MCUXIDECTUG

MCUXpresso Config Tools User's Guide (IDE)

Rev. 18.0 — 17 September 2026 User guide

Document information

Information	Content
Keywords	MCUXpresso Config Tools
Abstract	MCUXpresso Configuration Tools set is a suite of evaluation and configuration tools that help you from initial evaluation to production software development.

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:

1.1 Table MCUXpresso Config Tools

Table 1. MCUXpresso Config Tools

Name	Description
Pins Tool	Enables you to configure the pins of a device. Pins tool enables you to create, inspect, change, and modify any aspect of the pin configuration and muxing of the device.
Clocks Tool	Enables you to configure initialization of the system clock (core, system, bus, and peripheral clocks) and generates the C code with clock initialization functions and configuration structures.
Peripherals Tool	Enables you to configure the initialization for the MCUXpresso SDK drivers.
Device Configuration Tool	Enables you to generate a Device Configuration Data (DCD) image using the format and constraints specified in the Boot ROM reference manual.
TEE (Trusted Execution Environment) Tool	Enables you to configure security policies of memory areas, bus masters, and peripherals, in order to isolate and safeguard sensitive areas of your application.

1.2 Versions

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

Note: The desktop version of the tool contacts the NXP server and fetches the list of available processors. Once connected, processor 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.

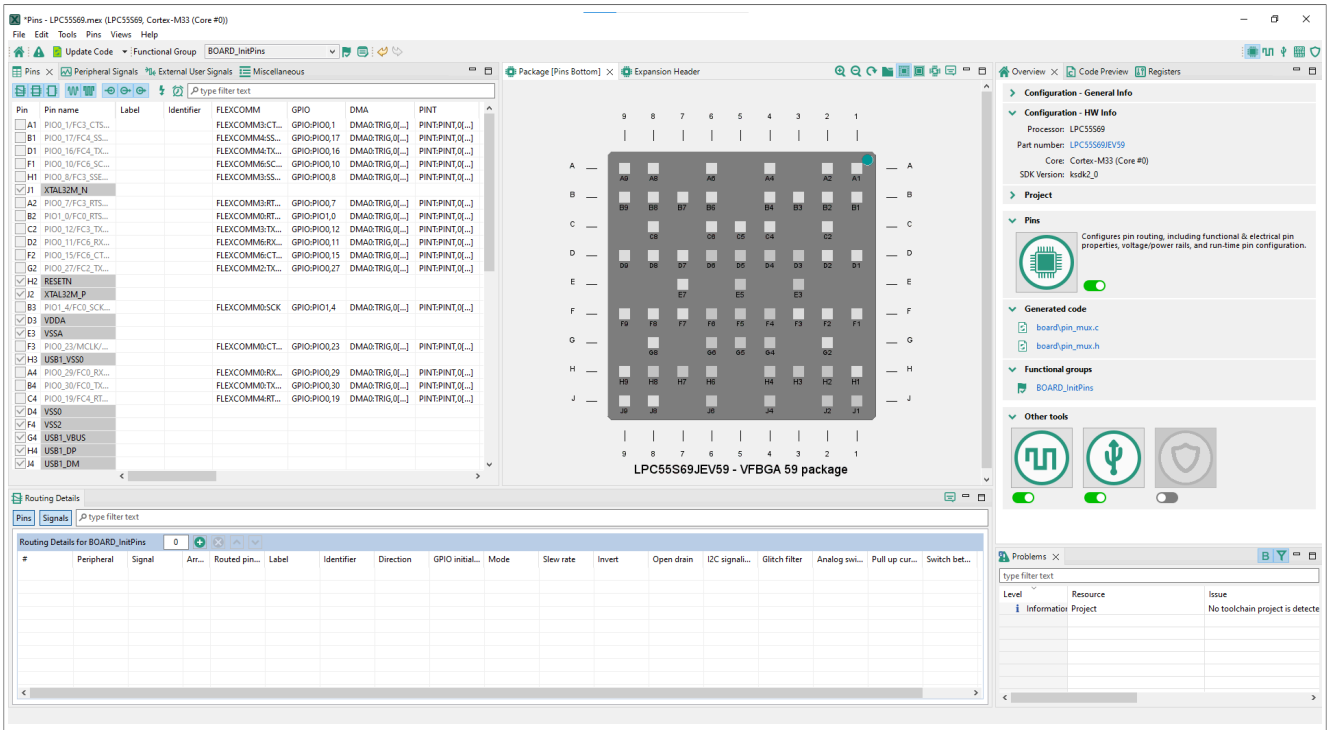


Figure 1. Desktop version of Pins tool

1.3 Config tools updates

In MCUXpresso IDE, the update site used by MCUXpresso Config Tools is disabled by default. To update MCUXpresso Config Tools integrated in MCUXpresso IDE, follow these steps:

1. Open MCUXpresso IDE.
2. Go to **Eclipse Preferences -> Install/Update -> Available Software Sites**
3. Select **Manage…**, check MCUXpresso Config Tools Update Site, click **Apply** and **Close**.

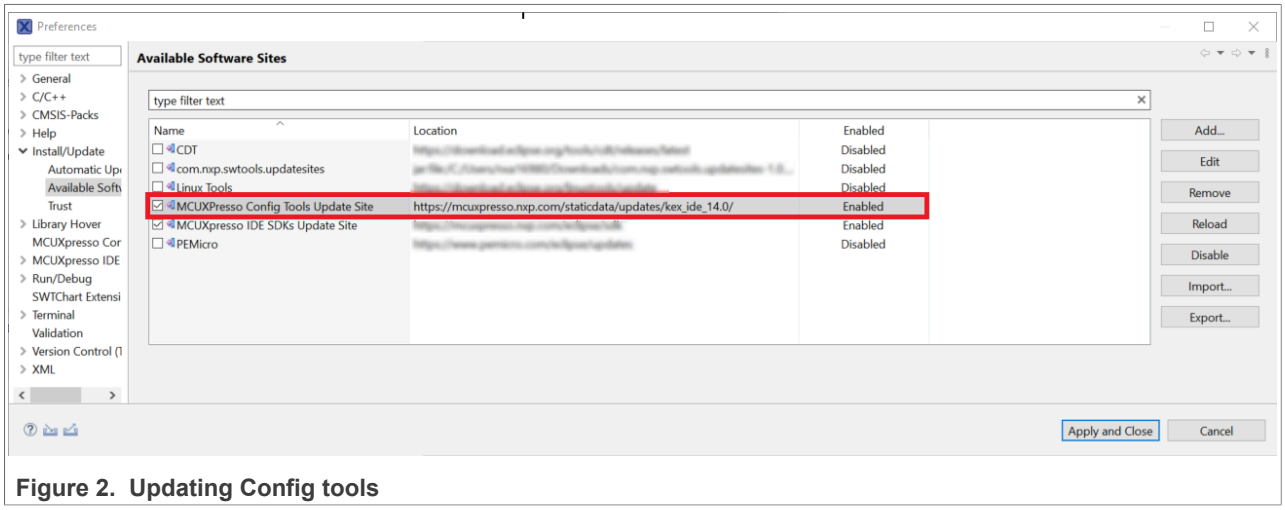


Figure 2. Updating Config tools

4. After enabling the MCUXpresso Config Tools update site, go to **Help** > **Check for Updates** to start the update process.

2 User interface

This section provides details on the 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. The folder with the saved MEX file must contain exactly one project file to be able to parse the toolchain project.

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

You can import source code files to use as a basis for further configuration.

Note: You can import only C or DTSI files containing valid YAML configuration blocks generated by the Config Tools. The configuration is reconstructed from the YAML block and the rest of the imported file is ignored.

To import source code files, do the following:

1. In the **Menu bar**, select **File > Import…**.
2. From the list, select **MCUXpresso Config Tools > Import Source**.

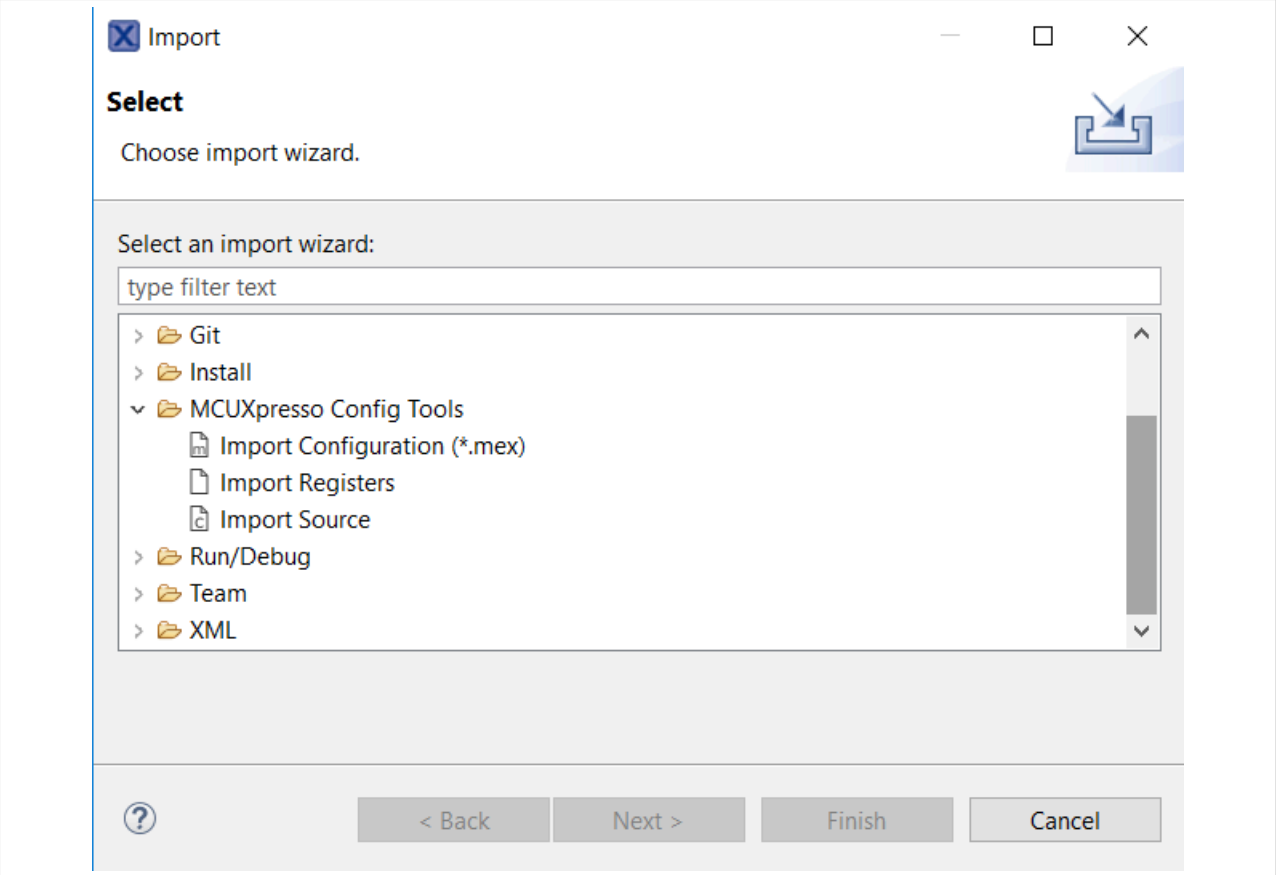


Figure 3. Import Source wizard

3. Click **Next**.
4. On the next page, click **Browse** to specify the location of the source file.
5. Select the source file that you wish to import and click **Open**.
6. On the next page, select which functional groups to import (based on tools) by selecting the checkbox in the left column.
7. Define how to import the functional groups by selecting one of the two available options in the dropdown menu in the right column:
   - **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` 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.
8. Click **Finish**.

2.1.3.1 Importing Board/Kit configuration

Use the import settings from the default board/kit templates provided in the CFG tools data for further configuration.

To import board/kit configuration, do the following:

1. In the **Menu bar**, select **File > Import...>**.

2. In the **Import** wizard, select **MCUXpresso Config Tools** > **Import Board/Kit Configuration**.
3. Click **Next**.
4. On the next page, select the board/kit variant from the dropdown menu.
5. Select which functional groups to import (based on tools) by selecting the checkbox in the left column.
6. Define how to import the functional groups by selecting one of the two available options in the dropdown menu in the right column:
   - **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 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.
7. Click **Finish**.

2.1.3.2 Importing configuration

To import an existing configuration from an MEX file, do the following:

1. In the **Menu bar**, select **File > Import…>**.
2. In the **Import** wizard, select **MCUXpresso Config Tools** > **Import configuration (*.mex)**.
3. Click **Next**.
4. On the next page, click **Browse** to specify the location of the registers file.
5. Select the MEX file that you wish to import and click **Open**.
6. On the next page, select which functional groups to import (based on tools) by selecting the checkbox in the left column.
7. Define how to import the functional groups by selecting one of the two available options in the dropdown menu in the right column:
   - **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` 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.
8. Click **Finish**.

2.1.4 Importing registers

You can import register configuration from a processor memory dump.

Note: Currently, register configuration can be imported into the Clocks tool only.

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

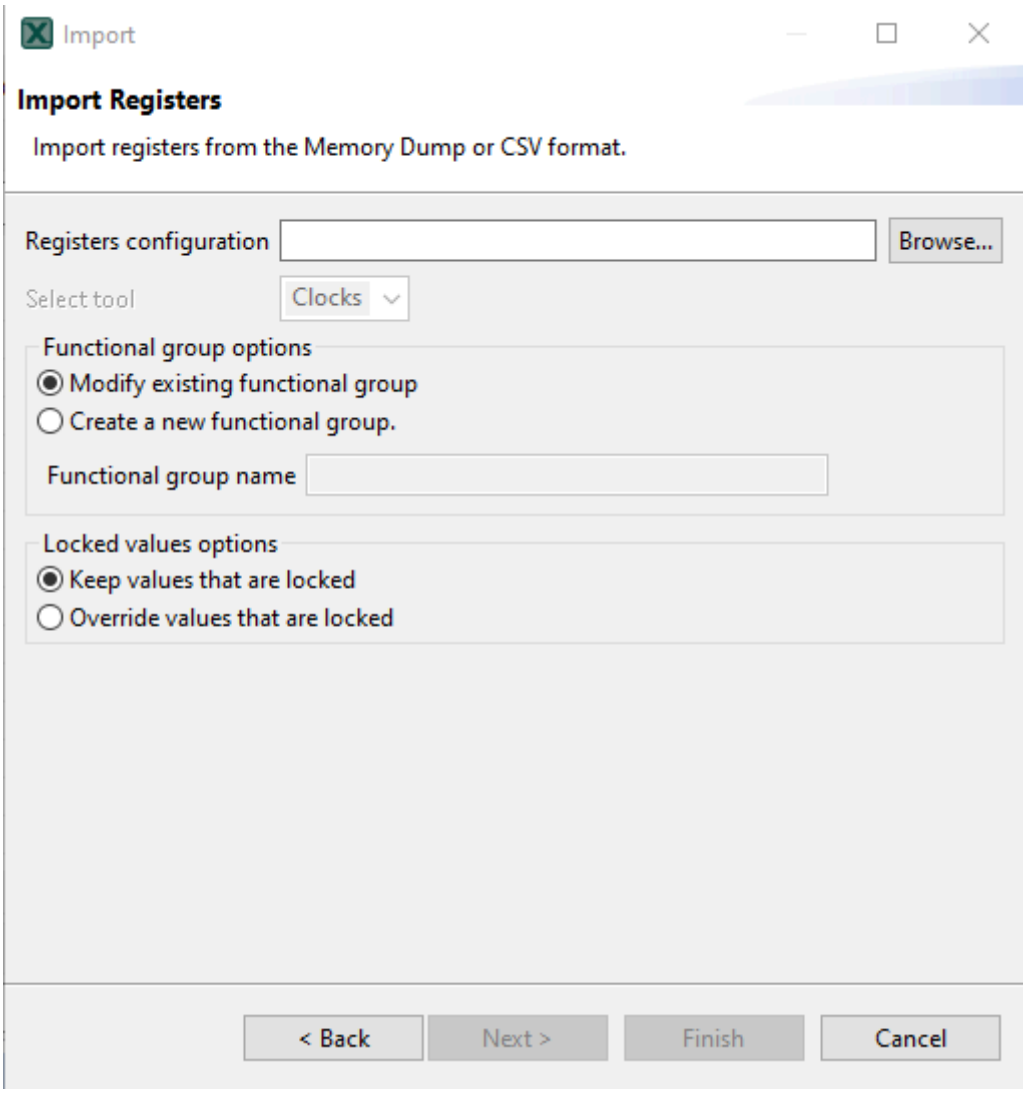


Figure 4. Import Registers

To import register configuration, do the following:

1. In the **Menu bar**, select **File > Import…**. Alternatively, click the **Import Registers Configuration** button in the **Registers** view, or drag-and-drop the memory dump file anywhere in the **Registers** view area.

