NXP Semiconductors
User's Guide

Document Number: USBPDMUG
Rev. 0, 10/2017

MCUXpresso USB PD Migration Guide

1. About this document

This document describes how to migrate from the FRDM-KL27Z USB Power Delivery software middleware to SDK v2.2-based platforms.

The migration between the two devices requires software changes and may require some hardware modifications, such as jumper configurations.

Contents

2. System overview

The USBPD-C-SHIELD uses Arduino-compatible headers to communicate between the PTN5110 Type-C Port Controller (TCPC) USB Power Delivery PHY and the FRDM-KL27Z working as a Type-C Port Manager (TCPM).

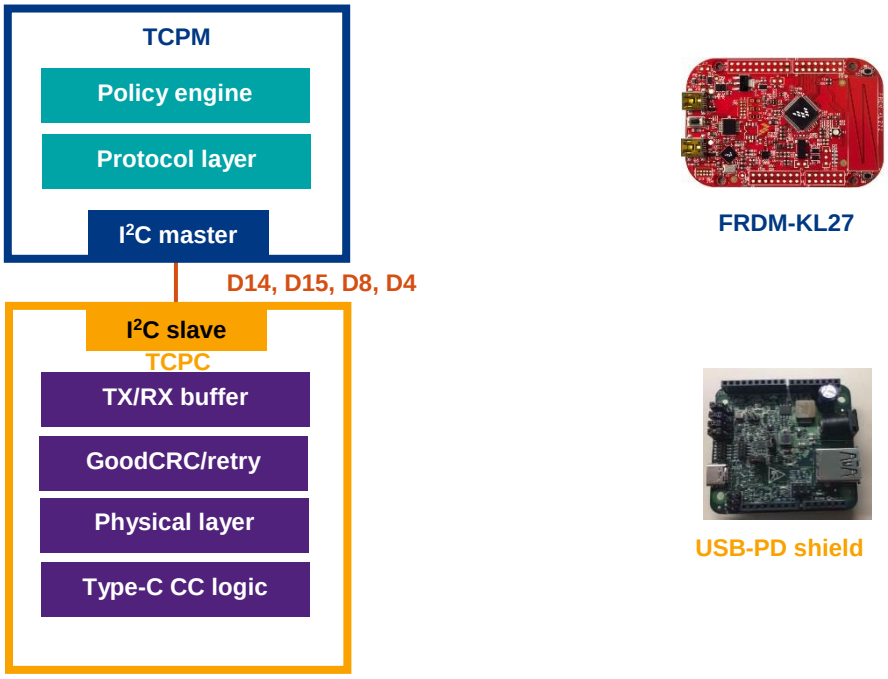

Figure 1. System overview

3. Hardware considerations

The communication between the TPCM MCU and the USB-PD shield requires six Arduino signals. To drive the PTN5110, two pins (D14, D15) are used for the I^2C and a third one (D8) as an interrupt-capable input nALERT. There is a second I^2C at A4 and A5 (multiplexed with D14 and D15) to provide flexibility on platforms using the I^2C instance at D14 and D15.

The TCPM output pin EXTRA_EN_SRC (D4) enables or disables the NX20P5090 unidirectional power switch that provides 9 V to the VBUS. Table 1 shows the pinout for different Kinetis FRDM boards.

Table 1. USB PD-C-SHIELD pinout

Arduino	Name	Shield	FRDM-KL27	FRDM-K64 revE
D4	EXTRA_EN_SRC	J403-05	PTA13	PTB23
D8	nALERT	J404-01	PTE31	PTC12
D14	PTN5110_SDA	J404-09	PTD6 I2C1	PTE25 I2C0
D15	PTN5110_SCL	J404-10	PTD7 I2C1	PTE24 I2C0
A4	SDA	J405-05	PTB1 I2C0	PTC11 I2C1
A5	SCL	J405-04	PTB0 I2C0	PTC10 I2C1

The usb_pd_freertos demo application uses two switches. One controls the power requests and the other switch calls for the power change. Table 2 shows the switch assignment (depending on the development board).

Table 2. Switch assignment

Switch	FRDM-KL27		FRDM-K64	
Power request	SW1	PTA4	SW2	PTC6
Power change	SW3	PTC1	SW3	PTA4

4. Software migration

4.1. FRDM-K64F rev. E

The FRDM-KL27Z USB Power Delivery MCUXpressso SDK middleware software enables you to migrate to other Kinetis platforms. The following subsections show how to enable the USBPD-C-SHIELD on the FRDM-K64F revE platform.

4.1.1. IAR Embedded Workbench® IDE

1. Follow the instructions in Appendix A to build an SDK package.
2. Follow the instructions in Appendix B to build an SDK package for a selected platform (for example; FRDM-K64F) using the IAR IDE.
3. Use the Windows® OS Explorer to copy the content from the *...\boards\frdmk64f\project_template\cproject_generator_templates\iar* folder.
4. Open a second Explorer window and navigate to *...\boards\frdmk64f\usb_examples*. Add a new folder with the name of the USB Power Delivery example project being ported (for example; *usb_pd*).
5. Inside the recently-created folder (*usb_pd*), add a new folder named *freertos*.
6. Add a new directory inside the *freertos* folder and name it *iar*.
7. Paste the files copied in step 2 into the *...\boards\frdmk64f\usb_examples\usb_pd\freertos\iar* folder.
8. Replace all *$[project_name]* file names with *usb_pd_freertos*.

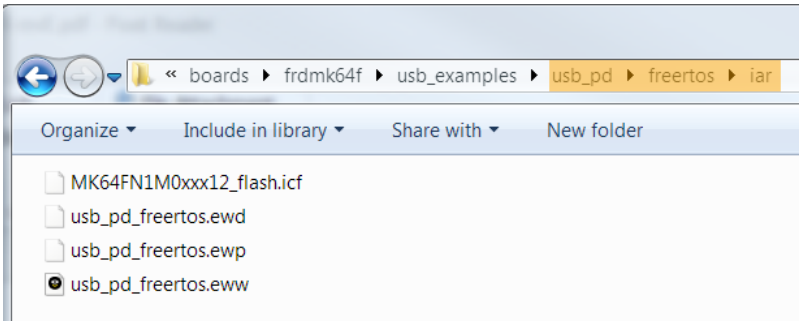

Figure 2. Replacing files

9. Use a text editor to open *usb_pd_freertos.eww*.

10. Inside the text editor, find all "$[project_name]" strings and replace them with "usb_pd_freertos".
11. Copy the board-support SDK template files *board.c*, *board.h*, *clock_config.c*, *clock_config.h*, *pin_mux.c*, and *pin_mux.h*, which can be found in the *...\boards\frdmk64f\project_template* folder.
12. In the second Explorer window, navigate to the *...\boards\frdmk64f\usb_examples\usb_pd\freertos* folder and paste the files copied in the previous step.
13. Copy the FreeRTOS configuration template file *FreeRTOSConfig.h*, which can be found in the *...\rtos\freertos_9.0.0\template_application\ARM_CM4F* folder.
14. Navigate to the *...\boards\frdmk64f\usb_examples\usb_pd\freertos* folder and paste the *FreeRTOSConfig.h* file copied in the previous step.
15. In the first Explorer window, navigate to the *...\boards\frdmk64f\cmsis_driver_examples\i2c\interrupt_transfer* folder and copy the *RTE_Device.h* file.
16. Paste *RTE_Device.h* into the second Explorer window (*...\boards\frdmk64f\usb_examples\usb_pd\freertos*).
17. Obtain the USB Power Delivery source code from the software downloaded in step 1 (MCUXpresso SDK for FRDM-KL27Z with the USB Power Delivery).
    a. Unzip the FRDM-KL27Z SDK folder.
    b. Use the Explorer to copy the *main.c*, *pd_app.c*, *pd_app.h*, *pd_app_demo.c*, *pd_command_app.c*, *pd_command_interface.c*, *pd_command_interface.h*, *pd_power_app.c*, *pd_power_interface.c*, *pd_power_interface.h*, *usb_io.h*, *usb_kinetis_io_drv.h*, *usb_pd_config.h*, and *usb_pit_drv.h* files located in the *...\boards\frdmkl27z\usb_examples\usb_pd\freertos* folder.

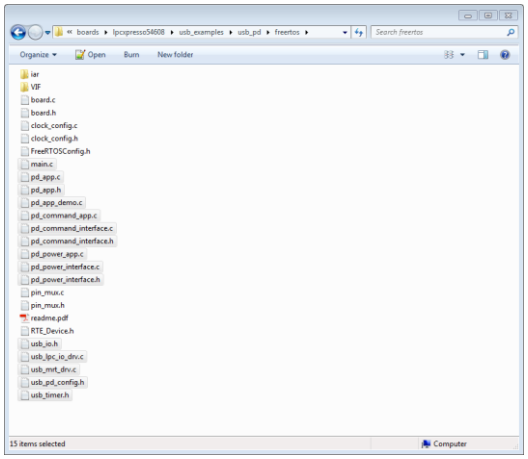

Figure 3. Copying files

    c. Paste the files into the *...\boards\frdmk64f\usb_examples\usb_pd\freertos* folder.
    d. To add the Power Delivery middleware support, copy the FRDM-KL27Z *pd* folder located in the *...\middleware\usb_1.7.0* folder.

e. Insert the copied *pd* folder into the FRDM-K64F middleware software located in the *...\middleware\usb_1.6.3* folder.

f. Replace the FRDM-K64F *usb_misc.h* file located in the *…\middleware\usb_1.6.3\include* folder with the version found in the FRDM-KL27Z SDK package.

18. Follow the steps in Appendix C to add the Arduino signals to the *pin_mux.c* and *pin_mux.h* files using the MCUXpresso Config Tools.
19. Using the IAR IDE, open the FRDM-KL27Z *usb_pd_freertos* workspace located in the *...\boards\frdmkl27z\usb_examples\usb_pd\freertos\iar* folder and the recently created FRDM-K64F workspace in the *...\boards\frdmk64f\usb_examples\usb_pd\freertos\iar* folder.
20. In the FRDM-K64F workspace, look for the *doc* folder in the "Workspace" window and remove it from the project by right-clicking it and selecting the "Remove" option.

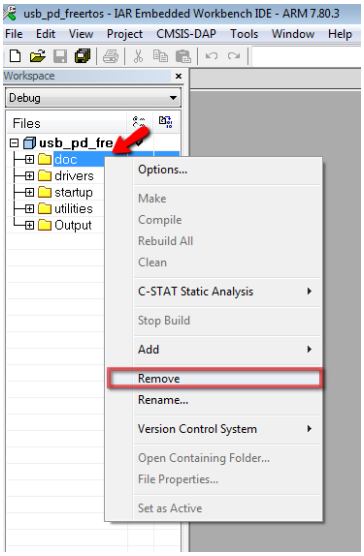

Figure 4. Removing the *doc* folder

21. Copy the folder structure from the FRDM-KL27Z workspace. To add a new folder, right-click the project name, expand the "Add" menu, and select the "Add Group…" option. In the "Add Group" wizard, write the folder name into the text box and click the "OK" button.

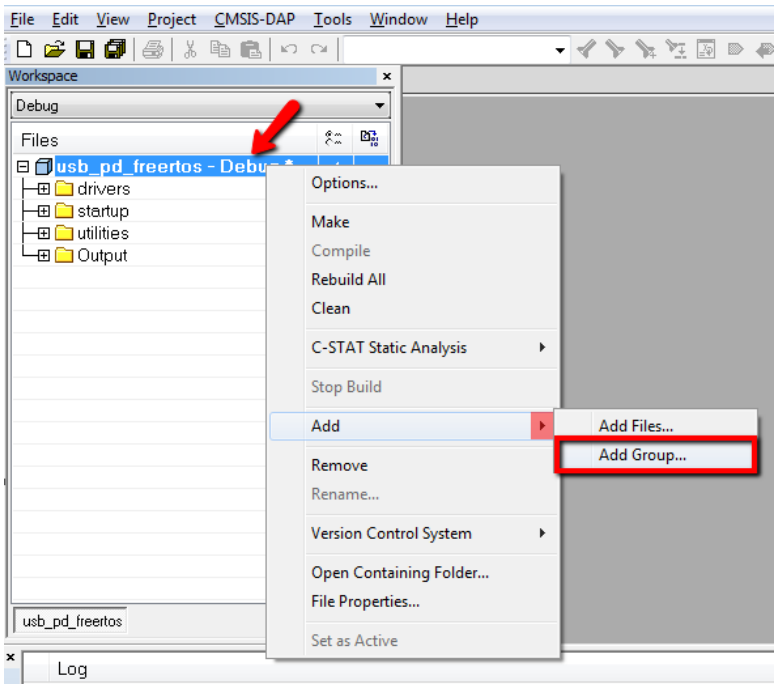

Figure 5. Copying folder structure

22. After adding all the groups, the folder structure should look like this:

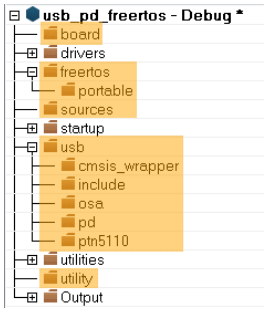

Figure 6. Resulting folder structure

23. Add the preprocessor paths to the workspace. To use the FRDM-KL27Z project options as a reference, navigate to "Project Options..." → "C/C++ Compiler" → "Preprocessor" and copy all the text in the "Additional include directories" field.

