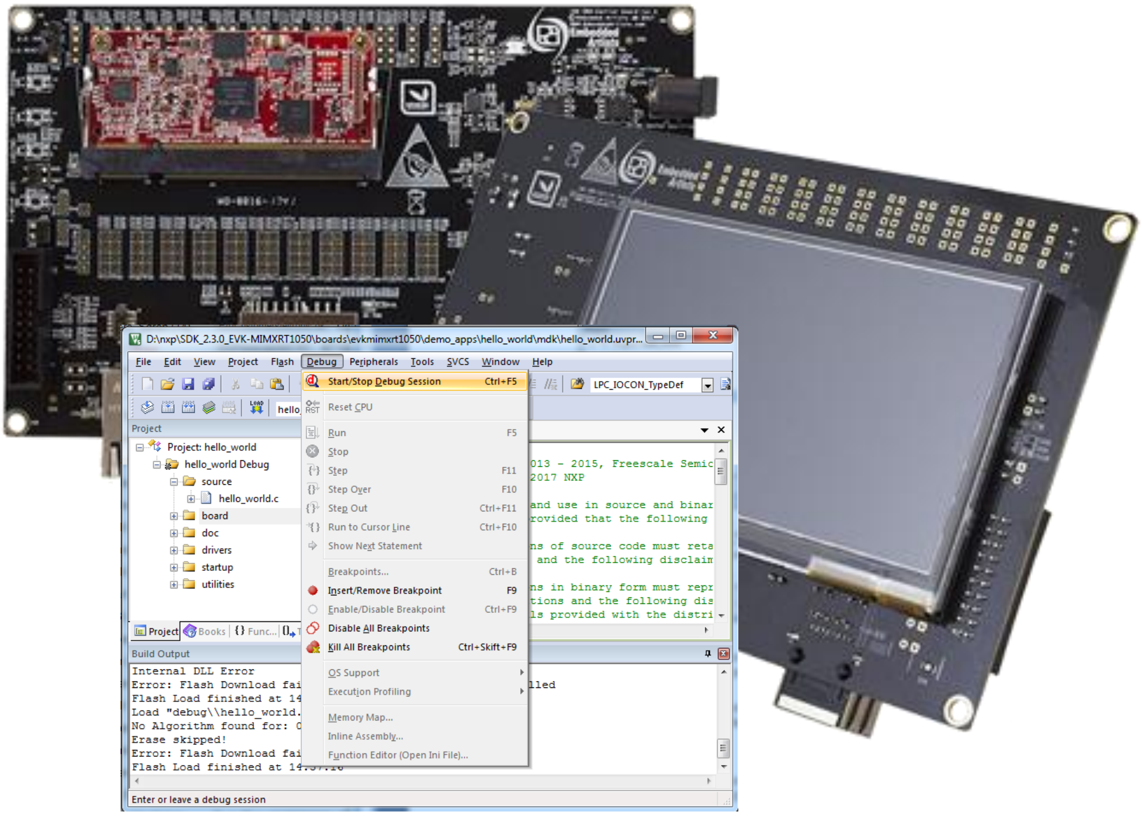
iMX RT1052/1062 Developer's Kit - Program Dev. Guide

iMX RT1052/1062 Developer's Kit
Program Development Guide


D:\nxp\SDK_2.3.0_EVK-MIMXRT1050\boards\evkmimxrt1050\demo_apps\hello_world\mdk\hello_world.uvpr...
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Start/Stop Debug Session Ctrl+F5
Reset CPU
Run F5
Stop
Step F11
Step Over F10
Step Out Ctrl+F11
Run to Cursor Line Ctrl+F10
Show Next Statement
Breakpoints... Ctrl+B
Insert/Remove Breakpoint F9
Enable/Disable Breakpoint Ctrl+F9
Disable All Breakpoints
Kill All Breakpoints Ctrl+Skift+F9
OS Support
Execution Profiling
Memory Map...
Inline Assembly...
Function Editor (Open Ini File)...
LPC_IOCON_TypeDef
Project
Project: hello_world
hello_world Debug
source
hello_world.c
board
doc
drivers
startup
utilities
Project Books Func...
Build Output
Enter or leave a debug session

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

Embedded Artists AB
Jörgen Ankersgatan 12
211 45 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*
PA1	2018-01-02	First released version.
PA2	2018-01-03	Added description of build targets and Mfgtool.
PA3	2018-04-16	Updated instructions for SDK and different IDEs.
PA4	2018-05-22	Updated instructions for MCUXpresso.
PA5	2018-06-25	Updated instructions to match version 2.4.0 of the SDK. Changed description for IAR Workbench as flash driver is now included in the IDE.
PA6	2018-08-24	Reorganize the document structure. Updated instructions around Mfgtool and flashing. Added J-LINK/J-TRACE instructions.
PA7	2018-09-10	Added information about CMSIS-PACK for Keil uVision
PA8	2018-12-17	Name changes and removed requirement to register on our website.
PA9	2019-02-08	Added references to iMX RT1062 OEM board.
PA10	2019-02-11	Updated instructions to match SDK 2.5.0
PA11	2019-04-10	Updated instructions to match SDK 2.5.1
PA12	2019-04-29	Corrected statement about EEPROM in section 2.3.2. Updated patches for Wi-Fi Examples

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 EcoXiP flash memory.

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

1. **MCUXpresso SDK -** this is a package of sample software that must be downloaded from NXPs website. You need to create an account on NXP's website for this.
2. **Apply a zip-file** to the SDK with projects specific to the *iMX RT Developer's Kit*. The zip-file is downloaded from the *iMX RT Developer's Kit* support page on Embedded Artists' website.
3. **Integrated Development Environment (IDE)**
   a. As of August 2018, **NXP MCUXpresso, Keil uVision/MDK and IAR Embedded Workbench** supports direct flash programming of the EcoXiP. Update to latest version of respective IDE.
   b. Programming of the EcoXiP flash is also supported via NXP's Mfgtool standalone application but it is not a suitable tool to use during program development.
4. **JTAG probe** to be able to download the application to SRAM, SDRAM or the EcoXiP flash memory and in general to be able to debug - set breakpoints, inspect memory, etc.
   a. The low-cost LPC-Link2 is an excellent choice. Keil ULINK2 and ULINKplus, as well as Segger JLINK, are also excellent debug probes.
   b. Technically it is possible to program/flash the EcoXiP without a JTAG probe (via NXP's Mfgtool application), but it is strongly recommended to use the proper tool for debugging - i.e., use a JTAG probe!
5. And of course the ***iMX RT Developer's Kit***!

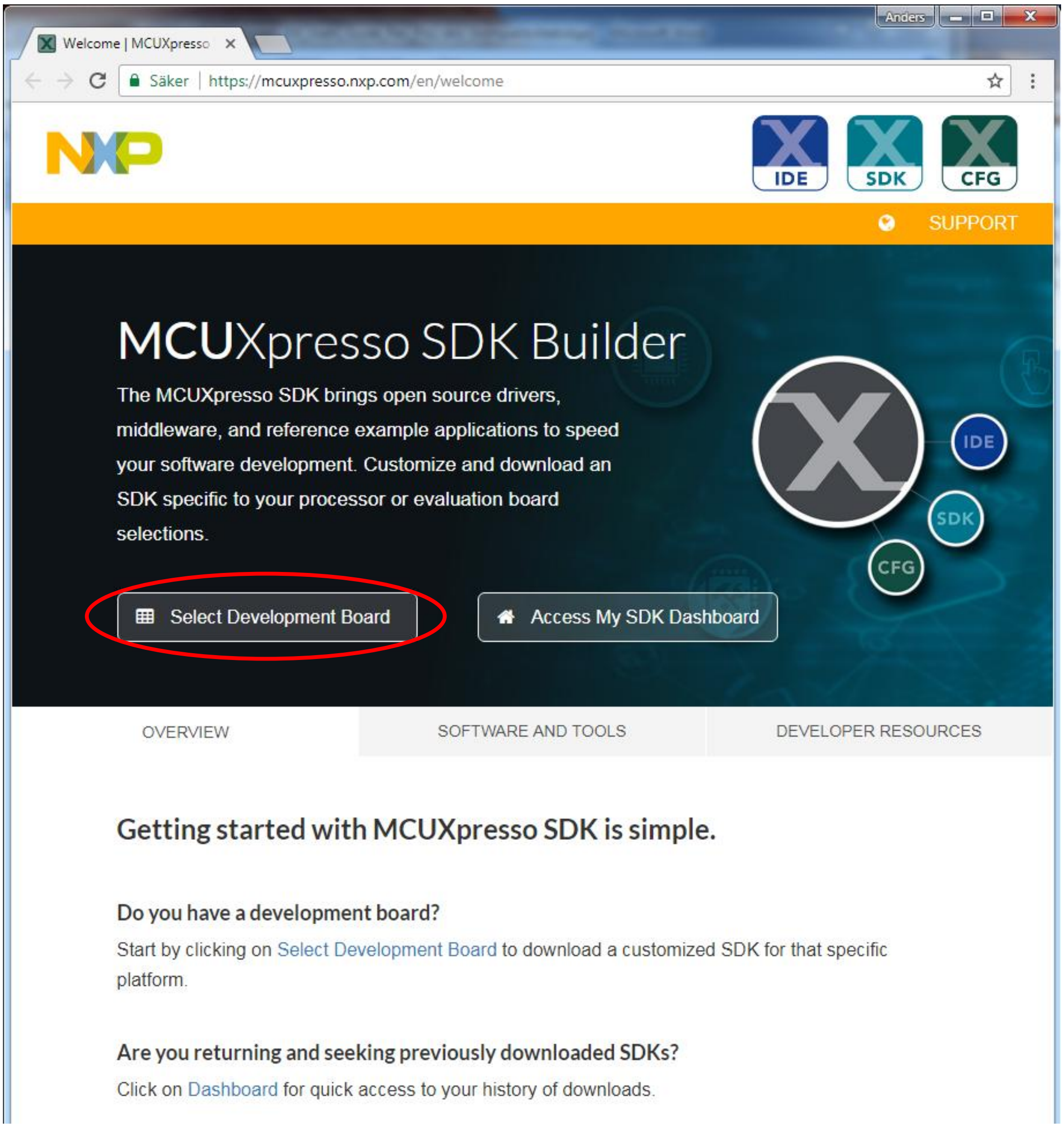
2.1 Downloading and Installing MCUXpresso SDK from NXP

This section will walk you through the installation of the MCUXpresso SDK, which is a package of sample software projects (with device drivers and peripheral examples and demos) that will get you started immediately with your i.MX RT software development. Note that even though the name of the SDK suggests the code package is for the MCUXpresso IDE, the sample projects have project files for other IDEs as well, such as Keil uVision/MDK, IAR Embedded Workbench, and more.

First download the MCUXpresso SDK by following this URL: https://mcuxpresso.nxp.com/en/welcome. This is the online SDK builder that makes sure you will get the latest version of the software.

Note that some details in the screen dumps in this section might change over time, but the basic walkthrough steps are the same.

You will need to login to your NXP account. Then click on *Select Development Board*.



MCUXpresso SDK Builder

The MCUXpresso SDK brings open source drivers, middleware, and reference example applications to speed your software development. Customize and download an SDK specific to your processor or evaluation board selections.

Getting started with MCUXpresso SDK is simple.

Do you have a development board?

Start by clicking on Select Development Board to download a customized SDK for that specific platform.

Are you returning and seeking previously downloaded SDKs?

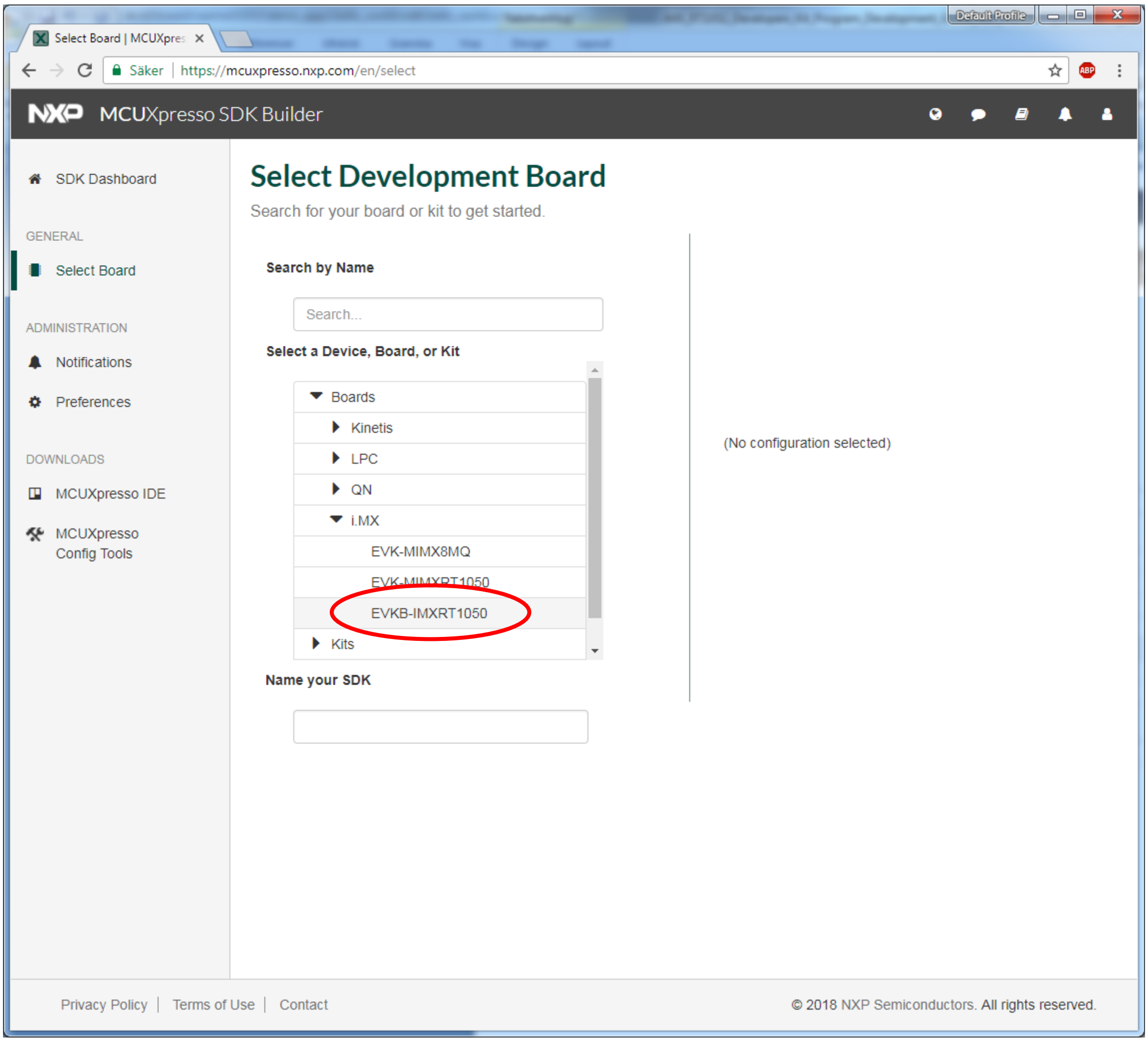
Click on Dashboard for quick access to your history of downloads.

Figure 1 – MCUXpresso SDK Builder

Then select the Boards → i.MX → EVKB-IMXRT1050 or EVK-MIMXRT1060 (depending on if you are using the iMX RT1052 or RT1062 Developer's Kit). This is the NXP evaluation board, but most of the samples will work unmodified on the *iMX RT Developer's Kit*.

Note that there are two different versions of the MIMXRT1050 EVK - EVK and EVKB. It is the latter that shall be selected because that supports i.MX RT1052 silicon revision A1.

For i.MX RT1062, only one version of the NXP EVK exists.



Select Development Board

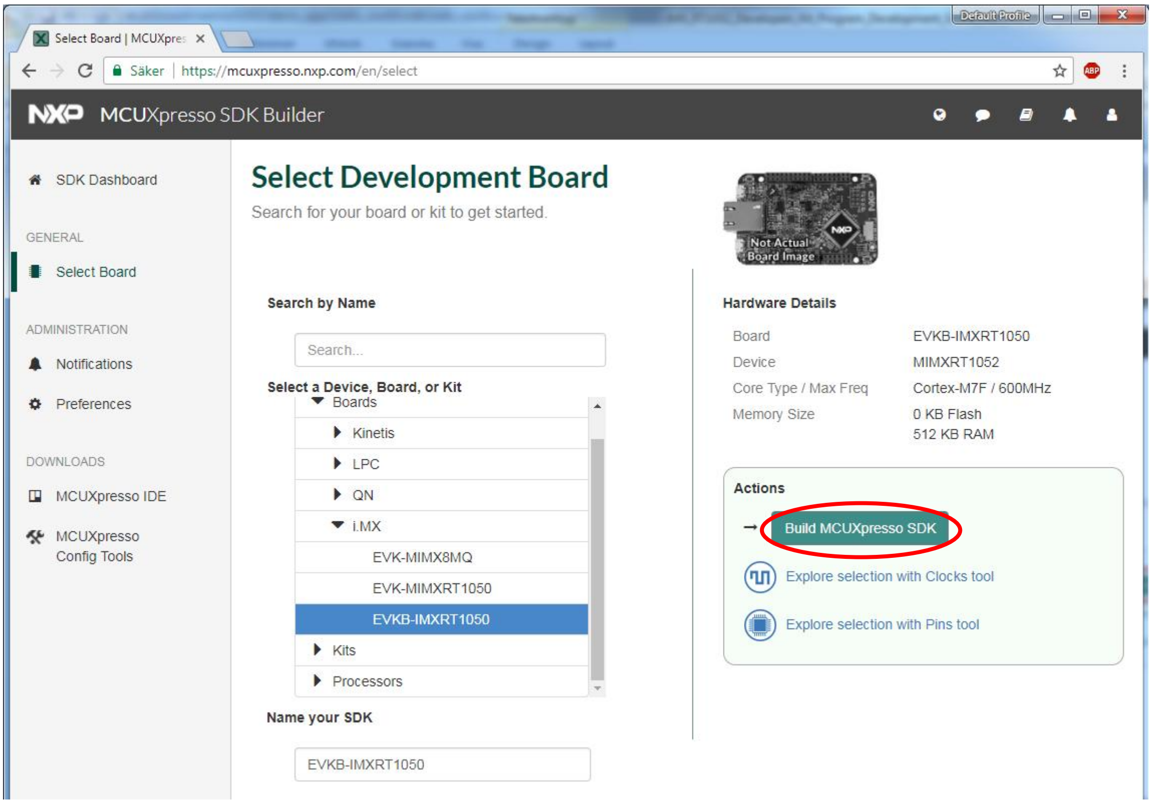
Search for your board or kit to get started.

(No configuration selected)

© 2018 NXP Semiconductors. All rights reserved.

Figure 2 – MCUXpresso SDK Builder - Select Development Board

After selecting the NXP EVK, click on the *Build MCUXpresso SDK* button.



Select Board | MCUXpres

Säker | https://mcuxpresso.nxp.com/en/select

NXP MCUXpresso SDK Builder

SDK Dashboard

GENERAL

Select Board

ADMINISTRATION

Notifications

Preferences

DOWNLOADS

MCUXpresso IDE

MCUXpresso Config Tools

Select Development Board

Search for your board or kit to get started.

Search by Name

Search...

Select a Device, Board, or Kit

- Boards
  - Kinetis
  - LPC
  - QN
  - i.MX
    - EVK-MIMX8MQ
    - EVK-MIMXRT1050
    - EVKB-IMXRT1050
- Kits
- Processors

Name your SDK

EVKB-IMXRT1050

Not Actual Board Image

Hardware Details

Board	EVKB-IMXRT1050
Device	MIMXRT1052
Core Type / Max Freq	Cortex-M7F / 600MHz
Memory Size	0 KB Flash 512 KB RAM

Actions

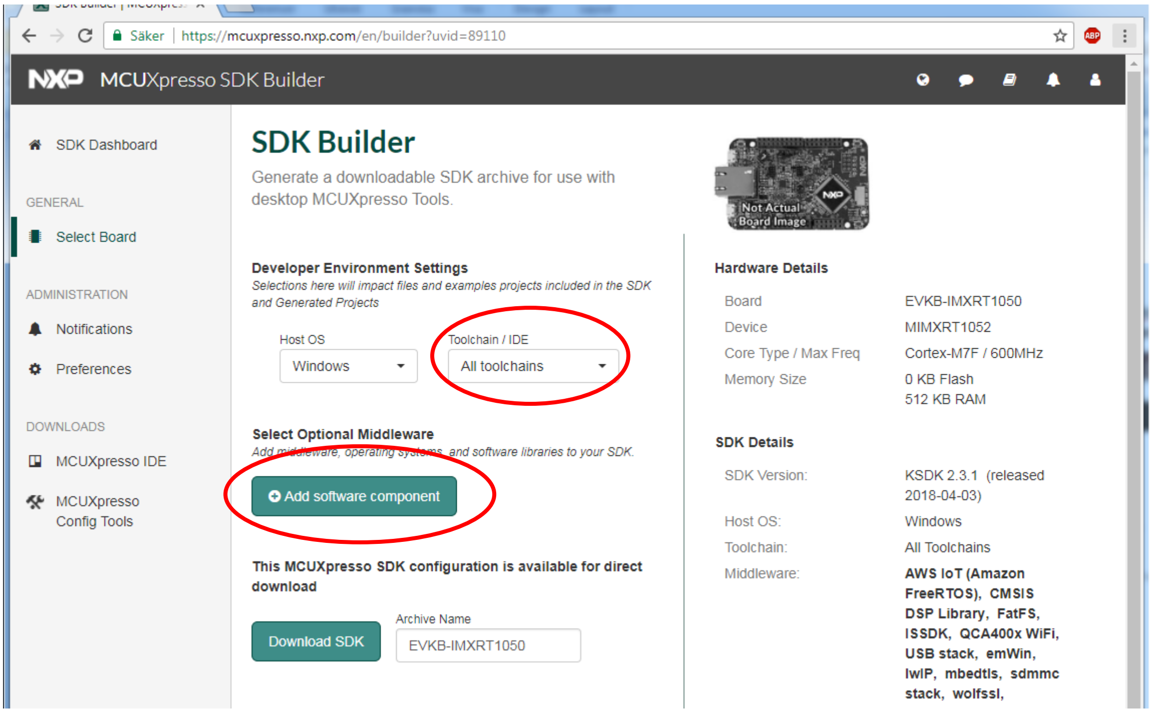
→ Build MCUXpresso SDK

Explore selection with Clocks tool

Explore selection with Pins tool

Figure 3 – MCUXpresso SDK Builder - Build MCUXpresso SDK

Select *All Toolchains* under the *Toolchain/IDE* dropdown list.



Säker | https://mcuxpresso.nxp.com/en/builder?uvid=89110

NXP MCUXpresso SDK Builder

SDK Dashboard

GENERAL

Select Board

ADMINISTRATION

Notifications

Preferences

DOWNLOADS

MCUXpresso IDE

MCUXpresso Config Tools

SDK Builder

Generate a downloadable SDK archive for use with desktop MCUXpresso Tools.

Developer Environment Settings

Selections here will impact files and examples projects included in the SDK and Generated Projects

Host OS: Windows

Toolchain / IDE: All toolchains

Select Optional Middleware

Add middleware, operating systems, and software libraries to your SDK.