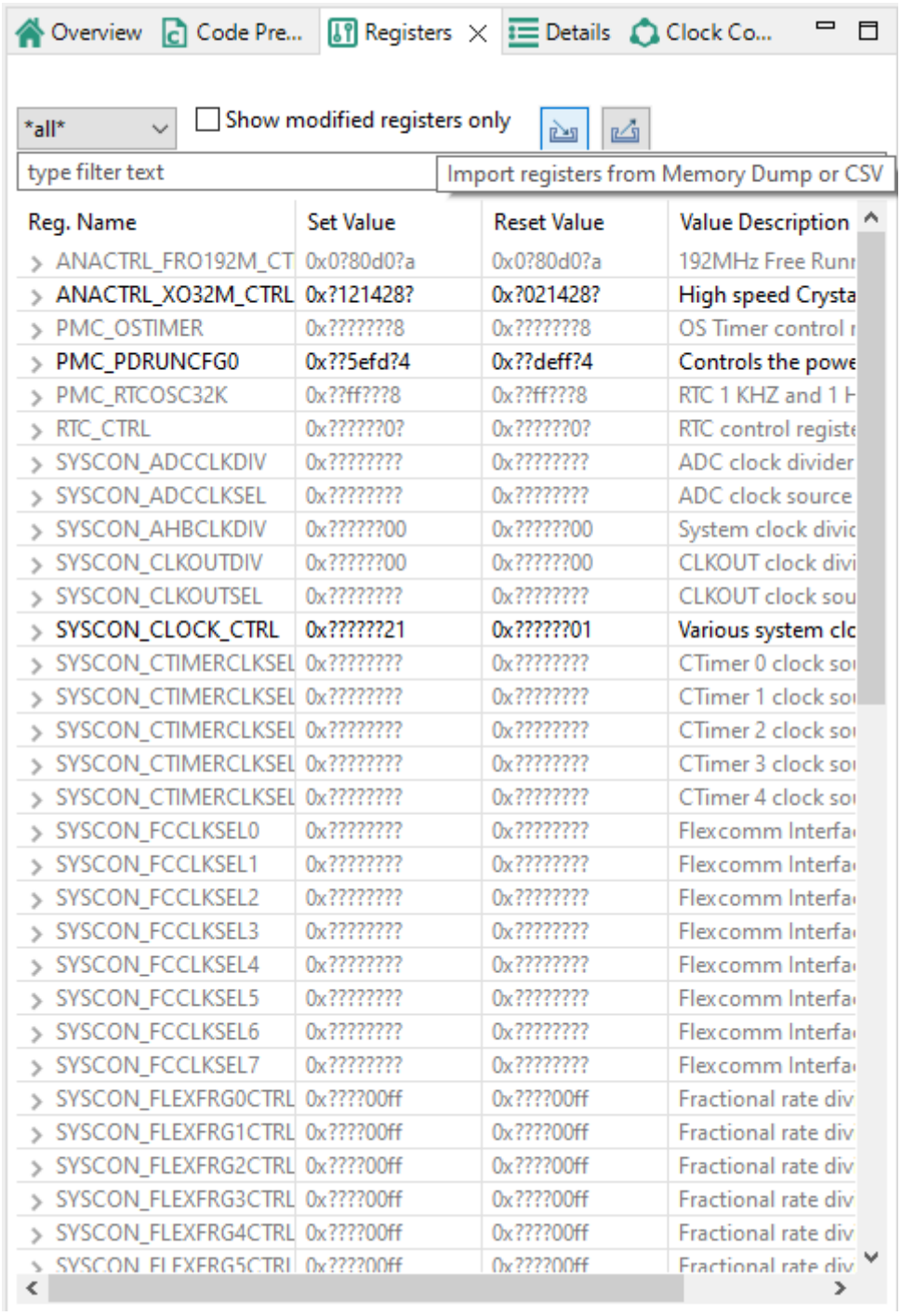


Figure 5. Import Registers Configuration

2. In the **Import** wizard, select **MCUXpresso Config Tools** > **Import Registers**.
3. Click **Next**.
4. On the next page, click **Browse** to specify the location of the registers configuration.
5. Select the registers file you wish to import, and click **OK**.

6. By default, the imported register configuration will overwrite the existing functional group. If you want a new functional group to be created instead, select the **Create new functional group** option button, and specify the functional group name.
7. 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.

2.1.5 Restoring configuration from source code

All Config tools have a possibility of restoring configuration from the source code. The generated code below contains information about the Clocks tool settings that are used in the tool (block within a comment in YAML format).

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

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

Figure 6. Setting Information in the source code

If this information is not corrupted, it is possible to reimport the clock settings into the tool using the following steps.

1. In the **Menu bar**, select **File** > **Import…**.
2. From the list, select **MCUXpresso Config Tools** > **Import Source Files**.
3. Click **Next**.
4. Click **Browse**.
5. Navigate and select the source file previously produced by one of the Config tools (for example, *clock_config.c*).
6. If the settings parse successfully, clock configurations are added into the current global configuration.

2.1.6 Preliminary processor data

Some processors are published for public release before their configuration data reaches full RFP (Ready-For-Production) quality. Such processors are marked as **preliminary** in the processor data.

When you use a processor whose data is marked as preliminary, MCUXpresso Config Tools indicates this in the **Problems** view by displaying a warning that notifies you that the current configuration relies on preliminary processor data.

Note: Preliminary processor data is subject to change. Configurations based on preliminary data should be re-validated once the final (RFP-quality) processor data becomes available.

2.2 Toolbar

The toolbar is at the top of the window and includes buttons and menus for frequently used actions common to all tools. See the following sections for more information.

2.2.1 Table Toolbar

Table 2. Toolbar

Item	Description
Config Tools Overview	Open the **Overview** dialog with information about currently used tools.
Show Problems View	Open the **Problems** view.
Update Code	Open the update dialog allowing you to update generated peripheral initialization code directly within specified toolchain project.
Generate Code	Regenerate source code when “Enable Code Preview” preference is disabled.
Functional group selection	Select functional group. 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.
Call from default initialization	Set the current functional group to be initialized by the default initialization function.
Functional group properties	Open the **Functional group properties** dialog to modify name and other properties of the function group.
Tool selection	Display icons of individual tools. Use them to switch between tools.
Undo/Redo	Undo/Redo last action.

In addition, the toolbar may contain additional items depending on the selected tool. See the chapters dedicated to individual tools for more information.

2.2.2 Eclipse project selection

You can use the **Eclipse project** drop-down menu to switch between projects. If a project contains several MEX configuration files (in its root folder), all are listed and can be switched individually.

2.2.3 Config Tools overview button

Click the **Config Tools Overview** button to open **Config Tools Overview** and inspect information about the configuration, hardware, and project. For more information, see the Config Tools Overview section.

2.2.4 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.

The button color depends on the issue type. Red indicates the presence of at least one error; yellow indicates the presence of at least one warning.

2.2.5 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.

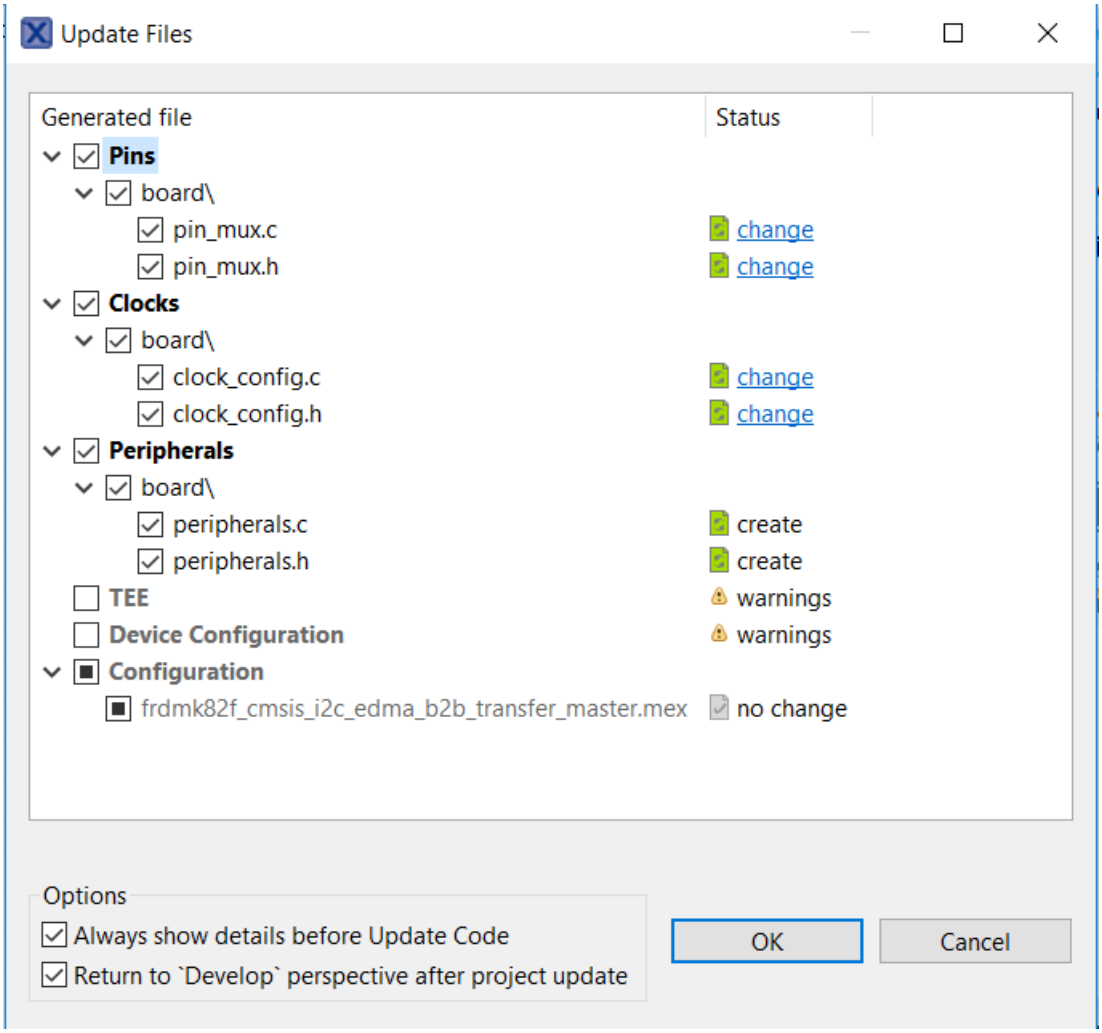


Figure 7. Update Files window

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

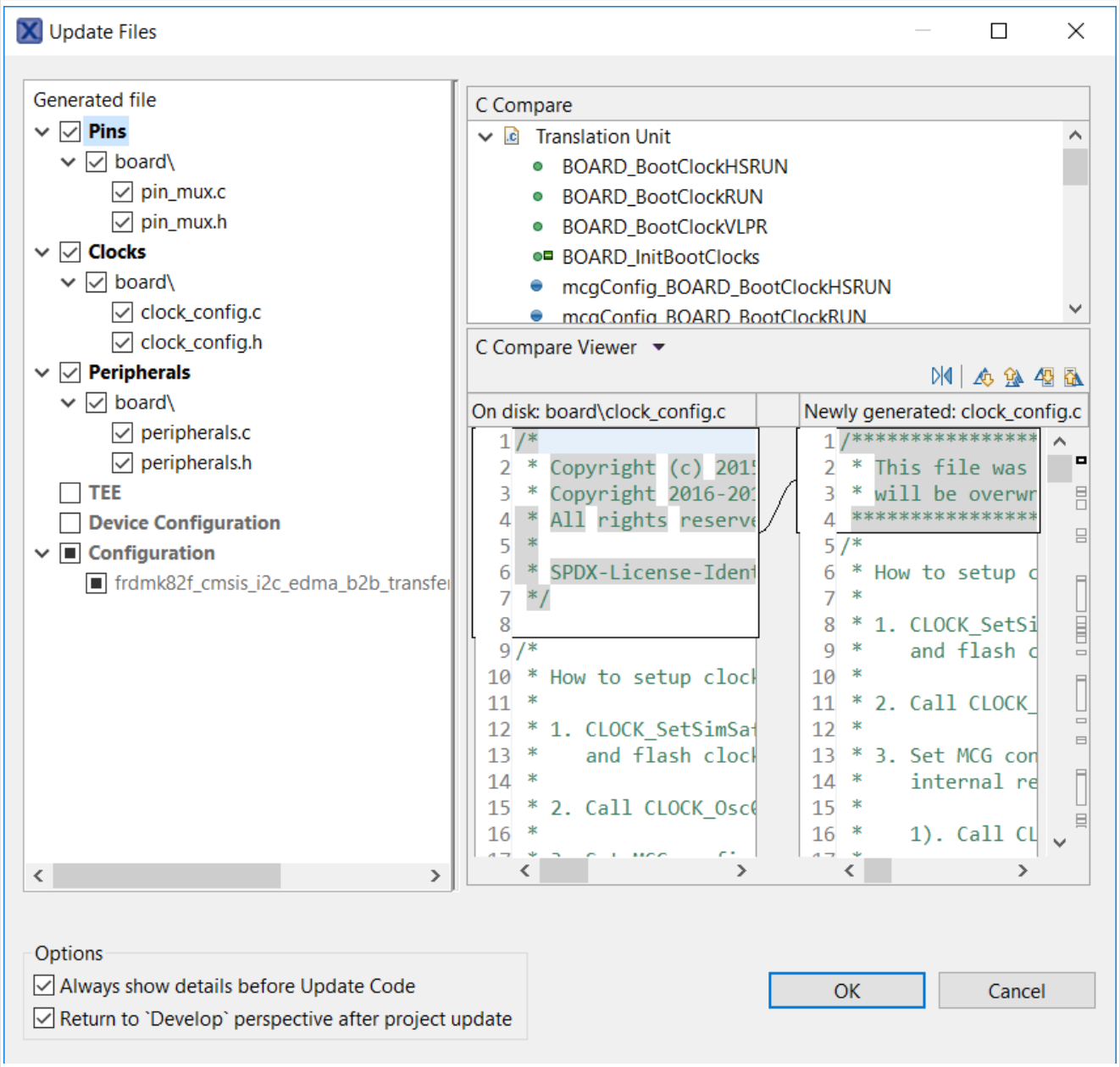


Figure 8. Show differences

To update the project without opening the **Update Files** dialog, deselect the **Always show details before Update Code** checkbox.

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

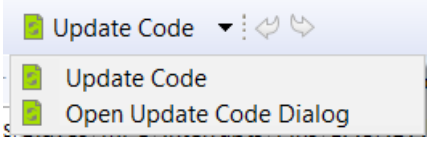


Figure 9. Update Code dropdown menu

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 the following conditions:

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

2.2.6 Functional groups

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

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

Figure 10. Functional groups

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

2.2.6.1 Table Functional Groups

Table 3. Functional Groups

Icon	Description
	Toggle “Called from default initialization function” feature (in source code)
	Opens the **Functional group properties** window

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

2.2.6.2 Functional group properties

In the **Functional Group Properties** window, you can configure several options for functions and code generation. Each setting applies to the selected function. You can specify the generated function name, select the core (for multicore processors only) that affects the generated source code, or write a function description (this description is 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 parameters for selected functional groups.