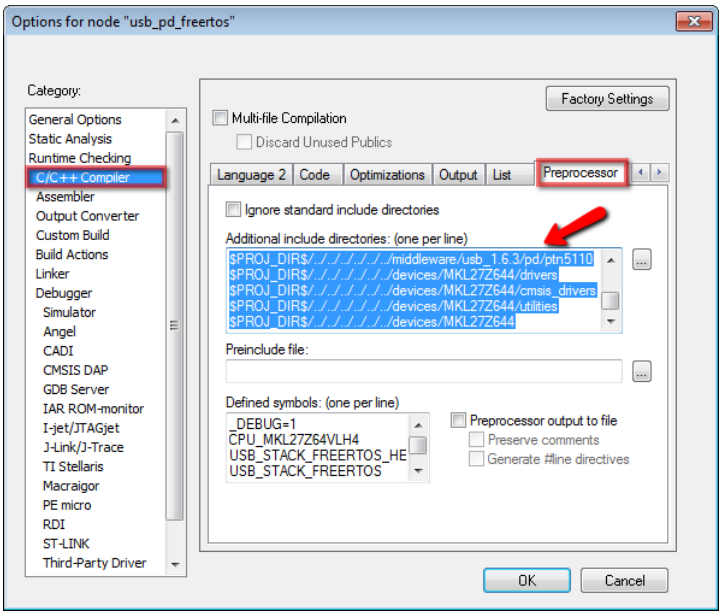

Figure 7. Adding preprocessor paths

24. Paste the paths into a text editor and replace all “MKL27Z644” occurrences with “MK64F12”. Replace the “ARM_CM0” occurrences with “ARM_CM4F” and copy the resulting text.

```
1  $PROJ_DIR$/../../../../../../../rtos/freertos_9.0.0/Source/portable/IAR/ARM_CM4F
2  $PROJ_DIR$/../../../../../../../rtos/freertos_9.0.0/Source/include
3  $PROJ_DIR$/../../../../../../../CMSIS/Include
4  $PROJ_DIR$/..
5  $PROJ_DIR$/../../../../../../../CMSIS/Driver/Include
6  $PROJ_DIR$/../../../../../../../rtos/freertos_9.0.0/Source
7  $PROJ_DIR$/../../../../../../../middleware/usb_1.6.3/pd
8  $PROJ_DIR$/../../../../../../../middleware/usb_1.6.3/osa
9  $PROJ_DIR$/../../../../../../../middleware/usb_1.6.3/include
10 $PROJ_DIR$/../../../../../../../middleware/usb_1.6.3/pd/cmsis_wrapper
11 $PROJ_DIR$/../../../../..
12 $PROJ_DIR$/../../../../../../../devices/MK64F12/drivers
13 $PROJ_DIR$/../../../../../../../devices/MK64F12/cmsis_drivers
14 $PROJ_DIR$/../../../../../../../middleware/usb_1.6.3/pd/ptn5110
15 $PROJ_DIR$/../../../../../../../devices/MK64F12/utilities
16 $PROJ_DIR$/../../../../../../../devices/MK64F12
```

Figure 8. Replacing text

25. Replace the FRDM-K64F workspace preprocessor paths with the previously-copied values. Before clicking the “OK” button, change the values inside the “Define symbols” text box: add “CPU_MK64FN1M0VLL12”, “USB_STACK_FREERTOS_HEAP_SIZE=32768”, “USB_STACK_FREERTOS”, and “FSL_RTOS_FREE_RTOS”.

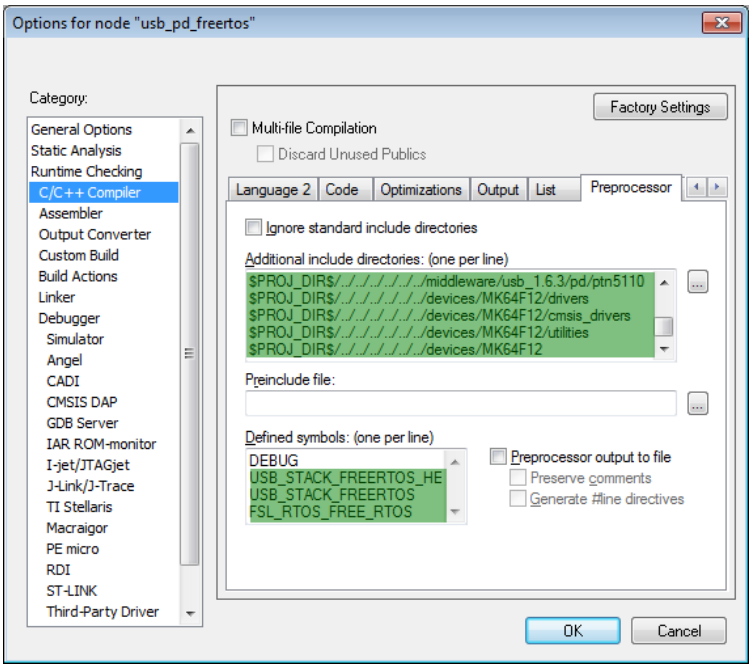

Figure 9. Changing values in text boxes

26. Navigate to "Project Options..." → "Assembler" → "Preprocessor", add the following paths into the "Additional include directories" section, and click the "OK" button:

```
$PROJ_DIR$/../../../..
$PROJ_DIR$/..
```

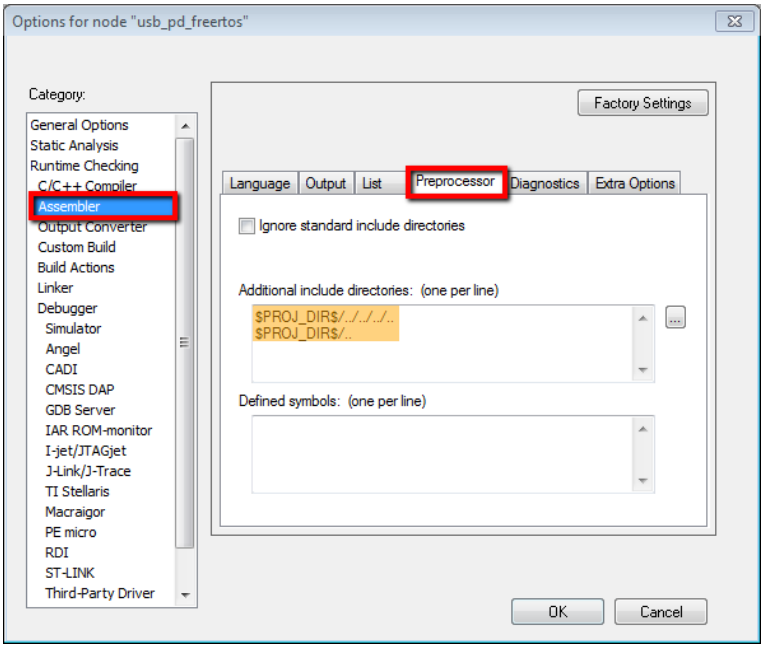

Figure 10. Adding paths

27. After including the directories, compiler paths, and defined symbols, add the source code to the workspace. To do that, right-click a group name (for example; board), expand the "Add" menu, and click the "Add Files..." option. In the "Add Files" wizard, use Table 3 as a guideline for which groups, paths, and files to add.

Table 3. FRDM-K64F source files

Group	Path	Files
board	*…\boards\frdmk64f\usb_examples\usb_pd\freertos*	*board.c, board.h, clock_config.c, clock_config.h, pin_mux.c, pin_mux.h*
drivers	*…\devices\MK64F12\drivers*	*fsl_clock.c, fsl_clock.h, fsl_dmamux.c, fsl_dmamux.h, fsl_edma.c, fsl_edma.h, fsl_gpio.c, fsl_gpio.h, fsl_i2c.c, fsl_i2c.h, fsl_i2c_edma.c, fsl_i2c_edma.h, fsl_lptmr.c, fsl_lptmr.h, fsl_pit.c, fsl_pit,h, fsl_port.h, fsl_uart.c, fsl_uart.h, fsl_sim.c, fsl_sim.h*
drivers	*…\devices\ MK64F12\drivers\cmsis_drivers*	*fsl_i2c_cmsis.c, fsl_i2c_cmsis.h*
freertos\portable	*…\rtos\freertos_9.0.0\Source\portable\IAR\ARM_CM4F*	*fsl_tickless_generic.c, fsl_tickless_systick.c, port.c, portasm.s, portmacro.h*
freertos	*…\rtos\freertos_9.0.0\Source*	*croutine.c, event_groups.c, list.c, queue.c, task.c, timers.c*
freertos	*…\rtos\freertos_9.0.0\Source\portable\MemMang*	*heap_4.c*
freertos	*…\rtos\freertos_9.0.0\Source\include*	*croutine.h, deprecated_definitions.h, event_groups.h, FreeRTOS.h, freertos_tasks_c_additions.h, list.h, mpu_prototypes.h, mpu_wrappers.h, portable.h, projdefs.h, queue.h, semphr.h, StackMacros.h, task.h, timers.h*
sources	*…\boards\frdmk64f\usb_examples\usb_pd\freertos*	*FreeRTOSConfig.h, main.c, pd_app.c, pd_app.h, pd_app_demo.c, pd_command_app.c, pd_command_interface.c, pd_command_interface.h, pd_power_app.c, pd_power_interface.c, pd_power_interface.h, RTE_Device.h, usb_pd_config.h, usb_timer.h*
usb\cmsis_wrapper	*…\middleware\usb_1.6.3\pd\cmsis_wrapper*	*usb_cmsis_iic_wrapper.c, usb_cmsis_wrapper.c, usb_cmsis_wrapper.h*
usb\include	*…\middleware\usb_1.6.3\include*	*usb.h, usb_misc.h*
usb\osa	*…\middleware\usb_1.6.3\osa*	*usb_osa.h, usb_osa_freertos.c, usb_osa_freertos.h*
usb\pd	*…\middleware\usb_1.6.3\pd*	*usb_pd.h, usb_pd_connect.c, usb_pd_interface.c, usb_pd_interface.h, usb_pd_msg.c, usb_pd_phy.h, usb_pd_policy.c, usb_pd_spec.h, usb_pd_timer.c, usb_pd_timer.h*
usb\ptn5110	*…\middleware\usb_1.6.3\pd\ptn5110*	*usb_pd_ptn5110.h, usb_pd_ptn5110_connect.c, usb_pd_ptn5110_hal.c, usb_pd_ptn5110_interface.c, usb_pd_ptn5110_msg.c, usb_pd_ptn5110_register.h*
Utility	*…\boards\frdmk64f\usb_examples\usb_pd\freertos*	*usb_io.h, usb_kinetis_io_drv.c, usb_pit_drv.c, usb_timers.h*

4.1.2. MCUXpresso IDE

1. Follow the instructions in Appendix A to build the SDK package for FRDM-KL27Z using the MCUXpresso IDE.
2. Follow the instructions in Appendix B to build the SDK for a selected platform (for example; FRDM-K64F) using the MCUXpresso IDE.
3. In Explorer, navigate to *...\boards\frdmk64f\usb_examples.* Add a new folder with the name of the USB Power Delivery example project to port (for example; *usb_pd*). Inside this folder, add a new folder named *freertos*.
4. Copy the SDK board template files *board.c*, *board.h*, *clock_config.c*, and *clock_config.h*, which can be found in the *...\boards\frdmk64f\project_template* folder.
5. In the second Explorer window, navigate to the *...\boards\frdmk64f\usb_examples\usb_pd\freertos* folder and paste the files copied in the previous step.
6. Copy the FreeRTOS configuration template file (*FreeRTOSConfig.h*) which can be found in the *...\rtos\freertos_9.0.0\template_application\ARM_CM4F* folder.
7. Navigate to the *...\boards\frdmk64f\usb_examples\usb_pd\freertos* folder and paste the *FreeRTOSConfig.h* file copied in the previous step.
8. In the first Explorer window, navigate to the *...\boards\frdmk64f\cmsis_driver_examples\i2c\interrupt_transfer* folder and copy the *RTE_Device.h* file.
9. Paste the *RTE_Device.h* file into the *...\boards\frdmk64f\usb_examples\usb_pd\freertos* folder.
10. Obtain the USB Power Delivery source code of the downloaded software (Appendix A).
    a. Unzip the FRDM-KL27Z SDK package folder.
    b. Use the Explorer to copy the *example.xml*, *main.c*, *pd_app.c*, *pd_app.h*, *pd_app_demo.c*, *pd_command_app.c*, *pd_command_interface.c*, *pd_command_interface.h*, *pd_power_app.c*, *pd_power_interface.c*, *pd_power_interface.h*, *usb_io.h*, *usb_kinetis_io_drv.c*, *usb_pd_config.h*, *usb_pd_freertos.xml*, *usb_pit_drv.c*, and *usb_timer.h* files located in the *...\boards\frdmkl27z\usb_examples\usb_pd\freertos* folder.

Figure 11. Copying files

c. Paste the files into the *...\boards\frdmk64f\usb_examples\usb_pd\freertos* folder.

d. To add the Power Delivery middleware support, copy the FRDM-KL27Z *pd* folder located in the *...\middleware\usb_1.7.0* folder.

e. Insert the copied *pd* folder into the FRDM-K64F middleware software located in the *...\middleware\usb_1.6.3* folder.

f. Replace the FRDM-K64F *usb_misc.h* file located in the *...\middleware\usb_1.6.3\include* folder with the version found in the FRDM-KL27Z SDK.

11. Follow Appendix C to add the Arduino signals to the *pin_mux.c* and *pin_mux.h* files using the MCUXpresso Config Tools.
12. Using a text editor, open the *example.xml* file located in the *...\boards\frdmk64f\usb_examples\usb_pd\freertos* folder. Replace all FRDM-**KL27Z**, frdm**kl27z**, fsl_**dma.** , fsl_i2c_**dma**, M**KL27Z644**, **1.7.0**, and ARM_CM**0** occurrences with FRDM-**K64F**, frdm**k64f**, fsl_**edma.** , fsl_i2c_**edma**, M**K64F12**, **1.6.3**, and ARM_CM**4F**.
13. Using a text editor, open the *usb_pd.xml* file located in the *...\boards\frdmk64f\usb_examples\usb_pd\freertos* folder. Replace all frdm**kl27z**, drivers.**dma**, M**KL27Z644**, and fpu.**none** occurrences with frdm**k64f**, drivers.**edma**, M**K64F12**, and fpu.**fpv4.hard**.
14. Repeat steps 3-13 for the *usb_pd_battery* and *usb_pd_source_charger* example projects.
15. Use a text editor (such as Notepad++) to open the *FRDM-KL2Z_manifest.xml* file located in the *...\SDK_2.2.1_FRDM-KL27Z* folder and the *FRDM-K64F_manifest.xml* file located in the*...\SDK_2.2_FRDM-K64F* folder.
16. Copy the lines *570-584* from the *FRDM-KL2Z_manifest.xml* file.

Figure 12. Copying lines

17. In the *FRDM-K64F_manifest.xml* file, paste the previously copied text after line 955. After that, replace all frdm**kl27z** occurrences with frdm**k64f**.

Figure 13. Copying lines

18. Copy lines 3086-3136 from the *FRDM-KL2Z_manifest.xml* file.
19. In the *FRDM-K64F_manifest.xml* file, paste the previously copied text after line 3866. After that, replace all M**KL27Z644**, M**KL27Z64xxx4**, and **1.7.0** occurrences with M**K64F12**, M**K64FN1M0xxx12**, and **1.6.3**.

