Getting Started with MCUXpresso SDK for QN9090



Contents

Chapter 1 Overview	3
Chapter 2 MCUXpresso SDK board support package folders	4
2.1 Example application structure	4
2.2 Locating example application source files	5
Chapter 3 Set up toolchain 3.1 Install GCC Arm Embedded tool chain 3.2 Add the new system environment variables 3.3 Install Python3	6
Chapter 4 Run a demo application using IAR.	9
4.1 Build an example application.	9
4.2 Run an example application.	10
Chapter 5 Run a demo using MCUXpresso IDE	13
5.1 Select the workspace location	13
5.2 Build an example application	13
5.3 Run an example application.	17
Appendix A How to determine COM port	24

Appendix B Updating Debugger	firmware26
------------------------------	------------

Chapter 1 Overview

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

For supported toolchain versions, see MCUXpresso SDK Release Notes for QN9090DK6 (document MCUXSDKQN9090RN).

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



Chapter 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

<code>board_name></code> folder, there are various sub-folders to classify the type of examples it contain. 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).
- rtos_examples: Basic FreeRTOSTM OS examples that show the use of various RTOS objects (semaphores, queues, and so on) and interfaces with the MCUXpresso SDK's RTOS drivers
- wireless_examples: Applications that use the bluetooth stack.

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:



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

2.2 Locating example application source files

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

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

- devices/<device name>: The device's CMSIS header file, MCUXpresso SDK feature file and a few other files
- devices/<device_name>/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
- devices/<devices name>/project template: Project template used by MCUXpresso IDE to create new projects
- devices/<devices name>/image tool: Post build scripts used by IAR and MCUXpresso IDE
- devices/<devices name>/project Project template used in CMSIS PACK new project creation

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

Chapter 3 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.

3.1 Install GCC Arm Embedded tool chain

Download and run the installer from developer.arm.com/open-source/gnu-toolchain/gnu-rm. This is the actual toolset (in other words, compiler, linker, and so on). The GCC toolchain should correspond to the latest supported version.

3.2 Add the new system environment variables

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:\GNU Tools Arm Embedded\7 2018-q2-update

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

	Environment Variables
	Edit User Variable X
	Variable name: ARMGCC, DIR Variable value: C:\GRUU Tools Arm Embedded\7 2018-q2-update Browse Directory Browse File OK Cancel
	ے۔ New Edit., Delete
igure 3. Add Armgcc_dir system	variable

Add the bin directory path of the GNU Arm GCC Embedded tools in System variables -> Path. For this example, the path is:

C:\GNU Tools Arm Embedded\7 2018-q2-update\bin

	System variables		
	Variable NUMBER.OF_PROCESSORS OS OUTPUTS_PATH Path PATHEXT PROCESSOR_ARCHITECTURE PROCESSOR_IDENTIFIER	Value 8 Windows_NT C\Workspace\ChipModel\mcu-chipmodel\npi-data\ C\Program Files (x86)N465arangXshell 6xC2>ythoa27\cC\Pytho. COM:LSEB.A7\cDVD.VBS:VEB.255EWSF.WSF.MSCC AMD64 Intel64 Family 6 Model 142 Stepping 10, GenuineIntel NewEditDelete OK Cancel	·
Figure 4. Select Path and click Edit			

	Edit environment variable	×	
		<u></u>	
	C:\Program Files (x86)\Intel\Intel(R) Management Engine Compon ^ C:\MinGW\bin	New	
	C:\Program Files\CMake\bin	Edit	
	C:\Program Files\Intel\Intel(K) Management Engine Components\ C:\Program Files\PuTTY\	Proviso	
	C:\Ruby24-x64\bin	blowse	
	C:\Program Files\TortoiseGit\bin C:\Program Files (x86)\\AR Systems\Embedded Workbench 8 3\co	Delete	
	C\nyn\I PCScn/nt\scrints		
	C\Program Files\doxygen\bin		
	C:\Strawberry\c\bin	Move Up	
	C:\Strawberry\perl\site\bin		
	C:\Strawberry\perl\bin	Move Down	
	C:\Program Files\MiKTeX 2.9\miktex\bin\x64\		
	C:\Workspace\ChipModel\mcu-chipmodel\agen		
	C:\Program Files\Sublime Text 3	Edit text	
	%SYSTEMROOT%\System32\OpenSSH\		
	C:\Program Files\dotnet\		
	C:\Program Files (x86)\Windows Kits\10\Windows Performance To		
	C:\Program Files\Git\cmd		
	C:\GNU Tools Arm Embedded\7 2018-q2-update\bin		
	ОК	Cancel	
Figure 5. Click New and add the path			

3.3 Install Python3

Download and run the Python3 installer from https://www.python.org/downloads/.

