NXP Semiconductors
User's Guide

Document Number: MCUXIDECTUG
Rev 3, 03/2018

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.

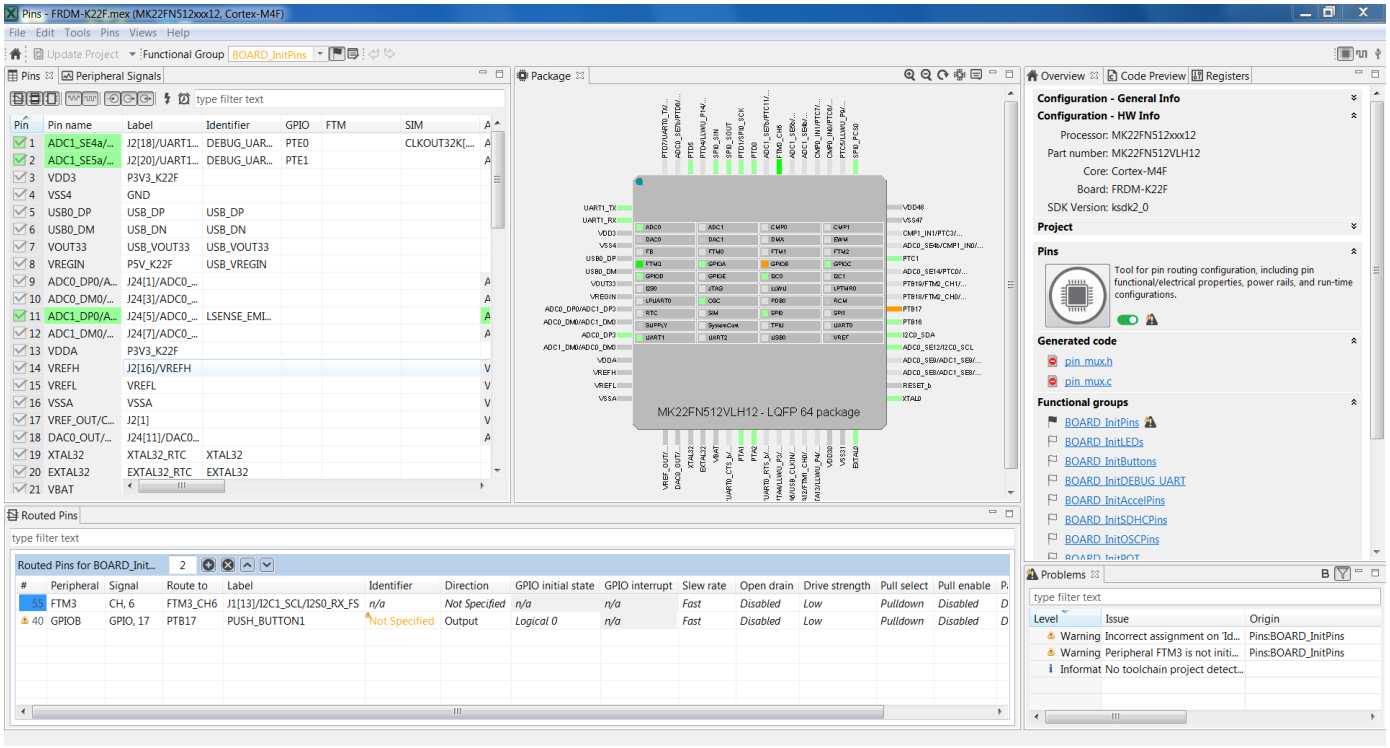


Figure 1. Desktop version of Pins Tool

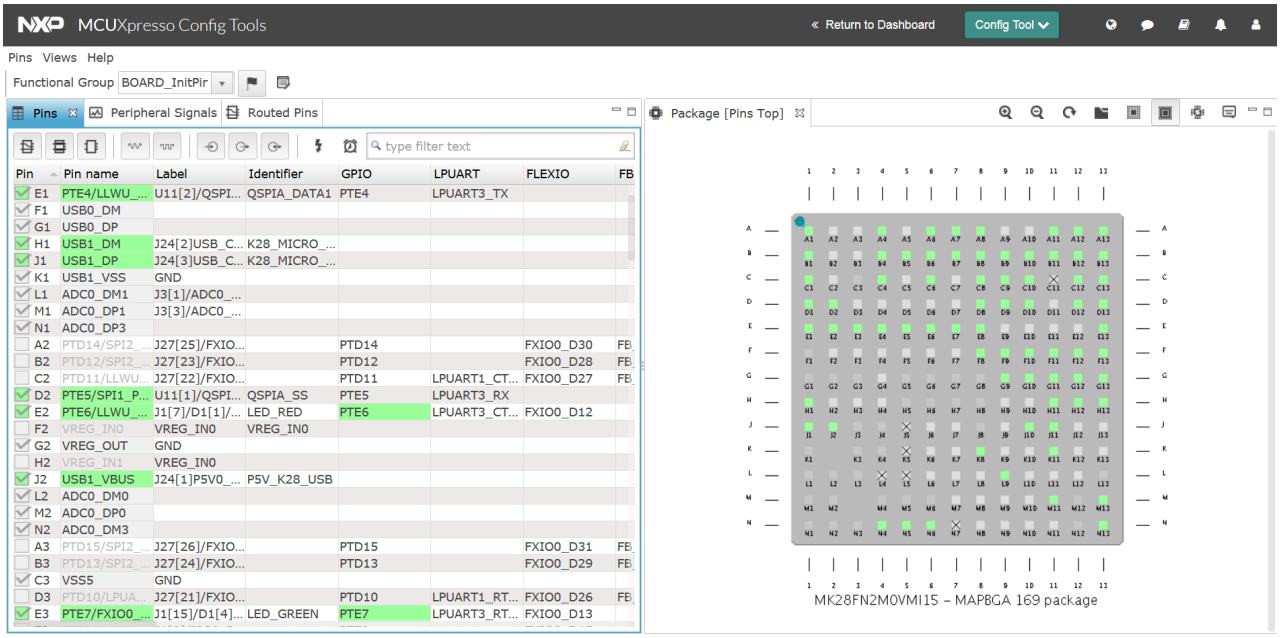


Figure 2. Web version of Pins Tool

Chapter 2
Config Tools User Interface

2.1 Configuration

Configuration stands for common tools settings stored in .mex 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 on the Eclipse project, which is based on MCUXpresso SDK, and select command **MCUXpresso Config Tool > Open Pins**. This command:

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

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

Current configuration can be saved using “Save” button on the toolbar or using main menu - File – Save. 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 *.mex file stored in the project root folder. If file does not exist, new one is created using current project name.

NOTE

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

2.1.3 Opening an existing configuration

Configuration can be opened by the same command as creating new configuration, e.g. in Project Explorer right click on the Eclipse project, which is based on MCUXpresso SDK, and select command **MCUXpresso Config Tool - Open Pins**.

Only one configuration can be opened at one time. If you open second configuration, the first configuration is automatically closed. If this configuration is not saved, tool offers to save it before closing.

NOTE

If you select different Eclipse project, you must explicitly open configuration for this project.

By default, last used configuration is re-opened during starting MCUX IDE. This feature can be affected in the preferences.

2.1.4 Importing sources

To import source code files:

1. Select **File > Import** from the main menu.
2. Select the **Import Source Files** option.

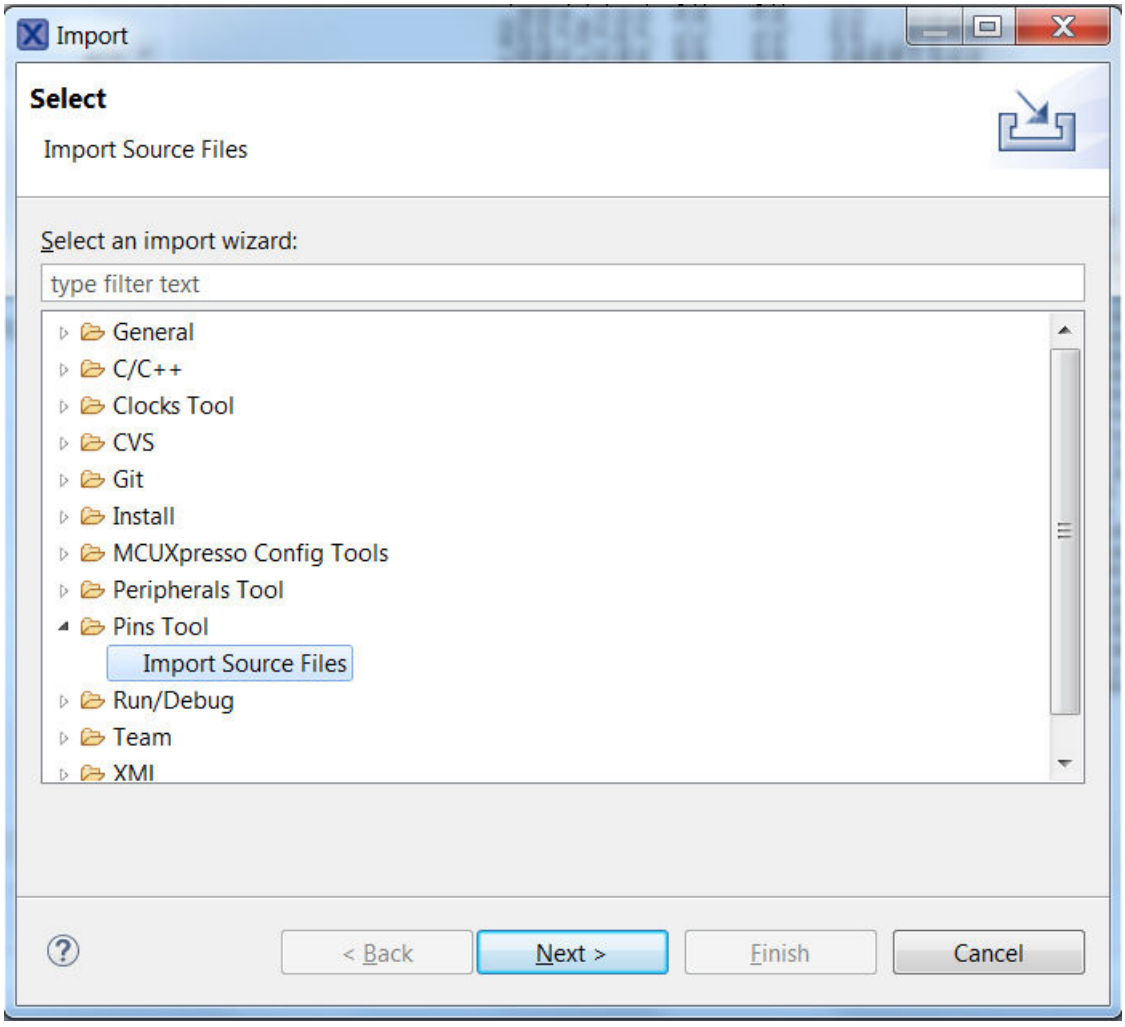


Figure 3. Import Source wizard

3. Click **Next**.
4. It is possible to 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.

2.1.5 Exporting sources

It is possible to export generated source using the Export wizard.

To launch the Export wizard:

1. Select **File > Export** from the main menu.
2. Select the **Export Source Files** option.

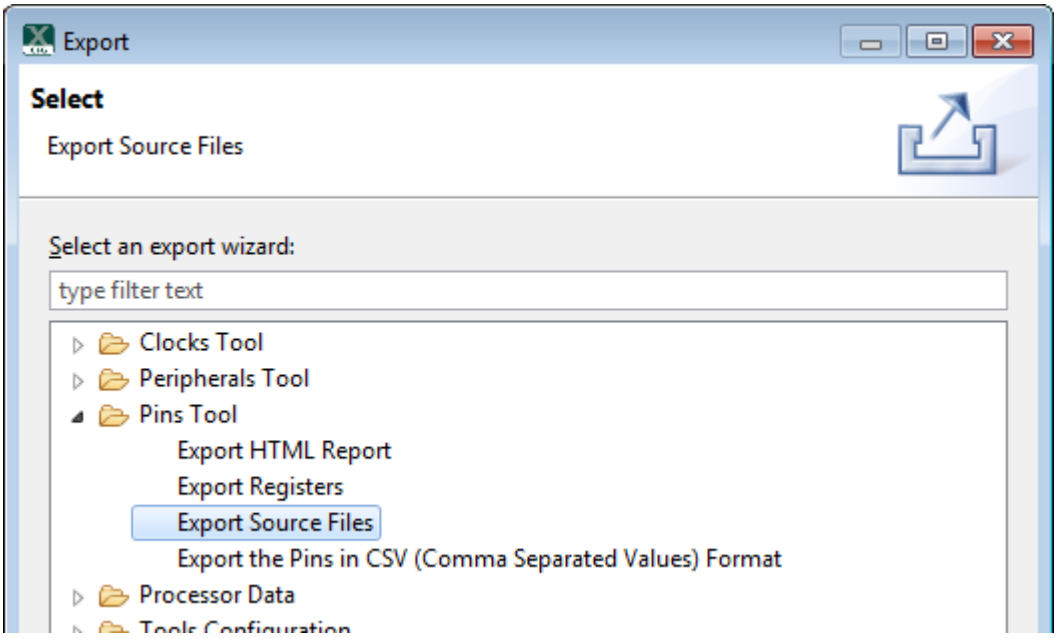


Figure 4. Export wizard

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

Figure 5. Select target folder

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

2.1.6 Restoring configuration from source code

The generated code contains information on 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 is possible to re-import the clock settings into the tool using the following steps.

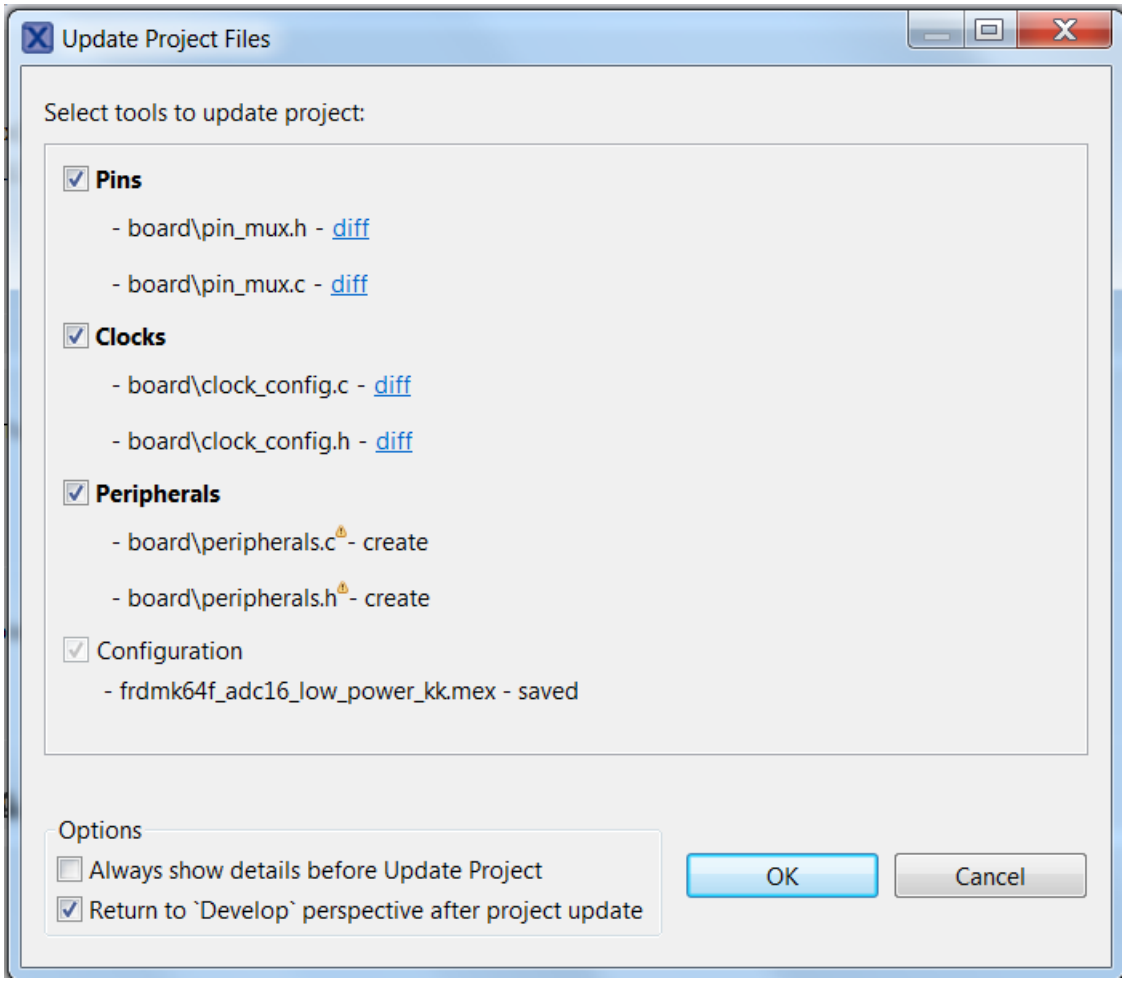
1. Select the command: **File > Import…**.
2. Select **Clocks Tool / Import Source Files**.
3. Click **Next**.
4. Click **Browse**.
5. Navigate and select the *clock_config.c* file previously produced by the Clocks Tool.
6. If the settings parse successfully, the clock configurations are added into the 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 you want to update. If code update is not possible, the button is highlighted in gray with reason displayed in the tooltip.