4.1.3. Migrating source files in both IDEs

At this stage, only the *pin_mux.c* and *pin_mux.h* files align with the FRDM-K64F development board. Follow these steps to migrate the software to a new platform:

- *utility\usb_io.h*—add the following lines of code:

```
typedef enum _k64_ports
{
        kPTA = 0,
        kPTB,
        kPTC,
        kPTD,
        kPTE
}k64_ports;
```

- *sources\FreeRTOSConfig.h*—replace these definitions with the values highlighted in **orange**:

```
#define configMAX_PRIORITIES            8
#define configUSE_TIME_SLICING          1
#define configTOTAL_HEAP_SIZE     ((size_t)(5*1024))
#define configTIMER_TASK_PRIORITY (configMAX_PRIORITIES - 1)
```

- *sources\main.c*—inside HW_TimerInit, replace PIT_IRQn with PIT0_IRQn:

```
NVIC_SetPriority(PIT0_IRQn, PD_TIMER_INTERRUPT_PRIORITY);
```

- Use Table 1 to find the GPIO assigned to the PTN5150 I²C signals A4 and A5. For example, the FRDM-K64F rev E IOs are PTC11-I2C1_SDA and PTC10-I2C1_SCL.

```
#define I2C1_SCL (10U)
#define I2C1_SDA (11U)
```

- Look for the BOARD_I2C1_ReleaseBus function and change the number highlighted in green to the I²C instance connected to D14 – D15. For example, this function can be named BOARD_I2C0_ReleaseBus for the FRDM-K64F.

```
void BOARD_I2C0_ReleaseBus(void)
```

- Look for the BOARD_I2C0_ReleaseBus function and change the number highlighted in green to the I²C instance connected to A4 – A5. For example; this function can be named BOARD_I2C1_ReleaseBus for the FRDM-K64F.

```
void BOARD_I2C1_ReleaseBus(void)
```

- Modify the BOARD_I2C1_ReleaseBus function IO numbers so that they are compatible with the GPIOs used for the FRDM-K64F A4 – A5 signals.

```
void BOARD_I2C1_ReleaseBus(void)
{
    uint8_t i = 0;
    gpio_pin_config_t pin_config;
    port_pin_config_t i2c_pin_config = {0};

    /* Config pin mux as gpio */
    i2c_pin_config.pullSelect = kPORT_PullUp;
    i2c_pin_config.mux = kPORT_MuxAsGpio;

    pin_config.pinDirection = kGPIO_DigitalOutput;
    pin_config.outputLogic = 1U;
    CLOCK_EnableClock(kCLOCK_PortC);
    PORT_SetPinConfig(PORTC, I2C1_SCL, &i2c_pin_config);
    PORT_SetPinConfig(PORTC, I2C1_SDA, &i2c_pin_config);

    GPIO_PinInit(GPIOC, I2C1_SCL, &pin_config);
    GPIO_PinInit(GPIOC, I2C1_SDA, &pin_config);

    /* Drive SDA low first to simulate a start */
    GPIO_WritePinOutput(GPIOC, I2C1_SDA, 0U);
    i2c_release_bus_delay();

    /* Send 9 pulses on SCL and keep SDA high */
    for (i = 0; i < 9; i++)
    {
        GPIO_WritePinOutput(GPIOC, I2C1_SCL, 0U);
        i2c_release_bus_delay();

        GPIO_WritePinOutput(GPIOC, I2C1_SDA, 1U);
        i2c_release_bus_delay();

        GPIO_WritePinOutput(GPIOC, I2C1_SCL, 1U);
        i2c_release_bus_delay();
        i2c_release_bus_delay();
```

```
    }

    /* Send stop */
    GPIO_WritePinOutput(GPIOC, I2C1_SCL, 0U);
    i2c_release_bus_delay();

    GPIO_WritePinOutput(GPIOC, I2C1_SDA, 0U);
    i2c_release_bus_delay();

    GPIO_WritePinOutput(GPIOC, I2C1_SCL, 1U);
    i2c_release_bus_delay();

    GPIO_WritePinOutput(GPIOC, I2C1_SDA, 1U);
i2c_release_bus_delay();
}
```

- Modify the BOARD_I2C0_ReleaseBus function IO numbers so that they are compatible with the GPIOs used for the FRDM-K64F D14 – D15 signals.

```
void BOARD_I2C0_ReleaseBus(void)
{
    uint8_t i = 0;
    gpio_pin_config_t pin_config;
    port_pin_config_t i2c_pin_config = {0};

    /* Config pin mux as gpio */
    i2c_pin_config.pullSelect = kPORT_PullUp;
    i2c_pin_config.mux = kPORT_MuxAsGpio;

    pin_config.pinDirection = kGPIO_DigitalOutput;
    pin_config.outputLogic = 1U;
    CLOCK_EnableClock(kCLOCK_PortE);
    PORT_SetPinConfig(PORTE, 24u, &i2c_pin_config);
    PORT_SetPinConfig(PORTE, 25u, &i2c_pin_config);

    GPIO_PinInit(GPIOE, 24u, &pin_config);
    GPIO_PinInit(GPIOE, 25u, &pin_config);

    /* Drive SDA low first to simulate a start */
    GPIO_WritePinOutput(GPIOE, 25u, 0U);
    i2c_release_bus_delay();

    /* Send 9 pulses on SCL and keep SDA high */
    for (i = 0; i < 9; i++)
    {
        GPIO_WritePinOutput(GPIOE, 24u, 0U);
        i2c_release_bus_delay();

        GPIO_WritePinOutput(GPIOE, 25u, 1U);
        i2c_release_bus_delay();

        GPIO_WritePinOutput(GPIOE, 24u, 1U);
        i2c_release_bus_delay();
        i2c_release_bus_delay();
    }

    /* Send stop */
    GPIO_WritePinOutput(GPIOE, 24u, 0U);
    i2c_release_bus_delay();
```

```
    GPIO_WritePinOutput(GPIOE, 25u, 0U);
    i2c_release_bus_delay();

    GPIO_WritePinOutput(GPIOE, 24u, 1U);
    i2c_release_bus_delay();

    GPIO_WritePinOutput(GPIOE, 25u, 1U);
    i2c_release_bus_delay();
}
```

- Go to the main function and rename BOARD_I2C**0**_ReleaseBus with the I^2C**n** instance used for A4 – A5 (BOARD_I2C**1**_ReleaseBus).
- Use Table 1 to find the GPIO assigned to the EXTRA_EN_SRC output signal (D4). On the FRDM-K64F rev.E, the **PTB23** output pin is used. Update the HW_GpioExternalSourceEnable IO software to be compatible with **PTB23**, as shown below.

```
void HW_GpioExternalSourceEnable(uint8_t enable)
{
 if (enable)
 {
    USB_GpioOutputWritePin(kPTB, kPTB, 23, 1);
 }
 else
 {
    USB_GpioOutputWritePin(kPTB, kPTB, 23, 0);
 }
}
```

- Modify the port and pin number values inside the HW_GpioReadPowerRequestSW and HW_GpioReadPRSwapSW functions using the information in Table 2.

```
uint8_t HW_GpioReadPowerRequestSW(void)
{
  return USB_GpioInputReadPin(kPTC, kPTC, 6);
}

uint8_t HW_GpioReadPRSwapSW(void)
{
  return USB_GpioInputReadPin(kPTA, kPTA, 4);
}
```

- Look for the HW_GpioInit function and modify the GPIO initialization with the FRDM-K64F IOs. Use Table 1 and Table 2 as a reference (external power control = EXTRA_EN_SRC, SW1 = SW2, SW3= SW3, PD PHY interrupt = nALERT).

```
/* the external power control pin */
USB_GpioOutputInit(kPTB, kPTB, 23);

/* SW1 */
USB_GpioInputInit(kPTC, kPTC, 6);
/* SW3 */
USB_GpioInputInit(kPTA, kPTA, 4);

/* the PD PHY interrupt gpio */
USB_GpioInterruptInit(kPTC, kPTC, 12u, kUSB_GpioInterruptLogicZero,
HW_GpioPDPHYIntCallback);
```

- Look for the PORTB_PORTC_PORTD_PORTE_IRQHandler function and change the IRQHandler portion to match the port used on the nALERT pin.

```
void PORTC_IRQHandler(void)
{
```

```
    if (PORTC->ISFR & (1U << 12u))
    {
        PORTC->ISFR = 1U << 12u;
        if (!(GPIOC->PDIR & (1U << 12u)))
        {
            HW_GpioPDPHYIntCallback();
        }
    }
}
```

- *board\board.h*—using a text editor, open the *...\boards\frdmk64f\demo_apps\hello_world\board.h* file and copy all the BOARD_SWn definitions. Navigate to the IAR IDE workspace, double-click the *board\board.h* file, and paste the content after line 55.

```
/* Define the port interrupt number for the board switches */
#define BOARD_SW2_GPIO GPIOC
#define BOARD_SW2_PORT PORTC
#define BOARD_SW2_GPIO_PIN 6U
#define BOARD_SW2_IRQ PORTC_IRQn
#define BOARD_SW2_IRQ_HANDLER PORTC_IRQHandler
#define BOARD_SW2_NAME "SW2"

#define BOARD_SW3_GPIO GPIOA
#define BOARD_SW3_PORT PORTA
#define BOARD_SW3_GPIO_PIN 4U
#define BOARD_SW3_IRQ PORTA_IRQn
#define BOARD_SW3_IRQ_HANDLER PORTA_IRQHandler
#define BOARD_SW3_NAME "SW3"
```

  Using Table 1, add these three macros: BOARD_ARDUINO_INT_IRQ, BOARD_ARDUINO_I2C_IRQ, and BOARD_ARDUINO_I2C_INDEX.

```
#define BOARD_ARDUINO_INT_IRQ    PORTC_IRQn
#define BOARD_ARDUINO_I2C_IRQ    I2C0_IRQn
#define BOARD_ARDUINO_I2C_INDEX  0
```

- *sources\pd_app.h*—replace all sw1State and sw1Time occurrences with sw2State and sw2Time. Use CTRL + H to speed up the replacing process.
- *sources\pd_app_demo.c*—replace all sw1State and sw1Time occurrences with sw2State and sw2Time. Use CTRL + H to speed up the replacing process.

To import and build the MCUXpresso IDE projects, follow these steps:

1. Import the K64 SDK into the MCUXpresso IDE by dragging and dropping the unzipped *SDK_2.2_FRDM-K64F* folder into the “Installed SDK” view.
2. Import the *usb_pd* example application by clicking “Import SDK example(s)…” located in the “Quick Start Panel” in the lower left-hand corner.
3. Click the “frdmk64f” board image to select the project that can run on that board and click the “Next” button.

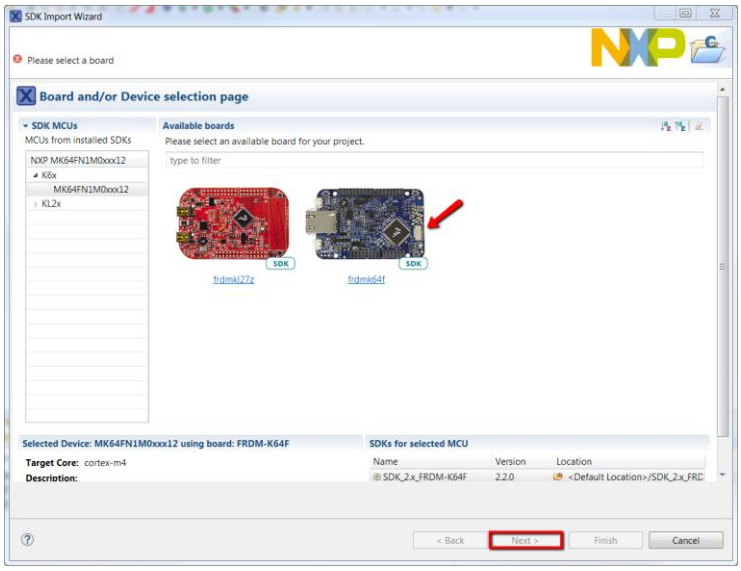

Figure 14. Selecting a project

4. Use the arrow button to expand the “usb_examples” category, and then click the checkbox next to “usb_pd” to select that project. Deselect the “Enable semihost” option in the project options and click the “Next” button.

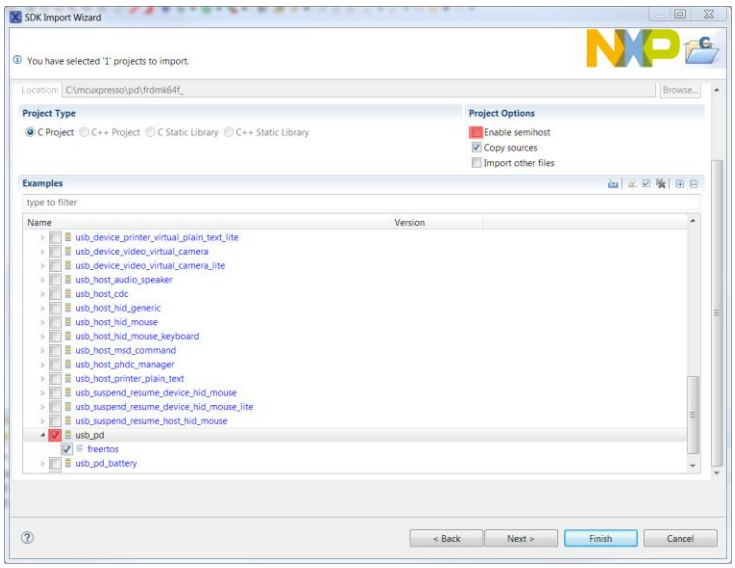

Figure 15. Importing a project

5. In the “Advanced Settings” area, untick the “Redirect SDK “PRINTF” to C library “printf” checkbox to use the MCUXpresso SDK console functions for printing instead of the generic C library functions. Click the “Finish” button.

