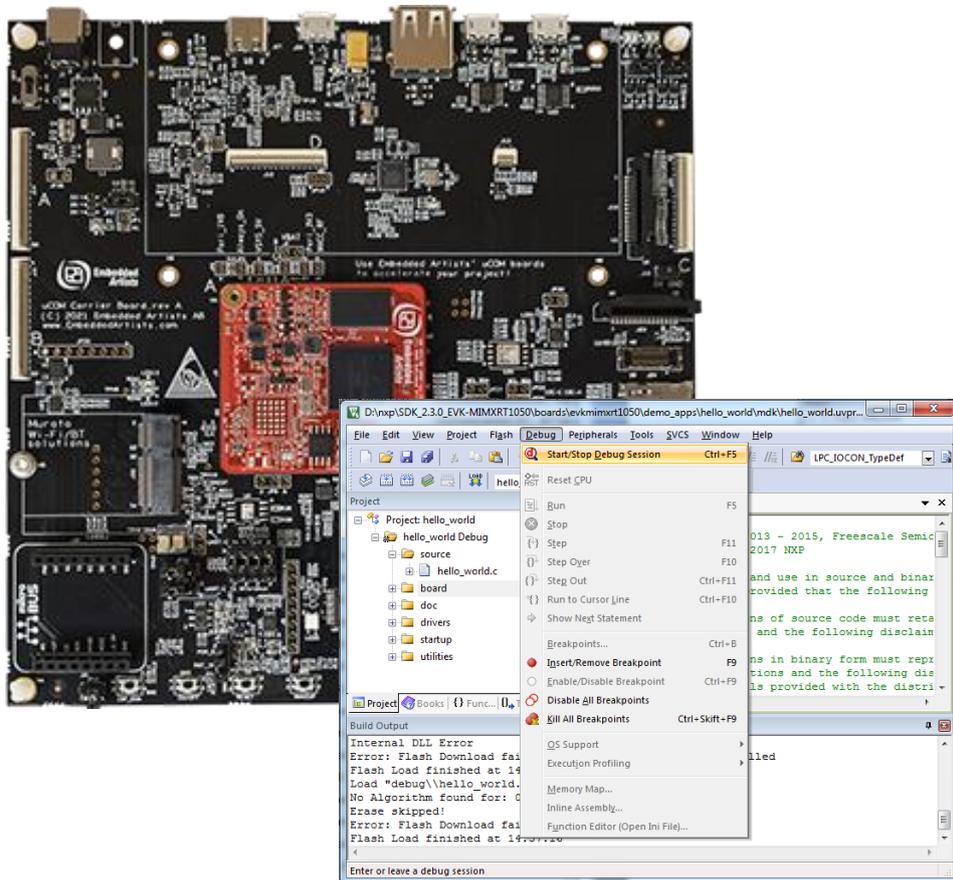


iMX RT Developer's Kit Program Development Guide



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

Embedded Artists AB

Rundelsgatan 14
211 36 Malmö
Sweden

<https://www.EmbeddedArtists.com>

Copyright 2021 © Embedded Artists AB. All rights reserved.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of Embedded Artists AB.

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	5
2	Get Started with Program Development	6
2.1	Downloading the SDK	6
2.2	About the SDK	7
2.3	Preparing the Hardware	7
2.4	Getting Started with a Specific IDE	7
3	Getting Started with Keil uVision/MDK	9
3.1	Install the SDK	9
3.2	Install CMSIS Device Pack	9
3.3	Build an Example Application	10
3.4	Run an Example Application	11
4	Getting Started with IAR Embedded Workbench	13
4.1	Install the SDK	13
4.2	Build an Example Application	13
4.3	Run an Example Application	14
5	Getting Started with NXP MCUXpresso IDE	16
5.1	Install the SDK	16
5.2	Build an Example Application	16
5.3	Run an Example Application	18
5.4	Target Memory	19
5.5	Troubleshooting	23
6	Debug Interface	24
6.1	J-LINK/J-TRACE Support	24
6.1.1	Install J-LINK Software	24
6.1.2	MCUXpresso	24
6.1.3	Keil uVision	25
7	Standalone Program Download	27
7.1	Install the Required Software	27
7.2	Prepare the Program to Flash	27
7.3	Booting an Unsigned Image	28
7.4	Booting an Authenticated or Encrypted Image	30
8	Terminal Application Setup	31
8.1	UART-to-USB Bridge	31
8.2	Terminal Application on the PC	31
8.2.1	Tera Term Terminal Emulation Application	32
8.2.2	PuTTY terminal emulation application	33

9	Booting from External Memory	35
10	Troubleshooting	38
10.1	Cannot debug / download	38
11	Things to Note	40
11.1	ESD Precaution	40
11.2	General Handling Care	40
11.3	OTP Fuse Programming	40
12	Disclaimers	41
12.1	Definition of Document Status	42

1 Document Revision History

<i>Revision</i>	<i>Date</i>	<i>Description</i>
PA1	2020-03-03	First release
PA2	2021-03-07	Moved troubleshooting section
PA3	2021-12-02	Updated Standalone Program Download section. Minor version updates.

2 Get Started with Program Development

This chapter contains information about how to get started with program development on the iMX RT Developer's Kit. The document covers the following kits:

Kit (and MCU)
iMX RT1064 Developer's Kit
iMX RT1166 Developer's Kit
iMX RT1176 Developer's Kit

Note that the *iMX RT1052/RT1062 OEM Developer's Kit* is not covered in this document. There is a separate document for that kit.

Most of the examples in this document refer to the *iMX RT1176 Developer's Kit* (and associated *iMX RT1176 uCOM* board). For other kits/MCUs, the paths and names may change slightly but it will be obvious where the changes are.

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**. Update to latest version of respective IDE.
 - b. Programming of the 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 flash memory and in general to be able to debug - set breakpoints, inspect memory, etc.
 - a. The low-cost MCU-Link (Pro) 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 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

`ea<mcu>_sdk_<version>_<date>.zip`

2.2 About the SDK

NXP's SDK builder (<https://mcuxpresso.nxp.com/en/>) was used to generate an SDK.

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** and that is the combination that was selected for the patched SDK. SDK builder for 2.8.6 forced a choice between FreeRTOS or Azure RTOS and in that case FreeRTOS was selected. Starting with 2.9.0 the SDKs have support for both operating systems.

In general, this is what has been patched (see the release notes for more detail):

- The memory configuration (e.g., size of SDRAM) is changed to match the iMX RT Developer's Kit
- The flash configuration (e.g., size/model/make) is changed to match the iMX RT Developer's Kit. This includes changes to selected flash algorithms and flash headers
- Use of I2C busses, UARTs and pinning is changed where needed
- Buttons and LEDs are remapped to work on the iMX RT Developer's Kit
- Drivers for peripherals unique to the iMX RT Developer's Kit are added. This includes for example
 - GPIO Expander PCA6416
 - PWM Expander PCA9530
 - EEPROM for MAC address
- LWIP projects are modified to read the MAC address

This has been added:

- Wi-Fi/Bluetooth support is extended to support a larger selection of Embedded Artists' M.2 wireless modules.
- HDMI display support is added if it can be supported by the MCU
- Parallel RGB display support is added if it can be supported by the MCU

This has been removed:

- All projects for the expansion board AGM01

2.3 Preparing the Hardware

Please follow the instructions on the getting started page for the iMX RT Developer's Kit to power the board and connect to the console (UART-to-USB bridge).

<https://www.embeddedartists.com/getting-started-with-ucom-developers-kit/>

2.4 Getting Started with a Specific IDE

The following chapters will describe how to get started with a number of different IDEs.

Chapter 3 describes Keil uVision/MDK.

Chapter 4 describes IAR Embedded Workbench.

Chapter 5 describes NXP MCUXpresso.

3 Getting Started with Keil uVision/MDK

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

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 too long path can cause problem 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

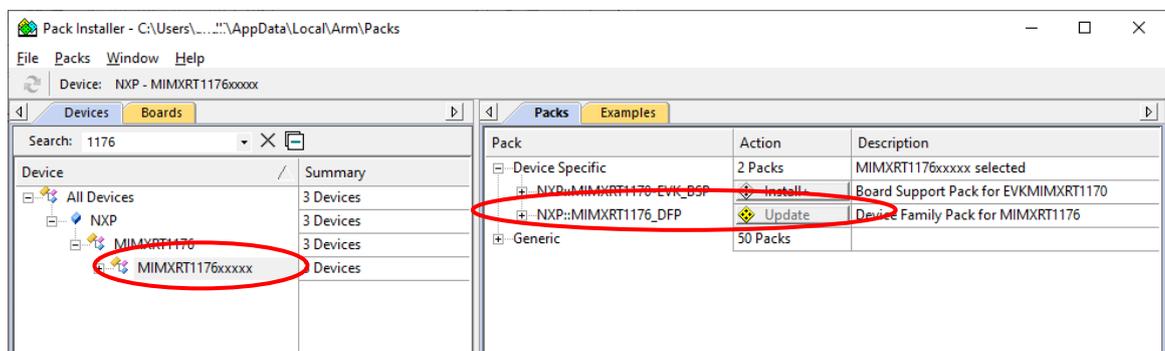
After the MDK tools are installed, Cortex Microcontroller Software Interface Standard (CMSIS) device packs must be installed to fully support the device from a debug perspective. These packs include things such as memory map information, register definitions and flash programming algorithms.

In most cases, if you open one of the projects from the unpacked SDK in Keil uVision, it will detect if the CMSIS pack is needed and ask if it should be downloaded/installed. Accept it and let it be installed.

Note that this could take a long time (~20 minutes) but it is a onetime operation so don't abort and let it complete.

In the rare cases where Keil uVision cannot detect the package automatically for you, follow these steps to install the CMSIS pack manually.

1. Start Keil uVision
2. Start the Pack Installer tool with this button on the toolbar: 
3. Browse to the device tree to the left and locate the device (for example NXP→MIMXRT1176→MIMXRT1176xxxxx or similar for other MCUs) and then locate the package in the right side with a `_DFP` in the name and click the Install button next to it. Note that the button in the image below has "Update" as the package has already been installed.



4. Note that this operation can take a very long time to complete (~20 minutes) so don't abort if it looks like the installer has frozen.

3.3 Build an Example Application

The following steps will guide you through opening the hello_world application for the Cortex-M7 core of the iMX RT1176. These steps may change slightly for other MCUs and other example applications as some of these applications may have additional layers of folders in their path. Note that the Keil uVision workspace files are always placed in subfolders named **mdk/** so look for that.