Add software component

This MCUXpresso SDK configuration is available for direct download

Download SDK

Archive Name: EVKB-IMXRT1050

Not Actual Board Image

Hardware Details

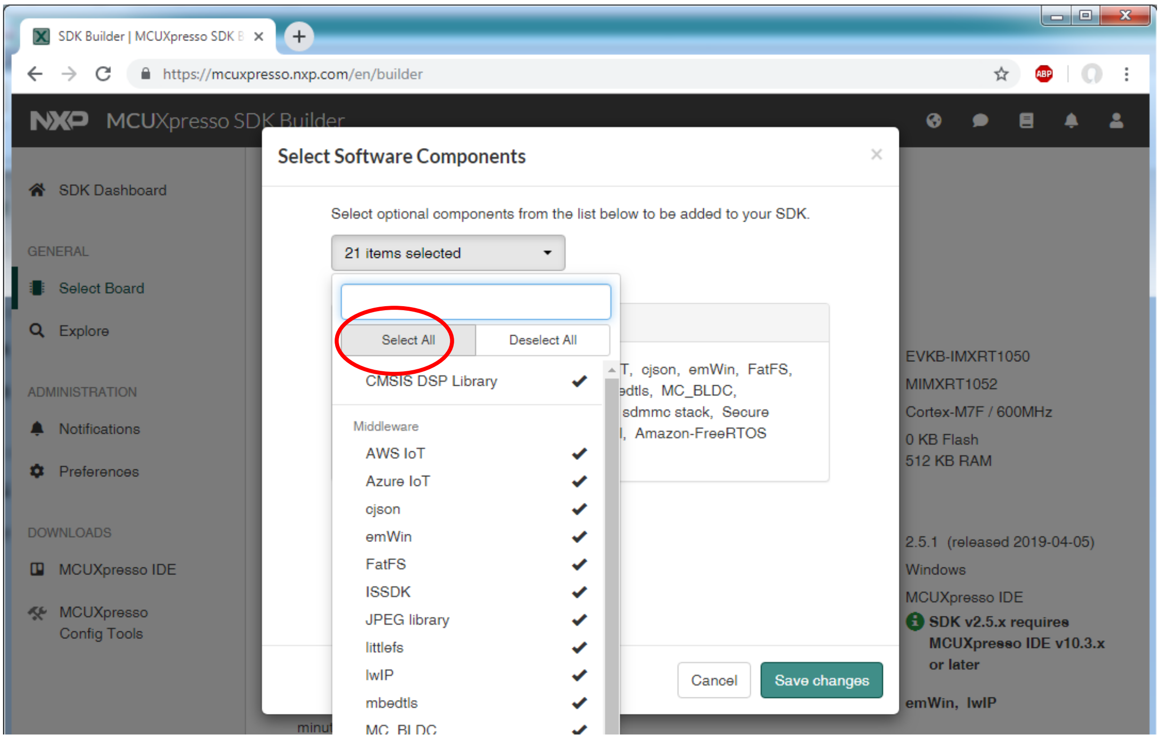
Board	EVKB-IMXRT1050
Device	MIMXRT1052
Core Type / Max Freq	Cortex-M7F / 600MHz
Memory Size	0 KB Flash 512 KB RAM

SDK Details

SDK Version:	KSDK 2.3.1 (released 2018-04-03)
Host OS:	Windows
Toolchain:	All Toolchains
Middleware:	**AWS IoT (Amazon FreeRTOS), CMSIS DSP Library, FatFS, ISSDK, QCA400x WiFi, USB stack, emWin, lwIP, mbedtls, sdmmc stack, wolfssl,**

Figure 4 – MCUXpresso SDK Builder - Select Toolchain

Click the *Add software component* button and then choose the *Select All* option in the dropdown that appears. Save the changes.



Select Software Components

Select optional components from the list below to be added to your SDK.

21 items selected

Select All | Deselect All

- CMSIS DSP Library ✓
- Middleware
- AWS IoT ✓
- Azure IoT ✓
- cjson ✓
- emWin ✓
- FatFS ✓
- ISSDK ✓
- JPEG library ✓
- littlefs ✓
- lwIP ✓
- mbedtls ✓
- MC_BLDC ✓

T, cjson, emWin, FatFS, edtls, MC_BLDC, sdmmc stack, Secure l, Amazon-FreeRTOS

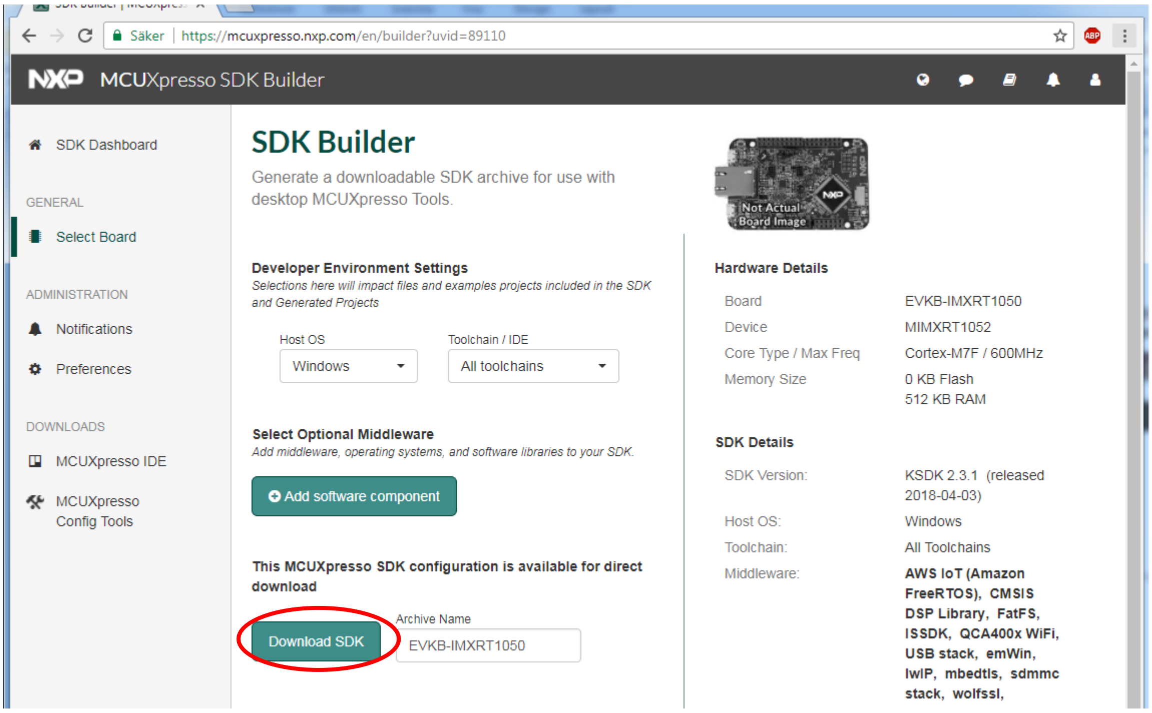
EVKB-IMXRT1050
MIMXRT1052
Cortex-M7F / 600MHz
0 KB Flash
512 KB RAM

2.5.1 (released 2019-04-05)
Windows
MCUXpresso IDE
SDK v2.5.x requires MCUXpresso IDE v10.3.x or later

emWin, lwIP

Figure 5 – MCUXpresso SDK Builder - Configure Software Components

Press *Download SDK* button. This will give you a complete package with all software.



SDK Builder

Generate a downloadable SDK archive for use with desktop MCUXpresso Tools.

Developer Environment Settings

Selections here will impact files and examples projects included in the SDK and Generated Projects

Host OS: Windows

Toolchain / IDE: All toolchains

Select Optional Middleware

Add middleware, operating systems, and software libraries to your SDK.

This MCUXpresso SDK configuration is available for direct download

Archive Name: EVKB-IMXRT1050

Hardware Details

Board	EVKB-IMXRT1050
Device	MIMXRT1052
Core Type / Max Freq	Cortex-M7F / 600MHz
Memory Size	0 KB Flash 512 KB RAM

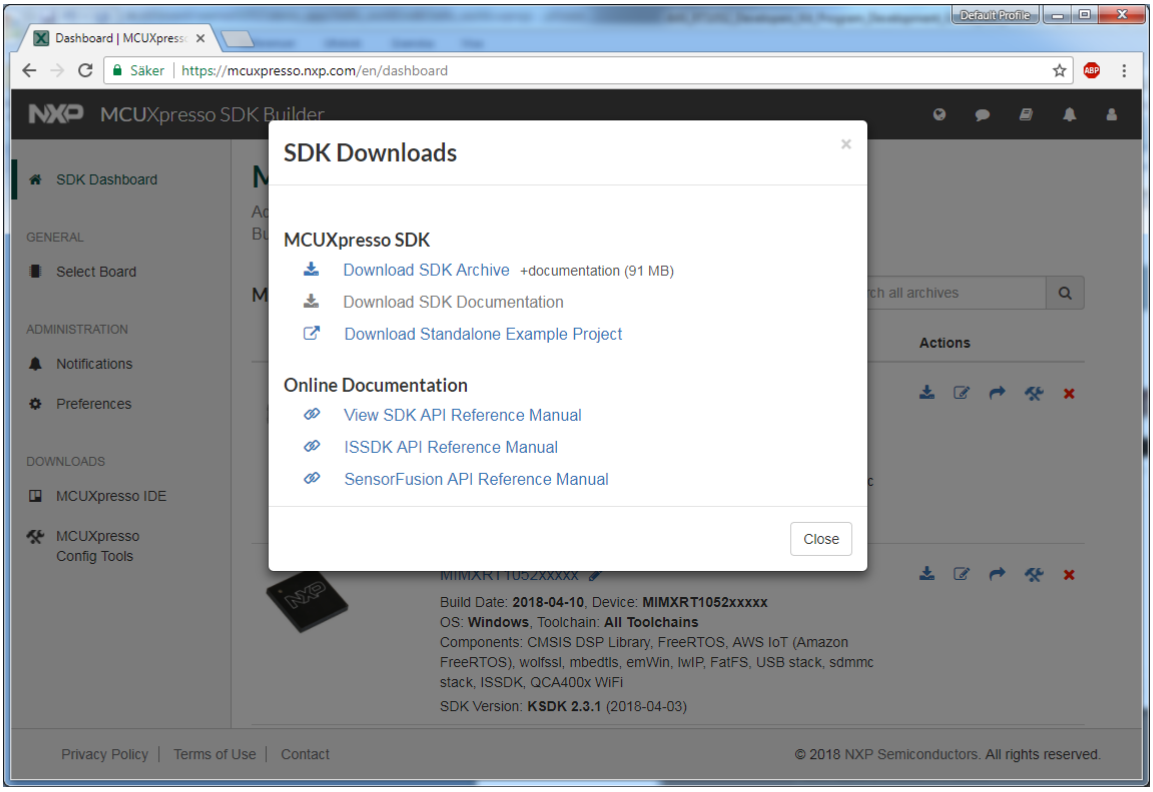
SDK Details

SDK Version:	KSDK 2.3.1 (released 2018-04-03)
Host OS:	Windows
Toolchain:	All Toolchains
Middleware:	**AWS IoT (Amazon FreeRTOS), CMSIS DSP Library, FatFS, ISSDK, QCA400x WiFi, USB stack, emWin, lwIP, mbedtls, sdmmc stack, wolfssl,**

Figure 6 – MCUXpresso SDK Builder - SDK Builder

Also agree to the *Software Terms and Conditions* by clicking on the *I Agree* button.

Download of a (about) 91MByte zip-file will begin.



SDK Downloads

MCUXpresso SDK

- Download SDK Archive +documentation (91 MB)
- Download SDK Documentation
- Download Standalone Example Project

Online Documentation

- View SDK API Reference Manual
- ISSDK API Reference Manual
- SensorFusion API Reference Manual

MIMXRT1052xxxxx

Build Date: **2018-04-10**, Device: **MIMXRT1052xxxxx**
OS: **Windows**, Toolchain: **All Toolchains**
Components: CMSIS DSP Library, FreeRTOS, AWS IoT (Amazon FreeRTOS), wolfssl, mbedtls, emWin, lwIP, FatFS, USB stack, sdmmc stack, ISSDK, QCA400x WiFi
SDK Version: **KSDK 2.3.1** (2018-04-03)

Privacy Policy | Terms of Use | Contact © 2018 NXP Semiconductors. All rights reserved.

Figure 7 – MCUXpresso SDK Builder - SDK Downloads

Save the file on your computer. Note the download location. As of April 2019, the file name is `SDK_2.5.1_EVKB-IMXRT1050.zip` or `SDK_2.5.1_EVK-MIMXRT1060.zip` but the version number (i.e., 2.5.1) will likely increment over time. Extract the content of the zip-file to the desired installation location in your file system. Note that the installation root shall have a short path. For example: `c:\NXP_SDK\`

2.2 Modify the SDK to Support iMX RT Developer's Kit

This section describes how to modify the previously downloaded SDK to be compatible with the features of the *iMX RT Developer's Kit*, like the EcoXiP.

The SDK supports hyper flash but the *iMX RT Developer's Kit* has EcoXip flash from Adesto. Ideally the EcoXip flash driver should be placed in a new file with _ecoxip in the name but that would require changes to every project for every IDE in the SDK and that is a huge task. The solution is instead to replace the file content but keep the filename. This way none of the projects have to be modified.

Download the zip-file from the *Resources* tab on the product page on Embedded Artists' website. The name of the file is:
`imxrt1052_ea_files_<date>.zip`, or
`imxrt1062_ea_files_<date>.zip`

Unpack the downloaded zip-file in a temporary directory, for example: **`c:/temp/ea_files`**

Copy

`c:/temp/ea_files/evkbimxrt1050_flexspi_nor_config.c` , or
`c:/temp/ea_files/evkmimxrt1060_flexspi_nor_config.c`

to

`<SDK_Install_Directory>/boards/evkbimxrt1050/xip/`, or
`<SDK_Install_Directory>/boards/evkmimxrt1060/xip/`

replacing the existing file. This will add support for the EcoXip.

2.3 SDK Project Comments

The MCUXpresso SDK comes with a long list of example application code. To see what's available, browse to the boards folder of your SDK installation and select your board:
`<SDK_Install_Directory>/boards/evkbimxrt1050`, or
`<SDK_Install_Directory>/boards/evkmimxrt1060`

Only use the projects found under this sub-directory.

To learn more about specific example code, open the readme.txt file in an example's directory.

The following sections will describe some projects that will require extra configuration to work.

2.3.1 USB Examples

The USB interface is different on the NXP EVK that the SDK is for and the *iMX RT Developer's Kit.* The USB examples all have a block of code (most of the time in app.h) to define the CONTROLLER_ID like this:

```
#if ((defined USB_HOST_CONFIG_KHCI) && (USB_HOST_CONFIG_KHCI))
#define CONTROLLER_ID kUSB_ControllerKhci0
#endif /* USB_HOST_CONFIG_KHCI */
#if ((defined USB_HOST_CONFIG_EHCI) && (USB_HOST_CONFIG_EHCI))
#define CONTROLLER_ID kUSB_ControllerEhci0
#endif /* USB_HOST_CONFIG_EHCI */
#if ((defined USB_HOST_CONFIG_OHCI) && (USB_HOST_CONFIG_OHCI))
#define CONTROLLER_ID kUSB_ControllerOhci0
#endif /* USB_HOST_CONFIG_OHCI */
#if ((defined USB_HOST_CONFIG_IP3516HS) && (USB_HOST_CONFIG_IP3516HS))
#define CONTROLLER_ID kUSB_ControllerIp3516Hs0
#endif /* USB_HOST_CONFIG_IP3516HS */
```

If the USB example is not working, then try to change the 0 (marked in red above) to a 1 and test again. Note that in version 2.5.0 of the SDK several of the examples work without the modification as the code is written to enable both USB interfaces.

2.3.2 Network Examples

The *iMX RT1052 OEM and iMX RT1062 OEM* board have an EEPROM with a unique id that can be used as MAC address in the network examples, found in the demo_apps/lwip directory. Follow these steps to use it:

1. Start by copying the mac_addr.c and mac_addr.h files from the imxrt1052_ea_files_<date>.zip / imxrt1062_ea_files_<date>.zip archive in section 2.2 into your project
2. Add the mac_addr.c file to the project. How this is done depends on the IDE but for Keil uVision, right click on the *source* folder in the project and select *Add existing files to group 'source'.*
3. Add the I2C driver to the project the same way as in the previous step. The driver is called fsl_lpi2c and can be found in the <SDK_Install_Directory>/devices/MIMXRT1052/drivers/ folder or <SDK_Install_Directory>/devices/MIMXRT1062/drivers/ folder.
4. Add `#include "mac_addr.h"` to main.c after the other include statements
5. Find the declaration in the main.c file that looks like this:

```
    ethernetif_config_t fsl_enet_config0 = {
        .phyAddress = EXAMPLE_PHY_ADDRESS,
        .clockName = EXAMPLE_CLOCK_NAME,
        .macAddress = configMAC_ADDR,
    };
```

   and add this line after it:

```
MAC_Read(fsl_enet_config0.macAddress);
```

Now the program will attempt to read the MAC address from the EEPROM before the LWIP stack is initialized.

2.3.3 Wi-Fi Examples

The *iMX RT Developer's Kit* support M.2 modules. As of April 2019 the SDK contained two projects that support this module: *wiced_iperf* and *wiced_mfg_test.* The hardware setup is a bit different for the *iMX RT Developer's Kit* and a number of files must be copied into the project from the imxrt1052_ea_files_<date>.zip / imxrt1062_ea_files_<date>.zip archive in section 2.2

File to copy	Destination	Comment
pin_mux.c	board/	Replace existing file
fsl_lpi2c.c	drivers/	New file
fsl_lpi2c.h	drivers/	New file
pca6416.c	source/	New file
pca6416.h	source/	New file
wwd_platform.c	43xxx_wifi/WICED/platform/MCU/LPC/WWD/	Replace existing file
wwd_SDIO.c	43xxx_wifi/WICED/platform/MCU/LPC/WWD/	Replace existing file

After copying the files into the project the IDE must be told to look for the new files. How this is done depends on the IDE but for Keil uVision, right click on for example the *source* folder in the project and select *Add existing files to group 'source'.*

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 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 as installed in chapter 2 . It is assumed that you have this development environment installed on your computer.

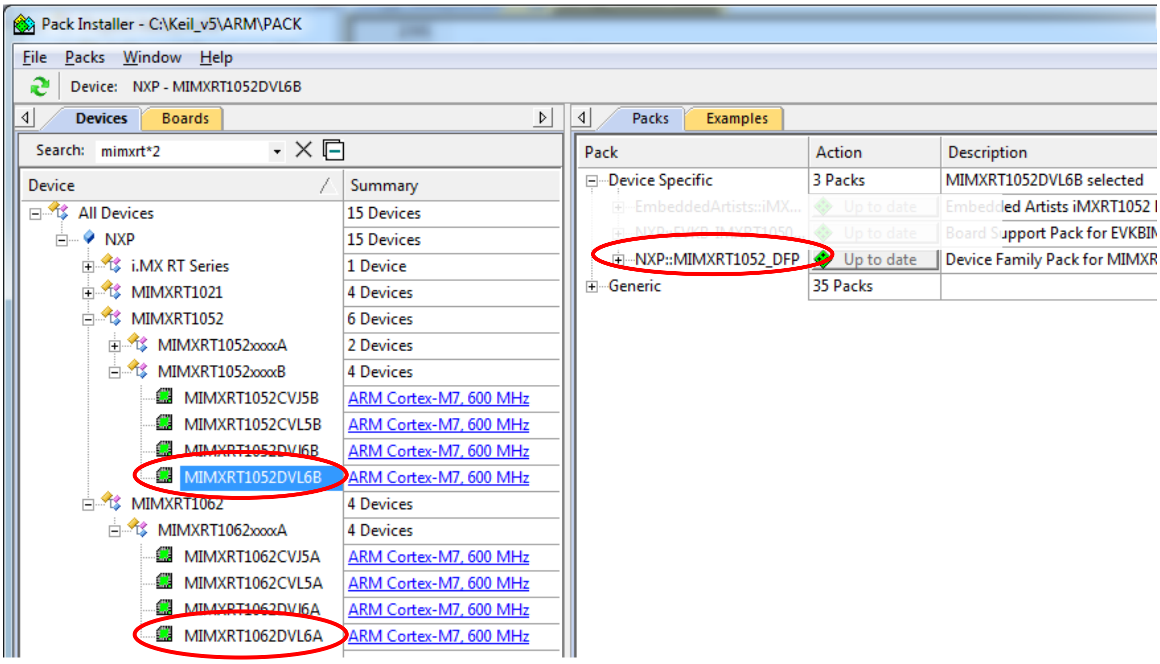
Following the instructions in this section will make it possible to run all examples in the downloaded SDK. Note that none of the examples use the RTE (Run-Time Environment) feature of the Keil uVision/MDK. The use of RTE is described in chapter 4 .

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

3.1 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→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.



Pack Installer - C:\Keil_v5\ARM\PACK

Device: NXP - MIMXRT1052DVL6B

Search: mimxrt*2

Device	Summary
All Devices	15 Devices
NXP	15 Devices
i.MX RT Series	1 Device
MIMXRT1021	4 Devices
MIMXRT1052	6 Devices
MIMXRT1052xxxxA	2 Devices
MIMXRT1052xxxxB	4 Devices
MIMXRT1052CVJ5B	ARM Cortex-M7, 600 MHz
MIMXRT1052CVL5B	ARM Cortex-M7, 600 MHz
MIMXRT1052DVJ6B	ARM Cortex-M7, 600 MHz
MIMXRT1052DVL6B	ARM Cortex-M7, 600 MHz
MIMXRT1062	4 Devices
MIMXRT1062xxxxA	4 Devices
MIMXRT1062CVJ5A	ARM Cortex-M7, 600 MHz
MIMXRT1062CVL5A	ARM Cortex-M7, 600 MHz
MIMXRT1062DVJ6A	ARM Cortex-M7, 600 MHz
MIMXRT1062DVL6A	ARM Cortex-M7, 600 MHz

Pack	Action	Description
Device Specific	3 Packs	MIMXRT1052DVL6B selected
EmbeddedArtists::iMX...	Up to date	Embedded Artists iMXRT1052 I
NXP::EVKB-IMXRT1050...	Up to date	Board Support Pack for EVKBIM
NXP::MIMXRT1052_DFP	Up to date	Device Family Pack for MIMXR
Generic	35 Packs	

3.2 Build an Example Application

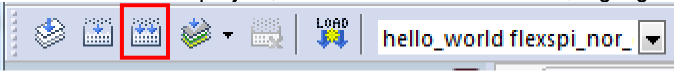
The following steps will guide you through opening the hello_world application. These steps may change slightly for other example applications as some of these applications may have additional layers of folders in their path.

1. If not already done, open the desired example application workspace in:
`<install_dir>/boards/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`

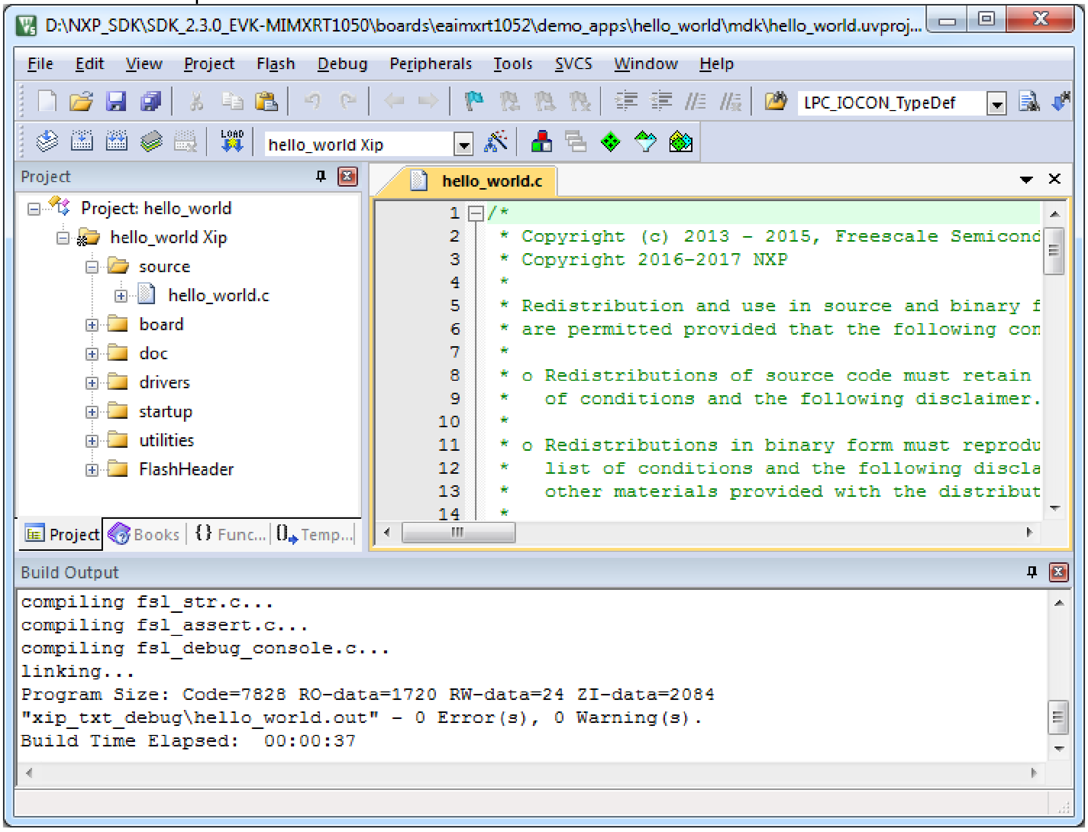
   or

   `<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.


