NXP Semiconductors

User's Guide

Document Number: MCUXIDECTUG

Rev. 4, 28 October 2018

MCUXpresso Config Tools User's Guide (IDE)

Contents

Chapter 1 Introduction

The MCUXpresso Config Tools set is a suite of evaluation and configuration tools that helps you from first evaluation to production software development. It includes the following tools.

Table 1. MCUXpresso Config Tools

Name	Description
Pins Tool	Enables you to configure the pins of a device. Pins Tool enables you to create, inspect, change, and modify any aspect of the pin configuration and muxing of the device.
Clocks Tool	Enables you to configure initialization of the system clock (core, system, bus, and peripheral clocks) and generates the C code with clock initialization functions and configuration structures.
Peripherals Tool	Enable you to configure the intilization for the MCUXpresso SDK drivers.

1.1 Versions

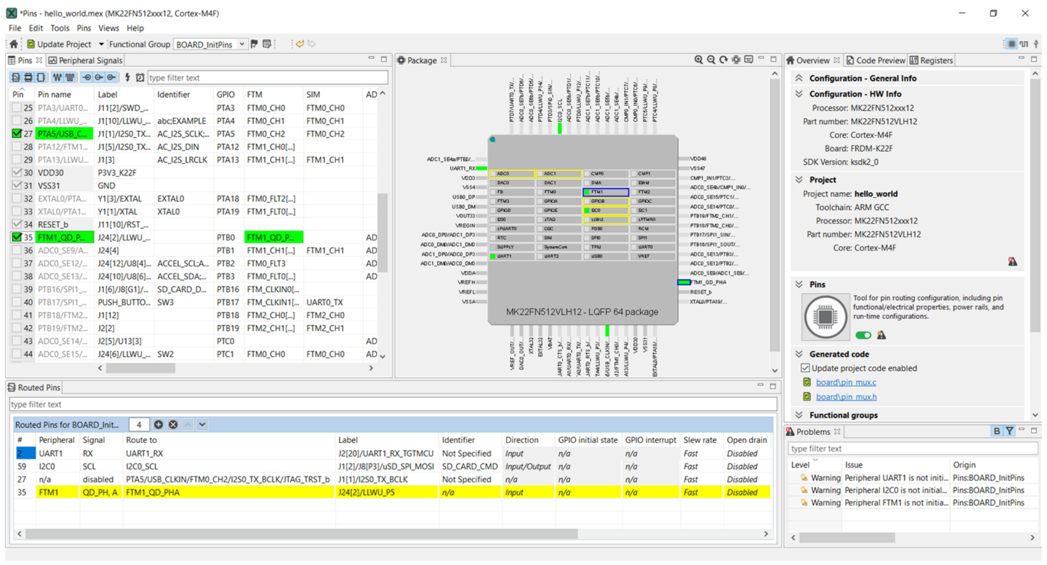
The suite of these tools is called MCUXpresso Config Tools. These tools are provided as an online Web application or as a desktop application or as integrated version in MCUXpresso IDE.

NOTE

The desktop version of the tool contacts the NXP server and fetches the list of the available processors. Once used, the processors data is retrieved on demand.

TIP

To use the desktop tool in the offline mode, create a configuration for the given processor while online. The tool will then store the processors locally in the user folder and enable faster access and offline use. Otherwise, it is possible to download and export the data using the **Export** menu.



Pin	Pin name	Label	Identifier	GPIO	FTM	SIM	AD
25	PTA3/UART0...	J11[2]/SWD_...		PTA3	FTM0_CH0	FTM0_CH0	
26	PTA4/LLWU_...	J1[10]/LLWU_...	abc;EXAMPLE	PTA4	FTM0_CH1	FTM0_CH1	
✓ 27	PTA5/USB_C...	J1[1]/I2S0_TX...	AC_I2S_SCLK;...	PTA5	FTM0_CH2	FTM0_CH2	
28	PTA12/FTM1...	J1[5]/I2S0_TX...	AC_I2S_DIN	PTA12	FTM1_CH0[...]		
29	PTA13/LLWU...	J1[3]	AC_I2S_LRCLK	PTA13	FTM1_CH1[...]	FTM1_CH1	
✓ 30	VDD30	P3V3_K22F					
✓ 31	VSS31	GND					
32	EXTAL0/PTA...	Y1[3]/EXTAL	EXTAL0	PTA18	FTM0_FLT2[...]		
33	XTAL0/PTA1...	Y1[1]/XTAL	XTAL0	PTA19	FTM1_FLT0[...]		
✓ 34	RESET_b	J11[10]/RST_...					
✓ 35	FTM1_QD_P...	J24[2]/LLWU_...		PTB0	FTM1_QD_P...		AD
36	ADC0_SE9/A...	J24[4]		PTB1	FTM1_CH1[...]	FTM1_CH1	AD
37	ADC0_SE12/...	J24[12]/U8[4]...	ACCEL_SCL;A...	PTB2	FTM0_FLT3		AD
38	ADC0_SE13/...	J24[10]/U8[6]...	ACCEL_SDA;...	PTB3	FTM0_FLT0[...]		AD
39	PTB16/SPI1_...	J1[6]/J8[G1]/...	SD_CARD_D...	PTB16	FTM_CLKIN0[...		
40	PTB17/SPI1_...	PUSH_BUTTO...	SW3	PTB17	FTM_CLKIN1[...	UART0_TX	
41	PTB18/FTM2...	J1[12]		PTB18	FTM2_CH0[...]	FTM2_CH0	
42	PTB19/FTM2...	J2[2]		PTB19	FTM2_CH1[...]	FTM2_CH1	
43	ADC0_SE14/...	J2[5]/U13[3]		PTC0			AD
44	ADC0_SE15/...	J24[6]/LLWU_...	SW2	PTC1	FTM0_CH0	FTM0_CH0	AD

Routed Pins for BOARD_Init... 4

#	Peripheral	Signal	Route to	Label	Identifier	Direction	GPIO initial state	GPIO interrupt	Slew rate	Open drain
2	UART1	RX	UART1_RX	J2[20]/UART1_RX_TGTMCU	Not Specified	*Input*	*n/a*	*n/a*	*Fast*	*Disabled*
59	I2C0	SCL	I2C0_SCL	J1[2]/J8[P3]/uSD_SPI_MOSI	SD_CARD_CMD	*Input/Output*	*n/a*	*n/a*	*Fast*	*Disabled*
27	n/a	disabled	PTA5/USB_CLKIN/FTM0_CH2/I2S0_TX_BCLK/JTAG_TRST_b	J1[1]/I2S0_TX_BCLK	Not Specified	*n/a*	*n/a*	*n/a*	*Fast*	*Disabled*
35	FTM1	QD_PH, A	FTM1_QD_PHA	J24[2]/LLWU_P5	*n/a*	*Input*	*n/a*	*n/a*	*Fast*	*Disabled*

Configuration - General Info

Configuration - HW Info

Processor: MK22FN512xxx12
Part number: MK22FN512VLH12
Core: Cortex-M4F
Board: FRDM-K22F
SDK Version: ksdk2_0

Project

Project name: **hello_world**
Toolchain: ARM GCC
Processor: MK22FN512xxx12
Part number: MK22FN512VLH12
Core: Cortex-M4F

Pins

Tool for pin routing configuration, including pin functional/electrical properties, power rails, and run-time configurations.

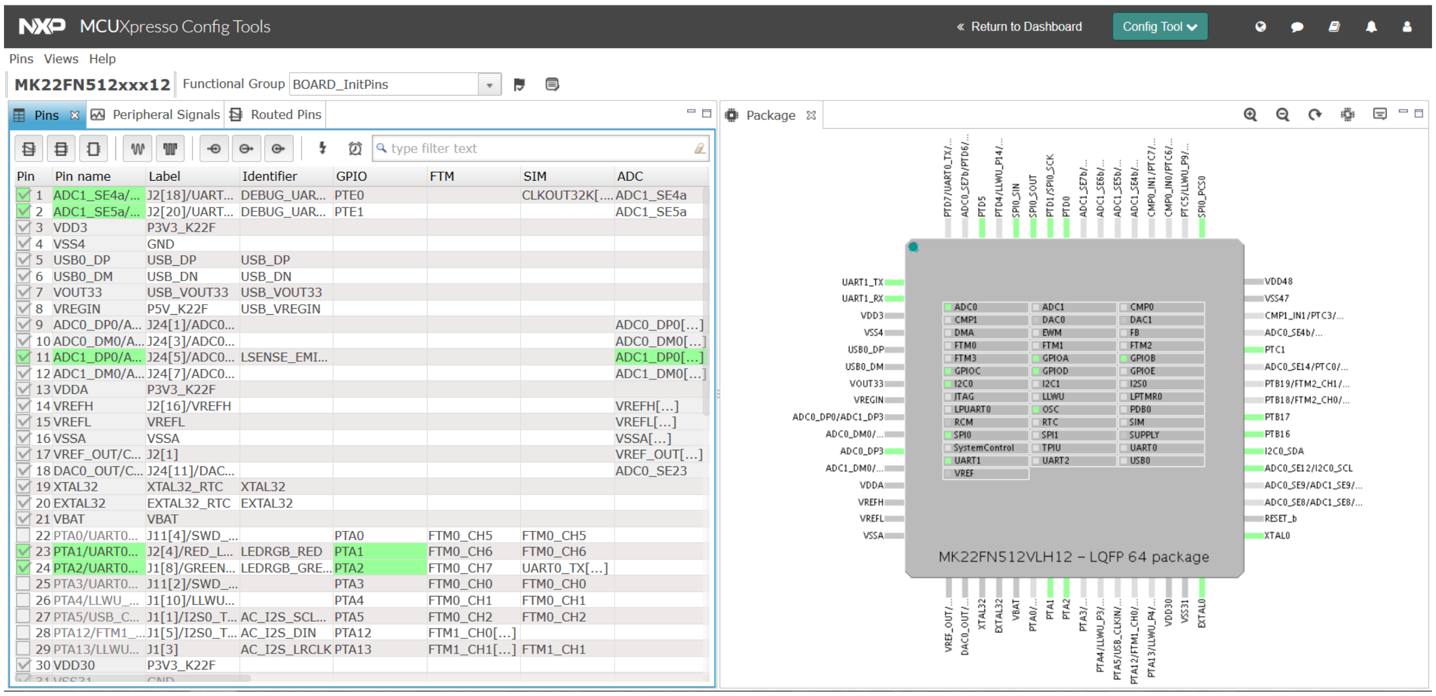
Generated code

☑ Update project code enabled
board\pin_mux.c
board\pin_mux.h

Functional groups

Level	Issue	Origin
Warning	Peripheral UART1 is not initi...	Pins:BOARD_InitPins
Warning	Peripheral I2C0 is not initial...	Pins:BOARD_InitPins
Warning	Peripheral FTM1 is not initia...	Pins:BOARD_InitPins

Figure 1. Desktop version of Pins Tool



MK22FN512xxx12 Functional Group BOARD_InitPins

Pins | Peripheral Signals | Routed Pins

Pin	Pin name	Label	Identifier	GPIO	FTM	SIM	ADC
✓ 1	ADC1_SE4a/...	J2[18]/UART...	DEBUG_UAR...	PTE0		CLKOUT32K[....	ADC1_SE4a
✓ 2	ADC1_SE5a/...	J2[20]/UART...	DEBUG_UAR...	PTE1			ADC1_SE5a
✓ 3	VDD3	P3V3_K22F					
✓ 4	VSS4	GND					
✓ 5	USB0_DP	USB_DP	USB_DP				
✓ 6	USB0_DM	USB_DN	USB_DN				
✓ 7	VOUT33	USB_VOUT33	USB_VOUT33				
✓ 8	VREGIN	P5V_K22F	USB_VREGIN				
✓ 9	ADC0_DP0/A...	J24[1]/ADC0...					ADC0_DP0[...]
✓ 10	ADC0_DM0/A...	J24[3]/ADC0...					ADC0_DM0[...]
✓ 11	ADC1_DP0/A...	J24[5]/ADC0...	LSENSE_EMI...				ADC1_DP0[...]
✓ 12	ADC1_DM0/A...	J24[7]/ADC0...					ADC1_DM0[...]
✓ 13	VDDA	P3V3_K22F					
✓ 14	VREFH	J2[16]/VREFH					VREFH[...]
✓ 15	VREFL	VREFL					VREFL[...]
✓ 16	VSSA	VSSA					VSSA[...]
✓ 17	VREF_OUT/C...	J2[1]					VREF_OUT[...]
✓ 18	DAC0_OUT/C...	J24[11]/DAC...					ADC0_SE23
✓ 19	XTAL32	XTAL32_RTC	XTAL32				
✓ 20	EXTAL32	EXTAL32_RTC	EXTAL32				
✓ 21	VBAT	VBAT					
22	PTA0/UART0...	J11[4]/SWD_...		PTA0	FTM0_CH5	FTM0_CH5	
✓ 23	PTA1/UART0...	J2[4]/RED_L...	LEDRGB_RED	PTA1	FTM0_CH6	FTM0_CH6	
✓ 24	PTA2/UART0...	J1[8]/GREEN...	LEDRGB_GRE...	PTA2	FTM0_CH7	UART0_TX[...]	
25	PTA3/UART0...	J11[2]/SWD_...		PTA3	FTM0_CH0	FTM0_CH0	
26	PTA4/LLWU_...	J1[10]/LLWU...		PTA4	FTM0_CH1	FTM0_CH1	
27	PTA5/USB_C...	J1[1]/I2S0_T...	AC_I2S_SCL...	PTA5	FTM0_CH2	FTM0_CH2	
28	PTA12/FTM1_...	J1[5]/I2S0_T...	AC_I2S_DIN	PTA12	FTM1_CH0[...]		
29	PTA13/LLWU...	J1[3]	AC_I2S_LRCLK	PTA13	FTM1_CH1[...]	FTM1_CH1	
✓ 30	VDD30	P3V3_K22F					

Figure 2. Web version of Pins Tool

Chapter 2
Config Tools User Interface

2.1 Creating, saving, and opening a configuration

In this context, configuration stands for common tools settings stored in an MEX (Microcontrollers Export Configuration) file. This file contains settings of all available tools and can be used in both web and desktop versions.

2.1.1 Creating a new configuration

In **Project Explorer** right-click the Eclipse project based on MCUXpresso SDK, and select **MCUXpresso Config Tool > Open Pins**. One of the following actions takes place:

- If the project contains an MEX file in the root folder, the file is opened.
- If the project contains any source file with tool configuration (pin_mux.c, clock_config.c and/or peripheral.c), the tool configuration is imported from this file.
- Otherwise, an empty/default configuration for selected processor is created.

NOTE

The same command can be invoked also from popup menu on the MEX file or from toolbar in **Project Explorer** view.

2.1.2 Saving a configuration

You can save your configuration by clicking the **Save** button on the toolbar or selecting **File>Save** from the **Main Menu**. The command is enabled only if the configuration is dirty (unsaved) and one of MCUXpresso Config Tool perspective is opened. The configuration is always saved into an MEX file stored in the project root folder. If file doesn't exist, new one is created using current project name.

NOTE

Configuration is also automatically saved during **Update Project Code** action.

2.1.3 Importing sources

To import source code files:

1. Select **File** > **Import...** from the **Main Menu**.
2. Select **MCUXpresso Config Tools>Import Source**.

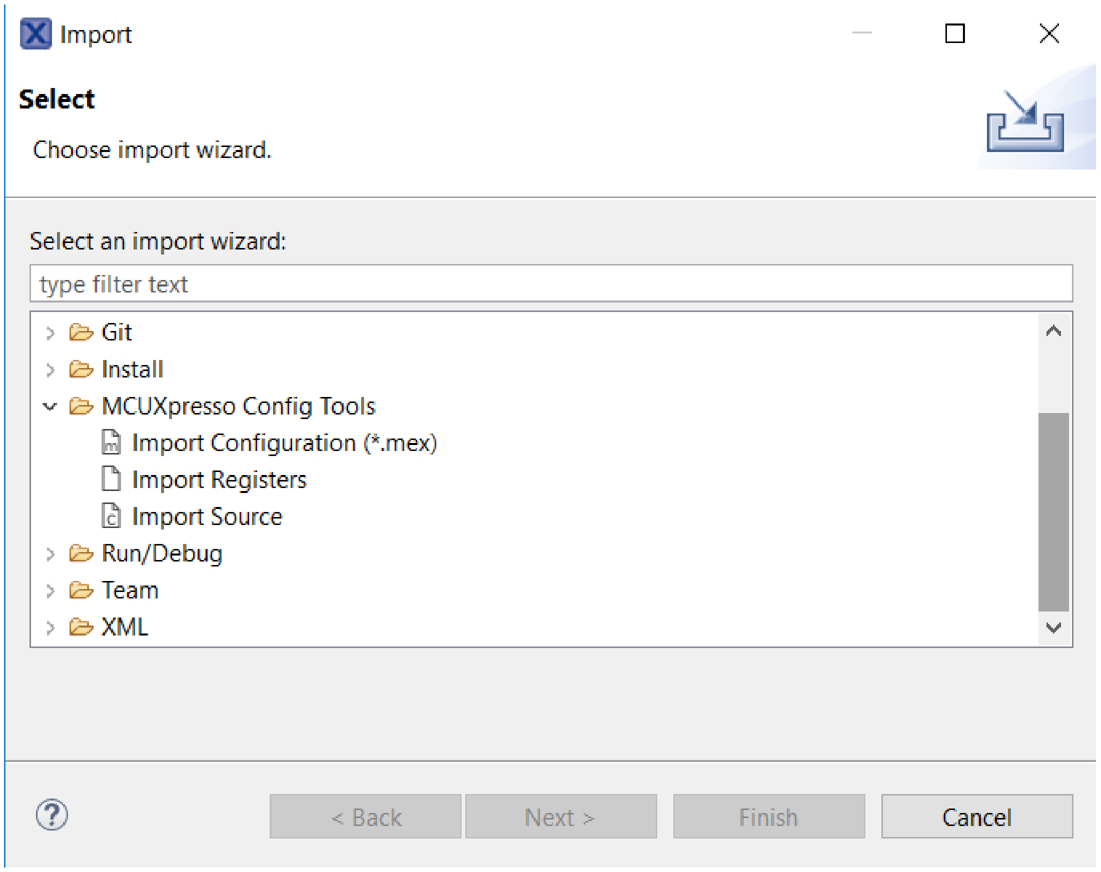


Figure 3. Import Source wizard

3. Click **Next**.
4. You can select one or more C files to import using the **Browse** button in the **Import Pins Source Files** dialog.
5. Select how to import the files:
   - **Rename** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as as an existing one, it is automatically renamed to the indexed one. For example, if BOARD_InitPins already exists in the configuration then the imported function is renamed to BOARD_InitPins1.
   - **Overwrite** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as as an existing one, then the existing one is replaced with the imported one.
6. Click **Finish**.

NOTE

Only C files with valid YAML configuration can be imported. It imports the configuration only, then the whole C file is re-created based on this setting. The rest of the *.c and *.dtsi files are ignored.

2.1.3.1 Importing configuration

2.1.3.2 Importing registers

You can import register configuration from a processor memory dump.

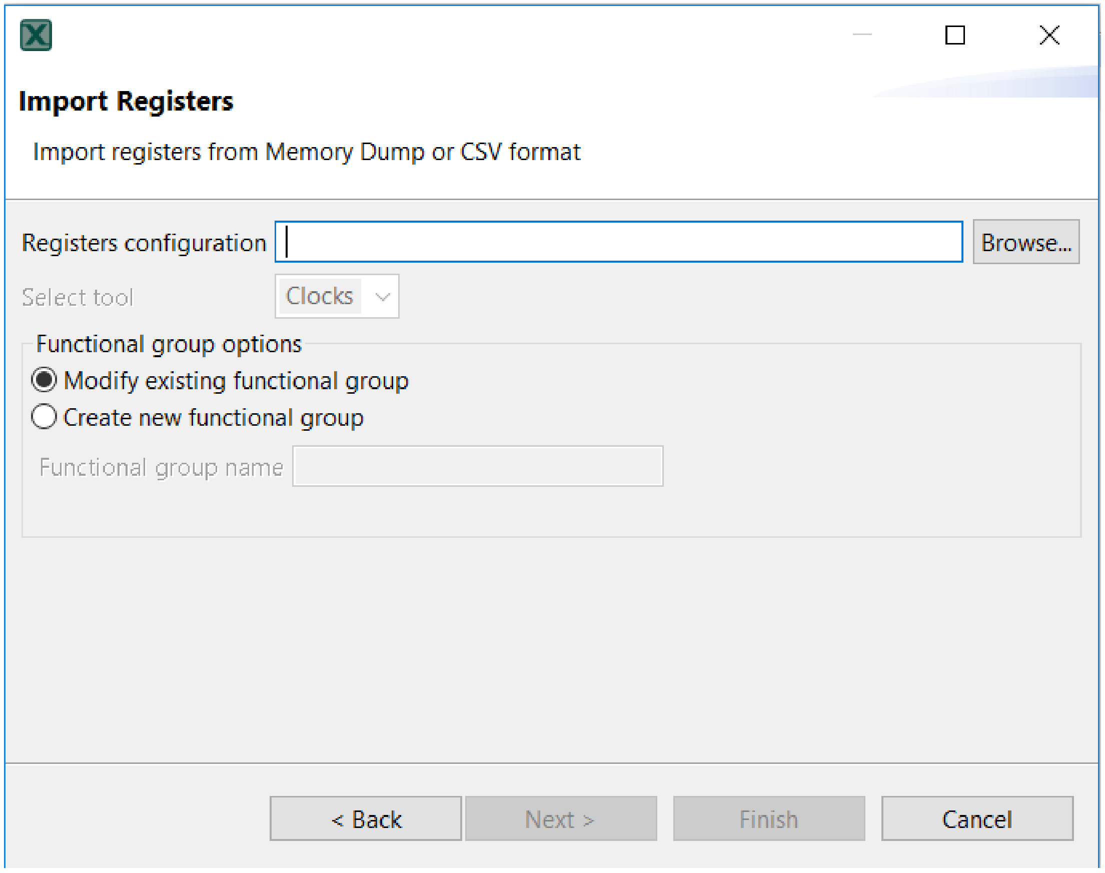
NOTE

Currently, register configuration can be imported into the Clocks Tool only.

NOTE

A processor memory-dump file in the CSV or S19 format is required for importing register configuration.

Figure 4. Import Registers



Import Registers

Import registers from Memory Dump or CSV format

Registers configuration:

Select tool: Clocks

Functional group options

[✓ Modify existing functional group]

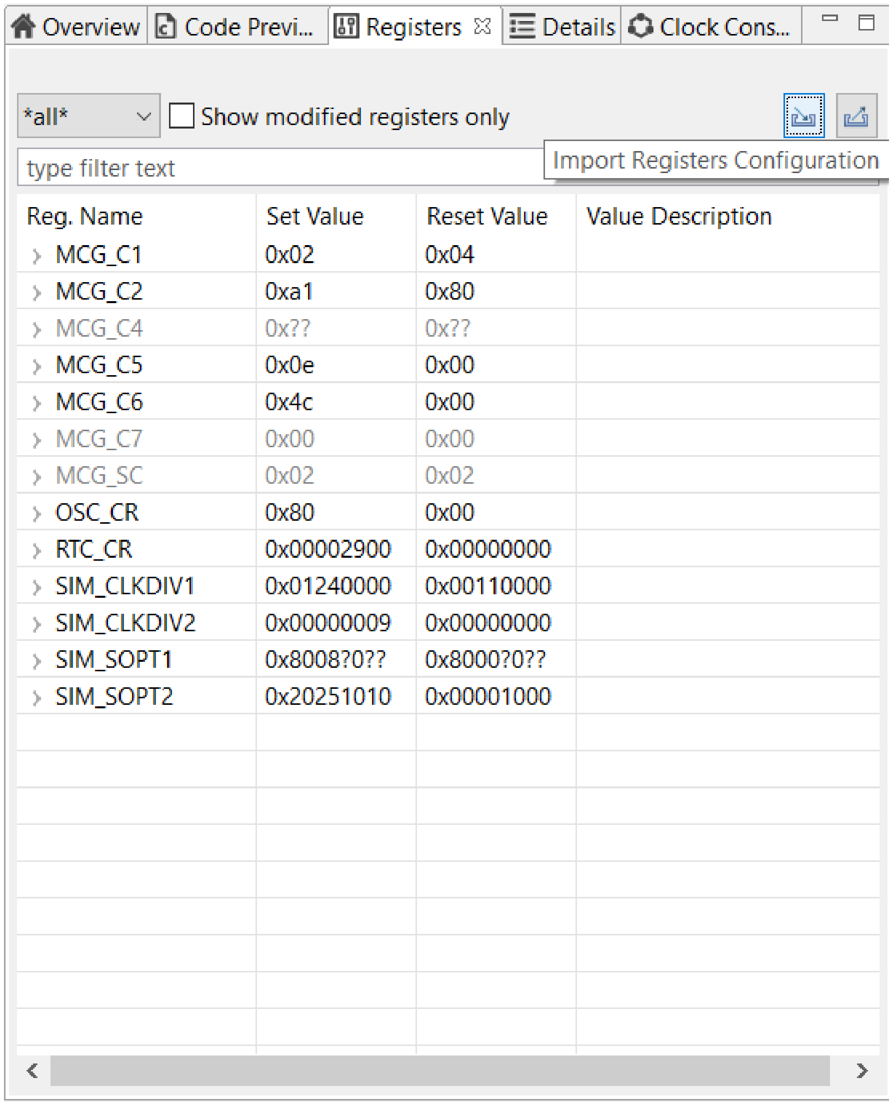
Create new functional group

Functional group name:

To import register configuration, do the following:

1. Select **File>Import…>Import Registers** from the **Main Menu**. Alternatively, click the **Import Registers Configuration** button in the **Registers** view, or drag-and-drop the memory dump file anywhere in the **Registers** view.

Figure 5. Import Registers Configuration



Reg. Name	Set Value	Reset Value	Value Description
MCG_C1	0x02	0x04	
MCG_C2	0xa1	0x80	
MCG_C4	0x??	0x??	
MCG_C5	0x0e	0x00	
MCG_C6	0x4c	0x00	
MCG_C7	0x00	0x00	
MCG_SC	0x02	0x02	
OSC_CR	0x80	0x00	
RTC_CR	0x00002900	0x00000000	
SIM_CLKDIV1	0x01240000	0x00110000	
SIM_CLKDIV2	0x00000009	0x00000000	
SIM_SOPT1	0x8008?0??	0x8000?0??	
SIM_SOPT2	0x20251010	0x00001000	

2. In the **Import Registers** wizard, specify the location of the registers configuration. If you want a new functional group to be created, select the option, and specify the functional group name.
3. Click **Finish**.

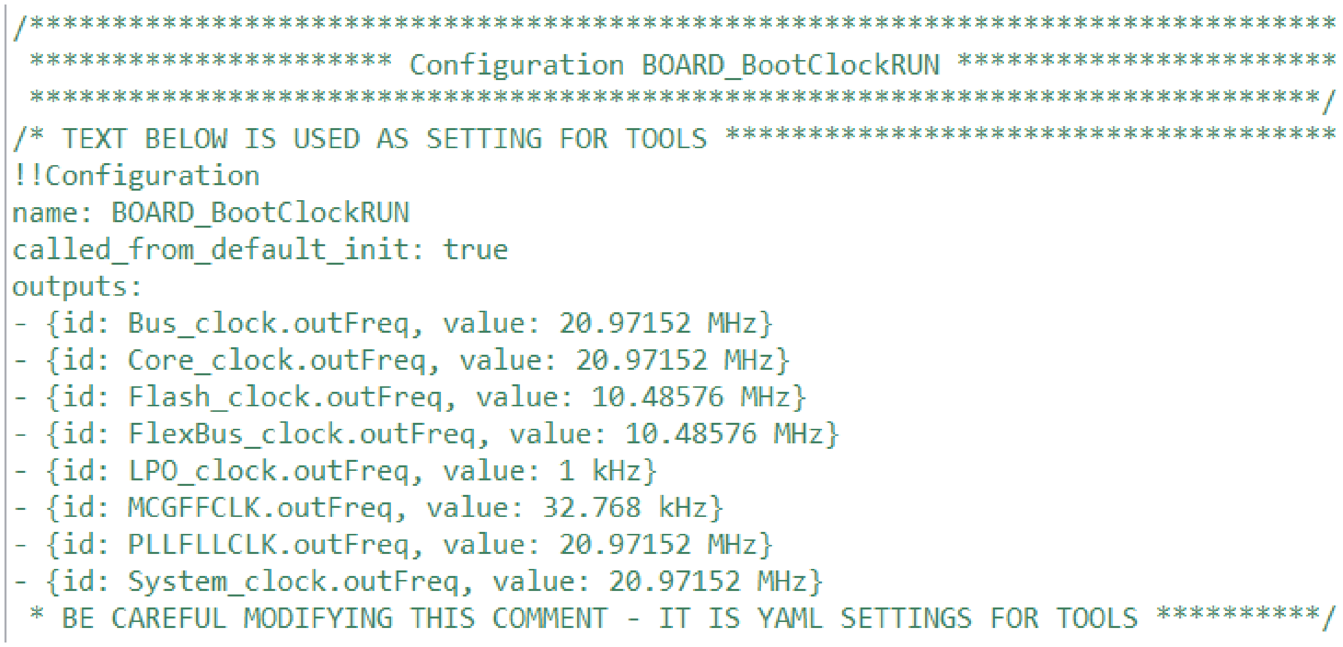
NOTE

All registers are imported from the dump file regardless of their relevance to clock configuration, therefore, the list can contain registers not needed by the Clocks Tool.

2.1.4 Restoring configuration from source code

The generated code contains information about the Clocks Tool settings that are used in the tool (block within a comment in YAML format).

The following is an example of the settings information in the generated source code.



```
/*******************************************************************************
 ********************** Configuration BOARD_BootClockRUN ***********************
 ******************************************************************************/
/* TEXT BELOW IS USED AS SETTING FOR TOOLS *************************************
!!Configuration
name: BOARD_BootClockRUN
called_from_default_init: true
outputs:
- {id: Bus_clock.outFreq, value: 20.97152 MHz}
- {id: Core_clock.outFreq, value: 20.97152 MHz}
- {id: Flash_clock.outFreq, value: 10.48576 MHz}
- {id: FlexBus_clock.outFreq, value: 10.48576 MHz}
- {id: LPO_clock.outFreq, value: 1 kHz}
- {id: MCGFFCLK.outFreq, value: 32.768 kHz}
- {id: PLLFLLCLK.outFreq, value: 20.97152 MHz}
- {id: System_clock.outFreq, value: 20.97152 MHz}
 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS **********/
```

Figure 6. Setting Information in the source code

If this information is not corrupted, it's possible to re-import the clock settings into the tool using the following steps.

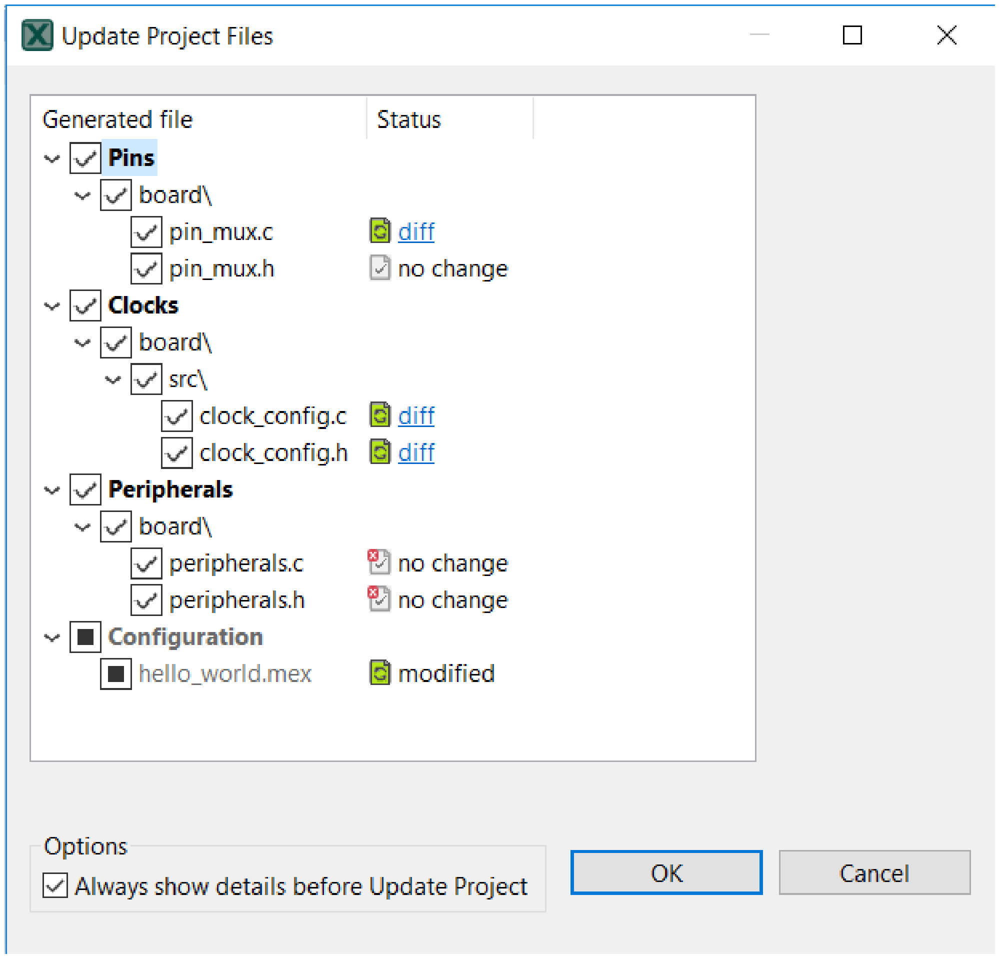
1. Select **File** > **Import…**.
2. Select **Clocks Tool / Import Source Files**.
3. Click **Next**.
4. Click **Browse**.
5. Navigate and select the *clock_config.c* file previously produced by the Clocks Tool.
6. If the settings parse successfully, clock configurations are added into the current global configuration.

