# Getting Started with MCUXpresso SDK for EVK-MIMXRT1020

# **1** Overview

The MCUXpresso Software Development Kit (SDK) provides comprehensive software support for Kinetis and LPC Microcontrollers. The MCUXpresso SDK includes a flexible set of peripheral drivers designed to speed up and simplify development of embedded applications. Along with the peripheral drivers, the MCUXpresso SDK provides an extensive and rich set of example applications covering everything from basic peripheral use case examples to full demo applications. The MCUXpresso SDK contains FreeRTOS and various other middleware to support rapid development.

For supported toolchain versions, see *MCUXpresso SDK Release Notes for EVK-MIMXRT1020* (document MCUXSDKMIMXRT1021RN).

For more details about MCUXpresso SDK, refer to MCUXpresso-SDK: Software Development Kit for MCUXpresso.

### Contents

1	Overview	1
2	MCUXpresso SDK board support package folders	2
3	Run a demo application using IAR	3
4	Run a demo using Keil® MDK/ μVision	7
5	Run a demo using Arm® GCC	11
6	Run a demo using MCUXpresso IDE	19
7	MCUXpresso Config Tools	28
8	MCUXpresso IDE New Project Wizard	29
9	How to determine COM port	29
10	How to add or remove boot header for XIP targets	30



#### MCUXpresso SDK board support package folders

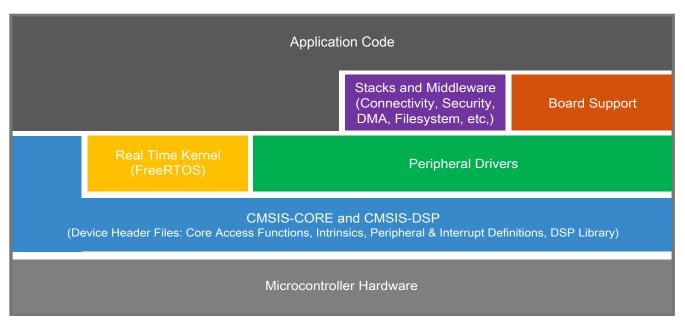


Figure 1. MCUXpresso SDK layers

# 2 MCUXpresso SDK board support package folders

MCUXpresso SDK board support package provides example applications for NXP development and evaluation boards for Arm<sup>®</sup> Cortex<sup>®</sup>-M cores including Freedom, Tower System, and LPCXpresso boards. Board support packages are found inside the top level boards folder and each supported board has its own folder (an MCUXpresso SDK package can support multiple boards). Within each <board\_name> folder, there are various sub-folders to classify the type of examples it contain. These include (but are not limited to):

- demo\_apps: Full-featured applications that highlight key functionality and use cases of the target MCU. These applications typically use multiple MCU peripherals and may leverage stacks and middleware.
- driver\_examples: Simple applications that show how to use the MCUXpresso SDK's peripheral drivers for a single use case. These applications typically only use a single peripheral but there are cases where multiple peripherals are used (for example, SPI conversion using DMA).
- rtos\_examples: Basic FreeRTOS<sup>TM</sup> OS examples that show the use of various RTOS objects (semaphores, queues, and so on) and interfaces with the MCUXpresso SDK's RTOS drivers
- wireless\_examples: Applications that use the Zigbee and OpenThread stacks.

# 2.1 Example application structure

This section describes how the various types of example applications interact with the other components in the MCUXpresso SDK. To get a comprehensive understanding of all MCUXpresso SDK components and folder structure, see *MCUXpresso SDK API Reference Manual*.

Each <board\_name> folder in the boards directory contains a comprehensive set of examples that are relevant to that specific piece of hardware. Although we use the hello\_world example (part of the demo\_apps folder), the same general rules apply to any type of example in the <board\_name> folder.

In the hello\_world application folder you see the following contents:

armgcc		
📜 iar	-	Toolchain folders: project and linker files
mdk		
📔 board.c		Board macro definitions (LEDs, buttons, etc)
📔 board.h		board macro deminions (LEDS, buttons, etc)
clock_config.c		Application encoding clock configuration
📓 clock_config.h	5	Application-specific clock configuration
hello_world.bin	$\rightarrow$	Pre-compiled application
📓 hello_world.c	$\rightarrow$	Application main source file
kello_world.mex		Application-specific MCUXpresso Config Tool configuration
hello_world.xml		Project definition file for MCUXpresso IDE and PG
in_mux.c	7	Application aposition in configuration
📓 pin_mux.h		Application-specific pin configuration
readme.txt		Description and instructions for running

### Figure 2. Application folder structure

All files in the application folder are specific to that example, so it is easy to copy and paste an existing example to start developing a custom application based on a project provided in the MCUXpresso SDK.

# 2.2 Locating example application source files

When opening an example application in any of the supported IDEs, a variety of source files are referenced. The MCUXpresso SDK devices folder is the central component to all example applications. It means the examples reference the same source files and, if one of these files is modified, it could potentially impact the behavior of other examples.

The main areas of the MCUXpresso SDK tree used in all example applications are:

- devices/<device name>: The device's CMSIS header file, MCUXpresso SDK feature file and a few other files
- devices/<device\_name>/drivers: All of the peripheral drivers for your specific MCU
- devices/<device\_name>/<tool\_name>: Toolchain-specific startup code, including vector table definitions
- devices/<device\_name>/utilities: Items such as the debug console that are used by many of the example applications
- devices/<devices\_name>/project Project template used in CMSIS PACK new project creation

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

# 3 Run a demo application using IAR

# 3.1 Build an example application

Do the following steps to build the hello\_world example application.

#### Run a demo application using IAR

1. Open the desired demo application workspace. Most example application workspace files can be located using the following path:

<install\_dir>/boards/<board\_name>/<example\_type>/<application\_name>/iar

Using the EVK-MIMXRT1020 hardware platform as an example, the hello\_world workspace is located in:

<install\_dir>/boards/evkmimxrt1020/demo\_apps/hello\_world/iar/hello\_world.eww

Other example applications may have additional folders in their path.

