### Getting Started with MCUXpresso SDK

Rev. 2.15.000 — 10 January 2024

User guide

#### **Document information**

Information	Content
Keywords	MCUXSDKGSUG, Getting Started, MCUXpresso SDK, MCUXSDK
Abstract	This document describes the steps to get started with MCUXpresso SDK.



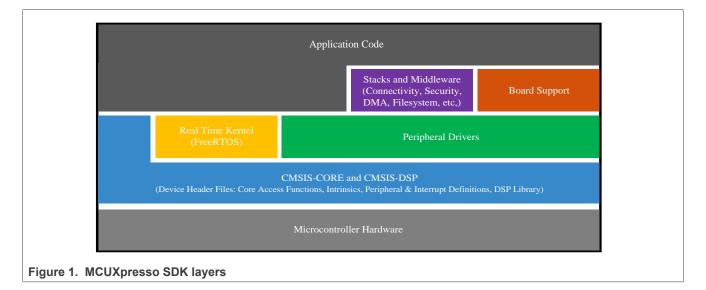
Getting Started with MCUXpresso SDK

### 1 Overview

The NXP MCUXpresso software and tools offer comprehensive development solutions designed to optimize, ease, and help accelerate embedded system development of applications based on general purpose, crossover, and Bluetooth-enabled MCUs from NXP. 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 optional RTOS integrations such as FreeRTOS and Azure RTOS, and various other middleware to support rapid development.

For supported toolchain versions, see MCUXpresso SDK Release Notes (document MCUXSDKRN).

For more details about MCUXpresso SDK, see MCUXpresso Software Development Kit (SDK).



### 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 <br/><br/>d\_name> folder, there are various subfolders to classify the type of examples it contains. These include (but are not limited to):

- cmsis driver examples: Simple applications intended to show how to use CMSIS drivers.
- 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).
- emwin\_examples: Applications that use the emWin GUI widgets.
- rtos\_examples: Basic FreeRTOSOS examples that show the use of various RTOS objects (semaphores, queues, and so on) and interfaces with the MCUXpresso SDK's RTOS drivers
- usb examples: Applications that use the USB host/device/OTG stack.

#### Getting Started with MCUXpresso SDK

### 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	
📓 clock_config.c	Application-specific clock configuration
📓 clock_config.h	Application-specific clock configuration
📄 hello_world.bin 🛛 🔶	Pre-compiled application
📔 hello_world.c 🛛 🛶	Application main source file
🔀 hello_world.mex>	Application-specific MCUXpresso Config Tool configuration
hello_world.xml>	Project definition file for MCUXpresso IDE and PG
📔 pin_mux.c	Application-specific pin configuration
📔 pin_mux.h 📃	Application opcome 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, various source files are referenced. The MCUXpresso SDK devices folder is the central component to all example applications. It means that 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>/cmsis drivers: All the CMSIS drivers for your specific MCU
- 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

#### Getting Started with MCUXpresso SDK

• devices/<devices name>/project: Project template used in CMSIS PACK new project creation

For examples containing middleware/stacks or an RTOS, there are references to the appropriate source code. Middleware source files are located in the middleware folder and 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 using MCUXpresso IDE

**Note:** Ensure that the MCUXpresso IDE toolchain is included when generating the MCUXpresso SDK package.

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 hardware platform is used as an example, though these steps can be applied to any example application in the MCUXpresso SDK.

#### 3.1 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 the MCUXpresso SDK tree.

#### 3.2 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 an SDK. In the window that appears, click **OK** and wait until the import has finished.

	🎁 Installed SDKs 🛛	Properties	📃 Console	🖹 Problems	🚺 Memory	🚯 Instruction -
	🕅 Installed SD	Ks				
	To install an SDK, sim	ply drag and dro	op an SDK (zip	file/folder) into	the 'Installed	SDKs' view.
	Name	١	/ersion	Location		
Figure 3. Install an SD	K					

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

### **NXP Semiconductors**

# MCUXSDKGSUG

### Getting Started with MCUXpresso SDK

O Quickstart Panel       № Global Variables № Variables № Breakpoints № Outline       0         Import Spice       No project selected       0         Import Spice       Import Spice       0         Import Spice       0       0         Import Spice			
<ul> <li>Create or import selected</li> <li>Create or import spoject</li> <li>New project</li> <li>Build your project</li> <li>Build Clean</li> <li>Debug your project</li> <li>Debug</li> <li>Terminate, Build and Debug</li> <li>Miscellaneous</li> <li>Edit project settings</li> <li>Quick Settings&gt;</li> <li>Eport project(s) and references to archive (zip)</li> </ul>		🙂 Quickstart Panel 🔤 Global Variables 🔅 Variables 🗣 Breakpoints 😂 Outline	
New project.         Import SDK example(s)         Import project(s) from file system         • Build your project         So         • Debug your project         So         • Debug         • Terminate, Build and Debug         • Miscellaneous         © Edit project settings         © Quick Settings>         © Export project(s) tan drive (zip)         © Export project(s) tan drive (zip)		MCUXpresso IDE - Quickstart Panel No project selected	
<ul> <li>Import SDK example(s)</li> <li>Import project(s) from file system</li> <li>Build your project</li> <li>Build Clean</li> <li>Debug your project</li> <li>P Debug</li> <li>Terminate, Build and Debug</li> <li>Miscellaneous</li> <li>Edit project settings</li> <li>Quick Settings&gt;</li> <li>Export project(s) and references to archive (zip)</li> </ul>		- Create or import a project	
Build Clean         > Debug your project         > Debug %         > Debug %         > Miscellaneous         > Edit project settings         > Dickic Settings>         > Eport project(s) to archive (zip)         > Export project(s) and references to archive (zip)		🚧 💹 Import SDK example(s)	
Debug your project     M ▼		* Build your project	
Conservation     C		S Build	
<ul> <li>Miscellaneous</li> <li>Edit project settings</li> <li>Quick Settings&gt;&gt;</li> <li>Export project(s) to archive (zip)</li> <li>Export project(s) and references to archive (zip)</li> </ul>			🗙 🔻 🔛 🗶 🗶
<ul> <li>Miscellaneous</li> <li>⊕ Edit project settings</li> <li>@ Quick Settings&gt;&gt;</li> <li></li></ul>		Debug * Terminate, Build and Debug	
<ul> <li>Quick Settings&gt;&gt;</li> <li>Export project(s) to archive (zip)</li> <li>Export project(s) and references to archive (zip)</li> </ul>			
		Quick Settings>>  Export project(s) to archive (zip)  Export project(s) and references to archive (zip)	
	Figure 4 Import on CDK out	omnlo	
	Figure 4. Import an SDK exa	ampie	

- 3. Expand the demo apps folder and select hello world.
- 4. Click Next.

MCUXSDKGSUG

Getting Started with MCUXpresso SDK

The source from the SDK will be copied into the workspace.				
If you want to use linked files, please unzip the 'SDK_2.x_FRDM-K64F' SDK.				
Import projects				
Project name prefix: frdmk64f_	ect name suffix:			a
✓ Use default location	L			
	orkspace\frdmk64	f_		B <u>r</u> owse.
Project Type		Project Optio	ns	
O Project ○ C++ Project ○ C Static Library ○ C++ Static Library		Copy sour	ces	
Examples			2	🍇 🕀 E
Name		V	ersion	
Image: Second			131011	
🔺 🔲 🗧 demo_apps				
▷ 🔲 🗧 Iwip				
▷				
▷ III III IIIIIIIIIIIIIIIIIIIIIIIIIIII				
🔲 🗮 adc16_low_power				
bubble				
☐				=
☐				
🔲 🚆 ftm_guad_decoder				
Image: Provide the second s				
power_manager				
☐				
🕅 🗏 chell				
〒 chell 				

5. Ensure Redlib: Use floating-point version of printf is selected if the example prints floating-point numbers on the terminal for demo applications such as adc\_basic, adc\_burst, adc\_dma, and adc\_interrupt. Otherwise, it is not necessary to select this option. Then, click Finish.

### 3.3 Run an example application

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

To download and run the application, perform the following steps:

MCUXSDKGSUG

#### Getting Started with MCUXpresso SDK

- 1. See the table in <u>Section 11</u> to determine the debug interface that comes loaded on your specific hardware platform. For LPCXpresso boards, install the DFU jumper for the debug probe, then connect the debug probe USB connector.
  - For boards with CMSIS-DAP/mbed/DAPLink interfaces, visit <u>developer.mbed.org/handbook/Windows-</u> serial-configuration and follow the instructions to install the Windowsoperating system serial driver. If running on Linux OS, this step is not required.
  - For boards with a P&E Micro interface, see <u>PE micro</u> to download and install the P&E Micro Hardware Interface Drivers package.
  - For the MRB-KW01 board, see <u>www.nxp.com/USB2SER</u> to download the serial driver. This board does not support the OpenSDA. Therefore, an external debug probe (such as a J-Link) is required. The steps below referencing the OpenSDA do not apply because there is only a single USB connector for the serial output.
  - If using J-Link with either a standalone debug pod or OpenSDA, install the J-Link software (drivers and utilities) from <u>www.segger.com/jlink-software.html</u>.
  - For boards with the OSJTAG interface, install the driver from <u>www.keil.com/download/docs/408</u>.
- 2. Connect the development platform to your PC via a 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 <u>Section 9</u>). Configure the terminal with these settings:
  - a. 115200 or 9600 baud rate, depending on your board (reference <code>BOARD\_DEBUG\_UART\_BAUDRATE variable in board.h file)</code>
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit

Getting Started with MCUXpresso SDK

ategory:		
Session	Basic options for your PuTTY ses	
Terminal	Specify the destination you want to connect	to
- Keyboard	Serial line	Speed
- Bell - Features	COM4	115200
Window Appearance	Connection type: Raw Telnet Rlogin SSH	<ul> <li>Serial</li> </ul>
- Behaviour - Translation - Selection - Colours	Load, save or delete a stored session Saved Sessions	
Connection Data	Default Settings	Load
Proxy Telnet Rlogin		Save
Elogin		Delete
	Close window on exit: Always Never Only on clear	ean exit
About	Open	Cancel

- 4. On the **Quickstart Panel**, click **Debug** to launch the debug session.
- 5. 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 **OK**. (For any future debug sessions, the stored probe selection is automatically used, unless the probe cannot be found.)