2.2 Toolbar

The toolbar is located on the top of the window and includes frequently used actions.

2.2.1 Update project

To update the generated code in the related toolchain project, click the **Update Project** button. In the dialog, select the tools or files you want to update. If update is not possible, the button is filled with a black square. The reason is displayed in the tooltip.



Update Project Files

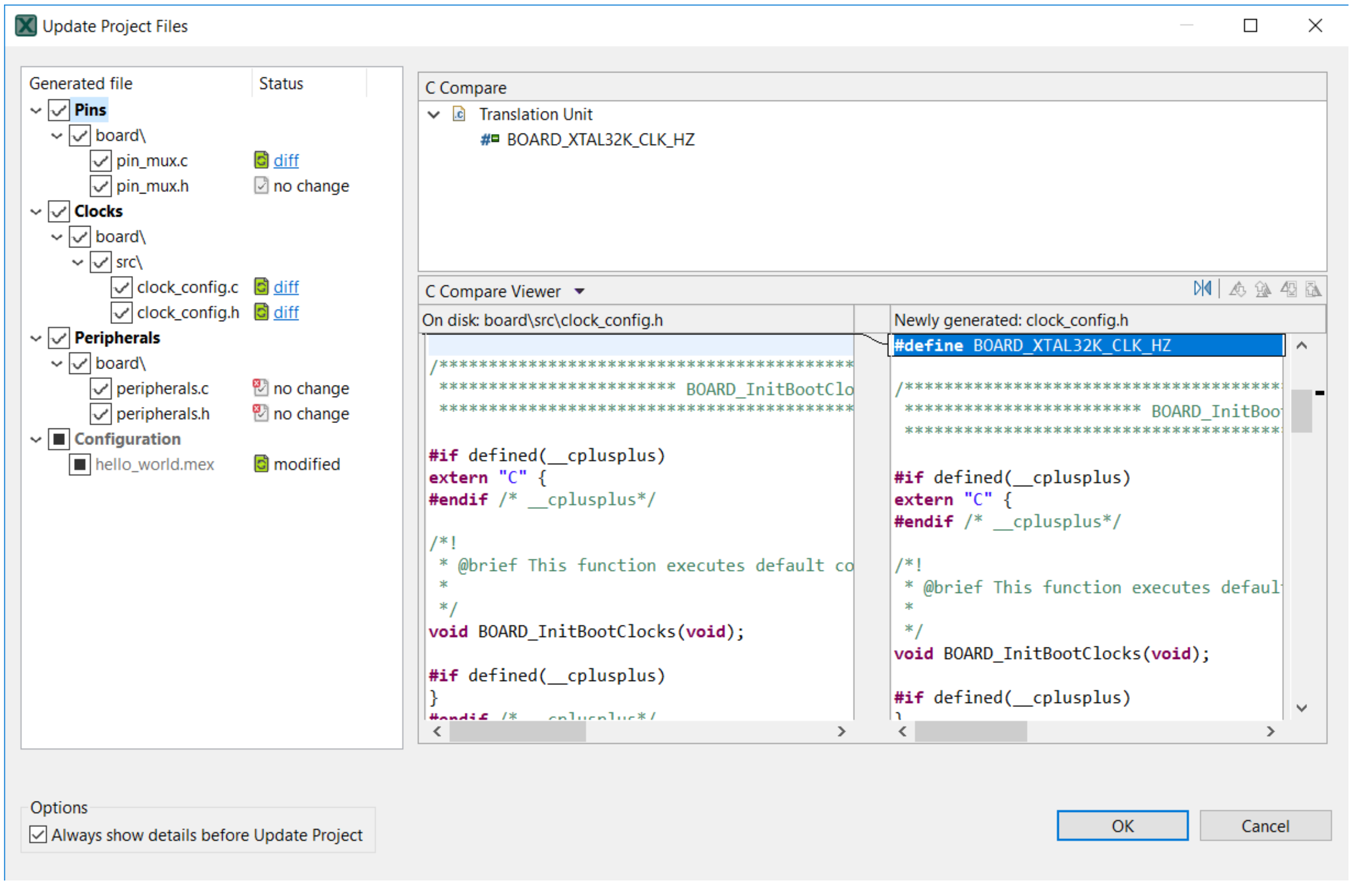
Generated file	Status
✓ **Pins**	
✓ board\	
✓ pin_mux.c	diff
✓ pin_mux.h	no change
✓ **Clocks**	
✓ board\	
✓ src\	
✓ clock_config.c	diff
✓ clock_config.h	diff
✓ **Peripherals**	
✓ board\	
✓ peripherals.c	no change
✓ peripherals.h	no change
■ **Configuration**	
■ hello_world.mex	modified

Options

✓ Always show details before Update Project

Figure 7. Update Project Files dialog

To inspect the code difference between the versions, click the **show differences** link.



Update Project Files

Generated file	Status
Pins	
board\	
pin_mux.c	diff
pin_mux.h	no change
Clocks	
board\	
src\	
clock_config.c	diff
clock_config.h	diff
Peripherals	
board\	
peripherals.c	no change
peripherals.h	no change
Configuration	
hello_world.mex	modified

C Compare
Translation Unit
BOARD_XTAL32K_CLK_HZ

C Compare Viewer

On disk: board\src\clock_config.h

```
/*******************************************
 ************************ BOARD_InitBootClo
 *******************************************

#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus*/

/*!
 * @brief This function executes default co
 *
 */
void BOARD_InitBootClocks(void);

#if defined(__cplusplus)
}
```

Newly generated: clock_config.h

```
#define BOARD_XTAL32K_CLK_HZ

/*******************************************
 ************************ BOARD_InitBoo
 *******************************************

#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus*/

/*!
 * @brief This function executes defaul
 *
 */
void BOARD_InitBootClocks(void);

#if defined(__cplusplus)
```

Options
☑ Always show details before Update Project

OK Cancel

Figure 8. Show differences

To update the project without opening the **Update Project Files** dialog, clear the **Always show details before Update Project** option.

To access the the **Update Project Files** dialog from the **Update Project** drop-down menu, select **Open Update Project Dialog**.

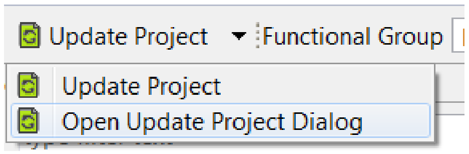


Figure 9. Update Project drop-down menu

NOTE
The generated code is always overwritten.

NOTE
Previous version of the file can be retrieved from Eclipse local history.

The **Update Project** action is enabled under following conditions:

- Processor selected in the tool matches with processor selected in the toolchain project
- Core is selected (for multicore processors)

2.2.2 Eclipse project selection

You can use the **Eclipse project** drop-down menu to switch between projects.

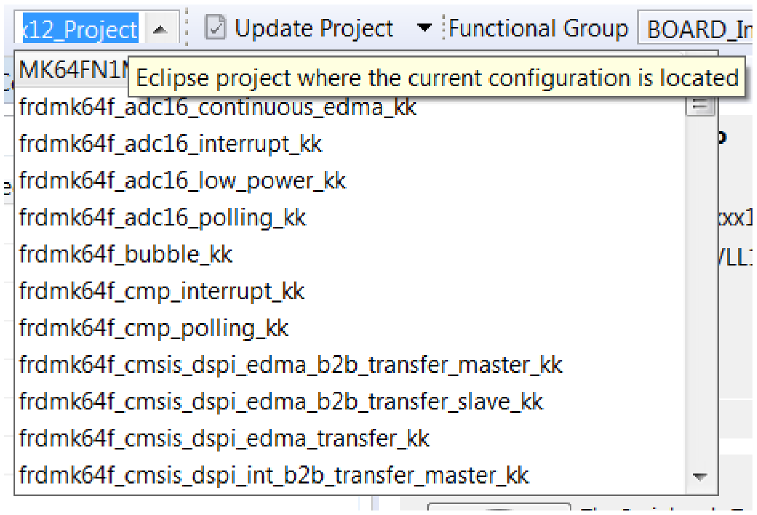

k12_Project
Update Project
Functional Group
BOARD_In
MK64FN1M
Eclipse project where the current configuration is located
frdmk64f_adc16_continuous_edma_kk
frdmk64f_adc16_interrupt_kk
frdmk64f_adc16_low_power_kk
frdmk64f_adc16_polling_kk
frdmk64f_bubble_kk
frdmk64f_cmp_interrupt_kk
frdmk64f_cmp_polling_kk
frdmk64f_cmsis_dspi_edma_b2b_transfer_master_kk
frdmk64f_cmsis_dspi_edma_b2b_transfer_slave_kk
frdmk64f_cmsis_dspi_edma_transfer_kk
frdmk64f_cmsis_dspi_int_b2b_transfer_master_kk

Figure 10. Eclipse project selection

2.2.3 Functional groups

Each configuration can contain several functional groups. These groups represent functions which will be generated into source code. Use the drop-down menu to switch between functional groups and configure them.

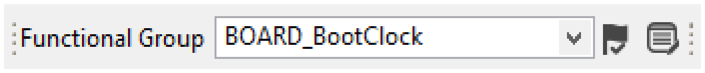


Figure 11. Functional groups

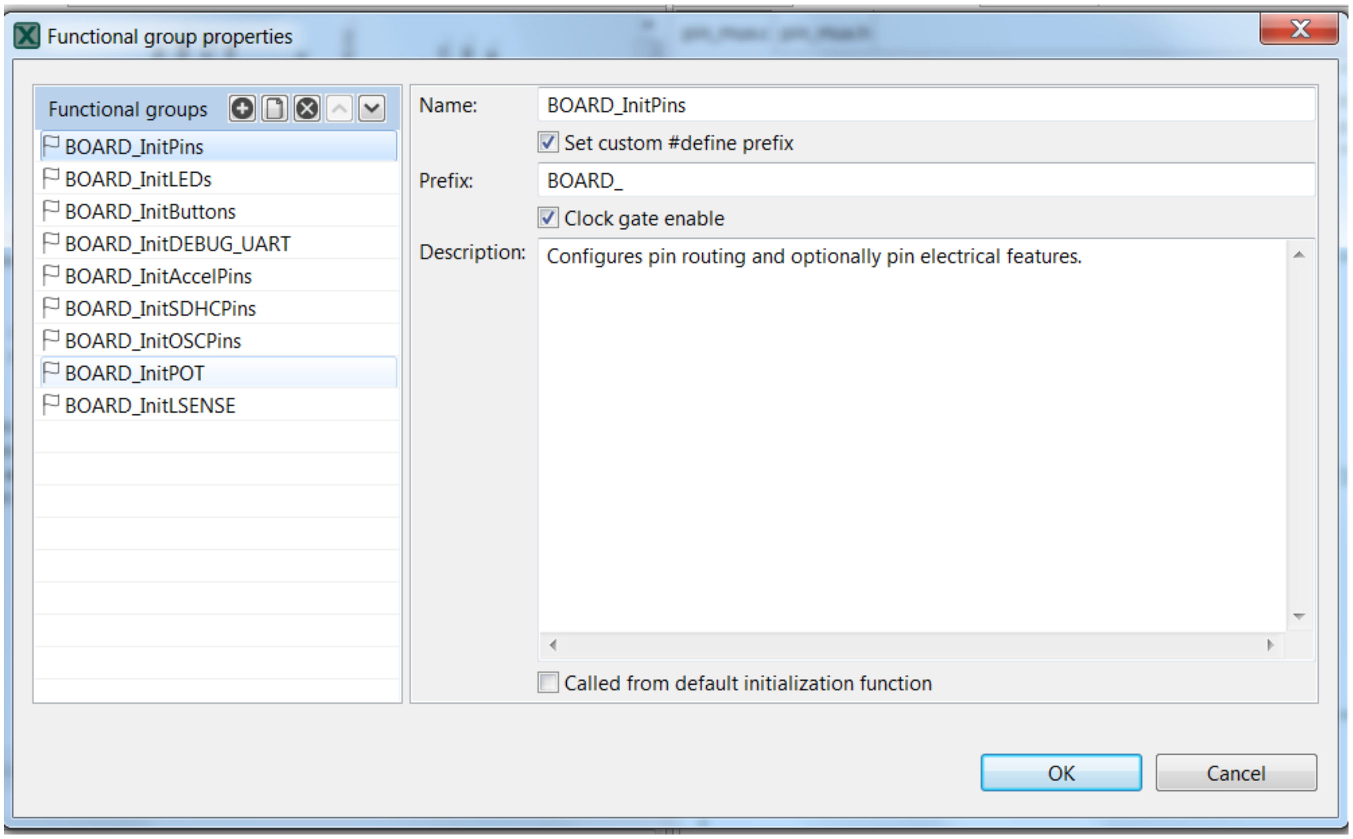
Additional buttons can be used on functional groups:

– Toggle "Called from default initialization function" feature (in source code)

– Open the **Functional group properties** dialog

Red/orange background indicates errors/warnings in the configuration.

2.2.3.1 Functional group properties



BOARD_InitPins
BOARD_InitLEDs
BOARD_InitButtons
BOARD_InitDEBUG_UART
BOARD_InitAccelPins
BOARD_InitSDHCPins
BOARD_InitOSCPins
BOARD_InitPOT
BOARD_InitLSENSE

Name: BOARD_InitPins
[✓] Set custom #define prefix
Prefix: BOARD_
[✓] Clock gate enable
Description: Configures pin routing and optionally pin electrical features.
[ ] Called from default initialization function

Figure 12. Functional group properties for Pins Tool

In this dialog, you can configure several options for functions and code generation. Each settings is applicable for the selected function. you can specify generated function name, select core (for multicore processors only) that is affecting the generated source code, or write function description (this description will be generated in the C file).

Set custom #define prefix: If enabled, it uses the specified prefix for the identifiers in the source code. You can also modify functions order (on the left), the order is applied in the generated code.[1]

Configure the **Called from default initialization function** option to set it for the function. If the option is set, the function is called from the default initialization function.

2.2.4 Switching the tools

Buttons on the right side of the toolbar represent available tools. Click them to quickly navigate between Clocks, Pins and Peripherals tools.

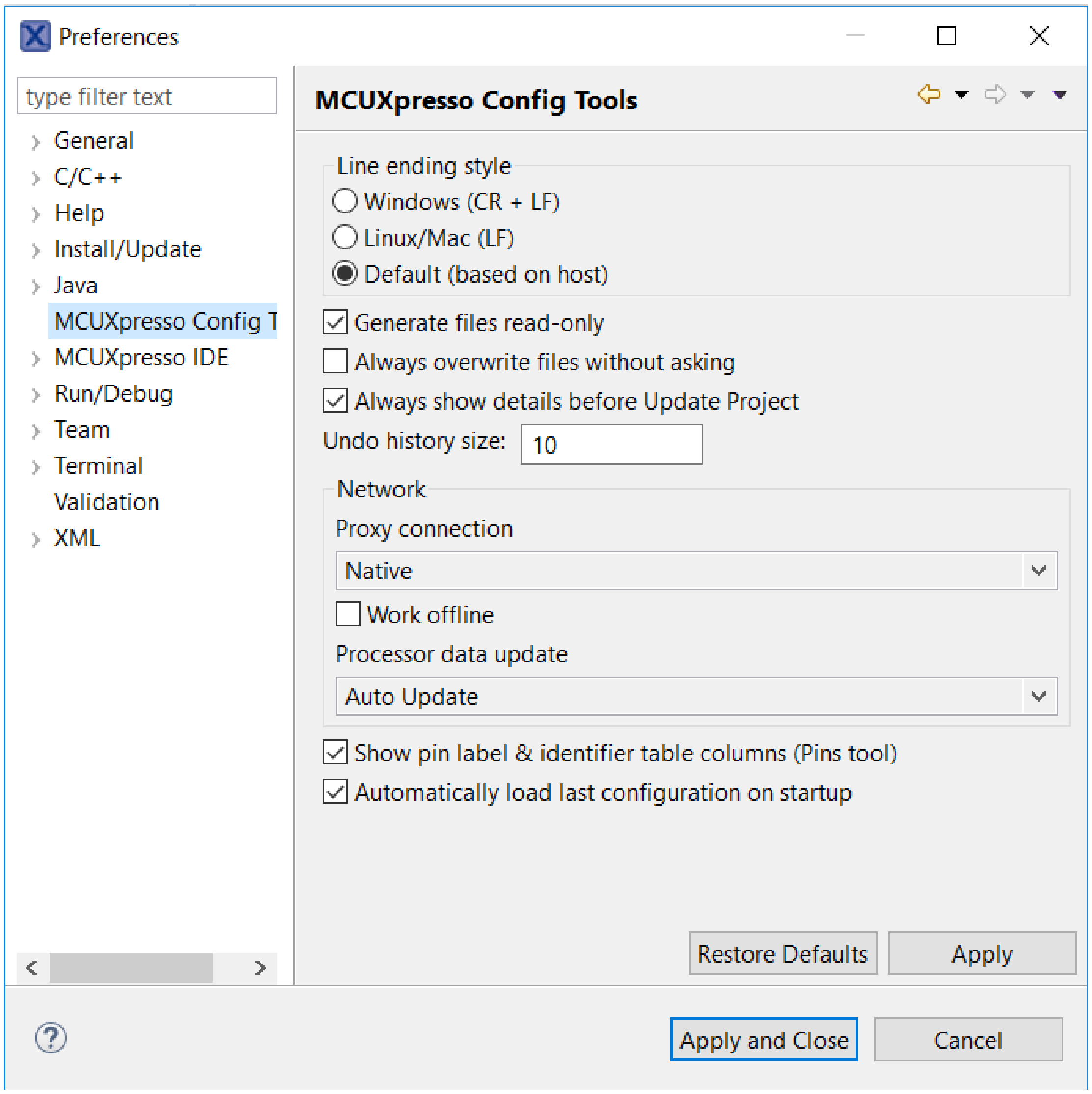
2.3 Status bar

The status bar is visible at the bottom part of the GUI. Status bar indicates error and warning state of the currently selected functional group.

[1] *if supported by processor

2.4 Preferences

To configure preferences, select **Window>Preferences>MCUXpresso Config Tools** from the **Main Menu**. The **Preferences** dialog appears.



MCUXpresso Config Tools

Line ending style

- ○ Windows (CR + LF)
- ○ Linux/Mac (LF)
- ● Default (based on host)

- [✓] Generate files read-only
- [ ] Always overwrite files without asking
- [✓] Always show details before Update Project

Undo history size: 10

Network

Proxy connection: Native

- [ ] Work offline

Processor data update: Auto Update

- [✓] Show pin label & identifier table columns (Pins tool)
- [✓] Automatically load last configuration on startup

Figure 13. Preferences dialog

In this dialog you can set the following:

- **Line ending style** – Select between **Windows (CR + LF)**, **Linux/Mac (LF)**, or **Default (based on host)**.
- **Generate files read-only** – Prevents modifying the source files unintentionally. Generated source files are marked as read-only.
- **Always overwrite files without asking** – Select to update existing files automatically, without prompting.
- **Always show details before Update Project** – Select to review changes before the project is updated.
- **Undo history size** – Enter the number of steps you want to undo. Enter 0 to disable.
- Proxy connection

- **Direct** – Select to connect directly and avoid a proxy connection.
- **Native** – Select to use system proxy configuration for network connection.

- **Work offline** – Select to disable both the connection to NXP cloud and the download of processor/board/kit data.
- **Processor data update** – Select from the following options:
  - **Auto Update** – Select to update the processor data automatically.
  - **Manual** – Select to be update processor data after confirmation.
  - **Disabled** – Select to disable processor data update.
- **Show pin label & identifier table columns (Pins tool)** – Select to show the pin label and the label identifier in the relevant views.
- **Automatically load last configuration on startup** – Select to avoid the startup dialog and load the last used configuration instead.

2.5 Configuration preferences

The configuration preferences are general preferences stored within the configuration storage file (MEX).

To configure the preferences related to the configuration, uses popup menu on the Eclipse project, select **Properties** and then **MCUXpresso Config Tools** in the left pane.

The following preferences are available:

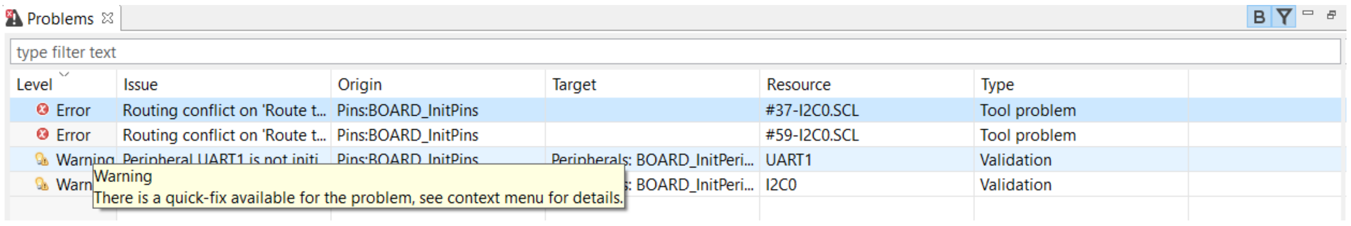
- **Validate boot init only** – Select to validate tools dependencies only against 'boot init' function group.When selected, dependencies from all functional groups of all tools must be satisfied in the functional groups marked for default initialization. Clearing this option hides warnings in case the user is using complex scenarios with alternating functional groups within the application code.
- **Generate YAML** – Select to generate YAML into C sources files.
- **Generate extended information into header file** – Select to generate extended information into the header file. For projects created in earlier MCUXpresso versions, this option is selected by default.

WARNING

When the source does not contain YAML code, it can't be imported.

2.6 Problems view

This view displays problems in the tools and the inter-dependencies between the tools.



Level	Issue	Origin	Target	Resource	Type
Error	Routing conflict on 'Route t...	Pins:BOARD_InitPins		#37-I2C0.SCL	Tool problem
Error	Routing conflict on 'Route t...	Pins:BOARD_InitPins		#59-I2C0.SCL	Tool problem
Warning	Peripheral UART1 is not initi	Pins:BOARD_InitPins	Peripherals: BOARD_InitPeri...	UART1	Validation
Warn			: BOARD_InitPeri...	I2C0	Validation

Warning
There is a quick-fix available for the problem, see context menu for details.

Figure 14. Problems view

To open the **Problems** view, select **Views > Problems**.

The table contains the following information:

- **Level** – Lists the severity of the problem: Information, Warning, or Error.
- **Issue** – Description of the problem.
- **Origin** – Information on the dependency source.
- **Target** – Lists the tool that handled the dependency and where it should be fulfilled.
- **Resource** – Lists the resource which is related to the problem,. For example, the signal name, the clock signal, and so on.
- **Type** – The type of the problem. It is either the validation that is checking dependencies between the tools, or the tool problem that describes problem related just to one tool.

Context-menu

Every problem comes with a context menu accessible by right-clicking the problem row. Use this menu to access information about the problem or to apply a quick fix where applicable.

NOTE
Quick fix is only available for problems highlighted with the "lightbulb" icon.

Filter buttons

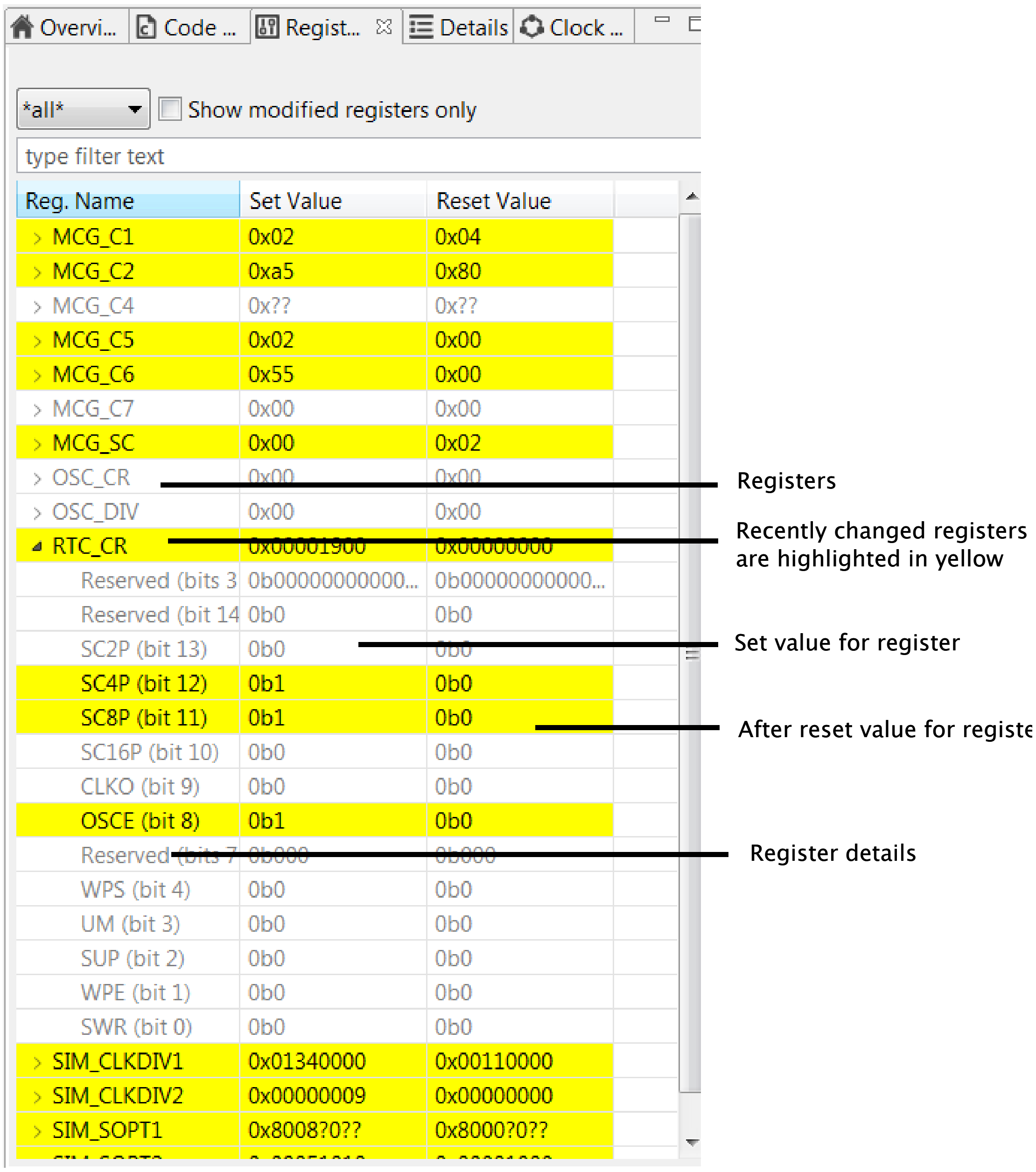
The filter buttons are available on the right side of the problems view.

- **B**

  – Enables the 'Validate boot init only' preference. See Configuration preferences section for details.
- – Filters messages in the **Problems** view. If selected, only problems for the active tool are displayed. See Configuration preferences section for details.

2.7 Registers view

The **Registers** view lists the registers handled by the tool models. You can see the state of the processor registers that correspond to the current configuration settings and also the state that is in the registers by default after the reset. The values of the registers are displayed in the hexadecimal and binary form. If the value of the register (or bit) is not defined, an interrogation mark "?" is displayed instead of the value.



Reg. Name	Set Value	Reset Value
MCG_C1	0x02	0x04
MCG_C2	0xa5	0x80
MCG_C4	0x??	0x??
MCG_C5	0x02	0x00
MCG_C6	0x55	0x00
MCG_C7	0x00	0x00
MCG_SC	0x00	0x02
OSC_CR	0x00	0x00
OSC_DIV	0x00	0x00
RTC_CR	0x00001900	0x00000000
Reserved (bits 3	0b00000000000...	0b00000000000...
Reserved (bit 14	0b0	0b0
SC2P (bit 13)	0b0	0b0
SC4P (bit 12)	0b1	0b0
SC8P (bit 11)	0b1	0b0
SC16P (bit 10)	0b0	0b0
CLKO (bit 9)	0b0	0b0
OSCE (bit 8)	0b1	0b0
Reserved (bits 7	0b000	0b000
WPS (bit 4)	0b0	0b0
UM (bit 3)	0b0	0b0
SUP (bit 2)	0b0	0b0
WPE (bit 1)	0b0	0b0
SWR (bit 0)	0b0	0b0
SIM_CLKDIV1	0x01340000	0x00110000
SIM_CLKDIV2	0x00000009	0x00000000
SIM_SOPT1	0x8008?0??	0x8000?0??

Figure 15. Registers view

The **Registers view** contains:

- **Peripheral filter** drop-down list – Use to list the registers only for the selected peripheral. Select “all” to list registers for all the peripherals.
- **Show modified registers only** checkbox – Select this option to hide the registers that are left in their after-reset state or are not configured.
- **Text filter** – Use to filter content by text.

The following table lists the color highlighting styles used in the **Registers** view.

Table 2. Color codes

Color	Description
Yellow background	Indicates that the bit-field has been affected by the last change made in the tool.
Gray text color	Indicates the bit-field is not edited and the value is the after-reset value.
Black text	Indicates the bit-fields that the tool modifies.

NOTE

This view contains registers for the seleted tool. The view uses registers as internal parameters but it might not handle all the register writes needed in the code. The register writes are done inside the SDK functions that are called by the generated code. There might be additional registers accessed in the SDK code during the setup process, and such register writes are not known to the tool and are not displayed in the registers view.

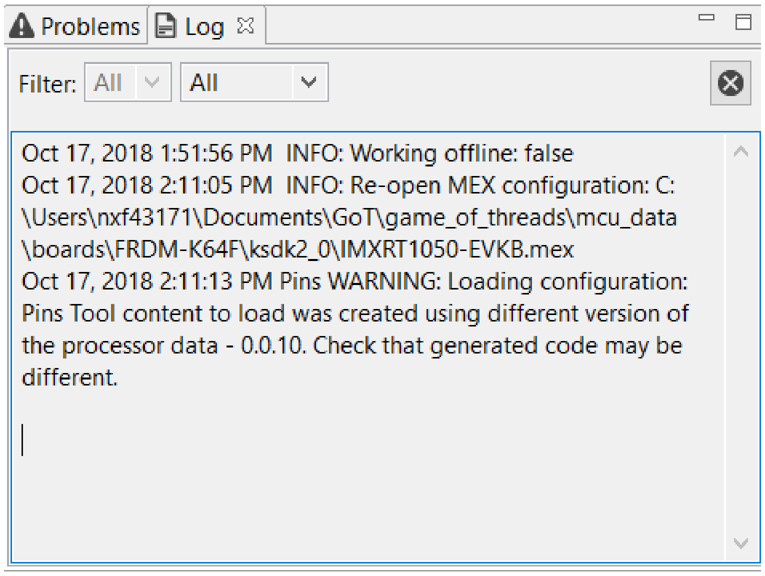
2.8 Log view

The **Log** view shows user-specific information about the progress of the tools. The **Log** view can show up to 100 records across all tools in the chronological order.

Each record consists of the timestamp, the name of the tool responsible for the record, the severity level, and the actual message. If no tool name is specified, the records is created by the shared functionality.

The content of the **Log** view can be filtered using the combo boxes to display only specific tool and/or severity level information. Filters in different tools can be set independently.

Buffered log records are cleared using the clear button. This affects **Log** views of all tools.



```
Oct 17, 2018 1:51:56 PM  INFO: Working offline: false
Oct 17, 2018 2:11:05 PM  INFO: Re-open MEX configuration: C:
\Users\nxf43171\Documents\GoT\game_of_threads\mcu_data
\boards\FRDM-K64F\ksdk2_0\IMXRT1050-EVKB.mex
Oct 17, 2018 2:11:13 PM Pins WARNING: Loading configuration:
Pins Tool content to load was created using different version of
the processor data - 0.0.10. Check that generated code may be
different.
```

Figure 16. Log view

2.9 Config tools overview

By default, the **Config Tools Overview** icon is located on the left side of the toolbar, and opens a dialog with the following options:

- **Configuration – General Info** – Displays the name of and the path to the MEX file of the current configuration. Click the link to open the folder containing the MEX file. To import additional settings, click the **Import additional settings into current configuration** button.
- **Configuration – HW Info** – Displays the processor, part number, core, and SDK-version information of the current configuration.
- **Project** – Displays the toolchain project information.
- **Pins/Clocks/Peripherals** – Displays basic information about the **Pins ,Clocks**, and **Peripherals** tools.

