

# MCUXpresso Config Tools User's Guide (IDE)



# Contents

<b>Chapter 1 Introduction.....</b>	<b>5</b>
1.1 Versions.....	5
<b>Chapter 2 User Interface .....</b>	<b>7</b>
2.1 Creating, saving, and opening a configuration.....	7
2.1.1 Creating a new configuration.....	7
2.1.2 Saving a configuration.....	7
2.1.3 Importing sources.....	7
2.1.3.1 Importing configuration.....	8
2.1.3.2 Importing registers.....	8
2.1.4 Restoring configuration from source code.....	10
2.2 Toolbar.....	11
2.2.1 Eclipse project selection.....	11
2.2.2 Config Tools Overview.....	11
2.2.3 Show Problems View.....	11
2.2.4 Update code.....	12
2.2.5 Functional groups.....	14
2.2.5.1 Functional group properties.....	14
2.2.6 Undo/Redo actions.....	15
2.2.7 Selecting the tools.....	15
2.3 Status bar.....	15
2.4 Preferences.....	16
2.5 Configuration preferences.....	17
2.6 Problems view.....	18
2.7 Registers view.....	19
2.8 Log view.....	21
2.9 Config tools overview.....	21
<b>Chapter 3 Pins Tool.....</b>	<b>23</b>
3.1 Pins routing principle.....	23
3.1.1 Beginning with peripheral selection.....	23
3.1.2 Beginning with pin/internal signal selection.....	24
3.2 Workflow.....	24
3.3 Example usage.....	25
3.4 User interface.....	27
3.4.1 Functions.....	28
3.4.2 Package.....	29
3.4.3 Routed Pins view.....	31
3.4.3.1 View controls.....	32
3.4.3.2 Filtering routed pins.....	33
3.4.4 Peripheral Signals view.....	33
3.4.5 Pins view.....	35
3.4.6 Labels and identifiers.....	36
3.4.7 Filtering in the Pins and Peripheral Signals views.....	37
3.4.8 Highlighting and color coding.....	38
3.4.9 Expansion Header.....	40
3.5 Errors and warnings.....	44
3.5.1 Incomplete routing.....	45
3.6 Code generation.....	45

3.7 Using Pins Definitions in Code.....	46
<b>Chapter 4 Clocks Tool.....</b>	<b>48</b>
4.1 Features.....	48
4.2 User interface overview.....	48
4.3 Clock configuration.....	49
4.4 Global settings.....	49
4.5 Clock sources.....	50
4.6 Setting states and markers.....	50
4.7 Frequency settings.....	51
4.7.1 Pop-up menu commands.....	52
4.7.2 Frequency precision.....	52
4.8 Dependency arrows.....	52
4.9 Details view.....	52
4.10 Clock diagram.....	54
4.10.1 Mouse actions in diagram.....	54
4.10.2 Color and line styles.....	55
4.10.3 Clock model structure.....	55
4.11 Clocks menu.....	57
4.12 Troubleshooting problems.....	57
4.13 Code generation.....	58
4.13.1 Working with the code.....	59
4.14 Clock Consumers view.....	60
<b>Chapter 5 Peripherals Tool.....</b>	<b>61</b>
5.1 Features.....	61
5.2 Basic Terms and Definitions.....	61
5.3 Workflow.....	61
5.4 User interface overview.....	62
5.5 Common toolbar.....	62
5.6 Documentation view.....	63
5.7 Peripherals view.....	64
5.8 Components view.....	65
5.9 Settings Editor.....	67
5.9.1 Quick selections.....	68
5.9.2 Settings.....	68
5.9.3 Settings Editor header.....	70
5.10 SEMC Validation tool.....	70
5.10.1 Validation view.....	71
5.11 Problems.....	71
5.12 Code generation.....	72
<b>Chapter 6 Device Configuration Tool.....</b>	<b>74</b>
6.1 Device Configuration Data (DCD) view.....	74
6.1.1 Device Configuration Data (DCD) view actions.....	74
6.2 Code generation.....	76
<b>Chapter 7 Trusted Execution Environment Tool.....</b>	<b>77</b>
7.1 User Memory Regions.....	77
7.2 Security Access Configuration view.....	81
7.2.1 Masters/Slaves.....	81

7.2.2 MPC.....	83
7.2.3 Interrupts.....	84
7.2.4 Pins.....	85
7.2.5 SAU.....	86
7.2.6 Miscellaneous.....	87
7.3 Memory attribution map.....	88
7.3.1 Core 0.....	88
7.3.2 Other masters.....	90
7.4 Access Overview.....	91
7.5 Code Generation.....	92

## **Chapter 8 Advanced Features..... 94**

8.1 Switching the processor .....	94
8.2 Exporting the Pins table.....	95
8.3 Tools advanced configuration.....	96
8.4 Generating HTML report.....	96
8.5 Exporting sources.....	96
8.6 Exporting registers.....	98
8.7 Command line execution.....	98
8.7.1 Command line execution - Pins Tool.....	100
8.7.2 Command line execution - Clocks Tool.....	101
8.7.3 Command line execution - Peripherals Tool.....	102
8.7.4 Command line execution - Project Cloner.....	102
8.8 Managing data and working offline.....	103
8.8.1 Working offline.....	104
8.8.2 Downloading data.....	104
8.8.3 Exporting data.....	104
8.8.4 Importing data.....	105
8.8.5 Updating data.....	105

## **Chapter 9 Support..... 106**

# Chapter 1

## Introduction

The **MCUXpresso Config Tools** set is a suite of evaluation and configuration tools that help you from initial evaluation to production software development. Following tools are included:

**Table 1. MCUXpresso Config Tools**

Name	Description
<a href="#">Pins Tool</a>	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.
<a href="#">Clocks Tool</a>	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.
<a href="#">Peripherals Tool</a>	Enable you to configure the initialization for the MCUXpresso SDK drivers.
<a href="#">Device Configuration Tool</a>	Enables you to generate a Device Configuration Data (DCD) image using the format and constrains specified in the Boot ROM reference manual.
<a href="#">TEE Tool</a>	Enables you to configure security policies of memory areas, bus masters, and peripherals, in order to isolate and safeguard sensitive areas of your application.

### 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.

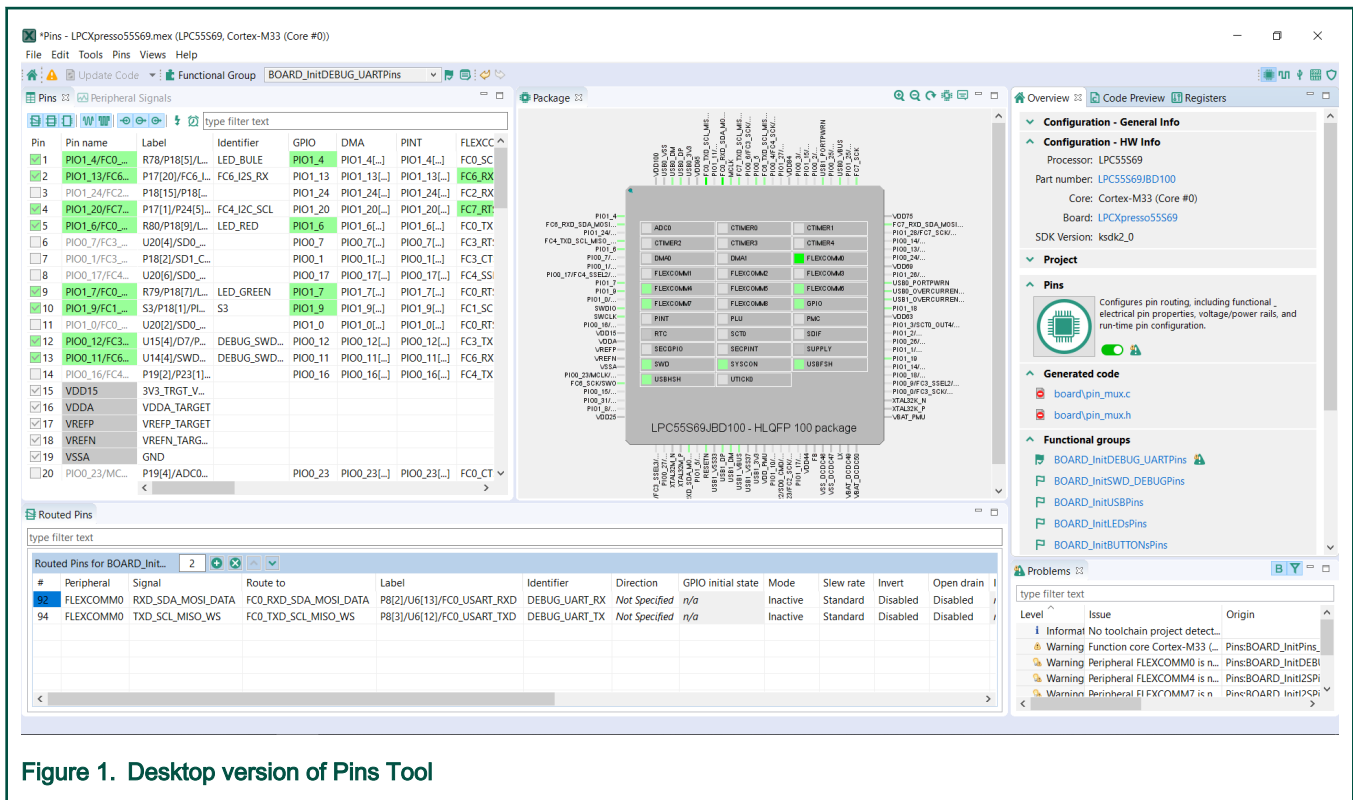


Figure 1. Desktop version of Pins Tool

# Chapter 2

## User Interface

### 2.1 Creating, saving, and opening a configuration

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

It's important there is exactly one project file in the directory where the MEX is located to be able parse the toolchain project (either IAR EW project file EWP, or MDK uVision UVPROJX, or ARM GCC CMakeLists.txt).

#### 2.1.1 Creating a new configuration

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

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

#### NOTE

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

#### 2.1.2 Saving a configuration

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

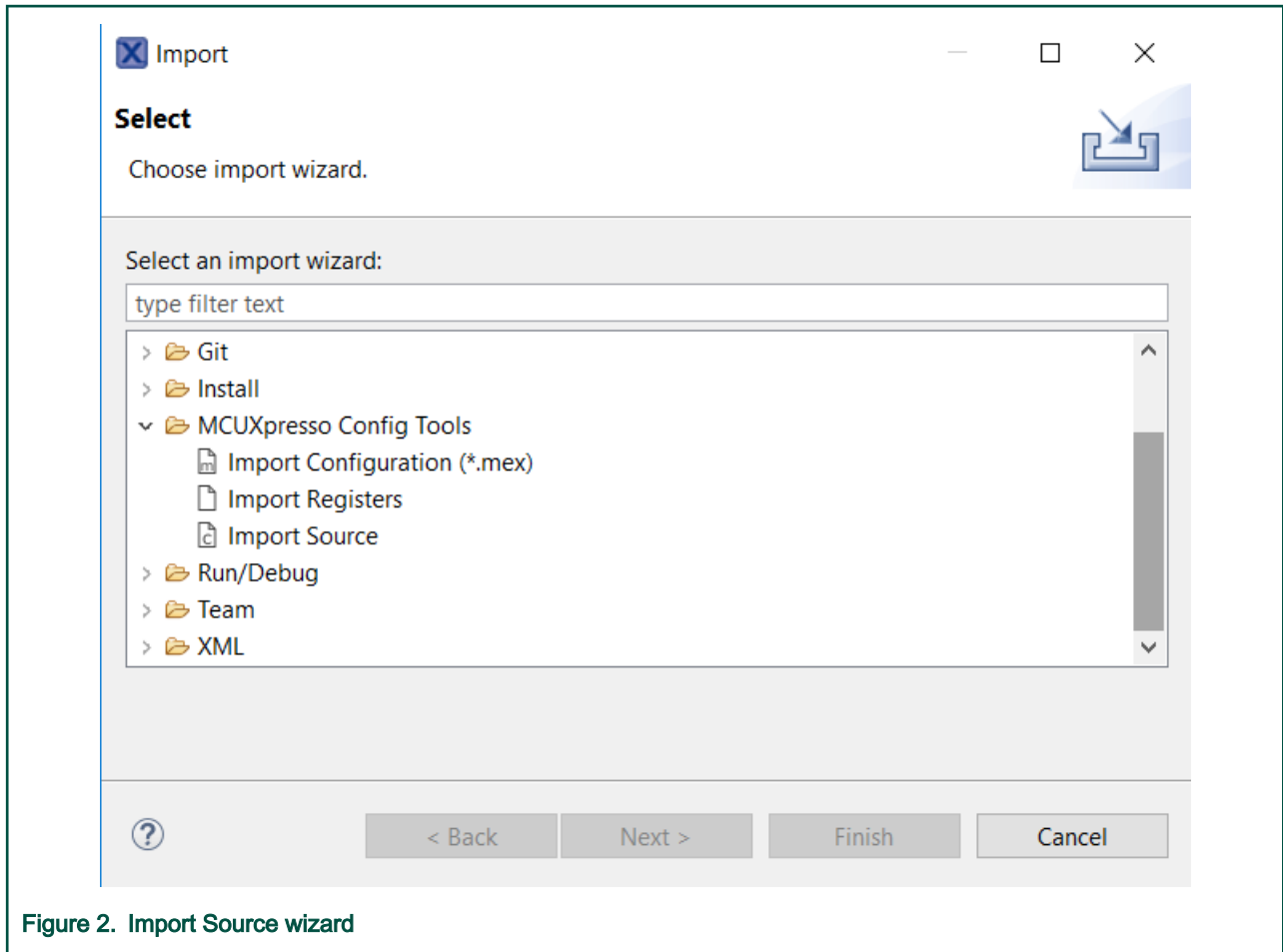
#### NOTE

Configuration is also saved when you select **Update Code** in the toolbar.

#### 2.1.3 Importing sources

To import source code files:

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



**Figure 2. Import Source wizard**

3. Click **Next**.
4. You can select one or more C files to import using the **Browse** button in the **Import Pins Source Files** window.
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 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 an existing one, then the existing one is replaced with the imported one.
6. Click **Finish**.

#### NOTE

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

### 2.1.3.1 Importing configuration

### 2.1.3.2 Importing registers

You can import register configuration from a processor memory dump.

#### NOTE

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



**NOTE**

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

**Figure 3. Import Registers**

**Import Registers**  
Import registers from Memory Dump or CSV format

Registers configuration

Select tool

Functional group options

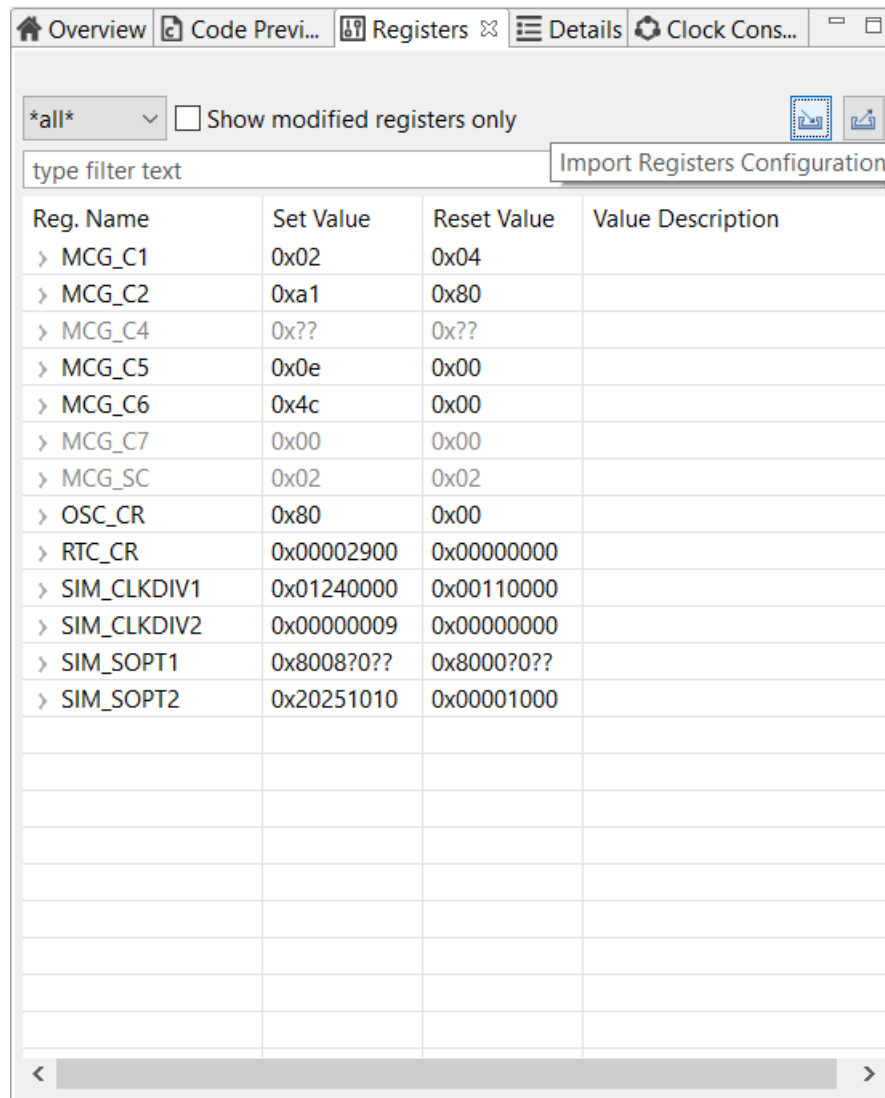
☒ Modify existing functional group  
☐ Create new functional group

Functional group name

To import register configuration, do the following:

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

### Figure 4. Import Registers Configuration



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

## NOTE

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

### 2.1.4 Restoring configuration from source code

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

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

```

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

```

**Figure 5. Setting Information in the source code**

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

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

## 2.2 Toolbar

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

### 2.2.1 Eclipse project selection

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

### 2.2.2 Config Tools Overview

Click the **Config Tools Overview** button to open **Config Tools Overview** and inspect information about the configuration, hardware, and project. See [Config Tools Overview for more information](#).

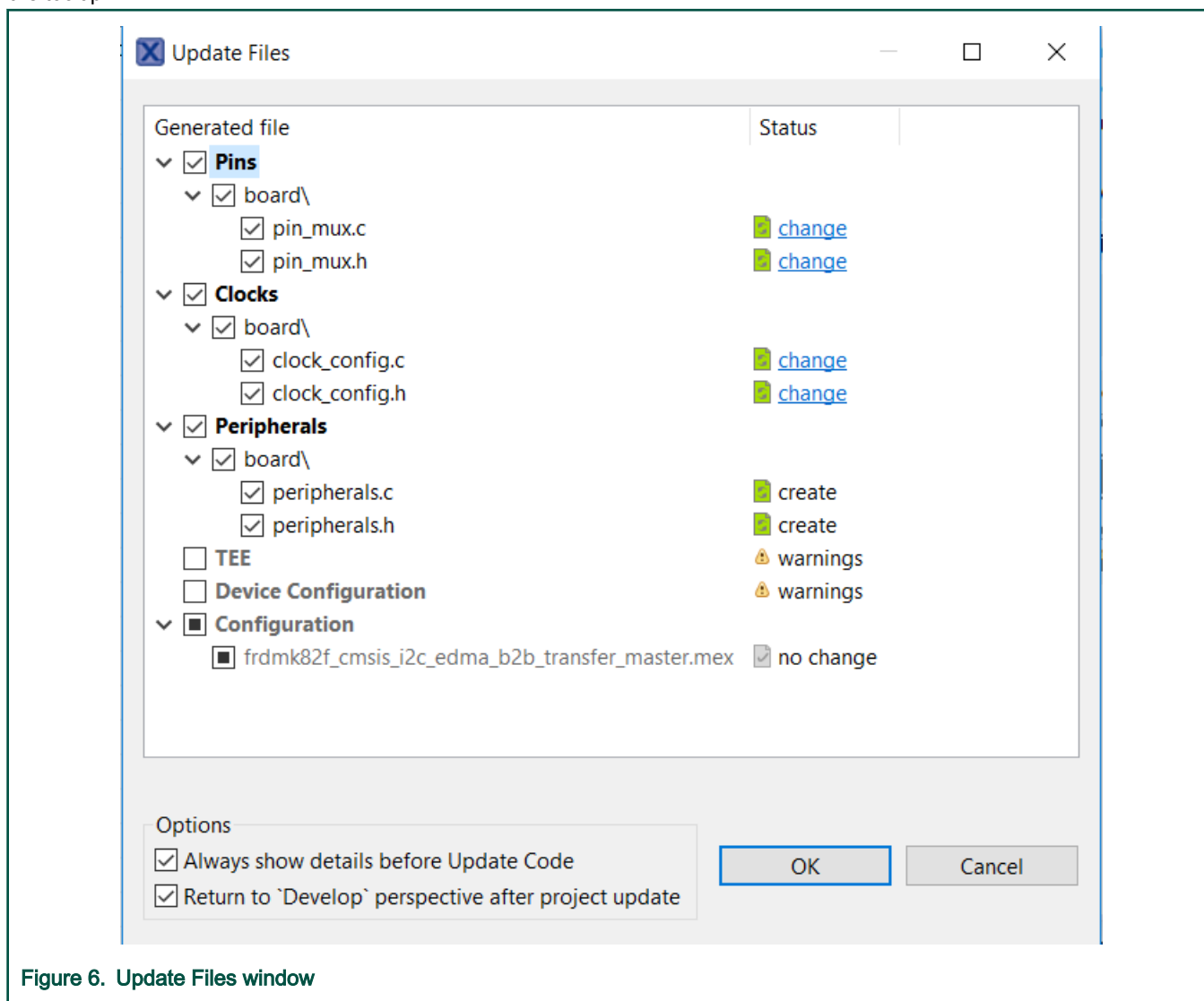
### 2.2.3 Show Problems View

Click the **Show Problems View** to open/highlight the **Problems view** and inspect any errors in your configuration. See [Problems view](#) for more information.

Button color depends on issue type. Red indicates the presence of at least one error, yellow indicates the presence of at least one warning.

## 2.2.4 Update code

To update the generated code in the related toolchain project, click the **Update Code** button. In the window, select the tools or files you want to update. If the file is updated automatically, the button is filled with a black square. The reason is displayed in the tooltip.



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

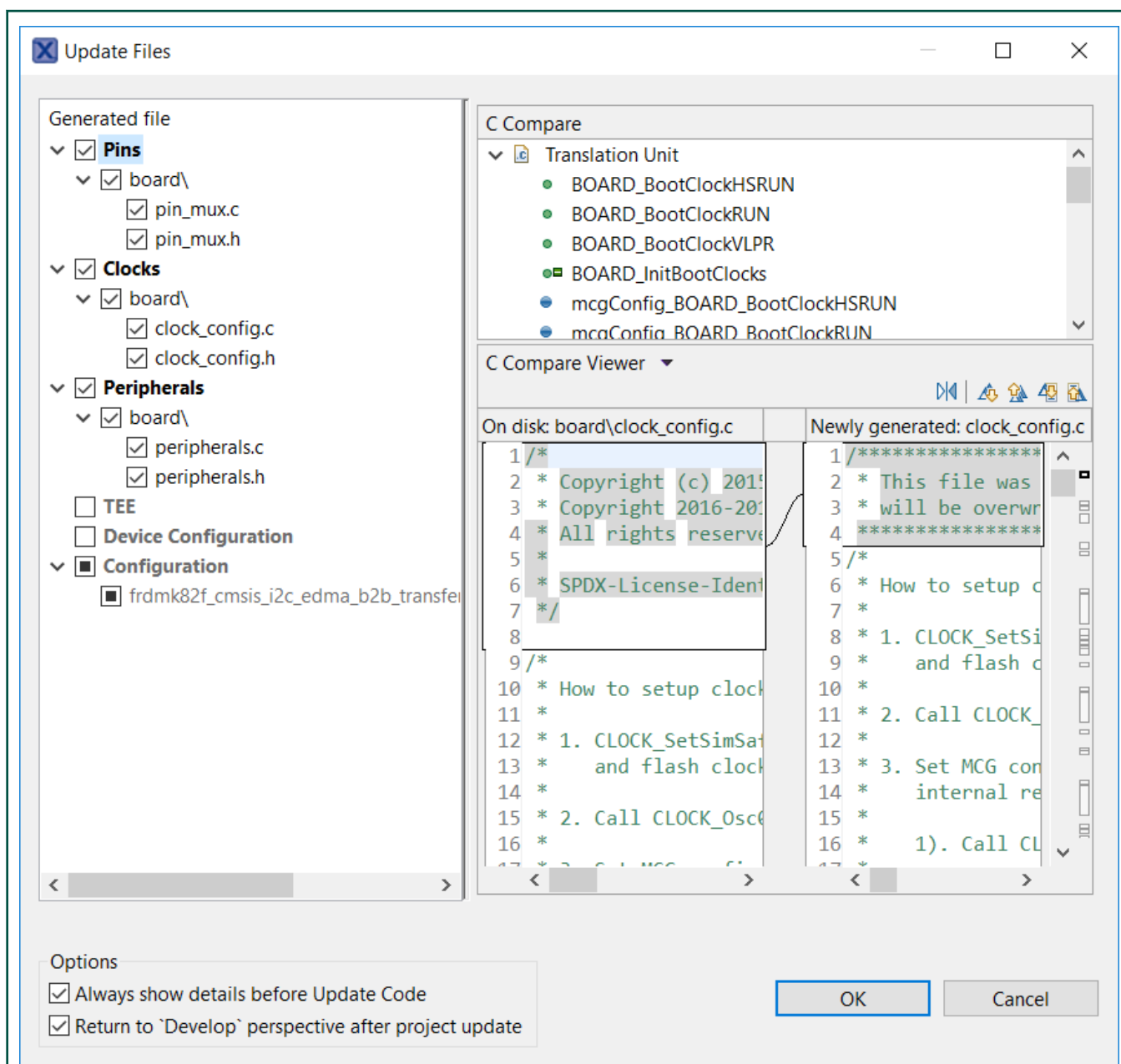


Figure 7. Show differences

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

To access the **Update Code** window from the **Update Code** dropdown menu, select **Open Update Code Dialog**.

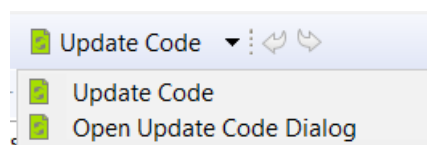


Figure 8. Update Code dropdown menu

NOTE

The generated code is always overwritten.

NOTE

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

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

- If the MEX configuration is saved in a toolchain project, the processor selected in the tool matches with processor selected in the toolchain project
- Core is selected (for multicore processors)

2.2.5 Functional groups

Every **Pins/Clocks/Peripherals** configuration can contain several functional groups.

These groups represent functions which will be generated into source code. Use the dropdown menu to switch between functional groups and configure them.

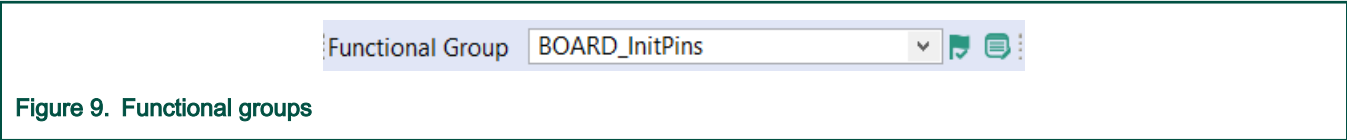




Figure 9. Functional groups

You can use two additional buttons to further configure functional groups:

Table 2. Functional Groups

Icon	Description
	Toggle "Called from default initialization function" feature (in source code)
	Opens the <b>Functional group properties</b> window

NOTE

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

2.2.5.1 Functional group properties

In the **Functional Group Properties** window, you can configure several options for functions and code generation. Each settings is applicable for the selected function. you can specify generated function name, select core (for multicore processors only) that is affecting the generated source code, or write function description (this description will be generated in the C file). You can also add, copy, and remove functional groups as needed.

Aside from name and description, you can choose to set the following parameters for selected functional groups:

- **Set custom #define prefix** - Enable to use the specified prefix for the identifiers in the source code. You can also modify the functions order (on the left), the order is applied in the generated code.

NOTE

Not all processors support this option.

- **Called from default initialization function** - Enable to call the function is called from the default initialization function.
- **Clock gate enable**

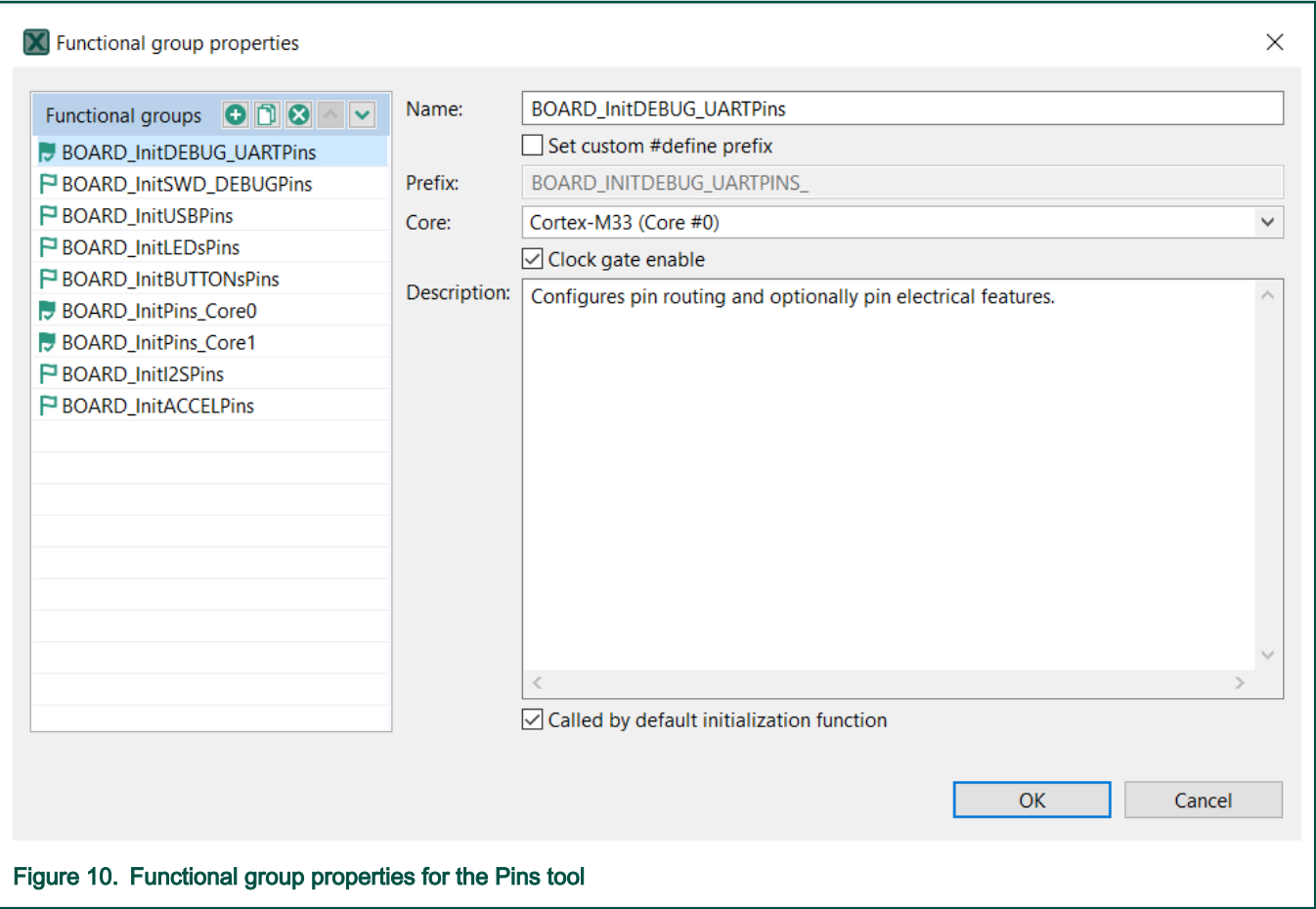


Figure 10. Functional group properties for the Pins tool

2.2.6 Undo/Redo actions

You can reverse your actions by clicking the following buttons:

Table 3. Undo/reto options

Icon	Description
	Cancels the previous action
	Cancels the previous undo action

2.2.7 Selecting the tools

Buttons on the extreme right-hand side of the toolbar represent available tools. Click the icons to quickly navigate between .

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>MCUXpresso Config Tools** from the **Main Menu**. The **Preferences** window will open.

### NOTE

You can restore settings to default by selecting **Restore Defaults** in the lower right corner of the window.