Getting Started with MCUXpresso SDK

Name       Serial number/ID       Type       Manu       IDE Debug Mod         USB1 - OpenSDA (7A790E49       7A790E49       USB1       P&E Mi       All-Stop         IDE Debug Mod       IDE Debug Mod       IDE Debug Mod       All-Stop         IDE Debug Mod       IDE Debug Mod       IDE Debug Mod         IDE Debug Mod       IDE Debug Mod       All-Stop         IDE Debug Mod       IDE Debug Mod       IDE Debug Mod         IDE Debug Mod       IDE Debug Mod       All-Stop         IDE Debug Mod       IDE Debug Mod       IDE Debug Mod         IDE Debug Mod       IDE Debug Mod       IDE Debug Mod         IDE Debug Mod       IDE Debug Mod       IDE Debug Mod         IDE Debug Mod       IDE Debug Mod       IDE Debug Mod         IDE Debug Mod       IDE Debug Mod       IDE Debug Mod         IDE Debug Mod       IDE Debug Mod       IDE Debug Mod         IDE Debug Mod       IDE Debug Mod       IDE Debug Mod         IDE Debug Mod       IDE Debug Mod       IDE Debug Mod         IDE Debug Mod       IDE Debug Mod       IDE Debug Mod         Pottor       IDE Debug Mod       IDE Debug Mod         MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes       IDE Debug Mod         Pde Search op	vailable attached prol	bes			
ported Probes (tick/untick to enable/disable) MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes P&E Micro probes SEGGER J-Link probes be search options	Name	Serial number/ID	Туре	Manu	IDE Debug Mode
MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes P&E Micro probes SEGGER J-Link probes be search options	USB1 - OpenSDA (7A790E49	7A790E49	USB1	P&E Mi	All-Stop
MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes P&E Micro probes SEGGER J-Link probes be search options					
MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes P&E Micro probes SEGGER J-Link probes be search options					
MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes P&E Micro probes SEGGER J-Link probes be search options					
MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes P&E Micro probes SEGGER J-Link probes be search options					
MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes P&E Micro probes SEGGER J-Link probes be search options					
MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes P&E Micro probes SEGGER J-Link probes be search options					
MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes P&E Micro probes SEGGER J-Link probes be search options	upported Probes (tick/untick to	enable/disable)			
P&E Micro probes SEGGER J-Link probes be search options			robes		
SEGGER J-Link probes		(			
be search options	SEGGER J-Link probes				
	•				
arch again					
	Search again				
member my selection (for this Launch configuration)					

- 6. The application is downloaded to the target and automatically runs to main().
- 7. Start the application by clicking **Resume**.

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

#### Getting Started with MCUXpresso SDK



### 3.4 Build a multicore example application

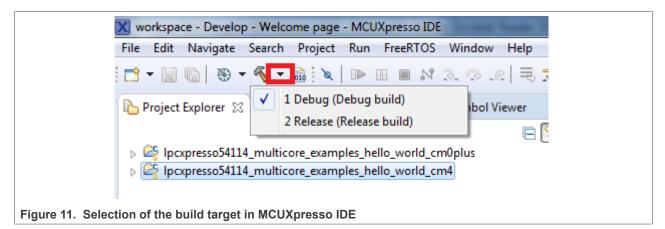
This section describes the steps required to configure MCUXpresso IDE to build, run, and debug multicore example applications. The following steps can be applied to any multicore example application in the MCUXpresso SDK. Here, the dual-core version of hello\_world example application targeted for the LPCXpresso54114 hardware platform is used as an example.

- Multicore examples are imported into the workspace in a similar way as single core applications, explained in <u>Section 3.2</u>. When the SDK zip package for LPCXpresso54114 is installed and available in the **Installed** SDKs view, click **Import SDK example(s)...** on the Quickstart Panel. In the window that appears, expand the LPCxx folder and select LPC54114J256. Then, select lpcxpresso54114 and click Next.
- 2. Expand the multicore\_examples/hello\_world folder and select cm4. The cm0plus counterpart project is automatically imported with the cm4 project, because the multicore examples are linked together and there is no need to select it explicitly. Click Finish.

	SDK Import Wizard The source from the SDK will be copied into the workspace. If you want to use linked files, please unsip the "SDK_Z_FRDM-K32	L3A6' SDK. The advanced options page is disabled when eithe	more than one	
	🔀 Import projects			
	Project name prefix frdmk32l3a6	2 Project name suffic	<i>Q</i>	
	✓ Use default location			
	Location: C:\Users\nxa12829\Documents\MCUXpressolDE_11.0.1_2	530_alpha\workspace\frdmk32l3a6	Browse	
	Project Type	Project Options		
	○ C Project ○ C++ Project ○ C Static Library ○ C++ Static Libra	ry SDK Debug Console ○ Semihost ○	UART	
	Examples		画 🖉 🖗 🕀 🖻	
	type to filter			
	Name       >     □ ≣ fatfs_examples       >     □ ≣ mbedtls_examples	Description	Version	
	erpc_matrix_multiply_mu_rtos_cm0plus     erpc_matrix_multiply_mu_rtos_cm0plus     erpc_matrix_multiply_pmsg_cm0plus     erpc_matrix_multiply_pmsg_cm0+1 Linked to: erpc_matrix     erpc_matrix_multiply_pmsg_rtos_cm0plus	The Multicore eRPC Matrix Multiply project is a simple demo The Multicore eRPC Matrix Multiply project is a simple demo The Multicore eRPC Matrix Multiply MDS project is a simple The Multicore eRPC Matrix Multiply MDS project is a simple the Multicore eRPC Matrix Multiply project is a simple demo The Multicore eRPC Matrix Multiply project is a simple demo The Multicore eRPC Matrix Multiply project is a simple The Multicore eRPC Matrix Multiply MDS project is a simple The Multicore eRPC Matrix Multiply MDS project is a simple The Multicore Hello World demo application demonstrates in The Multicore Manager example application demonstrates and The Mu	Instation program. Mat demonstration program stration program. Mat stration program. Mat demonstration program demonstration program own to set up projects f own to set up projects f	
	0	< Back Next >	Finish Cancel	
Figure 10. Select the hell	o_world multicore exar	nple		

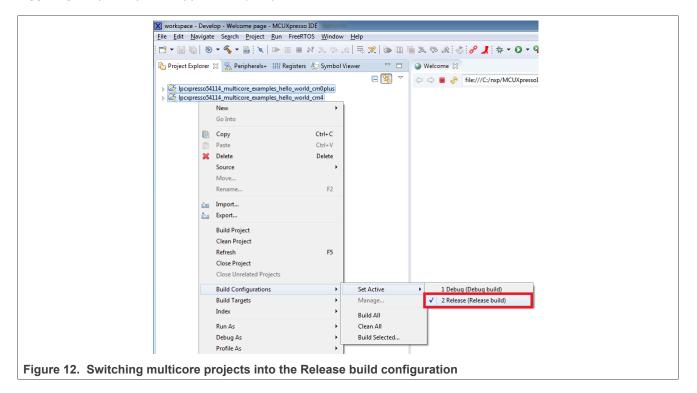
Getting Started with MCUXpresso SDK

3. Now, two projects should be imported into the workspace. To start building the multicore application, highlight the lpcxpresso54114\_multicore\_examples\_hello\_world\_cm4 project (multicore master project) in the Project Explorer. Then choose the appropriate build target, **Debug** or **Release**, by clicking the downward facing arrow next to the hammer icon, as shown in <a href="Figure 11">Figure 11</a>. For this example, select **Debug**.



The project starts building after the build target is selected. Because of the project reference settings in multicore projects, triggering the build of the primary core application (cm4) also causes the referenced auxiliary core application (cm0plus) to build.

**Note:** When the **Release** build is requested, it is necessary to change the build configuration of both the primary and auxiliary core application projects first. To do this, select both projects in the Project Explorer view and then right click which displays the context-sensitive menu. Select **Build Configurations** -> **Set Active** -> **Release**. This alternate navigation using the menu item is **Project** -> **Build Configuration** -> **Set Active** -> **Release**. After switching to the **Release** build configuration, the build of the multicore example can be started by triggering the primary core application (cm4) build.



#### Getting Started with MCUXpresso SDK

#### 3.5 Run a multicore example application

The primary core debugger handles flashing of both the primary and the auxiliary core applications into the SoC flash memory. To download and run the multicore application, switch to the primary core application project and perform all steps as described in <u>Section 3.3</u>. These steps are common for both single-core applications and the primary side of dual-core applications, ensuring both sides of the multicore application are properly loaded and started. However, there is one additional dialogue that is specific to multicore examples which requires selecting the target core. See the following figures as reference.

© Quickstart Panel III International Project: frdmk32l3a6_hello_world_cm4 [Released in the section of the secti		V or structures of or take or yes reasons of the please consult the supplied MCUXpret     I Help -> MCUXpress IDE User Gu     Further product documentation is provi     II Help -> Help Contents
<ul> <li>Create or import a project</li> <li>New project</li> <li>Import SDK example(s)</li> <li>Import project(s) from file system</li> </ul>		Installed SDKs     Properties     Problems     C       CDT Build Console (frdmk32l3a6_hello_world_cm)
- Build your project Build Clean - Debug your project	œ • ≅ • № •	<pre>makeno-print-directory post-buil Performing post-build steps arm-none-eabi-size "frdmk3213a6_hel text data bss dec 7000 0 8488 15488</pre>
Construction of the second sec	Attach to a runn Program flash ad	kServer probes (CTRL+SHIFT+ALT+L) ing target using LinkServer (CTRL+ALT+L) ction using LinkServer n using LinkServer
3. Debug "frdmk32l3a6 hello world cm4"	case	

MCUXSDKGSUG

Getting Started with MCUXpresso SDK

C-LINK2 CMSIS-DAP V5.134 A00000002 LinkServe NXP Semi Non-Stop C-LINK2 CMSIS-DAP V5.134 A00000002 LinkServe NXP Semi Non-Stop UXPresso IDE LinkServer (inc. CMSIS-DAP) probes E Micro probes iGER J-Link probes earch options					
ted Probes (tick/untick to enable/disable) UXpresso IDE LinkServer (inc. CMSIS-DAP) probes E Micro probes iGER J-Link probes	PC-LINK2 CMSIS-DAP V5.134 A000	000002	LinkServe	NXP Semi	Non-Stop
UXpresso IDE LinkServer (inc. CMSIS-DAP) probes E Micro probes iGER J-Link probes earch options					
UXpresso IDE LinkServer (inc. CMSIS-DAP) probes E Micro probes iGER J-Link probes earch options					
UXpresso IDE LinkServer (inc. CMSIS-DAP) probes E Micro probes iGER J-Link probes earch options					
UXpresso IDE LinkServer (inc. CMSIS-DAP) probes E Micro probes iGER J-Link probes earch options					
UXpresso IDE LinkServer (inc. CMSIS-DAP) probes E Micro probes iGER J-Link probes earch options					
UXpresso IDE LinkServer (inc. CMSIS-DAP) probes E Micro probes iGER J-Link probes earch options					
UXpresso IDE LinkServer (inc. CMSIS-DAP) probes E Micro probes iGER J-Link probes earch options					
UXpresso IDE LinkServer (inc. CMSIS-DAP) probes E Micro probes iGER J-Link probes earch options					
UXpresso IDE LinkServer (inc. CMSIS-DAP) probes E Micro probes iGER J-Link probes earch options					
UXpresso IDE LinkServer (inc. CMSIS-DAP) probes E Micro probes iGER J-Link probes earch options					
UXpresso IDE LinkServer (inc. CMSIS-DAP) probes E Micro probes iGER J-Link probes earch options					
E Micro probes GER J-Link probes earch options	rted Probes (tick/untick to enable/disat	le)			
E Micro probes GER J-Link probes earch options	CUV presso IDE LinkSenver (inc. CMSIS-F	AD) probes			
iGER J-Link probes	•	AP) probes			
earch options	&E Micro probes				
earch options					
	GGER L Link prober				
	GGER J-Link probes				
	GGER J-Link probes				
again	•				
	search options				
again	search options				
dudiii	•				
dudiii					
again	search options				
	earch options				
	search options				
mber my selection (for this Launch configuration)	search options h again	figuration			

	X			_	X			
	🔔 2 avai	nfiguration lable SWD Devices dete t 'Cortex-M4' has been		ay be incompatible!				
	Device		TAP Id	Details				
	0	Cortex-M4 Cortex-M0+	0x2ba01477 0x2ba01477	APID:24770011 APID:24770011				
	?			ОК	Cancel			
ure 15. Target core selection of	lialog	lne				1		