D:\NXP_SDK\SDK_2.3.0_EVK-MIMXRT1050\boards\eaimxrt1052\demo_apps\hello_world\mdk\hello_world.uvproj...
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
LPC_IOCON_TypeDef
LOAD
hello_world Xip
Project
Project: hello_world
hello_world Xip
source
hello_world.c
board
doc
drivers
startup
utilities
FlashHeader
Project Books {} Func... 0→ Temp...
hello_world.c
Build Output

```
/*
 * Copyright (c) 2013 - 2015, Freescale Semicond
 * Copyright 2016-2017 NXP
 *
 * Redistribution and use in source and binary f
 * are permitted provided that the following con
 *
 * o Redistributions of source code must retain
 *   of conditions and the following disclaimer.
 *
 * o Redistributions in binary form must reprodu
 *   list of conditions and the following discla
 *   other materials provided with the distribut
 *
```

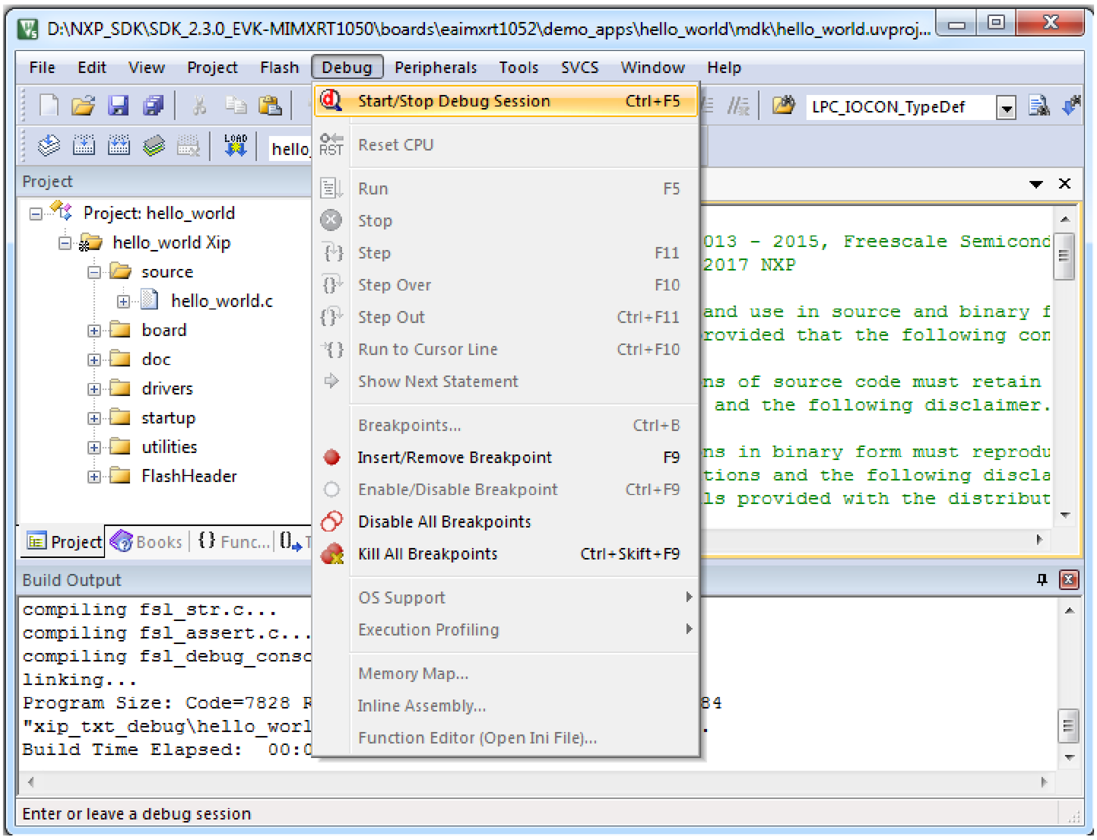
```
compiling fsl_str.c...
compiling fsl_assert.c...
compiling fsl_debug_console.c...
linking...
Program Size: Code=7828 RO-data=1720 RW-data=24 ZI-data=2084
"xip_txt_debug\hello_world.out" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:37
```

3.3 Run an Example Application

To download and run the application, perform these steps:

4. Connect the debug probe to development platform to your PC via USB cable. See chapter 7 for details how to connect the debug probe.

5. Select the *Debug* menu and then *Start/Stop Debug Session*, or simply press Ctrl+F5. The application will then be downloaded into SRAM.


D:\NXP_SDK\SDK_2.3.0_EVK-MIMXRT1050\boards\eaimxrt1052\demo_apps\hello_world\mdk\hello_world.uvproj...

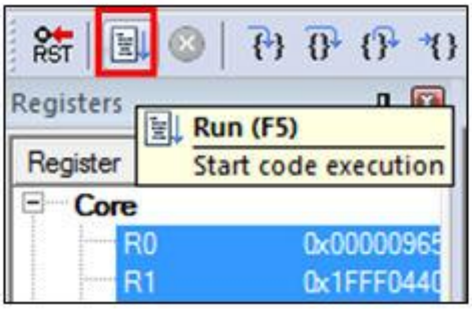
```
013 - 2015, Freescale Semicond
2017 NXP

and use in source and binary f
rovided that the following con

ns of source code must retain
 and the following disclaimer.

ns in binary form must reprodu
tions and the following discla
ls provided with the distribut
```

```
compiling fsl_str.c...
compiling fsl_assert.c...
compiling fsl_debug_consc
linking...
Program Size: Code=7828 R                                       84
"xip_txt_debug\hello_worl                                       .
Build Time Elapsed:   00:0
```

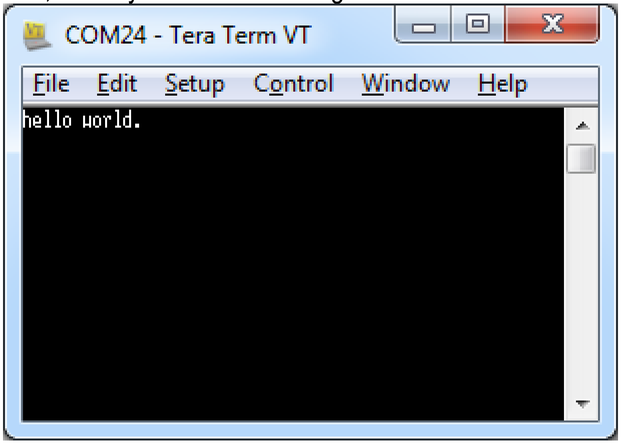
Enter or leave a debug session

6. 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`

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



8. 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.



```
hello world.
```

3.4 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.

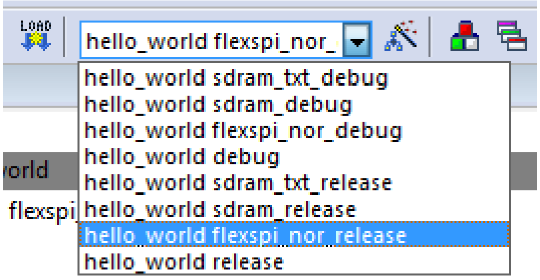

LOAD
hello_world flexspi_nor_
hello_world sdram_txt_debug
hello_world sdram_debug
hello_world flexspi_nor_debug
hello_world debug
hello_world sdram_txt_release
hello_world sdram_release
hello_world flexspi_nor_release
hello_world release
orld
flexspi

Figure 8 – Build Targets in Keil uVision/MDK

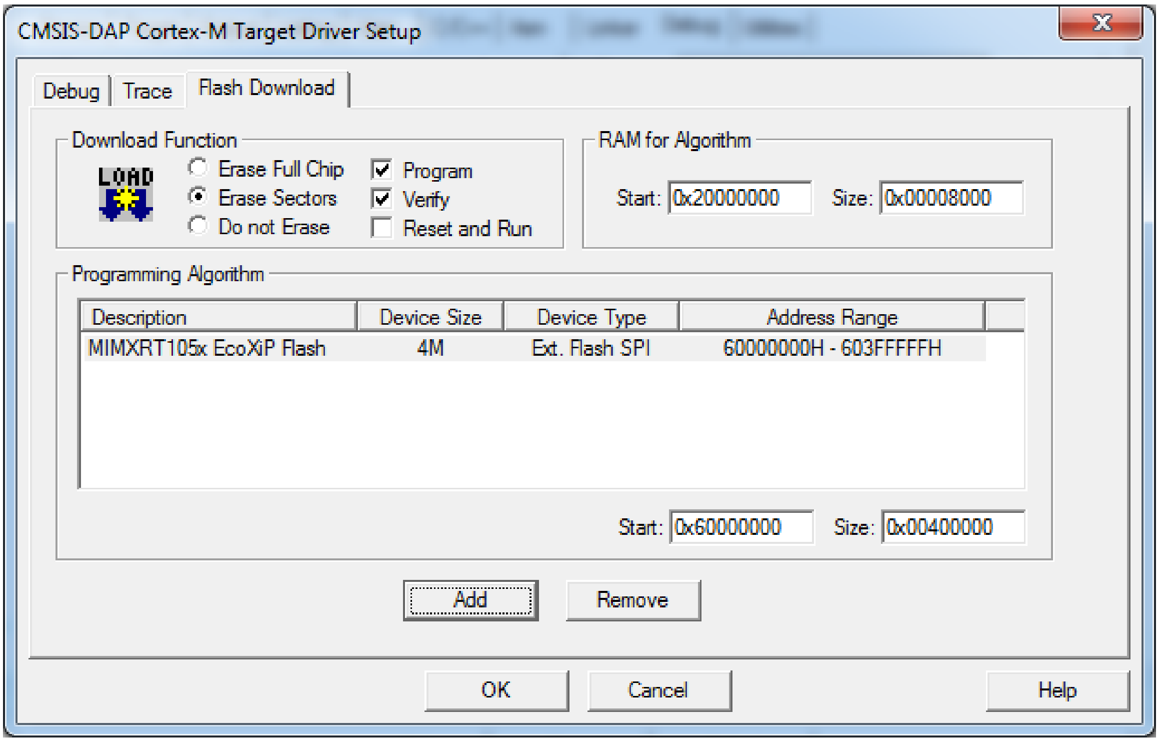
The target name starts with the project name, **hello_world** in this case, and then the nomenclature as outlined in the table below is used. Release targets have a higher compiler optimization degree making the target less suitable for debugging.

<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. Need extra configuration, see section 3.5 .

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.

3.5 Configuring Flash Algorithm

To setup a project for storing/executing the application from the EcoXiP 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 and replace the existing driver with the EcoXiP flash driver as shown below.



Download Function

○ Erase Full Chip
● Erase Sectors
○ Do not Erase

☑ Program
☑ Verify
☐ Reset and Run

RAM for Algorithm

Start: 0x20000000 Size: 0x00008000

Programming Algorithm

Description	Device Size	Device Type	Address Range
MIMXRT105x EcoXiP Flash	4M	Ext. Flash SPI	60000000H - 603FFFFFH

Start: 0x60000000 Size: 0x00400000

Figure 9 – Keil uVision/MDK Flash Algorithm Settings

3.6 Build Output

Building a project results in one important file:

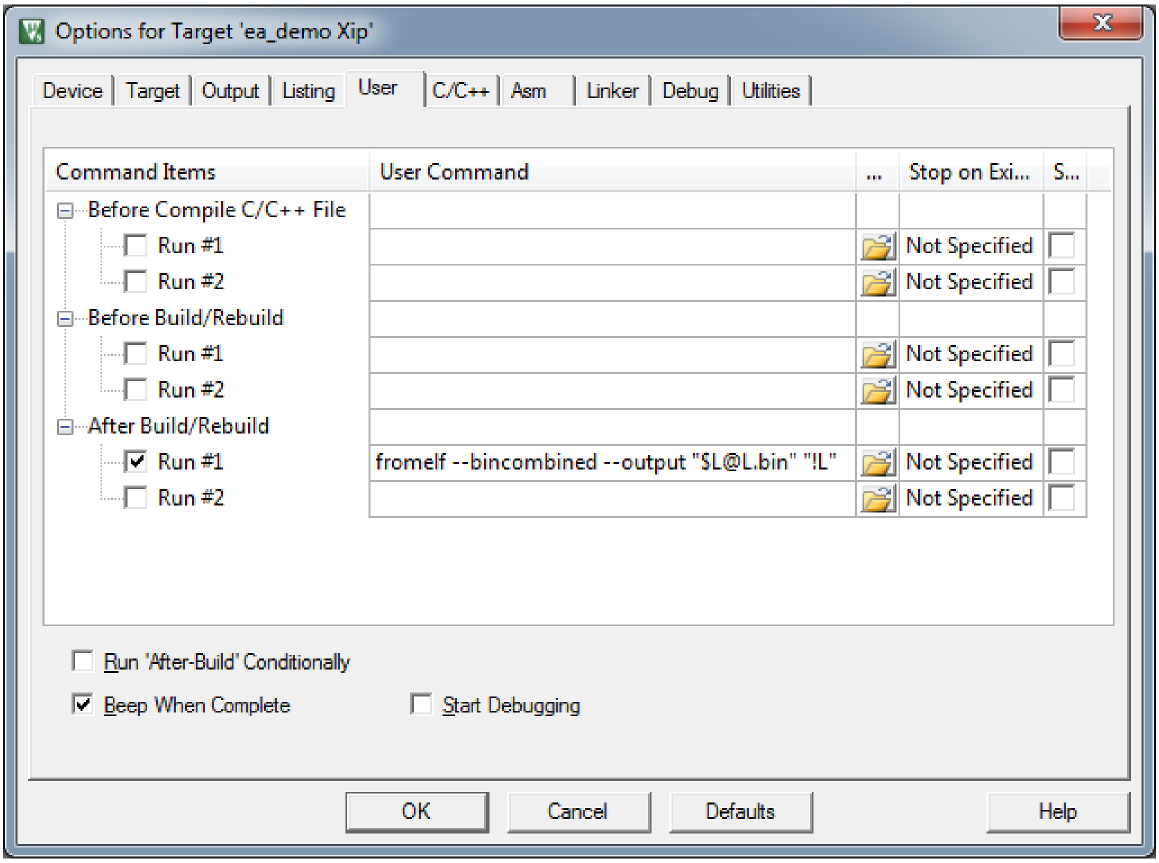
- An `*.out` file (for example `ea_demo.out`) which is perfect when debugging or using the debugger to flash the program

If the goal is to use the MfgTool to download the program (as explained in chapter 8) then another file is more important:

- An `*.bin` file (for example `ea_demo.bin`)

To get the `*.bin` file created automatically, add a User Command, as shown in the picture below. The command is:

```
fromelf --bincombined --output "$L@L.bin" "!L"
```



Command Items	User Command	...	Stop on Exi...	S...
⊟ Before Compile C/C++ File				
□ Run #1			Not Specified	□
□ Run #2			Not Specified	□
⊟ Before Build/Rebuild				
□ Run #1			Not Specified	□
□ Run #2			Not Specified	□
⊟ After Build/Rebuild				
☑ Run #1	fromelf --bincombined --output "$L@L.bin" "!L"		Not Specified	□
□ Run #2			Not Specified	□

□ Run 'After-Build' Conditionally

☑ Beep When Complete □ Start Debugging

Figure 10 – After Build Command to Create BIN File in Keil uVision/MDK

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 chapter 2 . 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.

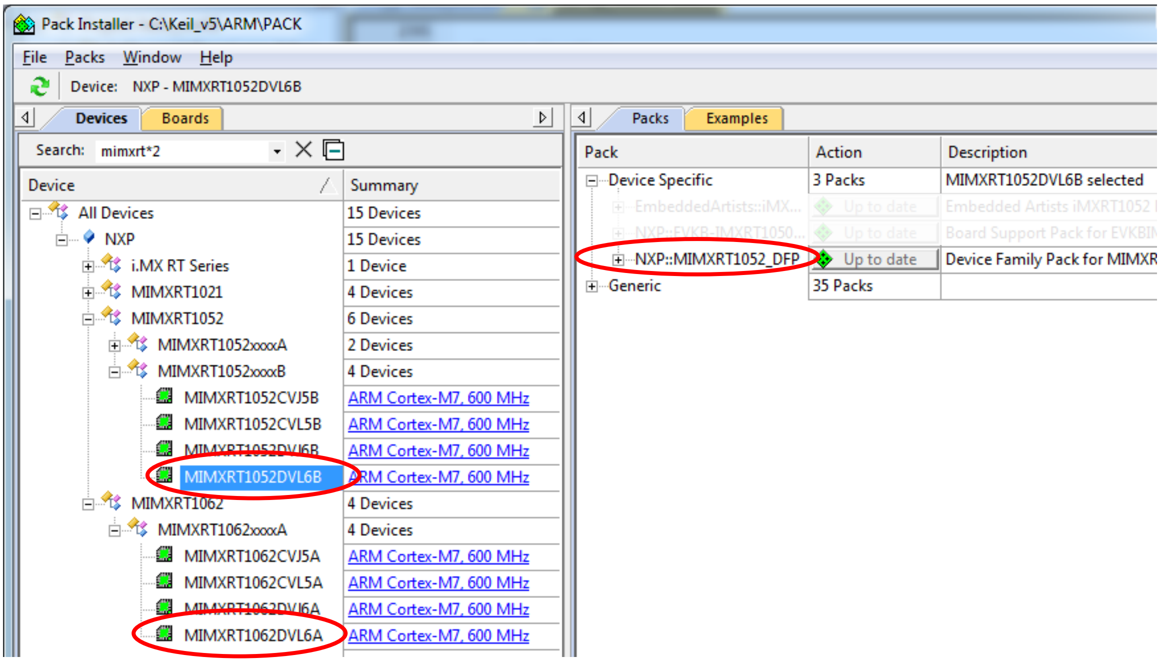
The recommendation is to use the SDK unless you know exactly what you are doing. The use of the SDK is described in chapter 3 .

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



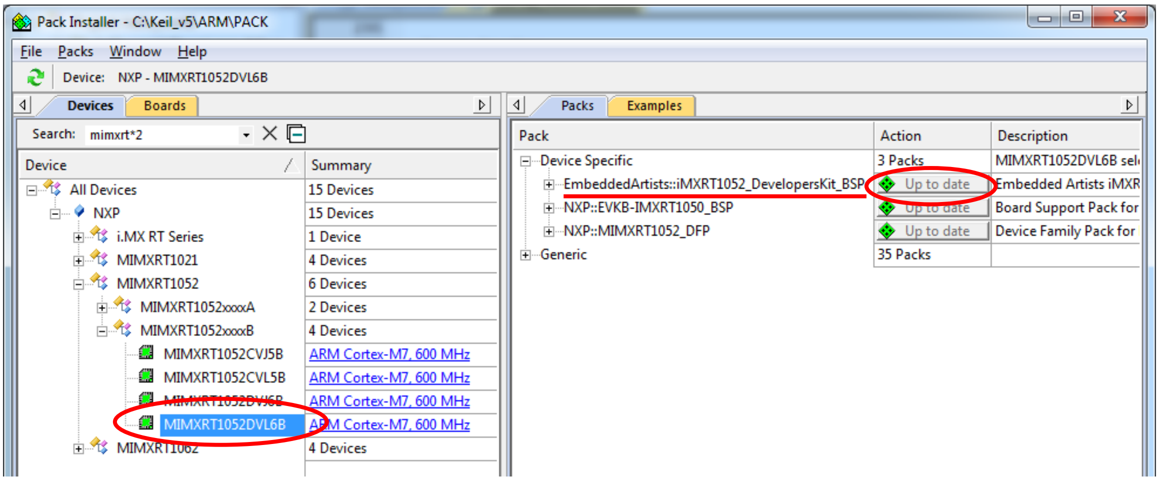
Device: NXP - MIMXRT1052DVL6B

Search: mimxrt*2

Device	Summary
All Devices	15 Devices
NXP	15 Devices
i.MX RT Series	1 Device
MIMXRT1021	4 Devices
MIMXRT1052	6 Devices
MIMXRT1052xxxxA	2 Devices
MIMXRT1052xxxxB	4 Devices
MIMXRT1052CVJ5B	ARM Cortex-M7, 600 MHz
MIMXRT1052CVL5B	ARM Cortex-M7, 600 MHz
MIMXRT1052DVJ6B	ARM Cortex-M7, 600 MHz
MIMXRT1052DVL6B	ARM Cortex-M7, 600 MHz
MIMXRT1062	4 Devices
MIMXRT1062xxxxA	4 Devices
MIMXRT1062CVJ5A	ARM Cortex-M7, 600 MHz
MIMXRT1062CVL5A	ARM Cortex-M7, 600 MHz
MIMXRT1062DVJ6A	ARM Cortex-M7, 600 MHz
MIMXRT1062DVL6A	ARM Cortex-M7, 600 MHz

Pack	Action	Description
Device Specific	3 Packs	MIMXRT1052DVL6B selected
EmbeddedArtists::iMX...	Up to date	Embedded Artists iMXRT1052 I
NXP::EVKB-IMXRT1050...	Up to date	Board Support Pack for EVKBIM
NXP::MIMXRT1052_DFP	Up to date	Device Family Pack for MIMXR
Generic	35 Packs	

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



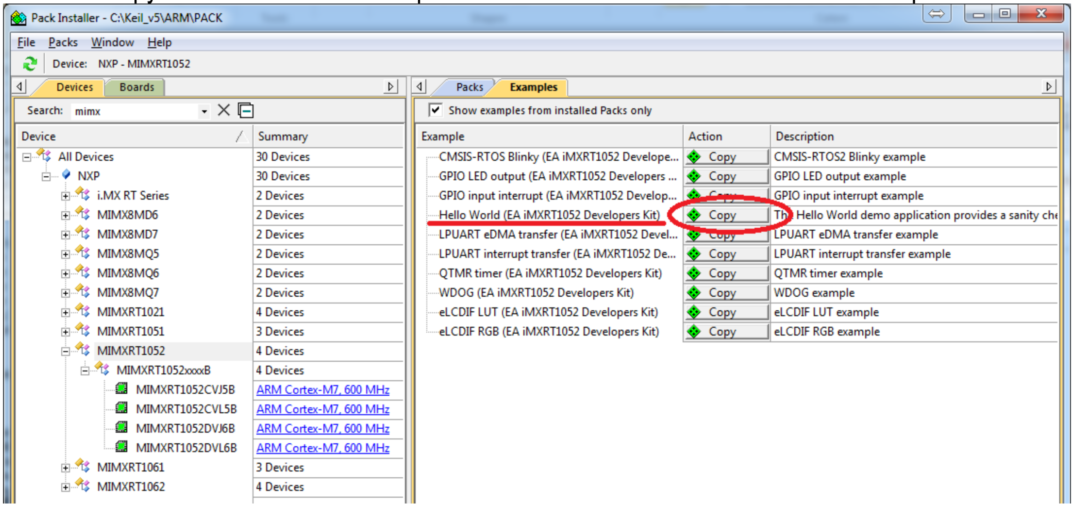
Device	Summary
All Devices	15 Devices
NXP	15 Devices
i.MX RT Series	1 Device
MIMXRT1021	4 Devices
MIMXRT1052	6 Devices
MIMXRT1052xxxxA	2 Devices
MIMXRT1052xxxxB	4 Devices
MIMXRT1052CVJ5B	ARM Cortex-M7, 600 MHz
MIMXRT1052CVL5B	ARM Cortex-M7, 600 MHz
MIMXRT1052DVJ6B	ARM Cortex-M7, 600 MHz
MIMXRT1052DVL6B	ARM Cortex-M7, 600 MHz
MIMXRT1062	4 Devices

Pack	Action	Description
Device Specific	3 Packs	MIMXRT1052DVL6B sel
EmbeddedArtists::iMXRT1052_DevelopersKit_BSP	Up to date	Embedded Artists iMXR
NXP::EVKB-IMXRT1050_BSP	Up to date	Board Support Pack for
NXP::MIMXRT1052_DFP	Up to date	Device Family Pack for
Generic	35 Packs	

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.

