NXP Semiconductors Application Note

QCA4002/4 KSDK Porting Guide

1 Introduction

QCA4002/4 MCUXpresso Software Development Kit (MCUXpresso SDK) is available for various NXP evaluation boards and currently supports two Wi-Fi modules, Arrow GT202 and Silex SX-ULPAN. Please refer to Arrow or Silex for further information on their modules.

This guide shows how to start from an existing MCUXpresso SDK and adapt it to another Kinetis Freedom board not supported in the SDK. It was designed to facilitate the migration between boards or for custom hardware, and written focused on the MCUXpresso Config tools.

2 GT202 and SX-ULPAN

2.1 Boards Description

These boards focus on Internet of Things (IoT) applications and are based on Qualcomm Wi-Fi System-on-Chip (SoC) QCA4002 for GT202 and QCA4004 for SX-ULPAN. QCA4004 is a dual band (2.4 GHz and 5.8 GHz) Wi-Fi device and QCA4002 is a single band (2.4 GHz) solution.

Both the boards have UART and SPI connections. A UART is usually used for a quick internet connection using simple data transfers. SPI communication allows greater flexibility and is an advanced connection.

The MCUXpresso SDK supports only a SPI connection.

Contents

1	Intro	duction	1
2	GT2	02 and SX-ULPAN	1
	2.1	Boards Description	1
	2.2	Hardware Connections	2
	2.3	Freedom Board Support for Wi-Fi Shield	2
	2.4	MCUXpresso SDK Support for QCA4002/4	3
3	MCU	JXpresso Config Tools	4
	3.1	How to get MCUXpresso Tools	4
	3.2	Install and initial steps	4
4	MCU	JXpresso SDK Migration	6
	4.1	Selecting devices and copying files	6
5	Porti	ng Example with MCUXpresso Config Tools	9
	5.1	Overview of MCUXpresso Config Tools	9
	5.2	MCUXpresso SDK Standalone Porting	10
	5.3	MCUXpresso SDK-Linked Porting	27
	5.4	Considerations for Other MCUXpresso SDKs.	30



2.2 Hardware Connections

The QCA4002 modules and development boards must be connected using at minimum the following signals:

Signal Name	Function	GT202	Silex
CS	SPI CS	J2.6	J2.6
SCK	SPI SCK	J2.12	J2.12
MOSI	SPI SOUT	J2.8	J2.8
MISO	SPI SIN	J2.10	J2.10
PWRON	GPIO (out)	J2.2	J1.10
IRQ	GPIO (in with IRQ)	J1.16	J1.6

Table 1.	Signal and FRDM Pinout
----------	------------------------

The pin configuration should be in accordance with the Table 1 while configuring a new board. However, a rework may be required for some boards.

2.3 Freedom Board Support for Wi-Fi Shield

Pinout can change between the development boards. Some functions and peripherals are specific to microcontrollers.

Usually SPI are available in shield connectors and no modification is necessary. However, in microcontrollers with more than one SPI, it is necessary to check the module getting connected to QCA shield-specific pins.

For GPIOs, J1.16 and J2.2 are not standard pins. In some Freedom boards these pins cannot be used as is. For example, in FRDM-KW41Z, PTA0 and PTA1 are available in these pin positions. As PTA0 and PTA1 are also shared with SDA debug functions, it will have high impact the development of final application. So, in some cases, simple PCB rework is necessary.

For this application note, the two boards used are FRDM-KL43Z and FRDM-K66F. For FRDM-K66F, no rework is required, but for FRDM-KL43Z, a rework is required. In FRDM-KL43Z, J1.16 must be disconnected from PTE30, and another IRQ capable signal should be routed to J1.16. For this application note, a solder jump between J1.16 and J1.15 was made, as shown in Figure 1:

GT202 and SX-ULPAN



Figure 1. FRDM-KL43Z rework

This connects the IRQ pin from QCA4002/4 module to pin PTC5 of FRDM-KL43Z, which is interrupt capable.

2.4 MCUXpresso SDK Support for QCA4002/4

During porting activities, it is necessary to choose the base MCUXpresso SDK. To reduce the porting workload, some components should be considered for this decision. The most important in this application note are:

- SPI
- DMA
- UART

NXP microcontrollers have certain combinations of peripherals in each family/subfamily of products. For example, SPI can be SPI, DSPI, and LPSPI. They have different features and performance.

Currently, QCA4002/4 middleware and examples are available in MCUXpresso SDK 2.x for or the following boards:

- FRDM-K22F
- FRDM-K64F
- FRDM-K82F
- FRDM-KL28Z
- FRDM-KL46Z

To obtain the MCUXpresso SDK for these platforms, check QCA4002 quick start guide Section 2.2.

3 MCUXpresso Config Tools

3.1 How to get MCUXpresso Tools

With MCUXpresso config tools it is possible to port MCUXpresso SDK drivers and examples to another Freedom board or to a custom board.

The MCUXpresso tools can be used remotely, via web version, or locally, or via desktop application. In this document, only the desktop version is covered. To access MCUXpresso tools on the web, login at mcuxpresso.nxp.com/en/welcome and select the required tool.



Figure 2. MCUXpresso Config Tools online version

To download the desktop version, select your platform-specific application at <u>www.nxp.com/products/software-and-tools/run-time-software/mcuxpresso-software-and-tools/mcuxpresso-config-tools:MCUXpresso-Config-Tools</u>.

3.2 Install and initial steps

1. After downloading MCUXpresso Config Tools, install the executable file for your platform. Select 'Pins and Clock Tool', and select Java, as shown in Figure 3.

Kinetis Expert Tools Setup	X						
Custom Setup Select the way you want features to be installed.							
Click on the icons in the tree below to change the	way features will be installed.						
Kinetis Expert Tools Kinetis Expert Tools F Pins & Clocks Tool F JRE 1.8	Kinetis Expert Tools						
	This feature requires 336KB on your hard drive. It has 2 of 2 subfeatures selected. The subfeatures require 239MB on your hard drive.						
Location: C:\nxp\KEx_v2\	Br <u>o</u> wse						
Advanced Installer Disk <u>U</u> sage	<u>N</u> ext > Cancel						

Figure 3. Installing MCUXpresso Config tools

- 2. Click the 'Next' button until installation ends.
- 3. To execute MCUXpresso, access the Start Menu (or corresponding location of your system) and open *NXP*→*MCUXpresso Config Tools* as shown in Figure 4.