Getting Started with MCUXpresso SDK

<u>F</u> ile	<u>E</u> dit	Source	Refac <u>t</u> or <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un FreeRTOS <u>W</u> indow <u>H</u> elp
<b>1</b>	- 8	8	▼ 🗞 ▼ 🗟 ! 🍬   📭 🗉 🔳 🖄 🎿 👁 🗽   🗮 🛒 🎼 🏛 🖷 🎿 😎 📌 🤣 🖋 🖌 ! 🎋 ▼ 🔘 ▼ 💁 ▼
8		ebug 🔀	
6	-		esso54114_multicore_examples_hello_world_cm4 Debug [C/C++ (NXP Semiconductors) MCU Application]
묾			xpresso54114_multicore_examples_hello_world_cm4.axf [LPC54114J256 (cortex-m0plus)]
1010		4 d <sup>99</sup>	Thread #11 (Stopped) (Suspended : Breakpoint)
*			main() at hello_world_core0.c:85 0x98a
		N dir	n-none-eabi-gdb (7.12.0.20161204)
8			
ö	🕞 h	ello worl	d_core0.c 🔀
(×)=	_	B {	
	69	· ·	int32 t core1 image size;
(x)=	70	0 <b>#if d</b>	lefined(CC_ARM)
•	73		<pre>core1_image_size = (uint32_t)&amp;Image\$\$CORE1_REGION\$\$Length;</pre>
			defined(ICCARM) ma section = " sec core"
	74		core1 image size = (uint32 t) section end(" sec core") - (uint32 t)&core1 image start;
	75		
	70		eturn core1_image_size;
	71	7	f
		9⊜ /*!	•
	80		orief Main function
	81		main(void)
		3 {	
	84		<pre>/* Define the init structure for the switches*/</pre>
	♦ 85		<pre>pio_pin_config_t sw_config = {kGPIO_DigitalInput, 0};</pre>
	80		/* Init board hardware.*/
	88		/* attach 12 MHz clock to FLEXCOMMØ (debug console) */
	89	9 (	:LOCK_AttachClk( <i>kFR012M_to_FLEXCOMM0</i> );
	90		
	91 92		0ARD_InitPins_Core0(); 0ARD_BootClockFROHF48M();
	93		BOARD_InitDebugConsole();
	94	4	
	95		(* Init switches */
	90		<pre>iPIO_PinInit(BOARD_SW1_GPIO, BOARD_SW1_GPIO_PORT, BOARD_SW1_GPIO_PIN, &amp;sw_config); iPIO PinInit(BOARD SW2 GPIO, BOARD SW2 GPIO_PORT, BOARD_SW2 GPIO_PIN, &amp;sw_config);</pre>
	93		" 10, 111112(00, m0_3m2_0, 10, 00, m0_3m2_0, 10, 00, 10, 5m2_0, 10, 11, 03m2000, 18/)
		Cton	the primary core application at main() when running debugging

After clicking the "Resume All Debug sessions" button, the hello\_world multicore application runs and a banner is displayed on the terminal. If this is not the case, check your terminal settings and connections.

	COM25:115200baud-Tera Term VT File Edit Setup Control Window KanjiCode Help Hello World from the Primary Core! Starting Secondary core. The secondary core application has been started.	
Figure 17. Hello Wo	orld from the primary core message	
MCUXSDKGSUG	All information provided in this document is subject to legal disclaimers.	© 2024 NXP B.V. All rights reserved.

Getting Started with MCUXpresso SDK

An LED controlled by the auxiliary core starts flashing, indicating that the auxiliary core has been released from the reset and running correctly. It is also possible to debug both sides of the multicore application in parallel. After creating the debug session for the primary core, perform same steps also for the auxiliary core application. Highlight the lpcxpresso54114\_multicore\_examples\_hello\_world\_cm0plus project (multicore slave project) in the Project Explorer. On the Quickstart Panel, click "Debug 'lpcxpresso54114\_multicore\_examples\_hello\_world cm0plus' [Debug]" to launch the second debug session.

😃 Quickstart Pa	(x)= Global Varia (x)= Variables 💁 Breakp	oints 🗄 Outline	- 8	Fu
	resso IDE - Quickstart Panel cxpresso54114_hello_world_cm0plus [Debu	g]	^	ণী Installed SDKs 🔲 Properties 🛛 📮 Consol Property
▼ Create or imp	ort a project			
	roject SDK example(s) project(s) from file system			
- Build your pro	oject			
Suild 🗸 🔨 Build				
- Debug your p	roject	🗙 🔻 🔛 🗶		
Termin	ate, Build and Debug	X Attach to Program	a running	erver probes (CTRL+SHIFT+L) ) target using LinkServer (CTRL+ALT+L) g LinkServer nkServer
Figure 18. Det	oug "lpcxpresso54114_multicore_exa			

Getting Started with MCUXpresso SDK

x workspace - Develop - Ipcxpresso54114_multicore_examples_hello_world_cm4/source/hello_world_core0.c - MCUXpresso IDE
! 🗂 ▾ 🔚 🐚   🏵 ▾ 🗞 ▾ 🛗   🔌   🕨 == 🛤 🎿 👁 .rk   ≂, 🌫   🕪 🖬 🖷 🎿 👁 .rk   & 🏷 / 🖌 ! ♦ ▼ 🔿 ▼
🔓 🏘 Debug 🛛
<ul> <li>Ipcxpresso54114_multicore_examples_hello_world_cm4 Debug [C/C++ (NXP Semiconductors) MCU Application]</li> <li>Ipcxpresso54114_multicore_examples_hello_world_cm4.axf [LPC54114J256 (cortex-m0plus)]</li> <li>Ipcxpresso54114_multicore_examples_hello_world_cm0plus Debug [C/C++ (NXP Semiconductors) MCU Application]</li> <li>Ipcxpresso54114_multicore_examples_hello_world_cm0plus Debug [C/C++ (NXP Semiconductors) MCU Application]</li> <li>Ipcxpresso54114_multicore_examples_hello_world_cm0plus Debug [C/C++ (NXP Semiconductors) MCU Application]</li> <li>Ipcxpresso54114_multicore_examples_hello_world_cm0plus.axf [LPC54114J256 (cortex-m0plus)]</li> </ul>
●●
<pre>     hello_world_core0.c      for the switches*/</pre>
<pre>\$ 85 gpio_pin_config_t sw_config = {kGPIO_DigitalInput, 0}; 86 87 /* Init board hardware.*/ 88 /* attach 12 MHz clock to FLEXCOMM0 (debug console) */ 89 CLOCK_AttachClk(kFR012M_to_FLEXCOMM0); 90 91 BOARD_InitPins_Core0(); 92 BOARD_BootClockFR0HF48M(); 93 BOARD_InitDebugConsole(); 94 95 /* Init_switches */ 96 GPIO_PinInit(BOARD_SW1_GPIO, BOARD_SW1_GPIO_PORT, BOARD_SW1_GPIO_PIN, &amp;sw_config); 97 GPIO_PinInit(BOARD_SW2_GPIO, BOARD_SW2_GPIO_PORT, BOARD_SW2_GPIO_PIN, &amp;sw_config); 97 Figure 19. Two opened debug sessions</pre>

Now, the two debug sessions should be opened, and the debug controls can be used for both debug sessions depending on the debug session selection. Keep the primary core debug session selected by clicking the "Resume" button. The hello\_world multicore application then starts running. The primary core application starts the auxiliary core application during runtime, and the auxiliary core application stops at the beginning of the

MCUXSDKGSUG

Getting Started with MCUXpresso SDK

main() function. The debug session of the auxiliary core application is highlighted. After clicking the "Resume" button, it is applied to the auxiliary core debug session. Therefore, the auxiliary core application continues its execution.

X wo	orkspace - Develop - Ip	ocxpresso54114_multic	ore_examples_hel	lo_world_cm0	plus/source	e/hello_wo	orld_core1.c - MCUX	(presso IDE
<u>F</u> ile	<u>E</u> dit <u>S</u> ource Refac	<u>t</u> or <u>N</u> avigate Se <u>a</u> rc	n <u>P</u> roject <u>R</u> un	FreeRTOS	<u>W</u> indow	<u>H</u> elp		
: 📬	- 🛛 🕼   🛞 - 🍕	• 🗟 🔌 🕨 🗉	- N 3. 3	.e 33	2   🕩 🗓	] 🖷 🕄	🗟 🔊 🔊	📕 💠 🔹 🔿 🕶 💁 🕶
	🎄 Debug 😒						Step Retur	n All Debug sessions
● × × ●	<ul> <li>X Ipcxpresso54:</li> <li>A Place</li> <li>Phrea</li> <li>arm-none</li> <li>X Ipcxpresso54:</li> <li>A Place</li> <li>Ipcxpresso54:</li> <li>A Place</li> <li>Threa</li> <li>Threa</li> </ul>	114_multicore_example o54114_multicore_exam d #1 1 (Stopped) (Runn e-eabi-gdb (7.12.0.2016 114_multicore_example o54114_multicore_exam d #1 1 (Stopped) (Suspa ain() at hello_world_con e-eabi-gdb (7.12.0.2016	ples_hello_world_ ing) 1204) s_hello_world_cm ples_hello_world_ inded : Breakpoin e1.c:71 0x2001084	.cm4.axf [LPC 0plus Debug   .cm0plus.axf [ t)	54114J256 ( [C/C++ (N)	(cortex-m0 (P Semico	ctors) MCU Applicat (plus)] nductors) MCU App	tion]
₽								
	hello_world_core	0.c 🔥 fsl_mailbox	.h 💽 hello_	world_core1.c	23			
	65 */ 66⊜ int main(\ 67 { 68 uint32 69 70 /* Def	2_t startupData, i	cture for the	output LE	D pin*/			
		oin_config_t led_c SPIO_DigitalOutput	<u> </u>					
	76 MCMGR_ 77 /* Get 79 MCMGR_ 80 81 /* Mak 82 /* Use 83 for (i	itialize MCMGR bef Init(); t the startup data GetStartupData(kM ke a noticable del e startup paramete i = 0; i < startup elay();	*/ CMGR_Core1, & ay after the r from the ma	startupData reset */				
Figu	re 20. Auxiliary o	core application s	tops at the ma	ain functio	n			

At this point, it is possible to suspend and resume individual cores independently. It is also possible to make synchronous suspension and resumption of both the cores. This is done either by selecting both opened debug sessions (multiple selections) and clicking the "Suspend" / "Resume" control button, or just using the "Suspend All Debug sessions" and the "Resume All Debug sessions" buttons.

