NXP Semiconductors
Application Note

Document Number: AN11981
Rev. 0, 06/2017

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.

Contents

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.

2.2 Hardware Connections

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

Table 1. **Signal and FRDM Pinout**

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

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:

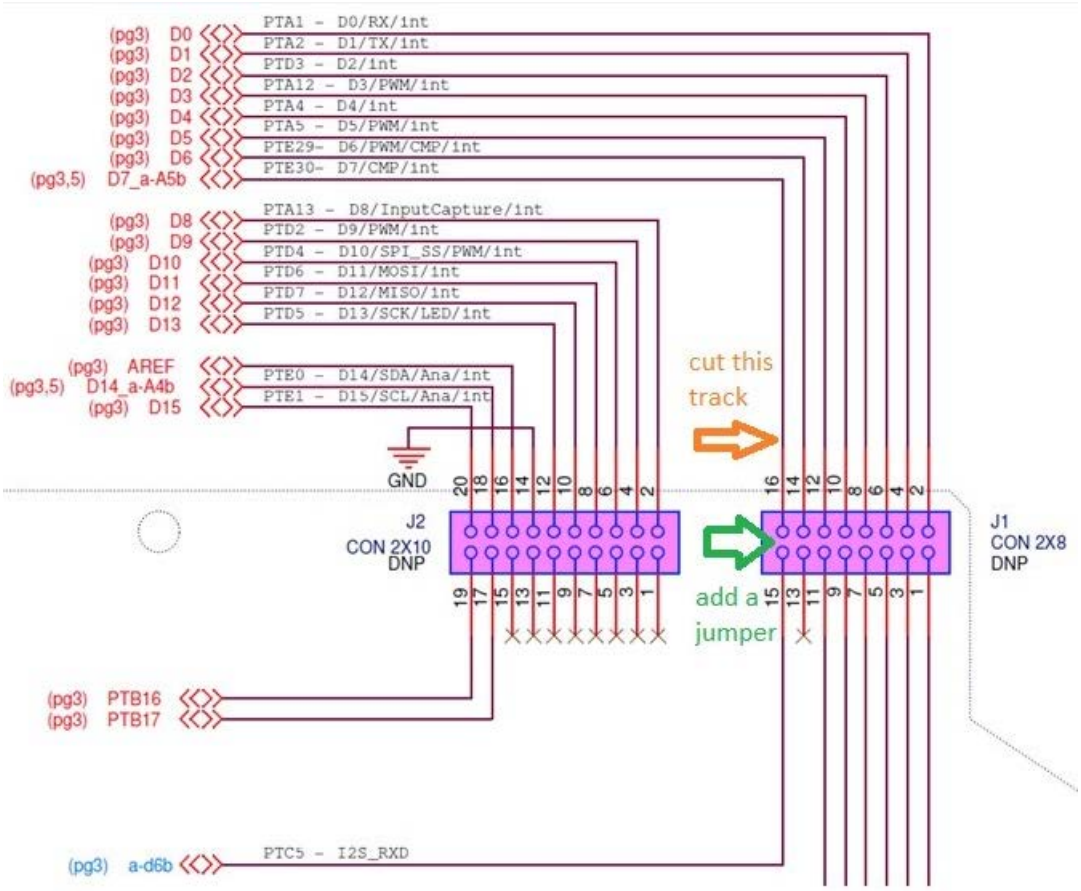

Figure 1. FRDM-KL43Z rework

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.

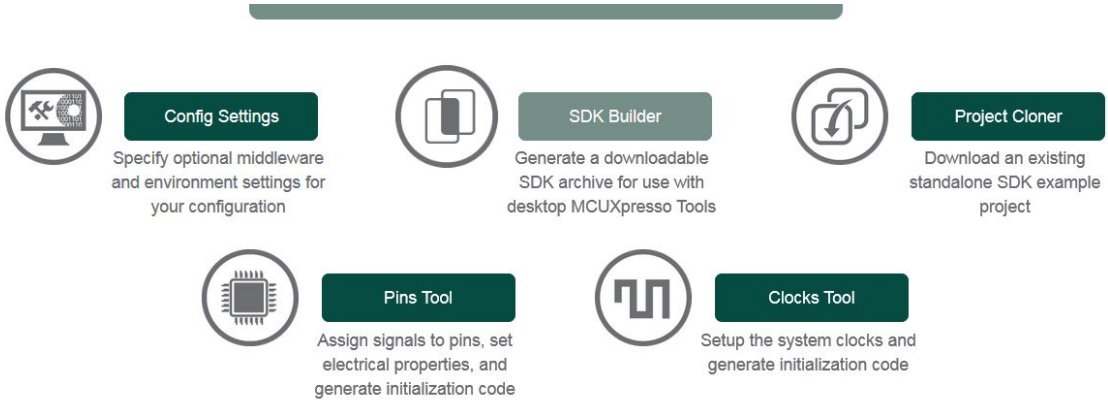

Figure 2. MCUXpresso Config Tools online version

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

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.

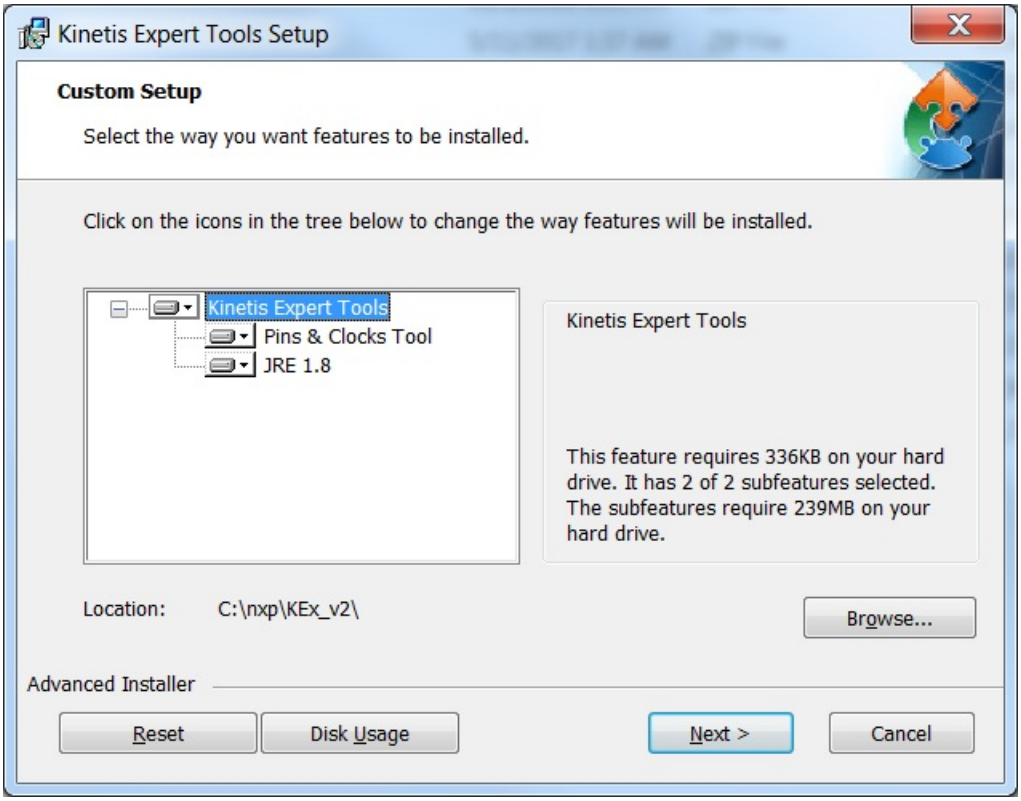

Figure 3. Installing MCUXpresso Config tools

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.

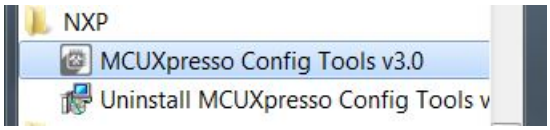

Figure 4. MCUXpresso for Windows® OS 7

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.

1. 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.

