

## 1 Introduction

This document describes how to migrate from Kinetis MKW38A512xxx4 MCUs to Kinetis MKW39A512xxx4 MCUs with emphasis on connectivity software. In this document, the MKW38A512xxx4 and MKW39A512xxx4 are referred to as MKW38 and MKW39, respectively. The document is intended for software engineers, software testers, software integrators, and customers designing their own hardware.

## 2 Hardware considerations

The MKW39 wireless MCU is pin-to-pin compatible with MKW38 and almost all peripherals are the same in both devices. The main difference between the MKW38 and MKW39 MCUs is the number of communication interfaces. The MKW38 MCU presents a second instance of LPUART and a FlexCAN modules.

**Table 1. Differences between MKW38 and MKW39**

Device	Second instance of LPUART	FlexCAN module
MKW38	X	X
MKW39		

### 2.1 Peripherals instantiation

The KW38/39 devices in the 48-pin HVQFN package are pin-to-pin compatible. The MKW39 MCU does not support LPUART1 and FlexCAN. Therefore, the signals of the last instances are not available for multiplexing in this device. The **bold** alternatives are only included for the MKW38 MCU.

**Table 2. MKW38/39 instance comparative**

Pin Name	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	ALT8	ALT9
PTA16		PTA16/ LLWU_P 4	SPI1_SO UT	<b>LPUART 1_ RTS_b</b>		TPM0_C H0				
PTA17		PTA17/ LLWU_P 5	SPI1_SIN	<b>LPUART 1_ RX</b>	<b>CAN0_T X</b>	TPM_ CLKIN1				

*Table continues on the next page...*

### Contents

- 1 Introduction..... 1
- 2 Hardware considerations..... 1
- 3 Software development kit download and install ..... 3
- 4 Software migration in IAR Embedded Workbench IDE.....5
- 5 Software migration in MCUXpresso IDE..... 9
- 6 Build and run Bluetooth LE connectivity stack examples.....16



**Table 2. MKW38/39 instance comparative (continued)**

<b>PTA18</b>		PTA18/ LLWU_P 6	SPI1_SC K	<b>LPUART 1_ TX</b>	<b>CAN0_R X</b>	TPM2_C H0				
<b>PTA19</b>	ADC0_S E5	PTA19/ LLWU_P 7	SPI1_PC S0	<b>LPUART 1_ CTS_b</b>		TPM2_C H1				
<b>PTB0</b>		PTB0/ LLWU_P 8/ RF_ RFOSC_ EN		I2C0_SC L	CMP0_O UT	TPM0_C H1		CLKOUT	<b>CAN0_T X</b>	
<b>PTB1</b>	ADC0_S E1/ CMP0_IN 5	PTB1/ RF_ PRIORIT Y	DTM_RX	I2C0_SD A	LPTMR0 _ ALT1	TPM0_C H2		CMT_IR O	<b>CAN0_R X</b>	
<b>PTB3</b>	ADC0_S E2/ CMP0_IN 4	PTB3/ ERCLK3 2K/ RF_ACTI VE	<b>LPUART 1_ RTS_b</b>	TPM0_C H1	CLKOUT	TPM1_C H1		RTC_ CLKOUT	TPM2_C H1	
<b>PTB16</b>	EXTAL32 K	PTB16	<b>LPUART 1_ RX</b>	I2C1_SC L		TPM2_C H0				
<b>PTB17</b>	XTAL32K	PTB17	<b>LPUART 1_ TX</b>	I2C1_SD A		TPM2_C H1				
<b>PTB18</b>	ADC0_S E4/ CMP0_IN 2	PTB18	<b>LPUART 1_ CTS_b</b>	I2C1_SC L	TPM_ CLKIN0	TPM0_C H0		NMI_b		
<b>PTC3</b>		PTC3/ LLWU_P 11	RX_ SWITCH	I2C1_SD A	LPUART 0_ TX	TPM0_C H1		DTM_TX	SPI1_SIN	<b>CAN0_T X</b>

*Table continues on the next page...*

**Table 2. MKW38/39 instance comparative (continued)**

<b>PTC4</b>		PTC4/ LLWU_P 12/ RF_ACTI VE	ANT_A	EXTRG_I N	LPUART 0_ CTS_b	TPM1_C H0		I2C0_SC L	SPI1_PC S0	<b>CAN0_R X</b>
<b>PTC16</b>		PTC16/ LLWU_P 0/ RF_ STATUS	SPI0_SC K	I2C0_SD A	LPUART 0_ RTS_b	TPM0_C H3			<b>LPUART 1_ RTS_b</b>	
<b>PTC17</b>		PTC17/ LLWU_P 1/ RF_EXT_ OSC_EN	SPI0_SO UT	I2C1_SC L	LPUART 0_ RX			DTM_RX	<b>LPUART 1_ RX</b>	
<b>PTC18</b>		PTC18/ LLWU_P 2	SPI0_SIN	I2C1_SD A	LPUART 0_ TX			DTM_TX	<b>LPUART 1_ TX</b>	
<b>PTC19</b>		PTC19/ LLWU_P 3/ RF_ EARLY_ WARNIN G	SPI0_PC S0	I2C0_SC L	LPUART 0_ CTS_b				<b>LPUART 1_ CTS_b</b>	

**NOTE**

Table 2 is not a full description of the MKW38/39 pinout. It can be found in the MKW39/38/37 reference manual.

### 3 Software development kit download and install

This chapter shows how to download the Software Development Kit (SDK) for MKW39A512xxx4 MCUs, because it is used as a starting point to migrate MKW38 code to the MKW39 MCU. Perform the following steps to download the SDK package:

1. Go to the MCUXpresso web page ([mcuxpresso.nxp.com](http://mcuxpresso.nxp.com)).
2. Log in with your registered account.
3. Search for the “KW39” device. Then click on the suggested processor and click the “Build MCUXpresso SDK” button.

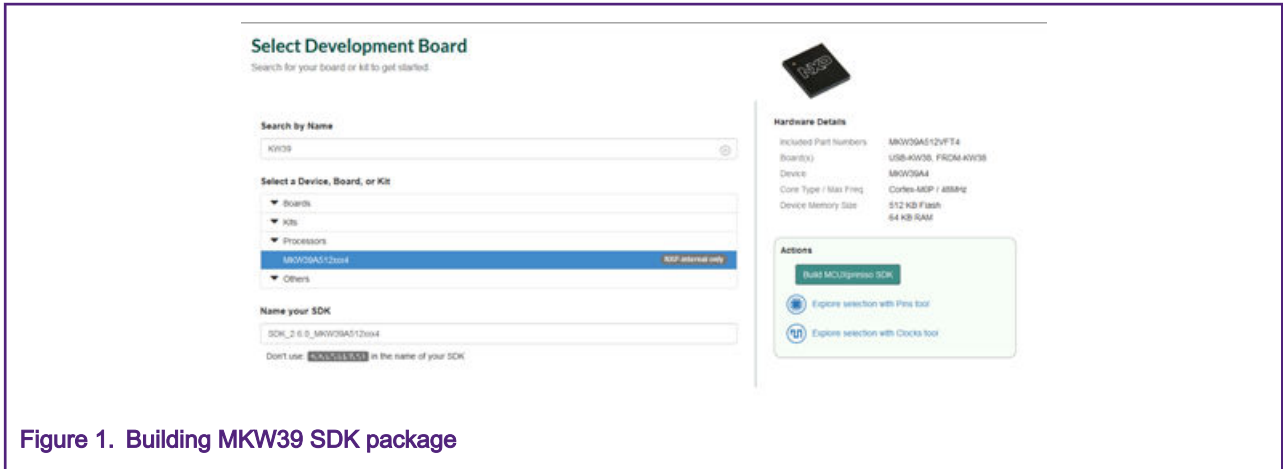


Figure 1. Building MKW39 SDK package

- The next page is displayed. Select “All toolchains” option in the “Toolchain / IDE” box and provide a name to identify the package.

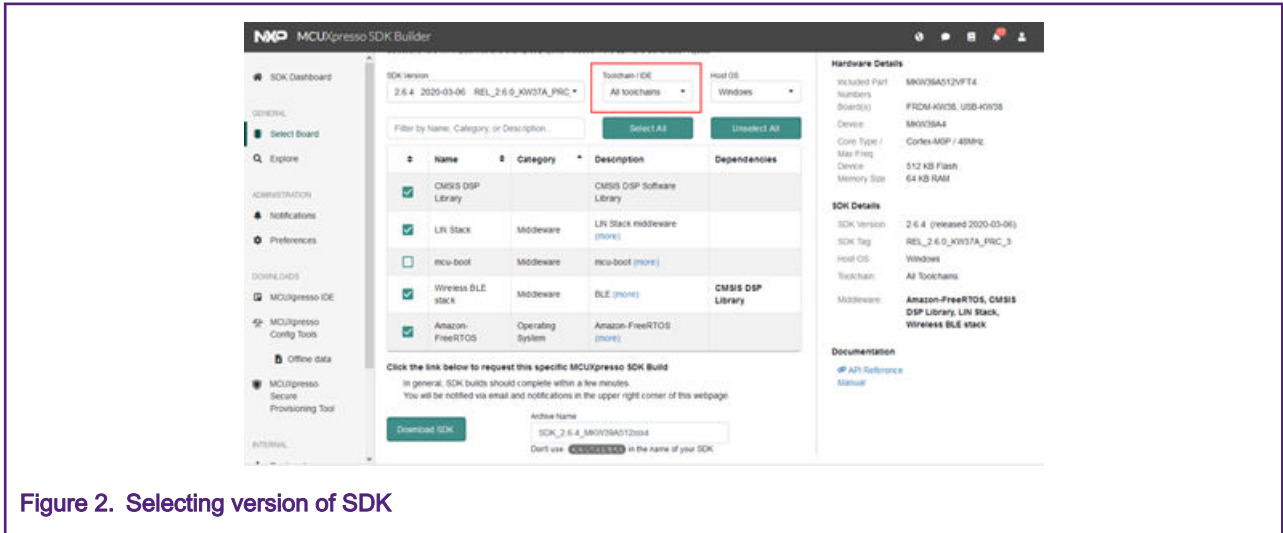


Figure 2. Selecting version of SDK

- Click the “Download SDK” button. This starts the building process of the desired SDK. It takes a few minutes until the system gets the package into your profile at the MCUXpresso web page.
- When the SDK is ready to be downloaded, the “Software Terms and Conditions” are displayed. Accept them and the download process starts automatically.
- If the download does not start automatically, click the “Download SDK Archive” option.