Functional group properties are specific for individual Config Tools:

The Pins tool:

- **Set custom #define prefix**- If this property is set, the specific custom prefix is used for macros generated into the `pin_mux.h`. Otherwise the name of the functional group is used as the prefix.
- **Prefix** - The custom prefix string. If it is empty, no prefix is used.
- **Clocks gate enable** - If this property is enabled, the clock gate is enabled in the generated code. The clock gate is needed for access to the peripherals, so have it enabled elsewhere.
- **Core** (for multicore processors only) - Selects the core that is used for executing this function.
- **Full pins initialization** - If this property is set, all features of the pins are fully initialized in the generated function even if the state matches the after-reset state of the processor. If it is not set, values “Not specified” or “No init” can be selected.

- **De-initialization function** - If this feature is set, an additional function that sets all pins and peripheral signals in this functional group to their after-reset state is generated. The new function has a suffix `_deinit` by default.
- **Set custom de-initialization function name** - Allows specifying a user-defined name of the de-initialization function.

Clocks tool:

- **Set custom #define prefix** - If this property is set, the custom prefix is used for macros define in `clock_config.h`. Otherwise the name of the functional group is used as the prefix.
- **Prefix** - The custom prefix string. If it is empty, no prefix is used.
- **Other settings** - The processor-specific settings are specific for each processor. See the tooltips for details.

Peripherals tool:

- **Prefix** - It is used for identifiers, constants, and functions related to the functional group that is used in generated code. If it is not specified, no prefix is used.

TEE tool:

- **Set custom #define prefix** - If this property is checked, the custom prefix is used for macros define in generate code. Otherwise, the name of the functional group is used as the prefix.

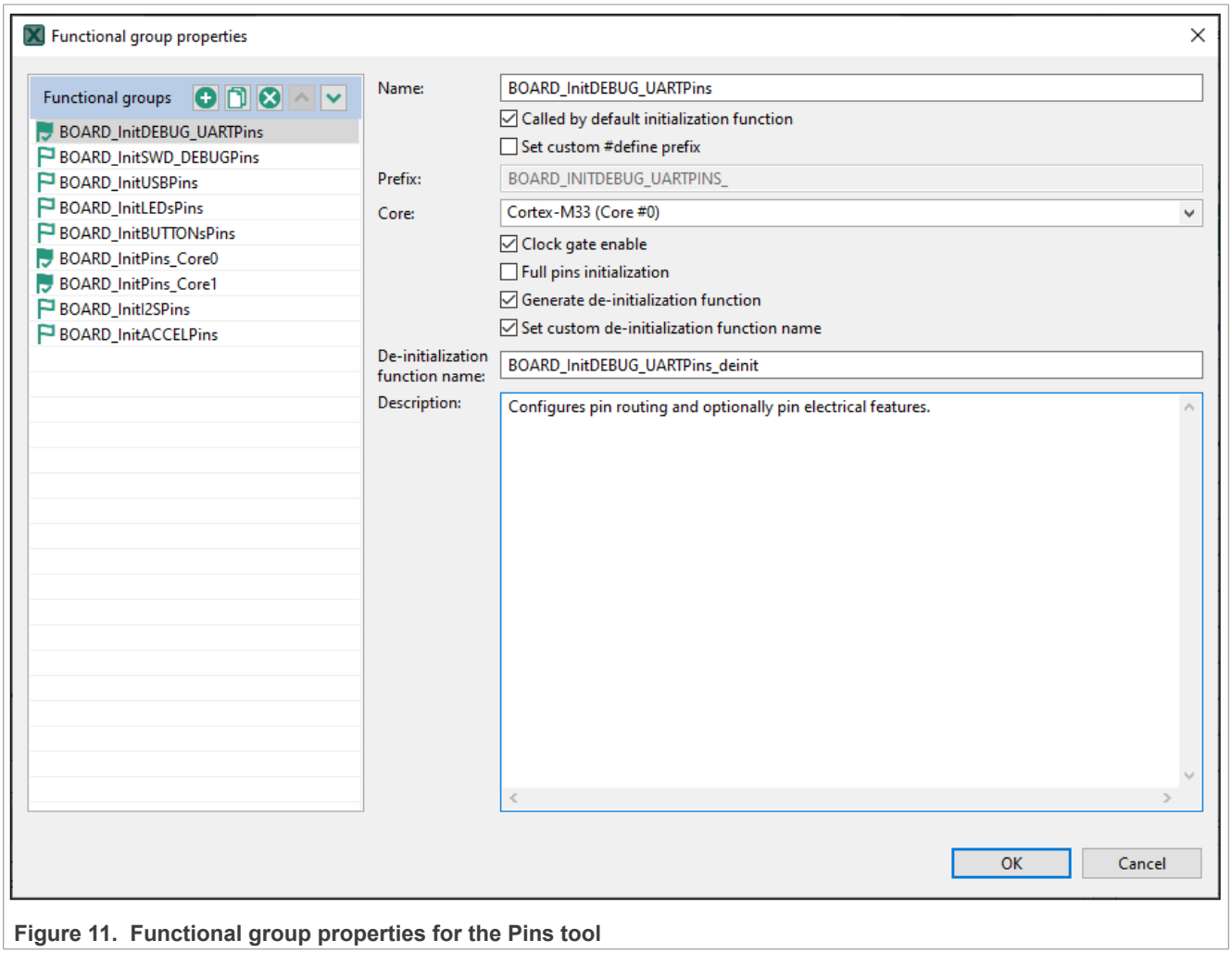


Figure 11. Functional group properties for the Pins tool

2.2.7 Undo/Redo actions

You can reverse your actions by using Undo/Redo buttons available in the **Toolbar**. You can also perform these actions from the **Edit** menu in the **Menu bar**.

2.2.7.1 Table Undo/redo actions

Table 4. Undo/redo actions

Icon	Description
	Cancels the previous action
	Cancels the previous undo action

You can also discard all unsaved changes using a command from the main menu **ConfigTools>Reload Configuration (*.mex)**.

2.2.8 Selecting the tools

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

2.3 Status bar

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

2.4 Preferences

To configure preferences in the **Preferences** dialog, select **Window>Preferences>MCUXpresso Config Tools** from the **Menu bar**.

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

Functional group properties for the Pins tool

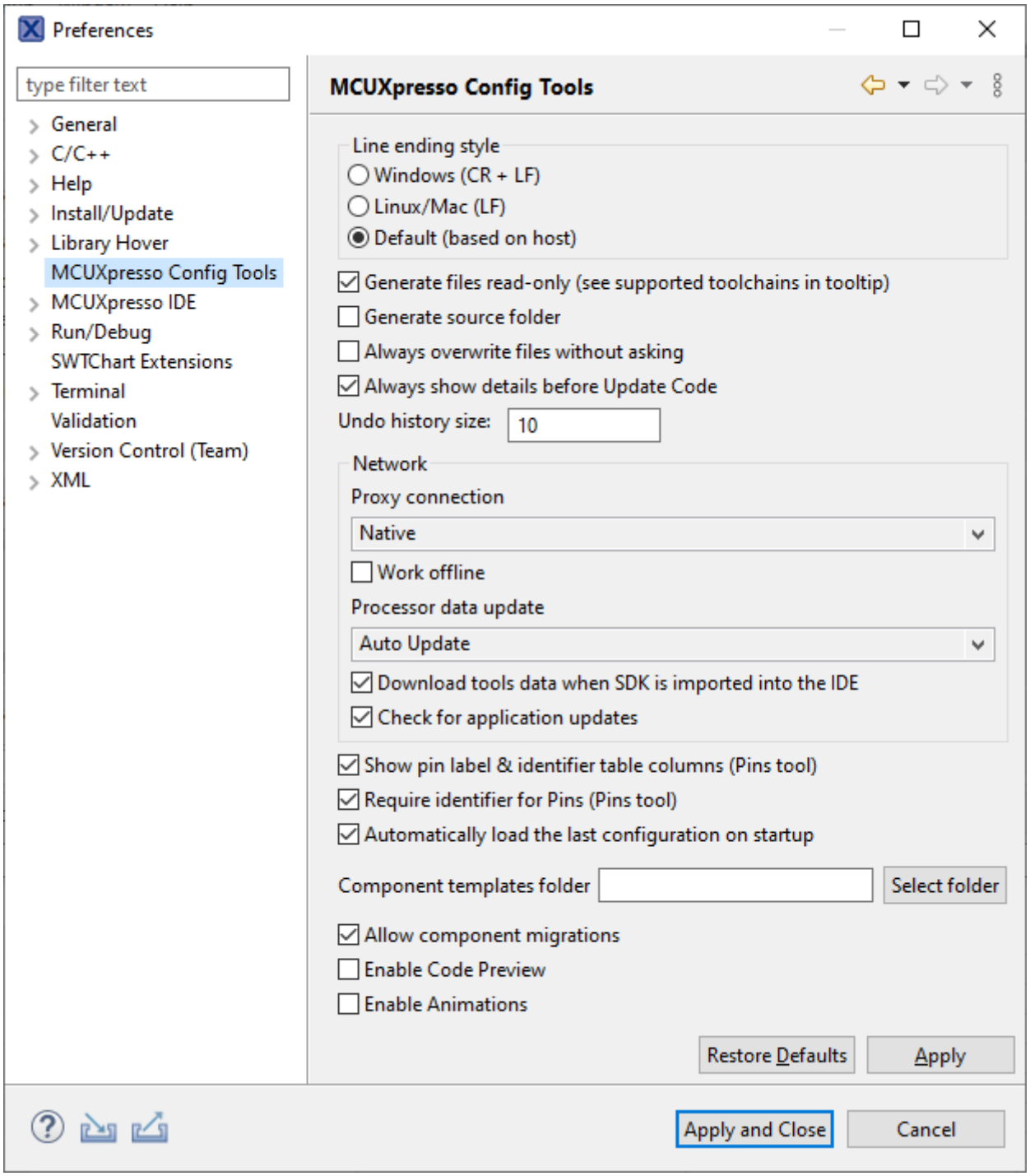


Figure 12. Preferences

Several settings are available.

2.4.1 Table Preferences

Table 5. Preferences

Item	Description
Line ending style	Select between **Windows (CR + LF)**, **(LF)**, or **Default (based on host)**.
Always overwrite files without asking	Update existing files automatically, without prompting.
Always show details before Update Code	Review changes before the project is updated.
Undo history size	Enter the maximum number of steps that can be undone. Enter 0 to disable.
Component template folder (Peripherals Tool)	The path to the folder with component templates. Keep empty to use the default path. The default path is to the folder component_templates in data of the Config tools.
Allow component migrations (Peripherals Tool)	When a configuration associated with a toolchain project is open, the peripheral tool automatically checks if the configuration components match the project and suggests a migration if they are not.
Show internal IDs in Peripherals Tool	When checked, the tooltips in the Peripherals tool contain internal identifications
Enable animations	Enables animations in the user interface, such as smoother scrolling or opening a drop-down menu.

2.5 Configuration preferences

In the **Configuration preferences** window, you can set your preferences for the configuration storage file (MEX).

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

Several preferences are available.

2.5.1 Table Configuration preferences

Table 6. Configuration preferences

Item	Description
Validate boot init only	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	Generate YAML into C sources files.
Custom source file copyright header	Add a custom copyright header to generated source files that do not already contain copyright.
Generate defines of clock registers	This feature is disabled by default (**Edit->Configuration Preferences**). The new `registers.h` file with registers defines is generated in the **Code Preview** tab. The custom prefix can be defined in the Functional group properties.

Table 6. Configuration preferences...*continued*

Item	Description
Generate code only for registers that are different from the after-reset state	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.
Output path overrides	Rules that are used to override the path of the output files are generated by tools. They are applied in the Update code and Exports commands. A special dialog allows editing. For more information, see Output path overrides.

Warning: When the source does not contain YAML code, it cannot be imported.

2.6 Problems view

The **Problems** view displays issues in individual tools and in the inter-dependencies between the tools.

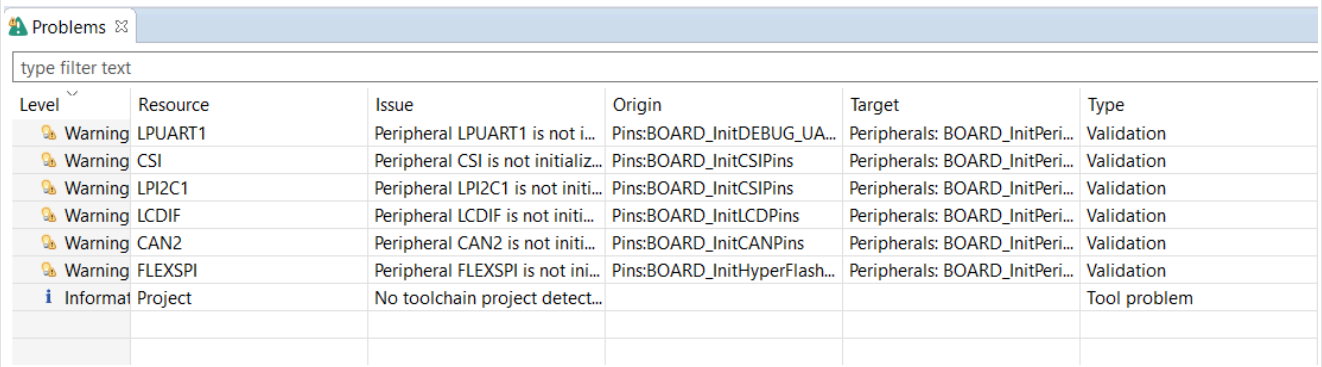


Figure 13. Problems view

To open the **Problems** view, click the **Show Problems view** button in the **Toolbar**, or select **Views > Problems** from the **Menu bar**.

The **Problems** table contains the following information:

2.6.1 Table Problems view

Table 7. Problems view

Item	Description
Level	Severity of the problem: Information, Warning, or Error.
Resource	Resource related to the problem, such as signal name, the clock signal.
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 is 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 rows for later use by right-clicking a row and selecting **Copy** or using **Ctrl+C**. Use **Ctrl+left-click** to add additional rows to the selection.

Note: Quick fix is only available for problems highlighted with the “light bulb” icon.

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

2.6.2 Table Filter buttons

Table 8. Filter buttons

Button	Description
	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.

