

NXP FRDM and QCA400x Shields for Wi-Fi Solutions

1 Overview

This guide describes how to setup a Wi-Fi solution using the NXP Freedom boards along with the SX-ULPAN-2401-SHIELD. The guide describes the required hardware and software, how to connect the boards, and how to program and run the demo applications.

There are two demo applications available; one for exercising the common Wi-Fi commands and the second one is a throughput demo.

This document is intended to be used by software engineers, system engineers, and test engineers.

Contents

1	Overview	1
2	Hardware overview	2
2.1	Hardware configurations	2
2.2	Assembly instructions	3
2.3	Additional hardware	3
3	Software overview	4
3.1	Using Kinetis SDK 2.x	4
4	Running the Wi-Fi Shell demo	12
4.1	Prepare the demo	12
4.2	Exercise the console commands	14
5	Running the Wi-Fi throughput demo	16
6	References	20
7	Revision history	21



2 Hardware overview

2.1 Hardware configurations

The following hardware configurations are currently supported:

- FRDM-K22F plus SX-ULPAN-2401-SHIELD
- FRDM-KL46 plus SX-ULPAN-2401-SHIELD
- FRDM-K64F plus SX-ULPAN-2401-SHIELD
- FRDM-K82F plus SX-ULPAN-2401-SHIELD
- FRDM-KL28Z plus SX-ULPAN-2401-SHIELD
- FRDM-K32W042 plus SX-ULPAN-2401-SHIELD
- EVK-MIMXRT1050 plus SX-ULPAN-2401-SHIELD
- EVK-MIMXRT1060 plus SX-ULPAN-2401-SHIELD

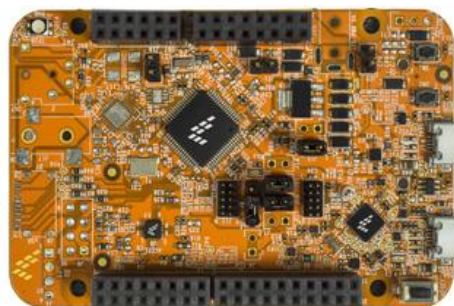


Figure 1: NXP FRDM-K22F board

The SX-ULPAN-2401-SHIELD kit contains the following parts:

- SX-ULPAN-2401 Wi-Fi Module (Soldered on Adapter Board)
- Adapter Board with Freedom compatible headers



Figure 2: Silex SX-ULPAN-2401 Shield

2.2 Assembly instructions

The assembly procedure is simple. Just plug the Silex/ULPAN shield into the Freedom board, taking the board orientation into consideration. The USB connectors from the Freedom board must remain visible after the assembly is done.



Figure 3: Assembling the boards

2.3 Additional hardware

Besides the modules described above, the following materials are also necessary:

- USB A to micro USB cable
- Personal Computer

The QCA WIFI application does not call for any special hardware configuration. Although not required, the recommendation is to leave the development board jumper settings and configurations in the default state when running this demo.

3 Software overview

The Freedom board to be used must be programmed with the demo application binary before use. The demo projects are provided in the source code and must be built. The SX-ULPAN module is pre-programmed with the Wi-Fi firmware and no further actions are required. For custom Kinetis development, NXP offers the Kinetis SDK 2.x drivers as the current option.

3.1 Using Kinetis SDK 2.x

3.1.1 Software requirements

- QCA400x drivers, libraries and demo applications (available on www.nxp.com)
- Kinetis SDK v2.x (available on mcuxpresso.nxp.com)
 - It is recommended to use always the latest version of SDK v2
- MCUXpresso IDE (available on <http://www.nxp.com/mcuxpresso/ide>)
- PC Virtual COM port software (TeraTerm, puTTY, etc)

3.1.2 Software installation

To get the right software go to mcuxpresso.nxp.com:

- 1 - Select your Development Board:

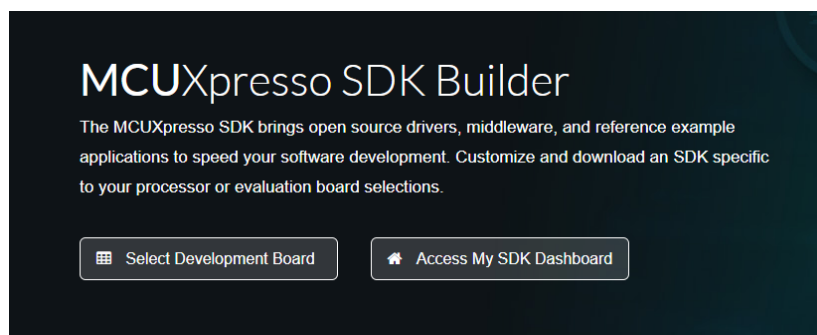


Figure 4 – Accessing Board

If asked use your login and password or create one.

- 2 - Select your Board in list:

Select Development Board

Search for your board or kit to get started.

Search by Name

Search...

