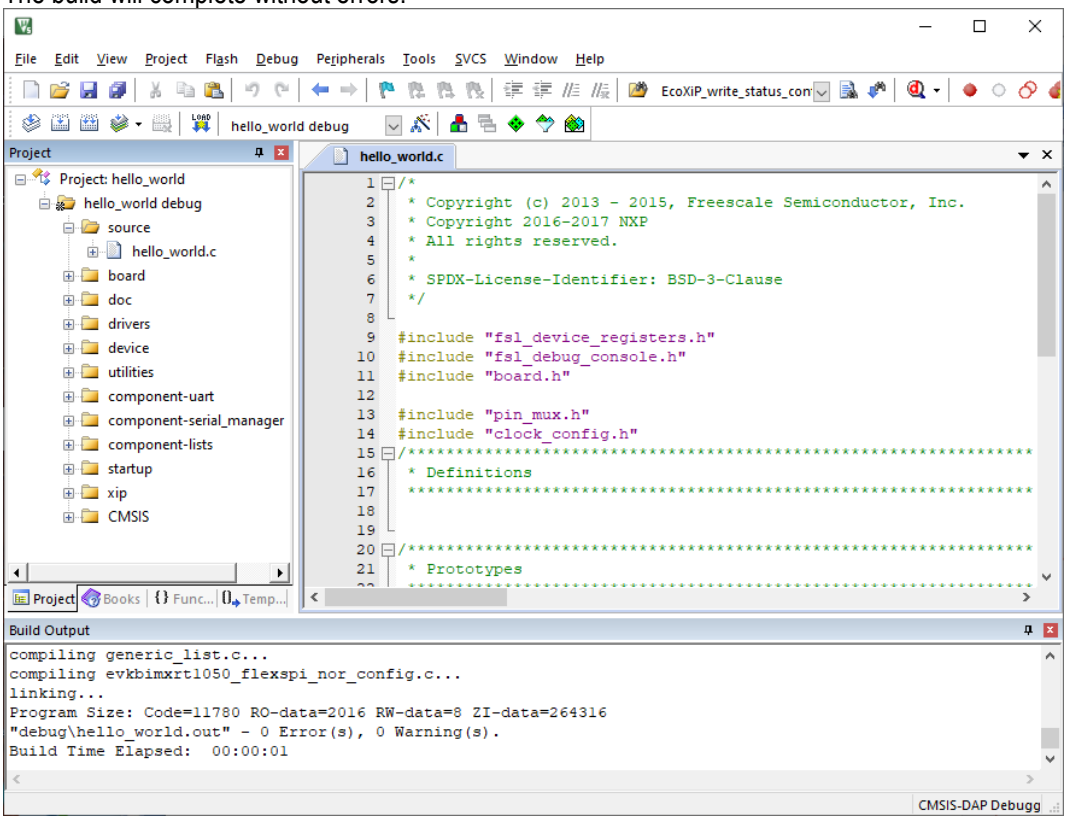
1. If not already done, open the desired example application workspace in:
   <install_dir>/boards/evkbimxrt1052/<example_type>/<application_name>/mdk
   The workspace file is named **<demo_name>.uvmpw**, so for this specific example, the actual path is:
   <install_dir>/boards/evkbimxrt1052/demo_apps/hello_world/mdk/hello_world.uvmpw

   or

   <install_dir>/boards/evkmimxrt1062/<example_type>/<application_name>/mdk
   The workspace file is named **<demo_name>.uvmpw**, so for this specific example, the actual path is:
   <install_dir>/boards/evkmimxrt1062/demo_apps/hello_world/mdk/hello_world.uvmpw

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

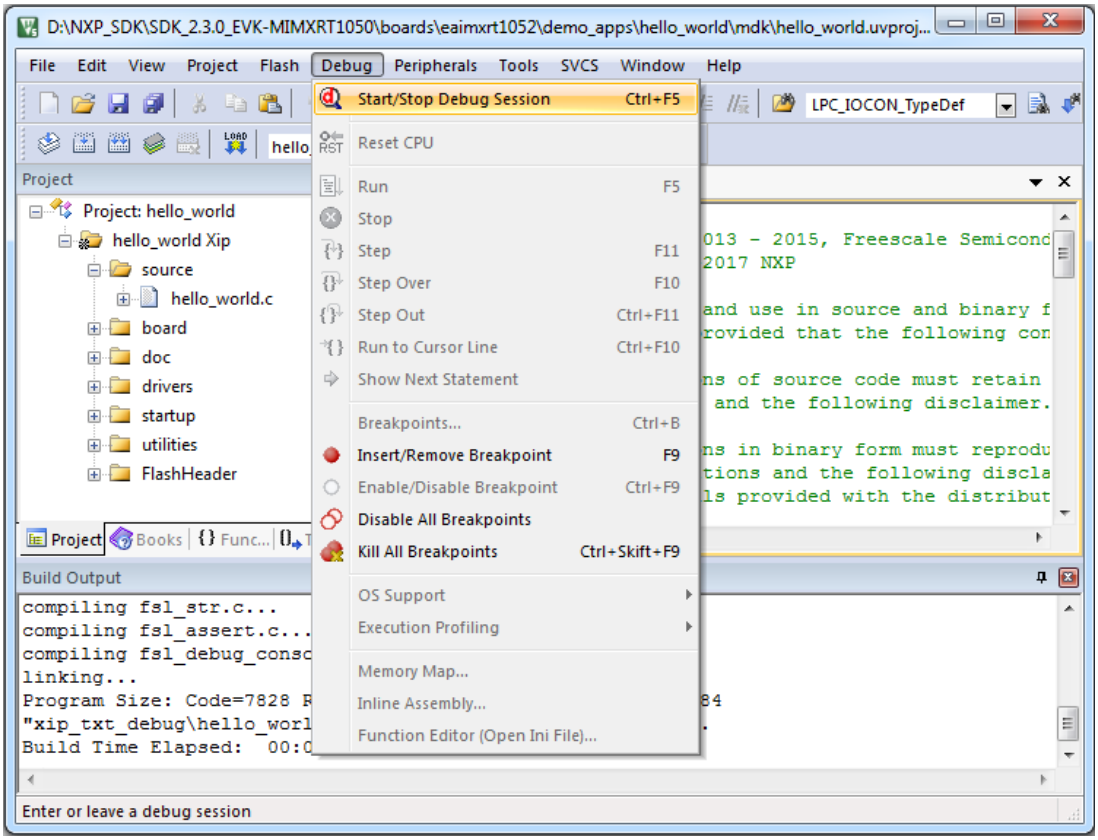
3. The build will complete without errors.



3.4 Run an Example Application

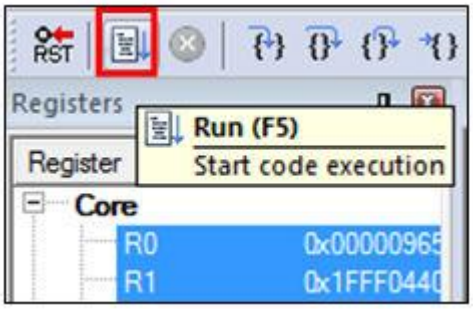
To download and run the application, perform these steps:

1. Connect the debug probe to the development platform to your PC via USB cable. See chapter 7 for details how to connect the debug probe.
2. Select the *Debug* menu and then *Start/Stop Debug Session*, or simply press Ctrl+F5.
   The application will then be downloaded into SRAM.

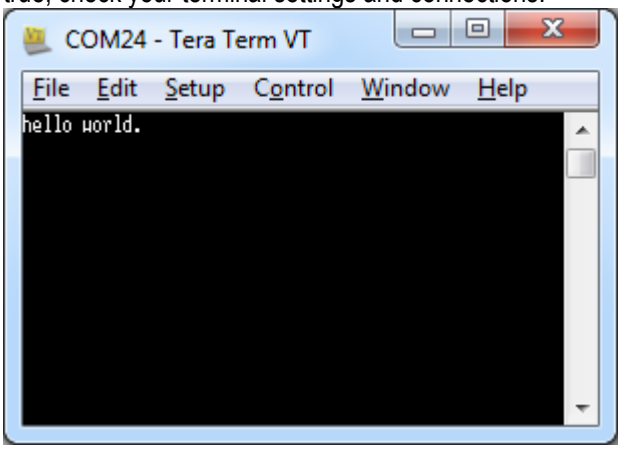


3. Open the terminal application on the PC, such as TeraTerm or PuTTY, and connect to the debug serial port number. Configure the terminal with 115200 baud, 8N1.
   You can alter the baud rate be searching for the reference BOARD_DEBUG_UART_BAUDRATE variable in file: **`board.h`**

4. Run the code by clicking the "Run" button (or tress F5) to start the application.



5. 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.



3.5 Selecting Build Target

A target is a variant of the project that for example can change compiler flags or dictate in what memory region the program should execute. All projects have one or more targets as shown in the picture below.

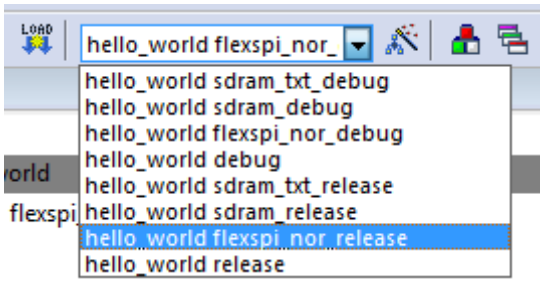


Figure 1 – Build Targets in Keil uVision/MDK

The target name starts with the project name, **hello_world** in this case, and then the nomenclature as outlined in the table below is used. Release targets have a higher compiler optimization degree making the target less suitable for debugging.

<project> sdram_debug <project> sdram_release	The application runs in internal SRAM but with data in external SDRAM
<project> sdram_txt_debug <project> sdram_txt_release	The application runs in external SDRAM but with data in internal SRAM
<project> debug <project> release	The application runs in internal SRAM
<project> flexspi_nor_debug <project> flexspi_nor_release	The application runs in external flash, which must be programmed/flashed before use.

Note that each of the targets has its own set of options so changing for example include path in one target does not affect the other targets.

4 Getting Started with Keil uVision/MDK and RTE

This section is a guide to open, build and debug an example application using Keil uVision/MDK with the Board Support Package (BSP) installed using the Pack Installer and **not the SDK described in section 2.1** . It is assumed that you have this development environment installed on your computer.

The BSP contains a small subset of all the examples available in the SDK. The main difference between the SDK and the BSP examples is that the BSP examples use the Run-Time Environment (RTE) feature in Keil uVision/MDK and the SDK doesn't.

The RTE is a way to manage software components (e.g. drivers, RTOS, network stacks, USB stacks) to add/remove features in a project without having to add each source file and include path individually.

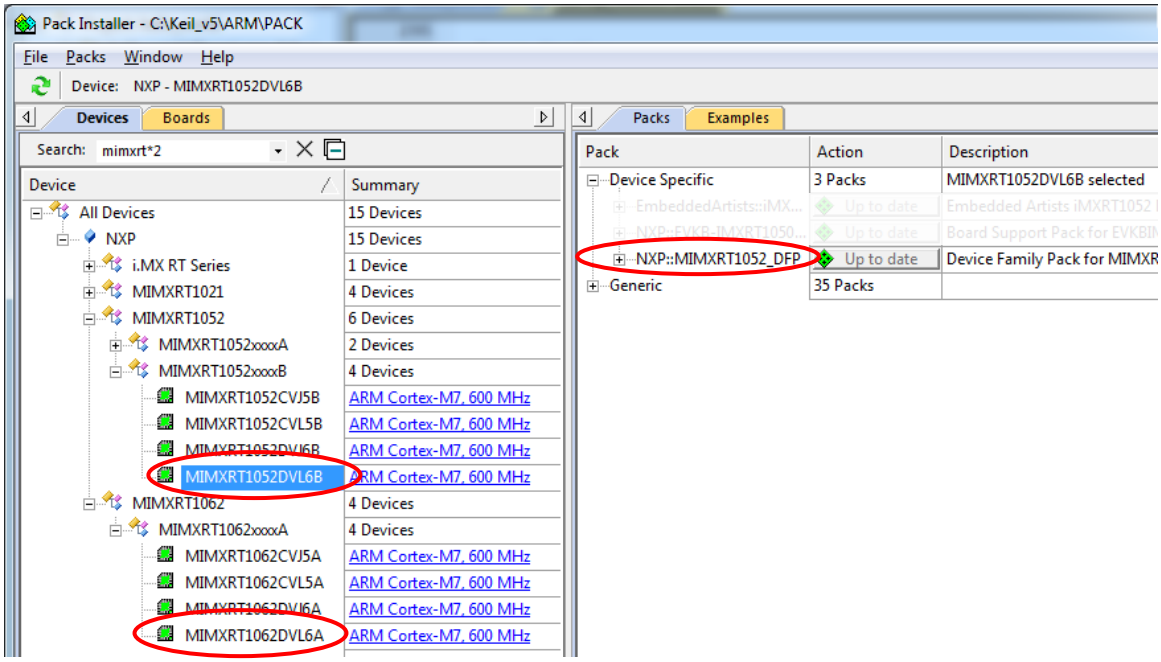
> The recommendation is not to use RTE, but instead to use the SDK (as described in chapter 3) examples as a base for your programming. There are only a very limited set of examples are available for RTE.

Make sure the version of uVision is v5.25.2, or later. Else there is no support for flash download/programming to the OctalSPI flash memory directly.

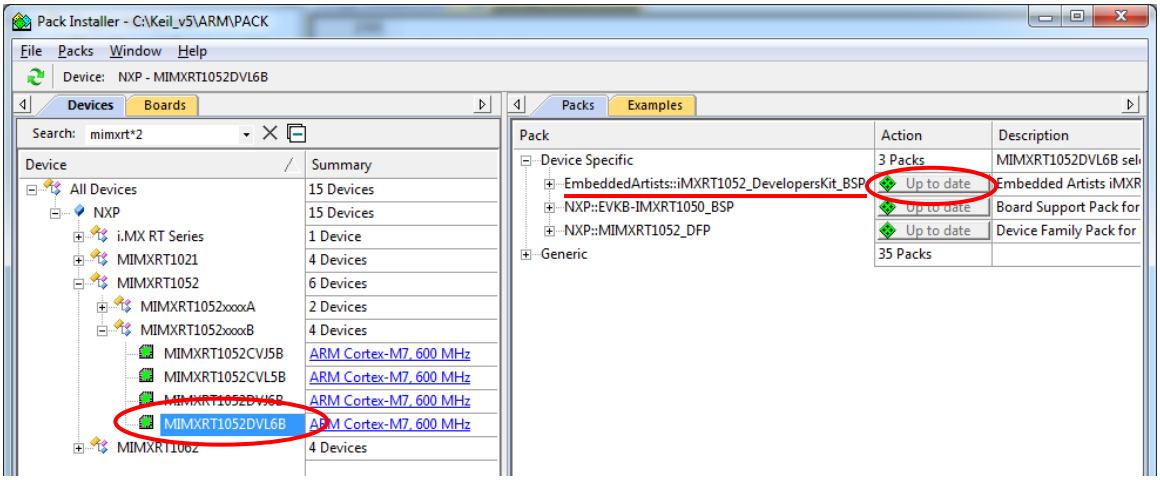
4.1 Install CMSIS Device Pack

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 IMXRT105x CMSIS pack or IMXRT106x CMSIS pack.