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

There are twelve project configurations (build targets) supported for most MCUXpresso SDK projects:

- Debug Compiler optimization is set to low, and debug information is generated for the executable. The linker file is RAM linker, where text and data section is put in internal TCM.
- Release Compiler optimization is set to high, and debug information is not generated. The linker file is RAM linker, where text and data section is put in internal TCM.
- ram\_0x1400\_debug Project configuration is same as the debug target. The linker file is RAM\_0x1400 linker, where text is put in ITCM with offset 0x1400 and data put in DTCM.
- ram\_0x1400\_release Project configuration is same as the release target. The linker file is RAM\_0x1400 linker, where text is put in ITCM with offset 0x1400 and data put in DTCM.
- sdram\_debug Project configuration is same as the debug target. The linker file is SDRAM linker, where text is put in internal TCM and data put in SDRAM.
- sdram\_release Project configuration is same as the release target. The linker file is SDRAM linker, where text is put in internal TCM and data put in SDRAM.
- sdram\_txt\_debug Project configuration is same as the debug target. The linker file is SDRAM\_txt linker, where text is put in SDRAM and data put in OCRAM.
- sdram\_txt\_release Project configuration is same as the release target. The linker file is SDRAM\_txt linker, where text is put in SDRAM and data put in OCRAM.
- flexspi\_nor\_debug Project configuration is same as the debug target. The linker file is flexspi\_nor linker, where text is put in flash and data put in TCM.
- flexspi\_nor\_release Project configuration is same as the release target. The linker file is flexspi\_nor linker, where text is put in flash and data put in TCM.
- flexspi\_nor\_sdram\_release Project configuration is same as the release target. The linker file is flexspi\_nor\_sdram linker, where text is put in flash and data put in SDRAM.
- flexspi\_nor\_sdram\_debug Project configuration is same as the debug target. The linker file is flexspi\_nor\_sdram linker, where text is put in flash and data put in SDRAM.

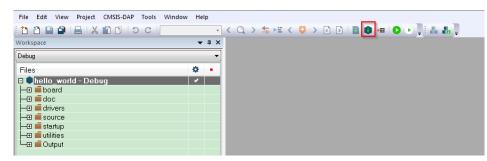
For some examples need large data memory, only sdram\_debug and sdram\_release targets are supported.

For this example, select hello\_world – debug.

Workspace 🔫	ά×
debug	~
debug	
flexspi_nor_debug	
sdram_debug	
flexspi_nor_sdram_debug	
sdram_txt_debug	
ram_0x1400_debug	
release	
flexspi_nor_release	
sdram_release	
flexspi_nor_sdram_release	
sdram_txt_release	
ram_0x1400_release	_
	•
- 🕀 📫 utilities	•

### Figure 3. Demo build target selection

3. To build the demo application, click Make, highlighted in red in Figure 4.



# Figure 4. Build the demo application

4. The build completes without errors.

# 3.2 Run an example application

To download and run the application, perform these steps:

- 1. This board supports the CMSIS-DAP/mbed/DAPLink debug probe by default. Visit os.mbed.com/handbook/Windowsserial-configuration and follow the instructions to install the Windows<sup>®</sup> operating system serial driver. If runni ng on Linux OS, this step is not required.
- 2. Connect the development platform to your PC via USB cable. Connect the USB cable to J41 and make sure SW7[1:4] is **0010b**.
- 3. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug COM port (to determine the COM port number, see Appendix A). Configure the terminal with these settings:
  - a. 115200 or 9600 baud rate, depending on your board (reference BOARD\_DEBUG\_UART\_BAUDRATE variable in the board.h file)
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit

Run a demo application using IAR

Session	Basic options for yo	ur PuTTY exercion
Logging	the second se	2.550 CO.0
- Terminal	Specify the destination you w	
- Keyboard	Serial line	Speed
Bell	COM16	115200
- Features Window	Connection type: Raw D Telnet Ro	ogin 💿 <u>S</u> SH 💿 Serjal
Appearance Behaviour Translation Selection Colours Connection	Load, save or delete a stored Sav <u>e</u> d Sessions	session
	Debug	
	Default Settings Debug	Load
Data Proxy		Sa <u>v</u> e
- Telnet Rlogin ⊕- SSH Serial		Delete
	Close window on exit: Always Never	Only on clean exit

# Figure 5. Terminal (PuTTY) configuration

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



# Figure 6. Download and Debug button

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

File Edit View Project Debug	Disassembly	CMSIS-DAP Tools Window Help
1 1 🖻 🖬 📲 🕹 🖍 🛍 🗅	5 C	🖬 < Q > ⇆ म्म < 📮 > 🕢 🗈 📄 🗰 🖷 🕒 C 😆 📮 🕩 म 🖻 म 🧃
Vorkspace	<b>→</b> ‡ ×	hello_world.c x
debug	~	main()
Files	۰ ¢	54 * @brief Main function
🗆 🌒 hello_world - debug	~	55 L */
- 🕀 🛋 board		➡ 56 int main(void)
—⊞ 🛋 doc		57 🖯 {
⊢⊕ 🛋 drivers		58 char ch;
- 🕀 🛋 source		59
🗕 🖬 🖬 startup		60 /* Init board hardware. */
🗕 🖅 🛋 utilities		
–⊞ 🛋 xip		61 BOARD_ConfigMPU();
🖵 🖬 Output		62 BOARD_InitPins();
		63 BOARD_BootClockRUN();

### Figure 7. Stop at main() when running debugging

6. Run the code by clicking the Go button to start the application.



# Figure 8. Go button

7. The hello\_world application is now running and a banner is displayed on the terminal. If this is not true, check your terminal settings and connections.



Figure 9. Text display of the hello\_world demo

# 4 Run a demo using Keil<sup>®</sup> MDK/µVision

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

# 4.1 Install CMSIS device pack

After the MDK tools are installed, Cortex<sup>®</sup> 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 MIMXRT102x CMSIS pack.

1. Download the MIMXRT1020 and MIMXRT1021 packs.

#### Run a demo using Keil® MDK/µVision

2. After downloading the DFP, double click to install it.

# 4.2 Build an example application

1. Open the desired example application workspace in:

<install\_dir>/boards/<board\_name>/<example\_type>/<application\_name>/mdk

The workspace file is named as <demo\_name>.uvmpw. For this specific example, the actual path is:

<install\_dir>/boards/evkmimxrt1020/demo\_apps/hello\_world/mdk/hello\_world.uvmpw
2. To build the demo project, select Rebuild, highlighted in red.