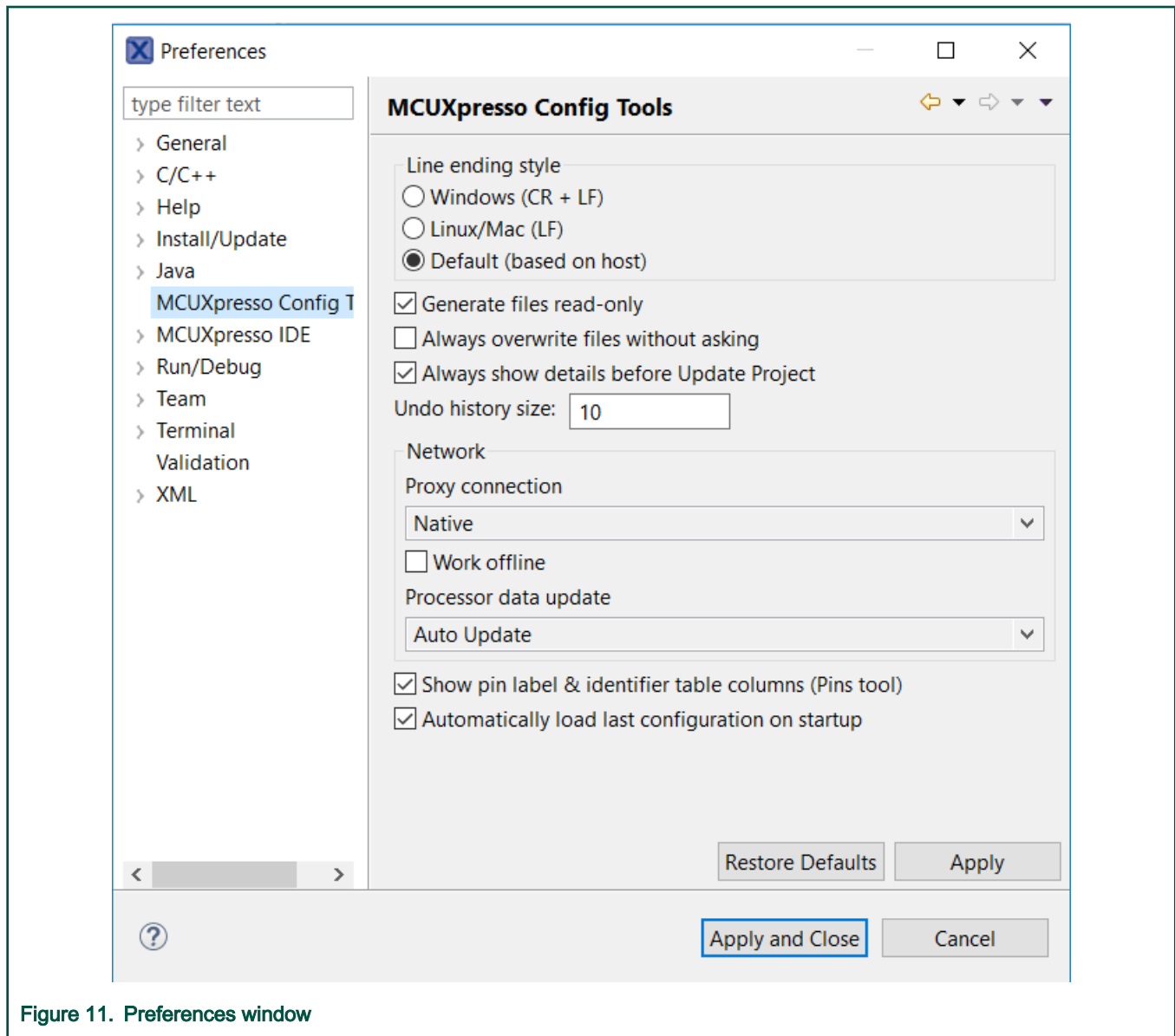


Figure 11. Preferences window

In this window you can set the following:

- **Line ending style** – Select between **Windows (CR + LF)**, **Linux/Mac (LF)**, or **Default (based on host)**.
- **Generate files read-only** – Prevents modifying the source files unintentionally. Generated source files are marked as read-only.
- **Generate source folder** - At build time, automatically creates a folder including source files.
- **Create empty configuration if no yaml is available** - Will generate a configuration even if no yaml is present.
- **Always overwrite files without asking** – Select to update existing files automatically, without prompting.



- **Always show details before Update Code** – 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.

---

**NOTE**

The proxy settings are copied from operating system settings. In case of error, you can specify proxy information in the tools.ini file, located in the <install\_dir>/bin/ folder. Make sure the file contains the following lines:

- Djava.net.useSystemProxies=true (already present by default)
- Dhttp.proxyHost=<somecompany.proxy.net>
- Dhttp.proxyPort=80

---

**NOTE**

Authentication is not supported.

---

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

## 2.5 Configuration preferences

In the **Configuration preferences** window, you can set your preferences for to 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.

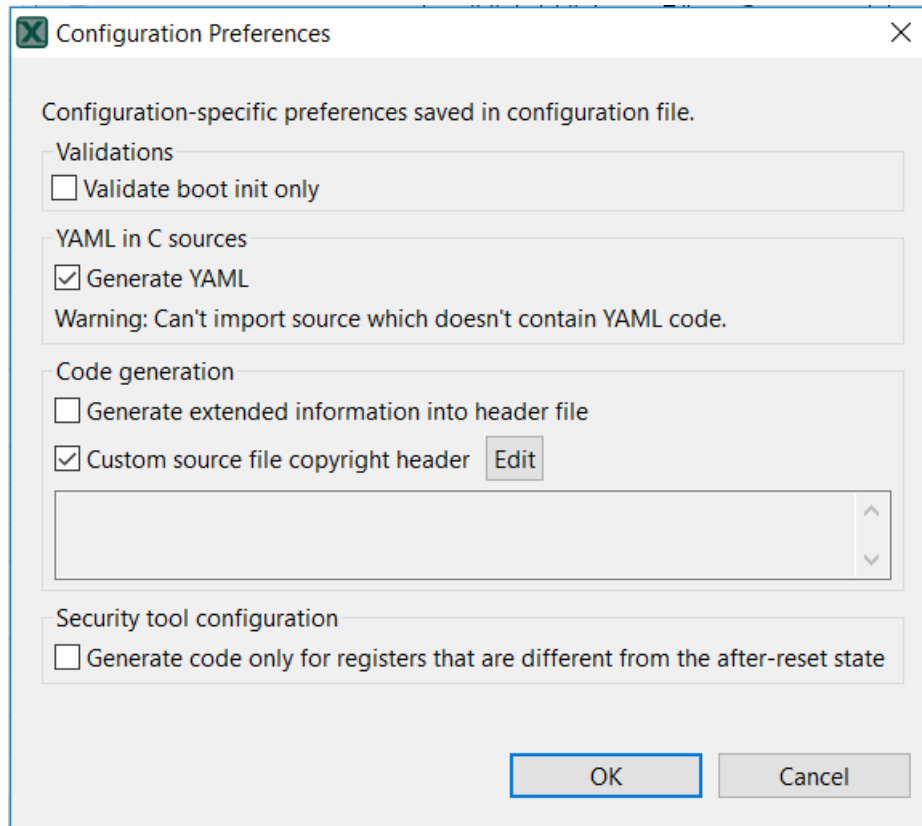


Figure 12. Configuration Preferences

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.
- **Custom source file copyright header** - Select to add a custom copyright header to generated source files that don't already contain copyright.
- **Generate code only for registers that are different from the after-reset state** – Select to generate code only for registers that are different from the after-reset state. For projects created in earlier MCUXpresso versions, this option is selected by default.

#### WARNING

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

## 2.6 Problems view

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

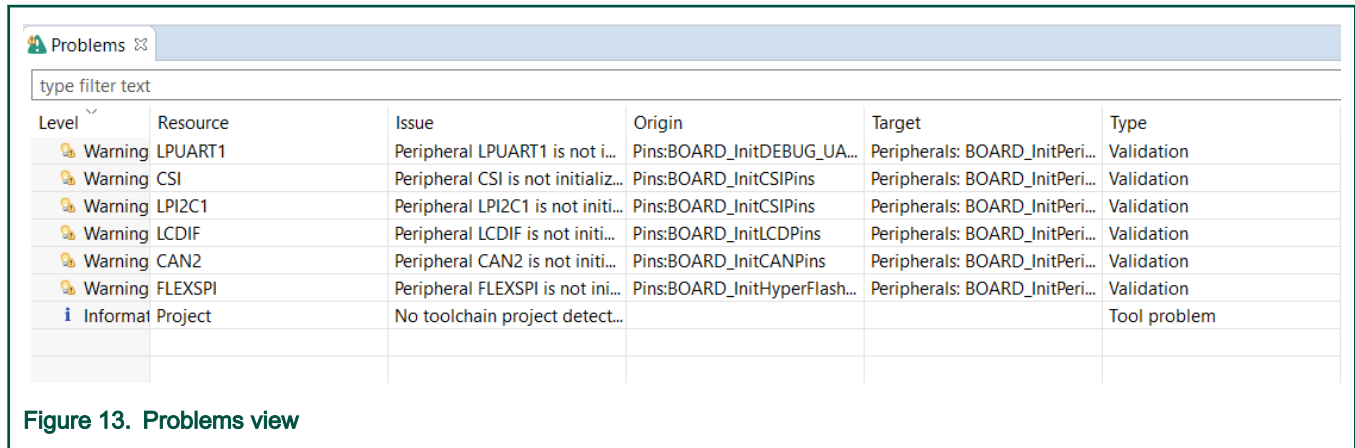


Figure 13. Problems view

To open the **Problems** view, click the **Show Problems view** button, or select **Views > Problems**.

The table contains the following information:

- **Level** – Severity of the problem: Information, Warning, or Error.
- **Resource** – Resource related to the problem, such as signal name, the clock signal, and so on.
- **Issue** – Description of the problem.
- **Origin** – Information on the dependency source.
- **Target** – Tool that handles the dependency and its resolution.
- **Type** – Type of the problem. It's either the validation checking dependencies between tools, or a single tool issue.



Every issue comes with a context menu accessible by right-clicking the table row. Use this menu to access information about the problem or to apply a quick fix where applicable. You can also copy the rows for later use by right-clicking the row and selecting **Copy** or by using the **Ctrl+C** shortcut. You can use the **Ctrl+left-click** shortcut to add additional rows to the selection.

#### NOTE

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

#### Filter buttons

The filter buttons are available on the right side of the **Problems** view ribbon.

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

## 2.7 Registers view

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

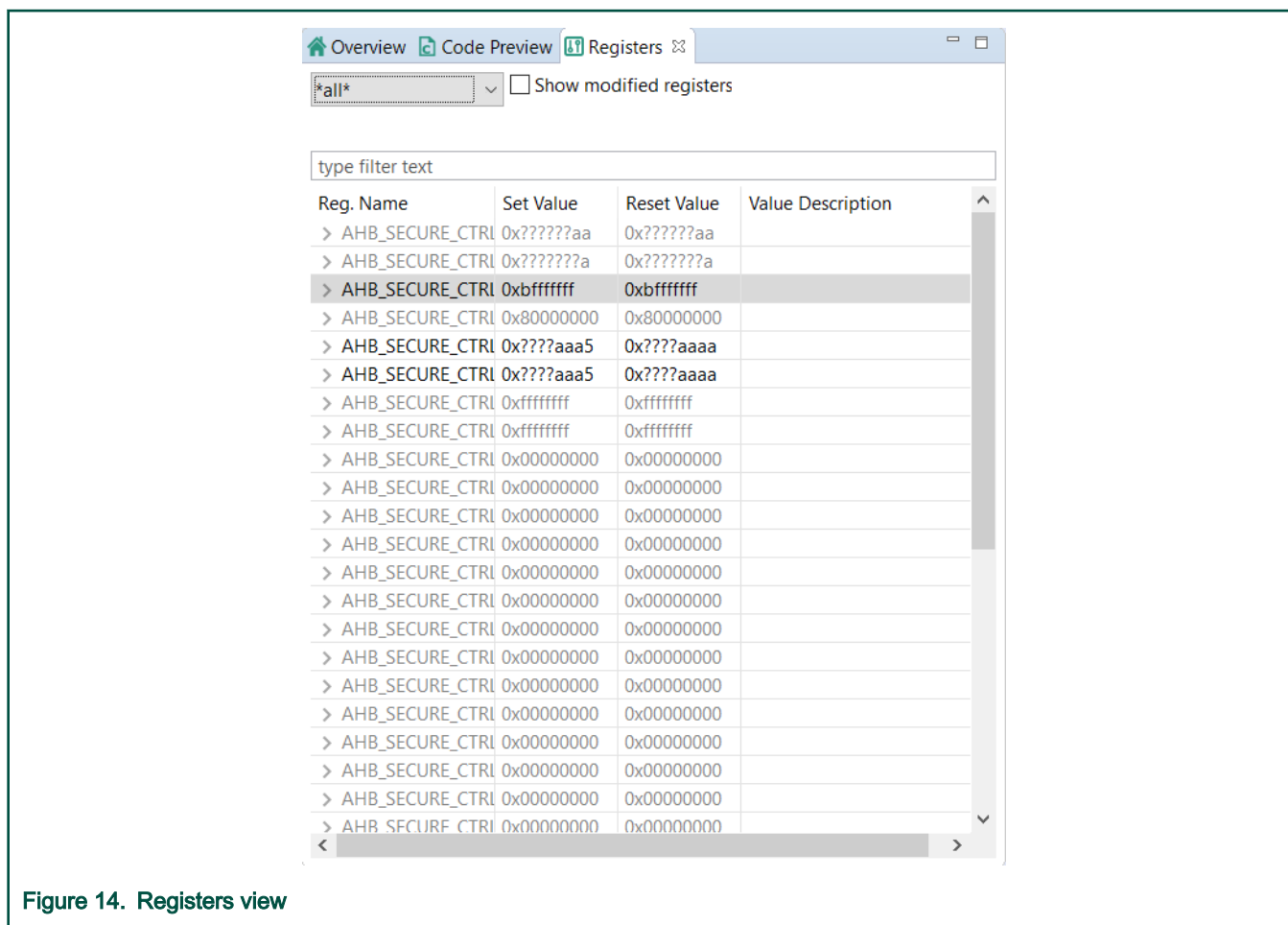


Figure 14. Registers view

The **Registers view** contains:

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

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

Table 4. 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 selected tool. The view uses registers as internal parameters but it might not handle all the register writes needed in the code. The register writes are done inside the SDK functions that are called by the generated code. There might be additional registers accessed in the SDK code during the setup process, and such register writes are not known to the tool and are not displayed in the registers view.

## 2.8 Log view

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

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

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

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

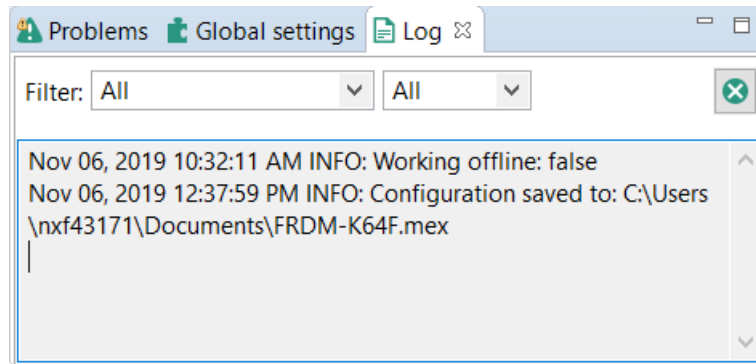


Figure 15. Log view

## 2.9 Config tools overview

**Config Tools Overview** provides you with general information about your currently active configuration, hardware, and project. It also provides a quick overview of the used/active and unused/inactive tools, generated code, and functional groups. By default, the **Config Tools Overview** icon is located on the left side of the toolbar.

**Config Tools Overview** contains the following options:

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

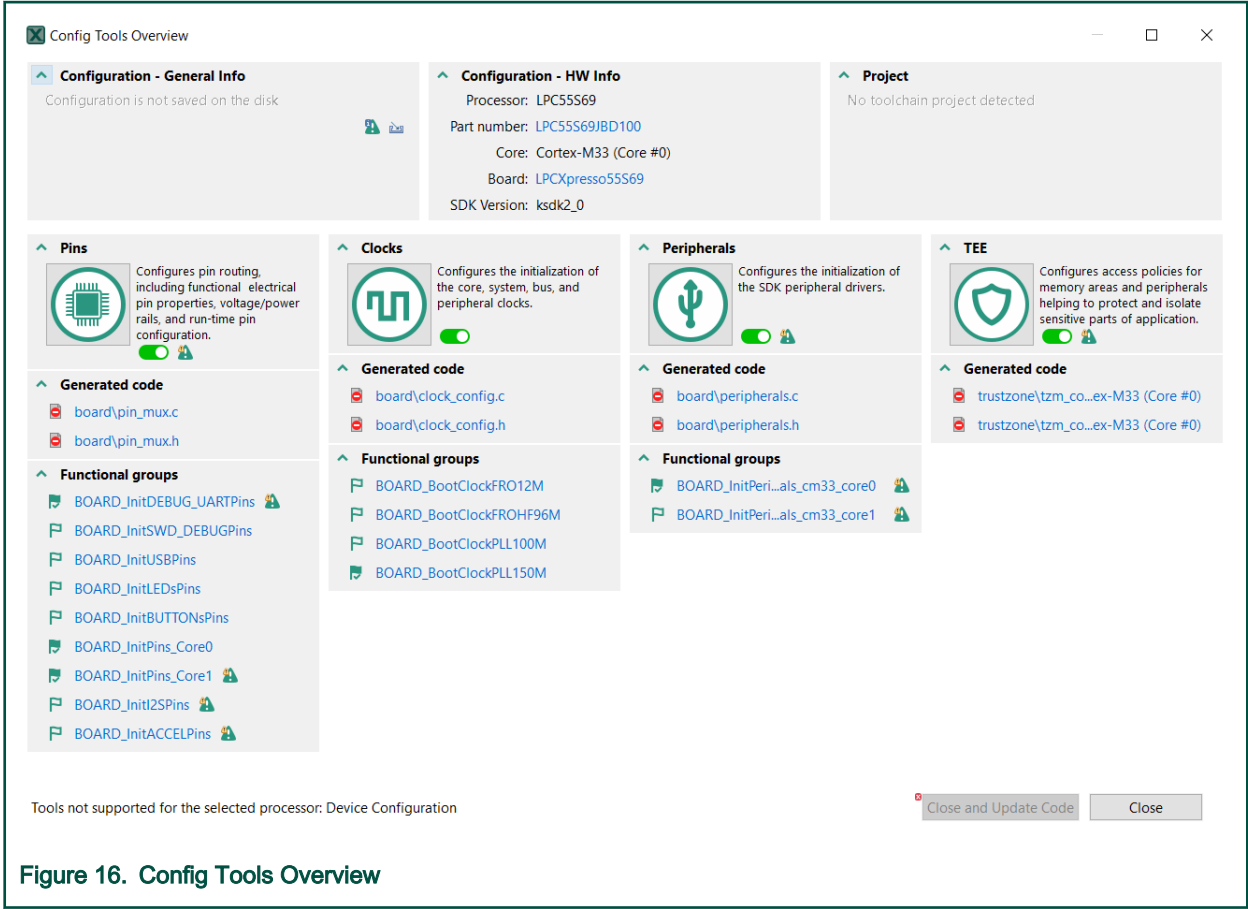


Figure 16. Config Tools Overview

NOTE

Unsupported tools are not displayed in the overview.

# Chapter 3

## Pins Tool

**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.

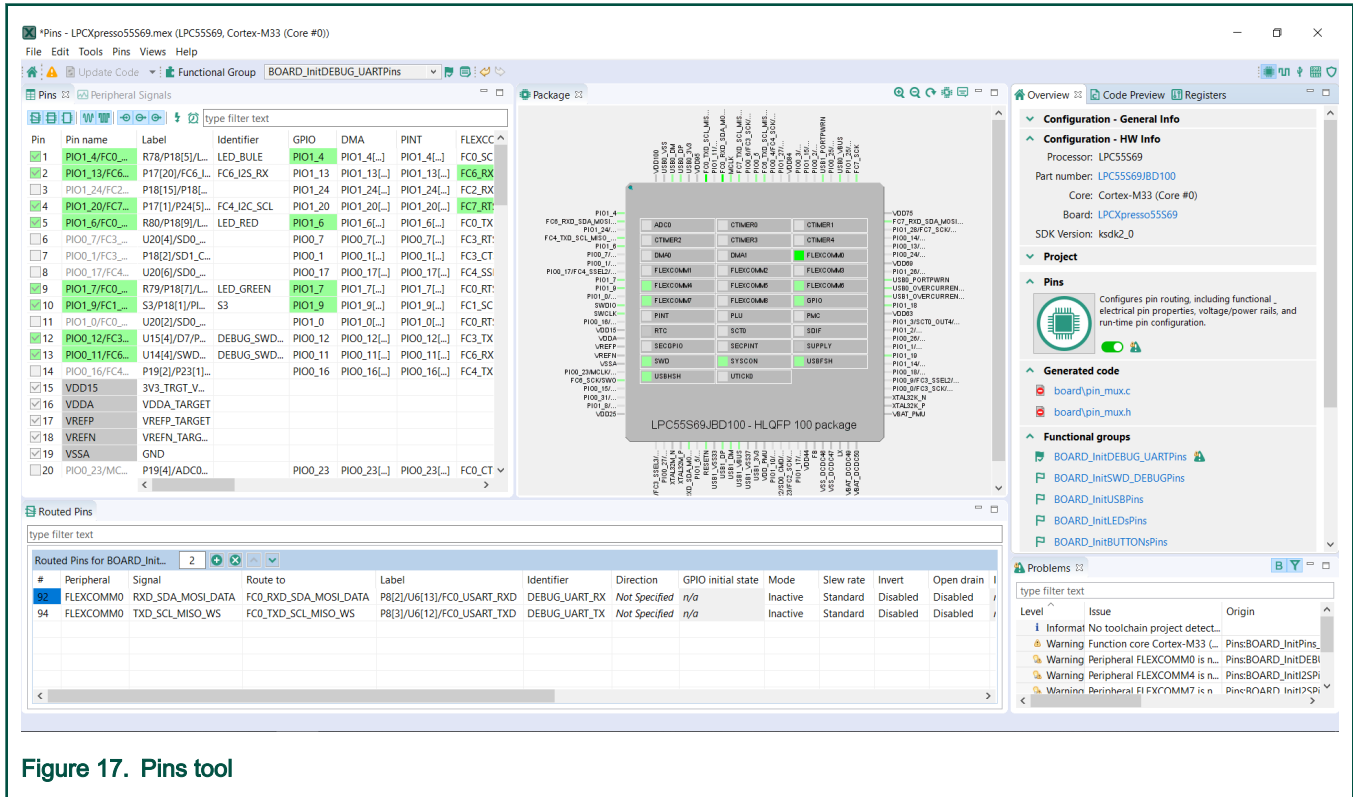


Figure 17. Pins tool

### 3.1 Pins routing principle

**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.

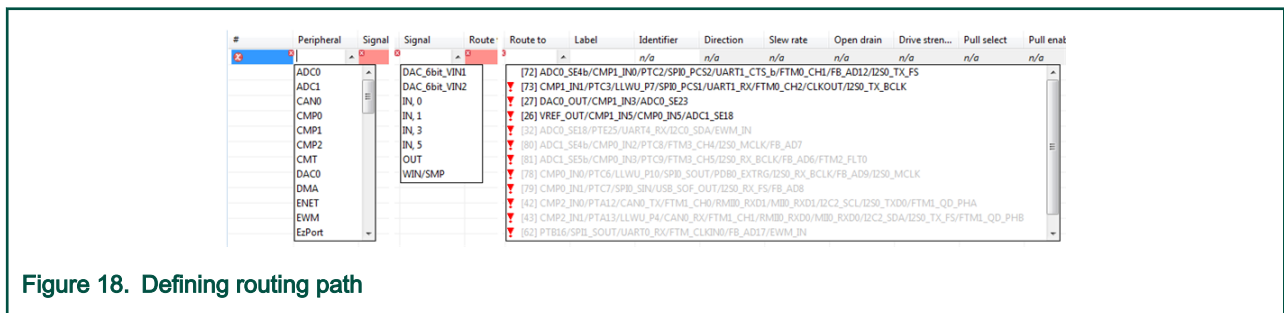


Figure 18. Defining routing path

### 3.1.2 Beginning with pin/internal signal selection

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

- Select the pin/internal signal (**Route to**).
- 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.
- For the selected peripheral, select one of the available **Signals**.

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

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

#### NOTE

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

## 3.2 Workflow

The following steps briefly describe the basic workflow in the **Pins** tool.

- 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 views **Peripheral Signals**, **Package**, **Routed Pins**.

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



**NOTE**

Source code is automatically generated.

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

### 3.3 Example usage

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

In this example, three pins (**UART3\_RX**, **UART3\_TX** and **PTB20**) on a board are configured.

You can use the generated files with the application code.

1. In the **Pins** view on the left, select the **UART3\_RX** and **TX** signals. For this, you can click into the cells to make them 'green'.

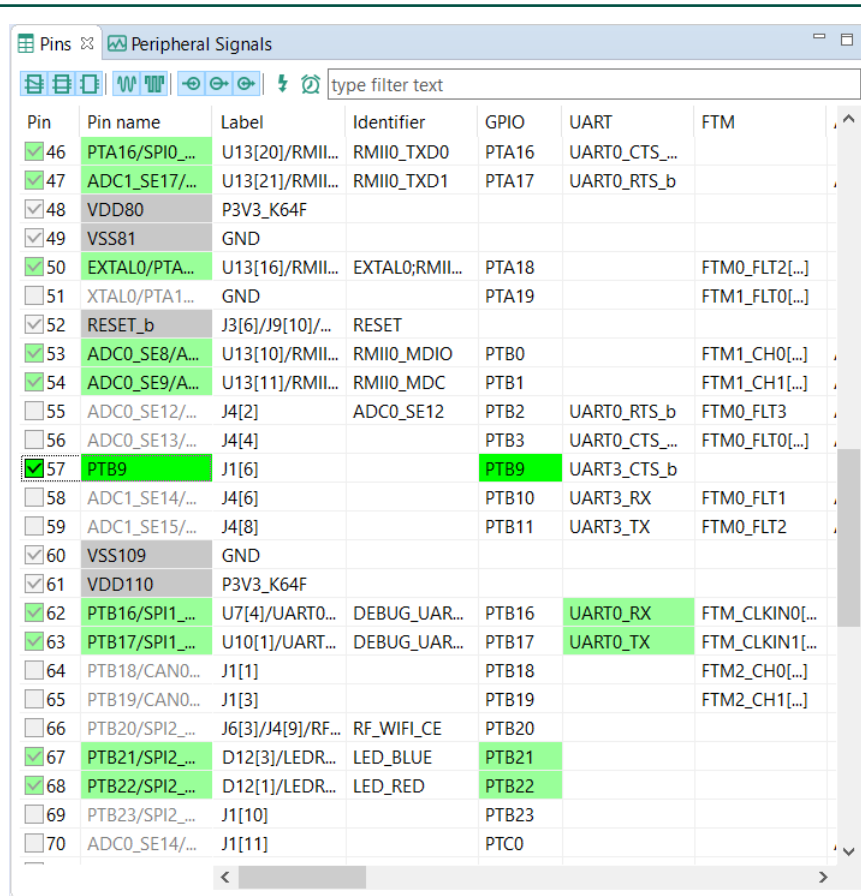


Figure 19. Configure Signals in Pins View

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

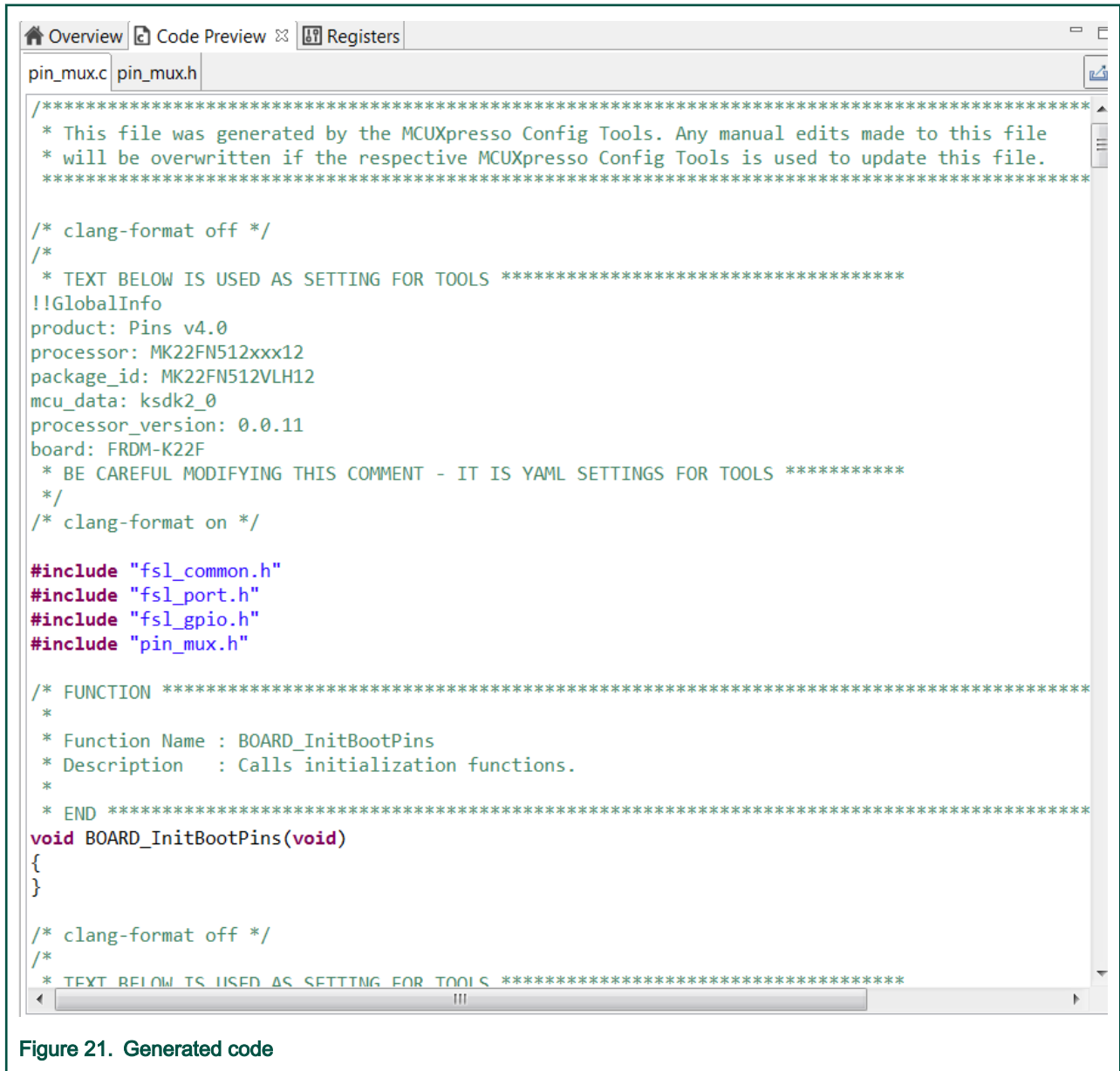
#	Peripheral	Signal	Route to	Label	Identifier	Direction	GPIO initial state	GPIO interrupt	Slew rate	Open drain	Drive strength	Pull select	Pull enable	Passiv
55	FTM3	CH, 6	FTM3_CH6	J1[13]/I2C1_SCL/I2S0_RX_FS	n/a	Not Specified	n/a	n/a	Fast	Disabled	Low	Pulldown	Disabled	Disab
40	GPIOB	GPIO, 17	PTB17	PUSH_BUTTON1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

Figure 20. Select Direction

**NOTE**

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

- 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.



```

Overview Code Preview Registers
pin_mux.c pin_mux.h

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

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

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

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

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

```

**Figure 21. Generated code**

- 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.

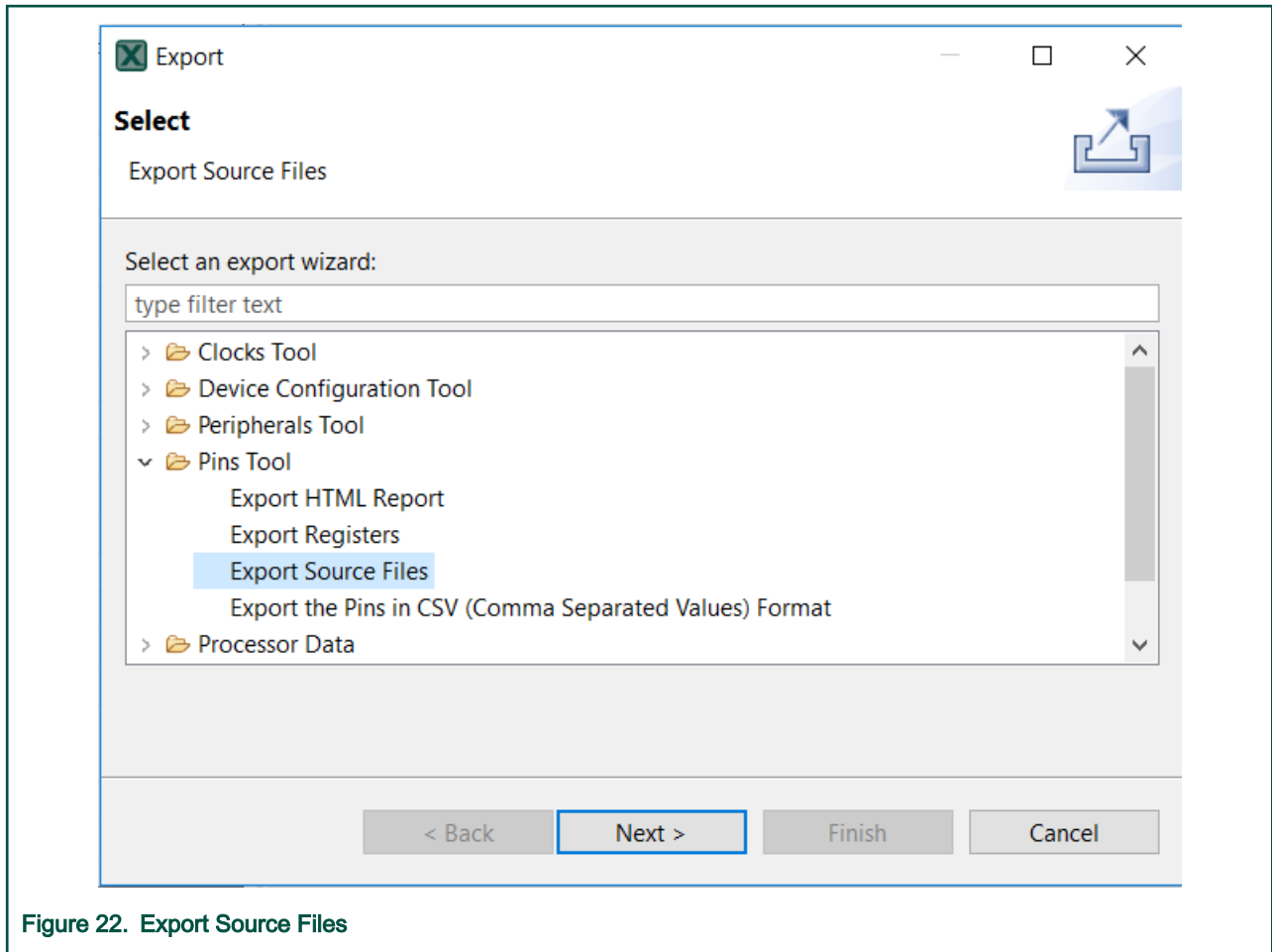


Figure 22. Export Source Files

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

## 3.4 User interface

Pins Tool consists of several views.