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

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**.

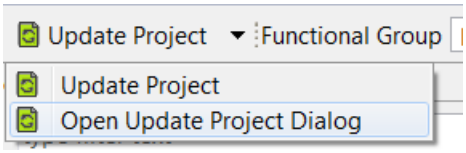


Figure 7. Update Project drop-down menu

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.

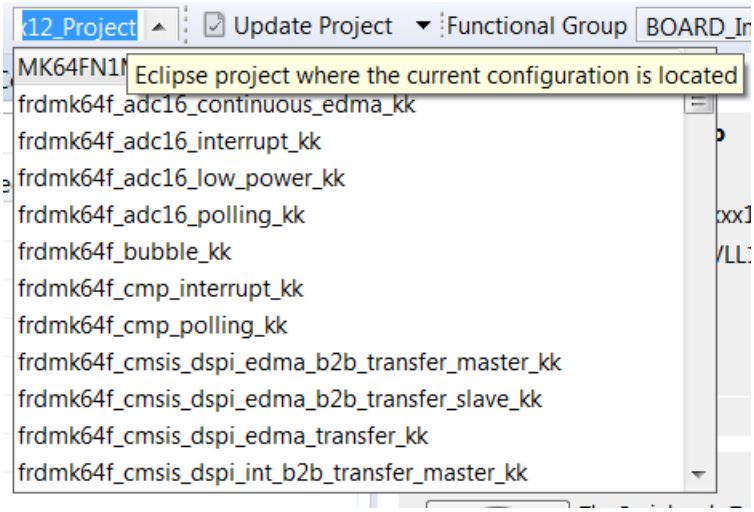


Figure 8. Eclipse project selection

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.

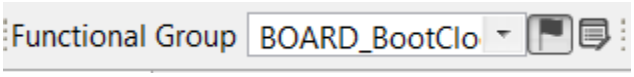


Figure 9. Functional groups

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

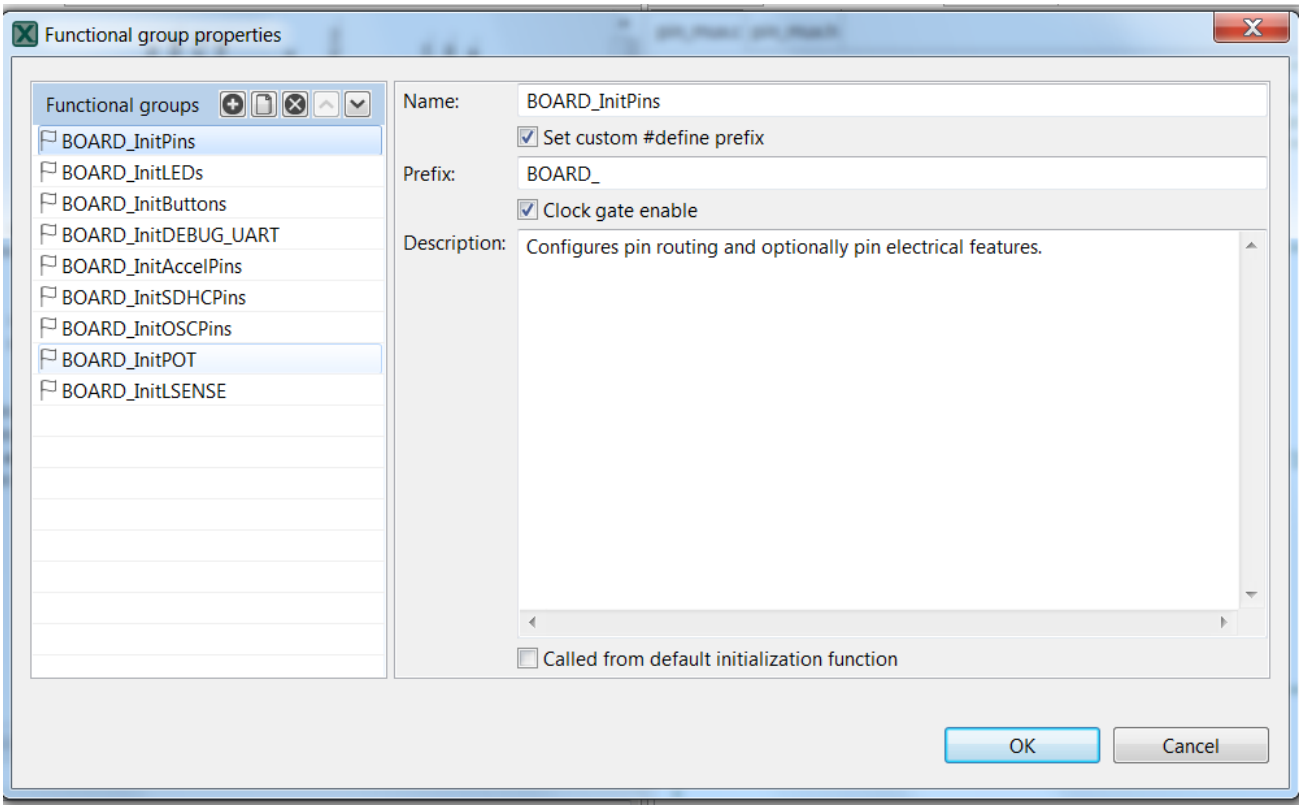


Figure 10. Functional group properties for Pins Tool

In this dialog, it is possible to configure several options for functions and code generation. Each settings is applicable for selected function. It is possible to 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. It is also possible to 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 Global clock settings

Global clock settings, for example: Run Mode and MCG mode, are shown in Clocks Tool only. Use this menu to select desired processor global settings. Hover the items to see the exact description of each mode.

2.2.5 Switching the tools

The buttons on the right side of the toolbar represent the available tools. You can click them to quickly navigate between Clocks, Pins and Peripherals tools.

[1] *if supported by processor

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.

2.4 Preferences

To configure preferences, select **Window > Preferences** from the main menu. The **Preferences** dialog appears. Select **MCUXpresso Config Tools** preference in the left pane.

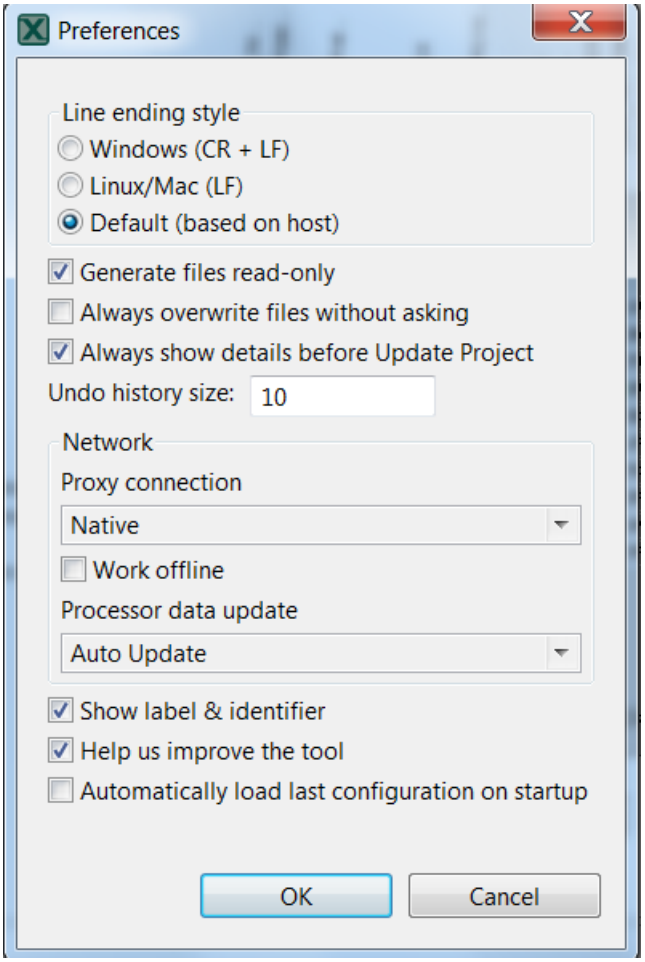


Figure 11. Preferences dialog

In this dialog it is possible to set:

- **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 label & identifier** – Select to show the pin label and the label identifier in the relevant views.
- **Help us to improve the tool** – Select to send device-configuration and tool-use information to NXP. Sending this information to NXP helps fix issues and improve the tools.
- **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 source does not contain YAML code, it is not importable.

2.6 Updates

To perform a check for updates select the **Help > Check for updates** menu. It contacts the server and checks whether there is a new version available.

NOTE

To check updates, internet connection is required.

2.7 Problems view

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

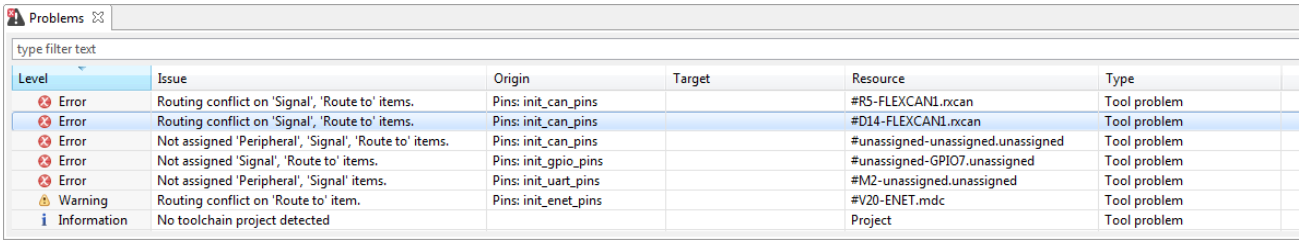


Figure 12. Problems view

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

There is a context-menu for each problem that shows the problem in the tool (to see context of the problem) or the quick-fix to the problem (if available).

NOTE

The quick-fix is not provided for all the listed problems.

Filter buttons

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

-  – 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.

2.8 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.

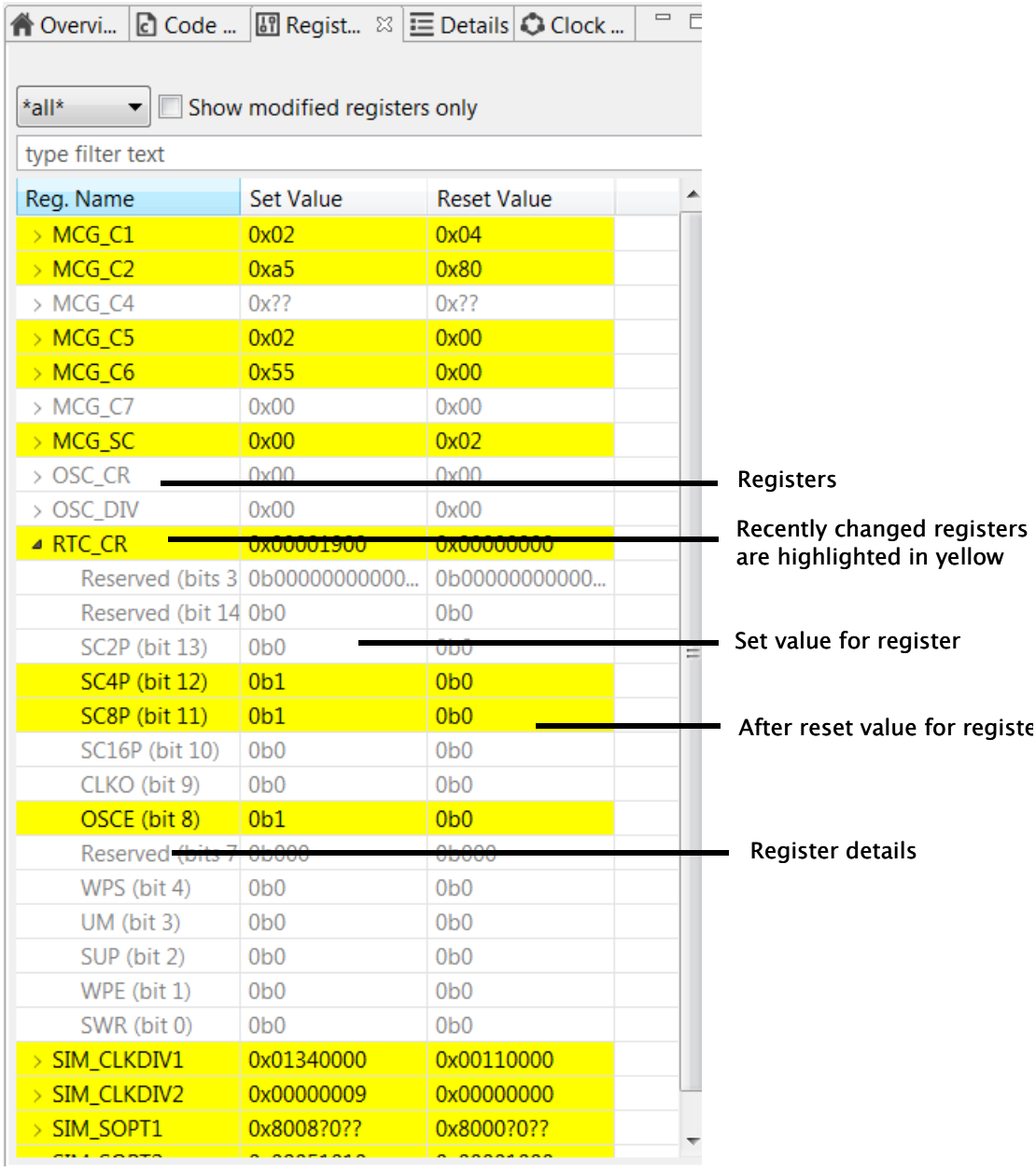


Figure 13. Registers view

The **Registers view** contains:

- **Peripheral filter** drop-down list – Use this filter 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** – Enables you 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.

2.9 Log view

The **Log** view shows user-specific information about the progress of the tools. The **Log** view can show up to 100 records throughout the 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 record is created by the shared functionality.

The content of the **Log** view is filtered using the combo boxes and shows only the specific tool and/or severity of the record.

The buffered log records are cleared using the clear button.

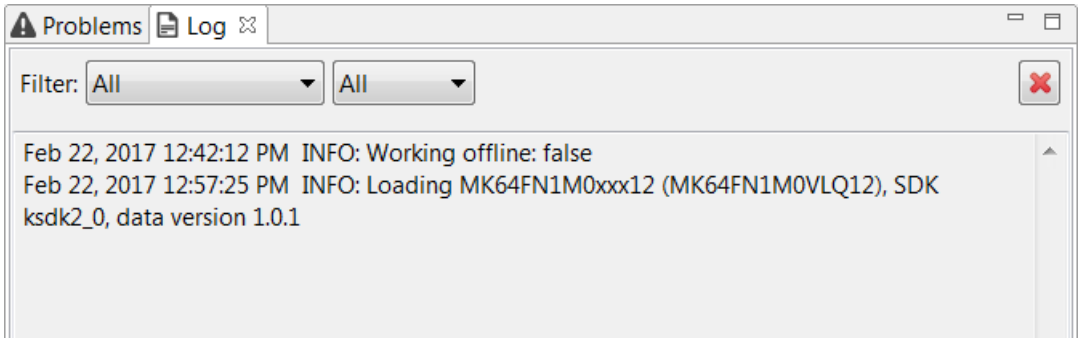


Figure 14. Log view

2.10 Config tools overview

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

- **Configuration – General Info** – Shows the name of and the path to the *.mex file of the current configuration. Click the link and open the folder containing the *.mex file. To import additional settings, click the **Import additional settings into current configuration** button.
- **Configuration – HW Info** – Shows the processor, part number, core, and SDK-version information of the current configuration.

- **Project** – Shows the toolchain project information.
- **Pins/Clocks/Peripherals** – Shows the 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.