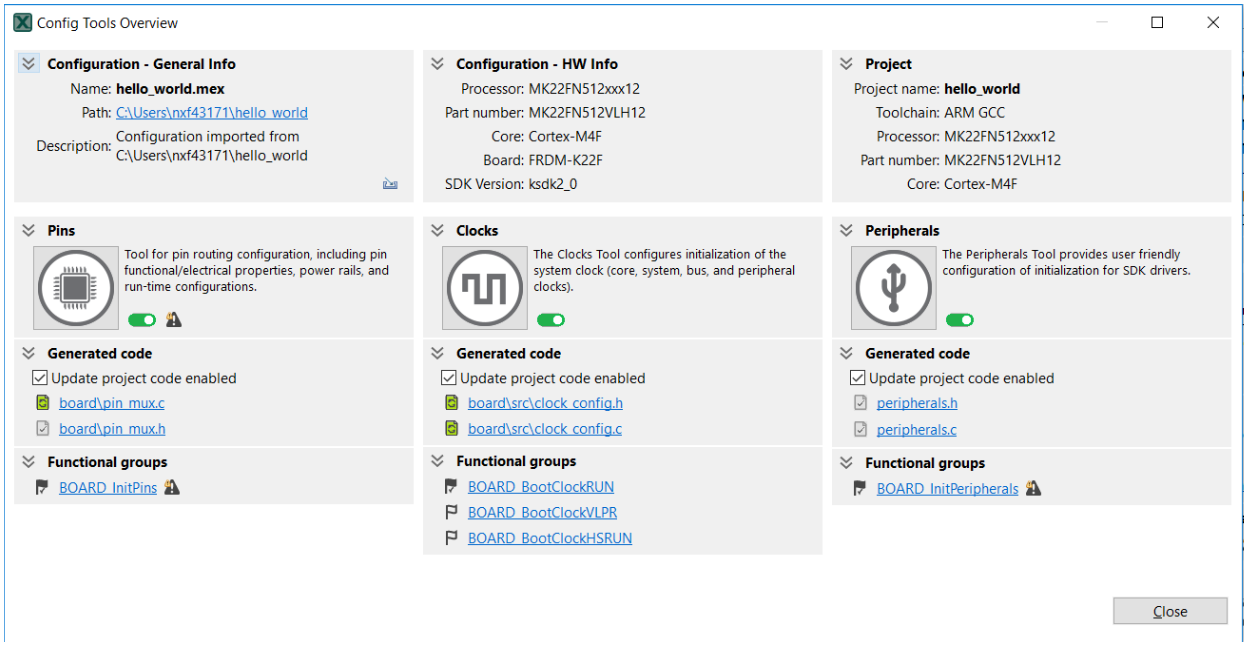
NOTE

If you have disabled a tool and want to reopen it, click the tool icon in the upper right corner or select it from the Main Menu. The **Config Tools Overview** opens automatically.

To enable/disable the tools, click the toggle button. You can navigate to the tools by clicking their icons. The following information about the tools is also available:

— **Generated code** – Contains the list of source-code files. Click the links to open the files in the **Code Preview** view.

— **Functional groups** – Contains the list of the currently active functional groups. To select the groups in the **Functional groups** tab in the toolbar, select the relevant links.



Configuration - General Info
Name: **hello_world.mex**
Path: C:\Users\nxf43171\hello_world
Description: Configuration imported from C:\Users\nxf43171\hello_world

Configuration - HW Info
Processor: MK22FN512xxx12
Part number: MK22FN512VLH12
Core: Cortex-M4F
Board: FRDM-K22F
SDK Version: ksdk2_0

Project
Project name: **hello_world**
Toolchain: ARM GCC
Processor: MK22FN512xxx12
Part number: MK22FN512VLH12
Core: Cortex-M4F

Pins
Tool for pin routing configuration, including pin functional/electrical properties, power rails, and run-time configurations.

Clocks
The Clocks Tool configures initialization of the system clock (core, system, bus, and peripheral clocks).

Peripherals
The Peripherals Tool provides user friendly configuration of initialization for SDK drivers.

Generated code
☑ Update project code enabled
board\pin_mux.c
board\pin_mux.h

Generated code
☑ Update project code enabled
board\src\clock_config.h
board\src\clock_config.c

Generated code
☑ Update project code enabled
peripherals.h
peripherals.c

Functional groups
BOARD_InitPins

Functional groups
BOARD_BootClockRUN
BOARD_BootClockVLPR
BOARD_BootClockHSRUN

Functional groups
BOARD_InitPeripherals

Figure 17. Config Tools Overview dialog

Chapter 3
Pins Tool

The Pins Tool is an easy-to-use tool for configuration of device pins. The Pins Tool software helps create, inspect, change, and modify any element of pin configuration and device muxing.

3.1 Pins routing principle

The Pins Tool is designed to configure routing peripheral signals either to pins or to internal signals.

Internal signal is an interconnection node which peripheral signals can be connected to (without any pin interaction). Connecting two peripheral signals to internal signal makes an interconnection of these two peripheral signals.

This routing configuration can be done in either of these views:

- Pins
- Peripheral Signals
- Package
- Routed Pins

The following two sections describe the two methods you can use to define the routing path.

3.1.1 Beginning with peripheral selection

You can select peripheral in the **Routed Pins** view and the **Peripheral Signals** view.

1. Select the **Peripheral**.
2. In **Routed Pins** view, select one of the available **Signals** or expand the peripheral in **Peripheral Signals** view.
3. Selected the desired pin/internal signal.

   Items (pins/internal signals) in the **Route to** column in the **Routed Pins** view have following decorators:

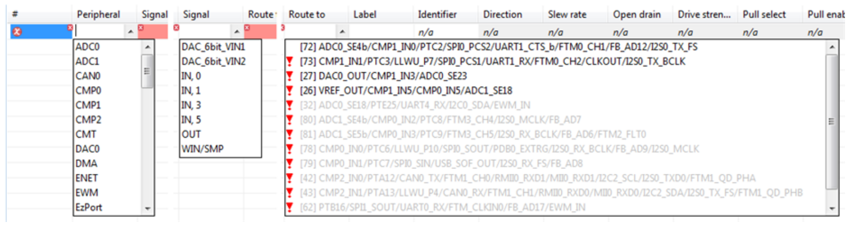
   - Exclamation mark and default text color indicates that such item selection causes a register conflict or the item cannot be routed to the selected peripheral signal (some other peripheral signal can be).
   - Exclamation mark and gray text color indicates that the item cannot be routed to any signal of the selected peripheral. The item is available for different peripheral using the same signal.

NOTE

Route to field in Routed Pins view contains items that are connectable to the selected signal (without its channel if applicable). So when selected signal is “GPIO, 6” then the **Route to** provides items connectable to “GPIO”.

NOTE

In the **Package** view there is no possibility to select pin/internal signal when a peripheral signal is connectable to more pins/internal signals.



#	Peripheral	Signal	Signal	Route	Route to	Label	Identifier	Direction	Slew rate	Open drain	Drive stren...	Pull select	Pull enab
							n/a	n/a	n/a	n/a	n/a	n/a	n/a

ADC0
ADC1
CAN0
CMP0
CMP1
CMP2
CMT
DAC0
DMA
ENET
EWM
EzPort

DAC_6bit_VIN1
DAC_6bit_VIN2
IN, 0
IN, 1
IN, 3
IN, 5
OUT
WIN/SMP

[72] ADC0_SE4b/CMP1_IN0/PTC2/SPI0_PCS2/UART1_CTS_b/FTM0_CH1/FB_AD12/I2S0_TX_FS
[73] CMP1_IN1/PTC3/LLWU_P7/SPI0_PCS1/UART1_RX/FTM0_CH2/CLKOUT/I2S0_TX_BCLK
[27] DAC0_OUT/CMP1_IN3/ADC0_SE23
[26] VREF_OUT/CMP1_IN5/CMP0_IN5/ADC1_SE18
[32] ADC0_SE18/PTE25/UART4_RX/I2C0_SDA/EWM_IN
[80] ADC1_SE4b/CMP0_IN2/PTC8/FTM3_CH4/I2S0_MCLK/FB_AD7
[81] ADC1_SE5b/CMP0_IN3/PTC9/FTM3_CH5/I2S0_RX_BCLK/FB_AD6/FTM2_FLT0
[78] CMP0_IN0/PTC6/LLWU_P10/SPI0_SOUT/PDB0_EXTRG/I2S0_RX_BCLK/FB_AD9/I2S0_MCLK
[79] CMP0_IN1/PTC7/SPI0_SIN/USB_SOF_OUT/I2S0_RX_FS/FB_AD8
[42] CMP2_IN0/PTA12/CAN0_TX/FTM1_CH0/RMII0_RXD1/MII0_RXD1/I2C2_SCL/I2S0_TXD0/FTM1_QD_PHA
[43] CMP2_IN1/PTA13/LLWU_P4/CAN0_RX/FTM1_CH1/RMII0_RXD0/MII0_RXD0/I2C2_SDA/I2S0_TX_FS/FTM1_QD_PHB
[62] PTB16/SPI1_SOUT/UART0_RX/FTM_CLKIN0/FB_AD17/EWM_IN

Figure 18. Defining routing path

3.1.2 Beginning with pin/internal signal selection

You can select a pin or an internal signal in the **Routed Pins** view.

1. Select the pin/internal signal (**Route to**).
2. Select one of the available **Peripherals**. In the **Pins view**, see all available peripherals/signals by clicking on the checkbox in the first column or scroll the columns to the required peripheral type.
3. For the selected peripheral, select one of the available **Signals**.

   Items in **Peripheral** column in Routed Pins view have following symbols:

   - Exclamation mark and default text color indicates that such item selection can cause a register conflict or the item does not support selected signal.
   - Exclamation mark and gray text color indicates that the item cannot be routed to the selected pin/internal signal. The item is available for different pin/internal signal using the same signal.

NOTE
In the **Pins** view and the **Package** view you can configure only pins and not internal signals.

3.2 Workflow

The following steps briefly describe the basic workflow in the Pins Tool.

1. In the **Pins** view on the left find a pin and peripheral signal in the table and configure the routing by clicking on the signal cell.

   NOTE
   This routing configuration can be similarly done in other Pins views **Peripheral Signals**, **Package, Routed Pins**.

2. Optionally, configure the electrical properties in the **Routed pins** view in the middle by selecting required state.

   NOTE
   Source code is automatically generated.

3. Open the **Code Preview** view to inspect the source code.
4. Click the **Update Project** button in the **Toolbar** to update the code.

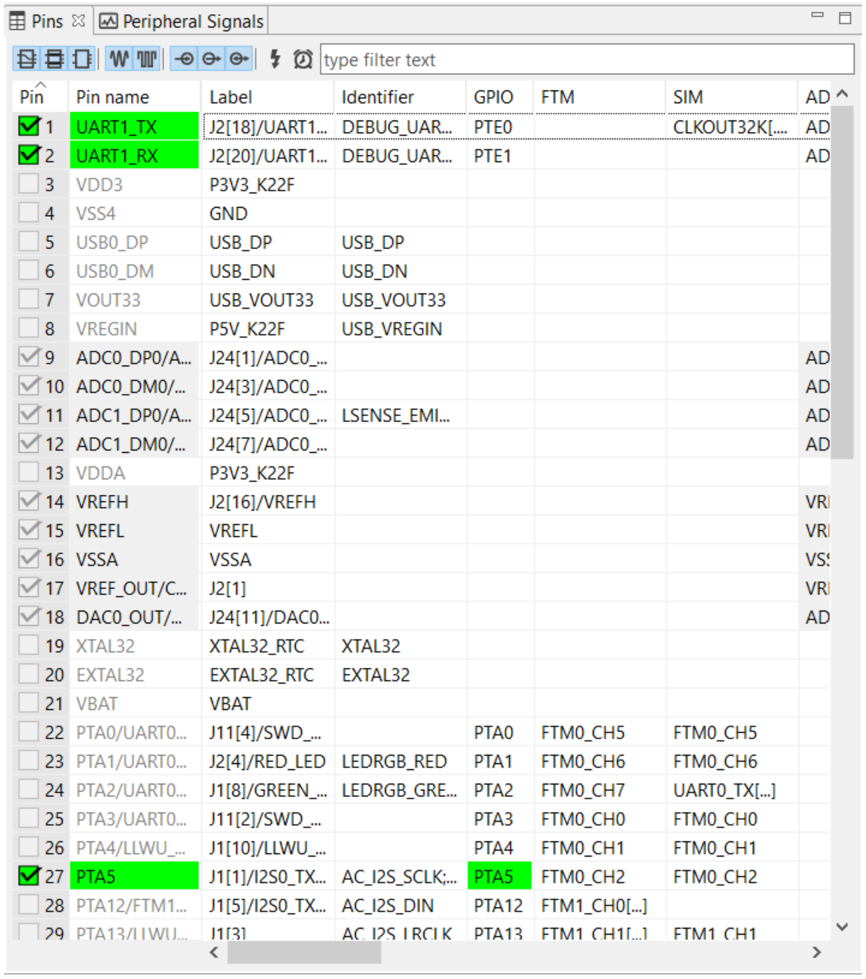
3.3 Example usage

This section lists the steps to create an example pin configuration, which can then be used in a project.

In this example, three pins **(UART3_RX, UART3_TX and PTB20)** on a board are configured.

You can use the generated files with the application code.

1. In the **Pins** view on the left, select the **UART3_RX** and **TX** signals. For this, you can click into the cells to make them 'green'.



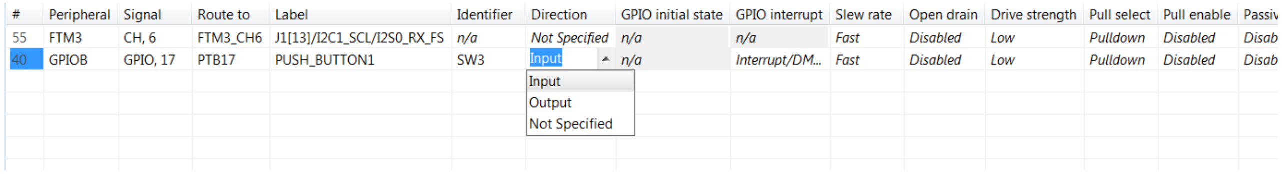
Pins | Peripheral Signals

type filter text

Pin	Pin name	Label	Identifier	GPIO	FTM	SIM	AD
✓ 1	UART1_TX	J2[18]/UART1...	DEBUG_UAR...	PTE0		CLKOUT32K[....	AD
✓ 2	UART1_RX	J2[20]/UART1...	DEBUG_UAR...	PTE1			AD
3	VDD3	P3V3_K22F					
4	VSS4	GND					
5	USB0_DP	USB_DP	USB_DP				
6	USB0_DM	USB_DN	USB_DN				
7	VOUT33	USB_VOUT33	USB_VOUT33				
8	VREGIN	P5V_K22F	USB_VREGIN				
✓ 9	ADC0_DP0/A...	J24[1]/ADC0_...					AD
✓ 10	ADC0_DM0/...	J24[3]/ADC0_...					AD
✓ 11	ADC1_DP0/A...	J24[5]/ADC0_...	LSENSE_EMI...				AD
✓ 12	ADC1_DM0/...	J24[7]/ADC0_...					AD
13	VDDA	P3V3_K22F					
✓ 14	VREFH	J2[16]/VREFH					VRI
✓ 15	VREFL	VREFL					VRI
✓ 16	VSSA	VSSA					VSS
✓ 17	VREF_OUT/C...	J2[1]					VRI
✓ 18	DAC0_OUT/...	J24[11]/DAC0...					AD
19	XTAL32	XTAL32_RTC	XTAL32				
20	EXTAL32	EXTAL32_RTC	EXTAL32				
21	VBAT	VBAT					
22	PTA0/UART0...	J11[4]/SWD_...		PTA0	FTM0_CH5	FTM0_CH5	
23	PTA1/UART0...	J2[4]/RED_LED	LEDRGB_RED	PTA1	FTM0_CH6	FTM0_CH6	
24	PTA2/UART0...	J1[8]/GREEN_...	LEDRGB_GRE...	PTA2	FTM0_CH7	UART0_TX[...]	
25	PTA3/UART0...	J11[2]/SWD_...		PTA3	FTM0_CH0	FTM0_CH0	
26	PTA4/LLWU_...	J1[10]/LLWU_...		PTA4	FTM0_CH1	FTM0_CH1	
✓ 27	PTA5	J1[1]/I2S0_TX...	AC_I2S_SCLK;...	PTA5	FTM0_CH2	FTM0_CH2	
28	PTA12/FTM1...	J1[5]/I2S0_TX...	AC_I2S_DIN	PTA12	FTM1_CH0[...]		
29	PTA13/LLWU...	J1[3]	AC_I2S_LRCLK	PTA13	FTM1_CH1[...]	FTM1_CH1	

Figure 19. Configure Signals in Pins View

2. In the middle view, called the **Routed Pins** view, select the **Output** direction for the TX and PTB20 signals.



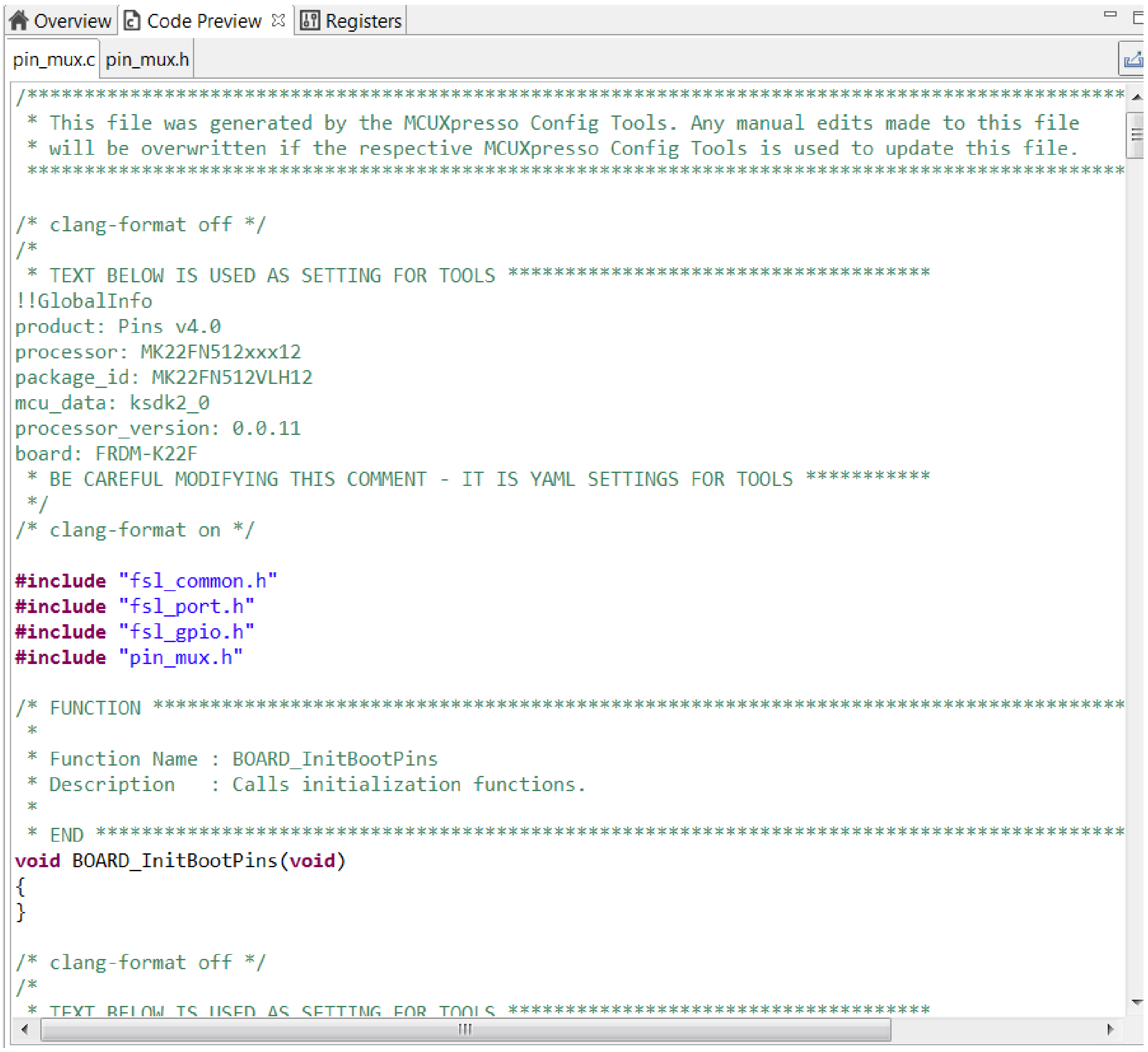
#	Peripheral	Signal	Route to	Label	Identifier	Direction	GPIO initial state	GPIO interrupt	Slew rate	Open drain	Drive strength	Pull select	Pull enable	Passiv
55	FTM3	CH, 6	FTM3_CH6	J1[13]/I2C1_SCL/I2S0_RX_FS	n/a	Not Specified	n/a	n/a	Fast	Disabled	Low	Pulldown	Disabled	Disab
40	GPIOB	GPIO, 17	PTB17	PUSH_BUTTON1	SW3	Input	n/a	Interrupt/DM...	Fast	Disabled	Low	Pulldown	Disabled	Disab

Figure 20. Select Direction

NOTE

For GPIO peripherals, you can set the **Direction** by clicking the cell and selecting from the drop-down menu. If you select **Output** you can also set **GPIO initial state** by clicking the cell in the **GPIO initial state** column. If you select Input you can also set GPIO interrupt by clicking the cell in the **GPIO interrupt** column.

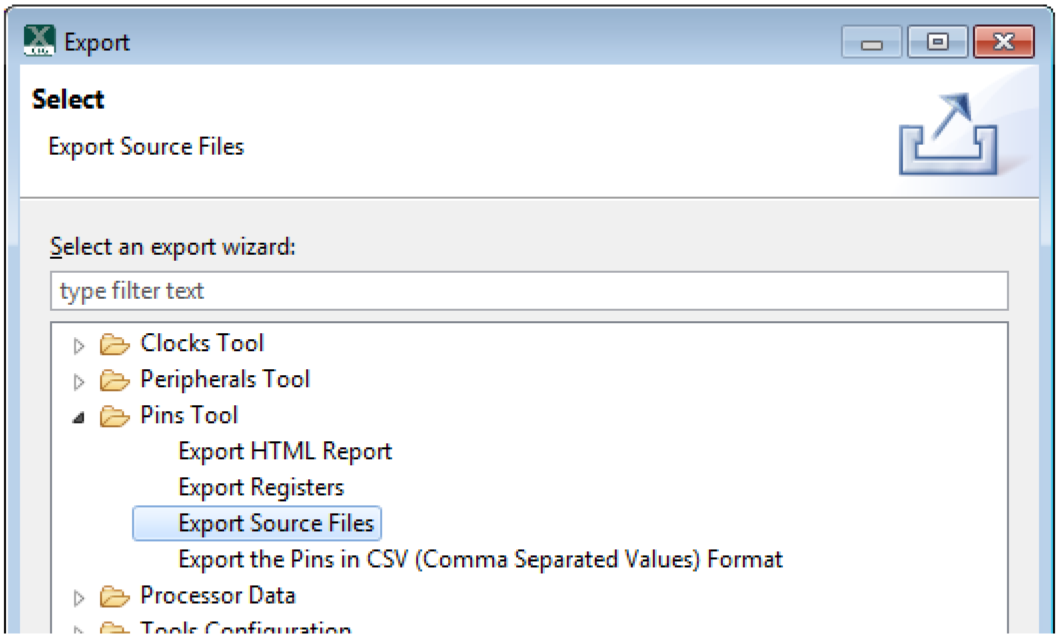
3. The Pins Tool automatically generates the source code for `pin_mux.c` and `pin_mux.h` on the right panel of the **Code Preview** view.



```
/***********************************************************************************************
 * This file was generated by the MCUXpresso Config Tools. Any manual edits made to this file
 * will be overwritten if the respective MCUXpresso Config Tools is used to update this file.
 ***********************************************************************************************

/* clang-format off */
/*
 * TEXT BELOW IS USED AS SETTING FOR TOOLS *************************************
!!GlobalInfo
product: Pins v4.0
processor: MK22FN512xxx12
package_id: MK22FN512VLH12
mcu_data: ksdk2_0
processor_version: 0.0.11
board: FRDM-K22F
 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS ***********
 */
/* clang-format on */

#include "fsl_common.h"
#include "fsl_port.h"
#include "fsl_gpio.h"
#include "pin_mux.h"

/* FUNCTION ************************************************************************************
 *
 * Function Name : BOARD_InitBootPins
 * Description   : Calls initialization functions.
 *
 * END *****************************************************************************************
void BOARD_InitBootPins(void)
{
}

/* clang-format off */
/*
 * TEXT BELOW IS USED AS SETTING FOR TOOLS *************************************
```

Figure 21. Generated code

4. You can now copy-paste the content of the source(s) to your application and IDE. Alternatively, you can export the generated files. To export the files, select the menu **File > Export** (in the desktop version) or select the menu **Pins > Export** menu (in the Web version). In the **Export** dialog expand the tree control for the tool you want to export sources for and select the **Export Source Files** option. **Export**, select the **Export Source Files** option.


Export

Select

Export Source Files

Select an export wizard:

type filter text

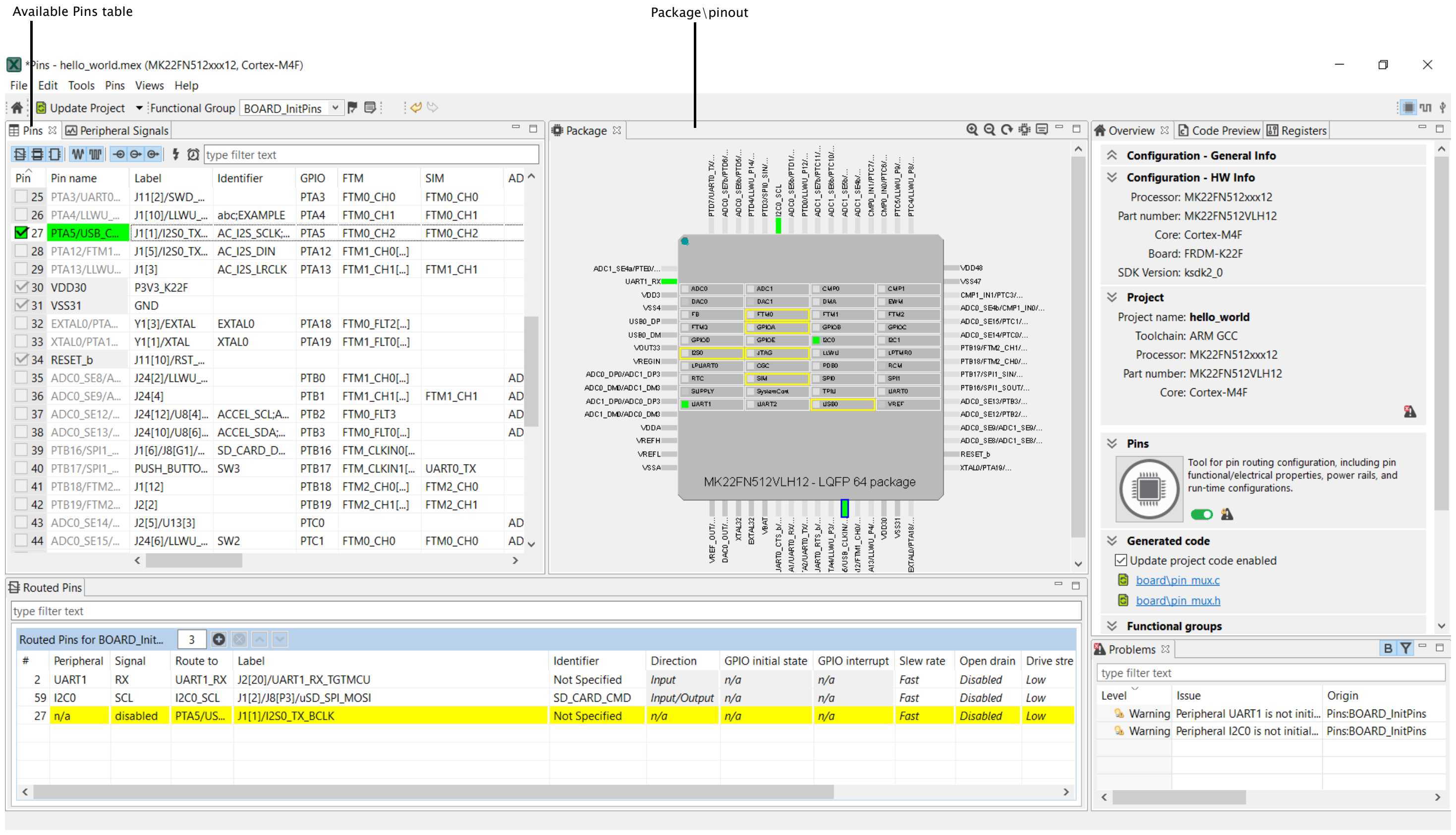
- Clocks Tool
- Peripherals Tool
- Pins Tool
  - Export HTML Report
  - Export Registers
  - Export Source Files
  - Export the Pins in CSV (Comma Separated Values) Format
- Processor Data
- Tools Configuration

Figure 22. Export Source Files

5. Click **Next** and specify the directory for each respective core (in multicore configuration) where you want to store the exported files for each individual core (in case of multicore configuration).
6. Click **Finish** to export the files.
7. Integrate and use the exported files in your application as source files.

3.4 User interface

Pins Tool consists of several views.


Available Pins table
Package\pinout
*Pins - hello_world.mex (MK22FN512xxx12, Cortex-M4F)
File Edit Tools Pins Views Help
Update Project
Functional Group BOARD_InitPins
Pins
Peripheral Signals
Package
Overview
Code Preview
Registers
MK22FN512VLH12 - LQFP 64 package
Routed Pins
Problems

Pin	Pin name	Label	Identifier	GPIO	FTM	SIM	AD
25	PTA3/UART0...	J11[2]/SWD_...		PTA3	FTM0_CH0	FTM0_CH0	
26	PTA4/LLWU_...	J1[10]/LLWU_...	abc;EXAMPLE	PTA4	FTM0_CH1	FTM0_CH1	
✓ 27	PTA5/USB_C...	J1[1]/I2S0_TX...	AC_I2S_SCLK;...	PTA5	FTM0_CH2	FTM0_CH2	
28	PTA12/FTM1...	J1[5]/I2S0_TX...	AC_I2S_DIN	PTA12	FTM1_CH0[...]		
29	PTA13/LLWU...	J1[3]	AC_I2S_LRCLK	PTA13	FTM1_CH1[...]	FTM1_CH1	
✓ 30	VDD30	P3V3_K22F					
✓ 31	VSS31	GND					
32	EXTAL0/PTA...	Y1[3]/EXTAL	EXTAL0	PTA18	FTM0_FLT2[...]		
33	XTAL0/PTA1...	Y1[1]/XTAL	XTAL0	PTA19	FTM1_FLT0[...]		
✓ 34	RESET_b	J11[10]/RST_...					
35	ADC0_SE8/A...	J24[2]/LLWU_...		PTB0	FTM1_CH0[...]		AD
36	ADC0_SE9/A...	J24[4]		PTB1	FTM1_CH1[...]	FTM1_CH1	AD
37	ADC0_SE12/...	J24[12]/U8[4]...	ACCEL_SCL;A...	PTB2	FTM0_FLT3		AD
38	ADC0_SE13/...	J24[10]/U8[6]...	ACCEL_SDA;...	PTB3	FTM0_FLT0[...]		AD
39	PTB16/SPI1_...	J1[6]/J8[G1]/...	SD_CARD_D...	PTB16	FTM_CLKIN0[...		
40	PTB17/SPI1_...	PUSH_BUTTO...	SW3	PTB17	FTM_CLKIN1[...	UART0_TX	
41	PTB18/FTM2...	J1[12]		PTB18	FTM2_CH0[...]	FTM2_CH0	
42	PTB19/FTM2...	J2[2]		PTB19	FTM2_CH1[...]	FTM2_CH1	
43	ADC0_SE14/...	J2[5]/U13[3]		PTC0			AD
44	ADC0_SE15/...	J24[6]/LLWU_...	SW2	PTC1	FTM0_CH0	FTM0_CH0	AD

Configuration - General Info

Configuration - HW Info
Processor: MK22FN512xxx12
Part number: MK22FN512VLH12
Core: Cortex-M4F
Board: FRDM-K22F
SDK Version: ksdk2_0

Project
Project name: **hello_world**
Toolchain: ARM GCC
Processor: MK22FN512xxx12
Part number: MK22FN512VLH12
Core: Cortex-M4F

Pins
Tool for pin routing configuration, including pin functional/electrical properties, power rails, and run-time configurations.

Generated code
Update project code enabled
board\pin_mux.c
board\pin_mux.h

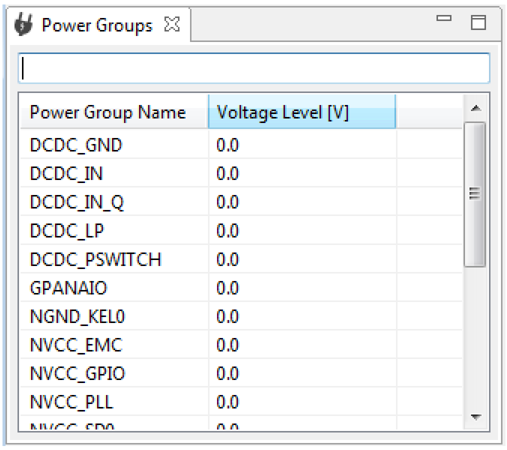
Functional groups

Routed Pins for BOARD_Init... 3

#	Peripheral	Signal	Route to	Label	Identifier	Direction	GPIO initial state	GPIO interrupt	Slew rate	Open drain	Drive stre
2	UART1	RX	UART1_RX	J2[20]/UART1_RX_TGTMCU	Not Specified	*Input*	*n/a*	*n/a*	*Fast*	*Disabled*	*Low*
59	I2C0	SCL	I2C0_SCL	J1[2]/J8[P3]/uSD_SPI_MOSI	SD_CARD_CMD	*Input/Output*	*n/a*	*n/a*	*Fast*	*Disabled*	*Low*
27	n/a	disabled	PTA5/US...	J1[1]/I2S0_TX_BCLK	Not Specified	*n/a*	*n/a*	*n/a*	*Fast*	*Disabled*	*Low*

Level	Issue	Origin
Warning	Peripheral UART1 is not initi...	Pins:BOARD_InitPins
Warning	Peripheral I2C0 is not initial...	Pins:BOARD_InitPins

Figure 23. Pins Tool user interface



Power Group Name	Voltage Level [V]
DCDC_GND	0.0
DCDC_IN	0.0
DCDC_IN_Q	0.0
DCDC_LP	0.0
DCDC_PSWITCH	0.0
GPANAIO	0.0
NGND_KEL0	0.0
NVCC_EMC	0.0
NVCC_GPIO	0.0
NVCC_PLL	0.0

Figure 24. Selecting power group

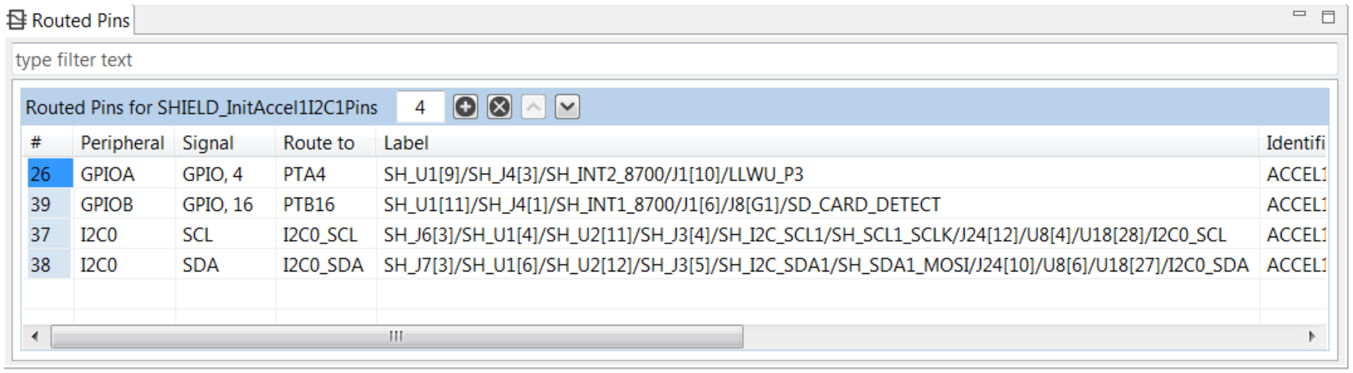
NOTE

Power Groups are not supported for all processors.