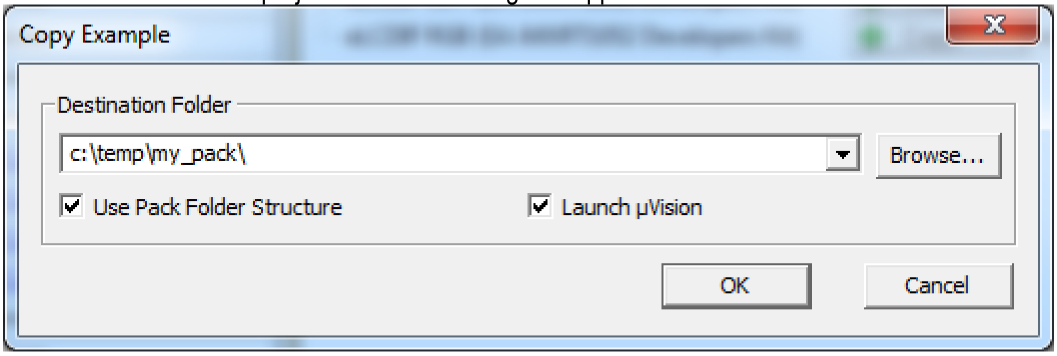


Device	Summary
All Devices	30 Devices
NXP	30 Devices
i.MX RT Series	2 Devices
MIMX8MD6	2 Devices
MIMX8MD7	2 Devices
MIMX8MQ5	2 Devices
MIMX8MQ6	2 Devices
MIMX8MQ7	2 Devices
MIMXRT1021	4 Devices
MIMXRT1051	3 Devices
MIMXRT1052	4 Devices
MIMXRT1052xxxxB	4 Devices
MIMXRT1052CVJ5B	ARM Cortex-M7, 600 MHz
MIMXRT1052CVL5B	ARM Cortex-M7, 600 MHz
MIMXRT1052DVJ6B	ARM Cortex-M7, 600 MHz
MIMXRT1052DVL6B	ARM Cortex-M7, 600 MHz
MIMXRT1061	3 Devices
MIMXRT1062	4 Devices

☑ Show examples from installed Packs only

Example	Action	Description
CMSIS-RTOS Blinky (EA iMXRT1052 Develope...	Copy	CMSIS-RTOS2 Blinky example
GPIO LED output (EA iMXRT1052 Developers ...	Copy	GPIO LED output example
GPIO input interrupt (EA iMXRT1052 Develop...	Copy	GPIO input interrupt example
Hello World (EA iMXRT1052 Developers Kit)	Copy	The Hello World demo application provides a sanity che
LPUART eDMA transfer (EA iMXRT1052 Devel...	Copy	LPUART eDMA transfer example
LPUART interrupt transfer (EA iMXRT1052 De...	Copy	LPUART interrupt transfer example
QTMR timer (EA iMXRT1052 Developers Kit)	Copy	QTMR timer example
WDOG (EA iMXRT1052 Developers Kit)	Copy	WDOG example
eLCDIF LUT (EA iMXRT1052 Developers Kit)	Copy	eLCDIF LUT example
eLCDIF RGB (EA iMXRT1052 Developers Kit)	Copy	eLCDIF RGB example

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

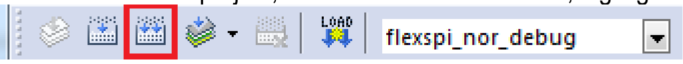


Destination Folder: c:\temp\my_pack\

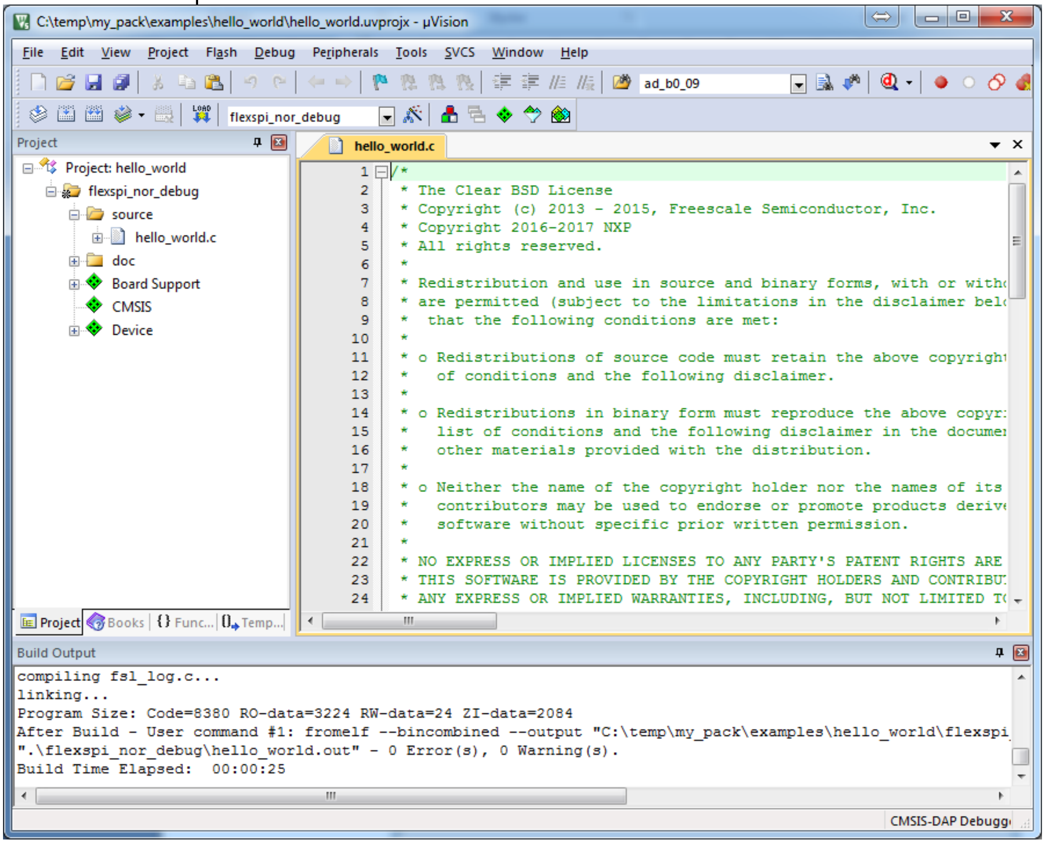
☑ Use Pack Folder Structure ☑ Launch µVision

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.



```
/*
 * The Clear BSD License
 * Copyright (c) 2013 - 2015, Freescale Semiconductor, Inc.
 * Copyright 2016-2017 NXP
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or witho
 * are permitted (subject to the limitations in the disclaimer belo
 *  that the following conditions are met:
 *
 * o Redistributions of source code must retain the above copyright
 *   of conditions and the following disclaimer.
 *
 * o Redistributions in binary form must reproduce the above copyr:
 *   list of conditions and the following disclaimer in the documen
 *   other materials provided with the distribution.
 *
 * o Neither the name of the copyright holder nor the names of its
 *   contributors may be used to endorse or promote products derive
 *   software without specific prior written permission.
 *
 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT RIGHTS ARE
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBU
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO
```

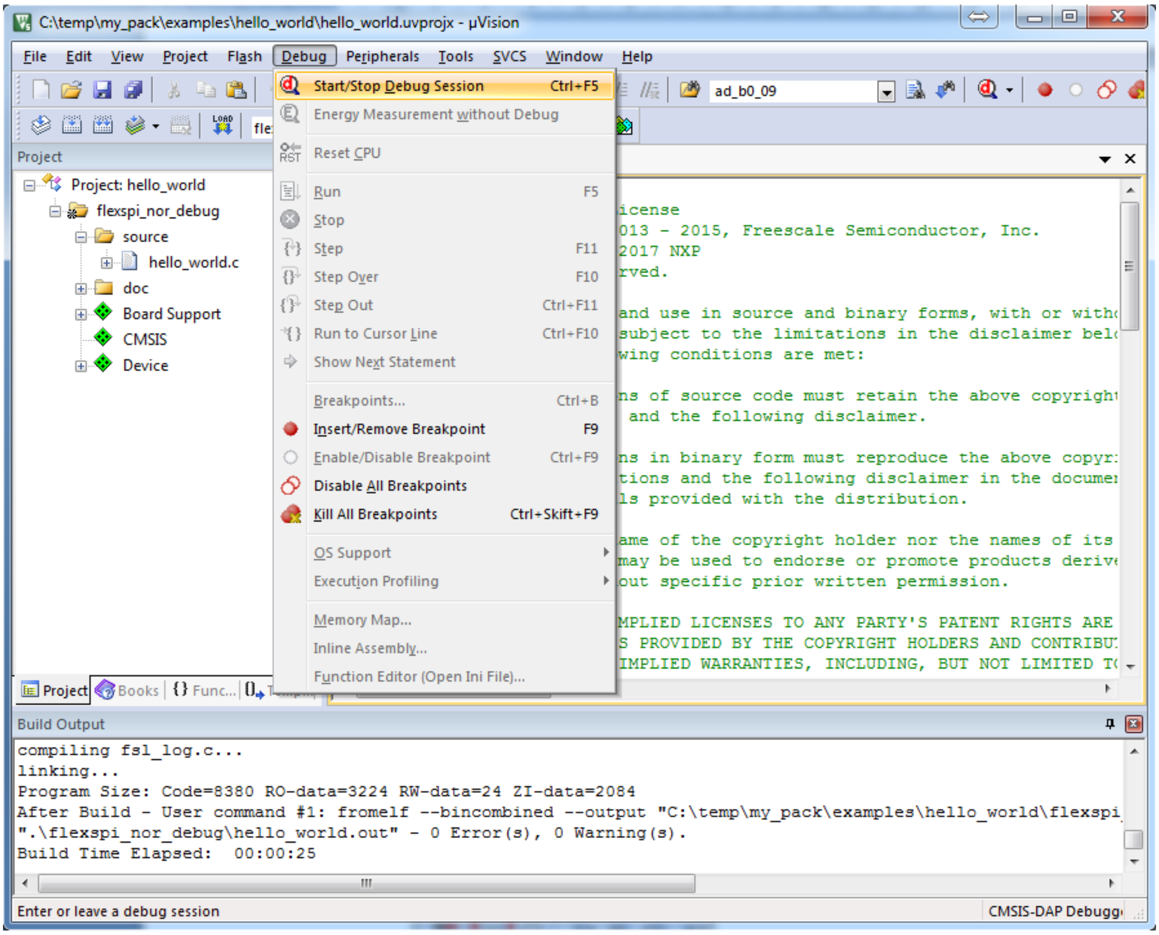
```
compiling fsl_log.c...
linking...
Program Size: Code=8380 RO-data=3224 RW-data=24 ZI-data=2084
After Build - User command #1: fromelf --bincombined --output "C:\temp\my_pack\examples\hello_world\flexspi
".\flexspi_nor_debug\hello_world.out" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:25
```

4.3 Run an Example Application

To download and run the application, perform these steps:

1. Connect the debug probe to 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.

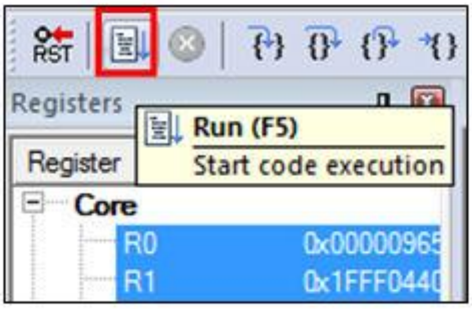

C:\temp\my_pack\examples\hello_world\hello_world.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Start/Stop Debug Session Ctrl+F5
Energy Measurement without Debug
Reset CPU
Run F5
Stop
Step F11
Step Over F10
Step Out Ctrl+F11
Run to Cursor Line Ctrl+F10
Show Next Statement
Breakpoints... Ctrl+B
Insert/Remove Breakpoint F9
Enable/Disable Breakpoint Ctrl+F9
Disable All Breakpoints
Kill All Breakpoints Ctrl+Skift+F9
OS Support
Execution Profiling
Memory Map...
Inline Assembly...
Function Editor (Open Ini File)...
ad_b0_09
Project
Project: hello_world
flexspi_nor_debug
source
hello_world.c
doc
Board Support
CMSIS
Device
Project Books {} Func... 0+ T...
Build Output
Enter or leave a debug session
CMSIS-DAP Debugg

```
icense
013 - 2015, Freescale Semiconductor, Inc.
2017 NXP
rved.

and use in source and binary forms, with or with
subject to the limitations in the disclaimer bel
wing conditions are met:

ns of source code must retain the above copyright
 and the following disclaimer.

ns in binary form must reproduce the above copyr
tions and the following disclaimer in the docume
ls provided with the distribution.

ame of the copyright holder nor the names of its
may be used to endorse or promote products deriv
out specific prior written permission.

MPLIED LICENSES TO ANY PARTY'S PATENT RIGHTS ARE
S PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBU
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED T
```

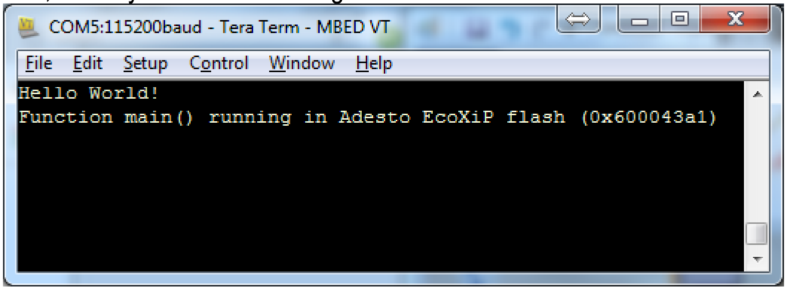
```
compiling fsl_log.c...
linking...
Program Size: Code=8380 RO-data=3224 RW-data=24 ZI-data=2084
After Build - User command #1: fromelf --bincombined --output "C:\temp\my_pack\examples\hello_world\flexspi
".\flexspi_nor_debug\hello_world.out" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:25
```

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.


COM5:115200baud - Tera Term - MBED VT
File Edit Setup Control Window Help

```
Hello World!
Function main() running in Adesto EcoXiP flash (0x600043a1)
```

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.

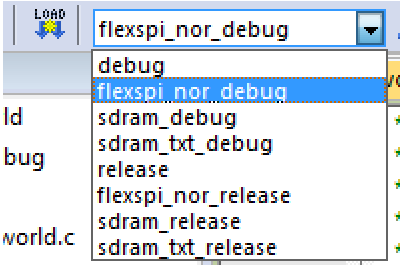


Figure 11 – Build Targets in Keil uVision/MDK

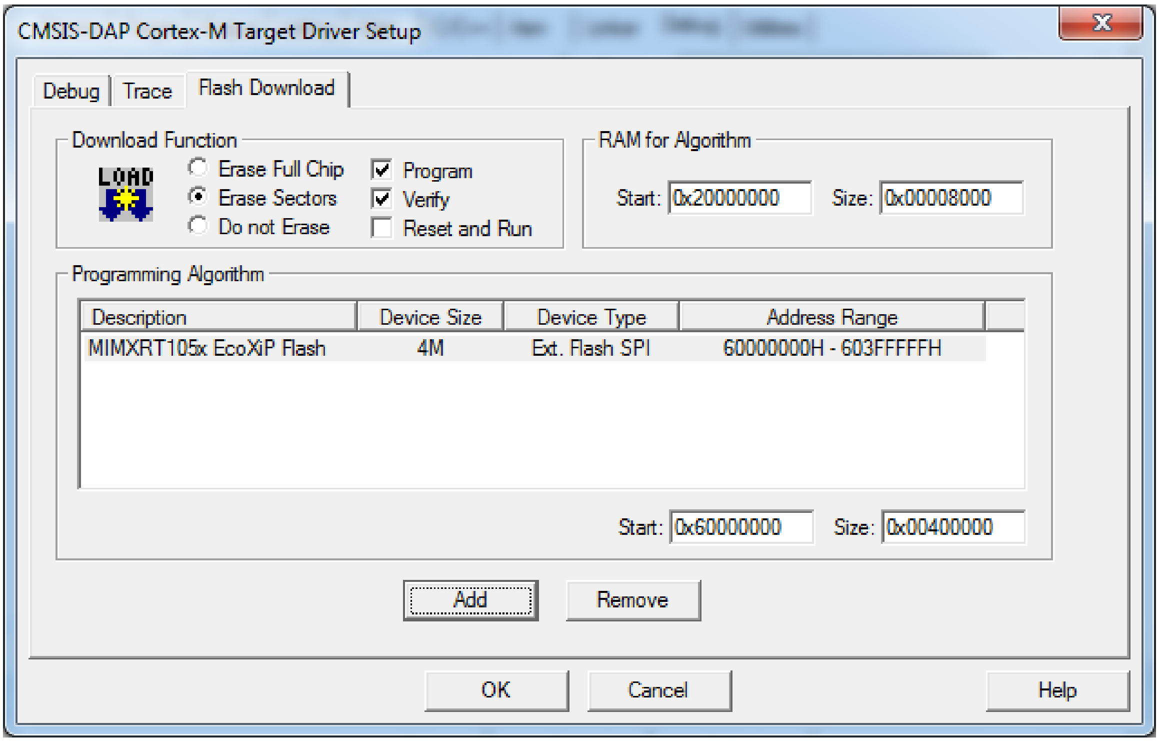
The default target is **flexspi_nor_debug**. The table below shows each of the targets. Release targets have a higher compiler optimization degree making the target less suitable for debugging.

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 have 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

To setup a project for storing/executing the application from the EcoXiP 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 EcoXiP flash driver as shown below.



Download Function

○ Erase Full Chip ☑ Program
● Erase Sectors ☑ Verify
○ Do not Erase □ Reset and Run

RAM for Algorithm

Start: 0x20000000 Size: 0x00008000

Programming Algorithm

Description	Device Size	Device Type	Address Range
MIMXRT105x EcoXiP Flash	4M	Ext. Flash SPI	60000000H - 603FFFFFH

Start: 0x60000000 Size: 0x00400000

Figure 12 – Keil uVision/MDK Flash Algorithm Settings

4.6 Build Output

Building a project results in one important file:

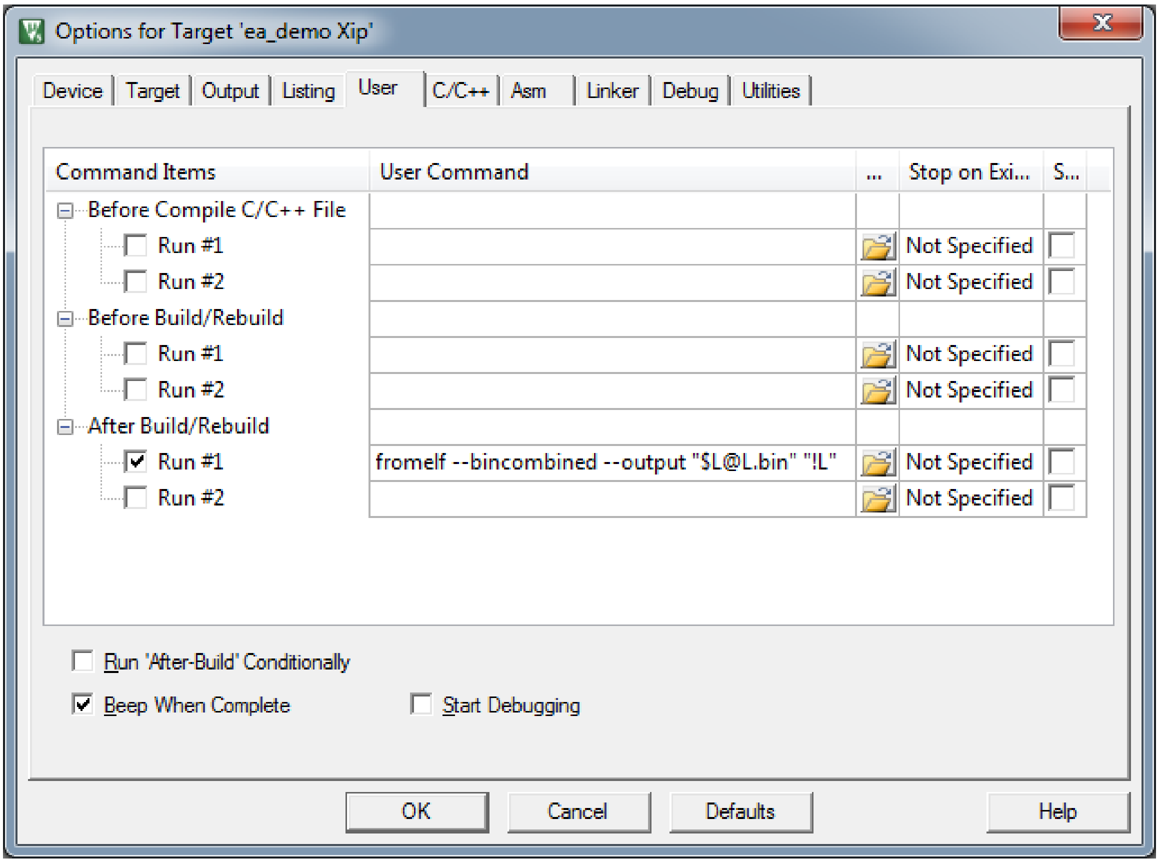
- An `*.out` file (for example `ea_demo.out`) which is perfect when debugging or using the debugger to flash the program

If the goal is to use the MfgTool to download the program (as explained in chapter 8) then another file is more important:

- An `*.bin` file (for example `ea_demo.bin`)

To get the `*.bin` file created automatically, add a User Command, as shown in the picture below. The command is:

```
fromelf --bincombined --output "$L@L.bin" "!L"
```



Command Items	User Command	...	Stop on Exi...	S...
⊟ Before Compile C/C++ File				
□ Run #1			Not Specified	□
□ Run #2			Not Specified	□
⊟ Before Build/Rebuild				
□ Run #1			Not Specified	□
□ Run #2			Not Specified	□
⊟ After Build/Rebuild				
☑ Run #1	fromelf --bincombined --output "$L@L.bin" "!L"		Not Specified	□
□ Run #2			Not Specified	□

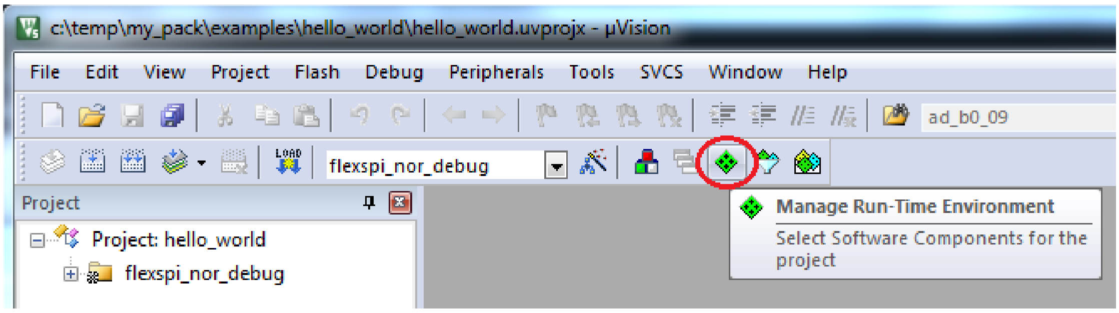
□ Run 'After-Build' Conditionally

☑ Beep When Complete □ Start Debugging

Figure 13 – After Build Command to Create BIN File in Keil uVision/MDK

4.7 Working with RTE

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

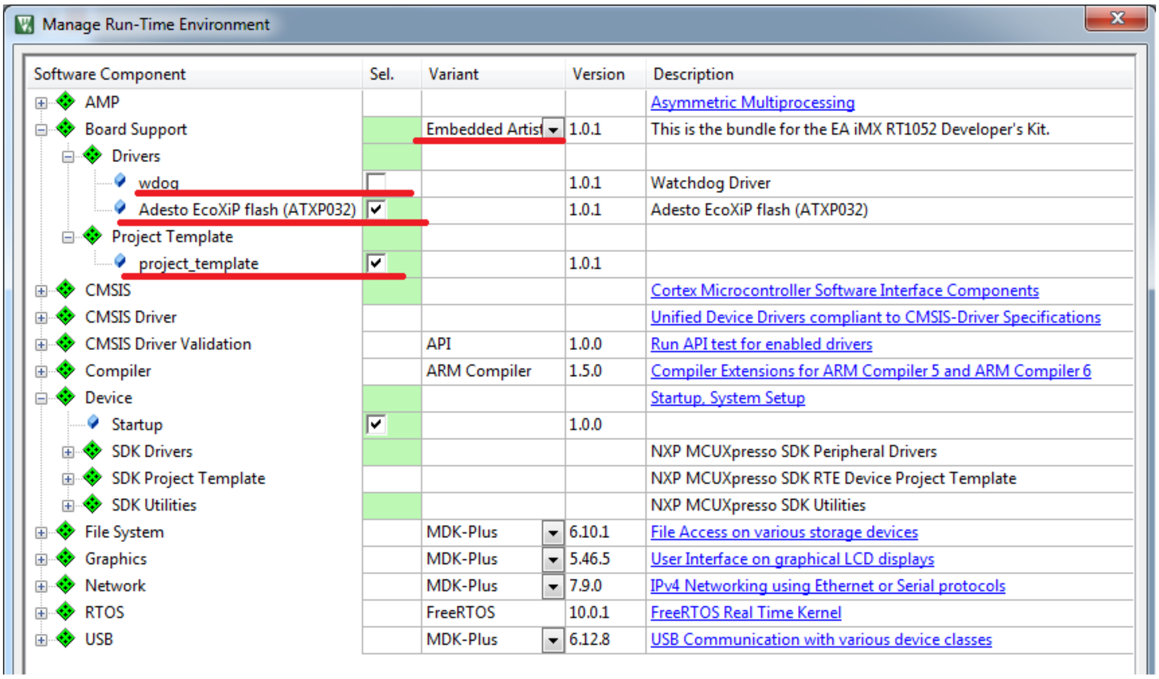


c:\temp\my_pack\examples\hello_world\hello_world.uvprojx - µVision

Manage Run-Time Environment

Select Software Components for the project

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



Manage Run-Time Environment

Software Component	Sel.	Variant	Version	Description
AMP				Asymmetric Multiprocessing
Board Support		Embedded Artist	1.0.1	This is the bundle for the EA iMX RT1052 Developer's Kit.
Drivers				
wdog	□		1.0.1	Watchdog Driver
Adesto EcoXiP flash (ATXP032)	☑		1.0.1	Adesto EcoXiP flash (ATXP032)
Project Template				
project_template	☑		1.0.1	
CMSIS				Cortex Microcontroller Software Interface Components
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specifications
CMSIS Driver Validation		API	1.0.0	Run API test for enabled drivers
Compiler		ARM Compiler	1.5.0	Compiler Extensions for ARM Compiler 5 and ARM Compiler 6
Device				Startup, System Setup
Startup	☑		1.0.0	
SDK Drivers				NXP MCUXpresso SDK Peripheral Drivers
SDK Project Template				NXP MCUXpresso SDK RTE Device Project Template
SDK Utilities				NXP MCUXpresso SDK Utilities
File System		MDK-Plus	6.10.1	File Access on various storage devices
Graphics		MDK-Plus	5.46.5	User Interface on graphical LCD displays
Network		MDK-Plus	7.9.0	IPv4 Networking using Ethernet or Serial protocols
RTOS		FreeRTOS	10.0.1	FreeRTOS Real Time Kernel
USB		MDK-Plus	6.12.8	USB Communication with various device classes

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) 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 EcoXiP flash memory directly.