Select a Device, Board, or Kit

▼ Boards

▼ Kinetis

- FRDM-K22F
- FRDM-K28F
- FRDM-K28FA
- FRDM-K64F**
- FRDM-K66F
- FRDM-K82F
- FRDM-KE02Z40M
- FRDM-KE04Z

Name your SDK

FRDM-K64F

Don't use: < > : " / \ | ? * % \



Hardware Details

Board	FRDM-K64F
Device	MK64F12
Core Type / Max Freq	Cortex-M4F / 120MHz
Memory Size	1024 KB Flash 256 KB RAM

Actions

- Build MCUXpresso SDK
- Explore selection with Clocks tool
- Explore selection with Pins tool

Figure 5 - SDK Board selection

3 - Click on Build MCUXpresso SDK button:

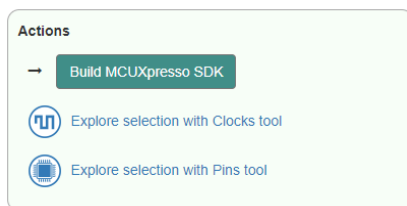


Figure 6 – Build MCUXpresso SDK

4 - Select your OS and then the preferable Toolchain (or all of them):

Developer Environment Settings
Selections here will impact files and examples projects included in the SDK and Generated Projects

Host OS: Windows ▼

Toolchain / IDE: MCUXpresso IDE ▼

Select Optional Middleware
Add middleware, operating systems, and so on

+ Add software component

- MCUXpresso IDE
- GCC ARM Embedded
- IAR Embedded Workbench for ARM
- Keil MDK
- All toolchains

Figure 7 - OS and Toolchain selection

- 5 - Now click on Add software component button in order to select RTOS and Middleware
- 6 - Select all necessary software components for your project. For QCA400x, it is mandatory Amazon-FreeRTOS Kernel and QCA400x WiFi and click in Save Changes button.

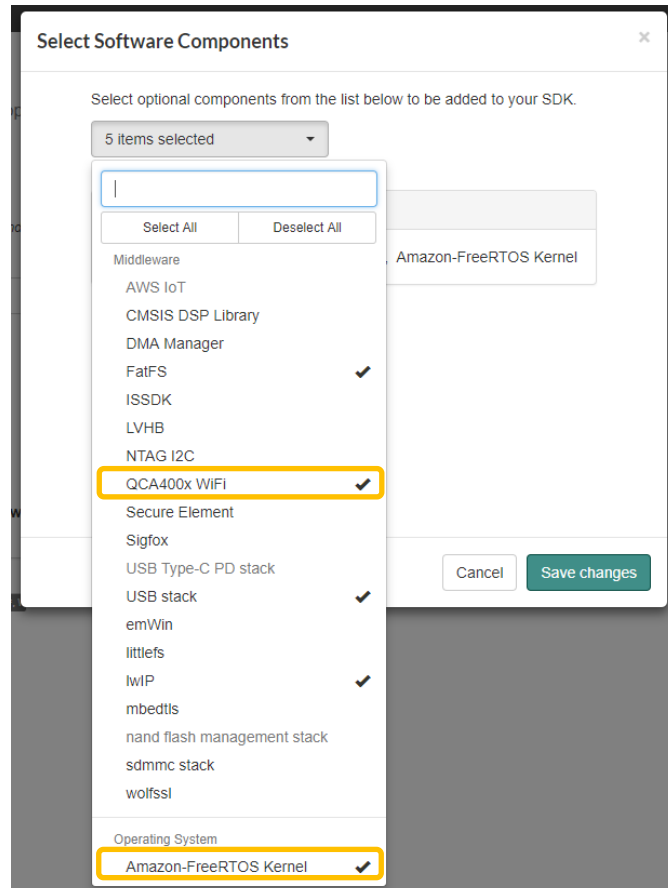


Figure 8 – Additional Components

- 7 - Then request a New Build, clicking on Request Build Button:

Click the link below to request this specific MCUXpresso SDK Build

In general, SDK builds should complete within a few minutes.

You will be notified via email and notifications in the upper right corner of this webpage.

[Request Build →](#)

Archive Name

Don't use: < > : \ / . , ! ? * ~ in the name of your SDK

Figure 9 – Additional Components

- 8 - You should receive an e-mail when SDK starts to build and one after it ends with a link to Download. Another way to download the new generated SDK is enter in MCUXpresso SDK Dashboard and Click on Download icon of desired configuration:

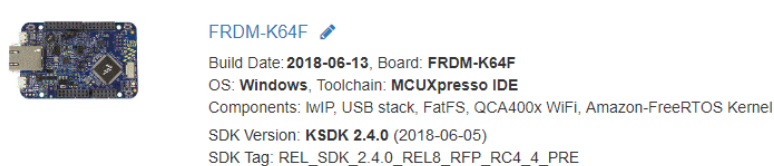


Figure 10 – SDK Dashboard

- 9 - Download and unzip the file to a local folder (e.g. C:\NXP\SDK_2.2_FRDM-K22F)

- 10 - Your local file structure shall be like:

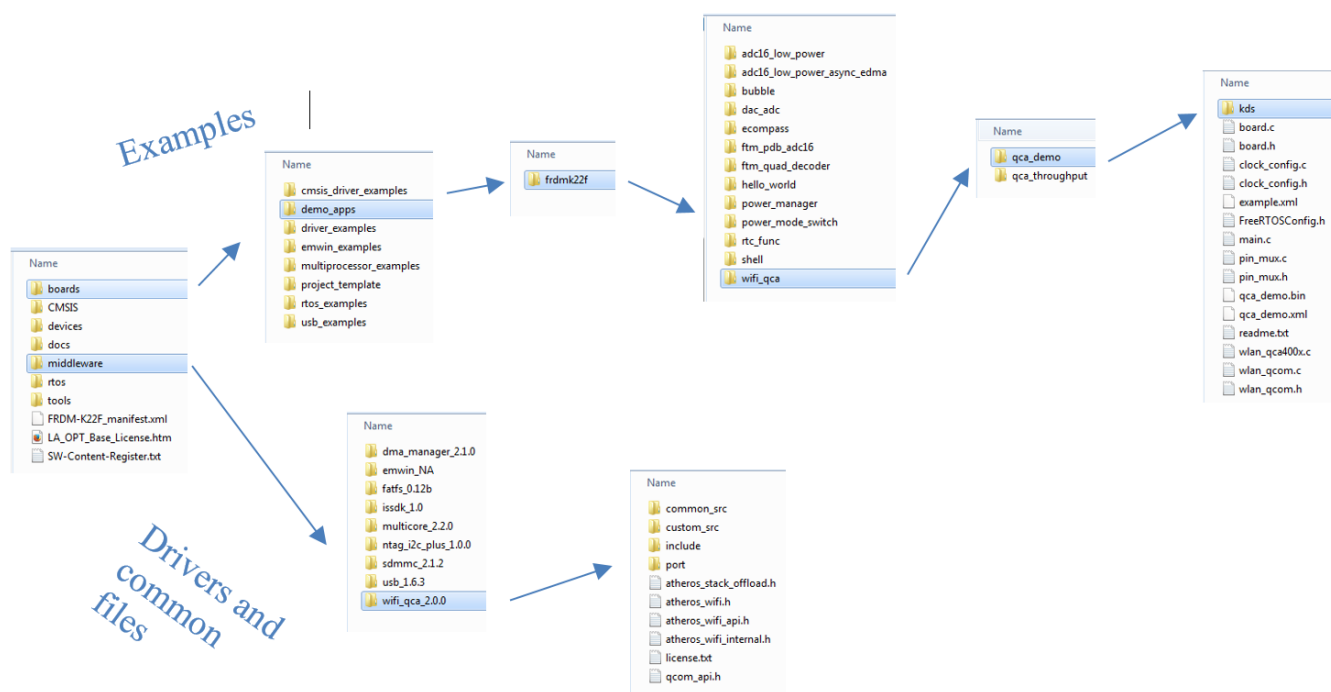


Figure 12 – SDK2 Folder Structure

As shown in the above figure, the examples can be accessed in the `wifi_qca` folder. In the `qca_demo` folder, you can access all board-specific files. In these files, you can change the pins, SPI, UART, and other items that are board-dependent. This is the first place you must change to migrate the demo project to your custom board.

In the `wifi_qca_2.0.0` folder, all the drivers and common files included in the demo projects are located. Look at these folders to understand how the demos work and/or add more features or functions to your custom project.

3.1.3 Using demo application with MCUXpresso IDE

Open MCUXpresso IDE and Drag and Drop your FRDM-K64 SDK folder inside Installed SDK area:

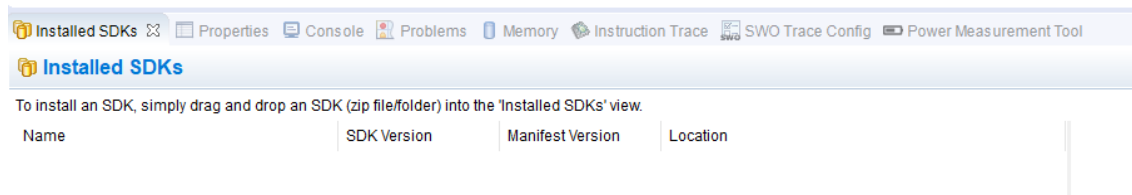


Figure 13 – Drag and Drop SDK here

Project structure should be:

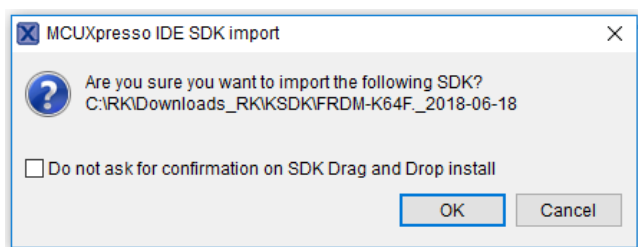


Figure 14 – Accept Installation

In bottom left area of MCUXpresso click on Import SDK Examples

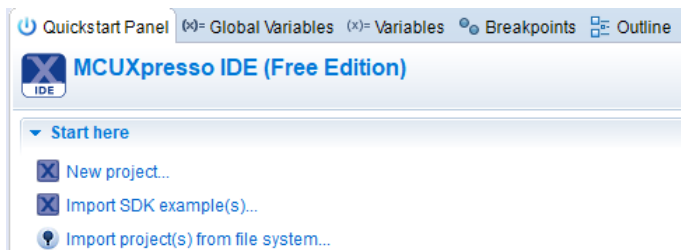


Figure 15 – Import Examples

A new window will open and the desired board needs to be selected. After selection, click in Next Button. Then select the example(s) to open in MCUXpresso IDE.