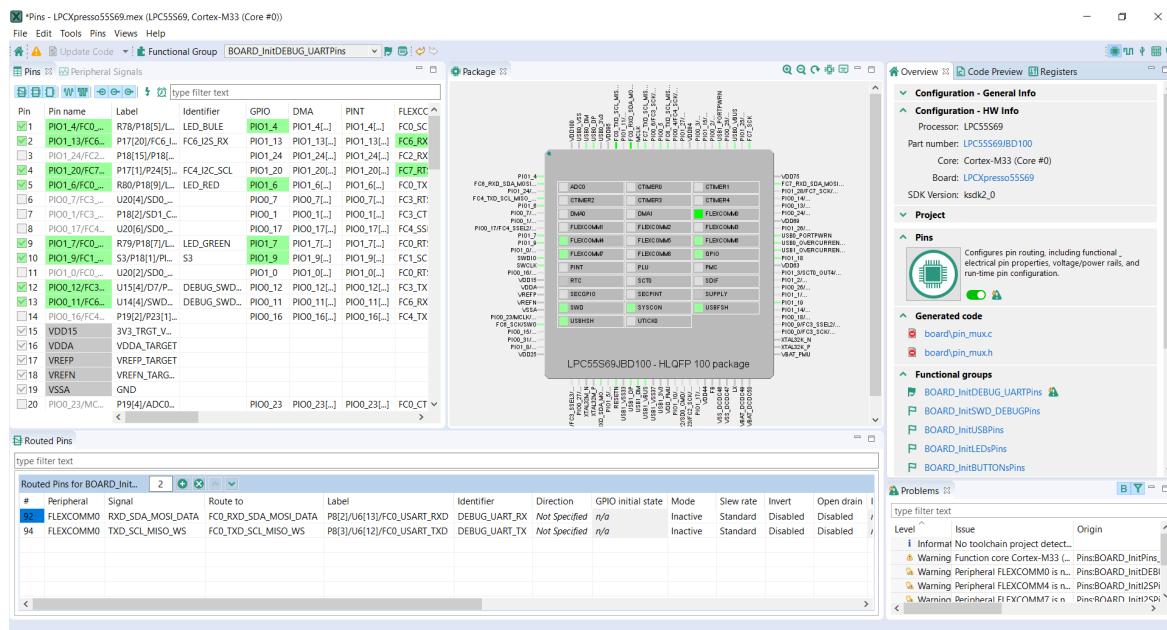


Figure 23. Pins Tool user interface

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

Figure 24. Selecting power group

**NOTE**

Power Groups are not supported for all processors.

### 3.4.1 Functions

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

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

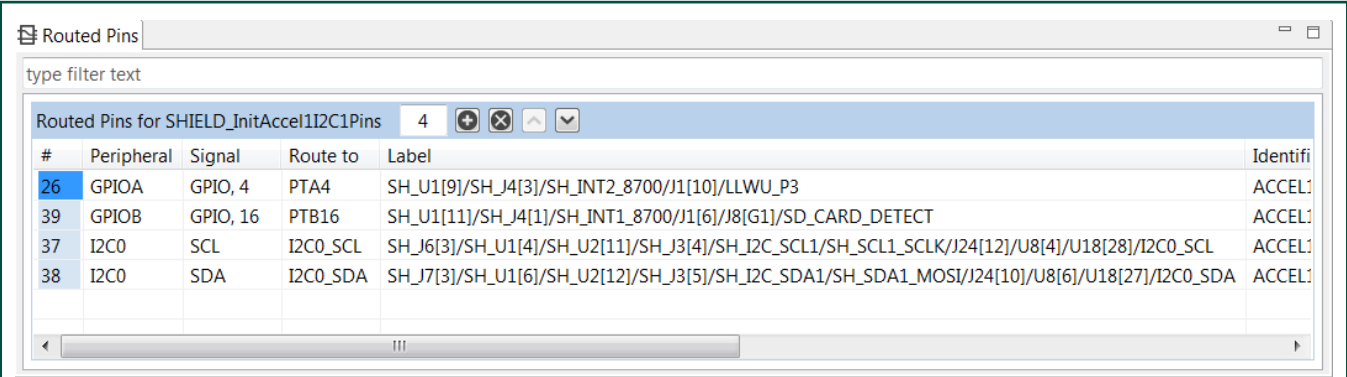


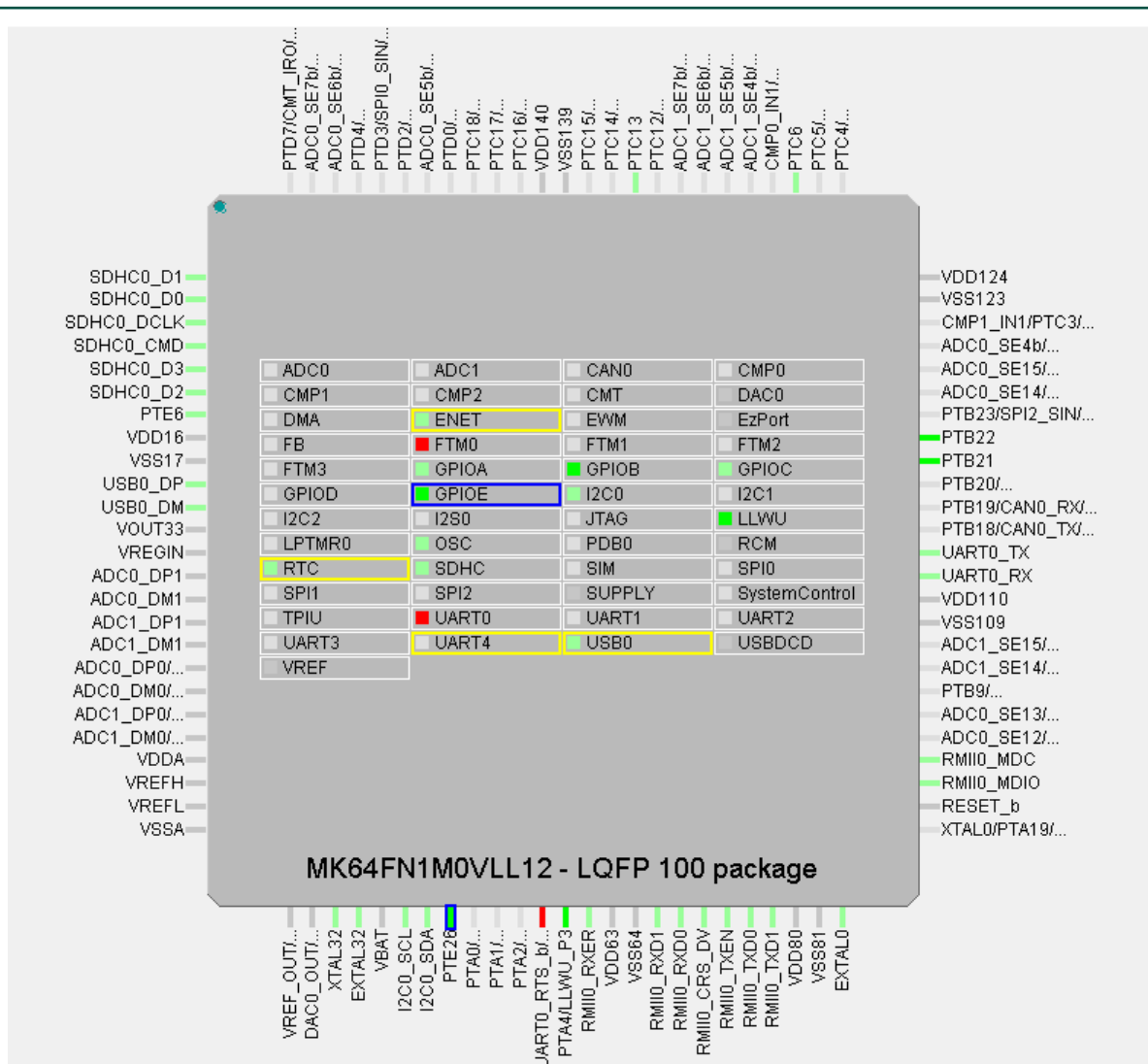
Figure 25. Routed Pins view

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

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

### 3.4.2 Package

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



**Figure 26. Processor package**

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

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

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

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








The following icons are available in the toolbar:

**Table 5. Toolbar options**

Icon	Description
	Zoom in package image.

*Table continues on the next page...*

Table 5. 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 for the Processor** window shows list of available processor packages, showing package type and number of pins.

### 3.4.3 Routed Pins view

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

**NOTE**

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

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

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

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

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

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

Figure 27. Routed Pins view

The gray background indicates the read-only items.

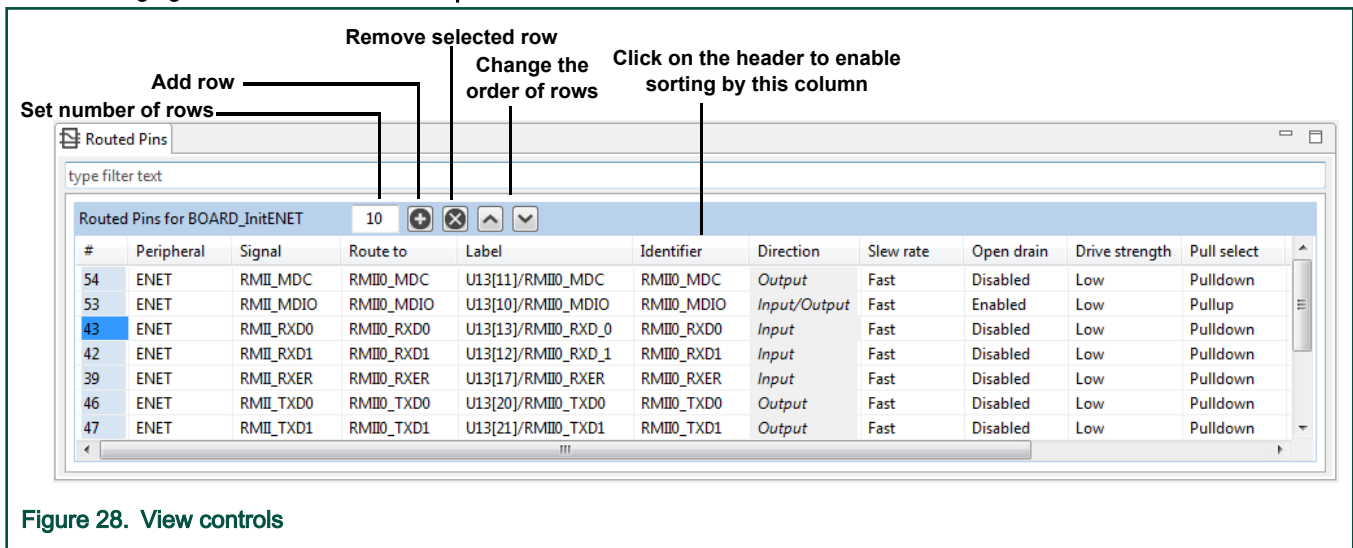
The italic value indicates that the value is not configured and it shows the after-reset value and no code is generated, so the configuration relies on the after reset value or the values configured from the different functions.

#### 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.



#### Add / remove rows:

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

#### Add a specific number of rows or clear the table:

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

#### Change the order of the rows:

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

#### Filter table entries:

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

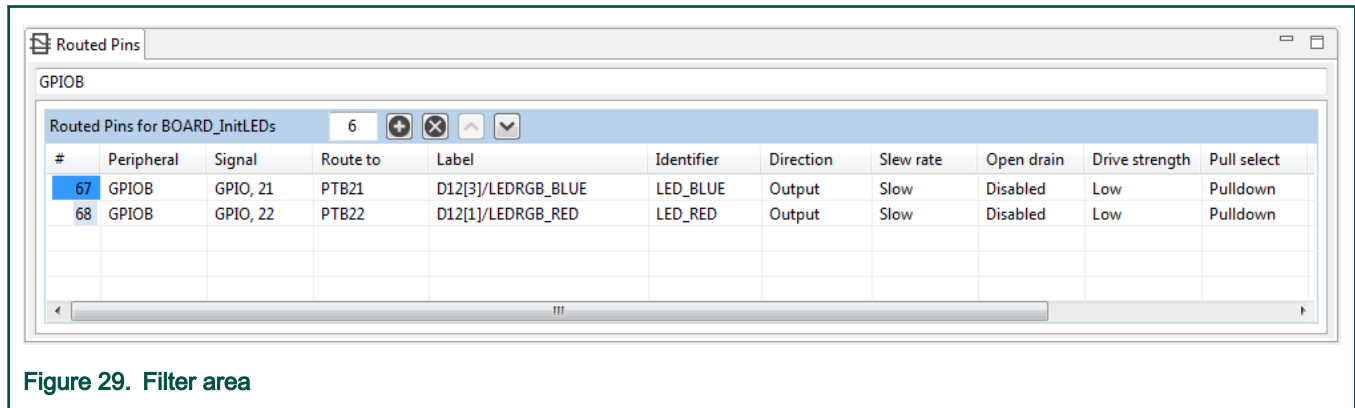
#### Copy-paste rows:

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



### 3.4.3.2 Filtering routed pins

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



**Figure 29. 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 6. Status codes**

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

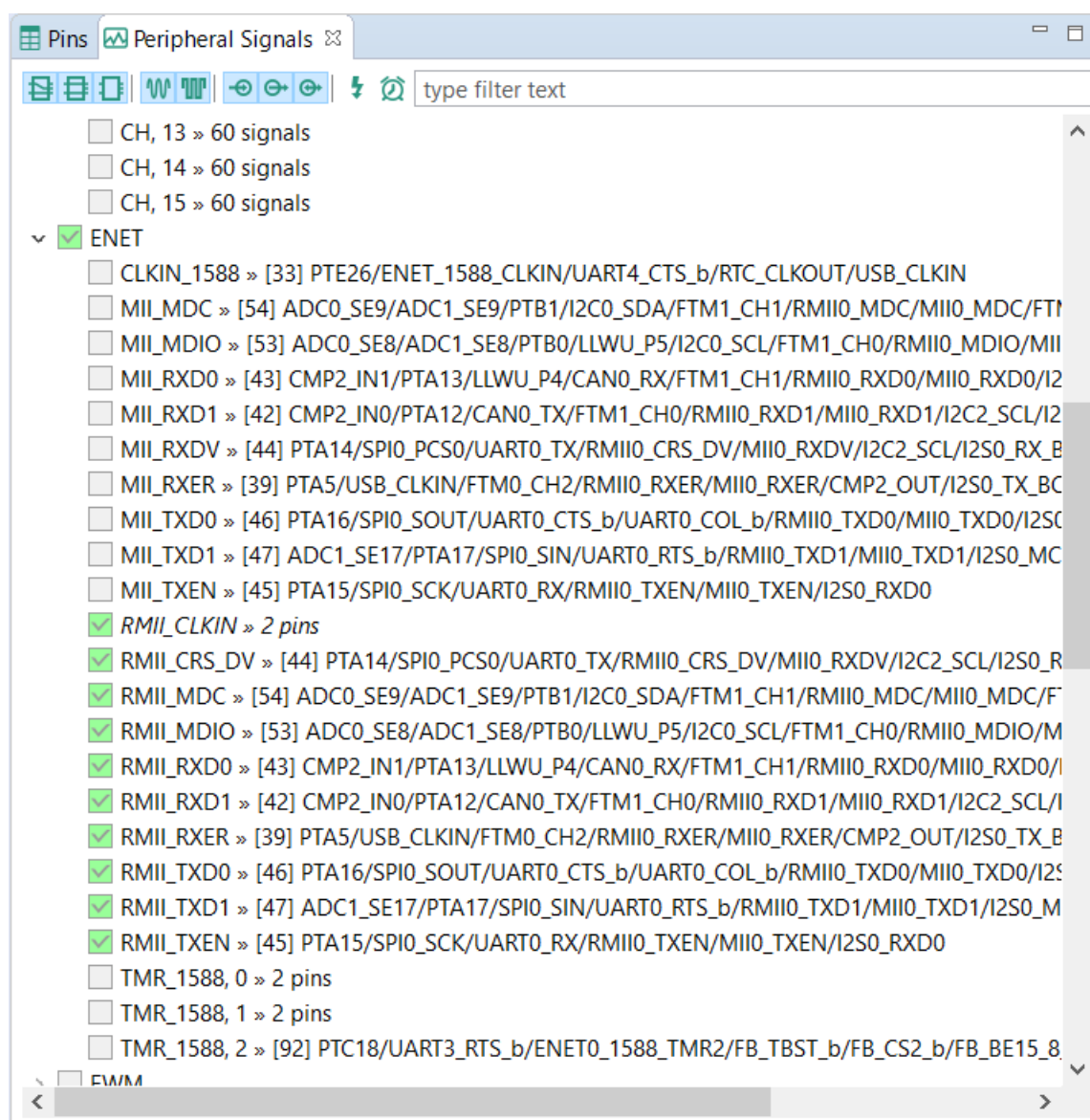


Figure 30. Peripheral Signals view

Use the checkbox to route/unroute the selected pins.

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

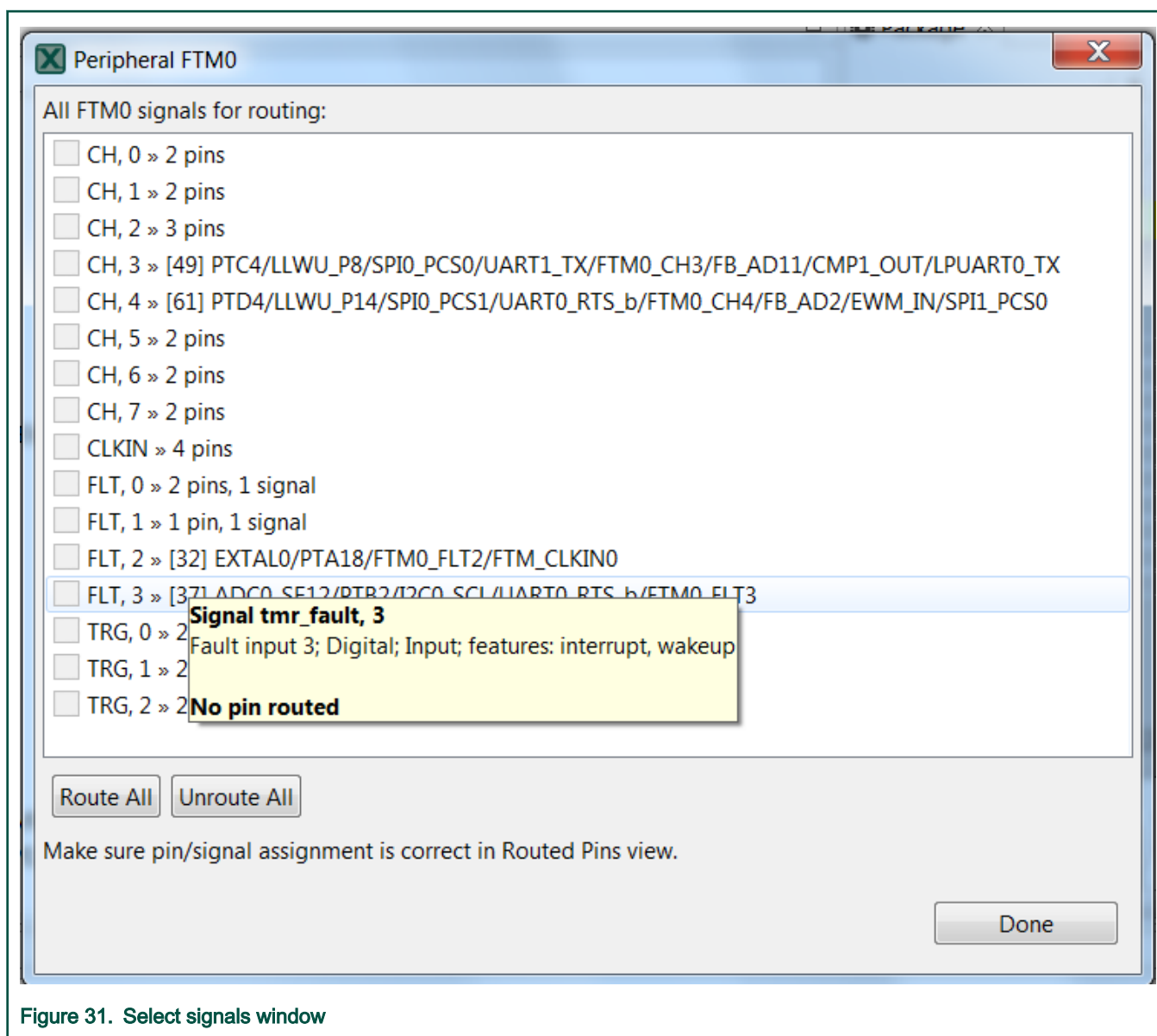


Figure 31. Select signals window

### 3.4.5 Pins view

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

Pin	Pin name	Label	Identifier	GPIO	DMA	PINT	FLEXCC
82	PIO1_15/UTI...	P18[12]/SD1_...		PIO1_15	PIO1_15[...]	PIO1_15[...]	FC5_RT:
83	PIO0_3/FC3_...	U3[11]/FC3_S...		PIO0_3	PIO0_3[...]	PIO0_3[...]	FC3_RX
84	VDD84	3V3_TRGT_V...					
85	PIO1_27/FC2...	P17[16]/PLU...		PIO1_27	PIO1_27[...]	PIO1_27[...]	FC2_RT:
86	PIO0_4/FC4_...	U3[14]/FC3_S...		PIO0_4	PIO0_4[...]	PIO0_4[...]	FC4_SC
87	PIO1_16/FC6...	P18[17]/SD1_...	FC6_I2S_WS	PIO1_16	PIO1_16[...]	PIO1_16[...]	FC6_TX
88	PIO0_5/FC4_...	S1/J10[1]/U3[...	S1	PIO0_5	PIO0_5[...]	PIO0_5[...]	FC4_RX
89	PIO0_6/FC3_...	U3[13]/FC3_S...		PIO0_6	PIO0_6[...]	PIO0_6[...]	FC3_SC
90	PIO0_19/FC4...	P17[12]/FC7_L...	FC7_I2S_WS	PIO0_19	PIO0_19[...]	PIO0_19[...]	FC4_RT:
91	PIO1_31/MC...	P19[7]/P19[8]...	MCLK	PIO1_31	PIO1_31[...]	PIO1_31[...]	
92	PIO0_29/FC0...	P8[2]/U6[13]/...	DEBUG_UAR...	PIO0_29	PIO0_29[...]	PIO0_29[...]	FC0_RX
93	PIO1_11/FC1...	P17[6]/PIO1_...		PIO1_11	PIO1_11[...]	PIO1_11[...]	FC1_TX
94	PIO0_30/FC0...	P8[3]/U6[12]/...	DEBUG_UAR...	PIO0_30	PIO0_30[...]	PIO0_30[...]	FC0_TX
95	VDD95	3V3_TRGT_V...					
96	USB0_3V3	VDD_TARGET...					
97	USB0_DP	P10[3]/D10[3]...	USB0_DP				
98	USB0_DM	P10[2]/D10[2]...	USB0_DM				
99	USB0_VSS	GND					
100	VDD100	3V3_TRGT_V...					

Figure 32. 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/unroute a pin to the given peripheral, select the relevant cell in the **Pin** column. Routed pins are highlighted in green. If a conflict in routing exists, the pins are highlighted in red.

Every routed pin appears in the **Routed pins** table.

When multiple functions are specified in the configuration, the **Pins** 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.

You can double-right-click a row to open a dropdown menu that offers the following options:

- Route/Unroute the pin.
- Highlight the pin in the **Package** view.
- Set the label and identifier for the pin.
- Add a comment to the pin. You can later inspect the comment in the **Code Preview** view.

#### TIP

The option to route more signals to a single pin is indicated by an ellipsis (...). Select the cell to open a dialog to choose from multiple available signals. The dialog also displays which signals are routed by default.

### 3.4.6 Labels and identifiers

You can define the label of any pin that can be displayed in user interface for ease of identification.

Boards and kits have pre-defined labels. However, it's also possible to define a pin label listed in the **Pins** and **Routed Pins** views. To set/update the **Labels and Identifier** columns visibility, select **Edit > Preferences**.

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

Pin	Pin name	Label	Identifier	GPIO
49	VSS81	GND		
50	EXTAL0/PTA18/...	U13[16]/RMII_RXCLK	EXTAL0;RMII_RXCLK	PTA18
51	XTAL0/PTA19/F...	GND		PTA19
52	RFSFT h	I3I6I/I9I10I/D1/RFSFT	RFSFT	

Figure 33. Pin Identifier

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

#	Peripheral	Signal	Route to	Label	Identifier	Directi
50	GPIOA	GPIO, 18	PTA18	U13[16]/RMII_R...	EXTAL0	Input
					EXTAL0	
					RMII_RXCLK	
					Not Specified	

Figure 34. Identifier in Routed Pins table

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

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

Figure 35. 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.

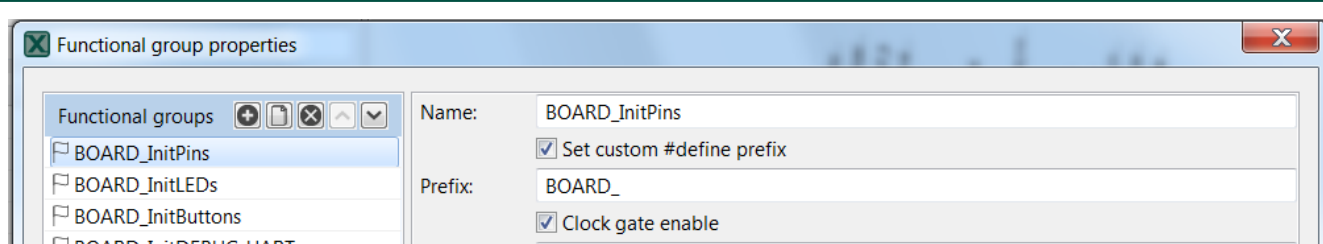
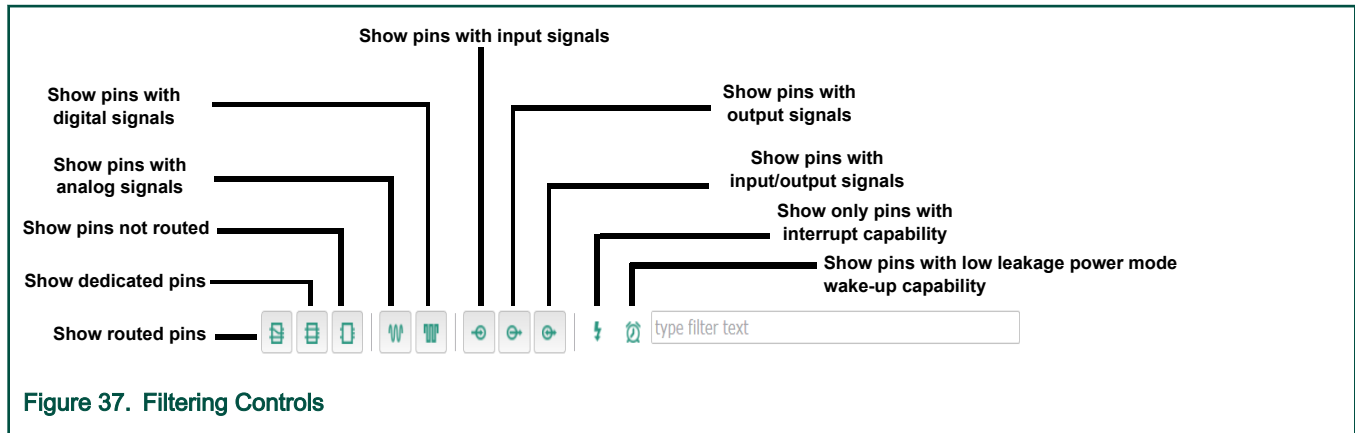


Figure 36. Pins macros prefix

If multiple functions are used, each individual function can include a special prefix. Check the **Pins > Functional Group Properties > Set custom #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.

### 3.4.7 Filtering in the Pins and Peripheral Signals views

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



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

### 3.4.8 Highlighting and color coding

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

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

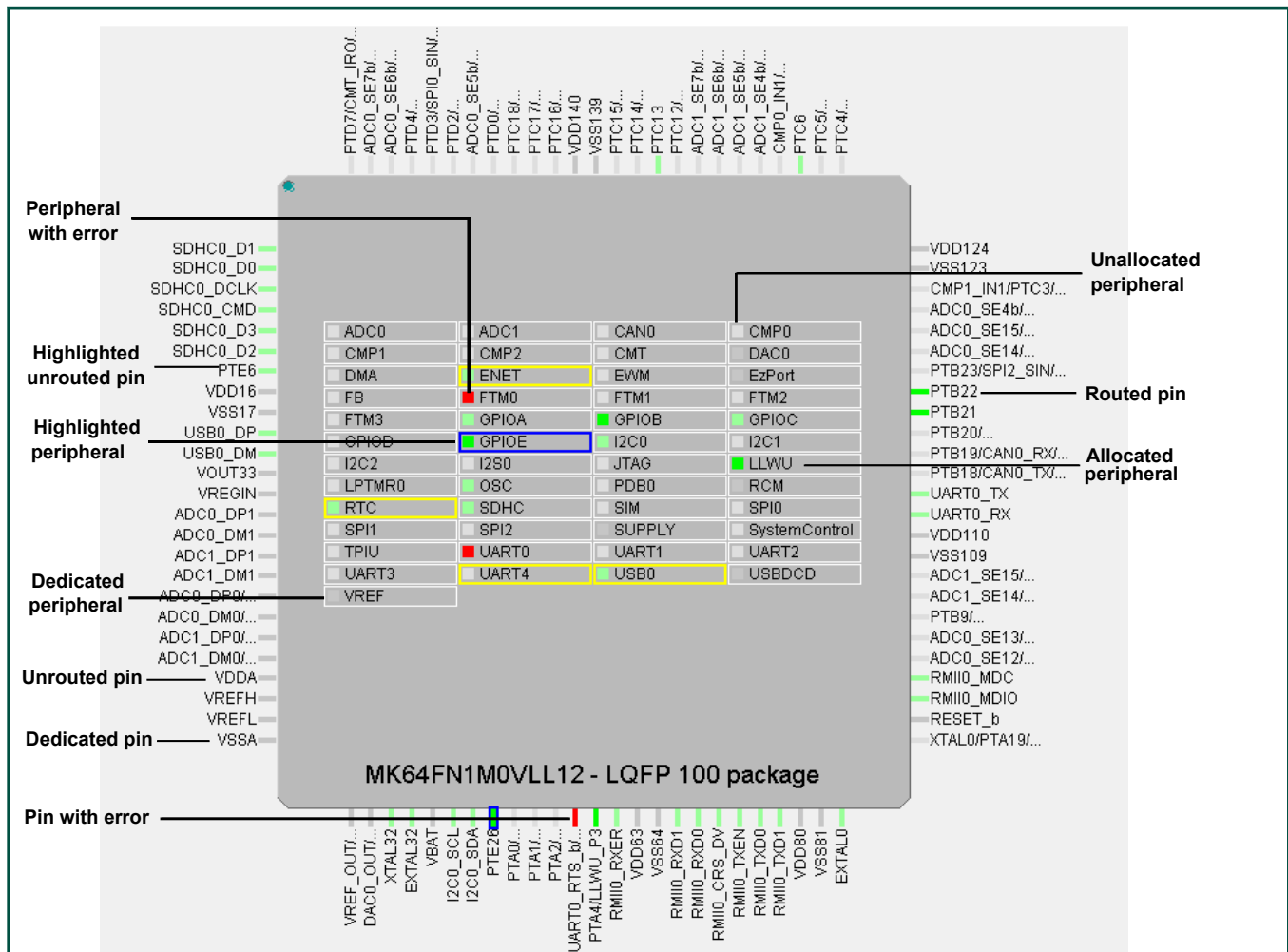


Figure 38. Highlighting and color coding

#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Open drain	Drive strength
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 39. 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 40. 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.

### 3.4.9 Expansion Header

In the **Expansion Header** view, you can add and modify an expansion header configuration, and map the connectors.