MCUXSDKGSUG

Getting Started with MCUXpresso SDK

🗙 workspace - Develop - Ipcxpresso54114_multicore_examples_hello_world_cm0plus/source/hello_world_core1.c - MCUXpresso IDE	
<u>File Edit Source Refactor Navigate Search Project Run FreeRTOS Window Help</u>	
:	0 -
P ☆ Debug ⊠	
Ipcxpresso54114_multicore_examples_hello_world_cm4 Debug [C/C++ (NXP Semiconductors) MCU Application]	
Ipcxpresso54114_multicore_examples_hello_world_cm4.axf [LPC54114J256 (cortex-m0plus)]	
1010	
iiii       iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii	
A Place provide the second	
Intered #11 (Stopped) (Running)	
() arm-none-eabi-gdb (7.12.0.20161204)	
(×)=	
(x)=	
9 <sub>0</sub>	
c hello_world_core0.c h fsl_mailbox.h c hello_world_core1.c ☆ c 0x190	
59asm("NOP"); /* delay */	
60 }	
61 } 62	
63⊖ /*!	
64 * @brief Main function	
65 */ 66⊖ int main(void)	
67 {	
68 uint32_t startupData, i;	
69 70 /* Define the init structure for the output LED pin*/	
71 gpio_pin_config_t led_config = {	
72 kGPIO_DigitalOutput, 0, 73 };	
73 }; 74	
75 /* Initialize MCMGR before calling its API */	
76 MCMGR_Init(); 77	
78 /* Get the startup data */	
<pre>79 MCMGR_GetStartupData(kMCMGR_Core1, &amp;startupData);</pre>	
80 81 /* Make a noticable delay after the reset */	
82 /* Use startup parameter from the master core */	
<pre>83 for (i = 0; i &lt; startupData; i++)</pre>	
84 delay(); 85	
Figure 21. Synchronous suspension/resumption of both cores using the multiple selections of debug session	ins
and "Suspend"/"Resume" controls	

Getting Started with MCUXpresso SDK

ile	orkspace - Develop - Ipcxpresso54114_multicore_examples_hello_world_cm0plus/source/hello_world_core1.c - MCUXpresso IDE Edit Source Refactor Navigate Search Project Run FreeRTOS Window Help
<u> </u>	▾ 🔚 🕼   🥸 ▾ 🐔 ፣ 🗞   🕨 🗉 🔳 🕅 ଅଧ୍ୟ ରହ 🖉   🧮 🐨 🚺 🖬 ଅଧ୍ୟରହ 🔊 🚺 👘 🦓 🏒 ! 🎋 ▾ 🔘
•••	to be bug ⊠
8	Ipcxpresso54114_multicore_examples_hello_world_cm4 Debug [C/C++ (NXP Semiconductors) MCU Application]
ò	Ipcxpresso54114_multicore_examples_hello_world_cm4.axf [LPC54114J256 (cortex-m0plus)]
8	Incorrection of the second
010 101	GPIO_ReadPinInput() at fsl_gpio.h:146 0x95c
5	main() at hello world core0.c:134 0xa10
	arm-none-eabi-gdb (7.12.0.20161204)
 8	Incorresso 54114_multicore_examples_hello_world_cm0plus Debug [C/C++ (NXP Semiconductors) MCU Application]
	Ipcxpresso54114_multicore_examples_hello_world_cm0plus.axf [LPC54114J256 (cortex-m0plus)]
5	Thread #1 1 (Stopped) (Suspended : Signal : SIGINT:Interrupt)
<)=	delay() at hello_world_core1.c:59 0x20010824
<)=	main() at hello_world_core1.c:99 0x200108a0
•	📕 arm-none-eabi-gdb (7.12.0.20161204)
Þ	
	Image: Control of the state of the sta
	<pre>&gt; 59 asm("NOP"); /* delay */</pre>
	60 }
	61 }
	62 63⊜ /*!
	64 * @brief Main function
	65 */
	66⊖ int main(void)
	67 { 68 uint32 t startupData, i;
	<pre>68 uint32_t startupData, i; 69</pre>
	<pre>70 /* Define the init structure for the output LED pin*/</pre>
	<pre>71 gpio_pin_config_t led_config = {</pre>
	72 kGPIO_DigitalOutput, 0,
	73 }; 74
	<pre>/* Initialize MCMGR before calling its API */</pre>
	76 MCMGR_Init();
	<pre>78 /* Get the startup data */ 79 MCMGR GetStartupData(kMCMGR Core1, &amp;startupData);</pre>
	80
	81 /* Make a noticable delay after the reset */
	82 /* Use startup parameter from the master core */
	<pre>83 for (i = 0; i &lt; startupData; i++)</pre>
	84 delay();
	re 22. Synchronous suspension/resumption of both cores using the "Suspend All Debug sessions" and the

### 4 Run a demo application using IAR

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

MCUXSDKGSUG

Getting Started with MCUXpresso SDK

**Note:** IAR Embedded Workbench for Arm version 8.32.3 is used in the following example, and the IAR toolchain should correspond to the latest supported version, as described in the MCUXpresso SDK Release Notes.

#### 4.1 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

Other example applications may have additional folders in their path.

2. Select the desired build target from the drop-down menu. For this example, select **hello\_world – debug**.

Workspace	
Debug	•
Debug	
Release	
🗆 🗇 hello_world - Deb	¥
🛛 🛏 🗀 board	
doc	
🗕 🛏 🗀 drivers	
🛛 🛏 🗀 source	
📕 🛏 🗀 startup	
📕 🛏 🗀 utilities	
📙 🖵 🗀 Output	

Figure 23. Demo build target selection

3. To build the demo application, click **Make**, highlighted in red in Figure 24.

Workspace	-	ąх	
r		1	
Debug		•	
Files	•	•	
🗆 🌒 hello_world - Debug	~		
🛏 🛋 board		•	
– ⊞ 🛋 doc			
🗕 🖽 🛋 drivers		•	
- 🕀 🛋 source		•	
⊢⊞ 🛋 startup		•	
🛏 🖬 utilities		•	
🖵 🖬 🖬 Output			

4. The build completes without errors.

#### 4.2 Run an example application

To download and run the application, perform these steps:

- 1. See the table in <u>Section 11</u> to determine the debug interface that comes loaded on your specific hardware platform.
  - The user should install LPCScrypt or MCUXpresso IDE to ensure that LPC board drivers are installed.

### Getting Started with MCUXpresso SDK

- For boards with P&E Micro interfaces, visit <u>www.pemicro.com/support/downloads\_find.cfm</u> and download the P&E Micro Hardware Interface Drivers package.
- 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 COM port (to determine the COM port number, see <u>Section 9</u>). Configure the terminal with these settings:
  - a. 115200 or 9600 baud rate, depending on your board (reference <code>BOARD\_DEBUG\_UART\_BAUDRATE variable in the board.h file)</code>
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit

PuTTY Configuration	×
Category: Session Logging Terminal Keyboard Bell Features Window Appearance Behaviour Translation Selection Colours Connection Data Proxy Telnet Rlogin SSH Serial	Basic options for your PuTTY session   Specify the destination you want to connect to   Serial line Speed   COM4 115200   Connection type: Image: Serial   Raw Telnet Rlogin   Saved Sessions Image: Serial   Default Settings Load   Delete Save   Close window on exit: Image: Serial   Always Never Only on clean exit   Open Cancel

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



Figure 26. Download and Debug button

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

Getting Started with MCUXpresso SDK



6. Run the code by clicking the Go button.



Figure 28. Go button

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

COM4 - PuTTY		×
hello world.		^
		$\sim$
Figure 29. Text display of the hello_world demo		

### 4.3 Build a multicore example application

This section describes the steps to build and run a dual-core application. The demo applications workspace files are located in this folder:

<install\_dir>/boards/<board\_name>/multicore\_examples/<application\_name>/<core\_type>/iar

Getting Started with MCUXpresso SDK

Begin with a simple dual-core version of the Hello World application. The multicore Hello World IAR workspaces are located in this folder:

```
<install dir>/boards/lpcxpresso54114/multicore examples/hello world/cm0plus/iar/
hello world cm0plus.eww
```

<install dir>/boards/lpcxpresso54114/multicore examples/hello world/cm4/iar/hello world cm4.eww

Build both applications separately by clicking the Make button. Build the application for the auxiliary core (cm0plus) first, because the primary core application project (cm4) must know the auxiliary core application binary when running the linker. It is not possible to finish the primary core linker when the auxiliary core application binary is not ready.

#### 4.4 Run a multicore example application

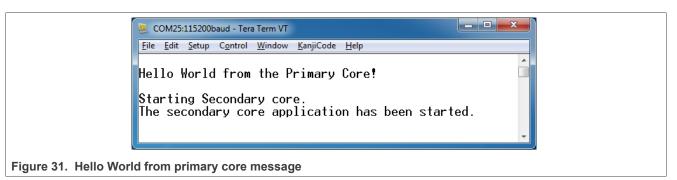
The primary core debugger handles flashing both primary and the auxiliary core applications into the SoC flash memory. To download and run the multicore application, switch to the primary core application project and perform steps 1 - 4 as described in <u>Section 4.2</u>. These steps are common for both single core and dual-core applications in IAR.

After clicking the "Download and Debug" button, the auxiliary core project is opened in the separate EWARM instance. Both the primary and auxiliary images are loaded into the device flash memory and the primary core application is executed. It stops at the default C language entry point in the main() function.

Run both cores by clicking the "Start all cores" button to start the multicore application.



During the primary core code execution, the auxiliary core is released from the reset. The hello world multicore application is now running and a banner is displayed on the terminal. If this does not appear, check the terminal settings and connections.



An LED controlled by the auxiliary core starts flashing, indicating that the auxiliary core has been released from the reset and is running correctly. When both cores are running, use the "Stop all cores", and "Start all cores" control buttons to stop or run both cores simultaneously.

0: 🐟 👻 1: 📑 👻 🚅 💕

Figure 32. "Stop all cores" and "Start all cores" control buttons

© 2024 NXP B.V. All rights reserved.

#### Getting Started with MCUXpresso SDK

### 5 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.

#### 5.1 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 appropriate CMSIS pack.

1. Open the MDK IDE, which is called µVision. In the IDE, select the **Pack Installer** icon.

W μVision												
File Edit	View Pr	oject Flash	Debug	Peripherals	Tools S	SVCS Wind	ow					
🗋 💕	a 🔊 🕽	6 5 6 6	9 6	(= =) 🖗	隐 隐	機律の	j= //					
🔗 🕮 I	iii 🧼 🗮	LOAD		<b>v</b> 4	8 🔒	팀 🚸 🔶						

Figure 33. Launch the Pack Installer

2. After the installation finishes, close the Pack Installer window and return to the µVision IDE.

### 5.2 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: 2. To build the demo project, select **Rebuild**, highlighted in red.



3. The build completes without errors.

#### 5.3 Run an example application

To download and run the application, perform these steps:

- 1. See the table in <u>Section 11</u> to determine the debug interface that comes loaded on your specific hardware platform.
  - For boards with the CMSIS-DAP/mbed/DAPLink interface, visit <u>mbed Windows serial configuration</u> and follow the instructions to install the Windows operating system serial driver. If running on Linux OS, this step is not required.
  - The user should install LPCScrypt or MCUXpresso IDE to ensure that LPC board drivers are installed.