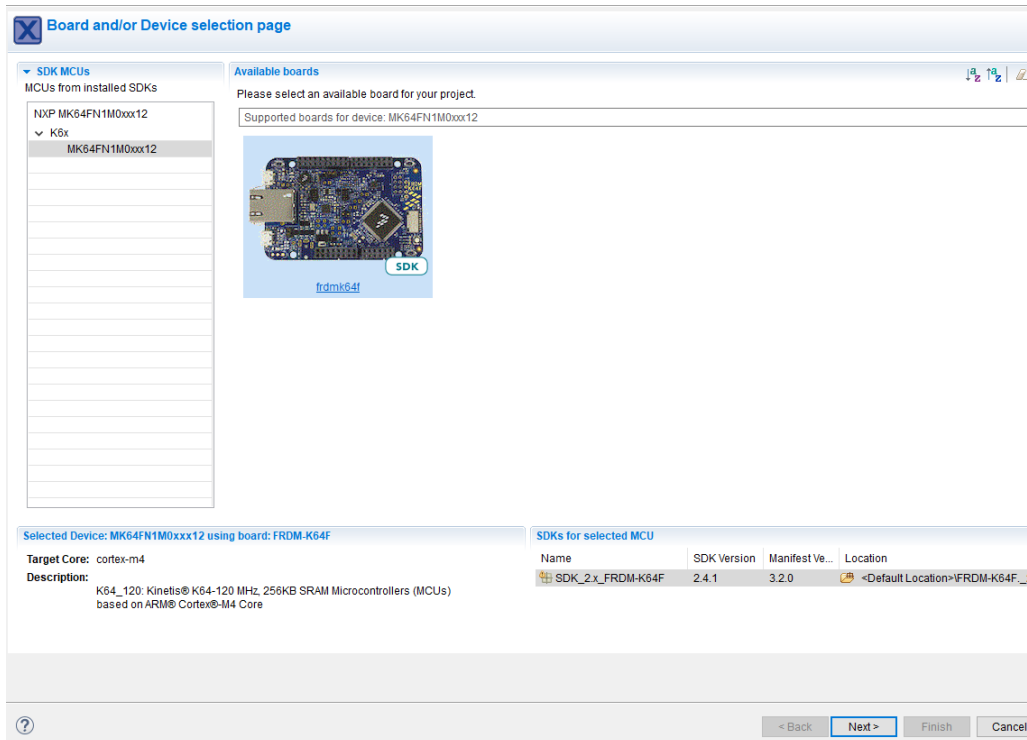


Figure 16 - Select Target Board

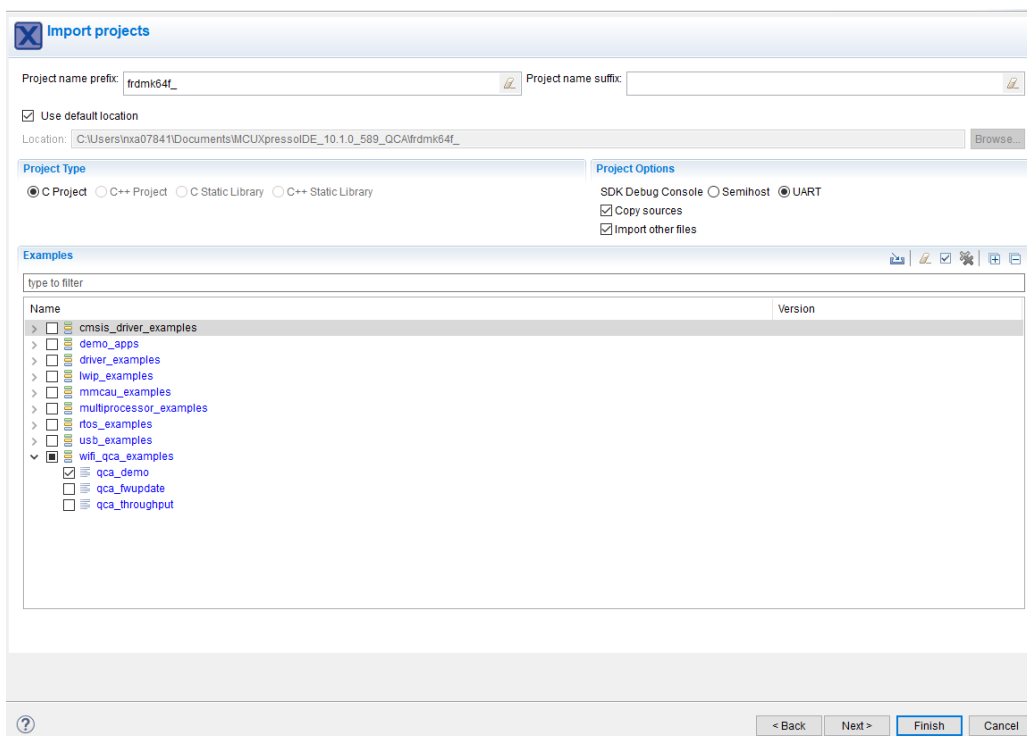


Figure 17 - Select SDK Example

3.1.4 Using demo application with IAR

In IAR, open the *qca_demo.eww*. It will import qca_demo project files and configuration. Workspace will be:

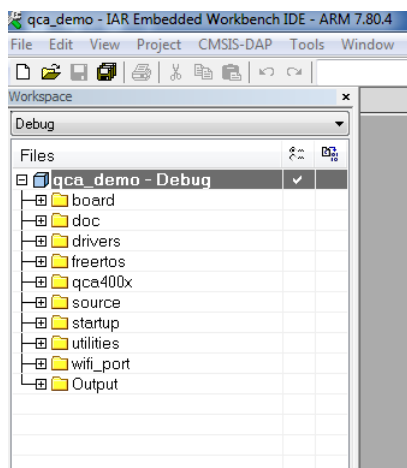


Figure 20 - Workspace IAR IDE

Enter in Menu *Project* -> *Rebuild All*. After complete, enter again in *Project*, and then *Download and Debug*. It will open the debug window in IAR and you can run/step run the code.