Figure 4. MCUXpresso for Windows® OS 7

4 MCUXpresso SDK Migration

The SDK can be downloaded and migrated to any board. Some MCUXpresso SDK require less effort to migrate than others depending on the peripherals, selected microcontroller and IP differences between source and target devices.

4.1 Selecting devices and copying files

In this first example, a migration to FRDM-KL43Z25 is made. On the MCUXpresso website, download FRDM-KL43Z SDK, then select FreeRTOS from the drop-down menu.

 FRDM-KL46 is the reference MCUXpresso SDK, so if it is not available locally, download the FRDM-KL46Z SDK. Select FreeRTOS then QCA400x Wi-Fi, as shown in Figure 5 and Figure 6.

NXP MCL	JXpresso overview	TOOLS - MANAGE -	•
	<u> </u>	ettings	
\smile	Select All Deselect All	cros selections, and development preferences.	
	Middleware CMSIS DSP Library	Hardware Details	
	FatFS 🗸	Board	FRDM-KL46Z
			MKL46Z4
	USB stack emWin IwIP	Vies projects included in the SDK Download and Generated Projects thain / IDE toolchaipe	256 KB Flash 32 KB RAM
	mbedtls wolfssl		
	Operating systems FreeRTOS	K download, generated projects, and will impact Peripheral Tool settings	
	5 items selected	Selected Middleware	
×		CMSIS DSP Library, FatFS, QCA400x WiFi, USB stack, FreeRTOS	
Feedbac	Actum to Overview	Go to SDK Builder Jump start your configuration	

Figure 5. FRDM-KL46Z SDK download

- 2. After downloading both MCUXpresso SDKs, unzip them to a selected local folder. In examples, they are:
- {YOUR PATH}\SDK_2.2_FRDM-KL46Z
- {*YOUR PATH*}*SDK*_2.2_*FRDM*-*KL*43Z

Copy {YOUR PATH}\SDK_2.2_FRDM-KL46Z\boards\frdmkl46z\demo_apps\wifi_qca into {YOUR PATH}\SDK_2.2_FRDM-KL43Z\boards\frdmkl43z\demo_apps.

Include in library 🔻	Share with 🔻	New folder
L adc16_low_power		
Ladc16_low_power_a	sync_dma	
📙 bubble		
📙 dac_adc		
📙 flexio_pwm		
hello_world		
bower_manager		
L power_mode_switch	า	
📙 rtc_func		
📙 shell		
📙 wifi gca		

Figure 6. MCUXpresso SDK FRDM-KL43Z examples folder after copy

3. Also, copy *{YOUR PATH}\SDK_2.2_FRDM-KL46Z\middleware\wifi_qca_2.0.0* to *{YOUR PATH}\SDK_2.2_FRDM-KL43Z\middleware* as shown in Figure 7.



Figure 7. MCUXpresso SDK FRDM-KL43Z middleware folder after copy

- 4. After copying both the folders, go to *{YOUR PATH}\SDK_2.2_FRDM-KL43Z\middleware\wifi_qca_2.0.0\port\boards* and rename the folder frdmkl46z to frdmkl43z.
- 5. In the main folder of FRDM-KL43Z, find the file called FRDM-KL43Z_manifest.xml and open it in a text editor. Open the FRDM-KL46Z_manifest.xml file from the FRDM-KL46Z folder as well.
- 6. Include the *qca_demo* example and corresponding source files in the new MCUXpresso SDK. Copy the *qca_demo* example from FRDM-KL46Z to FRDM-KL43Z. Copy it after last example in category "demo_apps". It must be copied from *<example* ... to the following *<\example>*.



Figure 8. Adding example to manifest file

7. Copy all components related to wifi_qca. Copy after last "middleware" group of files.

Each component starts at *<component* ... and finish at *<\component>*.

1372	<pre>comp</pre>	nent id="middleware.	multicore.erpc.MKL43Z4" name="erpc" full name="Embedded Remote Procedure Call" type="other" brief="eRPC" dependency="middleware.multicore.erpc.eRPC arb;
3183	the <compo< th=""><th>nent id="middleware.</th><th>wifi qca.MKL46Z4" name="wifi qca" type="other" dependency="middleware.freertos.MKL46Z4 middleware.template_application.freertos.MKL46Z4 middleware.wifi</th></compo<>	nent id="middleware.	wifi qca.MKL46Z4" name="wifi qca" type="other" dependency="middleware.freertos.MKL46Z4 middleware.template_application.freertos.MKL46Z4 middleware.wifi
3337	<compo< th=""><th>nent id="middleware.</th><th>wifi gca.board.frdmkl46z.gt202.MKL46Z4" name="gt202" type="other" dependency="middleware.freertos.MKL46Z4 middleware.template_application.freertos.MKL4</th></compo<>	nent id="middleware.	wifi gca.board.frdmkl46z.gt202.MKL46Z4" name="gt202" type="other" dependency="middleware.freertos.MKL46Z4 middleware.template_application.freertos.MKL4
3351	<compo< th=""><th>nent id="middleware.</th><th>wifi qca.driver.spi dma freertos.MKL46Z4" name="spi dma freertos" type="other" dependency="middleware.freertos.MKL46Z4 middleware.template application.:</th></compo<>	nent id="middleware.	wifi qca.driver.spi dma freertos.MKL46Z4" name="spi dma freertos" type="other" dependency="middleware.freertos.MKL46Z4 middleware.template application.:
3359	d <comp< th=""><th>nent id="middleware.</th><th>wifi gca.env.freertos.MKL46Z4" name="freertos" type="other" dependency="middleware.freertos.MKL46Z4 middleware.template application.freertos.MKL46Z4 mid</th></comp<>	nent id="middleware.	wifi gca.env.freertos.MKL46Z4" name="freertos" type="other" dependency="middleware.freertos.MKL46Z4 middleware.template application.freertos.MKL46Z4 mid
3369	<compo< th=""><th>nent id="tools.MKL43</th><th>Z4" name="tools" type="other" devices="MKL432256xxx4" version="1.0.0"></th></compo<>	nent id="tools.MKL43	Z4" name="tools" type="other" devices="MKL432256xxx4" version="1.0.0">
3375	d <compo< th=""><th>nent id="middleware.</th><th>dma_manager.NKL4324" name="dma_manager" type="middleware" dependency="platform.Include_common platform.Include_core_cm0plus platform.devices.NKL4324_CM</th></compo<>	nent id="middleware.	dma_manager.NKL4324" name="dma_manager" type="middleware" dependency="platform.Include_common platform.Include_core_cm0plus platform.devices.NKL4324_CM
2202		and the base of the second sec	And some stands were stand and and and and and and and and and

