Getting Started with MCUXpresso SDK for EVKB-IMXRT1050

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 the *MCUXpresso SDK Release Notes Supporting EVKB-IMXRT1050* (document MCUXSDKMIMXRT105XRN).

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	10
6	Run a demo using MCUXpresso IDE	16
7	How to determine COM port	
8	How to add or remove boot header for	
	XIP targets	24
9	Revision history	27



MCUXpresso SDK board support package folders

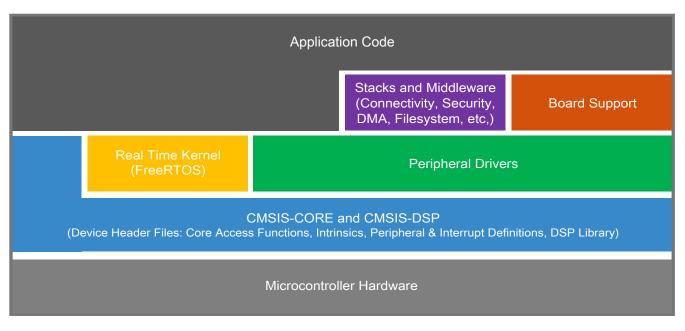


Figure 1. MCUXpresso SDK layers

2 MCUXpresso SDK board support package folders

MCUXpresso SDK board support package provides example applications for NXP development and evaluation boards for Arm[®] Cortex[®]-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):

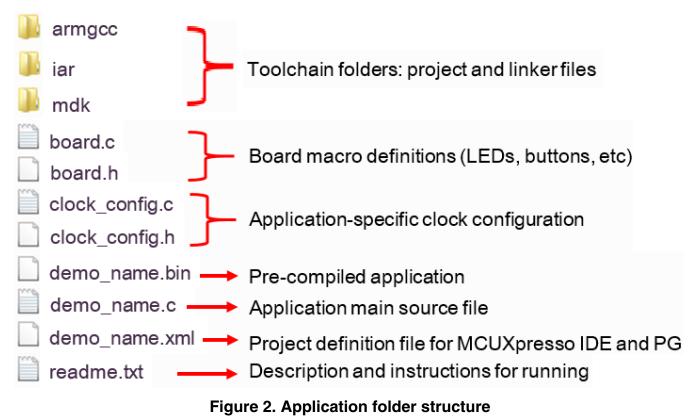
- demo_apps: Full-featured applications that highlight key functionality and use cases of the target MCU. These applications typically use multiple MCU peripherals and may leverage stacks and middleware.
- driver_examples: Simple applications that show how to use the MCUXpresso SDK's peripheral drivers 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 FreeRTOSTM OS examples that show the use of various RTOS objects (semaphores, queues, and so on) and interfaces with the MCUXpresso SDK's RTOS drivers

2.1 Example application structure

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

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:



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
- device_name>/<tool_name>: Toolchain-specific startup code, including vector table definitions
- devices/<device_name>/utilities: Items such as the debug console that are used by many of the example applications

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

3 Run a demo application using IAR

Run a demo application using IAR

This section describes the steps required to build, run, and debug example applications provided in the MCUXpresso SDK. The hello_world demo application targeted for the EVKB-IMXRT1050 hardware platform is used as an example, although these steps can be applied to any example application in the MCUXpresso SDK.

3.1 Build an example application

Do the following steps to build the hello_world example application.

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

<install_dir>/boards/<board_name>/<example_type>/<application_name>/iar

Using the IMXRT1050-EVKB hardware platform as an example, the hello_world workspace is located in:

><install_dir>/boards/evkbimxrt1050/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 the **hello_world – debug** target.

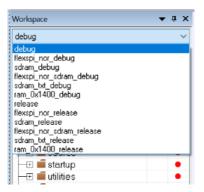


Figure 3. Demo build target selection

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

File Edit View	Project CMSIS	DAP Tools	Window H						_		
i 🗅 🗅 🔛 🕋 🛙	🗄 X 🗈 Ö	D C		Ŧ	< Q >	> 🖛 🔁 🤇	< 🙂 >	< >		0 💽 📜	
Workspace				▼ ‡ ×					_		
Debug				•							
Files			\$	•							
🗆 🌒 hello_world	l - Debug		· · · ·								
- ⊞ 🖬 board											
– ⊞ 🖬 doc – ⊞ 🗐 drivers											
🖃 🛋 startup											
🕂 🕀 🔳 utilities											
L-⊞ 🛑 Output											