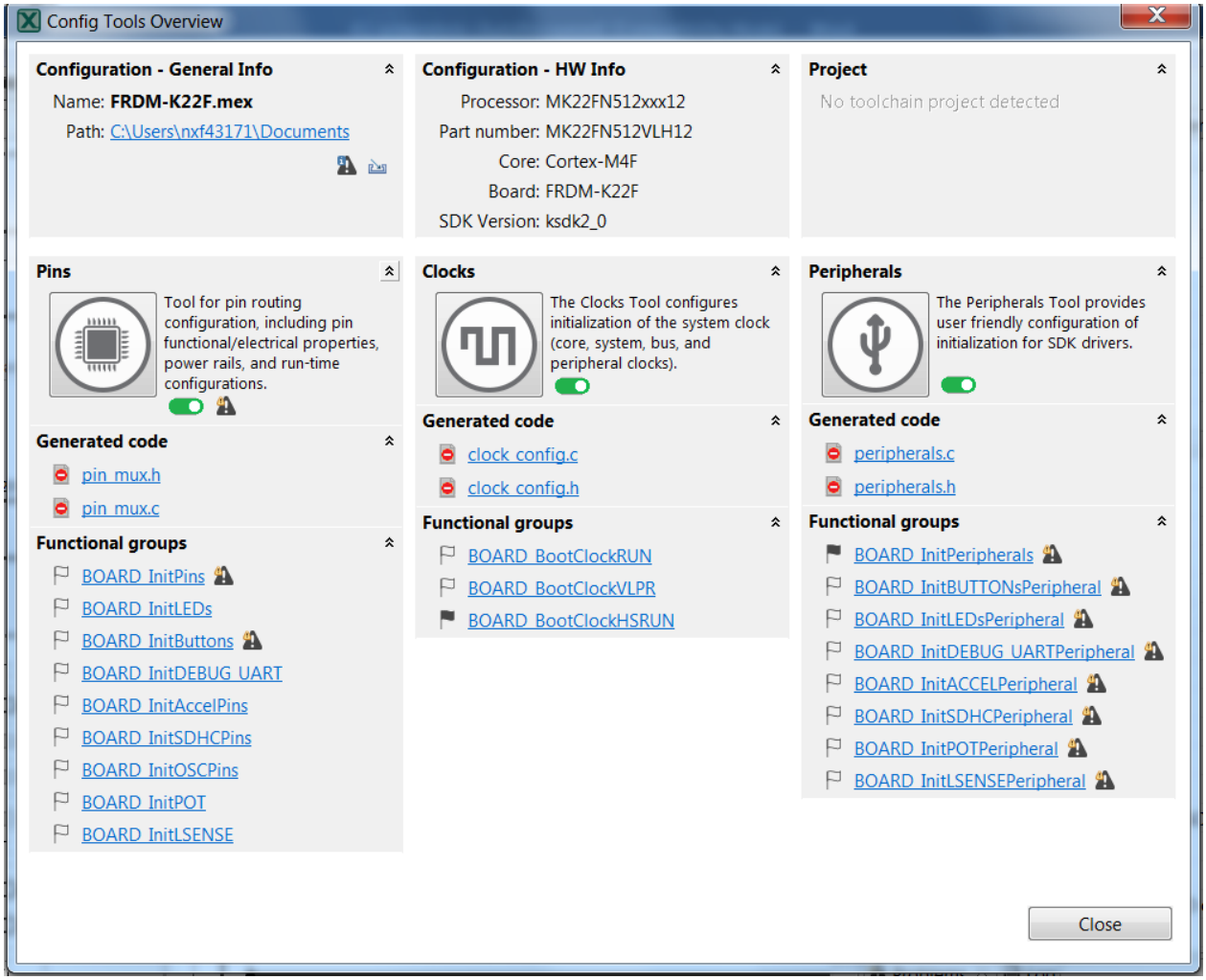


Figure 15. Config Tools Overview dialog

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.

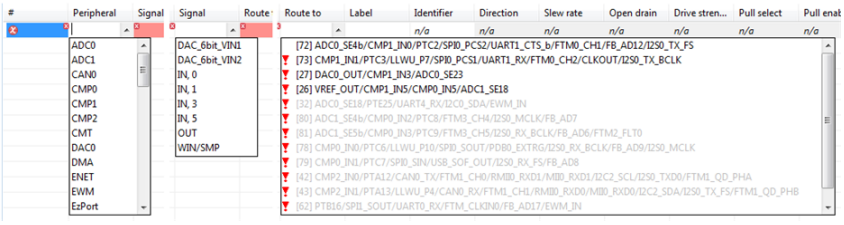


Figure 16. Defining routing path

3.1.2 Beginning with pin/internal signal selection

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

1. Begin with the pin/internal signal selection (**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 decorators:

   - 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

The source code is automatically generated.

3. Open the **Code Preview** view and see the output source code.
4. Export the source code.
   a. For the *Desktop* version: Select **File > Export** from the main menu.
   b. For the *Web* version: Select **Pins > Export** from the main menu.

NOTE

To export the source code, you can also click the **Export** button in the **Code Preview** view. The **Export** button is available in both the Desktop and Web versions.

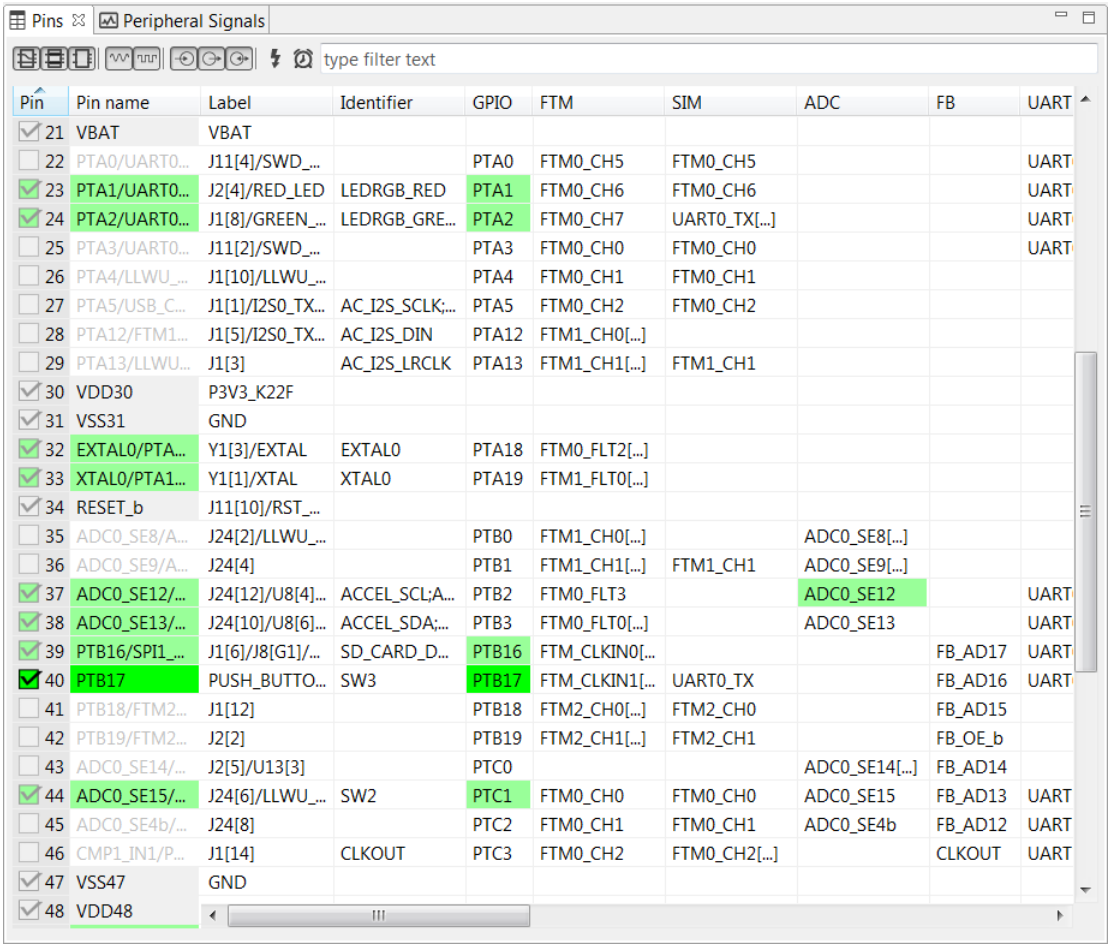
3.3 Example usage

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.

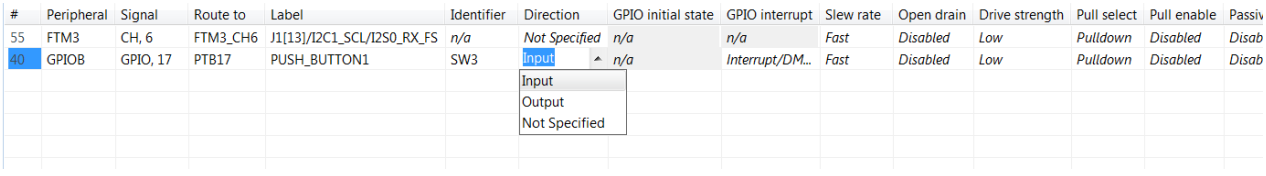
1. In the **Pins** view on the left, select the **UART3_RX** and **TX** signals. For this, you can click into the cells to make them 'green'.



Pin	Pin name	Label	Identifier	GPIO	FTM	SIM	ADC	FB	UART
21	VBAT	VBAT							
22	PTA0/UART0...	J11[4]/SWD_...		PTA0	FTM0_CH5	FTM0_CH5			UART
23	PTA1/UART0...	J2[4]/RED_LED	LEDRGB_RED	PTA1	FTM0_CH6	FTM0_CH6			UART
24	PTA2/UART0...	J1[8]/GREEN_...	LEDRGB_GRE...	PTA2	FTM0_CH7	UART0_TX[...]			UART
25	PTA3/UART0...	J11[2]/SWD_...		PTA3	FTM0_CH0	FTM0_CH0			UART
26	PTA4/LLWU_...	J1[10]/LLWU_...		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[...]		ADC0_SE8[...]		
36	ADC0_SE9/A...	J24[4]		PTB1	FTM1_CH1[...]	FTM1_CH1	ADC0_SE9[...]		
37	ADC0_SE12/...	J24[12]/U8[4]...	ACCEL_SCL;A...	PTB2	FTM0_FLT3		ADC0_SE12		UART
38	ADC0_SE13/...	J24[10]/U8[6]...	ACCEL_SDA;...	PTB3	FTM0_FLT0[...]		ADC0_SE13		UART
39	PTB16/SPI1_...	J1[6]/J8[G1]/...	SD_CARD_D...	PTB16	FTM_CLKIN0[...			FB_AD17	UART
40	PTB17	PUSH_BUTTO...	SW3	PTB17	FTM_CLKIN1[...	UART0_TX		FB_AD16	UART
41	PTB18/FTM2...	J1[12]		PTB18	FTM2_CH0[...]	FTM2_CH0		FB_AD15	
42	PTB19/FTM2...	J2[2]		PTB19	FTM2_CH1[...]	FTM2_CH1		FB_OE_b	
43	ADC0_SE14/...	J2[5]/U13[3]		PTC0			ADC0_SE14[...]	FB_AD14	
44	ADC0_SE15/...	J24[6]/LLWU_...	SW2	PTC1	FTM0_CH0	FTM0_CH0	ADC0_SE15	FB_AD13	UART
45	ADC0_SE4b/...	J24[8]		PTC2	FTM0_CH1	FTM0_CH1	ADC0_SE4b	FB_AD12	UART
46	CMP1_IN1/P...	J1[14]	CLKOUT	PTC3	FTM0_CH2	FTM0_CH2[...]		CLKOUT	UART
47	VSS47	GND							
48	VDD48								

Figure 17. 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

Input
Output
Not Specified

Figure 18. 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.

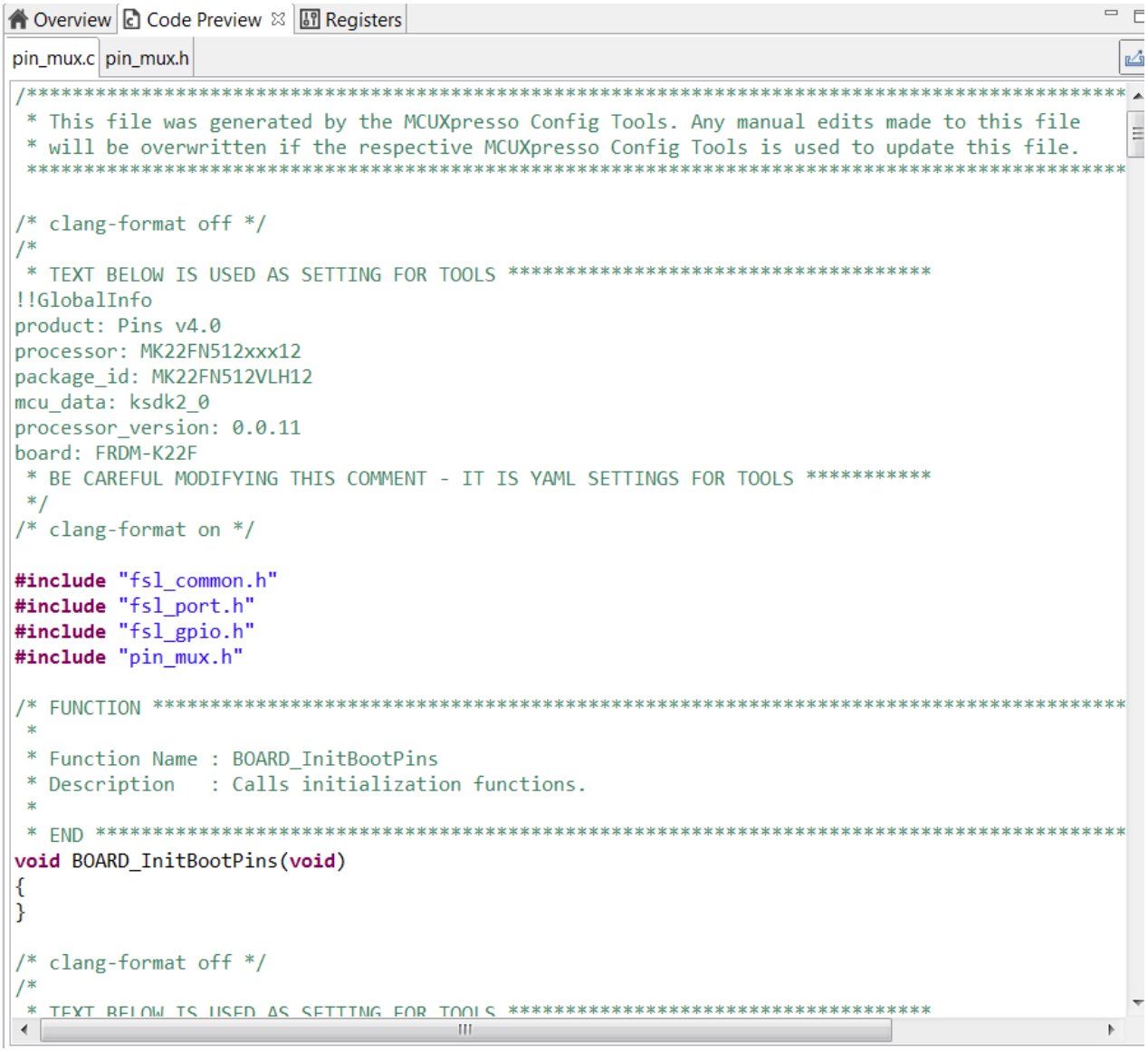


Figure 19. Generated code

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.

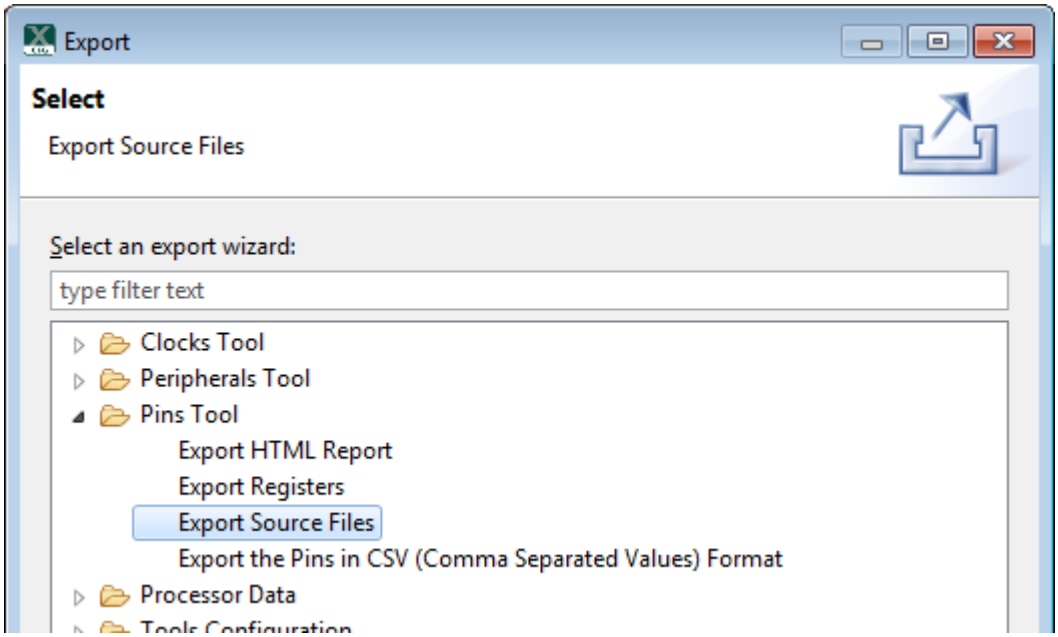


Figure 20. Export Source Files

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

The Pins Tool consists of several views.