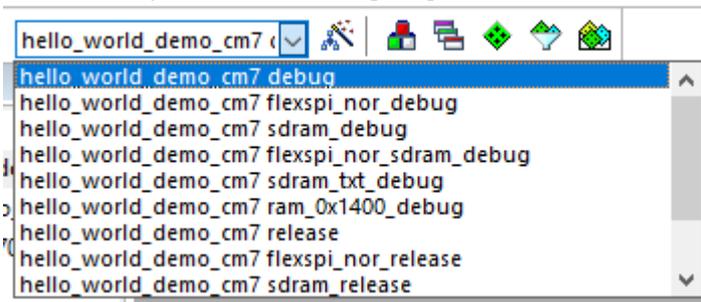
1. Open the desired example application either by double clicking the .uvmpw file in the explorer or by starting Keil uVision and selecting Project->Open Project... from the menu. The hello world example has this path:

```
<install_dir>boards/evkmimxrt1170/demo_apps/hello_world/cm7/
mdk/hello_world_demo_cm7.uvmpw
```

2. Customize the project. For projects using a display this includes selecting which display to use, for networking projects it may include setting the IP number/netmask, for Wi-Fi projects it includes selecting which M.2 module to use and which Wi-Fi network to connect to.

These settings are handled differently for different SDK versions and different MCUs so consult the readme.txt file available in the docs/ folder of each example for information.

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



The target name indicates from where the application is executing, see table below. Release targets have a higher compiler optimization degree making the target less suitable for debugging. Available targets differ from example to example but here are some of the more common ones.

<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 before use.
<project> flexspi_nor_sdram_debug <project> flexspi_nor_sdram_release	The application runs in external flash and use SDRAM. The external flash must be programmed before use.

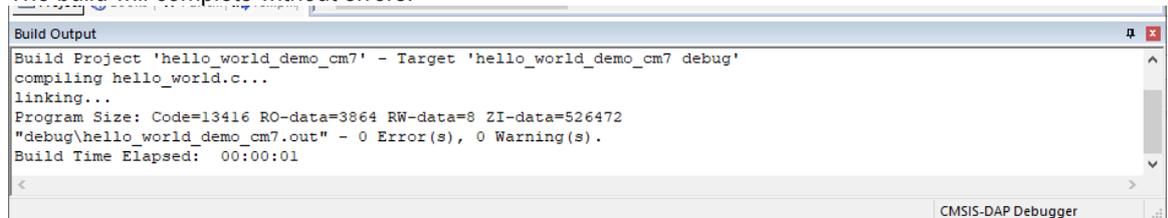
Note that each of the targets have its own set of options so changing for example include path in

one target does not affect the other targets.

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



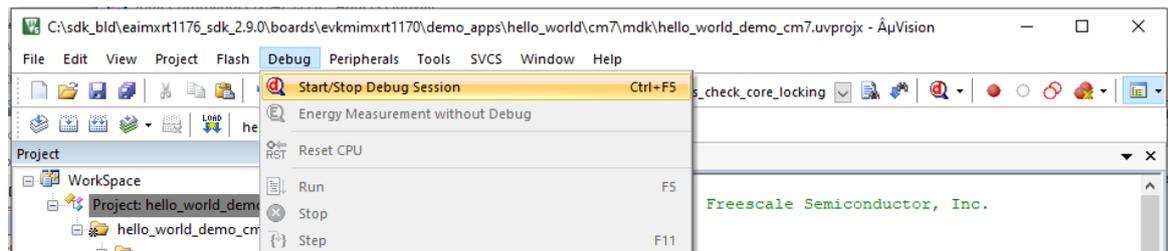
- The build will complete without errors.



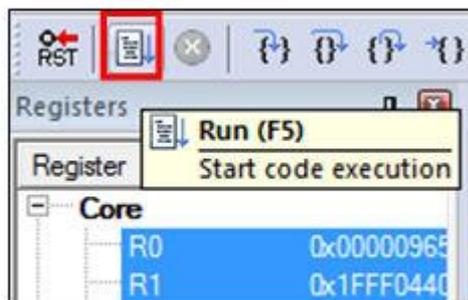
3.4 Run an Example Application

To download and run the application, perform these steps:

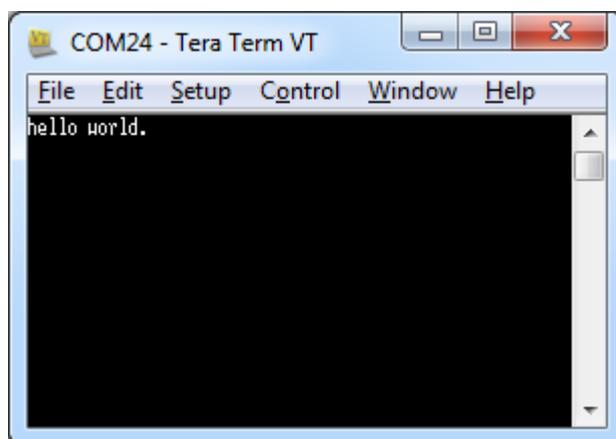
- Prepare the hardware as described in chapter 2.3 and power it on.
- Open the terminal application on the PC, such as TeraTerm or PuTTY, and connect to the virtual COM port. Configure the terminal with 115200 baud, 8N1. You can alter the baud rate by searching for the BOARD_DEBUG_UART_BAUDRATE define in file: **board.h**
- Select the *Debug* menu and then *Start/Stop Debug Session*, or simply press Ctrl+F5. The application will then be downloaded into SRAM.



- Run the code by clicking the "Run" button (or press 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 Getting Started with IAR Embedded Workbench

This section is a guide to open, build and debug an example application from the SDK that was downloaded in 2.1 using IAR Embedded Workbench. It is assumed that you have this development environment installed on your computer.

4.1 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 too long path can cause problem in Windows.

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

4.2 Build an Example Application

The following steps will guide you through opening the `hello_world` application for the Cortex-M7 core of the iMX RT1176. These steps may change slightly for other example applications as some of these applications may have additional layers of folders in their path. Note that the IAR Embedded Workbench workspace files are always placed in subfolders named `iar/` so look for that.

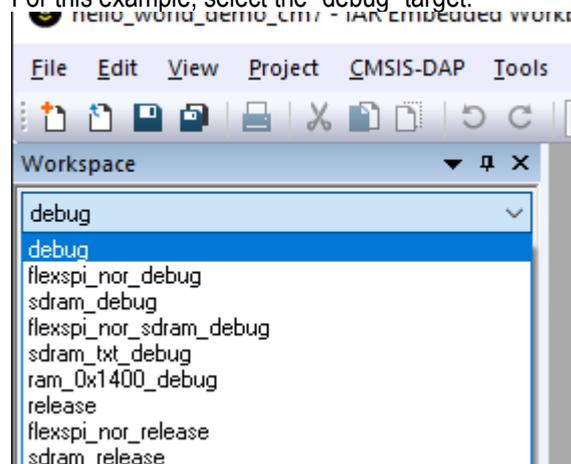
1. Open the desired example application either by double clicking the `.eww` file in the explorer or by starting IAR Embedded Workbench and selecting File->Open Workspace... from the menu. The hello world example has this path:

```
<install_dir>boards/evkmimxrt1170/demo_apps/hello_world/cm7/
iar/hello_world_demo_cm7.eww
```

2. Customize the project. For projects using a display this includes selecting which display to use, for networking projects it may include setting the IP number/netmask, for Wi-Fi projects it includes selecting which M.2 module to use and which Wi-Fi network to connect to.

These settings are handled differently for different SDK versions and different MCUs so consult the `readme.txt` file available in the `docs/` folder of each example for information.

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

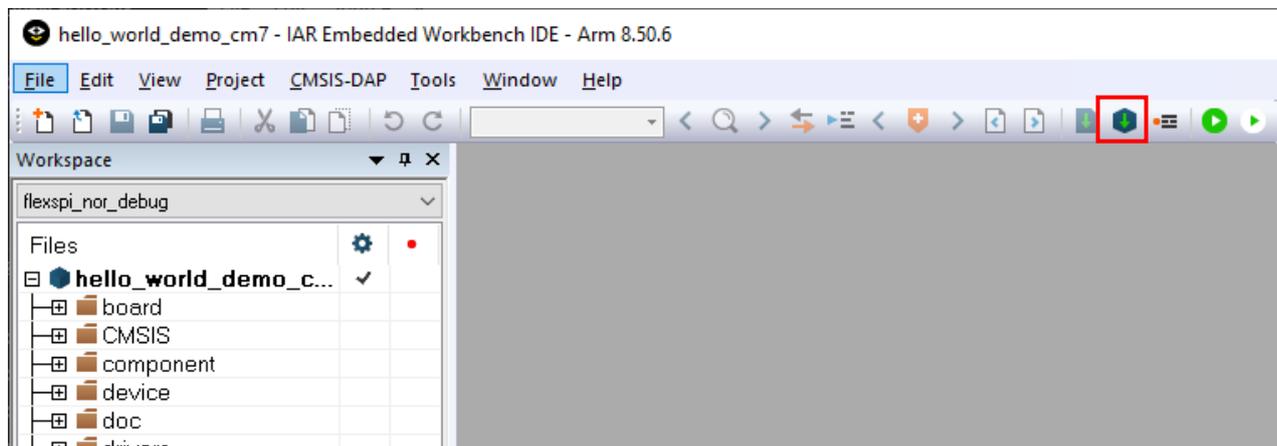


The target name indicates from where the application is executing, see table below. The name also indicates if it is a release or debug target. Release targets have a higher compiler optimization degree making the target less suitable for debugging. Available targets differ from example to example but here are some of the more common ones.

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 before use.
flexspi_nor_sdram_debug flexspi_nor_sdram_release	The application runs in external flash and use SDRAM. The external flash must be programmed before use.

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

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



- The build will complete without errors

4.3 Run an Example Application

To download and run the application, perform these steps:

- Prepare the hardware as described in chapter 2.3 and power it on.
- Open the terminal application on the PC, such as TeraTerm or PuTTY, and connect to the virtual COM port. Configure the terminal with 115200 baud, 8N1. You can alter the baud rate by searching for the BOARD_DEBUG_UART_BAUDRATE define in file: **board.h**

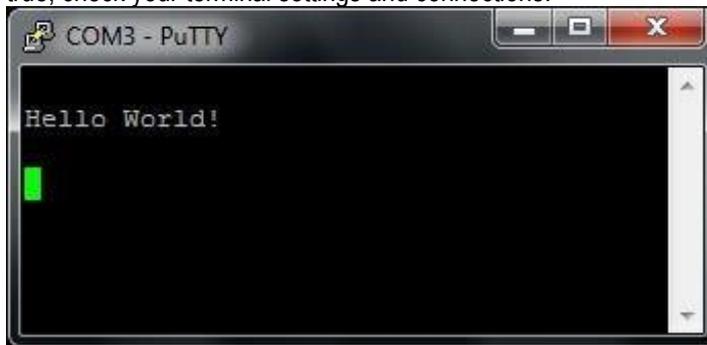
3. Click the "Download and Debug" button to download the application to the target.