Figure 9. Adding example components to manifest file

Four components should be copied:

- middleware.wifi_qca.MKL46Z4
- middleware.wifi_qca.board.frdmkl46z.gt202.MKL46Z4
- middleware.wifi_qca.driver.spi_dma_freertos.MKL46Z4
- middleware.wifi_qca.env.freertos.MKL46Z4
- 8. Using a text editor, open *qca_demo.xml* and *example.xml*. They are located at *{YOUR PATH}\SDK_2.2_FRDM-KL43Z\boards\frdmkl43z\demo_apps\wifi_qca\qca_demo*.
- 9. Now, replace every FRDM-KL46Z reference for the corresponding FRDM-KL43Z equivalent.

For KL46Z, the strings replacements are as follows:

- frdmkl46z to frdmkl43z
- FRDM_KL46Z to FRDM_KL43Z
- MKL46Z4 to MKL43Z4
- MKL46Z256VLL4 to MKL43Z256VLH4
- MKL46Z256xxx4 to MKL43Z256xxx4

Now, MCUXpresso Config Tools can be used to configure the new board.

5 Porting Example with MCUXpresso Config Tools

5.1 Overview of MCUXpresso Config Tools

The MCUXpresso Config Tools is an integrated suite of configuration tools that helps guide users from first evaluation to production software development when designing with NXP's microcontrollers based on ARM[®] Cortex[®]-M cores, including LPC and Kinetis MCUs. Available in both online and desktop editions, these tools allow developers to quickly build a custom MCUXpresso SDK, leverage pins, clocks and peripheral tools to generate initialization C code for custom board support and estimate system power consumption and battery life.



Figure 10. MCUXpresso tools

MCUXpresso can be used online or offline. For this tutorial, offline version is used.

5.2 MCUXpresso SDK Standalone Porting

5.2.1 Opening MCUXpresso SDK Qca_demo Example

For a Standalone project, use the full set of offline MCUXpresso Config Tools:

- Pins
- Clocks
- Project Generator
- 1. Open MCUXpresso Config Tools in start menu, as shown in Figure 11.



Figure 11. Windows OS start menu

2. Select the required MCUXpresso SDK to use:

Create a new configuration	
Select SDK Package	
Start development with the selected MCUXpresso SDKv2 Package (SDK can be obtained)	d at <u>http://mcuxpresso.nxp.com</u>)
Select SDK folder: C:\NXP\KSDK\SDK_2.2_FRDM-KL43Z	
Create new configuration	
Use this option to create empty configuration for selected processor/board/kit/ten	plate or create configuration from existing SDK example project.
Select this option if you want example project with all sources for selected toolchai	n. The project will not be editable using MCUXpresso Config Tools.
Start development without an MCUXpresso SDK Package	
Use this option if you have not downloaded an SDK package yet. Tool will be limited t	o only Pins and Clocks Tools. It is possible to specify SDK path later.
< <u>E</u>	ack Next > Einish Cancel

Figure 12. Selecting MCUXpresso SDK

3. If all previous steps are completed, the *wifi_qca/qca_demo* example appears under the *Demo_Apps* menu. Select it. Click the 'OK' button in the pop-up informing some pins and clock are wrongly set.

5.2.2 Setting up Clock

Usually when porting applications between different microcontrollers, clock needs to be fixed.

1. To set up new clock configuration on FRDM-KL43Z, select the boot clock to set. Click on either **BOARD_BootClockRUN** or **BOARD_BootClockVLPR**. Select **BOARD_BootClockRUN**.

	III	4	•	III	•				
Run	Mode RUN MCG Lite Mode High-frequency	/ Internal Referer	nce Clock 4	8MHz 🔻					
No	No probler RUN ad								
BO	BOARD_BootClockRUN X BOARD_BootClockVLPR								

Figure 13. Selecting boot clock

2. It is necessary to fix all the wrong parameters here. In this case, enable HIRC to get a 48 MHz clock. After this, the parameters display what is shown in Figure 12.

🗉 Details 🛛 🕻 Sources 👪	Register	rs OModule Clocks	Log	
Path Details: Core_clock				
Name	A., L.,	Value		Acy
Core clock		48 MHz		±0.1%
CORECLK Frequency		48 MHz		
□ OUTDIV1		/1		
OUTDIV1 Frequency		48 MHz		
MCGOUTCLK Frequency		48 MHz		
CLKS		HIRC clock		
□ HIRC		48 MHz		
HIRC enable		Enabled		



The final configuration for RUN clock is shown in Figure 15.





- 亘 Details 🛛 🔂 Sources 🔠 Registers 🗘 Module Clocks 📄 Log Path Details: Core_clock A., L., Value Ac...y Name 🔒 <mark>4 MHz</mark> ±0.1% Core clock 4 MHz CORECLK Frequency □ OUTDIV1 / 2 4 MHz **OUTDIV1** Frequency MCGOUTCLK Frequency 8 MHz LIRC clock CLKS □ LIRC DIV1 /1 LIRC DIV1 Frequency 8 MHz □ LIRC 8 MHz Internal ref...lock output Enabled Internal re... Stop mode Disabled
- 3. Now, select **BOARD_BootClockVLP**, and change clock source to LIRC in Core clock.