#### Getting Started with MCUXpresso SDK

- For boards with a P&E Micro interface, visit <u>www.pemicro.com/support/downloads\_find.cfm</u> and download and install the P&E Micro Hardware Interface Drivers package.
- If using J-Link either a standalone debug pod or OpenSDA, install the J-Link software (drivers and utilities) from <u>www.segger.com/jlink-software.html</u>.
- For boards with the OSJTAG interface, install the driver from www.keil.com/download/docs/408.
- 2. Connect the development platform to your PC via USB cable using OpenSDA USB connector.
- 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 <u>Section 9</u>). Configure the terminal with these settings:
  - a. 115200 or 9600 baud rate, depending on your board (reference <code>BOARD\_DEBUG\_UART\_BAUDRATE variable in the board.h file)</code>
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit

MCUXSDKGSUG

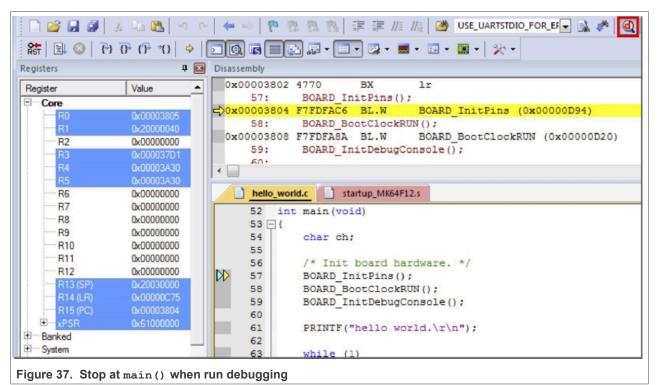
Getting Started with MCUXpresso SDK

4. In µVision, after the application is built, click the **Download** button to download the application to the target.

Getting Started with MCUXpresso SDK

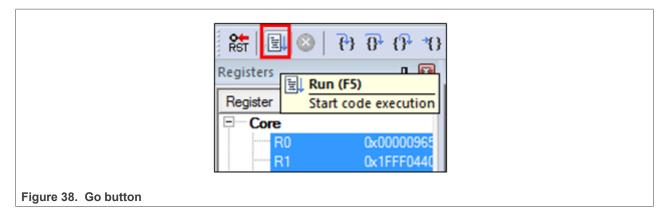


5. After clicking the **Download** button, the application downloads to the target and is running. To debug the application, click the **Start/Stop Debug Session** button, highlighted in red.



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

Getting Started with MCUXpresso SDK



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



### 5.4 Build a multicore example application

This section describes the steps to build and run a dual-core application. The demo applications workspace files are located in this folder:

<install\_dir>/boards/<board\_name>/multicore\_examples/<application\_name>/<core\_type>/mdk

Begin with a simple dual-core version of the Hello World application. The multicore Hello World Keil MSDK/ µVision workspaces are located in this folder:

```
<install_dir>/boards/lpcxpresso54114/multicore_examples/hello_world/cm0plus/mdk/
hello_world_cm0plus.uvmpw
```

<install\_dir>/boards/lpcxpresso54114/multicore\_examples/hello\_world/cm4/mdk/hello\_world\_cm4.uvmpw

Build both applications separately by clicking the **Rebuild** button. Build the application for the auxiliary core (cm0plus) first because the primary core application project (cm4) must know the auxiliary core application

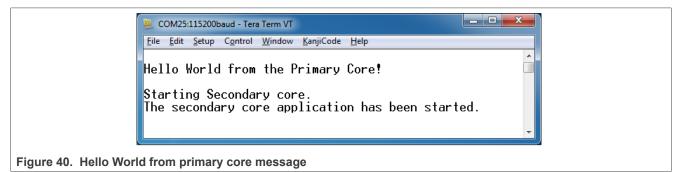
#### Getting Started with MCUXpresso SDK

binary when running the linker. It is not possible to finish the primary core linker when the auxiliary core application binary is not ready.

#### 5.5 Run a multicore example application

The primary core debugger flashes both the primary and the auxiliary core applications into the SoC flash memory. To download and run the multicore application, switch to the primary core application project and perform steps 1 - 5 as described in <u>Section 5.3</u>. These steps are common for both single-core and dual-core applications in  $\mu$ Vision.

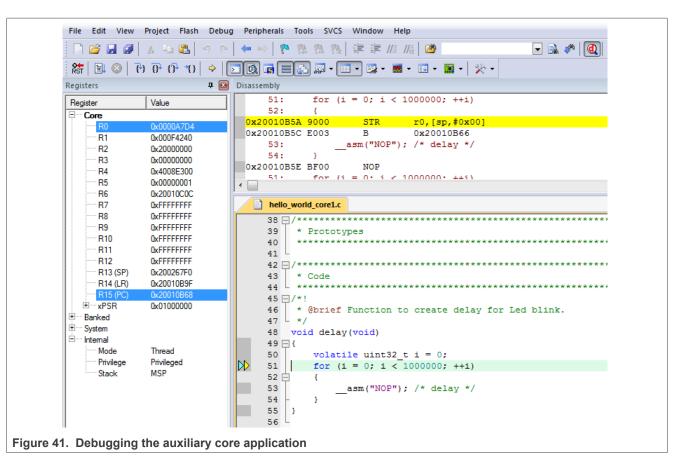
Both the primary and the auxiliary image is loaded into the device flash memory. After clicking the "Run" button, the primary core application is executed. During the primary core code execution, the auxiliary core is released from the reset. The hello\_world multicore application is now running and a banner is displayed on the terminal. If this does not appear, check your terminal settings and connections.



An LED controlled by the auxiliary core starts flashing indicating that the auxiliary core has been released from the reset and is running correctly.

Attach the running application of the auxiliary core by opening the auxiliary core project in the second µVision instance and clicking the "Start/Stop Debug Session" button. After this, the second debug session is opened and the auxiliary core application can be debugged.

Getting Started with MCUXpresso SDK



Arm describes multicore debugging using the NXP LPC54114 Cortex-M4/M0+ dual-core processor and Keil uVision IDE in Application Note 318 at <u>www.keil.com/appnotes/docs/apnt\_318.asp</u>. The associated video can be found <u>here</u>.

### 6 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.

### 6.1 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.

#### 6.1.1 Install GCC Arm Embedded tool chain

Download and run the installer from GNU Arm Embedded Toolchain. This is the actual toolset (in other words, compiler, linker, and so on). The GCC toolchain should correspond to the latest supported version, as described in *MCUXpresso SDK Release Notes*.

#### Getting Started with MCUXpresso SDK

#### 6.1.2 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 MinGW.
- 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 42. Set up MinGW and MSYS

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

MinGW Installation M Installation Package	-	_		
Update Catalogue			Package	
Mark All Upgrades			mingw-developer-tool	
Apply Changes			mingw32-base	
· · · · · · · · · · · · · · · · · · ·			mingw32-gcc-ada	
Quit	Alt+F4		mingw32-gcc-fortran	
		тШ	mingw32-gcc-g++	
			mingw32-gcc-objc	
		5	msys-base	

Figure 43. Complete MinGW and MSYS installation

 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 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.

#### Getting Started with MCUXpresso SDK

	System Properties
	Computer Name Hardware Advanced System Protection Remote
	Environment Variables
	Edit System Variable
	Variable name: Path
	Variable value: pgram Files (x86)\CMake\bin;C:\MinGW\bin
	OK Cancel
	System variables
	Variable Value ^
	OS Windows_NT
	Path C:\Program Files (x86)\Parallels\Parallels
	PATHEXT .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;
	PROCESSOR_A AMD64
	New Edit Delete
	OK Cancel
Figure 44. Add Path to system	s environment

#### 6.1.3 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\8 2018-q4-major

See the installation folder of the GNU Arm GCC Embedded tools for the exact pathname 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.



Figure 45. Convert path to short path

Getting Started with MCUXpresso SDK

Variable         Value           OneDrive         C:\Users\         \OneDrive - NXP           OneDriveConfimercial         C:\Users\         \OneDrive - NXP           Path         C:\Ruby24-x64\bin;C:\Users\nxa07599\AppData           PATHEXT         .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;JSE;.WSF;.WSI           TEMP         C:\Users\         \AppData\Local\Temp	N	
OneDriveConFinercial         C:\Users\         \OneDrive - NXP           Path         C:\Ruby24-x64\bin;C:\Users\nxa07599\AppData           PATHEXT         .COM;.EXE;;BAT;.CMD;.VBS;.VBE;JS;JSE;.WSF;.WSF           TEMP         C:\Users\         \AppData\Local\Temp	March	
PATHEXT .COM;:EXE;:BAT;:CMD;:VBS;:VBE;JS;JSE;:WSF;:WSF TEMP C:\Users\ \AppData\Local\Temp	A see Marine	
TEMP C:\Users\ \AppData\Local\Temp	a\Local\Micros	
	H;.MSC;.RB;.RB	
TMP C:\Users\ \AppData\Local\Temp		
Variable name: ARMGCC_DIR		×
Variable value: C:\PROGRA~2\GNUTOO~1\82018-~1		
Browse Directory Browse File	OK Car	icel
IAR_WORKBENCH C:\Program Files (x86)\IAR Systems\Embedded	Workbench 8.2	TT
JLINK_DIR C:\Program Files (x86)\SEGGER\JLink_V640		
KEIL C:\Keil_v5\UV4		
myCleanUp No	~	·
New Edit	Delete	

#### 6.1.4 Install CMake

- 1. Download CMake 3.0.x from <u>www.cmake.org/cmake/resources/software.html</u>.
- 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.

#### Getting Started with MCUXpresso SDK

CMake 3.0.2 Setup	
Install Options Choose options for installing CMake 3.0.2	
By default CMake does not add its directory to the system PATH.	
<ul> <li>Do not add CMake to the system PATH</li> <li>Add CMake to the system PATH for all users</li> <li>Add CMake to the system PATH for current user</li> </ul>	
Create CMake Desktop Icon	
Nullsoft Install System v2.46	Cancel
Figure 47. 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.

#### 6.2 Build an example application

To build an example application, follow these steps.

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
i Uninstall GNU Tools for ARM Embe
Figure 48. 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:

*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 49.

Getting Started with MCUXpresso SDK

[ 84%] Building C object CMakeFiles/hello_world.elf.dir/C_/nxp/SDK_2.0_FRDM-K64F
/devices/MK64F12/drivers/fsl_smc.c.obj
[ 92%] Building C object CMakeFiles/hello_world.elf.dir/C_/nxp/SDK_2.0_FRDM-K64F
/devices/MK64F12/drivers/fsl_clock.c.obj
[100%] Linking C executable debughello_world.elf
[100%] Built target hello_vorld.elf
C:\nxp\SDK_2.0_FRDM-K64F\boards\frdmk64f\demo_apps\hello_world\armgcc>IF "" == "
" (pause )
Press any key to continue

Figure 49. hello\_world demo build successful

### 6.3 Run an example application

This section describes steps to run a demo application using J-Link GDB Server application. To update the onboard LPC-Link2 debugger to Jlink firmware, see <u>Section 12</u>.

**Note:** J-Link GDB Server application is not supported for TFM examples. Use CMSIS DAP instead of J-Link for flashing and debugging TFM examples.

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

- 1. Connect the development platform to your PC via USB cable between the LPC-Link2 USB connector (may be named OSJTAG for some boards) and the PC USB connector. If using a standalone J-Link debug pod, connect it to the SWD/JTAG connector of the board.
- 2. 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 <u>Section 9</u>). Configure the terminal with these settings:
  - a. 115200 or 9600 baud rate, depending on your board (reference <code>BOARD\_DEBUG\_UART\_BAUDRATE variable in board.h file)</code>
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit

#### Getting Started with MCUXpresso SDK

Category:	
Session	Basic options for your PuTTY session
Logging	Specify the destination you want to connect to
Keyboard	Serial line Speed
Bell	COM4 115200
<ul> <li>Features</li> <li>Window</li> <li>Appearance</li> <li>Behaviour</li> <li>Translation</li> <li>Selection</li> </ul>	Connection type: Raw Telnet Rlogin SSH Serial Load, save or delete a stored session Saved Sessions
Colours Connection Data Proxy Telnet Rlogin SSH	Default Settings       Load         Save       Delete
About	Close window on exit: Always Never Only on clean exit Open Cancel

**Note:** Make sure that the board is set to FlexSPI flash boot mode (ISP2: ISP1: ISP0 = ON, OFF, ON) before use GDB debug.