Every project that use the flash must be manually modified as they come pre-configured with the wrong flash type. The instructions for this can be found in section 5.3 .

Many of the SRAM/SDRAM sample projects created for NXP's EVK will run on the iMX RT Developer's Kit, but not all of them.

5.1 Build an Example Application

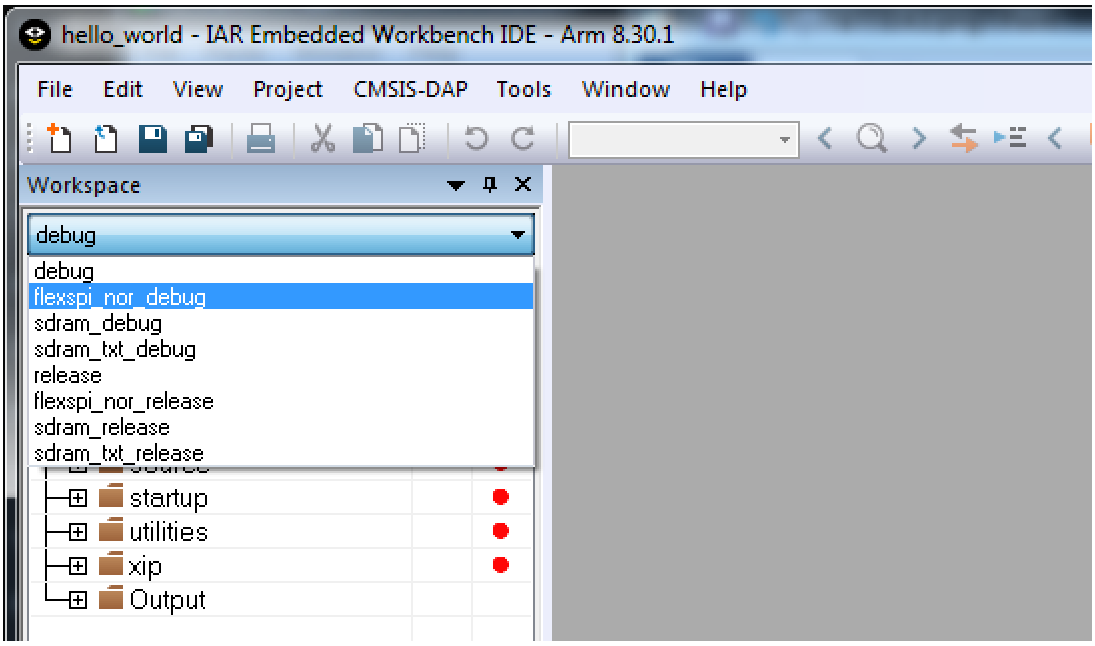
The following steps will guide you through opening the hello_world application. These steps may change slightly for other example applications as some of these applications may have additional layers of folders in their path.

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.

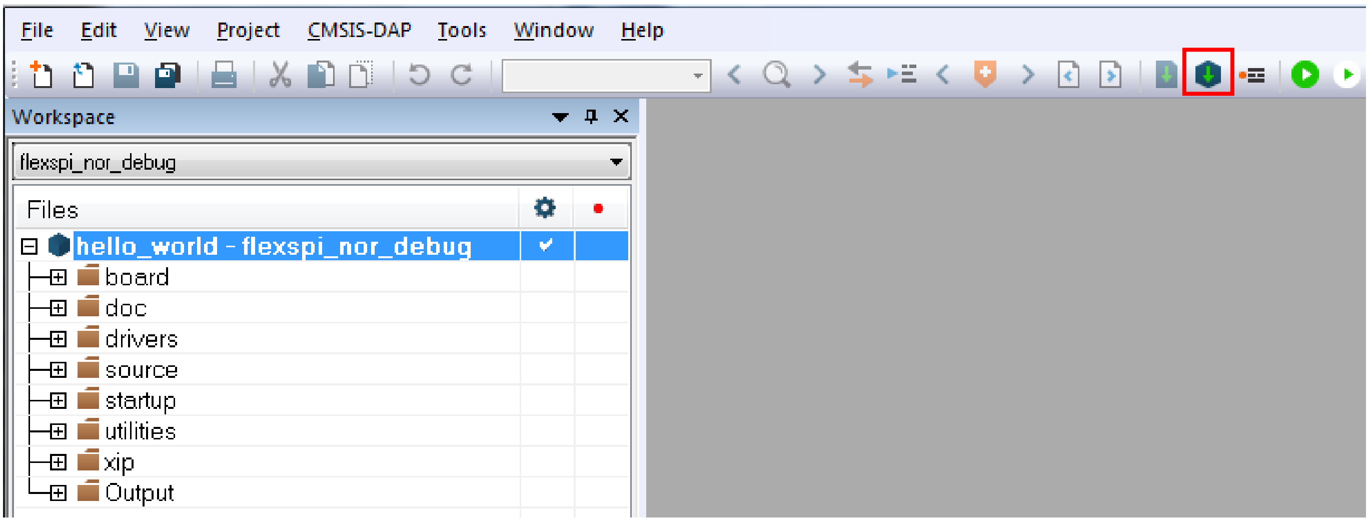


The target name indicates from where the application is executing, see table below. The name also indicates if it is a release or debug target. Release targets have a higher compiler optimization degree making the target less suitable for debugging.

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. Need extra configuration, see section 5.3 .

3. 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.

4. To build the demo application, click the "Make" button, highlighted in red below.

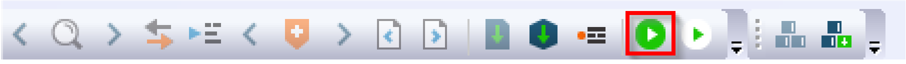

File Edit View Project CMSIS-DAP Tools Window Help
Workspace
flexspi_nor_debug
Files
hello_world - flexspi_nor_debug
board
doc
drivers
source
startup
utilities
xip
Output

5. The build completes without errors.

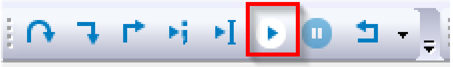
5.2 Run an Example Application

To download and run the application, perform these steps:

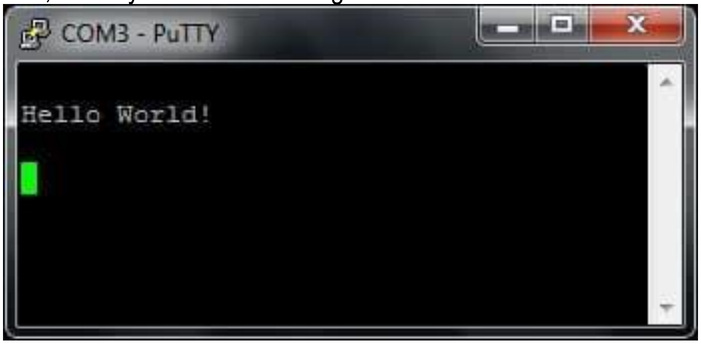
1. Connect the debug probe to development platform to your PC via USB cable. See chapter 7 for details how to connect the debug probe.
2. Open the terminal application on the PC, such as TeraTerm or PuTTY, and connect to the debug serial port number. Configure the terminal with 115200 baud, 8N1.
   You 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.

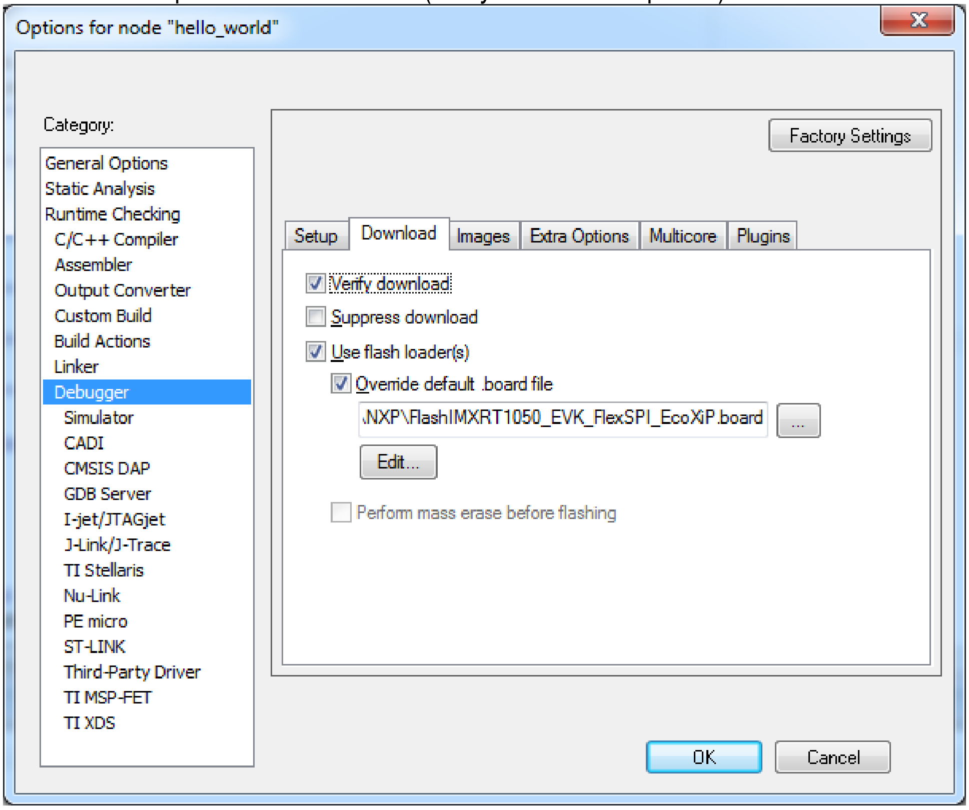


```
Hello World!
```

5.3 Configuring Flash Algorithm

To setup a project for storing/executing the application from the EcoXiP flash:

1. Select one of the flexspi_nor_* targets, right click on the project and select *Options* (or press Alt+F7).
2. Select the *Debugger* in the list and then the *Download* tab and select *Settings* for the debugger.
3. Select the two options as shown below (*Verify download* is optional)

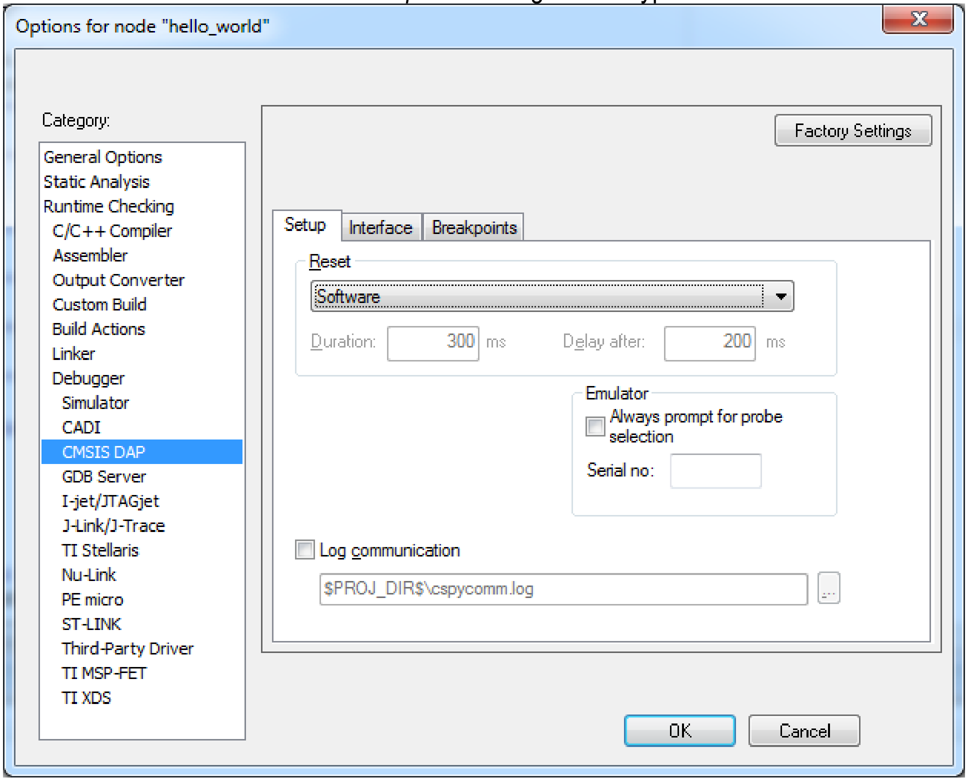


☑ Verify download
□ Suppress download
☑ Use flash loader(s)
☑ Override default .board file
.NXP\FlashIMXRT1050_EVK_FlexSPI_EcoXiP.board
□ Perform mass erase before flashing

The path in the box is
$TOOLKIT_DIR$\config\flashloader\NXP\FlashIMXRT1050_EVK_FlexSPI_EcoXiP.board

It is the same path for the iMX RT1052 and the RT1062.

4. Select *CMSIS DAP* and then the *Setup* tab. Change Reset type to Software as shown here:



Reset: Software

Duration: 300 ms   Delay after: 200 ms

Emulator

□ Always prompt for probe selection

Serial no:

□ Log communication

$PROJ_DIR$\cspycomm.log

5.4 Build Output

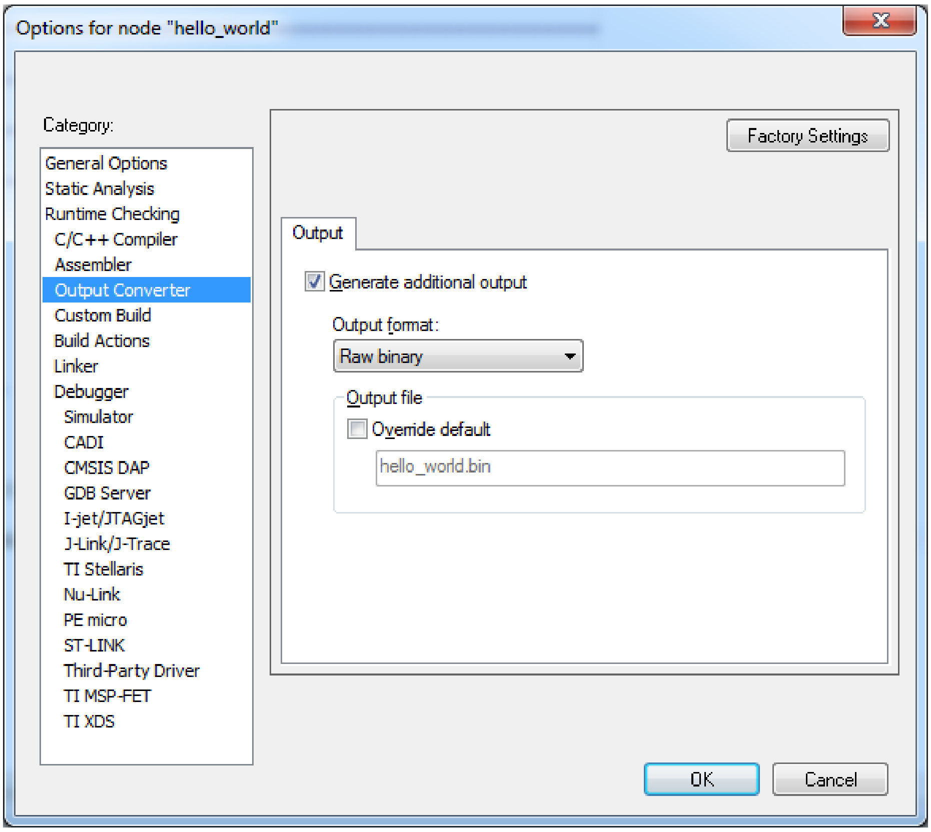
Building a project results in one important file:

- An `*.out` file (for example `hello_world.out`) which is perfect when debugging or using the debugger to flash the program

If the goal is to use the MfgTool to download the program (as explained in chapter 8) then another file is more important:

- An `*.bin` file (for example `hello_world.bin`)

To get the `*.bin` file built automatically, open *Project Options,* select *Output Converter* in the list and then configure like this.



Options for node "hello_world"

Category:

- General Options
- Static Analysis
- Runtime Checking
- C/C++ Compiler
- Assembler
- Output Converter
- Custom Build
- Build Actions
- Linker
- Debugger
- Simulator
- CADI
- CMSIS DAP
- GDB Server
- I-jet/JTAGjet
- J-Link/J-Trace
- TI Stellaris
- Nu-Link
- PE micro
- ST-LINK
- Third-Party Driver
- TI MSP-FET
- TI XDS

Output

☑ Generate additional output

Output format: Raw binary

Output file

□ Override default

hello_world.bin

Figure 14 – Generating bin file in IAR Embedded Workbench

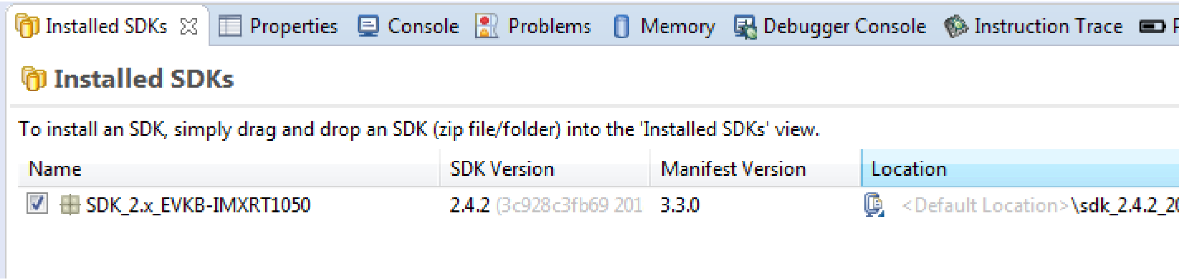
6 Getting Started with NXP MCUXpresso IDE

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.

Make sure to use MCUXpresso version 10.2.1 or later. Older versions did not have support for flash download/programming to the EcoXiP flash memory directly.

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 folder (`<SDK_Install_Directory>/`) that was unpacked and modified in section 2.2 on to the "Installed SDKs" tab in MCUXpresso:



To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Installed SDKs' view.