### Figure 10. Build the demo

3. The build completes without errors.

# 4.3 Run an example application

To download and run the application, perform these steps:

- 1. This board supports the CMSIS-DAP/mbed/DAPLink debug probe by default. Visit os.mbed.com/handbook/Windowsserial-configuration and follow the instructions to install the Windows<sup>®</sup> operating system serial driver. If running on Linux OS, this step is not required.
- 2. Connect the development platform to your PC via USB cable.
- 3. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug serial port number (to determine the COM port number, see How to determine COM port). Configure the terminal with these settings:
  - a. 115200 or 9600 baud rate, depending on your board (reference BOARD\_DEBUG\_UART\_BAUDRATE variable in board.h file)
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit

tegory:								
	Basic options for your PuTTY session							
Logging	Specify the destination you wa	ant to connect to						
- Keyboard	Serial li <u>n</u> e	Speed						
Bell	COM16	115200						
Features Window Appearance Behaviour Translation Selection Colours Connection	Connection type: Raw <u>T</u> elnet      Rlog	gin 🔘 <u>S</u> SH 🔘 Serial						
	Load, save or delete a stored Sav <u>e</u> d Sessions	session						
	Debug							
	Default Settings	Load						
Data Proxy		Sa <u>v</u> e						
- Telnet Rlogin ⊕- SSH Serial		Delete						
	Close window on e <u>x</u> it: Always Never (	Only on clean exit						

# Figure 11. Terminal (PuTTY) configurations

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

#### Run a demo using Keil® MDK/µVision

File	Edit View	Project	Flash	Deb	ug	Peripherals	Tools	SVCS	Window	Help								
	i 🚽 🚽	χþ	<b>1</b>	9 0		┩   ← ┥	12.12	1.19		//= // <sub>%</sub>	1	BL_TAR	GET_RAM	I	~ 🗟	an   🤇	Q 🕘	୍ 🔗
RST	🗐 🚳 🗐	b 6 6	P +()	\$	>	Q. 🖪 🚍	æ 🖧	a - 📖	I - 📴 ·	- 👿 -	<b>III</b> -	• -	※ -					
Registe	rs		д	x	Disas	sembly												<b>д </b> 🗙
Registe	er	Value				61:			nfigMPU	J();								^
Cc	ore	_			<mark>≺&gt;</mark> 0×	00002480			BL.W	0	BC	DARD_0	ConfigN	1PU (	0 <b>x</b> 0000	0934)		
	R0	0x0000			0.1	62: 00002484		_	itPins BL.W	07	PC		ni+Di.		x000001	9701		~
	R1	0x2000			<	00002464	P/PP.	rn/0	D11.W		n.	ARI	nn.en	18 10.	<b>K</b> ()()()()	n/01		>
	R2	0x0000				· · · ·												
	R3 R4	0x0000 0x0000				hello_wo												<b>▼</b> ×
	R5	0x0000				54	∗ @br	ief	Main f	functi	on							^
	R6	0x0000				55 L	*/											
	R7	0x0000				56 i	nt ma	in(v	oid)									
	R8	0x0000	0000			57 ⊟ {												
	R9	0x0000				58	ch	ar c	h:									
	R10	0x0000				59			1									
	R11	0x0000				60	/*	Ini	t boar	d har	dwar	е. *	/					
	···· R12 ···· R13 (SP)	0x0000 0x2001			$\triangleright$	61			Config				·					
		0x2001			~~	62			InitPi									
	R15 (PC)	0x0000				63		_	BootCl									
÷	xPSR	0x6100				64		_	InitDe									
E Ba	anked					65	DU	ARD_	TUICDE	bugco	nsot	.e();						
	stem						DE	TNTE	(11-11			A	· ·					
Internet	emal	_			-	66	PF		("hell	.o wor	. ra . /	(r \n"	);					
	Mode	Thread			L.,	67												
	···· Privilege ···· Stack	Privileg MSP	ea			68		ile	(1)									
	- States	6542				<b>69</b> 🛱	{											
	···· Sec	0.0006	5420			70			= GET		);							
						71		PU	TCHAR (	(ch);								
						72 -	}	_										~
🖭 Proj	ject 🛛 🗮 Reg	isters			<													>

### Figure 12. Stop at main() when run debugging

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

RST	1 🛛	<b>{</b> *}	0	{}	*{}
Registers	E Ru	n (F5)	)		-
Register		rt co	_	ecut	ion
E Core	•				
	RO		0x0	0000	965
-	R1		0x1	FFF04	440

Figure 13. Run button

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

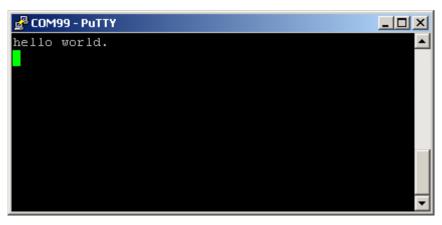


Figure 14. Text display of the hello\_world demo

# 5 Run a demo using Arm<sup>®</sup> GCC

This section describes the steps to configure the command line Arm<sup>®</sup> GCC tools to build, run, and debug demo applications and necessary driver libraries provided in the MCUXpresso SDK. The hello\_world demo application is targeted which is used as an example.

### NOTE

GCC ARM Embedded 8.2.1 is used as an example in this document. The latest GCC version for this package is as described in the *MCUXpresso SDK Release Notes*.

# 5.1 Set up toolchain

This section contains the steps to install the necessary components required to build and run an MCUXpresso SDK demo application with the Arm GCC toolchain, as supported by the MCUXpresso SDK. There are many ways to use Arm GCC tools, but this example focuses on a Windows operating system environment.

# 5.1.1 Install GCC ARM Embedded tool chain

Download and run the installer from launchpad.net/gcc-arm-embedded. This is the actual toolset (in other words, compiler, linker, etc.). The GCC toolchain should correspond to the latest supported version, as described in *MCUXpresso SDK Release Notes Supporting MIMXRT1020-EVK* (document MCUXSDKMIMXRT102XRN).

# 5.1.2 Install MinGW (only required on Windows OS)

The Minimalist GNU for Windows (MinGW) development tools provide a set of tools that are not dependent on third-party C-Runtime DLLs (such as Cygwin). The build environment used by the MCUXpresso SDK does not use the MinGW build tools, but does leverage the base install of both MinGW and MSYS. MSYS provides a basic shell with a Unix-like interface and tools.

- 1. Download the latest MinGW mingw-get-setup installer from sourceforge.net/projects/mingw/files/Installer/.
- 2. Run the installer. The recommended installation path is C:\MinGW, however, you may install to any location.

#### NOTE

The installation path cannot contain any spaces.

3. Ensure that the mingw32-base and msys-base are selected under Basic Setup.

🏇 MinGW Installation Manager					
Installation Package Settings					
Basic Setup	Package	Class	Installed Version	Repository Version	Description
All Packages	mingw-developer-tool	bin		2013072300	An MSYS Installation for MinGW Developers (meta)
	mingw32-base	bin		2013072200	A Basic MinGW Installation
	mingw32-gcc-ada	bin		4.8.1-4	The GNU Ada Compiler
	mingw32-gcc-fortran	bin		4.8.1-4	The GNU FORTRAN Compiler
	mingw32-gcc-g++	bin		4.8.1-4	The GNU C++ Compiler
	mingw32-gcc-objc	bin		4.8.1-4	The GNU Objective-C Compiler
	🐑 msys-base	bin		2013072300	A Basic MSYS Installation (meta)

# Figure 15. Set up MinGW and MSYS

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

Installation Package Sett	ings		
Update Catalogue			Package
Mark All Upgrades			mingw-developer-tool
Apply Changes		_	mingw32-base
			mingw32-gcc-ada
Quit	Alt+F4		mingw32-gcc-fortran
111129	11		mingw32-gcc-g++
			mingw32-gcc-objc
		5	msys-base

### Figure 16. Complete MinGW and MSYS installation

5. Add the appropriate item to the Windows operating system path environment variable. It can be found under Control Panel->System and Security->System->Advanced System Settings in the Environment Variables... section. The path is:

<mingw\_install\_dir>\bin

Assuming the default installation path, C:\MinGW, an example is shown below. If the path is not set correctly, the toolchain will not not work.