- Open the J-Link GDB Server application. Assuming the J-Link software is installed, the application can be launched by going to the Windows operating system Start menu and selecting Programs -> SEGGER -> J-Link <version> J-Link GDB Server.
- 4. After it is connected, the screen should look like this figure:

#### Getting Started with MCUXpresso SDK

File Help	R J-Link GDB Server V6.46g			_		<
J-Link	Waiting for connection     SWI       Connected     SWI       .3A60xxx_M4 (Halted)     3.29		4000 kHz little endian	Gene	on top v log window erate logfile v download	
Hardware S/N: 3610 Checking Target vo Listening Connectin Connected	: J-Link Lite-FSL V1 co : V1.00	ompiled Jun 25	2012 16:40:07			^
0 bytes dow	nloaded		Conne	cted to target		

Figure 51. 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.

Description
bocumentation
GCC Command Prompt
🐻 Uninstall GNU Tools for ARM Embec

#### Figure 52. Launch command prompt

6. 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/frdmk32l3a6/demo\_apps/hello\_world/cm4/armgcc/debug

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

### Getting Started with MCUXpresso SDK

GCC Command Prompt - arm-none-eabi-gdb.exe C:\Users\nxa12829\Desktop\k32l3\boards\frdmk32l3a6\demo_apps\hello_world\cm4\armgcc\debu 🛛 🗙
C:\Program Files (x86)\GNU Tools ARM Embedded\8 2018-q4-major>arm-none-eabi-gdb.exe C:\Users\nxa12829\Desktop\k3213\boar ds\frdmk3213a6\demo_apps\hello_world\cm4\armgcc\debug\hello_world_demo_cm4.elf C:\Program Files (x86)\GNU Tools ARM Embedded\8 2018-q4-major\bin\arm-none-eabi-gdb.exe: warning: Couldn't determine a p ath for the index cache directory. GNU gdb (GNU Tools for Arm Embedded Processors 8-2018-q4-major) 8.2.50.20181213-git Copyright (C) 2018 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <http: gnu.org="" gpl.html="" licenses=""> 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:>
For help, type "help". Type "apropos word" to search for commands related to "word" Reading symbols from C:\Users\nxa12829\Desktop\k3213\boards\frdmk3213a6\demo_apps\hello_world\cm4\armgcc\debug\hello_wor ld_demo_cm4.elf (gdb) Figure 53. Run arm-none-eabi-gdb

#### 8. Run these commands:

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

```
b. \ \text{monitor reset} \\
```

- C. monitor halt
- **d**. load
- e. monitor reset
- 9. The application is now downloaded and halted at the watchpoint. 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 does not appear, check your terminal settings and connections.

COM4 - PuTTY	_	×
hello world.		^
		$\sim$

Figure 54. Text display of the hello\_world demo

#### 6.4 Build a multicore example application

This section describes the steps to build and run a dual-core application. The demo application build scripts are located in this folder:

<install\_dir>/boards/<board\_name>/multicore\_examples/<application\_name>/<core\_type>/armgcc

#### Getting Started with MCUXpresso SDK

Begin with a simple dual-core version of the Hello World application. The multicore Hello World GCC build scripts are located in this folder:

<install\_dir>/boards/lpcxpresso54114/multicore\_examples/hello\_world/cm0plus/armgcc/build\_debug.bat

<install\_dir>/boards/lpcxpresso54114/multicore\_examples/hello\_world/cm4/armgcc/build\_debug.bat

#### Build both applications separately following steps for single core examples as described in Section 6.2.

🔤 GCC Command Prompt - build_debug.bat	_		×
[ 47%] Building C object CMakeFiles/hello_world_cm0plus.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/de	vices/	K32L3A	60/ ^
drivers/fsl_common.c.obj			
[ 52%] Building C object CMakeFiles/hello_world_cm0plus.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/de drivers/fsl msmc.c.obj	vices/	K32L3A	60/
[ 56%] Building C object CMakeFiles/hello world cm0plus.elf.dir/C /packages/SDK 2.6.0 FRDM-K32L3A6 RC1/de	vices/	K32L3A	607
utilities/debug console/fsl debug console.c.obj			
[ 60%] Building C object CMakeFiles/hello_world_cm0plus.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/de		K32L3A	60/
utilities/fsl_assert.c.obj			
[ 65%] Building C object CMakeFiles/hello_world_cm0plus.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/de	vices/	K32L3A	60/
utilities/str/fsl_str.c.obj [ 69%] Building C object CMakeFiles/hello_world_cm0plus.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/co	mpopen	ts/uan	+/1
[ Use] outsing C Use Charles Merro, we is a compression of the packages Suc. 10.0, Non-KSEESKO_KEI/CO		c3/ dai	C7 ±
73%] Building C object CMakeFiles/hello_world_cm0plus.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/co	mponen		ial
_manager/serial_manager.c.obj			
[78%] Building C object CMakeFiles/hello_world_cm0plus.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/co		ts/ser	ial
_manager/serial_port_uart.c.obj [ 82%] Building C object CMakeFiles/hello_world_cm0plus.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/co	mpopen	tc/lic	+c/
generic list.c.obj			-37
[ 86%] Building C object CMakeFiles/hello_world_cm0plus.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/de	vices/	K32L3A	60/
system_K32L3A60_cm0plus.c.obj			
[ 91%] Building ASM object CMakeFiles/hello_world_cm0plus.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/	device	s/K32L	3A6
0/gcc/startup_K32L3A60_cm0plus.S.obj [ 95%] Building C object CMakeFiles/hello_world_cm0plus.elf.dir/C /packages/SDK 2.6.0 FRDM-K32L3A6 RC1/mi	ddl ewa	re/mul	tic
[ Job] officially conject characteristics metro_works_emopres.err.ur/c_/packages/Jok_z.o.o_non-k5205Ac_ker/mi	autewa	r cy mar	
[100%] Linking C executable debug\hello_world_cm0plus.elf			
[100%] Built target hello_world_cm0plus.elf			
	<b>F H H</b>		
c:\packages\SDK_2.6.0_FRDM-K32L3A6_RC1\boards\frdmk32l3a6\multicore_examples\hello_world\cm0plus\armgcc>I se )	=	= (	pau
Press any key to continue			~
Figure 55. hello world cm0plus example build successful			
rigure 33. heno_wonu_chiopius example bunu successiui			

MCUXSDKGSUG

Getting Started with MCUXpresso SDK

		_	
GCC Command Prompt - build_debug.bat	_		×
[ 50%] Building C object CMakeFiles/hello_world_cm4.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/devic	es/K321		lriv /
ers/fsl_lpuart.c.obj [ f49] mulling o phint curbesiles/balls world and all dis/o (archang/opy a c o spou yaarang double			
[ 54%] Building C object CMakeFiles/hello_world_cm4.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/devic ers/fsl common.c.obj	25/K32L	L3A60/0	Iriv
[ 58%] Building C object CMakeFiles/hello_world_cm4.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/devic	as/K321	3460/0	leiv
issi sung e bijeet enskerites, keite_wordwordwordesteriobekeges, sskrefe_, kan kszeske_keite			
[ 62%] Building C object CMakeFiles/hello_world_cm4.elf.dir/C_/packages/SDK_2.6.0 FRDM-K32L3A6_RC1/devic	es/K321		itil
ties/str/fsl_str.c.obj			
[ 66%] Building C object CMakeFiles/hello_world_cm4.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/devic	25/K321	L3A60/u	util
ities/fsl_assert.c.obj			
70%] Building C object CMakeFiles/hello_world_cm4.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/devic ties/debug console/fsl debug console.c.obj	25/K32L	L3A60/1	1111
75%] Building C object CMakeFiles/hello world cm4.elf.dir/C /packages/SDK 2.6.0 FRDM-K32L3A6 RC1/compo	ients/i	uart/lr	buar
adapter.c.obj			
79%] Building C object CMakeFiles/hello_world_cm4.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/compo	ients/s		man
ager/serial_port_uart.c.obj			
83%] Building C object CMakeFiles/hello_world_cm4.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/compo	ients/s	serial_	_man
ger/serial_manager.c.obj 87%] Building C object CMakeFiles/hello_world_cm4.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/compo	oonts /	licte//	Topo
oral building cobject chargeriles/helio_world_cma.elf.dlf/c_/packages/suk_2.0.0_rkbm-ks2csko_kci/compo		11212/8	jene
918] Building C object CMakeFiles/hello world cm4.elf.dir/C /packages/SDK 2.6.0 FRDM-K32L3A6 RC1/devic	es/K321	L3A60/9	syst
m_K32L3A60_cm4.c.obj			
95%] Building ASM object CMakeFiles/hello_world_cm4.elf.dir/C_/packages/SDK_2.6.0_FRDM-K32L3A6_RC1/dev	.ces/K3	32L3A60	∂/gc
/startup_K32L3A60_cm4.S.obj			
100%] Linking C executable debug\hello_world_cm4.elf 100%] Built target hello world cm4.elf			
loos built carget hello_world_care.ell			
::\packages\SDK_2.6.0_FRDM-K32L3A6_RC1\boards\frdmk32l3a6\multicore_examples\hello_world\cm4\armgcc>IF "	""	" (paus	se
Press any key to continue			
Figure F6, hollo world and example build successful			
igure 56. hello_world_cm4 example build successful			

### 6.5 Run a multicore example application

When running a multicore application, the same prerequisites for J-Link/J-Link OpenSDA firmware, and the serial console as for the single-core application, applies, as described in <u>Section 6.3</u>.

The primary core debugger handles flashing of both the primary and the auxiliary core applications into the SoC flash memory. To download and run the multicore application, switch to the primary core application project and perform steps 1 to 10, as described in <u>Section 6.3</u>. These steps are common for both single-core and dual-core applications in Arm GCC.

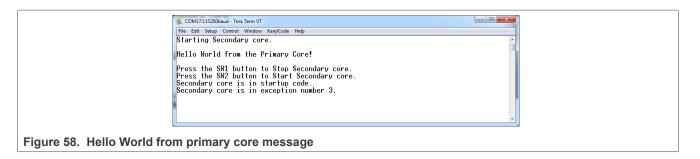
Both the primary and the auxiliary image is loaded into the SPI flash memory. After execution of the monitor go command, the primary core application is executed. During the primary core code execution, the auxiliary core code is reallocated from the flash memory to the RAM, and the auxiliary core is released from the reset. The hello\_world multicore application is now running and a banner is displayed on the terminal. If this is not true, check your terminal settings and connections.