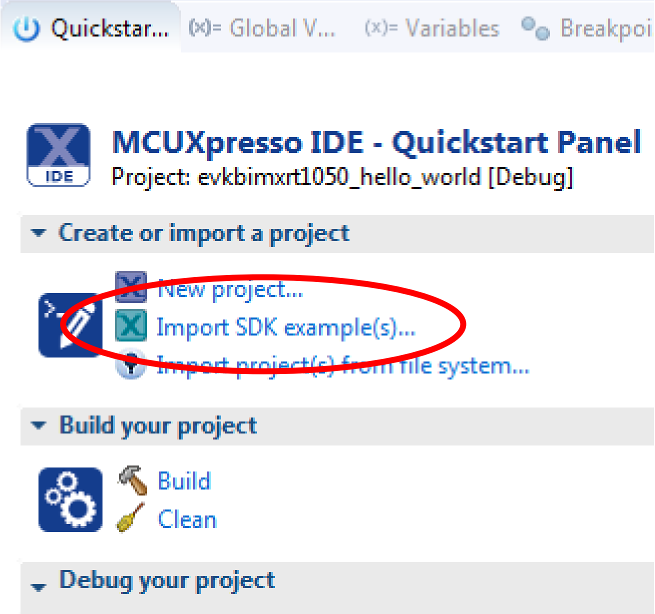
Name	SDK Version	Manifest Version	Location
☑ SDK_2.x_EVKB-IMXRT1050	2.4.2 (3c928c3fb69 201	3.3.0	<Default Location>\sdk_2.4.2_20

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

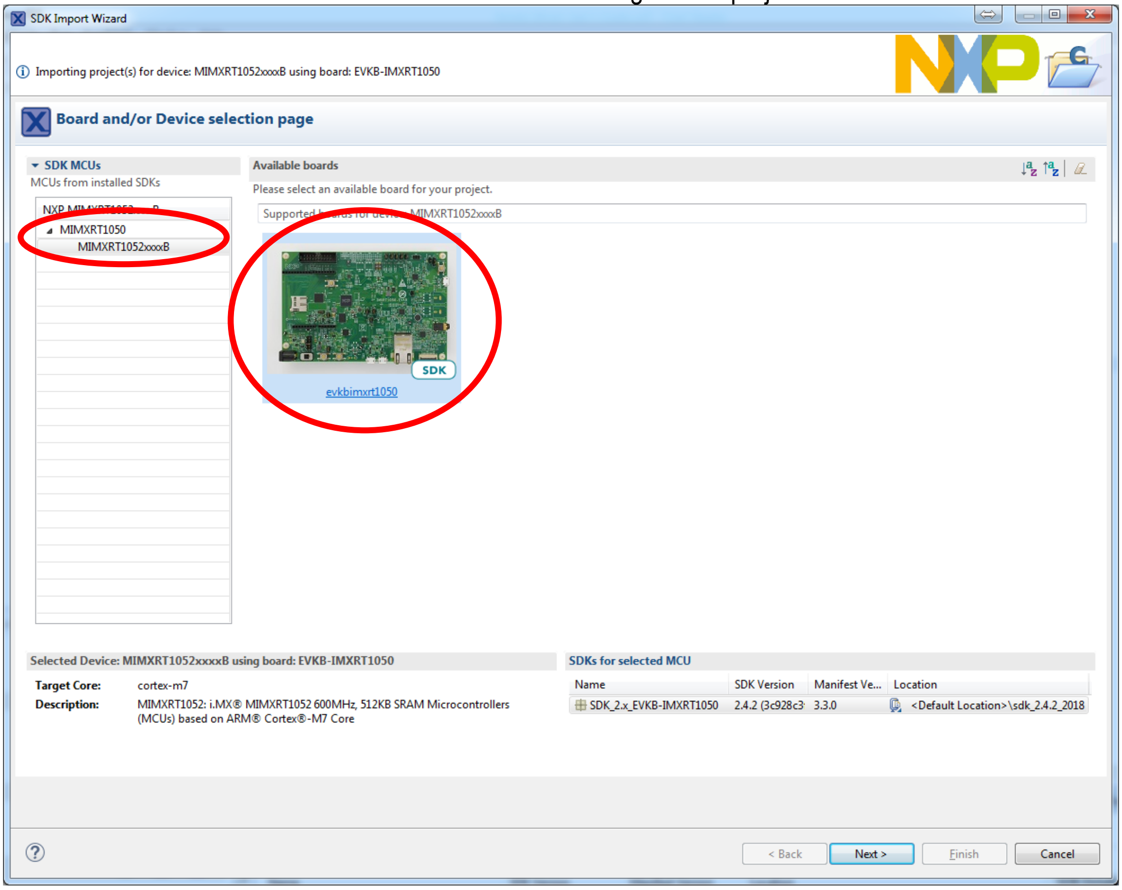
6.2 Build an Example Application

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

1. Install the SDK as described in section 2.1 if you have not done so already
2. Click the "Import SDK example(s)…" link in the Quickstart Panel


Quickstar...
(x)= Global V...
(x)= Variables
Breakpoi
MCUXpresso IDE - Quickstart Panel
Project: evkbimxrt1050_hello_world [Debug]
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Debug your project

3. Select the MIMXRT1050 and evkbimxrt1050. Click Next to go to the project selector.



SDK Import Wizard

Importing project(s) for device: MIMXRT1052xxxxB using board: EVKB-IMXRT1050

Board and/or Device selection page

SDK MCUs

MCUs from installed SDKs

NXP MIMXRT1052xxxxB

- MIMXRT1050
  - MIMXRT1052xxxxB

Available boards

Please select an available board for your project.

Supported boards for device: MIMXRT1052xxxxB

evkbimxrt1050

Selected Device: MIMXRT1052xxxxB using board: EVKB-IMXRT1050

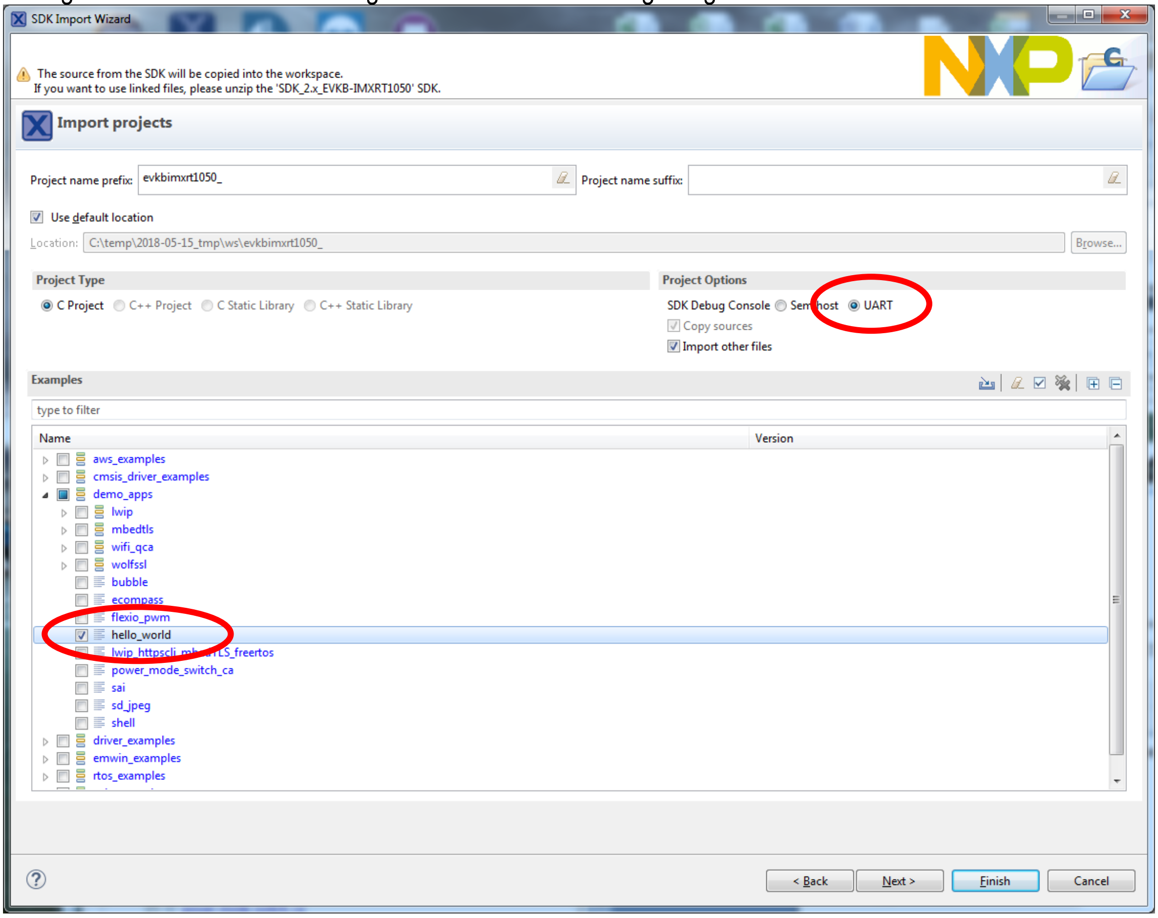
Target Core: cortex-m7

Description: MIMXRT1052: i.MX® MIMXRT1052 600MHz, 512KB SRAM Microcontrollers (MCUs) based on ARM® Cortex®-M7 Core

SDKs for selected MCU

Name	SDK Version	Manifest Ve...	Location
SDK_2.x_EVKB-IMXRT1050	2.4.2 (3c928c3	3.3.0	<Default Location>\sdk_2.4.2_2018

4. Select the hello_world example and make sure to switch from Semihost to UART for the SDK Debug Console then click Next to go to the Advanced Settings Page.



SDK Import Wizard

The source from the SDK will be copied into the workspace.
If you want to use linked files, please unzip the 'SDK_2.x_EVKB-IMXRT1050' SDK.

Import projects

Project name prefix: evkbimxrt1050_ Project name suffix:

☑ Use default location

Location: C:\temp\2018-05-15_tmp\ws\evkbimxrt1050_

Project Type: ● C Project ○ C++ Project ○ C Static Library ○ C++ Static Library

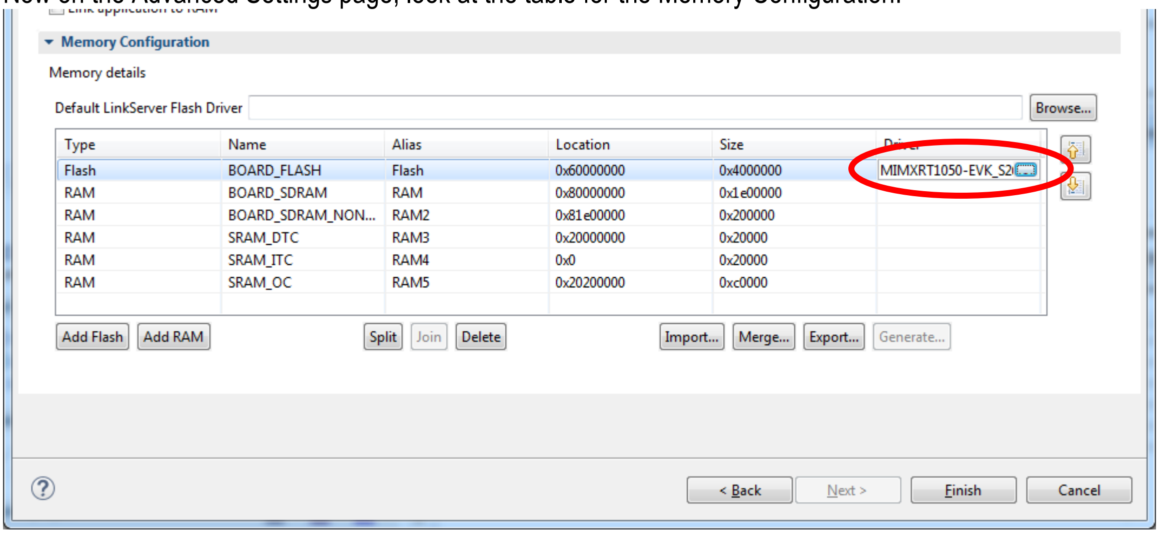
Project Options: SDK Debug Console ○ Semihost ● UART
☑ Copy sources
☑ Import other files

Examples

type to filter

Name	Version
□ aws_examples	
□ cmsis_driver_examples	
■ demo_apps	
□ lwip	
□ mbedtls	
□ wifi_qca	
□ wolfssl	
□ bubble	
□ ecompass	
□ flexio_pwm	
☑ hello_world	
□ lwip_httpscli_mbedTLS_freertos	
□ power_mode_switch_ca	
□ sai	
□ sd_jpeg	
□ shell	
□ driver_examples	
□ emwin_examples	
□ rtos_examples	

5. Now on the Advanced Settings page, look at the table for the Memory Configuration:



Memory Configuration

Memory details

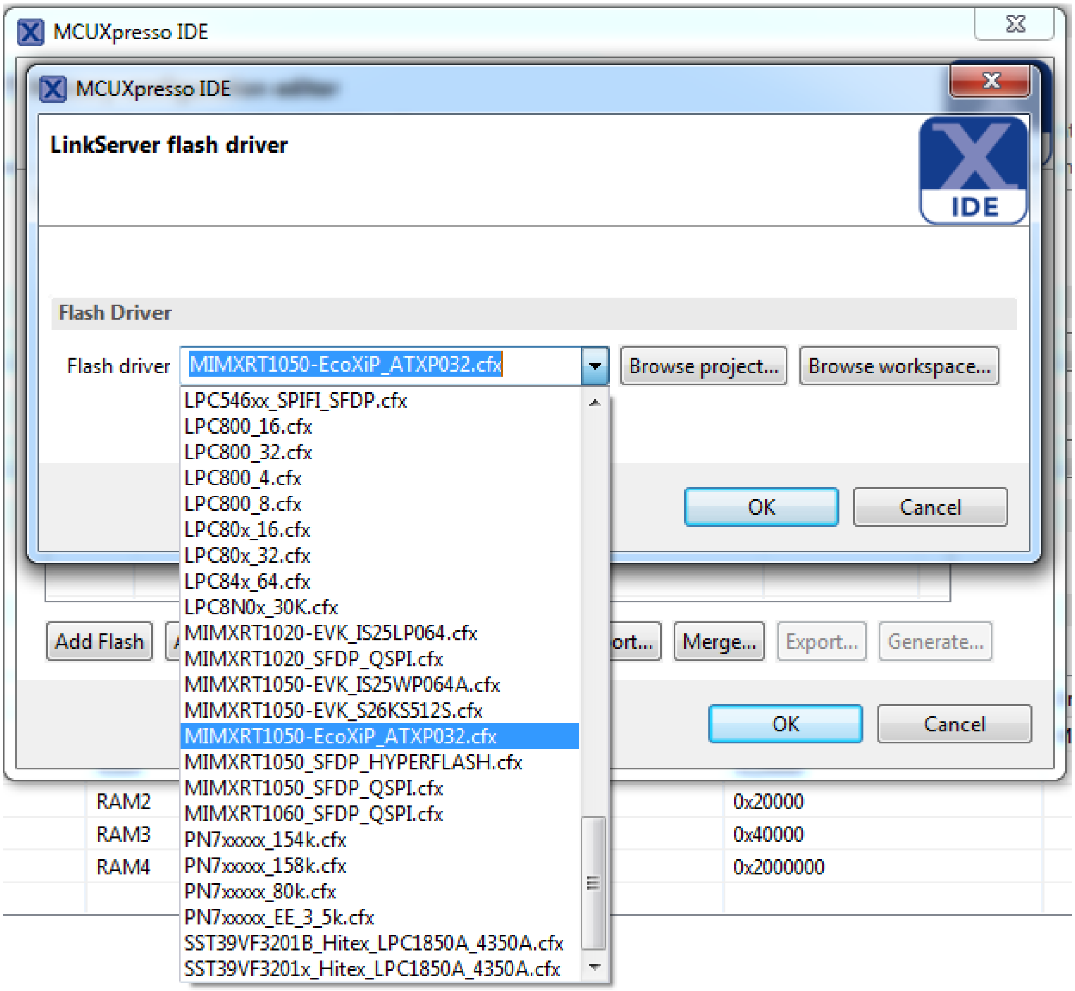
Default LinkServer Flash Driver

Type	Name	Alias	Location	Size	Driver
Flash	BOARD_FLASH	Flash	0x60000000	0x4000000	MIMXRT1050-EVK_S2
RAM	BOARD_SDRAM	RAM	0x80000000	0x1e00000	
RAM	BOARD_SDRAM_NON...	RAM2	0x81e00000	0x200000	
RAM	SRAM_DTC	RAM3	0x20000000	0x20000	
RAM	SRAM_ITC	RAM4	0x0	0x20000	
RAM	SRAM_OC	RAM5	0x20200000	0xc0000	

6. Click and change the Size for BOARD_FLASH from 0x4000000 to 0x400000

7. Click in the table cell for the driver and then on the small button that appears. Change the Driver for BOARD_FLASH from MIMXRT1050-EVK_S26KS512S.cfx to **MIMXRT1050-EcoXiP_ATXP032.cfx** using the dropdown menu.

The same flash driver can be used for both iMX RT1052 and 1062.



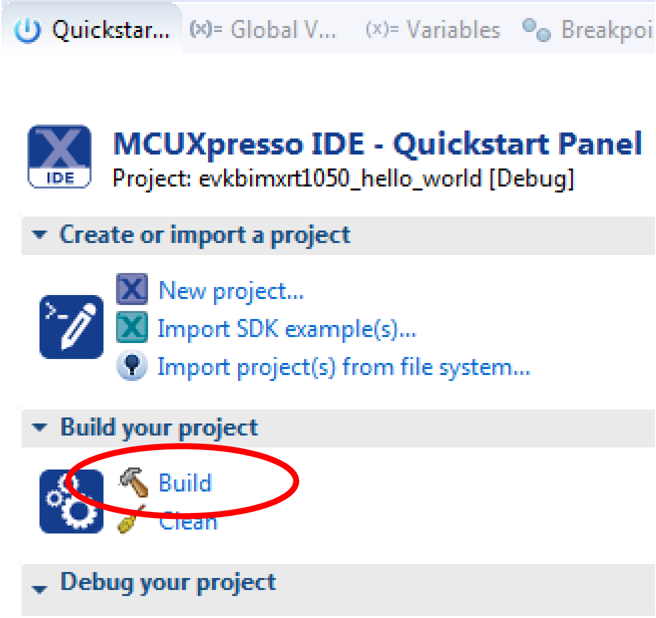
Flash driver: MIMXRT1050-EcoXiP_ATXP032.cfx

LPC546xx_SPIFI_SFDP.cfx
LPC800_16.cfx
LPC800_32.cfx
LPC800_4.cfx
LPC800_8.cfx
LPC80x_16.cfx
LPC80x_32.cfx
LPC84x_64.cfx
LPC8N0x_30K.cfx
MIMXRT1020-EVK_IS25LP064.cfx
MIMXRT1020_SFDP_QSPI.cfx
MIMXRT1050-EVK_IS25WP064A.cfx
MIMXRT1050-EVK_S26KS512S.cfx
MIMXRT1050-EcoXiP_ATXP032.cfx
MIMXRT1050_SFDP_HYPERFLASH.cfx
MIMXRT1050_SFDP_QSPI.cfx
MIMXRT1060_SFDP_QSPI.cfx
PN7xxxxx_154k.cfx
PN7xxxxx_158k.cfx
PN7xxxxx_80k.cfx
PN7xxxxx_EE_3_5k.cfx
SST39VF3201B_Hitex_LPC1850A_4350A.cfx
SST39VF3201x_Hitex_LPC1850A_4350A.cfx

RAM2	0x20000
RAM3	0x40000
RAM4	0x2000000

8. With all the changes made, click ok and then finish to have MCUXpresso complete the project setup

9. Click Build in the Quickstart Panel



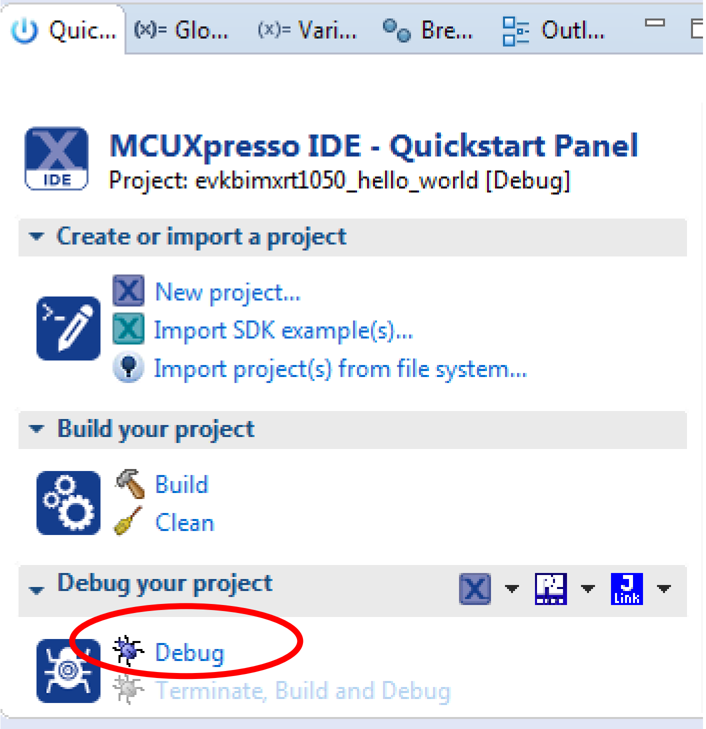
Project: evkbimxrt1050_hello_world [Debug]

10. The program builds without errors

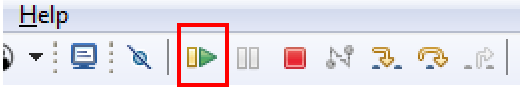
6.3 Run an Example Application

To download and run the application, perform these steps:

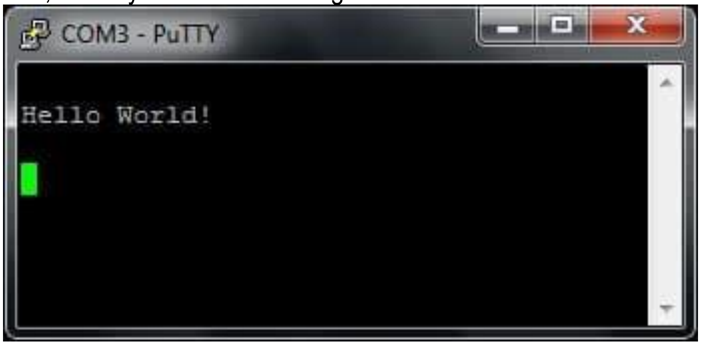
1. Connect the debug probe to development platform to your PC via USB cable. See chapter 7 for details how to connect the debug probe.
2. Open the terminal application on the PC, such as TeraTerm or PuTTY, and connect to the debug serial port number. Configure the terminal with 115200 baud, 8N1.
   You 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.


Quic...
Glo...
Vari...
Bre...
Outl...
MCUXpresso IDE - Quickstart Panel
Project: evkbimxrt1050_hello_world [Debug]
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Debug your project
Debug
Terminate, Build and Debug

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.



1. 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.



```
Hello World!
```

6.4 Build Output

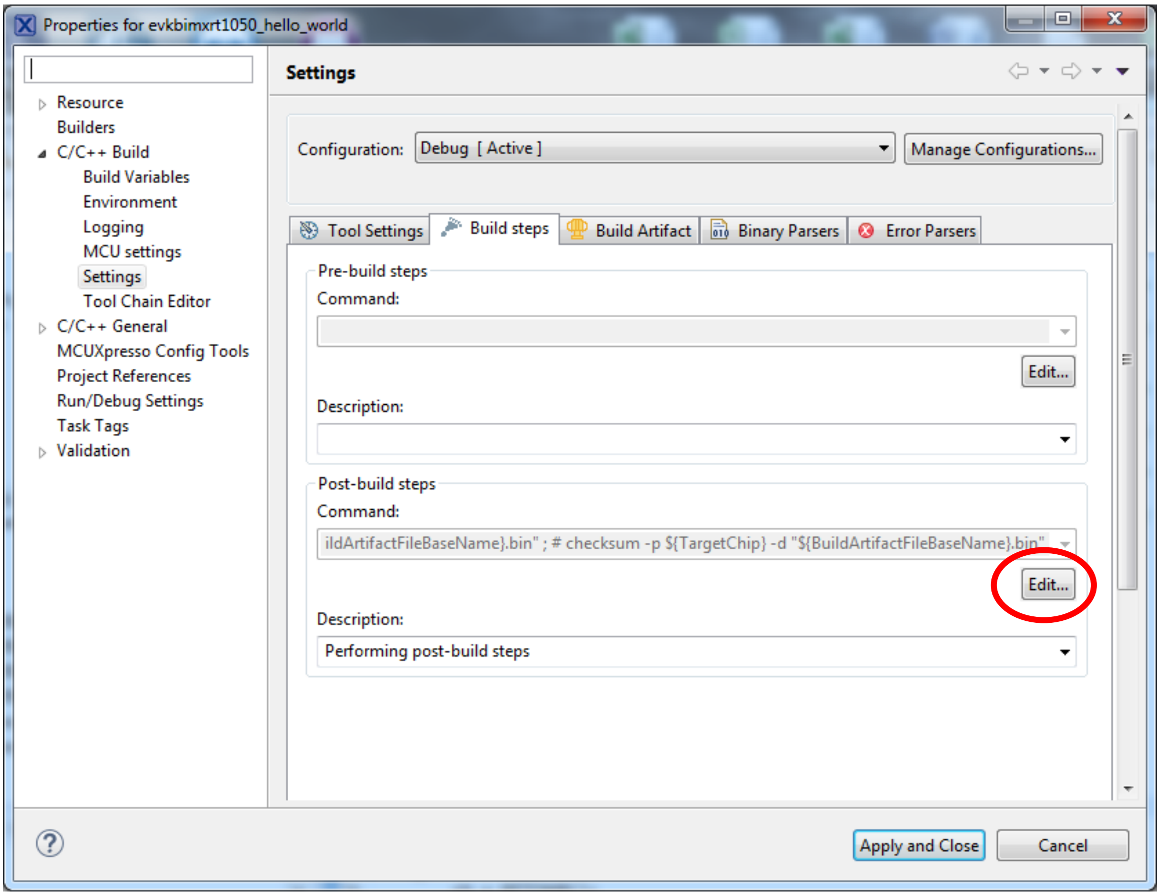
Building a project results in one important file:

- An `*.axf` file (for example `hello_world.axf`) which is perfect when debugging or using the debugger to flash the program

If the goal is to use the MfgTool to download the program (as explained in section 8.3 below) then another file is more important:

- An `*.bin` file (for example `wohello_world.bin`)

To get the `*.bin` file built automatically, select the Project->Properties menu and then select C/C++ Build, select Settings and then the Build steps tab



Properties for evkbimxrt1050_hello_world

Settings

Configuration: Debug [Active]

Pre-build steps

Command:

Description:

Post-build steps

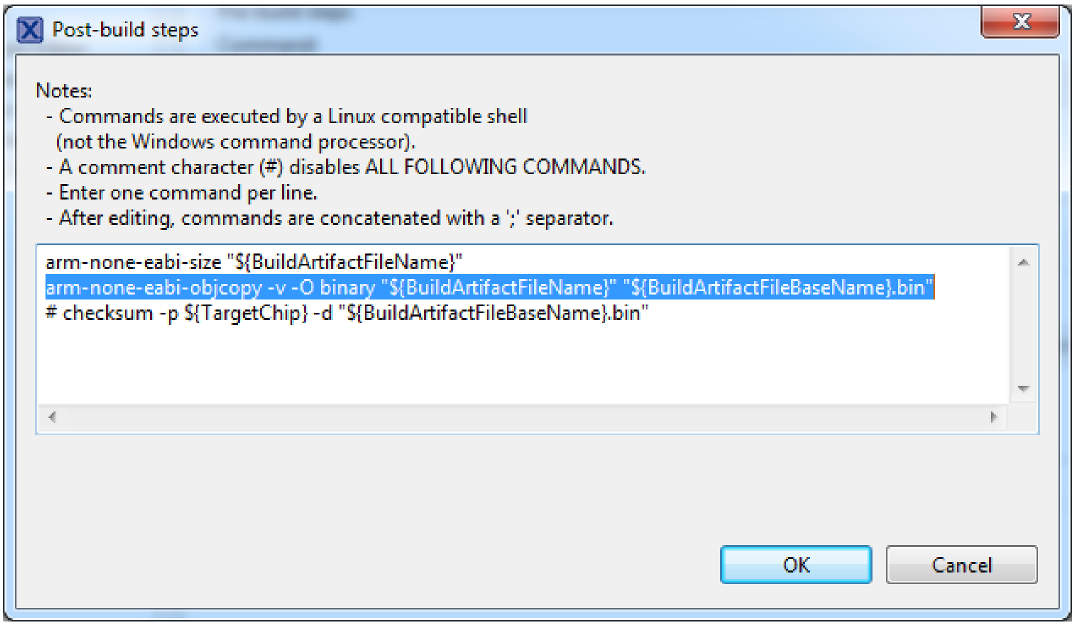
Command: ildArtifactFileBaseName}.bin" ; # checksum -p ${TargetChip} -d "${BuildArtifactFileBaseName}.bin"

Description: Performing post-build steps

Click the Edit button and then add the following in the dialog that appears:

```
arm-none-eabi-objcopy -v -O binary "${BuildArtifactFileName}"
  "${BuildArtifactFileBaseName}.bin"
```

It should look like this:



Notes:
- Commands are executed by a Linux compatible shell
  (not the Windows command processor).
- A comment character (#) disables ALL FOLLOWING COMMANDS.
- Enter one command per line.
- After editing, commands are concatenated with a ';' separator.

```
arm-none-eabi-size "${BuildArtifactFileName}"
arm-none-eabi-objcopy -v -O binary "${BuildArtifactFileName}" "${BuildArtifactFileBaseName}.bin"
# checksum -p ${TargetChip} -d "${BuildArtifactFileBaseName}.bin"
```

Close and save. The *.bin file will be created the next time the project is built.