1. Start Keil uVision
2. Start the Pack Installer tool with this button on the toolbar: 
3. Browse to the MIMXRT1052 in the device tree (NXP→MIMXRT1052→MIMXRT1052xxxxB) or MIMXRT1062 in the device tree (NXP→MIMXRT1062→MIMXRT1062xxxx) and then click the Install button next to the package name. Note that the button in the image below has "Up to date" as the package has already been installed.



4. Next click the Install button for the EmbeddedArtists::iMXRT1052_DevelopersKit_BSP or EmbeddedArtists::iMXRT1062_DevelopersKit_BSP. Note that the button in the image below has "Up to date" as the package has already been installed

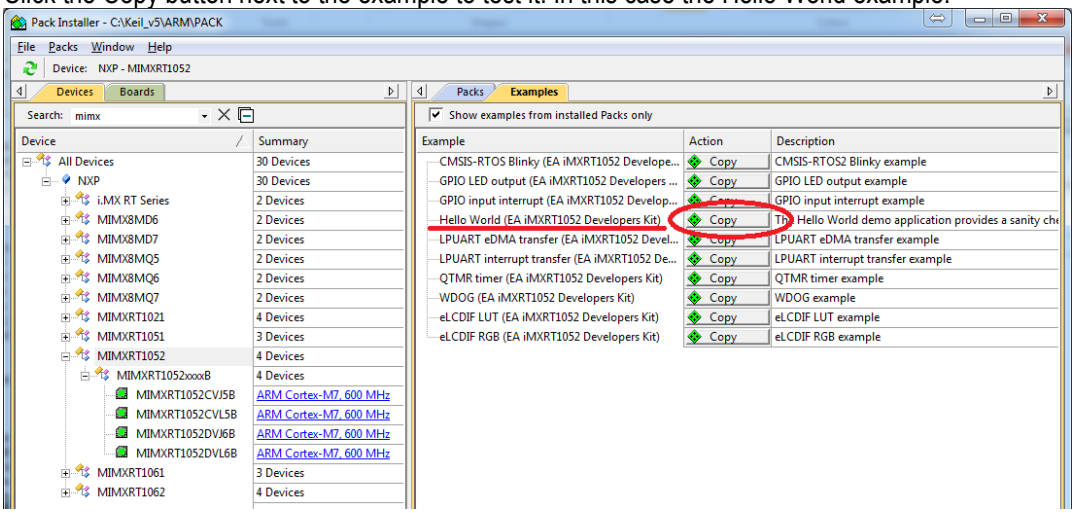


4.2 Build an Example Application

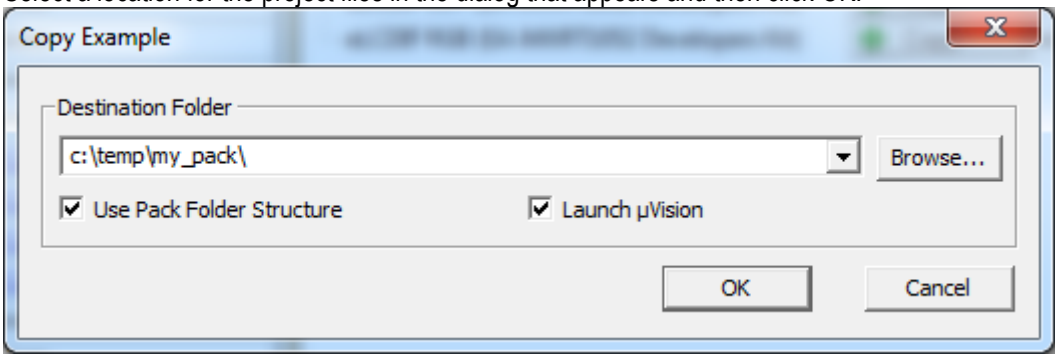
The following steps will guide you through opening the Hello World application. These steps may change slightly for other example applications as some of these applications may have additional layers of folders in their path.

1. Start Keil uVision
2. Start the Pack Installer tool with this button on the toolbar: 
3. Browse to the MIMXRT1052 in the device tree (NXP→MIMXRT1052→MIMXRT1052xxxxB) or MIMXRT1062 in the device tree (NXP→MIMXRT1062→MIMXRT1062xxxxB) and then select the Examples tab

4. Click the Copy button next to the example to test it. In this case the Hello World example.



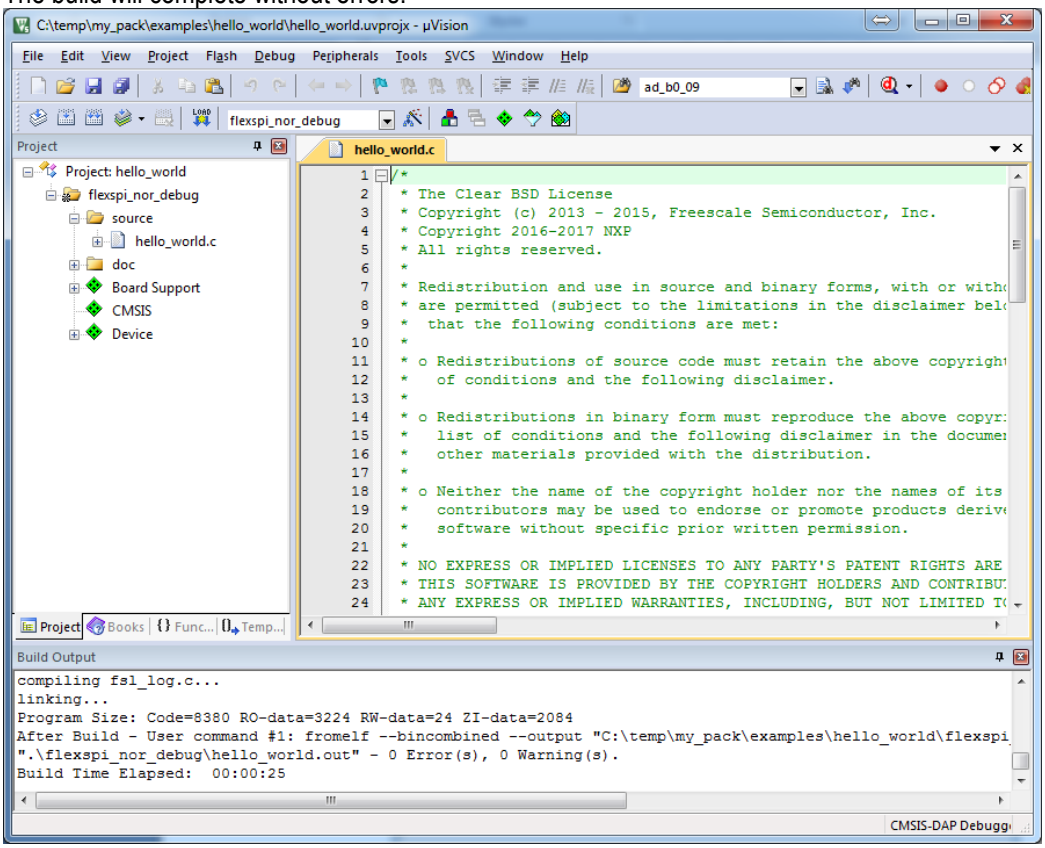
5. Select a location for the project files in the dialog that appears and then click OK.



6. A new uVision window will appear after a few seconds with the newly created project

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

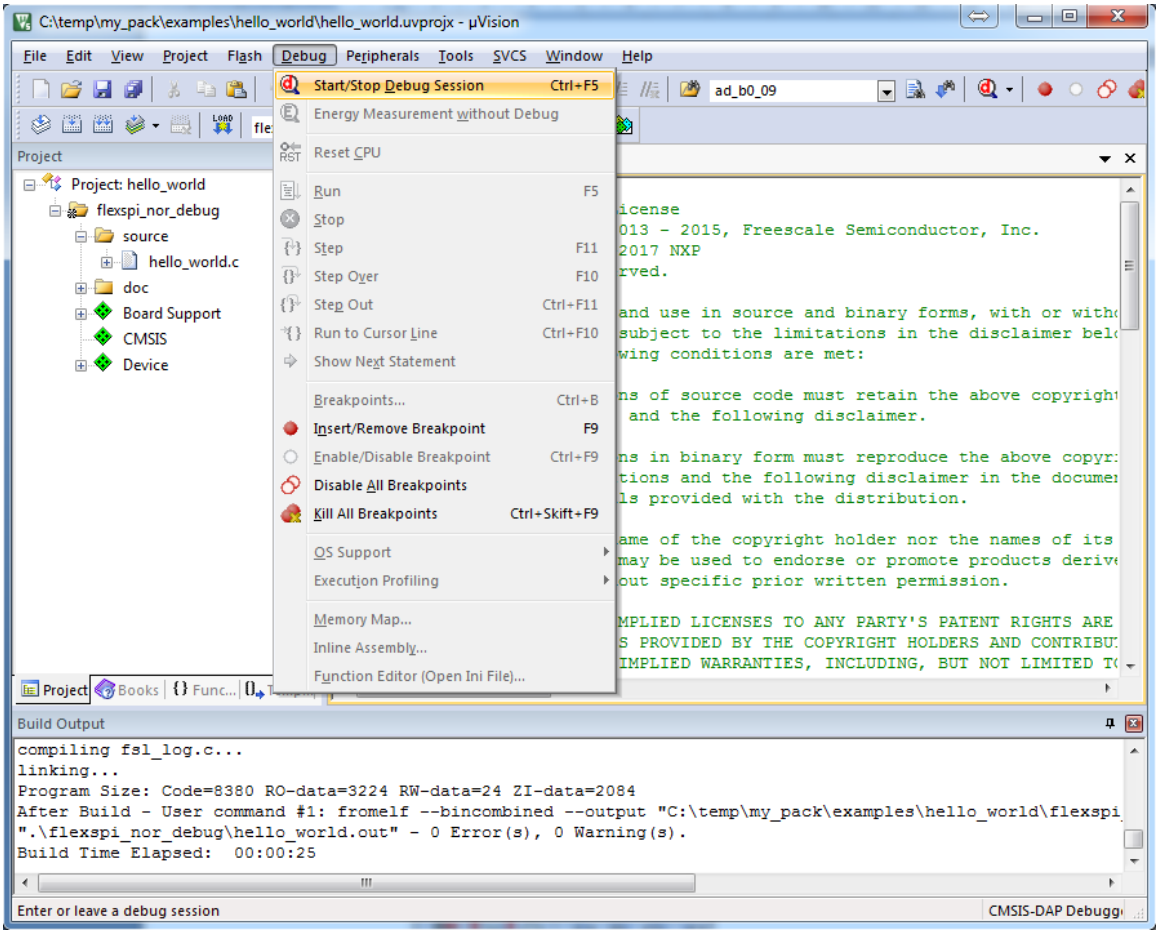
8. The build will complete without errors.



4.3 Run an Example Application

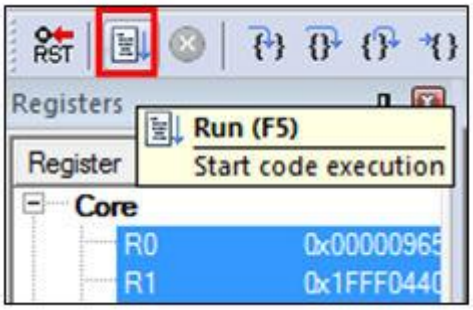
To download and run the application, perform these steps:

1. Connect the debug probe to the development platform to your PC via USB cable. See chapter 7 for details how to connect the debug probe.
2. Select the *Debug* menu and then *Start/Stop Debug Session*, or simply press Ctrl+F5.
   The application will then be downloaded into flash and started.

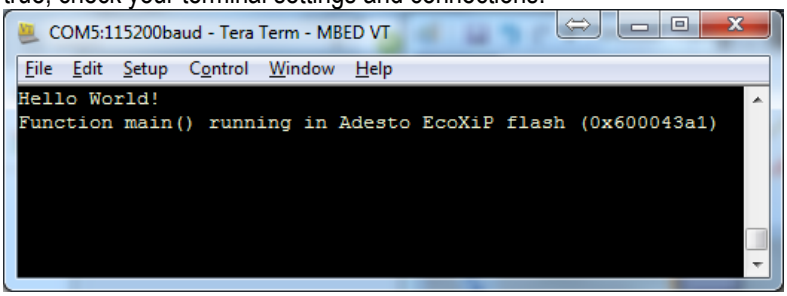


3. Open the terminal application on the PC, such as TeraTerm or PuTTY, and connect to the debug serial port number. Configure the terminal with 115200 baud, 8N1.
   You can alter the baud rate be searching for the reference BOARD_DEBUG_UART_BAUDRATE variable in file: **`board.h`**

4. Run the code by clicking the "Run" button (or tress F5) to start the application.



5. 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.



4.4 Selecting Build Target

A target is a variant of the project that for example can change compiler flags or dictate in what memory region the program should execute. All projects have one or more targets as shown in the picture below.

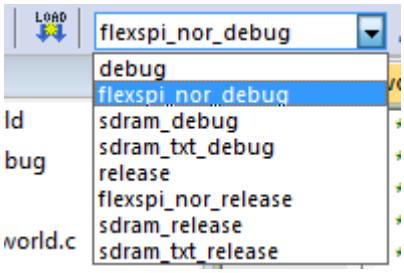


Figure 2 – Build Targets in Keil uVision/MDK

The default target is **flexspi_nor_debug**. The table below shows each of the targets. Release targets have a higher compiler optimization degree making the target less suitable for debugging.

sdram_debug sdram_release	The application runs in internal SRAM but with data in external SDRAM
sdram_txt_debug sdram_txt_release	The application runs in external SDRAM but with data in internal SRAM
debug release	The application runs in internal SRAM
flexspi_nor_debug flexspi_nor_release	The application runs in external flash, which must be programmed/flashed before use. Need extra configuration, see section 4.5 .