3.1.5 Migrating Demo Software to your custom design

The demo software is supplied to test the FRDM-K22F and SX-ULPAN.

In qca_demo, the *middleware\wifi_qca\port\boards\frdmk64f\freertos\silex2401* folder contains all the board-dependent files. The *wifi_shield_silex2401.h* file contains the main settings and it is the first to be reviewed to port and/or debug a custom board. It includes the definitions for the SPI module, correct pin selection for each SPI signal, Power-on Pin (GPIO used to turn the shield on or off), IRQ, and DMA channel. All of them must be assigned correctly.

Also, a session called *BOARD_InitSilex2401Shield* in *pin_mux.h* exists to set every GPIO necessary to use SX-ULPAN shield.

For a new or a custom board, all these files could be cloned and modified.

The GPIO settings are taken from the *pin_mux.h* file generated by the MCUXpresso PinmuxTool. Part of file for Silex2401 configuration:

```
/*FUNCTION*****
*
* Function Name : BOARD_InitSilex2401Shield
* Description   : Configures pin routing and optionally pin electrical features.
*
*END*****/
void BOARD_InitSilex2401Shield(void) {
    CLOCK_EnableClock(kCLOCK_PortB); /* Port B Clock Gate Control: Clock enabled */
```

```

CLOCK_EnableClock(kCLOCK_PortD);                /* Port D Clock Gate Control: Clock enabled */

PORT_SetPinMux(PORTB, PIN23_IDX, kPORT_MuxAsGpio); /* PORTB23 (pin 69) is configured as PTB23 */
PORTB->PCR[23] = ((PORTB->PCR[23] &
(~(PORT_PCR_PS_MASK | PORT_PCR_ISF_MASK))) /* Mask bits to zero which are setting */
| PORT_PCR_PE_MASK | PORT_PCR_ISF_MASK) /* Pull Enable: Internal pullup or pulldown resistor is
enabled on the corresponding pin, if the pin is configured as a digital input. */
);
PORT_SetPinMux(PORTB, PIN9_IDX, kPORT_MuxAsGpio); /* PORTB9 (pin 57) is configured as PTB9 */
PORTB->PCR[9] = ((PORTB->PCR[9] &
(~(PORT_PCR_PS_MASK | PORT_PCR_PE_MASK | PORT_PCR_ISF_MASK))) /* Mask bits to zero which are setting */
| PORT_PCR_PS_MASK | PORT_PCR_ISF_MASK) /* Pull Select: Internal pullup resistor is enabled on the
corresponding pin, if the corresponding PE field is set. */
| PORT_PCR_PE_MASK | PORT_PCR_ISF_MASK) /* Pull Enable: Internal pullup or pulldown resistor is
enabled on the corresponding pin, if the pin is configured as a digital input. */
);
PORT_SetPinMux(PORTD, PIN0_IDX, kPORT_MuxAlt2); /* PORTD0 (pin 93) is configured as SPI0_PCS0 */
PORT_SetPinMux(PORTD, PIN1_IDX, kPORT_MuxAlt2); /* PORTD1 (pin 94) is configured as SPI0_SCK */
PORT_SetPinMux(PORTD, PIN2_IDX, kPORT_MuxAlt2); /* PORTD2 (pin 95) is configured as SPI0_SOUT */
PORT_SetPinMux(PORTD, PIN3_IDX, kPORT_MuxAlt2); /* PORTD3 (pin 96) is configured as SPI0_SIN */
}

```

There are other files that are used to configure the clock gating and functions for the pins (such as `pin_mux.c`), but these files are usually spread out in the KSDK examples. You can select and configure the pins using the MCUXpresso Pin Configuration tool (available as a web service at <https://mcuxpresso.nxp.com/>). The necessary signals are:

- SPI MOSI
- SPI MISO
- SPI SCK
- SPI CS
- WLAN IRQ (GPIO)
- WLAN PWRUP (GPIO)

The WLAN IRQ must be configured to enable the pull-up and it must support the GPIO interrupts. Some of the KL chips do not provide GPIO interrupts for all GPIO ports. The WLAN PWRUP must be set to pull down.

If using another Freedom board (instead of FRDM-K22F), the information above is still valid. Some MCUs have a slightly different way to configure different peripherals, but the files you must change are the same as described.

For swap between shields or in case of more supported boards (in case the same structure was kept), a define must be set in order to select the correct board. In `middleware\wifi_qca\port\boards\frdmk64f\freertos\wifi_shield.h` a selection between shields can be easily done:

```

/* Select specific shield support */
// #define WIFISHIELD_IS_GT202
#define WIFISHIELD_IS_SILEX2041

```

4 Running the Wi-Fi Shell demo

4.1 Prepare the demo

To prepare the demo, follow these steps:

- Connect the FRDM-K22F board to the PC using the USB cable. The USB connector used on the Freedom board is the OpenSDA USB.