Remember to select the Add Python 3.7 to PATH checkbox.

NOTE

Only the version newer than 3.2 is supported while generating the JN518x binaries.

Download and run the VCForPython installer from https://download.microsoft.com.

After the installation, please confirm that the installation path is added into System variables Path. If not, please do it manually.

System variables	
Variable	Value
myWindowsServerPassiveM.	. No
myWorkspaceSwitch	No
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
OUTPUTS_PATH	C:\Workspace\ChipModel\mcu-chipmodel\npi-data\
Path	C:\Program Files (x86)\NetSarang\Xshell 6\;C:\Python37\;C:\Pytho
PATHEXT	.COM;:EXE;:BAT;.CMD;:VBS;:VBE;:JS;:JSE;:WSF;:WSH;:MSC
	- MIDC4
	New Edit Delete
Figure 6. Select Path and click Edit	

	Edit environment variable	×	
	C:\Python37\	^ New	
	C:\Python37\Scripts		
	C:\Program Files (x86)\NetSarang\Xshell 6\	Edit	
	C:\Program Files (x86)\Intel\Intel(R) Management Engine Compon	Curt	
	C:\Program Files\Intel\Intel(R) Management Engine Components\i	Province	
	C:\Program Files (x86)\Common Files\Oracle\Java\javapath	browse	
	%SystemRoot%\system32	Delate	
	%SystemRoot%	Delete	
	%SystemRoot%\System32\Wbem		
	%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\		
	C:\Program Files (x86)\Intel\Intel(R) Management Engine Compon	Move Up	
	C:\MinGW\bin		
	C:\Program Files\CMake\bin	Move Down	
	C:\Program Files\Intel\Intel(R) Management Engine Components\	-	
	C:\Program Files\PuTTY\		
	C:\Ruby24-x64\bin	Edit text	
	C:\Program Files\TortoiseGit\bin		
	C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.50.5\		
	C:\nxp\LPCScrypt\scripts		
	C:\Program Files\doxygen\bin		
	C:\Strawberry\c\bin		
	C\Strawhern\nerl\site\bin	~	
	ОК	Cancel	
	La construction de la construction		
Figure 7 Confirm the path is added			
rigure r. commune paul is added			

Follow the below steps to install the Crypto library for Python 3.

- 1. Press Windows+R to open the **Run** box.
- 2. Type **cmd** and then click **OK** to open a regular Command Prompt.
- 3. Type and run the command: pip3 install pycryptodome.

Chapter 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. The hello_world demo application targeted for the qn9090dk6 hardware platform is used as an example, although these steps can be applied to any example application in the MCUXpresso SDK.

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

Using the qn9090dk6 hardware platform as an example, the hello world workspace is located in:

<install_dir>/boards/qn9090dk6/demo_apps/hello_world/iar/hello_world.eww

Other example applications may have additional folders in their path.

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

For this example, select hello_world - debug.

Workspace	▼ ‡ ×
debug	~
debug	
release	
🖂 🛑 hello_world - deb	ug 🗸
🛛 🖃 🖬 board	
- 🕀 🖬 component	
🛛 🗁 🖬 device	•
🛛 🗁 🖬 doc	
🛛 🗁 🖬 drivers	
🛛 🗁 🖬 source	
🛛 🛏 🖬 startup	
🛛 🛏 🖬 utilities	
🛛 🖵 🖬 Output	
no build target selection	