Note that each of the targets has its own set of options so changing for example include path in one target does not affect the other targets.

4.5 Configuring Flash Algorithm

If you have an iMX RT1052/62 OEM board up to, and including, rev B2 you must setup projects for storing/executing the application from the OctalSPI flash. Select one of the `flexspi_nor_*` targets, right click on the project and select *Options for Target <name>* (or press Alt+F7). Select the *Debug tab* and select *Settings* for the debugger. Then select the *Flash Download* tab. Replace the existing driver with the OctalSPI flash driver as shown below.

> This is **NOT** applicable for iMX RT1062 boards of **revision C1 or higher** as those use the same flash as the NXP EVK so it is already correct.

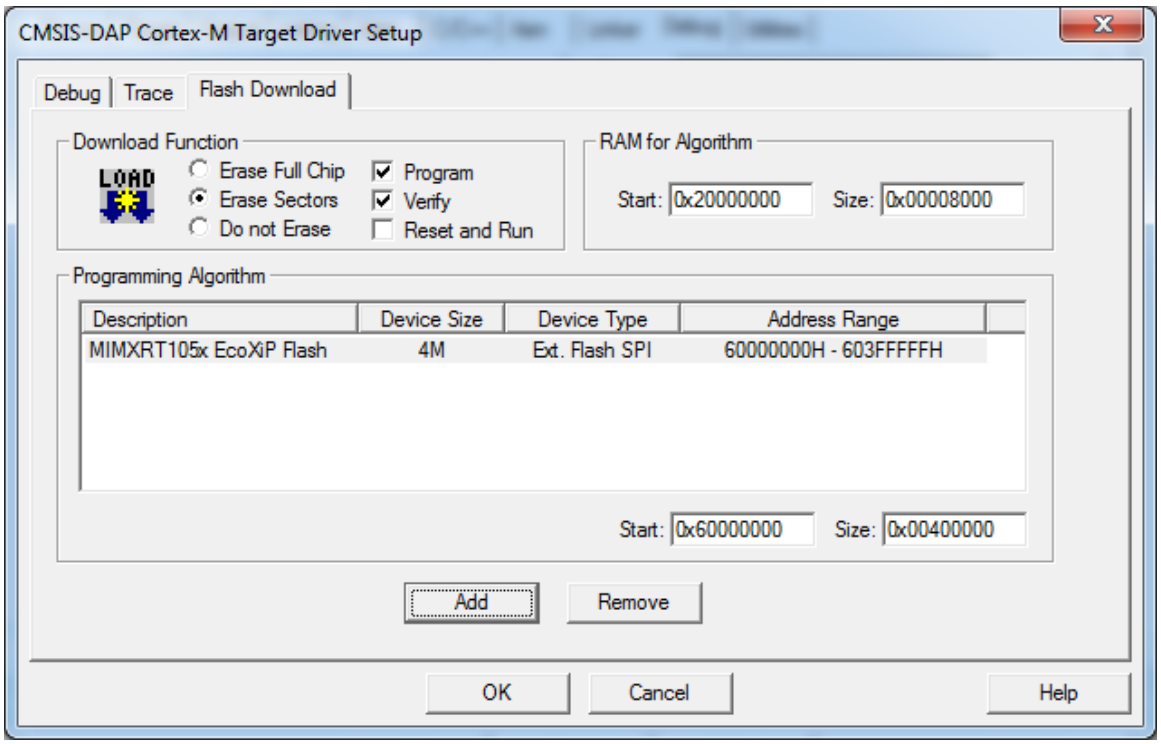
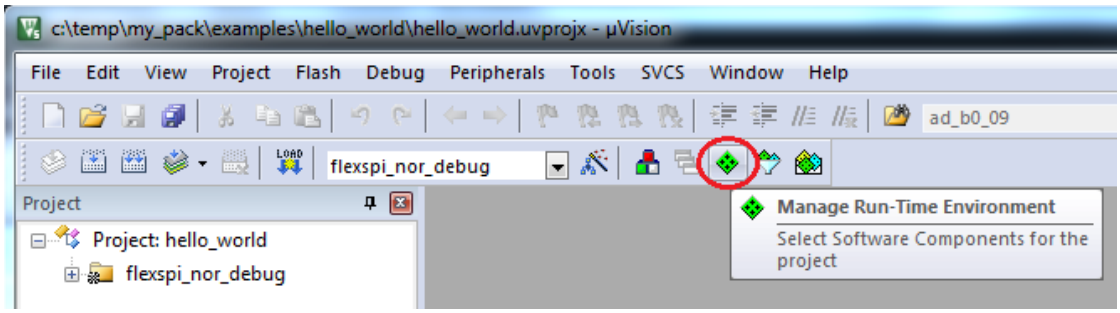


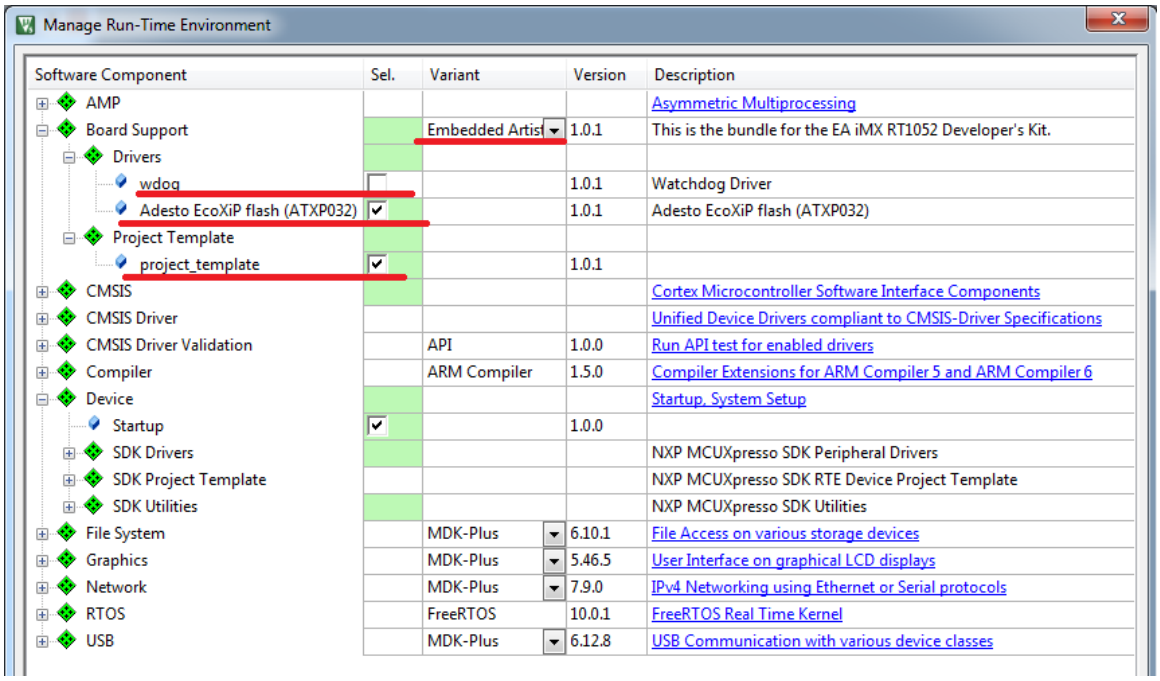
Figure 3 – Keil uVision/MDK Flash Algorithm Settings

4.6 Working with RTE

The Manage Run-Time Environment tool is started with a button on the toolbar:



The example projects are already configured correctly, and it is possible to add more software components by ticking the corresponding checkboxes.



There are a couple of very important things to note:

1. The Board Support must be "Embedded Artists" to get the drivers and templates as seen above.
2. If you have an iMX RT1052/62 OEM board with revision up to, and including, rev B2 then the Adesto EcoXip flash (ATXP032) must be selected otherwise applications will not start after being flashed.
3. If the watchdog driver (wdog) is used it must be selected from Board Support->Drivers->wdog and not the Device->SDK Drivers->wdog. The reason is that the one in Board Support is patched to work with the hardware.

The system for managing the RTE is described by Keil here:
http://www.keil.com/support/man/docs/uv4/uv4_ca_rtemanager.htm .

5 Getting Started with IAR Embedded Workbench

This section is a guide to open, build and debug an example application using IAR Embedded Workbench. It is assumed that you have this development environment installed on your computer.

Make sure the version of Embedded Workbench is 8.30.1, or later. Else there is no support for flash download/programming to the OctalSPI/QuadSPI flash memory directly.

5.1 Install the SDK

Unpack the SDK zip file that was downloaded (as described in section 2.1) somewhere on the local file system. It is suggested to use a very short path as the SDK has a deep folder structure and a path that is too long can cause problems in Windows.

The folder that the SDK is unpacked into will be referred to as **<install_dir>** in the following sections.

5.2 Build an Example Application

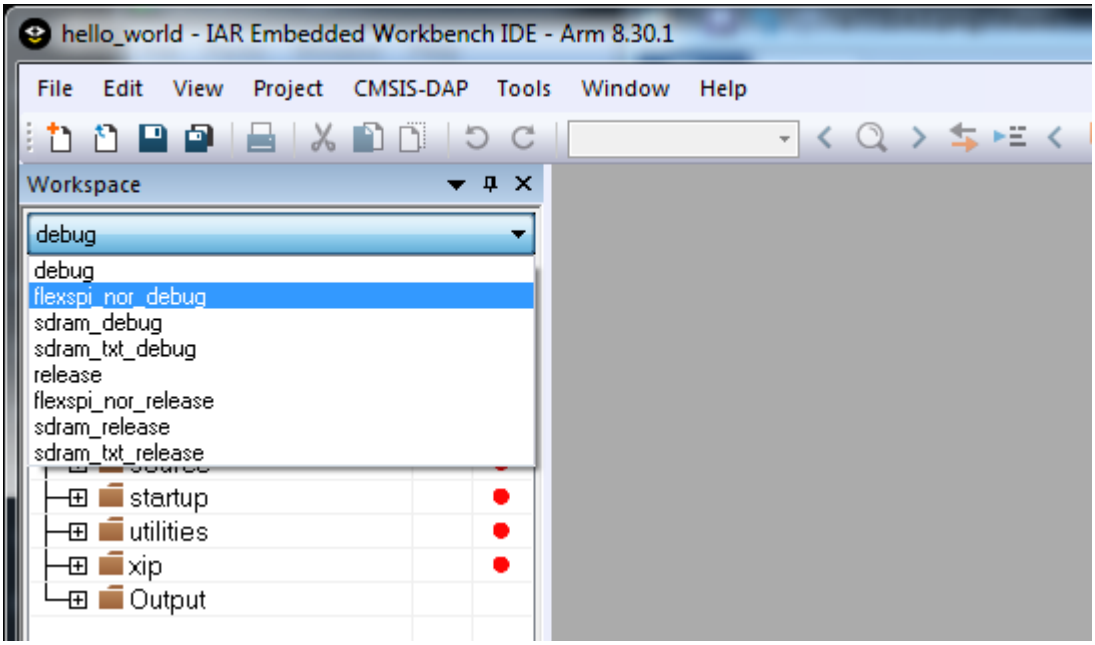
The following steps will guide you through opening the hello_world application. These steps may change slightly for other example applications as some of these applications may have additional layers of folders in their path.

1. If not already done, open the desired example application workspace in:
   <install_dir>/boards/evkbimxrt1050/<example_type>/<application_name>/iar
   The workspace file is named **<demo_name>.eww**, so for this specific example, the actual path is:
   <install_dir>/boards/evkbimxrt1050/demo_apps/hello_world/iar/hello_world.eww

   or

   <install_dir>/boards/evkmimxrt1060/<example_type>/<application_name>/iar
   The workspace file is named **<demo_name>.eww**, so for this specific example, the actual path is:
   <install_dir>/boards/evkmimxrt1060/demo_apps/hello_world/iar/hello_world.eww

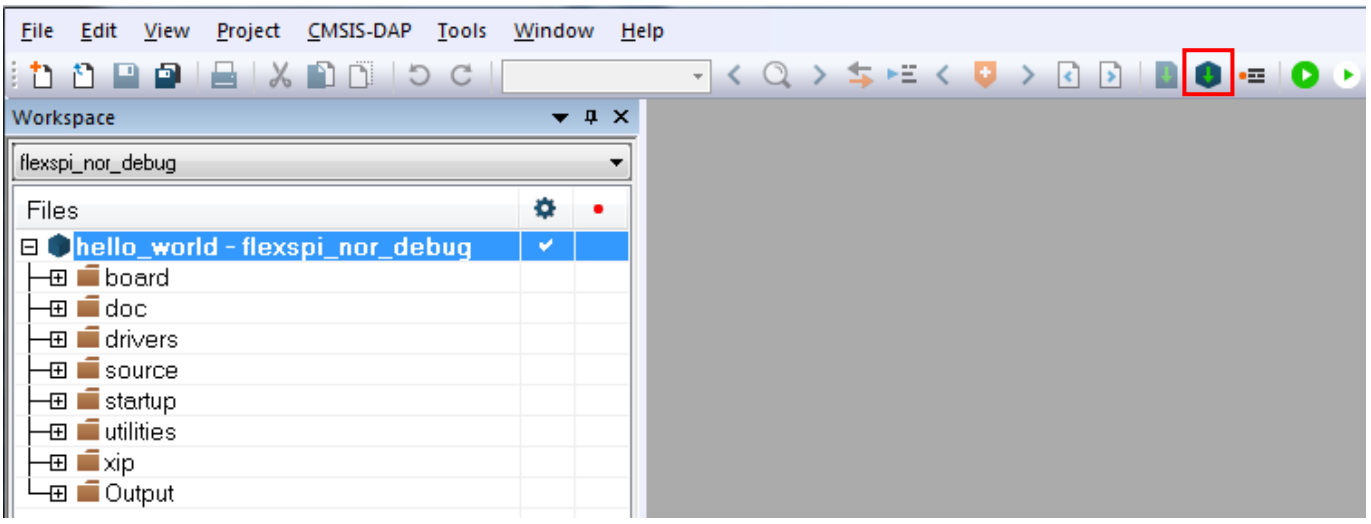
2. Select the desired build target from the drop-down.
   For this example, select the "flexspi_nor_debug" target.