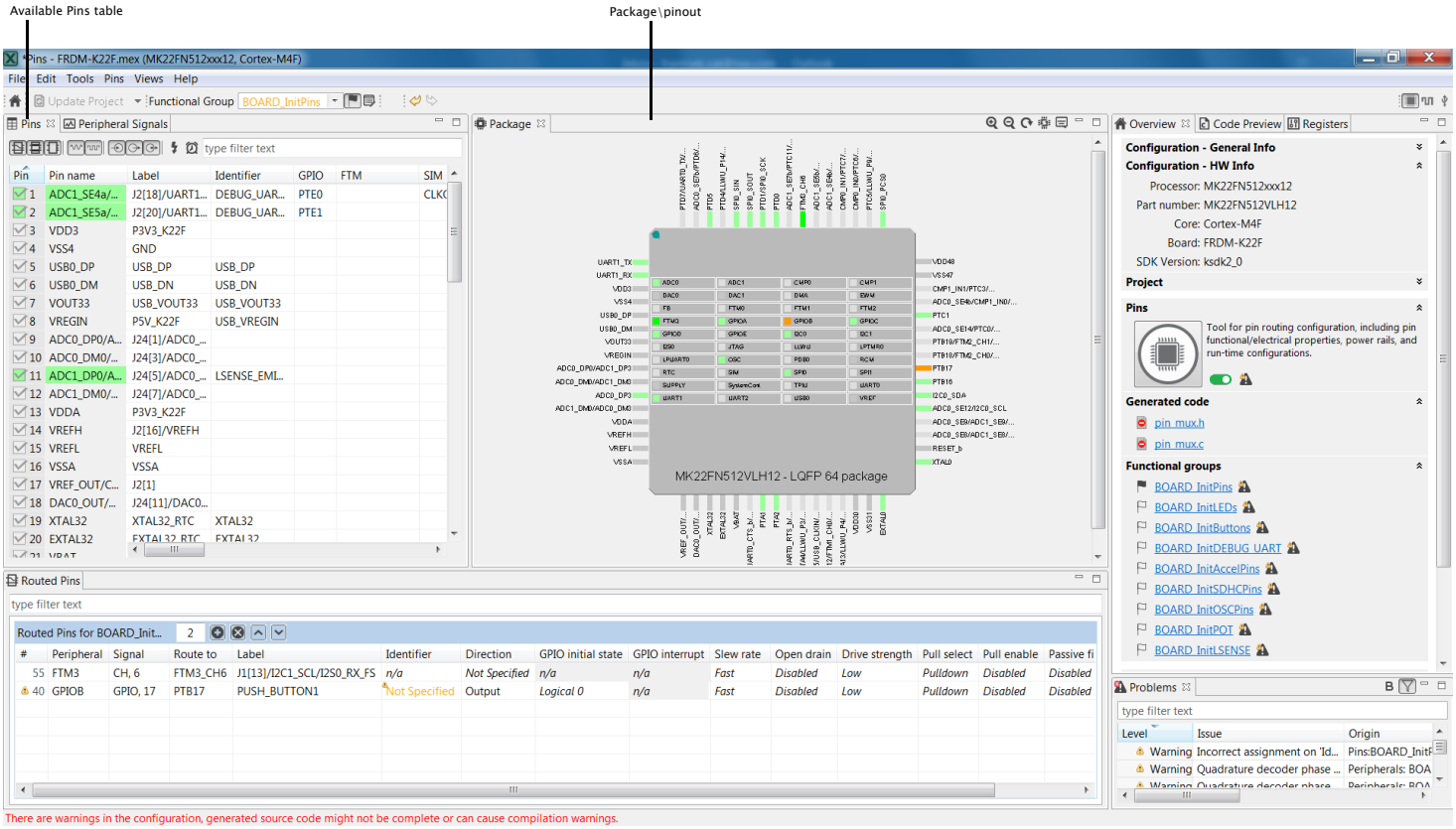


Figure 21. Pins Tool user interface

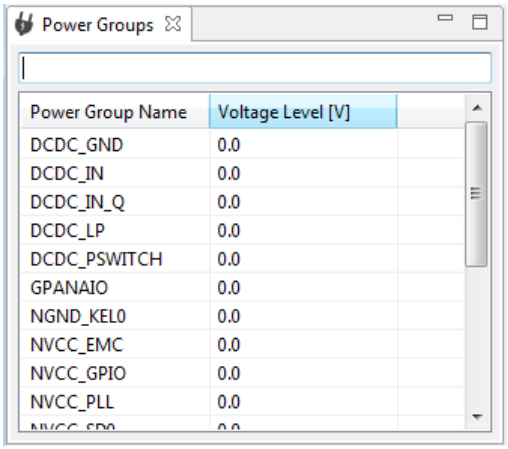


Figure 22. Selecting power group

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.

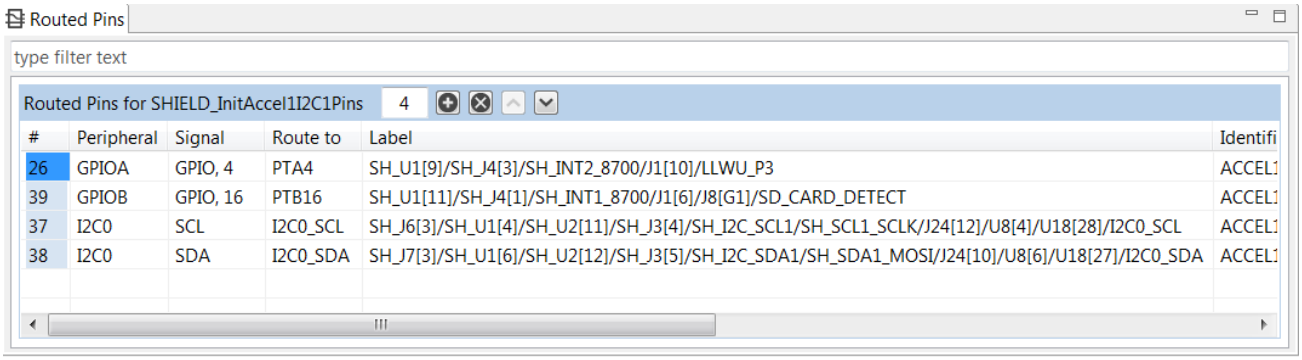


Figure 23. Routed Pins view

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 the resources allocation.

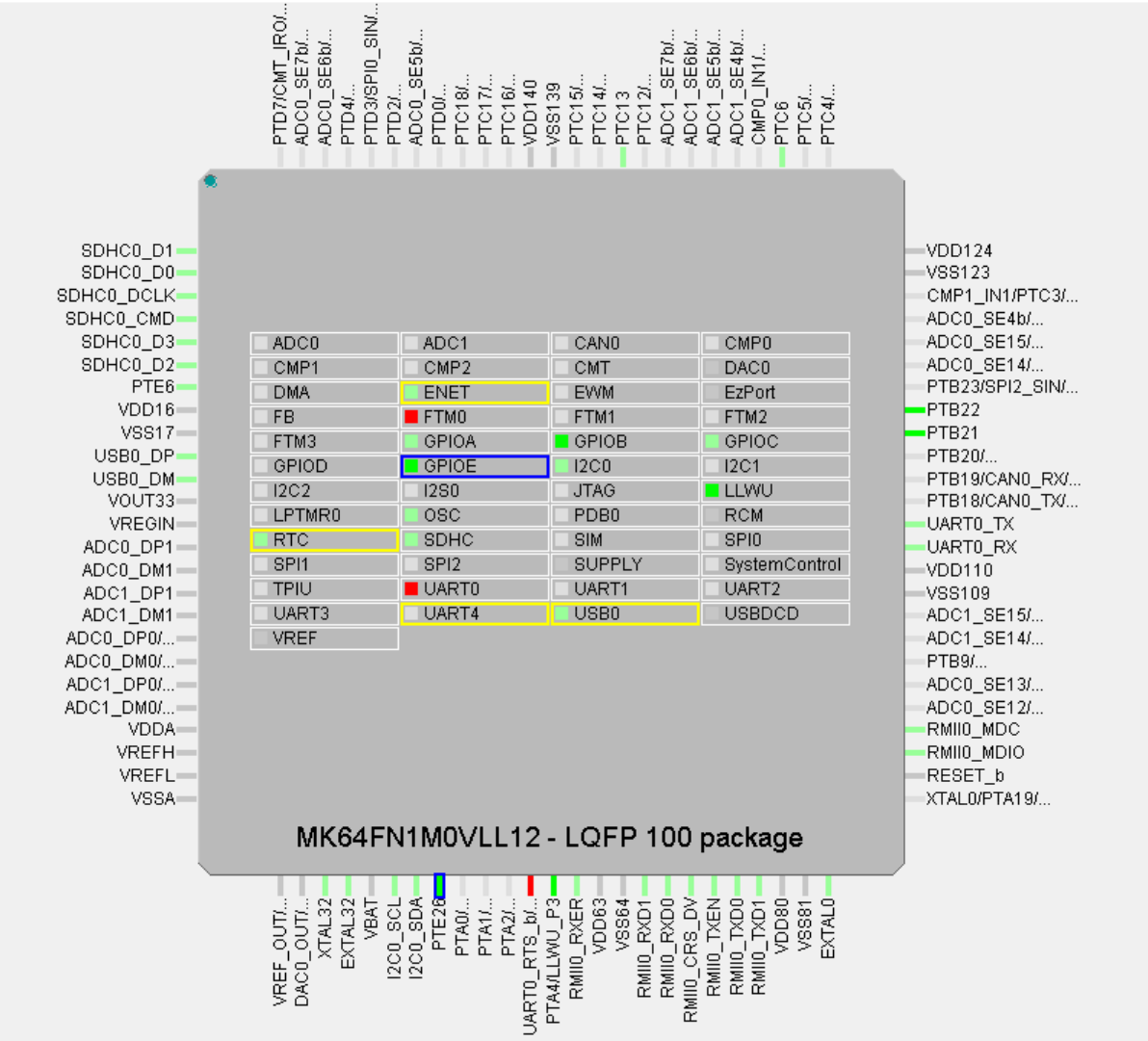


Figure 24. Processor package

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**.

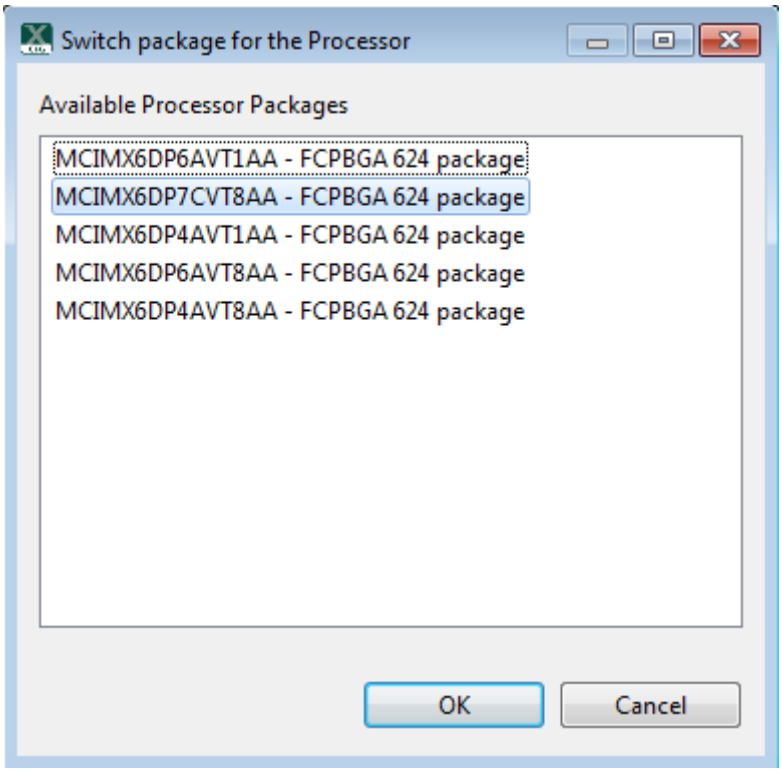


Figure 25. Switch package

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 shows a list of routed pins and allows 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 configured. Each row represents configuration of one 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 select required signal, and finally select the routed pin.

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

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

#	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 26. 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.

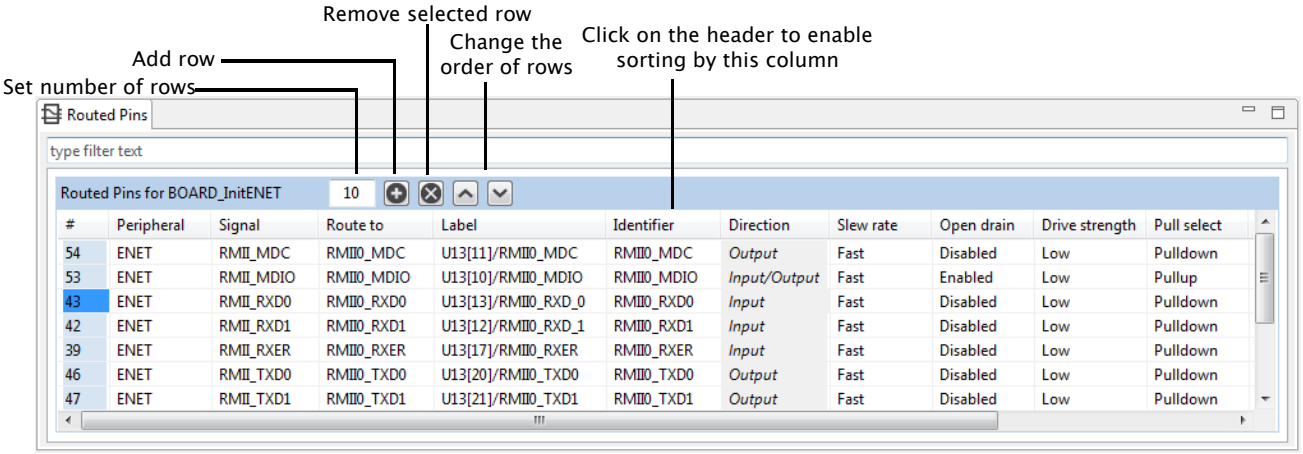


Figure 27. View controls

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.

3.4.3.2 Filtering routed pins

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

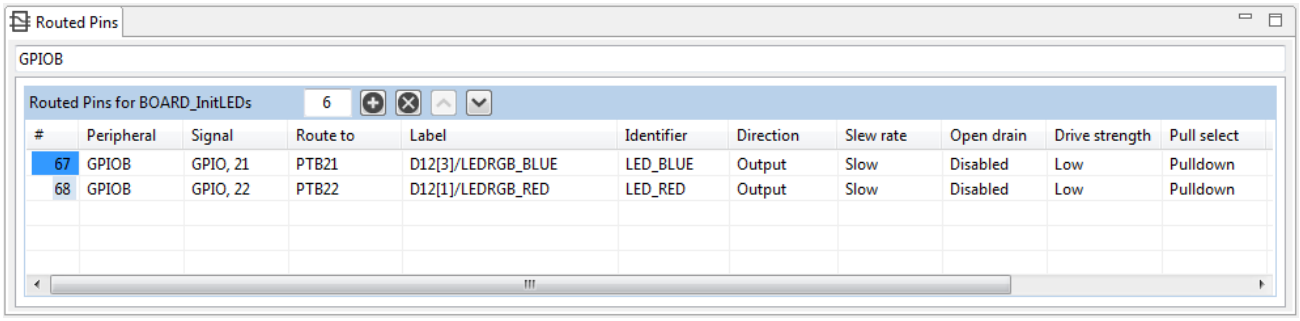


Figure 28. Filter area

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.