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.

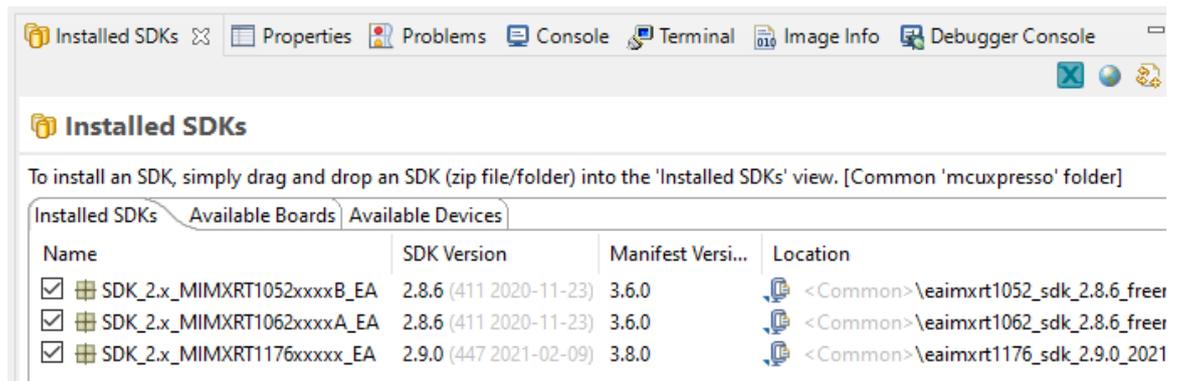


5 Getting Started with NXP MCUXpresso IDE

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.

5.1 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 in 2.1 to the "Installed SDKs" tab in MCUXpresso:

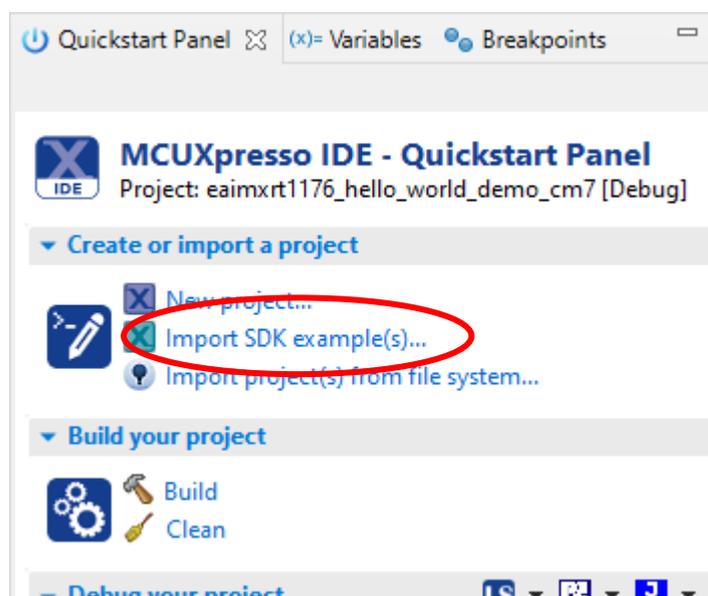


During the installation MCUXpresso will make a copy of the zip file that you drag-n-dropped so it is ok to delete it afterwards if you don't need it for one of the other IDEs.

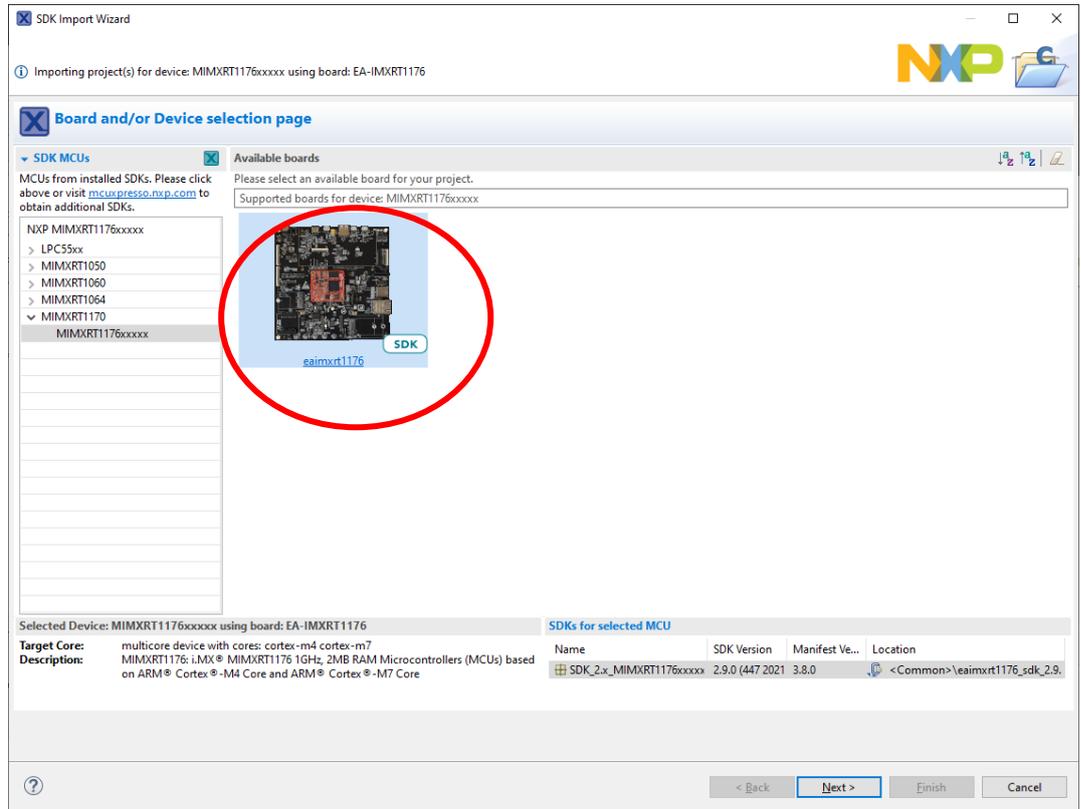
5.2 Build an Example Application

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

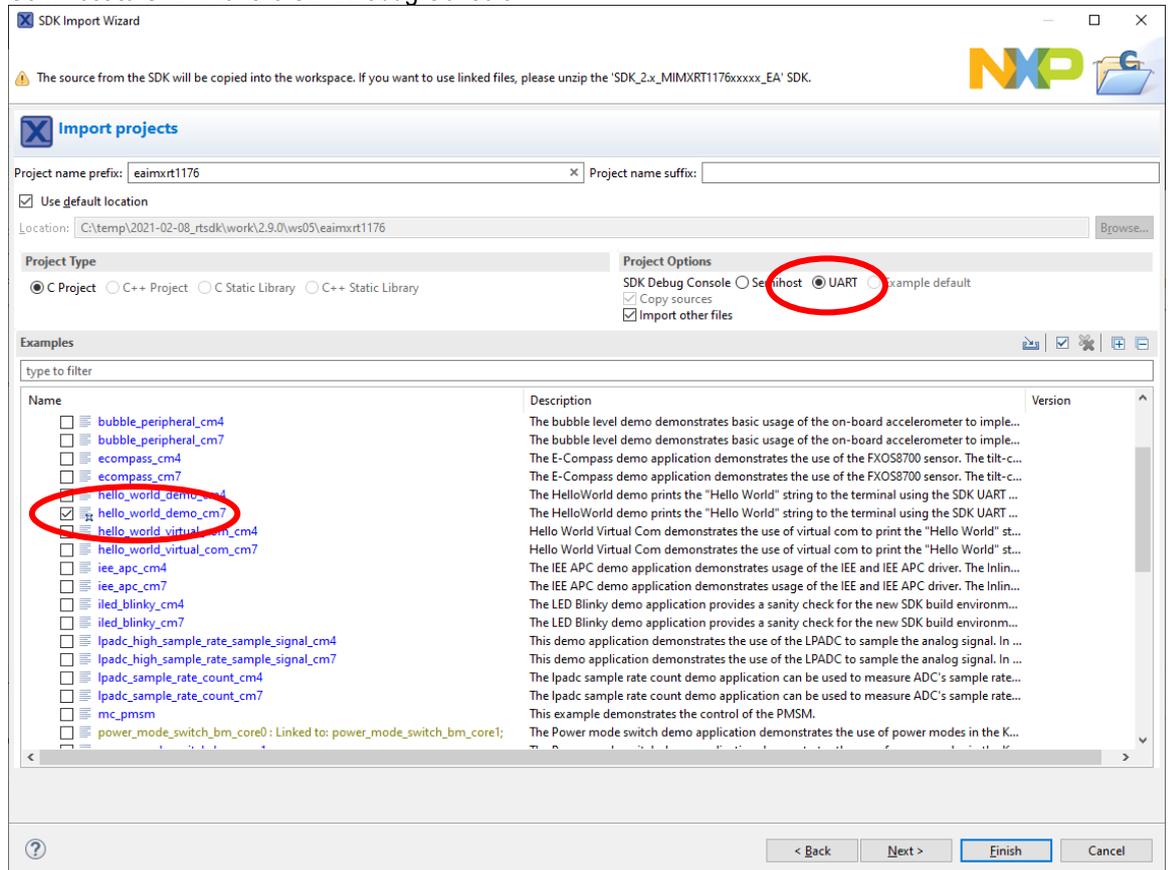
1. Install the SDK as described in section 5.1 if you have not done so already
2. Click the "Import SDK example(s)..." link in the Quickstart Panel



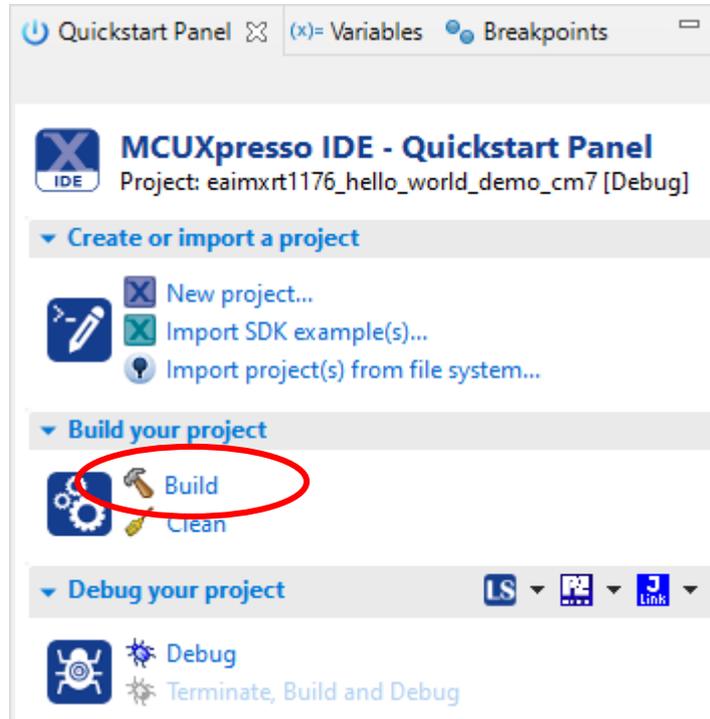
3. Select the iMX RT Developer's Kit (in this case the eaimxrt1176). Click Next to go to the project selector.