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[®] 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. Connect the USB cable to J11 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	our PuTTY session
Logging	The second	2220
- Teminal	Specify the destination you w	
 Keyboard Bell Features Window Appearance Behaviour Translation 	Serial line COM16	Speed 115200
	Connection type:	ogin © <u>S</u> SH 💿 Serial
	Load, save or delete a stored Sav <u>e</u> d Sessions	d session
Selection	Debug	
Colours Connection	Default Settings Debug	Load
- Data - Proxy	Coolig	Sa <u>v</u> e
Telnet Rlogin		Delete
⊡ SSH ⊡ SSH Serial	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.
- 6. Run the code by clicking the **Go** button to start the application.



Figure 7. 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.

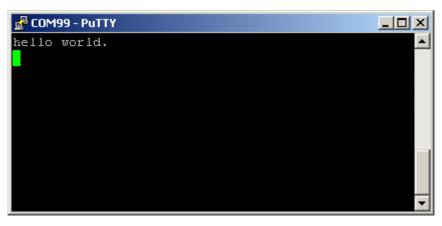


Figure 8. Text display of the hello_world demo

4 Run a demo using Keil[®] MDK/µVision

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

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 MIMXRT105x CMSIS pack.

- 1. Download the MIMXRT1051 and MIMXRT1052 packs.
- 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/evkbimxrt1050/demo_apps/hello_world/mdk/hello_world.uvmpw
2. To build the demo project, select Rebuild, highlighted in red.



Figure 9. 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® 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 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 board.h file)
 - b. No parity
 - c. 8 data bits
 - d. 1 stop bit

tegory:						
- Session	Basic options for yo	Basic options for your PuTTY session				
Logging	Specify the destination you w	ant to connect to				
Terminal	Serial li <u>n</u> e	Speed				
Keyboard Bell Features Window Appearance Behaviour Translation	COM16	115200				
	Connection type: Raw Delnet Re	ogin 💿 <u>S</u> SH 💿 Serjal				
	Load, save or delete a stored Saved Sessions	session				
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 on clean exit				

Figure 10. Terminal (PuTTY) configurations

4. To debug the application, click **load** (F8) if the flexspi_nor target is used. Then, click **Start/Stop Debug Session**, highlighted in red. If using **J-Link** as the debugger, click **Project option-->Debug -->Settings-->Debug-->Port**, acnd select **SW**.

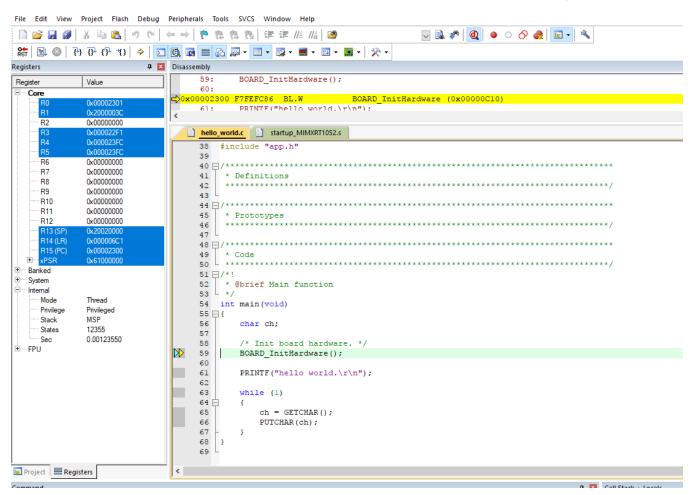


Figure 11. Stop at main() when run debugging

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

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



Figure 12. Text display of the hello_world demo

5 Run a demo using Arm[®] GCC

This section describes the steps to configure the command line Arm[®] GCC tools to build, run, and debug demo applications and necessary driver libraries provided in the MCUXpresso SDK. The hello_world demo application is targeted which is used as an example.

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 IMXRT1050-EVKB* (document MCUXSDKMIMXRT105XRN).