Figure 21: OpenSDA USB port to be used

- Wait for the debug and virtual COM port drivers to install on the PC.
- Within the IAR IDE, start a debug session to program the K22F chip. The debug configuration used depends on the debug interface used on the FRDM-K22F board (for example, Segger J-Link, PEMicro, OpenOCD, mbed CMSIS-DAP, and other).
- After programming, terminate the debug session. The board is ready to be used.
- Open the serial COM port application on the PC (TeraTerm, puTTY) and configure the communication parameters for the port that corresponds to FRDM-K22F (available in the Device Manager): 115200 baud, 8N1, no flow control.
- Reset the FRDM-K22F board.
- The demo application starts and the terminal application shows the version information:

```
Host version:      3.3.0.0
Target version:    0x31c80997
Firmware version:  3.3.4.91
Interface version: 1
```

- The menu is displayed (it is displayed again each time the help is called by pressing the “h” key):

```
s  AP Scan
c  AP Connect (SSID='nxp',
pass='NXP0123456789')
D  AP Disconnect
d  Get DHCP address
g  HTTP GET nxp.com
w  HTTP GET from gateway
p  Ping gateway
P  Ping nxp.com
i  Print IP configuration
R  Resolve some hosts
h  Help (print this menu)
H  Print extended help
```

4.2 Exercise the console commands

4.2.1 AP Connect

Pressing “c” to connect to an AP with the SSID='nxp', pass='NXP0123456789'

Key 'c': AP Connect (SSID='nxp', pass='NXP0123456789')

Reading connection params

opMode=0 (Station)

phyMode=mixed

ssid=nxp

EVENT: CLIENT connected

EVENT: 4 way handshake success for device=0

EVENT: CLIENT connected

EVENT: 4 way handshake success for device=0

4.2.2 AP Disconnect

Pressing “D” to disconnect again

Key 'D': AP Disconnect

EVENT: CLIENT disconnect

4.2.3 Get DHCP address

It is necessary to Press “d” after connection to get and display the address assigned e.g.

Getting DHCP address...

EVENT: CLIENT connected

EVENT: 4 way handshake success for device=0

DNS 0: 192.168.43.1

addr: 192.168.10.81 mask: 255.255.255.0 gw: 192.168.10.1

4.2.4 HTTP GET www.nxp.com

Pressing “g” to get from the www.nxp.com

```
Key 'g': HTTP GET www.nxp.com
*****
Looked up www.nxp.com as 104.80.15.112
HTTP GET from 104.80.15.112:80
GET / HTTP/1.0
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:50.0) Gecko/20100101
Firefox/50.0
Accept-Language: en-us
Host: www.nxp.com
```

```
TCP sent 148 bytes
Waiting for response (with t_select)
qcom_recv() receiving response
TCP received 461 bytes
(...)
```

4.2.5 Resolve Hosts

Pressing “R” to get some host name resolved by dns

```
Key 'R': Resolve some hosts
Looked up google.com as 172.217.29.110
Looked up cr.yp.to as 131.193.32.109
Looked up kernel.org as 198.145.29.83
Looked up www.nxp.com as 104.80.15.112
```

4.2.6 Ping gateway

Pressing “p” to ping the AP

```
Key 'p': Ping gateway
Pinging 192.168.10.1... OK (0 ms)
```

4.2.7 Ping nxp.com

Pressing “P” to ping *www.nxp.com*

```
Key 'P': Ping nxp.com
Looked up www.nxp.com as 23.44.183.148
Pinging 23.44.183.148... OK (20 ms)
```

4.2.8 Print IP configuration

Pressing “i” to display the IP configuration e.g.

```
Key 'i': Print IP configuration
addr: 192.168.10.81 mask: 255.255.255.0 gw: 192.168.10.1
```

4.2.9 Resolve some hosts

Pressing “R” to resolve and display some (hardcoded) hosts

```
Key 'R': Resolve some hosts
Looked up google.com as 216.58.209.110
Looked up cr.yp.to as 131.155.70.11
Looked up kernel.org as 199.204.44.194
Looked up nxp.com as 192.88.156.33
```

5 Running the Wi-Fi throughput demo

The throughput demo requires two FRDM+SX-ULPAN setups, not necessarily identical. Both setups connect to the same Access Point (AP) and the data are passed from one device to the other. This figure describes the setup:

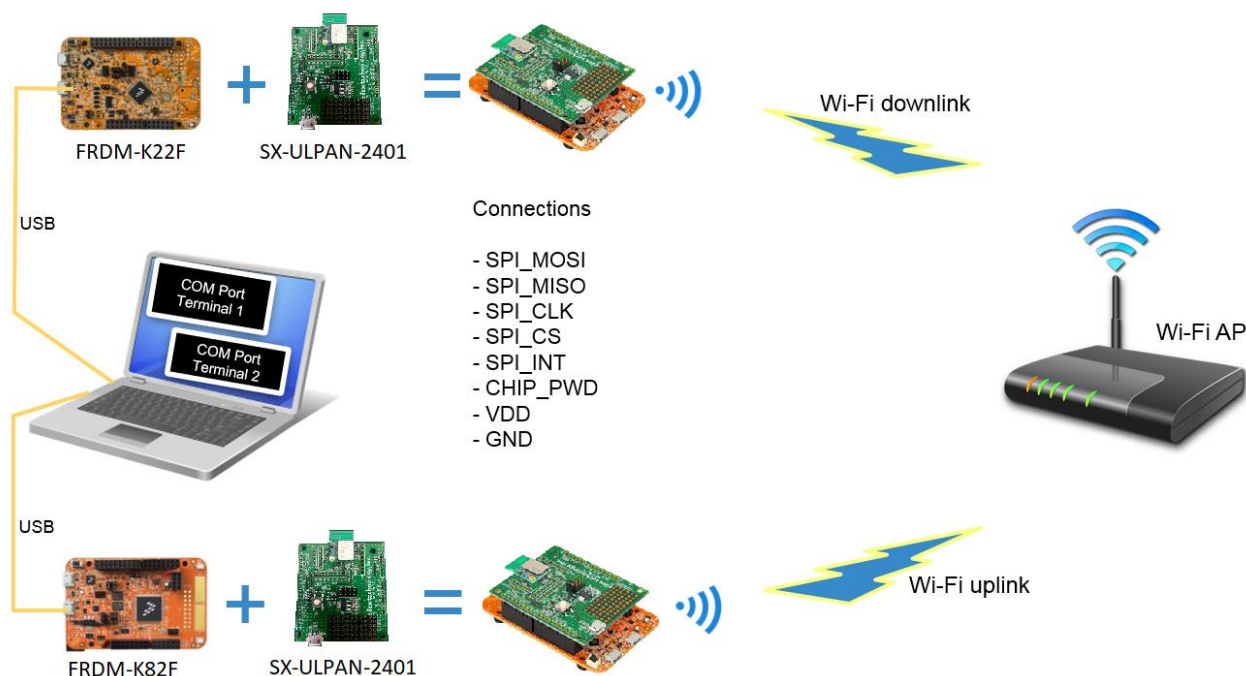


Figure 22: Wi-Fi Throughput demo setup

The throughput example files are located at (for example, for FRDM-K22F):

<SDK_Install>\boards\frdmk22f\demo_apps\wifi_qca\qca_throughput\

If using similar setups, both Freedom boards must be programmed with the same firmware, otherwise a different example project (for example, for FRDM-K82F) must be added and built in the KDSK workspace.

When the two Freedom boards are programmed, the throughput test can be performed. To do this, the Freedom boards must be connected to the USB ports on a PC/laptop and two serial COM port terminal applications must be opened. Each terminal application connects the PC/laptop to one Freedom board.

The following tables list the commands that must be executed on each device, always starting with Device 1 which is the TCP listener.

Table 1 – Device 1

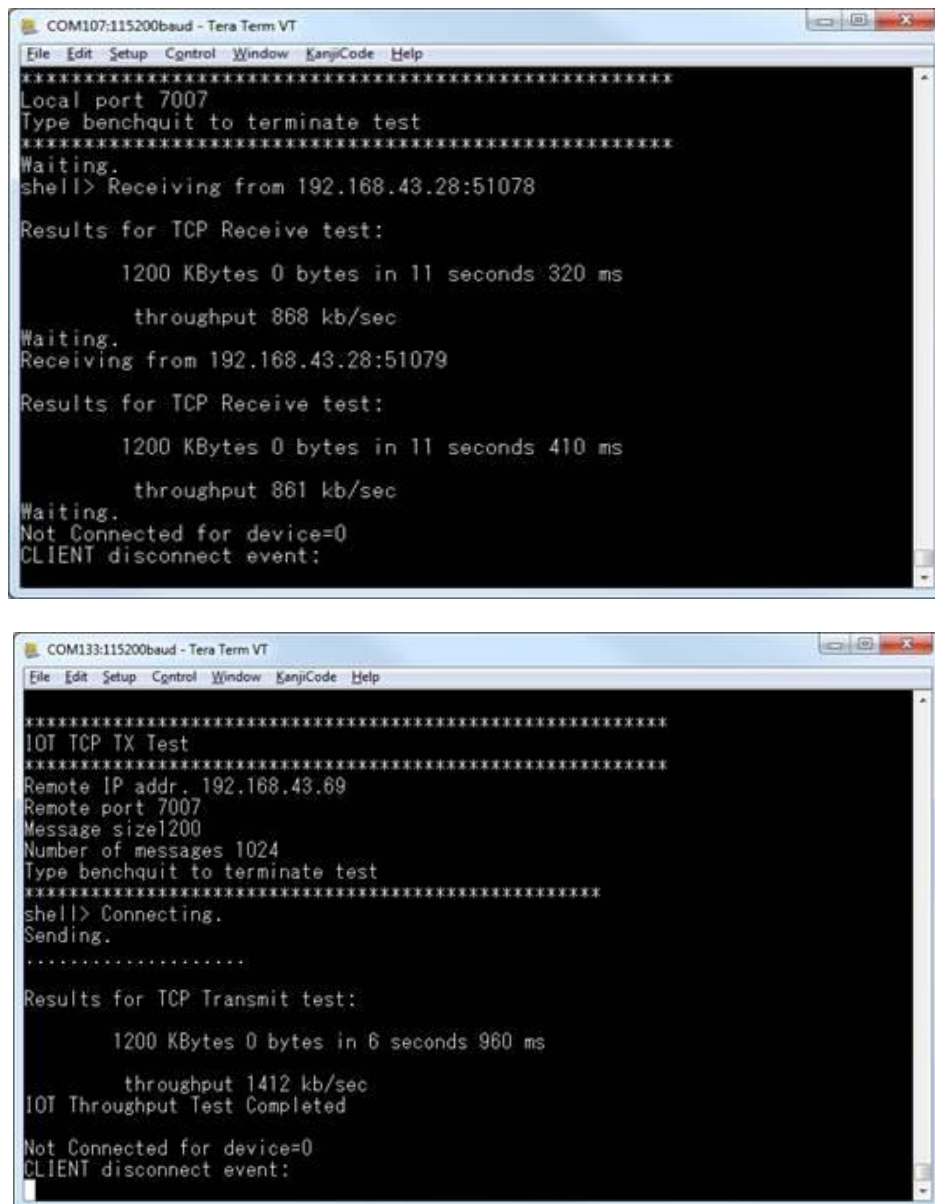
Command	Description
wmiconfig --p freescale	Provide the AP password
wmiconfig --wpa 2 CCMP CCMP	Setup the security and encryption protocol