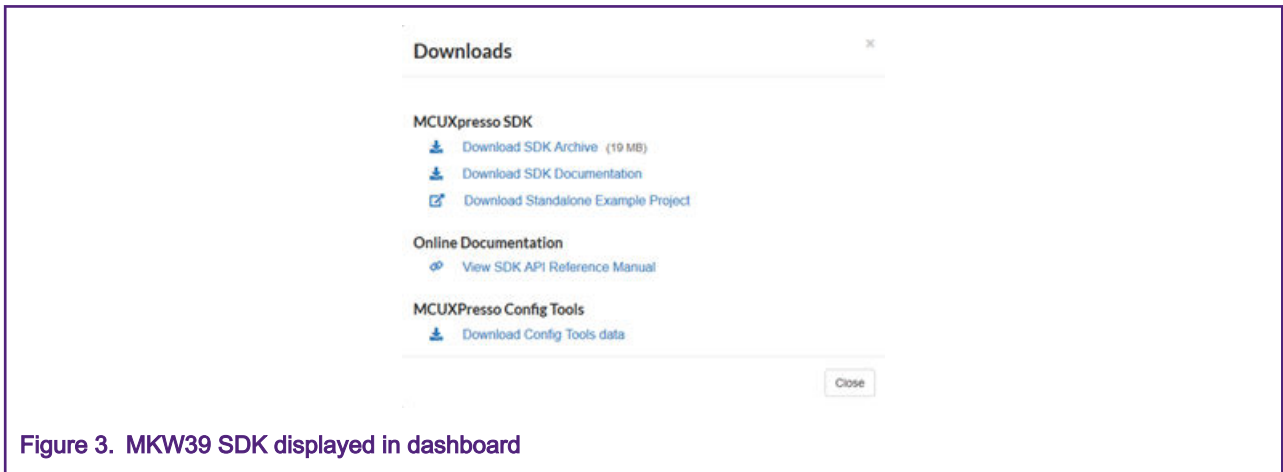


Figure 3. MKW39 SDK displayed in dashboard

- If the above picture is not displayed, click the “Download SDK archive and documentation” button from “MCUXpresso SDK Dashboard”.

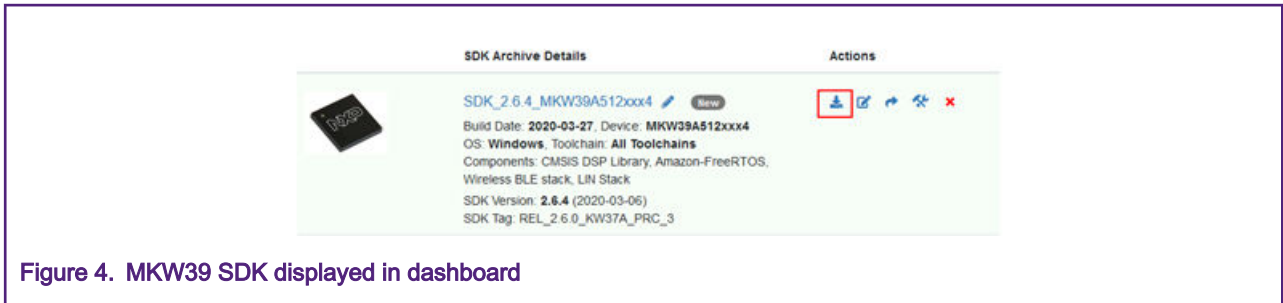


Figure 4. MKW39 SDK displayed in dashboard

At this point, you have downloaded and installed the SDK package for MKW39A devices.

## 4 Software migration in IAR Embedded Workbench IDE

This chapter shows how to migrate the MKW38 example code to the MKW39 MCU in the IAR Embedded Workbench IDE. The heart rate sensor project is used as the base in this document, because it has easy-to-understand example and involves the Bluetooth LE connectivity software stack (included in the SDK).

### 4.1 Changes required in project options and settings

#### NOTE

In this section, a “bare-metal” version of the project was used. However, the same steps apply for FreeRTOS projects. Some of the paths related to “bare-metal” projects may differ for FreeRTOS versions.

- Open the heart rate sensor project located at the following path:

*<SDK\_root>/boards/frdmkw38/wireless\_examples/bluetooth/hrs/bm/iar/heart\_rate\_sensor\_bm.eww*

- Select the project in the workspace and press Alt + F7 to open project options.

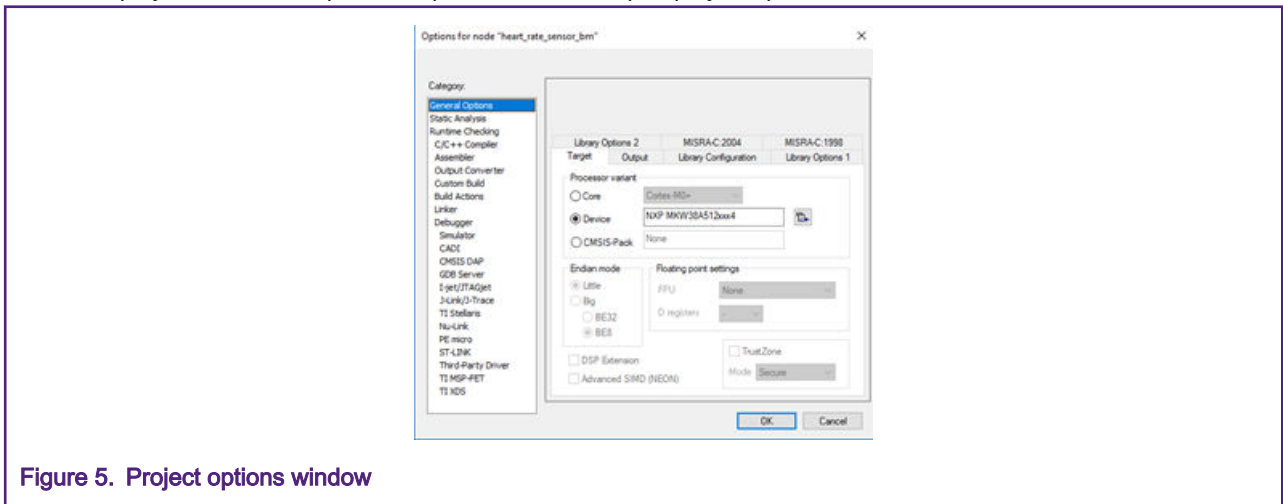


Figure 5. Project options window

- In the "General Options/Target" window, click the icon next to the device name and select the appropriate device (NXP->KinetisKW->KW3x->NXP MKW37A512xxx4) and click the "OK" button.

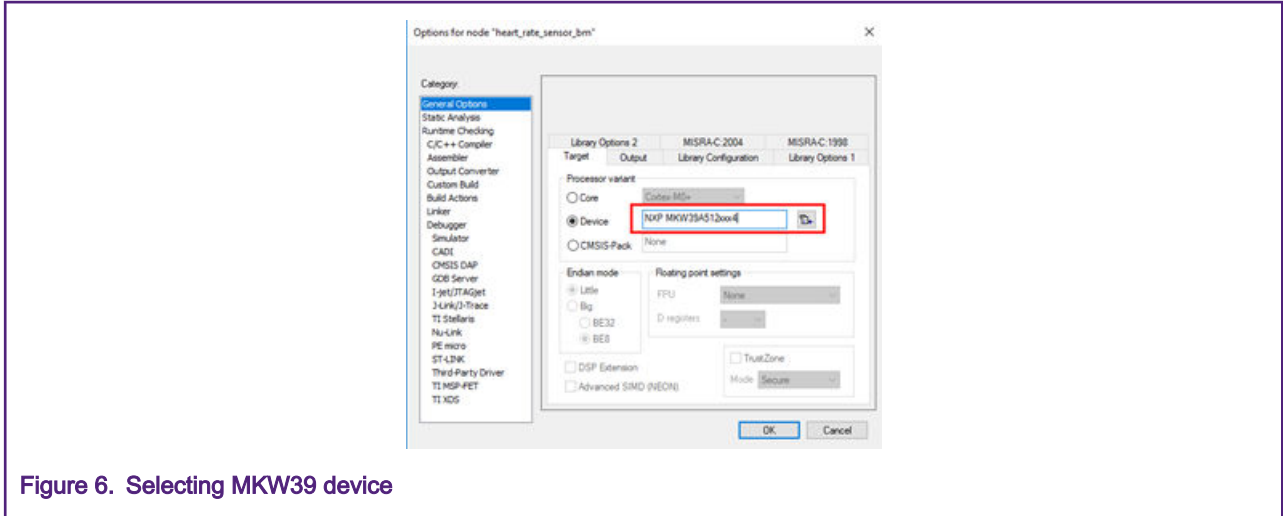


Figure 6. Selecting MKW39 device

4. Create a new folder named *MKW39A4* at following paths:  
`<SDK_root>/middleware/wireless/framework/DCDC/Interface`  
`<SDK_root>/middleware/wireless/framework/DCDC/Source`  
`<SDK_root>/middleware/wireless/framework/LowPower/Interface`  
`<SDK_root>/middleware/wireless/framework/LowPower/Source`  
`<SDK_root>/middleware/wireless/framework/XCVR`

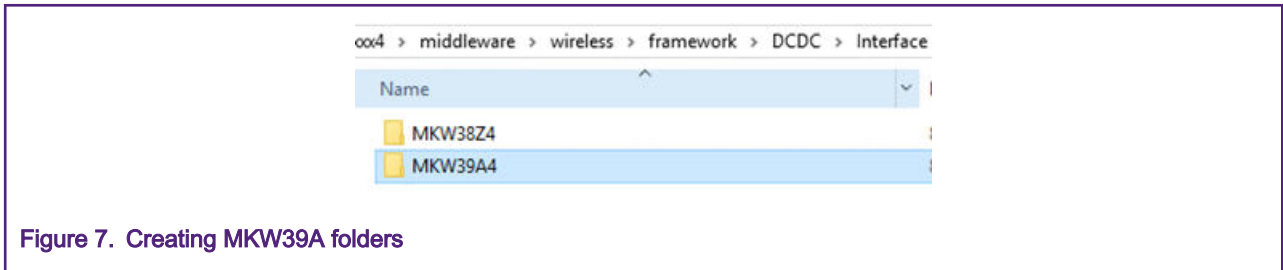


Figure 7. Creating MKW39A folders

5. Copy all files from the *MKW38Z4* folders located at the path mentioned above and paste them into the *MKW39A4* folders.