3. To build the demo application, click Make, highlighted in red in Figure 9.

· · · · · · · · · · · · · · · · · · ·		_	
debug		~	
Files	•	•	
🗆 🌒 hello_world - debug	~		
— ⊞ 🛑 board			
- E CMSIS			
- 🖽 🛋 component			
		•	
- 🗄 🛑 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. Download and install LPCScrypt or the Windows[®] operating systems driver for LPCXpresso boards from www.nxp.com/ lpcutilities. This installs required drivers for the board.
- 2. Connect the development platform to your PC via USB cable between the Link2 USB connector (J2) and the PC USB connector. Ensure JP5 is removed so the Link2 boots from internal flash. If connecting for the first time, allow about 30 seconds for the devices to enumerate.
- 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 #unique_12). Configure the terminal with these settings:
 - a. 115200 or 9600 baud rate, depending on your settings (reference the <code>BOARD_DEBUG_UART_BAUDRATE</code> variable in the <code>board.h</code> file)
 - b. No parity
 - c. 8 data bits
 - d. 1 stop bit

- Session	Basic options for	your PuTTY session					
Logging	Specify the destination you	want to connect to					
- Ieminal	Serial li <u>n</u> e	Speed					
Bell	COM16	115200					
- Features ∋- Window	Connection type: Raw <u>T</u> elnet	Rlogin 🔘 <u>S</u> SH 🔘 Serial					
Appearance Behaviour Translation	Load, save or delete a stor Sav <u>e</u> d Sessions	ed session					
Selection	Debug						
Colours	Default Settings Debug	Load					
Data		Sa <u>v</u> e					
Telnet Rlogin		Delete					
⊕- SSH Serial	Close window on exit: Always Never Only on clean exit						
About	lelo	Open Cancel					

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



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

Vorkspace	•	ψ×	hello_wor	ld.c x
debug		~	main()	
Files Files Files CNSIS CNSIS COMPONENT	÷	•	20 21 22 22 22 24 22 24 22 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 41 42 43 43 41 42 43 43 41 42 43 43 41 42 43 43 41 42 43 43 41 42 43 43 41 42 43 43 41 42 43 43 41 42 43 43 41 42 43 43 41 41 41 41 41 41	<pre>* Prototypes * Code * Code * Code */ # @brief Main function */ int main(void) { char ch; /* Init board hardware. */ /* Security code to allow debug access */ SYSCON->CODESECURITYPROT = 0x87654320; /* attach clock for USART(debug console) */ CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH); /* reset FLEXCOWN for USART */ RESET_PeripheralReset(KFCQ_RST_SHIFT_RSTn); RESET_PeripheralReset(KGPI00_RST_SHIFT_RSTn);</pre>
		_	44 45 46	BOARD_BootClockRUN(); BOARD_InitDebugConsole(); BOARD_InitPins():

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

< Q, >	\$ ►E <	٥	>	<	>	۵	0	•==	G	c	8	Ţ i ſ	، د	t 🗗	N	۰I	• 0	•	• -		Ŧ
Figure 13.	Go button																				

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



Chapter 5 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 QN9090DK6 hardware platform is used as an example, though these steps can be applied to any example application in the MCUXpresso SDK.

5.1 Select the workspace location

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

5.2 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 🛛 🔲 Propert Installed SDKs	iies 📮 Console	Problems	Memory	🚯 Instruction
	To install an SDK, simply drag and	l drop an SDK (zip	file/folder) into	the 'Installed	SDKs' view.
	Name	Version	Location		
Figure 15. Install an SDI	<				

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

MCUXpresso IDE (Free Edition)	Â
▼ Start here	
New project	E
Import SDK example(s)	
Import project(s)	
🐔 Build '' 🗋	
Clean " []	
🍄 Debug '' []	
🔯 Terminate, Build and Debug " []	
Edit " project settings	
Quick Settings>>	
Export projects to archive (zip)	-

3. In the window that appears, expand the QN909x folder and select QN9090 . Then, select qn9090dk6 and click Next.

SDK Import Wizard						×
Importing project(s) for devi	ce: QN9090 using board: QN9090DK6					5
Board and/or Devi	ice selection page					
▼ SDK MCUs	Available boards				↓ <mark>a</mark> _ta <mark>z</mark>	a.
MCUs from installed SDKs	Please select an available board for your project	t.				
NXP QN9090 QN9090 QN9090	Supported boards for device: QN9090					
Selected Device: QN9090 u	sing board: QN9090DK6	SDKs for selected MCU	J. Construction			
Target Core: cm4		Name	SDK Vers	Manifest	Location	
Description: The QN909 embedded	00 are ARM Cortex-M4 based microcontrollers for applications.	⊕ SDK_2.x_QN9090D	2.6.0	3.5.0	Common>\QN9090DK	
?		< Back	Next >	Fi	inish Cancel	
Figure 17. Select QN9	090DK6 board					