4. Select the hello_world example for Cortex-M7 (ends in _cm7) 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



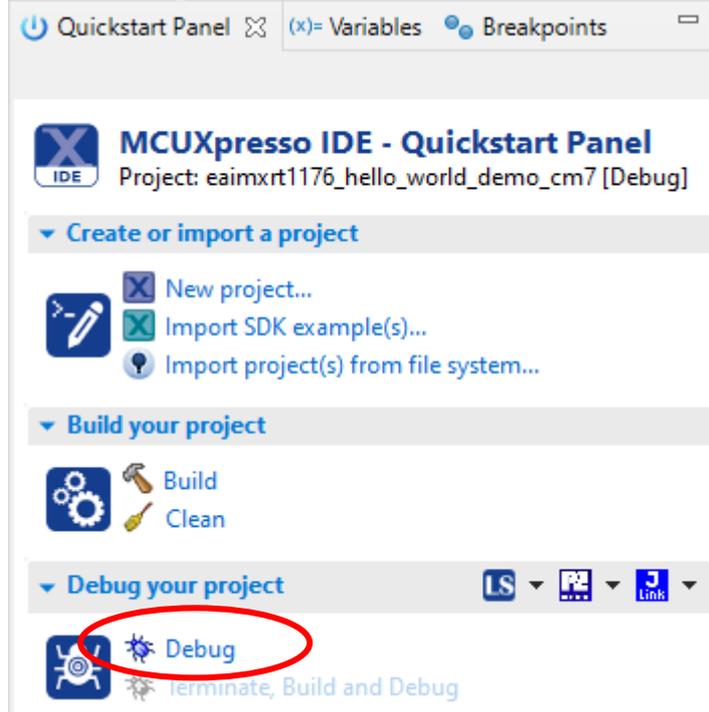
7. The build will complete without errors

5.3 Run an Example Application

To download and run the application, perform these steps:

1. Prepare the hardware as described in chapter 2.3 and power it on.
2. Open the terminal application on the PC, such as TeraTerm or PuTTY, and connect to the virtual COM port. Configure the terminal with 115200 baud, 8N1. You can alter the baud rate by searching for the BOARD_DEBUG_UART_BAUDRATE define in file: **board.h**

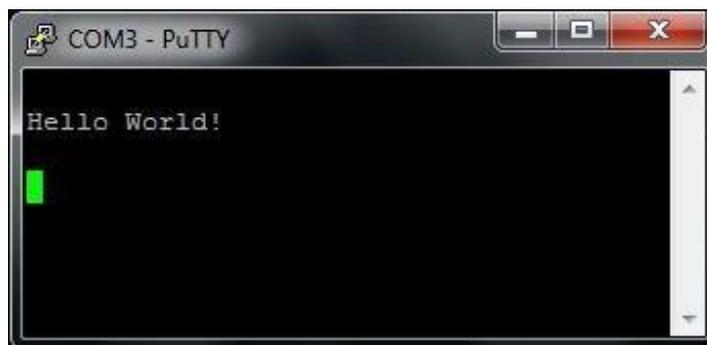
- Click the "Debug" button in the Quickstart Panel to download the application to the target.



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

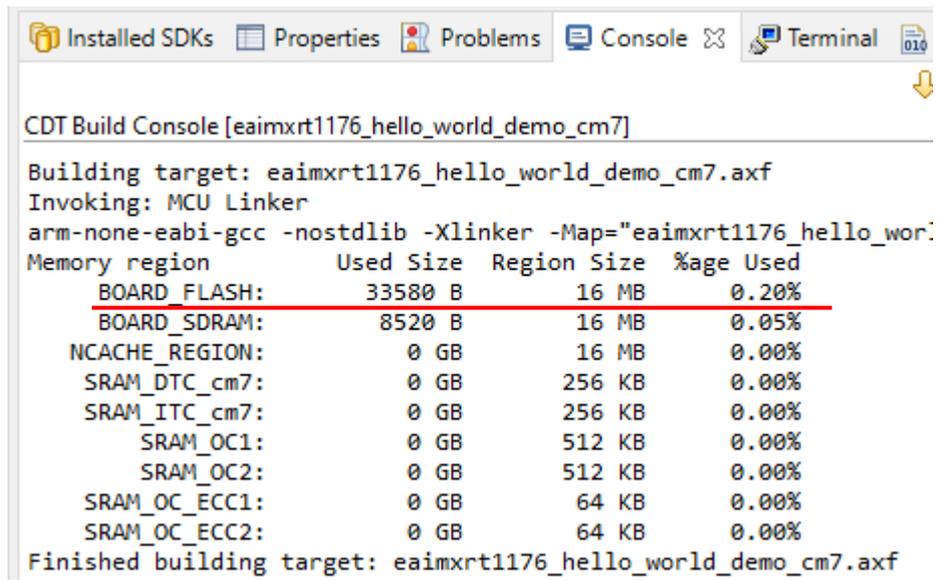


- The hello_world application is now running, and a banner is displayed on the terminal. If this is not true, check your terminal settings and connections.



5.4 Target Memory

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



The screenshot shows the CDT Build Console for the target `eaimxrt1176_hello_world_demo_cm7`. The console output indicates that the target is being built and the MCU Linker is invoked. The linker command is `arm-none-eabi-gcc -nostdlib -Xlinker -Map="eaimxrt1176_hello_wor:`. A table of memory usage is displayed, with the `BOARD_FLASH` row highlighted in red. The table shows the used size, region size, and percentage used for various memory regions.

```
CDT Build Console [eaimxrt1176_hello_world_demo_cm7]

Building target: eaimxrt1176_hello_world_demo_cm7.axf
Invoking: MCU Linker
arm-none-eabi-gcc -nostdlib -Xlinker -Map="eaimxrt1176_hello_wor:
Memory region      Used Size  Region Size  %age Used
BOARD_FLASH:      33580 B    16 MB        0.20%
BOARD_SDRAM:      8520 B     16 MB        0.05%
NCACHE_REGION:    0 GB       16 MB        0.00%
SRAM_DTC_cm7:     0 GB       256 KB       0.00%
SRAM_ITC_cm7:     0 GB       256 KB       0.00%
SRAM_OC1:         0 GB       512 KB       0.00%
SRAM_OC2:         0 GB       512 KB       0.00%
SRAM_OC_ECC1:     0 GB       64 KB        0.00%
SRAM_OC_ECC2:     0 GB       64 KB        0.00%
Finished building target: eaimxrt1176_hello_world_demo_cm7.axf
```

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

Properties for eaimxrt1176_hello_world_demo_cm7

type filter text

MCU settings

Available parts

SDK MCUs

Preinstalled MCUs

Target architecture: cortex-m7

Preserve memory configuration

Preserve project configuration

Memory details (MIMXRT1176xxxxx)*

Default LinkServer Flash Driver Browse...

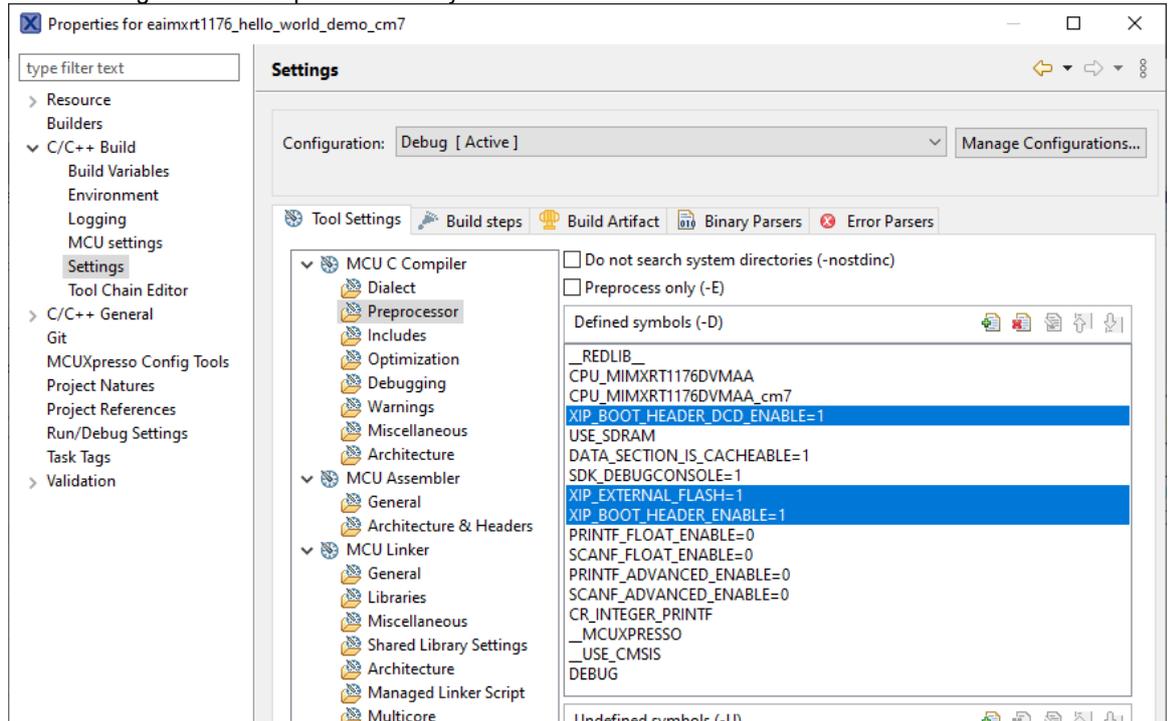
Type	Name	Alias	Location	Size	Driver
Flash	BOARD_FL...	Flash	0x30000000	0x1000000	MIMXRT11...
RAM	BOARD_SD...	RAM	0x80000000	0x1000000	
RAM	NCACHE_...	RAM2	0x81000000	0x1000000	
RAM	SRAM_DTC...	RAM3	0x20000000	0x40000	
RAM	SRAM_ITC...	RAM4	0x0	0x40000	
RAM	SRAM_OC1	RAM5	0x20240000	0x80000	
RAM	SRAM_OC2	RAM6	0x202c0000	0x80000	
RAM	SRAM_OC...	RAM7	0x20340000	0x10000	
RAM	SRAM_OC...	RAM8	0x20350000	0x10000	

Add Flash Add RAM Split Join Delete Import... Merge... Export... Generate...