The target name indicates from where the application is executing, see table below. The name also indicates if it is a release or debug target. Release targets have a higher compiler optimization degree making the target less suitable for debugging.

sdram_debug sdram_release	The application runs in internal SRAM but with data in external SDRAM
sdram_txt_debug sdram_txt_release	The application runs in external SDRAM but with data in internal SRAM
debug release	The application runs in internal SRAM
flexspi_nor_debug flexspi_nor_release	The application runs in external flash, which must be programmed/flashed before use.

3. Note that the target has its own set of options so changing, for example, the include path in one target does not affect the other targets.
4. To build the demo application, click the "Make" button, highlighted in red below.



5. The build completes without errors.

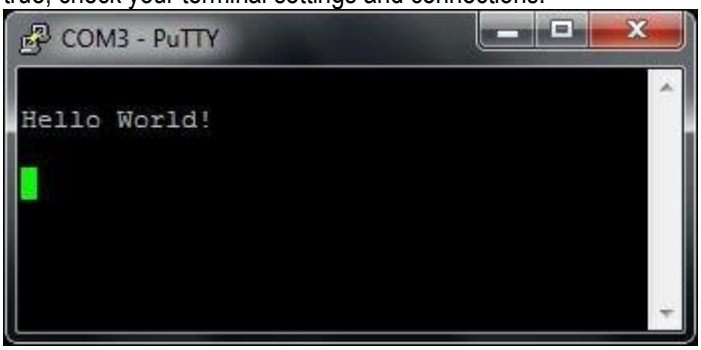
5.3 Run an Example Application

To download and run the application, perform these steps:

1. Connect the debug probe to the development platform to your PC via USB cable. See chapter 7 for details how to connect the debug probe.
2. Open the terminal application on the PC, such as TeraTerm or PuTTY, and connect to the debug serial port number. Configure the terminal with 115200 baud, 8N1.
   You can alter the baud rate be searching for the reference BOARD_DEBUG_UART_BAUDRATE variable in file: **`board.h`**
3. Click the "Download and Debug" button to download the application to the target.

4. The application is then downloaded to the target and automatically runs to the main() function.
5. Run the code by clicking the "Go" button to start the application.

6. 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.

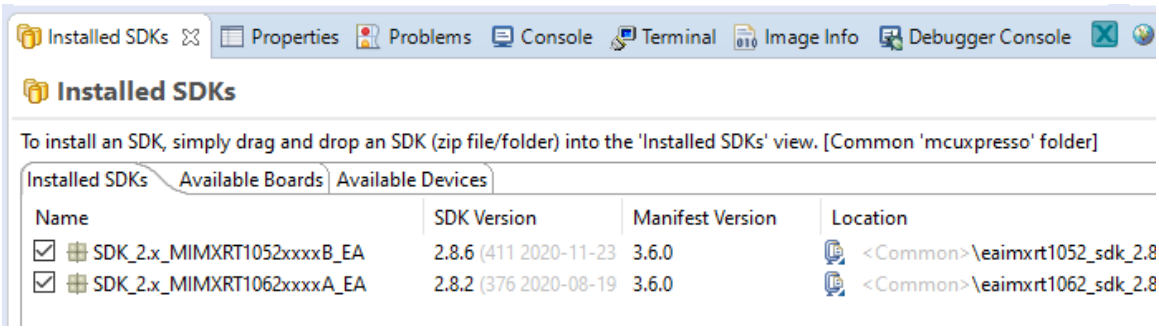


6 Getting Started with NXP MCUXpresso IDE

This section is a guide to open, build and debug an example application using NXP MCUXpresso IDE. It is assumed that you have this development environment installed on your computer.

6.1 Install the SDK

MCUXpresso requires the SDK to be installed before it can be used. To do that start MCUXpresso and then drag-n-drop the SDK zip file that was downloaded (as described in section 2.1) to the "Installed SDKs" tab in MCUXpresso:

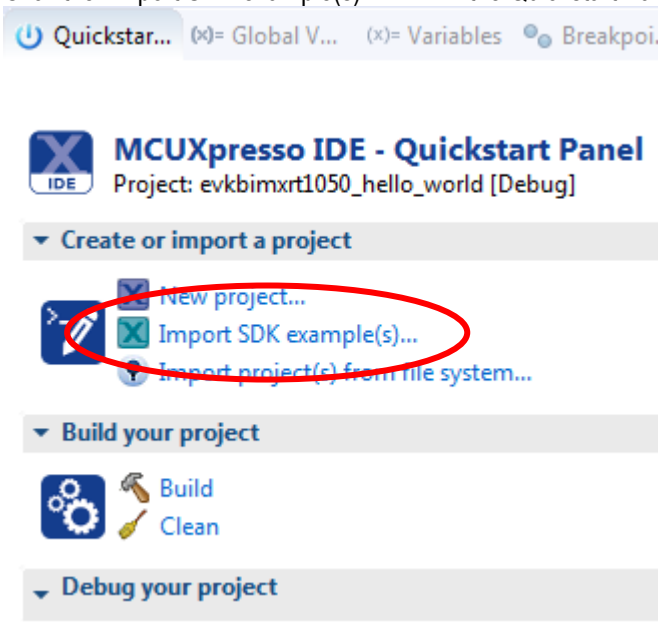


During the installation MCUXpresso will make a copy of the zip file that you drag-n-dropped so it is ok to delete it afterwards if you don't need it for one of the other IDEs.

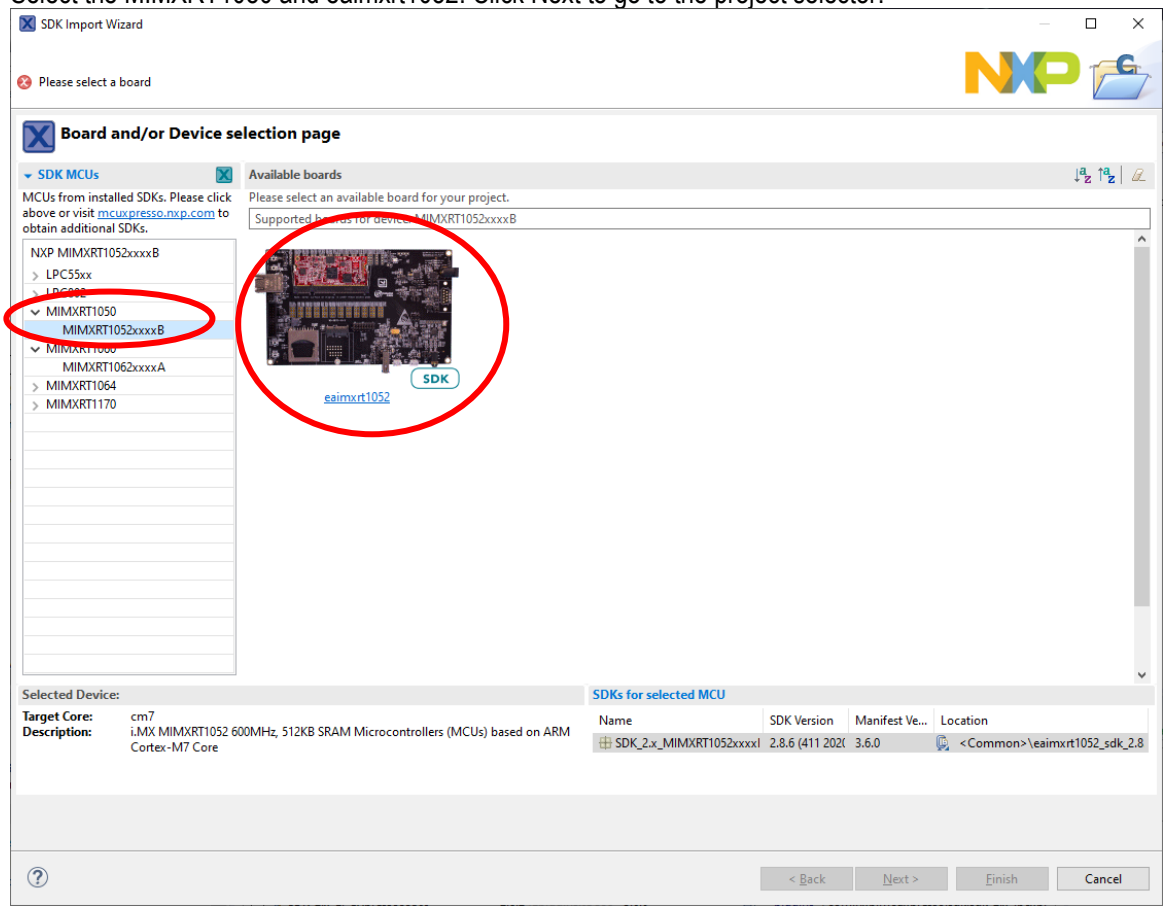
6.2 Build an Example Application

The following steps will guide you through opening the hello_world application from the SDK.

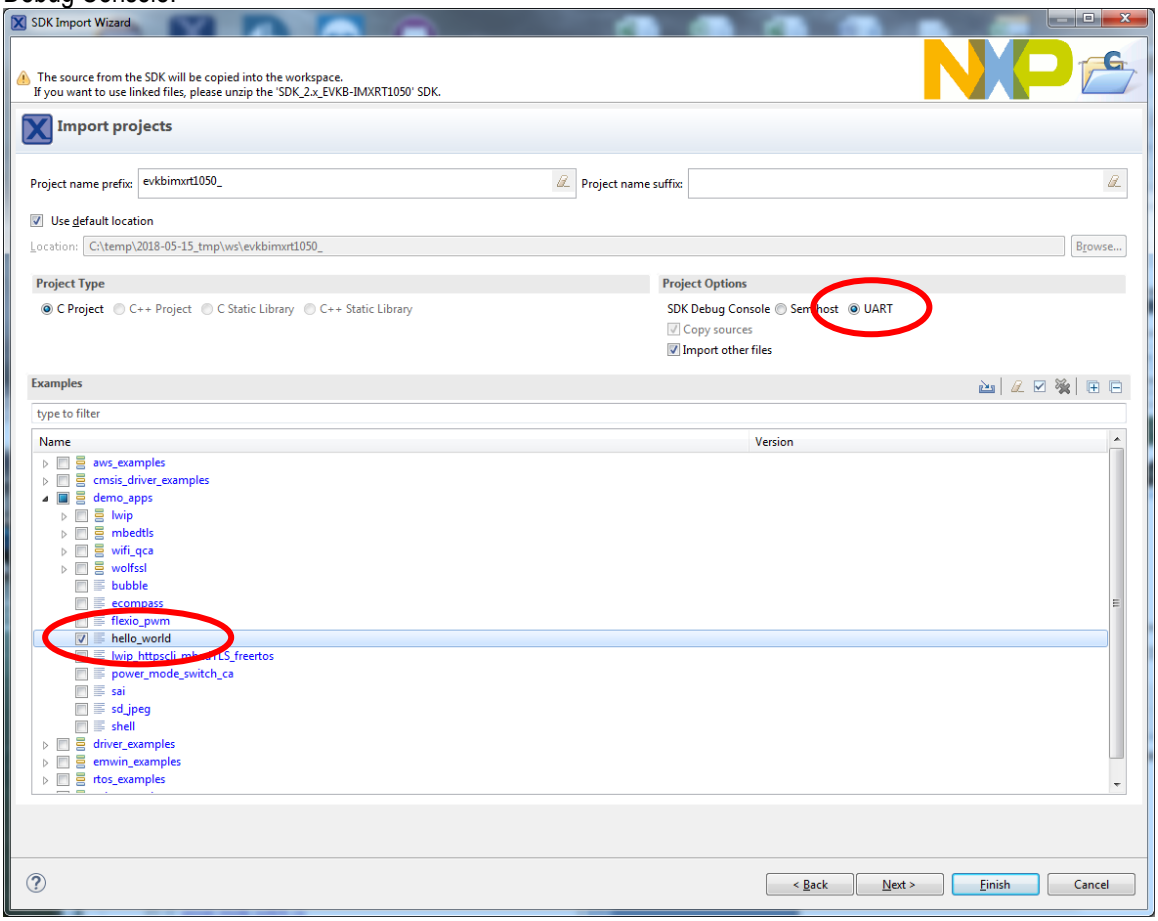
1. Install the SDK as described in section 6.1 if you have not done so already
2. Click the "Import SDK example(s)…" link in the Quickstart Panel



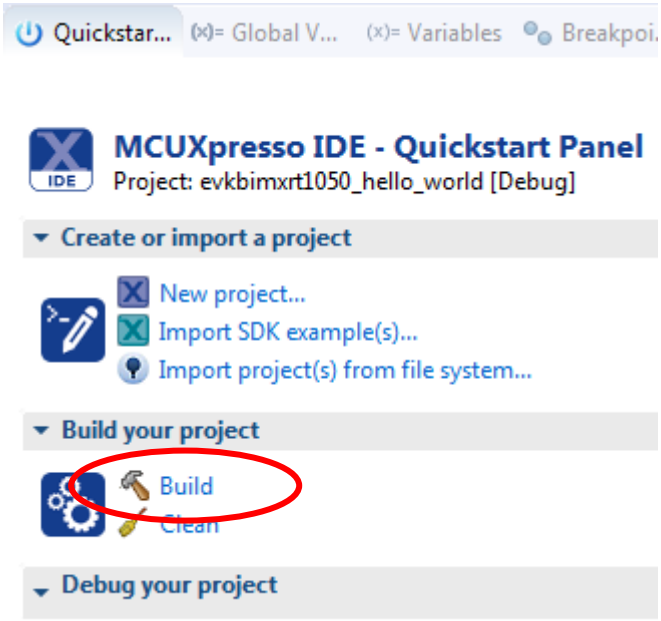
3. Select the MIMXRT1050 and eaimxrt1052. Click Next to go to the project selector.