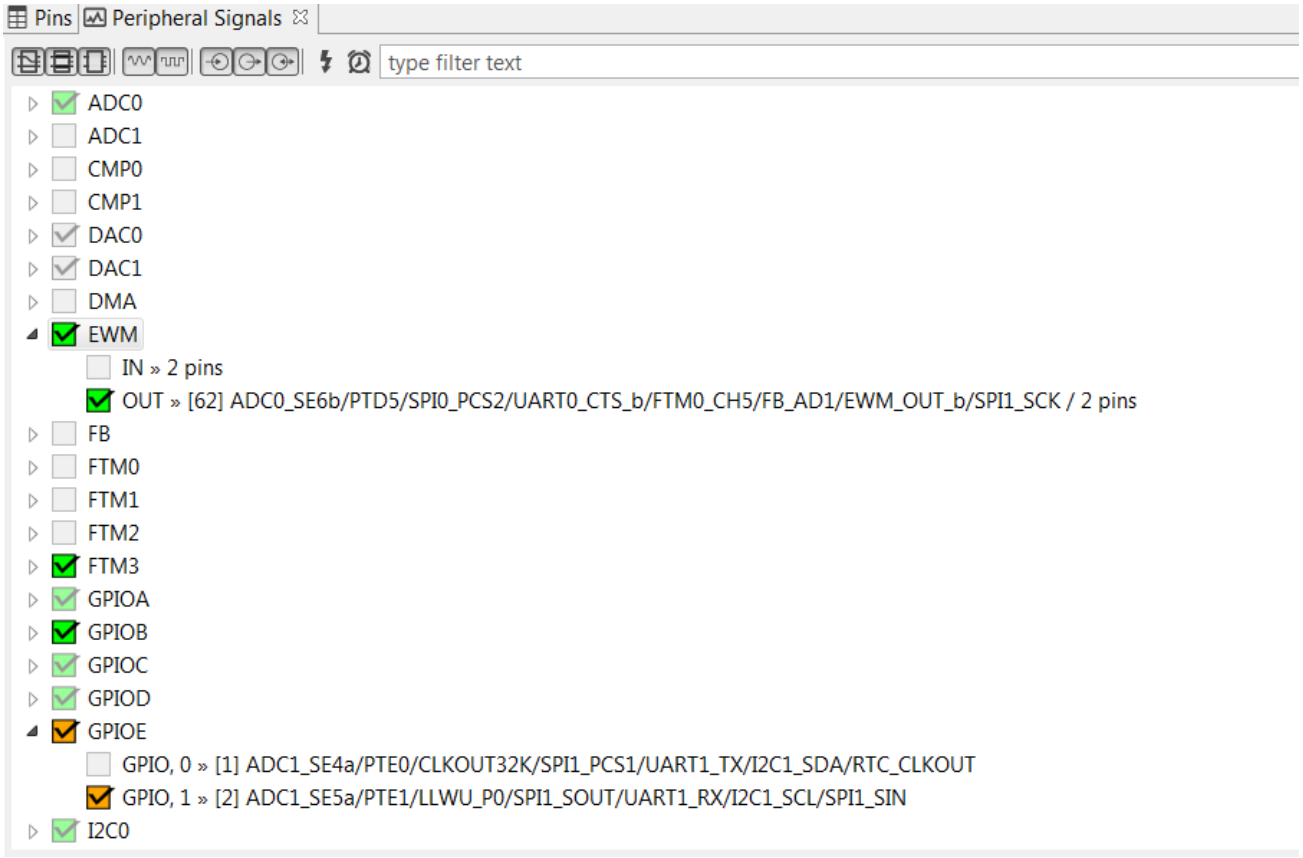


Figure 29. Peripheral Signals view

Use the checkbox to route/unroute the selected pins.

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

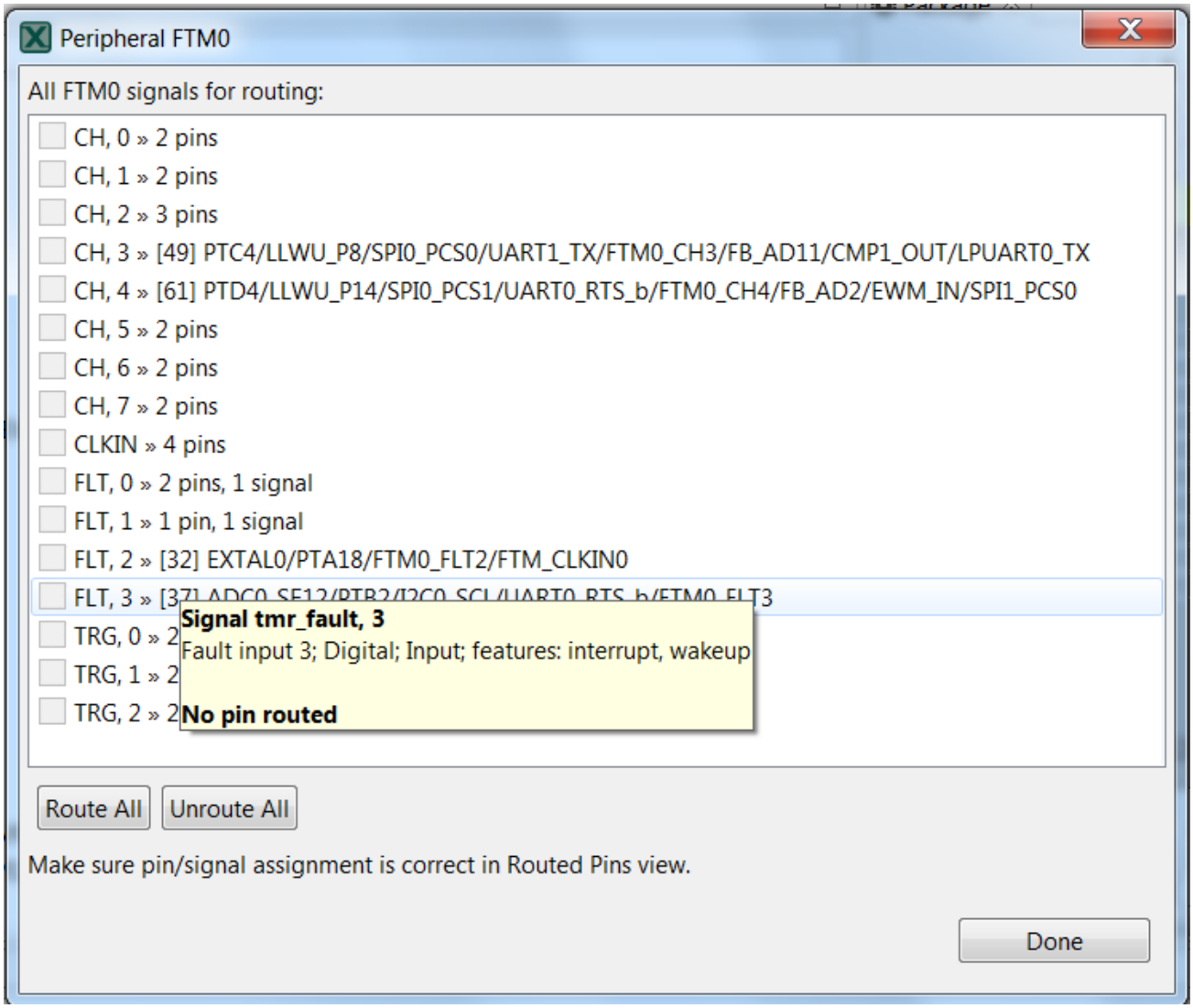


Figure 30. Select signals dialog

3.4.5 Pins table view

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

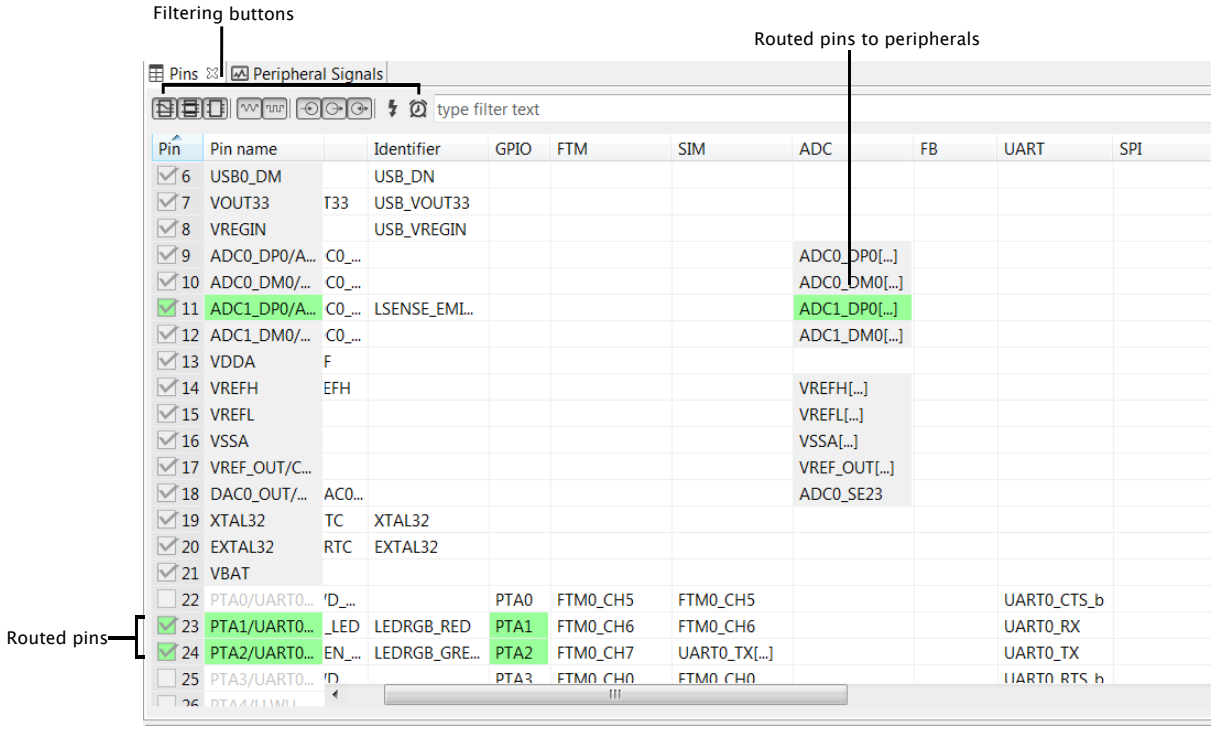


Figure 31. Pins table view

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

If more signals can be routed to one pin, it is indicated by […]. The **Multiple Signals Configuration** dialog appears, if clicked.

3.4.5.1 Labels and identifiers

It is 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 is 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.

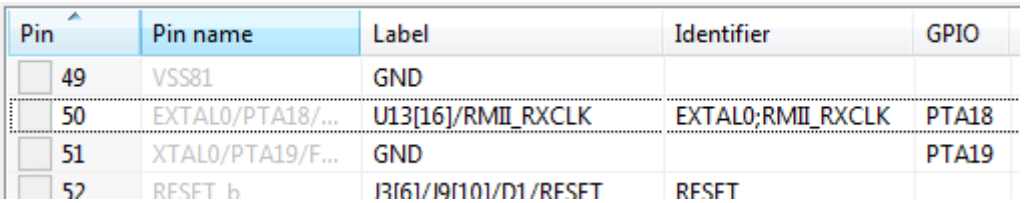


Figure 32. Pin Identifier

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

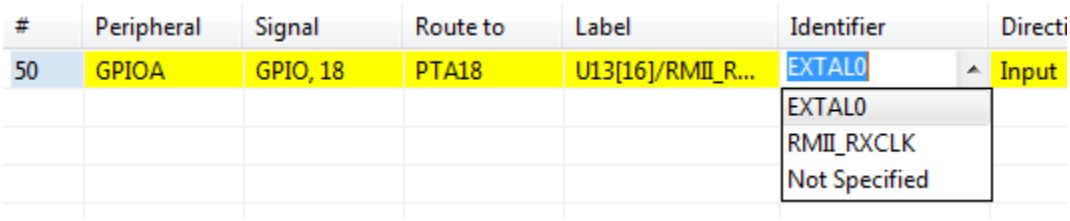


Figure 33. Identifier in Routed Pins table

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.

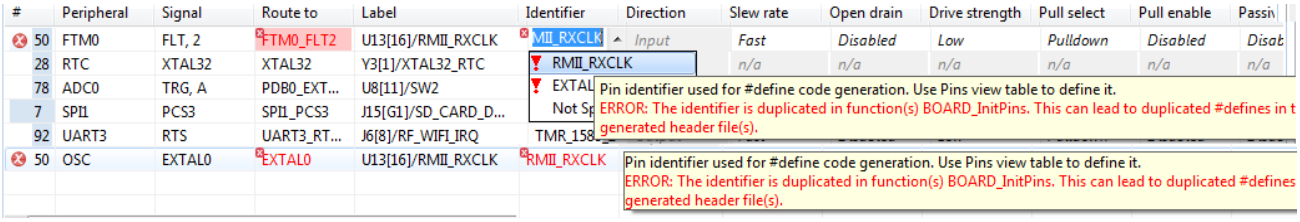


Figure 34. Identifier errors

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.

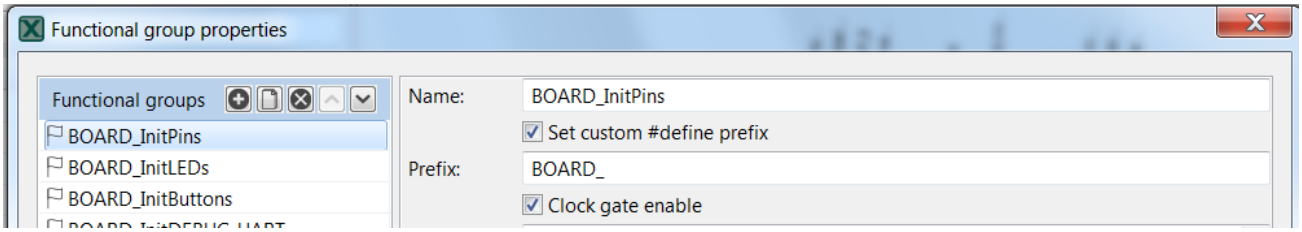


Figure 35. Pins macros prefix

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.

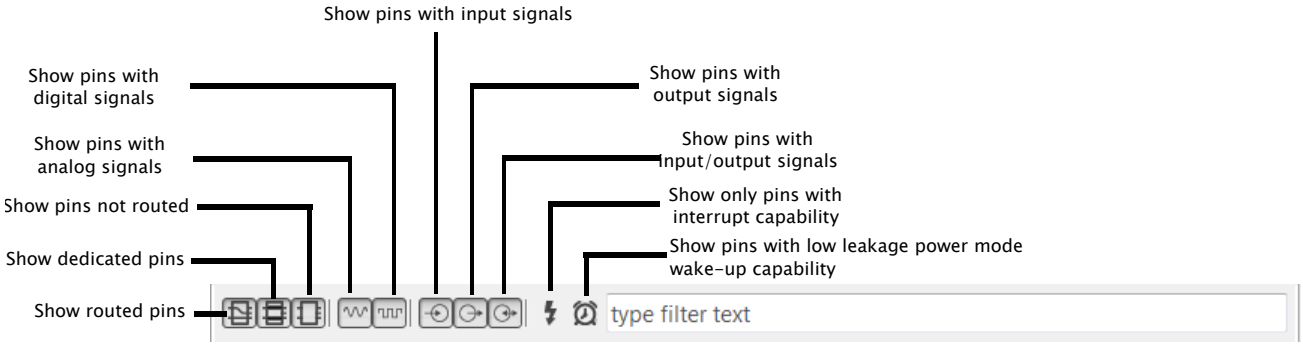


Figure 36. Filtering Controls

Type any text to search across the table/tree. It will search for the pins/peripheral signals containing the specified text.

3.4.7 Highlighting and color coding

It is 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.