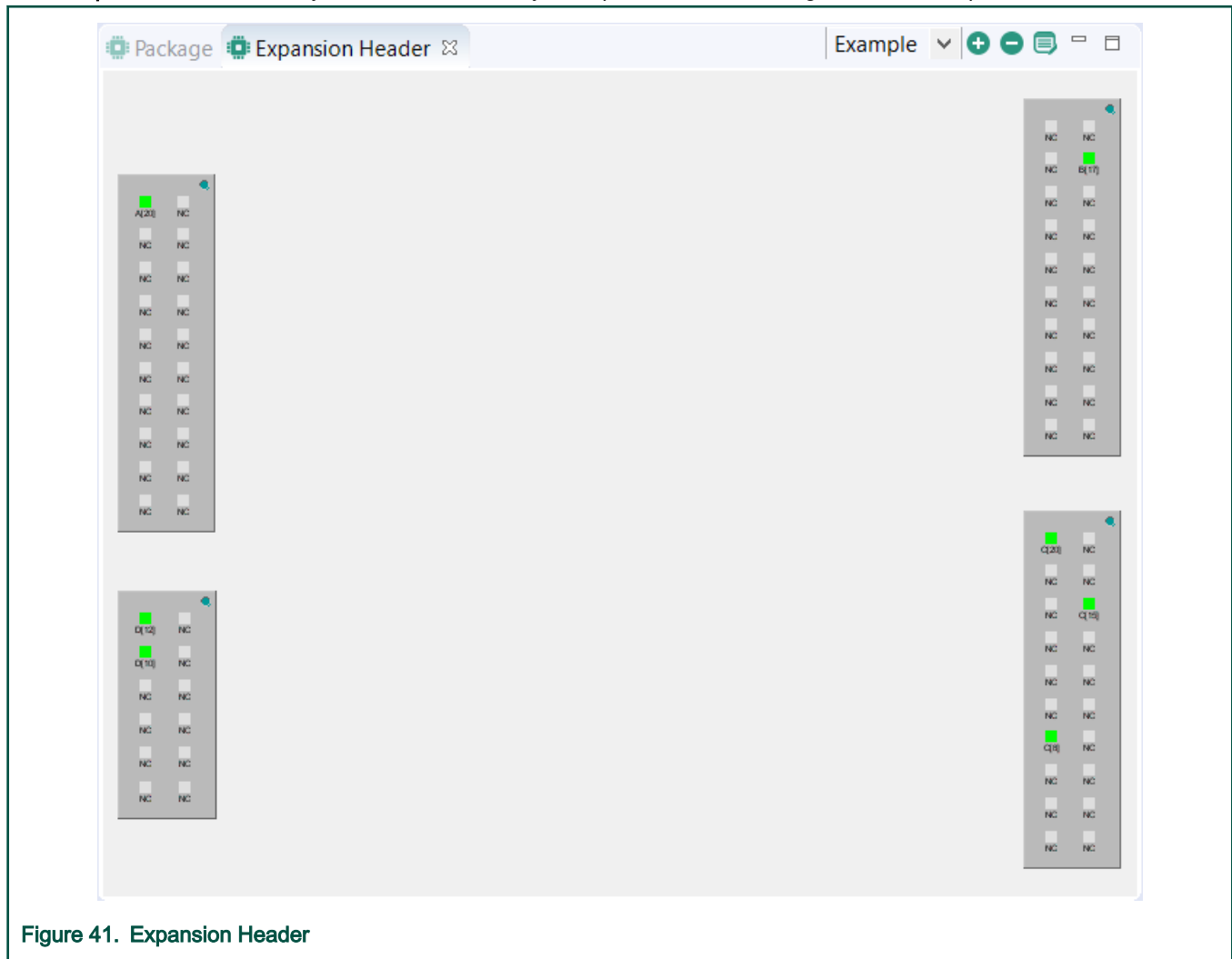


Figure 41. Expansion Header

The expansion header is not automatically preset for every supported device. If the header is not preconfigured, follow these steps to create and modify an expansion header configuration:

1. Open the view by selecting **Views>Expansion Header** from the **Toolbar**.
2. Add a new header by selecting the **Add** button in the view toolbar.
3. In the **Add New Expansion Header** window, select the **Header type** from the dropdown menu.



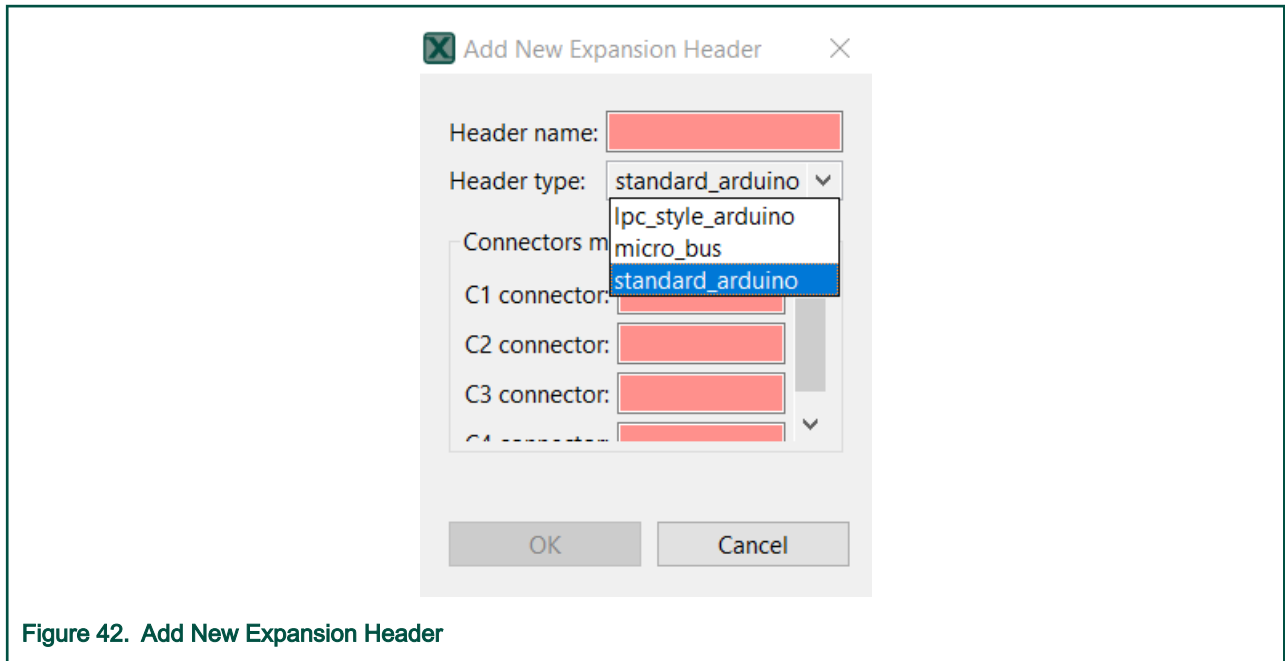


Figure 42. Add New Expansion Header

4. Name the header and map the connectors.

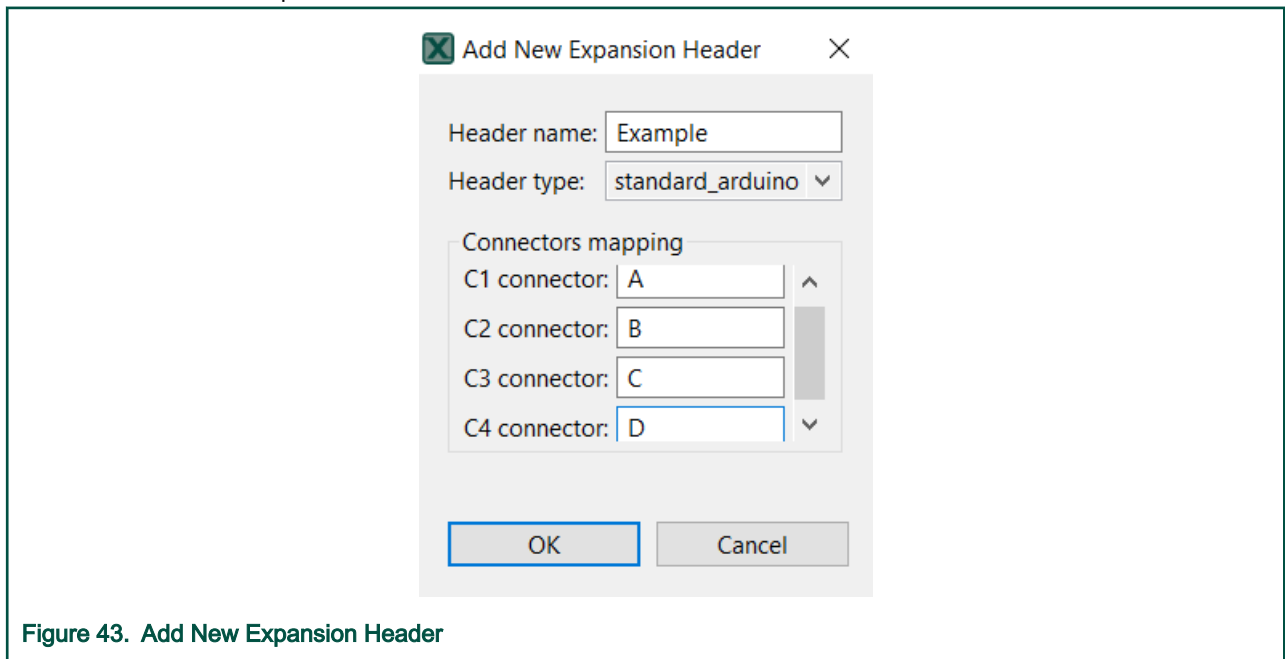


Figure 43. Add New Expansion Header

5. Select OK.

**Expansion Header** view now displays the connector layout. You can point your cursor over the pins to display additional information.

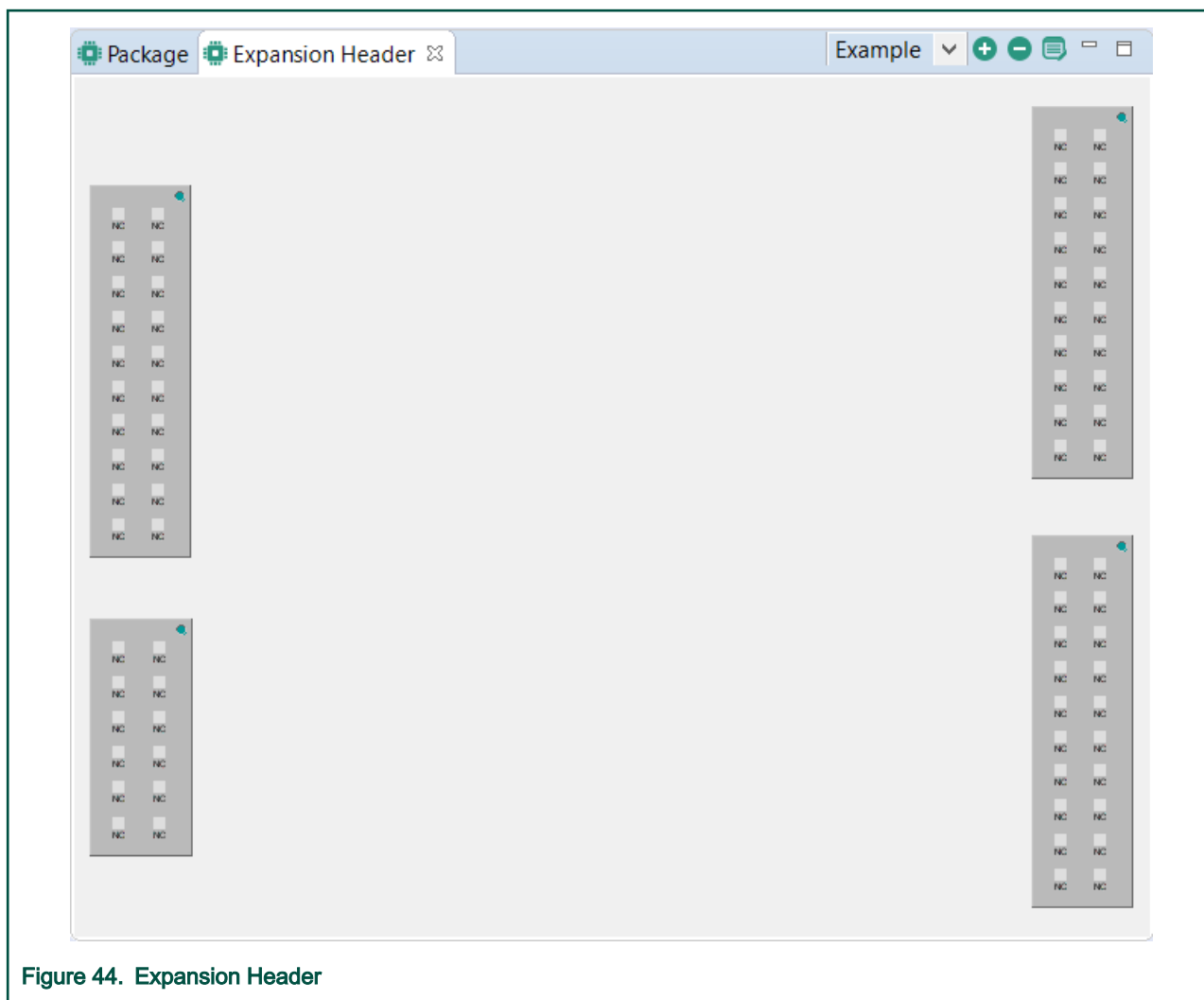


Figure 44. Expansion Header

6. In the **Pins** view, map the connector pin to the processor pin by entering the connector pin label in the new expansion header column.

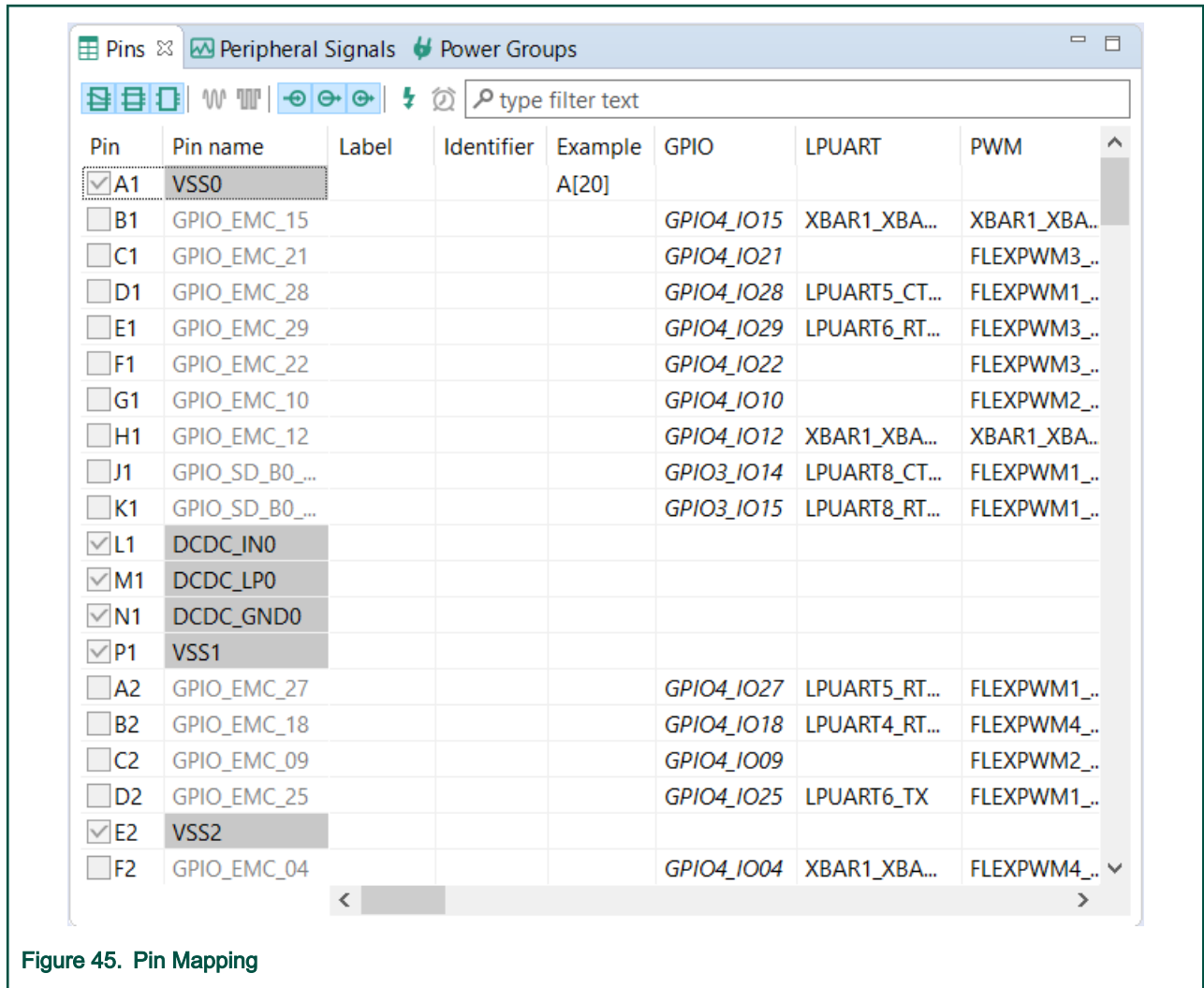


Figure 45. Pin Mapping

- To route the pin, left-click the pin checkbox or right-click the pin row and select **Route**.

The connector pin is now routed.

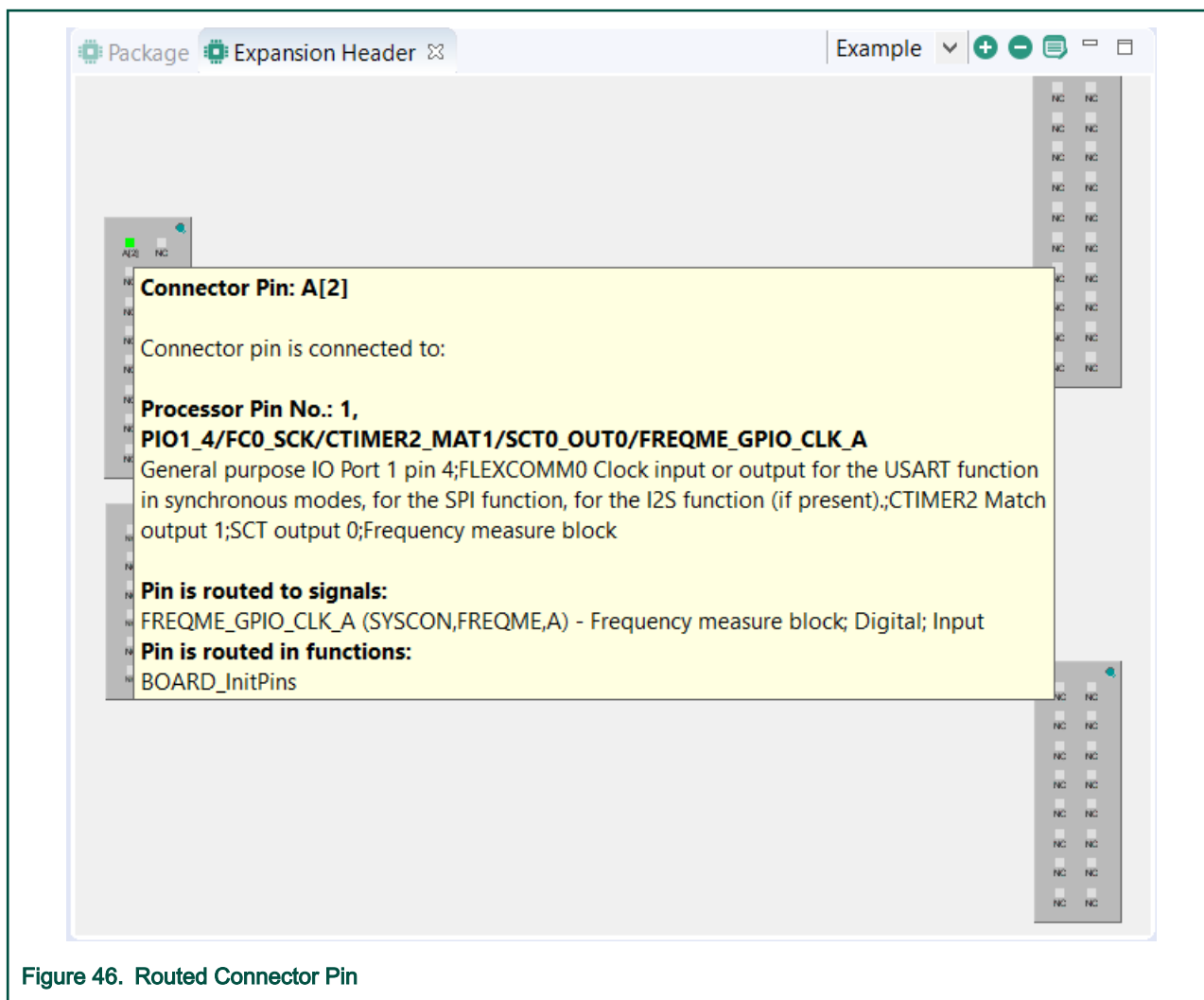


Figure 46. Routed Connector Pin

You can create more than one expansion header configuration. Switch between the configurations in the view's dropdown menu.

You can modify the configuration parameters by selecting the **Edit** button. Information in the **Pins** view is updated automatically.

You can remove a configuration by selecting the **Remove** button.

### 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.

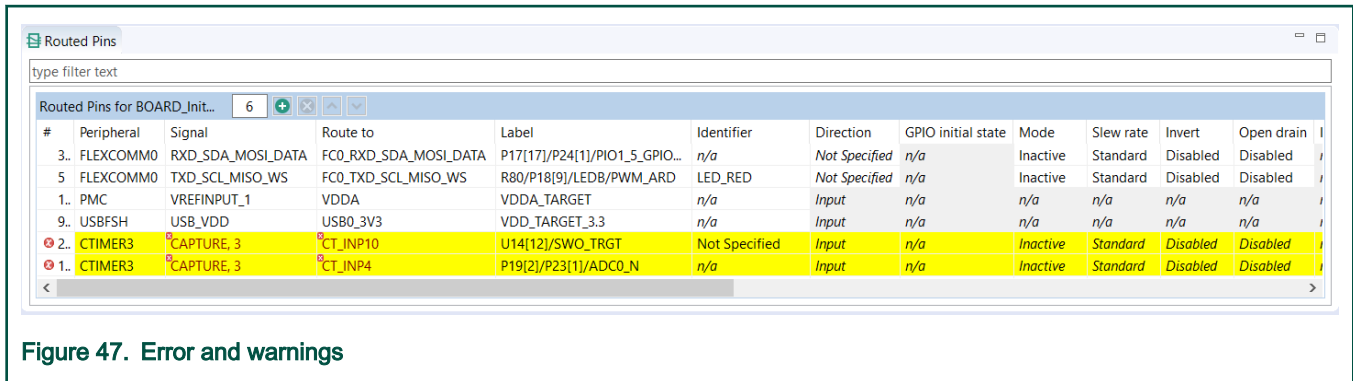


Figure 47. Error and warnings

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

For more information on error and warnings color, see the [Highlighting and Color Coding](#) section.

### 3.5.1 Incomplete routing

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

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

Figure 48. Incomplete routing

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

You can also select **Pins > Automatic Routing** from the Main menu to resolve any routing issues.

#### NOTE

Not all routing issues can be resolved automatically. In some cases, manual intervention is required.

## 3.6 Code generation

If the settings are correct and no error is reported, the code generation engine instantly re-generates the source code. You can view the resulting code the **Code Preview** view of the **Pins** tool.

**Code Preview** automatically highlights differences between the current and immediately preceding iteration of the code. You can choose between two modes of highlighting by clicking the **Set viewing style for source differences**. You can also disable highlighting altogether from the same dropdown menu.

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

#### NOTE

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

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

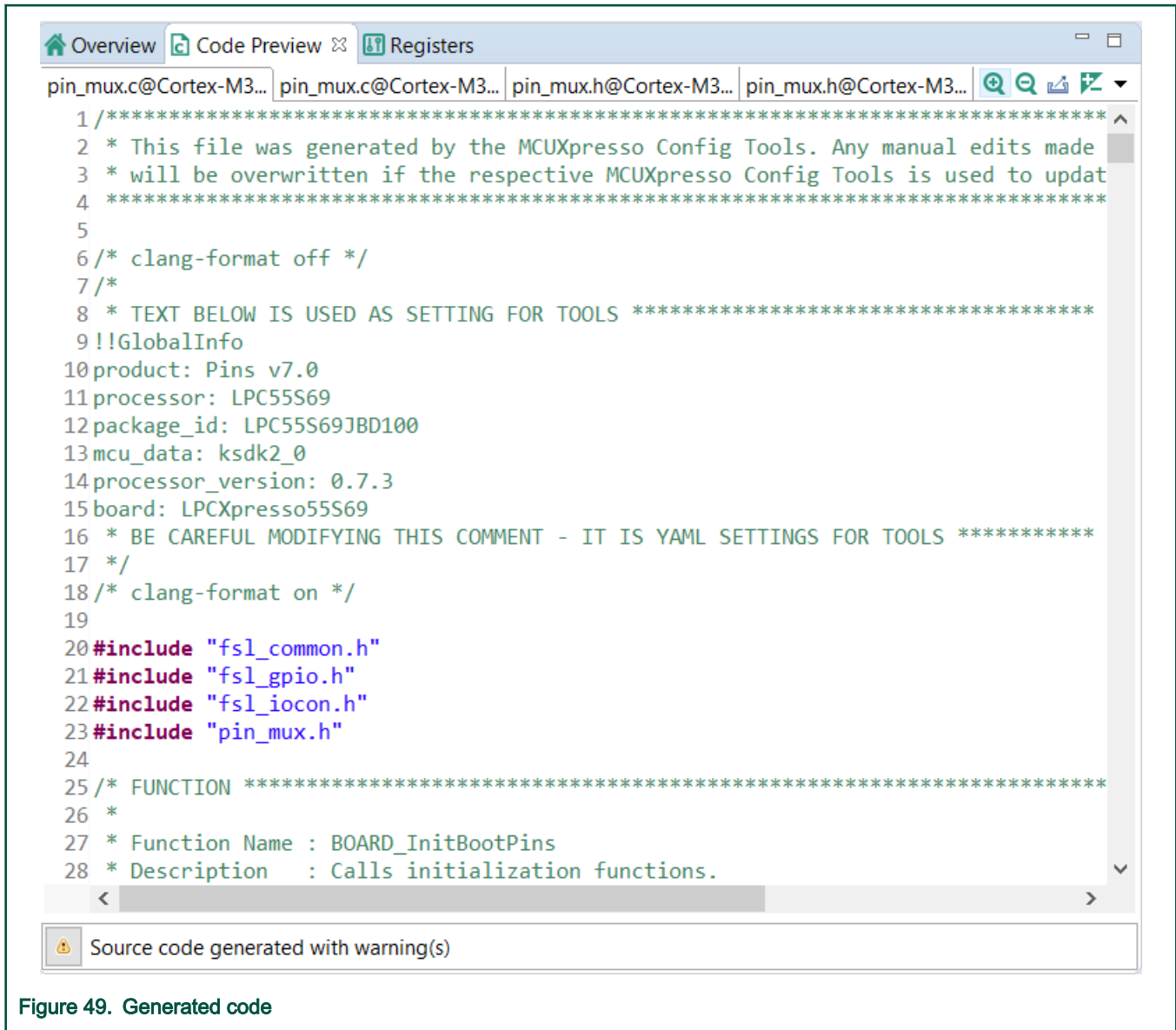


Figure 49. Generated code

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.

## 3.7 Using Pins Definitions in Code

The **Pins** tool generates definitions of named constants that can be leveraged in the application code. Using such constants based on user-specified identifiers allows you to write code which is independent of configured routing. In case the you change the pin where the signal is routed, the application will still refer to the proper pin.

For example, when the `LED_RED` is specified an identifier of a pin routed to `PTB22`, the following defines are generated into the `pin_mux.h`:

```
#define BOARD_LED_RED_GPIO GPIOB /*!<@brief GPIO device name: GPIOB */
#define BOARD_LED_RED_PORT PORTB /*!<@brief PORT device name: PORTB */
```

```
#define BOARD_LED_RED_PIN 22U /*!<@brief PORTB pin index: 22 */
```

The name of the define is composed from function group prefix and pin identifier. For more details, see [Functional groups](#) and [Labels and identifiers](#) sections.

To write to this GPIO pin in application using the SDK driver (`fsl_gpio.h`), you can for example use the following code referring to the generated defines for the pin with identifier *LED\_RED*:

```
GPIO_PinWrite(BOARD_LED_RED_GPIO, BOARD_LED_RED_PIN, true);
```

# Chapter 4

## Clocks Tool

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

### 4.1 Features

Following are the features of the Clock Tool:

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

### 4.2 User interface overview

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



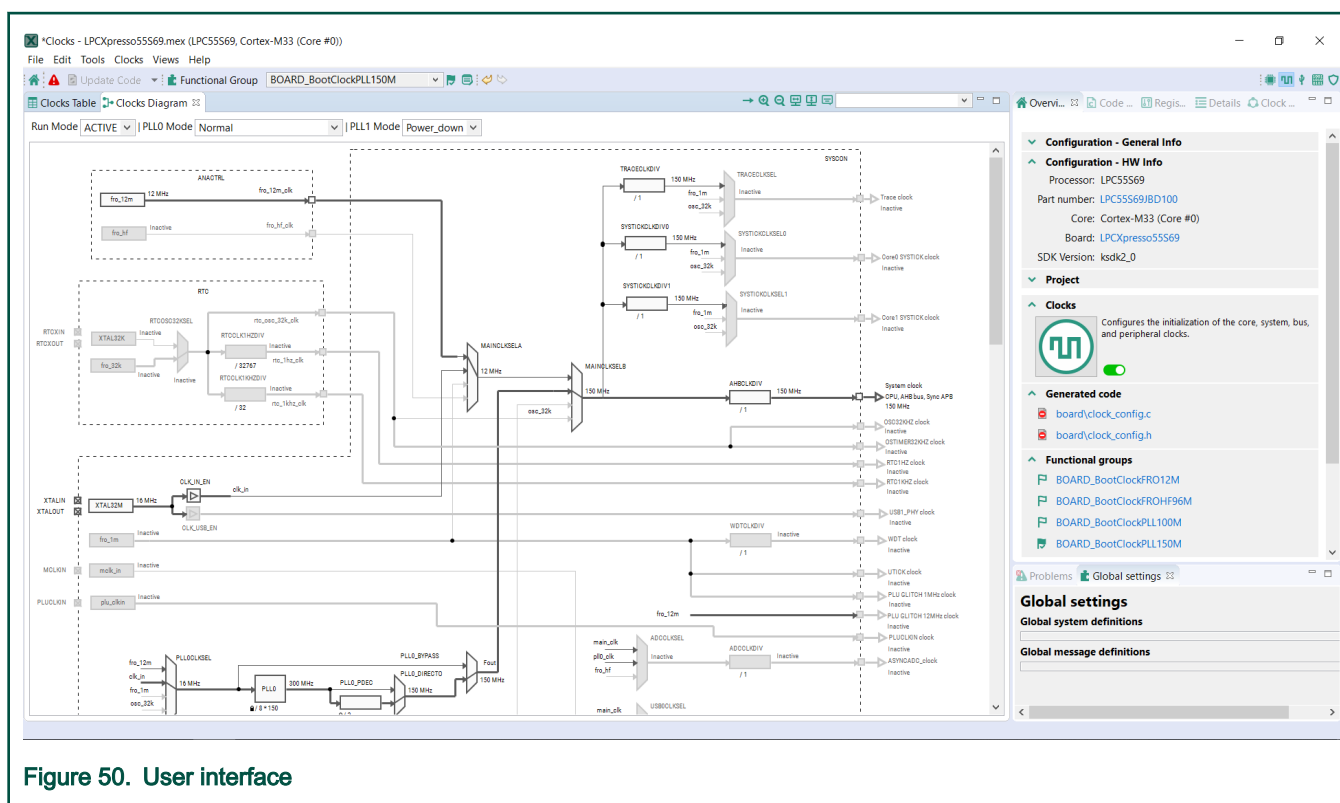


Figure 50. User interface

### 4.3 Clock configuration

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

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

#### NOTE

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

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

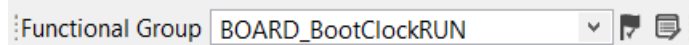


Figure 51. Default clock configuration

#### NOTE

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

### 4.4 Global settings

Global settings, such as Run Mode and MCG mode, influence the entire clock system. It's recommended to set them first. Global settings can be modified in **Clock Table**, **Clock Diagram**, and **Details** views.

NOTE

Global settings can be changed at any time.

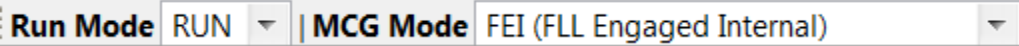


Figure 52. Global settings

4.5 Clock sources

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

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

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

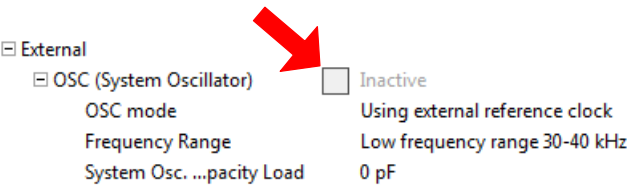


Figure 53. External clock source configuration

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 7. Setting states and markers

State/Style/Marker	Icon	Description
Error marker		Indicates that there is an error in the settings or something related to it. See the tooltip of the setting for details.
Warning marker		Indicates that there is a warning in the settings or something related to it. See the tooltip of the setting for details.
Lock icon		Indicates that the settings (that may be automatically adjusted by the tool) are locked to prevent any automatic adjustment. If the setting can be locked, they are automatically locked

Table continues on the next page...

Table 7. Setting states and markers (continued)

State/Style/Marker	Icon	Description
		<p>when you change the value. To add/remove the lock manually, use the pop-up menu command <b>Lock/Unlock</b>.</p> <p style="text-align: center;"><b>NOTE</b></p> <p>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.</p>
Yellow background	100 MHz	Indicates that the field is directly or indirectly changed by the previous user action.
Gray text	FCTRIM	Indicates that the value of setting does not actively influence the clock. It is disabled or relates to an inactive clock element. For example, on the clock path following the unavailable clock source or disabled element. The frequency signal also show the text "inactive" instead of frequency. The value is also gray when the value is read-only. In such a state it is not possible to modify the value.

## 4.7 Frequency settings

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

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

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

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

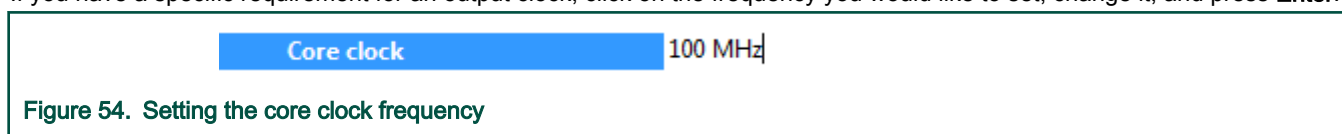


Figure 54. Setting the core clock frequency

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

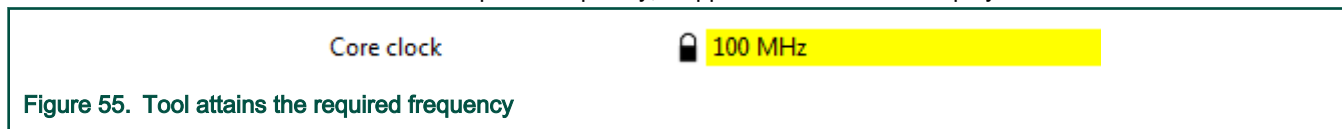


Figure 55. Tool attains the required frequency

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



Figure 56. 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.