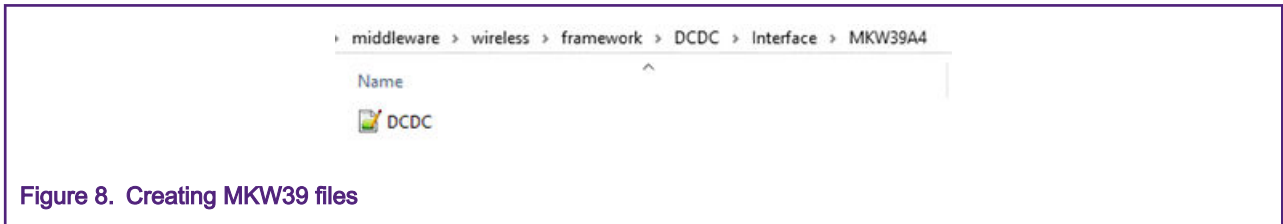


Figure 8. Creating MKW39 files

6. Select the heart rate sensor application in the workspace and press Alt+F7 to open the "project options" window. In the "C/C++ Compiler/Preprocessor" window, rename the paths related to the *MKW38A/Z4* folders to "MKW39A4" by modifying the name in the "additional include directories" text box. The results must look similar to the following:  
`$PROJ_DIR$/../../../../../../devices/MKW39A4/utilities`  
`$PROJ_DIR$/../../../../../../devices/ MKW39A4/drivers`  
`$PROJ_DIR$/../../../../../../middleware/wireless/framework/LowPower/Interface/MKW39A4`  
`$PROJ_DIR$/../../../../../../devices/MKW39A4/utilities/str`  
`$PROJ_DIR$/../../../../../../devices/MKW39A4/utilities/debug_console`

```
$PROJ_DIR$/../../../../../../../../middleware/wireless/framework/DCDC/Interface/ MKW39A4
$PROJ_DIR$/../../../../../../../../middleware/wireless/framework/XCVR/MKW39A4/drv
$PROJ_DIR$/../../../../../../../../middleware/wireless/framework/XCVR/MKW39A4/drv/nb2p4ghz
$PROJ_DIR$/../../../../../../../../middleware/wireless/framework/XCVR/MKW39A4/drv/nb2p4ghz/configs/gen35
```

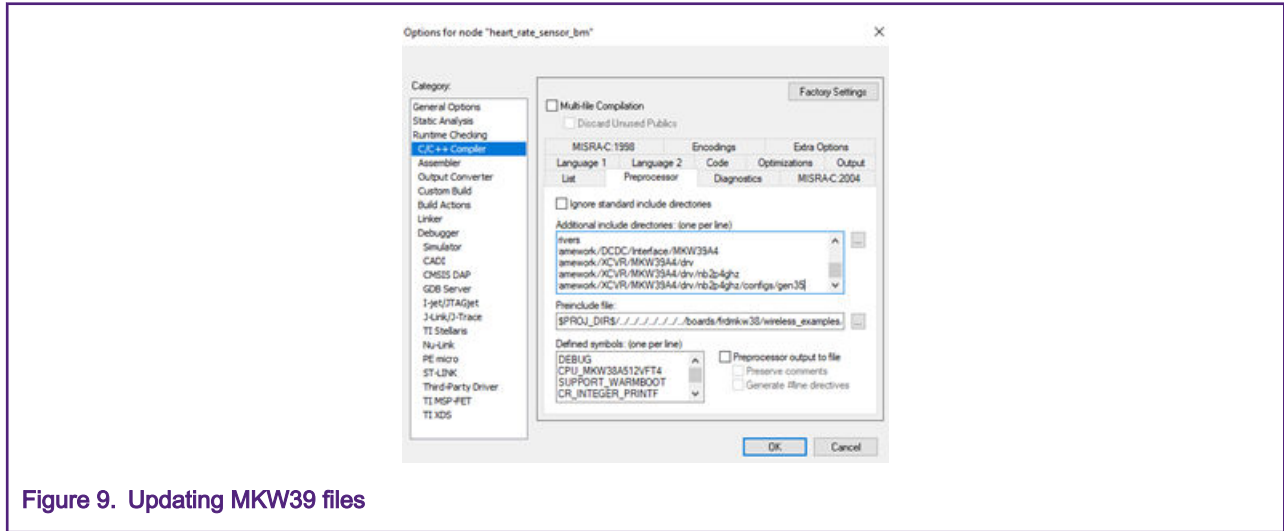


Figure 9. Updating MKW39 files

- Rename the CPU\_MKW38A512VFT4 macro to CPU\_MKW39A512VFT4 and delete the FRDM\_KW38 macro in the "defined" symbols text box. Click the "OK" button.

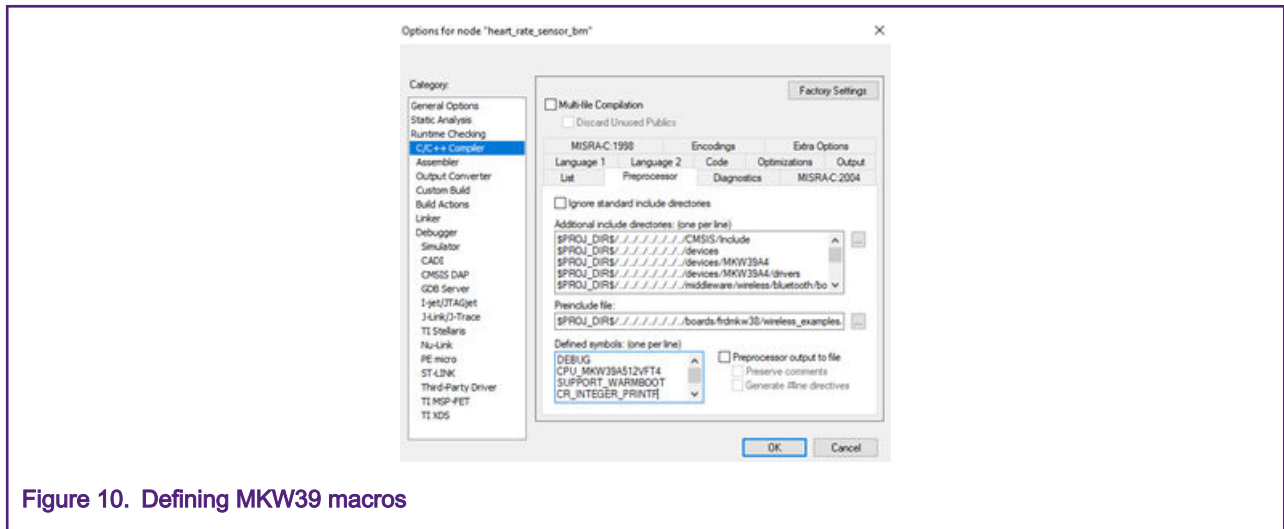


Figure 10. Defining MKW39 macros

- Expand the startup folder, select all files, right-click and select the "Remove" option. Right-click the folder and select "Add/Add files". Add the *startup\_MKW39A4.s* file located at this path:

```
<SDK_root>/devices/MKW39A4/iar/startup_MKW39A4.s
```

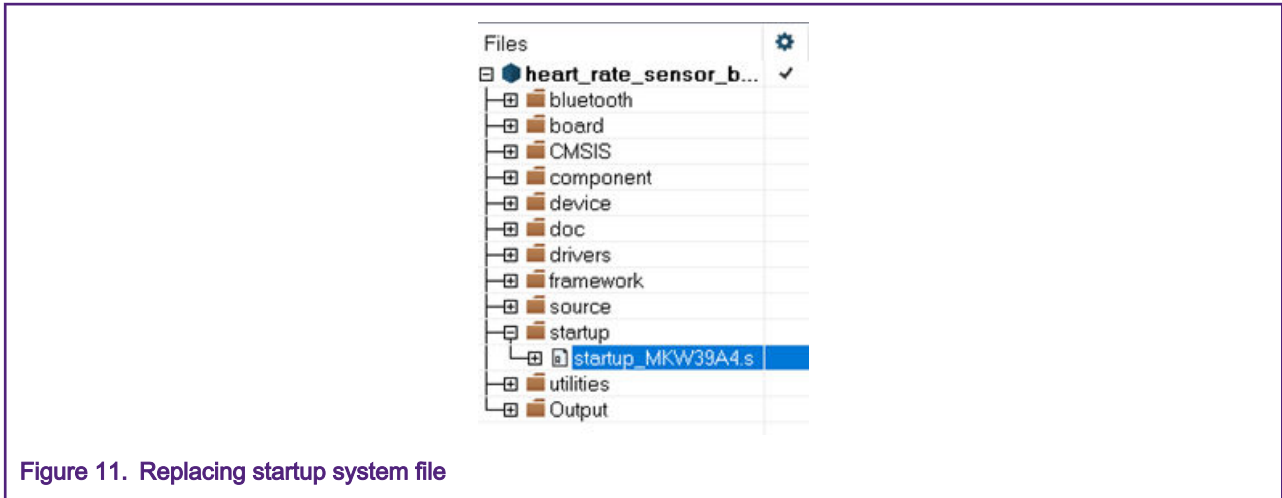


Figure 11. Replacing startup system file

- Expand the device folder, select all files, right-click, and select the “Remove” option. Right-click the folder and select “Add/Add files”. Add the *fsl\_device\_registers.h*, *MKW39A4.h*, *MKW39A4\_features.h*, *system\_MKW39A4.h*, and *system\_MKW39A4.c* files at the following path:

<SDK\_root>/devices/MKW39A4

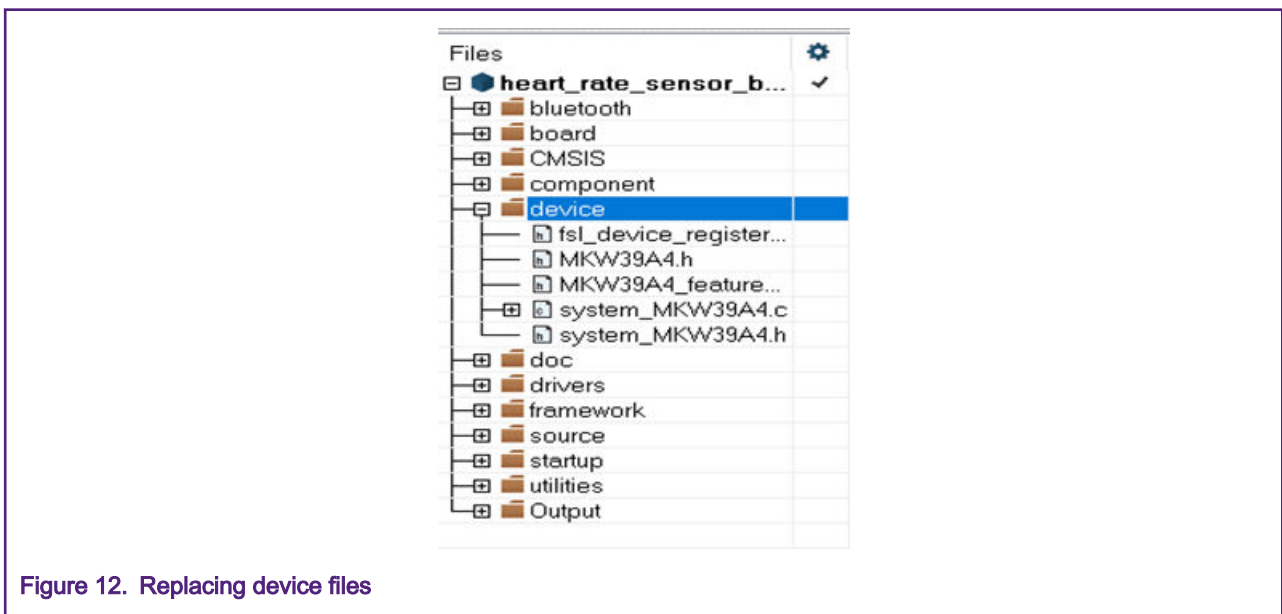


Figure 12. Replacing device files

- Expand the *drivers* folder, select the *fsl\_flexcan.h* and *fsl\_flexcan.c* files, right-click, and select the “Remove” option.
- Some files must be updated. Open the *clock\_config.c* file in the *board* folder. Delete the `CLOCK_SetLpuart1Clock` call to function in the `BOARD_BootClockRUN` function.

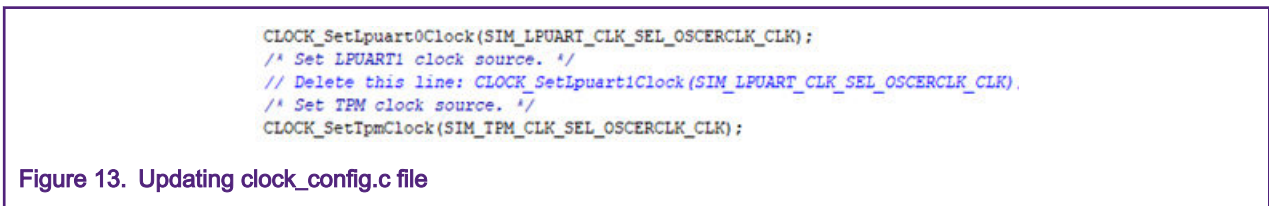


Figure 13. Updating clock\_config.c file

- If necessary, open the *app\_preinclude.h* file in the *source* directory in the workspace. This file contains important information about the board, such as the number of switches and LEDs, timers, power consumption settings, and so on. Examine and update this file to fit into your own custom board.



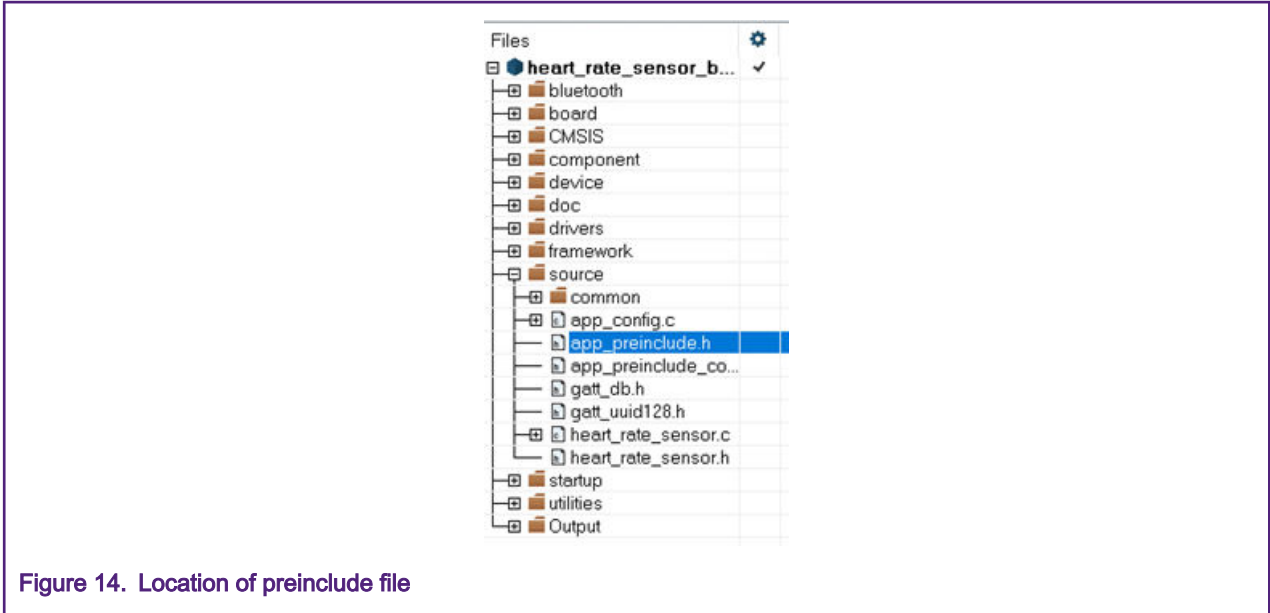


Figure 14. Location of preinclude file

13. If it necessary, open the pin mux files (*pin\_mux.c* and *pin\_mux.h*) and gpio files (*gpio\_pins.c* and *gpio\_pins.h*) in the *board* directory in the workspace. These files contain alternatives, options, and multiplexing information of the pins. Examine and update them to fit into your own custom board.

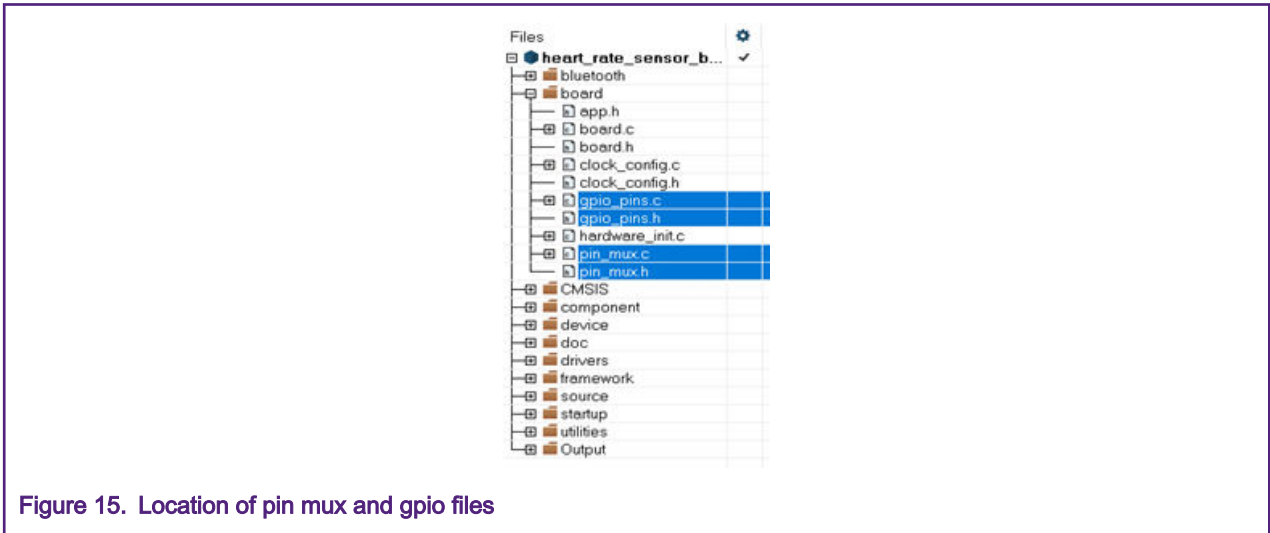


Figure 15. Location of pin mux and gpio files

14. Select the project in the workspace, right-click and press clean.
15. Rebuild the project. At this point, the project is already migrated and you can customize the application for your own goal using the MKW39 device.