Figure 16. Set VLPR Core clock

*Clocks - qca_demo.mex (N	/KL43Z256xxx4)						
File Edit Tools Clocks Vi	iews Help						
🗄 Clocks Table 🛛 🕻 Clocks	Diagram						- 8
Clock Sources			C	lock Outputs			
Name	Available	Value	1	Name	L	Value	A
Internal			E	System			
□ HIRC		Inactive	L+	Core clock	a	4 MHz	±
HIRC enable		Disabled	÷	System clock		4 MHz	
H LIRC		8 MHz	↦	Bus clock		800 kHz	
LPO		1 kHz	÷	Flash clock		800 kHz	
External			E	Peripheral			
OSC (System Oscillator)		Inactive		MCGPCLK		Inactive	
OSC mode		Using oscillator wi	4	MCGIRCLK		8 MHz	
Frequency Range		High frequency rang		OSCERCLK		Inactive	
System Osity Load		0 pF		ERCLK32K		1 kHz	
USB clock input		Inactive		RTC_CLKOUT		Inactive	
RTC_CLK external input		Inactive		LPO clock		1 kHz	
				USB FS clock		Inactive	
				CLKOUT		Inactive	
				LPUART0 clock		Inactive	
				LPUART1 clock		Inactive	
				TPM clock		Inactive	
				FLEXIO clock		Inactive	
				COP clock		Inactive	

Figure 17. Final VLPR clock config

5.2.3 Setting up Pins

New pin configuration is necessary when changing microcontroller family or package types in the same family of products. Sometimes microcontrollers are pin-to-pin compatible, but some peripherals are different at the same pin. They have the same function but they are not entirely software compatible. Therefore, some minor changes are necessary. Check pins during migration.

As an example, from FRDM-KL46Z to FRDM-KL43Z, PTA1 and PTA2 can be connected to UART. However, in FRDM-KL46Z they are UART and in FRDM-KL43Z they are LPUART. The functionality is same for most applications but it is mandatory to associate the pins in Pins Tool.

To continue the migration process, select 'Pins' in the 'Tools' menu:

*Clocks	- qca_demo.mex (MKL43Z256xxx4)	
File Edit	Tools Clocks Views Help	
E Clocks T	Clocks	
Clock Sour	Pins	
Name	8 Project Generator	Value
□ Internal		
⊟ HIRC		Inactive

Figure 18. Move to pins configuration

Then, configure all the pins as shown in Figure 19:

Rout	ed Pins											- 0
type fi	lter text											
Route	ed Pins 🔘	2 🖸										
#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Drive strength	Pull select	Pull enable	Passive filter	
23	LPUART0	RX	LPUART0_RX	J1[2]/D0/UART	DEBUG	Input	Slow	Low	Pullup	Disabled	Disabled	
24	LPUART0	TX	LPUART0_TX	J1[4]/D1/UART	DEBUG	Not Spec	Slow	Low	Pullup	Disabled	Disabled	
BOAR	D_InitPins	BOARD	InitGT202Shield	BOARD_InitSilex24	101Shield	0						

Figure 19. Init pins

1 Pins											
er text											
Pins 🕒	6	<u>~ ~</u>									
Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Drive strength	Pull select	Pull enable	Passive filter	
GPIOA	GPIO, 13	PTA13	PWRON	PWRON	Output	Slow	Low	Pulldown	Enabled	Disabled	
GPIOC	GPIO, 5	PTC5	IRQ	IRQ	Input	Fast	Low	Pullup	Enabled	Disabled	
SPI1	PCS0	SPI1_SS	CS	n/a	Not Spec	Fast	Low	Pullup	Disabled	Disabled	
SPI1	SCK	SPI1_SCK	SCK	n/a	Not Spec	Fast	Low	Pullup	Disabled	Disabled	
SPI1	MOSI	SPI1_MOSI	MOSI	n/a	Not Spec	Fast	Low	Pullup	Disabled	Disabled	
SPI1	MISO	SPI1_MISO	MISO	n/a	Not Spec	Fast	Low	Pullup	Disabled	Disabled	
InitPins	BOARD_I	nitGT202Shield	BOARD_I	nitSilex2401	Shield O						
	Pins Pins Peripheral SPIOA SPIOC SPI1 SPI1 SPI1 SPI1 SPI1 SPI1 SPI1 SPI1	Pins C 6 C Peripheral Signal GPIOA GPIO, 13 GPIOC GPIO, 5 SPI1 PCS0 SPI1 SCK SPI1 MOSI SPI1 MISO InitPins BOARD_I	Pins 6 Peripheral Signal Route to SPIOA GPIO, 13 PTA13 SPIOC GPIO, 13 PTA13 SPIOC GPIO, 5 PTC5 SPI1 PCSO SPI1_SS SPI1 SCK SPI1_SCK SPI1 MOSI SPI1_MOSI SPI1 MISO SPI1_MISO	Pins 6 A A Peripheral Signal Route to Label SPIOA GPIO, 13 PTA13 PWRON SPIOC GPIO, 5 PTC5 IRQ SPI1 PCS0 SPI1_SS CS SPI1 SCK SPI1_SCK SCK SPI1 MOSI SPI1_MOSI MOSI SPI1 MISO SPI1_MISO MISO SPI1 MISO SPI1_MISO MISO SPI1 BOARD_InitGT202Shield BOARD_I	Pins 6 ▲ ▼ Peripheral Signal Route to Label Identifier SPIOA GPIO, 13 PTA13 PWRON PWRON SPIOC GPIO, 5 PTC5 IRQ IRQ SPI1 PCS0 SPI1_SS CS n/a SPI1 SCK SPI1_SCK SCK n/a SPI1 MOSI SPI1_MOSI MOSI n/a SPI1 MISO SPI1_MISO MISO n/a SPI1 BOARD_InitGT202Shield BOARD_InitSilex24015	Pins 6 Image: Second seco	Pins 6 Image: Second seco	Pins 6 Image: Signal integration of the strength	Pins 6 Image: Signal Route to Label Identifier Direction Siew rate Drive strength Pull select SPIOA GPIO, 13 PTA13 PWRON PWRON Output Slow Low Pullup SPIOC GPIO, 5 PTC5 IRQ IRQ Input Fast Low Pullup SPI1 PCS0 SPI1_SS CS n/a Not Spec Fast Low Pullup SPI1 Pullup SPI1 SPI1 SCK SPI1_SCK SCK n/a Not Spec Fast Low Pullup SPI1 PUILup SPI1 SPI1 MOSI SPI1_SCK SCK n/a Not Spec Fast Low Pullup SPI1 PUILup SPI1 SPI1 MOSI SPI1_MOSI MOSI n/a Not Spec Fast Low Pullup SPI1 SPI1 MISO SPI1_MISO MISO n/a Not Spec Fast Low Pullup SPI1 SPI1 MISO SPI1_MISO MISO n/a Not Spec Fast Low Pullup SPI1 SPI1 MISO SPI1_MISO MISO n/a Not Spec Fast Low Pullup SPI1 SPI1 MISO SPI1_MISO MISO n/a Not Spec Fast Low Pullup SPI1 SPI1 MISO SPI1_MISO Intereet Spece Spiter Spiteret Spece Spece Spiteret Spece Spiteret Spec	Pins 6 Image: Constraint of the second	Pins 6 Image: Signal Signal Route to Label Identifier Direction Slew rate Drive strength Pull select Pull enable Passive filter Peripheral Signal Route to Label Identifier Direction Slew rate Drive strength Pull select Pull enable Passive filter Peripheral Signal Route to Label Identifier Direction Slew rate Drive strength Pull select Pull enable Passive filter Peripheral Signal Route to Label Identifier Direction Slew rate Drive strength Pull select Pull enable Passive filter Peripheral Signal Route to Label Identifier Direction Slew rate Drive strength Pull select Pull enable Disabled Disabled SPIOA GPIO, 13 PTA13 PWRON PWRON Output Slow Low Pullup Enabled Disabled Disabled SPIOC GPIO, 5 PTC5 IRQ IRQ Input Fast Low Pullup Disabled Disabled Disabled SPI1 PCS0 SPI1_SC CS n/a Not Spec Fast Low Pullup Disabled Disabled SPI1 MOSI SPI1_MOSI MOSI n/a Not Spec Fast Low Pullup Disabled Disabled Disabled SPI1 MISO SPI1_MISO MISO n/a Not Spec Fast Low Pullup Disabled Disabled <td< td=""></td<>