4. Expand the ${\tt demo_apps}$ folder and select ${\tt hello_world}$. Then, click Next .

🔀 SDK Import Wizard			- 🗆 X
A The source from the SDK will be copied into the work of you want to use linked files, please unzip the 'SDK	rkspace. _2.x_QN9090DK6' SDK.		
Import projects			
Project name prefix: qn9090dk6		2 Project name suffix:	Ø.
☑ Use default location			
Location: C:\Users\nxf49881\Documents\MCUXpre	ssoIDE_11.0.0_2486_alpha	workspace\qn9090dk6	Browse
Project Type		Project Options	
● C Project ○ C++ Project ○ C Static Library ○	C++ Static Library	SDK Debug Console 〇 Semihost 🖲 🛛 Copy sources Import other files	ART Example default
Examples			ѐ 🖉 🗹 🙀 🕀 🖻
type to filter			
Name	Description		Version
 ■ e demo_apps □ ≡ hello_world □ ≡ led_blinky > □ ≡ driver_examples > □ ≡ rtos_examples 	The Hello Wo	orld demo application provides a sanity check for ky demo application provides a sanity check for	r the new SDK buil the new SDK build
0		< Back Next >	Finish Cancel
Figure 18. Select hello_world			

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_interrupt. Otherwise, it is not necessary to select this option. Then, click Finish.

DK Import Wizar	rd					
						P
Advanced S	Settings					
C/C++ Library S	Settings					
Set library type (a	and hosting variant) Redlib (no	ohost-nf)	~			
Redlib: Use floa	ating point version of printf		New	libNano: Use floating p	pint version of printf	
Redlib: Use cha	racter rather than string based	l printf	New	libNano: Use floating po	pint version of scanf	
Redirect SDK "F	PRINTF" to C library "printf"		Redi	rect printf/scanf to ITM		
Include semiho	ost HardFault handler		Redi	rect printf/scanf to UAR	Т	
Hardware settin	ngs					
et Floating Point	type None					\sim
MCU C Compile	er					
anguage standar	d GNU C99 (-std=gnu99)					~
MCII Linker						
Link application	n to RAM					
Memory Confid	nuration					
lemory details						
 Default LinkServe	er Flash Driver					Browse
Туре	Name	Alias	Location	Size	Driver	
Flash	PROGRAM_FLASH	Flash	0x0	0xa0000	QN9090_640K.cfx	8
RAM	SRAM	RAM	0x4000000	0x15fe0		1201
KAM	SRAM1	RAMZ	0x4020000	0x10000		
Add Flash Add	RAM	Split Join Delete	e Imr	oort Merge Expor	t Generate	
			c Pack	Next ~	Finish	Cancel
)			> DOLK	INCXL >	1 1111511	Calicel

5.3 Run an example application

The application can be downloaded into DK6 QN9090/K32W061 module either:

- via the LPC-LINK2 USB port using an IDE debugger (MCUXpresso or IAR)
- via UART0 using the DK6 Flash Programmer



UART0 can be routed to several ports of the DK6:

- the LPC-LINK2 USB port (JP4 and JP7 in the leftmost position as shown in Figure 20)
- the FTDI USB connector (JP4 and JP7 in the middle position)
- J3 connector pins 8 and 9 using a USB-to-Serial converter (e.g. Prolific model obtainable from nxp.com)

UART0 is user port and UART1 instance is a debug console. The debug console can be accessed via J3 pins 10 and 11 using a USB-to-Serial converter. The easiest way to utilize BLE applications on the DK6 board is to plug a mini-USB cable into LPC-LINK2 USB connector. This acts as a power source and UART interface that can be used to download the binaries. Refer to the *DK6-UG-3127-Production-Flash-Programmer.pdf* for additional information.