3.4.1 Functions

'Functions' are used to group a set of routed pins, and they create code for the configuration in a function which then can be called by the application.

The tool allows to creates multiple functions that can be used to configure pin muxing.


Routed Pins

type filter text

Routed Pins for SHIELD_InitAccel1I2C1Pins 4

#	Peripheral	Signal	Route to	Label	Identifi
26	GPIOA	GPIO, 4	PTA4	SH_U1[9]/SH_J4[3]/SH_INT2_8700/J1[10]/LLWU_P3	ACCEL1
39	GPIOB	GPIO, 16	PTB16	SH_U1[11]/SH_J4[1]/SH_INT1_8700/J1[6]/J8[G1]/SD_CARD_DETECT	ACCEL1
37	I2C0	SCL	I2C0_SCL	SH_J6[3]/SH_U1[4]/SH_U2[11]/SH_J3[4]/SH_I2C_SCL1/SH_SCL1_SCLK/J24[12]/U8[4]/U18[28]/I2C0_SCL	ACCEL1
38	I2C0	SDA	I2C0_SDA	SH_J7[3]/SH_U1[6]/SH_U2[12]/SH_J3[5]/SH_I2C_SDA1/SH_SDA1_MOSI/J24[10]/U8[6]/U18[27]/I2C0_SDA	ACCEL1

Figure 25. Routed Pins view

The usage of pins is indicated by 50% opacity in **Pins**, **Peripheral Signals**, and **Package** views. Each function can define a set of routed pins or re-configure already routed pins.

When multiple functions are specified in the configuration, the package view primarily shows the pins and the peripherals for the selected function. Pins and peripherals for different functions are shown with light transparency and cannot be configured, until switched to this function.

3.4.2 Package

The processor package appears in the middle of the Pins Tool window. The processor package shows an overall overview of the package including resource allocation.

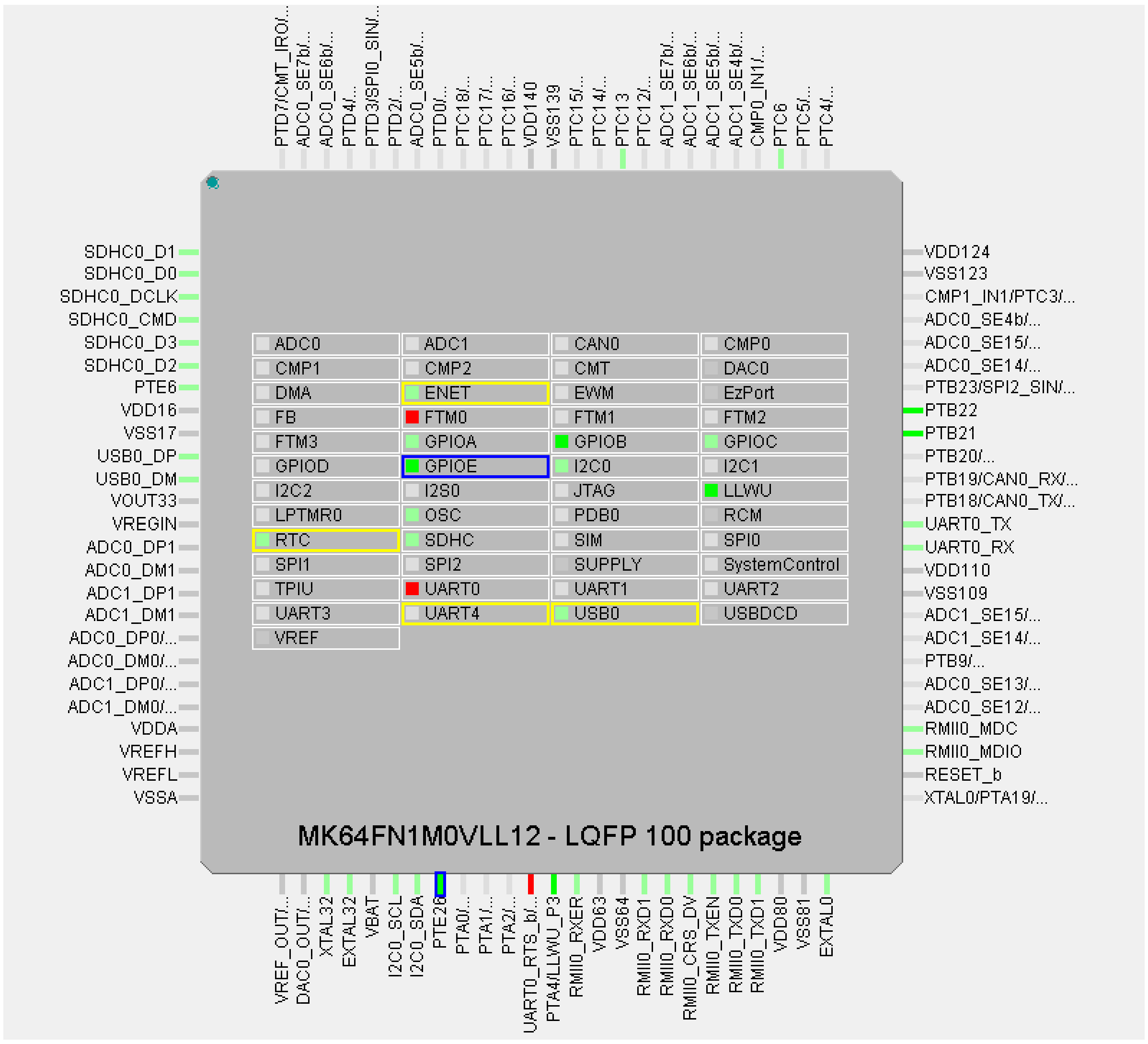


Figure 26. Processor package

This view shows Package overview with pins location. In the center are the peripherals.

For BGA packages, use the **Resources** icon to see them.

- Green color indicates the routed pins/peripherals.
- Gray color indicates that the pin/peripheral is not routed.
- Dark Gray color indicates that the pin/peripheral is dedicated. It is routed by default and has no impact on generated code.

The view also shows the package variant and the description (type and number of pins).

The following icons are available in the toolbar:

Table 3. Toolbar options

Icon	Description
🔍	Zoom in package image.

Table continues on the next page...

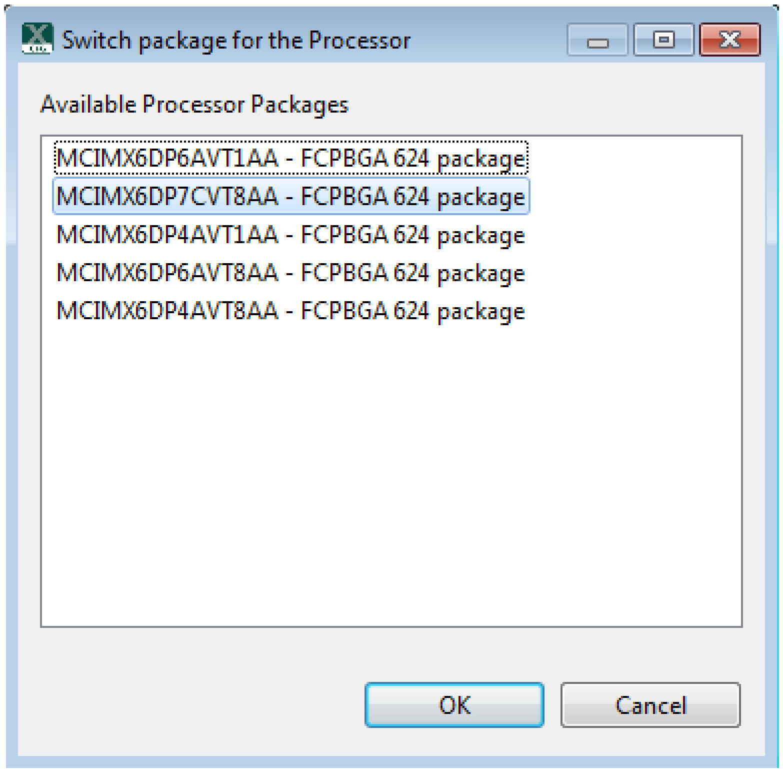
Table 3. Toolbar options (continued)

Icon	Description
	Zoom out package image.
	Rotate package image.
	Show pins as you can see it from the bottom. This option is available on BGA packages only.
	Show pins as you can see it from the top. This option is available on BGA packages only.
	Show resources. This option is available on BGA packages only.
	Switch package.
	Package legend

NOTE

Depending on the processor package selected, not all views are available.

The **Switch package** icon launches **Switch package for the Processor**.


Switch package for the Processor
Available Processor Packages
OK
Cancel

MCIMX6DP6AVT1AA - FCPBGA 624 package
MCIMX6DP7CVT8AA - FCPBGA 624 package
MCIMX6DP4AVT1AA - FCPBGA 624 package
MCIMX6DP6AVT8AA - FCPBGA 624 package
MCIMX6DP4AVT8AA - FCPBGA 624 package

Figure 27. Switch package

The **Switch package for the Processor** dialog shows list of available processor packages, showing package type and number of pins.

3.4.3 Routed Pins view

The **Routed Pins** view displays a list of routed pins and allows further configuration. This view also allows the configuration of the electrical properties of pins and displays all the pins. It displays the pad configuration available in a configuration where each pin is associated with the signal name and the function.

NOTE

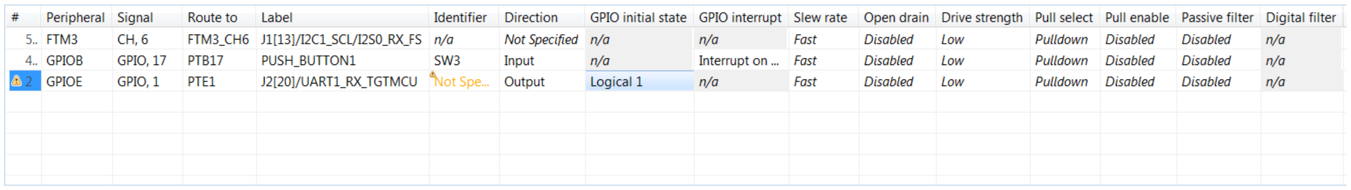
The electrical features are configured only for pins in the table. For example, the routed pins.

The table is empty when the new configuration is created, which means no pin is configured. Each row represents configuration of a single pin and if there are no conflicts, then the code is immediately updated. For Boards/Kits the pins are routed already

Use the table drop down menu to configure the pin. To configure pins, start from left to right – select the peripheral first, then the required signal, and finally, the routed pin.

See the right part of the table to configure the electrical features.

If the feature is not supported, n/a is displayed.



#	Peripheral	Signal	Route to	Label	Identifier	Direction	GPIO initial state	GPIO interrupt	Slew rate	Open drain	Drive strength	Pull select	Pull enable	Passive filter	Digital filter
5..	FTM3	CH, 6	FTM3_CH6	J1[13]/I2C1_SCL/I2S0_RX_FS	*n/a*	*Not Specified*	*n/a*	*n/a*	*Fast*	*Disabled*	*Low*	*Pulldown*	*Disabled*	*Disabled*	*n/a*
4..	GPIOB	GPIO, 17	PTB17	PUSH_BUTTON1	SW3	Input	*n/a*	Interrupt on ...	*Fast*	*Disabled*	*Low*	*Pulldown*	*Disabled*	*Disabled*	*n/a*
⚠ 2	GPIOE	GPIO, 1	PTE1	J2[20]/UART1_RX_TGTMCU	Not Spe...	Output	Logical 1	*n/a*	*Fast*	*Disabled*	*Low*	*Pulldown*	*Disabled*	*Disabled*	*n/a*

Figure 28. Routed Pins view

The gray background indicates the read-only items.

The italic value indicates that the value is not configured and it shows the after-reset value and no code is generated, so the configuration relies on the after reset value or the values configured from the different functions.

TIP

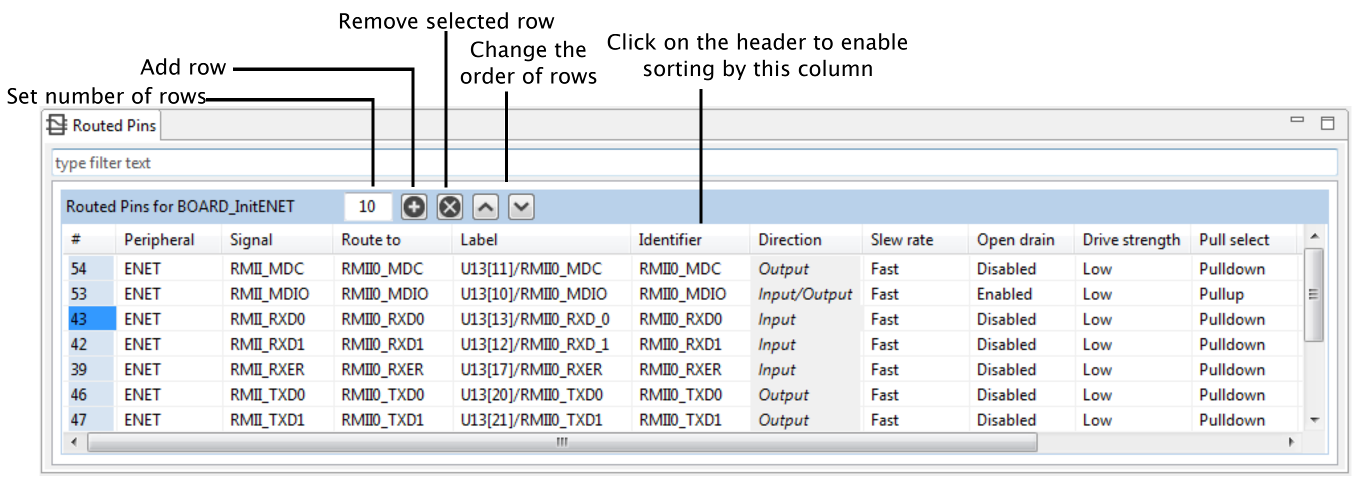
- The value shown using italic indicates the after-reset value. The real value may be different from the after reset value, if configured in other functions.

  Use the drop-down menu to select the required value.
- If you select the same value as the after-reset value, the tool will always generate code to set this feature.

  Use the drop-down "Reset" value to reset the value to its after-reset state.
- If an item does not support reset to after reset value, the **Reset** menu is not available. The first row shows pin number or coordinate on BGA package.

3.4.3.1 View controls

The following figure illustrates the **Routed pins** view controls.



#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Open drain	Drive strength	Pull select
54	ENET	RMII_MDC	RMII0_MDC	U13[11]/RMII0_MDC	RMII0_MDC	Output	Fast	Disabled	Low	Pulldown
53	ENET	RMII_MDIO	RMII0_MDIO	U13[10]/RMII0_MDIO	RMII0_MDIO	Input/Output	Fast	Enabled	Low	Pullup
43	ENET	RMII_RXD0	RMII0_RXD0	U13[13]/RMII0_RXD_0	RMII0_RXD0	Input	Fast	Disabled	Low	Pulldown
42	ENET	RMII_RXD1	RMII0_RXD1	U13[12]/RMII0_RXD_1	RMII0_RXD1	Input	Fast	Disabled	Low	Pulldown
39	ENET	RMII_RXER	RMII0_RXER	U13[17]/RMII0_RXER	RMII0_RXER	Input	Fast	Disabled	Low	Pulldown
46	ENET	RMII_TXD0	RMII0_TXD0	U13[20]/RMII0_TXD0	RMII0_TXD0	Output	Fast	Disabled	Low	Pulldown
47	ENET	RMII_TXD1	RMII0_TXD1	U13[21]/RMII0_TXD1	RMII0_TXD1	Output	Fast	Disabled	Low	Pulldown

Figure 29. View controls

Add / remove rows:

- To add a new row to the end of table, click on the [+] button.
- To remove the selected row, click on the [x] button.
- To delete a specific row or insert a new row at a given position, right-click and use the pop-up menu commands.

Add a specific number of rows or clear the table:

- To add a specific number of rows, specify the exact number of rows.
- To clear the table, type 0.

Change the order of the rows:

To change the order of the rows, use the arrow icons to move one row up or down.

Filter table entries:

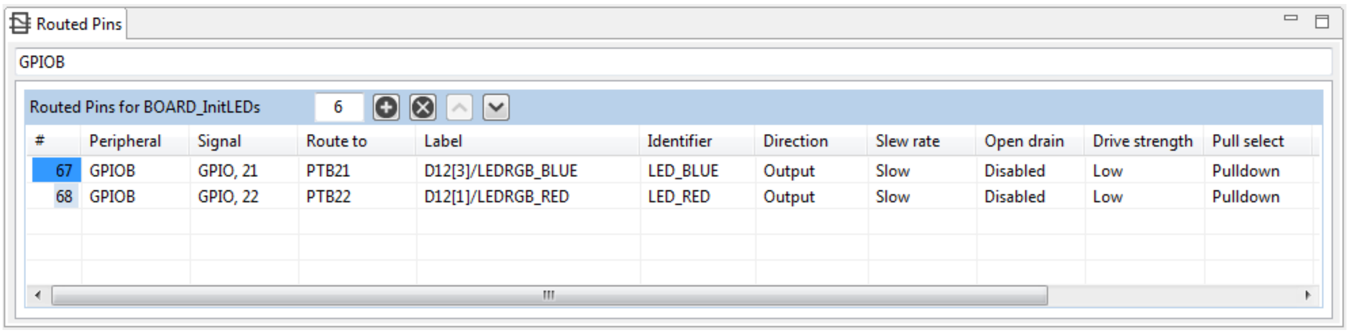
To filter table entries by text, enter the text string in the **type filter text** field.

Copy-paste rows:

To copy the row, right-click any cell in the row and choose **Copy**. You can later paste the copied row into the **Routed Pins** view of another functional group or configuration by right-clicking the table are and choosing **Paste**.

3.4.3.2 Filtering routed pins

The following image illustrates the filter area of the **Routed Pins** view.



#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Open drain	Drive strength	Pull select
67	GPIOB	GPIO, 21	PTB21	D12[3]/LEDRGB_BLUE	LED_BLUE	Output	Slow	Disabled	Low	Pulldown
68	GPIOB	GPIO, 22	PTB22	D12[1]/LEDRGB_RED	LED_RED	Output	Slow	Disabled	Low	Pulldown

Figure 30. Filter area

To instantly filter rows, type the text or the search phrase in the filter area (type filter text).

NOTE

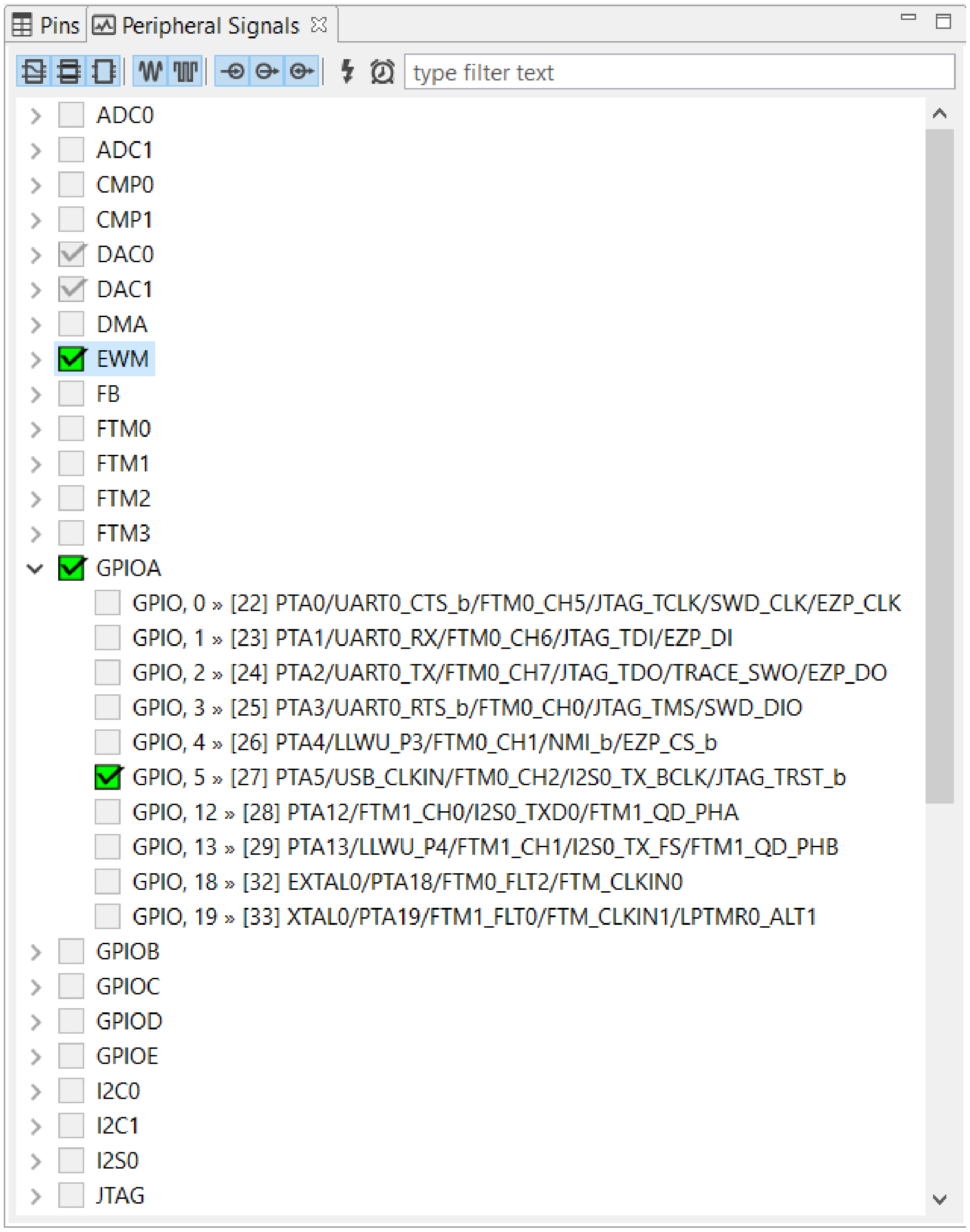
When you enter the search text, it also searches the text in the full pin names displays rows that contain the search text.

3.4.4 Peripheral Signals view

The **Peripheral Signals** view shows a list of peripherals and their signals. Only the **Peripheral Signals** and **Pins** view shows the checkbox (allocated) with status.

Table 4. Status codes

Color code	Status
🟥✓	Error
🟩✓	Configured
☐	Not configured
🟧✓	Warning
✓	Dedicated: Device is routed by default and has no impact on the generated code.



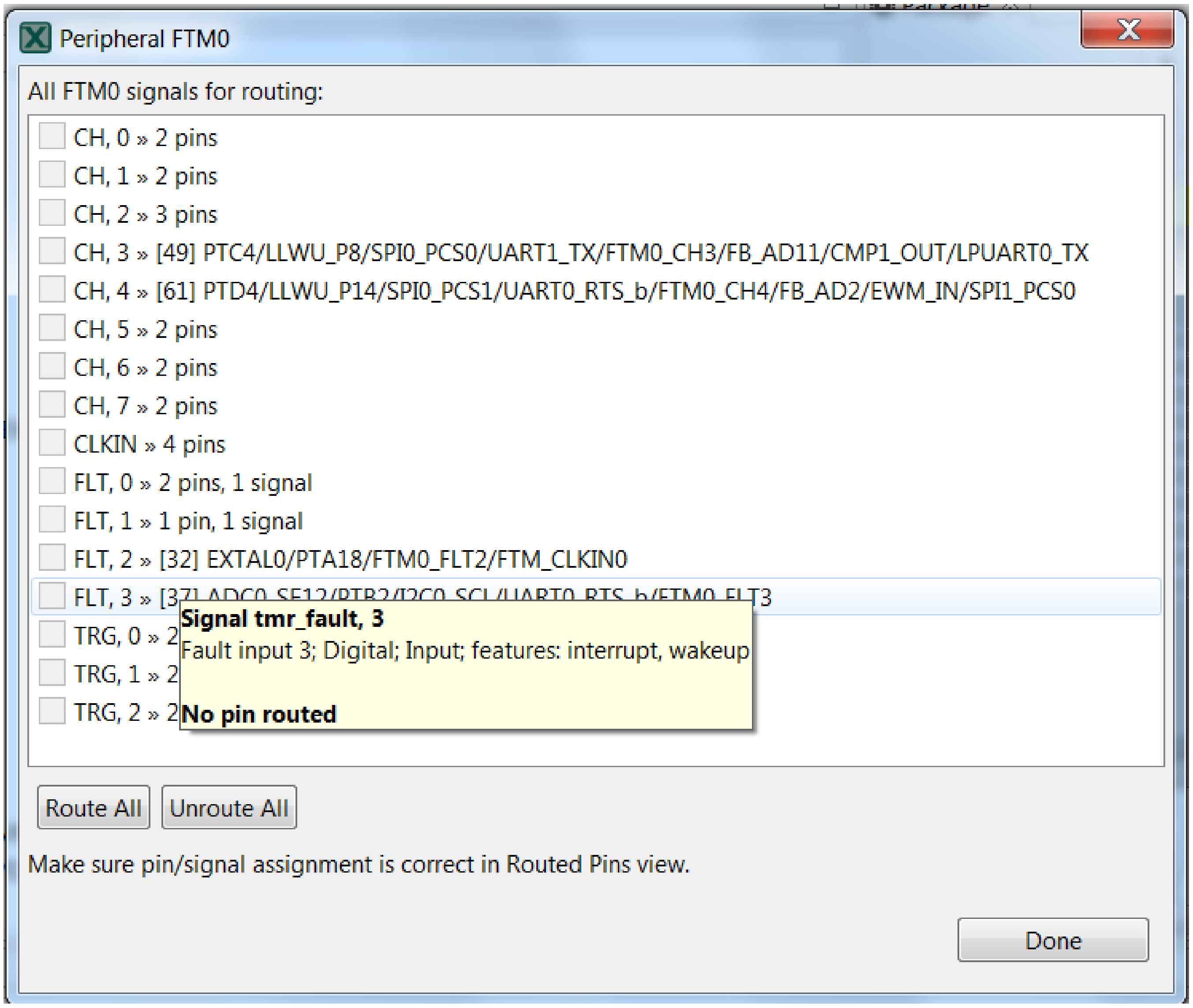
ADC0
ADC1
CMP0
CMP1
✓ DAC0
✓ DAC1
DMA
✓ EWM
FB
FTM0
FTM1
FTM2
FTM3
✓ GPIOA
- GPIO, 0 » [22] PTA0/UART0_CTS_b/FTM0_CH5/JTAG_TCLK/SWD_CLK/EZP_CLK
- GPIO, 1 » [23] PTA1/UART0_RX/FTM0_CH6/JTAG_TDI/EZP_DI
- GPIO, 2 » [24] PTA2/UART0_TX/FTM0_CH7/JTAG_TDO/TRACE_SWO/EZP_DO
- GPIO, 3 » [25] PTA3/UART0_RTS_b/FTM0_CH0/JTAG_TMS/SWD_DIO
- GPIO, 4 » [26] PTA4/LLWU_P3/FTM0_CH1/NMI_b/EZP_CS_b
- ✓ GPIO, 5 » [27] PTA5/USB_CLKIN/FTM0_CH2/I2S0_TX_BCLK/JTAG_TRST_b
- GPIO, 12 » [28] PTA12/FTM1_CH0/I2S0_TXD0/FTM1_QD_PHA
- GPIO, 13 » [29] PTA13/LLWU_P4/FTM1_CH1/I2S0_TX_FS/FTM1_QD_PHB
- GPIO, 18 » [32] EXTAL0/PTA18/FTM0_FLT2/FTM_CLKIN0
- GPIO, 19 » [33] XTAL0/PTA19/FTM1_FLT0/FTM_CLKIN1/LPTMR0_ALT1

GPIOB
GPIOC
GPIOD
GPIOE
I2C0
I2C1
I2S0
JTAG

Figure 31. Peripheral Signals view

Use the checkbox to route/unroute the selected pins.

To route/unroute multiple pins, click the peripheral and select the options in the **Select signals** dialog.


Peripheral FTM0

All FTM0 signals for routing:

- CH, 0 » 2 pins
- CH, 1 » 2 pins
- CH, 2 » 3 pins
- CH, 3 » [49] PTC4/LLWU_P8/SPI0_PCS0/UART1_TX/FTM0_CH3/FB_AD11/CMP1_OUT/LPUART0_TX
- CH, 4 » [61] PTD4/LLWU_P14/SPI0_PCS1/UART0_RTS_b/FTM0_CH4/FB_AD2/EWM_IN/SPI1_PCS0
- CH, 5 » 2 pins
- CH, 6 » 2 pins
- CH, 7 » 2 pins
- CLKIN » 4 pins
- FLT, 0 » 2 pins, 1 signal
- FLT, 1 » 1 pin, 1 signal
- FLT, 2 » [32] EXTAL0/PTA18/FTM0_FLT2/FTM_CLKIN0
- FLT, 3 » [37] ADC0_SE12/PTB2/I2C0_SCL/UART0_RTS_b/FTM0_FLT3
- TRG, 0 » 2
- TRG, 1 » 2
- TRG, 2 » 2

Signal tmr_fault, 3
Fault input 3; Digital; Input; features: interrupt, wakeup

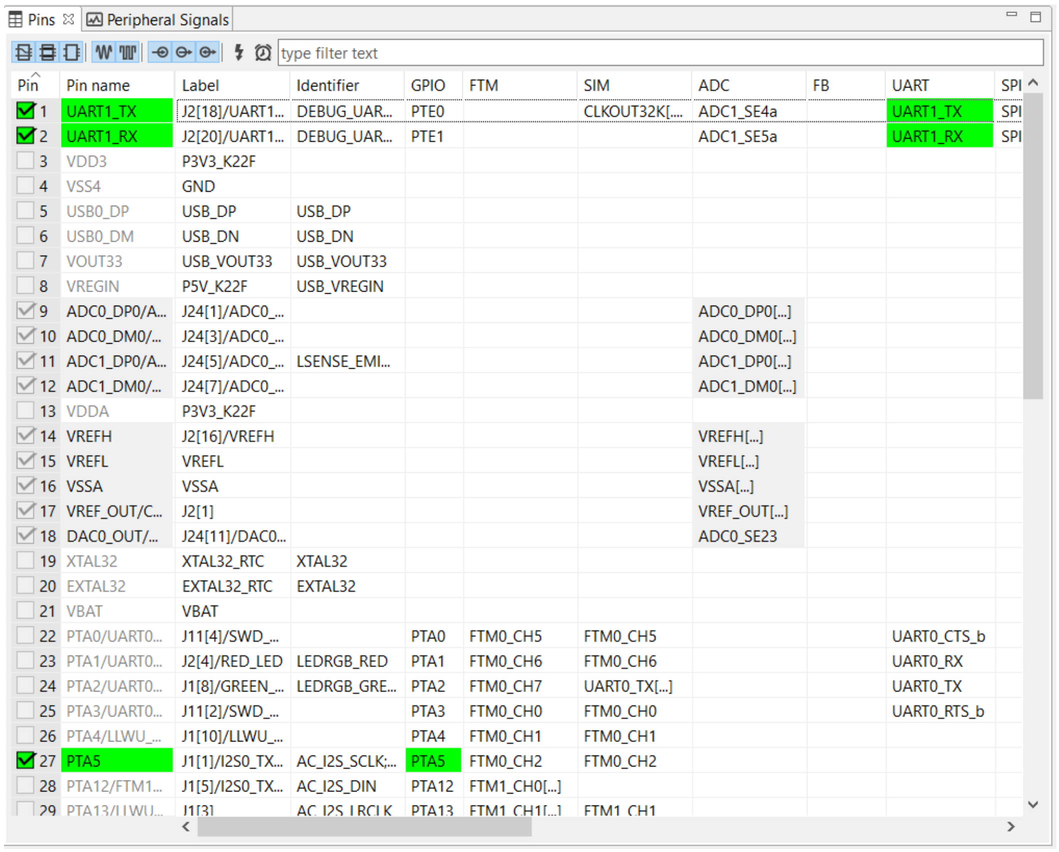
No pin routed

Make sure pin/signal assignment is correct in Routed Pins view.