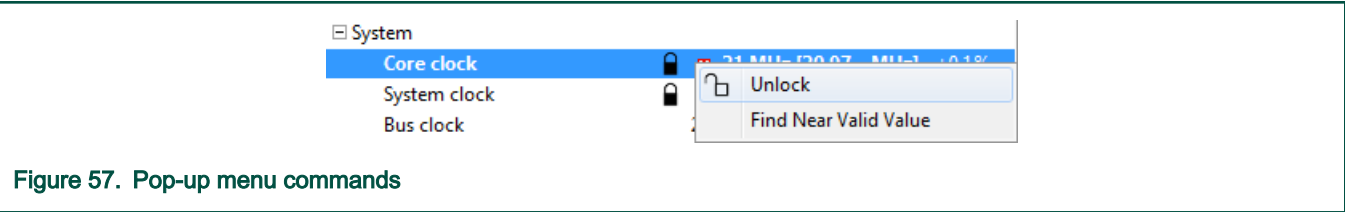


Figure 57. Pop-up menu commands

4.7.2 Frequency precision

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

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

Figure 58. 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.

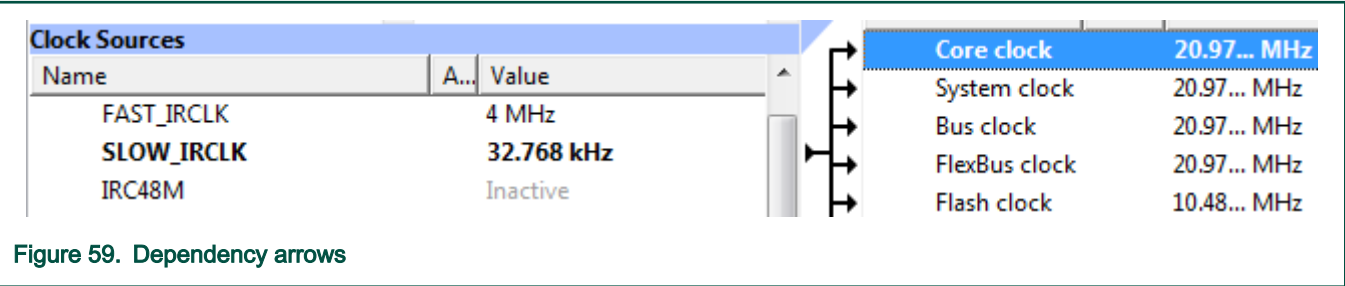


Figure 59. Dependency arrows

4.9 Details view

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

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

Name	C..	L..	Value
☐ Clock input	<input checked="" type="checkbox"/>		16 MHz
Enable CLKIN			Enabled and clocked
XTAL32M Oscillator mod			Normal crystal oscillation mod
☐ 32 MHz clock output			Enabled and clocked
XTAL32MHz ... Frequency			16 MHz
☐ 32 MHz clock output			Disabled
Enable XO32...te Frequen			Inactive
☐ Master clock input	<input type="checkbox"/>		Inactive
MCLK PIN mode			Input mode.
☐ FRO_1M			Inactive
fro 1MHz clock output			Disabled
PLU clock input	<input type="checkbox"/>		Inactive
PLL0 clock select			CLKIN clock.
PLL1 clock select			none
☐ PLL0			
PLL0 Frequency			300 MHz
PLL0 direct input enable			Active
PLL0 feedbac...divider by			Active
Frequency m...ation deptl			k = 0 (fractional mode)
Programmab...n frequenc			Nss = 512 (fm = 3.9 - 7.8 kHz)
Modulation ...orm contro			no compensation
Modulation f...uency dith			Fixed
PLL0N_DIV		<input checked="" type="checkbox"/>	/ 8

Figure 60. Details view

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

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

## 4.10 Clock diagram

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

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

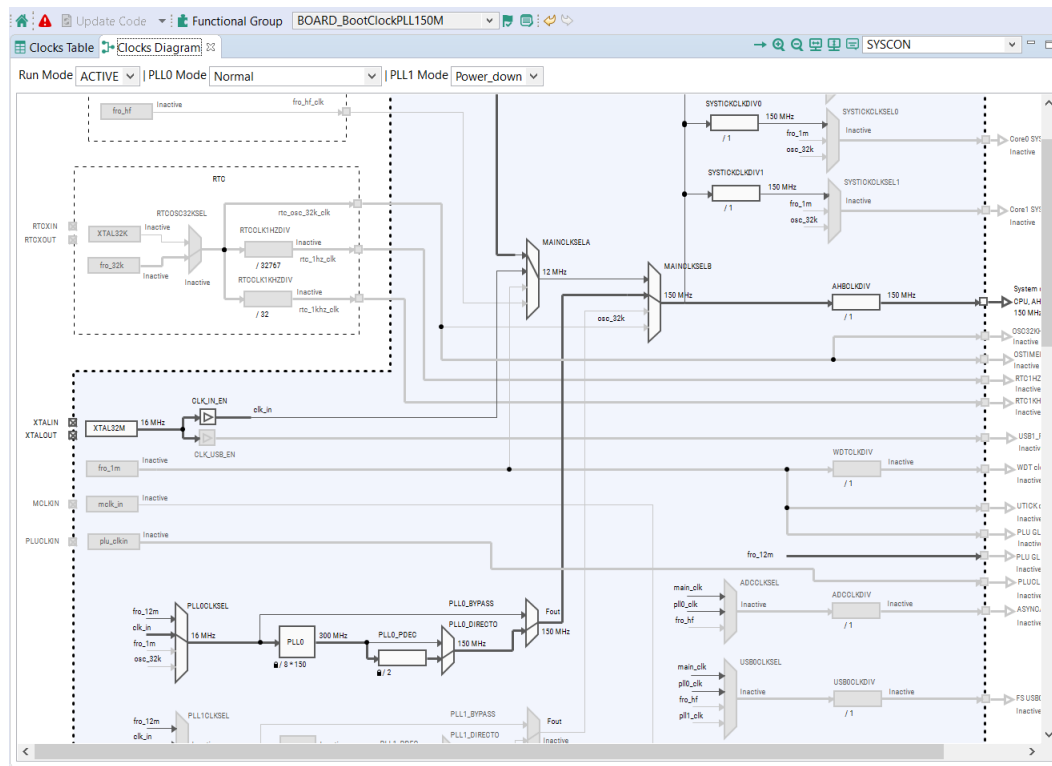
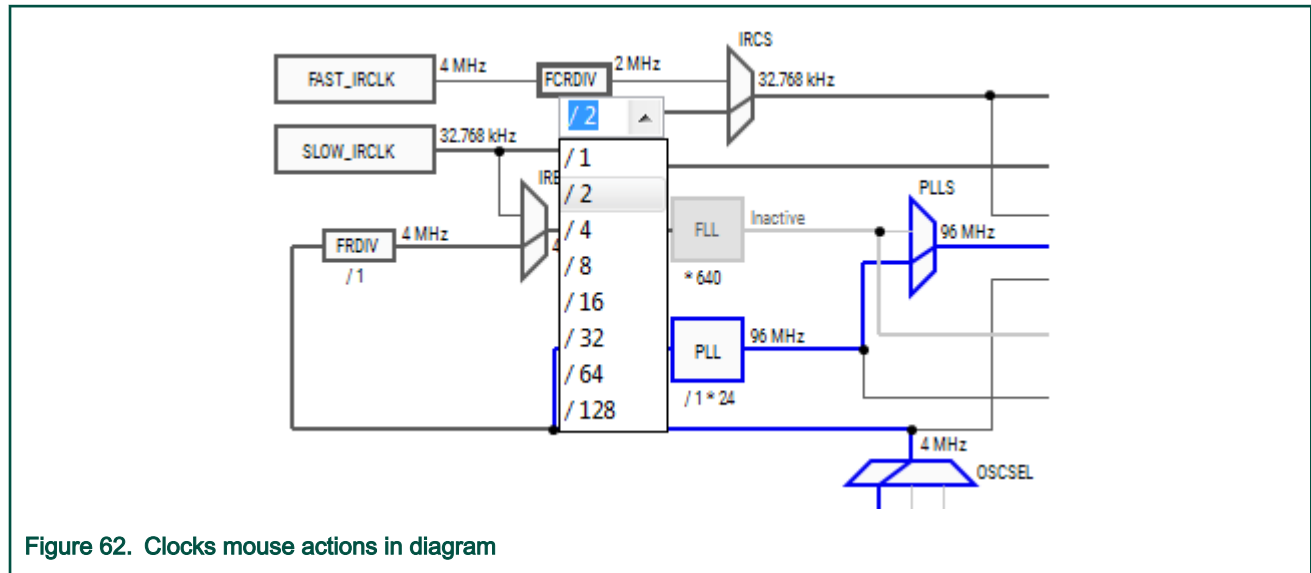


Figure 61. Clock diagram

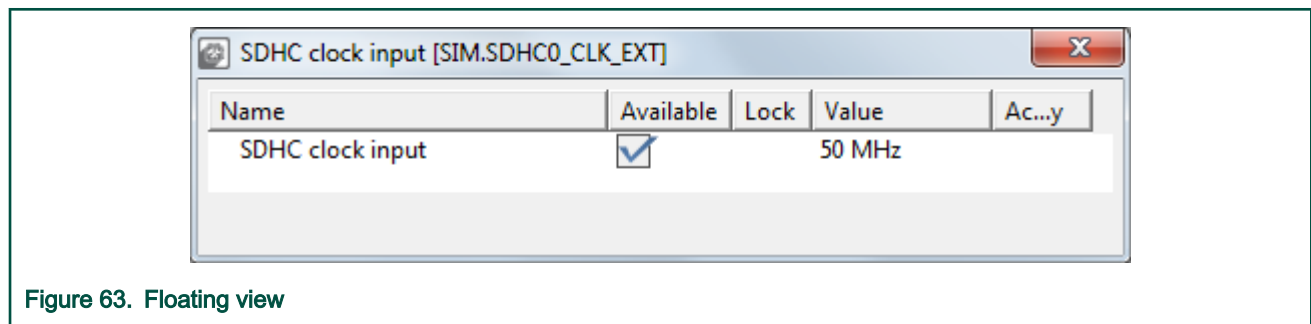
### 4.10.1 Mouse actions in diagram

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

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



- **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.



### 4.10.2 Color and line styles

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

The color and line styles can indicate:

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

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

### 4.10.3 Clock model structure

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

The following are the clock model elements.

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



Figure 64. Clock source

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

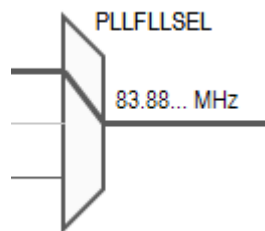


Figure 65. Clocks selector

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



Figure 66. Prescaler

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

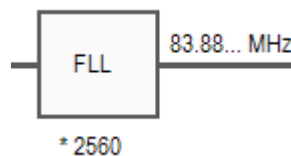


Figure 67. Frequency Locked Loop

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

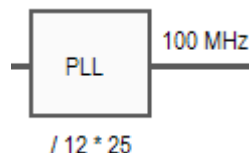


Figure 68. Phase Locked Loop

- **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 69. 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.

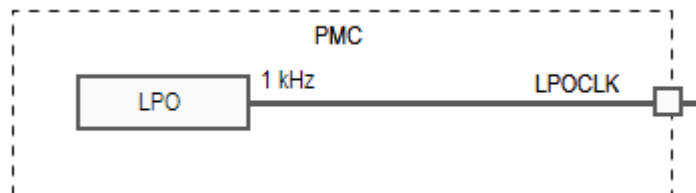


Figure 70. Clock component

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

## 4.11 Clocks menu

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

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

## 4.12 Troubleshooting problems

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

You may encounter any of the following problems:

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

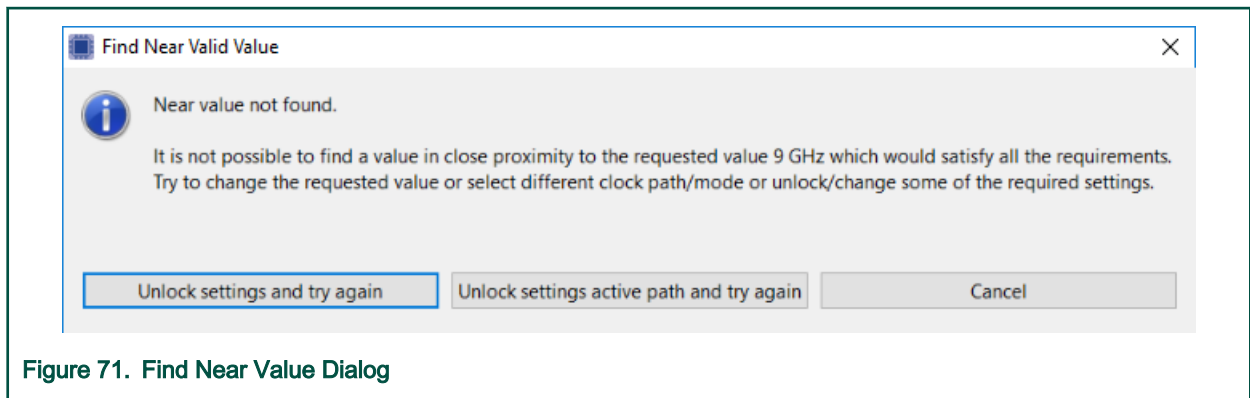
The following are some tips to troubleshoot encountered problems.

1. Find the elements and settings with marked errors in the diagram or tables and see the details in the tooltip.
2. Start with only one locked frequency and let the tool find and calculate other ones. After you are successful you can add more.
3. Go through the locked outputs, if there are any, and verify the requirements (possible errors in the required frequency, wrong units, and so on).
4. If you are OK to have a near around of the requested value, right-click and from the pop-up menu select **Clock output > Find near value**.

5. If you cannot reach the values you need, see the clock paths leading to the clock output you want to adjust and check the selectors if it's possible to switch to another source of clock.
6. Try to remove locks by selecting **Clocks > Unlock All Settings**. In case many changes are required, you can simply reset the model to the default values and start from the beginning. To reset, select **Clocks > Reset to processor defaults**.

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

- **Show problem** - Shows the problem in the **Clocks Diagram** view. If one of 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.

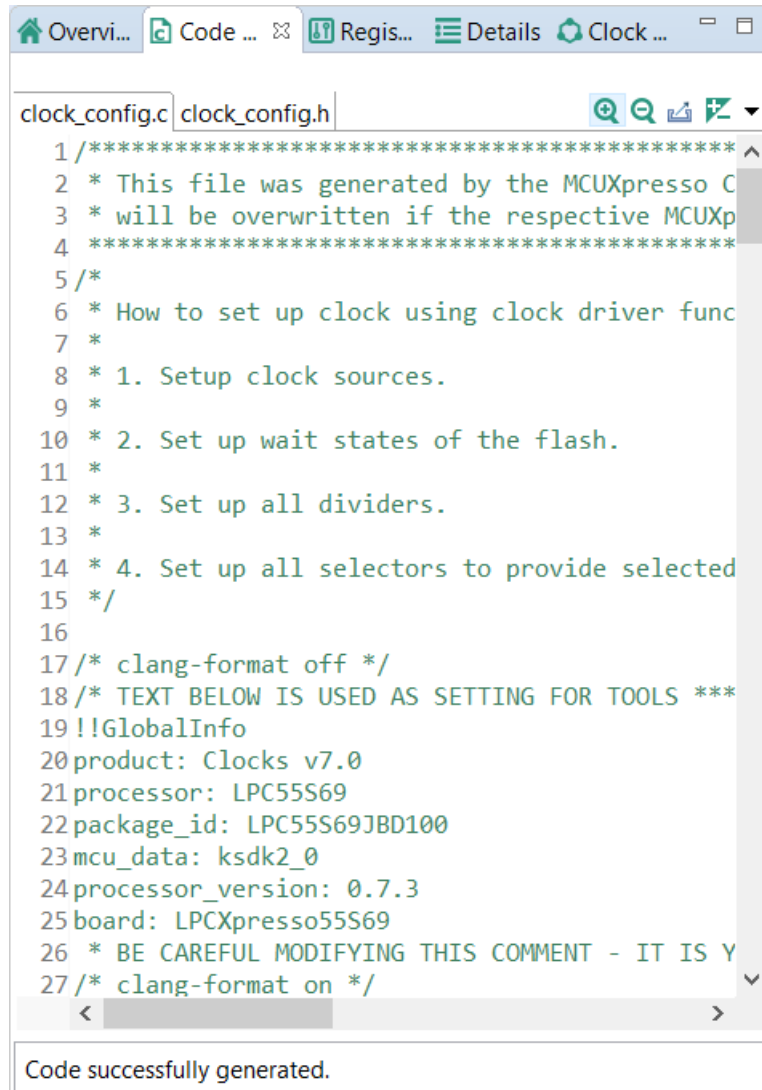


- **Unlock settings active path and try again** - unlocks all elements that lead to selected output and tries to recompute.
- **Unlock settings and try again** - unlocks all locked values and tries to recompute. If automatic value computation fails, nothing will be changed.
- **Cancel** - cancels the modifications.

## 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.

**Code Preview** automatically highlights differences between the current and immediately preceding iteration of the code. You can choose between two modes of highlighting by clicking the **Set viewing style for source differences**. You can also disable highlighting altogether from the same dropdown menu.



The screenshot shows the 'Code ...' tab in the MCUXpresso Config Tools interface. The file 'clock\_config.c' is open, displaying the following content:

```

1 /*****
2  * This file was generated by the MCUXpresso C
3  * will be overwritten if the respective MCUXp
4  *****/
5 /*
6  * How to set up clock using clock driver func
7  *
8  * 1. Setup clock sources.
9  *
10 * 2. Set up wait states of the flash.
11 *
12 * 3. Set up all dividers.
13 *
14 * 4. Set up all selectors to provide selected
15 */
16
17/* clang-format off */
18/* TEXT BELOW IS USED AS SETTING FOR TOOLS ***
19!!GlobalInfo
20product: Clocks v7.0
21processor: LPC55S69
22package_id: LPC55S69JBD100
23mcu_data: ksdk2_0
24processor_version: 0.7.3
25board: LPCXpresso55S69
26 * BE CAREFUL MODIFYING THIS COMMENT - IT IS Y
27/* clang-format on */

```

At the bottom of the window, a status bar indicates: "Code successfully generated."

Figure 72. Code Preview

### 4.13.1 Working with the code

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

To transfer the code into your project, do the following in the **Code Preview**:

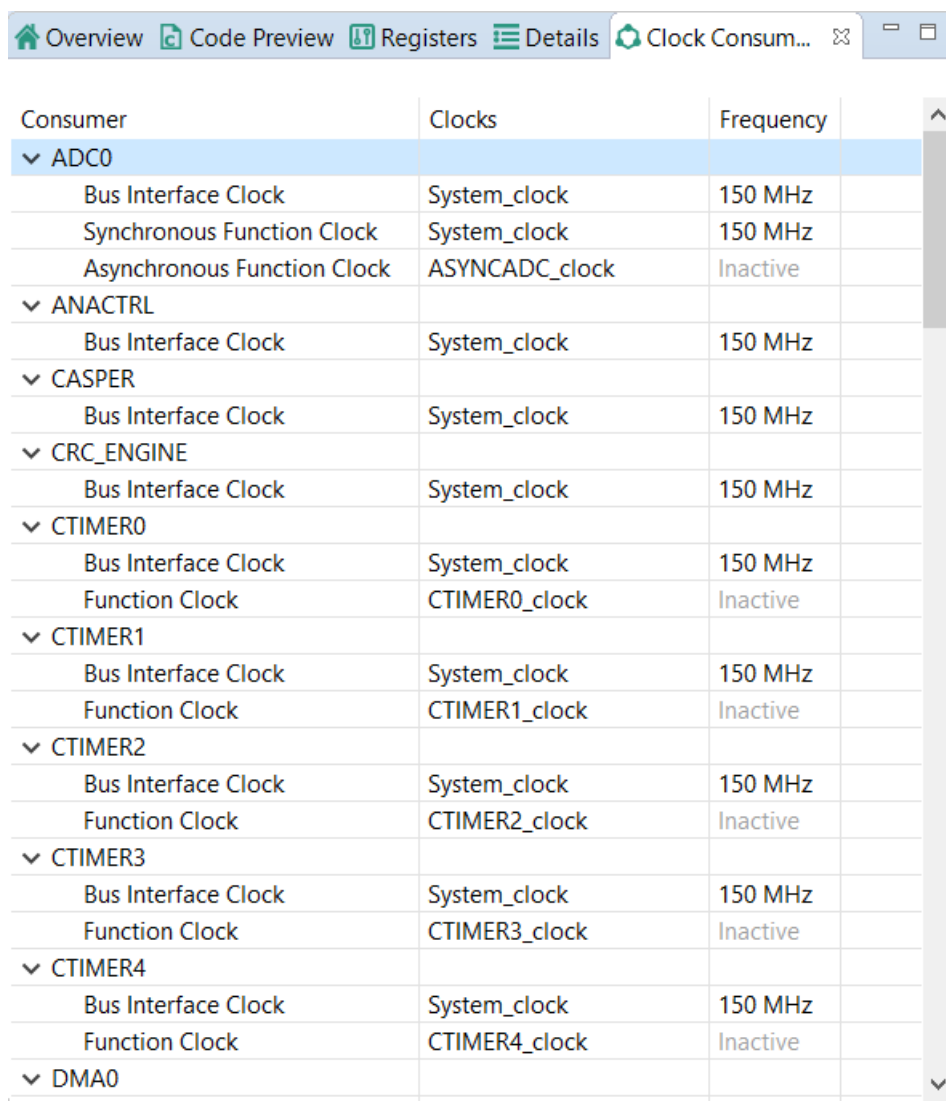
- Copy the content using the COPY command, either by pressing the CTRL+C keys or the pop-up menu after the whole text is selected.
- Use export command.
- Click the **Export** button in **Code Preview** view.
- Click **Update Code** in the toolbar.

## 4.14 Clock Consumers view

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

### NOTE

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



Consumer	Clocks	Frequency
▼ ADC0		
Bus Interface Clock	System_clock	150 MHz
Synchronous Function Clock	System_clock	150 MHz
Asynchronous Function Clock	ASYNADC_clock	Inactive
▼ ANACTRL		
Bus Interface Clock	System_clock	150 MHz
▼ CASPER		
Bus Interface Clock	System_clock	150 MHz
▼ CRC_ENGINE		
Bus Interface Clock	System_clock	150 MHz
▼ CTIMER0		
Bus Interface Clock	System_clock	150 MHz
Function Clock	CTIMER0_clock	Inactive
▼ CTIMER1		
Bus Interface Clock	System_clock	150 MHz
Function Clock	CTIMER1_clock	Inactive
▼ CTIMER2		
Bus Interface Clock	System_clock	150 MHz
Function Clock	CTIMER2_clock	Inactive
▼ CTIMER3		
Bus Interface Clock	System_clock	150 MHz
Function Clock	CTIMER3_clock	Inactive
▼ CTIMER4		
Bus Interface Clock	System_clock	150 MHz
Function Clock	CTIMER4_clock	Inactive
▼ DMA0		

Figure 73. Clock Consumers view

# Chapter 5

## Peripherals Tool

### 5.1 Features

The Peripherals Tool features:

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

### 5.2 Basic Terms and Definitions

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

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

### 5.3 Workflow

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

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

**NOTE**

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

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

**NOTE**

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

6. You can use **Update Code** command from the toolbar. If not, you can export the source code by selecting **File>Export...** from the **Main Menu**.

**NOTE**

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

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

## 5.4 User interface overview

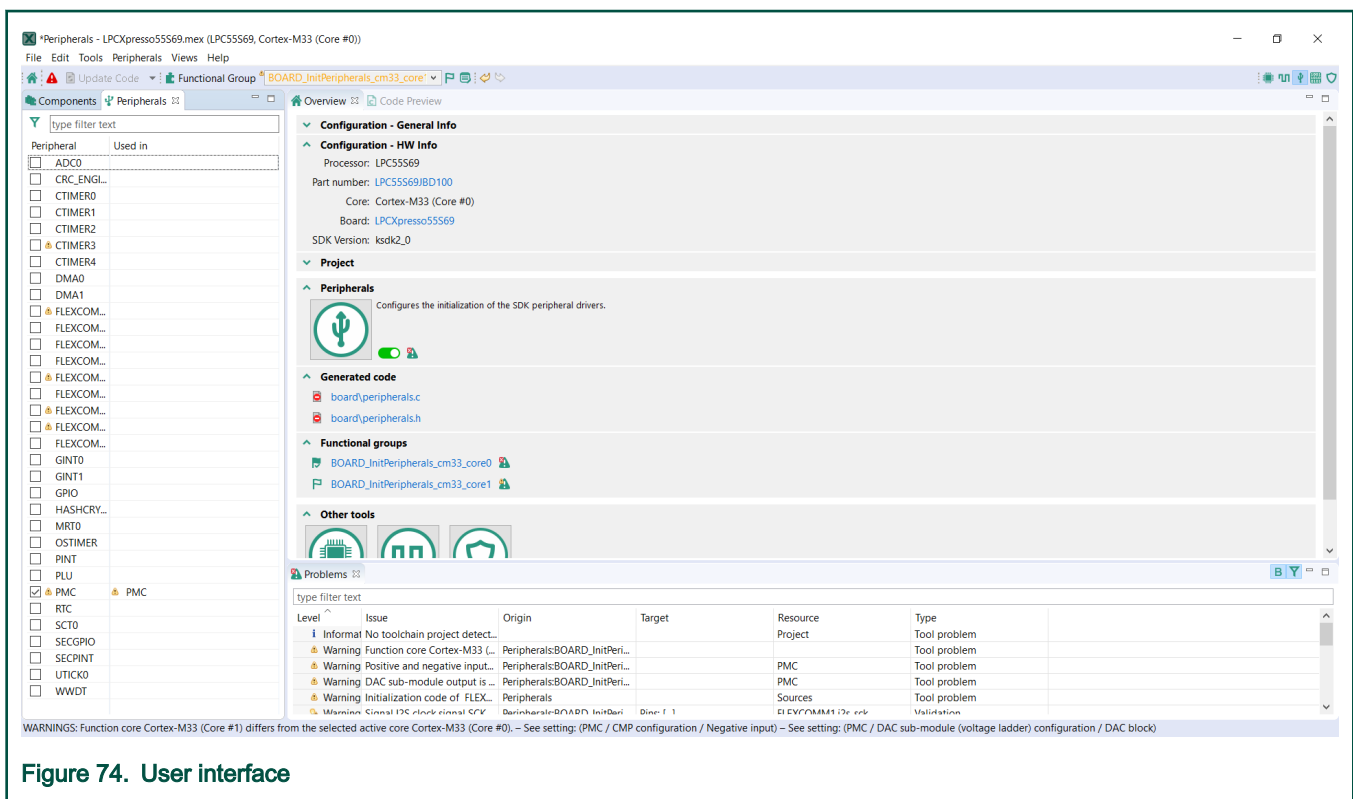


Figure 74. User interface

## 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:

- **Config Tools Overview** - Opens the **Overview** with information about currently-used tools.
- **Show Problems View** - Opens the **Problems** view.
- **Update Code** - Opens update dialog allowing you to update generated peripheral initialization code directly within specified toolchain project.
- **Global settings** - Opens a tab aggregating global settings of all configuration sets.

- **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** – Contains icons of individual tools. Click these icons to switch the currently visible tool.
- **Undo/Redo** - Allows you to undo/redo last actions.
- 

---

**NOTE**

---

For details on other commands, refer [Toolbar](#)

---

## 5.6 Documentation view

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

---

**NOTE**

---

Not all components might have this option enabled.

---

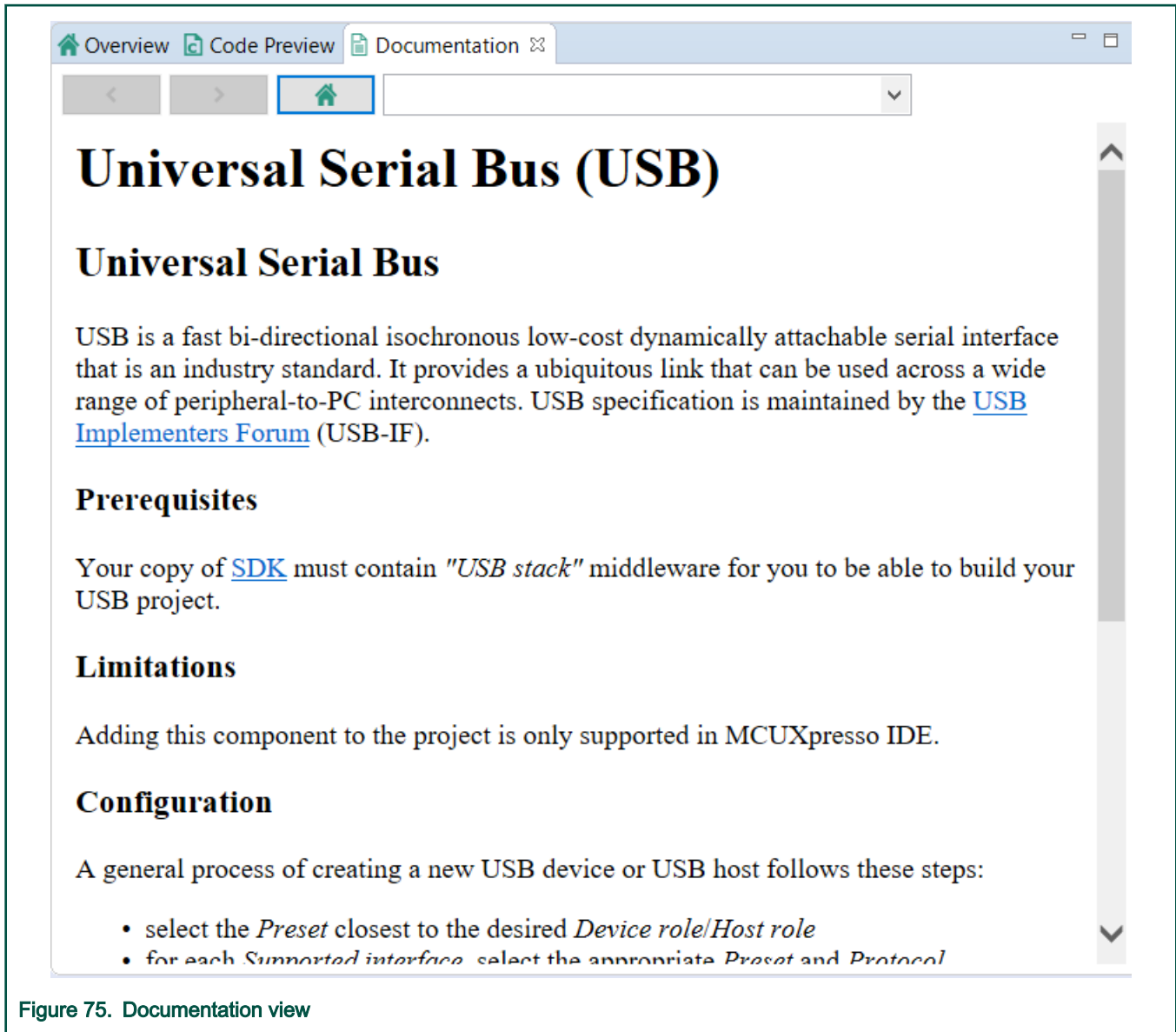


Figure 75. Documentation view

You can open the **Documentation** view in three different ways:

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

## 5.7 Peripherals view

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

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

Second column contains a name of component instance handling the peripheral. This name is customizable in the settings editor and it is used in generated code. The name of the component instance can't contain spaces.

You can enable an instance by selecting the checkbox, or by clicking the switch in the settings editor of the component instance.



Disable a component instance by deselecting it.

Double-click on the second column to open the **Settings Editor** for the component instance.

Right-click the row to open a context menu. The context menu allows you to:

- **Open** - Open the component instance in the **Settings Editor** (if enabled). In case more instances are enabled for the peripheral, you need to choose between them.
- **Add a component instance** - Add a component instance to the peripheral.
- **Documentation** - Open the **Documentation view** (if applicable).
- **Remove** - Remove the component instance (if more instances are in use, a confirmation window will allow you to select which instance you want to remove).
- **Migrate** - Migrate the component to a different component type or to a component with a newer driver version.
- **Edit comment** - Add/edit custom notes for the component instance.
- **Enable/Disable** - Enable/Disable the component instance. In case more instances are enabled for the peripheral, you need to choose between them.
- **Move to** - Move the component instance to a different **Functional Group**.
- **Copy to** - Copy the component instance to another **Functional Group**.