7 Debug Interface

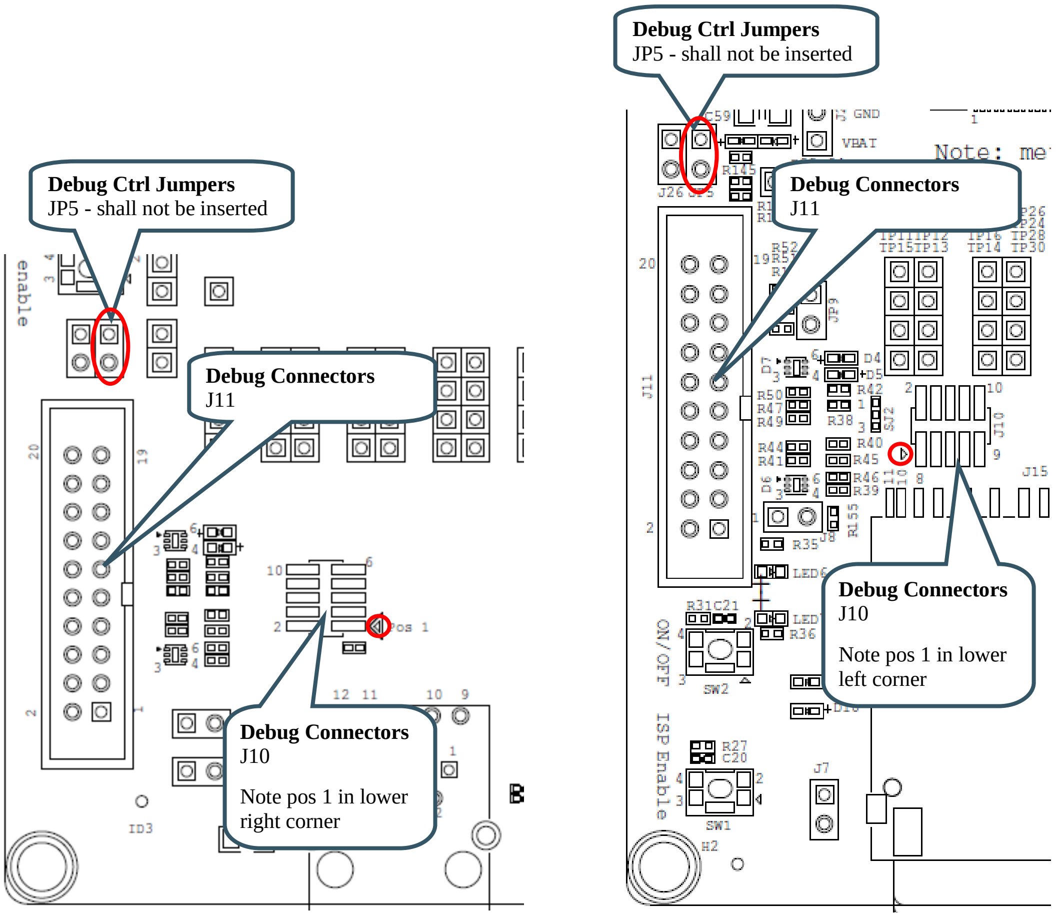
It is strongly recommended to use a debug/JTAG probe during program development. The low-cost LPC-Link2 is an excellent choice. Keil ULINK2 and ULINKplus, as well as Segger JLINK, are also excellent debug probes.

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 in specifically marked on the PCB silkscreen. It is located in the lower right corner, see Figure 15 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 connector are defined and supports both the SWD and JTAG type of debug interfaces.

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



Debug Ctrl Jumpers
JP5 - shall not be inserted

Debug Connectors
J11

Debug Connectors
J10

Note pos 1 in lower right corner

Debug Ctrl Jumpers
JP5 - shall not be inserted

Debug Connectors
J11

Debug Connectors
J10

Note pos 1 in lower left corner

Figure 15 – 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

The software for the J-TRACE/J-LINK is installed in a central location (i.e. outside of the IDEs). The latest version (v6.42b) as of February 2019 have support for the iMX RT1052/1062 but needs a configuration change to support the EcoXiP flash memory:

1. Download and install the latest version of Segger's software: https://www.segger.com/downloads/jlink/JLink_Windows.exe (v6.42, or later). The rest of the steps will use `<jdir>` as abbreviation for the folder that the driver was installed in, for example c:\Program Files (x86)\SEGGER\JLink_V642b\
2. Open `<jdir>\JLinkDevices.xml`
3. Search for MCIMXRT1052 or MCIMXRT1062 to find <Device> entries (at the time this document was written there were MCIMXRT1052, MCIMXRT1052xxxxA and MCIMXRT1052xxxxB, MCIMXRT1062)
4. In each of the <Device> entries, change

```
Loader="Devices/NXP/iMXRT105x/NXP_iMXRT105x_HyperFlash.elf"
or
Loader="Devices/NXP/iMXRT106x/NXP_iMXRT106x_HyperFlash.elf"
```

to

```
Loader="Devices/NXP/iMXRT105x/NXP_iMXRT105x_QSPI.elf"
or
Loader="Devices/NXP/iMXRT106x/NXP_iMXRT106x_QSPI.elf"
```

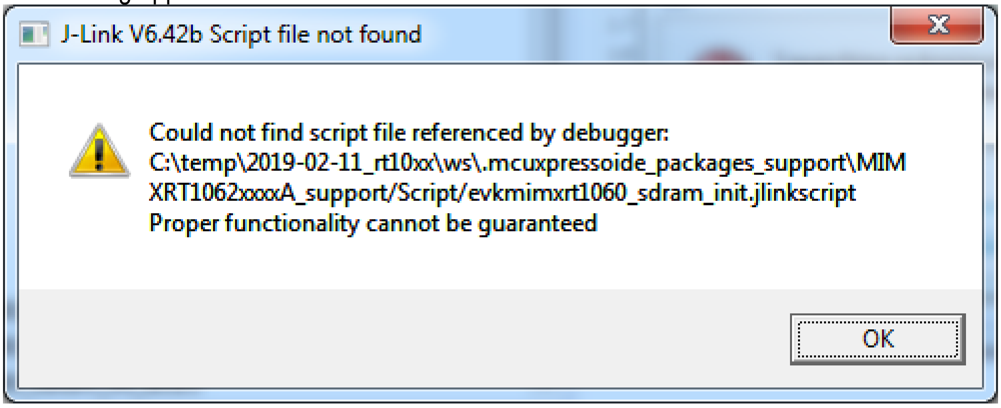
Change the `MaxSize="0x04000000"` to `MaxSize="0x00400000"` and then save the file.

7.1.2 MCUXpresso 10.3.0

Chapter 6 describes how to import a project and set it up correctly for the Adesto EcoXip flash. It is important that the flash algorithm is changed to MIMXRT1050-EcoXiP_ATXP032.cfx and that the flash size is changed to 0x400000. Note that same flash driver shall be used for both iMX RT1052 and RT1062.

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

If this dialog appears:



J-Link V6.42b Script file not found

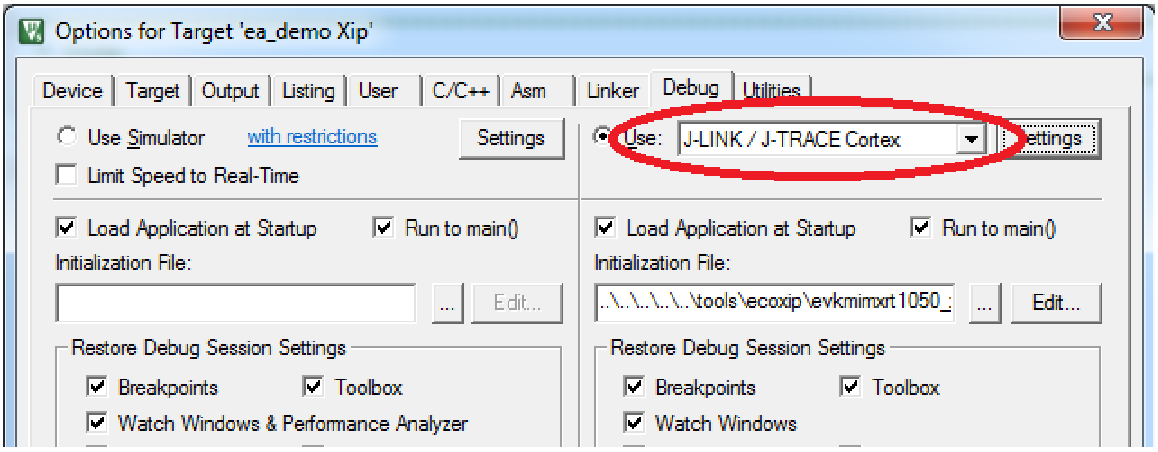
Could not find script file referenced by debugger:
C:\temp\2019-02-11_rt10xx\ws\.mcuxpressoide_packages_support\MIMXRT1062xxxxA_support/Script/evkmimxrt1060_sdram_init.jlinkscript
Proper functionality cannot be guaranteed

Then copy the missing file from the folder that was created in see section 2.2 (`c:/temp/ea_files/`)

7.1.3 Keil uVision v5.25.2

Chapter 6 describes how to import a project and set it up correctly for the Adesto EcoXip flash. It is important that the flash algorithm is changed to MIMXRT105x EcoXiP Flash and that the flash size is changed to 0x400000. Note that 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:



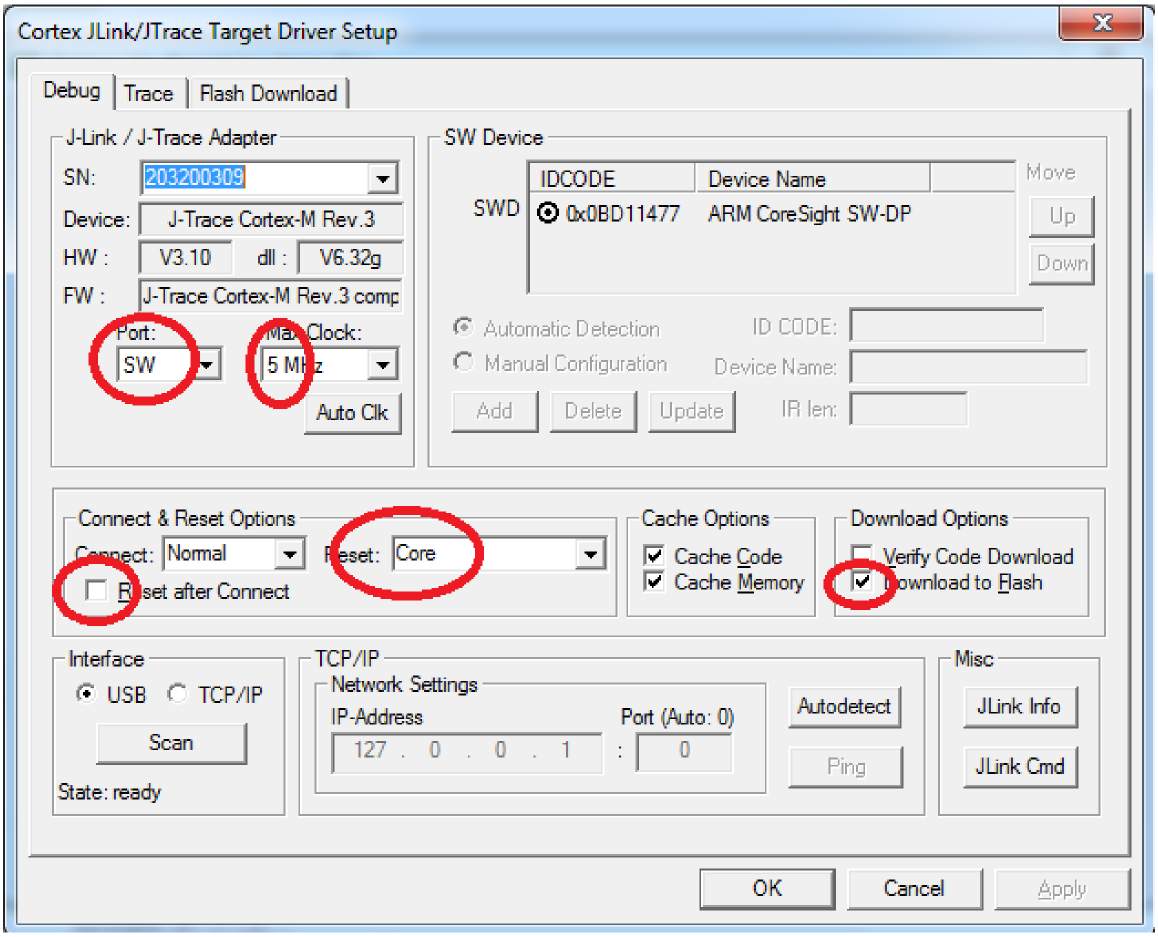
Options for Target 'ea_demo Xip'
Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities
with restrictions
Settings

○ Use Simulator
□ Limit Speed to Real-Time
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows & Performance Analyzer

● Use: J-LINK / J-TRACE Cortex
☑ Load Application at Startup ☑ Run to main()
Initialization File: ..\..\..\..\tools\ecoxip\evkmimxrt1050_
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows

Figure 16 – Setting Debug Interface in Keil uVision

Open the settings dialog and change to the following settings:



J-Link / J-Trace Adapter

SN: 203200309

Device: J-Trace Cortex-M Rev.3

HW : V3.10 dll : V6.32g

FW : J-Trace Cortex-M Rev.3 comp

Port: SW

Max Clock: 5 MHz

SW Device

	IDCODE	Device Name
SWD	● 0x0BD11477	ARM CoreSight SW-DP

● Automatic Detection ID CODE:

○ Manual Configuration Device Name:

IR len:

Connect & Reset Options

Connect: Normal Reset: Core

□ Reset after Connect

Cache Options

☑ Cache Code

☑ Cache Memory

Download Options

□ Verify Code Download

☑ Download to Flash

Interface

● USB ○ TCP/IP

State: ready

TCP/IP

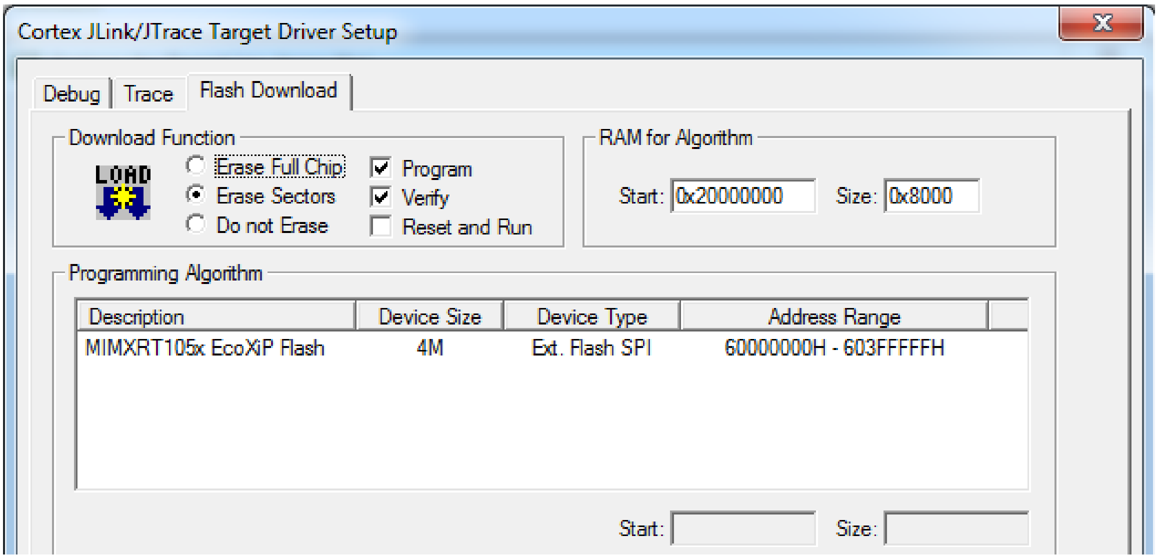
Network Settings

IP-Address 127 . 0 . 0 . 1

Port (Auto: 0) : 0

Misc

Figure 17 – Configuring J-TRACE/J-LINK Interface in Keil uVision



Download Function

○ Erase Full Chip ☑ Program

● Erase Sectors ☑ Verify

○ Do not Erase □ Reset and Run

RAM for Algorithm

Start: 0x20000000 Size: 0x8000

Programming Algorithm

Description	Device Size	Device Type	Address Range
MIMXRT105x EcoXiP Flash	4M	Ext. Flash SPI	60000000H - 603FFFFFH

Start: Size:

Figure 18 – Configuring Flash Programming for J-TRACE/J-LINK Interface in Keil uVision

7.1.4 J-LINK/J-TRACE 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

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 exist that represent the application program.

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

- **SWD/JTAG Debug Interface**
  Using this method, the application can be downloaded to internal SRAM, to external SDRAM or external EcoXiP 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 LPC-LINK2, Keil has ULINK2/ULINKpro, Segger has J-LINK, etc.
- **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, Mfgtool2 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.
  The ISP over USB download is accomplished by using the Mfgtool application running on a PC. Note the following limitations:
    - The Mfgtool application is only available for Windows.
    - The Mfgtool application must be downloaded from the NXP website for license reasons.
    - The setup requires a couple of small modifications to work with the EcoXiP flash from Adesto. These modifications are explained below.

8.1 Install the Required Software

8.1.1 i.MX RT1052

First of all, download the "Flashloader i.MX-RT1050" application from NXP's website. It can be found under the "Tools & Software" tab on the product page for the i.MX RT1050 MCU.

As of February 2019, the file name is **`Flashloader_i.MXRT1050_GA.zip`**. Extract the content of the zip-file to the desired installation location in your file system. Note that the installation root shall have a short path. For example: **`c:\NXP_SDK\`**

Modification #1:

1. Open the **<install directory>\Tools\mfgtools-rel\cfg.ini** file
2. Replace this line:

```
name = MXRT105X-DevBoot
```

with

```
name = MXRT105x-DevBootSerialFlashXiP
```

Modification #2:

1. Open the **<install directory>\Tools\mfgtools-rel\Profiles\MXRT105X\OS Firmware\ucl2.xml** file
2. Scroll down to a line that starts with *<LIST name="MXRT105x-DevBootSerialFlashXiP"*
3. A couple of lines below, replace this command:

   ```
   fill-memory 0x2000 4 0xc0233007
   ```

   with

   ```
   fill-memory 0x2000 4 0xc0803007
   ```

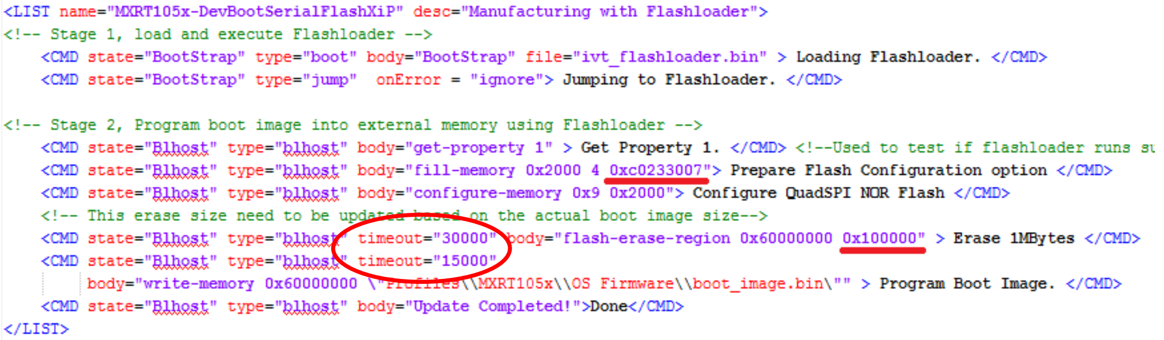
4. (Optional) If the binary file that you are planning to flash is larger than 1Mb in size then replace this command:

   ```
   flash-erase-region 0x60000000 0x100000
   ```

   with for example

   ```
   flash-erase-region 0x60000000 0x400000
   ```

   This will erase the entire flash (4MByte). Any size can be chosen as long as it is a multiple of 0x1000. The larger the erase size the longer it will take to flash so keep it as small as possible. The increased erase size may require an increase of the timeout from the 30000/15000ms default values to 60000ms.
   The picture below illustrates where the two constants can be found in the *.xml file.



```
<LIST name="MXRT105x-DevBootSerialFlashXiP" desc="Manufacturing with Flashloader">
<!-- Stage 1, load and execute Flashloader -->
    <CMD state="BootStrap" type="boot" body="BootStrap" file="ivt_flashloader.bin" > Loading Flashloader. </CMD>
    <CMD state="BootStrap" type="jump"  onError = "ignore"> Jumping to Flashloader. </CMD>

<!-- Stage 2, Program boot image into external memory using Flashloader -->
    <CMD state="Blhost" type="blhost" body="get-property 1" > Get Property 1. </CMD> <!--Used to test if flashloader runs su
    <CMD state="Blhost" type="blhost" body="fill-memory 0x2000 4 0xc0233007"> Prepare Flash Configuration option </CMD>
    <CMD state="Blhost" type="blhost" body="configure-memory 0x9 0x2000"> Configure QuadSPI NOR Flash </CMD>
    <!-- This erase size need to be updated based on the actual boot image size-->
    <CMD state="Blhost" type="blhost" timeout="30000" body="flash-erase-region 0x60000000 0x100000" > Erase 1MBytes </CMD>
    <CMD state="Blhost" type="blhost" timeout="15000"
        body="write-memory 0x60000000 \"Profiles\\MXRT105x\\OS Firmware\\boot_image.bin\"" > Program Boot Image. </CMD>
    <CMD state="Blhost" type="blhost" body="Update Completed!">Done</CMD>
</LIST>
```

8.1.2 i.MX RT1062

First of all, download the "Flashloader i.MX-RT1060" application from NXP's website. It can be found under the "Tools & Software" tab on the product page for the RT1060 MCU.