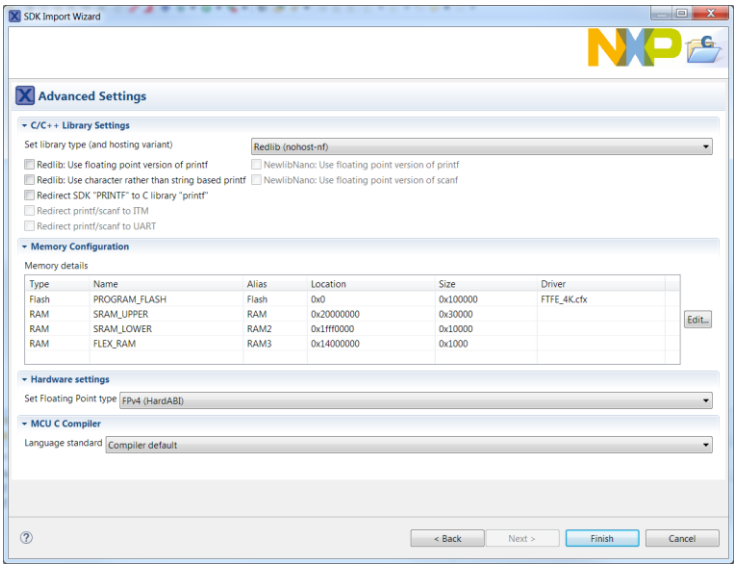

Figure 16. Importing a project

Compile and flash the example.

For more information, see the *MCUXPresso SDK USB Type-C PD Stack User's Guide* located in the *...SDK_2.2.1_FRDM-KL27Z\docs\usb* folder.

Appendix A. SDK Builder FRDM-KL27Z

Use the online MCUXpresso Config Tools to create a custom SDK package for the FRDM-K64F board.

1. Open a web browser and navigate to the MCUXpresso homepage mcuxpresso.nxp.com.

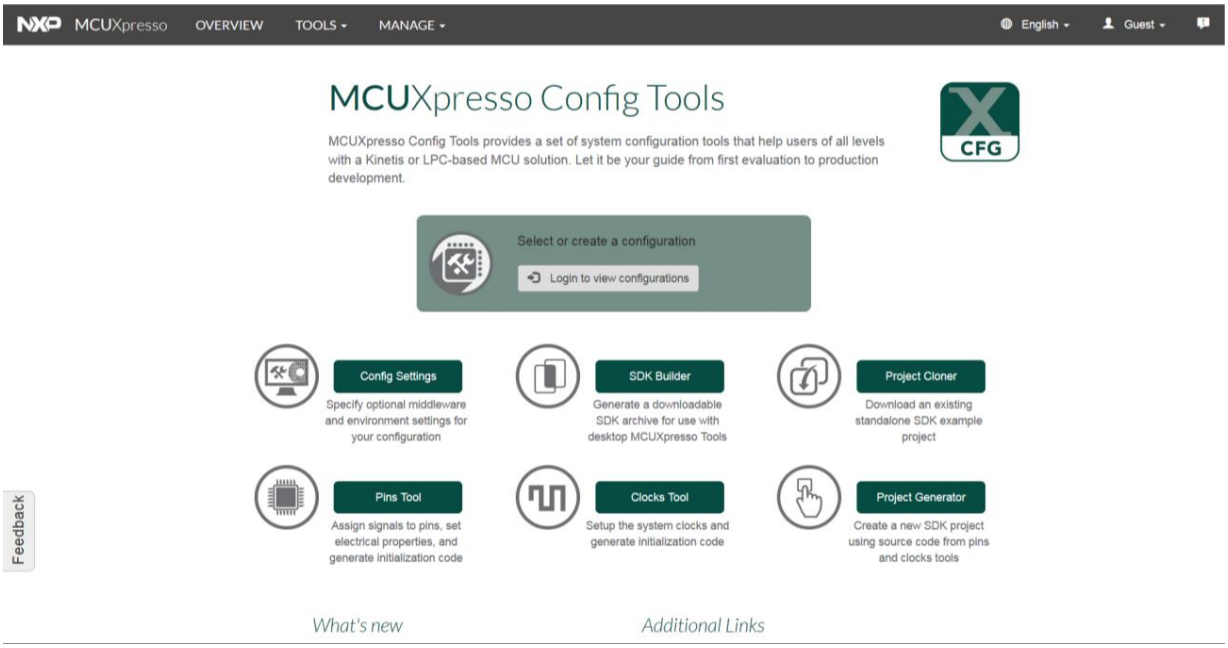

Figure 17. MCUXpresso Config Tools

2. Click the “Login to view configurations” button to create a new configuration.

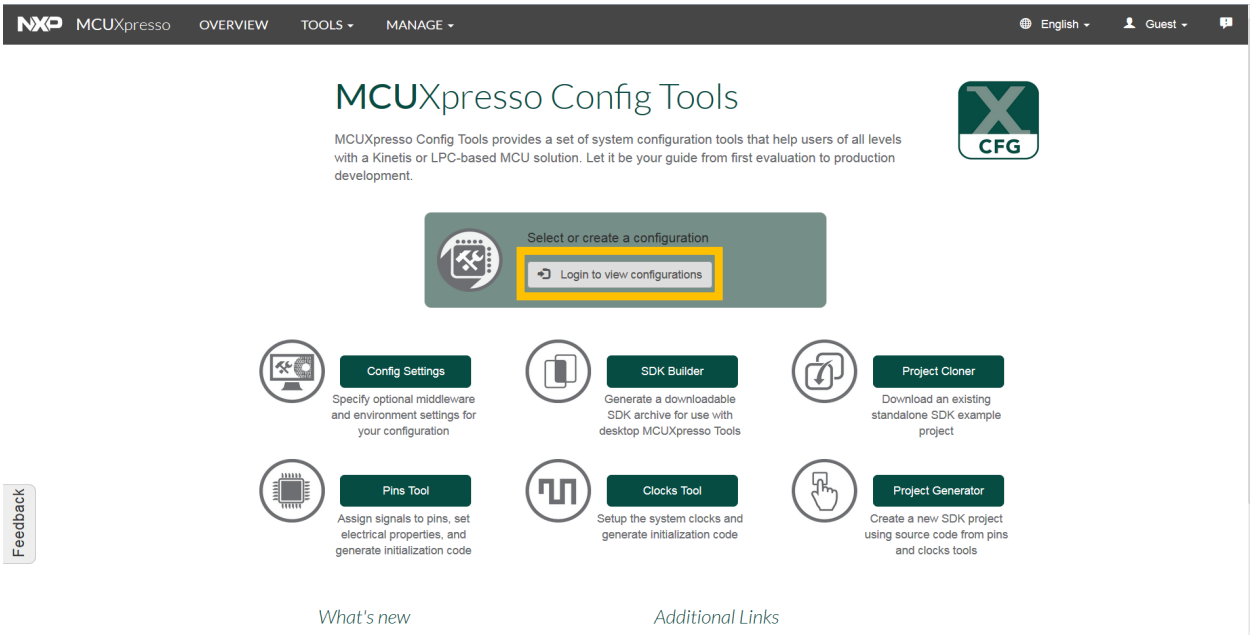

Figure 18. MCUXpresso Config Tools

3. You are redirected to the www.nxp.com login page. Enter your account information or create a

new account.

4. Back on the MCUXpresso homepage, click the drop-down box and select the “New Configuration” option.

Figure 19. Creating a configuration

5. Search for the board name (for example; FRDM-KL27).

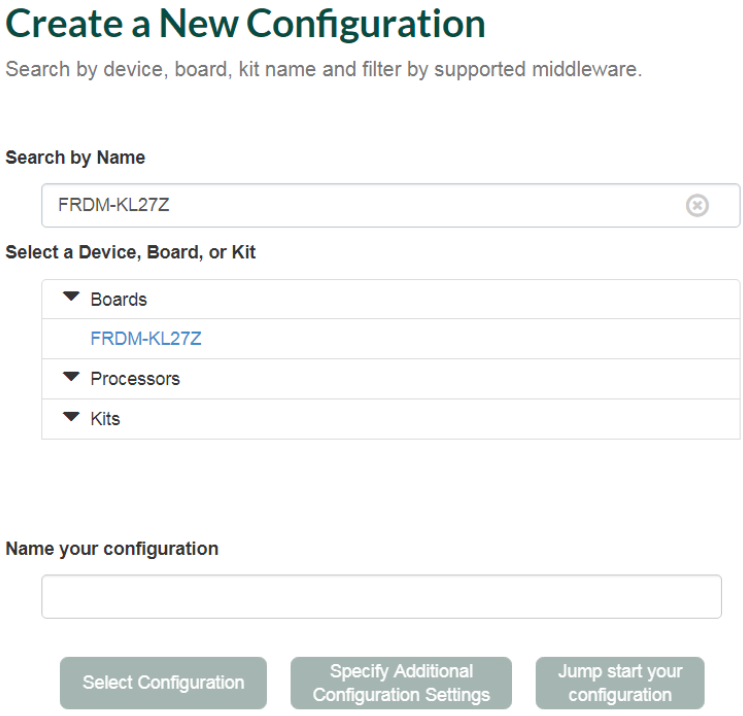

Figure 20. Creating a configuration

6. Select a board from the list and provide a name for the configuration. Click the “Specific Additional Configuration Settings” button to select from FreeRTOS, IAR toolchain, and USB middleware.

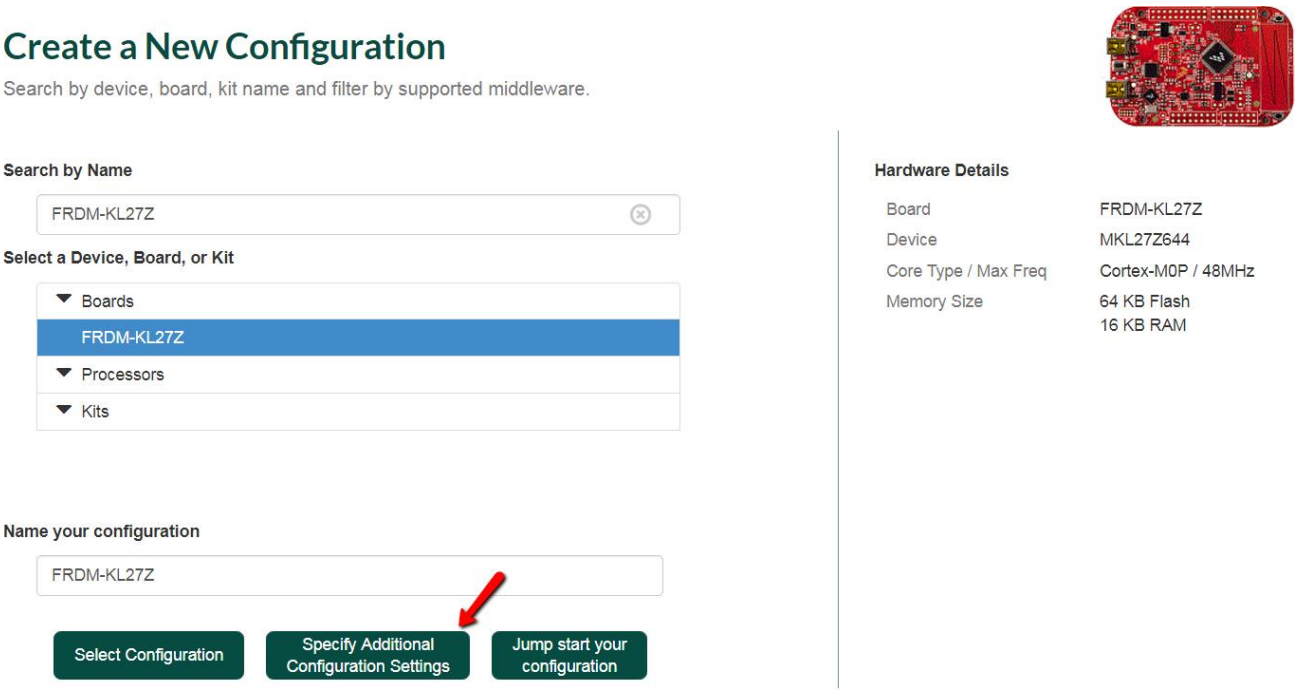

Figure 21. Creating a configuration

7. In the "Configuration Settings" section, set the following:

- Host OS → Windows.
- Toolchain/IDE → IAR Embedded Workbench for Arm.
- Middleware → USB Stack, FreeRTOS.

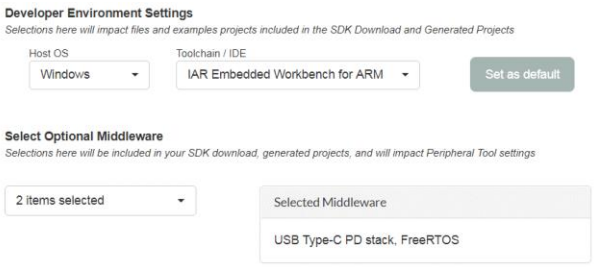

Figure 22. IAR IDE

Or:

- Host OS → Windows.
- Toolchain/IDE → MCUXpresso IDE.
- Middleware → USB Stack, FreeRTOS.