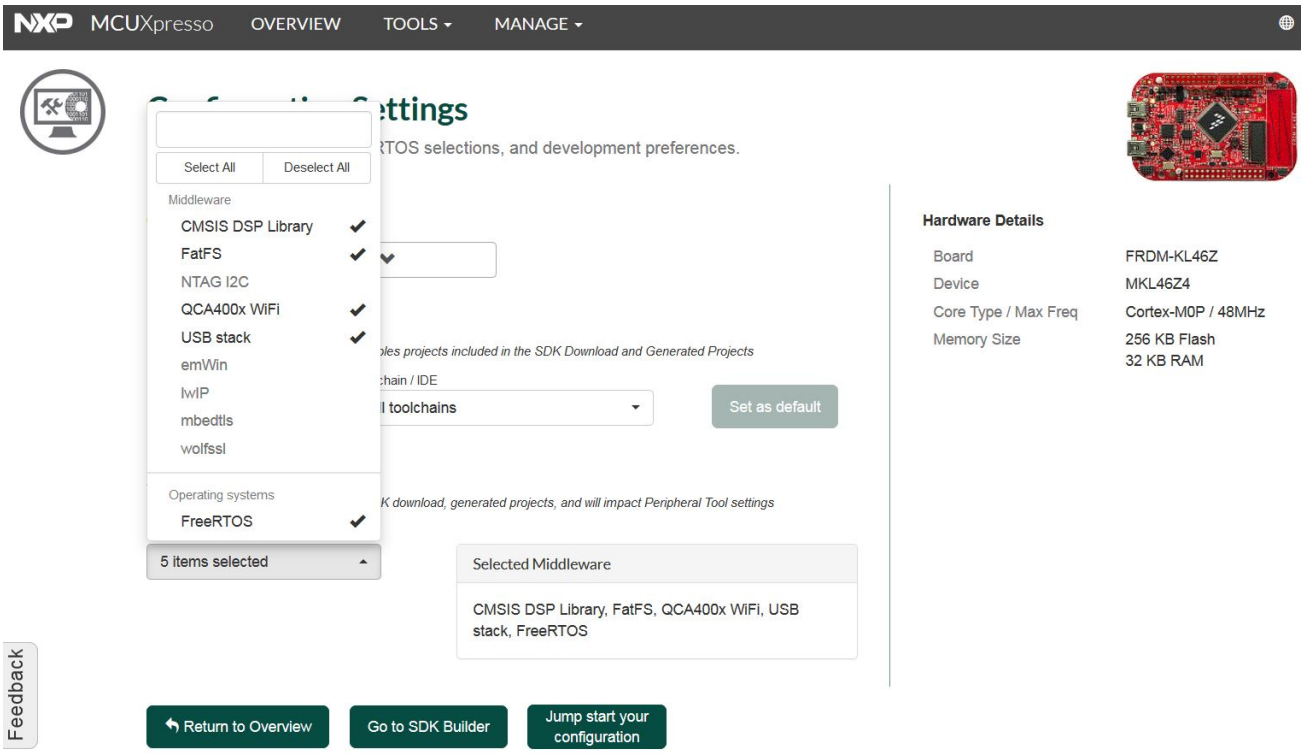

Figure 5. FRDM-KL46Z SDK download

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-KL43Z*

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.***

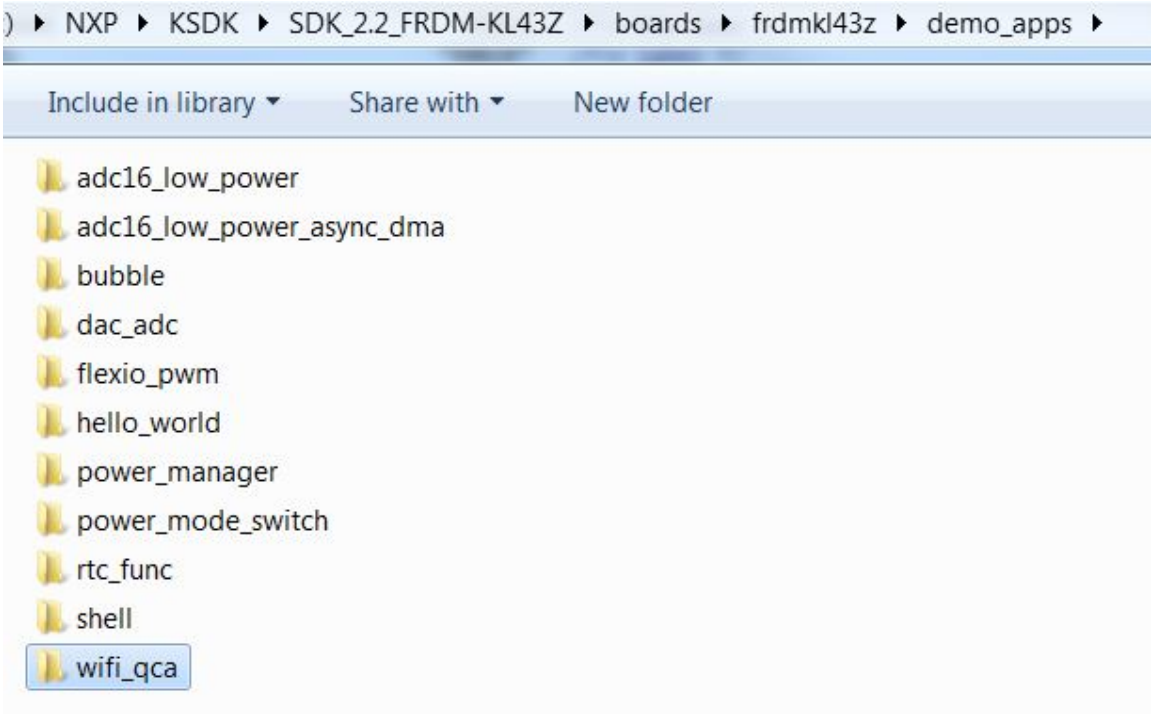

Figure 6. MCUXpresso SDK FRDM-KL43Z examples folder after copy

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.

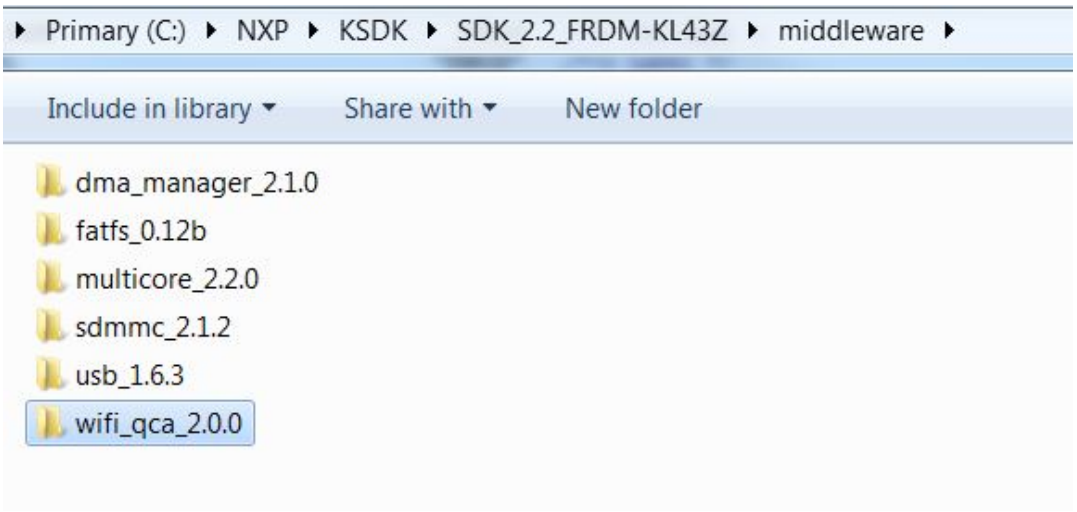

Figure 7. MCUXpresso SDK FRDM-KL43Z middleware folder after copy

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*>.

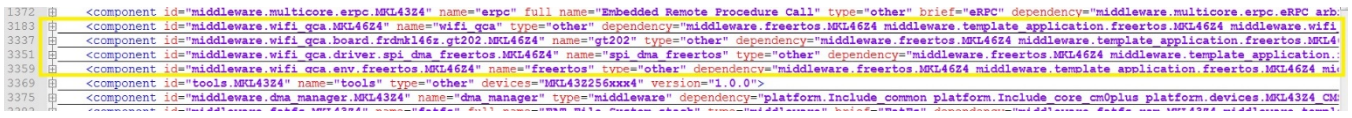

Figure 9. Adding example components to manifest file

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.

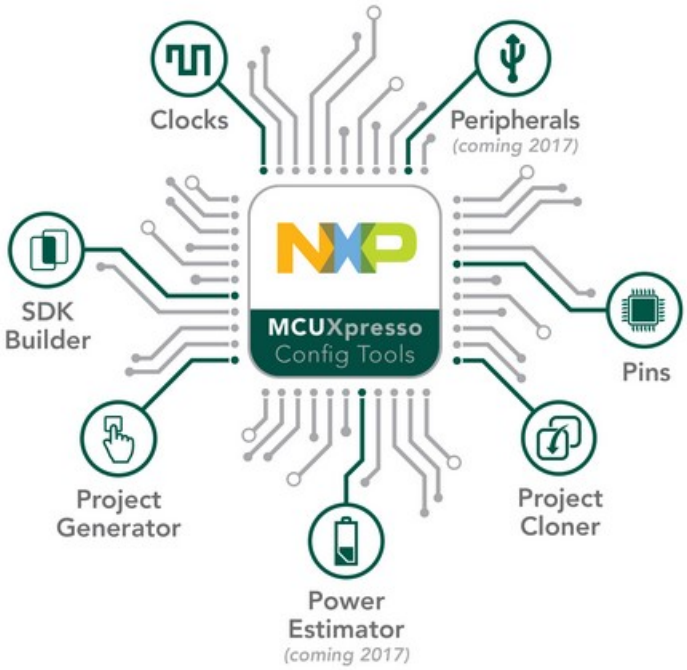

Figure 10. MCUXpresso tools

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.

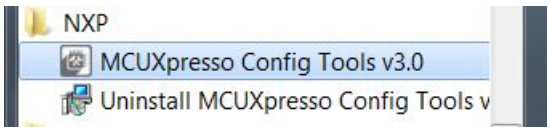

Figure 11. Windows OS start menu

2. Select the required MCUXpresso SDK to use:

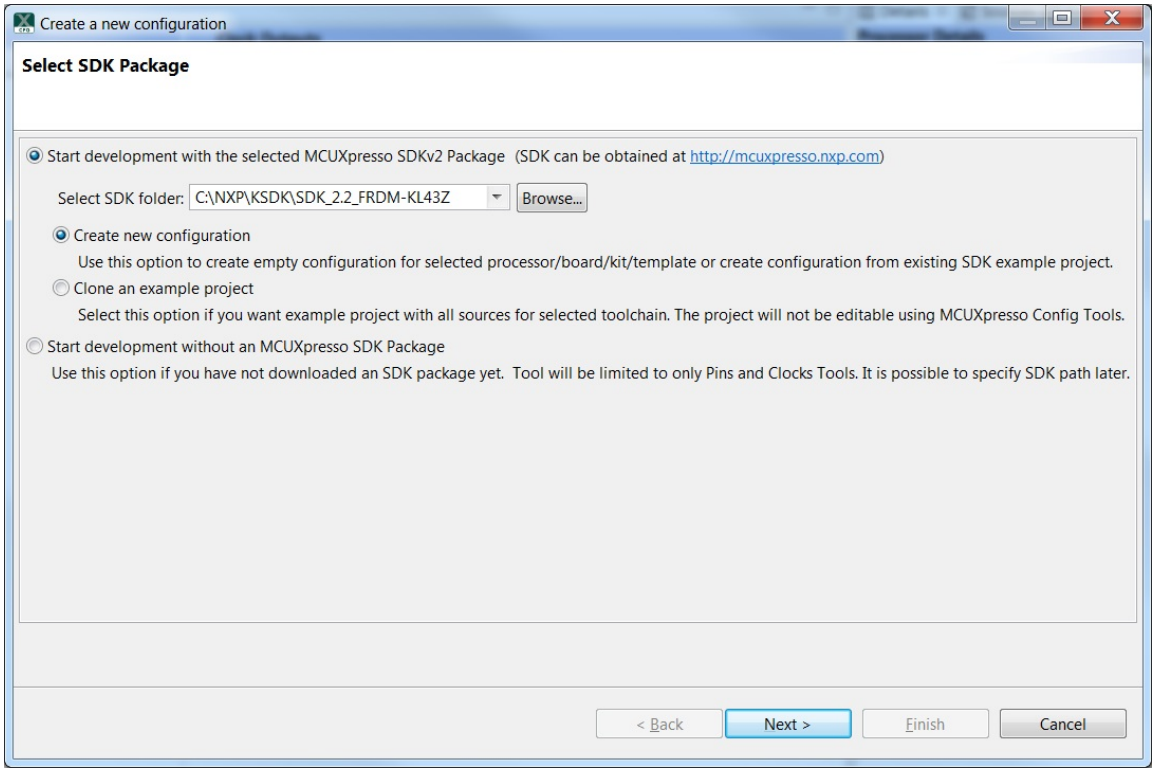

Figure 12. Selecting MCUXpresso SDK

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**.

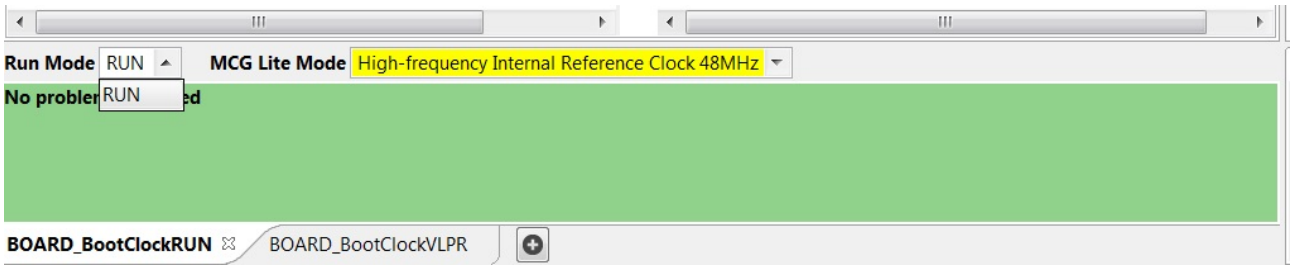

Figure 13. Selecting boot clock

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.