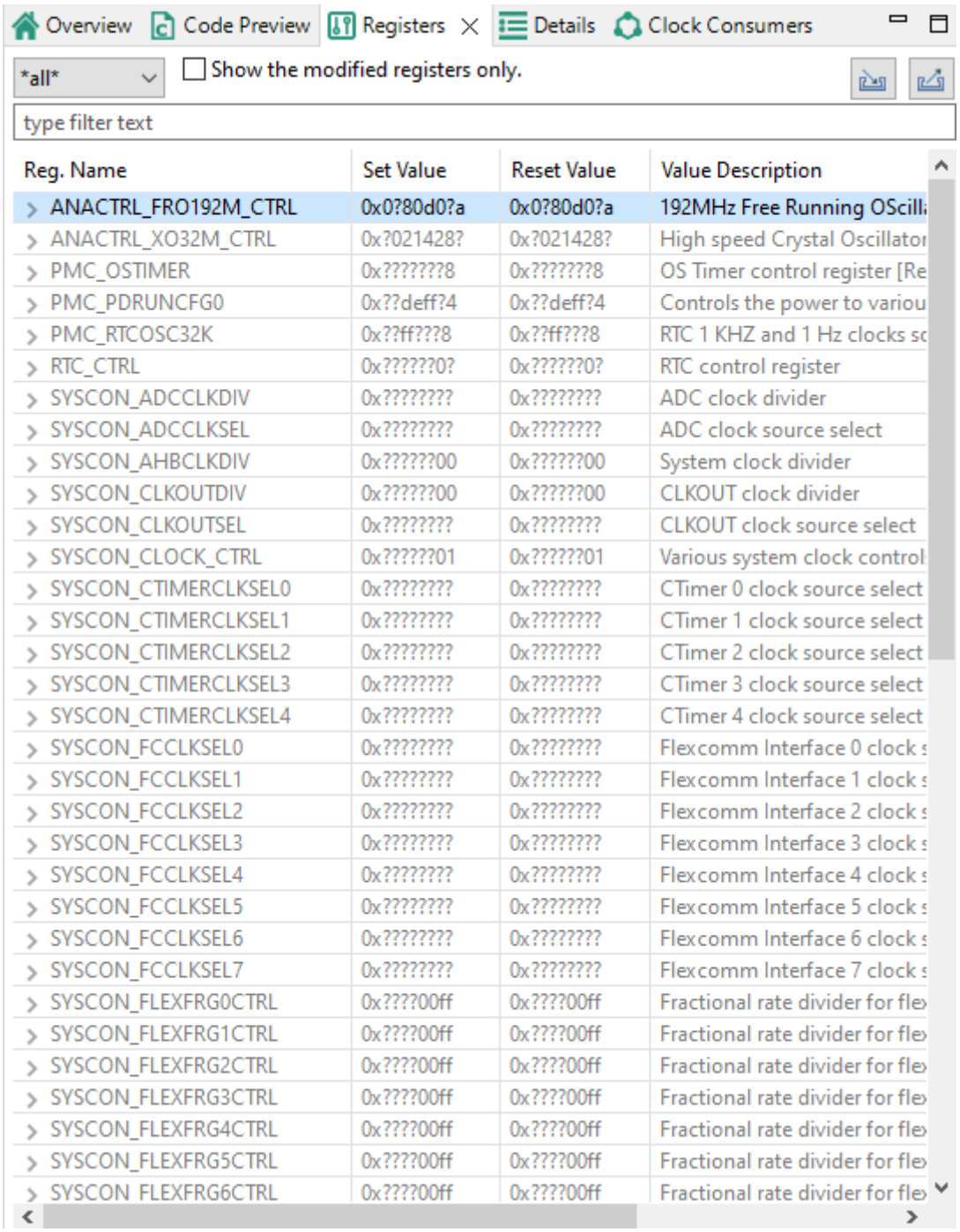


Figure 14. Registers view

The **Registers view** contains several items.

2.7.1 Table Registers

Table 9. Registers

Item	Description
Peripheral filter drop-down list	List the registers only for the selected peripheral. Select **all** to list registers for all the peripherals.
Show modified registers only checkbox	Hide the registers that are left in their after-reset state or are not configured.
Text filter	Filter content by text.
Import/Export registers	Import/Export registers from/to CSV (Import is available only for the Clocks tool)

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

2.7.2 Table Color codes

Table 10. Color codes

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

Note: When the Peripherals tool is active and register initialization components are used, the user can perform manual changes to some of the displayed values.

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 MCUXpresso Config Tools operations. The **Log** view can show up to 100 records across all tools in chronological order.

Each log entry consists of a timestamp, the name of the tool responsible for the entry, severity level, and the actual message. If no tool name is specified, the entry was triggered by shared functionality.

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

Click the clear button to clear buffered log records. It affects **Log** views across all tools.

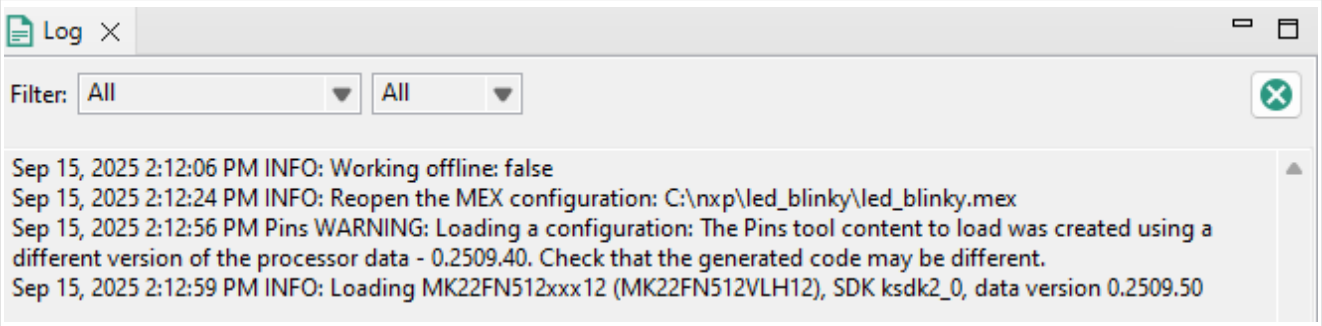


Figure 15. Log view

2.9 Config tools overview

The **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 on the left side of the toolbar.

Config Tools Overview contains several items.

2.9.1 Table Config Tools Overview

Table 11. Config Tools Overview

Item	Description
Configuration - General Info	Displays the name of and the path to the MEX file of the current configuration. To import additional settings, click the link to open the folder containing the MEX file.
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.

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:

2.9.2 Table Config Tools Overview

Table 12. Config Tools Overview

Item	Description
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.

To open a tool-specific overview, select **Views > Overview** from the main menu.

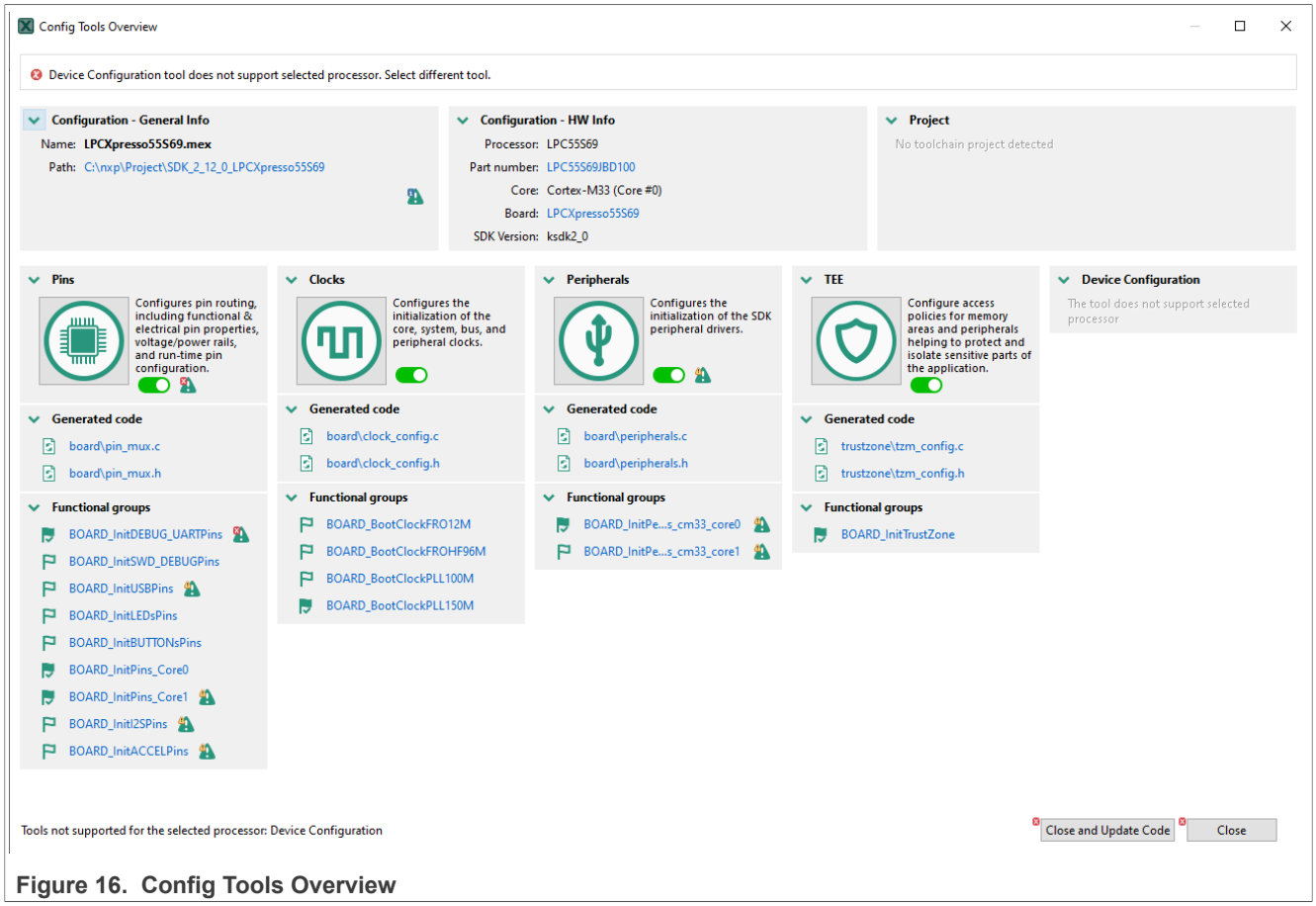


Figure 16. Config Tools Overview

2.10 Config Tools snippets

The **Config tools snippets** view can be opened in the Config Tools perspective. The snippets view shows snippets related to the currently selected project. If you edit a particular file, the view shows items related to the file's project. The snippets can be categorized; the hierarchy is controlled by the tool. Double-click or use the specific icon to place the source code of the selected snippet into the active editor.

Note: In the current version, the **Config tools snippets** view is supported for the Peripherals tool only.

3 Pins Tool

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

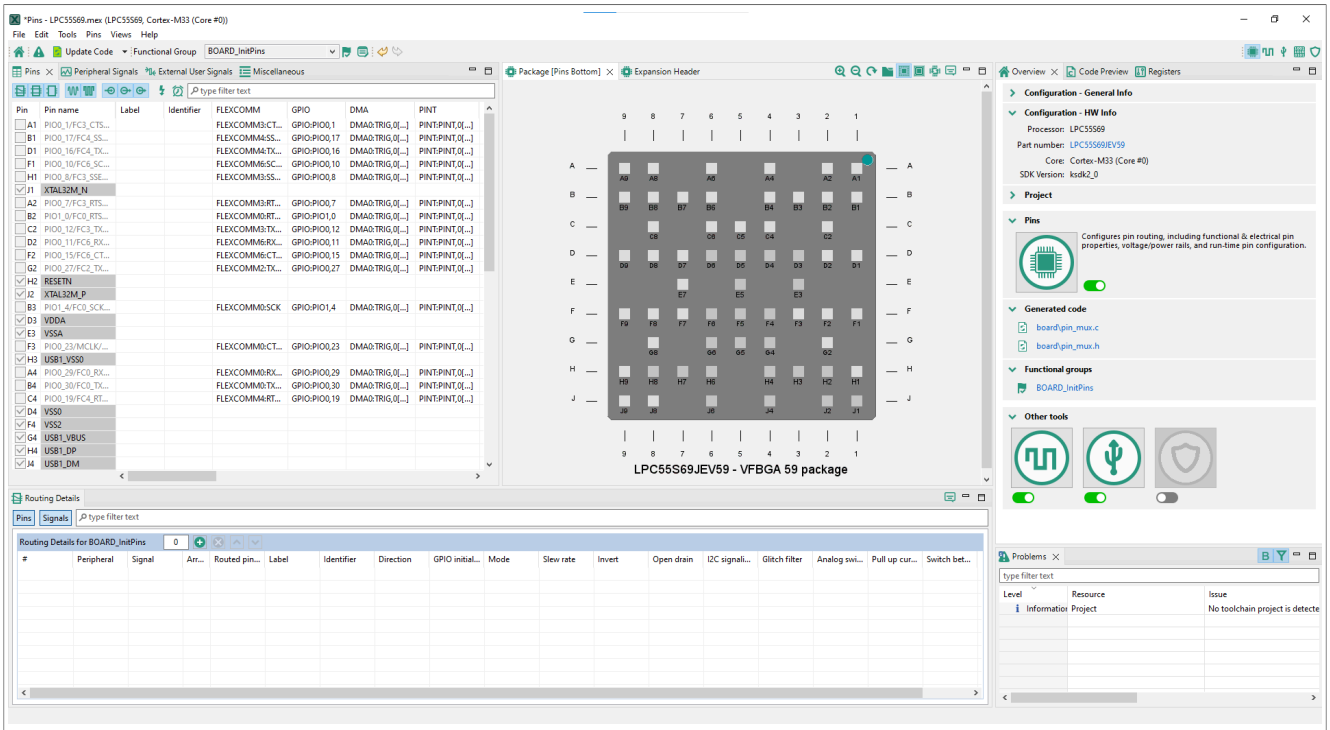


Figure 17. Pins tool

3.1 Pins routing principle

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

An internal signal is an interconnection node to which peripheral signals can be connected (without any pin interaction). Connecting two peripheral signals to an internal signal creates an interconnection between these two peripheral signals.

This routing configuration can be done in the following views:

- **Pins**
- **Peripheral Signals**
- **Package**
- **Routing Details**

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 the peripheral in the **Routing Details** view and the **Peripheral Signals** view.

1. Select the **Peripheral**.
2. In **Routing Details** view, select one of the available **Signals** or expand the peripheral in **Peripheral Signals** view.
3. Select the desired pin/internal signal.
   Items (pins/internal signals) in the **Routed pin/signal** column in the **Routing Details** view have the 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 the **Routing details** view contains items that are connectable to the selected signal (without its channel if applicable). So when selected signal is “GPIO, 6” the **Routed pin/signal** column provides items connectable to “GPIO”.

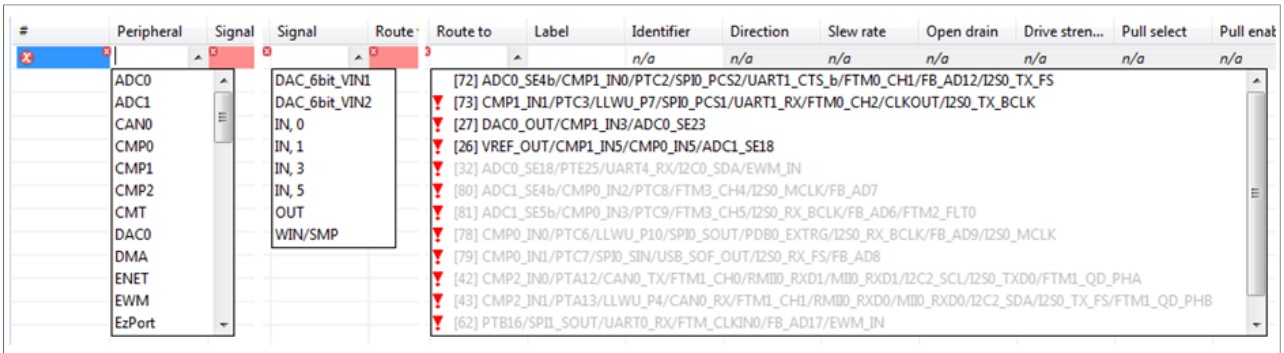


Figure 18. Defining routing path

3.1.2 Beginning with pin/internal signal selection

You can select a pin or an internal signal in the **Routing Details** view.

1. Select the pin/internal signal (**Routed pin/signal**).
2. Select one of the available **Peripherals**.
3. For the selected peripheral, select one of the available **Signals**.
   Items in **Peripheral** column in **Routing Details** view have the following symbols:
   - Exclamation mark and default text color indicate that such item selection can cause a register conflict or the item does not support selected signal.
   - Exclamation mark and gray text color indicate 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.

3.1.3 Routing of peripheral signals

Peripheral signals representing on-chip peripheral input or output can be connected to other on-chip peripherals or to a pin through an inter-peripheral crossbar. You can configure this connection in the **Routing Details** view.

Three types of peripheral signal routing are available:

1. Routing the signal from the output of an internal peripheral (A) into the input of another internal peripheral (B)
   The signal leads from the output of one internal peripheral (A) to the input node of another internal peripheral (B). In other words, the signal leads from A to B (A > B). To configure a signal in this way, perform the following steps (PWM triggering ADC (PWM > ADC) used as an example):
   a. Add a row in the **Routing Details** view.
   b. Select peripheral B from the drop-down list in the **Peripheral** column.