## 5.8 Components view

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

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

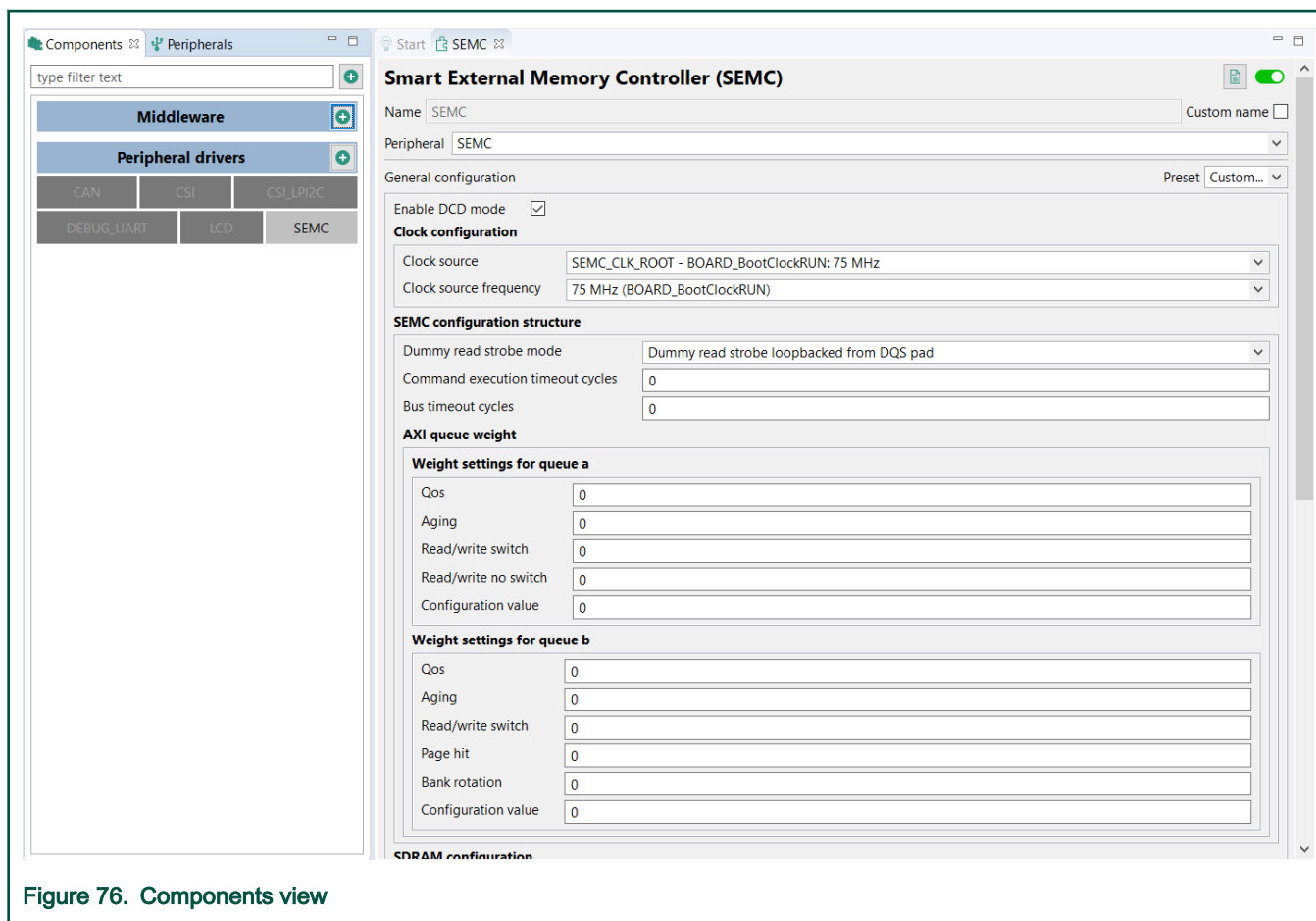


Figure 76. Components view

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

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

#### NOTE

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

- **Migrate** - Right-click the configuration component and choose **Migrate** to migrate the component to a different component type or to a component with a newer driver version.
- **Edit comment** - Right-click the configuration component and choose **Edit comment** to add/edit custom notes for that component.
- **Enable/disable the configuration component** - Right-click the configuration component and choose **Enable** or **Disable** to enable/disable the configuration component.
- **Move the configuration component to another functional group** - Right-click the configuration component and choose **Move to** to select from a list of functional groups to move the configuration component to.

- **Copy the configuration component to another functional group** - Right-click the configuration component and choose **Copy to** to select from a list of functional groups to copy the configuration component to.
- **Add new configuration components** - Left-click the **+** button and choose from the list to add a new component. You can filter the list to show only toolchain-project-relevant, or latest version components. You can also click the **+** buttons next to **Middleware/Peripheral drivers/Other** categories to add new components in them directly.

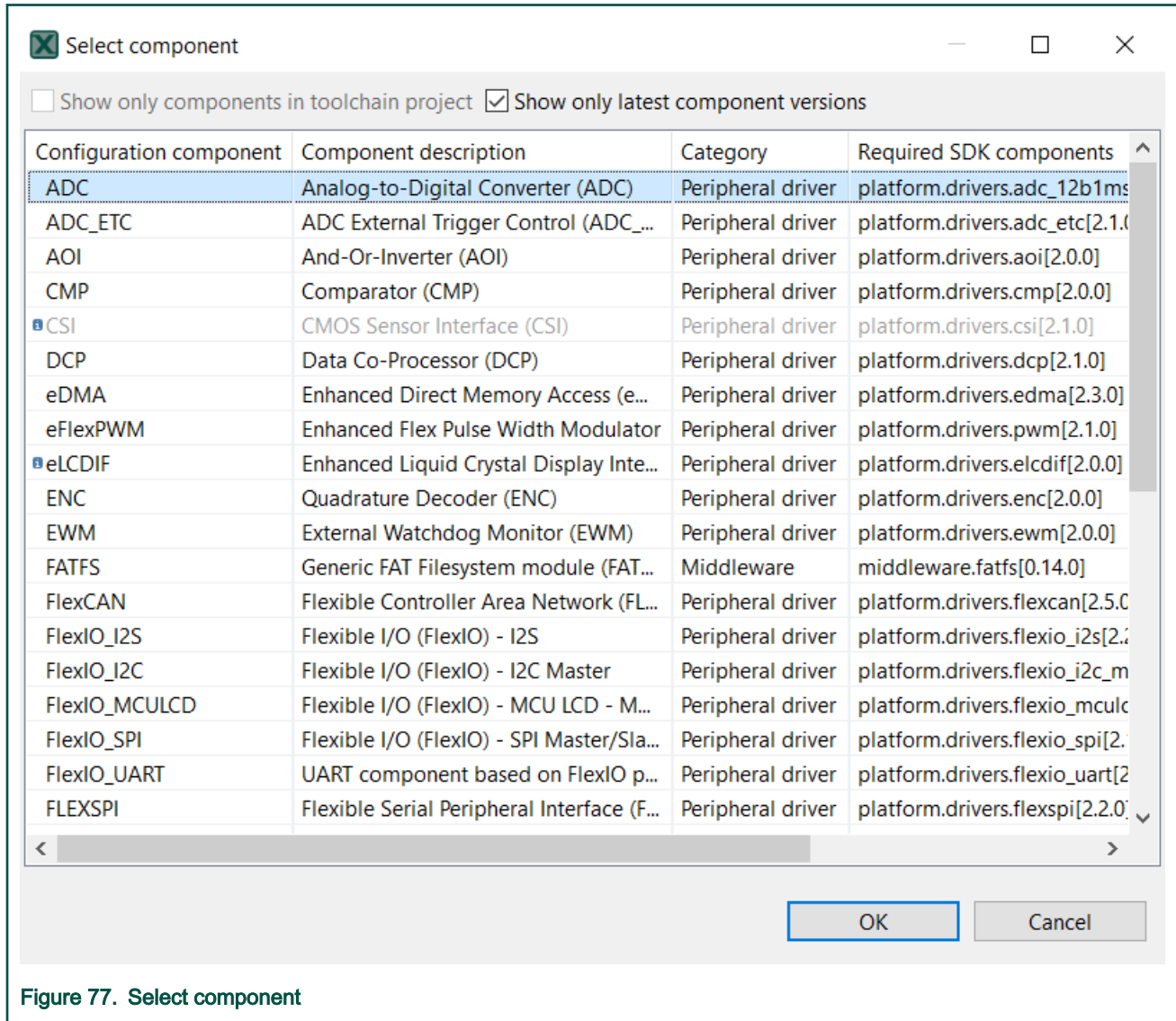


Figure 77. Select component

- **Filter configuration components by name** - Type a text string to filter configuration component names in the search bar.

## 5.9 Settings Editor

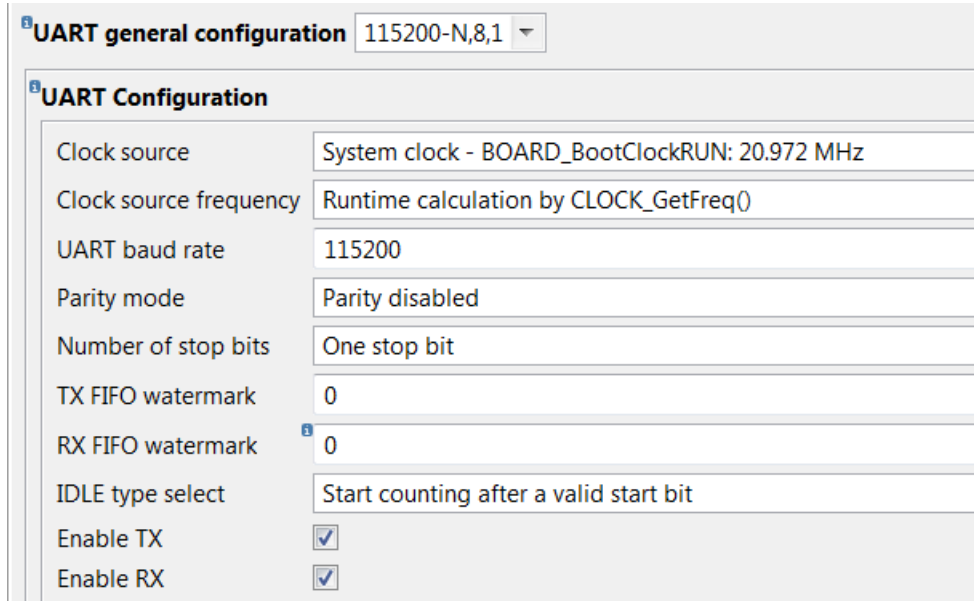
You can edit peripheral component settings in the **Settings Editor**. Open editors are shown in the central area of the screen, each with its own tab. Multiple editors can be opened at the same time. Changes done in the editor are immediately applied and kept even if the settings editor is closed. Settings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

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

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

### 5.9.1 Quick selections

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



The screenshot shows the 'UART general configuration' window with a dropdown menu set to '115200-N,8,1'. Below it is the 'UART Configuration' section with the following settings:

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

Figure 78. Quick selection example

### 5.9.2 Settings

The following settings occur in the editor.

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

Enable Rx/Tx interrupt ☒

Figure 79. Boolean setting example

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

Priority 100

Figure 80. Integer/Float setting example

- **String** – Textual input. More than a single entry can be supported.

Handler name Test

Figure 81. String setting example

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

Interrupt PORTA\_IRQn

Figure 82. Enumeration setting example

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

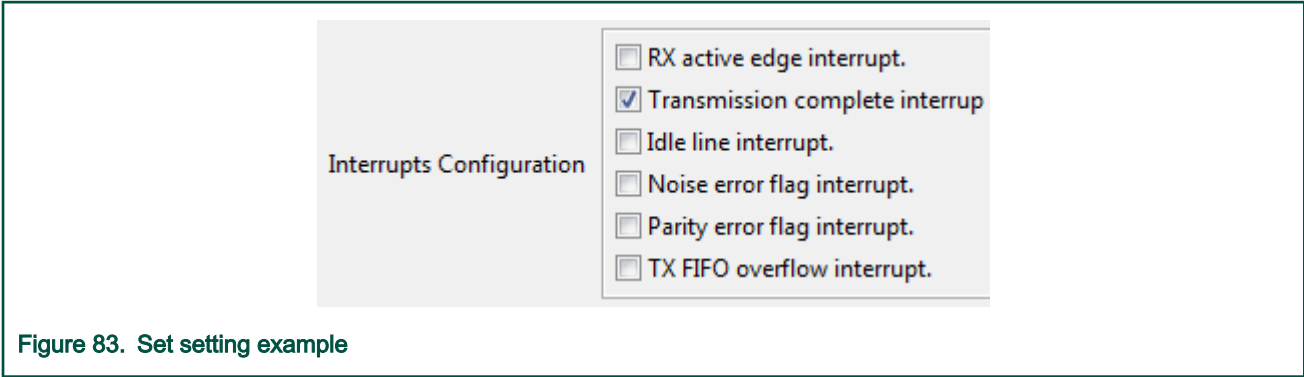


Figure 83. Set setting example

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

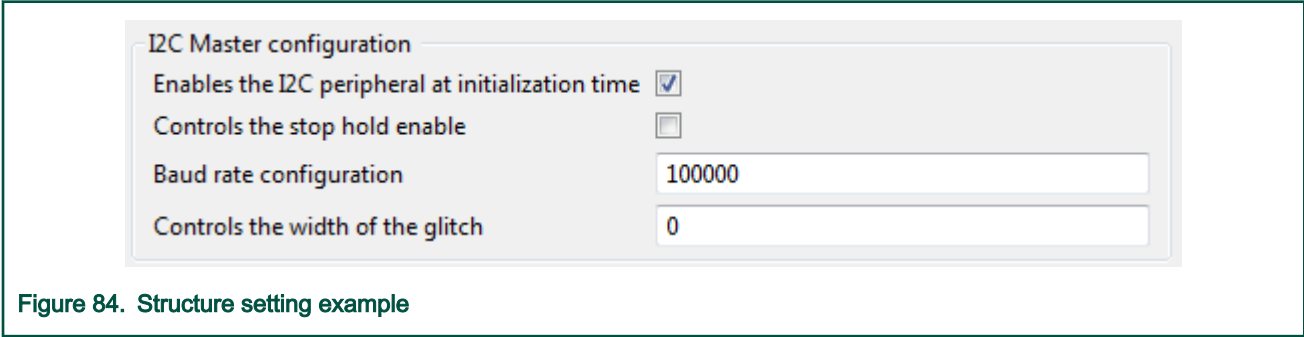


Figure 84. Structure setting example

- **Array** – Array of multiple settings of same type – you can add/remove items. The array of simple structures may also be represented as a table grid, master-detail, and as radio buttons.

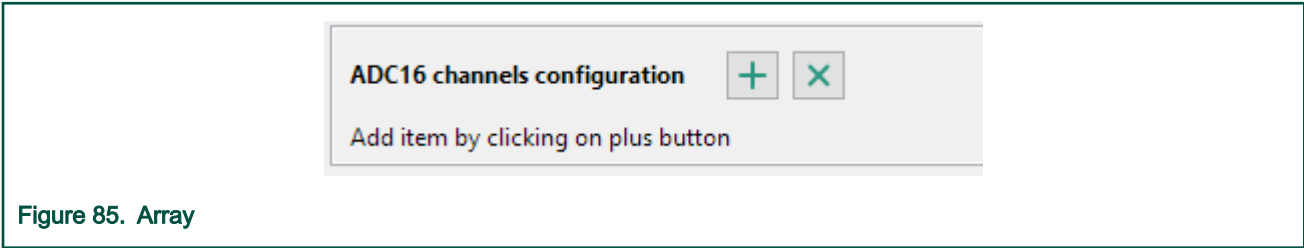


Figure 85. Array

The '+' button adds a new item at the end of array. To rearrange the position or delete an item, right-click the item and select one of the following options: **Move up**, **Move down**, **Move to top**, **Move to bottom**, or **Remove**. You can also copy-paste an array from one instance to another by right-clicking the array label and choosing **Copy**. You can then navigate to another instance array, right-click the table and choose **Paste** to add it.

**NOTE**  
System clipboard is not used for this purpose.

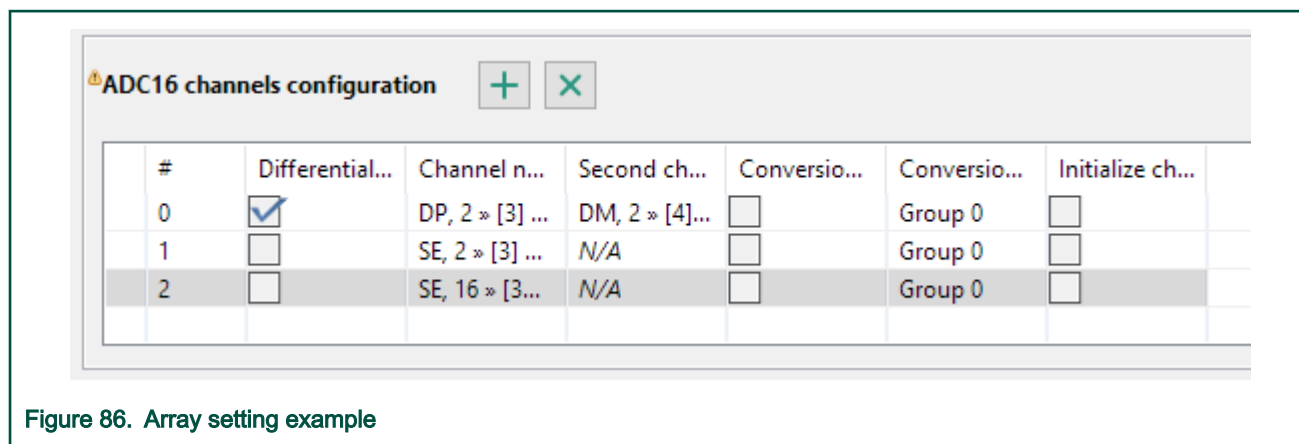


Figure 86. Array setting example

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

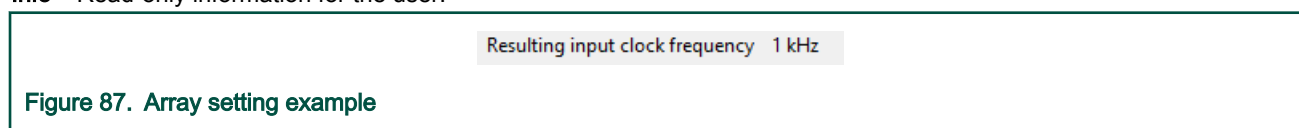


Figure 87. Array setting example

- **File setting** - Link/import an external settings file.

### 5.9.3 Settings Editor header

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

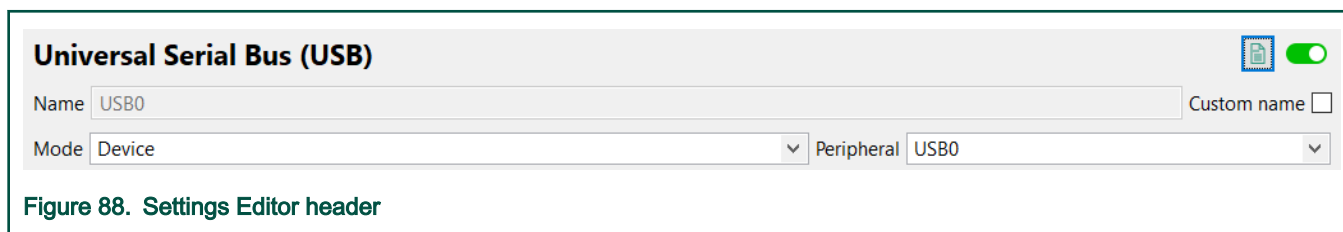


Figure 88. Settings Editor header

**Settings Editor** header contains the following:

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

## 5.10 SEMC Validation tool

If you are developing hardware with external memory, you can validate the memory settings with the **Memory Validation** tool. The tool is available for all SEMC, FlexSPI/NOR, and FCB peripherals and can be used after selecting these peripheral from the list in the **Peripherals** view.

Click the **Validation** button in the **Settings Editor** to open the **Validation** view and run validation scenarios for SEMC memory settings.

#### NOTE

The Memory Validation tool requires Python 2.7 to run. You must manually install the latest Python 2.7 from <https://www.python.org/downloads> and make sure arm-none-eabi-gdb-py starts without error. It's possible that Windows requires that python27.dll is installed.

Should the settings prove valid, you can click the **Sync with DCD** button to synchronize the memory settings with the **DCD** tool. You need to manually update the existing sdram configuration from the DCD tool with the one generated by clicking **Apply to DCD**. Also, if a configuration is not defined in the DCD tool, the configuration generated by clicking **Apply to DCD** will not work as it is, and additional configuration will need to be added in the Clocks or Pins tools.

### 5.10.1 Validation view

Use the **Validation** view to run validation scenarios for your memory settings and analyse the results. You can choose scenarios, tests to run in these scenarios, and view the test results, logs, and summary.

To run validation tests, do the following:

1. Select the correct connection type and COM port in the **Connections** area. Alternatively, scan for available COM ports by clicking the **Scan for available COM ports** button.
2. Choose a scenario you wish to test in the **Scenarios** sub-view.
  - a. To test the Phy configuration:
    - i. Select **Firmware Init > Firmware Init test**.
  - b. To perform a basic memory check by running Write-Read-Compare/Walking Ones/Walking Zeros:
    - i. Select **Operational > Operational tests**.
    - ii. Select **Choose Tests**.
    - iii. Select the tests to perform and set the test parameters.
  - c. To run stress tests:
    - i. Select **Stress Tests > Stress tests**.
    - ii. Select **Choose Tests** and set the test parameters.
  - d. To run diags tests:
    - i. Select **Diags** and choose between **Diags margin state/Diag Write Margin/Diag Read Margin**.
    - ii. Select **Choose Tests** and set the test parameters.
3. To start validation, click the **Start Validation** button.
4. Observe the results in real-time in the **Results** sub-view.
5. Inspect the results in the **Summary** and **Logs** sub-views.
6. View the U-boot-compatible code in **Core Preview**. You can export the generated code for importing into U-boot by selecting the **Export** button.
7. Download DDRV reports using the **Configure and generate DDRV reports** button.

## 5.11 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.

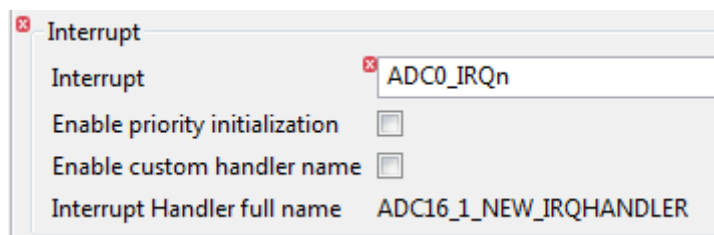


Figure 89. Error decorators

In the case of a dependency error, a quick-fix button is displayed.



Figure 90. Quick Fix 1

Right-click the button to display a list of issues, then left-click the issue to display possible solutions.

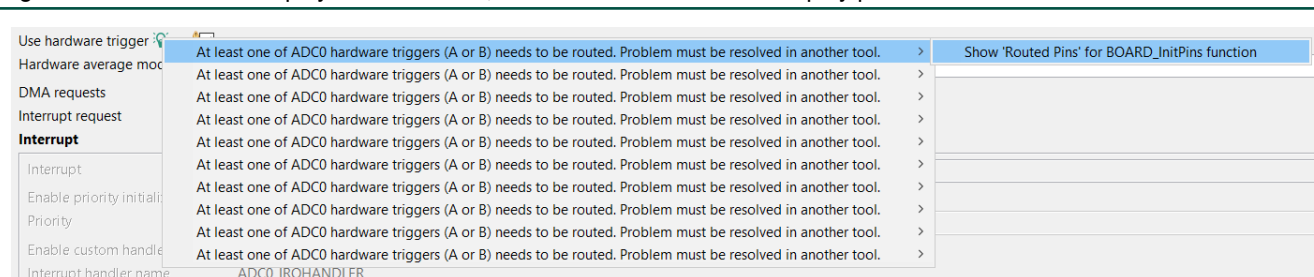


Figure 91. Quick Fix 2

## 5.12 Code generation

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

**Code Preview** automatically highlights differences between the current and immediately preceding iteration of the code. You can choose between two modes of highlighting by clicking the **Set viewing style for source differences**. You can also disable highlighting altogether from the same dropdown menu.

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.

### NOTE

Some components, such as the USB or FlexSPI, may generate additional output files.

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.

```

1  /*****
2  * This file was generated by the S32 Configuration Tools. Any manual edits ma
3  * will be overwritten if the respective S32 Configuration Tools is used to up
4  *****/
5
6  /* clang-format off */
7  /* TEXT BELOW IS USED AS SETTING FOR TOOLS *****/
8  !!GlobalInfo
9  product: Peripherals v6.0
10 processor: S32V234
11 package_id: S32V234_621
12 mcu_data: s32sdk_s32v23x_rtm_100
13 processor_version: 0.0.0
14 functionalGroups:
15 - name: BOARD_InitPeripherals
16   called_from_default_init: true
17   selectedCore: core0
18 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS *****/
19 /* clang-format on */
20
21 /*****
22 * Included files
23 *****/
24 #include "peripherals_crc_1.h"
25
26 /*****
27 * crc_1 initialization code
28 *****/
29 /* clang-format off test*/
30 /* TEXT BELOW IS USED AS SETTING FOR TOOLS *****/
31 instance:

```

Figure 92. Code Preview

# Chapter 6

## Device Configuration Tool

**Device Configuration** tool allows you to configure the initialization of memory interfaces of your hardware. Use the **Device Configuration Data (DCD)** view to create different types of commands and specify their sequence, define their address, values, sizes, and polls.

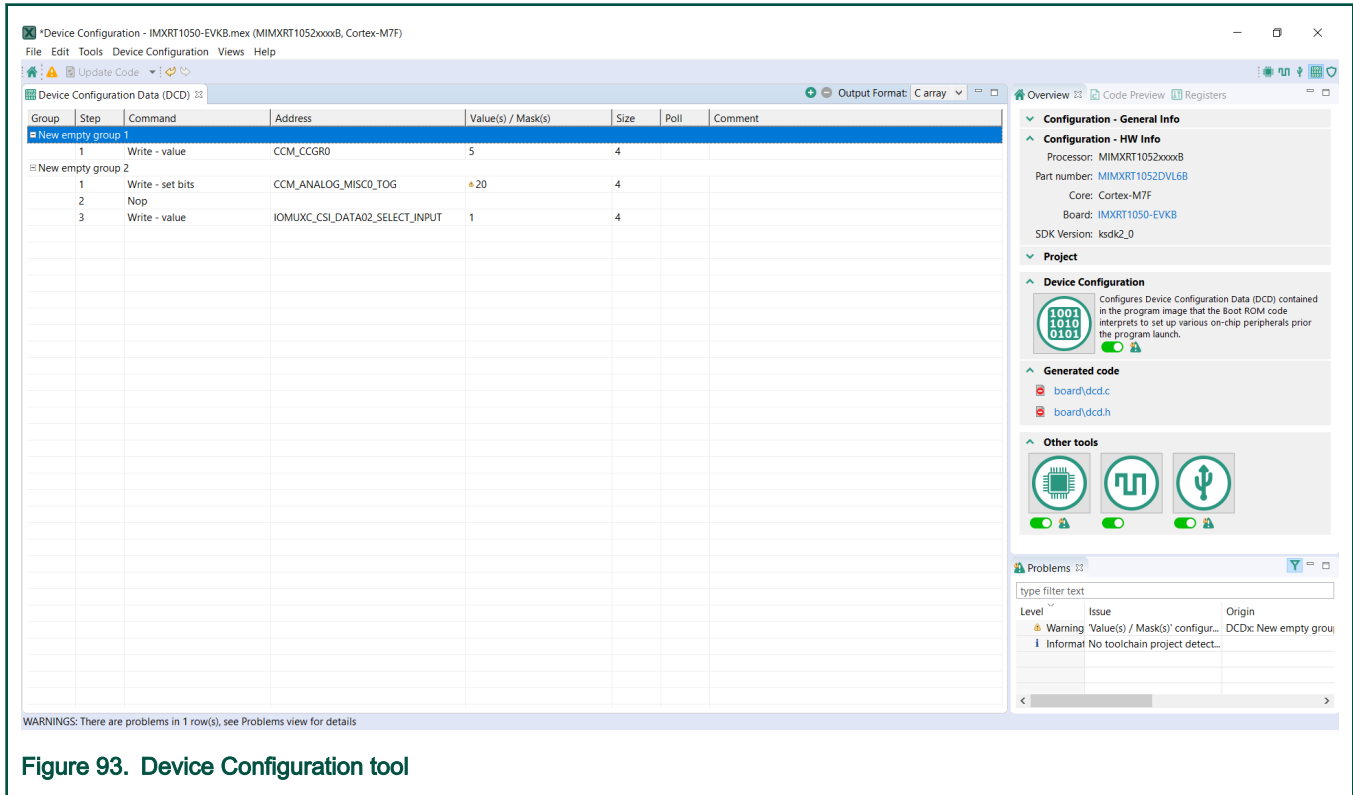


Figure 93. Device Configuration tool

### 6.1 Device Configuration Data (DCD) view

The **Device Configuration Data (DCD)** view displays memory initialization commands of your currently active configuration. Here, you can create new command groups and commands and specify their parameters.

Commands in the **Device Configuration Data (DCD)** can be synchronised from the **SEMC Validation** tool in the **Peripherals** tool.

#### 6.1.1 Device Configuration Data (DCD) view actions

The following is a list of command and command group-relevant actions you can perform in the **Device Configuration Data (DCD)** view:

- **Create a new command group** - Right-click the table and choose **Add Group** from the context menu.
- **Re/Name a command group** - Left-click the command group cell and enter the required name.
- **Disable a command group** - Right-click the command group row and choose **Disable Group** from the context menu.
- **Remove a command group** - Right-click the command group row and choose **Remove Group** from the context menu.
- **Collapse all command groups** - Right-click the the table and choose **Collapse All Groups** from the context menu.
- **Expand all command groups** - Right-click the table and choose **Expand All Groups** from the context menu.

- **Add a command to a group** - Right-click the table and choose **Add Command** from the context menu. Alternatively, click the **Add Command** button in the tool's toolbar.
- **Specify command type** - Left-click the row's **Command** cell and choose from the dropdown menu.
- **Specify register address for a command** - Left-click the row's **Address** cell and choose from the dropdown menu.
- **Specify a value or a mask for a command** - Left-click the row's **Value(s) / Mask(s)** cell to open the mask window. Enter the value into the field and select **OK**. Alternatively, select **Cancel** to cancel the operation, or **Reset** to reset the value.

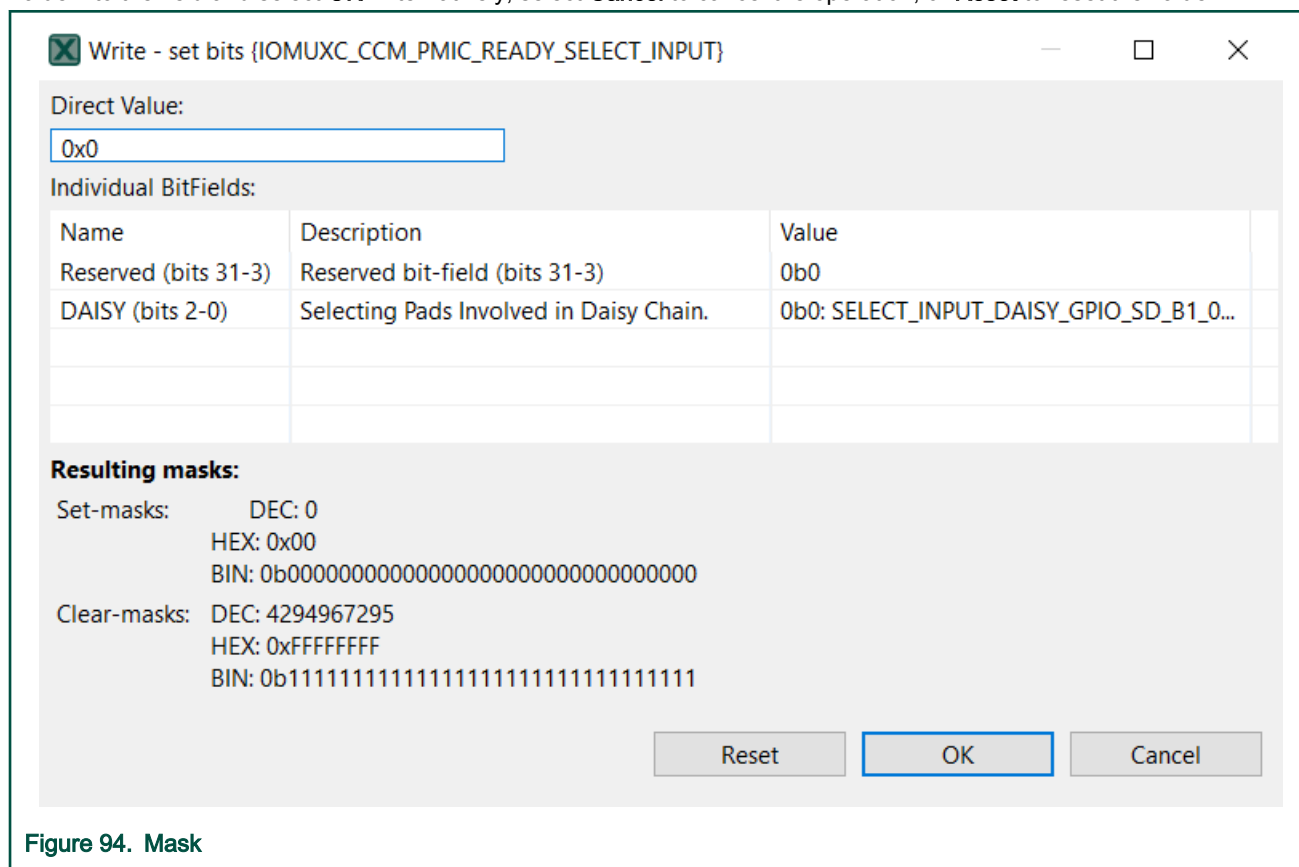


Figure 94. Mask

- **Specify the size of write/read data for a command** - Left-click the row's **Size** cell and choose from the dropdown menu.
- **Specify the number of polls of a command** - Left-click the row's **Poll** cell and enter the required value.
- **Add a comment to a command** - Left-click the row's **Comment** cell.
- **Remove a command** - Right-click the command row and choose **Remove Command** from the context menu. Alternatively, click the **Remove Command** button in the tool's toolbar.
- **Cut a command** - Right-click the command row and choose **Cut** from the context menu.
- **Copy a command** - Right-click the command row and choose **Copy** from the context menu.
- **Paste a command** - Right-click the command row and choose **Paste** from the context menu.

**NOTE**

You can remove all commands by clicking **Device Configuration** in the **Main Menu** and choosing **Clear All Commands** from the dropdown menu.

Basic cell selection shortcuts are applicable.

- **Select additional commands** - Ctrl+Left-click the command row.

## 6.2 Code generation

If the settings are correct and no error is reported, the code generation engine instantly re-generates the source code. You can view the resulting code the **Code Preview** view of the **Device Configuration** tool.

**Code Preview** automatically highlights differences between the current and immediately preceding iteration of the code. You can choose between two modes of highlighting by clicking the **Set viewing style for source differences**. You can also disable highlighting altogether from the same dropdown menu.