<code>wmiconfig --connect GL-iNet-d2d</code>	Connect to AP (here called GL-iNet-d2d)
<code>wmiconfig --ipdhcp</code>	Ask the AP for IP address via DHCP
<code>ipconfig</code>	Check the IP provided by the AP
<code>benchmode v4</code>	Setup bench mode (Internet v4)
<code>benchrx tcp 7007</code>	Start listening on TCP port 7007

Table 2 – Device 2

Command	Description
<code>wmiconfig --p freescale</code>	Provide the AP password
<code>wmiconfig --wpa 2 CCMP CCMP</code>	Setup the security and encryption protocol
<code>wmiconfig --connect GL-iNet-d2d</code>	Connect to AP (here called GL-iNet-d2d)
<code>wmiconfig --ipdhcp</code>	Ask the AP for IP address via DHCP
<code>ipconfig</code>	Check the IP provided by the AP
<code>benchmode v4</code>	Setup bench mode (Internet v4)
<code>benchtx 192.168.8.145 7007 tcp 1024 1 2000 0</code>	Start the transmission speed test using Device 1 IP address and port number.

After the throughput test is completed, both shells display the connection speed, as shown in this figure:



The image contains two screenshots of Tera Term VT console windows. The top window, titled 'COM107:115200baud - Tera Term VT', shows the results of a TCP Receive test. It displays the local port as 7007 and shows two successful receive operations from 192.168.43.28:51078, each with a throughput of approximately 860 kb/sec. The bottom window, titled 'COM133:115200baud - Tera Term VT', shows the results of a TCP Transmit test. It displays the remote IP as 192.168.43.69 and remote port as 7007, with a message size of 1200 and 1024 messages. The transmit test shows a throughput of 1412 kb/sec.

```
COM107:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help
*****
Local port 7007
Type benchquit to terminate test
*****
Waiting.
shell> Receiving from 192.168.43.28:51078

Results for TCP Receive test:

      1200 KBytes 0 bytes in 11 seconds 320 ms

      throughput 868 kb/sec
Waiting.
Receiving from 192.168.43.28:51079

Results for TCP Receive test:

      1200 KBytes 0 bytes in 11 seconds 410 ms

      throughput 861 kb/sec
Waiting.
Not Connected for device=0
CLIENT disconnect event:

COM133:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help
*****
IOT TCP TX Test
*****
Remote IP addr. 192.168.43.69
Remote port 7007
Message size 1200
Number of messages 1024
Type benchquit to terminate test
*****
shell> Connecting.
Sending.
.....

Results for TCP Transmit test:

      1200 KBytes 0 bytes in 6 seconds 960 ms

      throughput 1412 kb/sec
IOT Throughput Test Completed

Not Connected for device=0
CLIENT disconnect event:
```

Figure 23: Throughput result displayed on consoles output

6 References

References to FRDM boards are available on NXP website:

<http://www.nxp.com/products/software-and-tools/hardware-development-tools/freedom-development-boards:FREDEVPLA?tid=vanFREEDOM>

References to SX-ULPAN-2401-SHIELD and SX-ULPAN-2401-SHIELD(US) are in Silex Website:

<https://www.silextechnology.com/connectivity-solutions/embedded-wireless/sx-ulpan-shield>

Full documentation can be downloaded after a quick registration:

<https://www.silextechnology.com/productspecs/sx-ulpan-2401-shield-product-specifications>

References to GT-202 boards are available on Qualcomm and Arrow website:

<https://developer.qualcomm.com/hardware/qca4002-4?fsrch=1&sr=1&pageNum=1>

<https://www.arrow.com/en/products/search?q=GT202>

7 Revision history

Rev.	Date	Substantive change(s)
0	07/2018	Initial revision based on FRDMGT202QSG

How to Reach Us:

Home Page:

www.nxp.com

Web Support:

www.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, Kinetis, and Freedom are trademarks of NXP B.V. All other product or service names are the property of their respective owners.

ARM, the ARM Powered 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.