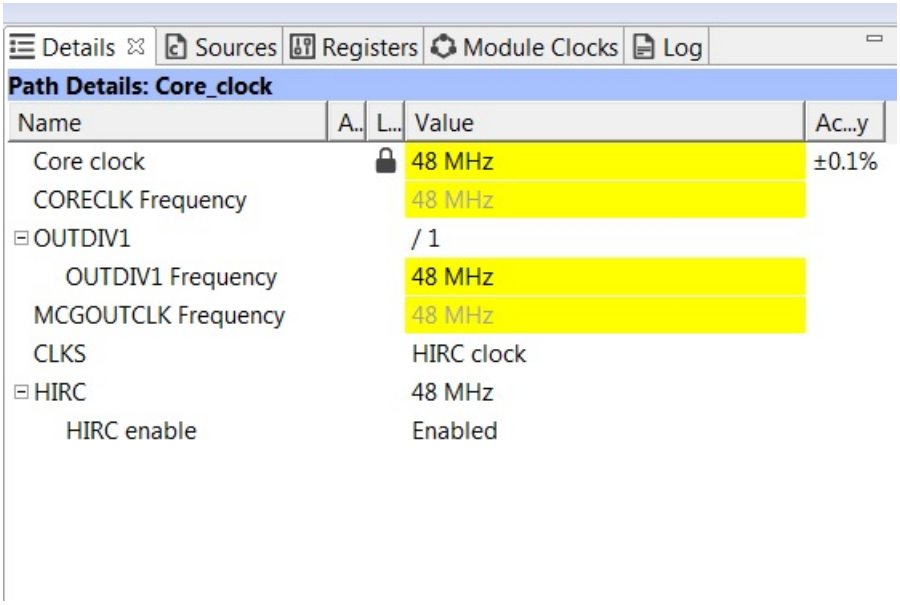

Figure 14. Set RUN clock

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

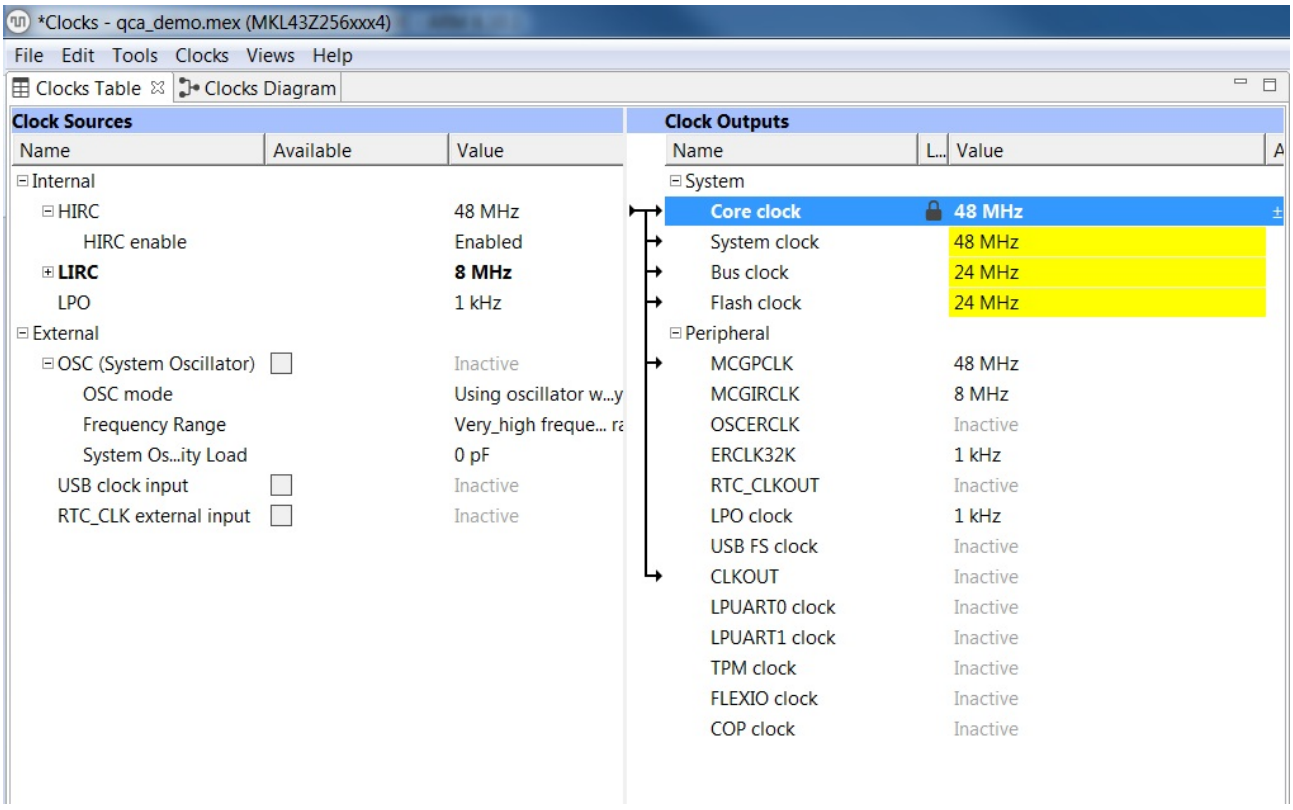

Figure 15. RUN clock final config

3. Now, select **BOARD_BootClockVLP**, and change clock source to LIRC in Core clock.

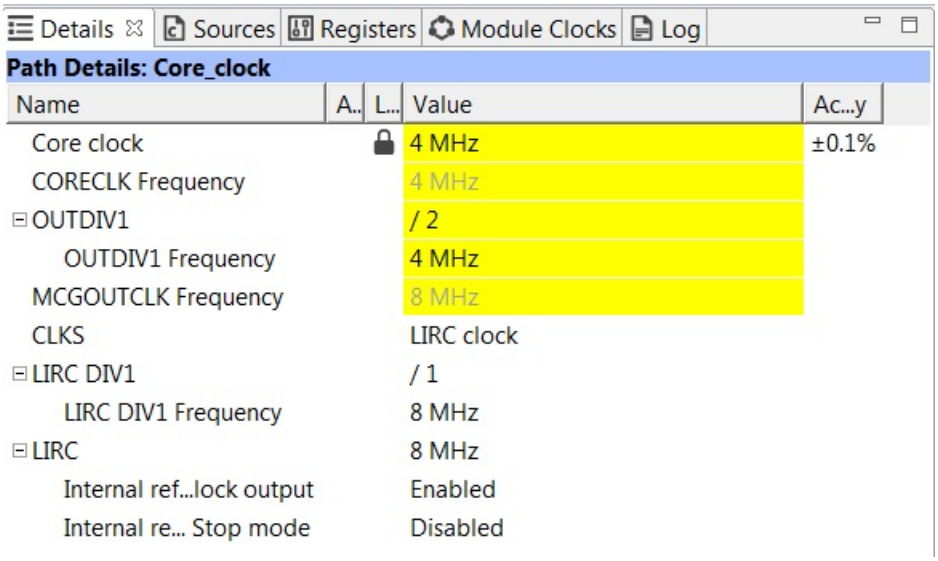

Figure 16. Set VLPR Core clock

The final VLPR configuration is shown in Figure 17:

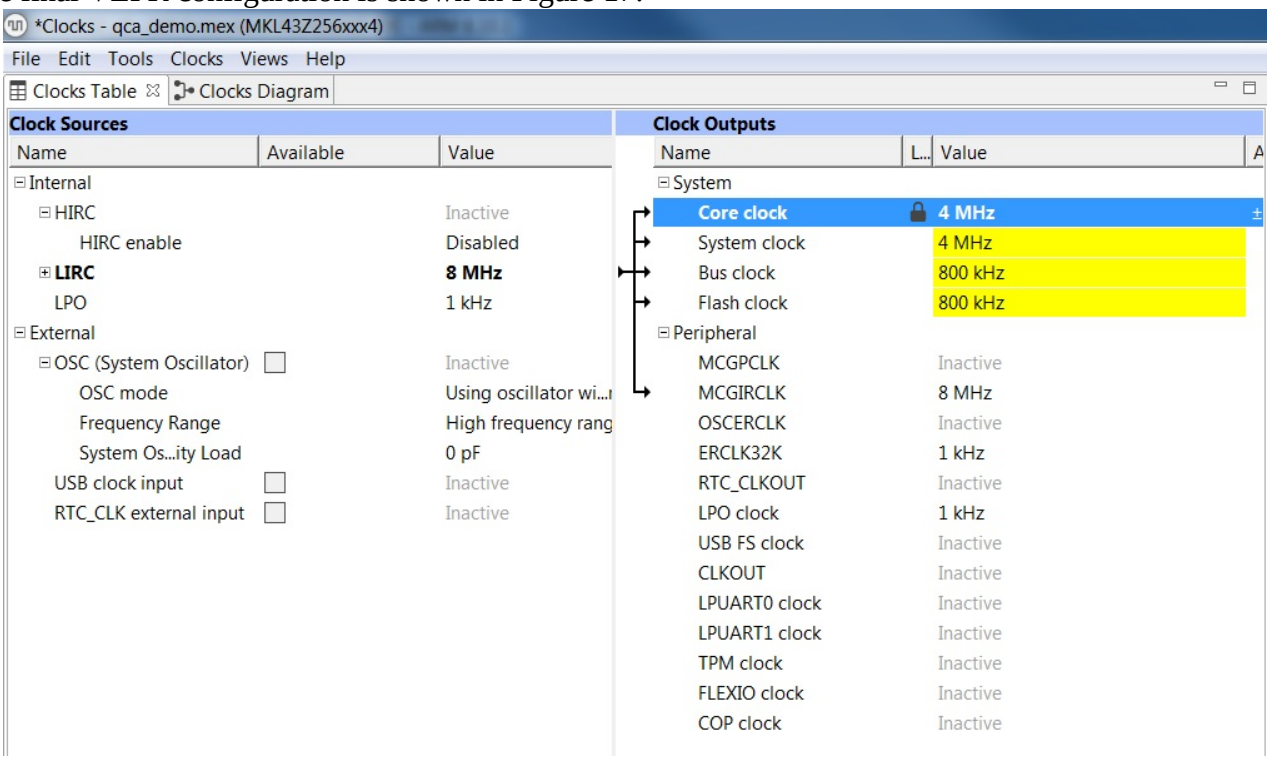

Figure 17. Final VLPR clock config

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:

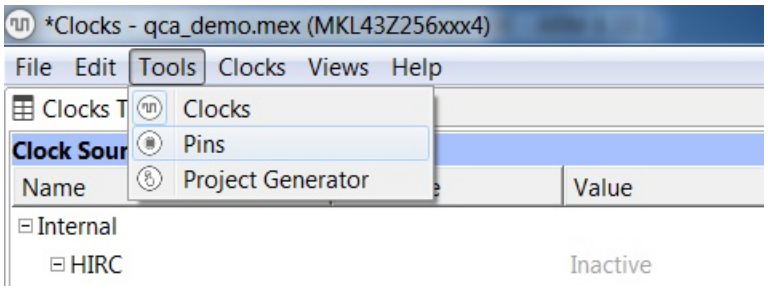

Figure 18. Move to pins configuration

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

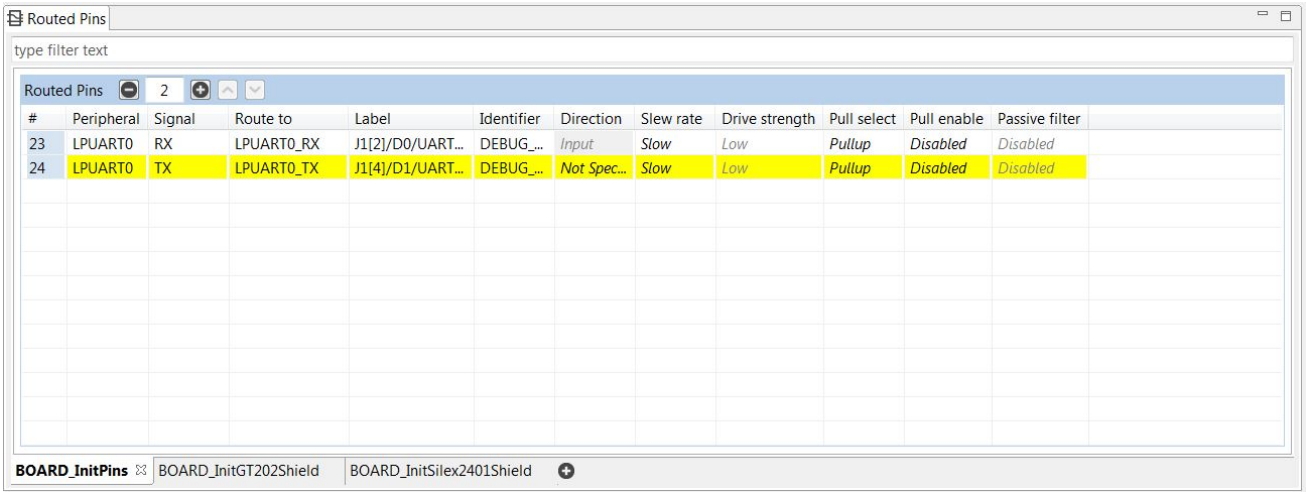

Figure 19. Init pins

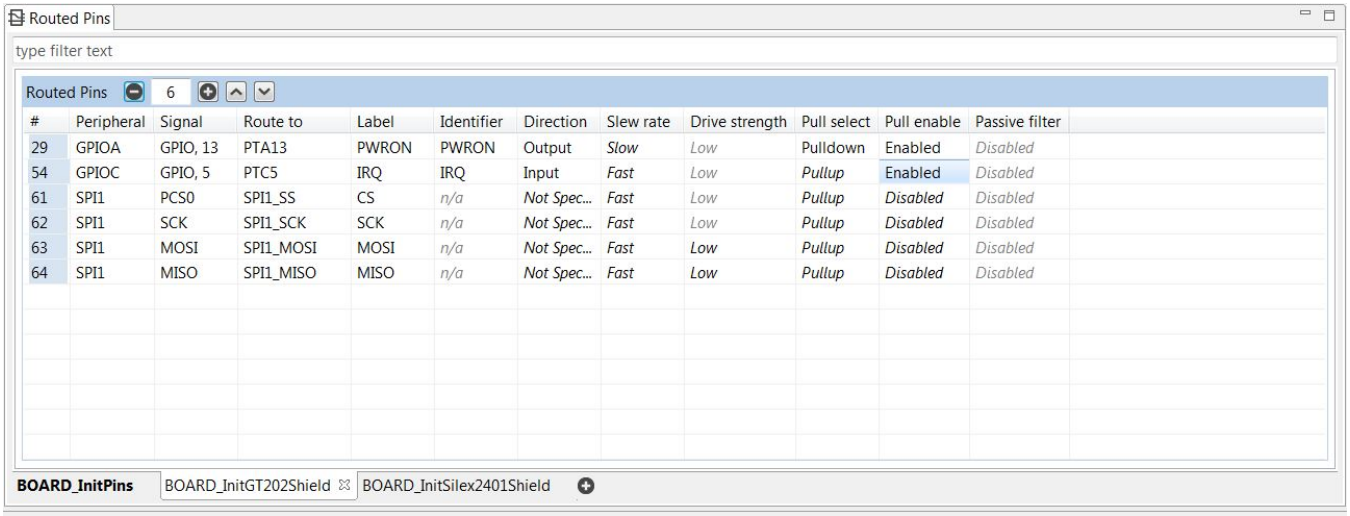

Figure 20. Specific init pins

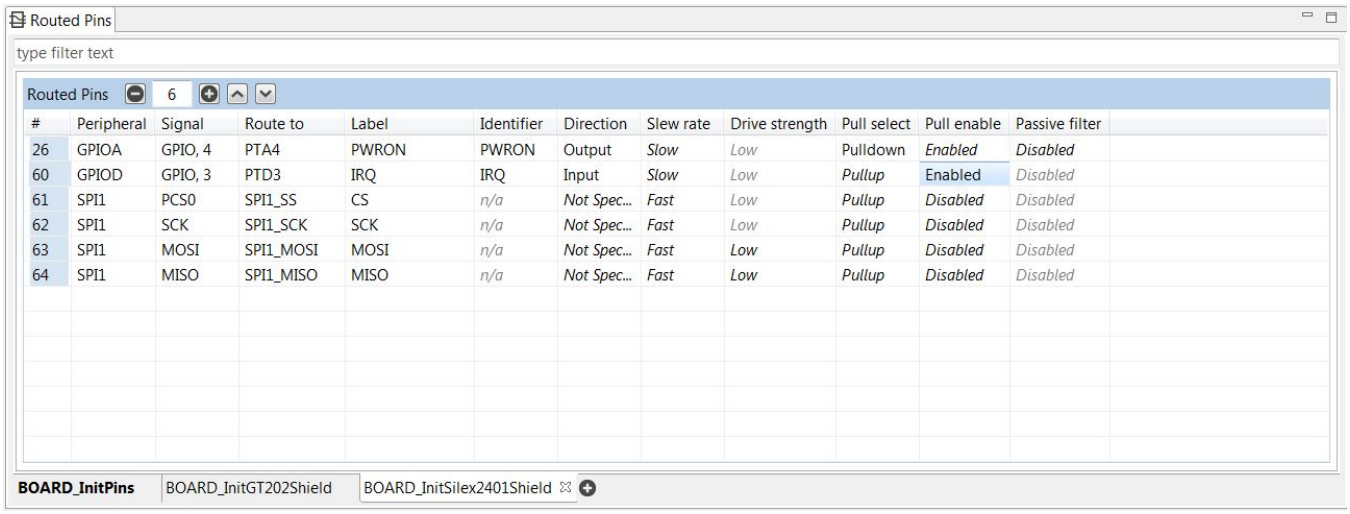

Figure 21. Silex2401 specific init pins

5.2.4 Project Generator

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

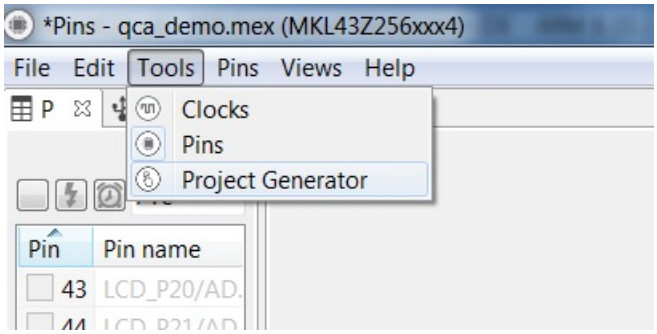

Figure 22. Selecting Project Generator

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.

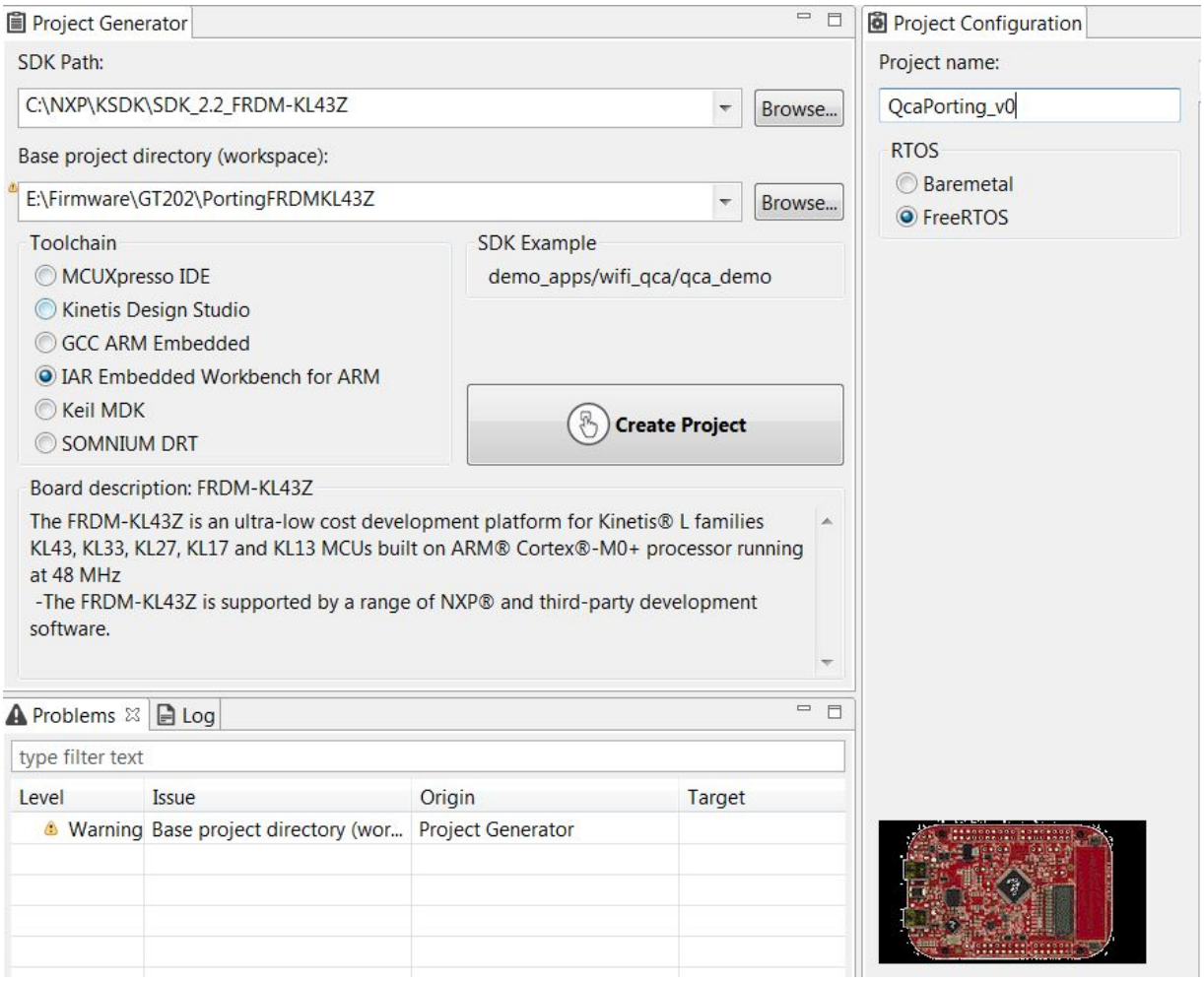

Figure 23. Project structure configuration

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.