#### NOTE

If you have C:\MinGW\msys\x.x\bin in your PATH variable (as required by Kinetis SDK 1.0.0), remove it to ensure that the new GCC build system works correctly.

Variable value: System variables Variable Val OS Wir	Path Dgram Files (x86)\CMake\bin;C:\MinGW\b OK Cancel
Edit System Variable Variable name: Variable value: System variables Variable Val OS Wii	Path ogram Files (x86)\CMake\bin;C:\MinGW\b
Variable name: Variable value: System variables Variable Val OS Wir	Path ogram Files (x86)\CMake\bin;C:\MinGW\b
Variable name: Variable value: System variables Variable Val OS Wir	Path ogram Files (x86)\CMake\bin;C:\MinGW\b
Variable value: System variables Variable Val OS Wir	ogram Files (x86)\CMake\bin;C:\MinGW\b
Variable value: System variables Variable Val OS Wir	ogram Files (x86)\CMake\bin;C:\MinGW\b
System variables Variable Val OS Wii	
Variable Val OS Wir	OK Cancel
Variable Val OS Wir	OK Cancel
Variable Val OS Wir	
Variable Val OS Wir	
OS Wir	
	ue
Path C:	ndows_NT
	Program Files (x86)\Parallels\Parallel
PATHEXT .CO	DM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;
PROCESSOR_A AM	D64
	New Edit Delete
PATHEXT .CC PROCESSOR_A AM	DM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS; D64

Figure 17. Add Path to systems environment

# 5.1.3 Add a new system environment variable for ARMGCC\_DIR

Create a new *system* environment variable and name it as ARMGCC\_DIR. The value of this variable should point to the Arm GCC Embedded tool chain installation path. For this example, the path is:

See the installation folder of the GNU Arm GCC Embedded tools for the exact path name of your installation.

Short path should be used for path setting, you could convert the path to short path by running command "for %I in (.) do echo %~sI" in above path.

C:\Program Files (x86)\GNU Tools Arm Embedded\8 2018-q4-major>for %I in (.) do echo %`sI C:\Program Files (x86)\GNU Tools Arm Embedded\8 2018-q4-major>echo C:\PROGRA^2\GNUTOO^1\82018-^1 C:\PROGRA^2\GNUTOO^1\82018-^1

Figure 18. Convert path to short path

Environment Va	riables						;	×	
User variable	s for								
Variable		Value							
OneDrive		C:\Users\	١	OneDrive - NXP	)				
OneDriveO	onAmercia	C:\Users\	1	OneDrive - NXP	)				
Path		C:\Ruby24->	:\Ruby24-x64\bin;C:\Users\nxa07599\AppData\Local\Micros						
PATHEXT		.COM; EXE; I	BAT;.CM	D;.VBS;.VBE;.JS;.J	ISE;.WSF;.WSH;.M	SC;.RB	;.RB		
TEMP		C:\Users\	\	AppData\Local	\Temp				
TMP		C:\Users\	١	AppData\Local	\Temp				
New User Variab	e							X	
Variable name:	ARN	IGCC_DIR							
Variable value:	C:\P	ROGRA~2\GNUTOO	~1\8201	8-~1					
valiable value.	C. (1		10201	U 1					
Browse Direct	ory	Browse File			OK		Canc	el	
IAR_WORK	BENCH	C:\Program	Files (x	36)\IAR Systems	Embedded Wor	kbencl	h 8.2		
JLINK_DIR		C:\Program	Files (x	86)\SEGGER\JLir	nk_V640				
KEIL		C:\Keil_v5\U	JV4						
myCleanUp	•	No							
		() ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( )	-				¥		
				New	Edit	C	Delete		
						~			
					ОК	Ca	ancel		

Figure 19. Add ARMGCC\_DIR system variable

# 5.1.4 Install CMake

- 1. Download CMake 3.0.x from www.cmake.org/cmake/resources/software.html.
- 2. Install CMake, ensuring that the option Add CMake to system PATH is selected when installing. The user chooses to select whether it is installed into the PATH for all users or just the current user. In this example, it is installed for all users.

🛕 CMake 3.0.2 Setup	1	
	Install Options Choose options for installing CMa	ke 3.0.2
By default CMake d	bes not add its directory to the system PAT	н.
Add CMake to the	ee to the system PATH e system PATH for all users e system PATH for current user	
🔲 Create CMake D	esktop Icon	
Nullsoft Install System	v2.46	Next > Cancel

### Figure 20. Install CMake

- 3. Follow the remaining instructions of the installer.
- 4. You may need to reboot your system for the PATH changes to take effect.
- 5. Make sure sh. exe is not in the Environment Variable PATH. This is a limitation of mingw32-make.

# 5.2 Build an example application

To build an example application, follow these steps.

1. Open a GCC Arm Embedded tool chain command window. To launch the window, from the Windows operating system **Start** menu, go to **Programs** ->**GNU Tools ARM Embedded <version>** and select **GCC Command Prompt**.

📕 GNU Tools for ARM Embedded Process
Documentation
GCC Command Prompt
🎯 Uninstall GNU Tools for ARM Embed

### Figure 21. Launch command prompt

2. Change the directory to the example application project directory which has a path similar to the following:

<install\_dir>/boards/<board\_name>/<example\_type>/<application\_name>/armgcc

For this example, the exact path is:

<install\_dir>/examples/evkmimxrt1020/demo\_apps/hello\_world/armgcc

#### NOTE

To change directories, use the cd command.

3. Type **build\_debug.bat** on the command line or double click on **build\_debug.bat** file in Windows Explorer to build it. The output is as shown in Figure 22.

[ 95%] Building C object CMakeFiles/hello\_world.elf.dir/C\_/repol/mcu-sdk-2.0/boards/evkmimxrt1020/xip /evkmimxrt1020\_sdram\_ini\_dcd.c.obj [100%] Linking C executable debug\hello\_world.elf [100%] Built target hello\_world.elf C:\repol\mcu-sdk-2.0\boards\evkmimxrt1020\demo\_apps\hello\_world\armgcc>IF "" == "" (pause ) Press any key to continue . . .