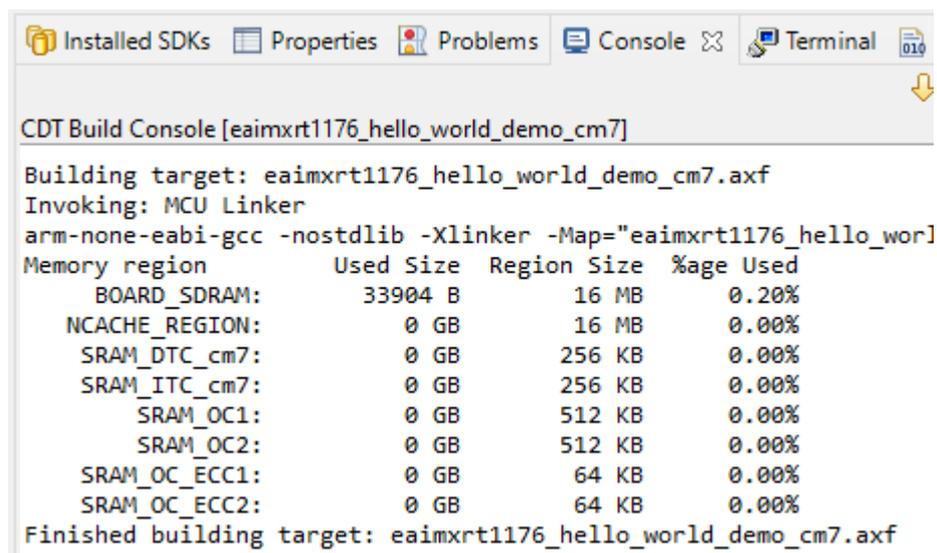
Refresh MCU Cache

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

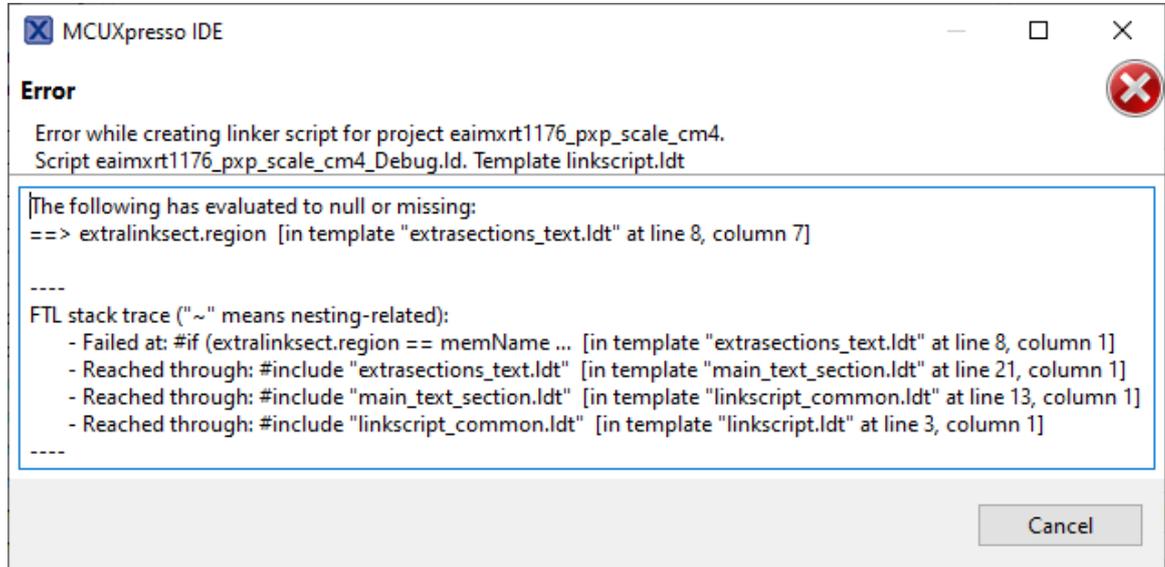


- There are three symbols that must be changed. The symbols are XIP_BOOT_HEADER_DCD_ENABLE, XIP_EXTERNAL_FLASH and XIP_BOOT_HEADER_ENABLE. Double-click each one and change the value from 1 to 0 to disable the feature.
- Build the project and look at the output. The BOARD_FLASH is no longer present and instead the SDRAM is used:



5.5 Troubleshooting

It is possible that a dialog like this pops up when building your project:



That error message appears because the project was not 100% imported and this happens sometimes if you select multiple projects to import at the same time. To fix the problem you have two choices:

Option 1: Right click the project, select Delete and make sure that the checkbox for deleting the content on the disk is filled in. You can then import that project again.

Option 2: Switch to a new workspace and only import that project.

6 Debug Interface

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

There is a Cortex Debug interface connector (J11) on the *uCOM Carrier board*. It is a 2x5 pos, 50 mil pitch connector with a shroud. Note that pin 7 is present. Some debug probe connectors have plugged pin 7. Such a cable connector cannot be used.

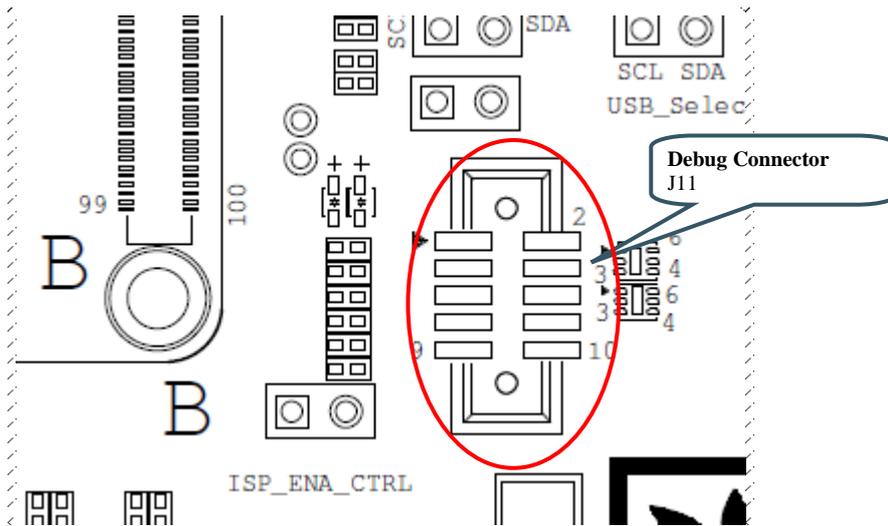


Figure 1 – Debug Interfaces on uCOM Carrier board

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

6.1 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.

6.1.1 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/>.

6.1.2 MCUXpresso

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

6.1.3 Keil uVision

All projects have been configured to use CMSIS-DAP as debug hardware. Follow the steps below to switch to J-LINK/J-TRACE.

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

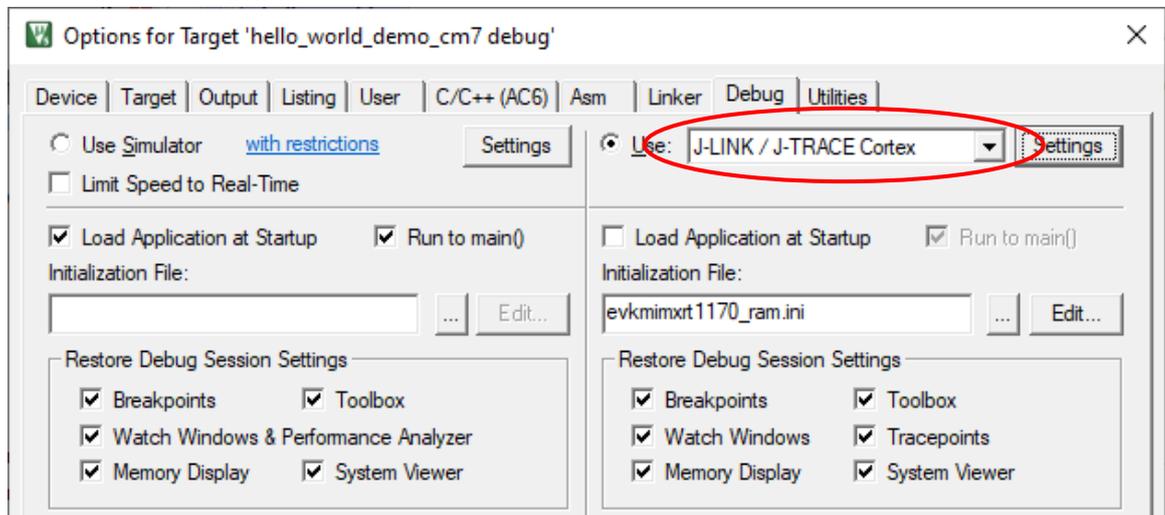


Figure 2 – Setting Debug Interface in Keil uVision

Open the settings dialog and change to the following settings:

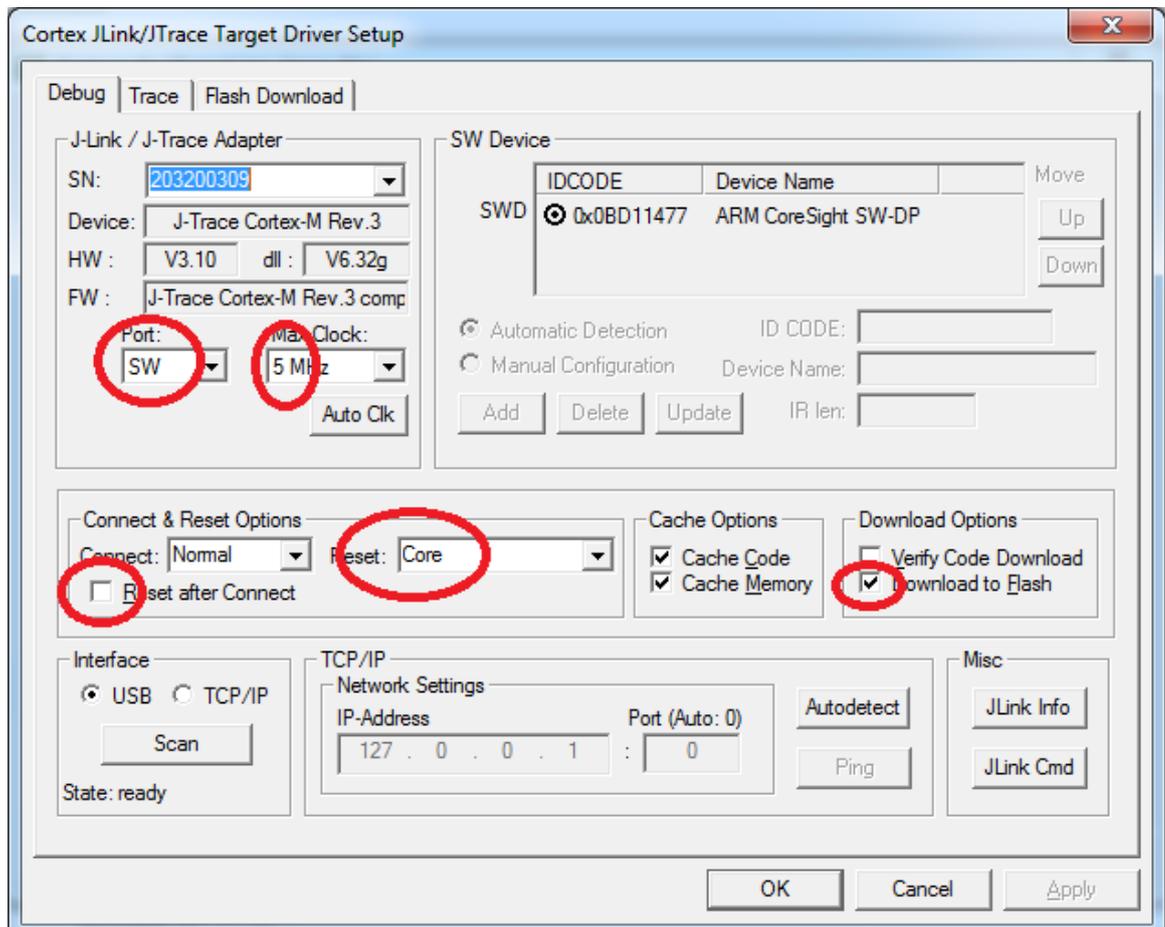


Figure 3 – Configuring J-TRACE/J-LINK Interface in Keil uVision

Switch to the Flash Download tab and make sure that the flash algorithm is correct for the MCU according to this table:

Product	Flash Description
iMX RT1064	MIMXRT106x 4mB Winbond QSPI Flash, 4M, On-chip Flash
iMX RT1166	MIMXRT117x 16mB QuadSPI NOR Flash, 16M, Ext Flash SPI
iMX RT1176	MIMXRT117x 16mB QuadSPI NOR Flash, 16M, Ext Flash SPI

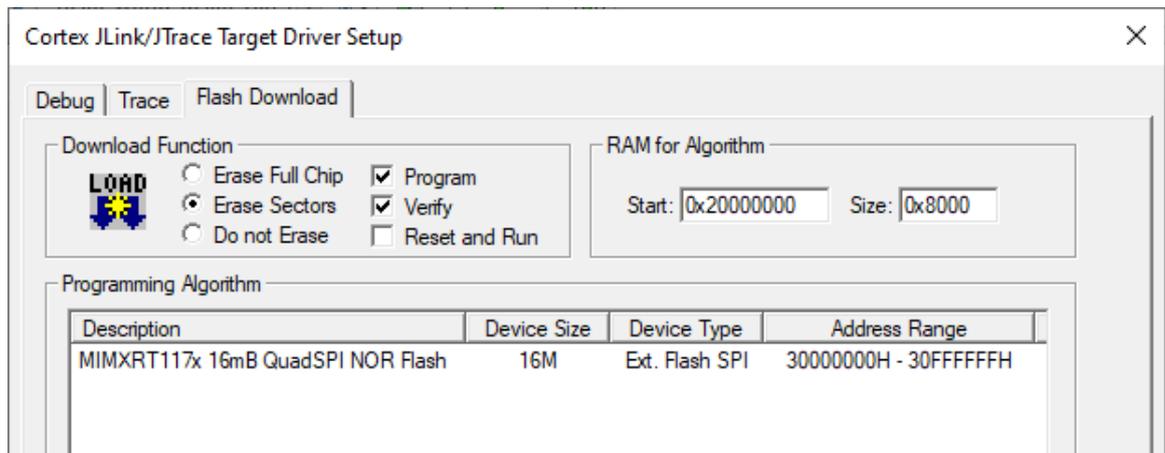


Figure 4 – Configuring Flash Programming for J-TRACE/J-LINK Interface in Keil uVision for iMX RT1176

7 Standalone Program Download

This chapter describes how to download an application to the *iMX RT 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 exist 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 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 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.

7.1 Install the Required Software

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_v2.1\` which will be referred to as <install directory> from now on.

7.2 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 *RT10xx/RT11xx Device Workflow* 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>

As an example, this is what was needed to prepare the `led_blinky` demo application:

- 1) Open the project
`<install_dir>\eaimxrt1176_sdk_2_10_1\boards\evkmimxrt1170\demo_apps\led_blinky\cm7\mdk\iled_blinky_cm7.uvprojx`
- 2) Select the "sdram_debug" target
- 3) Open Project->Options-Output and make sure that "Create HEX File" is selected
- 4) Switch to the Linker tab and click Edit to open the scatter file in the background
- 5) Close the dialog with the OK button (to save the change made in step 3)
- 6) Modify the following lines in the scatter file:

```
#define m_interrupts_start          0x00000000
#define m_interrupts_size          0x00000400

#define m_text_start                0x00000400
#define m_text_size                 0x0003FC00

#define m_data_start                0x80000000
#define m_data_size                 0x01000000
```

to look like this:

```
#define m_interrupts_start          0x80002000
#define m_interrupts_size          0x00000400

#define m_text_start                0x80002400
#define m_text_size                 0x0001DC00

#define m_data_start                0x80020000
#define m_data_size                 0x00DE0000
```

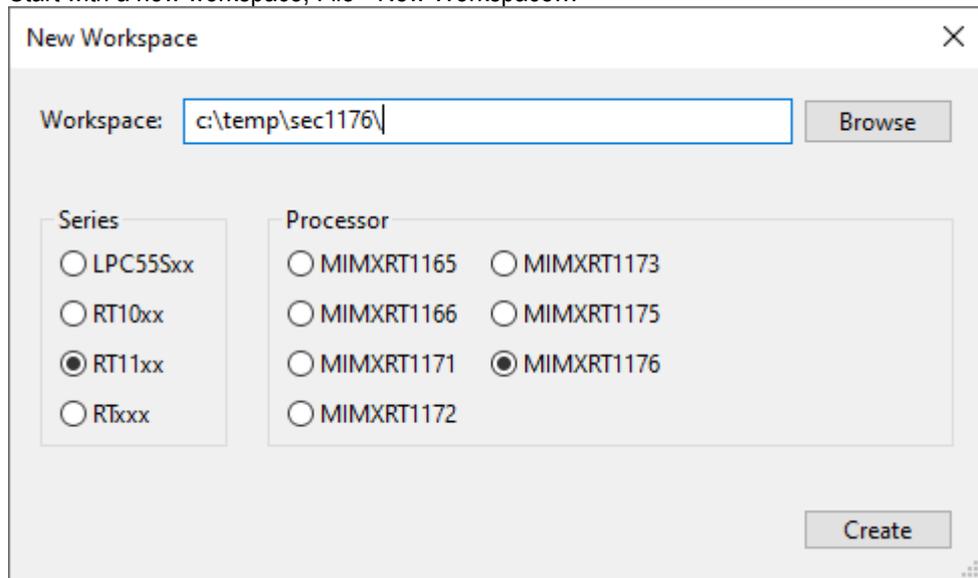
- 7) Save the file
- 8) Press F7 or Project->Build Target
- 9) There should now be a hex file here:
`<install_dir>\eaimxrt1176_sdk_2_10_1\boards\evkmimxrt1170\demo_apps\led_blinky\cm7\mdk\sdram_debug\iled_blinky_cm7.hex`

7.3 Booting an Unsigned Image

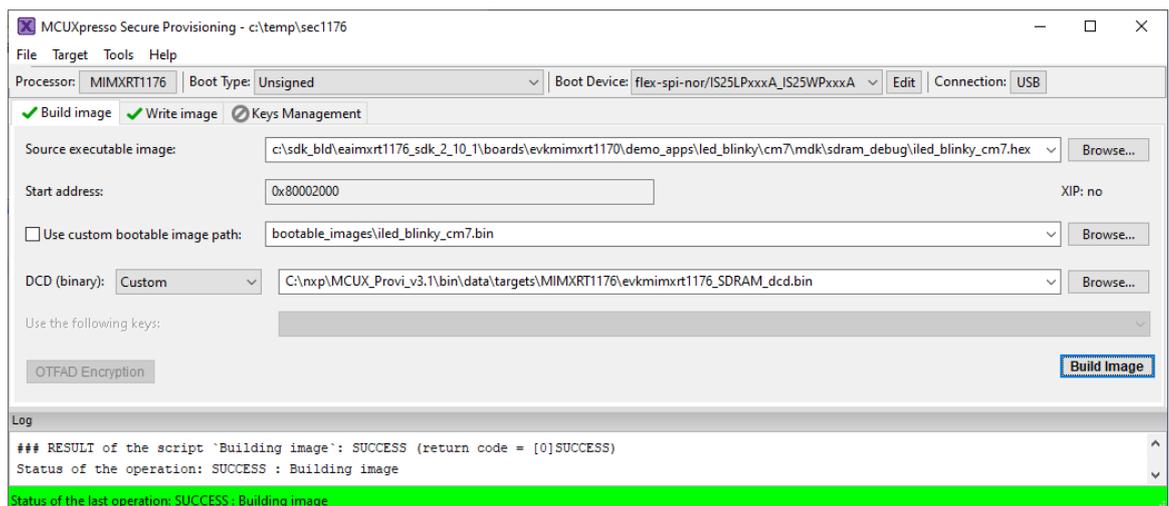
Unsigned image is typically used for development. It's recommended to start with this boot type before working with secured images to verify that the executable image works properly.