4. Select the hello_world example and make sure to switch from Semihost to UART for the SDK Debug Console.



5. Click finish to have MCUXpresso import and set up the selected project.
6. Click Build in the Quickstart Panel

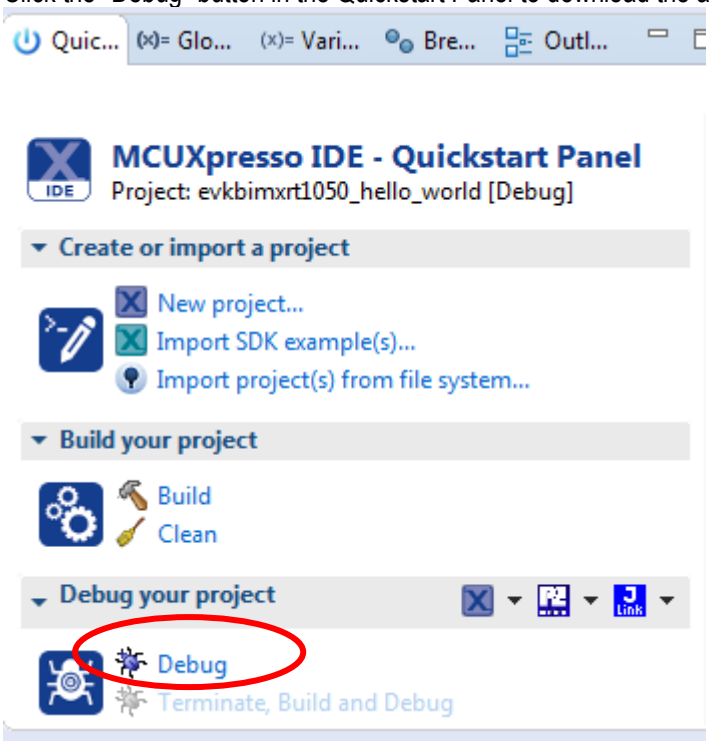


7. The program builds without errors

6.3 Run an Example Application

To download and run the application, perform these steps:

1. Connect the debug probe to the development platform to your PC via USB cable. See chapter 7 for details how to connect the debug probe.
2. Open the terminal application on the PC, such as TeraTerm or PuTTY, and connect to the debug serial port number. Configure the terminal with 115200 baud, 8N1.
   You can alter the baud rate be searching for the reference BOARD_DEBUG_UART_BAUDRATE variable in file: **`board.h`**
3. Click the "Debug" button in the Quickstart Panel to download the application to the target.

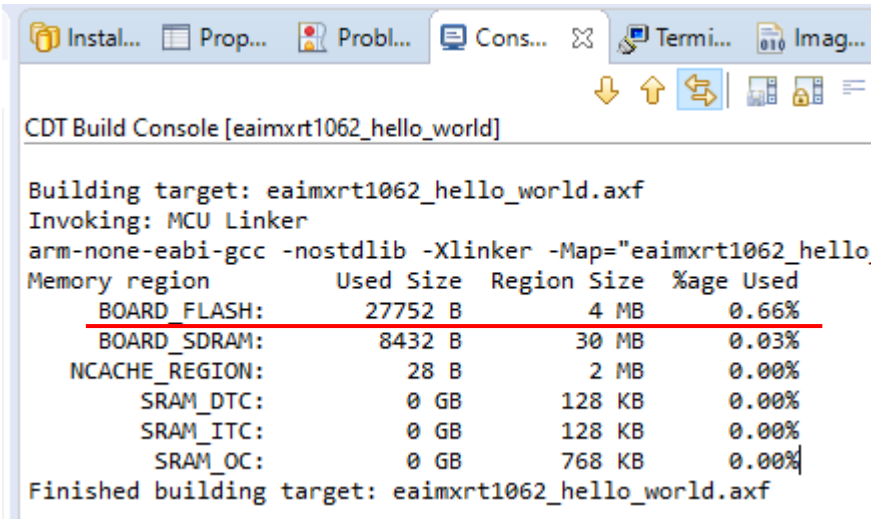


4. The application is then downloaded to the target and automatically runs to the main() function.
5. Run the code by clicking the "Resume" button to start the application.

6. 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.

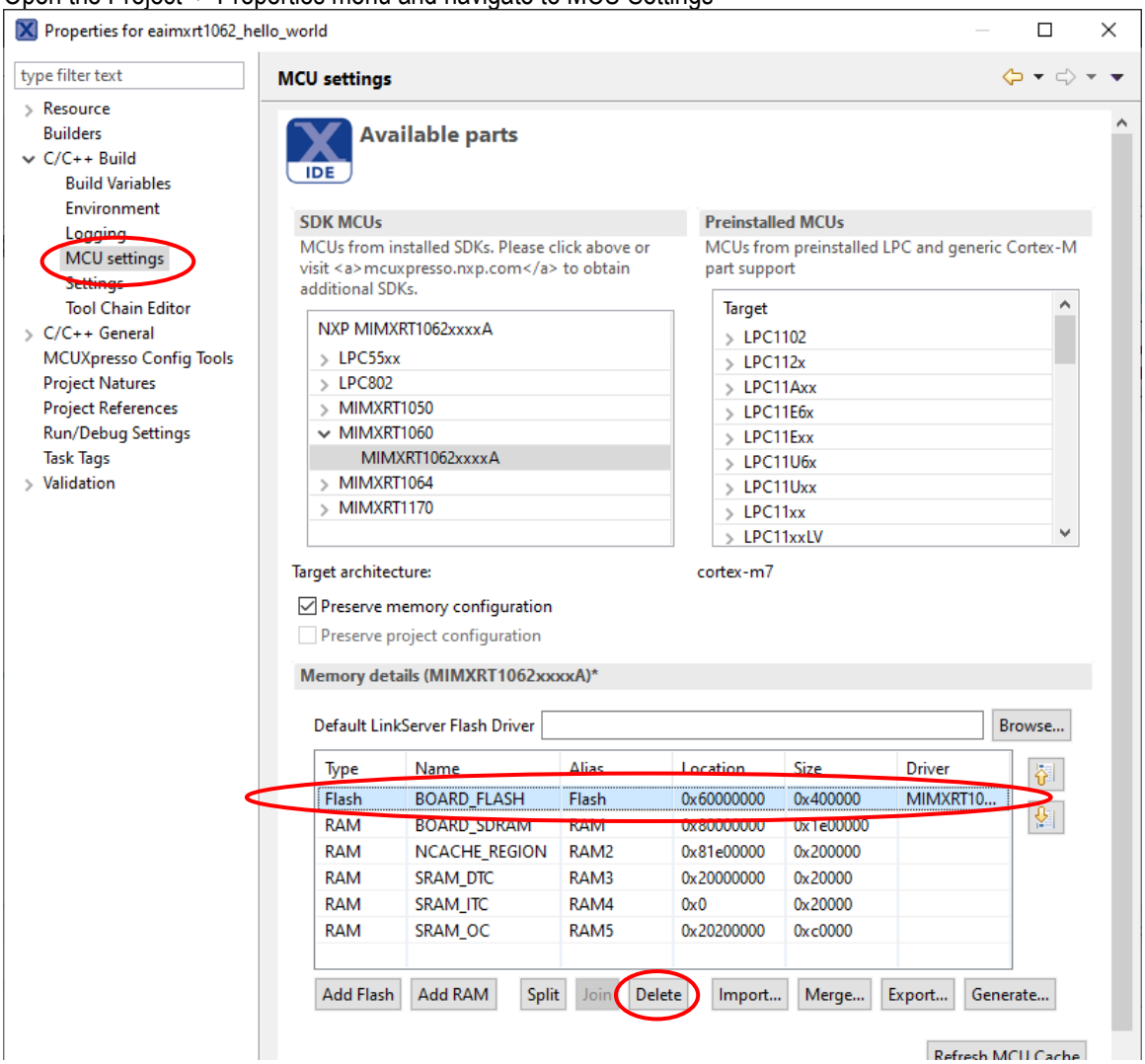
6.4 Target Memory

Almost all the examples for MCUXpresso are setup to run from flash. If the flash is used, or not, can be seen when compiling:



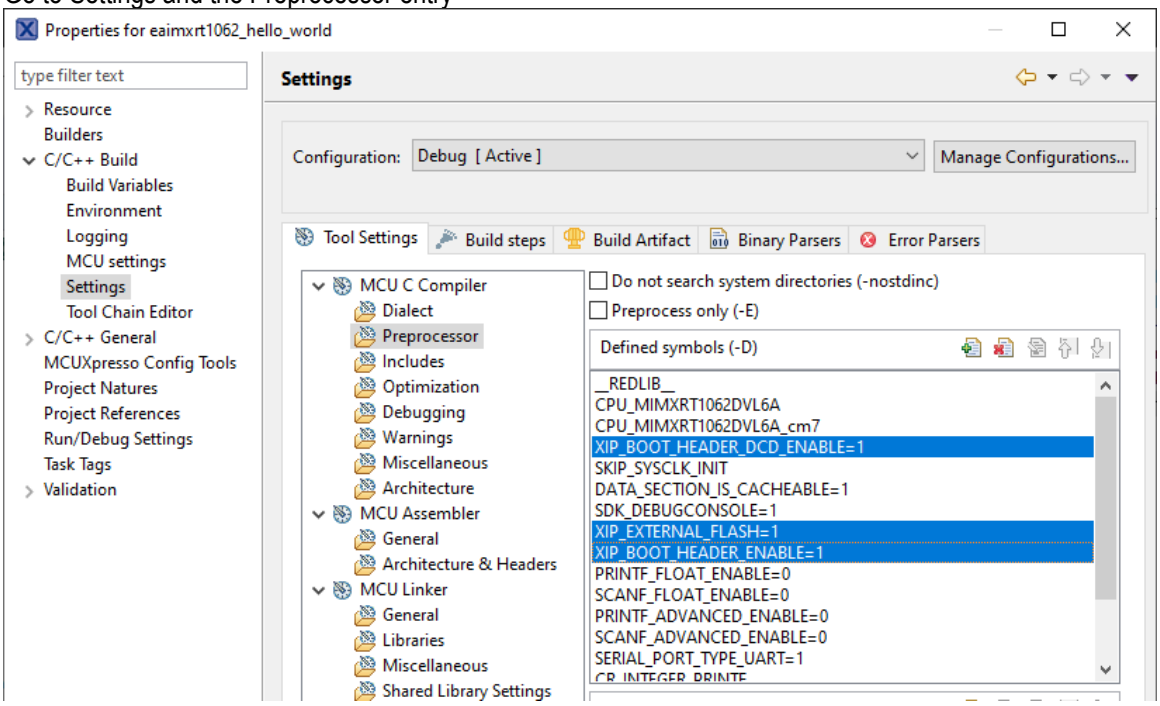
To run the program in RAM/SDRAM (if it is small enough to fit):

1. Open the Project -> Properties menu and navigate to MCU Settings



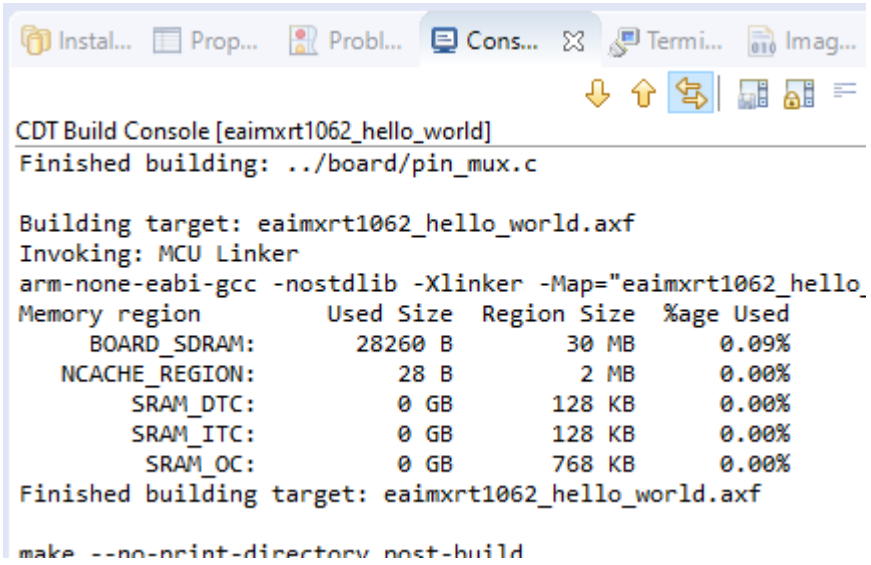
2. Select the BOARD_FLASH row in the table and then click the Delete button to remove the flash.

3. Go to Settings and the Preprocessor entry



4. There are three symbols that must be changed. The symbols are XIP_BOOT_HEADER_DCD_ENABLE, XIP_EXTERNAL_FLASH and XIP_BOOT_HEADER_ENABLE. Double-click each one and change the value from 1 to 0 to disable the feature.

5. Build the project and look at the output. The BOARD_FLASH is no longer present and instead the SDRAM is used:



7 Debug Interface

It is strongly recommended to use a debug/JTAG probe during program development. The low-cost MCU-Link or LPC-Link2 are excellent choices. Keil ULINK2 and ULINKplus, as well as Segger JLINK, are also excellent debug probes.

There are two debug interface connectors available on the *iMX OEM Carrier board*:

- J10 – this is a Cortex Debug connector. It is a 2x5 pos, 50 mil pitch connector without a shroud. Be careful when inserting the debug probe cable. Position 1 is specifically marked on the PCB silkscreen. It is located in the lower right corner, see Figure 4 below.
  The connector supports both the SWD and JTAG interfaces.
- J11 – this is an ARM Debug connector. It is a 2x10 pos, 100 mil pitch connector with shroud.

Both connectors are defined and support both the SWD and JTAG type of debug interfaces.

Note that to enable the JTAG/SWD interface on the i.MX RT, JP5 shall **not** be shorted/inserted.