### Figure 22. hello\_world demo build successful

# 5.3 Run an example application

This section describes steps to run a demo application using J-Link GDB Server application. To perform this exercise, make sure that either:

- The OpenSDA interface on your board is programmed with the J-Link OpenSDA firmware. If your board does not support OpenSDA, then a standalone J-Link pod is required.
- You have a standalone J-Link pod that is connected to the debug interface of your board.

#### NOTE

Some hardware platforms require hardware modification in order to function correctly with an external debug interface.

After the J-Link interface is configured and connected, follow these steps to download and run the demo applications:

- 1. This board supports the J-Link debug probe. Before using it, install SEGGER software, which can be downloaded from http://www.segger.com.
- 2. Connect the development platform to your PC via USB cable between the OpenSDA USB connector and the PC USB connector. If using a standalone J-Link debug pod, also connect it to the SWD/JTAG connector of the board.
- 3. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug serial port number (to determine the COM port number, see Appendix A). Configure the terminal with these settings:
  - a. 115200 or 9600 baud rate, depending on your board (reference BOARD\_DEBUG\_UART\_BAUDRATE variable in the board.h file)
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit

tegory:		
Session	Basic options for your PuTTY	session
Logging - Teminal	Specify the destination you want to con	nnect to
Keyboard	Serial li <u>n</u> e	Speed
Bell	COM16	115200
Features ⊒-Window	Connection type: Raw <u>T</u> elnet Rlogin <u></u>	SH 💿 Sețial
Appearance Behaviour Translation	Load, save or delete a stored session Sav <u>e</u> d Sessions	
- Selection	Debug	
Colours	Default Settings Debug	Load
- Data - Proxy		Sa <u>v</u> e
Telnet		Delete
⊕ SSH Serial	Close window on exit: Always Never Only o	n clean exit

### Figure 23. Terminal (PuTTY) configurations

- 4. Open the J-Link GDB Server application. Go to the SEGGER install folder. For example, C:\Program Files(x86)\SEGGER\JLink\_Vxxx. Open the command windows. For Debug and Release targets, use the JLinkGDBServer.exe command. For the sdram\_debug, sdram\_release, flexspi\_nor\_sdram\_debug, and flexspi\_nor\_sdram\_release targets, use the JLinkGDBServer.exe-scriptfile <install\_dir>/boards/ evkmimxrt1020/demo\_apps/hello\_world/evkmimxrt1020\_sdram\_init.jlinkscript command
- 5. The target device selection chosen for this example is MIMXRT1021DAG5A.
- 6. After it is connected, the screen should resemble this figure:

Run a demo using Arm® GCC		
SEGGER J-Link GDB Server V6.32c	_	
File Help		
GDB Waiting for connection J-Link Connected SWD Device RT1021DAG5A (Halted) 3.30V	4000 kHz ✓ Show	on top v log window rate logfile y download
ClearLog J-Link is connected. Firmware: J-Link V10 compiled Apr 20 2018 16: Hardware: V10.10 S/N: 600101602 Feature(s): RDI, FlashBP, FlashDL, JFlash, GI Checking target voltage Target voltage: 3.30 V		^
Listening on TCP/IP port 2331 Connecting to target Connected to target Waiting for GDB connection		~
0 bytes downloaded	Connected to target	

### Figure 24. SEGGER J-Link GDB Server screen after successful connection

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



# Figure 25. Launch command prompt

8. Change to the directory that contains the example application output. The output can be found in using one of these paths, depending on the build target selected:

<install\_dir>/boards/<board\_name>/<example\_type>/<application\_name>//armgcc/debug <install\_dir>/boards/<board\_name>/<example\_type>/<application\_name>//armgcc/release

For this example, the path is:

<install\_dir>/boards/evkmimxrt1020/demo\_apps/hello\_world/armgcc/debug

9. Run the arm-none-eabi-gdb.exe <application\_name>.elf. For this example, it is arm-none-eabi-gdb.exe hello\_world.elf.

GCC Command Prompt - arm-none-eabi-gdb.exe C:\repo1\mcu-sdk-2.0\boards\evkmimxrt1020\demo_apps\h
C:\Program Files (x86)\GNU Tools ARM Embedded\7 2017-q4-major\bin>arm-none-eabi-gdb.exe C:\rep ol\mcu-sdk-2.0\boards\evkmimxrt1020\demo_apps\hello_world\armgcc\debug\hello_world.elf GNU gdb (GNU Tools for Arm Embedded Processors 7-2017-q4-major) 8.0.50.20171128-git Copyright (C) 2017 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <http: gnu.org="" gpl.html="" licenses=""></http:>
This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law. Type "show copying" and "show warranty" for details. This GDB was configured as "host=i686-w64-mingw32target=arm-none-eabi". Type "show configuration" for configuration details.
For bug reporting instructions, please see: <http: bugs="" gdb="" software="" www.gnu.org=""></http:> . Find the GDB manual and other documentation resources online at: <http: documentation="" gdb="" software="" www.gnu.org=""></http:> . For help, type "help".
Type "apropos word" to search for commands related to "word" Reading symbols from C:\repol\mcu-sdk-2.0\boards\evkmimxrt1020\demo_apps\hello_world\armgcc\de bug\hello_world.elfdone. (gdb)

### Figure 26. Run arm-none-eabi-gdb

- 10. Run these commands:
  - a. target remote localhost:2331
  - b. monitor reset
  - $c.\ \mbox{monitor}$  halt
  - d. load
- 11. The application is now downloaded and halted at the reset vector. Execute the monitor go command to start the demo application.

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

# 6 Run a demo using MCUXpresso IDE

#### NOTE

Ensure that the MCUXpresso IDE toolchain is included when generating the MCUXpresso SDK Package.

MCUXPresso IDE is not supported in this release.

This section describes the steps required to configure MCUXpresso IDE to build, run, and debug example applications. The hello\_world demo application targeted for the MIMXRT1020-EVK platform is used as an example, though these steps can be applied to any example application in the MCUXpresso SDK.

#### NOTE

BY default. three macros, XIP\_EXTERNAL\_FLASH=1, XIP\_BOOT\_HEADER\_ENABLE=1, and XIP\_BOOT\_HEADER\_DCD\_ENABLE=1 are set in the project. If you do not use Board\_Flash in the project, these macros should be removed or set value to **0** in project settings.