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

1. Start with a new workspace, File->New Workspace...



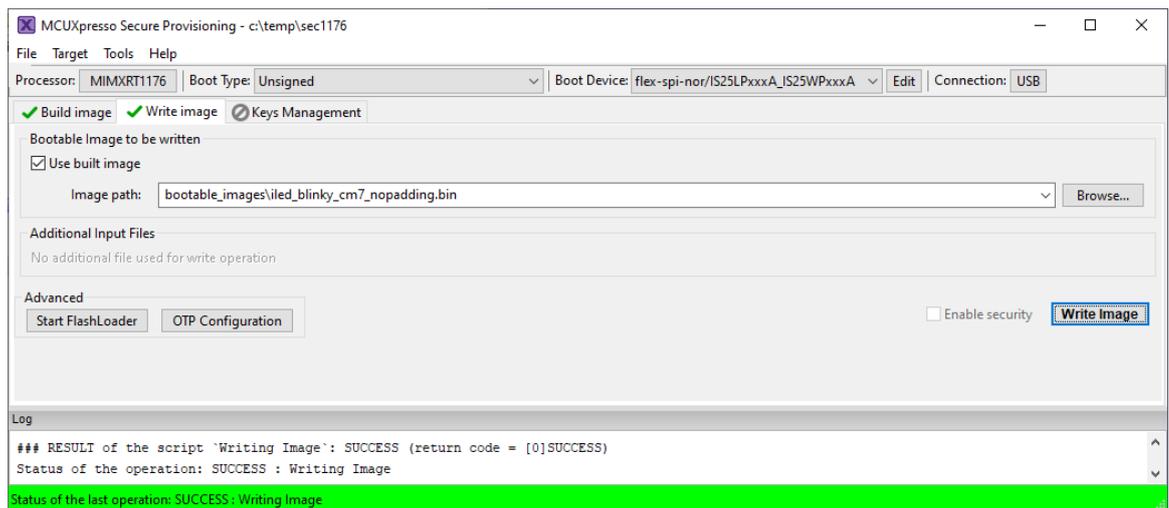
2. Select a location for the workspace and which processor to use. Click Create.
3. Make sure that Boot Type is *Unsigned*
4. Set Boot Device to *flex-spi-nor/IS25LPxxxA_IS25WPxxxA* for iMX RT11xx targets. For the iMX RT1064 set the Boot Device to *flex-spi-nor/W25Q32JV*.
5. Switch to the *Build Image* tab
6. Select the Source executable image that was prepared in the previous section, i.e. **iled_blinky_cm7.hex**
7. 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
8. 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 J26 and J29 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



7.4 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

8 Terminal Application Setup

This chapter contains information about the terminal connection that exist on the *uCOM 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 J29. The terminal is commonly used during program development.

8.1 UART-to-USB Bridge

The UART-to-USB bridge chip (FT230XS-R from FTDI) on the *uCOM 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 any more, especially not on laptops. The USB port also offers the possibility to power the board.

There are two LEDs, transmit from the board (LED9) and receive to the board (LED8), that signal communication activity.

See picture below for locating relevant components.

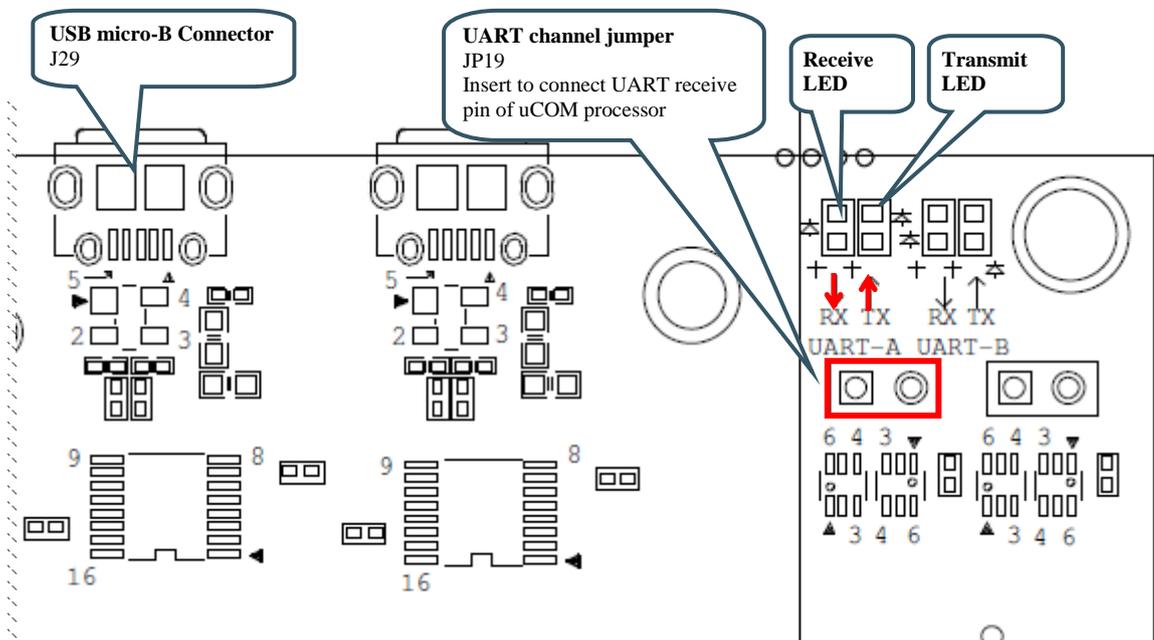


Figure 5 – UART-to-USB Bridge

8.2 Terminal Application on the PC

Begin by connecting the micro-B USB connector to J29, 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 be listed under “Ports” in the Device Manager as shown in Figure 6. Please note that the actual port number will most likely be different on your computer.

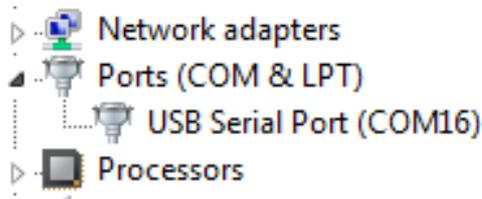


Figure 6 – Virtual COM port shown in device manager

The next step is to open a terminal application and attached 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.

8.2.1 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://tssh2.osdn.jp/index.html.en>
<http://sourceforge.jp/projects/tssh2/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 *uCOM Carrier Board*, there should be a COM port automatically populated in the list.

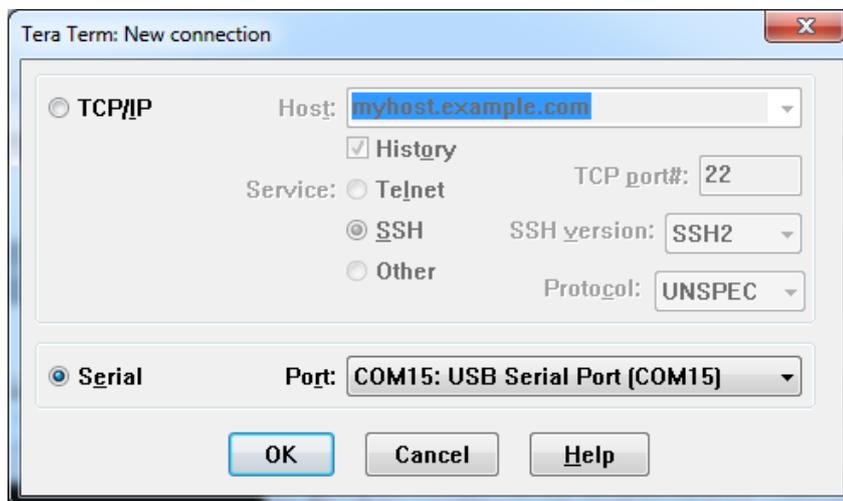


Figure 7 – Tera Term New Connection Window

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.

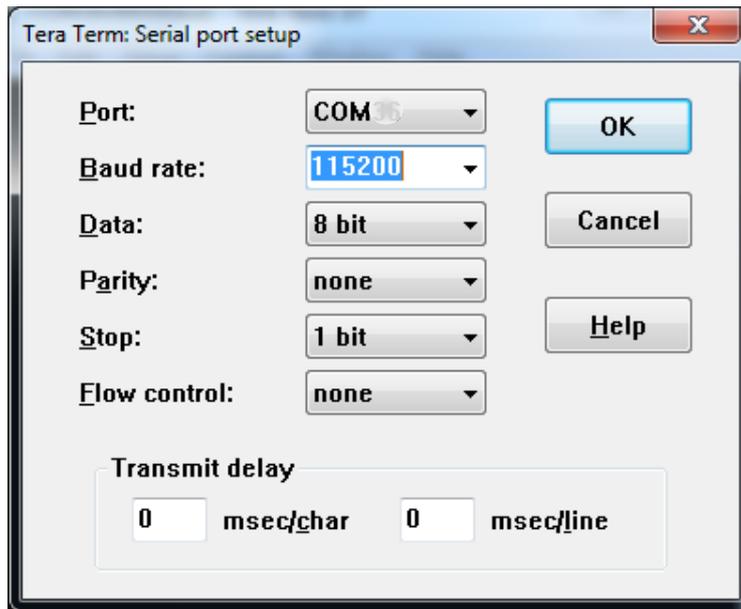


Figure 8 – Tera Term Serial Port Setup

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

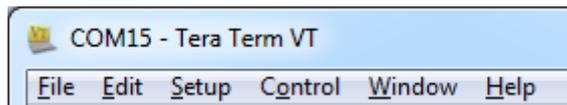


Figure 9 – Tera Term Menu

8.2.2 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.

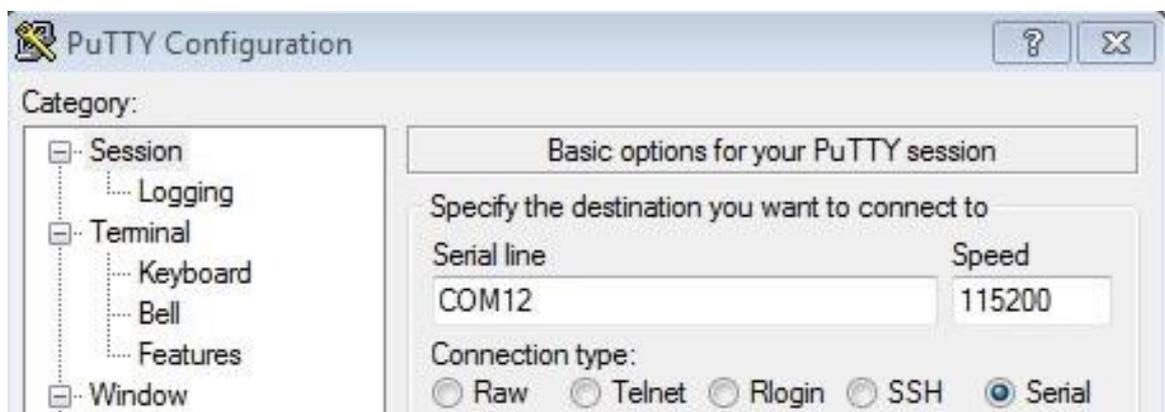


Figure 10 – PuTTY New Session Configuration

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.