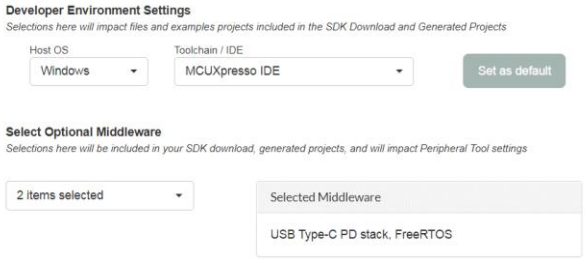

Figure 23. MCUXpresso IDE

8. After selecting the above settings, click the “Go to SDK Builder” button.

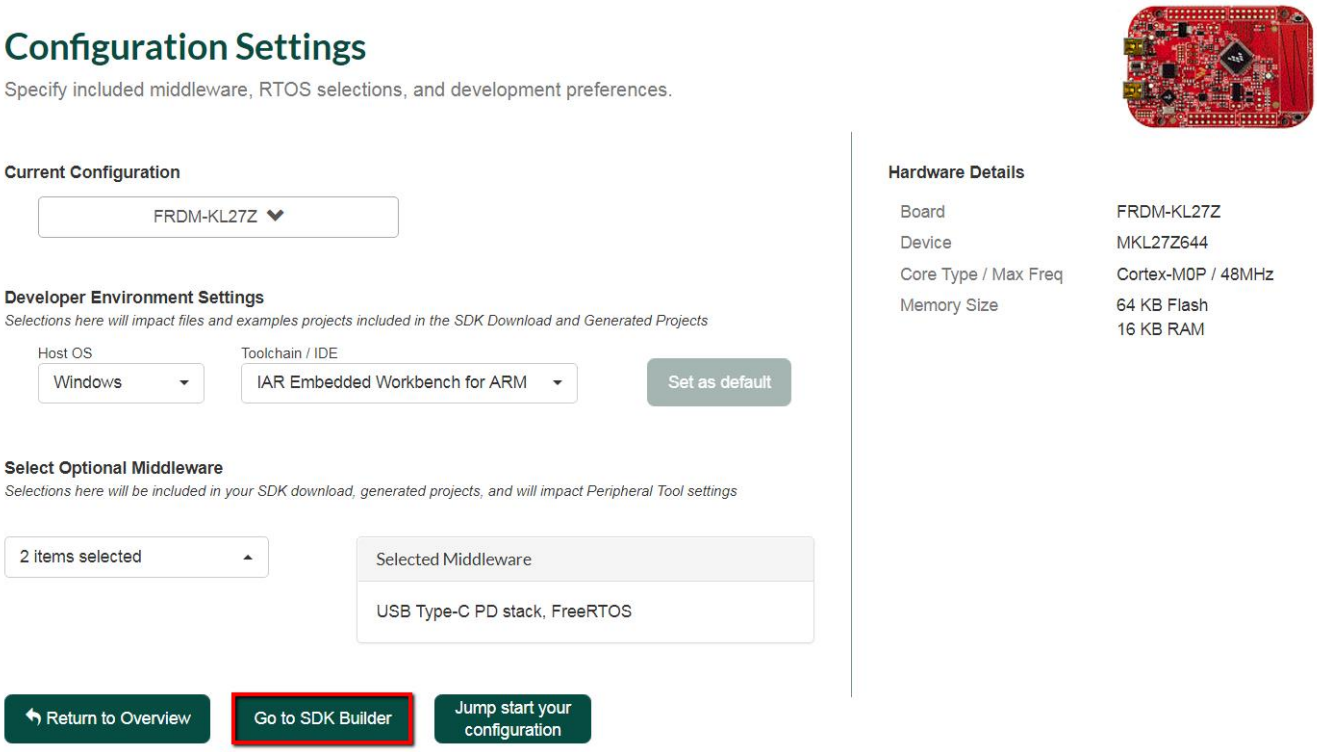

Figure 24. Navigating to the SDK Builder

9. Click the “Download Now” button to download the SDK package.

NOTE

You may see the “Request to Build” button instead of the “Download Now” button. Click the “Request to Build” button and when the package is built, a notification to download it appears in the “SDK Archive” section. To access the “SDK Archive” section, click the “MANAGE” tab and select the “SDK Archive” option.

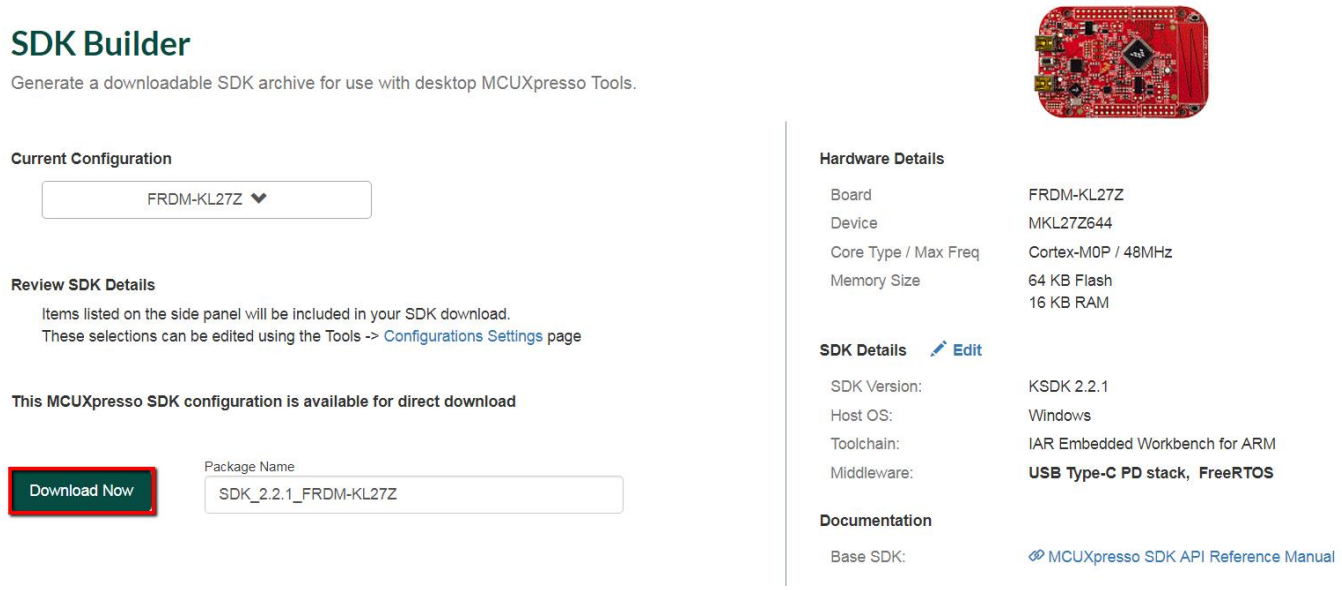

Figure 25. SDK Builder

10. Agree to the software terms and conditions.

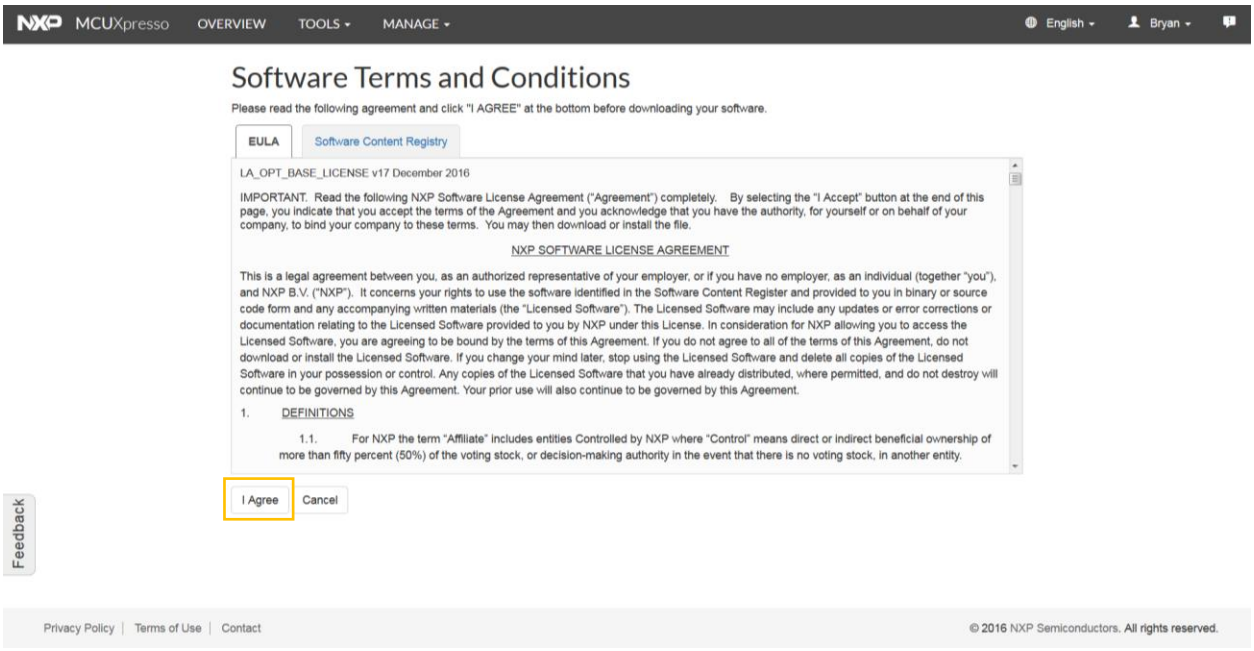

Figure 26. Software terms and conditions

11. Unzip the SDK to a folder (for example; *SDK_2.2_FRDM-KL27Z*).

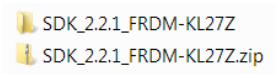

Figure 27. SDK folder

Appendix B. SDK Builder FRDM-K64F

Use the online MCUXpresso Config Tools to create a custom SDK package for the FRDM-K64F board.

1. Open a web browser and navigate to the MCUXpresso homepage mcuxpresso.nxp.com.

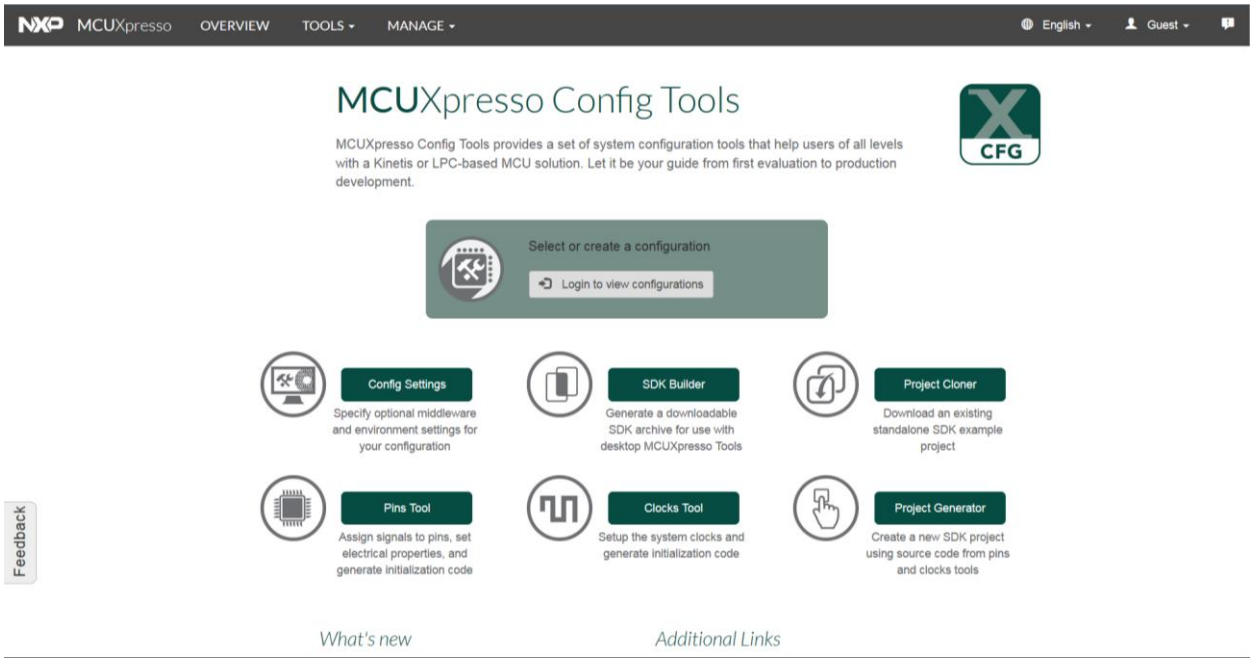

Figure 28. MCUXpresso homepage

2. Click the “Login to view configurations” button to create a new configuration.

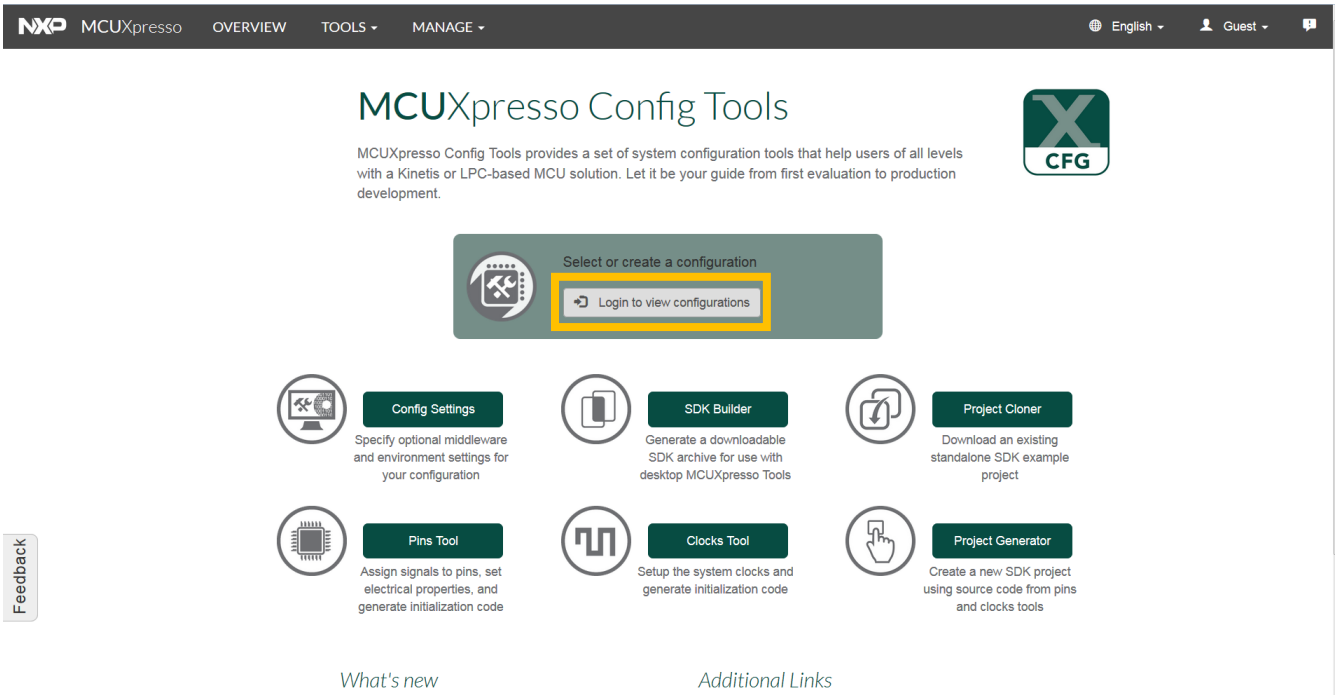

Figure 29. MCUXpresso Config Tools

3. You are redirected to the www.nxp.com login page. Enter your account information or create a new account.
4. Back on the MCUXpresso homepage, click the drop-down box and select the "New Configuration" option.