## 5 Software migration in MCUXpresso IDE

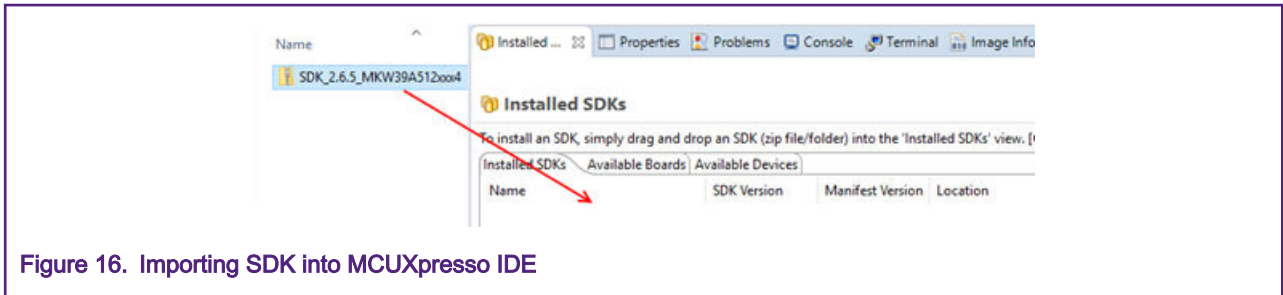
This chapter shows how to migrate MKW38 example code to the MKW39 MCU in the MCUXpresso IDE. The heart rate sensor project is used as a base because it has easy-to-understand example and involves the Bluetooth LE connectivity software stack (included in the SDK).

## 5.1 Changes required in project properties

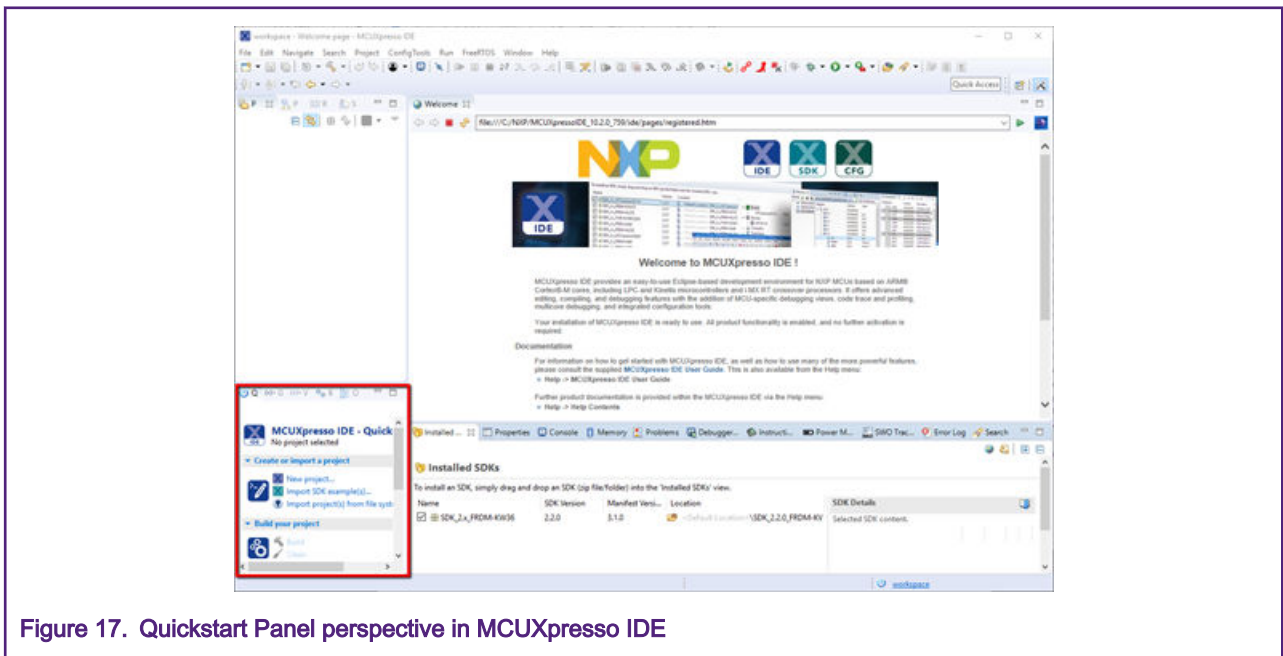
### NOTE

In this section, a “bare-metal” version of the project is used. However, the same steps apply for FreeRTOS projects. Some paths related to the “bare-metal” projects may differ in the FreeRTOS versions.

1. Import the KW39 SDK into the MCUXpresso IDE:
  - a. Open the MCUXpresso IDE.
  - b. Go to the “Installed SDKs” tab and drag and drop the KW39 SDK.



2. Find the "Quickstart Panel" in the lower left-hand corner.



3. Click the “Import SDK examples(s)...” option.

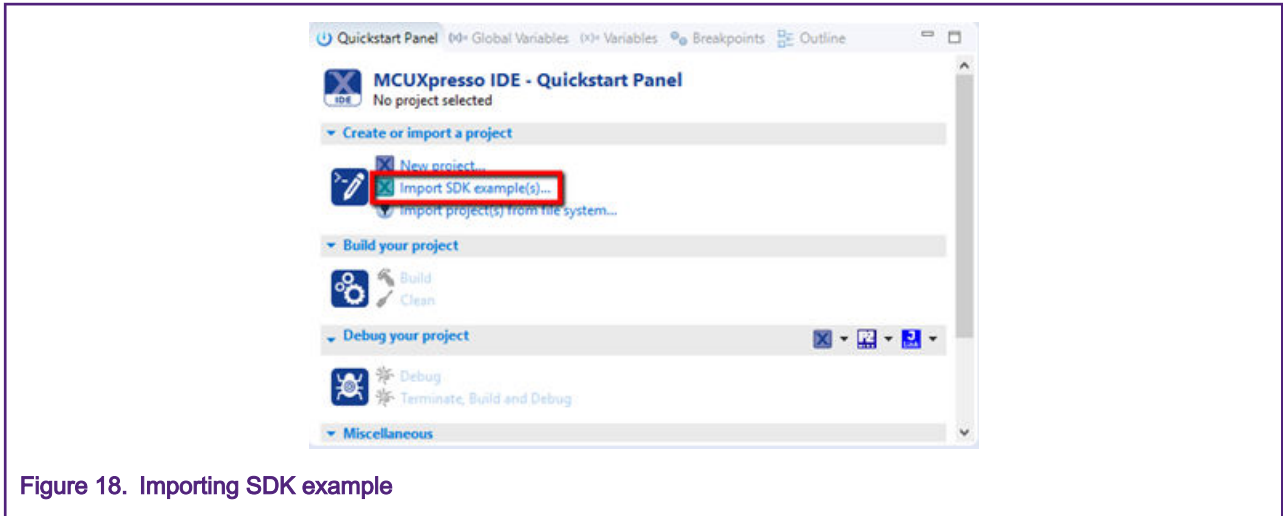


Figure 18. Importing SDK example

- Expand the *KW3X* folder, select "MKW38A512xxx4", click the "frdmkw38" board, and click "Next".

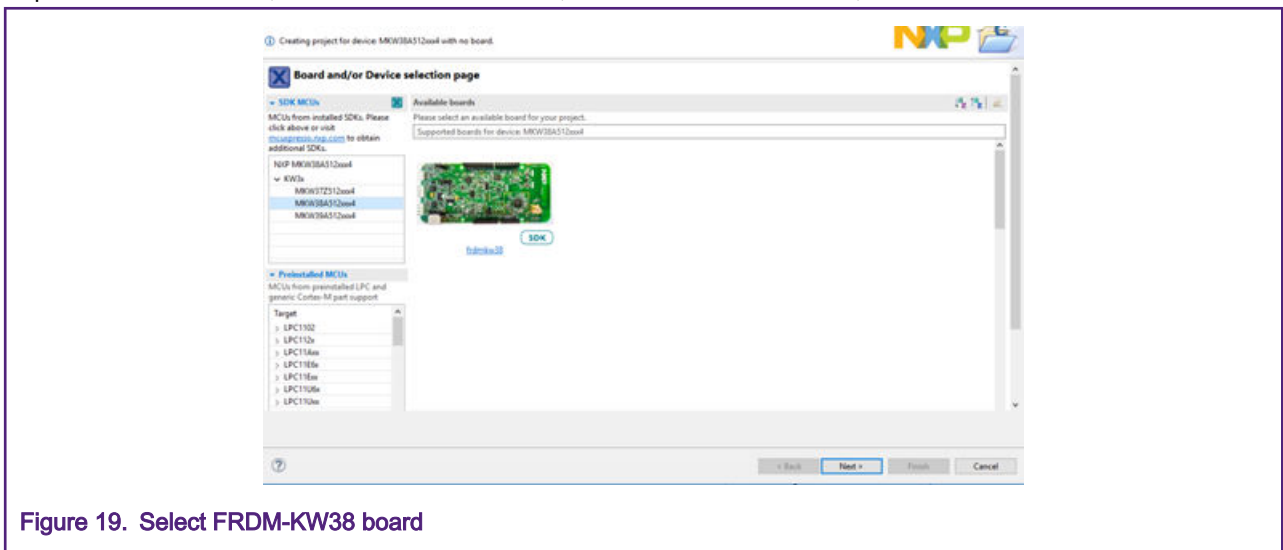


Figure 19. Select FRDM-KW38 board

- Use the arrow button to expand the list and locate the "hrs" project (wireless\_examples -> bluetooth -> hrs). Select the "bm" version of the project. In "SDK Debug Console", select "UART" and click "Finish".