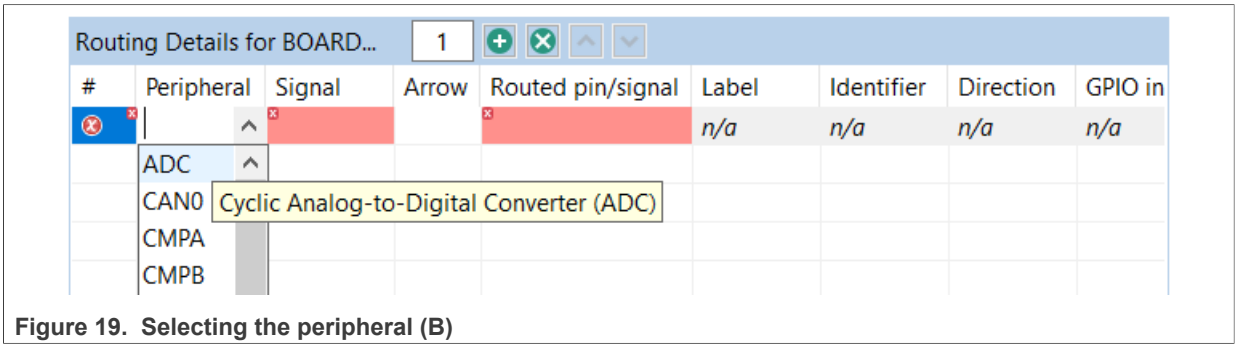


Figure 19. Selecting the peripheral (B)

c. Select the input node of peripheral B from the drop-down list in the **Signal** column.

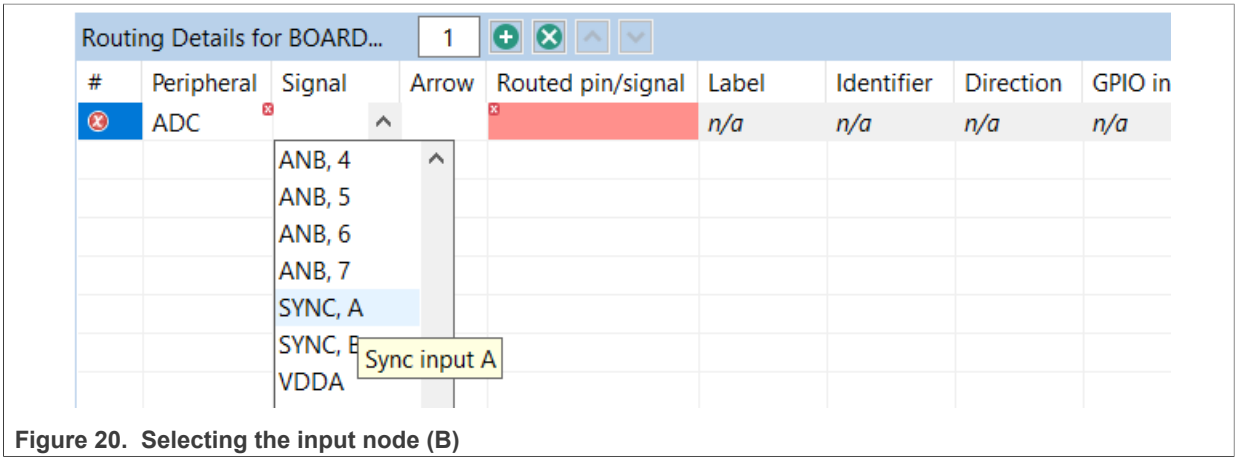


Figure 20. Selecting the input node (B)

d. Select the output signal of peripheral A from the drop-down list in the **Routed pin/signal** column.

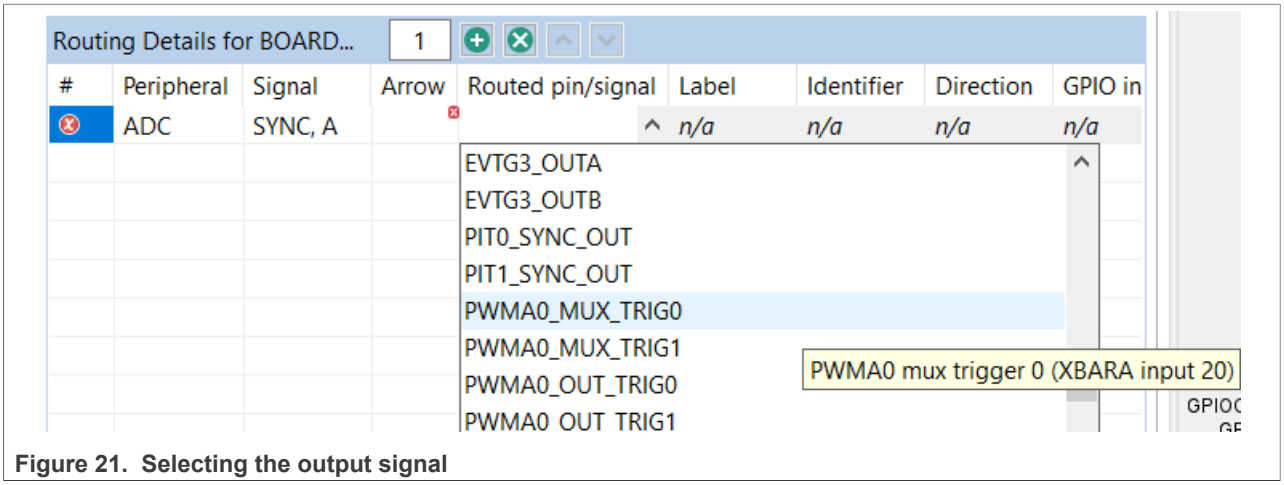


Figure 21. Selecting the output signal

Once the configuration is done, the row looks like this:

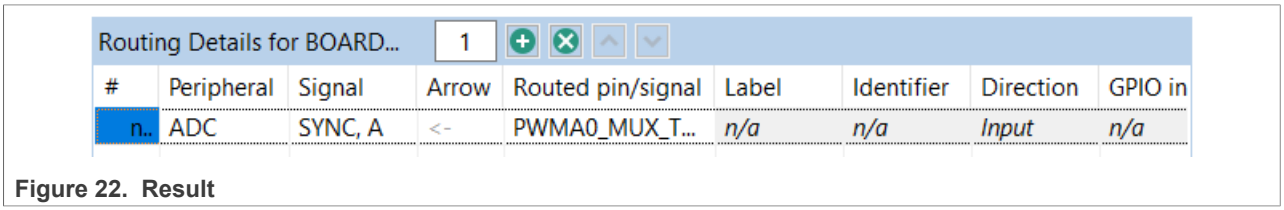


Figure 22. Result

Note: It is necessary to select the ADC peripheral where the signal leads to (input in ADC). It is a limitation of the Pins tool that the signal is not listed for the PWM peripheral (output). Notice the direction of the signal in the **Arrow** column.

2. Routing the signal from a pin on the package to an internal peripheral input signal through an inter-peripheral crossbar
   Note: Only if a crossbar switch is present.
   The signal leads from a pin on the package (XB_IN) connected through an inter-peripheral crossbar, to an internal peripheral (B) input node. In other words, the signal leads from XB_IN to B (XB_IN > B). To configure a signal in this way, perform the following steps (routing pin 55 using XB_IN6 to EVTG0 input A (XB_IN6 > EVTG0) used as example):
   a. Add a row in the **Routing Details** view.
   b. Select peripheral B from the drop-down list in the **Peripheral** column.

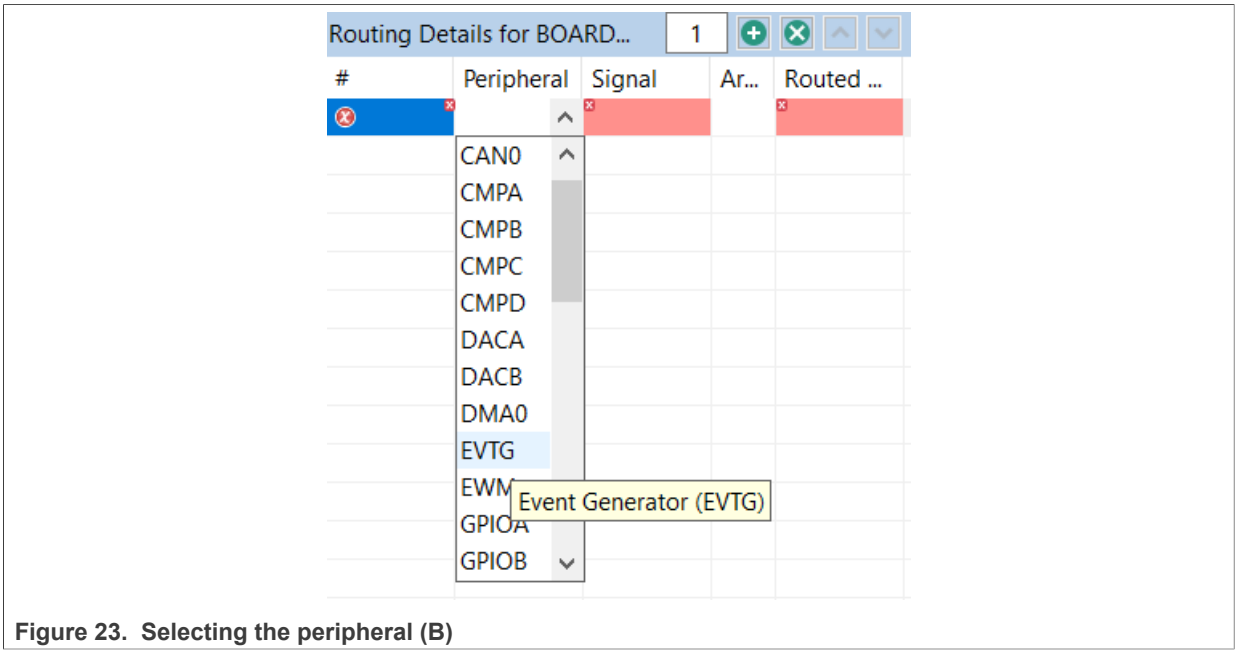


Figure 23. Selecting the peripheral (B)

   c. Select the input node of peripheral B from the drop-down list in the **Signal** column.

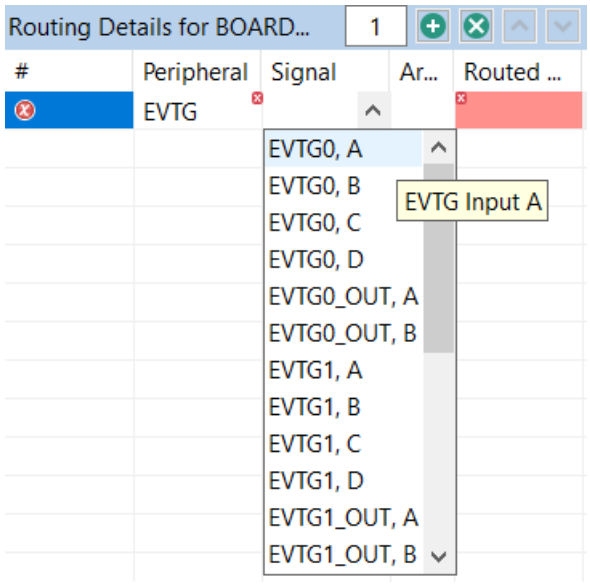


Figure 24. Selecting the input node (B)

d. Select the XB_IN pin from the drop-down list in the **Routed pin/signal** column.

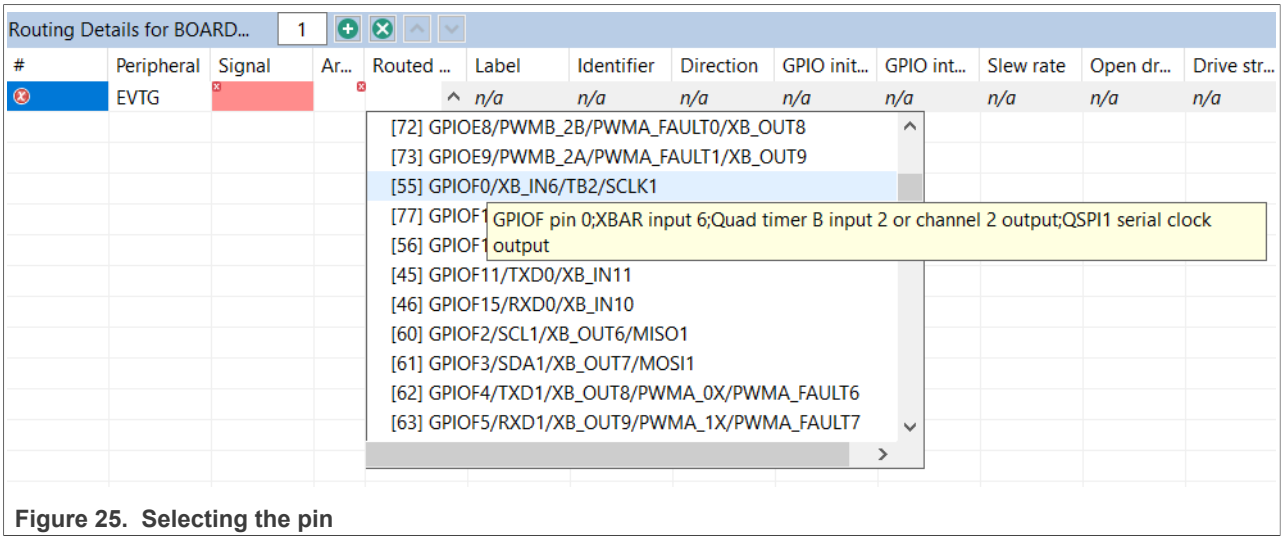


Figure 25. Selecting the pin

Once the configuration is done, the row looks like this:

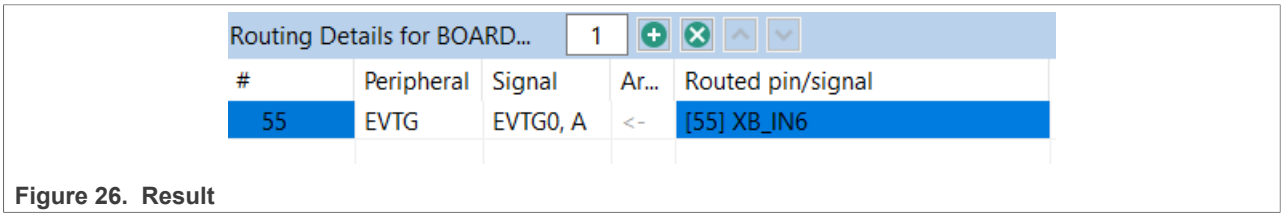


Figure 26. Result

Note: In this example, GPIOF0 is multiplexed with XB_IN6, QTimerB channel 2 output/input and QSPI1 SCLK signal. In this case, the tool will automatically pick XB_IN6 for the pin as XB_IN6 is the only option to be routed to EVTG0 input A.

3. Routing the signal from an internal peripheral (A) output to a pin via inter-peripheral crossbar
Note: Only if a crossbar switch is present.

The signal leads from an internal peripheral (A) output to a pin connected through an inter-peripheral crossbar on the package (XB_OUT). In other words, the signal leads from A to XB_OUT (A > XB_OUT). To configure a signal in this way, perform the following steps (routing EVTG0 output to a pin 87 using XB_OUT4 used as an example):

a. Add a row in the **Routing Details** view.

b. Select peripheral A from the drop-down list in the **Peripheral** column.