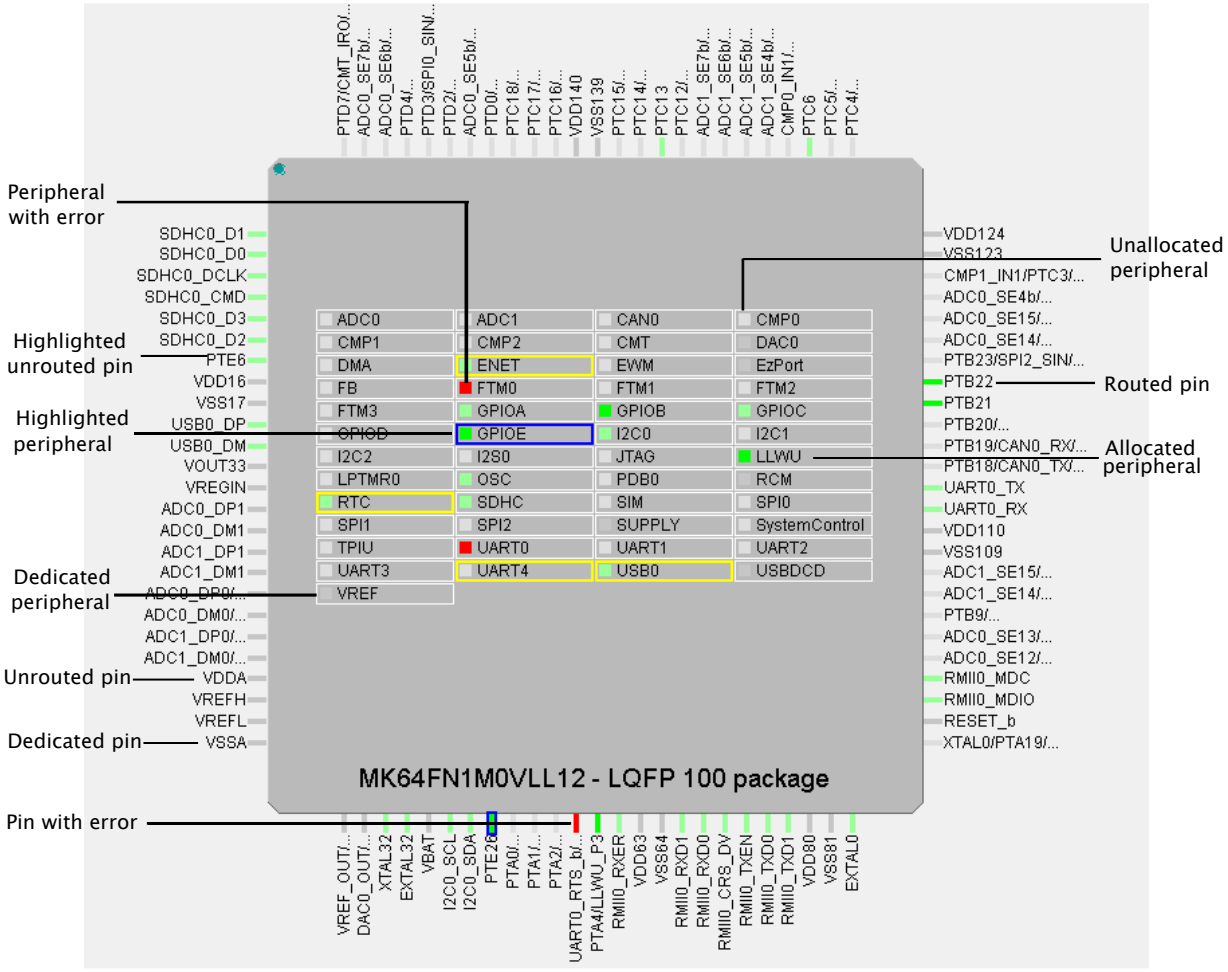


Figure 37. Highlighting and color coding

#	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 38. 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 39. 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 40. 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, refer to the Highlighting and Color Coding the 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 41. 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 the main menu **Pins > Refresh**. 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.

It is also possible to 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 YAML format. This comment is not intended for direct editing and can be used later to re-store the pins configuration.

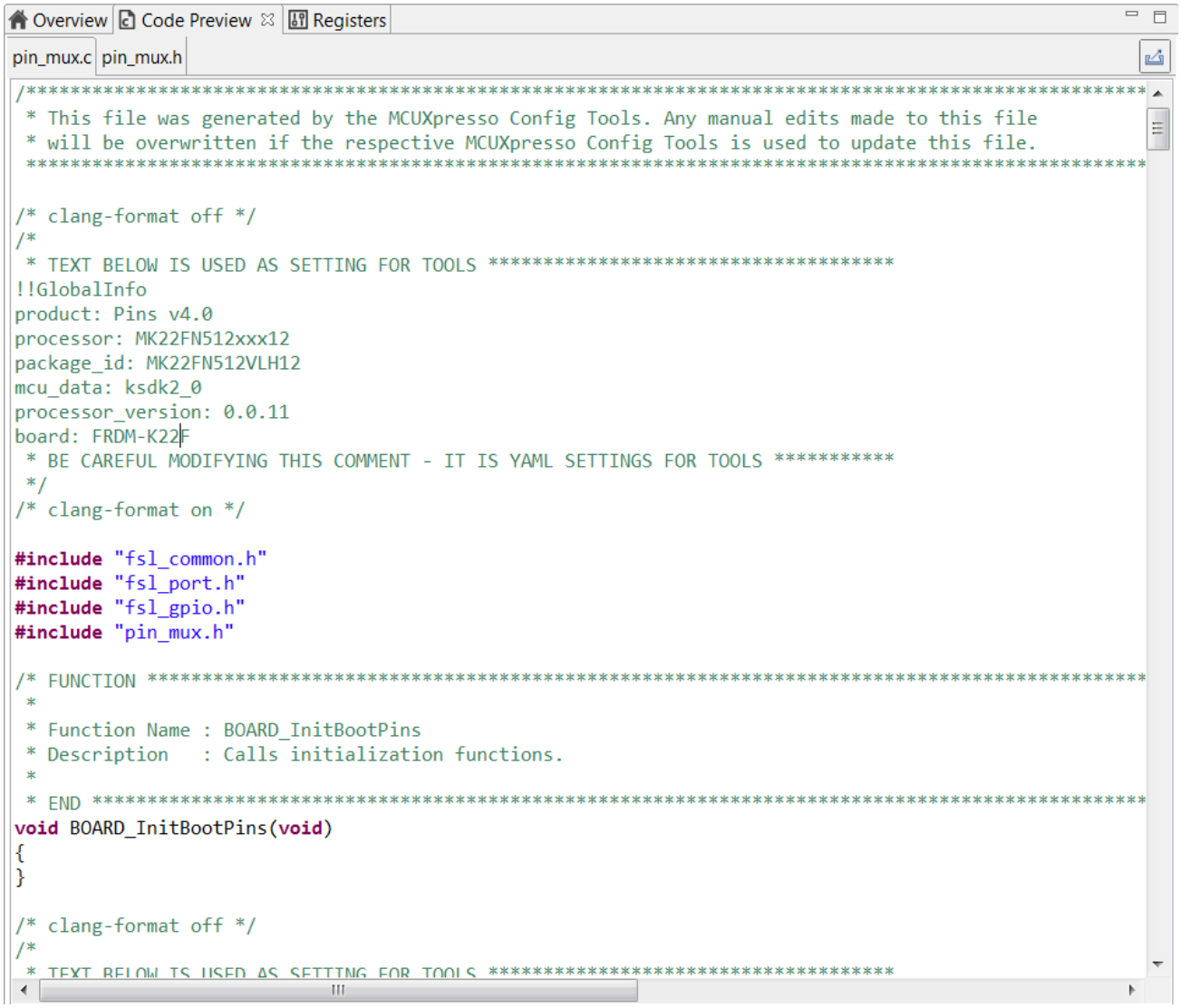


Figure 42. Generated code

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

The following are the Clock tool features:

- 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 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.

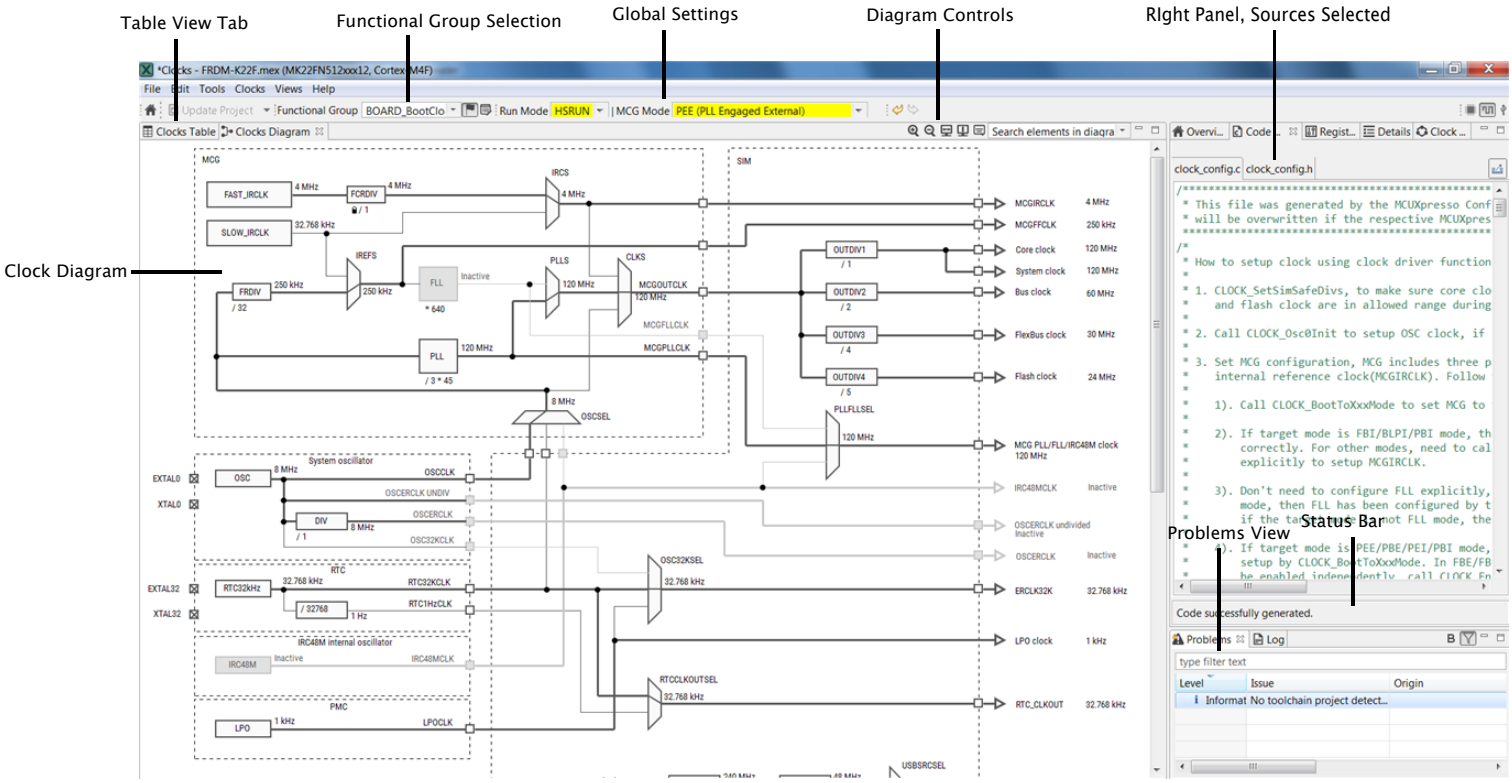


Figure 43. User interface

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 is 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 these clocks configurations, add more configurations using the '+' button, and remove configurations using '-' button.

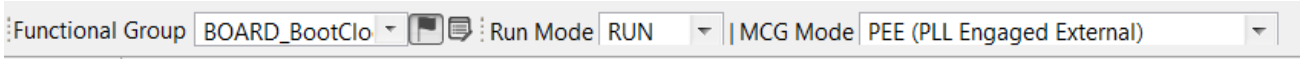


Figure 44. Default clock configuration

NOTE

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

4.4 Global settings

The global settings are the settings that influence the entire clock system. It is recommended to start with these settings, but they can be changed later.

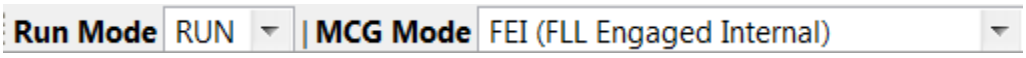


Figure 45. Global settings

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 the 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.

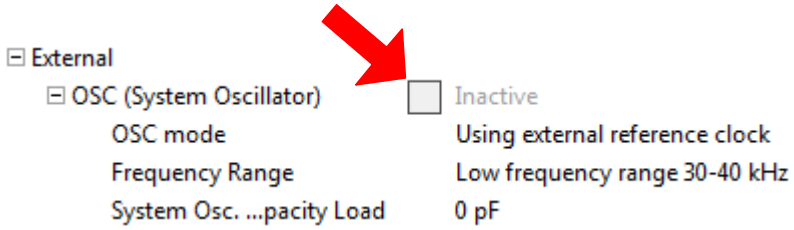


Figure 46. External clock source configuration

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.

Table continues on the next page...

Table 5. Setting states and markers (continued)

State/Style/ Marker	Icon	Description
Warning marker		Indicates that there is a warning in the settings or something related to it. See the tool-tip 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		Indicates that the field is directly or indirectly changed by the previous user action.
Gray text		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 value shown 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 is shortened and only two decimal places are shown with ellipsis ‘…’ character 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 the **Enter** key.

Figure 47. Setting the core clock frequency

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

Figure 48. Tool attains the required frequency

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

Figure 49. 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 50. Pop-up menu commands

4.7.2 Frequency precision

For the locked frequency settings (user indicated a requested value) the frequency precision value is also shown. By default, the value is 0.1% but can be individually adjusted by clicking on the value.

Figure 51. 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.

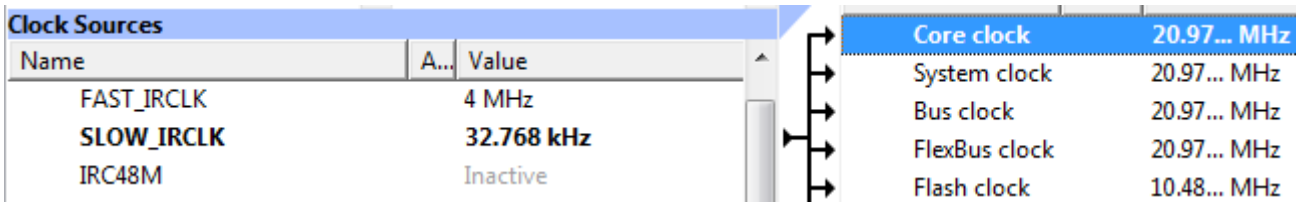


Figure 52. Dependency arrows

4.9 Details view

The **Details** view contains a list of settings on the selected element, clock path, component, or on the entire processor.

The content of the **Details** view depends on the selected element and can be one of the following.

- **Clock element** - Lists the settings of the selected clock source, prescaler, and so on.
- **Clock path** - Lists the settings of the element on the path from the selected output to used clock source.
- **Component** - Lists the settings for all elements located in the selected component.
- **Processor** - Lists all the settings related to the selected processor.

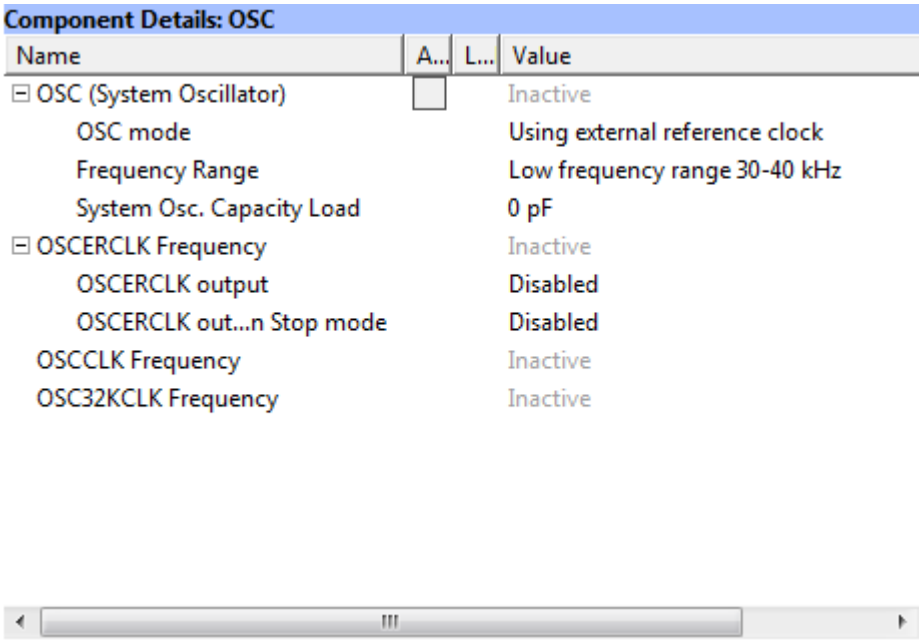


Figure 53. Details view

4.10 Clock diagram

The clock diagram shows the complete structure of the 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 is dynamically refreshed after every change and reflects the current state of the clock model.