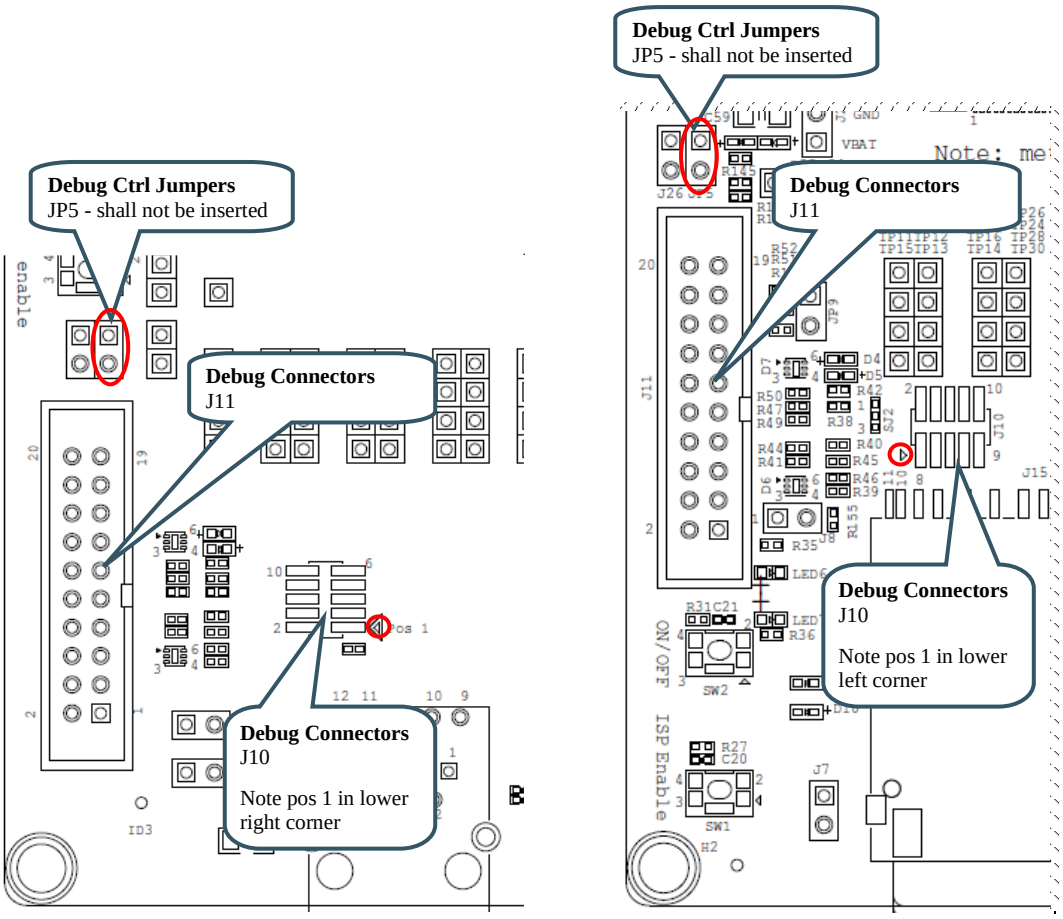


Figure 4 – Debug Interfaces on rev A and rev B1 iMX OEM Carrier board, respectively

Note that due to the powering sequencing requirements on the i.MX RT family, the debug probe I/O voltage **MUST** follow the i.MX RT I/O voltage.

The debug adapter must not drive any output higher than the Vcc/Vref voltage (and if that voltage is zero, then the debug adapter must not drive any output signal). Vcc/Vref is pin 1 on both J10 and J11.

Make sure the debug probe does not have a fixed output voltage, but rather follow Vcc/Vref. If using LPC-Link2 as debug interface, make sure there is **NO** jumper inserted in JP2 on the LPC-Link2.

7.1 J-LINK/J-TRACE Support

This section describes the steps necessary to get the Segger J-TRACE to work with NXP MCUXpresso and Keil uVision. The same instructions are likely to work for Segger J-LINK as well, but it has not been verified.

7.1.1 Install J-LINK Software

Use version v6.90a or later to get the best support for J-TRACE/J-LINK. The latest version can be found here: https://www.segger.com/downloads/jlink/.

7.1.2 MCUXpresso

Build and then launch the debugger. MCUXpresso will detect the J-LINK / J-TRACE and configure itself correctly.

7.1.3 Keil uVision

All projects have been configured to use CMSIS-DAP as debug hardware. When switching to J-LINK/J-TRACE it is important that the flash algorithm is changed to one of:

- **For iMX RT1062 OEM boards with revision >= C1**
  MIMXRT106x 8mB QuadSPI NOR Flash and that the flash size is changed to 0x01000000.
- **All other iMX RT1052/1062 boards**
  MIMXRT105x EcoXiP Flash and that the flash size is changed to 0x00400000. Note that the same flash driver shall be used for both iMX RT1052 and RT1062.

Change the Debugger from the default CMSIS-DAP to J-LINK / J-TRACE Cortex, as shown in picture below:

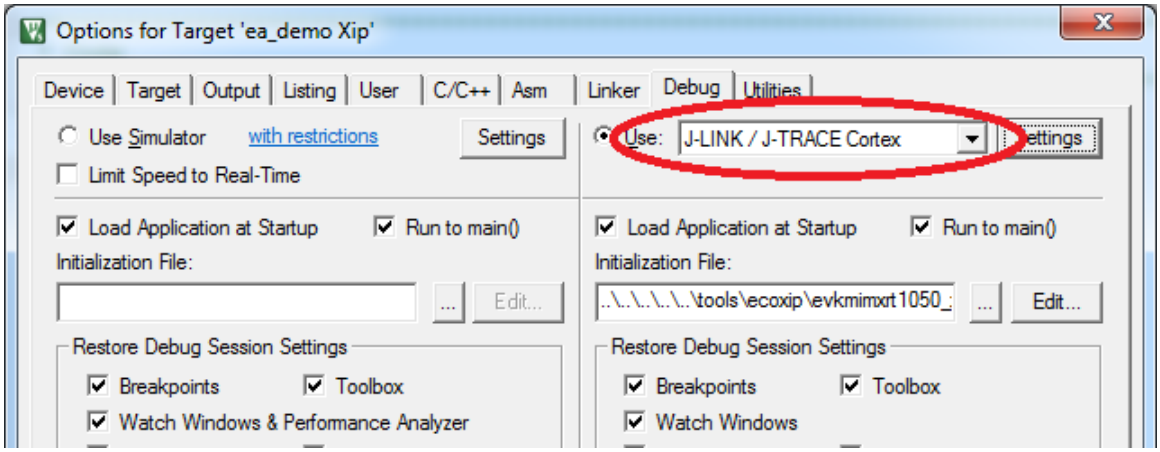


Figure 5 – Setting Debug Interface in Keil uVision

Open the settings dialog and change to the following settings:

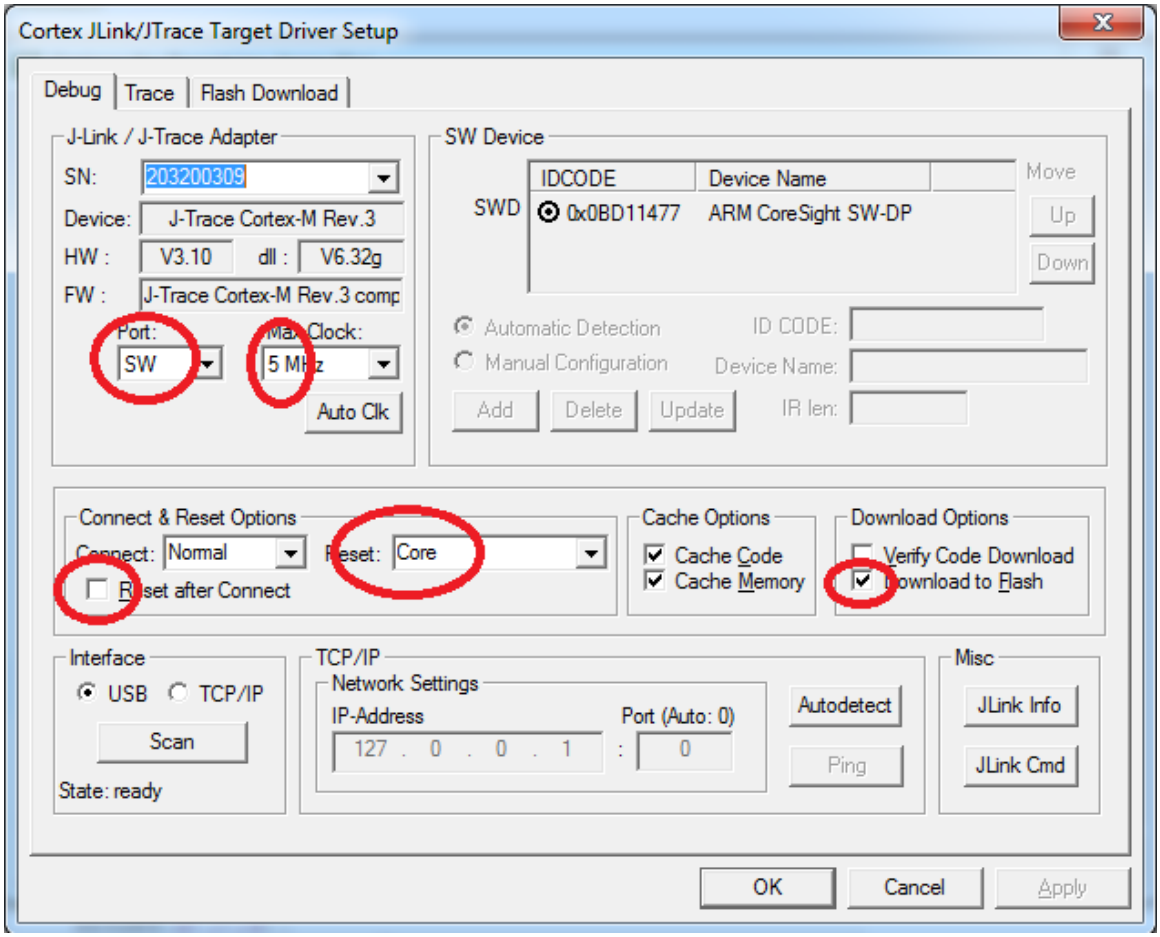


Figure 6 – Configuring J-TRACE/J-LINK Interface in Keil uVision

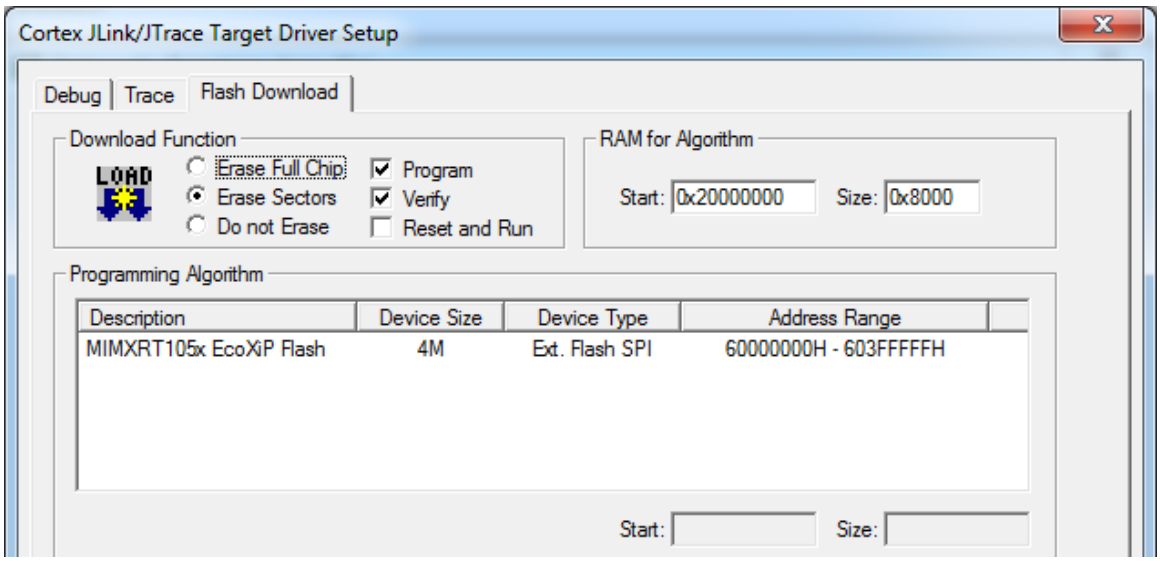


Figure 7 – Configuring Flash Programming for J-TRACE/J-LINK Interface in Keil uVision

7.2 Debug Troubleshooting

In some cases, the IDE complains about not being able to connect to the target. This is most likely because the program already running on the target is interfering. The solution is to put the hardware in ISP mode before starting the flash/debug operation in the IDE. To do this:

1. Push and hold down the ISP enable button
2. Press the Reset button
3. Release the Reset button
4. Wait 1 seconds
5. Release the ISP enable button

If the LPC-Link2 debugger is used, then there are some additional things to note:

1. Make sure that the J2 jumper on the LPC-Link2 is not inserted. If the jumper is inserted/closed then the target will be powered by the LPC-Link2 which might be too much power for the USB port that the LPC-Link2 is connected to.
2. If the LPC-Link2 is not found by the IDE and you are working on a laptop, then try using a powered USB hub instead.
3. The troubleshooting section in this forum post has a couple of additional things to try: https://community.nxp.com/thread/388964
4. There is a *Using and troubleshooting LPC-Link2* in the *Appendix - Additional Hints and Tips* of the User Guide for MCUXpresso IDE. The location of the document is c:\nxp\MCUXpressoIDE_11.0.0_2516\MCUXpresso_IDE_User_Guide.pdf if the IDE was installed with the default settings (correct path for your specific version of MCUXpresso).

Note that MCUXpresso does not support the "Restart" functionality when debugging. This issue has been reported to NXP. "Restart" works on Keil/ARM uVision/MDK and IAR Embedded Workbench.

8 Standalone Program Download

This chapter describes how to download an application to the *iMX RT OEM board* without using the IDE. Note that this section does not describe how to create the application code (create the application, compile and link it). It is assumed that a binary file exists that represent the application program.

As a reminder, there are two basic methods for program download:

- **ISP over USB Program Download**
  ISP is short for In-System Programming. The i.MX RT MCU contains a bootloader in ROM that can be enabled by pressing the ISP Enable push-button.
  An application (*MCUXpresso Secure Provisioning Tools*) provided by NXP is needed on the PC for downloading and flashing the application code. It is this method that will be described in this chapter.
  - This method of programming is useful during production
  - The *MCUXpresso Secure Provisioning Tools* application is needed to generate an authenticated or Encrypted image of the application.
  - Technically it is possible to program/flash the OctalSPI/QuadSPI flash without a JTAG probe (via NXP's *MCUXpresso Secure Provisioning Tools* application), but it is strongly recommended to use the proper tool for debugging - i.e., use a JTAG probe!
- **SWD/JTAG Debug Interface**
  Using this method, the application can be downloaded to internal SRAM, to external SDRAM or external OctalSPI/QuadSPI flash.
  This method is tightly integrated with the Integrated Development Environment (IDE) used. Specific scripts (and sometimes flash programming algorithms) must exist for the used IDE. Currently such scripts and drivers exist for Keil uVision/MDK, NXP MCUXpresso and IAR Embedded Workbench. For other IDEs, check supported functions.
  - There are many different SWD/JTAG interfaces on the market. NXP has created the low-cost MCU-Link and LPC-LINK2, Keil has ULINK2/ULINKpro, Segger has J-LINK, etc.

8.1 Install and Patch the Required Software

First, download *MCUXpresso Secure Provisioning Tools* from NXP's website. It can be found under the *Tools & Software* tab for each MCU or directly here. The tool is available for Windows, MacOS and Linux but this document only covers the Windows use case. Patching and using the MacOS and Linux versions should be very similar with only paths and file names varying.

The default installation location is **c:\nxp\MCUX_Provi_v5** which will be referred to as <install directory> from now on.