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 13. Set up MinGW and MSYS

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

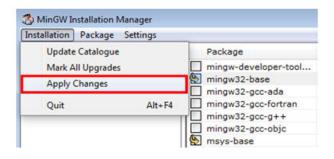


Figure 14. Complete MinGW and MSYS installation

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.

omputer Name	Hardware	Advanced	System Protection	on Remote
Environment \	/ariables			2
·····	·			
Edit System	m Variable	2		23
Variable n	ame:	Path		
Variable v	alue:	ogram Files	(x86)\CMake\bin	;C: (MinGW \bin
Variable v	alue:	ogram Files		
Variable v	alue:	ogram Files	(x86)\CMake\bin	;C: \MinGW \bin
Variable v		ogram Files		
	ables	ogram Files		
System varia	ables			Cancel
System varia	ables	/alue Vindows_NT		Cancel
System varia Variable OS	ables	/alue Vindows_NT ::\Program File	ОК	Cancel

Figure 15. 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:

C:\Program Files (x86)\GNU Tools ARM Embedded\6 2017-q2-update

Run a demo using Arm® GCC

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

nvironment Variable	25
New System Vari	able 🛛
Variable name:	ARMGCC_DIR
Variable value:	C:\Program Files (x86)\GNU Tools ARM Emb
	OK Cancel
System variables	
Variable	Value
Variable OS	Value ^
	value
OS	Windows_NT
OS Path	Value Windows_NT C:\Program Files (x86)\Parallels\Parallel .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;
OS Path PATHEXT	Value Windows_NT C:\Program Files (x86)\Parallels\Parallel .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;

Figure 16. 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.



Figure 17. 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 18. 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/evkbimxrt1050/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 shown in this figure:



Figure 19. 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)

Run a demo using Arm® GCC

- b. No parity
- c. 8 data bits
- d. 1 stop bit

egory:		
Session	Basic options for y	our PuTTY session
Logging Teminal	Specify the destination you	want to connect to
 Keyboard Bell Features Window Appearance Behaviour Translation 	Serial li <u>n</u> e	Speed
	COM16	115200
	Connection type:	Rogin () <u>S</u> SH () Serial
	Load, save or delete a store Sav <u>e</u> d Sessions	d session
Selection	Debug	
Colours Connection	Default Settings Debug	Load
Data	Debug	Sa <u>v</u> e
 Proxy Telnet Rlogin ⊕- SSH Serial 		Delete
	Close window on e <u>x</u> it: Always Never	 Only on clean exit

Figure 20. 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/ evkmimxrt1050/demo_apps/hello_world/evkmimxrt1050_sdram_init.jlinkscript command
- 5. The target device selection chosen for this example is Cortex-M7.
- 6. After it is connected, the screen should resemble this figure:

				Run a demo using Arm® GC0
				- ·· I
File He	elp			
GDB	Waiting for connection	l i i i i i i i i i i i i i i i i i i i		Stay on top
J-Link	Connected	SWD	4000 kHz	Show log window
Device	IMXRT1064xxx6A (Halted)	3.33V	little endian	Generate logfile
				Verify download
Clear L	log			
Target	endian:	little		^
Connec	ting to J-Link			
	is connected.			
	are: J-Link V10 compile are: V10.10	ed Sep 4 2018 11:24	:21	
	00101603			
-	e(s): RDI, FlashBP, F.	lashDL, JFlash, GDB		
	ng target voltage			
Target	; voltage: 3.33 V			
	ing on TCP/IP port 23			
	ting to targetConn	-		
Waitin	ng for GDB connection.	••		
				×
0 bytes do	ownloaded		Connected to	target

Figure 21. 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.

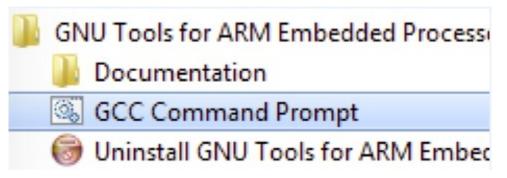


Figure 22. Launch command prompt

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/evkbimxrt1050/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.

Run a demo using MCUXpresso IDE

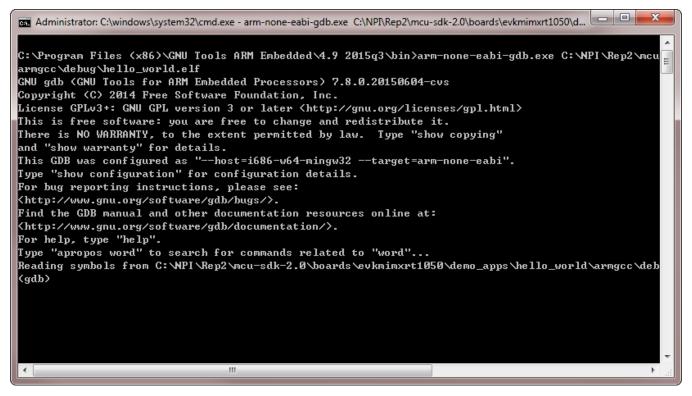


Figure 23. Run arm-none-eabi-gdb

10. Run these commands:

```
a. target remote localhost:2331
```

```
b. monitor reset
```

- c. 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 IMXRT1050-EVKB 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.