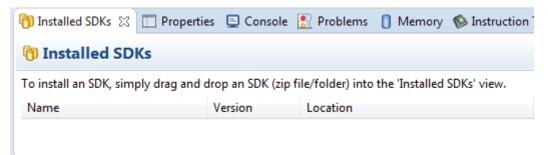
# 6.1 Select the workspace location

Every time MCUXpresso IDE launches, it prompts the user to select a workspace location. MCUXpresso IDE is built on top of Eclipse which uses workspace to store information about its current configuration, and in some use cases, source files for the projects are in the workspace. The location of the workspace can be anywhere, but it is recommended that the workspace be located outside of the MCUXpresso SDK tree.

# 6.2 Build an example application

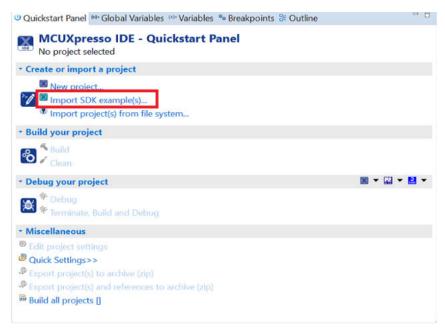
To build an example application, follow these steps.

1. Drag and drop the SDK zip file into the **Installed SDKs** view to install the MCUXpresso SDK. In the window that appears, click the **OK** button and wait until the import has finished.



### Figure 27. Install an SDK

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



### Figure 28. Import an SDK example

3. In the window that appears, expand the **MIMXRT1020** folder and select **MIMXRT1021xxxxx**. Then, select **evkmimxrt1020** and click the **Next** button.

#### Run a demo using MCUXpresso IDE



# Figure 29. Selecting MIMXRT1020-EVK board

4. Expand the demo\_apps folder and select hello\_world. Then, click the Next button.

Run a demo using MCUXpresso IDE				
🔀 SDK Import Wizard			_	пх
A The source from the SDK will be copied into the work				
If you want to use linked files, please unzip the 'SDK_'	2.x_EVK-MIMXRT1020' SI	DK.		
🔀 Import projects				
Project name prefix: evkmimxrt1020_	2 Projec	t name suffix:		Ø.
✓ Use default location				
Location: C:\Users\nxf37085\Documents\MCUXpress	oIDE_10.2.0_729_prc1e\v	vorkspace\evkmimxrt1020_		Browse
Project Type		Project Options		
● C Project ○ C++ Project ○ C Static Library ○ C	2++ Static Library	SDK Debug Console 🤇	Semihost OUAR	
		Copy sources		
		Import other files		
Examples			èn   4	2 🗹 🎕 🗎 🖻 🖻
type to filter				
Name		Version		^
✓ ■ a demo_apps				
> 🔄 🗄 mbedtls > 🗍 🗏 wifi_gca				
> 🗌 🗄 wolfssl				
= bubble				
□ ≡ ecompass				
✓ = hello_world □ = hello_world_virtual_com				
				•
0	< Back	Next >	Finish	Cancel
	< DOCK	INCAL 2		Cancer

### Figure 30. Selecting hello\_world

5. Ensure the option **Redlib:** Use floating point version of printf is selected if the cases print floating point numbers on the terminal (for demo applications such as dac32\_adc12, dac\_adc, dac\_cadc, ecompass, sai, coremark, mbedtls\_benchmark, wolfssl\_benchmark, and for mmcau\_examples such as mmcau\_api). Otherwise, there is no need to select it. Click the **Finish** button.

#### Run a demo using MCUXpresso IDE

X SDK Import Wizard



X Advanced Settings		^
• C/C++ Library Settings		
Set library type (and hosting variant) Redlib (nohost-nf)	~	
Redlib: Use floating point version of printf Redlib: Use character rather than string based printf		e floating point version of printf e floating point version of scanf
Redirect SDK "PRINTF" to C library "printf" Include semihost HardFault handler	Redirect printf/sc Redirect printf/sc	
<ul> <li>Hardware settings</li> </ul>		
Set Floating Point type	Pv5-SP-D16 (HardABI)	~
• MCU C Compiler		
Language standard GNU C99 (-std=gnu99)		~
* MCU Linker		
Link application to RAM		
Memory Configuration		
Memory details		
	Size Driver	
Type Name Alias Location Flash BOARD_FLASH Flash 0x600000		1020-EVK_IS25LP064.cfx
RAM SRAM_DTC RAM 0x200000		1020-EVR_1325EF004.CIX
RAM SRAM_ITC RAM2 0x0	0x10000	Edit
RAM SRAM_OC RAM3 0x2020000		
RAM BOARD_SDRAM RAM4 0x800000	0 0x2000000	
		ţ.
? < Bac	K Next >	Finish Cancel

# Figure 31. Selecting User floating point version of printf

NOTE

If you want to use semihost to print log, first select the **Semihost** button when importing projects.

Run a demo using MCUXpresso IDE		
X SDK Import Wizard		x c
A The source from the SDK will be copied into the workspace. If you want to use linked files, please unzip the 'SDK_2.x_EVK-MIMXRT1020' SDK.		F
Import projects		
Project name prefix: evkmimxrt1020_	ject name suffix:	<i>B</i> _
Use default location		
Location: C:\Users\nxf41402\Documents\MCUXpressoIDE_10.2.0_757_prc4\workspace\evkmi	mxrt1020_	Browse
Project Type	Project Options	
● C Project ○ C++ Project ○ C Static Library ○ C++ Static Library	SDK Debug Console Semihost UART Copy sources Import other files	
Examples	🚵   🖉 🔽	🍂   🕀 🖂
type to filter		
Name	Version	~
0	< Back Next > Finish C	ancel

Figure 32. Selecting Semihost

#### Run a demo using MCUXpresso IDE

Properties for evkmimx	rt1020_hello_world				$\times$
type filter text	Settings			<-> →	• •
<ul> <li>&gt; Resource Builders</li> <li>&gt; C/C++ Build Build Variables Environment Logging MCU settings Settings Tool Chain Editor</li> <li>&gt; C/C++ General MCUXpresso Config T Project References Run/Debug Settings</li> </ul>	Configuration:       Debug [ Active ]         Tool Settings       Build steps       Build Artifact       Binary Parsers       Error Parsers         MCU C Compiler       Do not search system directories (-nostdinc)         Dialect       Preprocessor         Preprocessor       Defined symbols (-D)         Includes       CPU_MIMXRT1021DAF5A         Optimization       Debugging         Warnings       SDK DEBUGCONSOLE=0         PRINTF_FLOAT_ENABLE=0			Configuration       Configuration       Image: Second secon	ons
Task Tags > Validation	<ul> <li>Miscellaneous</li> <li>Architecture</li> <li>MCU Assembler</li> <li>General</li> <li>Architecture &amp; Headers</li> <li>MCU Linker</li> <li>General</li> <li>Libraries</li> <li>Miscellaneous</li> <li>Shared Library Settings</li> <li>Architecture</li> <li>Managed Linker Script</li> <li>Multicore</li> </ul>	PRINTF_FLOAT_ENABLE=0 SCANF_FLOAT_ENABLE=0 PRINTF_ADVANCED_ENABLE=0 SCANF_ADVANCED_ENABLE=0 XIP_EXTERNAL_FLASH=1 XIP_BOOT_HEADER_DCD_ENABLE=1 CR_INTEGER_PRINTF MCUXPRESSO USE_CMSIS Undefined symbols (-U)			
< >			Restore Defaults	Apply	/
?			Apply and Close	Cancel	