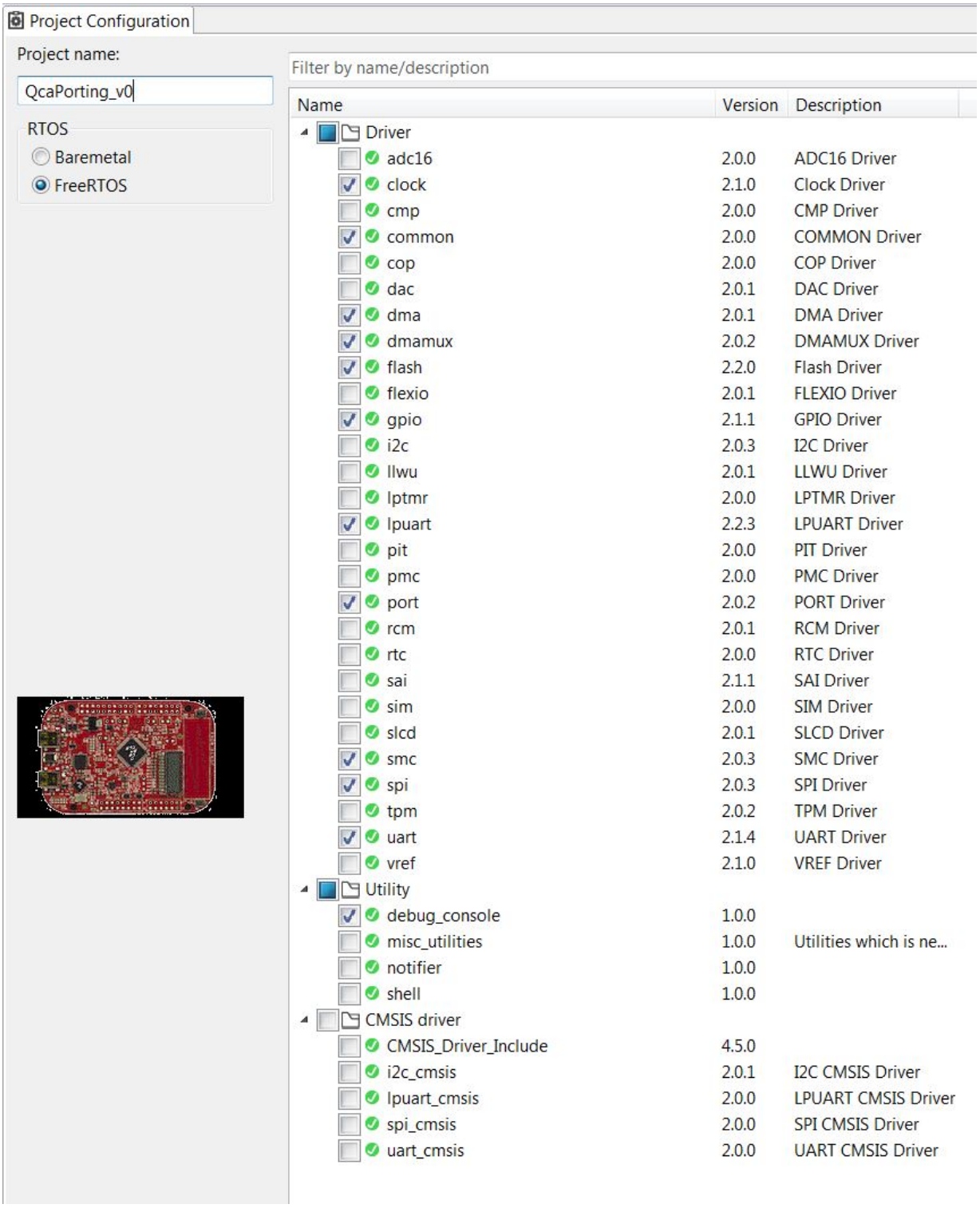

Figure 24. MCUXpresso SDK Drivers selection

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

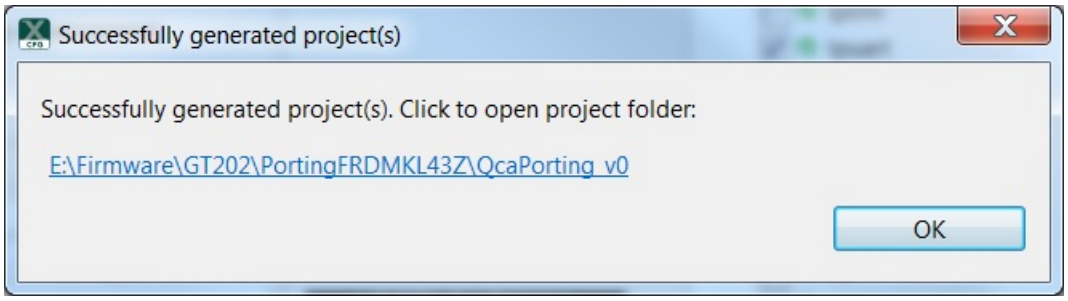

Figure 25. Project generated

The project will be in selected folder:

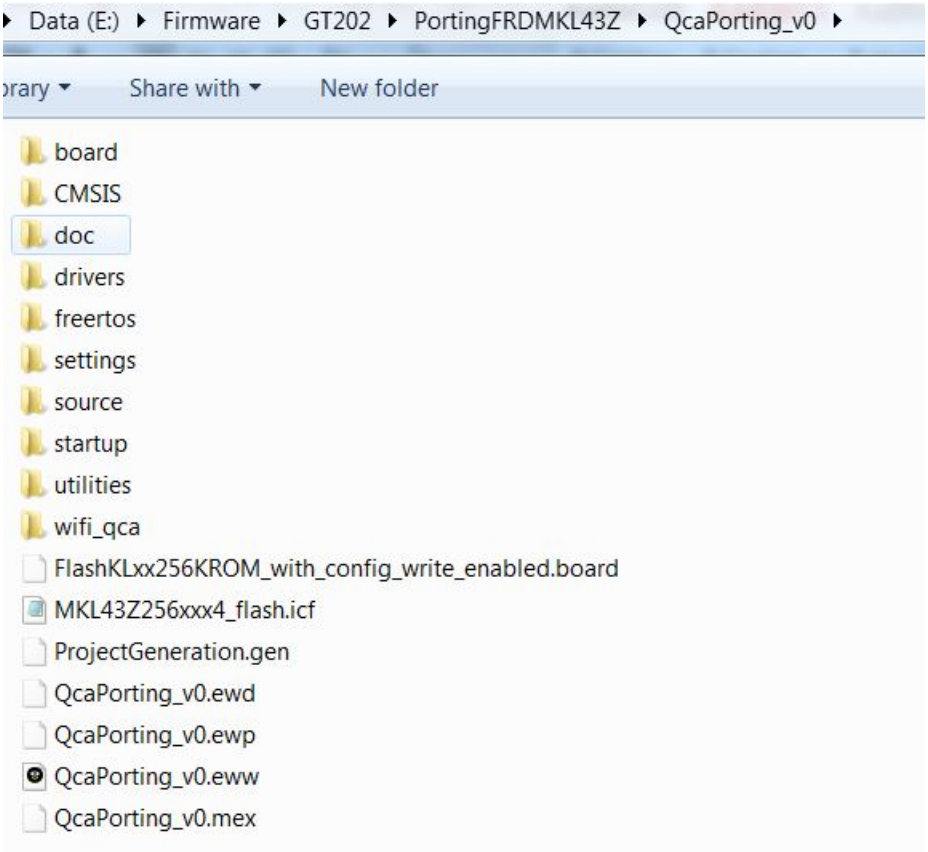

Figure 26. Migrated project folder

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.

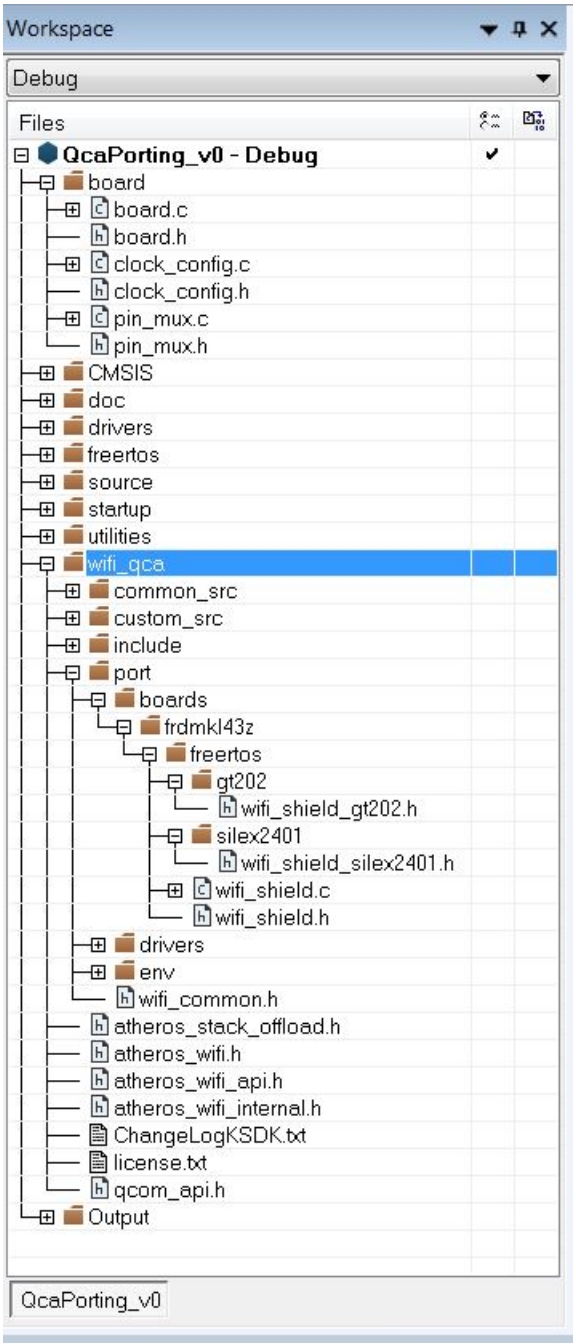

Figure 27. Workspace for standalone project

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.