Figure 30. Creating a configuration

5. Search for the board name (for example; FRDM-K64F).

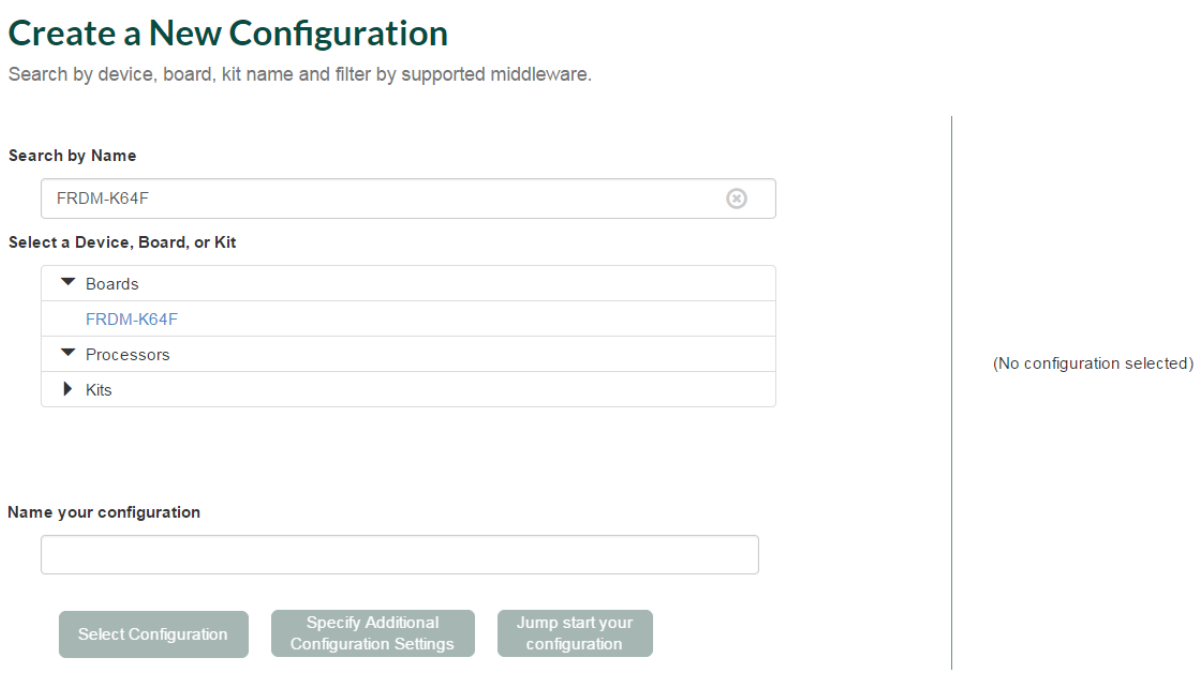

Figure 31. Creating a configuration

6. Choose a board from the list and provide a name for the configuration. Click the "Specify Additional Configuration Settings" button to select from FreeRTOS, IAR toolchain, and USB middleware.

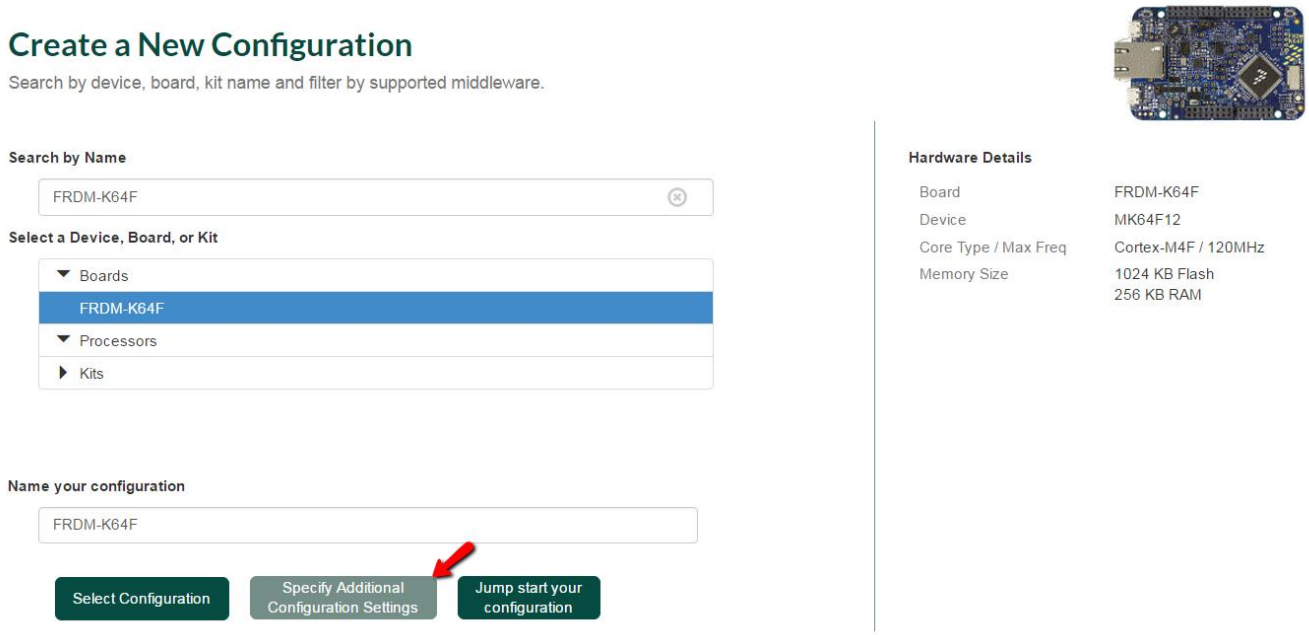

Figure 32. Creating a configuration

7. In the "Configuration Settings" section, set the following:

- Host OS → Windows.
- Toolchain/IDE → IAR Embedded Workbench for Arm.
- Middleware → USB Stack, FreeRTOS.

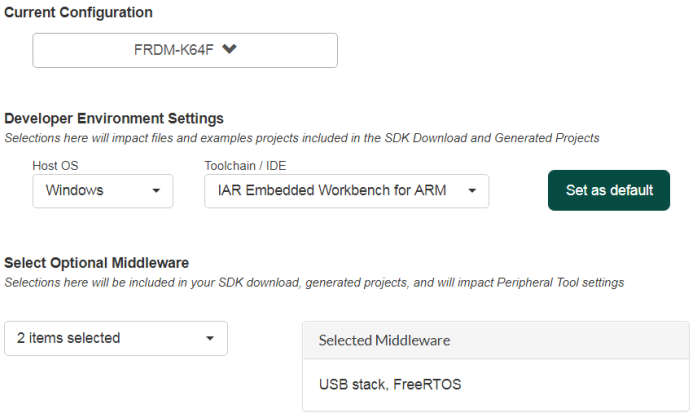

Figure 33. Configuration settings

Or:

- Host OS → Windows.
- Toolchain/IDE → MCUXpresso IDE.
- Middleware → USB Stack, FreeRTOS.

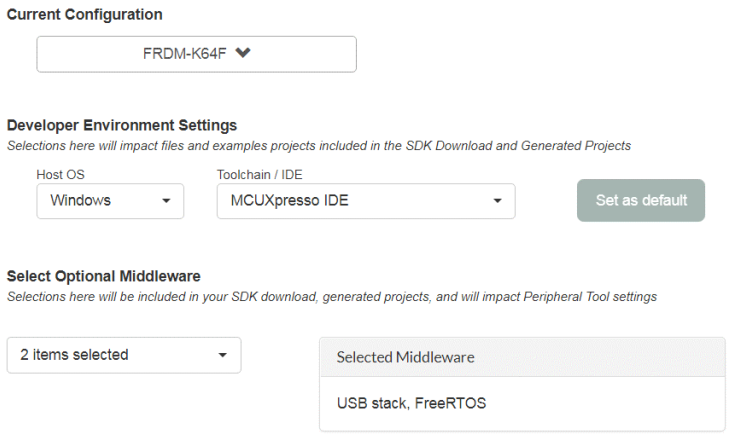

Figure 34. Configuration settings

8. After setting the configurations, click the "Go to SDK Builder" button.

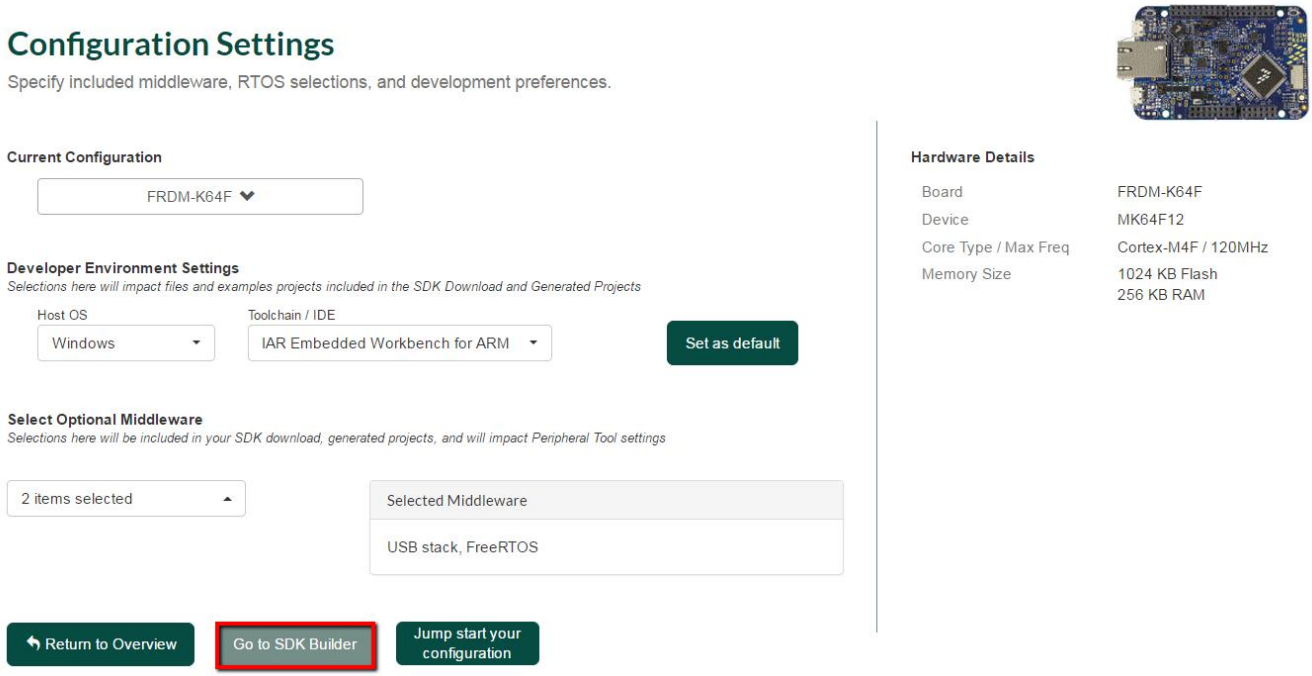

Figure 35. Configuration settings

9. Click the "Download Now" button to download the SDK package.

NOTE

If you see the "Request to Build" button instead of the "Go to SDK Builder" button, click the "Request to Build" button. When the package is built, a notification to download it appears in the "SDK Archive" section.