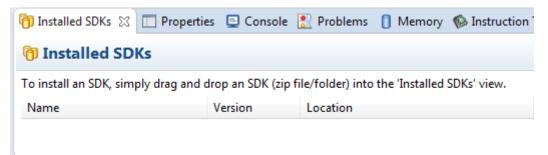


Figure 24. Install an SDK

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

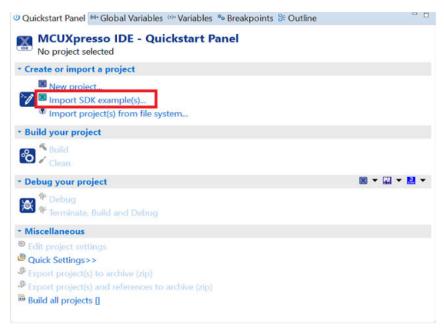


Figure 25. Import an SDK example

3. In the window that appears, expand the MIMXRT1050 folder and select MIMXRT1052xxxxx. Then, select evkbimxrt1050 and click the Next button.

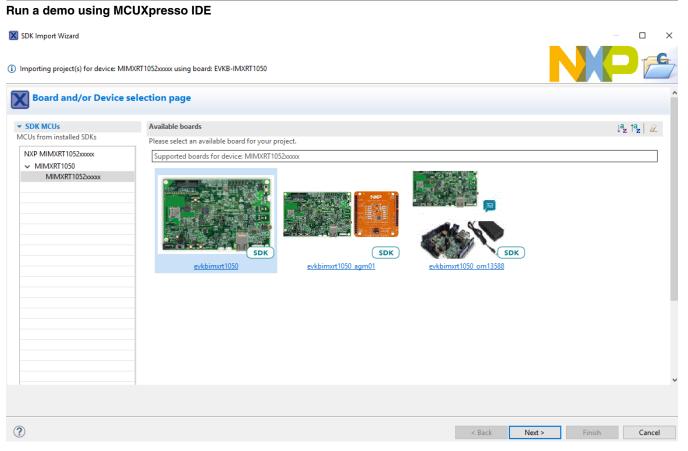


Figure 26. Selecting EVKB-IMXRT1050 board

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

			Run a dem	o using MCUXpresso IDE
🔀 SDK Import Wizard				- 🗆 X
A The source from the lift you want to use line	e SDK will be copied into the workspace. nked files, please unzip the 'SDK_2.x_EVKB-IMXRT1050' SDK.			
Import pro	jects			
Project name prefix:	evkbimxrt1050_	Reproject name suffix:		Ø.
Use default locat	ion			
Location: C:\Users\r	nxa12829\Documents\MCUXpressolDE_10.1.9_671_EB1\workspace\evkbimxrt1050_			Browse
Project Type		Projec	t Options	
● C Project ○ C	++ Project O C Static Library O C++ Static Library	Co	Debug Console () Semihost () UART upy sources port other files	
Examples				🔤 🖉 🗹 🙀 🕀 🖨
type to filter				
	ssi ile npass p_pwm		Version	* •
?			< Back Nex	t > Finish Cancel