Figure 32. Select signals dialog

3.4.5 Pins table view

The **Pins** table view shows all the pins in a table format.



Pin	Pin name	Label	Identifier	GPIO	FTM	SIM	ADC	FB	UART	SPI
✓ 1	UART1_TX	J2[18]/UART1...	DEBUG_UAR...	PTE0		CLKOUT32K[....	ADC1_SE4a		UART1_TX	SPI
✓ 2	UART1_RX	J2[20]/UART1...	DEBUG_UAR...	PTE1			ADC1_SE5a		UART1_RX	SPI
3	VDD3	P3V3_K22F								
4	VSS4	GND								
5	USB0_DP	USB_DP	USB_DP							
6	USB0_DM	USB_DN	USB_DN							
7	VOUT33	USB_VOUT33	USB_VOUT33							
8	VREGIN	P5V_K22F	USB_VREGIN							
✓ 9	ADC0_DP0/A...	J24[1]/ADC0_...					ADC0_DP0[...]			
✓ 10	ADC0_DM0/...	J24[3]/ADC0_...					ADC0_DM0[...]			
✓ 11	ADC1_DP0/A...	J24[5]/ADC0_...	LSENSE_EMI...				ADC1_DP0[...]			
✓ 12	ADC1_DM0/...	J24[7]/ADC0_...					ADC1_DM0[...]			
13	VDDA	P3V3_K22F								
✓ 14	VREFH	J2[16]/VREFH					VREFH[...]			
✓ 15	VREFL	VREFL					VREFL[...]			
✓ 16	VSSA	VSSA					VSSA[...]			
✓ 17	VREF_OUT/C...	J2[1]					VREF_OUT[...]			
✓ 18	DAC0_OUT/...	J24[11]/DAC0...					ADC0_SE23			
19	XTAL32	XTAL32_RTC	XTAL32							
20	EXTAL32	EXTAL32_RTC	EXTAL32							
21	VBAT	VBAT								
22	PTA0/UART0...	J11[4]/SWD_...		PTA0	FTM0_CH5	FTM0_CH5			UART0_CTS_b	
23	PTA1/UART0...	J2[4]/RED_LED	LEDRGB_RED	PTA1	FTM0_CH6	FTM0_CH6			UART0_RX	
24	PTA2/UART0...	J1[8]/GREEN_...	LEDRGB_GRE...	PTA2	FTM0_CH7	UART0_TX[...]			UART0_TX	
25	PTA3/UART0...	J11[2]/SWD_...		PTA3	FTM0_CH0	FTM0_CH0			UART0_RTS_b	
26	PTA4/LLWU_...	J1[10]/LLWU_...		PTA4	FTM0_CH1	FTM0_CH1				
✓ 27	PTA5	J1[1]/I2S0_TX...	AC_I2S_SCLK;...	PTA5	FTM0_CH2	FTM0_CH2				
28	PTA12/FTM1...	J1[5]/I2S0_TX...	AC_I2S_DIN	PTA12	FTM1_CH0[...]					
29	PTA13/LLWU...	J1[3]	AC_I2S_LRCLK	PTA13	FTM1_CH1[...]	FTM1_CH1				

Figure 33. Pins table view

This view shows the list of all the pins available on a given device. The **Pin name** column shows the default name of the pin, or if the pin is routed. The pin name is changed to show appropriate function for selected peripheral if routed. The next columns of the table shows peripherals and pin name(s) on given peripheral. Peripherals with few items are cumulated in the last column.

To route/un-route pin to the given peripheral, click in the cell of the table. Routed pins are marked with checkbox and green color. Colored cells indicate that a pin is routed to given peripherals. If there is conflict in routing, red color is used.

Unroute is possible by clicking on a given cell, or by checkbox in the first column.

Every routed pin appears in the Routed pins table.

When multiple functions are specified in the configuration, the Pins Table view shows pins for selected function primarily. Pins for different functions are shown with light transparency and cannot be configured until switched to this function.

TIP

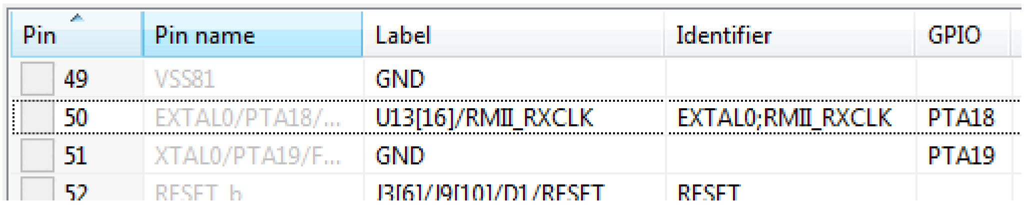
The option to route more signals to a single pin is indicated by an ellipsis (...). The **Multiple Signals Configuration** dialog appears, if clicked.

3.4.5.1 Labels and identifiers

It's possible to define label of any pin that can be shown in UI for easy pin identification.

The boards and kits have pre-defined labels. However, it is also possible to define a pin label listed in the **Routed Pins** view. To set\update the **Labels and Identifier** columns visibility, select **Edit > Preferences**.

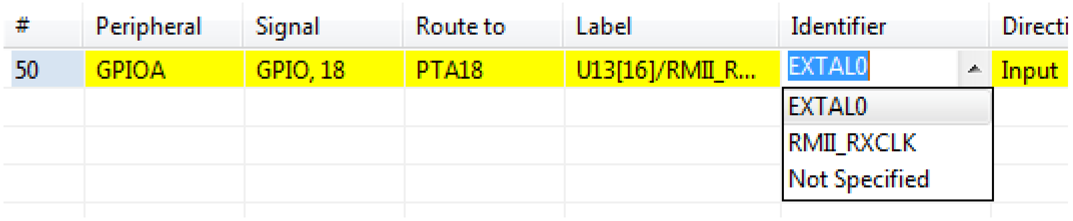
The pin identifier is used to generate the `#define` in the pin_mux.h file. However, it's an optional parameter. If the parameter is not defined, the code for `#define` is not generated. Additionally, you can define multiple identifiers, using the ";" character as a separator. You can also set the identifier by typing it directly into the cell in the **Identifier** column in the **Routed Pins** view.



Pin	Pin name	Label	Identifier	GPIO
49	VSS81	GND		
50	EXTAL0/PTA18/...	U13[16]/RMII_RXCLK	EXTAL0;RMII_RXCLK	PTA18
51	XTAL0/PTA19/F...	GND		PTA19
52	RESET_b	J3[6]/J9[10]/D1/RESET	RESET	

Figure 34. Pin Identifier

In this case it's possible to select from values if the pin is routed. See Routed pins table.

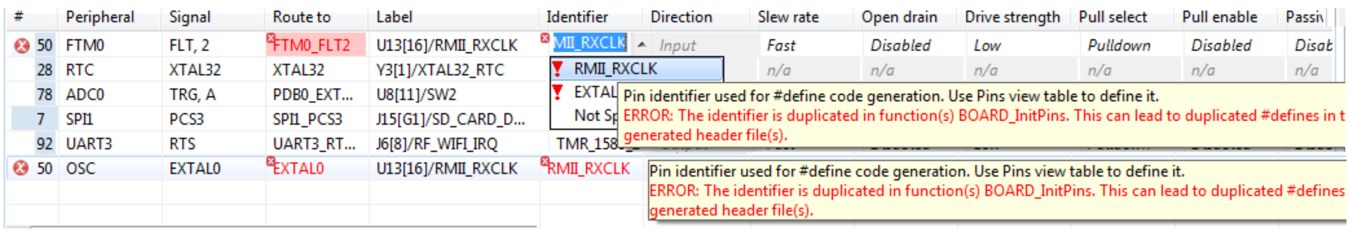


#	Peripheral	Signal	Route to	Label	Identifier	Directi
50	GPIOA	GPIO, 18	PTA18	U13[16]/RMII_R...	EXTAL0	Input

EXTAL0
RMII_RXCLK
Not Specified

Figure 35. Identifier in Routed Pins table

A check is implemented to ensure whether the generated defines are duplicated in the `pin_mux.h` file. These duplications are indicated in the identifier column as errors. See Identifier errors.



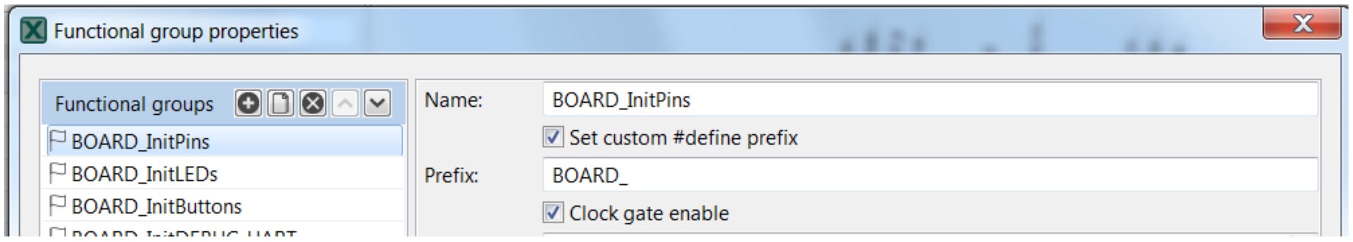
#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Open drain	Drive strength	Pull select	Pull enable	Passiv
50	FTM0	FLT, 2	FTM0_FLT2	U13[16]/RMII_RXCLK	MII_RXCLK	Input	Fast	Disabled	Low	Pulldown	Disabled	Disab
28	RTC	XTAL32	XTAL32	Y3[1]/XTAL32_RTC	RMII_RXCLK		n/a	n/a	n/a	n/a	n/a	n/a
78	ADC0	TRG, A	PDB0_EXT...	U8[11]/SW2	EXTAL							
7	SPI1	PCS3	SPI1_PCS3	J15[G1]/SD_CARD_D...	Not Sp							
92	UART3	RTS	UART3_RT...	J6[8]/RF_WIFI_IRQ	TMR_158							
50	OSC	EXTAL0	EXTAL0	U13[16]/RMII_RXCLK	RMII_RXCLK							

Pin identifier used for #define code generation. Use Pins view table to define it.
ERROR: The identifier is duplicated in function(s) BOARD_InitPins. This can lead to duplicated #defines in t
generated header file(s).

Pin identifier used for #define code generation. Use Pins view table to define it.
ERROR: The identifier is duplicated in function(s) BOARD_InitPins. This can lead to duplicated #defines
generated header file(s).

Figure 36. Identifier errors

You can also select the pin to use in a given routing from the **Routed Pins** view. However, the identifier must be a valid C identifier and should be used in the source code.



Functional group properties

Functional groups
BOARD_InitPins
BOARD_InitLEDs
BOARD_InitButtons
BOARD_InitDEBUG_UART

Name: BOARD_InitPins
☑ Set custom #define prefix
Prefix: BOARD_
☑ Clock gate enable

Figure 37. Pins macros prefix

If multiple functions are used, each individual function can include a special prefix. Check the **Pins** > **Functional Group Properties** > **Set custome #define prefix** checkbox to enter prefix of macros in particular function used in the generated code of the pin_mux.h file. Entered prefix text must be a C identifier. If unchecked, the **Function name** is used as a default prefix.

3.4.6 Filtering in the Pins and Peripheral Signals views

The following image illustrates the filtering controls in the **Pins** and **Peripheral Signals** views.

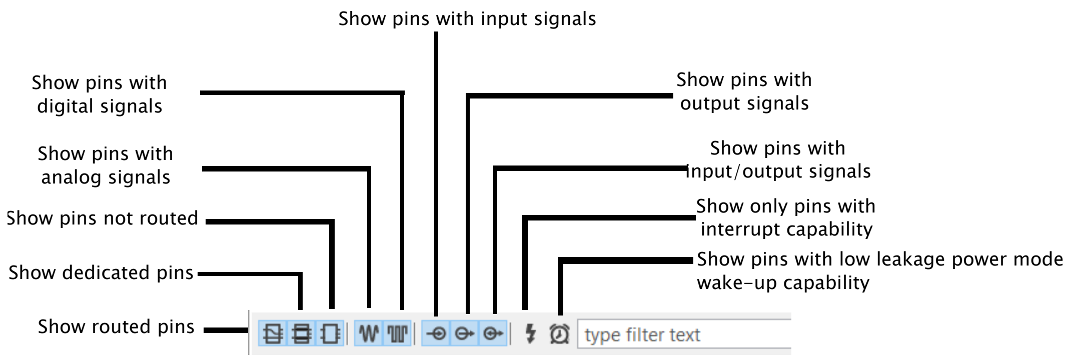

Show pins with input signals
Show pins with digital signals
Show pins with analog signals
Show pins not routed
Show dedicated pins
Show routed pins
Show pins with output signals
Show pins with input/output signals
Show only pins with interrupt capability
Show pins with low leakage power mode wake-up capability
type filter text

Figure 38. Filtering Controls

Type any text to search across the table/tree. It will search for the pins/peripheral signals containing the specified text. You can also use wildcards "*" and "?" to help you filter results you want. Use "space" to search for multiple strings at the same time.

3.4.7 Highlighting and color coding

It's possible to easily identify routed pins/peripherals in the package using highlighting. By default, the current selection (pin/ peripheral) is highlighted in the package view.

- The pin/peripheral is highlighted by yellow border around it in the Package view. If the highlighted pin/peripheral is selected then it has a blue border around it.
- Red indicates that the pin has an error.
- Green indicates that the pin is muxed or used.
- Light grey indicates that the pin is available for mux, but is not muxed or used.
- Dark gray indicates that the pin/peripheral is dedicated. It is routed by default and has no impact on generated code.

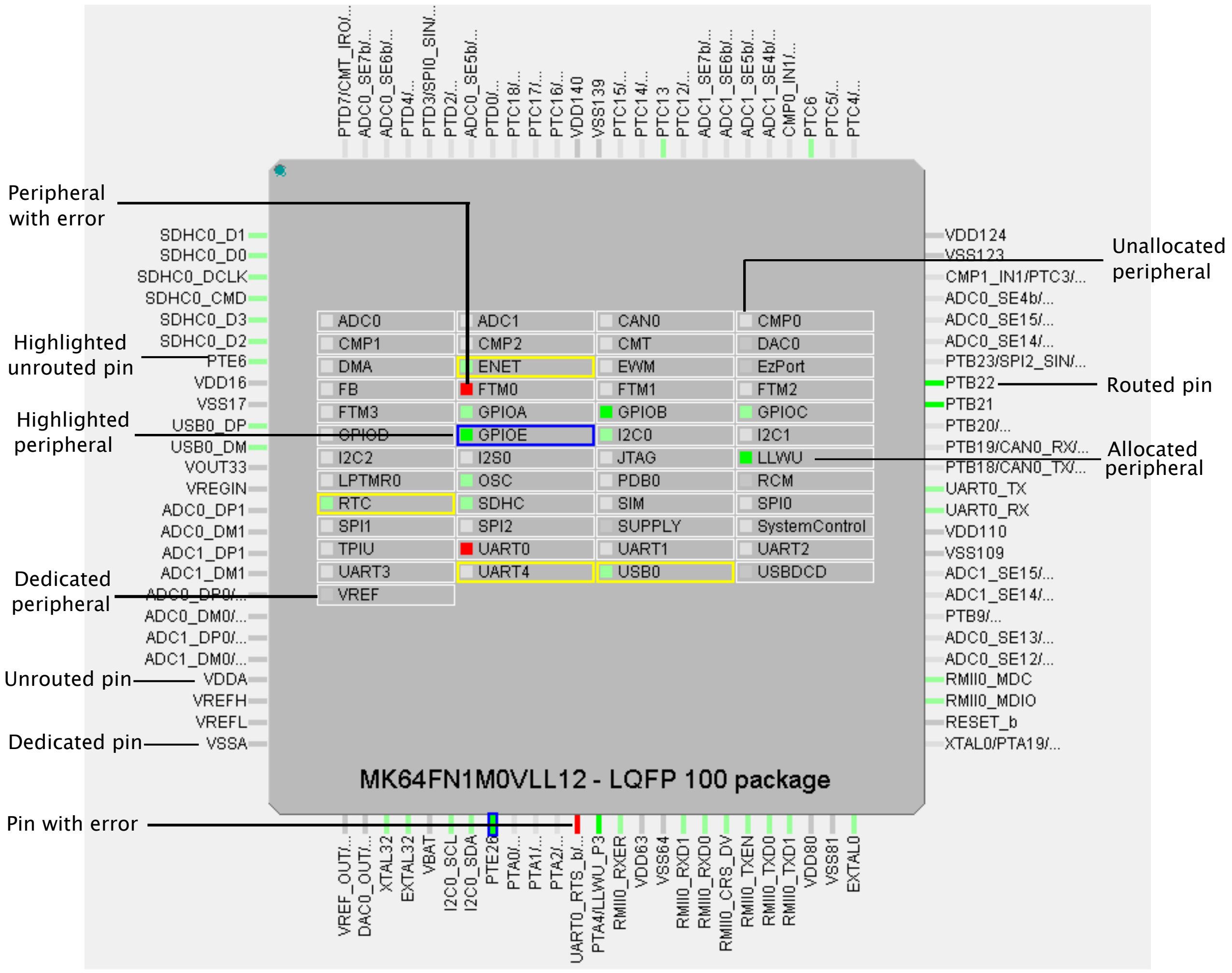
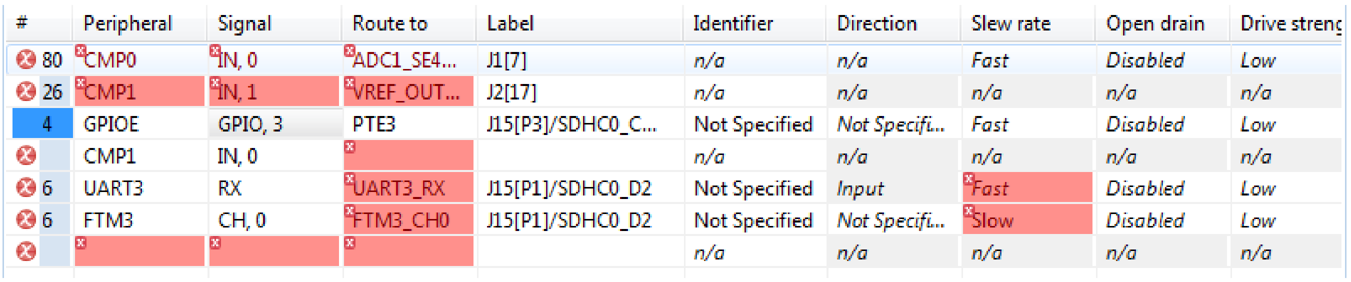
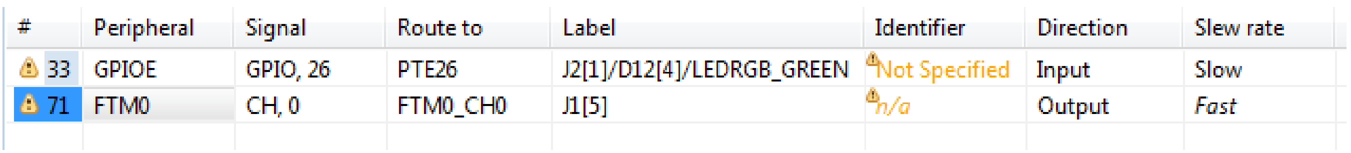


Figure 39. Highlighting and color coding



#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Open drain	Drive streng
✖ 80	CMP0	IN, 0	ADC1_SE4...	J1[7]	*n/a*	*n/a*	*Fast*	*Disabled*	*Low*
✖ 26	CMP1	IN, 1	VREF_OUT...	J2[17]	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*
4	GPIOE	GPIO, 3	PTE3	J15[P3]/SDHC0_C...	Not Specified	*Not Specifi...*	*Fast*	*Disabled*	*Low*
✖	CMP1	IN, 0			*n/a*	*n/a*	*n/a*	*n/a*	*n/a*
✖ 6	UART3	RX	UART3_RX	J15[P1]/SDHC0_D2	Not Specified	*Input*	*Fast*	*Disabled*	*Low*
✖ 6	FTM3	CH, 0	FTM3_CH0	J15[P1]/SDHC0_D2	Not Specified	*Not Specifi...*	*Slow*	*Disabled*	*Low*
✖					*n/a*	*n/a*	*n/a*	*n/a*	*n/a*

Figure 40. Pins conflicts



#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate
⚠ 33	GPIOE	GPIO, 26	PTE26	J2[1]/D12[4]/LEDRGB_GREEN	⚠Not Specified	Input	Slow
⚠ 71	FTM0	CH, 0	FTM0_CH0	J1[5]	⚠*n/a*	Output	*Fast*

Figure 41. Warnings

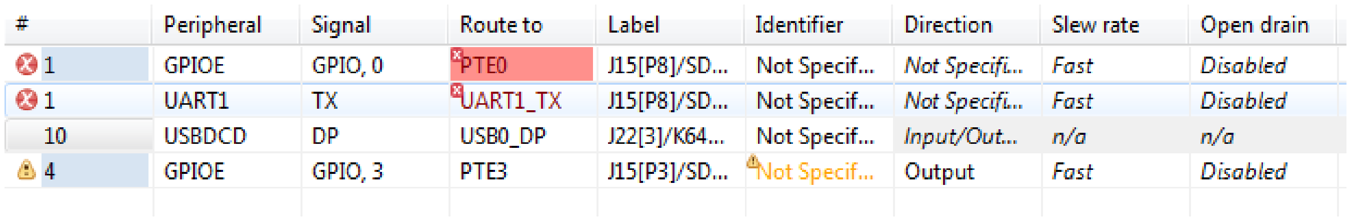
- **Package** view
  — Click on the peripheral or use the pop-up menu to highlight peripherals:

  - ◦ and all allocated pins (to selected peripheral).
  - ◦ or all available pins if nothing is allocated yet.
- — Click on the pin or use the pop-up menu to highlight the pin and the peripherals.
- — Click outside the package to cancel the highlight.

- **Peripherals / Pins** view
  - — The peripheral and pin behaves as described above image.

3.5 Errors and warnings

The Pins Tool checks for any conflict in the routing and also for errors in the configuration. Routing conflicts are checked only for the selected function . It is possible to configure different routing of one pin in different functions to allow dynamic pins routing re-configuration.



#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Open drain
1	GPIOE	GPIO, 0	PTE0	J15[P8]/SD...	Not Specif...	Not Specifi...	Fast	Disabled
1	UART1	TX	UART1_TX	J15[P8]/SD...	Not Specif...	Not Specifi...	Fast	Disabled
10	USBDCD	DP	USB0_DP	J22[3]/K64...	Not Specif...	Input/Out...	n/a	n/a
4	GPIOE	GPIO, 3	PTE3	J15[P3]/SD...	Not Specif...	Output	Fast	Disabled

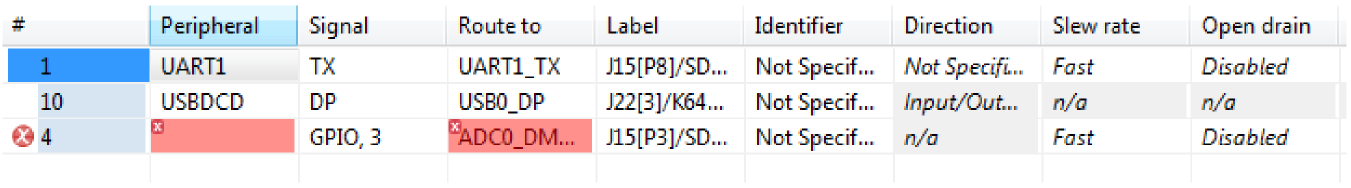
Figure 42. Error and warnings

If an error or warning is encountered, the conflict in the **Routed Pins** view is represented in the first column of the row and the error/warning is indicated in the cell, where the conflict was created. The first two rows in the figure above show the peripheral/ signal where the erroneous configuration occurs. The fourth row shows the warning on the unconfigured identifier while specifying a direction. The detailed error/warning message appears as a tooltip.

For more information on error and warnings color, see the Highlighting and Color Coding section.

3.5.1 Incomplete routing

A cell with incomplete routing is indicated by a red background. To generate proper pin routing, click on the drop down arrow and select the suitable value. A red decorator on a cell indicates an error condition.



#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Open drain
1	UART1	TX	UART1_TX	J15[P8]/SD...	Not Specif...	Not Specifi...	Fast	Disabled
10	USBDCD	DP	USB0_DP	J22[3]/K64...	Not Specif...	Input/Out...	n/a	n/a
4		GPIO, 3	ADC0_DM...	J15[P3]/SD...	Not Specif...	n/a	Fast	Disabled

Figure 43. Incomplete routing

The tooltip of the cell shows more details about the conflict or the error, typically it lists the lines where conflict occurs.

3.6 Code generation

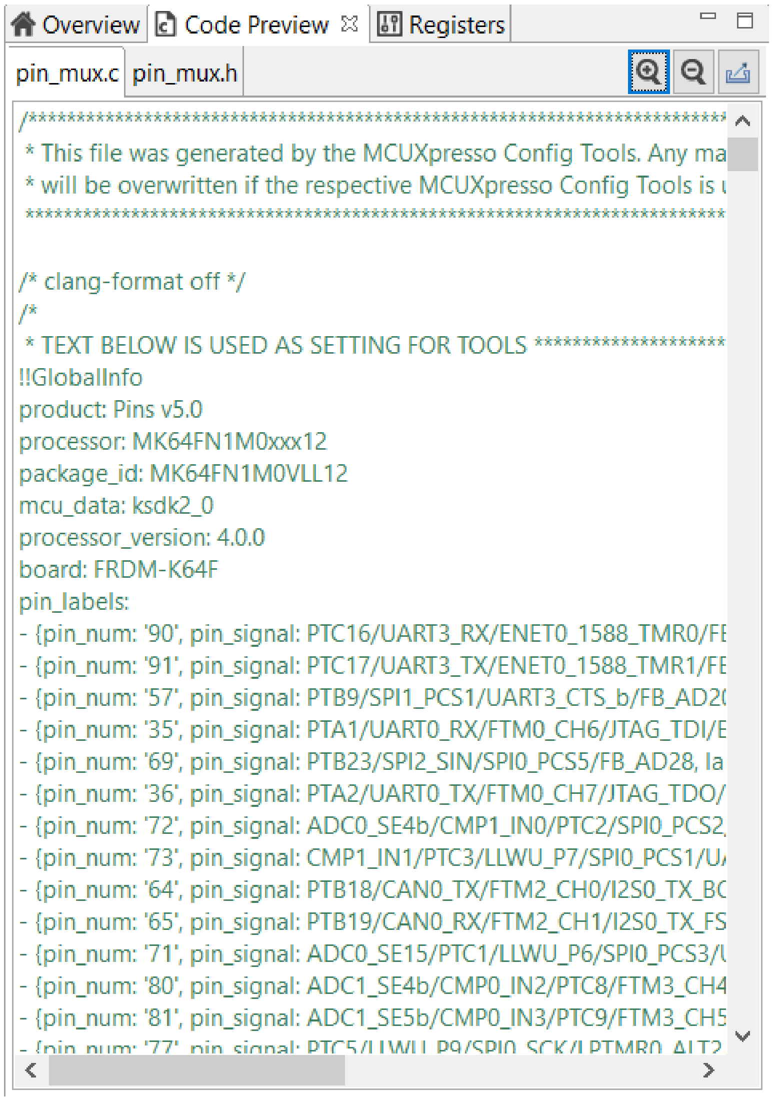
The tool generates source code that can be incorporated into an application to initialize pins routing. The source code is generated automatically on change or can be generated manually by selecting **Pins** > **Refresh** from the main menu. The generated code is shown in the **Code Preview** view. It shows all generated files and each file has its own tab.

For multicores, the sources are generated for each core. Appropriate files are shown with @Core #{number} tag.

NOTE

The tag name may be different depending on the selected multi-core processor family/type.

You can also copy and paste the generated code into the source files. The view generates code for each function. In addition to the function comments, the tool configuration is stored in a YAML format. This comment is not intended for direct editing and can be used later to restore the pins configuration.



```
/*****************************************************************************
 * This file was generated by the MCUXpresso Config Tools. Any ma
 * will be overwritten if the respective MCUXpresso Config Tools is u
 *****************************************************************************


/* clang-format off */
/*
 * TEXT BELOW IS USED AS SETTING FOR TOOLS *********************
!!GlobalInfo
product: Pins v5.0
processor: MK64FN1M0xxx12
package_id: MK64FN1M0VLL12
mcu_data: ksdk2_0
processor_version: 4.0.0
board: FRDM-K64F
pin_labels:
- {pin_num: '90', pin_signal: PTC16/UART3_RX/ENET0_1588_TMR0/FE
- {pin_num: '91', pin_signal: PTC17/UART3_TX/ENET0_1588_TMR1/FE
- {pin_num: '57', pin_signal: PTB9/SPI1_PCS1/UART3_CTS_b/FB_AD2
- {pin_num: '35', pin_signal: PTA1/UART0_RX/FTM0_CH6/JTAG_TDI/E
- {pin_num: '69', pin_signal: PTB23/SPI2_SIN/SPI0_PCS5/FB_AD28, la
- {pin_num: '36', pin_signal: PTA2/UART0_TX/FTM0_CH7/JTAG_TDO/
- {pin_num: '72', pin_signal: ADC0_SE4b/CMP1_IN0/PTC2/SPI0_PCS2
- {pin_num: '73', pin_signal: CMP1_IN1/PTC3/LLWU_P7/SPI0_PCS1/U
- {pin_num: '64', pin_signal: PTB18/CAN0_TX/FTM2_CH0/I2S0_TX_BC
- {pin_num: '65', pin_signal: PTB19/CAN0_RX/FTM2_CH1/I2S0_TX_FS
- {pin_num: '71', pin_signal: ADC0_SE15/PTC1/LLWU_P6/SPI0_PCS3/U
- {pin_num: '80', pin_signal: ADC1_SE4b/CMP0_IN2/PTC8/FTM3_CH4
- {pin_num: '81', pin_signal: ADC1_SE5b/CMP0_IN3/PTC9/FTM3_CH5
- {pin_num: '77', pin_signal: PTC5/LLWU_P9/SPI0_SCK/LPTMR0_ALT2
```

Figure 44. Generated code

YAML configuration contains configuration of each pin. It stores only non-default values.

TIP

For multicore processors, it will generate source files for each core. If processor is supported by SDK, it can generate BOARD_InitBootPins function call from main by default. You can specify "Call from BOARD_InitBootPins" for each function, in order to generate appropriate function call.

Chapter 4 Clocks Tool

The Clocks Tool configures initialization of the system clock (core, system, bus, and peripheral clocks) and generates the C code with clock initialization functions and configuration structures.

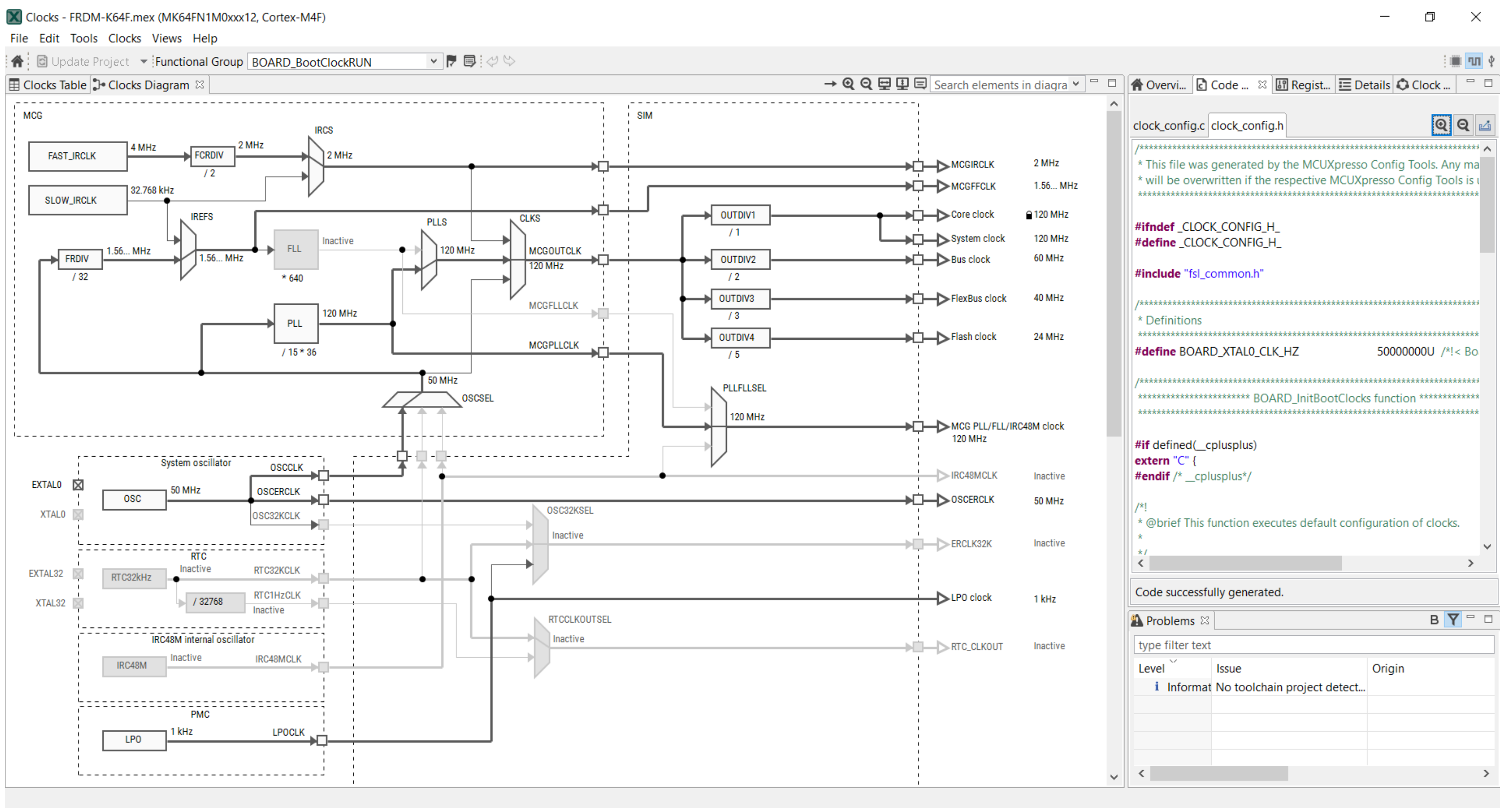
4.1 Features

Following are the features of the Clock Tool:

- Inspects and modifies element configurations on the clock path from the clock source up to the core/peripherals.
- Validates clock elements settings and calculates the resulting output clock frequencies.
- Generates a configuration code using the SDK.
- Modifies the settings and provides output using the table view of the clock elements with their parameters.
- Navigate, modify, and display important settings and frequencies easily in **Diagram** view.
- Edit detailed settings in Details view.
- Inspect the interconnections between peripherals and consuming clocks in Module Clocks view.
- Helps to find clock elements settings that fulfills given requirements for outputs.
- Fully integrated in tools framework along with other tools.
- Shows configuration problems in Problems view and guides the user for the resolution.