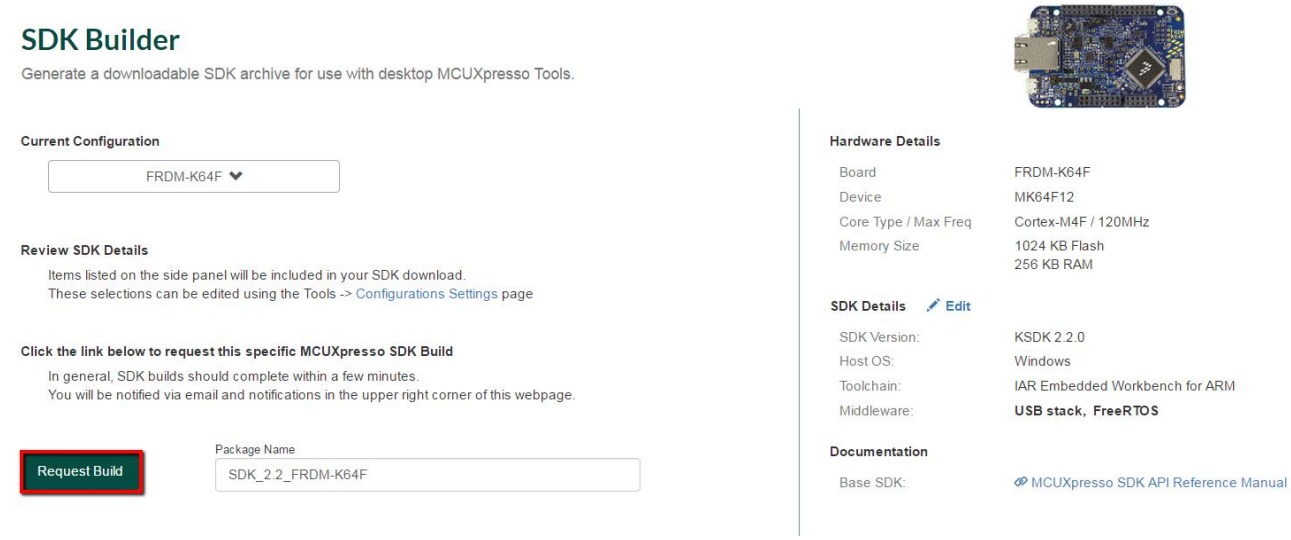

Figure 36. SDK Builder

10. To access the “SDK Archive” section, click the “MANAGE” tab and select the “SDK Archive” option.

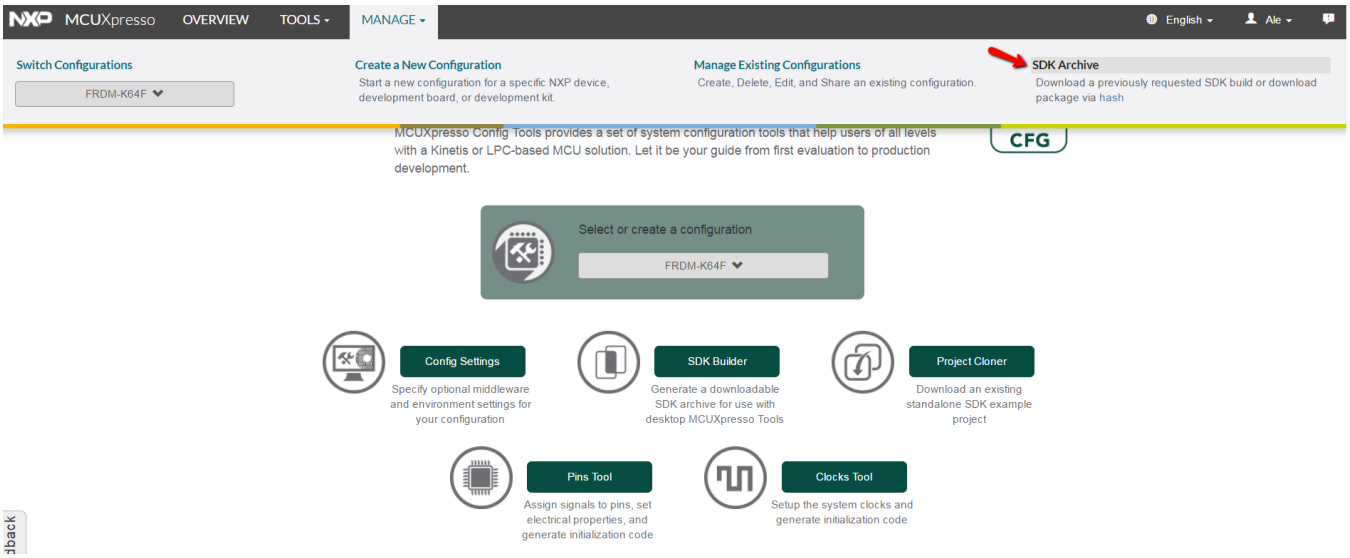

Figure 37. SDK Archive

11. Agree to the software terms and conditions.

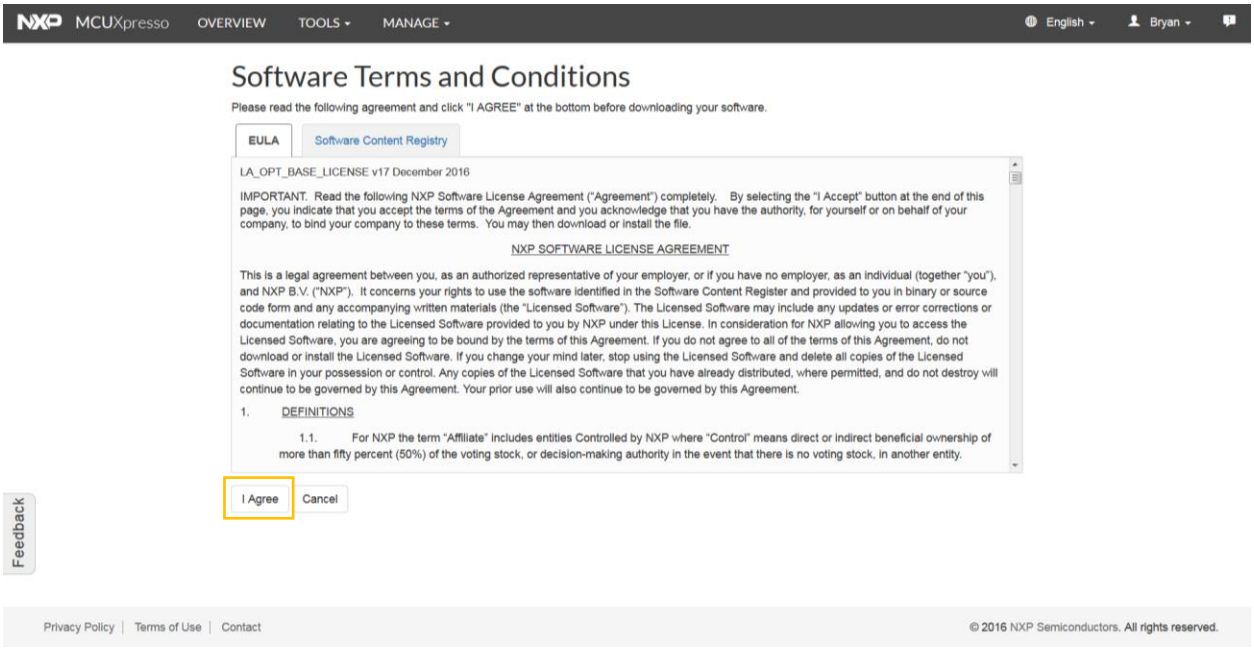

Figure 38. Terms and conditions

12. Unzip the SDK package to a folder (for example; *SDK_2.2_FRDM-K64F*)

Figure 39. SDK folder

Appendix C. Using MCUXpresso Config Tools for FRDM-K64F pins' initialization

1. Open the MCUXpresso Config Tools. If you don't have this program installed, visit www.nxp.com/mcuxpresso/config and click the "DOWNLOADS" tab to download and install it.

Figure 40. MCUXpresso icon

2. The wizard asks whether you want to start the development with or without an SDK package. Choose to start the development with the SDK package and select the "Create new configuration" option.

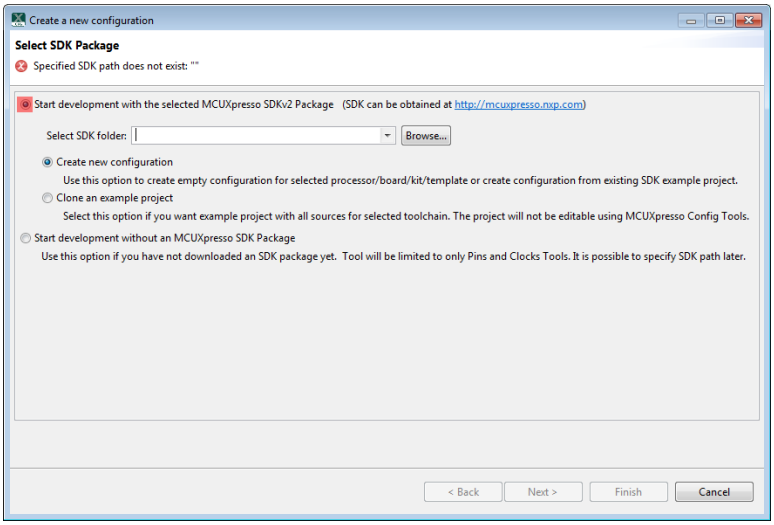

Figure 41. Creating a configuration

3. Use the "Browse…" button to navigate to the location where you unzipped the FRDM-K64F SDK installation. Select the SDK top-level folder from your file system and click the "OK" button.

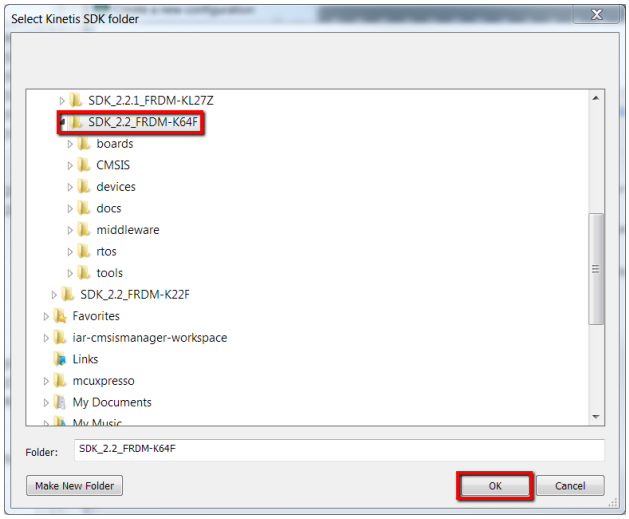

Figure 42. Selecting the SDK folder

4. The wizard asks you whether to create a new configuration or clone an example project. Select the “Create new configuration” option and click the “Next” button to continue.

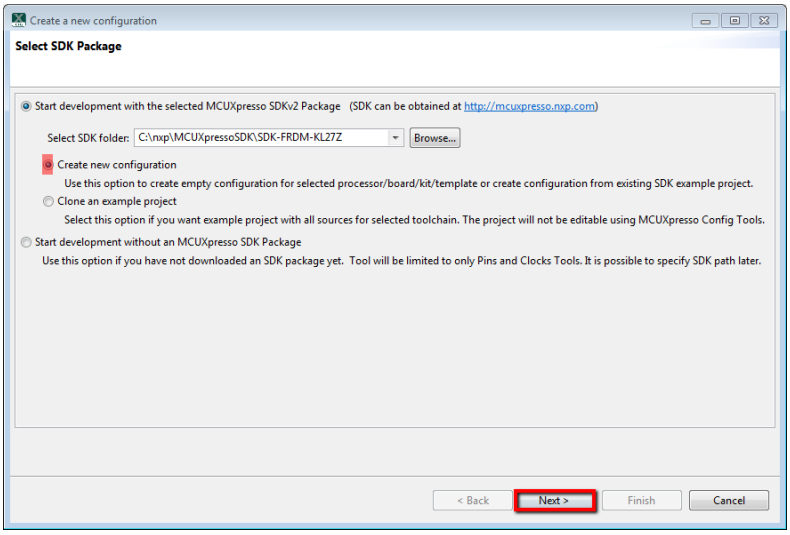

Figure 43. Selecting the SDK package

5. Select the “hello_world” option inside the “Name your configuration” section, rename it to “usbpd_k64”, and click the “Finish” button.

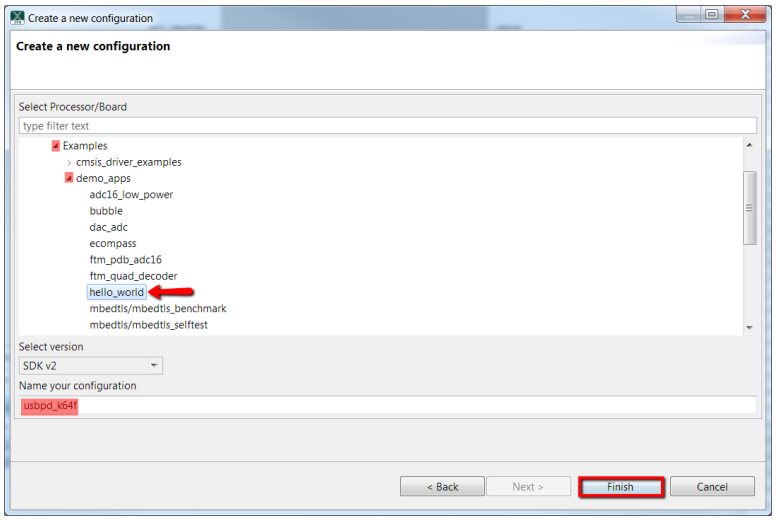

Figure 44. Selecting the "hello_world" option

6. If necessary (it may already be open), open the pins tool by selecting Tools → Pins from the toolbar.

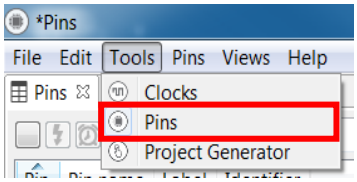

Figure 45. Opening the Pins tool

7. Use Table 2 to locate the FRDM-K64F pin assignment for the power request and power change switches. Route SW2 (PTC6) and SW3 (PTA4) as the GPIO inputs with the pull-up enabled. Search for PTC6 in the "Pins" view. Click the box under the GPIO column.

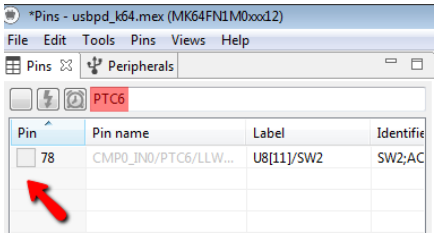

Figure 46. Pins tool

8. A new window pops up, showing all signals available on this pin. Select the GPIO functionality and click the "Done" button.

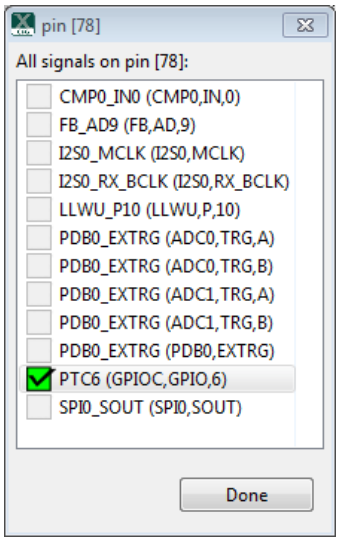

Figure 47. Available signals

9. Inside the “Routed Pins” tab, modify the PTC6 settings. Change the pin identifier to SW2 by right-clicking the identifier box and select the “Label” and “Identifier” tabs. Write “SW2” into the “Identifier” text box and click the “OK” button. Configure the input direction and select the “Pullup” in the “Pull select” tab and the “Enabled” option in the “Pull enable” tab.

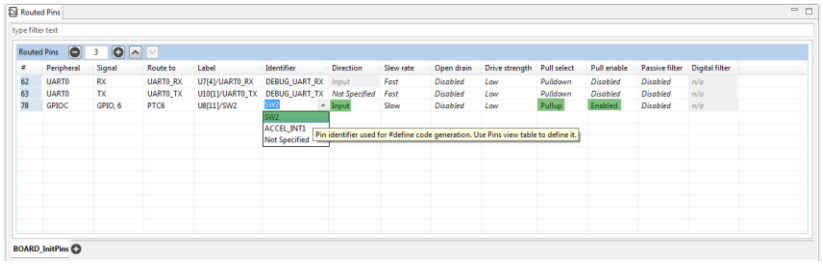

Figure 48. Modifying PTC6 settings

10. Repeat steps 7 to 9 to configure SW3 (PTA4).
11. Use Table 1 to locate the FRDM-K64F pin assignment for EXTRA_EN_SRC (PTB23). Route the PTB23 as a GPIO output. Search for PTB23 in the pins view. Click the box under the “GPIO” column. A new window pops up and displays all signals available on this pin. Select the “GPIO” functionality and click the “Done” button.
12. In the “Routed Pins” tab, change the pin identifier to EXTRA_EN_SRC by right-clicking the “Identifier” box and selecting the “Label” and “Identifier” tabs. Write “EXTRA_EN_SRC” into the “Identifier” text box and click the “OK” button. Change the direction to “Output”.