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

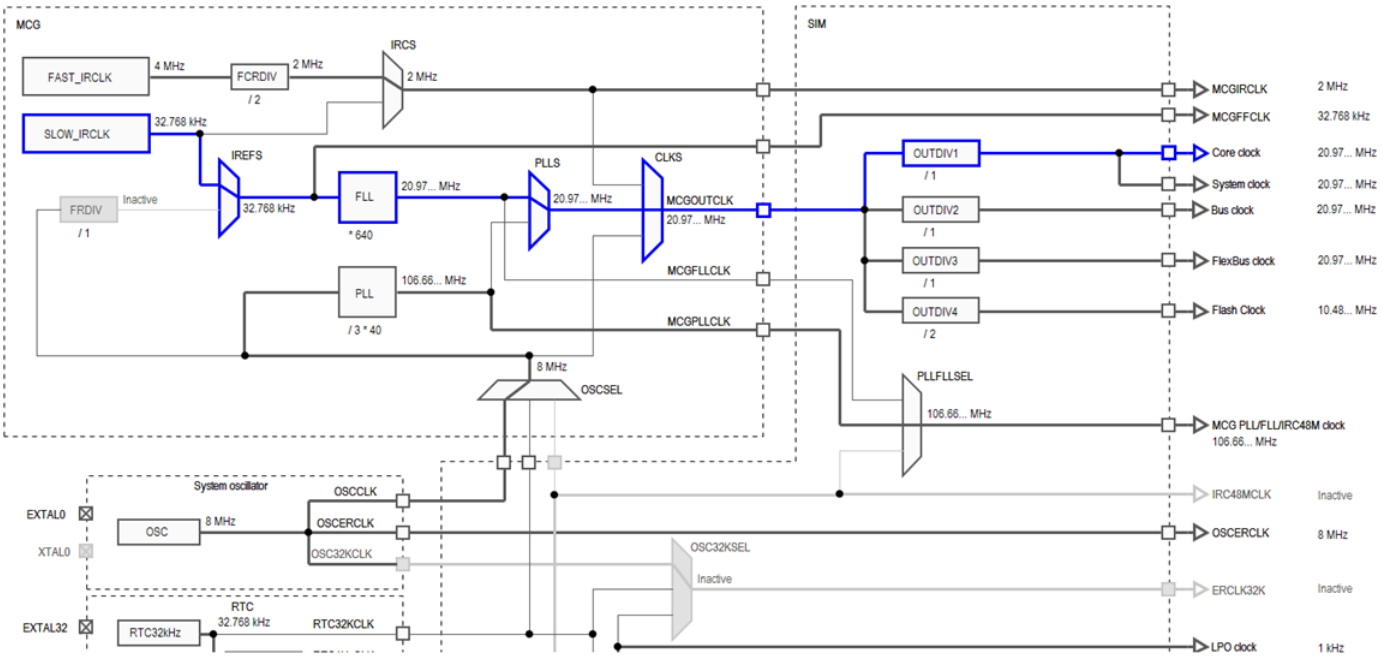


Figure 54. Clock diagram

4.10.1 Mouse actions in diagram

The following interactions are available in Clock diagram view.

- **Move 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.
- **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.

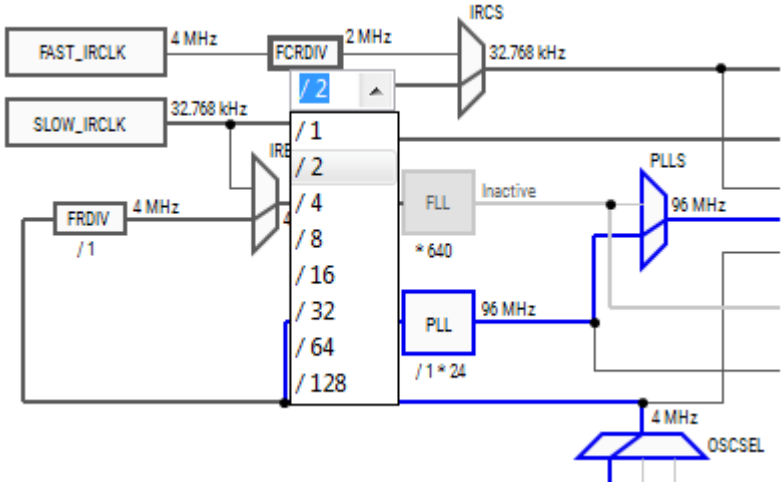


Figure 55. Clocks mouse actions in diagram

- **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.

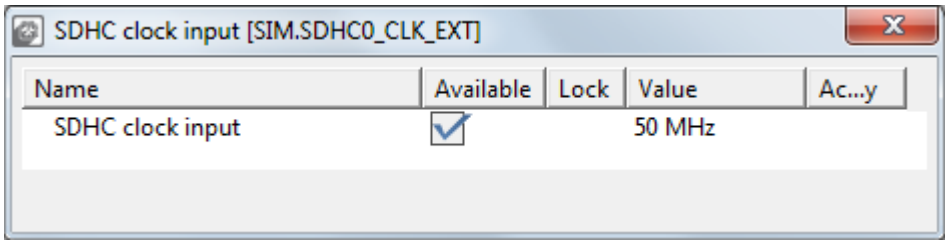


Figure 56. Floating view

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

4.10.3 Clock model structure

The clock model consists of the clock elements that are interconnected. The clock signal flows from the clock sources through the various clock elements to the clock outputs. The clock element can have specific enable conditions that can stop the signal from passing it to the successor. The clock element can also have specific constraints and limits that are watched by the clocks tool. To get these details, put the cursor on the element in the clock diagram and see its tooltip.

The following are the clock model elements.

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

Figure 57. Clock source

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

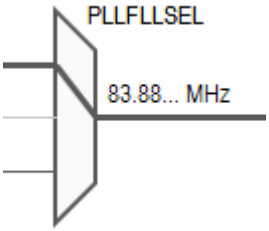


Figure 58. Clocks selector

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

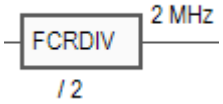


Figure 59. Prescaler

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

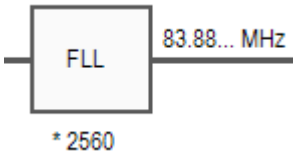


Figure 60. Frequency Locked Loop

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

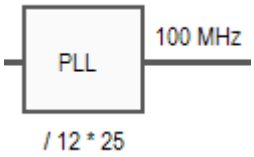


Figure 61. Phase Locaked Loop

- **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.

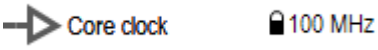


Figure 62. Clock output

- **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.

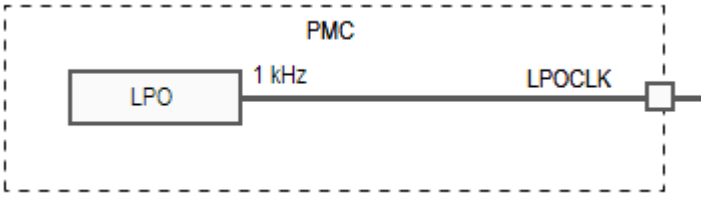


Figure 63. Clock component

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

4.11 Main menu

The commands related to Clocks are present in the **Clocks** menu and include the following commands:

- **Functional groups** – Invokes the **Functional group properties** dialog.
- **Unlock All Settings** – Unlocks all locks in all settings.

- **Reset To Board Defaults** – Resets the clock model to board defaults.
- **Reset To Processor Defaults** – Resets the clock model ito processor defaults.
- **Refresh** – Refreshes each clocks configuration with explicit invocation of code generation.

4.12 Troubleshooting problems

It is possible that while working with the tool some problems or mismatches occur. Such problems and the overall status is indicated in red on the central status bar of the tool. The status bar displays the 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 some settings is not valid. For example: The current state of settings is beyond the acceptable range.

The following are some tips to troubleshoot the 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 is 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.

clock_config.c | clock_config.h

```
/*FUNCTION**********************************************************************
 *
 * Function Name : CLOCK_CONFIG_SetFllExtRefDiv
 * Description   : Configure FLL external reference divider (FRDIV).
 * Param frdiv   : The value to set FRDIV.
 *
 *END**********************************************************************/
static void CLOCK_CONFIG_SetFllExtRefDiv(uint8_t frdiv)
{
    MCG->C1 = ((MCG->C1 & ~MCG_C1_FRDIV_MASK) | MCG_C1_FRDIV(frdiv));
}

/*******************************************************************************
 ************************ BOARD_InitBootClocks function ************************
 ******************************************************************************/
void BOARD_InitBootClocks(void)
{
    BOARD_BootClockHSRUN();
}

/*******************************************************************************
 ********************** Configuration BOARD_BootClockRUN ***********************
 ******************************************************************************/
/* clang-format off */
/* TEXT BELOW IS USED AS SETTING FOR TOOLS *************************************
!!Configuration
name: BOARD_BootClockRUN
outputs:
```

Figure 64. 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 is 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 the peripheral instances. It also provides the information on which clock can be consumed by the particular clock instance. This view is not editable and is for information only.

NOTE

The information on which peripherals are consuming a particular output clock is available in the clock output tooltip.

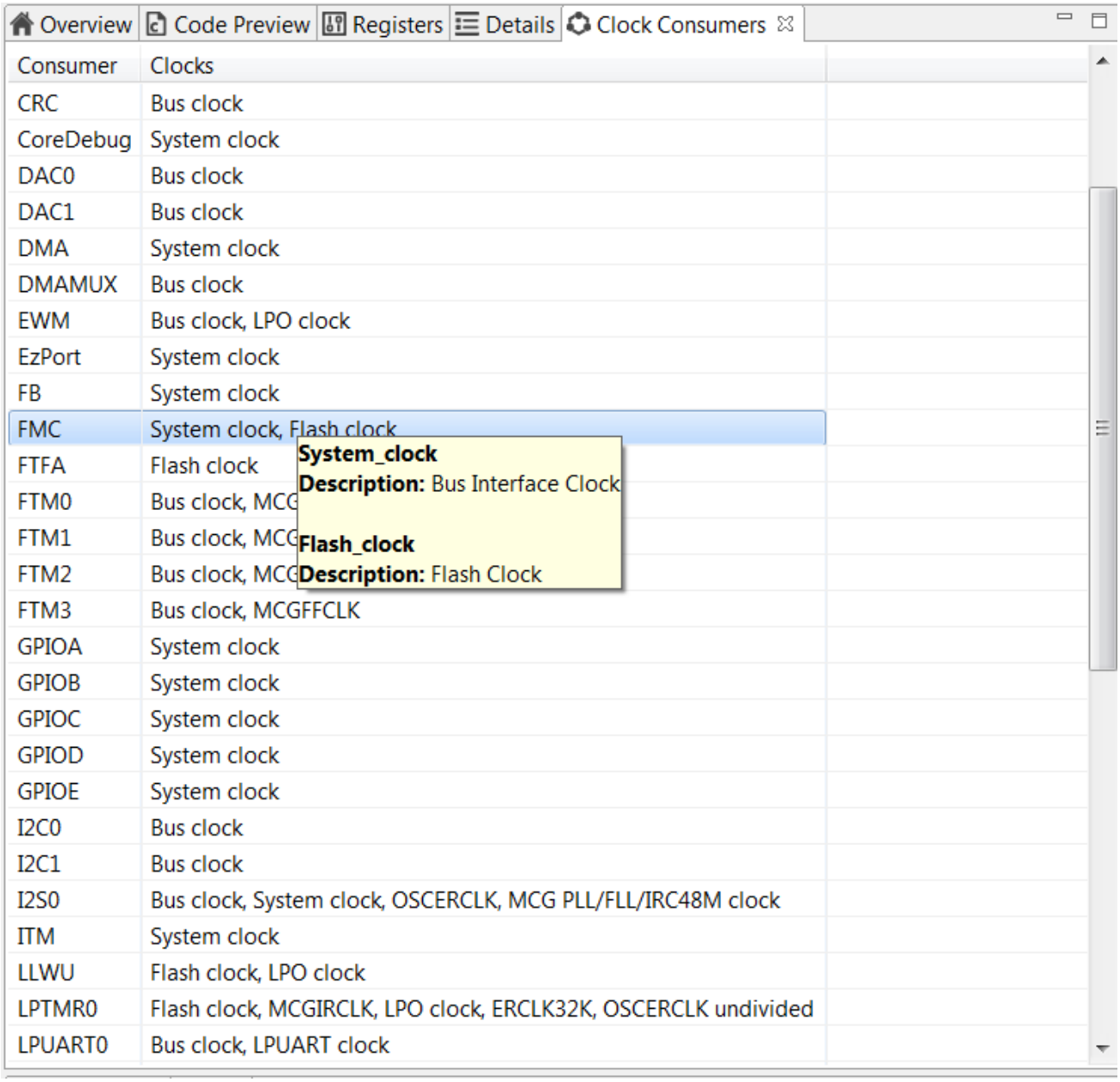


Figure 65. Clock Consumers view

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 function groups 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

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 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 located in the **Code Preview** view.

7. Settings can be saved to the **.mex** file (used for all settings of all tools) using the command **File > Save**.

5.4 User interface overview

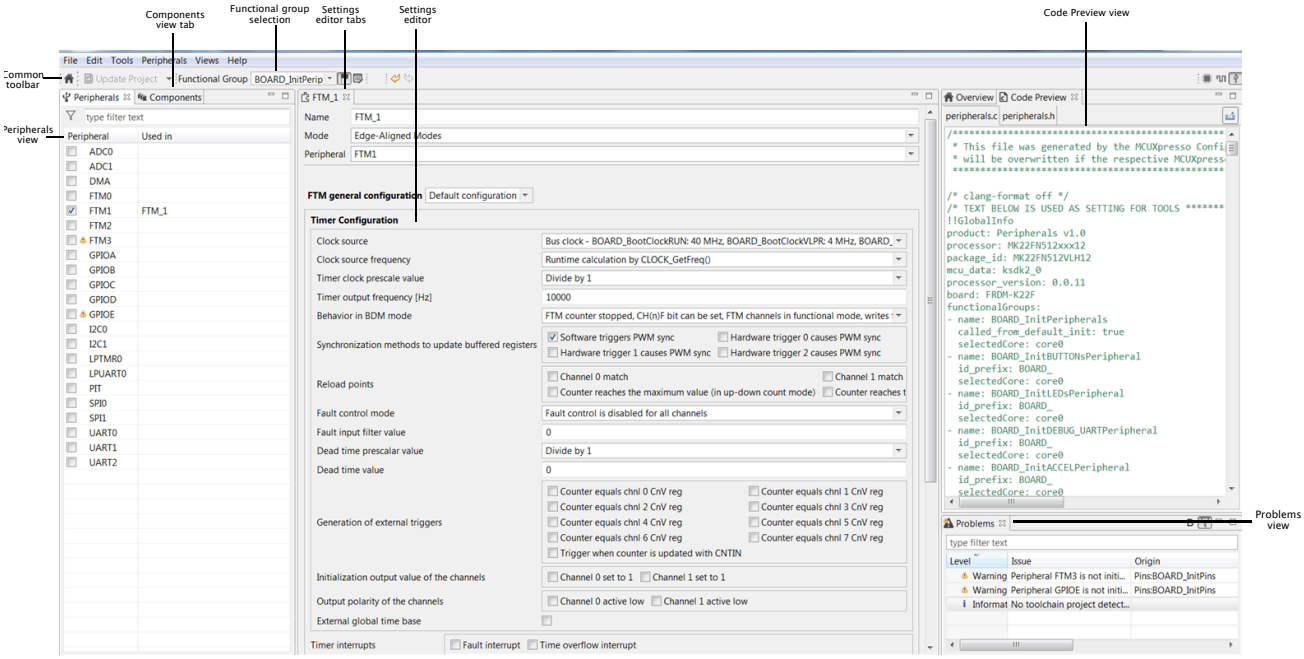


Figure 66. User interface

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 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 (e.g. UART0) occupies one row. First column contains peripheral name and a checkbox indicating whether the peripheral is used by any component instance.

Checking the check-box adds a new instance of the component and sets it to configure the selected peripheral 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.

5.7 Components view

The components view shows a tree of the configuration components and their instances under each component name. It shows all configuration components with instances, including the ones that do not use any peripheral and are not visible within the Peripherals view.

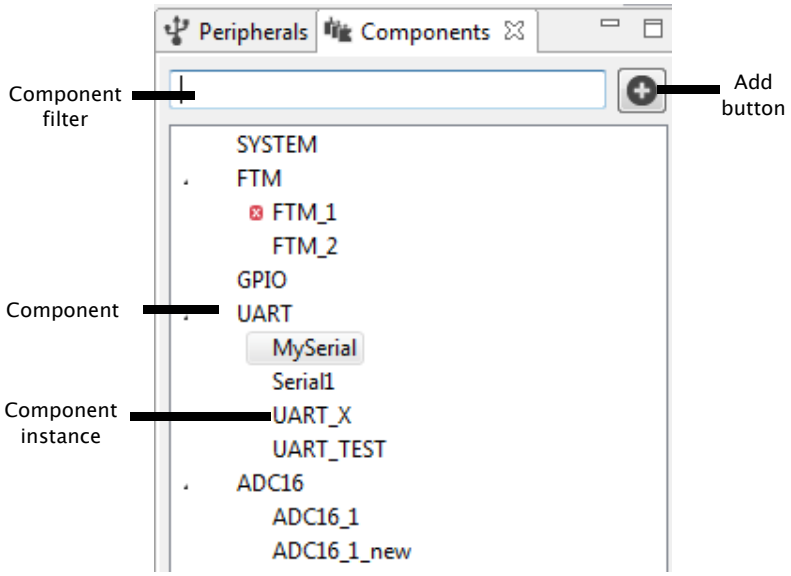


Figure 67. Component view

To add a new component instance, click the **Add button** to open the component selection dialog. It shows all components available for the currently selected processor. Select a component and click the **OK** button to confirm.