There is one thing that must be patched in the tool itself before it can be used. Download the zip-file from http://imx.embeddedartists.com. It will have a filename like

`imxrt_secure_provisioning_<date>.zip`

Unpack the downloaded file and move EA_iMXRT10x2_DevKit_4MB_OctalSPI_ATXP032 to

<install directory>\bin\data\boot_devices\devices\flex-spi-nor\

The *MCUXpresso Secure Provisioning Tool* is now ready to be used.

8.2 Prepare the Program to Flash

A program can be setup to run directly in external nor flash, in internal SRAM or in (external) SDRAM and the project must be modified accordingly. This is described in detail in section 7.2.1 of the *User Manual - MCUXpresso Secure Provisioning Tools* that comes with the installation of the tool (filename: MCUXpresso Secure Provisioning Tools.pdf) or it can be viewed here: https://www.nxp.com/docs/en/user-guide/MCUXSPTUG.pdf

8.3 Booting an Unsigned Image

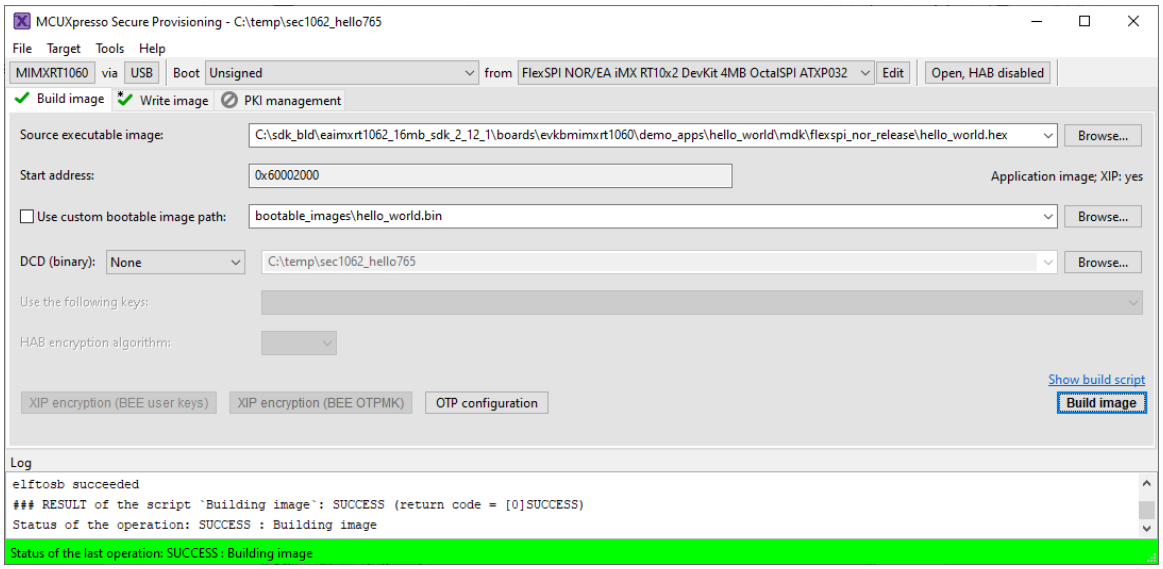
Unsigned images are typically used for development. It's recommended to start with this boot type before working with secured images to verify that the executable image works properly.

The first step is to convert the prepared application into a bootable image:

1. Select the correct processor
2. Make sure that Boot Type is *Unsigned*
3. Set Boot Device:

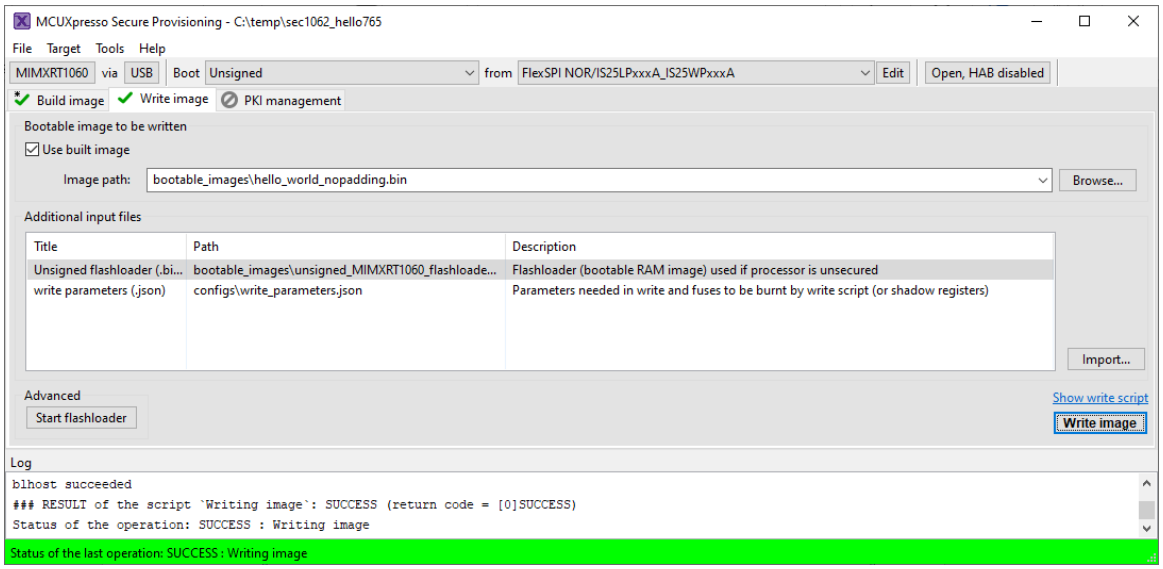
OEM Board	Boot Device
iMX RT1052 OEM, any revision	*flex-spi-nor/EA iMX RT10x2 DevKit 4MB OctalSPI ATXP032*
iMX RT1062 OEM, revision <= rev B2	*flex-spi-nor/EA iMX RT10x2 DevKit 4MB OctalSPI ATXP032*
iMX RT1062 OEM, revision >= rev C1	*flex-spi-nor/ IS25LPxxxA_IS25WPxxxA*

4. Switch to the *Build Image* tab
5. Select the Source executable image that was prepared in the previous section
6. If (and only if) the application uses SDRAM, select the *Use custom DCD* option and point to the dcd_sdram.bin file from the <install directory>\bin\data\targets\mimxrt*\ folder
7. Click *Build Image*



To write the image:

1. Switch to the *Write Image* tab
2. Make sure that the *Use built image* option is selected
3. Connect the hardware to the PC using micro-B to A USB cables in both J12 and J22 connectors
4. Put the hardware in ISP mode:
    a. Push and hold down the ISP enable button
    b. Press the Reset button
    c. Release the Reset button
    d. Wait 1 seconds
    e. Release the ISP enable button
5. Press the *Write Image* button
6. When finished, press the Reset button on the hardware to run the program



8.4 Booting an Authenticated or Encrypted Image

The *MCUXpresso Secure Provisioning Tools* support authenticated (signed) and encrypted (signed+encrypted) images. This is described in detail in the User Manual for the tool. A couple of **very important notes**:

1. Encrypted images cannot be used for applications that execute directly in the flash (XiP).
2. Burning the fuses in the processor is an irreversible operation. If the fuses are burned and the key is lost, then there is no way to burn anything again on that hardware so make sure to backup the (generated) keys **BEFORE** burning them to the hardware.
3. If the instructions mention setting the SW7 DIP to the board to 0001 it means putting the board into ISP mode. This is done on the Developers Kit by:
    a. Push and hold down the ISP enable button
    b. Press the Reset button
    c. Release the Reset button
    d. Wait 1 seconds
    e. Release the ISP enable button

9 Terminal Application Setup

This chapter contains information about the terminal connection that exists on the *iMX OEM Carrier Board*, and how to setup a terminal application on the PC. The terminal connection connects UART1 of the i.MX RT to a virtual COM port over the USB interface available on J22. The terminal is commonly used during program development.

9.1 UART-to-USB Bridge

The UART-to-USB bridge chip (FT230XS-R from FTDI) on the *iMX OEM Carrier Board* connects to UART channel #1 on the i.MX RT. It exists to simplify connection to a PC because serial ports are not so common anymore, especially not on laptops. The USB port also offers the possibility to power the board.

There are two LEDs, transmit from the board (LED11) and receive to the board (LED10), that signal communication activity.

See picture below for locating relevant components.

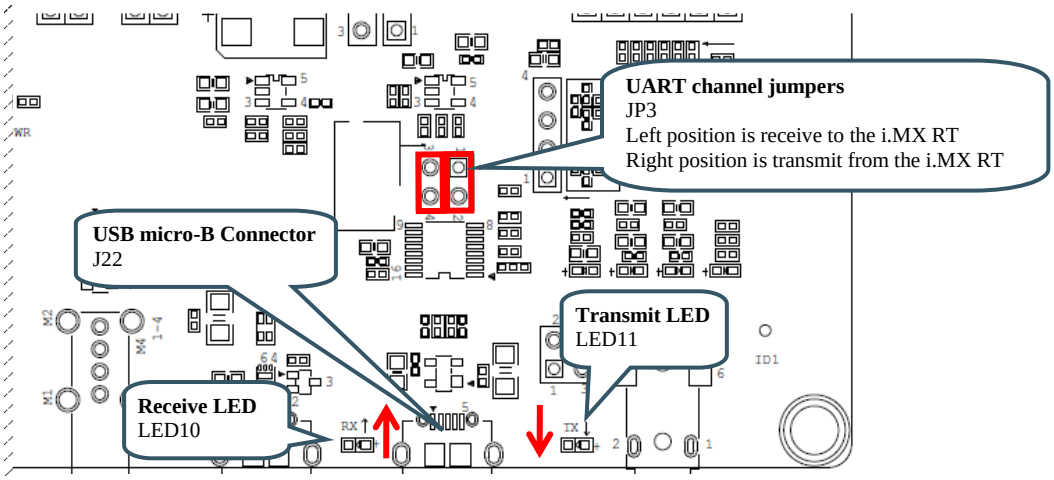


Figure 8 – UART-to-USB Bridge

9.2 Terminal Application on the PC

Begin by connecting the micro-B USB connector to J22, see picture above. Connect the other end of the USB cable to the PC. The PC will typically immediately begin installing drivers automatically for the UART-to-USB bridge that creates a Virtual COM port, if they are not already installed. If you have problems the drivers can be downloaded from the links below:

http://www.ftdichip.com/Drivers/VCP.htm
http://www.ftdichip.com/Support/Documents/InstallGuides.htm

When the driver has been installed, a new COM port will list under "Ports" in the Device Manager as shown in Figure 9. Please note that the actual port number will most likely be different on your computer.

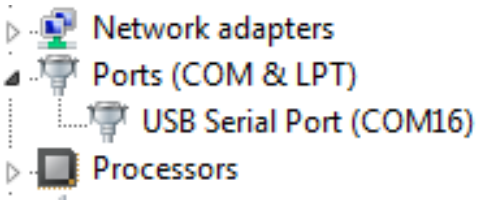


Figure 9 – Virtual COM port shown in device manager

The next step is to open a terminal application and attach it to the Virtual COM port that has just been created. The baud rate should be **115200**.

Some development environments/IDEs have a built-in terminal application that can be used. Sometimes it is better to have a terminal application with more features. For increased flexibility, we recommend using any of the two alternative terminal applications presented in the following sub-sections.

9.2.1 Tera Term Terminal Emulation Application

We recommend that you use **Tera Term** which can be downloaded and installed from either of the links below.

https://ttssh2.osdn.jp/index.html.en
http://sourceforge.jp/projects/ttssh2/releases/

Launch *Tera Term*. The first time it launches, it will show you the following dialog. Select the serial option. Assuming the USB cable is connected to the *iMX OEM Carrier Board*, there should be a COM port automatically populated in the list.

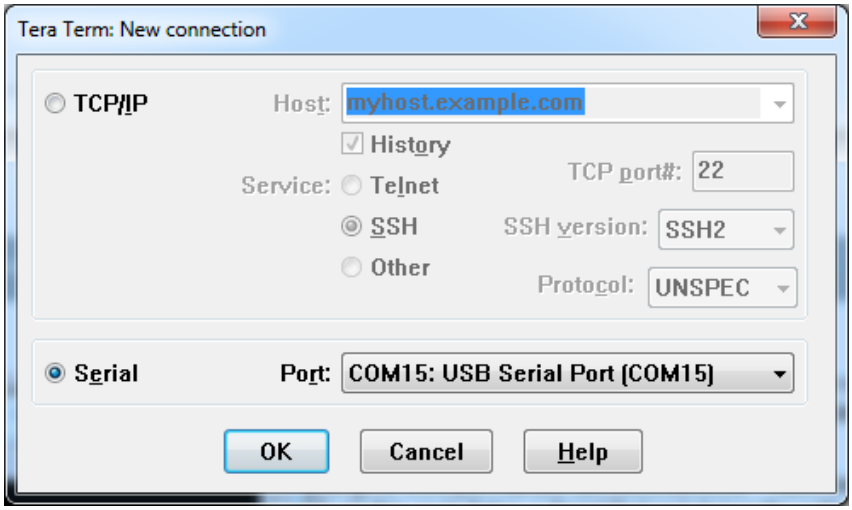


Figure 10 – Tera Term New Connection Window

Configure the serial port settings (using the COM port number identified earlier) to 115200 baud rate, 8 data bits, no parity and 1 stop bit. To do this, go to Setup → Serial Port and change the settings.

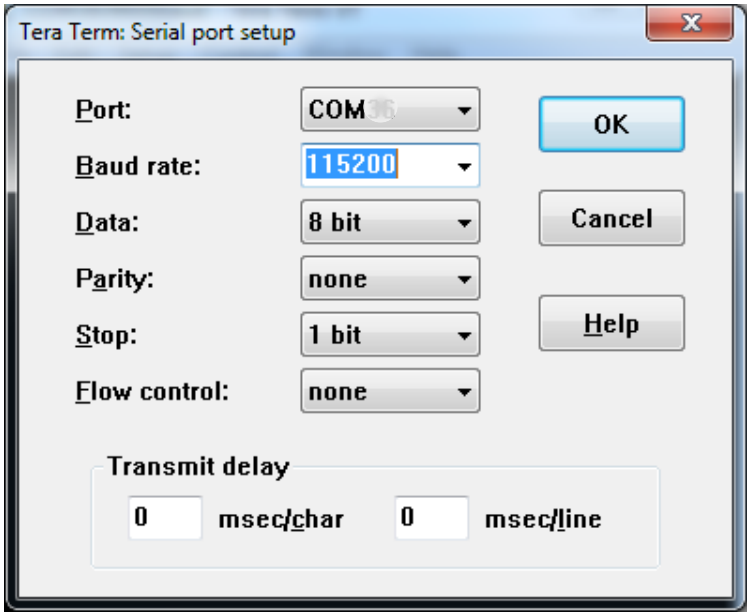


Figure 11 – Tera Term Serial Port Setup

Verify that the connection is open. If connected, *Tera Term* will show something like below in its title bar.