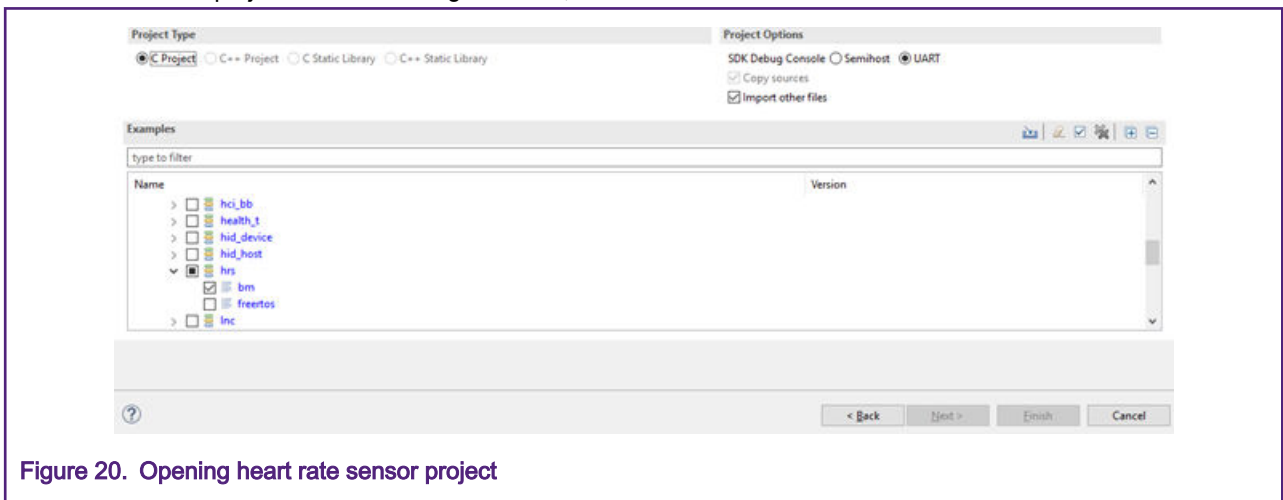


Figure 20. Opening heart rate sensor project

- Go to "Project/Properties". Expand "C/C++ Build/MCU settings" and select the "MKW39A512xxx4" MCU. Click the "Apply and Close" button to save the configuration.

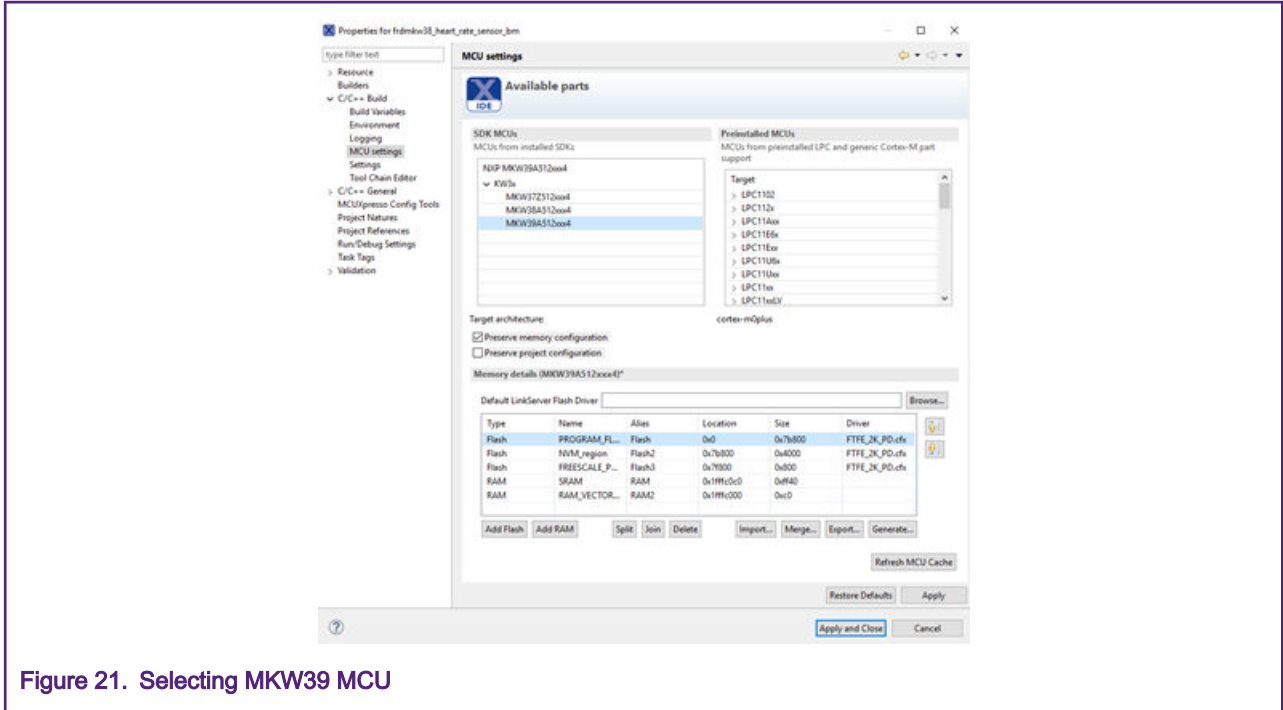


Figure 21. Selecting MKW39 MCU

7. Rename the *MKW38Z4* folder to "MKW39A4" at the following path:  
 <MCUXpresso\_hrs\_project\_root>/framework/XCVR  
 Default MCUXpresso root in Documents/MCUXpressoIDE<version>/workspace
8. Select the heart rate sensor project in the workspace and press F5 to refresh the modified folders.

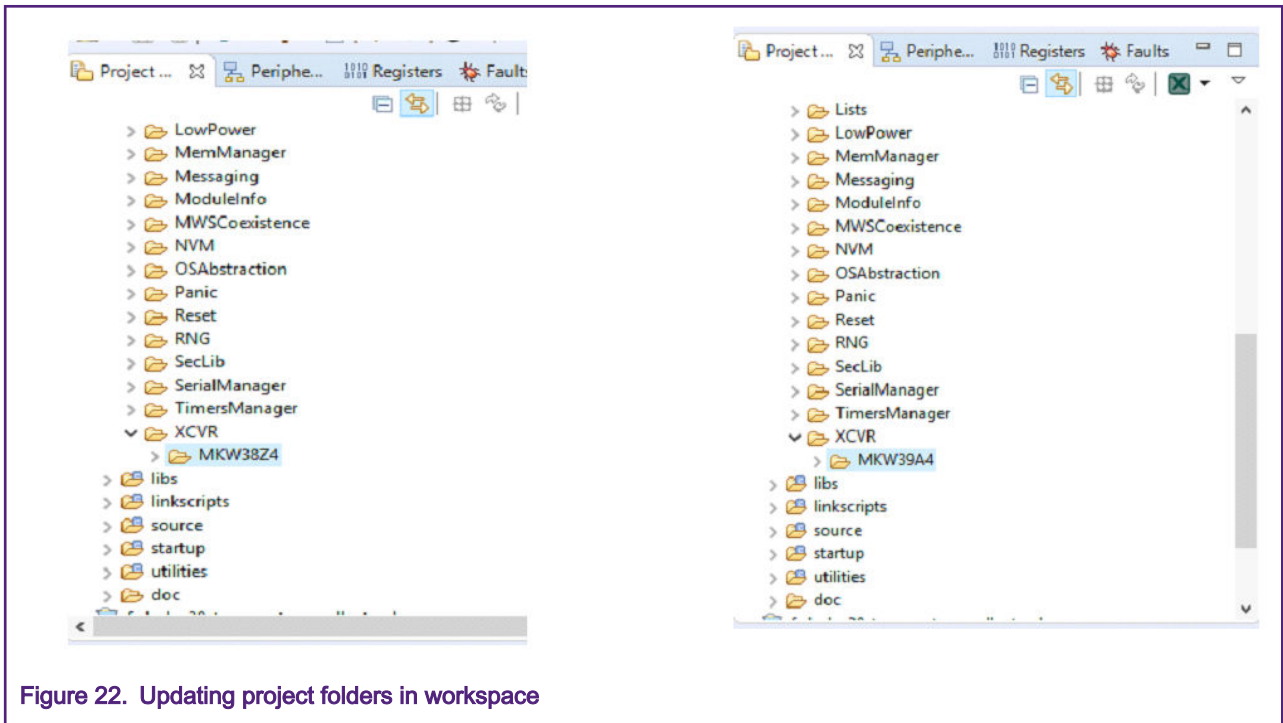


Figure 22. Updating project folders in workspace

9. Open the "Project/Properties" window in the MCUXpresso IDE. Go to "C/C++ Build/Settings" and select the *MCU C Compiler/Includes* folder in the "Tool Settings" window. Edit all paths related to the MKW38 MCU, in accordance with the *MKW39* folders created before. The results must look as follows:

```

../framework/XCVR/MKW39A4
../framework/XCVR/MKW39A4/nb2p4ghz/configs
../framework/XCVR/MKW39A4/nb2p4ghz
    
```