Figure 27. 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

🔀 SDK Import Wizard



Advanc	ed Settings					
• C/C++ Libra	ary Settings					
	pe (and hosting variant)	Redlib (nohost-nf)	~			
Redlib: Use	e floating point version of printf			NewlibNano: Use floating point	version of printf	
	e character rather than string base	ed printf		NewlibNano: Use floating point		
Redirect SI	DK "PRINTF" to C library "printf"			Redirect printf/scanf to ITM		
·····	mihost HardFault handler			Redirect printf/scanf to UART		
 Hardware s 						
Set Floating P	oint type		FPv5-SP-D1	6 (HardABI)		\sim
 MCU C Con 	npiler					
l anguage star	ndard GNU C99 (-std=gnu99)					~
 MCU Linke 	r					
Link applic	tation to RAM					
 Memory Co 	onfiguration					
Memory deta	ils					
Туре	Name	Alias	Location	Size	Driver	
Flash	BOARD_FLASH	Flash	0x6000000	0x4000000	MIMXRT1050-EVK_S26KS512S.cfx	
RAM	SRAM_DTC	RAM	0x20000000	0x20000		
RAM	SRAM_ITC	RAM2	0x0	0x20000		Edit
RAM	SRAM_OC	RAM3	0x20200000	0x40000		
RAM	BOARD_SDRAM	RAM4	0x8000000	0x2000000		
)					< Back Next > Finish	Cancel
					I I IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	0

Figure 28. 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. Then, change the value of $SDK_DEBUGCONSOLE$ from 1 to 0 in **Properties**.

6. On the Quickstart panel, click build .

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:

1. On the Quickstart panel, click Debug evkbimxrt1050_demo_apps_hello_world [Debug].

U Quickst	(×)= Global	(x)= Variables	● _● Breakp	e Outline		
	UXpresso I act: evkbimxrt10					Î
▼ Create or	[,] import a proje	ect				
²-🌈 💌 I	New project Import SDK exa Import project(:	mple(s) s) from file syst	em			
▼ Build you	r project					
ം 🍃	Build Clean					
👻 Debug ya	our project		2	 • •	J. 👻	
× *	<mark>Debug</mark> Terminate, Build	d and Debug				
▼ Miscellan	eous					T
🛞 Edit proj	-					
🖉 Quick Se	ettings>>					4

Figure 29. 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.)

X Probes discovered

Connect to target: MIMXRT1064xxxxA

1 probe found. Select the probe to use:

Available attached probes

	Name	Serial number/ID	Туре	Manufa	IDE Debug Mode			
X	CMSIS-DAP	0229000005d9b16cl	LinkServe	ARM	Non-Stop			
Sur	oported Probes (tick/untick to en	able/disable)						
$\overline{\checkmark}$	MCUXpresso IDE LinkServer (ind							
\checkmark	✓ P&E Micro probes							
	SEGGER J-Link probes							
Dre								
Pro	be search options							
Se	Search again							
⊡ R	Remember my selection (for this Launch configuration)							
6	\ \							
()			OK	Cancel			
SDR	AM: 0 GB	32 MB 0.00	0/					

Figure 30. Attached Probes: debug emulator selection

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



Figure 31. Stop at main() when running debugging

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



Figure 32. Resume button

Getting Started with MCUXpresso SDK for EVKB-IMXRT1050, Rev. 1, 12/2019

×

How to determine COM port

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.

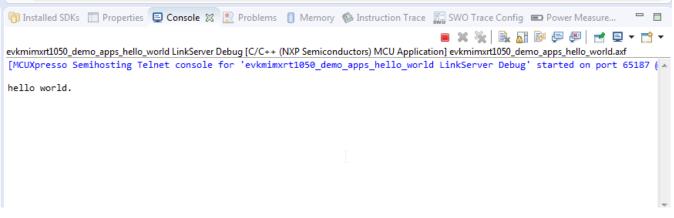


Figure 33. Text display of the hello_world demo

7 How to determine COM port

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

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

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

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

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

Control Panel (3)
🚔 Device Manager
View devices pad printer
Device Manager Update device View and update your hardware's settings and driver s
Pictures (9)
Companies.inc
hut.inc
PTPStillImageTables.inc
VIDs_PIDs.TXT
SCSI_CDB_RcvCpyRslts.inc
SCSI_CDB_SPC.inc
hci_command_table.inc
RNDIS_OID.inc
CDCRequests.inc
Files (1)
📄 dialog_settings.xml
₽ See more results
Device Manager × Shut down >
Device Manager X Shut uowh F

Figure 34. 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.

8 How to add or remove boot header for XIP targets

The MCUXpresso SDK for i.MX RT1050 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.

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.			
XIP_BOOT_HEADER_DCD_ENABLE	1: Add device configuration data to the image.			
	0: Do NOT add device configuration data to the image.			