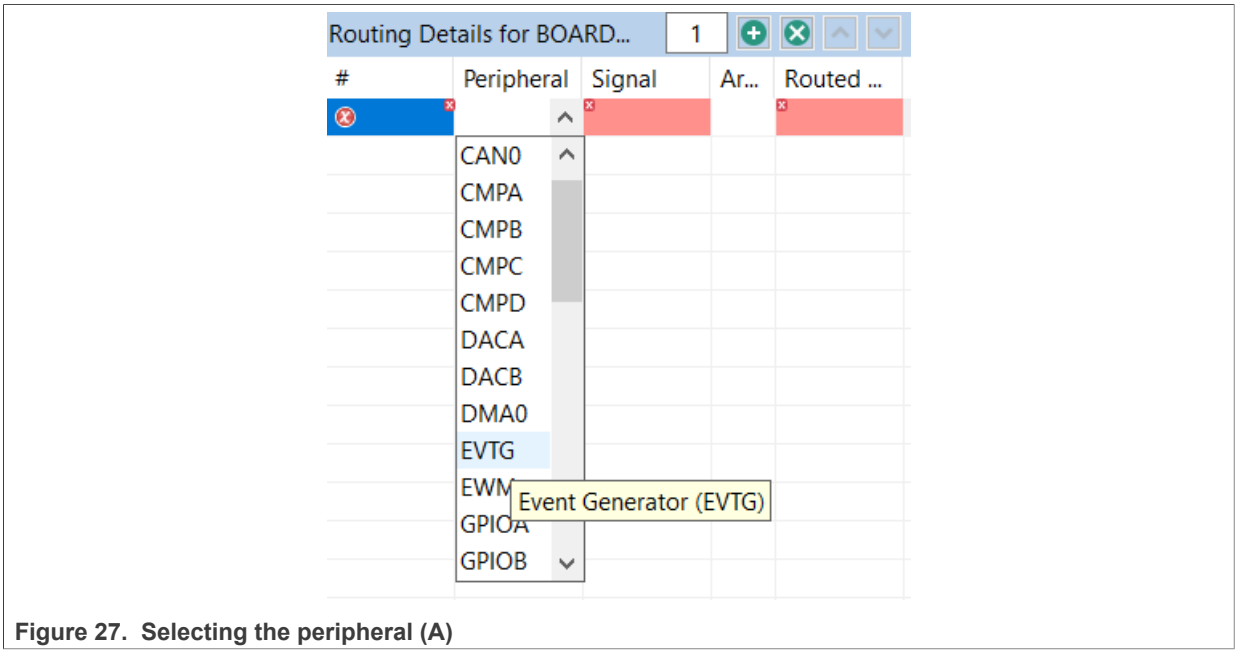


Figure 27. Selecting the peripheral (A)

c. Select the input node of peripheral A from the drop-down list in the **Signal** column.

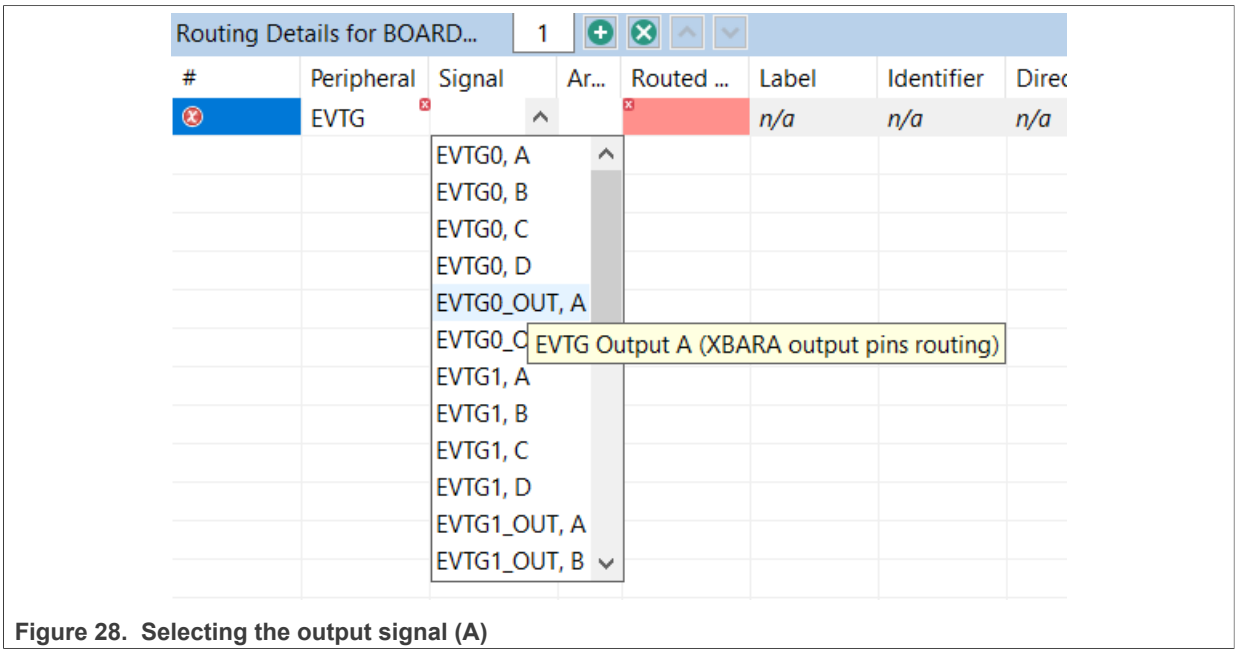


Figure 28. Selecting the output signal (A)

d. Select the XB_OUT pin from the drop-down list in the **Route to** column.

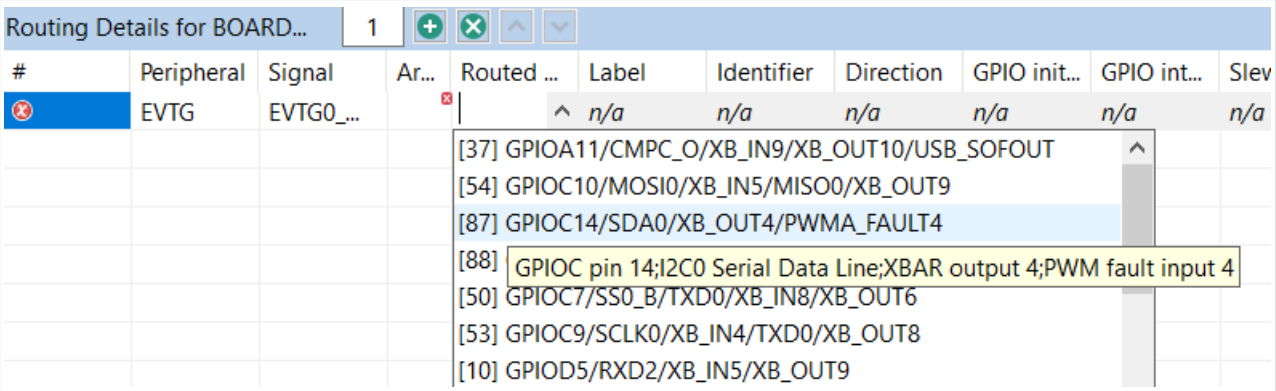


Figure 29. Selecting the pin

Once the configuration is done, the row looks like this:

Figure 30. Result

Note: In this example, GPIOC14 is multiplexed with XB_OUT4, SDA of I2C0 and fault4 of eFlexPWMA. In this case, the tool will automatically configure XB_OUT4 for the pin GPIOC14 (pin 87) as XB_OUT4 is the only option for EVTG0 output A.

Note: In some cases, multiple routings are configured by the same register change. When such routing is created, the user is offered an option to add the related routing to the functional group. Similarly, when a routing is removed, related routings are offered for removal.

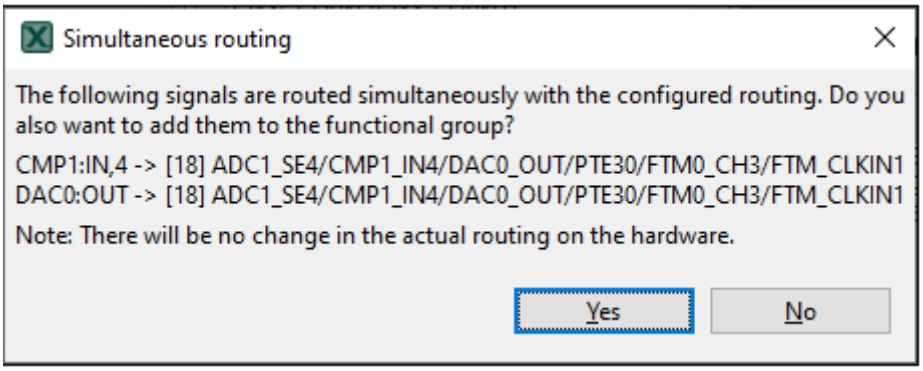


Figure 31. Simultaneous routing

3.2 Example workflow

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 it, you can click into the cells to make them ‘green’.

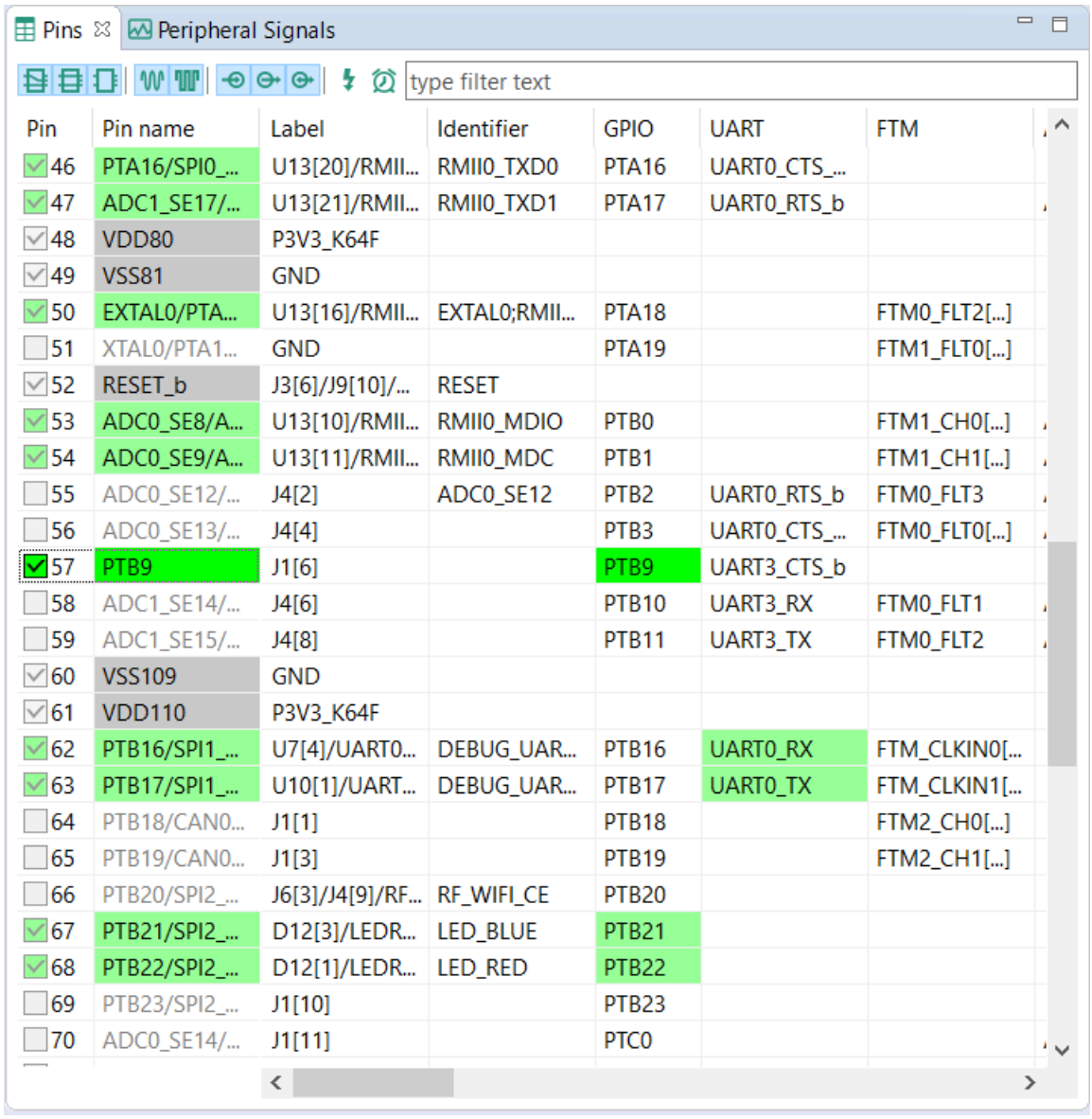


Figure 32. Configuring signals in the Pins view

2. In the **Routing Details** view, select the **Output** direction for the TX and PTB20 signals.

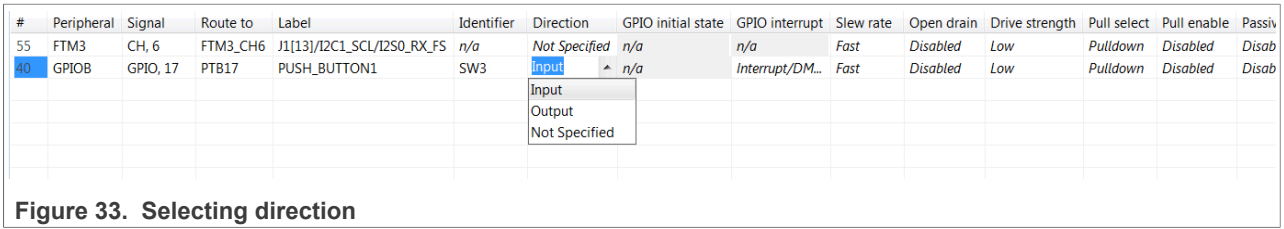


Figure 33. Selecting direction

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.

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

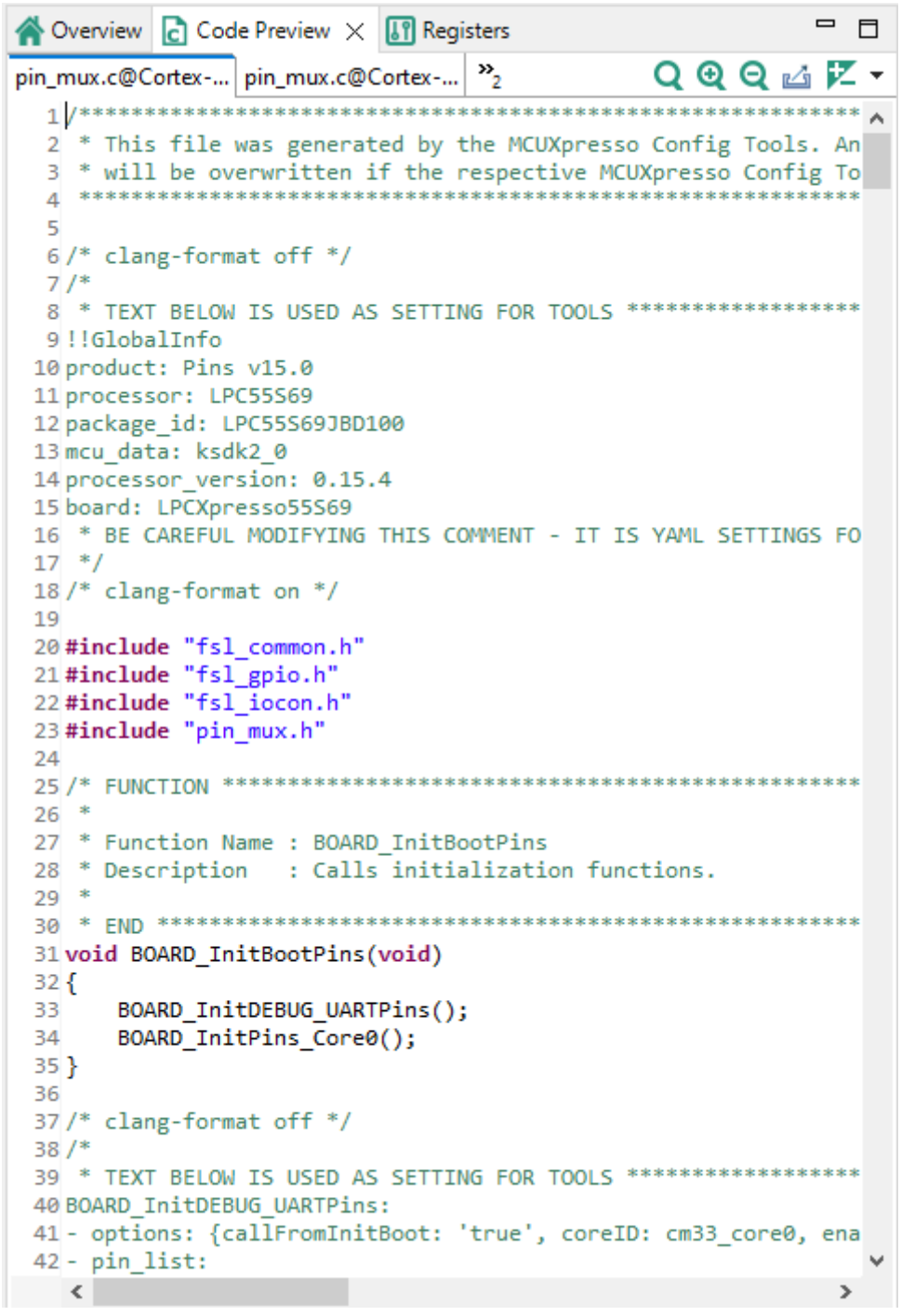


Figure 34. Generated code

2. You can now copy-paste the content of the source(s) to your application and IDE. Alternatively, you can export the generated files or update the code with the **Update Code** button in the **Toolbar**. To export the files, select **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 that you want to export sources for and select the **Export Source Files** option. **Export**, select the **Export Source Files** option.