4.2 User interface overview

The Clocks Tool is integrated and runs with the MCUXpresso Config Tools framework. For documentation on the common interface and menu items, see the Config Tools User Interface chapter.



```
/*************************************************************************
 * This file was generated by the MCUXpresso Config Tools. Any ma
 * will be overwritten if the respective MCUXpresso Config Tools is u
 *************************************************************************

#ifndef _CLOCK_CONFIG_H_
#define _CLOCK_CONFIG_H_

#include "fsl_common.h"

/*************************************************************************
 * Definitions
 *************************************************************************
#define BOARD_XTAL0_CLK_HZ                 50000000U  /*!< Bo

/*************************************************************************
 ************************ BOARD_InitBootClocks function *************
 *************************************************************************

#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus*/

/*!
 * @brief This function executes default configuration of clocks.
 *
 */
```

Code successfully generated.

Level	Issue	Origin
Informat	No toolchain project detect...	

Figure 45. User interface

4.3 Clock configuration

Each clock configuration (functional group) lists the settings for the entire clock system and is a part of the global configuration stored in the MEX file. Initially, after the new clock configuration is created, it's set to reflect the default after-reset state of the processor.

There can be one or more clock configurations handled by the Clocks Tool. The default clock configuration is created with the name “*BOARD_BootClockRUN*”. Multiple configurations means multiple options are available for the processor initialization.

NOTE

All clock settings are stored individually for each clock configuration so that each clock configuration is configured independently.

Clocks configurations (functional groups) are presented at the top of the view. You can switch between them by selecting them from the dropdown menu.

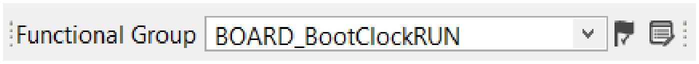

Functional Group BOARD_BootClockRUN

Figure 46. Default clock configuration

NOTE

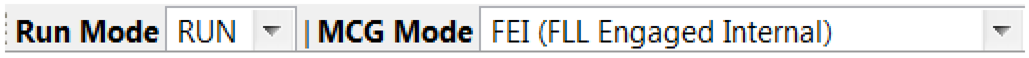
The code generation engine of the tool generates function with the name derived from the Clock configuration name.

4.4 Global settings

Global settings are settings that influence the entire clock system. It's recommended to set them first. Global settings can be modified in the **Clock Table** view.

NOTE

Global settings can be changed at any time.



Run Mode	RUN
MCG Mode	FEI (FLL Engaged Internal)

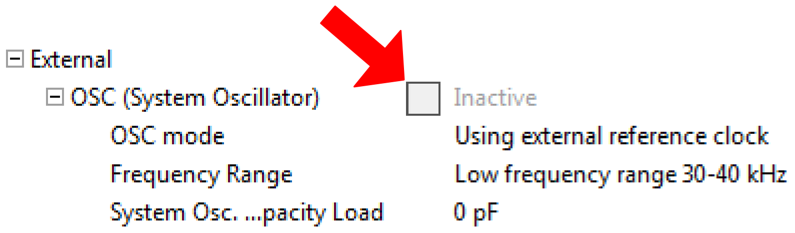
Figure 47. Global settings

4.5 Clock sources

The **Clock Sources** table is located in the **Clocks Table** view. You can also edit the clock sources directly from the **Diagram** view or from the **Details** view.

You can configure the availability of external clock sources (check the checkbox) and set their frequencies. Some sources can have additional settings available when you unfold the node.

If the external crystal or the system oscillator clock is available, check the checkbox in the clock source row and specify the frequency.



External	
OSC (System Oscillator)	Inactive
OSC mode	Using external reference clock
Frequency Range	Low frequency range 30-40 kHz
System Osc. ...pacity Load	0 pF

Figure 48. External clock source configuration

NOTE

Some clock sources remain inactive even though the checkbox is checked. This is because the clock sources functionality depends on other settings like power mode or additional enable/disable setting options. You can hover the cursor on the setting to see a tooltip with information on the element and possible limitations/options.

4.6 Setting states and markers

The following states, styles, and markers reflect the information shown in the settings' rows in the settings tables (clock sources, output, details or individual).

Table 5. Setting states and markers

State/Style/ Marker	Icon	Description
Error marker	☒	Indicates that there is an error in the settings or something related to it. See the tooltip of the setting for details.
Warning marker	⚠	Indicates that there is a warning in the settings or something related to it. See the tooltip of the setting for details.
Lock icon	🔒	Indicates that the settings (that may be automatically adjusted by the tool) are locked to prevent any automatic adjustment. If the setting can be locked, they are automatically locked when you change the value. To add/remove the lock manually, use the pop-up menu command **Lock/Unlock**. **NOTE** The clock element settings that cannot be automatically adjusted by the tool keep their value as is and do not allow locking. These are: clock sources, clock selectors and configuration elements.
Yellow background	100 MHz	Indicates that the field is directly or indirectly changed by the previous user action.
Gray text	FCTRIM	Indicates that the value of setting does not actively influence the clock. It is disabled or relates to an inactive clock element. For example, on the clock path following the unavailable clock source or disabled element. The frequency signal also show the text “inactive” instead of frequency. The value is also gray when the value is read-only. In such a state it is not possible to modify the value.

4.7 Frequency settings

The Clocks Tool instantly re-calculates the state of the entire clock system after each change of settings from the clock source up to the clock outputs.

The current state of all clock outputs is listed in the **Clock Outputs** view located on the right side of the clock sources. The displayed value can be:

- **Frequency** – Indicates that a clock signal is active and the output is fed with the shown frequency. The tool automatically chooses the appropriate frequency units. In case the number is too long or has more than three decimal places, it's shortened and only two decimal places are shown, followed by an ellipsis (‘…’), indicating that the number is longer.
- **“Inactive”** text – Indicates that no clock signal flows into the clock output or is disabled due to some setting.

If you have a specific requirement for an output clock, click on the frequency you would like to set, change it, and press **Enter**.

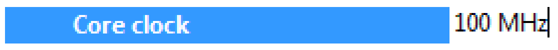

Core clock 100 MHz

Figure 49. Setting the core clock frequency

In case the tool has reached/attained the required frequency, it appears locked and is displayed as follows:

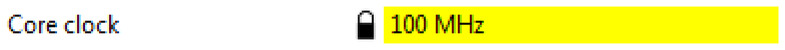

Core clock 🔒 100 MHz

Figure 50. Tool attains the required frequency

In case the tool is not able to reach/attain the required frequency or some other problem occurs, it's displayed as follows:

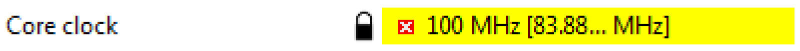


Figure 51. Tool encounters problem

The frequency value in square brackets [ ] indicates the value that the tool is actually using in the calculations instead of the value that has been requested.

NOTE

You can edit or set requirements only for the clock source and the output frequencies. The other values can be adjusted only when no error is reported.

4.7.1 Pop-up menu commands

- **Lock/Unlock** – Removes a lock on the frequency which enables the tool to change any valid value that satisfies all other requirements, limits, and constraints.
- **Find Near Valid Value** – Tries to find a valid frequency that lies near the specified value, in case the tool failed in reaching the requested frequency.

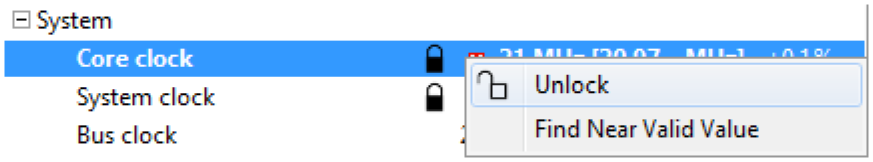
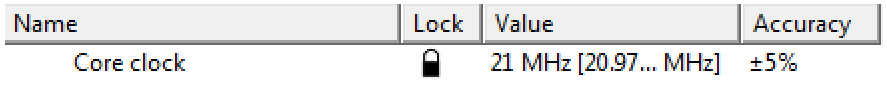


Figure 52. Pop-up menu commands

4.7.2 Frequency precision

For locked frequency settings (where user requests a specific value) the frequency precision value is also displayed. By default, the value is 0.1% but can be individually adjusted by clicking the value.

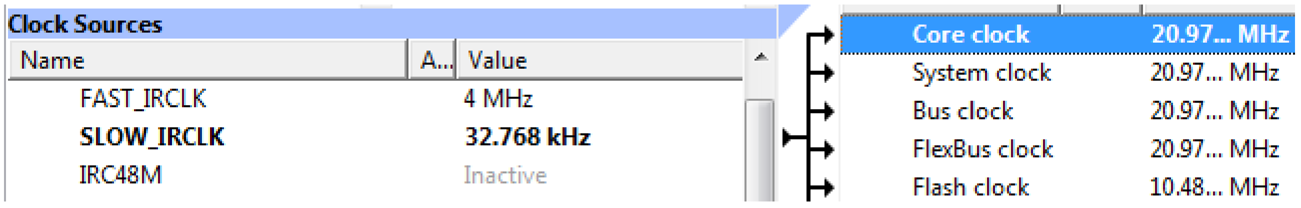


Name	Lock	Value	Accuracy
Core clock	🔒	21 MHz [20.97... MHz]	±5%

Figure 53. Frequency precision

4.8 Dependency arrows

In the **Table** view, the area between the clock sources and the clock output contains arrows directing the clock source to outputs. The arrows lead from the current clock source used for the selected output into all outputs that are using the signal from the same clock source. This identifies the dependencies and the influences when there is change in the clock source or elements on a shared clock path.



Clock Sources

Name	A...	Value
FAST_IRCLK		4 MHz
SLOW_IRCLK		**32.768 kHz**
IRC48M		Inactive

Core clock	20.97... MHz
System clock	20.97... MHz
Bus clock	20.97... MHz
FlexBus clock	20.97... MHz
Flash clock	10.48... MHz

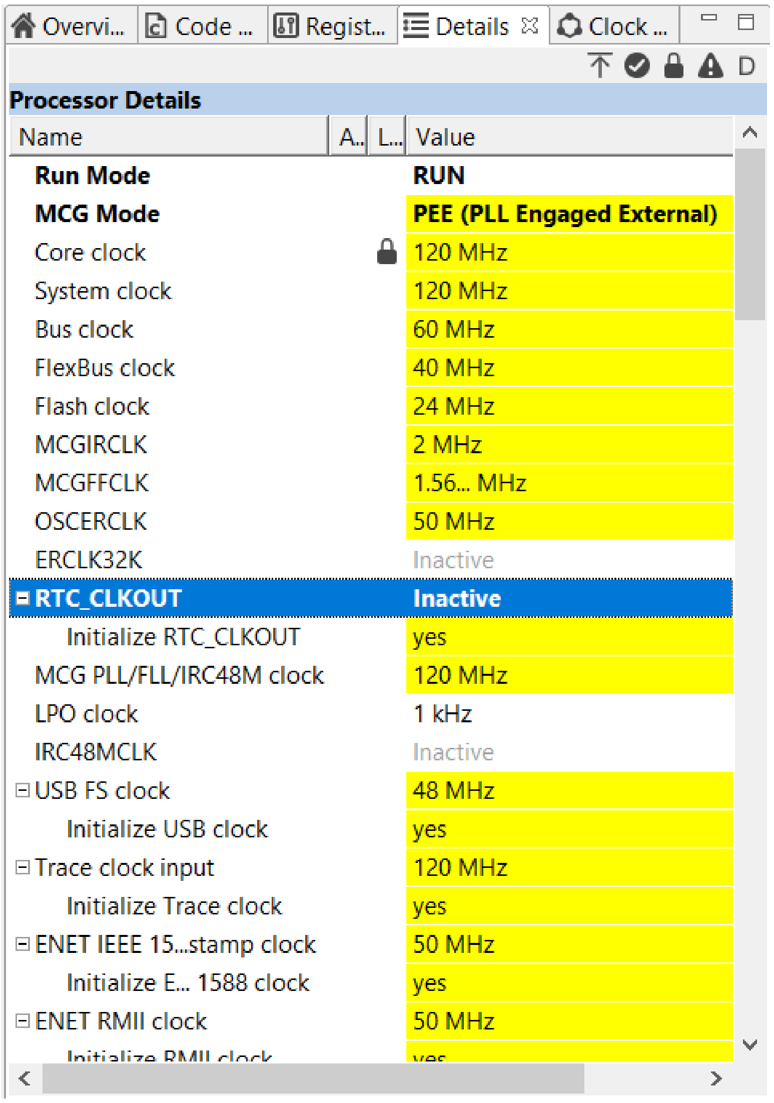
Figure 54. Dependency arrows

4.9 Details view

The **Details** view displays and allows you to change clock-element settings information.

The information is also updated in real-time based on any changes in the **Clocks Diagram** and **Clocks Table**.

•


Overvi... Code ... Regist... Details Clock ...

Processor Details

Name	A..	L...	Value
Run Mode			**RUN**
MCG Mode			**PEE (PLL Engaged External)**
Core clock		🔒	120 MHz
System clock			120 MHz
Bus clock			60 MHz
FlexBus clock			40 MHz
Flash clock			24 MHz
MCGIRCLK			2 MHz
MCGFFCLK			1.56... MHz
OSCERCLK			50 MHz
ERCLK32K			Inactive
⊟ **RTC_CLKOUT**			**Inactive**
Initialize RTC_CLKOUT			yes
MCG PLL/FLL/IRC48M clock			120 MHz
LPO clock			1 kHz
IRC48MCLK			Inactive
⊟ USB FS clock			48 MHz
Initialize USB clock			yes
⊟ Trace clock input			120 MHz
Initialize Trace clock			yes
⊟ ENET IEEE 15...stamp clock			50 MHz
Initialize E... 1588 clock			yes
⊟ ENET RMII clock			50 MHz
Initialize RMII clock			yes

Figure 55. Details view

Figure 56. Details view

In the **Details** view, you can perform the following actions:

- **Display clock-element information** - Point the mouse cursor at the clock element to display general clock-element information.
- **View the clock-element in Clocks Diagram or Clocks Table** - Left-click on a clock element to highlight it in the **Clocks Diagram** or **Clocks Table** views, depending on which is currently active.

- **View detailed clock-element information** - Double-click on a clock element to display element details, as well as highlight the element in **Clocks Diagram** or **Clocks Table**, depending on which is currently active. You can also view element details by clicking the **Open in new window** button in the upper right corner of the **Details** view.
- **Modify clock-element settings** - Left-click in the **Value** column to change clock element value, such as frequency, or select an option from the dropdown menu.
- **Lock/unlock clock elements** - Right-click on a clock element to lock/unlock the element.
- **Filter for active/locked/erroneous clock elements** - Use the buttons in the upper-right corner of the **Details** view to filter for active/locked/erroneous clock elements, or to remove all current filters.

4.10 Clock diagram

The clock diagram shows the structure of the entire clock model, including the clock functionality handled by the tool. It visualizes the flow of the clock signal from clock sources to clock output. It's dynamically refreshed after every change and always reflects the current state of the clock model.

At the same time it allows you to edit the settings of the clock elements.

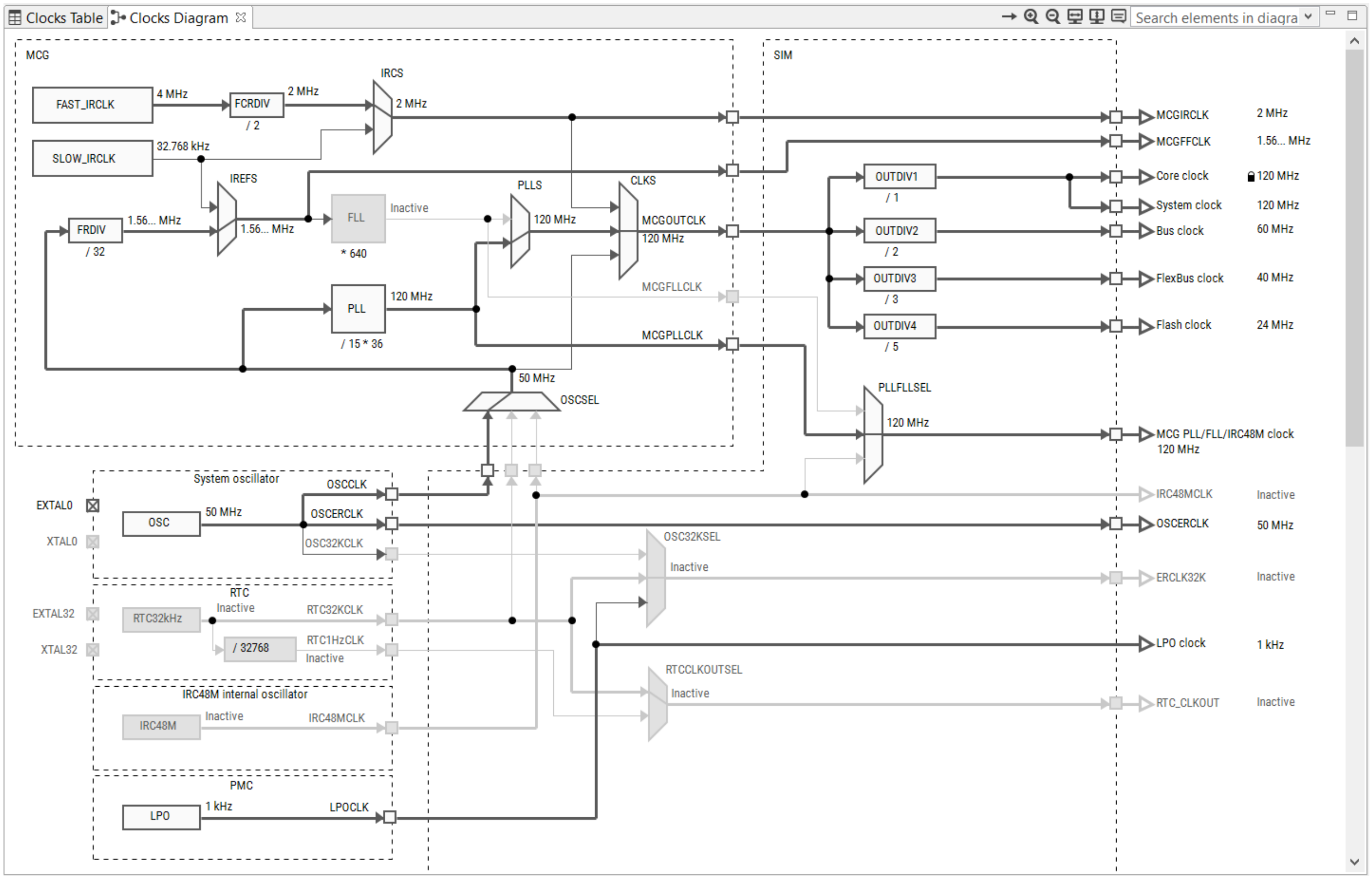


Figure 57. Clock diagram

4.10.1 Mouse actions in diagram

You can perform the following actions in the Clock diagram view.

- **Position the mouse cursor on the element** to see the tooltip with the information on the clock element such as status, description, output frequency, constraints, and enable/disable conditions.
- **Single-click on output frequency or scale** to change output frequency or scale.
- **Single-click on lock** to remove the lock.
- **Double-click on the element** to show its settings in the **Details** view (force to open the view if closed or not visible).
- **Single-click on the element** to show its settings in the **Details** view.
- **Single-click on a selected Clock source** to display a dropdown menu for enabling or disabling the source.
- **Single-click on a selected Clock selector** to display selector input options.

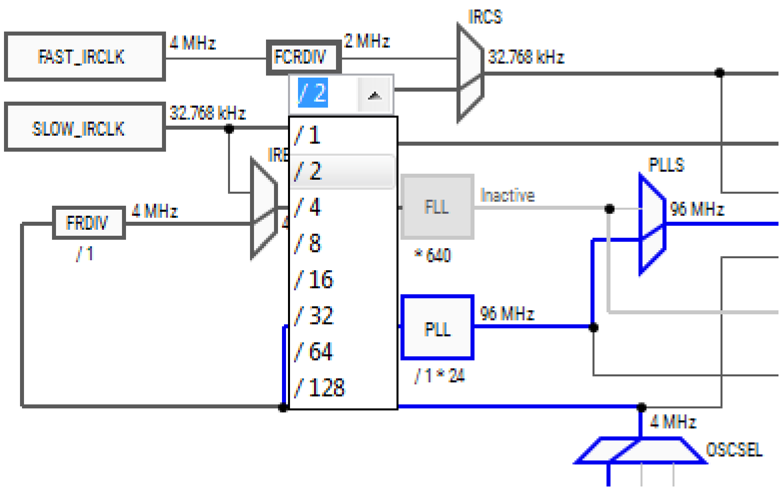
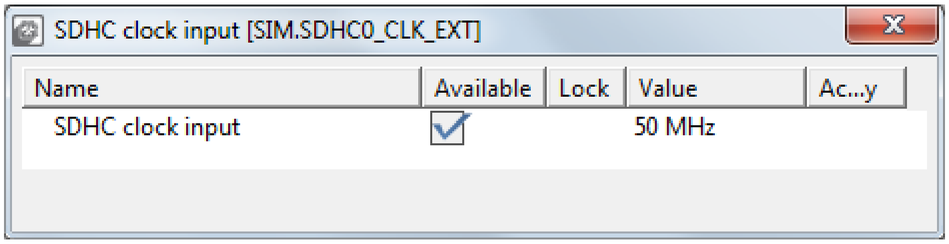


Figure 58. Clocks mouse actions in diagram

- **Right-click on the element**, **component, or clock output** to see a pop-up menu with the following options.
  - — **Edit settings of: {element}** – Invokes the floating view with the settings for a single element.
  - — **Edit all settings** – Invokes the floating view with all the settings for an element.
  - — **Edit settings on the path to: {clock output}** – Invokes the floating view with the settings for all elements on the clock path leading to the selected clock output.


SDHC clock input [SIM.SDHC0_CLK_EXT]

Name	Available	Lock	Value	Ac...y
SDHC clock input	✓		50 MHz	

Figure 59. Floating view

4.10.2 Color and line styles

Different color and line styles indicate different information for the element and clock signal paths.

The color and line styles can indicate:

- Active clock path for selected output
- Clock signal path states - used/unused/error/unavailable
- Element states – normal/disabled/error

To inspect colors and style appearance, select **Help** > **Show diagram legend** from the main menu.

4.10.3 Clock model structure

The clock model consists of interconnected clock elements. The clock signal flows from the clock sources through various clock elements to the clock outputs. The clock element can have specific enable conditions that can stop the signal from being passed to the successor. The clock element can also have specific constraints and limits that are watched by the Clocks Tool. To inspect these details, position the cursor on the element in the clock diagram to display the tooltip.

The following are the clock model elements.

- **Clock source** – Produces a clock signal of a specified frequency. If it's an external clock source, it can have one or more related pins.

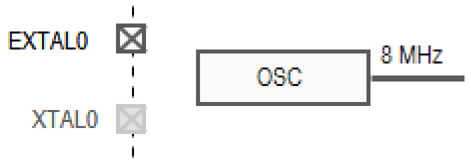


Figure 60. Clock source

- **Clocks selector (multiplexer) –** Selects one input from multiple inputs and passes the signal to the output.

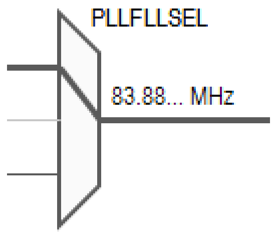


Figure 61. Clocks selector

- **Prescaler –** Divides or multiplies the frequency with a selectable or fixed ratio.

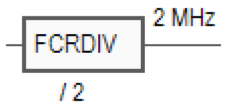


Figure 62. Prescaler

- **Frequency Locked Loop (FLL) –** Multiplies an input frequency with given factor.

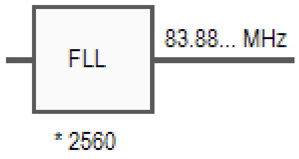


Figure 63. Frequency Locked Loop

- **Phase Locked Loop (PLL) –** Contains pre-divider and thus is able to divide/multiply with a given value.

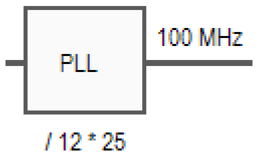


Figure 64. Phase Locaked Loop

- **Clock gate –** Stops the propagation of incoming signal.
- **Clock output** – Marks the clock signal output that has some name and can be further used by the peripherals or other parts of the processor. You can put a lock and/or frequency request.

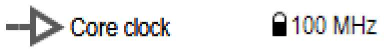


Figure 65. Clock output

- **Clock component –** Group of clock elements surrounded with a border. The clock component can have one or more outputs. The clock component usually corresponds to the processor modules or peripherals. The component output may behave like clock gates, allowing or preventing the signal flow out of the component.

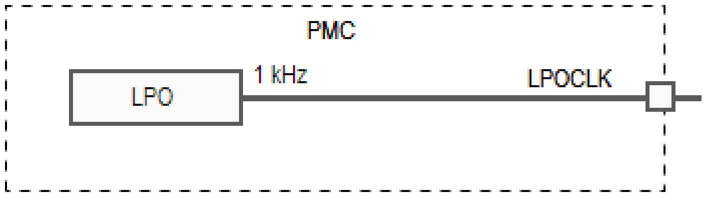


Figure 66. Clock component

- **Configuration element** – Additional setting of an element. Configuration elements do not have graphical representation in the diagram. They are shown in the setting table for the element or the clock path the element is on.

4.11 Main menu

Commands related to the Clocks Tool can be found in the **Clocks** menu and include the following commands:

- **Functional groups** – Invokes the **Functional group properties** dialog.
- **Refresh** – Refreshes each clocks configuration with explicit invocation of code generation.
- **Reset To Board Defaults** – Resets the clock model to board defaults.
- **Reset To Processor Defaults** – Resets the clock model ito processor defaults.
- **Unlock All Settings** – Unlocks all locks in all settings.

4.12 Troubleshooting problems

It's possible that problems or conflicts occur while working with the Clocks Tool. Such problems and the overall status are indicated in red on the central status bar of the Clocks Tool. The status bar displays global information on the reported problem.

You may encounter any of the following problems:

1. **Requirement(s) not satisfiable:** Indicates that there are one or more locked frequency or frequency constraints for which the tool is not able to find a valid settings and satisfy those requirements.
2. **Invalid settings or requirements:** [*element list*] – Indicates that the value of a settings is not valid. For example: The current state of settings is beyond the acceptable range.

The following are some tips to troubleshoot encountered problems.

1. Find the elements and settings with marked errors in the diagram or tables and see the details in the tooltip.
2. Start with only one locked frequency and let the tool find and calculate other ones. After you are successful you can add more.
3. Go through the locked outputs, if there are any, and verify the requirements (possible errors in the required frequency, wrong units, and so on).
4. If you are OK to have a near around of the requested value, right-click and from the pop-up menu select **Clock output > Find near value**.
5. If you cannot reach the values you need, see the clock paths leading to the clock output you want to adjust and check the selectors if it's possible to switch to another source of clock.

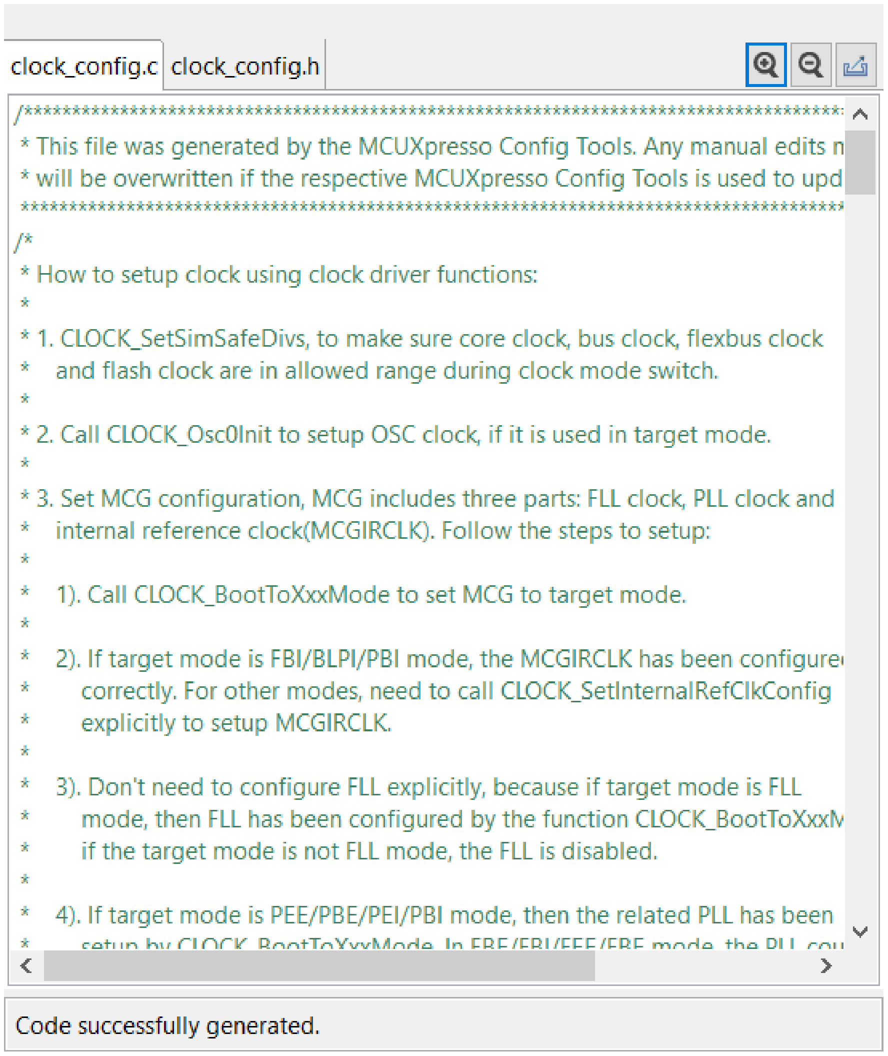
6. Try to remove locks by selecting **Clocks > Unlock All Settings**. In case many changes are required, you can simply reset the model to the default values and start from the beginning. To reset, select **Clocks > Reset to processor defaults**.

You can resolve most of the reported problems using the **Problems** view. Each problem is listed as a separate row. The following options appear when you right-click on a selected row in the **Problems** view.

- **Show problem** - Shows the problem in the **Clocks Diagram** view If one the solutions are possible then the pop up is extended by:
  - **Remove lock** - Removes the lock from erroneous element.
  - **Find Near value** - Finds the nearest value.

4.13 Code generation

If the settings are correct and no error is reported, the tool's code generation engine instantly re-generates the source code. The resulting code is found in the **Code Preview** view.



```
/*************************************************************************************
 * This file was generated by the MCUXpresso Config Tools. Any manual edits n
 * will be overwritten if the respective MCUXpresso Config Tools is used to upd
 ************************************************************************************
/*
 * How to setup clock using clock driver functions:
 *
 * 1. CLOCK_SetSimSafeDivs, to make sure core clock, bus clock, flexbus clock
 *    and flash clock are in allowed range during clock mode switch.
 *
 * 2. Call CLOCK_Osc0Init to setup OSC clock, if it is used in target mode.
 *
 * 3. Set MCG configuration, MCG includes three parts: FLL clock, PLL clock and
 *    internal reference clock(MCGIRCLK). Follow the steps to setup:
 *
 *    1). Call CLOCK_BootToXxxMode to set MCG to target mode.
 *
 *    2). If target mode is FBI/BLPI/PBI mode, the MCGIRCLK has been configure
 *        correctly. For other modes, need to call CLOCK_SetInternalRefClkConfig
 *        explicitly to setup MCGIRCLK.
 *
 *    3). Don't need to configure FLL explicitly, because if target mode is FLL
 *        mode, then FLL has been configured by the function CLOCK_BootToXxxM
 *        if the target mode is not FLL mode, the FLL is disabled.
 *
 *    4). If target mode is PEE/PBE/PEI/PBI mode, then the related PLL has been
 *        setup by CLOCK_BootToXxxMode. In FBE/FBI/FEE/FBE mode, the PLL cou
```

Code successfully generated.

Figure 67. Code Preview view

4.13.1 Working with the code

The generated code is aligned with the SDK. To use the code with the SDK project it's necessary to transfer the code into your project structure.