Table 1. Macros added in flexspi_nor

How to add or remove boot header for XIP targets

• 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	 Can be programmed to hyperflash by IDE and can run after POR reset if hyperflash is the boot source. SDRAM will be initialized. 	 Can be programmed to hyperflash by IDE, and can run after POR reset if hyperflash is the boot source. SDRAM will NOT be initialized.
	XIP_BOOT_HEADER _ENABLE=0	• CANNOT run after POR reset if it is programmed by IDE, even if hyperflash is the boot source.	
XIP_EXTERNAL_FLAS	5H=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 2. Effects on built image with different macros

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

Take hello_world as an example:

• IAR:

How to add or remove boot header for XIP targets

Options for node "hello_world"	and an other the	And a state of the			24
Category:				ſ	Factory Settings
General Options	📃 Multi-file Con	npilation			
Static Analysis	📃 Discard U	Jnused Publics			
Runtime Checking	Diagnostics	MISRA-C:2004	MISRA-C:1998	Encodin	gs Extra Options
C/C++ Compiler	Language 1	anguage 2 Code	Optimizations	Output L	ist Preprocessor
Assembler					
Output Converter Custom Build	Ignore sta	ndard include dired	tories		
Build Actions	Additional inc	lude directories: (o	ne per line)		
Linker		\$/////CMSIS/Ir	clude		A
Debugger	\$PROJ_DIR \$PROJ_DIR	\$/////devices			Ξ
Simulator	\$PROJ DIR	· •			
CADI	\$PROJ_DIR	\$/////devices/	MIMXRT1052/driv	ers	-
CMSIS DAP	Preinclude fil	e.			
GDB Server					
I-jet/JTAGjet					
J-Link/J-Trace	Defined sym	bols: (one per line)			
TI Stellaris	DEBUG				output to file
PE micro		NAL_FLASH=1 HEADER_ENABLE	· . =	Preserve co	
ST-LINK		HEADER_DCD_EN		aenerate #I	ine directives
Third-Party Driver					

Figure 35. Options node IAR

• MDK:

V N	Options for Target 'hello_world flexspi_nor_debug'					
Dev	ice Target	Output Listing User	C/C++ Asm	Linker Debug Utilities		
	Preprocesso	or Symbols				
	Define:	XIP_EXTERNAL_FLASH	H=1, XIP_BOOT_	_HEADER_ENABLE=1, XIP_	_BOOT_HEADER_DCD_ENABLE=1	. CPI
	Undefine:					
	Language / (Code Generation				
	Execute-	only Code		Strict ANSI C	Warnings:	
	Optimization	: Level 0 (-00) 🔻		Enum Container always int	All Warnings	•

• ARMGCC:

Figure 36. Options for target

Change the configuration in CMakeLists.txt.

SET (CMAKE_C_FLAGS_SDRAM_RELEASE " <i>\${CMAKE_C_FLAGS_SDRAM_RELEASE}</i> -std=gnu99")
<pre>SET(CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG "\${CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG} -DXIP_EXTERNAL_FLASH=1")</pre>
<pre>SET(CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG "\${CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG} -DXIP_BOOT_HEADER_ENABLE=1")</pre>
SET(CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG " <i>\${CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG}</i> -DXIP_BOOT_HEADER_DCD_ENABLE=1")
<pre>SET(CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG "\${CMAKE_C_FLAGS_FLEXSPI_NOR_DEBUG} -DCPU_MIMXRT1052DVL6A")</pre>

Figure 37. Change configuration CMakeLists.txt

• MCUX:

Properties for evkbimxrt	1050_demo_apps_hello_world
type filter text	Settings
 > Resource Builders C/C++ Build Build Variables Environment Logging MCU settings Settings Tool Chain Editor > C/C++ General MCUXpresso Config T Project References Run/Debug Settings Task Tags > Validation 	Configuration: Debug [Active] Manage Co Tool Settings Build steps Build Artifact Binary Parsers Compiler Binary Parsers Compiler Do not search system directories (-nosted) Dialect Do not search system directories (-nosted) Preprocessor Dialect Preprocessor Compiler Compiler Compiler Do not search system directories (-nosted) Defined symbols (-D) Compiler Com

Figure 38. Properties for evkbimxrt1050

9 Revision history

This table summarizes revisions to this document.

Table 3. Revision history

Revision number	Date	Substantive changes	
0	02/2018	Initial release	
1	06/2019	Updates for MCUXpresso SDK v2.6.0	

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, C-5, CodeTEST, 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 NXP B.V.

Document Number MCUXSDKMIMXRT105XGSUG Revision 1, 12/2019