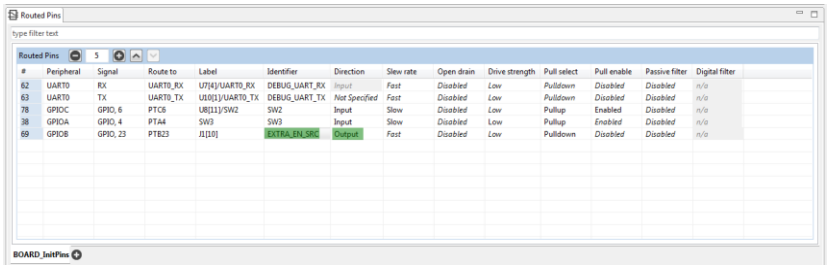

Figure 49. Changing the pin identifier

13. Use Table 1 to locate the FRDM-K64F pin assignment for nALERT (PTC12). Route PTC12 as a pin input. Search for PTC12 in the pins view. Click the box under the “GPIO” column. A new window pops up and displays all signals available on this pin. Select the GPIOC functionality and click the “Done” button.

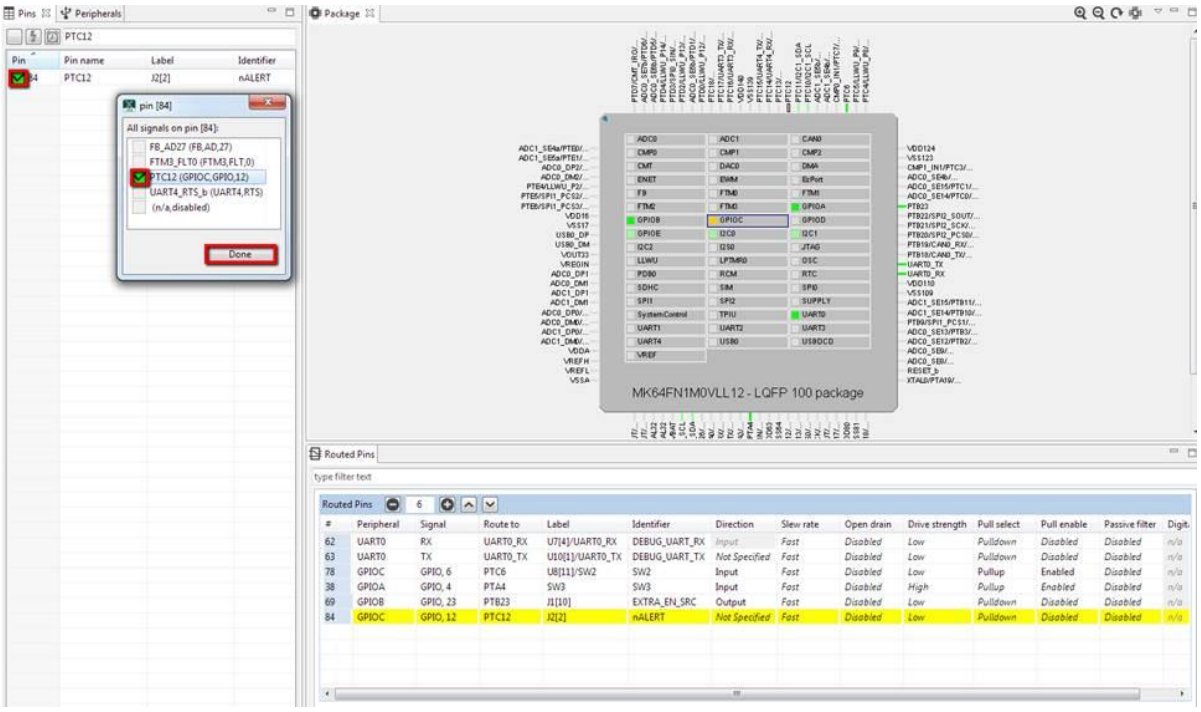

Figure 50. Available signals

14. Inside the “Routed Pins” tab, change the pin identifier to nALERT by right-clicking the “Identifier” box and select the “Label” and “Identifier” tabs. Write “nALERT” into the Identifier text box and click the “OK” button. Edit the “Mode” setting to “PullUp”.

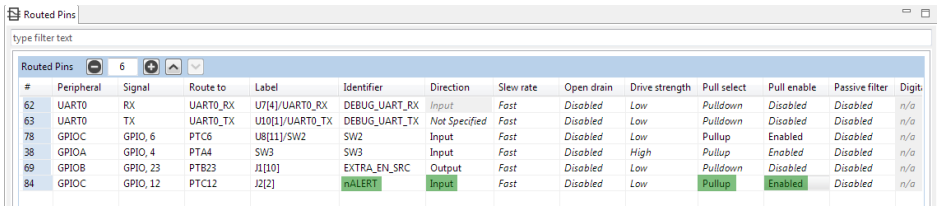

Figure 51. “Routed Pins” tab

15. Inside the “Routed Pins” tab, add a new function by clicking the “+” icon at the bottom of this tab.

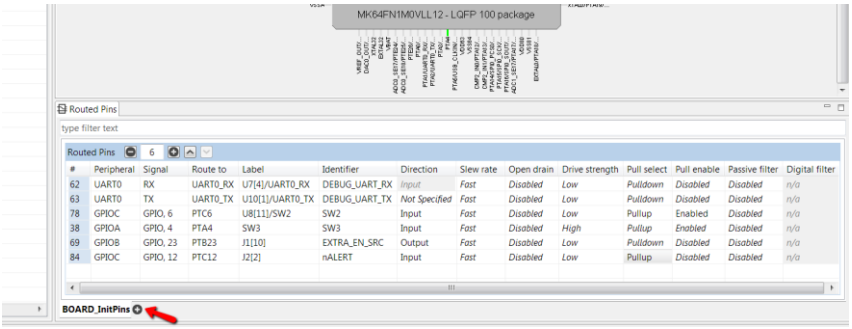

Figure 52. Adding a new function

16. Change the name of the recently created function to I2C**n**_InitPins, where **n** corresponds to the (D14, D15) I2C**n** number provided in Table 1. For the FRDM-K64F platform, the function should be named I2C0_InitPins. To rename the function, right-click its current name, then click the “Properties” option, write “I2C0_InitPins” into the “Function name” text box, and click the “OK” button.
17. Repeat step 16 for I2Cn_DeinitPins. 

18. Use Table 1 to locate the FRDM-K64F pin assignment for PTN5110_SDA (PTE25) and PTN5110_SCL (PTE24).
19. Select the I2C0_InitPins function and route the PTN5110_SDA and PTN5110_SCL as the I^2C pins. Search for “PTE25” in the “Pins” view and click the box under the GPIO column. A new window pops up and displays all signals available on this pin. Select the I2C0_SDA functionality and click the “Done” button.

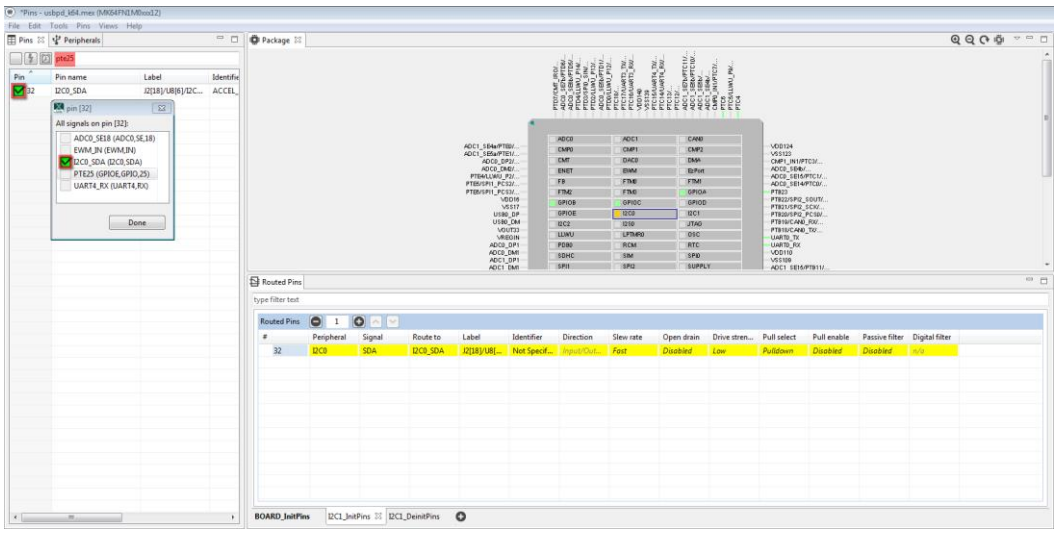

Figure 53. Available signals

20. In the “Routed Pins” tab, change the “Drive strength” to “High”.
21. Repeat steps 24 and 25 for the I2C4_SCL signal.

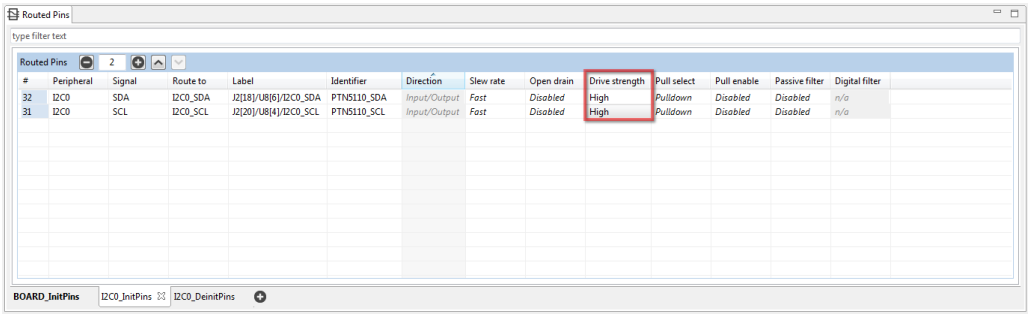

Figure 54. Changing drive strength

22. Select the I2C0_DeinitPins function and route PTN5110_SDA and PTN5110_SCL as GPIOs. Search for PTE25 in the pins view and click the box under the GPIO column. A new window pops up and displays all signals available on this pin. Tick the PTE25 GPIO checkbox and click the “Done” button.

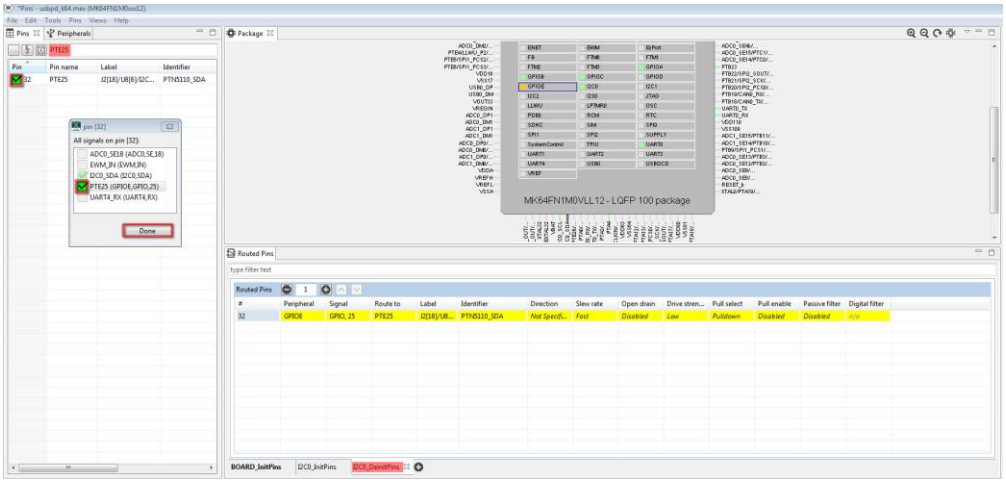

Figure 55. Available signals

23. Repeat the previous step for the PTN5110_SCL GPIO signal.
24. Repeat steps 19-23 for the A4-A5 routed pins.
25. Export the *pin_mux.c* and *pin_mux.h* files by clicking the “Source” tab on the right side and clicking the “export” icon.

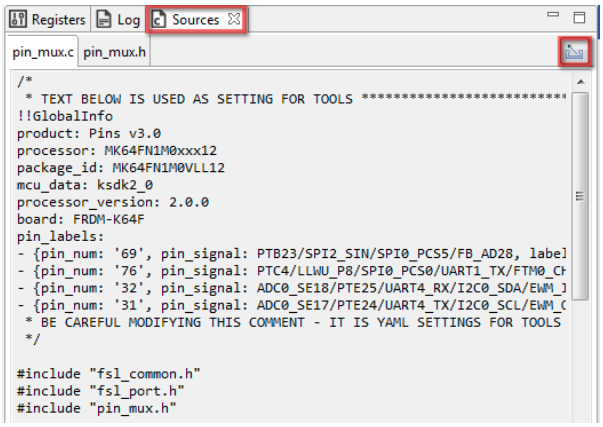

Figure 56. Sources tab

26. Select the folder to export the *pin_mux.c* and *pin_mux.h* files to. For the usb_pd_freertos project, select the *...\boards\frdmk64f\usb_examples\usb_pd\freertos* folder and click the "Finish" button.

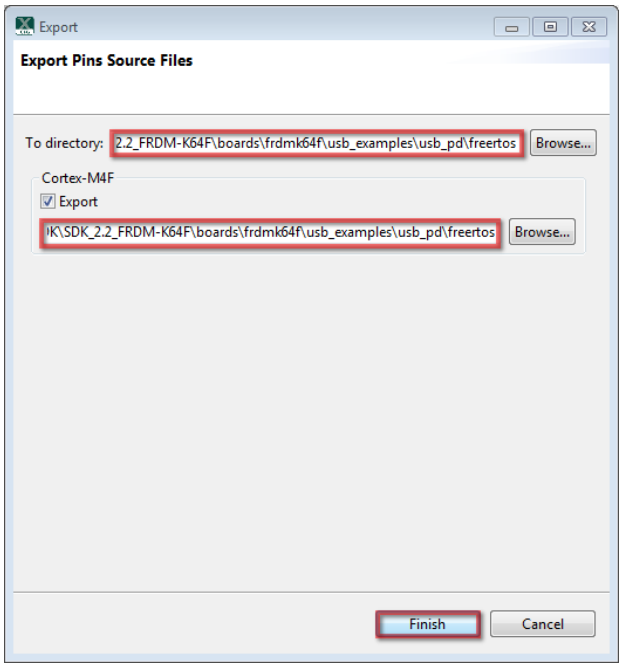

Figure 57. Export window

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: USBPDMUG
Rev. 0
10/2017