Getting Started with MCUXpresso SDK

Administrator: GCC Command Prompt
c:\D\SDK_2.0_LPCXpresso54114\boards\lpcxpresso54114\multicore_examples\hello_wor ld\cm4\armgcc>IF "" == "" (pause > Press any key to continue
c:\D\SDK_2.0_LPCXpresso54114\boards\lpcxpresso54114\multicore_examples\hello_wor ld\cm4\armgcc>cd_debug
c:\D\SDK_2.0_LPCXpresso54114\boards\lpcxpresso54114\multicore_examples\hello_wor ld\cm4\armgcc\debug>arm-none-eabi-gdb.exe hello_world_cm4.elf GNU gdb (GNU Tools for ARM Embedded Processors 6-2017-q2-update> 7.12.1.20170417
-git Copyright (C) 2017 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <a href="http://gnu.org/licenses/gpl.html">http://gnu.org/licenses/gpl.html</a> 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:
<pre>Khttp://www.gnu.org/software/gdb/bugs/&gt;. Find the GDB manual and other documentation resources online at:</pre>
Type "apropos word" to search for commands related to "word" Reading symbols from hello_world_cm4.elfdone. (gdb) target remote localhost:2331
Remote debugging using localhost:2331 0x00004290 in ?? <> <gdb> monitor reset Resetting target</gdb>
(gdb) monitor halt (gdb) load Loading section .interrupts, size 0xe0 lma 0x0
Loading section .text, size 0x3614 lma 0xe4 Loading section .ARM, size 0x8 lma 0x36f8 Loading section .init_array, size 0x4 lma 0x3700
Loading section .fini_array, size 0x4 1ma 0x3704 Loading section .data, size 0x68 1ma 0x3708 Loading section .m0code, size 0x1f64 1ma 0x30000 Start address 0x1d8, load size 22224
Transfer rate: 1973 KB/sec, 3174 bytes/write. (gdb) monitor reset Resetting target
(gdb) monitor go (gdb) q A debugging session is active.
Inferior 1 [Remote target] will be killed.
Quit anyway? (y or n) y c:\D\SDK_2.0_LPCXpresso54114\boards\lpcxpresso54114\multicore_examples\hello_wor—
ld\cm4\armgcc\debug>
Figure 57. Loading and running the multicore example

Figure 57. Loading and running the multicore example

Getting Started with MCUXpresso SDK



### 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 set up various on-chip peripherals prior to the program launch.	<b>0</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 <u>www.nxp.com/mcuxpresso</u>. Recommended for customers using IAR Embedded Workbench, Keil MDK μVision, or Arm GCC.
- **Online version** available on <u>mcuxpresso.nxp.com</u>. Recommended doing 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.

#### Getting Started with MCUXpresso SDK

### 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 <u>Figure 59</u>.

U Quickstart Panel 🛛 Global Variables 🌾 Variables 🗣 Breakpoints 🗄 Outline	
MCUXpresso IDE (Free Edition)	Â
▼ 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>>	-

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. All NXP boards ship with a factory programmed, onboard debug interface, whether it is based on OpenSDA or the legacy P&E Micro OSJTAG interface. To determine what your specific board ships with, see <u>Section 11</u>.

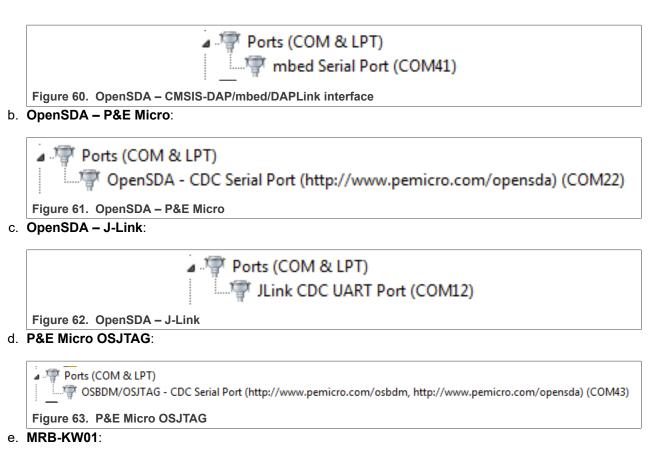
1. **Linux**: The serial port can be determined by running the following command after the USB Serial is connected to the host:

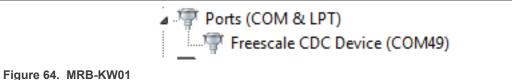
```
$ dmesg | grep "ttyUSB"
[503175.307873] usb 3-12: cp210x converter now attached to ttyUSB0
[503175.309372] usb 3-12: cp210x converter now attached to ttyUSB1
```

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 the **Start** menu and type **Device Manager** in the search bar.
- 3. In the Device Manager, expand the **Ports (COM & LPT)** section to view the available ports. The COM port names are different for all the NXP boards.
  - a. OpenSDA CMSIS-DAP/mbed/DAPLink interface:

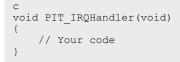
Getting Started with MCUXpresso SDK





### 10 How to define IRQ handler in CPP files

With MCUXpresso SDK, users could define their own IRQ handler in application level to override the default IRQ handler. For example, to override the default <code>PIT\_IRQHandler</code> define in <code>startup\_DEVICE.s</code>, application code like app.c can be implement like:



When application file is CPP file, like app.cpp, then extern "C" should be used to ensure the function prototype alignment.

```
cpp
extern "C" {
    void PIT_IRQHandler(void);
}
void PIT_IRQHandler(void)
{
    // Your code
```

© 2024 NXP B.V. All rights reserved.

Getting Started with MCUXpresso SDK

}

### 11 Default debug interfaces

The MCUXpresso SDK supports various hardware platforms that come loaded with various factory programmed debug interface configurations. <u>Table 2</u> lists the hardware platforms supported by the MCUXpresso SDK, their default debug interface, and any version information that helps differentiate a specific interface configuration.

Note: The OpenSDA details column in Table 2 is not applicable to LPC.

Hardware platform	Default interface	OpenSDA details
EVK-MC56F83000	P&E Micro OSJTAG	N/A
EVK-MIMXRT595	CMSIS-DAP	N/A
EVK-MIMXRT685	CMSIS-DAP	N/A
FRDM-K22F	CMSIS-DAP/mbed/DAPLink	OpenSDA v2.1
FRDM-K28F	DAPLink	OpenSDA v2.1
FRDM-K32L2A4S	CMSIS-DAP	OpenSDA v2.1
FRDM-K32L2B	CMSIS-DAP	OpenSDA v2.1
FRDM-K32W042	CMSIS-DAP	N/A
FRDM-K64F	CMSIS-DAP/mbed/DAPLink	OpenSDA v2.0
FRDM-K66F	J-Link OpenSDA	OpenSDA v2.1
FRDM-K82F	CMSIS-DAP	OpenSDA v2.1
FRDM-KE15Z	DAPLink	OpenSDA v2.1
FRDM-KE16Z	CMSIS-DAP/mbed/DAPLink	OpenSDA v2.2
FRDM-KL02Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL03Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL25Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL26Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL27Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL28Z	P&E Micro OpenSDA	OpenSDA v2.1
FRDM-KL43Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL46Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KL81Z	CMSIS-DAP	OpenSDA v2.0
FRDM-KL82Z	CMSIS-DAP	OpenSDA v2.0
FRDM-KV10Z	CMSIS-DAP	OpenSDA v2.1
FRDM-KV11Z	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KV31F	P&E Micro OpenSDA	OpenSDA v1.0
FRDM-KW24	CMSIS-DAP/mbed/DAPLink	OpenSDA v2.1
FRDM-KW36	DAPLink	OpenSDA v2.2
FRDM-KW41Z	CMSIS-DAP/DAPLink	OpenSDA v2.1 or greater
MCUXSDKGSUG	All information provided in this document is subject to legal dis	sclaimers. © 2024 NXP B.V. All rights reserved

Table 2. Hardware platforms supported by MCUXpresso SDK

### Getting Started with MCUXpresso SDK

	ported by MCUXpresso SDKcontinued	
Hardware platform	Default interface	OpenSDA details
Hexiwear	CMSIS-DAP/mbed/DAPLink	OpenSDA v2.0
HVP-KE18F	DAPLink	OpenSDA v2.2
HVP-KV46F150M	P&E Micro OpenSDA	OpenSDA v1
HVP-KV11Z75M	CMSIS-DAP	OpenSDA v2.1
HVP-KV58F	CMSIS-DAP	OpenSDA v2.1
HVP-KV31F120M	P&E Micro OpenSDA	OpenSDA v1
JN5189DK6	CMSIS-DAP	N/A
LPC54018 IoT Module	N/A	N/A
LPCXpresso54018	CMSIS-DAP	N/A
LPCXpresso54102	CMSIS-DAP	N/A
LPCXpresso54114	CMSIS-DAP	N/A
LPCXpresso51U68	CMSIS-DAP	N/A
LPCXpresso54608	CMSIS-DAP	N/A
LPCXpresso54618	CMSIS-DAP	N/A
LPCXpresso54628	CMSIS-DAP	N/A
LPCXpresso54S018M	CMSIS-DAP	N/A
LPCXpresso55s16	CMSIS-DAP	N/A
LPCXpresso55s28	CMSIS-DAP	N/A
LPCXpresso55s69	CMSIS-DAP	N/A
MAPS-KS22	J-Link OpenSDA	OpenSDA v2.0
MIMXRT1170-EVK	CMSIS-DAP	N/A
TWR-K21D50M	P&E Micro OSJTAG	N/AOpenSDA v2.0
TWR-K21F120M	P&E Micro OSJTAG	N/A
TWR-K22F120M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-K24F120M	CMSIS-DAP/mbed	OpenSDA v2.1
TWR-K60D100M	P&E Micro OSJTAG	N/A
TWR-K64D120M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-K64F120M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-K65D180M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-K65D180M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KV10Z32	P&E Micro OpenSDA	OpenSDA v1.0
TWR-K80F150M	CMSIS-DAP	OpenSDA v2.1
TWR-K81F150M	CMSIS-DAP	OpenSDA v2.1
TWR-KE18F	DAPLink	OpenSDA v2.1
TWR-KL28Z72M	P&E Micro OpenSDA	OpenSDA v2.1
TWR-KL43Z48M	P&E Micro OpenSDA	OpenSDA v1.0

## Table 2. Hardware platforms supported by MCUXpresso SDK continued

MCUXSDKGSUG User guide

#### Getting Started with MCUXpresso SDK

Hardware platform	Default interface	OpenSDA details
TWR-KL81Z72M	CMSIS-DAP	OpenSDA v2.0
TWR-KL82Z72M	CMSIS-DAP	OpenSDA v2.0
TWR-KM34Z75M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KM35Z75M	DAPLink	OpenSDA v2.2
TWR-KV10Z32	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KV11Z75M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KV31F120M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KV46F150M	P&E Micro OpenSDA	OpenSDA v1.0
TWR-KV58F220M	CMSIS-DAP	OpenSDA v2.1
TWR-KW24D512	P&E Micro OpenSDA	OpenSDA v1.0
USB-KW24D512	N/A External probe	N/A
USB-KW41Z	CMSIS-DAP\DAPLink	OpenSDA v2.1 or greater
	1	

Table 2. Hardware platforms supported by MCUXpresso SDK...continued

### 12 Updating debugger firmware

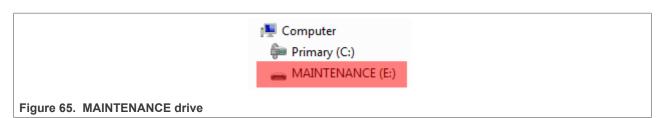
#### 12.1 Updating OpenSDA firmware

Any NXP hardware platform that comes with an OpenSDA-compatible debug interface has the ability to update the OpenSDA firmware. This typically means to switch from the default application (either CMSIS-DAP/mbed/ DAPLink or P&E Micro) to a SEGGER J-Link. This section contains the steps to switch the OpenSDA firmware to a J-Link interface. However, the steps can be applied to restoring the original image also. For reference, OpenSDA firmware files can be found at the links below:

- <u>J-Link</u>: Download appropriate image from <u>www.segger.com/opensda.html</u>. Choose the appropriate J-Link binary based on the table in <u>Section 11</u>. Any OpenSDA v1.0 interface should use the standard OpenSDA download (in other words, the one with no version). For OpenSDA 2.0 or 2.1, select the corresponding binary.
- CMSIS-DAP/mbed/DAPLink: DAPLink OpenSDA firmware is available at www.nxp.com/opensda.
- <u>P&E Micro</u>: Downloading P&E Micro OpenSDA firmware images requires registration with P&E Micro (<u>www.pemicro.com</u>).