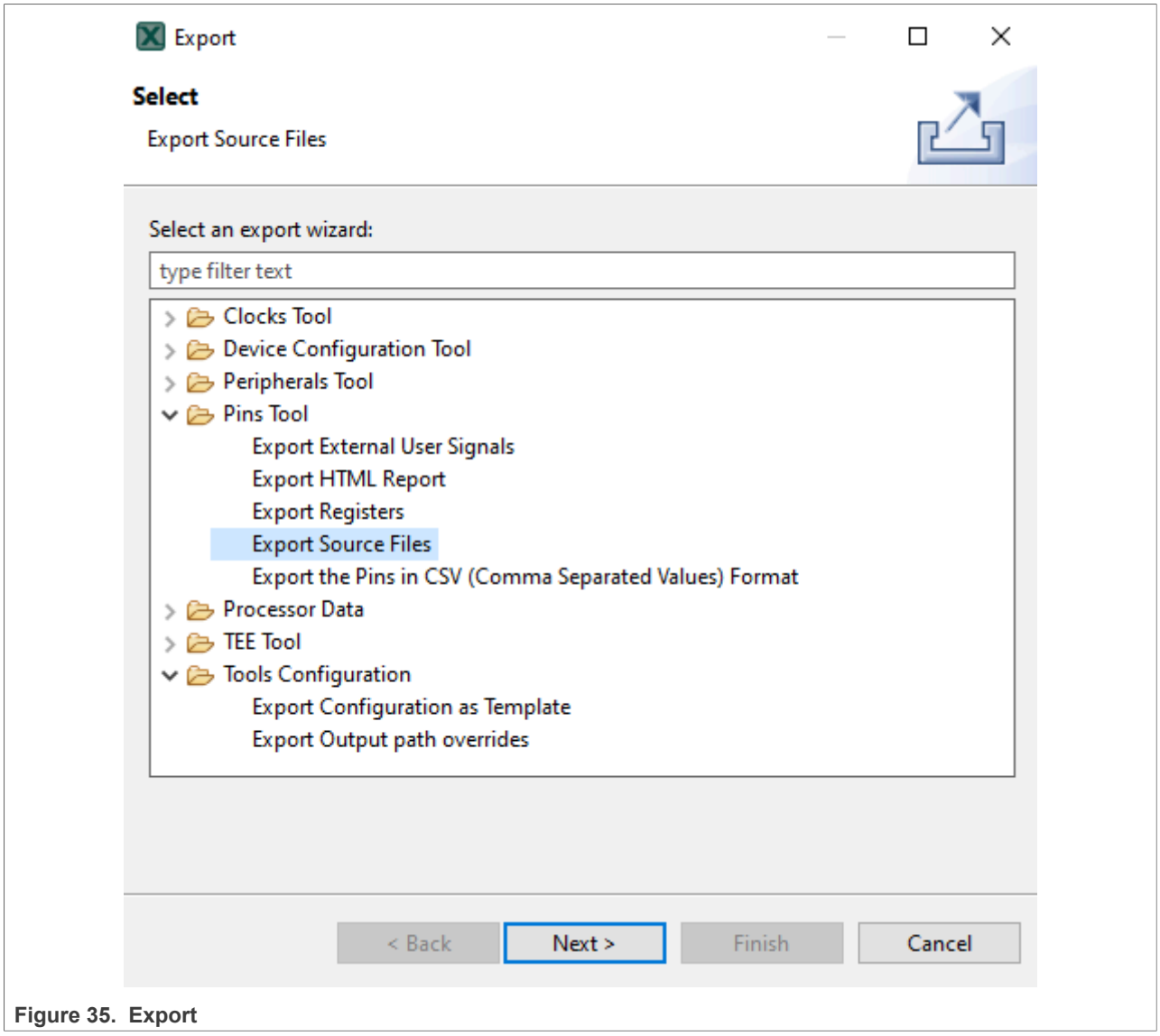


Figure 35. Export

3. 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).
4. Click **Finish** to export the files.
5. Integrate and use the exported files in your application as source files.

3.3 User interface

The Pins tool consists of several views.

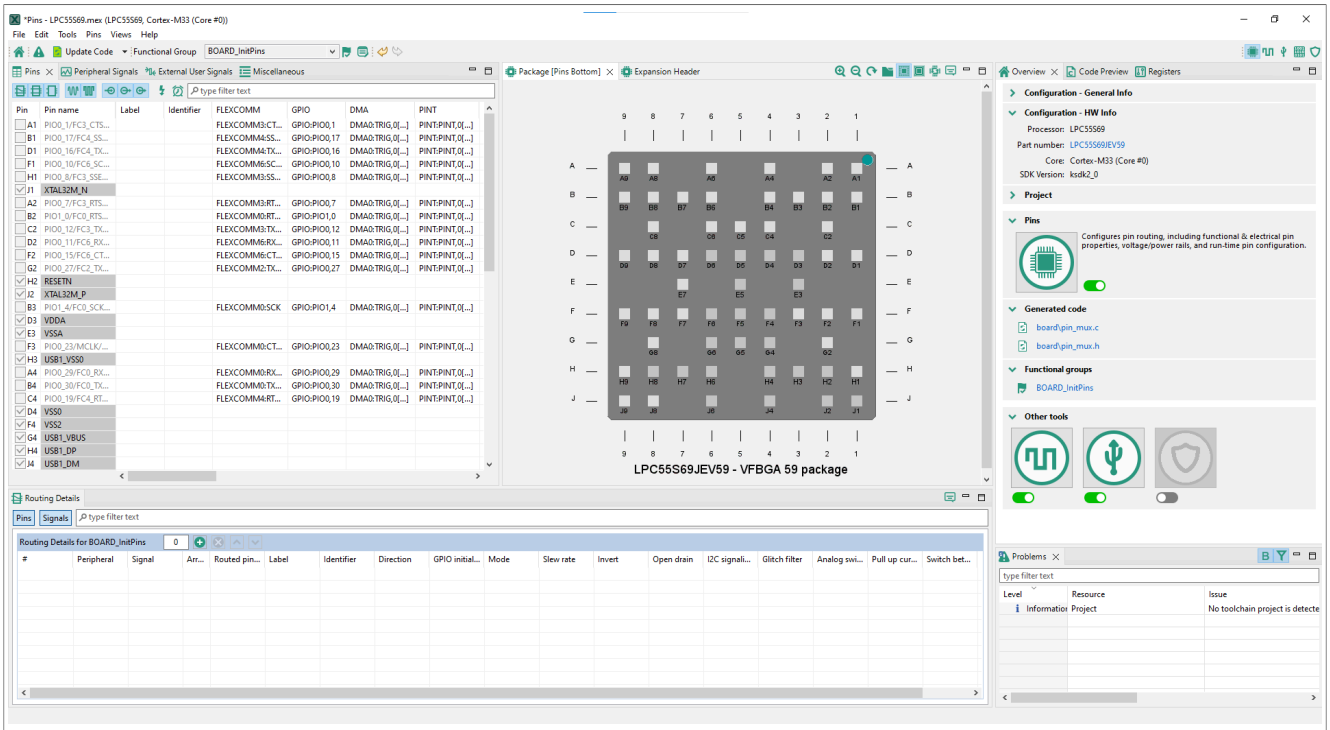


Figure 36. Pins tool user interface

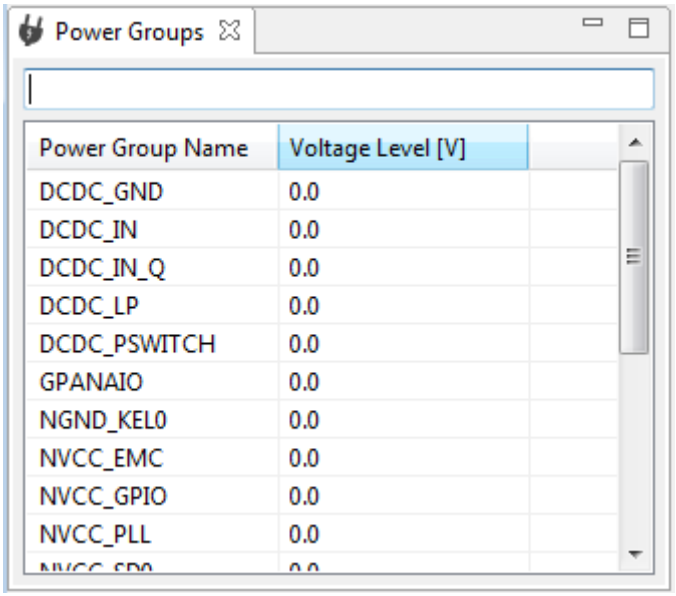
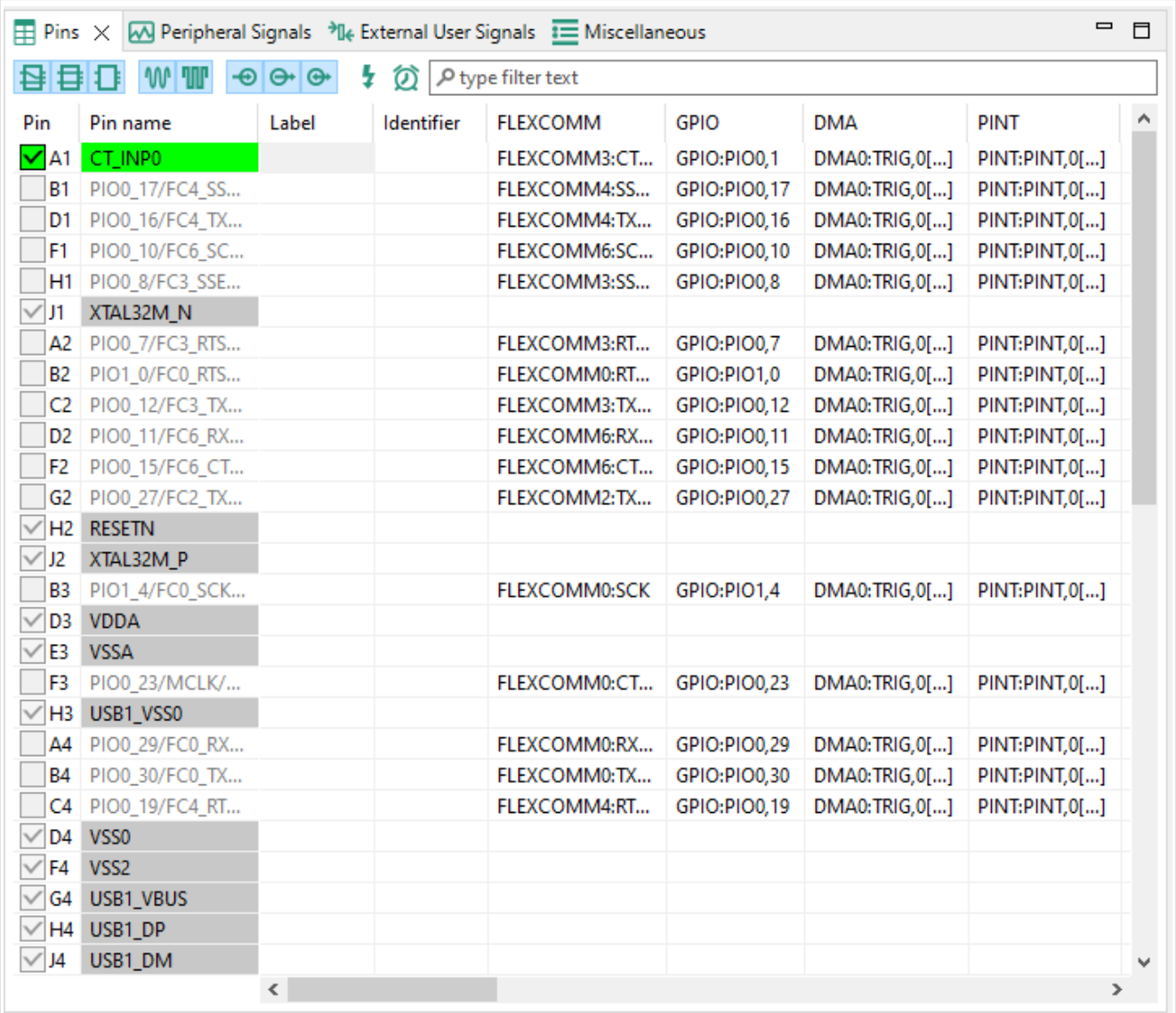


Figure 37. Selecting power group

Note: Power Groups are not supported for all processors.

3.3.1 Pins view

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



Pin	Pin name	Label	Identifier	FLEXCOMM	GPIO	DMA	PINT
A1	CT_INP0			FLEXCOMM3:CT...	GPIO:PIO0,1	DMA0:TRIG,0[...]	PINT:PINT,0[...]
B1	PIO0_17/FC4_SS...			FLEXCOMM4:SS...	GPIO:PIO0,17	DMA0:TRIG,0[...]	PINT:PINT,0[...]
D1	PIO0_16/FC4_TX...			FLEXCOMM4:TX...	GPIO:PIO0,16	DMA0:TRIG,0[...]	PINT:PINT,0[...]
F1	PIO0_10/FC6_SC...			FLEXCOMM6:SC...	GPIO:PIO0,10	DMA0:TRIG,0[...]	PINT:PINT,0[...]
H1	PIO0_8/FC3_SSE...			FLEXCOMM3:SS...	GPIO:PIO0,8	DMA0:TRIG,0[...]	PINT:PINT,0[...]
J1	XTAL32M_N						
A2	PIO0_7/FC3_RTS...			FLEXCOMM3:RT...	GPIO:PIO0,7	DMA0:TRIG,0[...]	PINT:PINT,0[...]
B2	PIO1_0/FC0_RTS...			FLEXCOMM0:RT...	GPIO:PIO1,0	DMA0:TRIG,0[...]	PINT:PINT,0[...]
C2	PIO0_12/FC3_TX...			FLEXCOMM3:TX...	GPIO:PIO0,12	DMA0:TRIG,0[...]	PINT:PINT,0[...]
D2	PIO0_11/FC6_RX...			FLEXCOMM6:RX...	GPIO:PIO0,11	DMA0:TRIG,0[...]	PINT:PINT,0[...]
F2	PIO0_15/FC6_CT...			FLEXCOMM6:CT...	GPIO:PIO0,15	DMA0:TRIG,0[...]	PINT:PINT,0[...]
G2	PIO0_27/FC2_TX...			FLEXCOMM2:TX...	GPIO:PIO0,27	DMA0:TRIG,0[...]	PINT:PINT,0[...]
H2	RESETN						
J2	XTAL32M_P						
B3	PIO1_4/FC0_SCK...			FLEXCOMM0:SCK	GPIO:PIO1,4	DMA0:TRIG,0[...]	PINT:PINT,0[...]
D3	VDDA						
E3	VSSA						
F3	PIO0_23/MCLK/...			FLEXCOMM0:CT...	GPIO:PIO0,23	DMA0:TRIG,0[...]	PINT:PINT,0[...]
H3	USB1_VSS0						
A4	PIO0_29/FC0_RX...			FLEXCOMM0:RX...	GPIO:PIO0,29	DMA0:TRIG,0[...]	PINT:PINT,0[...]
B4	PIO0_30/FC0_TX...			FLEXCOMM0:TX...	GPIO:PIO0,30	DMA0:TRIG,0[...]	PINT:PINT,0[...]
C4	PIO0_19/FC4_RT...			FLEXCOMM4:RT...	GPIO:PIO0,19	DMA0:TRIG,0[...]	PINT:PINT,0[...]
D4	VSS0						
F4	VSS2						
G4	USB1_VBUS						
H4	USB1_DP						
J4	USB1_DM						

Figure 38. 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 next columns are optional. They are **Label**, **Identifier, External User Signals** and **Expansion header connections** (One column for each expansion header). The pin name is changed to show the appropriate function for the selected peripheral if routed. The next column of the table shows peripherals and signals and pin name(s) on a given peripheral. Peripherals with a 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.