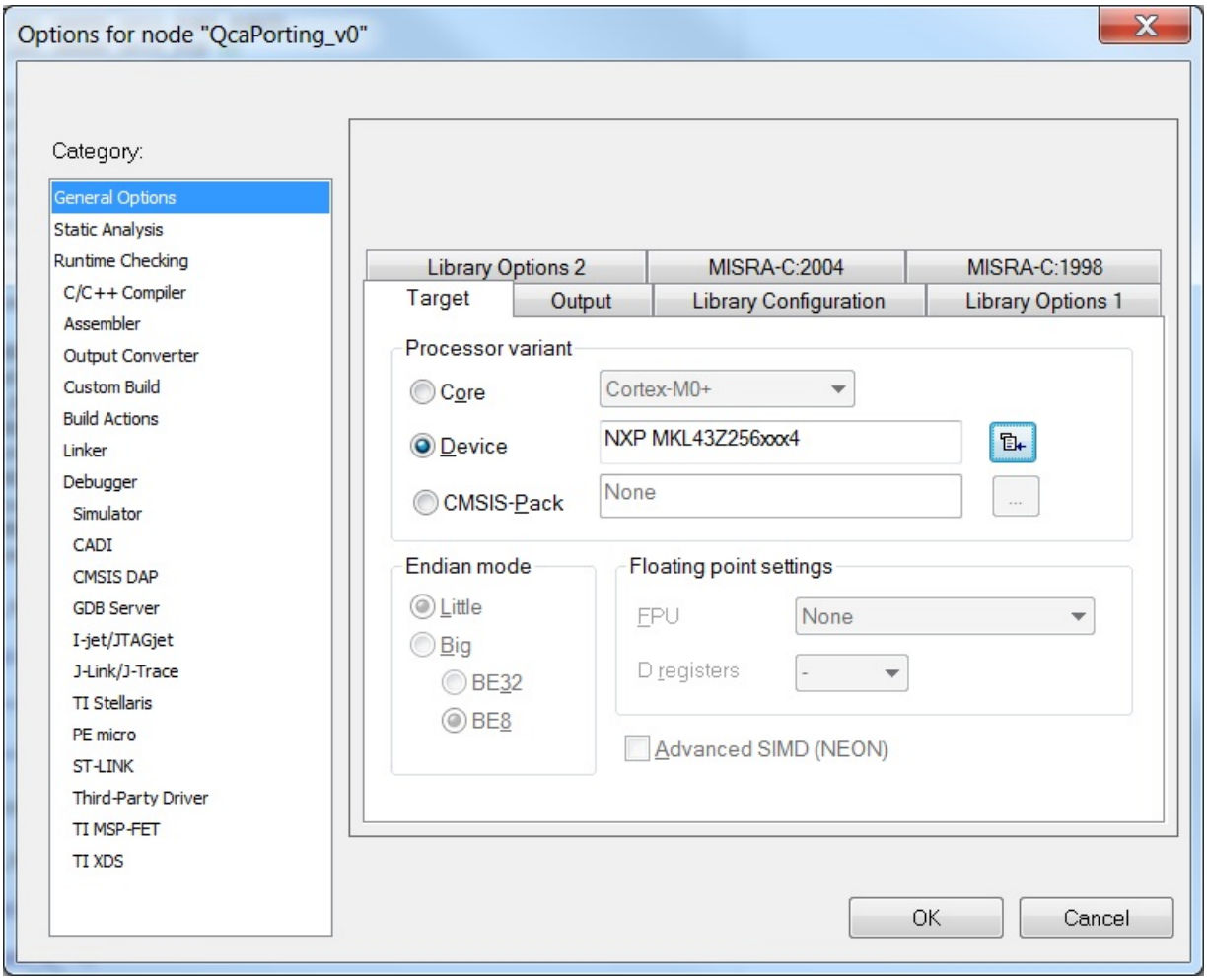

Figure 28. Select device

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

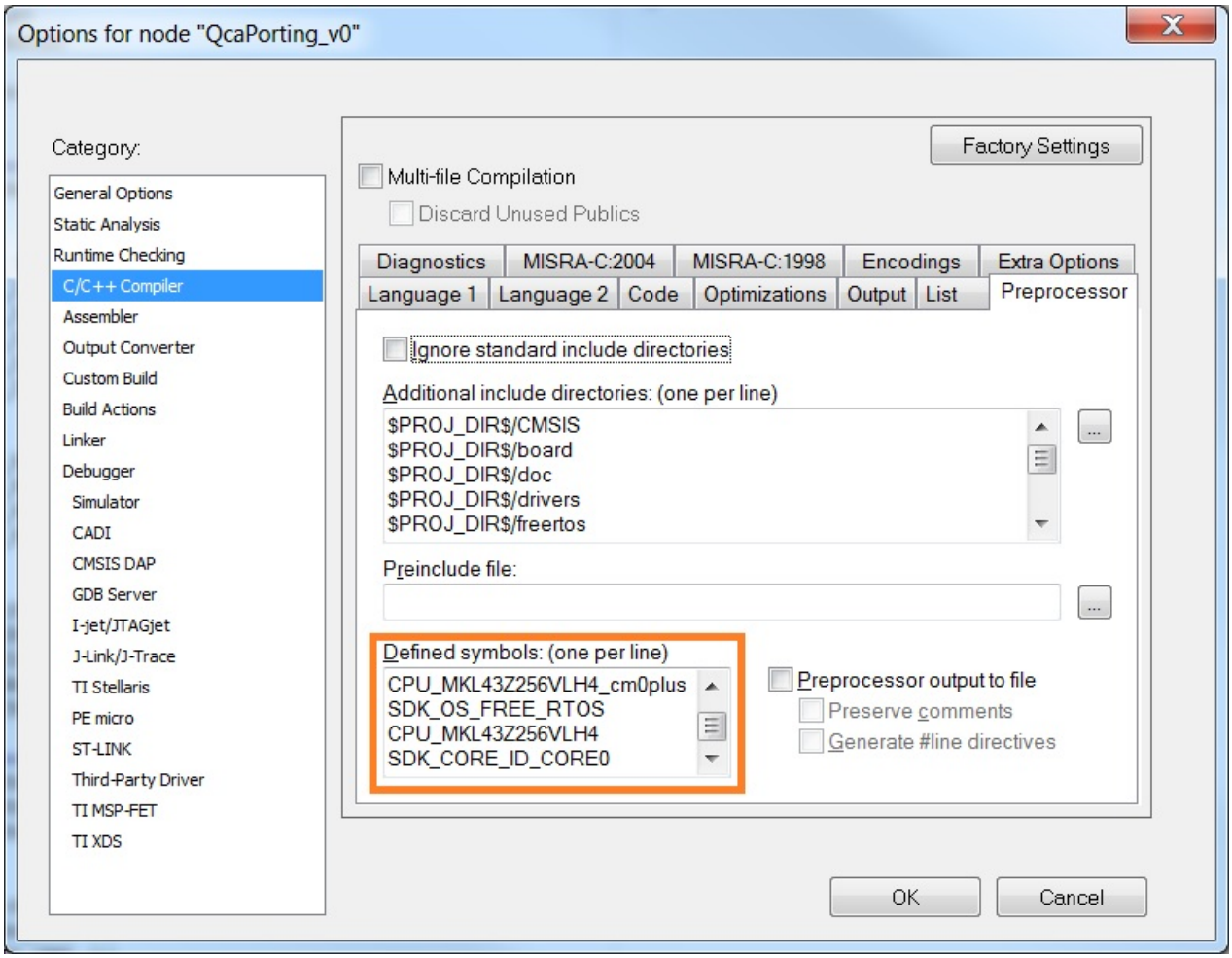

Figure 29. Symbol references

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.

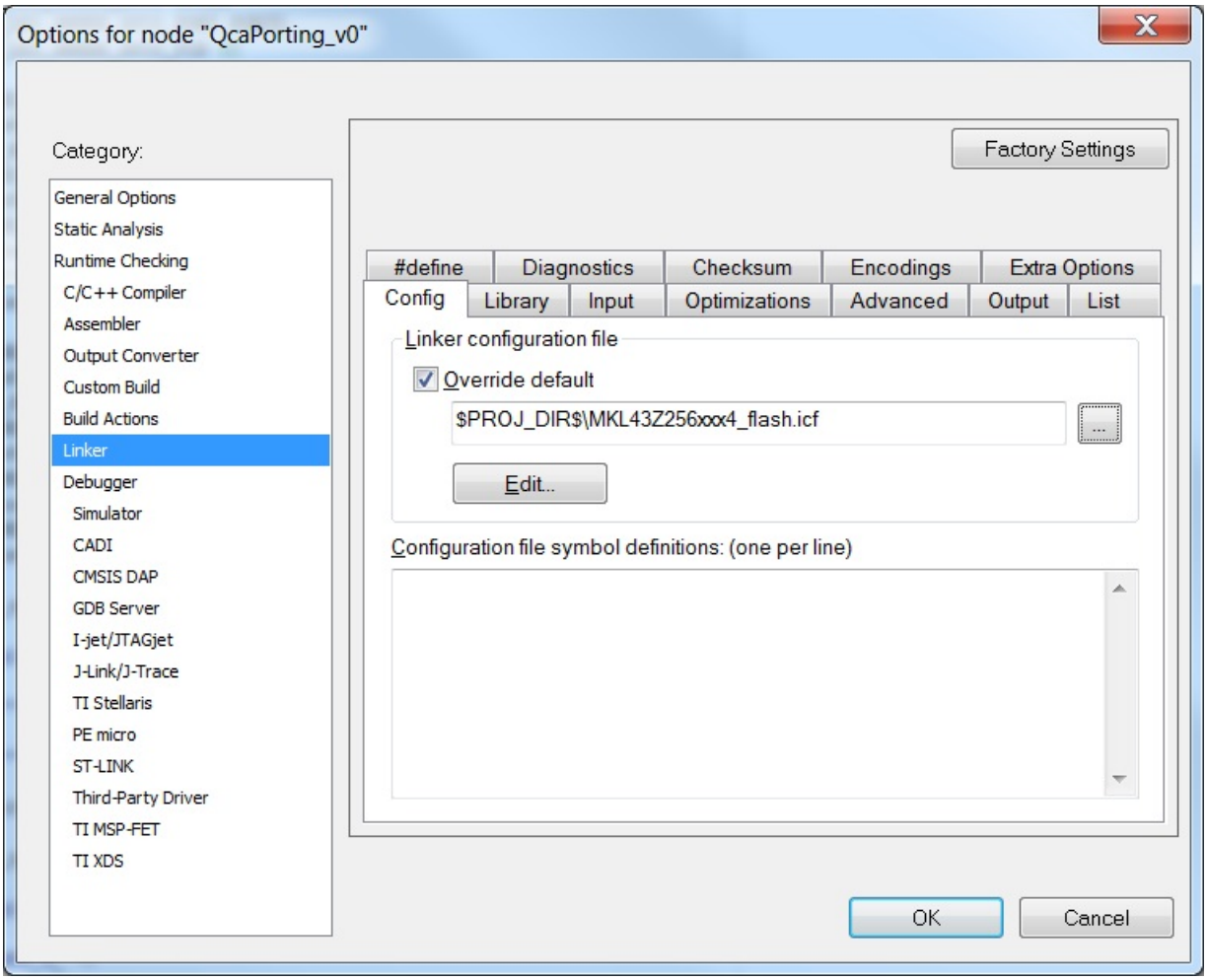

Figure 30. Linker configuration file

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.