Perform the following steps to update the OpenSDA firmware on your board for Windows and Linux OS users:

- 1. Unplug the board's USB cable.
- 2. Press the **Reset** button on the board. While still holding the button, plug the USB cable back into the board.
- 3. When the board re-enumerates, it shows up as a disk drive called MAINTENANCE.



4. Drag and drop the new firmware image onto the MAINTENANCE drive.

#### Getting Started with MCUXpresso SDK

**Note:** If for any reason the firmware update fails, the board can always reenter maintenance mode by holding down **Reset** button and power cycling.

These steps show how to update the OpenSDA firmware on your board for Mac OS users.

- 1. Unplug the board's USB cable.
- 2. Press the **Reset** button of the board. While still holding the button, plug the USB cable back into the board.
- 3. For boards with OpenSDA v2.0 or v2.1, it shows up as a disk drive called **BOOTLOADER** in **Finder**. Boards with OpenSDA v1.0 may or may not show up depending on the bootloader version. If you see the drive in **Finder**, proceed to the next step. If you do not see the drive in Finder, use a PC with Windows OS 7 or an earlier version to either update the OpenSDA firmware, or update the OpenSDA bootloader to version 1.11 or later. The bootloader update instructions and image can be obtained from P&E Microcomputer website.
- 4. For OpenSDA v2.1 and OpenSDA v1.0 (with bootloader 1.11 or later) users, drag the new firmware image onto the BOOTLOADER drive in **Finder**.
- 5. For OpenSDA v2.0 users, type these commands in a Terminal window:

> sudo mount -u -w -o sync /Volumes/BOOTLOADER > cp -X <path to update file> /Volumes/BOOTLOADER

**Note:** If for any reason the firmware update fails, the board can always reenter bootloader mode by holding down the **Reset** button and power cycling.

### 12.2 Updating LPCXpresso board firmware

The LPCXpresso hardware platform comes with a CMSIS-DAP-compatible debug interface (known as LPC-Link2). This firmware in this debug interface may be updated using the host computer utility called LPCScrypt. This typically used when switching between the default debugger protocol (CMSIS-DAP) to SEGGER J-Link, or for updating this firmware with new releases of these. This section contains the steps to reprogram the debug probe firmware.

**Note:** If MCUXpresso IDE is used and the jumper making DFUlink is installed on the board (JP5 on some boards, but consult the board user manual or schematic for specific jumper number), LPC-Link2 debug probe boots to DFU mode, and MCUXpresso IDE automatically downloads the CMSIS-DAP firmware to the probe before flash memory programming (after clicking **Debug**). Using DFU mode ensures that most up-to-date/ compatible firmware is used with MCUXpresso IDE.

NXP provides the LPCScrypt utility, which is the recommended tool for programming the latest versions of CMSIS-DAP and J-Link firmware onto LPC-Link2 or LPCXpresso boards. The utility can be downloaded from <a href="http://www.nxp.com/lpcutilities">www.nxp.com/lpcutilities</a>.

These steps show how to update the debugger firmware on your board for Windows operating system. For Linux OS, follow the instructions described in LPCScrypt user guide (<u>www.nxp.com/lpcutilities</u>, select **LPCScrypt**, and then the documentation tab).

- 1. Install the LPCScript utility.
- 2. Unplug the board's USB cable.
- 3. Make the DFU link (install the jumper labeled DFUlink).
- 4. Connect the probe to the host via USB (use Link USB connector).
- 5. Open a command shell and call the appropriate script located in the LPCScrypt installation directory (<LPCScrypt install dir>).
  - a. To program CMSIS-DAP debug firmware: <LPCScrypt install dir>/scripts/program\_CMSIS
  - b. To program J-Link debug firmware: <LPCScrypt install dir>/scripts/program\_JLINK
- 6. Remove DFU link (remove the jumper installed in <u>Step 3</u>).
- 7. Repower the board by removing the USB cable and plugging it in again.

© 2024 NXP B.V. All rights reserved.

Getting Started with MCUXpresso SDK

## 13 Revision history

This table summarizes revisions to this document.

#### Table 3. Revision history

Revision number	Date	Substantive changes
2.13.0	22 December 2022	Updated for MCUXpresso SDK v2.13.0
2.14.0	22 December 2022	Updated for MCUXpresso SDK v2.14.0
2.15.000	10 January 2024	Updated for MCUXpresso SDK v2.15.000

#### Getting Started with MCUXpresso SDK

### Legal information

### Definitions

**Draft** — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

### Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

**Right to make changes** — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

**Applications** — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect. Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

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

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

**Translations** — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at <u>PSIRT@nxp.com</u>) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

 $\ensuremath{\mathsf{NXP}}\xspace$  B.V. is not an operating company and it does not distribute or sell products.

### Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners. **NXP** — wordmark and logo are trademarks of NXP B.V.

#### Getting Started with MCUXpresso SDK

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 and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) 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. **Tower** — is a trademark of NXP B.V.

### Getting Started with MCUXpresso SDK

### **Tables**

Tab. 1.	MCUXpresso Config Tools42
Tab. 2.	Hardware platforms supported by
	MCUXpresso SDK45

## Figures

Fig. 1.	MCUXpresso SDK layers2
Fig. 2.	Application folder structure
Fig. 3.	Install an SDK4
Fig. 4.	Import an SDK example5
Fig. 5.	Select hello_world6
Fig. 6.	Terminal (PuTTY) configurations8
Fig. 7.	Attached Probes: debug emulator selection9
Fig. 8.	Resume button9
Fig. 9.	Text display of the hello_world demo10
Fig. 10.	Select the hello_world multicore example 10
Fig. 11.	Selection of the build target in
	MCUXpresso IDE11
Fig. 12.	Switching multicore projects into the
	Release build configuration11
Fig. 13.	Debug "frdmk32l3a6_hello_world_cm4"
	case
Fig. 14.	Attached Probes: debug emulator selection 13
Fig. 15.	Target core selection dialogue
Fig. 16.	Stop the primary core application at main()
	when running debugging14
Fig. 17.	Hello World from the primary core message 14
Fig. 18.	Debug "lpcxpresso54114_multicore_
F: 40	examples_hello_world_cm0plus" case15
Fig. 19.	Two opened debug sessions
Fig. 20.	Auxiliary core application stops at the main
	function
Fig. 21.	Synchronous suspension/resumption of both cores using the multiple selections of
	debug sessions and "Suspend"/"Resume"
	controls
Fig. 22.	Synchronous suspension/resumption of
T 19. 22.	both cores using the "Suspend All Debug
	sessions" and the "Resume All Debug
	sessions" controls
Fig. 23.	Demo build target selection
Fig. 24.	Build the demo application
Fig. 25.	Terminal (PuTTY) configuration
Fig. 26.	Download and Debug button
Fig. 27.	Stop at main() when running debugging
Fig. 28.	Go button
Fig. 29.	Text display of the hello_world demo

Tab. 3.	Revision history	/49

Fig. 30.	Start all cores button	23
Fig. 31.	Hello World from primary core message	
Fig. 32.	"Stop all cores" and "Start all cores" control	
•	buttons	23
Fig. 33.	Launch the Pack Installer	24
Fig. 34.	Build the demo	24
Fig. 35.	Terminal (PuTTY) configurations	
Fig. 36.	Download button	
Fig. 37.	Stop at main() when run debugging	27
Fig. 38.	Go button	
Fig. 39.	Text display of the hello world demo	
Fig. 40.	Hello World from primary core message	29
Fig. 41.	Debugging the auxiliary core application	30
Fig. 42.	Set up MinGW and MSYS	31
Fig. 43.	Complete MinGW and MSYS installation	31
Fig. 44.	Add Path to systems environment	32
Fig. 45.	Convert path to short path	32
Fig. 46.	Add ARMGCC_DIR system variable	33
Fig. 47.	Install CMake	34
Fig. 48.	Launch command prompt	34
Fig. 49.	hello_world demo build successful	35
Fig. 50.	Terminal (PuTTY) configurations	36
Fig. 51.	SEGGER J-Link GDB Server screen after	
	successful connection	37
Fig. 52.	Launch command prompt	37
Fig. 53.	Run arm-none-eabi-gdb	38
Fig. 54.	Text display of the hello_world demo	38
Fig. 55.	hello_world_cm0plus example build	
	successful	39
Fig. 56.	hello_world_cm4 example build successful	40
Fig. 57.	Loading and running the multicore example	41
Fig. 58.	Hello World from primary core message	42
Fig. 59.	MCUXpresso IDE Quickstart Panel	43
Fig. 60.	OpenSDA – CMSIS-DAP/mbed/DAPLink	
-	interface	44
Fig. 61.	OpenSDA – P&E Micro	
Fig. 62.	OpenSDA – J-Link	44
Fig. 63.	P&E Micro OSJTAG	
Fig. 64.	MRB-KW01	44
Fig. 65.	MAINTENANCE drive	47
5		

### Getting Started with MCUXpresso SDK

### Contents

1	Overview	2
2	MCUXpresso SDK board support	
	package folders	2
2.1	Example application structure	3
2.2	Locating example application source files	3
3	Run a demo using MCUXpresso IDE	4
3.1	Select the workspace location	4
3.2	Build an example application	4
3.3	Run an example application	6
3.4	Build a multicore example application	10
3.5	Run a multicore example application	12
4	Run a demo application using IAR	
4.1	Build an example application	
4.2	Run an example application	
4.3	Build a multicore example application	22
4.4	Run a multicore example application	23
5	Run a demo using Keil MDK/µVision	
5.1	Install CMSIS device pack	
5.2	Build an example application	
5.3	Run an example application	24
5.4	Build a multicore example application	28
5.5	Run a multicore example application	
6	Run a demo using Arm GCC	30
6.1	Set up toolchain	
6.1.1	Install GCC Arm Embedded tool chain	30
6.1.2	Install MinGW (only required on Windows	
	OS)	31
6.1.3	Add a new system environment variable for	
	ARMGCC DIR	32
6.1.4	Install CMake	33
6.2	Build an example application	34
6.3	Run an example application	35
6.4	Build a multicore example application	
6.5	Run a multicore example application	
7	MCUXpresso Config Tools	
8	MCUXpresso IDE New Project Wizard	
9	How to determine COM port	
10	How to define IRQ handler in CPP files	
11	Default debug interfaces	
12	Updating debugger firmware	
12.1	Updating OpenSDA firmware	
12.2	Updating LPCXpresso board firmware	
13	Revision history	
	Legal information	50
	~	

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© 2024 NXP B.V.

All rights reserved.

For more information, please visit: https://www.nxp.com

Date of release: 10 January 2024 Document identifier: MCUXSDKGSUG