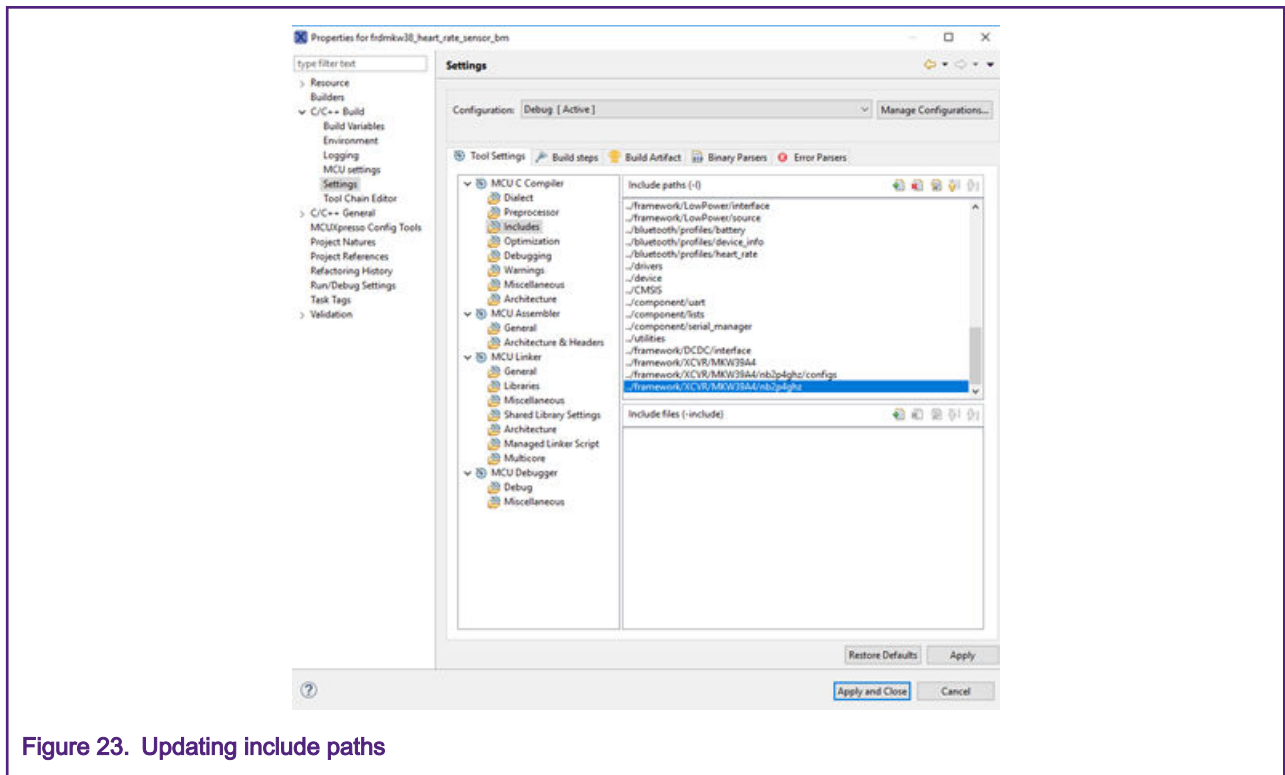


Figure 23. Updating include paths

10. Select the *MCU Assembler/General* folder in "Tool Settings". Edit the paths related to the MKW38 MCU. The results must look as follows:

```

../framework/XCVR/MKW39A4
../framework/XCVR/MKW39A4/nb2p4ghz/configs
../framework/XCVR/MKW39A4/nb2p4ghz
    
```

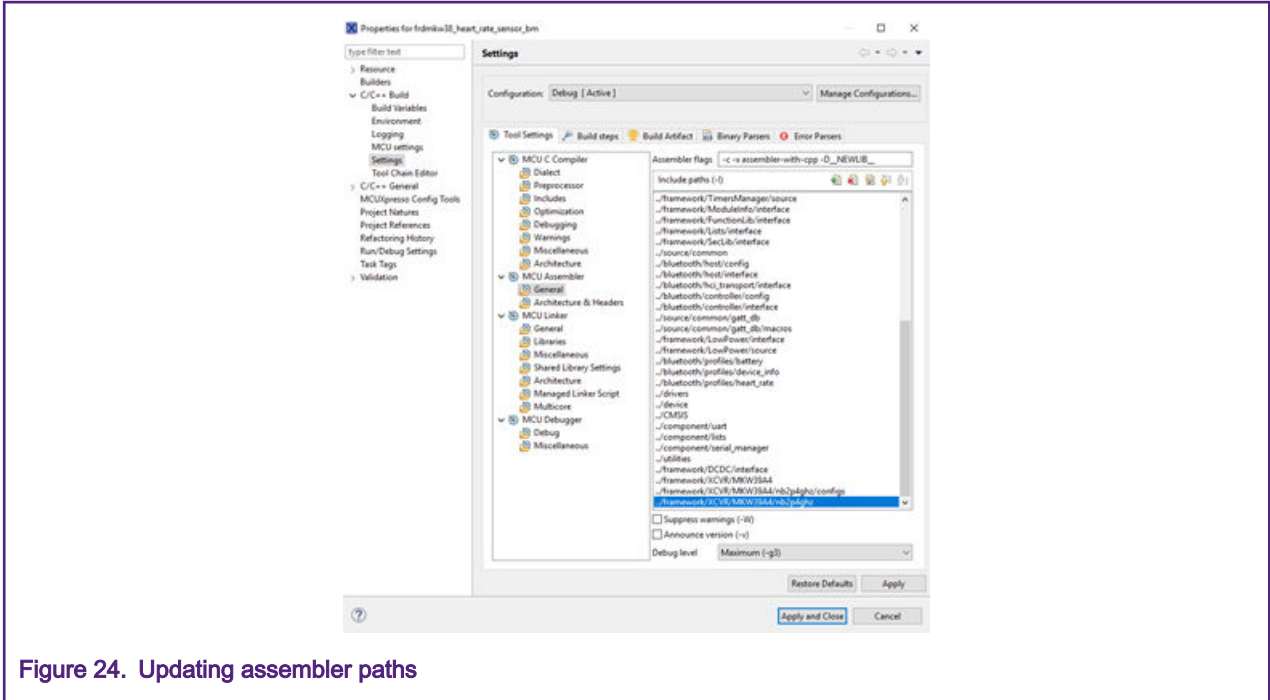


Figure 24. Updating assembler paths

- Open the "MCU C Compiler/Preprocessor" window and rename the CPU\_MKW38A512VFT4 and CPU\_MKW38A512VFT4\_cm0plus macros to CPU\_MKW39A512VFT4 and CPU\_MKW39A512VFT4\_cm0plus, respectively. Delete the FRDM\_KW38 macro. Click the "Apply and Close" button.

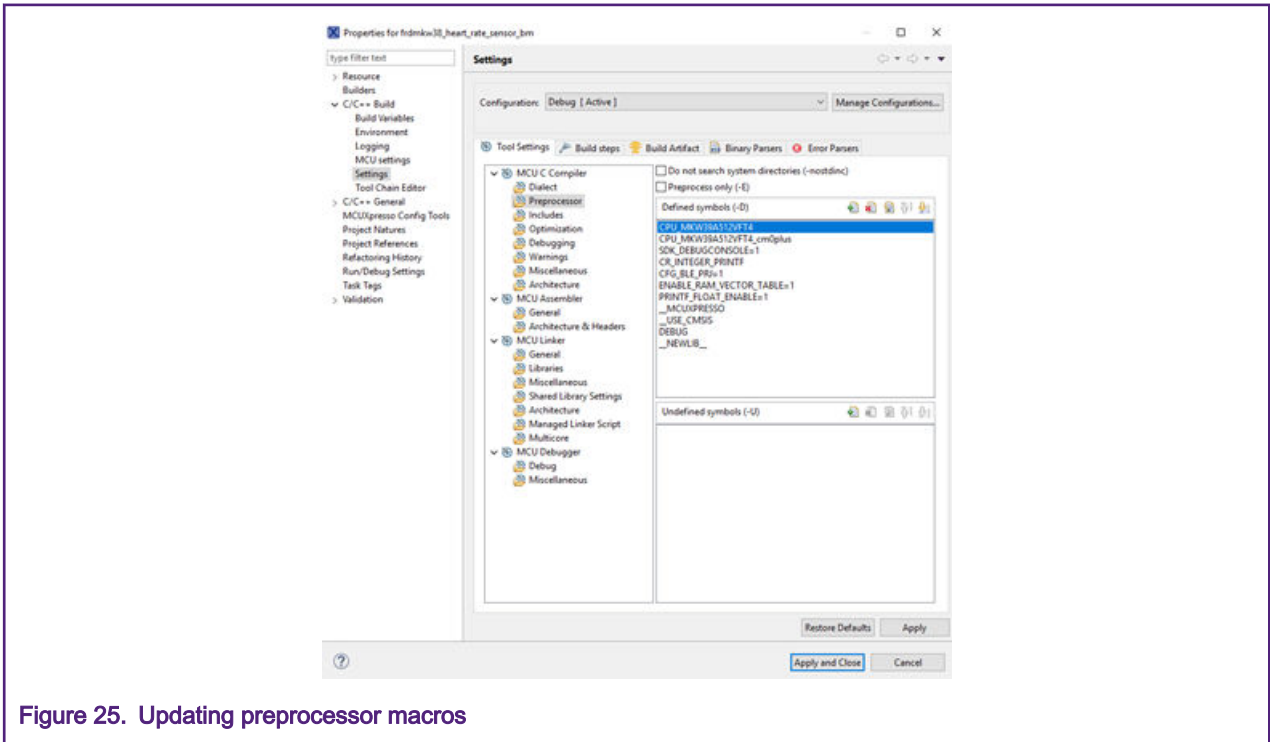


Figure 25. Updating preprocessor macros

- Delete the *fsl\_device\_registers.h*, *MKW38A4.h*, *MKW38A4\_features.h*, *system\_MKW38A4.h*, and *system\_MKW38A4.c* files located in the *device* folder in the workspace.
- Unzip the MKW39A SDK package and search for the *fsl\_device\_registers.h*, *MKW39A4.h*, *MKW39A4\_features.h*, *system\_MKW39A4.h*, and *system\_MKW39A4.c* files in this folder at the following paths:  
`<SDK_folder_root>/devices/MKW39A4/fsl_device_registers.h`

```
<SDK_folder_root>/devices/MKW39A4/MKW39A4.h
<SDK_folder_root>/devices/MKW39A4/MKW39A4_features.h
<SDK_folder_root>/devices/MKW39A4/system_MKW39A4.h
<SDK_folder_root>/devices/MKW39A4/system_MKW39A4.c
```

Drag and drop these files into the *device* folder in the workspace.