**Device Configuration** source code can be generated in a C array (default) or binary format.

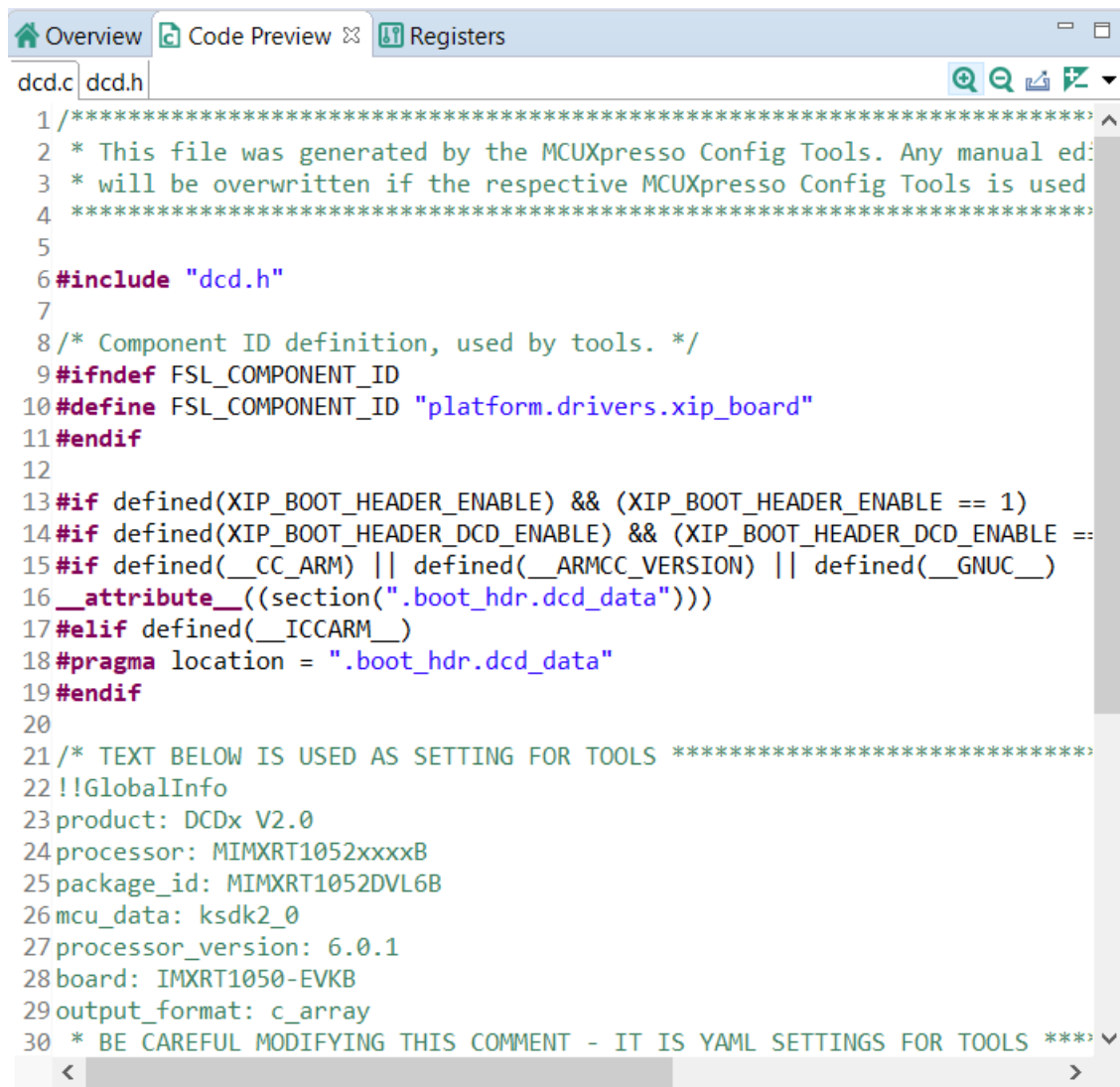
The code in a C array format is generated in two files:

- dcd.c
- dcd.h

The code in a binary format is generated in a single file:

- dcd.bin

To change the code format, choose the required option from the dropdown menu in the **Device Configuration Data (DCD)** view.



```

1 /*****
2  * This file was generated by the MCUXpresso Config Tools. Any manual ed
3  * will be overwritten if the respective MCUXpresso Config Tools is used
4  *****/
5
6 #include "dcd.h"
7
8 /* Component ID definition, used by tools. */
9 #ifndef FSL_COMPONENT_ID
10 #define FSL_COMPONENT_ID "platform.drivers.xip_board"
11 #endif
12
13 #if defined(XIP_BOOT_HEADER_ENABLE) && (XIP_BOOT_HEADER_ENABLE == 1)
14 #if defined(XIP_BOOT_HEADER_DCD_ENABLE) && (XIP_BOOT_HEADER_DCD_ENABLE ==
15 #if defined(__CC_ARM) || defined(__ARMCC_VERSION) || defined(__GNUC__)
16 __attribute__((section(".boot_hdr.dcd_data")))
17 #elif defined(__ICCARM__)
18 #pragma location = ".boot_hdr.dcd_data"
19 #endif
20
21 /* TEXT BELOW IS USED AS SETTING FOR TOOLS *****/
22 !!GlobalInfo
23 product: DCDx V2.0
24 processor: MIMXRT1052xxxxB
25 package_id: MIMXRT1052DVL6B
26 mcu_data: kSDK2_0
27 processor_version: 6.0.1
28 board: IMXRT1050-EVKB
29 output_format: c_array
30 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS ***>

```

Figure 95. Code Preview

# Chapter 7

## Trusted Execution Environment Tool

In the **Trusted Execution Environment**, or **TEE** tool, you can configure security policies of memory areas, bus masters, and peripherals, in order to isolate and safeguard sensitive areas of your application.

You can set security policies of different parts of your application in the **Security Access Configuration** and its sub-views, and review these policies in the **Memory Attribution Map** and **Access Overview** views. Use the **User Memory Regions** view to create a convenient overview of memory regions and their security levels.

You can also view registers handled by the **TEE** tool in the **Registers** view, and inspect the code in the **Code Preview** tool.

### NOTE

In order for your configuration to come into effect, make sure you have enabled the relevant enable secure check option in the **Miscellaneous** sub-view of the **Security Access Configuration** view.

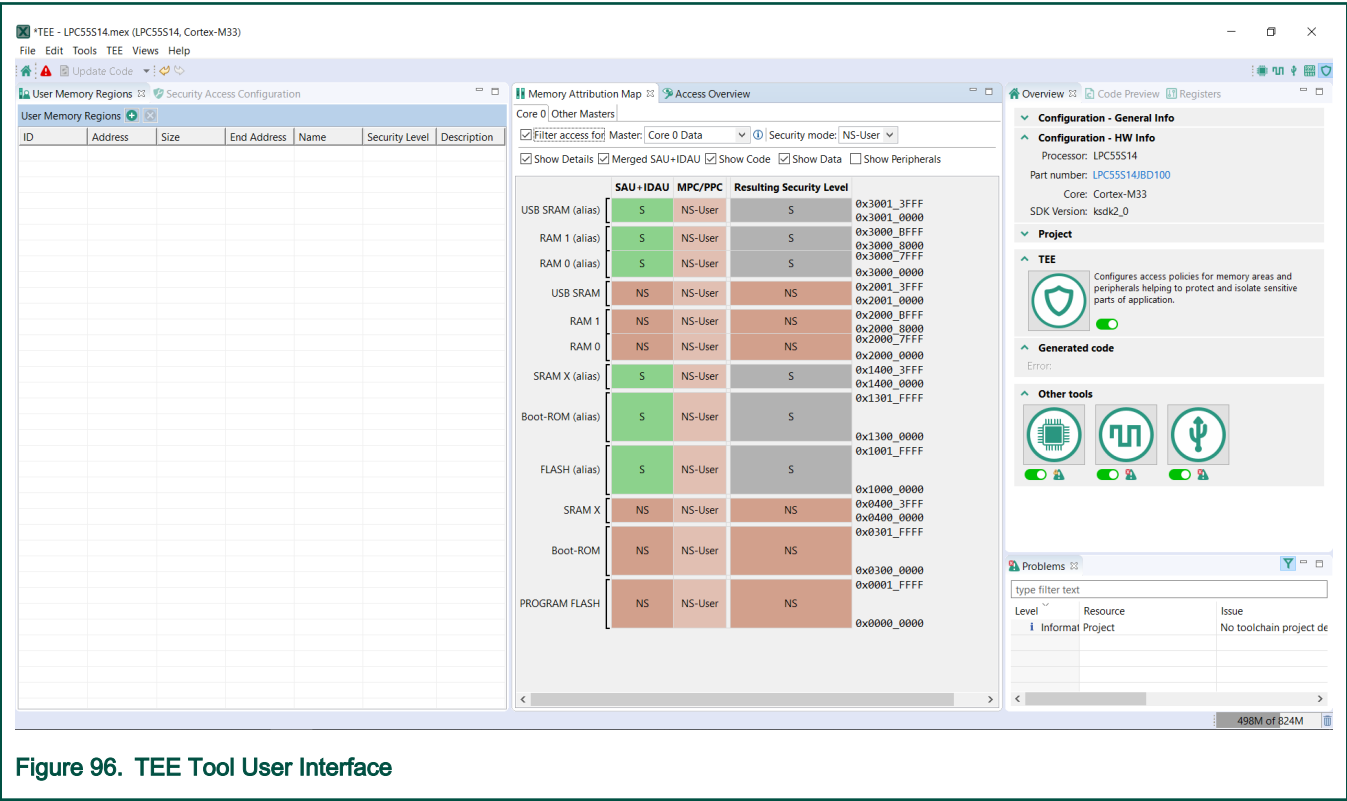


Figure 96. TEE Tool User Interface

### 7.1 User Memory Regions

In the **User Memory Regions** view, you can create and maintain a high-level configuration of memory regions and their security levels. You can create the regions, name them, specify their address, size, security level, and provide them with a description. You can then fix any errors in the settings with the help of the **Problems** view.

Create a new memory region by clicking the **Add new memory region button** in the view's header.

Enter/change the memory region's parameters by clicking the row's cells. In the **Security Level** column, you have these options to choose from:

- **NS-User** - Non-secure user
- **NS-Priv** - Non-secure privileged

- **S-User** - Secure user
- **S-Priv** - Secure privileged
- **NSC-User** - Non-secure callable user
- **NSC-Priv** - Non-secure callable privileged
- **Any**

Remove the memory region by selecting the table row and clicking the **Remove selected memory region(s)** button in the view's header.

[illegible]

### Figure 97. User Memory Regions

[illegible]

### Figure 98. User Memory Regions

You can import memory region configuration from other IDE projects by clicking the **Import memory regions configuration from the IDE project(s)** button in the view toolbar. Select the project you want from the list to import its memory regions settings into your current project.

## NOTE

After the import, you might have to correct some of the security levels manually.



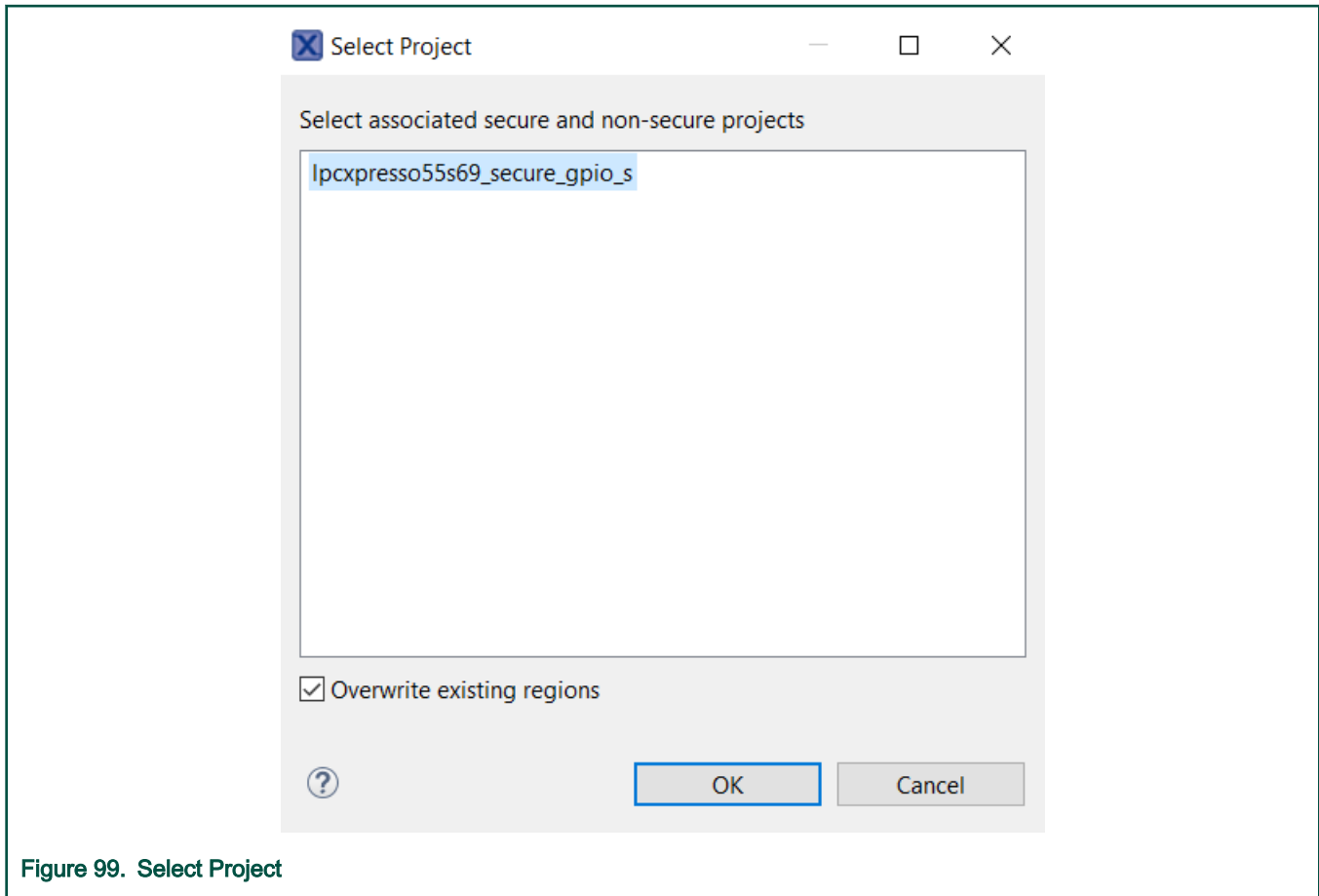


Figure 99. Select Project

## 7.2 Security Access Configuration view

In the **Security Access Configuration** view, you can configure your application's security policies in a number of ways. See the following sections for more details.

### 7.2.1 Masters/Slaves

In the **Masters/Slaves** sub-view, you can configure security levels for bus masters and slaves.

Set the bus master/slave security level by left-clicking the relevant cell in the **Security level** column and choosing from the dropdown list. Alternatively, you can right-click the relevant cell in the **Master** and **Slave** column and choose from the security level from the context menu. To select multiple entries, use the **Ctrl+Left-click** shortcut, then right-click the selected area for the context menu.

You have four security levels to choose from, in ascending order of security:

- **NS-User** - Non-secure user
- **NS-Priv** - Non-secure privileged
- **S-User** - Secure user
- **S-Priv** - Secure privileged

You can further specify the interrelation between master and slave security levels by selecting the following options:

- **Simple Master in Strict Mode** - Select to allow simple bus master to read and write on same level only. De-select to allow to read and write on same and lower level.
- **Smart Master in Strict Mode** - Select to allow smart bus master to execute, read, and write to memory at same level only. De-select to allow to execute on same level only, read and write on same and lower level.

**NOTE**

Instruction-type bus master security level must be equal to bus slave security level. Data and others security level must be equal or higher than bus slave security level.

The screenshot shows the 'Security Access Configuration' window with the 'Masters/Slaves' tab selected. The 'Options' section has two checked items: 'Simple Master in Strict Mode' and 'Smart Master in Strict Mode'. Below this is a table with columns 'Master', 'Security level', 'Slave', and 'Se...vel'.

Master	Security level	Slave	Se...vel
Simple master		ADC0	NS-User
CANFD	NS-User	AHB_SECURE_CTRL	NS-User
DMA0	NS-User	ANACTRL	NS-User
DMA1	NS-User	CAN0	NS-User
HASHCRYPT	NS-User	CASPER	NS-User
USBFS	NS-User	CRC_ENGINE	NS-User
USBFSH	NS-User	CTIMER0	NS-User
		CTIMER1	NS-User
		CTIMER2	NS-User
		CTIMER3	NS-User
		CTIMER4	NS-User
		DBGMAILBOX	NS-User
		DMA0	NS-User
		DMA1	NS-User
		FLASH	NS-User
		FLEXCOMM0	NS-User
		FLEXCOMM1	NS-User
		FLEXCOMM2	NS-User
		FLEXCOMM3	NS-User
		FLEXCOMM4	NS-User
		FLEXCOMM5	NS-User
		FLEXCOMM6	NS-User
		FLEXCOMM7	NS-User
		GINT0	NS-User
		GINT1	NS-User
		GPIO	NS-User
		HASHCRYPT	NS-User
		INPUTMUX	NS-User
		IOCON	NS-User
		MRT0	NS-User
		OSTIMER	NS-User

Figure 100. Masters/Slaves

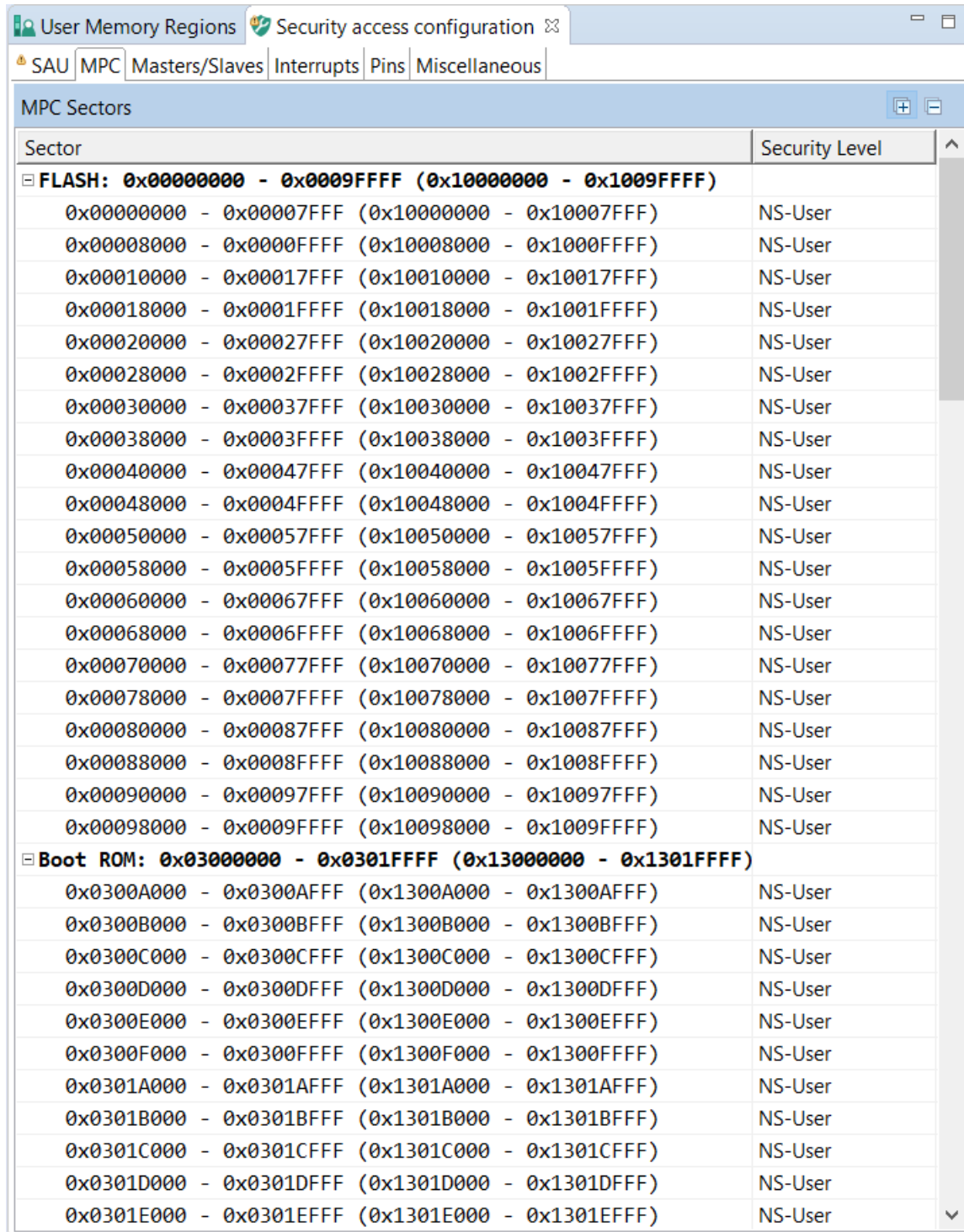
## 7.2.2 MPC

In the **MPC** (Memory Protection Checker) sub-view, you can set security policies on entire memory sectors as defined by physical addresses.

Set the memory sector security level by left-clicking the relevant cell in the **Security level** column and choosing from the dropdown list. Alternatively, you can right-click the relevant cell in the **Sector** column and choose the security level from the context menu. To select multiple entries, use the **Ctrl+Left-click** shortcut, then right-click the selected area for the context menu.

You have four security levels to choose from, in ascending order of security:

- **NS-User** - Non-secure user
- **NS-Priv** - Non-secure privileged
- **S-User** - Secure user
- **S-Priv** - Secure privileged



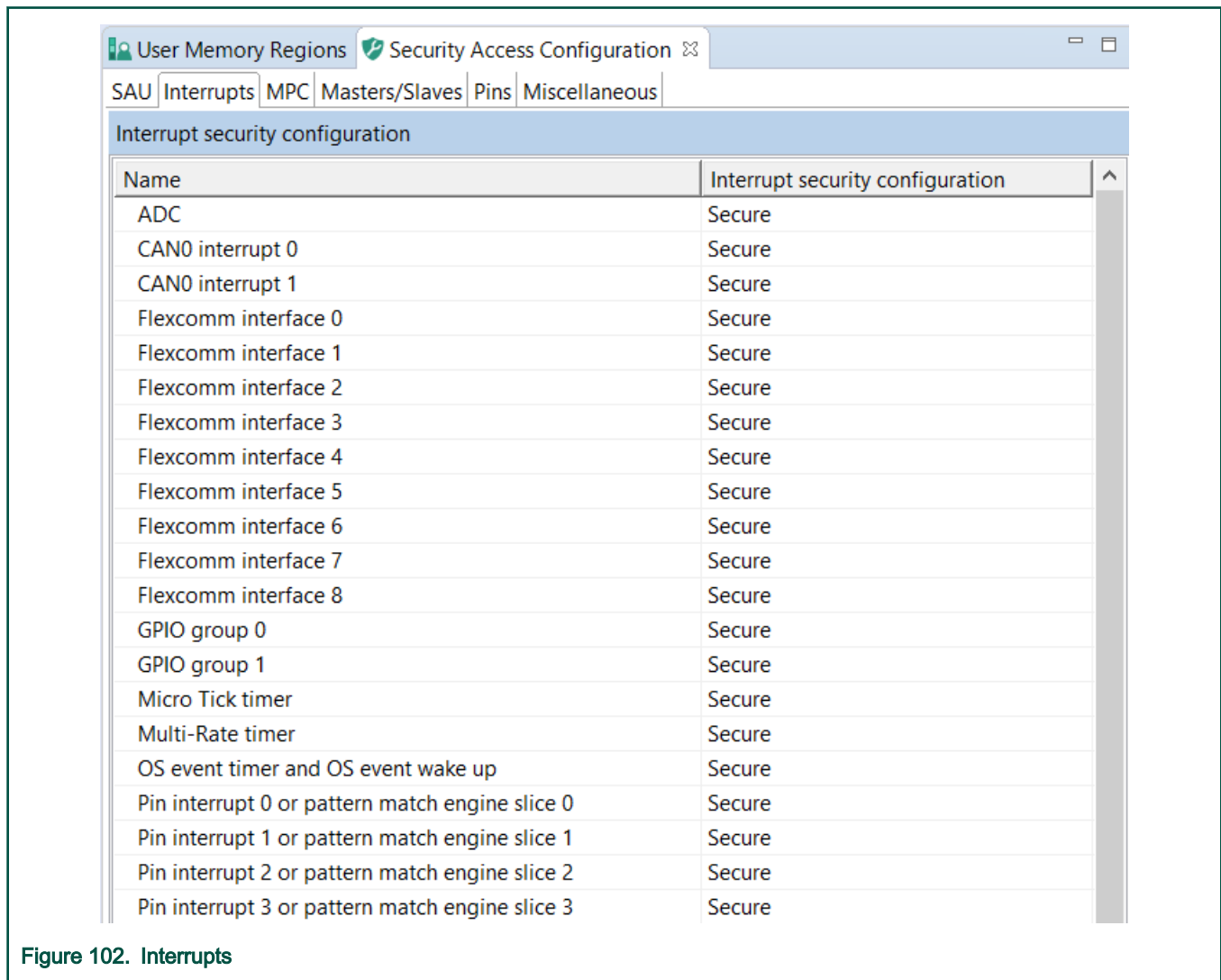
Sector	Security Level
<b>FLASH: 0x00000000 - 0x0009FFFF (0x10000000 - 0x1009FFFF)</b>	
0x00000000 - 0x00007FFF (0x10000000 - 0x10007FFF)	NS-User
0x00008000 - 0x0000FFFF (0x10008000 - 0x1000FFFF)	NS-User
0x00010000 - 0x00017FFF (0x10010000 - 0x10017FFF)	NS-User
0x00018000 - 0x0001FFFF (0x10018000 - 0x1001FFFF)	NS-User
0x00020000 - 0x00027FFF (0x10020000 - 0x10027FFF)	NS-User
0x00028000 - 0x0002FFFF (0x10028000 - 0x1002FFFF)	NS-User
0x00030000 - 0x00037FFF (0x10030000 - 0x10037FFF)	NS-User
0x00038000 - 0x0003FFFF (0x10038000 - 0x1003FFFF)	NS-User
0x00040000 - 0x00047FFF (0x10040000 - 0x10047FFF)	NS-User
0x00048000 - 0x0004FFFF (0x10048000 - 0x1004FFFF)	NS-User
0x00050000 - 0x00057FFF (0x10050000 - 0x10057FFF)	NS-User
0x00058000 - 0x0005FFFF (0x10058000 - 0x1005FFFF)	NS-User
0x00060000 - 0x00067FFF (0x10060000 - 0x10067FFF)	NS-User
0x00068000 - 0x0006FFFF (0x10068000 - 0x1006FFFF)	NS-User
0x00070000 - 0x00077FFF (0x10070000 - 0x10077FFF)	NS-User
0x00078000 - 0x0007FFFF (0x10078000 - 0x1007FFFF)	NS-User
0x00080000 - 0x00087FFF (0x10080000 - 0x10087FFF)	NS-User
0x00088000 - 0x0008FFFF (0x10088000 - 0x1008FFFF)	NS-User
0x00090000 - 0x00097FFF (0x10090000 - 0x10097FFF)	NS-User
0x00098000 - 0x0009FFFF (0x10098000 - 0x1009FFFF)	NS-User
<b>Boot ROM: 0x03000000 - 0x0301FFFF (0x13000000 - 0x1301FFFF)</b>	
0x0300A000 - 0x0300AFFF (0x1300A000 - 0x1300AFFF)	NS-User
0x0300B000 - 0x0300BFFF (0x1300B000 - 0x1300BFFF)	NS-User
0x0300C000 - 0x0300CFFF (0x1300C000 - 0x1300CFFF)	NS-User
0x0300D000 - 0x0300DFFF (0x1300D000 - 0x1300DFFF)	NS-User
0x0300E000 - 0x0300EFFF (0x1300E000 - 0x1300EFFF)	NS-User
0x0300F000 - 0x0300FFFF (0x1300F000 - 0x1300FFFF)	NS-User
0x0301A000 - 0x0301AFFF (0x1301A000 - 0x1301AFFF)	NS-User
0x0301B000 - 0x0301BFFF (0x1301B000 - 0x1301BFFF)	NS-User
0x0301C000 - 0x0301CFFF (0x1301C000 - 0x1301CFFF)	NS-User
0x0301D000 - 0x0301DFFF (0x1301D000 - 0x1301DFFF)	NS-User
0x0301E000 - 0x0301EFFF (0x1301E000 - 0x1301EFFF)	NS-User

Figure 101. MPC

### 7.2.3 Interrupts

In the **Interrupts** sub-view, you can set security designation for device's peripheral interrupts. In case that the processor contains more than a single core or processing unit, additional **Handling by Core** tables might appear. In these tables, you can specify if the interrupts coming from the peripheral can be handled by the core or processing unit.

All interrupts are set to **Secure** by default. If you want to change the interrupt source's security designation, left-click the **Secure** cell of the interrupt and choose from the dropdown menu. Alternatively, right-click the interrupt's **Name** cell and choose the security designation from the context menu. To select multiple entries, use the **Ctrl+Left-click** shortcut, then right-click the selected area for the context menu.



## 7.2.4 Pins

In the **Pins** sub-view, you can specify if the reading GPIO state is allowed or denied.

All pins' reading GPIO state is set to **Allow** by default. If you want to change the pins reading GPIO state, left-click the **Reading GPIO state** cell of the pin and choose from the dropdown menu. Alternatively, right-click the pin's **Name** cell and choose the reading GPIO state from the context menu. To select multiple entries, use the **Ctrl+Left-click** shortcut, then right-click the selected area for the context menu.

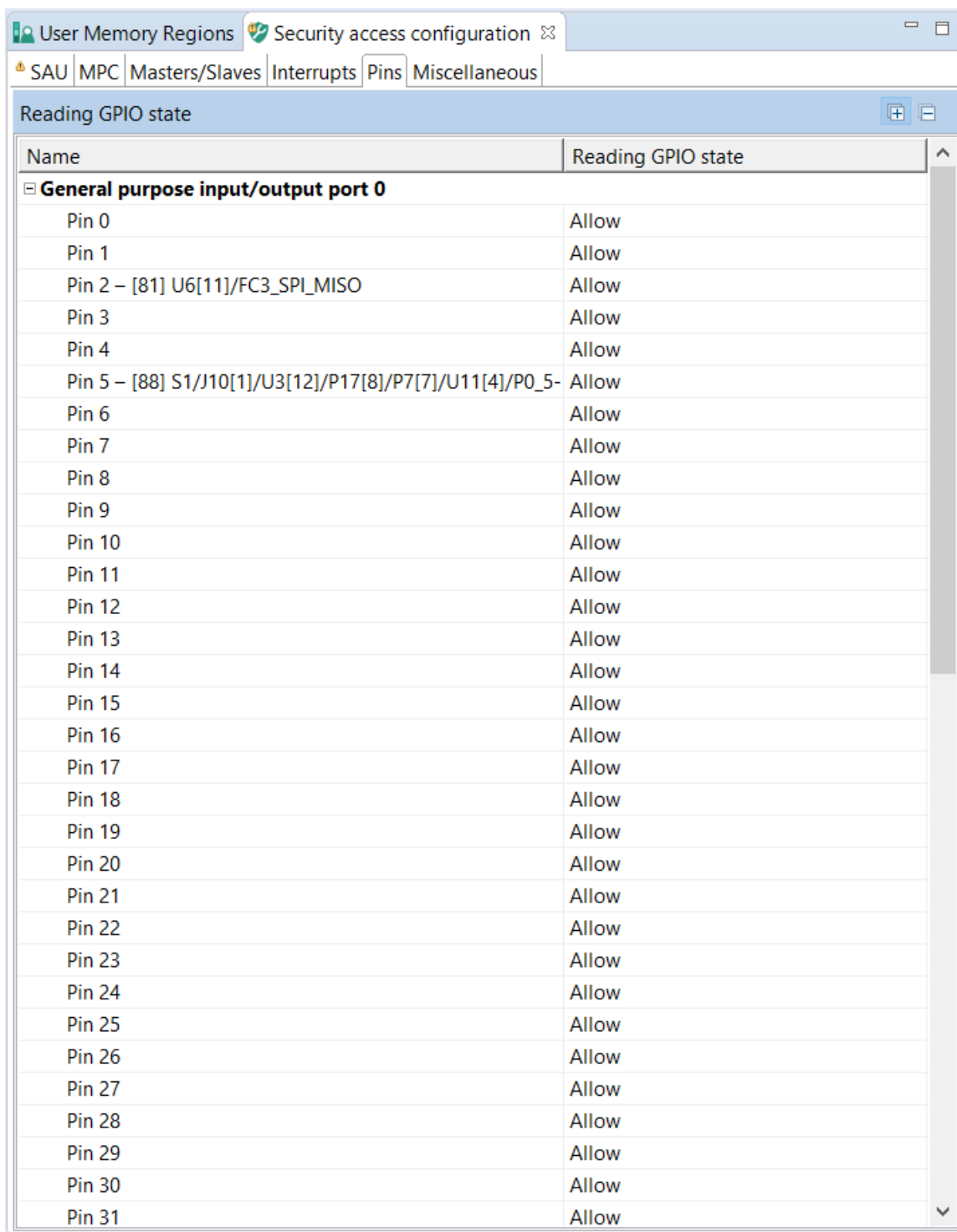


Figure 103. Pins

### 7.2.5 SAU

In the **SAU** sub-view, you can enable and configure SAU (Security attribution unit).

When enabled, you can set up SAU memory regions, specify their start and size or end address, and specify their security level. SAU automatically sets the entire memory space to a secure severity level when disabled. It also sets the entire memory space to a secure security level when enabled but without set memory regions.

You can choose between two security levels:

- **NS** - Non-secure
- **NSC** - Non-secure callable

Alternatively, you can set all the SAU memory regions to non-secure security level by selecting the **All Non-Secure**.

#### NOTE

This option is only available when SAU is disabled.

You can also decide to generate code even for disabled memory regions by selecting the option **Generate sources for disabled regions**.

The screenshot shows the 'Security Access Configuration' window with the 'SAU' tab selected. Under 'Options', the 'Enable SAU' checkbox is checked. Below this is a table titled 'SAU Memory Regions' with 6 columns: Index, Enable, Address, Size, End Address, and Security Level. The table contains 8 rows, with the first two rows (Index 0 and 1) having their 'Enable' checkboxes checked. The 'Address' and 'End Address' for Index 0 are 0x0000\_0000 and 0x0FFF\_FFFF respectively, and for Index 1 they are 0x2000\_0000 and 0xEFFF\_FFFF. All other rows have 'Enable' unchecked and identical 'Address' and 'End Address' values of 0x0000\_0000 and 0x0000\_001F. All 'Security Level' entries are 'NS'.