### Figure 33. Setting SDK\_DEBUGCONSOLE

6. On the **Quickstart** panel, click **build** evkmimxrt1020\_demo\_apps\_hello\_world [Debug].

Run a demo using MCUXpresso IDE	
	🙂 Quic 💁 Glob 🗠 Vari 🗣 Brea 🕃 Outli 🚍 🗖
	MCUXpresso IDE - Quickstart Panel Project: evkmimxrt1020_hello_world [Debug]
	<ul> <li>Create or import a project</li> </ul>
	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
	* Miscellaneous
	<ul> <li>Edit project settings</li> <li>Quick Settings&gt;&gt;</li> </ul>

Figure 34. Building hello world case

# 6.3 Run an example application

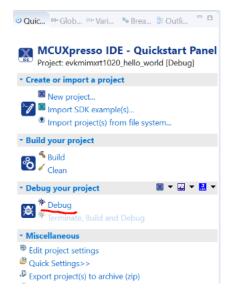
For more information on debug probe support in the MCUXpresso IDE, visit community.nxp.com.

To download and run the application, perform these steps:

### NOTE

Make sure that the board is on QSPI\_Flash mode before download (set SW8: 0010).

1. On the Quickstart panel, click Debug evkmimxrt1020\_demo\_apps\_hello\_world [Debug].



### Figure 35. Debugging hello\_world case

2. The first time you debug a project, the Debug Emulator Selection Dialog is displayed, showing all supported probes that are attached to your computer. Select the probe through which you want to debug and click the **OK** button. (For any future debug sessions, the stored probe selection is automatically used, unless the probe cannot be found.)

#### Run a demo using MCUXpresso IDE

X	Probes discovered				— [		×
Co	nnect to target: MIMXR	Т1021ххххх					
1	probe found. Select the prob	be to use:					
Av	ailable attached pro	bes					
	Name	Serial number/ID	Type	Manufa	IDE Debu	ug Mode	
×	CMSIS-DAP	0226000040214e450	LinkServer	ARM	Non-Sto	р	
-							
	un a stad Dark as (tick (untick t						
Su	pported Probes (tick/untick to MCUXpresso IDE LinkServer						î
Ē	P&E Micro probes	(inc. clasis brayprobes					
							~
Pr	obe search options						
Se	earch again						
	Remember my selection (for t	this Launch configuration)					
	2						
G	0			OK		ancel	

# Figure 36. Attached Probes: debug emulator selection

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

X workspace - evkmimxrt1020_hello_work	d/source/hello_world.c - MCUXpresso IDE —	×
	rch Project ConfigTools Run FreeRTOS Window Help (************************************	e 🔀
Pro ☆ Peri ☆ Re ☆ Sy ♡ □	Debug	▼ □
<ul> <li>&gt; Includes</li> <li>&gt; CMSIS</li> <li>&gt; Doard</li> <li>&gt; drivers</li> <li>&gt; source</li> <li>&gt; hello_world.c</li> <li>&gt; estartup</li> <li>&gt;</li></ul>	<pre># arm-none-eabi-gdb (8.0.50.20171128)  Welcome [[(gdb[0].proc[42000].threadGroup[i1].gdb[0].proc[42000].OSthread[1]) 55 */ 56*int main(void) 57 { 58 char ch; 59 60 /* Init board hardware. */ 61 BOARD_ConfigMPU();] 62 BOARD_InitPins(); 63 BOARD_BootClockRUN(); 64 BOARD_InitDebugConsole();</pre>	
<ul> <li>Commercial and the second secon</li></ul>	65 66 PRINTF("hello world.\r\n"); 67 68 while (1) 69 { 70 ch = GETCHAR(); 71 PUTCHAR(ch); 72 } Console ≅ ₽ Proble @ Memory ֎ Debug ♥ Instru ➡ Power 至SWO T	
< >	evkmimxrt1020_hello_world LinkServer Debug [C/C++ (NXP Semiconductors) MCU Application] evkmimxrt1020	

# Figure 37. Stop at main() when running debugging

4. Start the application by clicking the **Resume** button.

Project	Duns	Window
10 🔌		10 🔳 🕅

### Figure 38. Resume button

The hello\_world application is now running and a banner is displayed on the MCUX presso IDE console window. If this is not the case, check your terminal settings and connections.

🚰 COM99 - PuTTY	_ 🗆 🗵
hello world.	<b></b>
	-

Figure 39. Text display of the hello\_world demo

# 7 MCUXpresso Config Tools

MCUXpresso Config Tools can help configure the processor and generate initialization code for the on chip peripherals. The tools are able to modify any existing example project, or create a new configuration for the selected board or processor. The generated code is designed to be used with MCUXpresso SDK version 2.x.

Table 1 describes the tools included in the MCUXpresso Config Tools.

Config Tool	Description	Image
Pins tool	For configuration of pin routing and pin electrical properties.	
Clock tool	For system clock configuration	<b>I</b>
Peripherals tools	For configuration of other peripherals	<b>()</b>
TEE tool	Configures access policies for memory area and peripherals helping to protect and isolate sensitive parts of the application.	$\bigcirc$
Device Configuration tool	Configures Device Configuration Data (DCD) contained in the program image that the Boot ROM code interprets to setup various on-chip peripherals prior the program launch.	<b>*</b>

Table 1. MCUXpresso Config Tools

MCUXpresso Config Tools can be accessed in the following products:

- **Integrated** in the MCUXpresso IDE. Config tools are integrated with both compiler and debugger which makes it the easiest way to begin the development.
- Standalone version available for download from www.nxp.com/mcuxpresso. Recommended for customers using IAR Embedded Workbench, Keil MDK μVision, or Arm GCC.
- **Online version** available on mcuxpresso.nxp.com. Recommended to do a quick evaluation of the processor or use the tool without installation.

Each version of the product contains a specific *Quick Start Guide* document MCUXpresso IDE Config Tools installation folder that can help start your work.