Figure 20. Specific init pins

_												
pe fil	ter text											
Route	ed Pins 🔘	6	<u>~</u>									
#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Drive strength	Pull select	Pull enable	Passive filter	
26	GPIOA	GPIO, 4	PTA4	PWRON	PWRON	Output	Slow	Low	Pulldown	Enabled	Disabled	
60	GPIOD	GPIO, 3	PTD3	IRQ	IRQ	Input	Slow	Low	Pullup	Enabled	Disabled	
61	SPI1	PCS0	SPI1_SS	CS	n/a	Not Spec	Fast	Low	Pullup	Disabled	Disabled	
62	SPI1	SCK	SPI1_SCK	SCK	n/a	Not Spec	Fast	Low	Pullup	Disabled	Disabled	
63	SPI1	MOSI	SPI1_MOSI	MOSI	n/a	Not Spec	Fast	Low	Pullup	Disabled	Disabled	
64	SPI1	MISO	SPI1_MISO	MISO	n/a	Not Spec	Fast	Low	Pullup	Disabled	Disabled	

Figure 21. Silex2401 specific init pins

5.2.4 Project Generator

1. After completing the Pins Tool, start from the 'Tools' menu and select 'Project Generator'.



Figure 22. Selecting Project Generator

- 2. Project Generator copies all necessary files to run the *qca_demo* example to a selected directory. Enter the folder name at "Base Project directory" or browse for it, by clicking the 'Browse' button.
- 3. Select a toolchain from the available option. For this example, IAR Workbench is used.

4. Fill the project name in the "Project name" field. This is the name of project for all IAR generated files.

Project Gene	rator			- 8	Project Configuration
SDK Path:					Project name:
C:\NXP\KSDK	\SDK_2.2_FRDM-KL43Z		*	Browse	QcaPorting_v0
Base project d	irectory (workspace):				RTOS
E:\Firmware\@	GT202\PortingFRDMKL43Z		~	Browse	Baremetal Second December 2005
Toolchain		SDK Example			Preekios
MCUXpre	sso IDE	demo_apps/wifi	i_qca/qca_de	mo	
🔘 Kinetis De	esign Studio				
C GCC ARM	l Embedded				
IAR Embe	edded Workbench for ARM				
C Keil MDK		Bc	reate Proiect		
	M DRT				
Board descrip	ption: FRDM-KL43Z				
The FRDM-KI KL43, KL33, K at 48 MHz -The FRDM-I software.	L43Z is an ultra-low cost devel L27, KL17 and KL13 MCUs buil KL43Z is supported by a range	opment platform for Kin t on ARM® Cortex®-M0 of NXP® and third-party	etis® L famili + processor r y developmer	es * unning *	
)				
A Problems 🛙	Log			- 8	_
type filter text					
Level	Issue	Origin	Target		
💩 Warning	Base project directory (wor	Project Generator			A CONTRACTOR OF A

Figure 23. Project structure configuration

Check the drivers to be used in new project. All drivers used in example had been pre-selected. However, it is recommended to check any highlighted error. You may add the drivers your application may require in future, like I2C, timers, and so on.

Project Configuration			
Project name:	Filter by name/description		
QcaPorting_v0	Name	Version	Description
RTOS		Version	Description
Baremetal		200	ADC16 Driver
		210	Clock Driver
C HEERIOS		200	CMP Driver
		200	COMMON Driver
		200	COP Driver
		2.0.0	DAC Driver
		2.0.1	DMA Driver
		202	DMAMUX Driver
	I ash	2.0.2	Elash Driver
		2.2.0	
		211	GPIO Driver
		2.1.1	Driver
		2.0.3	LIWIL Driver
		2.0.1	LDTMP Driver
		2.0.0	LP INK DIVE
		2.2.5	DIT Driver
		2.0.0	PIT Driver
	o pric	2.0.0	PIVIC Driver
	V V Port	2.0.2	PORT Driver
	v rcm	2.0.1	RCM Driver
		2.0.0	CAL Driver
	sai	2.1.1	SALDriver
State Charles State	sim 🖉 sim	2.0.0	SIM Driver
	sicd	2.0.1	SLCD Driver
	Smc	2.0.3	SMC Driver
	Spi	2.0.3	SPI Driver
A CARLEN CONTRACTOR OF THE OWNER	tpm	2.0.2	TPM Driver
	V vart	2.1.4	UARI Driver
	vref	2.1.0	VREF Driver
		100	
	debug_console	1.0.0	100000
	misc_utilities	1.0.0	Utilities which is ne
	notifier	1.0.0	
		1.0.0	
	CMSIS driver	45.0	
	CMSIS_Driver_Include	4.5.0	
	12c_cmsis	2.0.1	12C CMSIS Driver
	Ipuart_cmsis	2.0.0	LPUART CMSIS Driver
	spi_cmsis	2.0.0	SPI CMSIS Driver
	uart_cmsis	2.0.0	UART CMSIS Driver