Index	Enable	Address	Size	End Address	Security Level
0	<input checked="" type="checkbox"/>	0x0000_0000	0x1000_0000	0x0FFF_FFFF	NS
1	<input checked="" type="checkbox"/>	0x2000_0000	0xD000_0000	0xEFFF_FFFF	NS
2	<input type="checkbox"/>	0x0000_0000	0x20	0x0000_001F	NS
3	<input type="checkbox"/>	0x0000_0000	0x20	0x0000_001F	NS
4	<input type="checkbox"/>	0x0000_0000	0x20	0x0000_001F	NS
5	<input type="checkbox"/>	0x0000_0000	0x20	0x0000_001F	NS
6	<input type="checkbox"/>	0x0000_0000	0x20	0x0000_001F	NS
7	<input type="checkbox"/>	0x0000_0000	0x20	0x0000_001F	NS

Figure 104. SAU/IDAU

## 7.2.6 Miscellaneous

In the **Miscellaneous** sub-view, you can set various configuration options. The list of these options depends on processor data, and varies greatly. All the options influence your register settings, and can be inspected in the **Register** view. Only some of the options directly influence configuration you have made in the **Security Access Configuration** view. Point your cursor over individual options to display a tooltip explaining the function of each option.

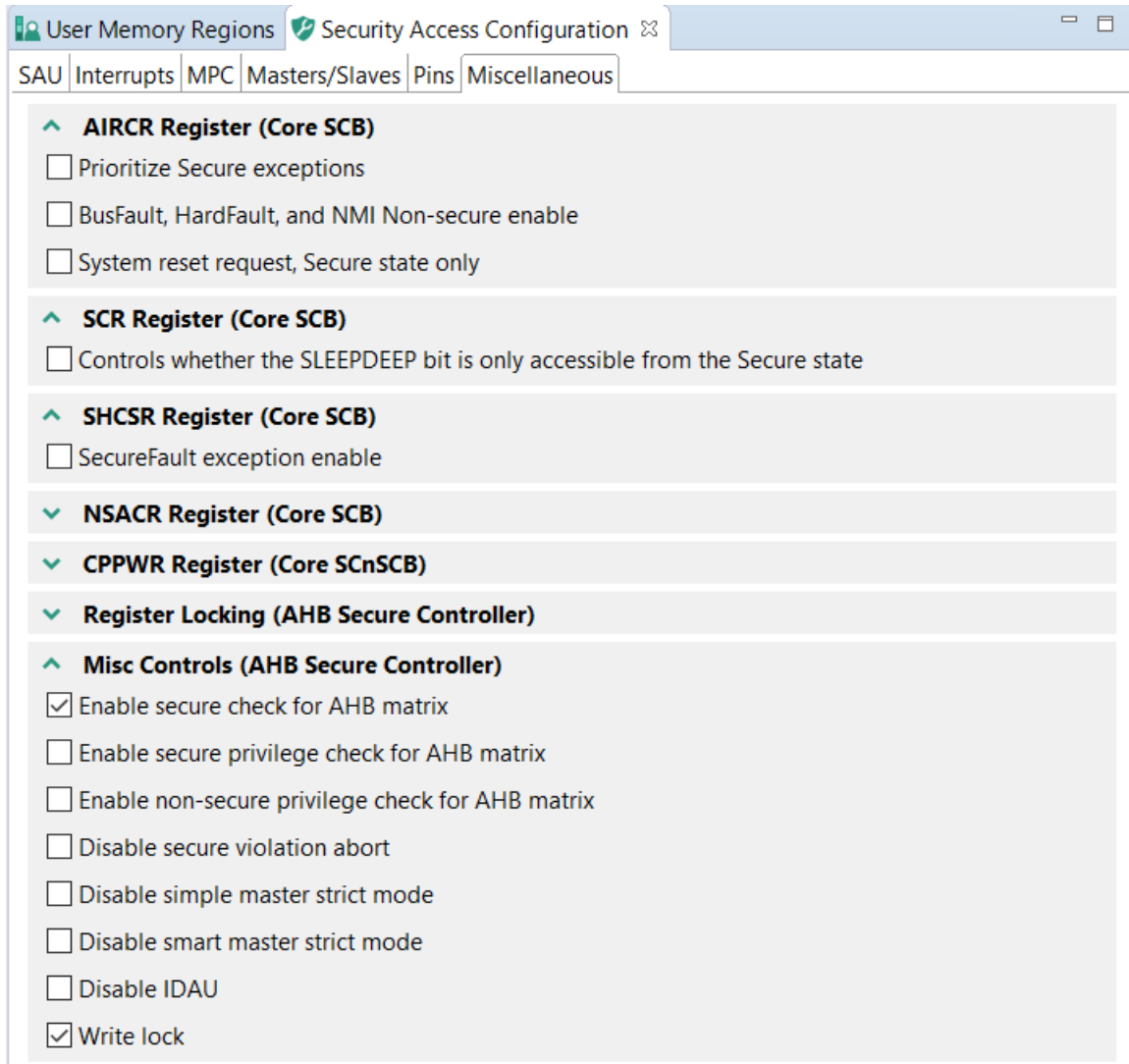


Figure 105. Miscellaneous

## 7.3 Memory attribution map

In the **Memory attribution map**, you can view security levels set for memory regions. This view is read-only.

### 7.3.1 Core 0

In the **Core 0** sub-view, you can review security levels set for Core 0 to the code, data, and peripherals memory regions. The table is read-only.

The **Access by Master** table displays **MSW** or **SAU+IDAU**, **MPC** (Memory Protection Checker) security level, and **Resulting Security Level** status of listed code, data, and peripherals memory regions, alongside their physical addresses.

To set the display options, do the following:

1. Click the **Filter access for** checkbox to enable filtering options.
2. Select the master security access you want to review by choosing from the **Master** dropdown menu.
3. Optionally, set the security level of the selected master by choosing from the **Security mode** dropdown menu. This setting has no effect on the configuration.



- Optionally, customize the output by de-selecting the **Show details** and **Merged SAU+IDAU** options.
- Optionally, filter displayed memory regions in the **Filter** area.

Point your cursor over the color-coded cells to display a tooltip with information about the security level combination.

Double-click the cell to open the pertinent settings in **Security Access Configuration**.

	SAU+IDAU	MPC/PPC	Resulting Security Level	
USB SRAM (alias)	S	NS-User	S	0x3001_3FFF 0x3001_0000
RAM 1 (alias)	S	NS-User	S	0x3000_BFFF 0x3000_8000
RAM 0 (alias)	S	NS-User	S	0x3000_7FFF 0x3000_0000
USB SRAM	NS	NS-User	NS	0x2001_3FFF 0x2001_0000
RAM 1	NS	NS-User	NS	0x2000_BFFF 0x2000_8000
RAM 0	NS	NS-User	NS	0x2000_7FFF 0x2000_0000
SRAM X (alias)	S	NS-User	S	0x1400_3FFF 0x1400_0000
Boot-ROM (alias)	S	NS-User	S	0x1301_FFFF 0x1300_0000
FLASH (alias)	S	NS-User	S	0x1001_FFFF 0x1000_0000
SRAM X	NS	NS-User	NS	0x0400_3FFF 0x0400_0000
Boot-ROM	NS	NS-User	NS	0x0301_FFFF 0x0300_0000
PROGRAM FLASH	NS	NS-User	NS	0x0001_FFFF 0x0000_0000

Figure 106. Core 0

### 7.3.2 Other masters

In the **Other Masters** sub-view, you can review security attributes of memory in relation to access rights by master other than Core 0. The table is read-only.

To set the display options, do the following:

1. Click the **Filter access for** checkbox to enable filtering options.
2. Select the master type security access you want to review by choosing from the **Master** dropdown menu.
3. Optionally, customize the output by de-selecting the **Show Details**, **Show Code**, **Show Data**, and **Show Peripherals** options.
4. Optionally, filter displayed memory regions in the **Filter** area.

Point your cursor over the color-coded fields to display a tooltip with information about the security level combination.

Double-click the cell to open the pertinent settings in **Security Access Configuration**.

Memory Attribution Map Access Overview

Core 0 Other Masters

☐ Filter access for:

☒ Show Details ☒ Merged SAU+IDAU ☒ Show Code ☒ Show Data ☐ Show Peripherals

	MSW	MPC/PPC	Resulting Security Level	
USB SRAM (alias)	S	NS-User	S	0x3001_3FFF 0x3001_0000
RAM 1 (alias)	S	NS-User	S	0x3000_BFFF 0x3000_8000
RAM 0 (alias)	S	NS-User	S	0x3000_7FFF 0x3000_0000
USB SRAM	NS	NS-User	NS	0x2001_3FFF 0x2001_0000
RAM 1	NS	NS-User	NS	0x2000_BFFF 0x2000_8000
RAM 0	NS	NS-User	NS	0x2000_7FFF 0x2000_0000
SRAM X (alias)	S	NS-User	S	0x1400_3FFF 0x1400_0000
Boot-ROM (alias)	S	NS-User	S	0x1301_FFFF 0x1300_0000
FLASH (alias)	S	NS-User	S	0x1001_FFFF 0x1000_0000
SRAM X	NS	NS-User	NS	0x0400_3FFF 0x0400_0000
Boot-ROM	NS	NS-User	NS	0x0301_FFFF 0x0300_0000
PROGRAM FLASH	NS	NS-User	NS	0x0001_FFFF 0x0000_0000

Figure 107. Other masters

## 7.4 Access Overview

In **Access Overview**, you can review security policies you have set in **Security Access Configuration** view.

The horizontal axis displays all masters, divided into color-coded groups by their security settings.

The vertical axis displays memory ranges and slave buses/peripherals.

Point your cursor at an entry to display a tooltip with information about the entry.

You can group the displayed information by security or by masters by using the button on the right-hand side of the toolbar.



Figure 108. Access Overview

## 7.5 Code Generation

If the settings are correct and no error is reported, the code generation engine instantly re-generates the source code. You can view the resulting code the **Code Preview** view of the **Trusted Execution Environment** tool.

**Code Preview** automatically highlights differences between the current and immediately preceding iteration of the code. You can choose between two modes of highlighting by clicking the **Set viewing style for source differences**. You can also disable highlighting altogether from the same dropdown menu.

Some devices support ROM preset as well as C code. You can choose to have the code generated in the ROM preset by selecting the option in the **Miscellaneous** sub-view.

# Chapter 8 Advanced Features

## 8.1 Switching the processor

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

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

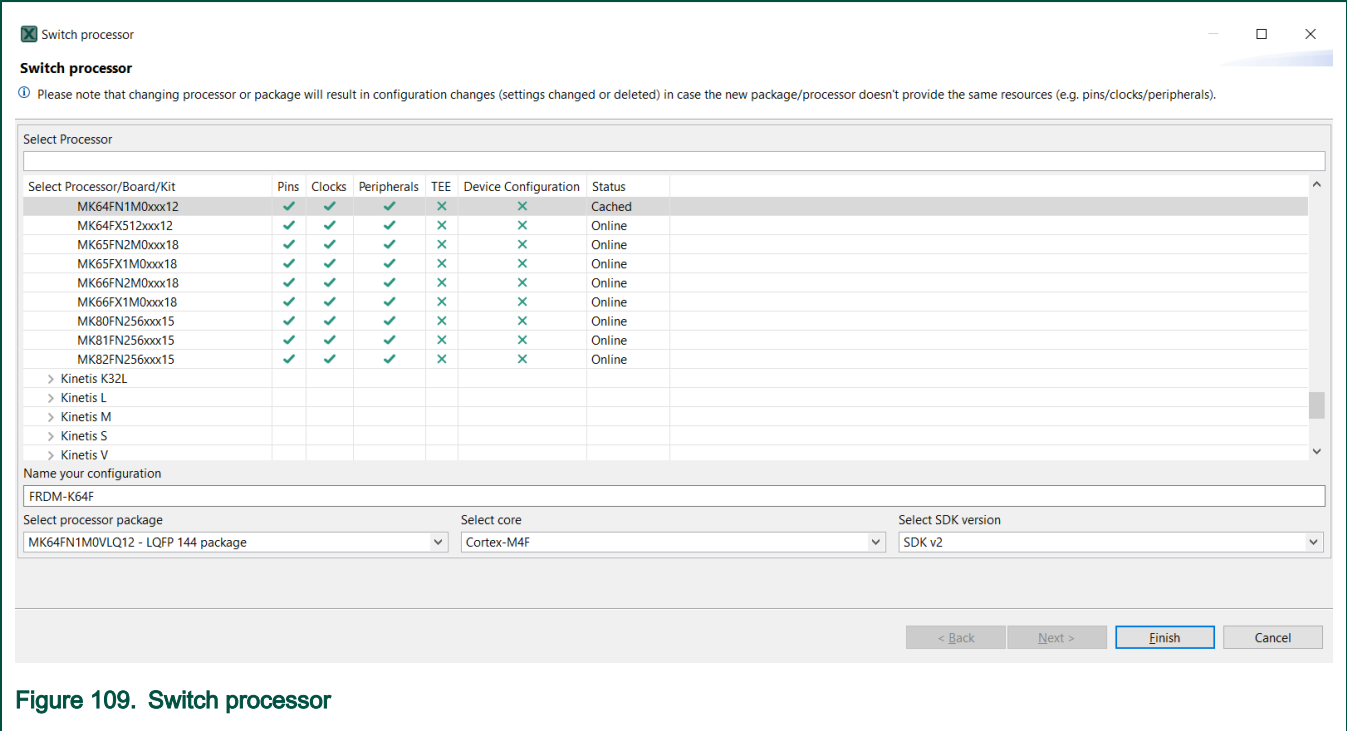


Figure 109. Switch processor

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

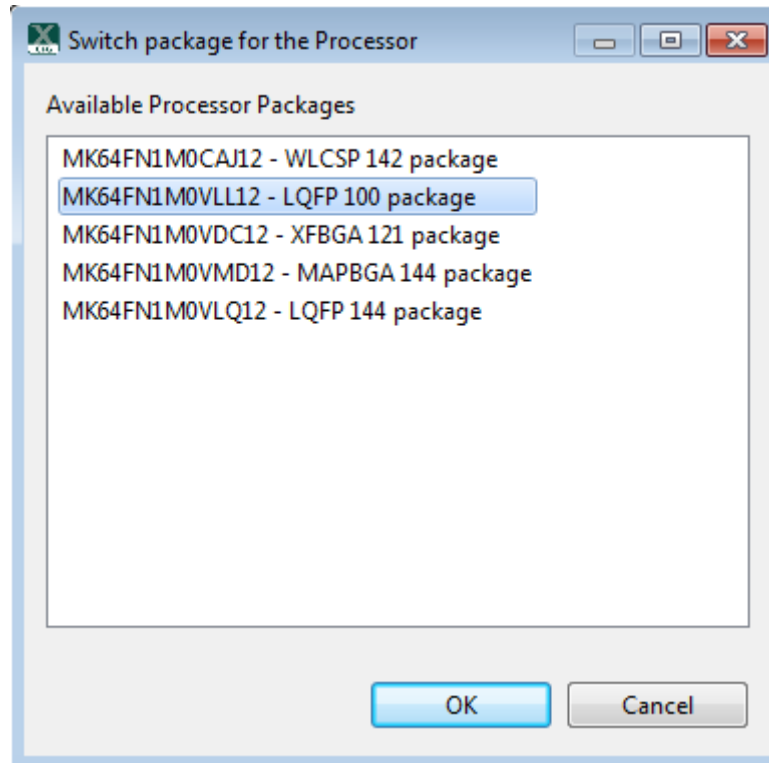


Figure 110. Switch package

## 8.2 Exporting the Pins table

To export the Pins table, do the following:

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

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

Figure 111. 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);      :PTE0/CLKOUT32K(GPIOE,GPIO,0);      :ADC1_SE4a(ADC1,SEa,4)
B1 :PTE1/LLWU_P0                            :PTE1/LLWU_P0(GPIOE,GPIO,1)         :PTE1/LLWU_P0(GPIOE,GPIO,1)         :ADC1_SE5a(ADC1,SEa,5)
C1 :PTD5                                    :PTD5(GPIOD,GPIO,5)                 :PTD5(GPIOD,GPIO,5)                 :ADC1_SE6b(ADC0,SEb,6)
D1 :USBD_DM                                 :                                  :                                  :
E1 :USBD_DP                                 :                                  :                                  :
F1 :ADCO_DM0/ADC1_DM3                       :                                  :                                  :ADC0_DM0/ADC1_DM3(ADC0,DM,0)/ADC0_DM0/ADC
G1 :ADCO_DP0/ADC1_DP3                       :                                  :                                  :ADC0_DP0/ADC1_DP3(ADC0,DP,0)/ADC0_DP0/ADC
H1 :VREF_OUT/CHP1_IN5/CHP0_IN5/ADC1_SE18    :                                  :                                  :VREF_OUT/CHP1_IN5/CHP0_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 :ADCO_SE7b/PTD6/LLWU_P15/SPI0_FCS3/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,FLT,0) :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 :ADCI_DM0/ADCO_DM3                       :                                  :                                  :ADC1_DM0/ADCO_DM3(ADC1,DM,0)/ADC1_DM0/ADC
G2 :ADCI_DP0/ADCO_DP3                       :                                  :                                  :ADC1_DP0/ADCO_DP3(ADC1,DP,0)/ADC1_DP0/ADC
H2 :DAC0_OUT/CHP1_IN3/ADCO_SE23             :                                  :                                  :DAC0_OUT/CHP1_IN3/ADCO_SE23(ADC0,SE,23)
A3 :PTD4/LLWU_P14/SPI0_FCS1/UART0_RTS_b/FTM0_CH4/ENM_IN/SPI1_FCS0 :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)         :FTM0_CH5(FTM0,CH,5)                 :
D3 :PTA0/UART0_CTS_b/FTM0_CH5/JTAG_TCLK/SWD_CLK/EZP_CLK :PTA0(GPIOA,GPIO,0)                 :                                  :
E3 :VSS80                                    :                                  :                                  :
F3 :VSSA                                    :                                  :                                  :VSSA(ADC0,SE,30)/VSSA(ADC1,SE,30)/VSSA(AI
G3 :VREFL                                    :                                  :                                  :VREFL(ADC0,SE,30)/VREFL(ADC1,SE,30)/VREFL
H3 :XTAL32                                  :                                  :                                  :
A4 :ADCO_SE5b/PTD1/SPI0_SCK/UART2_CTS_b/LPUART0_CTS_b :PTD1(GPIOD,GPIO,1)                 :                                  :ADC0_SE5b(ADC0,SEb,5)
B4 :ADCI_SE6b/PTC10/I2C1_SCL/I2S0_RX_FS     :PTC10(GPIOC,GPIO,10)               :                                  :ADC1_SE6b(ADC1,SEb,6)
C4 :VSS9                                    :                                  :                                  :
D4 :FTAL/UART0_RX/FTM0_CH6/JTAG_TDI/EZP_DI  :FTAL(GPIOA,GPIO,1)                 :FTM0_CH6(FTM0,CH,6)                 :
E4 :VDD81                                    :                                  :                                  :
F4 :VDDA                                    :                                  :                                  :VDDA(ADC0,SE,29)/VDDA(ADC1,SE,29)/VDDA(AI
G4 :VREFH                                    :                                  :                                  :VREFH(ADC0,SE,29)/VREFH(ADC1,SE,29)/VREFH
```

Figure 112. Aligning to block

## 8.3 Tools advanced configuration

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

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

## 8.4 Generating HTML report

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

## 8.5 Exporting sources

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

To launch the Export wizard:

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



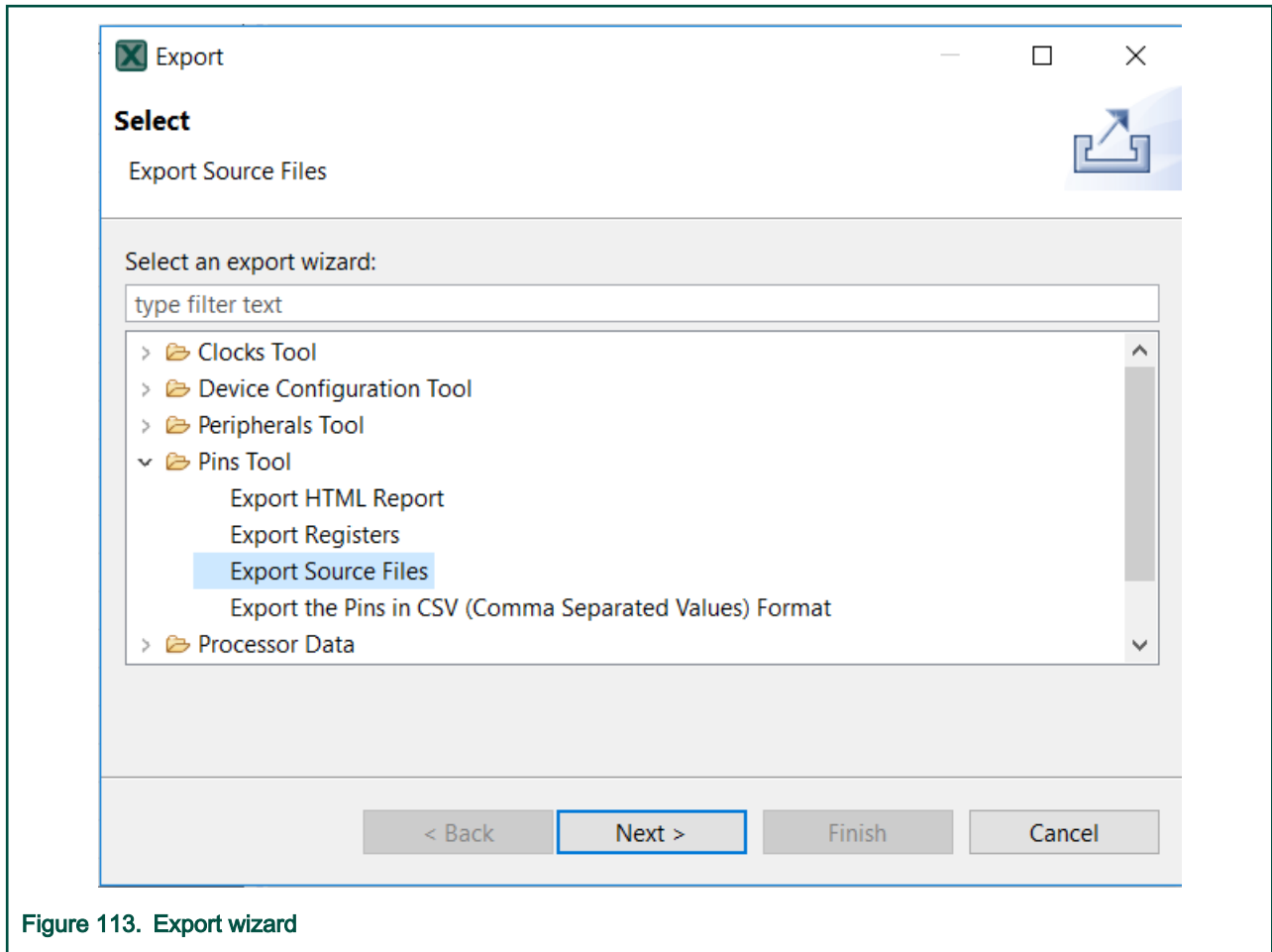


Figure 113. Export wizard

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

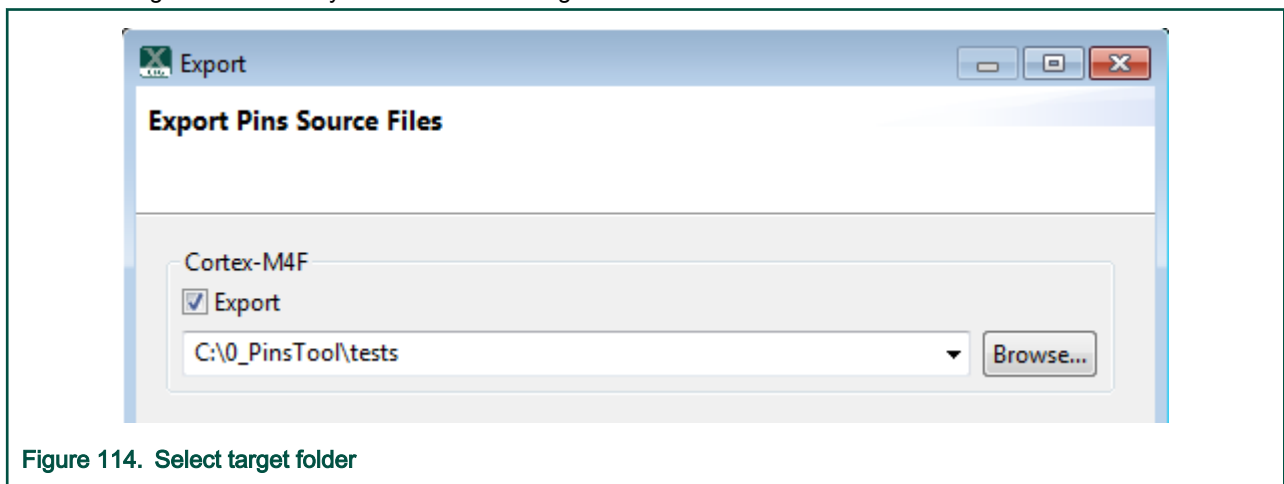


Figure 114. Select target folder

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

## 8.6 Exporting registers

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

To export registers, follow these steps:

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

## 8.7 Command line execution

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

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

Notes regarding command line execution:

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

Command example:

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

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

**Table 8. Commands supported in the framework**

Command name	Definition and parameters	Description	Restriction	Example
--------------	---------------------------	-------------	-------------	---------

*Table continues on the next page...*

**Table 8. Commands supported in the framework (continued)**

Force language	-nl {lang}	Force set language {lang} is in <a href="#">ISO-639-1</a> standard	Removal of the '.npx' 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
Create new configuration by importing toolchain project	-ImportProject {path}	Creates new configuration by importing toolchain project	Requires -HeadlessTool command	-HeadlessTool Pins -ImportProject c:\test\myproject

*Table continues on the next page...*

**Table 8. Commands supported in the framework (continued)**

		Parameter is path to the root of the toolchain project		
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

### 8.7.1 Command line execution - Pins Tool

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

**Table 9. Commands supported in Pins**

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

*Table continues on the next page...*

**Table 9. Commands supported in Pins (continued)**

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

## 8.7.2 Command line execution - Clocks Tool

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

**Table 10. Commands supported in Clocks**

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

*Table continues on the next page...*

**Table 10. Commands supported in Clocks (continued)**

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

### 8.7.3 Command line execution - Peripherals Tool

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

**Table 11. Commands supported in Peripherals Tool**

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

### 8.7.4 Command line execution - Project Cloner

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

**Table 12. Commands supported in Project Cloner**

Command name	Definition and parameters	Description	Restriction	Example
Specify SDK path	-SDKpath {path}	Specify absolute path to the root directory of the SDK package	@since v3.0	-SDKpath c:\nxp\SDK_2.0_MKL43Z256xxx4
Clone SDK example project	-PG_clone {board} {example} {toolchain} {wrkspc} {prjName}	<p>Clones specified SDK example project under new name</p> <ol style="list-style-type: none"> <li>1. {board} - subdirectory of the board in SDK package</li> <li>2. {example} - relative path from board sub-dir and name of the example, for example demo_apps/hello_world; use '/' as a path separator</li> <li>3. {toolchain} - id of the toolchain to create project (see toolchains - toolchain - id)</li> <li>4. {wrkspc} - absolute path where new project shall be created, e.g. projects workspace</li> <li>5. {prjName} - name of the new project</li> </ol>	<p>Requires - HeadlessTool PrjCloner and -SDKpath {path}</p> <p>@since v3.0</p>	-HeadlessTool PrjCloner -SDKpath c:\nxp\SDK_2.0_MKL43Z256xxx4 -PG_clone twrk64f120m demo_apps/hello kds c:\tmp exmpl

## 8.8 Managing data and working offline

You can download, import, and export processor data with the **Data Manager**. This feature is especially useful if you want to make the best out of the tools while staying offline.

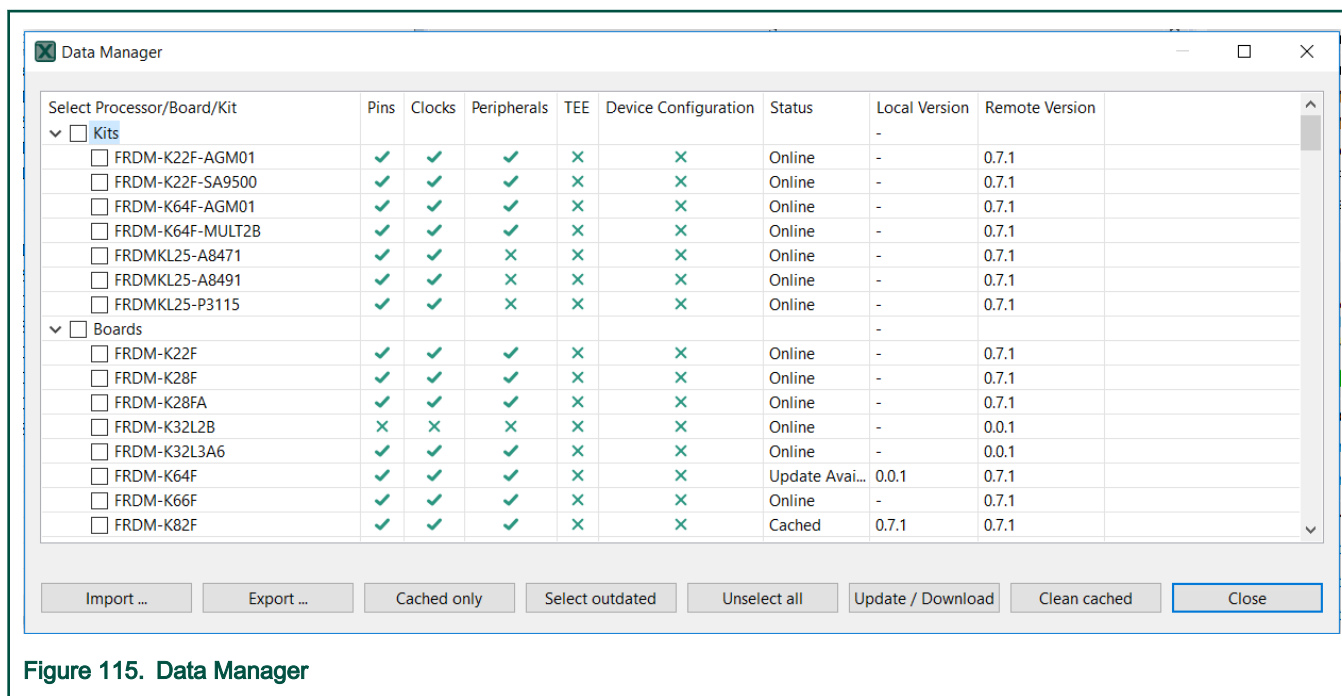


Figure 115. Data Manager

### 8.8.1 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.

### 8.8.2 Downloading data

You can download required processor data with **Data Manager**.

#### NOTE

By default, the data is downloaded and cached automatically during the **Creating a new standalone configuration for processor, board or kit process**.

To download processor data, do the following:

#### NOTE

Internet connection is required for data download.

1. Select **Config Tools >Data Manager** from the **Main Menu**.
2. In the **Data Manager** select the processor/board/kit you want to work with from the list.
3. Click **Update / Download** and confirm.

The data is now downloaded on your local computer, as shown by the **Cached** status in **Data Manager**.

### 8.8.3 Exporting data

You can export downloaded processor data with **Data Manager** in a ZIP format.

To export data, do the following:

1. Select **Config Tools >Data Manager** from the **Main Menu**.
2. In the **Data Manager**, click **Export**.
3. In the **Export Processor Data** window, select the processor data you want to export.



4. Click **Browse** to specify the location and name of the resulting ZIP file.
5. Click **Finish**,

Data is now saved on your local computer in a ZIP format. You can physically (for example, with an USB stick) move it to an offline computer.

---

**NOTE**

You can also export downloaded data by selecting **File>Export>Processor Data>Export Processor Data** from the **Main Menu**.

---

## 8.8.4 Importing data

You can import processor data from another computer with **Data Manager**, provided this data is available locally.

To import data, do the following:

1. Select **Config Tools >Data Manager** from the **Main Menu**.
2. In the **Data Manager**, select **Import**.
3. In the **Import Processor Data** window, specify the location of the ZIP file you want to import.
4. Choose the data to import by selecting the checkbox.
5. Click **Finish**.

The data is now imported to your offline computer, as shown by the **Cached** status in **Data Manager**. You can now work with the data by selecting **New...>Create new standalone configuration for processor, board or kit**.

---

**NOTE**

You can also import data by selecting **File>Import>MCUXpresso Config Tools>Import Processor Data** from the **Main Menu**

---

## 8.8.5 Updating data

You can keep cached data up to date with the **Data Manager**.

---

**NOTE**

If you select the relevant option in **Window > Preferences > MXUXpresso Config Tools** in the **Main Menu**, data will be updated automatically or after a prompt.

---

---

**NOTE**

Internet connection is required for data update.

---

To update cached data, do the following:

1. Select **Config Tools >Data Manager** from the **Main Menu**.
2. In the **Data Manager**, filter outdated data by clicking **Select outdated**.
3. Click **Update / Download** and confirm.

You can always check versions of your data by clicking **Cached only** and comparing version information in the **Local Version** and **Remote Version** columns.

You can clean all cached data by selecting **Clean cached**. This will remove all processor, board, kit, and component data, as well as SDK info files from your computer.

---

**NOTE**

Doesn't affect user templates.

---

## Chapter 9

# 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](http://nxp.com)

##### **Web Support:**

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

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

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

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro,  $\mu$ Vision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2016-2020.

All rights reserved.

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

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

Date of release: 6/2020

Document identifier: MCUXIDECTUG

**arm**