To download and run the application, perform these steps:

- Connect the development platform to your PC via USB cable between the Link2 USB connector (named Link for some boards or LPC-LINK2 or QN9090/K32W061 boards) and the PC USB connector. If connecting for the first time, allow about 30 seconds for the devices to enumerate.
- 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 How to determine COM port). Configure the terminal with these settings:
 - a. 115200 or 9600 baud rate, depending on your board (reference <code>BOARD_DEBUG_UART_BAUDRATE</code> variable in the <code>board.h</code> file)
 - b. No parity
 - c. 8 data bits
 - d. 1 stop bit

 Session Logging Terminal Keuboard 	Basic options for your PuTTY	session		
	Specify the destination you want to connect to			
	Serial line	Speed		
Bell	COM16	115200		
Features Window Appearance Behaviour Translation Selection Colours Connection	Connection type: Ra <u>w</u>	SH 💿 Se		
	Load, save or delete a stored session Saved Sessions			
	Debug			
	Default Settings	Load		
Data		Saye		
Telnet Rlogin ⊕ SSH Serial		Delet		
	Close window on exit: Always Never Only or	n clean exit		

3. On the Quickstart Panel, click Debug.

U Quickstart Panel 🕮 🗱 Variables 🎙	Breakpoints
MCUXpresso IDE - Qu Project: qn9090dk6_hello_worl	ickstart Panel
 Create or import a project 	
New project New project Import SDK example(s) Import project(s) from file s	ystem
 Build your project 	
Clean	
- Debug your project	🔝 🔻 🔛 👻 💌
 Terminate, Build and Debug Terminate, Build and Debug Debug project (Press Si 	HIFT to force probe re-discovery)
Edit project settings	
MCUXpresso Config Tools>>	
Quick Settings>>	~
🗳 qn9090dk6_hello_world	
Figure 22. Debug hello_world case	

4. 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.)

Av	ailable attached prol	pes			
	Name	Serial number/ID	Туре	Manu	IDE Debug Mode
X	LPC-LINK2 CMSIS-DAP V5.1	NQA5AQMQ	LinkServ	NXP Ser	Non-Stop
Su	pported Probes (tick/untick to	enable/disable)			
V	MCUXpresso IDE LinkServer	(inc. CMSIS-DAP) pr	robes		
V	P&E Micro probes				
V	SEGGER J-Link probes				
Pr	obe search options				
S	earch again				
	emember my selection (for th	is Launch configura	tion)		
/	· · · · ·				

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



6. Start the application by clicking Resume.



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



Appendix A 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, on-board debug interface LPC Link2.

1. To determine the COM port, open the Windows operating system Device Manager. This can be achieved by going to the Windows operating system Start menu and typing **Device Manager** in the search bar, as shown in Figure 27.

ſ	Control Panel (3)
	🚔 Device Manager
	View devices and printer Device Manager View and update your hardware's settings and driver s
	Pictures (9)
l	Companies.inc hut.inc
I	PTPStillImageTables.inc
	VIDs_PIDs.TXT
	SCSI_CDB_RcvCpyRslts.inc
	SCSI_CDB_SPC.inc
	hci_command_table.inc
	CDCRequests inc
	Files (1)
	dialog_settings.xml
	₽ See more results
	Device Manager × Shut down +
27	. Device manager

2. In the Device Manager, expand the Ports (COM & LPT) section to view the available ports.

Appendix B Updating Debugger firmware

The hardware platform comes with a CMSIS-DAP-compatible debug interface that has the ability to update the debugger firmware. This typically means switching from the default application (CMSIS-DAP) to a SEGGER J-Link. This section contains the steps to switch the CMSIS-DAP firmware to a J-Link interface. However, the steps can also be applied to restoring the original image.

NXP provides the LCPScrypt 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 www.nxp.com/lpcutilities.

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 (www.nxp.com/lpcutilities, select LPCScrypt, then select documentation tab).

- 1. Unplug the board's USB cable.
- 2. Install the LPCScript utility.
- 3. For LPCXpresso board: make DFU link (install the jumper labelled DFUlink (JP5)).
- 4. Connect the probe to the host via USB (use Link USB connector (J2)).
- 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 Step 3).
- 7. Re-power the board by removing the USB cable and plugging it again.

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

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

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. 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 PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, ICODE, JCOP, LIFE, VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, CodeWarrior, ColdFire, ColdFire+, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, Tower, TurboLink, EdgeScale, EdgeLock, elQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2020.

All rights reserved.

For more information, please visit: http://www.nxp.com For sales office addresses, please send an email to: salesaddresses@nxp.com

> Date of release: December 10, 2020 Document identifier: MCUXSDKQN9090GSUG

arm