To transfer the code into your project:

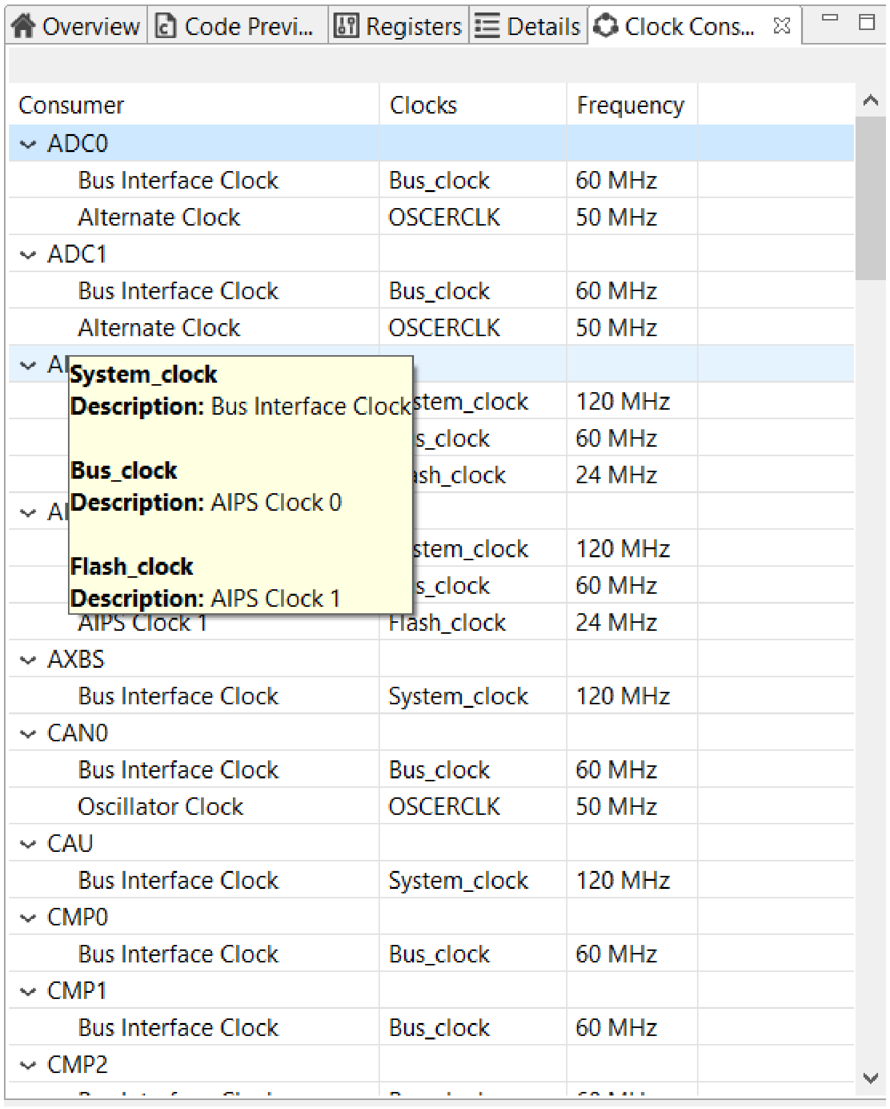
- Copy the content using the COPY command, either by pressing the CTRL+C keys or the pop-up menu after the whole text is selected.
- Use export command.
- Click the **Export** button in **Code Preview** view.
- Click **Update Project Code** in the main toolbar (works only for toolchain project).

4.14 Clock Consumers view

The **Clock Consumers** view provides an overview of peripheral instances. It also provides information on clock-clock instance pairing. This view is not editable and is for information only.

NOTE

Information about which peripherals are consuming which output clock is available in the clock output tooltip.


Overview | Code Previ... | Registers | Details | Clock Cons...

Consumer	Clocks	Frequency
ADC0		
Bus Interface Clock	Bus_clock	60 MHz
Alternate Clock	OSCERCLK	50 MHz
ADC1		
Bus Interface Clock	Bus_clock	60 MHz
Alternate Clock	OSCERCLK	50 MHz
AI		
	stem_clock	120 MHz
	s_clock	60 MHz
	ish_clock	24 MHz
AI		
	stem_clock	120 MHz
	s_clock	60 MHz
AIPS Clock 1	Flash_clock	24 MHz
AXBS		
Bus Interface Clock	System_clock	120 MHz
CAN0		
Bus Interface Clock	Bus_clock	60 MHz
Oscillator Clock	OSCERCLK	50 MHz
CAU		
Bus Interface Clock	System_clock	120 MHz
CMP0		
Bus Interface Clock	Bus_clock	60 MHz
CMP1		
Bus Interface Clock	Bus_clock	60 MHz
CMP2		

System_clock
Description: Bus Interface Clock

Bus_clock
Description: AIPS Clock 0

Flash_clock
Description: AIPS Clock 1

Figure 68. Clock Consumers view

Chapter 5 Peripherals Tool

5.1 Features

The Peripherals Tool features:

- Configuration of initialization for SDK drivers
- User friendly user interface allowing to inspect and modify settings
- Smart configuration component selection along the SDK drivers used in toolchain project
- Instant validation of basic constraints and problems in configuration
- Generation of initialization source code using SDK function calls
- Multiple functional-group support for initialization alternatives
- Configuration problems are shown in Problems view and marked with decorators in other views
- Integration in MCUXpresso Config Tools framework along with other tools
- Middleware configuration support (USB)

5.2 Basic Terms and Definitions

The following are the basic terms and definitions used in the chapter:

- Functional group - represents a group of peripherals that are initialized as a group. The tool generates a C function for each functional group that contains the initialization code for the peripheral instances in this group. Only one functional group can be selected as default initialization, the others are treated as alternatives that are not initialized by default.
- Peripheral instance – occurrence of a peripheral (device) of specific type. For example, UART peripheral has three instances on the selected processor, so there are UART0, UART1 and UART2 devices.
- Configuration component – provides user interface for configuring SDK software component (for example, peripheral driver) and generates code for its initialization.
- Component instance – configuration component can have multiple instances with different settings. (for example, for each peripheral instance like UART0, UART1).
- Component mode – specific use-case of the component instance (for example, TRANSFER mode of DSPI, or interrupt-based mode of communication).

5.3 Workflow

The following steps briefly describe the basic workflow in the Peripherals Tool.

1. In the **Peripherals view**, select the peripheral instance you would like to configure (use the checkbox).
2. In case more components are available for use by the peripheral, the **Select component** dialog appears. The **Select component** dialog shows the list of suitable configuration components for the selected peripheral matching the SDK driver for the selected processor.
3. Select the component you want to use and click **OK** to confirm.

4. In the settings editor that automatically opens, select the **Component mode** that you would like to use and configure individual settings.

NOTE

The selection of the component mode may impact appearance of some settings. Therefore, the selection of the mode should be always the first step.

5. Open the **Code Preview** view and see the output source code.

NOTE

Note: The source code preview is automatically generated after each change if no error is reported.

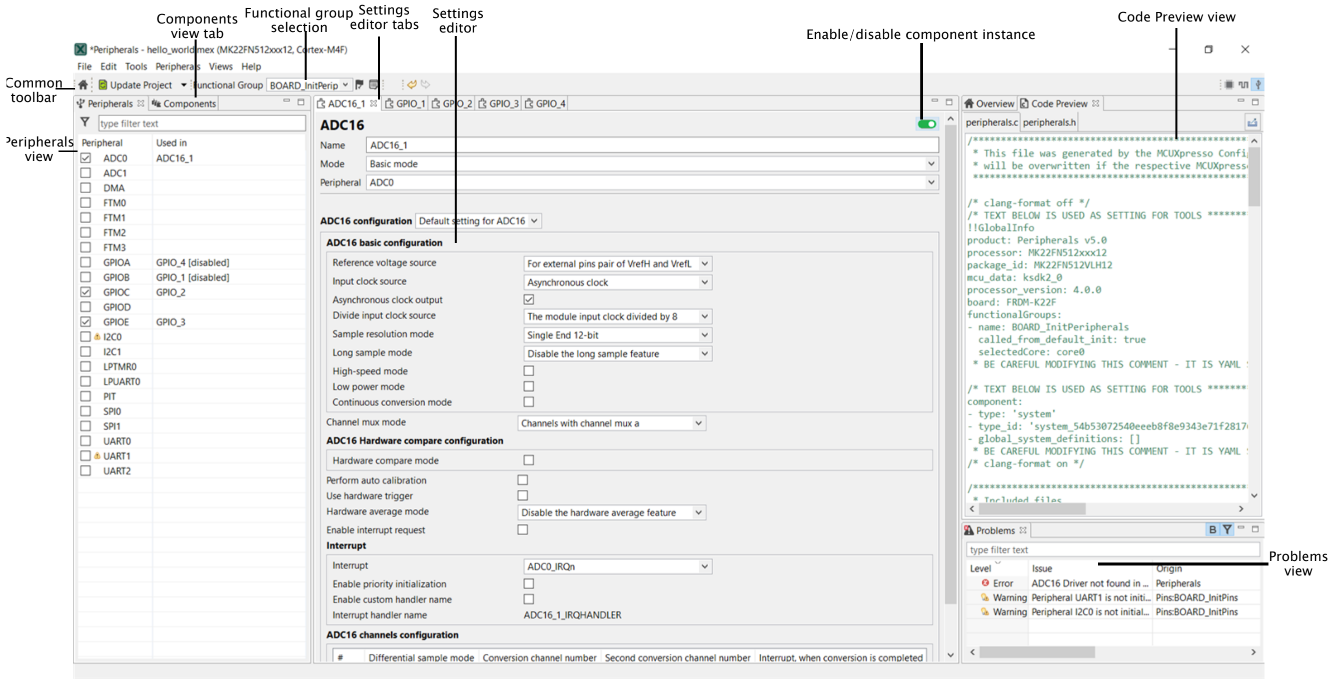
6. In case you are using a toolchain project, you can use **Update project** command from the toolbar. If not, you can export the source code by selecting **File>Export...** from the **Main Menu**.

NOTE

Note: To export the source code, you can also click the **Export** button in the **Code Preview** view.

7. Settings can be saved in a MEX format (used for all settings of all tools) by selecting **File>Save** from the **Main Menu**.

5.4 User interface overview



Peripheral	Used in
☑ ADC0	ADC16_1
☐ ADC1	
☐ DMA	
☐ FTM0	
☐ FTM1	
☐ FTM2	
☐ FTM3	
☐ GPIOA	GPIO_4 [disabled]
☐ GPIOB	GPIO_1 [disabled]
☑ GPIOC	GPIO_2
☐ GPIOD	
☑ GPIOE	GPIO_3
☐ I2C0	
☐ I2C1	
☐ LPTMR0	
☐ LPUART0	
☐ PIT	
☐ SPI0	
☐ SPI1	
☐ UART0	
☐ UART1	
☐ UART2	

ADC16

Name: ADC16_1
Mode: Basic mode
Peripheral: ADC0

ADC16 configuration: Default setting for ADC16

ADC16 basic configuration
- Reference voltage source: For external pins pair of VrefH and VrefL
- Input clock source: Asynchronous clock
- Asynchronous clock output: ☑
- Divide input clock source: The module input clock divided by 8
- Sample resolution mode: Single End 12-bit
- Long sample mode: Disable the long sample feature
- High-speed mode: ☐
- Low power mode: ☐
- Continuous conversion mode: ☐

Channel mux mode: Channels with channel mux a

ADC16 Hardware compare configuration
- Hardware compare mode: ☐

Perform auto calibration: ☐
Use hardware trigger: ☐
Hardware average mode: Disable the hardware average feature
Enable interrupt request: ☐

Interrupt
- Interrupt: ADC0_IRQn
- Enable priority initialization: ☐
- Enable custom handler name: ☐
- Interrupt handler name: ADC16_1_IRQHANDLER

ADC16 channels configuration

#	Differential sample mode	Conversion channel number	Second conversion channel number	Interrupt, when conversion is completed

```
/*************************************************
 * This file was generated by the MCUXpresso Confi
 * will be overwritten if the respective MCUXpress
 *************************************************

/* clang-format off */
/* TEXT BELOW IS USED AS SETTING FOR TOOLS *******
!!GlobalInfo
product: Peripherals v5.0
processor: MK22FN512xxx12
package_id: MK22FN512VLH12
mcu_data: ksdk2_0
processor_version: 4.0.0
board: FRDM-K22F
functionalGroups:
- name: BOARD_InitPeripherals
  called_from_default_init: true
  selectedCore: core0
 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML

/* TEXT BELOW IS USED AS SETTING FOR TOOLS *******
component:
- type: 'system'
- type_id: 'system_54b53072540eeeb8f8e9343e71f2817
- global_system_definitions: []
 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML
/* clang-format on */

/*************************************************
 * Included files
```

Level	Issue	Origin
Error	ADC16 Driver not found in ...	Peripherals
Warning	Peripheral UART1 is not initi...	Pins:BOARD_InitPins
Warning	Peripheral I2C0 is not initial...	Pins:BOARD_InitPins

Figure 69. User interface

5.5 Common toolbar

The common toolbar provides access to commands and selections that are available in context of all MCUXpresso Config Tools. It offers the following items:

- **Update project code** – this button opens update dialog allowing to update generated peripheral initialization code directly within specified toolchain project. This command is available only when the toolchain project has been specified.
- **Functional group selection –** Functional group in the Peripherals Tool represents a group of peripherals that are initialized as a group. The tool generates a C function for each function group that contains the initialization code.
- Function group related icons
  - **Call from default initialization** – sets the current functional group to be initialized by the default initialization function.
  - **Functional group properties** – opens the **Functional group properties** dialog to modify name and other properties of the function group
- **Tool switching icons** – section containing icons of individual tools. Click these icons to switch the currently visible tool.

NOTE

For details on other commands, refer Toolbar

5.6 Documentation view

You can display component-specific documentation by opening the **Documentation** view.

To open the **Documentation** view, do the following:

- In the **Peripherals** view, right-click the row and choose **Documentation** from the list.
- In the **Components** view, right-click the component and choose **Documentation** from the list.
- In the **Settings Editor**, click the **Documentation** button next to component name.

5.7 Peripherals view

The **Peripherals** view contains a table showing a list of available peripherals on the currently selected processor that can be configured by the Peripherals Tool. In case of multicore processors, the displayed peripherals are also core-specific.

Each instance of a peripheral (for example, UART0) occupies one row. First column contains peripheral name and a checkbox indicating whether the peripheral is used by any component instance.

De-selecting a previously-selected instance disables it. You can enable it by selecting the checkbox, or by clicking the switch in the settings editor of the component instance.

Second column contains a name of component instance handling the peripheral. This name is freely customizable in the settings editor and it is used in generated code.

Double-click on the second column opens the editor for the component instance.

5.8 Components view

The components view shows a list of configuration components, sorted by type into **Middleware/Peripheral drivers/Other**. The view displays configuration components differently based on their status:

- **Enabled** - Highlights the configuration component in dark gray.
- **Enabled/with warning** - Highlights the configuration component in dark gray with the alert symbol.
- **Enabled/with error** - Highlights the configuration component in red with the error symbol.

- **Disabled** - Highlights the configuration component in light gray.

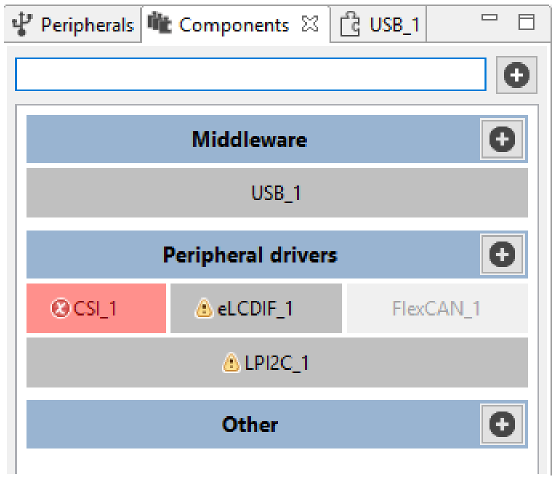


Figure 70. Components view

In the **Components** view, you can perform the following actions:

- **Display configuration-component information** - Point the mouse cursor at the configuration component to display general configuration-component information.
- **Open the Settings Editor of the configuration component** - Left-click the configuration component to open its **Settings Editor**.
- **View Configuration component documentation** - Right-click the configuration component and choose **Documentation** from the dropdown menu to view configuration-component documentation. If the configuration component isn't documented, the option is highlighted in gray.
- **Remove the component from configuration** - Right-click the configuration component and choose **Remove** from the dropdown menu.

NOTE

If the component has any global settings, a dialog appears prompting you to confirm the removal. If the component doesn't have any global settings, the component is deleted after removing the last instance.

- **Enable/disable the configuration component** - Right-click the configuration component and choose **Enable** or **Disable** to enable/disable the configuration component.
- **Move the configuration component to another functional group** - Right-click the configuration component and choose **Move to** to select from a list of functional groups to move the configuration component to.
- **Copy the configuration component to another functional group** - Right-click the configuration component and choose **Copy to** to select from a list of functional groups to copy the configuration component to.
- **Add new configuration components** - Left-click the **+** button and choose from the list to add a new component. You can filter the list to show only toolchain-project-relevant, or latest version components. You can also click the **+** buttons next to **Middleware/Peripheral drivers/Other** categories to add new components in them directly.
- **Filter configuration components by name** - Type a text string to filter configuration component names in the search bar.

5.9 Settings Editor

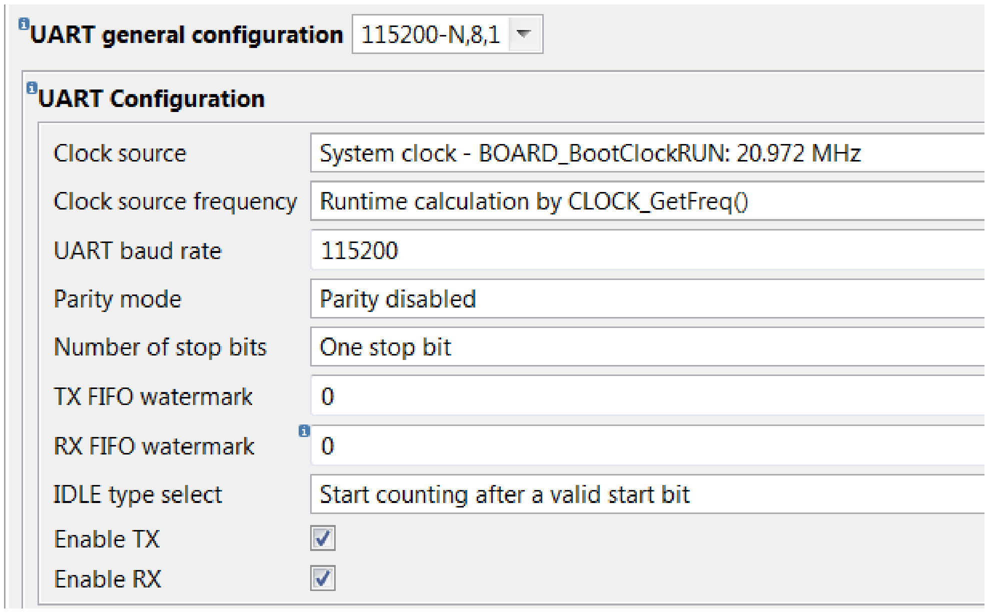
You can edit peripheral component settings in the **Settings Editor**. Open editors are shown in the central area of the screen, each with its own tab. Multiple editors can be opened at the same time. Changes done in the editor are immediately applied and kept even if the settings editor is closed. Settings that are disabled are highlighted in gray. In case that a component instance is disabled, all settings are highlighted in gray. Tooltips are displayed for all enabled settings when the mouse cursor is placed at settings.

To open **Settings Editor**, do the following:

- Double-click the component instance in the **Peripherals** or **Components** view to display component instance settings.
- Double-click the component in the **Components** view to display global settings of the component.

5.9.1 Quick selections

Settings are grouped to larger groups (config sets) that may provide presets with typical values. The user can use these presets to quickly set the desired typical combination of settings or return to the default state.



UART general configuration: 115200-N,8,1

UART Configuration

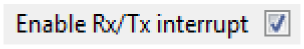
Clock source	System clock - BOARD_BootClockRUN: 20.972 MHz
Clock source frequency	Runtime calculation by CLOCK_GetFreq()
UART baud rate	115200
Parity mode	Parity disabled
Number of stop bits	One stop bit
TX FIFO watermark	0
RX FIFO watermark	0
IDLE type select	Start counting after a valid start bit
Enable TX	✓
Enable RX	✓

Figure 71. Quick selection example

5.9.2 Settings

The following settings occur in the editor.

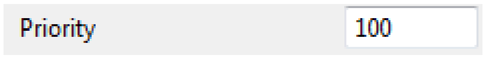
- **Boolean** – Two state setting (yes/no, true/false).



Enable Rx/Tx interrupt ✓

Figure 72. Boolean setting example

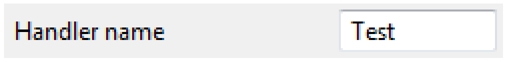
- **Integer, Float** – Integer or float number.



Priority: 100

Figure 73. Integer/Float setting example

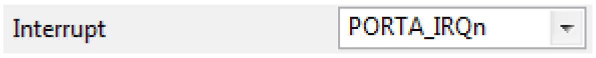
- **String** – Textual input.



Handler name: Test

Figure 74. String setting example

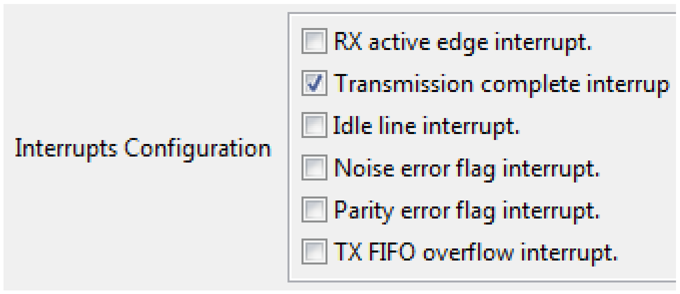
- **Enumeration** – Selection of one item from list of values.



Interrupt: PORTA_IRQn

Figure 75. Enumeration setting example

- **Set** – List of values, multiple of them can be selected.

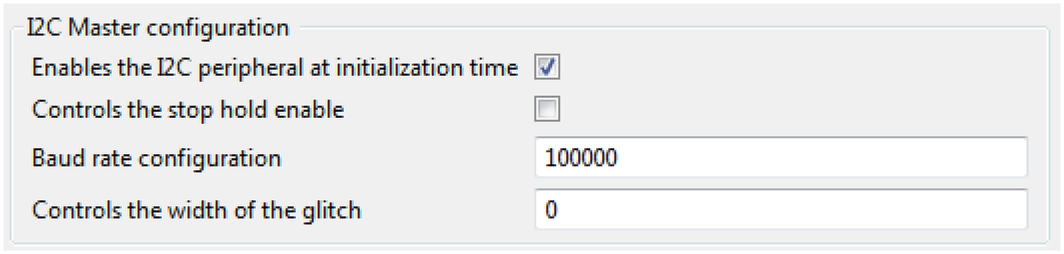


Interrupts Configuration

- [ ] RX active edge interrupt.
- [✓] Transmission complete interrup
- [ ] Idle line interrupt.
- [ ] Noise error flag interrupt.
- [ ] Parity error flag interrupt.
- [ ] TX FIFO overflow interrupt.

Figure 76. Set setting example

- **Structure** – Group of multiple settings of different types, may contain settings of any type including nested structures.



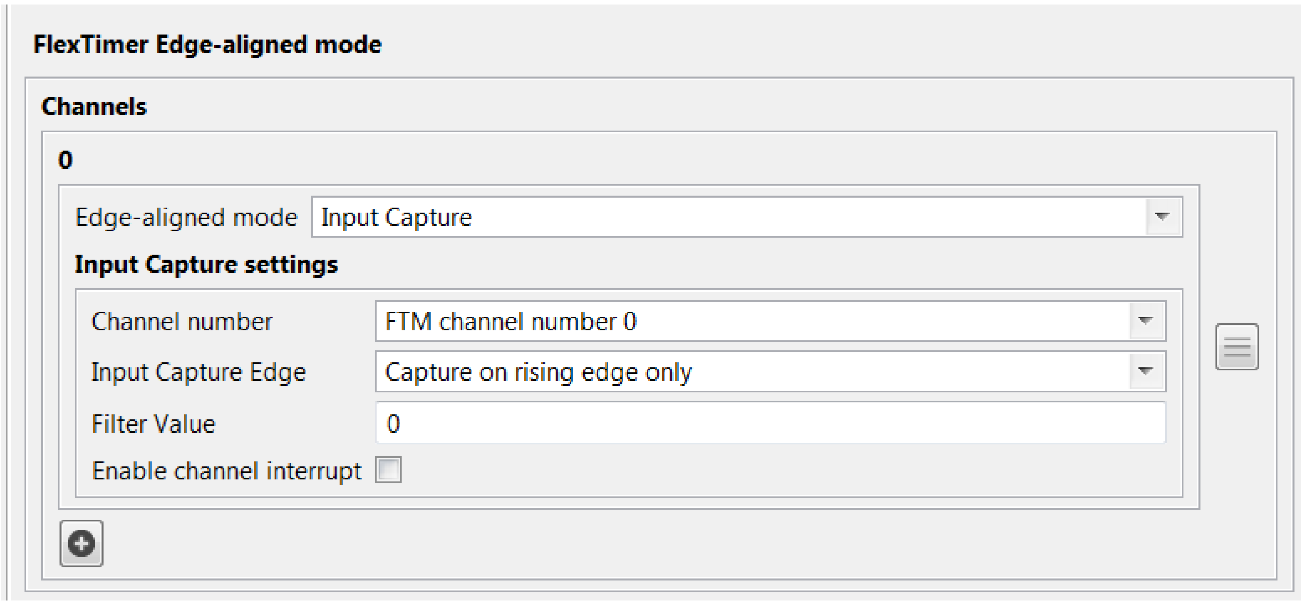
I2C Master configuration

Enables the I2C peripheral at initialization time	✓
Controls the stop hold enable	☐
Baud rate configuration	100000
Controls the width of the glitch	0

Figure 77. Structure setting example

- **Array** – Array of multiple settings of same type – user can add/remove items. The array of simple structures may also be represented as a table grid.

  The '+' button adds a new item at the end of array. To rearrange the position or delete an item, click on the menu icon and select one of the following options: Move up, Move down, Move to top, Move to bottom, or Remove.



FlexTimer Edge-aligned mode

Channels

0

Edge-aligned mode: Input Capture

Input Capture settings

Channel number	FTM channel number 0
Input Capture Edge	Capture on rising edge only
Filter Value	0
Enable channel interrupt	☐

Figure 78. Array setting example

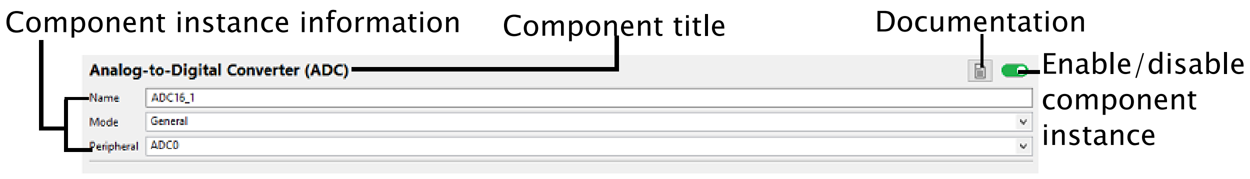
- **Info** – Read-only information for the user.

Resulting input clock frequency 1 kHz

Figure 79. Array setting example

5.9.3 Settings Editor header

All components share the **Settings Editor** header. In the header, you can view and change component information, enable or disable the component, and view component documentation (where applicable).


Component instance information
Component title
Documentation
Enable/disable component instance
Analog-to-Digital Converter (ADC)

Name ADC16_1

Mode General

Peripheral ADC0

Figure 80. Settings Editor header

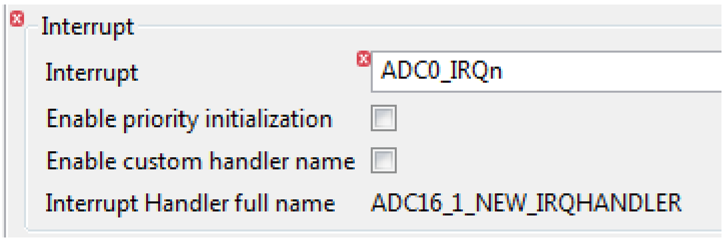
Settings Editor header contains the following:

- **Title** - Displays the configuration component title.
- **Name** - Displays the component instance name. This name is used in the generated code in constants and function identifiers. You can change it at any time.
- **Mode** - Displays the required usage for the component instance and influences available settings. Use the dropdown menu to change the mode (where applicable).
- **Peripheral** - Displays the name of the peripheral to be associated with the component instance. Use the dropdown menu to change it.
- **Enable/disable component instance switch** - Use the switch to enable or disable selected component instance. Note that by disabling the instance, you don't remove it from the tools configuration, but prevent its inclusion in the generated code.
- **Documentation button** - Click the button to view configuration component-specific documentation in the **Documentation** view. Note that not all configuration components are documented, therefore not all setting headers contain the **Documentation** icon.

5.10 Problems

The tool validates the settings and problems and errors are reported in the Problems view.

If there is an error related to the setting or component an error decorator is shown next to the element containing an error.


Interrupt

Interrupt ADC0_IRQn

Enable priority initialization ☐

Enable custom handler name ☐

Interrupt Handler full name ADC16_1_NEW_IRQHANDLER

Figure 81. Error decorators

5.11 Code generation

The code generation is performed automatically after every change in the configuration.

The Peripherals Tool produces the following C files:

- peripherals.c
- peripherals.h

NOTE

For multicore processors the peripherals.c/.h are generated for each core, containing functional groups associated with that core. This can be configured in functional group properties.

These files contain initialization code for peripherals produced by selected configuration components including:

- Constants and functions declaration in header file.
- Initialized configuration structures variables (constants).
- Global variables for the user application that are used in the initialization. For example, handles and buffers.
- Initialization function for each configuration component.
- Initialization function for each functional group. The name of the function is the same as the functional group name. These functions include execution of all assigned components' initialization functions.
- Default initialization function containing call to the function initializing the selected functional group of peripherals.

NOTE

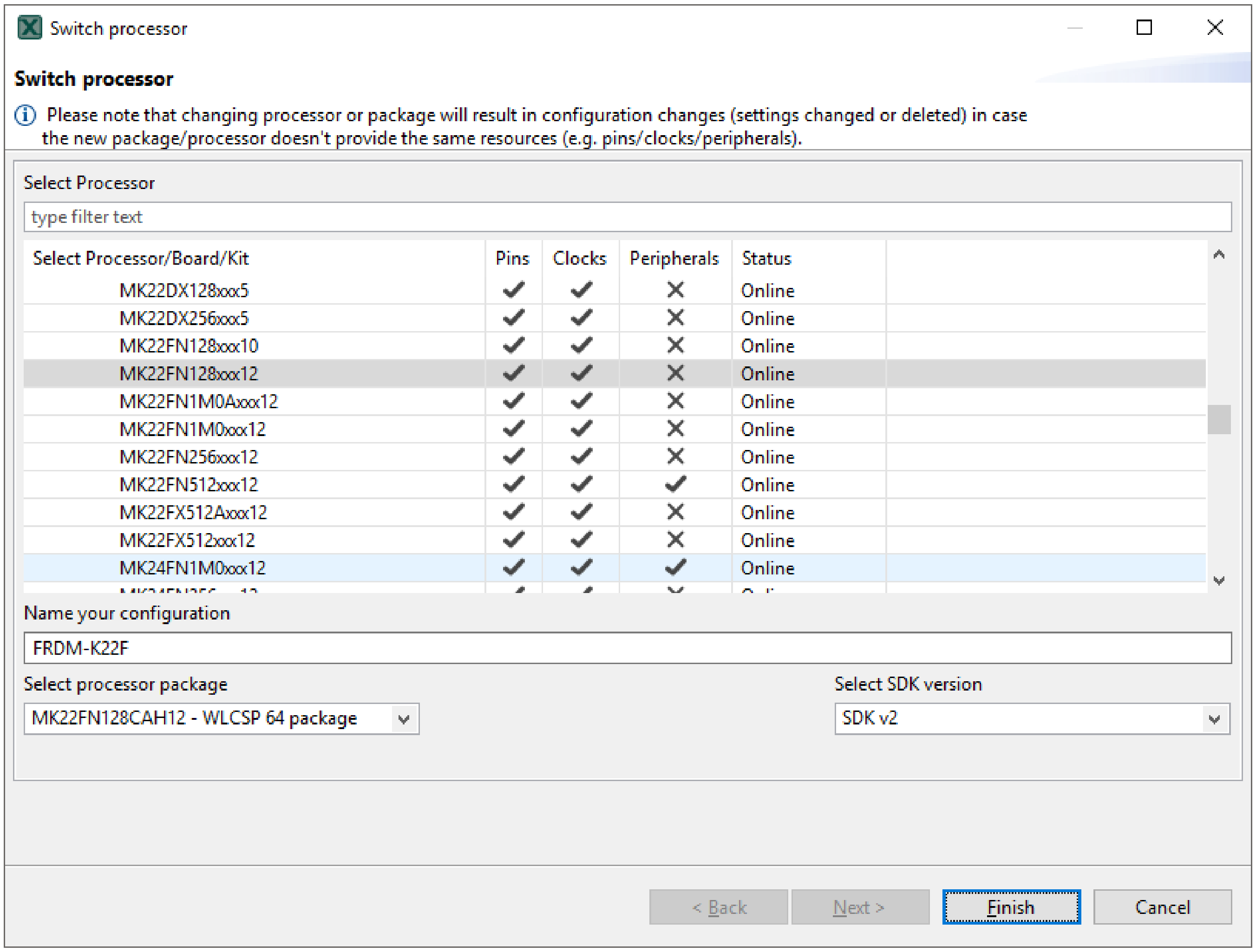
The prefixes of the global definitions (defines, constants, variables and functions) can be configured in the Properties of the functional group.