Figure 24. MCUXpresso SDK Drivers selection

5. To generate the project, click the 'Create Project' button. If everything is correct, a pop-up message appears.



Figure 25. Project generated

The project will be in selected folder:

orary Share with New folder
📙 board
L CMSIS
👢 doc 👘
📙 drivers
📙 freertos
📙 settings
I source
📙 startup
📙 utilities
📙 wifi_qca
FlashKLxx256KROM_with_config_write_enabled.board
MKL43Z256xxx4_flash.icf
ProjectGeneration.gen
QcaPorting_v0.ewd
QcaPorting_v0.ewp
QcaPorting_v0.eww
QcaPorting_v0.mex

Figure 26. Migrated project folder

5.2.5 Working with new project in IAR EWS

1. Open IAR using the Start menu. Then, open QcaPorting_v0 workspace or double-click QcaPorting_v0.eww. Figure 27 shows the new project structure. This has a slightly different folder structure when comparing with Linked Demo project.



Figure 27. Workspace for standalone project

2. To configure the project, right click the project name and select 'Options'. In the 'General Options' section, chose the microcontroller, as shown in Figure 28.

Category:				
General Options				
Static Analysis				
Runtime Checking	Library Options	2	MISRA-C:2004	MISRA-C:1998
C/C++ Compiler	Target Ou	tout	Library Configuration	Library Options 1
Assembler		put	Elbrary configuration	Elbrary options r
Output Converter	Processor variant			
Custom Build	Core	Cortex-	M0+ 💌	
Build Actions			(1.4270504	
Linker	Opevice	NAP MI	<pre>\L43Z25bX0X4</pre>	E +
Debugger	CMCIC Deals	None		
Simulator	CMSIS-Pack			
CADI	Endian mode	Floo	ting point pottings	
CMSIS DAP	Liidian niode	r iua	ung point settings	
GDB Server	() <u>L</u> ittle	EPU	None	-
I-jet/JTAGjet	<u>Big</u>			
J-Link/J-Trace	O BE <u>3</u> 2	Dre	egisters 🕘	
TI Stellaris	@ BE8			
PE micro		A	dvanced SIMD (NEON)	
ST-LINK				
TIMED FET				
11 AD3				

Figure 28. Select device

3. In C/C++ Compiler category, click the 'Preprocessor' tab and remove all symbol references to FRDM-KL46Z.

Options for node "QcaPorting_v	0"
Options for node "QcaPorting_v Category: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Custom Build Build Actions	0" Factory Settings Multi-file Compilation Discard Unused Publics Diagnostics MISRA-C:2004 MISRA-C:1998 Encodings Extra Options Language 1 Language 2 Code Optimizations Output List Preprocessor generated and include directories Additional include directories: (one per line) SERIOL DIRS/CMSIS
Linker Debugger Simulator CADI CMSIS DAP	\$PROJ_DIR\$/CMSIS \$PROJ_DIR\$/board \$PROJ_DIR\$/doc \$PROJ_DIR\$/doc \$PROJ_DIR\$/drivers \$PROJ_DIR\$/freertos Preinclude file:
GDB Server I-jet/JTAGjet J-Link/J-Trace TI Stellaris PE micro ST-LINK Third-Party Driver TI MSP-FET	Defined symbols: (one per line) CPU_MKL43Z256VLH4_cm0plus SDK_OS_FREE_RTOS CPU_MKL43Z256VLH4 SDK_CORE_ID_CORE0
TI XDS	OK Cancel

Figure 29. Symbol references

4. In the 'Linker' category, click the 'Config' tab and check if the linker configuration file points to corresponding FRDM-KL43Z one, as shown in Figure 30.

ptions for node "QcaPorting_"	v0"						X
Category: General Options Static Analysis Puntime Checking						Factory Setting	s
C/C++ Compiler	#define	Diag	nostics	Checksum	Encodings	Extra Option:	S
Assembler	Config	Library	Input	Optimizations	Advanced	Output List	
Output Converter	Linker	onfiguratio	on file				
Custom Build	<u>√</u> <u>O</u> v	erride defa	ult				
Build Actions	\$F	ROJ_DIR	\$\MKL43Z	256xxx4_flash.icf			
Linker			_			i	
Debugger		Edit					
Simulator							9
CADI	Configura	ation file sy	mbol defi	nitions: (one per li	ne)		
CMSIS DAP						-	
Liet/ITAGiet							
]-Link/]-Trace							
TI Stellaris							
PE micro							
ST-LINK						_	
Third-Party Driver							
TI MSP-FET							
TI XDS							
					ОК	Cance	1

Figure 30. Linker configuration file

5. In the 'Debugger' category, select a debugger tool, then click the 'Download' tab. Add the specific flash loader as in Figure 31 to avoid errors during programing at specific memory region (0x40C - 0x40F) in FRDM-KL43Z.