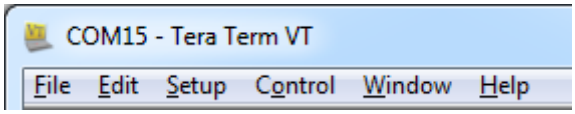


Figure 12 – Tera Term Menu

9.2.2 PuTTY terminal emulation application

Alternatively, you can use **PuTTY**. It is another commonly used terminal emulation application. PuTTY can be downloaded and installed from the link below.

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

Launch PuTTY by either double clicking on the *.exe file you downloaded or from the Start menu, depending on the type of download you selected.

In the window that launches, select the Serial radio button and enter the COM port number that you determined earlier. Also enter the baud rate, in this case 115200.

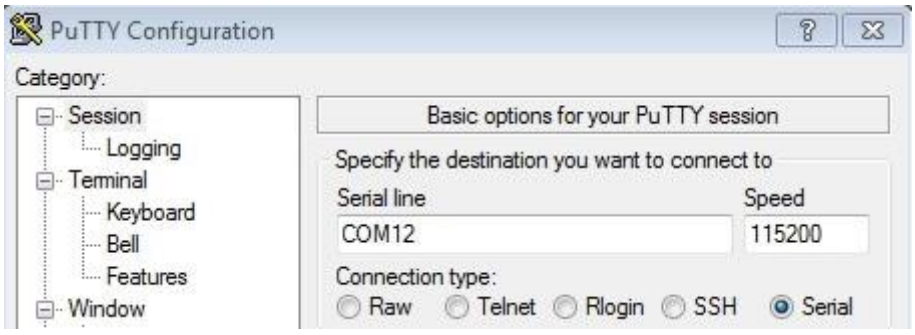


Figure 13 – PuTTY New Session Configuration

Click Open to open the serial connection. Assuming the FTDI cable is connected, and you entered the correct COM port, the terminal window will open. If the configuration is not correct, PuTTY will alert you.

10 Booting from External Memory

The i.MX RT does not have any internal flash memory for storing the application. It must be stored in an external memory. On the *iMX RT OEM boards*, with revision up to, and including rev B2 this external memory is a 4 MByte OctalSPI, called EcoXiP ATXP032 from Adesto Technologies. On rev C1 boards, and later, the external memory is a 16 Mbyte QuadSPI flash, called IS25WP128. The i.MX RT MCU always boots (i.e., starts executing) from this OctalSPI/QuadSPI memory.

First, let's investigate the three use-cases when executing an application. The picture below illustrates the first main use-case when executing an application.

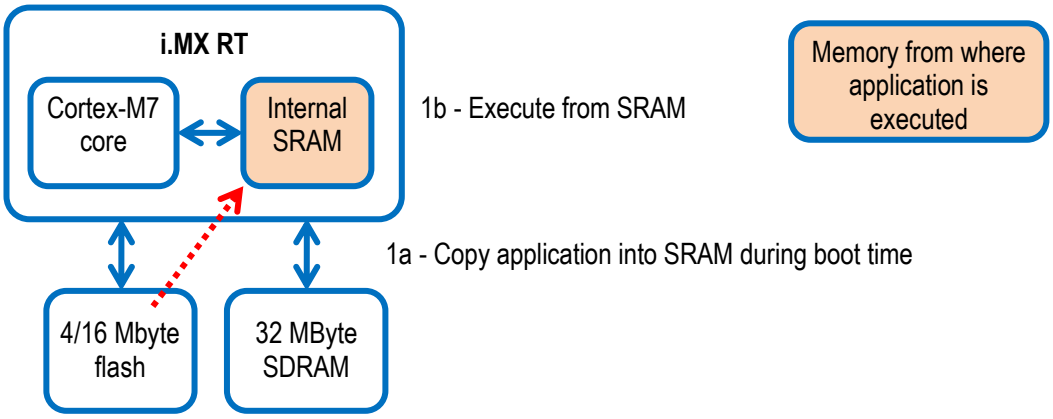


Figure 14 – i.MX RT and External Memories - Executing from SRAM

1. The application is stored in the OctalSPI/QuadSPI flash and the bootloader (inside the i.MX RT) copies it into internal SRAM and then run from there.
    a. The execution performance will be the highest in this setup.
    b. CoreMark score (about) 2600 / 3000 when running at 528 / 600 MHz
    c. During program development the application is just downloaded to internal SRAM by the debugger. There is no need to first download the application to the OctalSPI/QuadSPI flash memory. The address (in SRAM) where the application is downloaded is the same that it will be copied to by the on-chip bootloader in a final deployed system.

The second use-case is illustrated below. It is the default option supported when compiling and building the XiP targets.

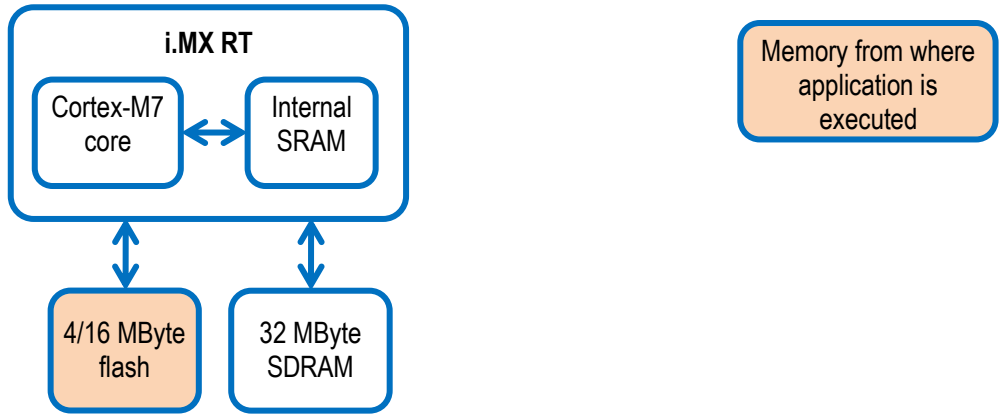


Figure 15 – i.MX RT and External Memories - Executing from External Flash

2. The application is stored in the OctalSPI/QuadSPI flash and also executed from there. In this case, the internal SRAM is probably too small for the application or is simply used for other things like data storage.
    a. When executing from an OctalSPI flash, the execution performance will be about 2/3 of the performance when executing from internal SRAM (assuming 5% cache miss, which is typical). CoreMark score is about 1736 / 1912 when running at 528 / 600 MHz
    b. When executing from a QuadSPI flash, the performance is about half that of the OctalSPI flash.
    c. During program development the application must be downloaded/flashed to the OctalSPI/QuadSPI memory before debugging actually starts. This is normally handled automatically by the IDE (Integrated Development Environment).

The third use-case is just a mixture of the two main ones. Two, or more memories, are used for storing executable code.

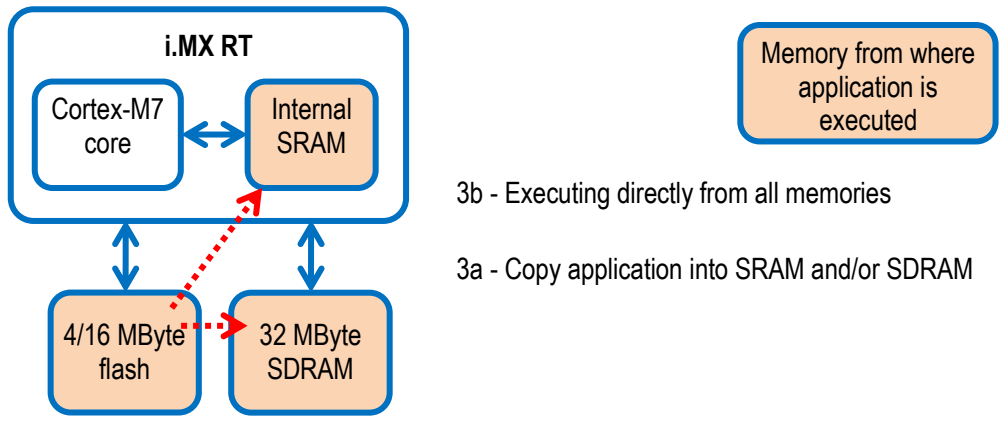


Figure 16 – i.MX RT and External Memories - Executing from all memories

3. The third setup is a mixture of the two above. Part of the application is copied into SRAM and/or SDRAM and part is executed directly from the OctalSPI/QuadSPI flash. A reason for placing part of an application in SRAM can be a need for highest performance for a data processing algorithm or a time critical interrupt service routine.
    a. Note that this is a more complicated system architecture. The application must implement a dynamic loader that can copy code from the OctalSPI/QuadSPI flash to SRAM, either on-demand or in a pre-scheduled way. The linker script can be much more complicated because of this. There is no general solution for this system solution. Every system must be individually investigated to select and implement the best solution.

11 Things to Note

11.1 ESD Precaution

Please note that the *iMX RT OEM Board* and *iMX OEM Carrier Board* come without any case/box and all components are exposed for finger touches – and therefore extra attention must be paid to ESD (electrostatic discharge) precautions.

Make it a habit always to first touch the metal surface of one of the USB, SD or Ethernet connectors for a few seconds with both hands before touching any other parts of the boards. That way, you will have the same potential as the board and therefore minimize the risk for ESD.

Never touch directly on the *iMX RT OEM Board* and in general as little as possible on the *iMX OEM Carrier Board*. The pushbuttons on the *iMX OEM Carrier Board* have grounded shields to minimize the effect of ESD.

Note that Embedded Artists does not replace boards that have been damaged by ESD.

11.2 General Handling Care

Handle the *iMX RT OEM Board* and *iMX OEM Carrier Board* with care. The boards are not mounted in a protective case/box and are not designed for rough physical handling. Connectors can wear out after excessive use. The *iMX OEM Carrier Board* is designed for prototyping use, and not for integration into an end-product.

For boards with LCD, do not exercise excessive pressure on the LCD glass area. That will damage the display. Also, do not apply pressure on the flex cables connecting the LCD/touch screen. These are relatively sensitive and can be damaged if too much pressure is applied to them.

Note that Embedded Artists does not replace boards where the LCD has been improperly handled.

11.3 OTP Fuse Programming

The i.MX RT MCU has on-chip OTP fuses that can be programmed, see NXP documents *IMXRT1050CEC, .MX RT1050 Crossover Processors for Consumer Products - Data Sheet* and *IMXRT1050RM, i.MX RT1050 Processor Reference Manual* and *IMXRT1060CEC, .MX RT1060 Crossover Processors for Consumer Products - Data Sheet* and *IMXRT1060RM, i.MX RT1060 Processor Reference Manual* for details. Once programmed, there is no possibility to reprogram them.

iMX RT OEM Boards are delivered without any OTP fuse programming. It is completely up to the OEM board user to decide if OTP fuses shall be programmed and, in that case, which ones.

Note that Embedded Artists does not replace iMX RT OEM Boards because of wrong OTP programming. It's the user's responsibility to be absolutely certain before OTP programming and not to program the fuses by accident.

11.4 Further Information

The following NXP documents are important reference documents and should be consulted for functional details:

- IMXRT1050CEC, .MX RT1050 Crossover Processors for Consumer Products - Data Sheet, latest revision
- IMXRT1050IEC, .MX RT1050 Crossover Processors for Industrial Products - Data Sheet, latest revision
- IMXRT1050RM, i.MX RT1050 Processor Reference Manual, latest revision
- IMXRT1050CE, Chip Errata for the i.MX RT1050, latest revision
  Note: It is the user's responsibility to make sure all errata published by the manufacturer are taken note of. The manufacturer's advice should be followed.
- AN12094, Power consumption and measurement of i.MXRT1050, latest revision

- IMXRT1060CEC, i.MX RT1060 Crossover Processors for Consumer Products - Data Sheet, latest revision
- IMXRT1060IEC, i.MX RT1060 Crossover Processors for Industrial Products - Data Sheet, latest revision
- IMXRT1060RM, i.MX RT1060 Processor Reference Manual, latest revision
- IMXRT1060CE, Chip Errata for the i.MX RT1060, latest revision
  Note: It is the user's responsibility to make sure all errata published by the manufacturer are taken note of. The manufacturer's advice should be followed.
- AN12245, Power consumption and measurement of i.MXRT1060, latest revision
- AN12253, i.MXRT1060 Product Lifetime Usage Estimates, latest revision

12 Disclaimers

Embedded Artists 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.

Customer is responsible for the design and operation of their applications and products using Embedded Artists' products, and Embedded Artists accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the Embedded Artists' 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. Customer is required to have expertise in electrical engineering and computer engineering for the installation and use of Embedded Artists' products.

Embedded Artists 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 Embedded Artists' 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). Embedded Artists does not accept any liability in this respect.

Embedded Artists does not accept any liability for errata on individual components. Customer is responsible to make sure all errata published by the manufacturer of each component are taken note of. The manufacturer's advice should be followed.

Embedded Artists does not accept any liability and no warranty is given for any unexpected software behavior due to deficient components.

Customer is required to take note of manufacturer's specification of used components, for example MCU, SDRAM and FLASH. Such specifications, if applicable, contains additional information that must be taken note of for the safe and reliable operation. These documents are stored on Embedded Artists' product support page.

All Embedded Artists' products are sold pursuant to Embedded Artists' terms and conditions of sale: http://www.embeddedartists.com/sites/default/files/docs/General_Terms_and_Conditions.pdf

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by Embedded Artists for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN EMBEDDED ARTISTS' TERMS AND CONDITIONS OF SALE EMBEDDED ARTISTS DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF EMBEDDED ARTISTS PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY THE CEO OF EMBEDDED ARTISTS, PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, NUCLEAR, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE.

Resale of Embedded Artists' products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by Embedded Artists

for the Embedded Artists' product or service described herein and shall not create or extend in any manner whatsoever, any liability of Embedded Artists.

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

12.1 Definition of Document Status

Preliminary – The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. Embedded Artists does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information. The document is in this state until the product has passed Embedded Artists product qualification tests.

Approved – The information and data provided define the specification of the product as agreed between Embedded Artists and its customer, unless Embedded Artists and customer have explicitly agreed otherwise in writing.