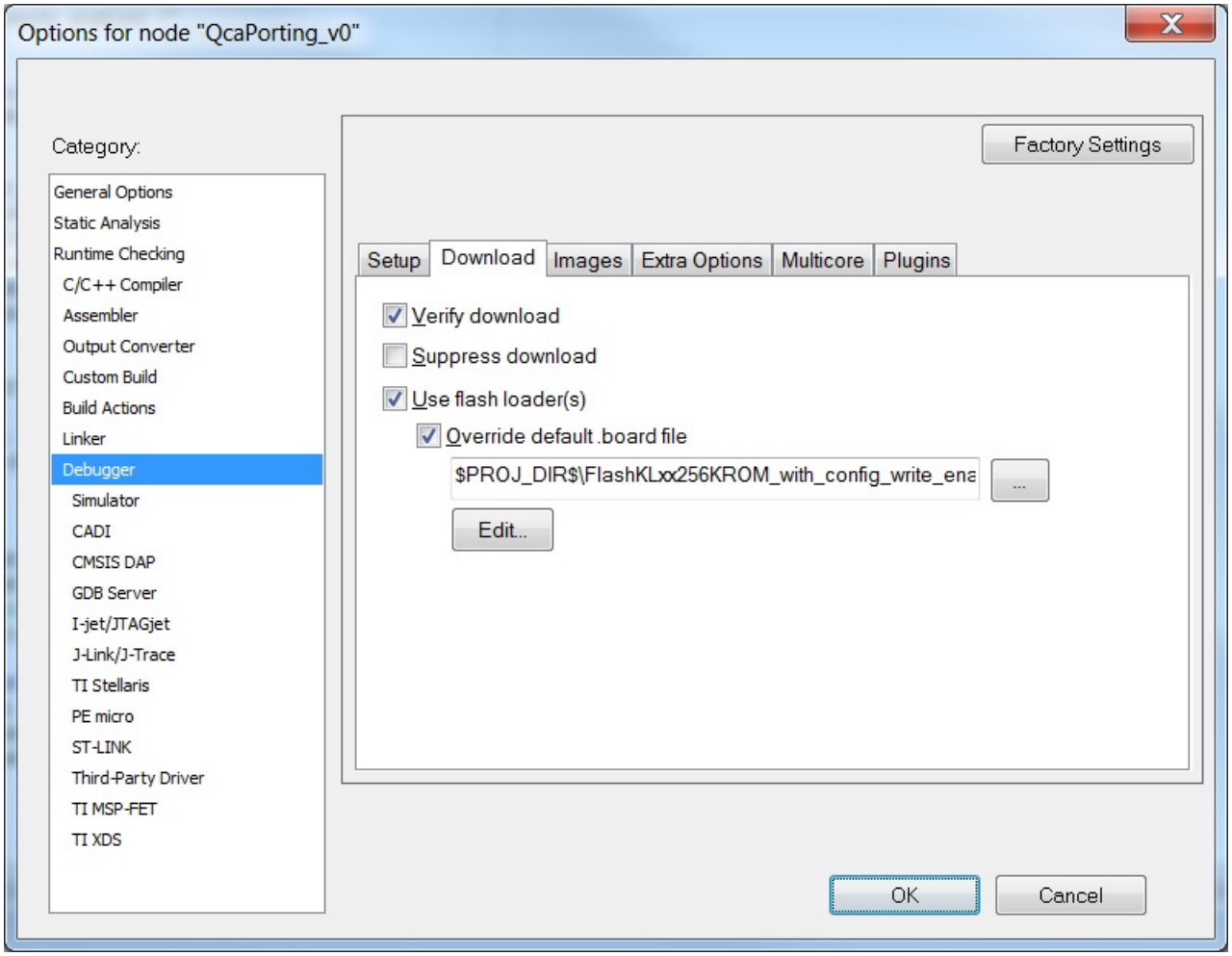

Figure 31. Include flash loader

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

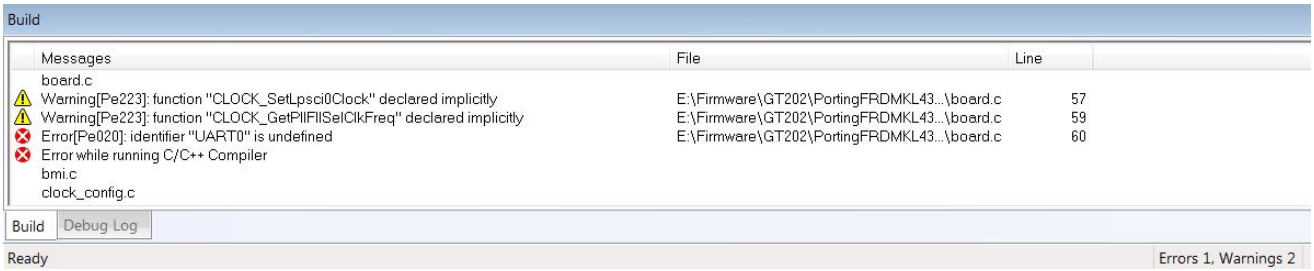

Figure 32. Build error log

7. UART0 is selected as default debug interface. LPUART0 The debug interface for FRDM-KL43Z is LPUART0. All clock references need to be fixed as well.