Category: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter General Options Setup Download Images Extra Options Multicore Plugins Verify download	otions for node "QcaPorting_v	70" ×
Custom Build Build Actions Linker Debugger Simulator CADI CADI CMSIS DAP GDB Server I-jet/JTAGjet J-Link/J-Trace TI Stellaris PE micro ST-LINK Third-Party Driver TI MSP-FET TI XDS OK Cancel	Category: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator CADI CMSIS DAP GDB Server I-jet/JTAGjet J-Link/J-Trace TI Stellaris PE micro ST-LINK Third-Party Driver TI MSP-FET TI XDS	Factory Settings Setup Download Verify download Suppress download Use flash loader(s) Override default.board file \$PROJ_DIR\$\FlashKLxx256KROM_with_config_write_ena Edit

Figure 31. Include flash loader

6. After this initial setup, build the project and check 'Build' log for errors.

Messages	File	Line	
board.c Warning[Pe223]: function "CLOCK_SetLpsci0Clock" declared implicitly Warning[Pe223]: function "CLOCK_GetPIIFIISelClkFreq" declared implicitly Error[Pe020]: identifier "UART0" is undefined Fror while running C/C++ Compiler bmi.c clock_config.c	E:\Firmware\GT202\PortingFRDMKL43\board.c E:\Firmware\GT202\PortingFRDMKL43\board.c E:\Firmware\GT202\PortingFRDMKL43\board.c	57 59 60	
uild Debug Log			
eady			Errors 1, Warnings



7. UARTO is selected as default debug interface. LPUARTO The debug interface for FRDM-KL43Z is LPUARTO. All clock references need to be fixed as well.



Figure 33. Fix UART0 definitions

8. On board.c, LPUART0 clock must be added.



Figure 34. LPUART0 Clock Enable

Some files need to be checked in order to guarantee the compatibility of new ported software. The file wifi_shield_gt202.h has all definitions for the Wi-Fi shield. It is basically an abstraction file to concentrate and make it easier to include new or change existent Wi-Fi modules/boards.

9. All parameters present in this file are not automatically updated by MCUXpresso Config Tools. Check IRQ signal interrupts if using different ports. The microcontroller may have a bundle interrupt vector, for example, PortB+PortC or PortC+PortD. This can be verified using the reference manual and the specific included file.

This configuration file also sets SPI speed and DMA.

```
board.h
                 fsl_clock.h
board.c
                              wifi shield gt202.h X main.c
    /* WLAN IRQ signal */
    #define WIFISHIELD_WLAN_IRQn (PORTC_PORTD_IRQn)
    #define WIFISHIELD_WLAN_ISR PORTC_PORTD_IRQHandler
    #define WIFISHIELD_WLAN_IRQ_DIRECTION (BOARD_INITGT202SHIELD_IRQ_DIRECTION)
    #define WIFISHIELD_WLAN_IRQ_PORT (BOARD_INITGT202SHIELD_IRQ_PORT)
    #define WIFISHIELD_WLAN_IRQ_GPIO (BOARD_INITGT202SHIELD_IRQ_GPIO)
    #define WIFISHIELD_WLAN_IRQ_PIN (BOARD_INITGT202SHIELD_IRQ_GPIO_PIN)
    /* WLAN PWRON signal */
    #define WIFISHIELD WLAN PWRON DIRECTION (BOARD INITGT202SHIELD PWRON DIRECTION)
    #define WIFISHIELD_WLAN_PWRON_PORT (BOARD_INITGT202SHIELD_PWRON_PORT)
    #define WIFISHIELD_WLAN_PWRON_GPIO (BOARD_INITGT202SHIELD_PWRON_GPIO)
    #define WIFISHIELD WLAN_PWRON_PIN (BOARD_INITGT202SHIELD_PWRON_GPIO_PIN)
    /* SPI settings */
    #define WIFISHIELD SPI (SPI1)
    #define WIFISHIELD_SPI_INIT_CS (kSPI_Pcs0)
    #define WIFISHIELD_SPI_CLOCKSRC (SPI1_CLK_SRC)
    #define WIFISHIELD_SPI_BAUDRATE (15000000)
    #define WIFISHIELD_SPI_THRESHOLD (0)
    /* DMAMUX settings, interconnect SPI with DMA */
    #define WIFISHIELD_DMAMUX (DMAMUX0)
    #define WIFISHIELD DMAMUX RX REQ (kDmaRequestMux0SPI1Rx)
    #define WIFISHIELD_DMAMUX_TX_REQ (kDmaRequestMux0SPI1Tx)
    /* DMA settings */
    #define WIFISHIELD DMA (DMA0)
    #define WIFISHIELD_DMA_RX_CHNL (0)
    #define WIFISHIELD DMA_TX_CHNL (1)
```



10. Rebuild the project. No errors and warnings should appear as in Figure 34.

Messages			
wifi_env.c			
wifi_shield.c			
wifi_spi.c			
wlan_qca400x	.c		
wlan_qcom.c			
wmi.c			
Linking			
Total number	of errors: 0		
i otal number	JT Warnings: U		

Figure 36. Final build

11. Lastly, flash new code and debug the **qca_demo** example.

5.3 MCUXpresso SDK-Linked Porting

All regular demo examples offered inside MCUXpresso SDK are linked. Linked projects can be easier to update and add new features.

5.3.1 Configure Clocks and Ports

- 1. Open MCUXpresso Config Tools and import project as described in Section 5.2.1, "Opening MCUXpresso SDK Qca_demo Example". Modify clock and pins as described in Section 5.2.2, "Setting up Clock" and Section 5.2.3, "Setting up Pins" respectively. However, instead of generating a project it is necessary to export clocks and pins configurations files independently.
- 2. Go to the 'File' menu and select 'Export'. Expand the clocks tools and select "Export source Files".



Figure 37. Exporting files in MCUXpresso Config tools

Export	
Select Export Source Files	Ľ
Select an export wizard:	
type filter text	
 Clocks Tool Export HTML Report Export Source Files Pins Tool Export HTML Report Export Registers Export Source Files Export the Pins in CSV (Comma Separated Values) Format Processor Data 	
< <u>B</u> ack Next > Einish	Cancel

Figure 38. Export Clock configuration files

3. Choose the FRDM-KL43Z SDK folder and export files to it. An alert is pop up. Click the 'Yes' button (MCUXpresso SDK renames old files to .bak as in shown in Figure 39).