Component filter allows to write any text that is searched within the component names and their instance names and only the ones that contain the entered text are shown.

Mouse actions:

- Double click on the component name to open global settings for component
- Double click on the component instance name to open the instance settings (e.g. MySerial)
- Right click on component to open the context menu with the following command:
  - **Remove** – removes the component completely from the configuration (all functional groups) including all its instances. A confirmation dialog is shown asking the user to confirm the action.
- Right click on instance of a component the context menu with the following commands:
  - **Remove** – removes the component instance from the current functional group. A confirmation dialog is shown asking the user to confirm the action.

NOTE

If the component has some global settings, a dialog confirms whether you want to remove component or not. If the component does not have any global settings, the component is deleted after removing the last instance.

  - **Disable** – disables the component instance so it's not used for code generation and its errors are not reported.
  - **Move to** – shows selection of functional groups and if you select a function group, the instance is moved there.
  - **Copy to** – shows selection of functional groups and if you select a functional group it creates a copy of the instance in that functional group.

NOTE

The SYSTEM component is a special global-only component that provides common infrastructure shared by other components. It is automatically added to the configuration and cannot be removed.

5.8 Settings editor

To open the Settings Editor dialog:

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

  -or-

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

The open editors are shown in the central area of the screen, each of them has its own tab. There can be multiple editors opened at the same time.

Changes done in the editor are immediately applied and kept regardless the settings editor is closed.

Settings that are disabled are grayed. In case that a component instance is disabled, all settings are grayed.

Tool-tips are provided for all settings that are not disabled when the mouse cursor is placed at settings.

5.8.1 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.

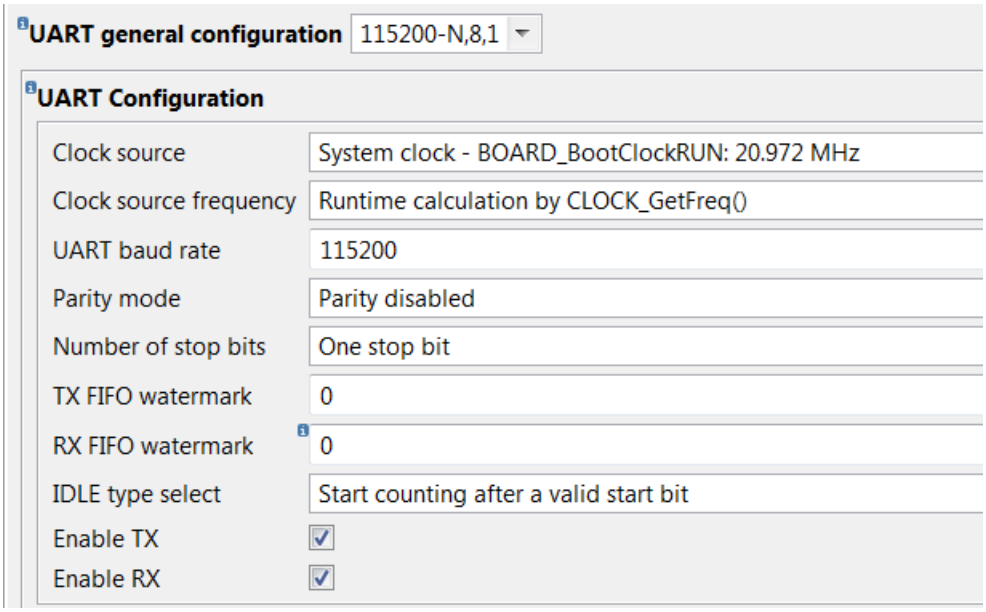


Figure 68. Quick selection example

5.8.2 Settings

The following settings occur in the editor.

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

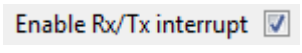


Figure 69. Boolean setting example

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

Figure 70. Integer/Float setting example

- **String** – textual input.

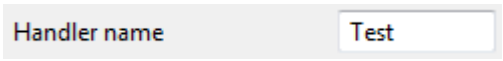


Figure 71. String setting example

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

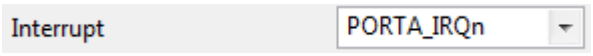


Figure 72. Enumeration setting example

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

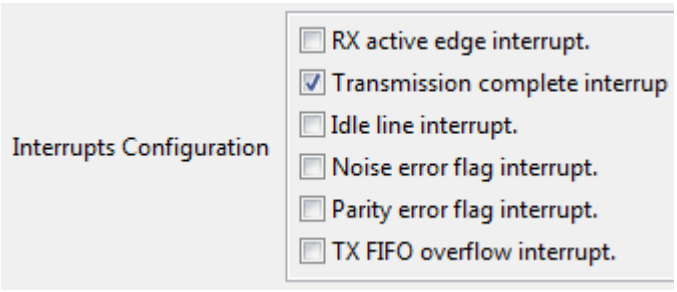


Figure 73. Set setting example

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

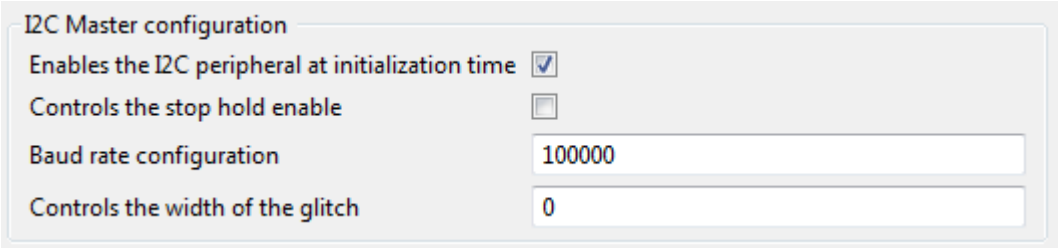


Figure 74. Structure setting example

- **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.

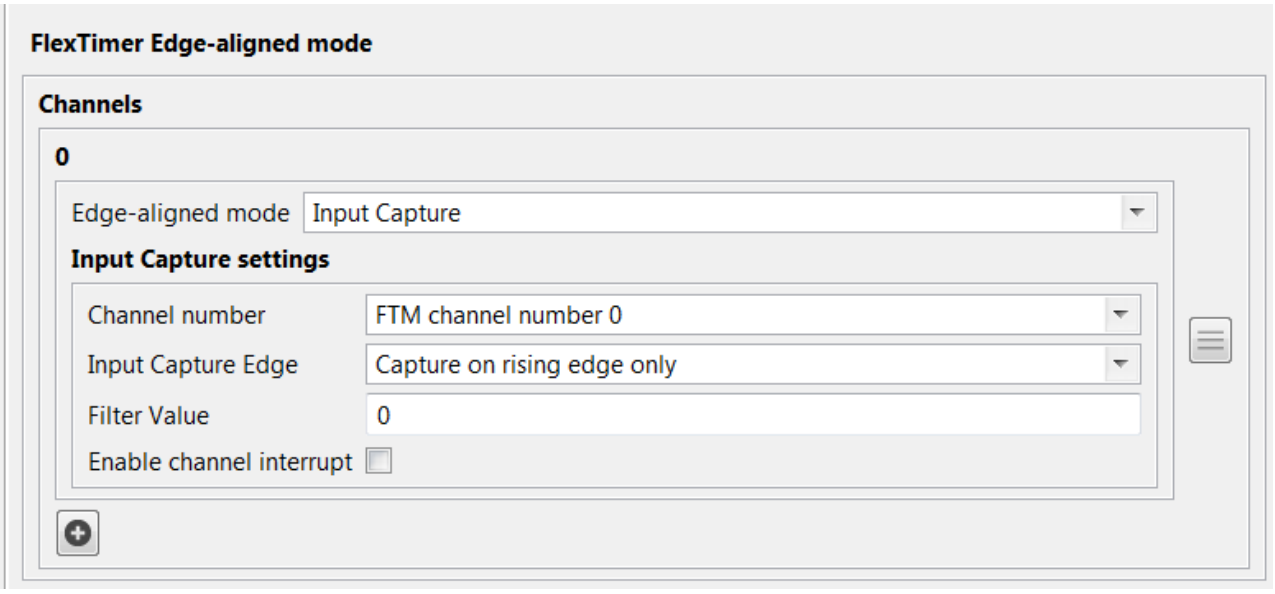


Figure 75. Array setting example

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

5.9 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.

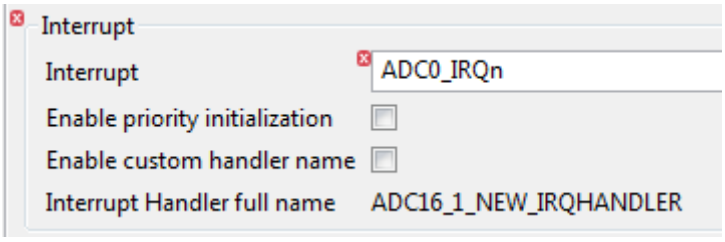


Figure 76. Error decorators

5.10 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 processor

It is possible to switch the processor or the package of the current configuration to a different one. However, if switched to a completely different processor, it may lead to a conflict or problems. For example, inaccessible pin routing or unsatisfiable clock output frequency. It is necessary to fix the problems manually. For example, go to the Pins Routing table and re-configure all pins which reports an error or conflicts. 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.

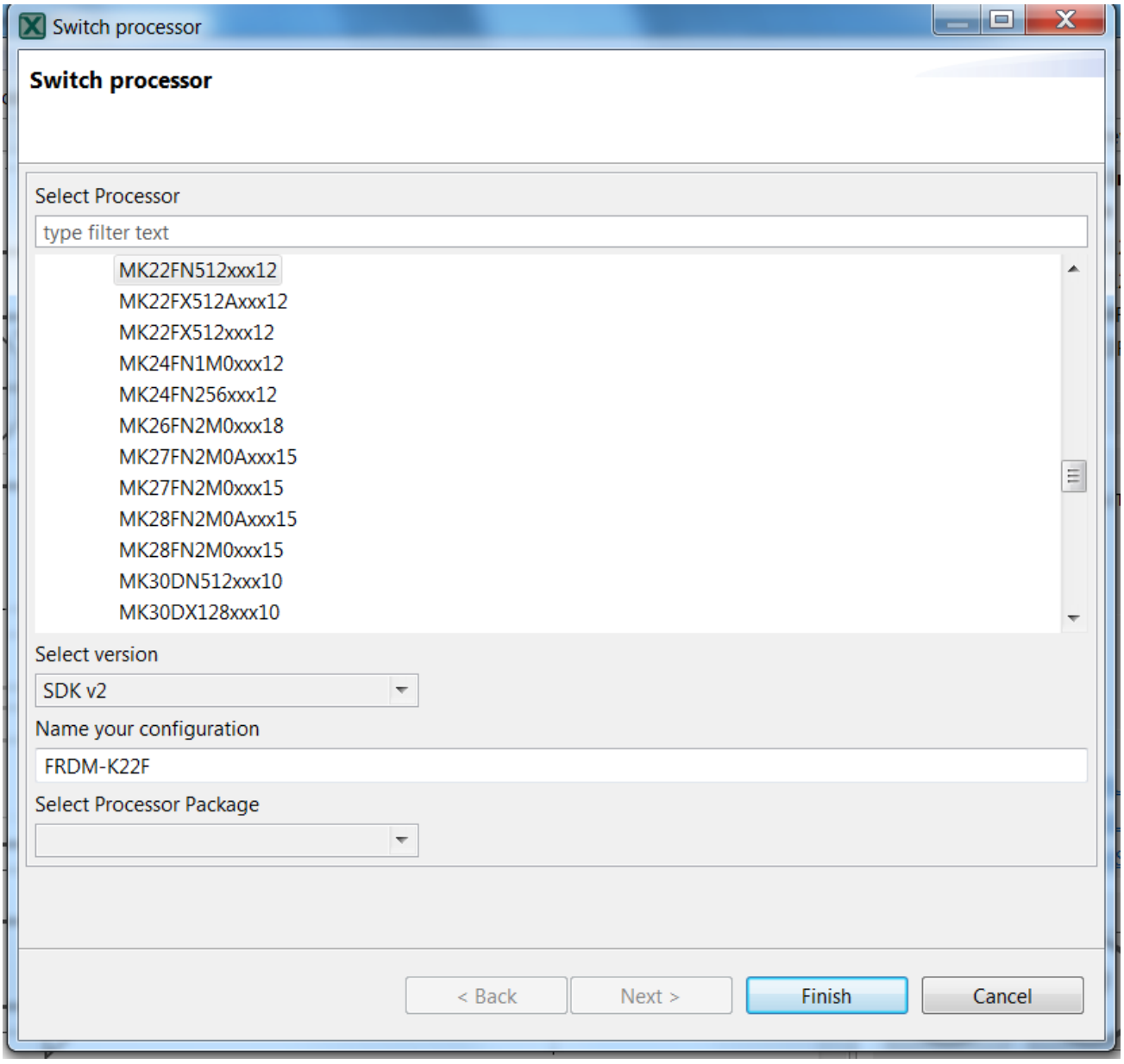


Figure 77. Switch processor

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

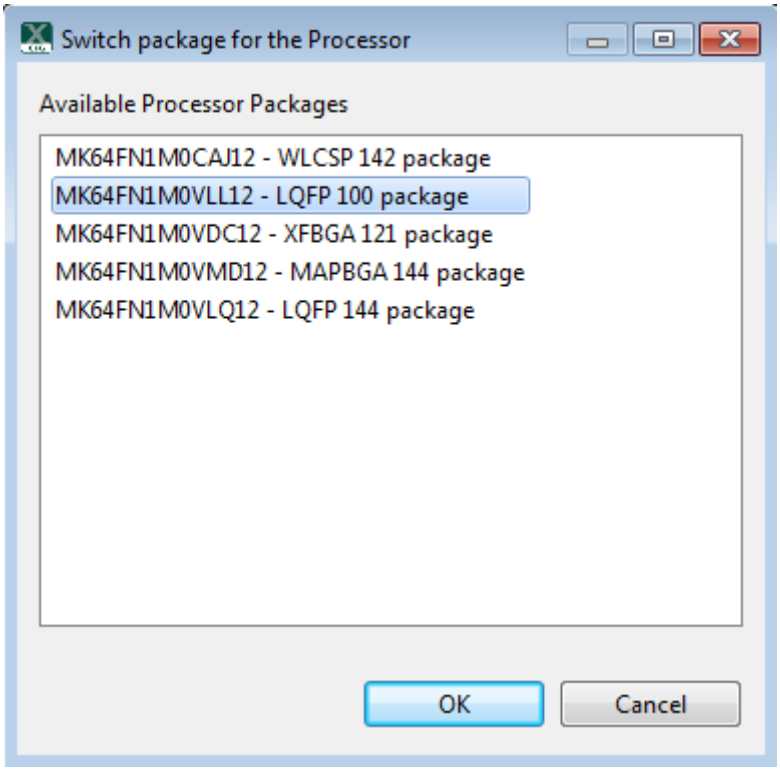


Figure 78. Switch package

6.2 Exporting Pins table

To export Pins table:

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 79. 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 80. Aligning to block

6.3 Downloading processor data

By default, only MK64 data is present after installing **MCUXpresso Config Tools**.

The data is downloaded on request during the Creating a new configuration process.

To explicitly download the available data to a local computer for offline work:

1. Select **File > Export**.
2. In the **Export** dialog, select **Processor Data > Download processor data**.
3. Click **Next**.
4. Select the series of data you want to download. Optionally, specify a custom download location.
5. Click **Next**. The download process will start. If you want to use the data on a computer without any internet connection at all, perform the previous procedure and manually copy the downloaded data folders from the specified download location to the desired computer. The data is saved to `<disk>/ProgramData>NXP>mcu_data_v4.1 folder`.

NOTE

In Windows, the **Program Data** folder is hidden by default. You can view the folder by selecting the **Show hidden files, folders, and drives** option in **Control Panel > Folder Options > View** tab.

6.4 Tools advanced configuration

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

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

6.5 Generating HTML report

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

6.6 Export registers

It is possible to export the tool modified registers data content using the Export wizard.

To launch the **Export** registers wizard:

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].`

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

Table 6. Commands supported in the framework

Command name	Definition and parameters	Description	Restriction	Example

Table continues on the next page...

Table 6. Commands supported in the framework (continued)

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
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

Table continues on the next page...

Table 6. Commands supported in the framework (continued)

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

Table continues on the next page...

Table 7. Commands supported in Pins (continued)

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
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
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

Table continues on the next page...

Table 8. Commands supported in Clocks (continued)

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
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

Table continues on the next page...

Table 9. Commands supported in Peripherals Tool (continued)

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		-SDKpath c:\nxp\SDK_2.0_MKL43Z256xxx4

Table continues on the next page...

Table 10. Commands supported in Project Cloner (continued)

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}	-HeadlessTool PrjCloner -SDKpath c: \nxp \SDK_2.0_MKL43Z25 6xxx4 -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 .

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support