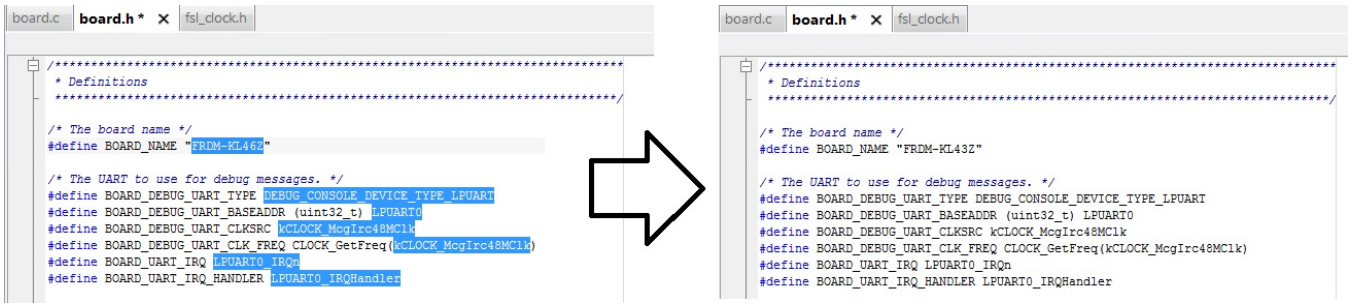

Figure 33. Fix UART0 definitions

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.c | board.h | fsl_clock.h | **wifi_shield_gt202.h** | 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)
```

Figure 35. Wi-Fi Shield configuration file

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

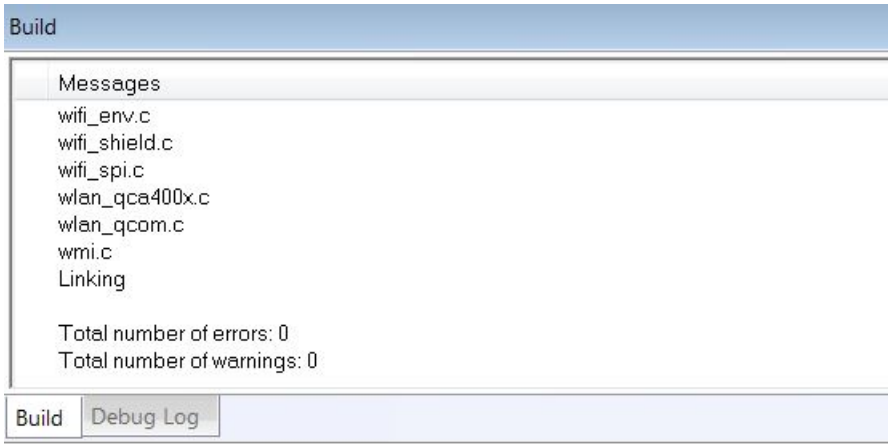

Figure 36. Final build

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”.

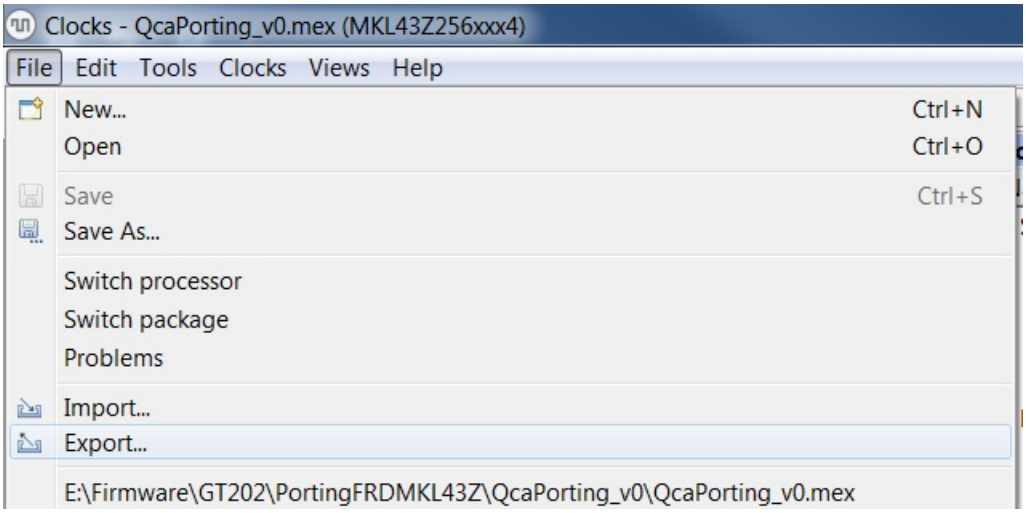

Figure 37. Exporting files in MCUXpresso Config tools

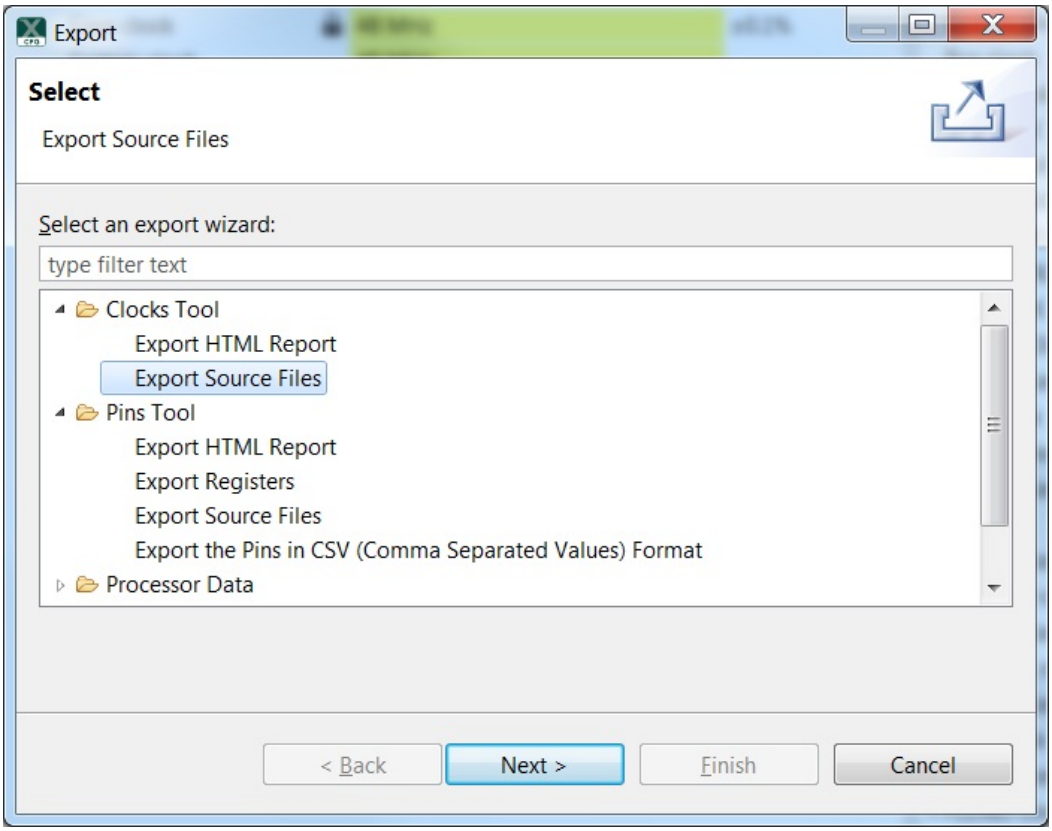

Figure 38. Export Clock configuration files

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.

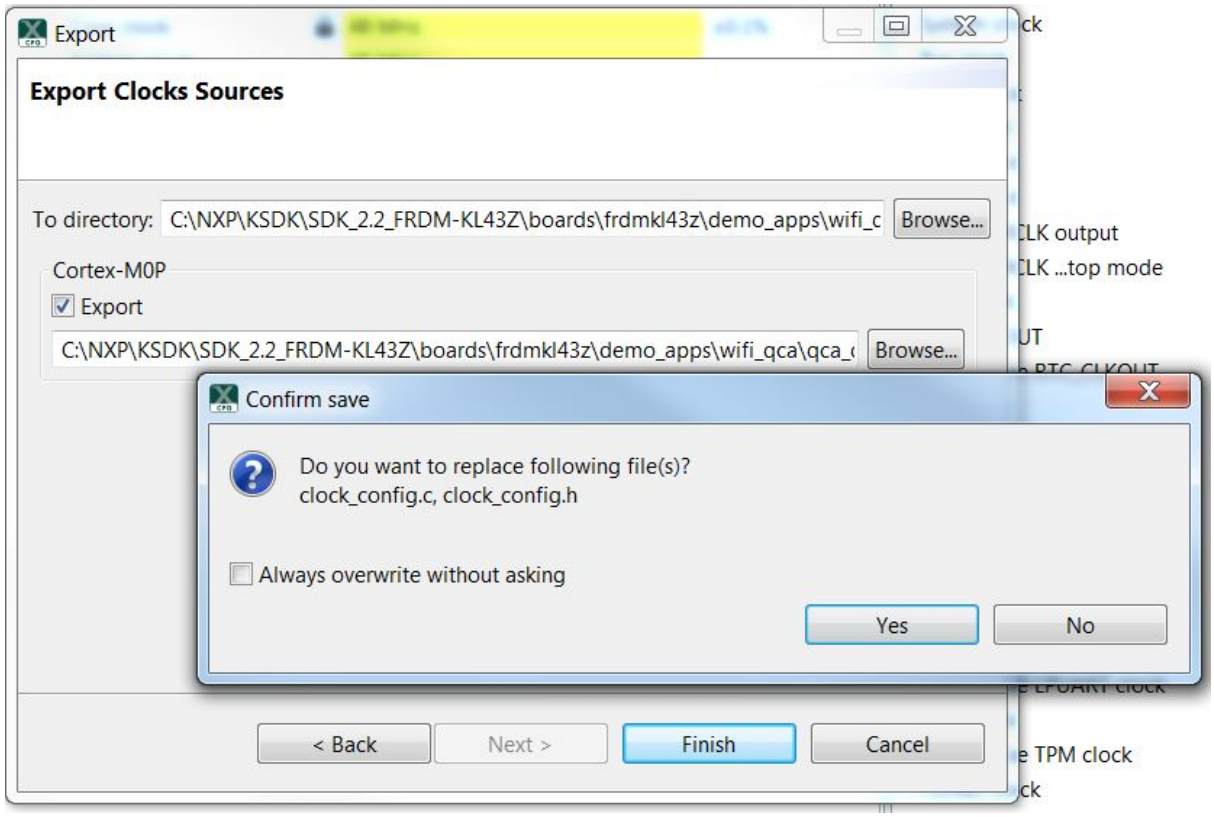

Figure 39. Exporting clocks config files

5.3.2 Fix IAR Configuration files

1. 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>
2168    </file>
2169    <file>
2170      <name>$PROJ_DIR$/../../../../../../devices/MKL43Z4/drivers/fsl_uart.c</name>
2171    </file>
2172    <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.

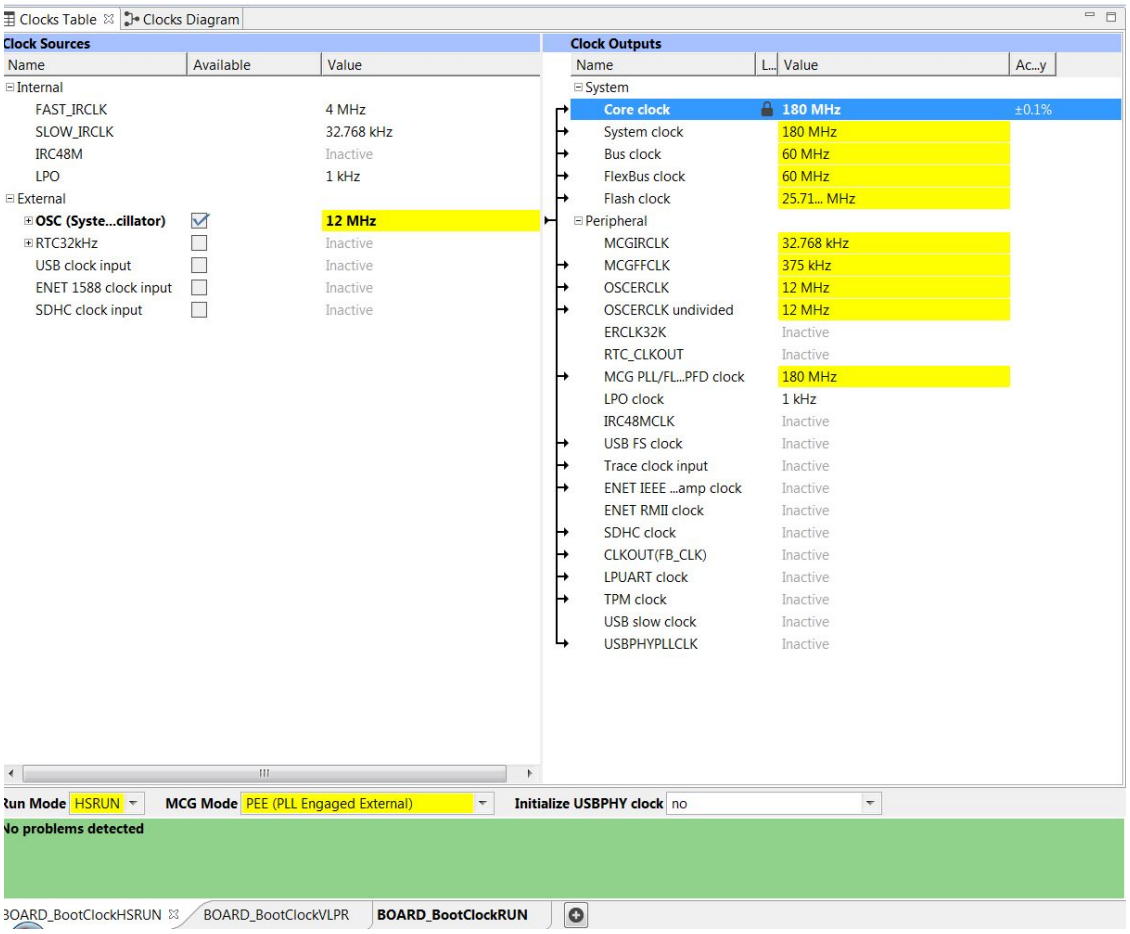

Figure 41. Configuring HSRUN in FRDM-K66

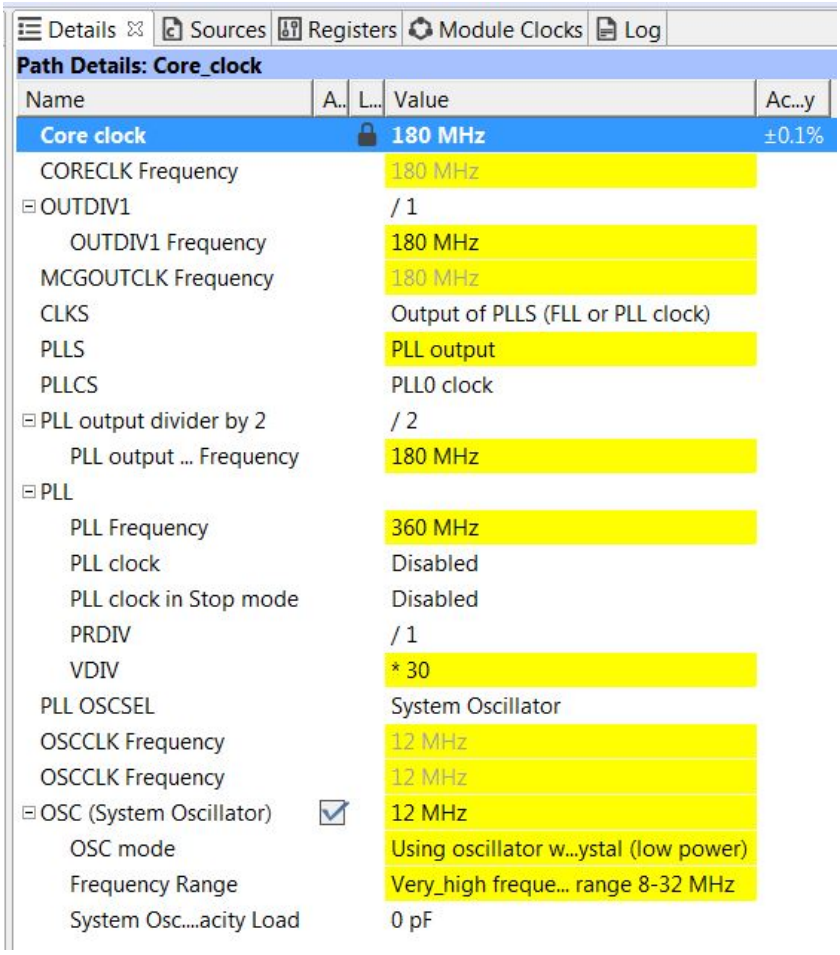

Figure 42. FRDM-K66 HSRUN Core Clock

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

Document Number: AN11981
Rev. 0
06/2017