Chapter 6
Advanced Features

6.1 Switching the processor

You can switch the processor or the package of the current configuration to a different one. However, switching to a completely different processor may lead to problems, such as inaccessible pin routing or unsatisfiable clock-output frequency. In that case, it's necessary to fix the problem manually. For example, go to the Pins Routing table and re-configure all pins which report an error or conflict. Alternatively, you may need to change the required frequencies on Clock output.

Select **File** > **Switch processor** menu to change the processor in the selected configuration.



Switch processor

Please note that changing processor or package will result in configuration changes (settings changed or deleted) in case the new package/processor doesn't provide the same resources (e.g. pins/clocks/peripherals).

Select Processor

type filter text

Select Processor/Board/Kit	Pins	Clocks	Peripherals	Status
MK22DX128xxx5	✓	✓	✗	Online
MK22DX256xxx5	✓	✓	✗	Online
MK22FN128xxx10	✓	✓	✗	Online
MK22FN128xxx12	✓	✓	✗	Online
MK22FN1M0Axxx12	✓	✓	✗	Online
MK22FN1M0xxx12	✓	✓	✗	Online
MK22FN256xxx12	✓	✓	✗	Online
MK22FN512xxx12	✓	✓	✓	Online
MK22FX512Axxx12	✓	✓	✗	Online
MK22FX512xxx12	✓	✓	✗	Online
MK24FN1M0xxx12	✓	✓	✓	Online

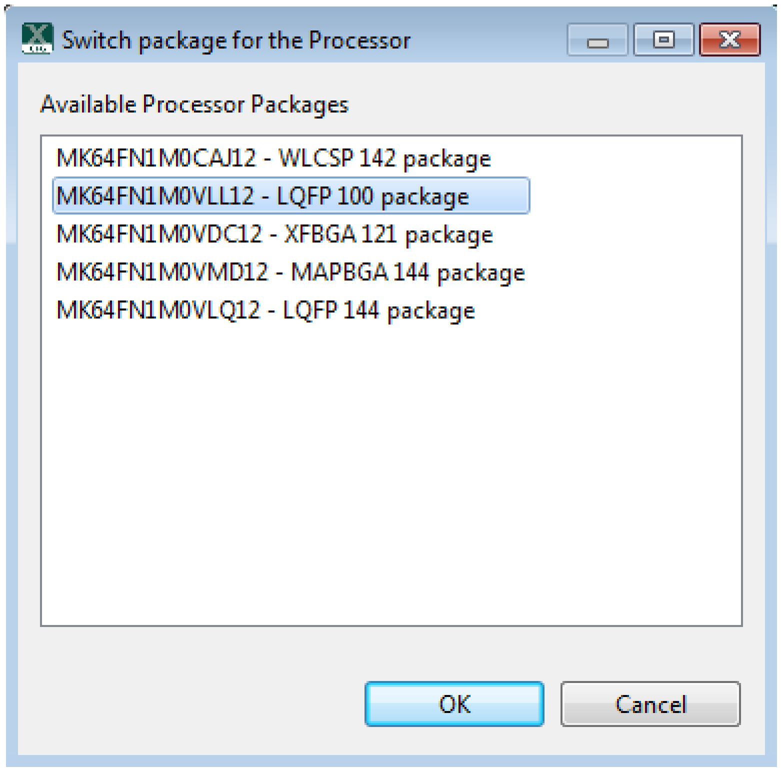
Name your configuration: FRDM-K22F

Select processor package: MK22FN128CAH12 - WLCSP 64 package

Select SDK version: SDK v2

Figure 82. Switch processor

Select **File** > **Switch package** to change the package of the current processor.


Switch package for the Processor
Available Processor Packages
OK
Cancel

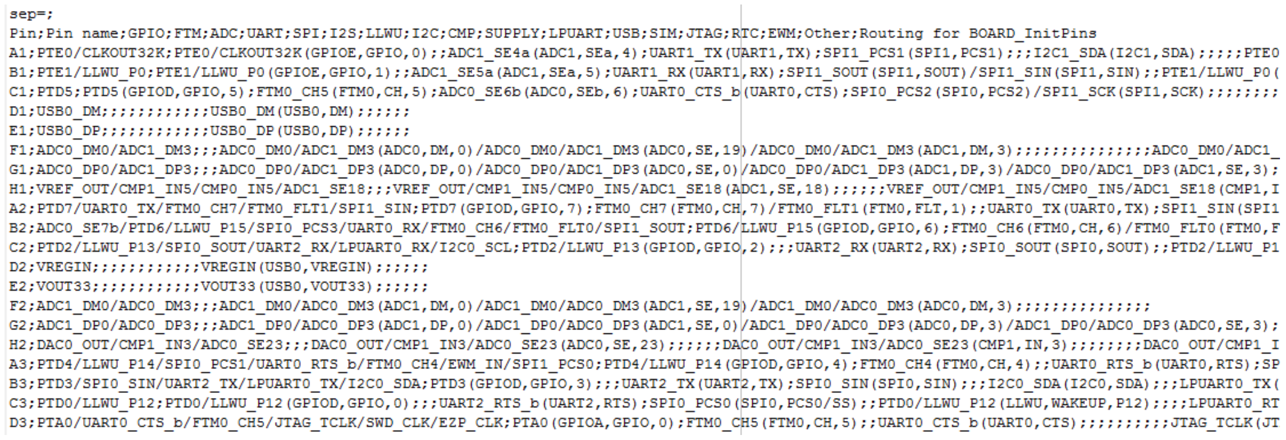
MK64FN1M0CAJ12 - WLCSP 142 package
MK64FN1M0VLL12 - LQFP 100 package
MK64FN1M0VDC12 - XFBGA 121 package
MK64FN1M0VMD12 - MAPBGA 144 package
MK64FN1M0VLQ12 - LQFP 144 package

Figure 83. Switch package

6.2 Exporting the Pins table

To export the Pins table, do the following:

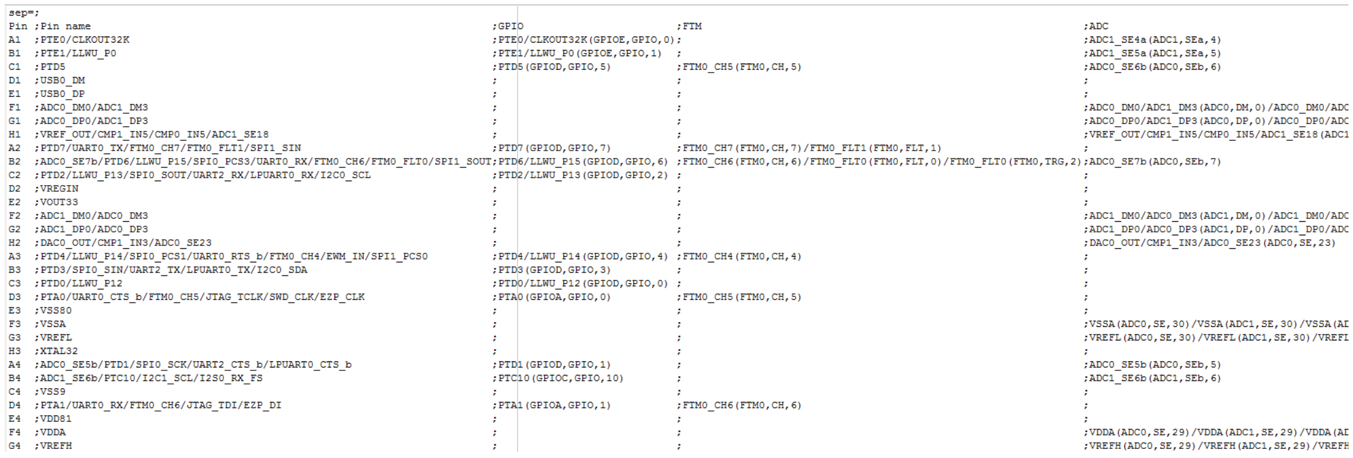
1. Select **File** > **Export** from the main menu.
2. In the **Export** dialog, select the **Export the Pins in CSV (Comma Separated Values) Format** option.
3. Click **Next**.
4. Select the folder and specify the file name to which you want to export.
5. The exported file contains content of the current Pins view table, plus lists the functions and the selected routed pins.



```
sep=;
Pin;Pin name;GPIO;FTM;ADC;UART;SPI;I2S;LLWU;I2C;CMP;SUPPLY;LPUART;USB;SIM;JTAG;RTC;EWM;Other;Routing for BOARD_InitPins
A1;PTE0/CLKOUT32K;PTE0/CLKOUT32K(GPIOE,GPIO,0);;ADC1_SE4a(ADC1,SEa,4);UART1_TX(UART1,TX);SPI1_PCS1(SPI1,PCS1);;;I2C1_SDA(I2C1,SDA);;;;;PTE0
B1;PTE1/LLWU_P0;PTE1/LLWU_P0(GPIOE,GPIO,1);;ADC1_SE5a(ADC1,SEa,5);UART1_RX(UART1,RX);SPI1_SOUT(SPI1,SOUT)/SPI1_SIN(SPI1,SIN);;PTE1/LLWU_P0(
C1;PTD5;PTD5(GPIOD,GPIO,5);FTM0_CH5(FTM0,CH,5);ADC0_SE6b(ADC0,SEb,6);UART0_CTS_b(UART0,CTS);SPI0_PCS2(SPI0,PCS2)/SPI1_SCK(SPI1,SCK);;;;;;;
D1;USB0_DM;;;;;;;;;;;;USB0_DM(USB0,DM);;;;;;
E1;USB0_DP;;;;;;;;;;;;USB0_DP(USB0,DP);;;;;;
F1;ADC0_DM0/ADC1_DM3;;ADC0_DM0/ADC1_DM3(ADC0,DM,0)/ADC0_DM0/ADC1_DM3(ADC0,SE,19)/ADC0_DM0/ADC1_DM3(ADC1,DM,3);;;;;;;;;;;;;;ADC0_DM0/ADC1_
G1;ADC0_DP0/ADC1_DP3;;ADC0_DP0/ADC1_DP3(ADC0,DP,0)/ADC0_DP0/ADC1_DP3(ADC0,SE,0)/ADC0_DP0/ADC1_DP3(ADC1,DP,3)/ADC0_DP0/ADC1_DP3(ADC1,SE,3);
H1;VREF_OUT/CMP1_IN5/CMP0_IN5/ADC1_SE18;;VREF_OUT/CMP1_IN5/CMP0_IN5/ADC1_SE18(ADC1,SE,18);;;;;;VREF_OUT/CMP1_IN5/CMP0_IN5/ADC1_SE18(CMP1,I
A2;PTD7/UART0_TX/FTM0_CH7/FTM0_FLT1/SPI1_SIN;PTD7(GPIOD,GPIO,7);FTM0_CH7(FTM0,CH,7)/FTM0_FLT1(FTM0,FLT,1);;UART0_TX(UART0,TX);SPI1_SIN(SPI1
B2;ADC0_SE7b/PTD6/LLWU_P15/SPI0_PCS3/UART0_RX/FTM0_CH6/FTM0_FLT0/SPI1_SOUT;PTD6/LLWU_P15(GPIOD,GPIO,6);FTM0_CH6(FTM0,CH,6)/FTM0_FLT0(FTM0,F
C2;PTD2/LLWU_P13/SPI0_SOUT/UART2_RX/LPUART0_RX/I2C0_SCL;PTD2/LLWU_P13(GPIOD,GPIO,2);;;UART2_RX(UART2,RX);SPI0_SOUT(SPI0,SOUT);;PTD2/LLWU_P1
D2;VREGIN;;;;;;;;;;;;VREGIN(USB0,VREGIN);;;;;;
E2;VOUT33;;;;;;;;;;;;VOUT33(USB0,VOUT33);;;;;;
F2;ADC1_DM0/ADC0_DM3;;ADC1_DM0/ADC0_DM3(ADC1,DM,0)/ADC1_DM0/ADC0_DM3(ADC1,SE,19)/ADC1_DM0/ADC0_DM3(ADC0,DM,3);;;;;;;;;;;;;;
G2;ADC1_DP0/ADC0_DP3;;ADC1_DP0/ADC0_DP3(ADC1,DP,0)/ADC1_DP0/ADC0_DP3(ADC1,SE,0)/ADC1_DP0/ADC0_DP3(ADC0,DP,3)/ADC1_DP0/ADC0_DP3(ADC0,SE,3);
H2;DAC0_OUT/CMP1_IN3/ADC0_SE23;;DAC0_OUT/CMP1_IN3/ADC0_SE23(ADC0,SE,23);;;;;;DAC0_OUT/CMP1_IN3/ADC0_SE23(CMP1,IN,3);;;;;;;;DAC0_OUT/CMP1_I
A3;PTD4/LLWU_P14/SPI0_PCS1/UART0_RTS_b/FTM0_CH4/EWM_IN/SPI1_PCS0;PTD4/LLWU_P14(GPIOD,GPIO,4);FTM0_CH4(FTM0,CH,4);;UART0_RTS_b(UART0,RTS);SP
B3;PTD3/SPI0_SIN/UART2_TX/LPUART0_TX/I2C0_SDA;PTD3(GPIOD,GPIO,3);;;UART2_TX(UART2,TX);SPI0_SIN(SPI0,SIN);;;I2C0_SDA(I2C0,SDA);;;LPUART0_TX(
C3;PTD0/LLWU_P12;PTD0/LLWU_P12(GPIOD,GPIO,0);;;UART2_RTS_b(UART2,RTS);SPI0_PCS0(SPI0,PCS0/SS);;PTD0/LLWU_P12(LLWU,WAKEUP,P12);;;;LPUART0_RT
D3;PTA0/UART0_CTS_b/FTM0_CH5/JTAG_TCLK/SWD_CLK/EZP_CLK;PTA0(GPIOA,GPIO,0);FTM0_CH5(FTM0,CH,5);;UART0_CTS_b(UART0,CTS);;;;;;;;;JTAG_TCLK(JT
```

Figure 84. Exported file content

The exported content can be used in other tools for further processing. For example, see it after aligning to blocks in the image below.



```
sep=;
Pin ;Pin name                                                               ;GPIO                        ;FTM                                                            ;ADC
A1  ;PTE0/CLKOUT32K                                                         ;PTE0/CLKOUT32K(GPIOE,GPIO,0);                                                               ;ADC1_SE4a(ADC1,SEa,4)
B1  ;PTE1/LLWU_P0                                                           ;PTE1/LLWU_P0(GPIOE,GPIO,1)  ;                                                               ;ADC1_SE5a(ADC1,SEa,5)
C1  ;PTD5                                                                   ;PTD5(GPIOD,GPIO,5)          ;FTM0_CH5(FTM0,CH,5)                                            ;ADC0_SE6b(ADC0,SEb,6)
D1  ;USB0_DM                                                                ;                            ;                                                               ;
E1  ;USB0_DP                                                                ;                            ;                                                               ;
F1  ;ADC0_DM0/ADC1_DM3                                                      ;                            ;                                                               ;ADC0_DM0/ADC1_DM3(ADC0,DM,0)/ADC0_DM0/ADC
G1  ;ADC0_DP0/ADC1_DP3                                                      ;                            ;                                                               ;ADC0_DP0/ADC1_DP3(ADC0,DP,0)/ADC0_DP0/ADC
H1  ;VREF_OUT/CMP1_IN5/CMP0_IN5/ADC1_SE18                                   ;                            ;                                                               ;VREF_OUT/CMP1_IN5/CMP0_IN5/ADC1_SE18(ADC1
A2  ;PTD7/UART0_TX/FTM0_CH7/FTM0_FLT1/SPI1_SIN                              ;PTD7(GPIOD,GPIO,7)          ;FTM0_CH7(FTM0,CH,7)/FTM0_FLT1(FTM0,FLT,1)                      ;
B2  ;ADC0_SE7b/PTD6/LLWU_P15/SPI0_PCS3/UART0_RX/FTM0_CH6/FTM0_FLT0/SPI1_SOUT;PTD6/LLWU_P15(GPIOD,GPIO,6) ;FTM0_CH6(FTM0,CH,6)/FTM0_FLT0(FTM0,FLT,0)/FTM0_FLT0(FTM0,TRG,2);ADC0_SE7b(ADC0,SEb,7)
C2  ;PTD2/LLWU_P13/SPI0_SOUT/UART2_RX/LPUART0_RX/I2C0_SCL                   ;PTD2/LLWU_P13(GPIOD,GPIO,2) ;                                                               ;
D2  ;VREGIN                                                                 ;                            ;                                                               ;
E2  ;VOUT33                                                                 ;                            ;                                                               ;
F2  ;ADC1_DM0/ADC0_DM3                                                      ;                            ;                                                               ;ADC1_DM0/ADC0_DM3(ADC1,DM,0)/ADC1_DM0/ADC
G2  ;ADC1_DP0/ADC0_DP3                                                      ;                            ;                                                               ;ADC1_DP0/ADC0_DP3(ADC1,DP,0)/ADC1_DP0/ADC
H2  ;DAC0_OUT/CMP1_IN3/ADC0_SE23                                            ;                            ;                                                               ;DAC0_OUT/CMP1_IN3/ADC0_SE23(ADC0,SE,23)
A3  ;PTD4/LLWU_P14/SPI0_PCS1/UART0_RTS_b/FTM0_CH4/EWM_IN/SPI1_PCS0          ;PTD4/LLWU_P14(GPIOD,GPIO,4) ;FTM0_CH4(FTM0,CH,4)                                            ;
B3  ;PTD3/SPI0_SIN/UART2_TX/LPUART0_TX/I2C0_SDA                             ;PTD3(GPIOD,GPIO,3)          ;                                                               ;
C3  ;PTD0/LLWU_P12                                                          ;PTD0/LLWU_P12(GPIOD,GPIO,0) ;                                                               ;
D3  ;PTA0/UART0_CTS_b/FTM0_CH5/JTAG_TCLK/SWD_CLK/EZP_CLK                    ;PTA0(GPIOA,GPIO,0)          ;FTM0_CH5(FTM0,CH,5)                                            ;
E3  ;VSS80                                                                  ;                            ;                                                               ;
F3  ;VSSA                                                                   ;                            ;                                                               ;VSSA(ADC0,SE,30)/VSSA(ADC1,SE,30)/VSSA(AD
G3  ;VREFL                                                                  ;                            ;                                                               ;VREFL(ADC0,SE,30)/VREFL(ADC1,SE,30)/VREFL
H3  ;XTAL32                                                                 ;                            ;                                                               ;
A4  ;ADC0_SE5b/PTD1/SPI0_SCK/UART2_CTS_b/LPUART0_CTS_b                      ;PTD1(GPIOD,GPIO,1)          ;                                                               ;ADC0_SE5b(ADC0,SEb,5)
B4  ;ADC1_SE6b/PTC10/I2C1_SCL/I2S0_RX_FS                                    ;PTC10(GPIOC,GPIO,10)        ;                                                               ;ADC1_SE6b(ADC1,SEb,6)
C4  ;VSS9                                                                   ;                            ;                                                               ;
D4  ;PTA1/UART0_RX/FTM0_CH6/JTAG_TDI/EZP_DI                                 ;PTA1(GPIOA,GPIO,1)          ;FTM0_CH6(FTM0,CH,6)                                            ;
E4  ;VDD81                                                                  ;                            ;                                                               ;
F4  ;VDDA                                                                   ;                            ;                                                               ;VDDA(ADC0,SE,29)/VDDA(ADC1,SE,29)/VDDA(AD
G4  ;VREFH                                                                  ;                            ;                                                               ;VREFH(ADC0,SE,29)/VREFH(ADC1,SE,29)/VREFH
```

Figure 85. Aligning to block

6.3 Tools advanced configuration

Use the ide\mcuxpressoide.ini file to configure the processor data directory location. You can define the "com.nxp.mcudata.dir" property to set the data directory location.

For example: `-Dcom.nxp.mcudata.dir=C:/my/data/directory.`

6.4 Generating HTML report

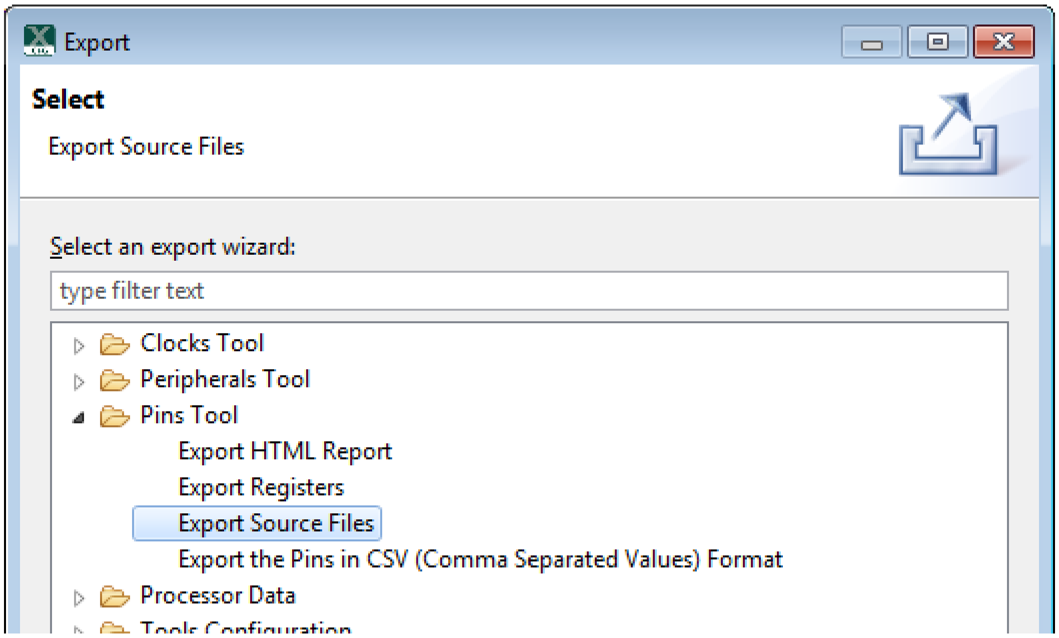
Select **Export** > **Pins/Clocks/Peripherals Tool** > **Export HTML Report** to generate the report.

6.5 Exporting sources

It's possible to export the generated source using the Export wizard.

To launch the Export wizard:

1. Select **File** > **Export** from the **Main Menu**.
2. Select **Export Source Files**.


Export

Select

Export Source Files

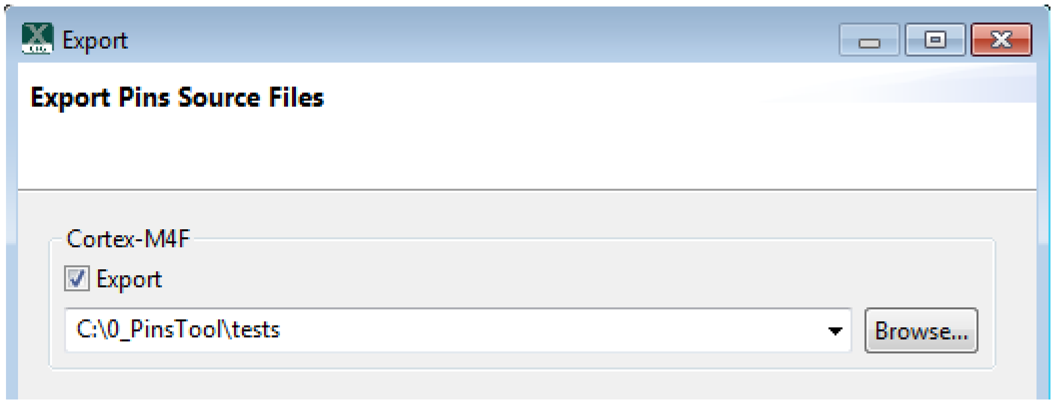
Select an export wizard:

type filter text

- Clocks Tool
- Peripherals Tool
- Pins Tool
  - Export HTML Report
  - Export Registers
  - Export Source Files
  - Export the Pins in CSV (Comma Separated Values) Format
- Processor Data
- Tools Configuration

Figure 86. Export wizard

3. Click **Next**.
4. Select the target folder where you want to store the generated files.


Export

Export Pins Source Files

Cortex-M4F

☑ Export

C:\0_PinsTool\tests

Figure 87. Select target folder

5. In case of multicore processors, select the cores you want to export.
6. Click **Finish**.

6.6 Exporting registers

You can export the content of tool-modified registers data using the Export wizard.

To export registers, follow these steps:

1. Select **File** > **Export** from the main menu.
2. Select the **Pins Tool** > **Export Registers** option.
3. Click **Next**.
4. Select the target file path where you want to export modified registers content.

5. Click **Finish**.

6.7 Command line execution

This section describes the Command Line Interface (CLI) commands supported by the desktop application.

MCUXpresso Config tools can be executed on command line with these parameters: `mcuxpressoide.exe -noSplash -application com.nxp.swtools.framework.application [tools commands].`

Notes regarding command line execution:

- Command **-HeadlessTool** is used as a separator of each command chain.
- Each command chain works independently.
- Every chain starts with **-HeadlessTool** command and continues to the next **-HeadlessTool** command, or end. (only exception are commands from framework which doesn´t need the **-HeadlessTool** command).
- Commands which don´t need the **-HeadlessTool** command, can be placed before the first **-HeadlessTool** if chained, or without -**HeadlessTool** when not chained.
- Commands from each tool are executed in given order.
- Commands from framework **are not executed in given order.**
- The following commands are not executed in given order:
  - ImportProject
  - Export MEX
  - ExportAll
- The application can exit with following codes when unexpected behavior occurs: hen parameter is missing:
  - When parameter is missing: 1
  - When tool error occurs: 2

Command example:

-HeadlessTool Clocks -MCU MKL43Z256xxx4 -SDKVersion ksdk2_0 -ExportSrc C:/exports/src -HeadlessTool Pins -MCU MK65FN2M0xxx18 -SDKVersion ksdk2_0 -ExportSrc C:/exports/src -HeadlessTool Peripherals -MCU MK64FX512xxx12 -SDKVersion ksdk2_0 -ExportSrc C:/exports/src

The following commands are supported in the **framework**:

Table 6. Commands supported in the framework

Command name	Definition and parameters	Description	Restriction	Example
Force language	-nl {lang}	Force set language {lang} is in ISO-639-1 standard	Removal of the '.nxp' folder from home directory is recommended, as some text might be cached Only 'zh' and 'en' are supported	-nl zh

Table continues on the next page...

Table 6. Commands supported in the framework (continued)

Show console	-consoleLog	Log output is also sent to Java's System.out (typically back to the command shell if any)	None	
Select MCU	-MCU	MCU to be selected by framework	Requires –SDKversion command	-MCU MK64FX512xxx12
Select SDK version	-SDKversion	Version of the MCU to be selected by framework	Requires -MCU command	-SDKversion test_ksdk2_0
Select part number	-PartNum	Select specific package of the MCU	Requires -MCU and -SDKversion commands	-PartNum MK64FX512VLL12
Configuration name	-ConfigName	Name of newly created configuration - used in export	Name is used when new configuration is created by -MCU and -SDKversion commands	-ConfigName "MyConfig"
Select tool	-HeadlessTool	Select a tool that should be run in headless mode	None	-HeadlessTool Clocks
Load configuration	-Load	Load existing configuration from (*.mex) file	None	-Load C:/conf/conf.mex
Export Mex	-ExportMEX	Export .mex configuration file after tools run Argument is expected as a folder name	None	-MCU xxx -SDKversion xxx -ExportMEX C:/exports/my_config_folder
Export all generated files	-ExportAll	Export generated files (with source code and so on. Code is regenerated before export Includes -ExportSrc and in framework -ExportMEX Argument is expected as a folder name. Argument is expected as a folder name	Requires -HeadlessTool command	-HeadlessTool Pins -ExportAll C:/exports/generated
Create new configuration by importing toolchain project	-ImportProject {path}	Creates new configuration by importing toolchain project Parameter is path to the root of the toolchain project	Requires -HeadlessTool command	-HeadlessTool Pins -ImportProject c:\test\myproject
Specify SDK path	-SDKpath {path}	Specify absolute path to the root directory of the SDK package.	@since v3.0	-SDKpath c:\nxp\SDK_2.0_MKL43Z256xxx4

6.7.1 Command line execution - Pins Tool

This section describes the Command Line Interface (CLI) commands supported in the **Pins Tool**.

Table 7. Commands supported in Pins

Command name	Definition and parameters	Description	Restriction	Example
Import C files	-ImportC	Import .c files into configuration Importing is done after loading mex and before generating outputs	Requires -HeadlessTool Pins	-HeadlessTool Pins -ImportC C:/imports/file1.c C:/imports/file2.c
Import DTSI files	-ImportDTSI	Import .dtsi files into configuration Importing is done after loading mex and before generating outputs	Requires -HeadlessTool Pins	-HeadlessTool Pins -ImportDTSI C:/imports/file1.dtsi C:/imports/file2.dtsi
Export all generated files (to simplify all exports commands to one command)	-ExportAll	Export generated files (with source code etc.) Code will be regenerated before export Includes -ExportSrc,-ExportCSV, -ExportHTML and in framework -ExportMEX Argument is expected as a folder name	Requires -HeadlessTool Pins	-HeadlessTool Pins -ExportAll C:/exports/generated
Export Source files	-ExportSrc	Export generated source files. Code will be regenerated before export Argument is expected as a folder name	Requires -HeadlessTool Pins	-HeadlessTool Pins -ExportSrc C:/exports/src
Export CSV file	-ExportCSV	Export generated csv file. Code will be regenerated before export Argument is expected as a folder name	Requires -HeadlessTool Pins	-HeadlessTool Pins -ExportSrc C:/exports/src

Table continues on the next page...

Table 7. Commands supported in Pins (continued)

Export HTML report file	-ExportHTML	Export generated html report file. Code will be regenerated before export Argument is expected as a folder name	Requires -HeadlessTool Pins	-HeadlessTool Pins -ExportHTML C:/exports/html
Export registers	-ExportRegisters	Export registers tab into folder. Code will be regenerated before export Argument is expected as a folder name	Requires -HeadlessTool Pins	-HeadlessTool Pins -ExportRegisters C:/exports/regs

6.7.2 Command line execution - Clocks Tool

This section describes the Command Line Interface (CLI) commands supported by the **Clocks Tool**.

Table 8. Commands supported in Clocks

Command name	**Definition and parameters**	**Description**	**Restriction**	**Example**
Import C files	-ImportC	Import .c files into configuration Importing is done after loading mex and before generating outputs	Requires -HeadlessTool Clocks	-ImportC C:/imports/file1.c C:/imports/file2.c
Export all generated files	-ExportAll	Export generated files (with source code and so on. Code is regenerated before export Includes -ExportSrc and in framework -ExportMEXArgument is expected as a folder name. Argument is expected as a folder name	Requires -HeadlessTool Clocks	-ExportAll C:/exports/generated

Table continues on the next page...

Table 8. Commands supported in Clocks (continued)

Export Source files	-ExportSrc	Export generated source files. Code will be regenerated before export Argument is expected as a folder name	Requires -HeadlessTool Clocks	-ExportSrc C:/ exports/src
Export HTML report file	-ExportHTML	Export generated html report file. Code will be regenerated before export Argument is expected as a folder name	Requires -HeadlessTool Clocks	-ExportHTML C:/ exports/html

6.7.3 Command line execution - Peripherals Tool

This section describes the Command Line Interface (CLI) commands supported by the **Peripherals Tool**.

Table 9. Commands supported in Peripherals Tool

Command name	Definition and parameters	Description	Restriction	Example
Export all generated files (to simplify all exports commands to one command)	-ExportAll	Export generated files (with source code etc.) Code will be regenerated before export Includes -ExportSrc, -ExportHTML and in framework -ExportMEX Argument is expected to be a folder	Requires - HeadlessTool Peripherals	-HeadlessTool Peripherals -ExportAll C:/exports/generated
Export Source files	-ExportSrc	Export generated source files. Code will be regenerated before export Argument is expected to be a folder	Requires - HeadlessTool Peripherals	-HeadlessTool Peripherals -ExportSrc C:/exports/src
* for internal commands, internal plugin must be installed into production application				

6.7.4 Command line execution - Project Cloner

This section describes the Command Line Interface (CLI) commands supported by the **Project Cloner**.

Table 10. Commands supported in Project Cloner

Command name	**Definition and parameters**	**Description**	**Restriction**	**Example**
Specify SDK path	-SDKpath {path}	Specify absolute path to the root directory of the SDK package	@since v3.0	-SDKpath c:\nxp \SDK_2.0_MKL43Z256 xxx4
Clone SDK example project	-PG_clone {board} {example} {toolchain} {wrkspc} {prjName}	Clones specified SDK example projecte under new name 1. {board} - subdirectory of the board in SDK package 2. {example} - relative path from board sub-dir and name of the example, for example demo_apps/ hello_world; use '/' as a path separator 3. {toolchain} - id of the toolchain to create project (see toolchains - toolchain - id) 4. {wrkspc} - absolute path where new project shall be created, e.g. projects workspace 5. {prjName} - name of the new project	Requires -HeadlessTool PrjCloner and -SDKpath {path} @since v3.0	-HeadlessTool PrjCloner -SDKpath c: \nxp \SDK_2.0_MKL43Z256 xxx4 -PG_clone twrk64f120m demo_apps/hello kds c: \tmp exmpl

6.8 Working offline

To work offline, you need to first download the processor-specific data. Once the configuration is created for the processor, the internet connection is not needed anymore.

Chapter 7
Support

If you have any questions or need additional help, perform a search on the forum or post a new question. Visit https://community.nxp.com/community/mcuxpresso/mcuxpresso-config .