9 Booting from External Memory

The i.MX RT MCU does not have any internal flash memory for storing the application. It must be stored in an external memory. The i.MX RT1064 is no exception, but NXP has packaged a QSPI flash chip inside the same package as the microcontroller itself, so it looks like the flash memory is internal. From a program development and memory layout point of view, all i.MX RT microcontrollers are designed the same.

On the *iMX RT1064 uCOM board*, the flash memory is inside the package (as mentioned above) and is a 4MByte QSPI memory.

On the *iMX RT1166/RT1176 uCOM boards*, the flash memory is an external 16Mbyte QSPI memory.

The i.MX RT MCU always boots (i.e., starts executing) from these respective QSPI flash memories.

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.

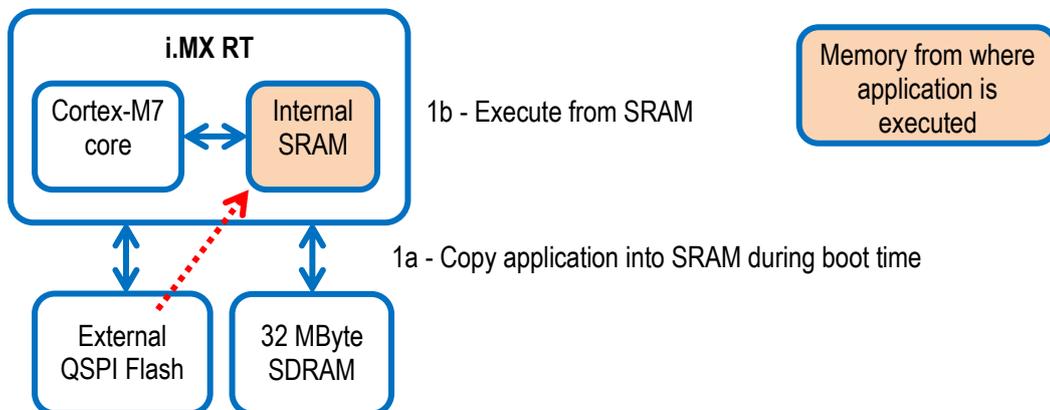


Figure 11 – i.MX RT and External QSPI flash - Executing from SRAM

1. The application is stored in the external 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. 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 external 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.

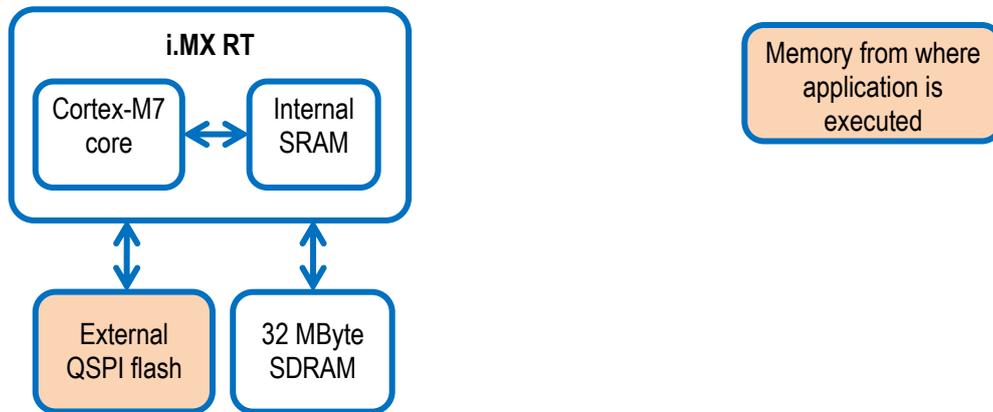


Figure 12 – i.MX RT and QSPI flash - Executing from external flash

Note that on the i.MX RT1064 microcontroller, the external flash is inside the same package, but the overall architecture is still the same.

2. The application is stored in the external flash and 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. The execution performance will be considerably lower than the performance when executing from internal SRAM.
 - b. During program development the application must be downloaded/flushed to the external flash memory before debugging 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.

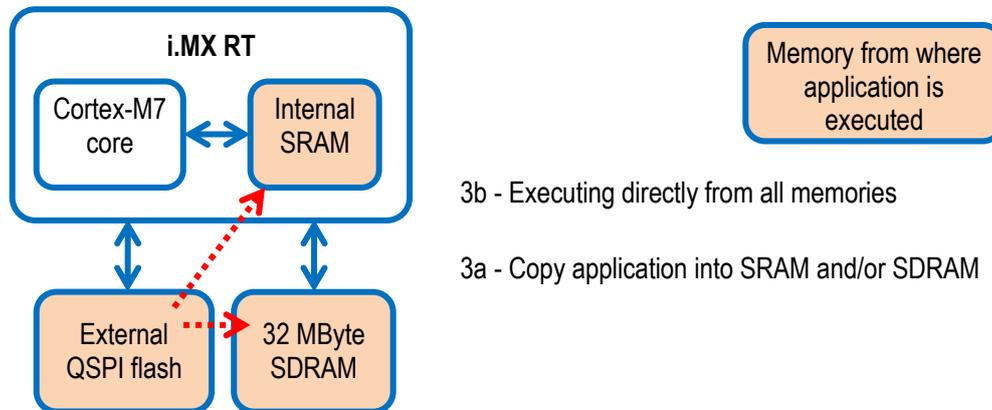


Figure 13 – i.MX RT and QSPI flash - Executing from all memories

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 external 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 external 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 in order to select and implement the best solution.

10 Troubleshooting

10.1 Cannot debug / download

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. If you, for example, get a dialog in MCUXpresso similar to Figure 14 below the reason is most likely because of an interfering application.

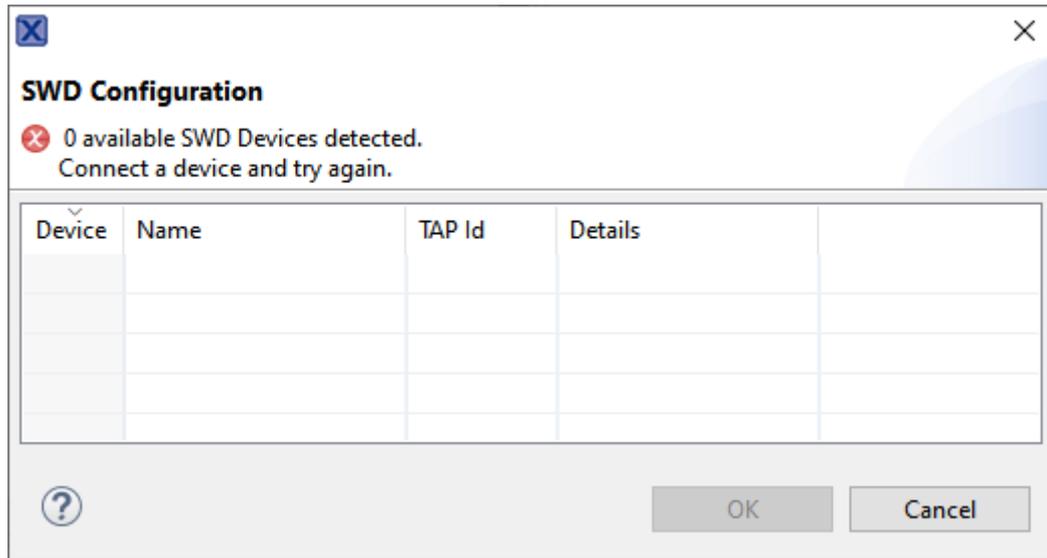


Figure 14 - No SWD Devices detected

The solution is to put the hardware in ISP mode before starting the flash/debug operation in the IDE. To do this you need to press a couple of buttons on the carrier board, see Figure 15.

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

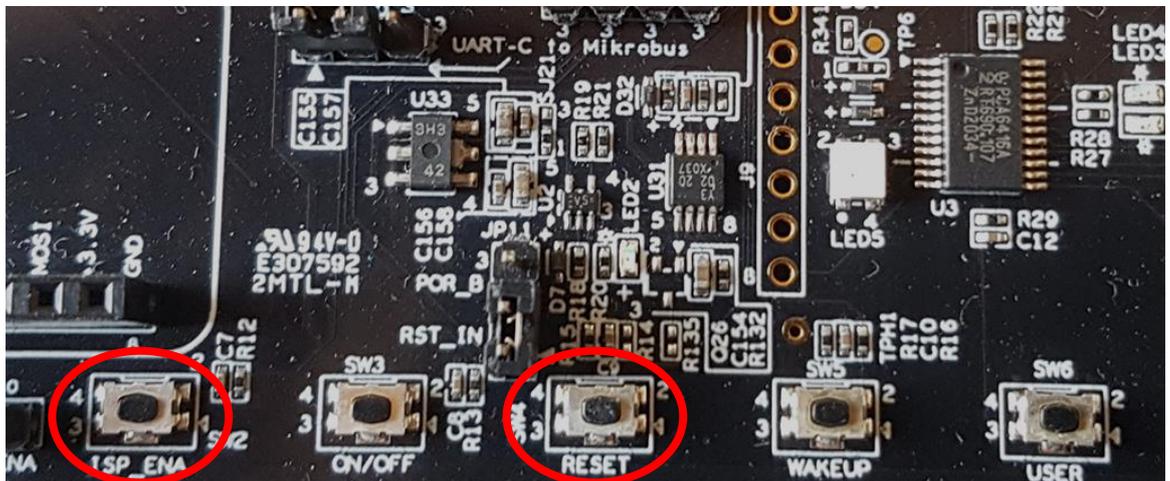


Figure 15 - Buttons on uCOM carrier board

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).

11 Things to Note

11.1 ESD Precaution

Please note that the *iMX RT uCOM Board* and *uCOM 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) precaution.

Make it a habit always to first touch the metal surface of one of the USB, uSD 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 uCOM Board* and in general as little as possible on the *uCOM Carrier Board*. The push-buttons on the *uCOM 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 uCOM Board* and *uCOM 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 *uCOM 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. Once programmed, there is no possibility to reprogram them.

iMX RT uCOM Boards are delivered with OTP fuse programming to boot from external QSPI flash. The rest of the fuses are completely up to the user to decide if OTP fuses shall be programmed and in that case, which ones.

Note that Embedded Artists does not replace iMX RT uCOM 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.

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.