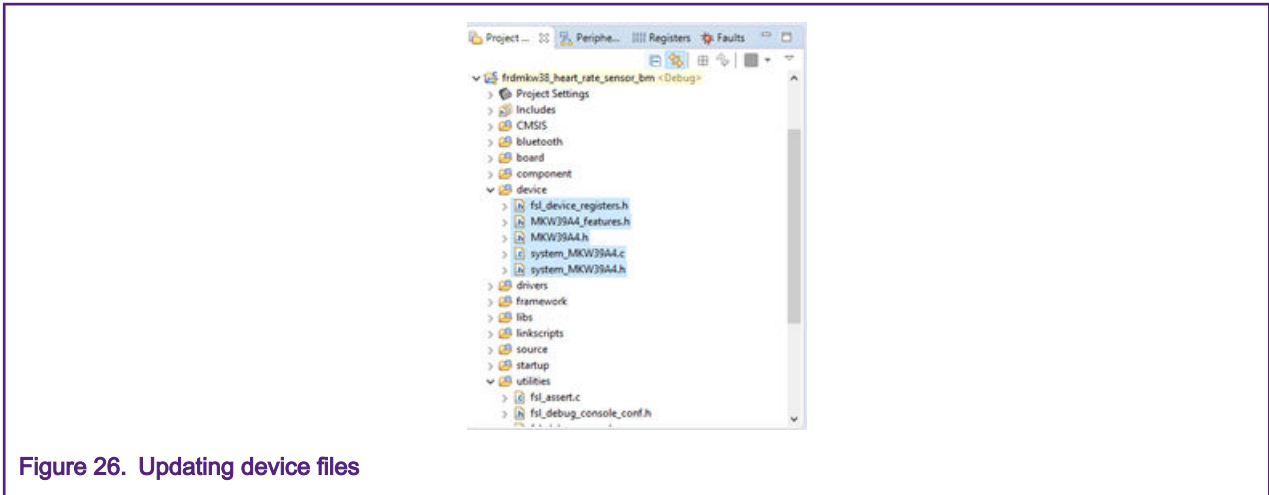


Figure 26. Updating device files

- Replace the *startup\_mkw38a4.c* file located in the *startup* folder in the workspace by the *startup\_mkw39a4.c* file. This file is located in the MKW39A SDK at the following path:

```
<SDK_folder_root>/devices/MKW39A4/mcuxpresso/startup_MKW39A4.c
```

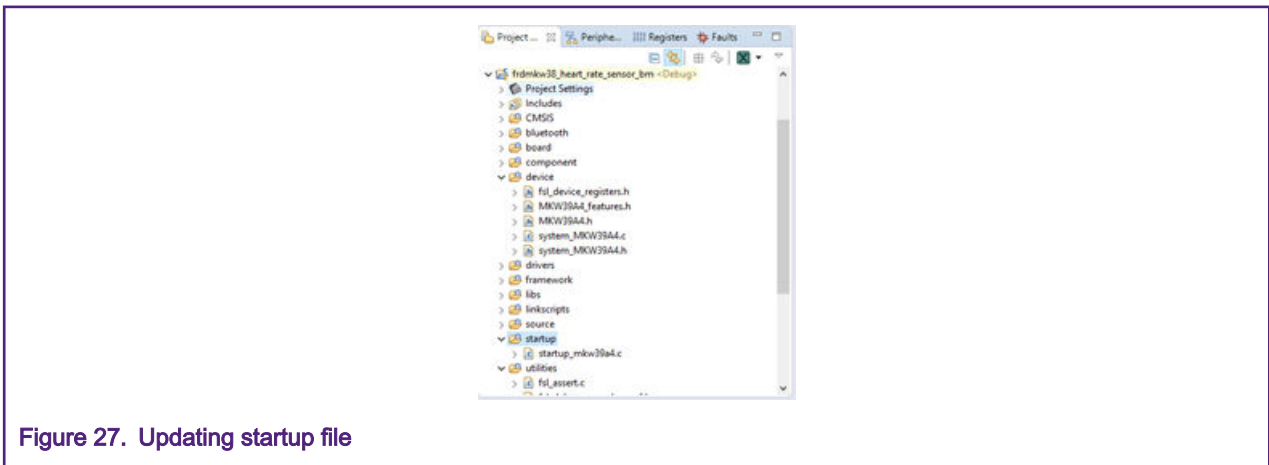


Figure 27. Updating startup file

- Some files must be updated. Open the *clock\_config.c* file in *board* folder. Delete the `CLOCK_SetLpuart1Clock` call to function in the `BOARD_BootClockRUN` function.

```
/* Set LPUART0 clock source. */
CLOCK_SetLpuart0Clock(SIM_LPUART_CLK_SEL_OSCERCLK_CLK);
/* Set LPUART1 clock source. */
//Delete this line: CLOCK_SetLpuart1Clock(SIM_LPUART_CLK_SEL_OSCERCLK_CLK);
/* Set TPM clock source. */
CLOCK_SetTpmClock(SIM_TPM_CLK_SEL_OSCERCLK_CLK);
```

Figure 28. Updating clock\_config.c file

- If necessary, open the *app\_preinclude.h* file under the *source* directory in the workspace. This file contains important information about the board, such as the number of switches and LEDs, timers, power consumption settings, and so on. Examine and update this file to fit into your own custom board.

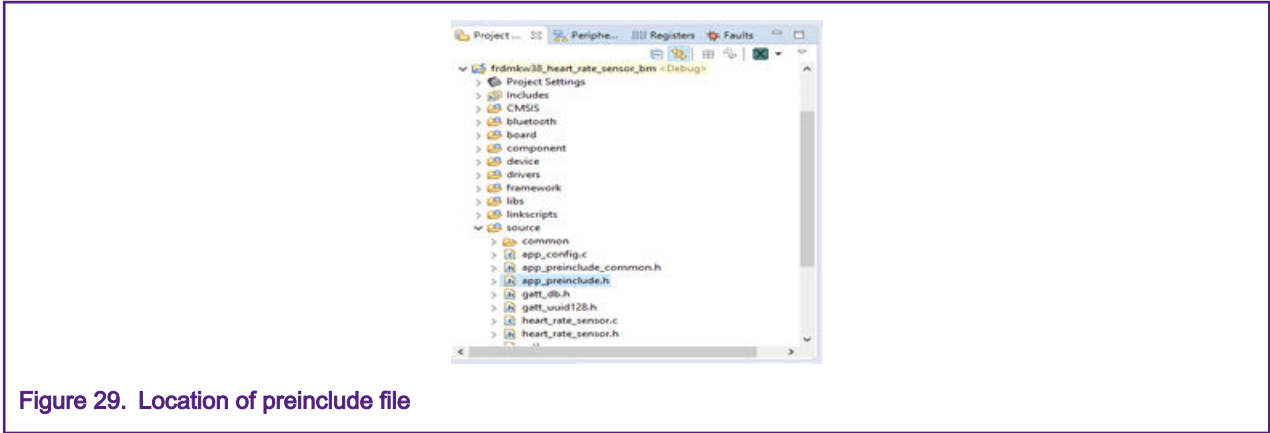


Figure 29. Location of preinclude file

17. If necessary, open the pin mux files (*pin\_mux.c* and *pin\_mux.h*) and the gpio files (*gpio\_pins.c* and *gpio\_pins.h*) in the *board* directory in the workspace. These files contain alternatives, options, and multiplexing information of the pins. Examine and update them to fit into your own custom board.

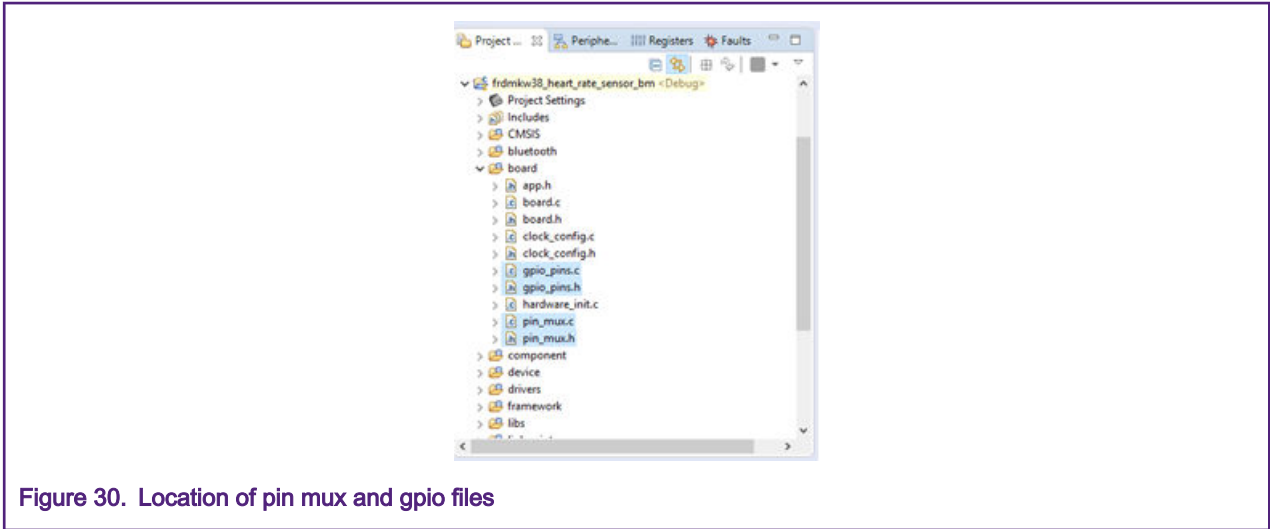


Figure 30. Location of pin mux and gpio files

18. Expand the *drivers* folder, select the *fs\_flexcan.h* and *fs\_flexcan.c* files, right-click, and select the “Delete” option.
19. Build the project. At this point, the project is already migrated and you are ready to run the demo using MKW39.

## 6 Build and run Bluetooth LE connectivity stack examples

All the examples referenced in the *Bluetooth LE Demo Applications User's Guide* are compatible with the MKW39 device after the modifications described in this document.



## How To Reach Us

### Home Page:

[nxp.com](http://nxp.com)

### Web Support:

[nxp.com/support](http://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](http://nxp.com/SalesTermsandConditions).

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

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

© NXP B.V. 2020.

All rights reserved.

For more information, please visit: <http://www.nxp.com>

For sales office addresses, please send an email to: [salesaddresses@nxp.com](mailto:salesaddresses@nxp.com)

Date of release: 04/2020

Document identifier: AN12559