As of February 2019, the file name is **`FLASHLOADER-RT106x-1-GA.zip`**. Extract the content of the zip-file to the desired installation location in your file system. Note that the installation root shall have a short path. For example: **`c:\NXP_SDK\`**

Modification #1:

3. Open the **<install directory>\Tools\mfgtools-rel\cfg.ini** file
4. Replace this line:

   ```
   name = MXRT106X-DevBoot
   ```

   with

   ```
   name = MXRT106x-DevBootSerialFlashXiP
   ```

Modification #2:

5. Open the **<install directory>\Tools\mfgtools-rel\Profiles\MXRT106X\OS Firmware\ucl2.xml** file
6. Scroll down to a line that starts with *<LIST name="MXRT106x-DevBootSerialFlashXiP"*
7. A couple of lines below, replace this command:

   ```
   fill-memory 0x2000 4 0xc0000007
   ```

   with

   ```
   fill-memory 0x2000 4 0xc0803007
   ```

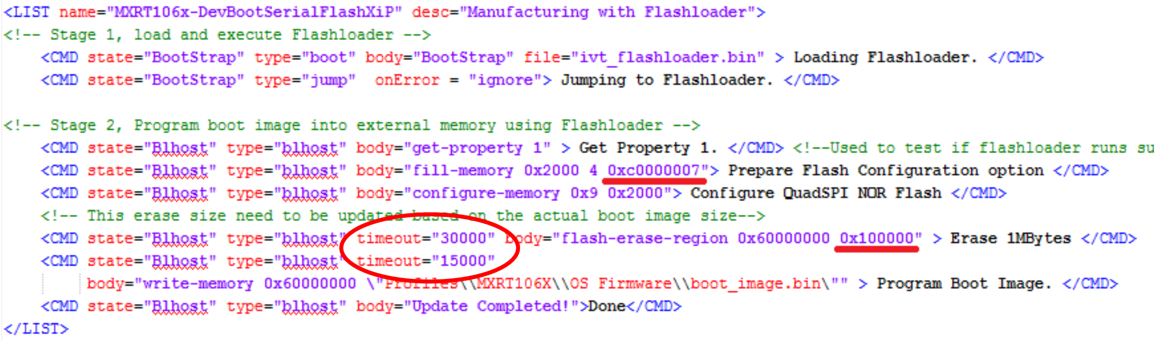
8. (Optional) If the binary file that you are planning to flash is larger than 1Mb in size then replace this command:

   ```
   flash-erase-region 0x60000000 0x100000
   ```

   with for example

   ```
   flash-erase-region 0x60000000 0x400000
   ```

   This will erase the entire flash (4MByte). Any size can be chosen as long as it is a multiple of 0x1000. The larger the erase size the longer it will take to flash so keep it as small as possible. The increased erase size may require an increase of the timeout from the 30000/15000ms default values to 60000ms.
   The picture below illustrates where the two constants can be found in the *.xml file.



```
<LIST name="MXRT106x-DevBootSerialFlashXiP" desc="Manufacturing with Flashloader">
<!-- Stage 1, load and execute Flashloader -->
    <CMD state="BootStrap" type="boot" body="BootStrap" file="ivt_flashloader.bin" > Loading Flashloader. </CMD>
    <CMD state="BootStrap" type="jump"  onError = "ignore"> Jumping to Flashloader. </CMD>

<!-- Stage 2, Program boot image into external memory using Flashloader -->
    <CMD state="Blhost" type="blhost" body="get-property 1" > Get Property 1. </CMD> <!--Used to test if flashloader runs su
    <CMD state="Blhost" type="blhost" body="fill-memory 0x2000 4 0xc0000007"> Prepare Flash Configuration option </CMD>
    <CMD state="Blhost" type="blhost" body="configure-memory 0x9 0x2000"> Configure QuadSPI NOR Flash </CMD>
    <!-- This erase size need to be updated based on the actual boot image size-->
    <CMD state="Blhost" type="blhost" timeout="30000" body="flash-erase-region 0x60000000 0x100000" > Erase 1MBytes </CMD>
    <CMD state="Blhost" type="blhost" timeout="15000"
        body="write-memory 0x60000000 \"Profiles\\MXRT106X\\OS Firmware\\boot_image.bin\"" > Program Boot Image. </CMD>
    <CMD state="Blhost" type="blhost" body="Update Completed!">Done</CMD>
</LIST>
```

8.2 Prepare the Program to Flash

As of SDK version 2.4.0 the procedure prepare the program has become much simpler. The only requirement is that program is in the binary format (*.bin). Setting up your IDE to create a binary file is explained in the Getting Started sections above (for the different development environments).

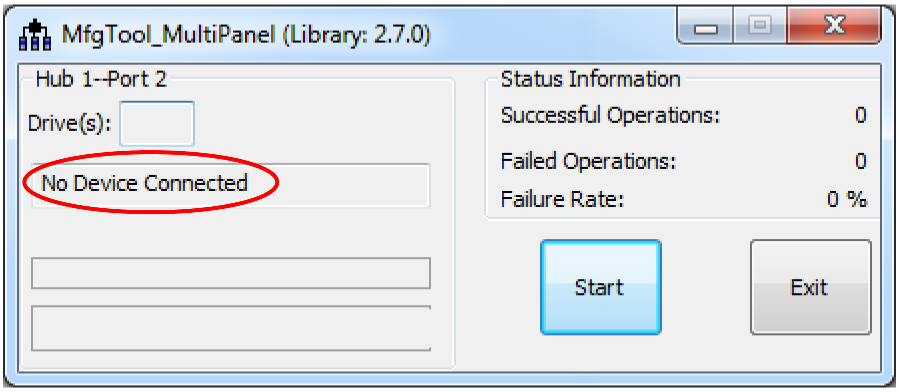
Follow the instructions below to prepare the application:

1. Copy your binary file to **Tools\mfgtools-rel\Profiles\MXRT105X\OS Firmware** or **Tools\mfgtools-rel\Profiles\MXRT106X\OS Firmware** and rename it to **boot_image.bin**.
2. If your binary file is larger than 1 Mb then see the modification steps of **ucl2.xml** above and update the erase region size.

8.3 Download the Program

Follow the instructions below to download an application via USB.

1. It is assumed that a downloadable image (*.bin) has already been created as described above.
2. Keep the ISP Enable push-button, SW1, pressed while powering-up the board. Alternatively insert a jumper in J7. Power the board either via J22 (the micro-B USB connector for the UART-to-USB bridge) or J1 (the 5V DC power jack).
   The i.MX RT MCU is now in ISP OTG mode.
3. Connect a micro-B USB cable between J12 and the PC.
4. Start the Mfgtool2.exe application on the PC. It is found here:
   `c:\NXP_SDK\<Flashloader_Install_Directory>\Tools\mfgtools-rel\MfgTool2.exe.`
   If no i.MX RT MCU in "ISP USB OTG" mode is detected, it will be flagged as "No Device Connected".



MfgTool_MultiPanel (Library: 2.7.0)

Hub 1--Port 2

Drive(s):

No Device Connected

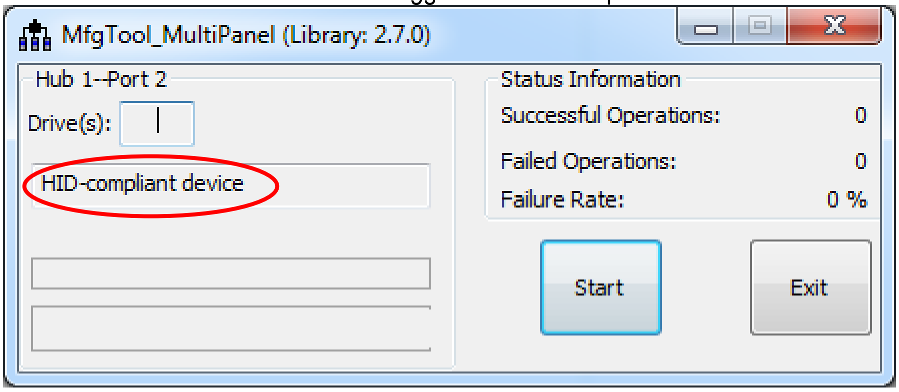
Status Information

Successful Operations: 0

Failed Operations: 0

Failure Rate: 0 %

When a device is detected it will be flagged as "HID-compliant device".



MfgTool_MultiPanel (Library: 2.7.0)

Hub 1--Port 2

Drive(s):

HID-compliant device

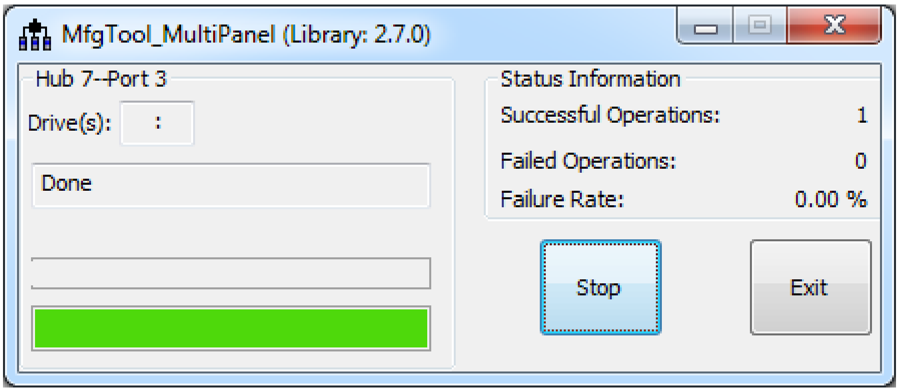
Status Information

Successful Operations: 0

Failed Operations: 0

Failure Rate: 0 %

5. Click on the Start button. A progress bar will illustrate the download process. When ready click on the Exit button.



MfgTool_MultiPanel (Library: 2.7.0)

Hub 7--Port 3

Drive(s): :

Done

Status Information

Successful Operations: 1

Failed Operations: 0

Failure Rate: 0.00 %

6. Disconnect the USB cable from J12. If there is a (short circuit) jumper in J7, remove that.

7. Now power cycle the board, for example by pressing the reset push-button. If a correct image has been flashed, it will now start executing.

9 Terminal Application Setup

This chapter contains information about the terminal connection that exist on the *iMX OEM Carrier Board*, and how to setup a terminal application on the PC. The terminal connection connect 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 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 (LED11) and receive to the board (LED10), that signal communication activity.

See picture below for locating relevant components.

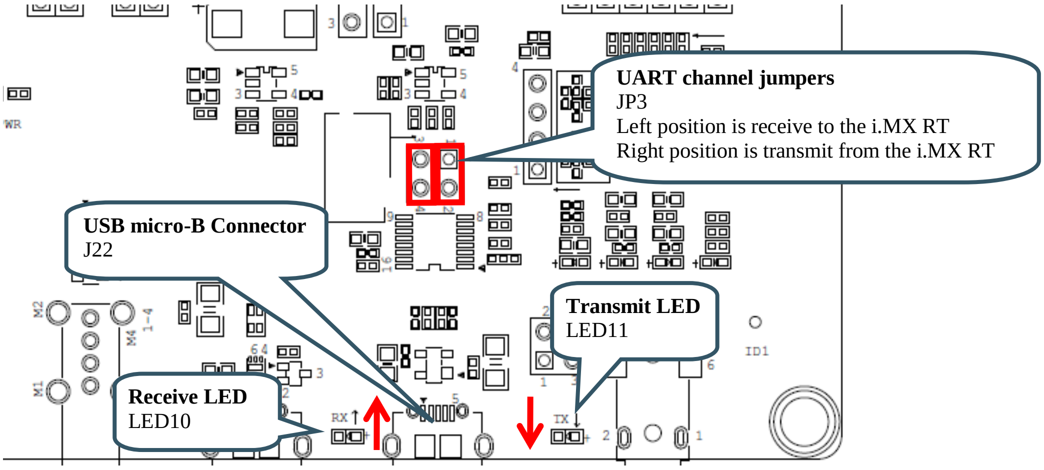


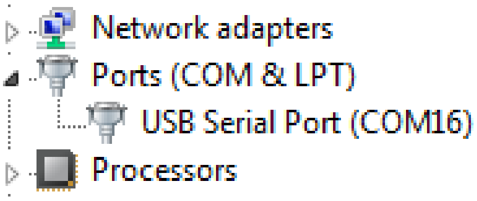
Figure 19 – UART-to-USB Bridge

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 listed under "Ports" in the Device Manager as shown in Figure 20. Please note that the actual port number will most likely be different on your computer.



- Network adapters
- Ports (COM & LPT)
  - USB Serial Port (COM16)
- Processors

Figure 20 – 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**.

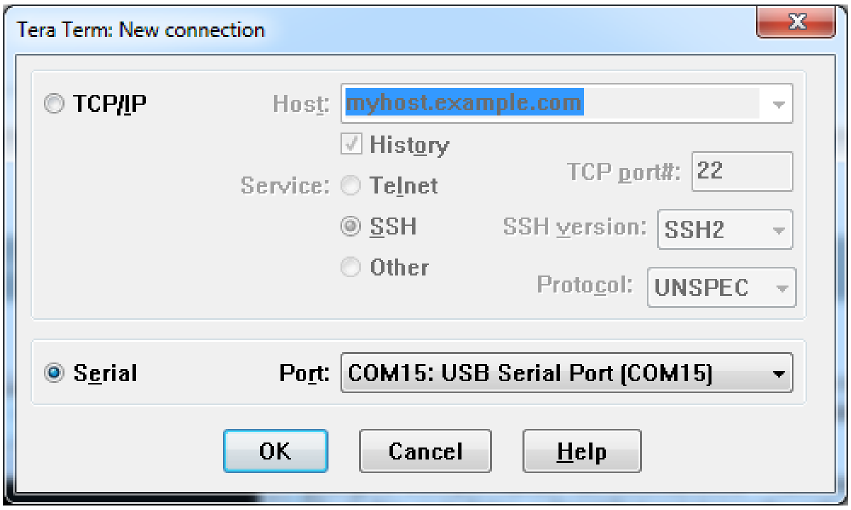
Some development environments/IDEs have a built-in terminal application that can be used. Sometimes it is better to have a terminal application with more features. For increased flexibility, we recommend using any of the two alternative terminal applications presented in the following sub-sections.

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.



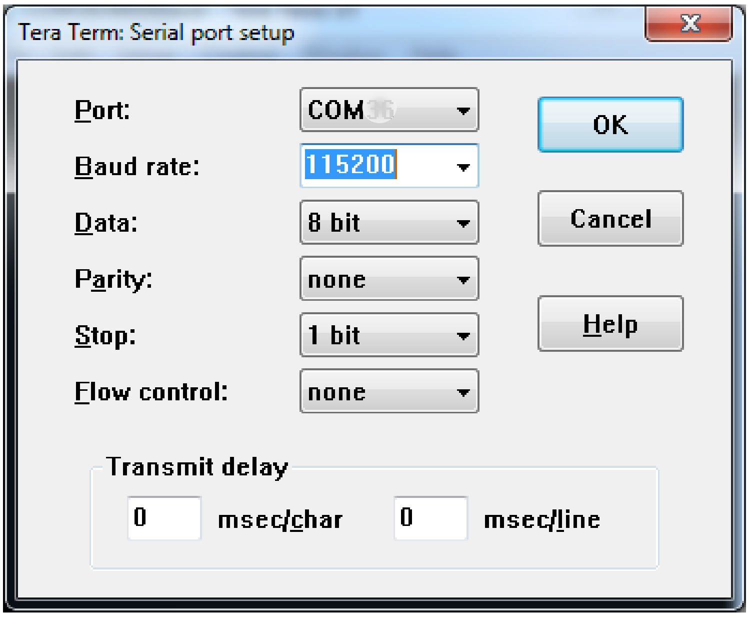
Tera Term: New connection

○ TCP/IP Host: myhost.example.com
☑ History
Service: ○ Telnet TCP port#: 22
● SSH SSH version: SSH2
○ Other Protocol: UNSPEC

● Serial Port: COM15: USB Serial Port (COM15)

Figure 21 – 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.



Tera Term: Serial port setup

Port: COM

Baud rate: 115200

Data: 8 bit

Parity: none

Stop: 1 bit

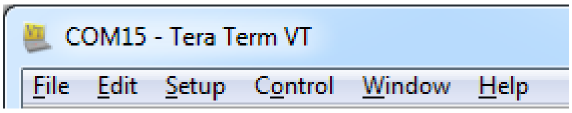
Flow control: none

Transmit delay

0 msec/char 0 msec/line

Figure 22 – Tera Term Serial Port Setup

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



COM15 - Tera Term VT

Figure 23 – Tera Term Menu

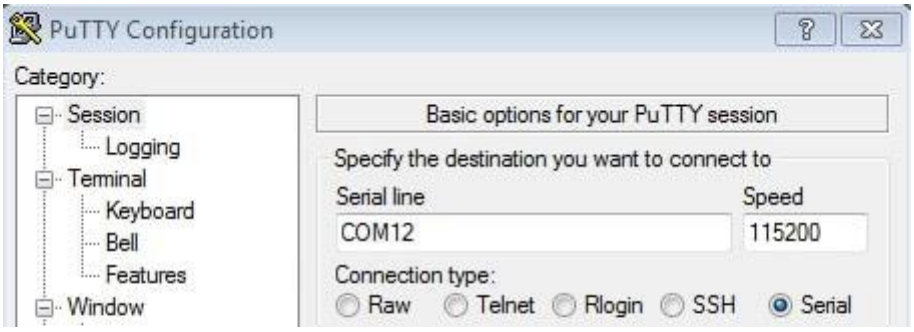
9.2.2 PuTTY terminal emulation application

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.



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: COM12

Speed: 115200

Connection type: ○ Raw ○ Telnet ○ Rlogin ○ SSH ● Serial

Figure 24 – 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 has to be stored in an external memory. On the *iMX RT OEM board*, this external memory is a 4 MByte OctalSPI, called EcoXiP ATXP032 from Adesto Technologies. The i.MX RT MCU always boots (i.e., starts executing) from this memory.

The EcoXiP memory is a low-power memory with a high-performance interface that operates in DDR (Double Data Rate) mode at 131 MHz. The performance when executing directly from the EcoXiP memory is very good. For more information about this product family, see http://www.adestotech.com/products/ecoxip/

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.

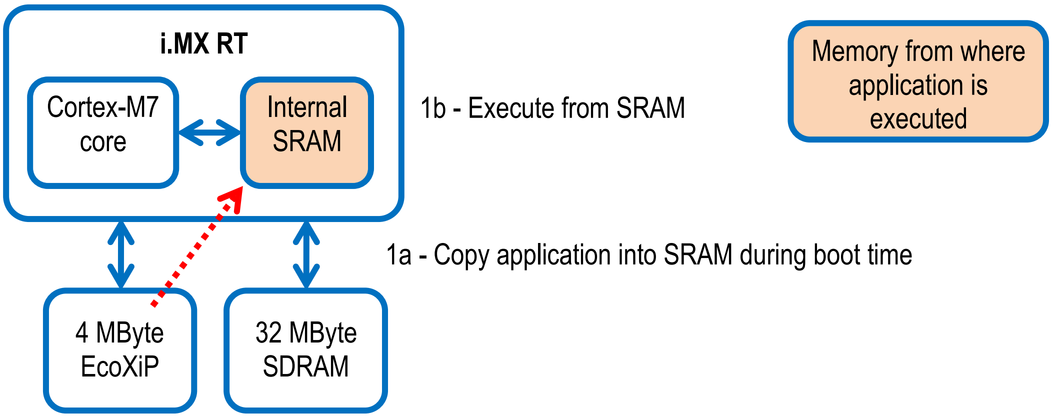

i.MX RT
Cortex-M7 core
Internal SRAM
1b - Execute from SRAM
Memory from where application is executed
1a - Copy application into SRAM during boot time
4 MByte EcoXiP
32 MByte SDRAM

Figure 25 – i.MX RT and EcoXiP - Executing from SRAM

1. The application is stored in the EcoXiP 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 EcoXiP 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.

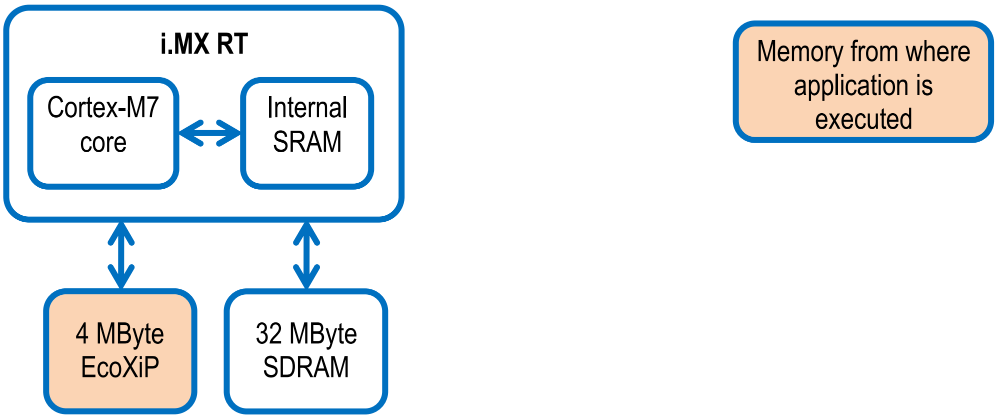


Figure 26 – i.MX RT and EcoXiP - Executing from EcoXiP

2. The application is stored in the EcoXiP 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. The execution performance will be about 2/3 of the performance when executing from the EcoXiP (assuming 5% cache miss, which is typical).
   b. CoreMark score (about) 1736 / 1912 when running at 528 / 600 MHz
   c. During program development the application must be downloaded/flashed to the EcoXiP 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.

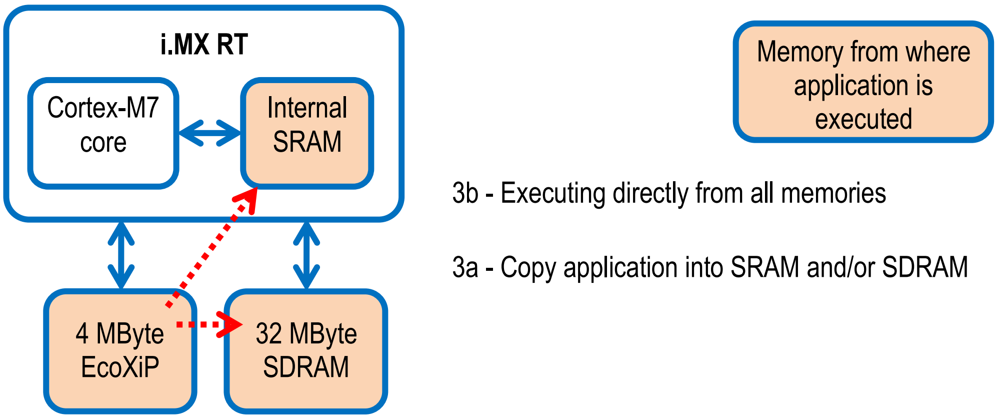

i.MX RT
Cortex-M7 core
Internal SRAM
4 MByte EcoXiP
32 MByte SDRAM
Memory from where application is executed
3b - Executing directly from all memories
3a - Copy application into SRAM and/or SDRAM

Figure 27 – i.MX RT and EcoXiP - 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 EcoXiP. 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 EcoXiP 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.

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) precaution.

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 push-buttons 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.