# 8 MCUXpresso IDE New Project Wizard

MCUXpresso IDE features a new project wizard. The wizard provides functionality for the user to create new projects from the installed SDKs (and from pre-installed part support). It offers user the flexibility to select and change multiple builds. The wizard also includes a library and provides source code options. The source code is organized as software components, categorized as drivers, utilities, and middleware.

To use the wizard, start the MCUXpresso IDE. This is located in the **QuickStart Panel** at the bottom left of the MCUXpresso IDE window. Select **New project**, as shown in Figure 40.

🙂 Quickstart Panel 🕺 = Global Variables 🔅 = Variables 🐁 Breakpoints 🗄 Outline	
MCUXpresso IDE (Free Edition)	^
▼ Start here	
New project	
Import SDK example(s)	=
Import project(s) from file system	=
🐔 Build " []	
🖌 Clean " []	
🎋 Debug " []	
🎋 Terminate, Build and Debug " []	
Edit " project settings	
Quick Settings>>	-

#### Figure 40. MCUXpresso IDE Quickstart Panel

For more details and usage of new project wizard, see the *MCUXpresso\_IDE\_User\_Guide.pdf* in the MCUXpresso IDE installation folder.

# 9 How to determine COM port

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

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

#### How to add or remove boot header for XIP targets

There are two ports, one is Cortex-A core debug console and the other is for Cortex M4.

2. Windows: To determine the COM port open Device Manager in the Windows operating system. Click on the **Start** menu and type **Device Manager** in the search bar.

Control Panel (3)		
🛃 Device Manager		
View devices Device Manager		
Update device View and update your hardware's settings and driver		
Pictures (9)		
Companies.inc		
hut.inc		
PTPStillImageTables.inc		
VIDs_PIDs.TXT		
SCSI_CDB_RevCpyRslts.inc		
SCSI_CDB_SPC.inc		
hci_command_table.inc		
RNDIS_OID.inc		
CDCRequests.inc		
Files (1)		
ialog_settings.xml		
₽ See more results		
Device Manager × Shut down +		

### Figure 41. Device Manager

3. In the Device Manager, expand the **Ports** (**COM & LPT**) section to view the available ports. The COM port names will be different for all the NXP boards.

# 10 How to add or remove boot header for XIP targets

The MCUXpresso SDK for i.MX RT1020 provides flexspi\_nor\_debug and flexspi\_nor\_release targets for each example and/or demo which supports XIP (eXecute-In-Place). These two targets add XIP\_BOOT\_HEADER to the image by default. Because of this, ROM can boot and run this image directly on external flash.

#### Macros for the boot leader:

• The following three macros are added in flexspi\_nor targets to support XIP.

flexspi_nor		
XIP_EXTERNAL_FLASH	1: Exclude the code which changes the clock of FLEXSPI.	
	0: Make no changes.	
XIP_BOOT_HEADER_ENABLE	1: Add FLEXSPI configuration block, image vector table, boot data, and device configuration data (optional) to the image by default.	
	0: Add nothing to the image by default.	

### Table 2. Macros added in flexspi\_nor

Table continues on the next page ...

# Table 2. Macros added in flexspi\_nor (continued)

XIP_BOOT_HEADER_DCD_ENABLE	1: Add device configuration data to the image.	
	0: Do <b>NOT</b> add device configuration data to the image.	

• The following table shows the different effect on the built image with a different combination of these macros:

		XIP_BOOT_HEADER _DCD_ENABLE=1	XIP_BOOT_HEADER _DCD_ENABLE=0
XIP_EXTERNAL_FLA SH=1	XIP_BOOT_HEADER _ENABLE=1	<ul> <li>Can be programmed to qspiflash by IDE and can run after POR reset if qspiflash is the boot source.</li> <li>SDRAM will be initialized.</li> </ul>	<ul> <li>Can be programmed to qspiflash by IDE, and can run after POR reset if qspiflash is the boot source.</li> <li>SDRAM will <b>NOT</b> be initialized.</li> </ul>
	XIP_BOOT_HEADER _ENABLE=0	• CANNOT run after POR reset if it is programmed by IDE, even if qspiflash is the boot source.	
XIP_EXTERNAL_FLAS	H=0	This image     CANNOT     complete XIP     because when     this macro is set     to 1, it excludes     the code, which     changes the     clock for     FLEXSPI.	

# Table 3. Effects on built image with different<br/>macros

### Where to change the macros for each toolchain in MCUXpresso SDK?

Take hello\_world as an example:

- IAR:
- MDK:

#### How to add or remove boot header for XIP targets

🔣 Options for Target 'hello_world flexspi_nor_debug'	Х
Device   Target   Output   Listing   User C/C++ (AC6) Asm   Linker   Debug   Utilities	
Preprocessor Symbols	
Define: XIP_EXTERNAL_FLASH=1,XIP_BOOT_HEADER_ENABLE=1,XIP_BOOT_HEADERDCD_ENABLE=1	
Undefine:	
Language / Code Generation         □ Execute-only Code       Warnings: AC5-like Warnings ▼       Language C: c99	]

Figure 42. Options for target

### • ARMGCC:

Change the configuration in CMakeLists.txt.

```
SET(CMAKE_C_FLAGS_SDRAM_RELEASE "${CMAKE_C_FLAGS_SDRAM_RELEASE} -std=gnu99")
SET(CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG "${CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG} -DXIP_EXTERNAL_FLASH=1")
SET(CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG "${CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG} -DXIP_BOOT_HEADER_ENABLE=1")
SET(CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG "${CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG} -DXIP_BOOT_HEADER_DCD_ENABLE=1")
SET(CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG "${CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG} -DXIP_BOOT_HEADER_DCD_ENABLE=1")
SET(CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG "${CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG} -DXIP_BOOT_HEADER_DCD_ENABLE=1")
```

# Figure 43. Change configuration CMakeLists.txt

• MCUX:

ype filter text	Settings	(-) ▼
<ul> <li>Resource Builders</li> <li>C/C++ Build Build Variables Environment Logging MCU settings</li> <li>Settings Tool Chain Editc</li> <li>C/C++ General MCUXpresso Cont Project References Run/Debug Settin Task Tags</li> <li>Validation</li> </ul>	Configuration: Debug [Active]	Manage Configurations         Build Artifact       Binary Parsers       Error Parsers         Do not search system directories (-nostdinc)         Preprocess only (-E)         Defined symbols (-D) <ul> <li></li></ul>

Figure 44. Properties for evkbimxrt1020

#### How to Reach Us:

Home Page: nxp.com

Web Support: nxp.com/support Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFIRE, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2019-2020 NXP B.V.

Document Number MCUXSDKMIMXRT102XGSUG Revision 2, 19 June 2020