4. Repeat all the previous steps to export pins.

Export	ck
Export Cloc	s Sources
To directory: Cortex-M0F	C:\NXP\KSDK\SDK_2.2_FRDM-KL43Z\boards\frdmkl43z\demo_apps\wifi_c Browse CLK output LK output
C:\NXP\KS	K\SDK 2.2 FRDM-KL43Z\boards\frdmkl43z\demo apps\wifi gca\gca (Browse UT
	Confirm save
	Do you want to replace following file(s)? clock_config.c, clock_config.h
	Always overwrite without asking
	Yes No
	C PETPOAINT CIOCK
	< Back Next > Finish Cancel e TPM clock

Figure 39. Exporting clocks config files

5.3.2 Fix IAR Configuration files

- Go to {YOUR PATH}\SDK_2.2_FRDM-KL43Z\boards\frdmkl43z\demo_apps\wifi_qca\qca_demo\iar and open the qca_demo.ewp file in text editor:
- 2. Replace all FRDM-KL46L strings to FRDM-KL43Z.

It is required to rename (or delete and include a new reference) LPSCI references to LPUART in all included files.

```
2160
          <file>
2161
            <name>$PROJ DIR$/../../../../devices/MKL43Z4/drivers/fsl lpuart.h</name>
2162
          </file>
2163
          <file>
2164
            <name>$PROJ_DIR$/../../../../devices/MKL43Z4/drivers/fsl_lpuart.c</name>
2165
          </file>
2166
          <file>
2167
            <name>$PROJ_DIR$/../../../../devices/MKL43Z4/drivers/fsl_uart.h</name>
           </file>
2168
2169
           <file>
2170
            <name>$PROJ DIR$/../../../../devices/MKL43Z4/drivers/fsl uart.c</name>
2171
           </file>
2172 白
          <file>
```

Figure 40. Change driver names

5.3.3 Working in IAR EWS

- 1. Open IAR using the Start menu, then open the **demo_qca** workspace or double-click **demo_qca.eww.**
- 2. Go to the 'Project Options' menu and check the information as described in Section 5.2.5, "Working with New project in IAR EWS". Except for flash loader, all other configuration should be correct. Include flash loader as shown in Figure 31.
- 3. Change board.h and board.c. See Figure 33 and Figure 34.

The Qca_demo example should build and debug without errors.

5.4 Considerations for Other MCUXpresso SDKs

The setup would change according to the MCUXpresso SDK used for migration.

To port FRDM-K64 SDK to FRDM-K66 follow same steps as used for FRDM-KL43Z. However, there are some additional steps in this platform.

FRDM-K66 can reach up to 180 MHz of CPU clock. This frequency is not possible in BOARD_BootClockRUN mode. Thus, this MCU has an additional mode, called BOARD_BootClockHSRUN. To configure BOARD_BootClockHSRUN, see Figure 41 and Figure 42.

Name Internal FAST_IRCLK SLOW_IRCLK IRC48M	Available V	alue			
Internal FAST_IRCLK SLOW_IRCLK IRC48M	4		Name	L Value	Acy
FAST_IRCLK SLOW_IRCLK IRC48M	4		🗆 System		
SLOW_IRCLK IRC48M		MHz	Core clock	🔒 180 MHz	±0.1%
IRC48M	3	2.768 kHz	→ System clock	180 MHz	
180	Ir	active	→ Bus clock	60 MHz	
LPO	1	kHz	→ FlexBus clock	60 MHz	
External			→ Flash clock	25.71 MHz	
OSC (Systecillator)	✓ 1	2 MHz 🛏	Peripheral		
■ RTC32kHz	Ir	active	MCGIRCLK	32.768 kHz	
USB clock input	Ir	active	→ MCGFFCLK	375 kHz	
ENET 1588 clock input	Ir Ir	active	→ OSCERCLK	12 MHz	
SDHC clock input	Ir	active	→ OSCERCLK undivided	12 MHz	
			ERCLK32K	Inactive	
			RTC_CLKOUT	Inactive	
			→ MCG PLL/FLPFD clock	k 180 MHz	
			LPO clock	1 kHz	
			IRC48MCLK	Inactive	
			→ USB FS clock	Inactive	
			→ Trace clock input	Inactive	
			→ ENET IEEEamp clock	Inactive	
			ENET RMII clock	Inactive	
			→ SDHC clock	Inactive	
			→ CLKOUT(FB_CLK)	Inactive	
			→ LPUART clock	Inactive	
			→ TPM clock	Inactive	
			USB slow clock	Inactive	
			USBPHYPLLCLK	Inactive	

Figure 41. Configuring HSRUN in FRDM-K66

🖻 Details 🖾 💽 Sources 🔐 Reg	isters 🚺 Module Clocks 📄 Log		
Path Details: Core_clock	The Design of the Second		
Name A	L Value Ac		
Core clock	180 MHz ±0.1		
CORECLK Frequency	180 MHz		
OUTDIV1	/1		
OUTDIV1 Frequency	180 MHz		
MCGOUTCLK Frequency	180 MHz		
CLKS	Output of PLLS (FLL or PLL clock)		
PLLS	PLL output		
PLLCS	PLL0 clock		
PLL output divider by 2	/2		
PLL output Frequency	180 MHz		
E PLL			
PLL Frequency	360 MHz		
PLL clock	Disabled		
PLL clock in Stop mode	Disabled		
PRDIV	/1		
VDIV	* 30		
PLL OSCSEL	System Oscillator		
OSCCLK Frequency	12 MHz		
OSCCLK Frequency	12 MHz		
🗆 OSC (System Oscillator) 🛛 🗹	12 MHz		
OSC mode	Using oscillator wystal (low power)		
Frequency Range	Very_high freque range 8-32 MHz		
System Oscacity Load	0 pF		

Figure 42. FRDM-K66 HSRUN Core Clock

Another difference between FRDM-K64 to FRDM-K66 is that the IRQ pin is at different port. IRQ is connected to PTC3 in FRDM-K64 and to PTA25 in FRDM-K66.

In this case, it is required to modify IRQ settings on *wifi_shield_gt202.h*.

#define WIFISHIELD_WLAN_IRQn (PORTA_IRQn) // instead of PORTB on FRDM-K64.

#define WIFISHIELD_WLAN_ISR PORTA_IRQHandler // instead of PORTB on FRDM-K64.

Additional procedures are similarly explained in the migration from FRDM-KL46Z to FRDM-KL43Z.

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.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, Freescale, the Freescale logo, and Kinetis, are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, the ARM logo, and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. mbed is a trademark of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

© 2017 NXP B.V.

Document Number: AN11981 Rev. 0 06/2017