Gray background color in the Pin name column indicates that the pin/peripheral is dedicated. It is routed by default and has no impact on the generated code.

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

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

Select a row to open a drop-down list 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.

3.3.2 Package

The **Package** view displays the processor package. The processor package provides an overview of the package including resource allocation.

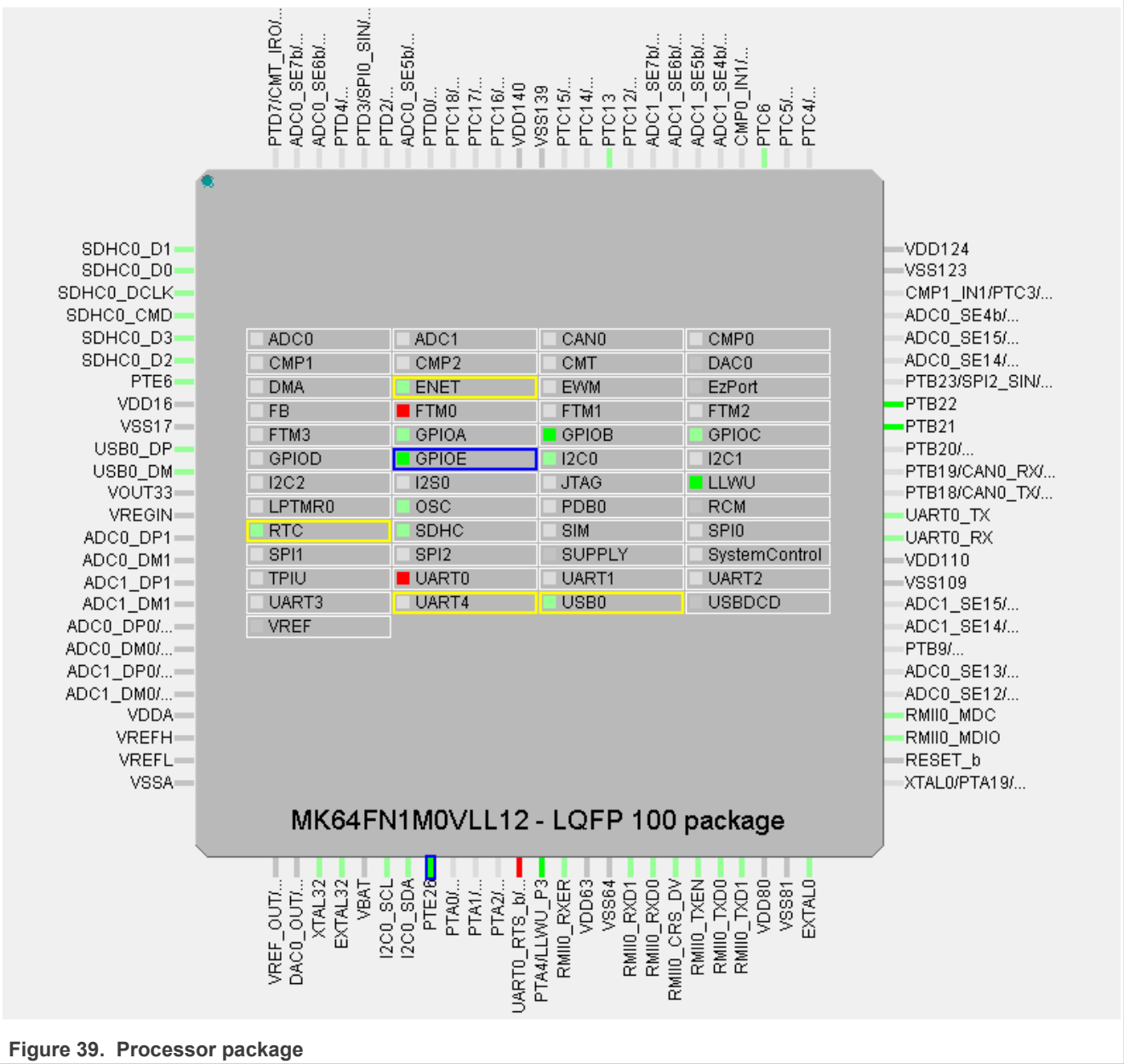


Figure 39. Processor package

This view shows a package overview with pin locations. The peripherals appear in the center.

To highlight the pin/peripheral configuration in the **Pins** and **Routing Details** views, right-click the pin or peripheral and select **Highlight**.

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:

3.3.2.1 Table Toolbar options

Table 13. Toolbar options

Icon	Description
	Zoom in package image.
	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.
	Select the information displayed as pin labels. This option is not available on BGA packages.

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

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

3.3.3 Peripheral Signals view

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

3.3.3.1 Table Status codes

Table 14. Status codes

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

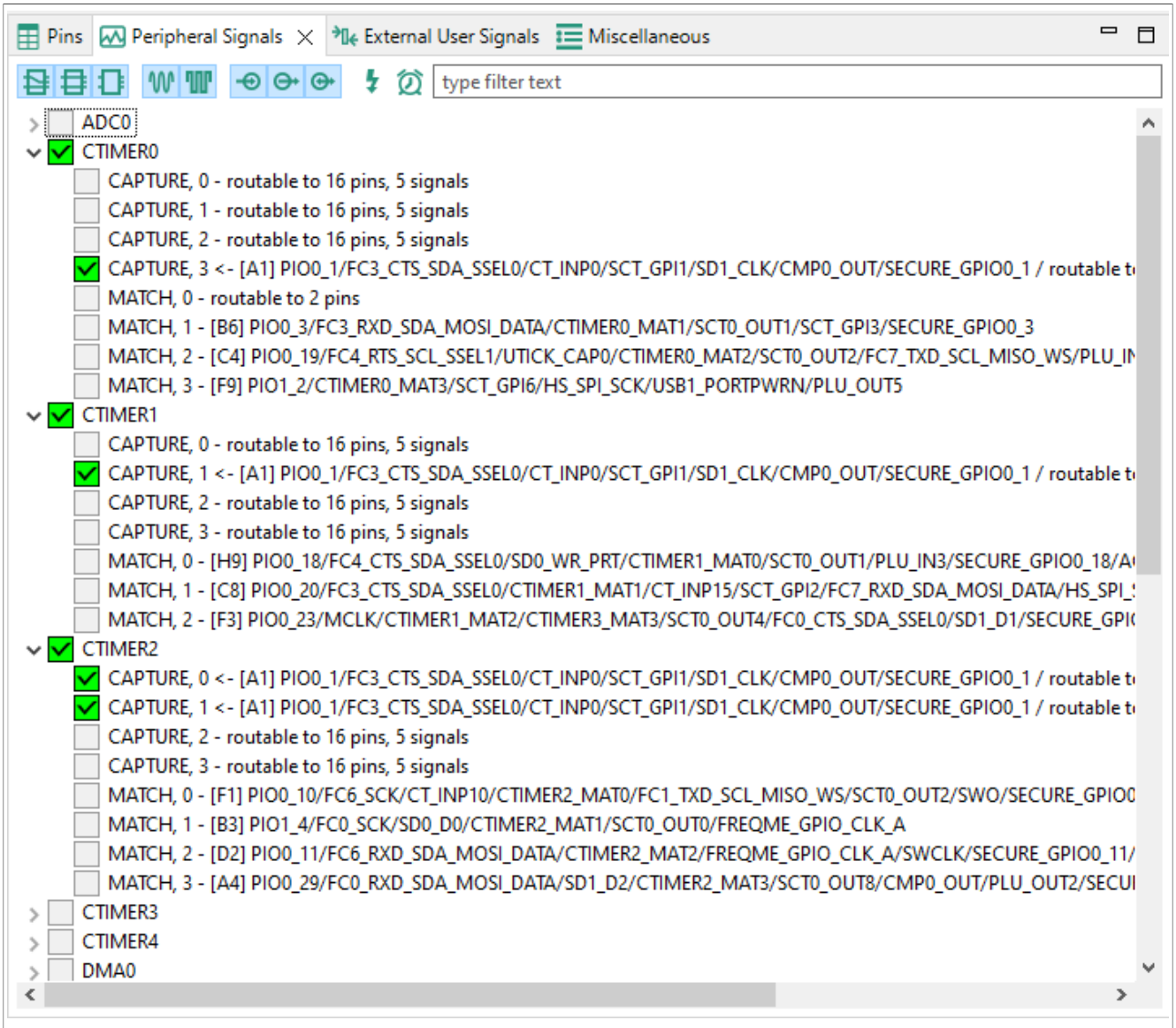


Figure 40. Peripheral Signals view

Use the checkbox to route/unroute the pins.

To highlight the pin/routing configuration about the peripheral in the **Package** and **Routing Details** views, right-click the signal and select **Highlight**.

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

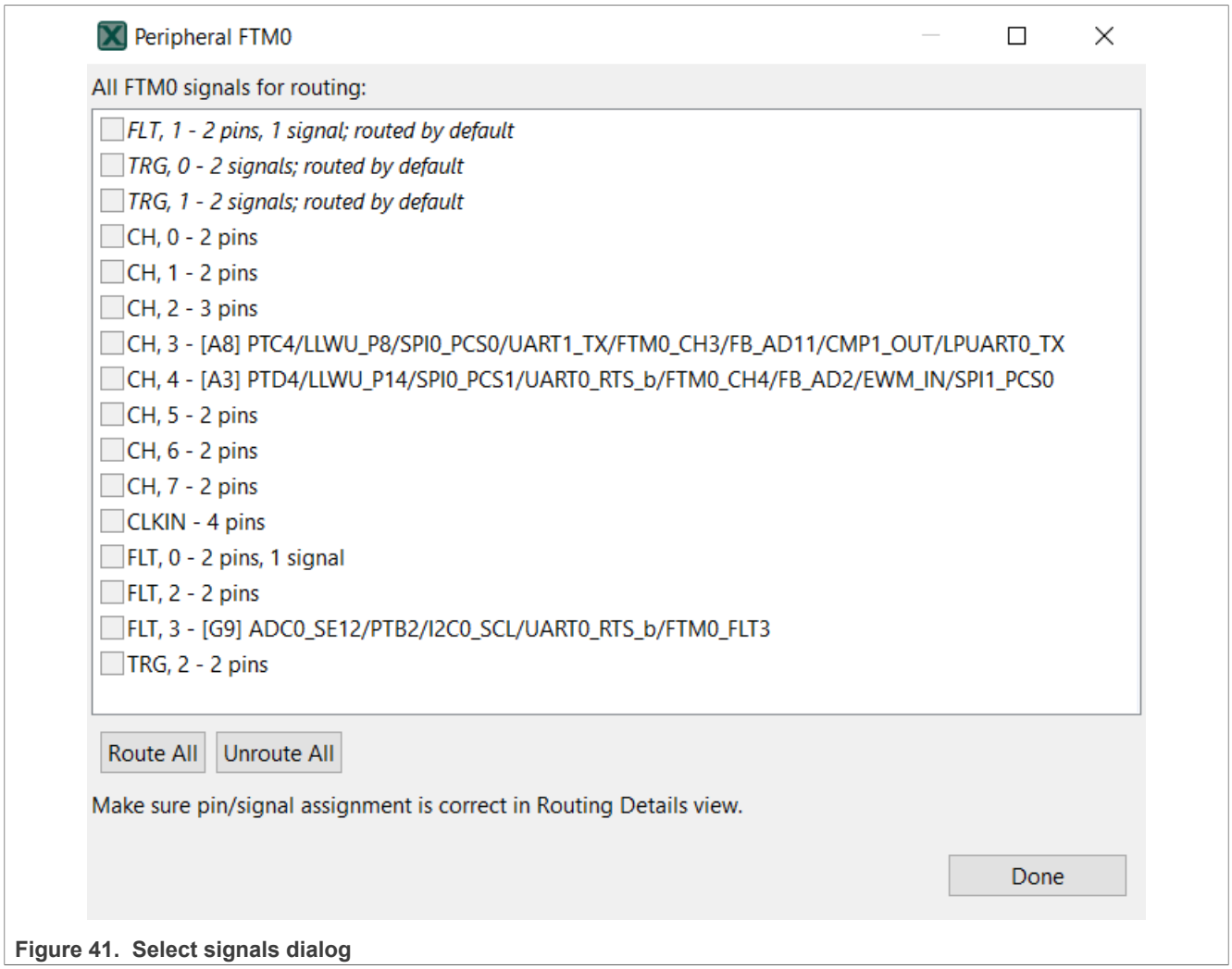


Figure 41. Select signals dialog

3.3.3.2 Filtering in the Pins and Peripheral Signals views

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

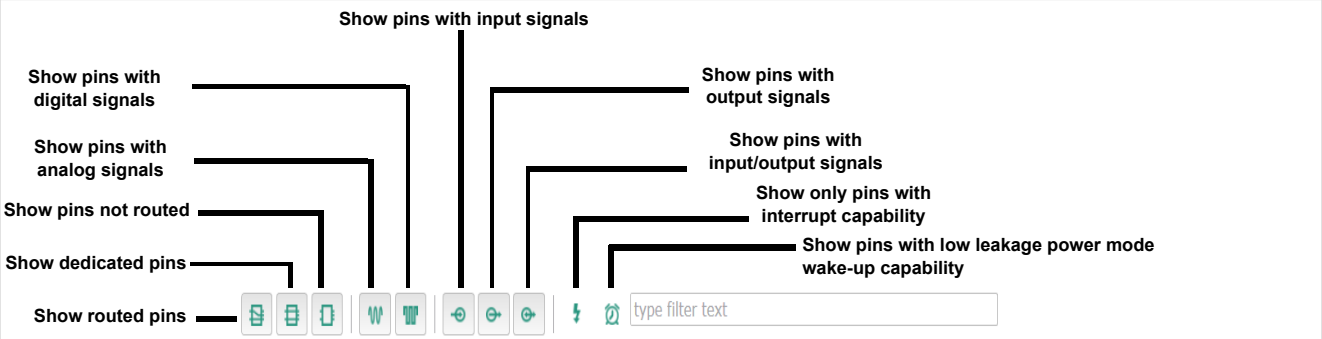


Figure 42. Filtering Controls

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

3.3.4 Routing Details view

In the **Routing Details** view, you can inspect and configure routed pins and internal signals. You can also configure the electrical properties of pins and view them. It displays the pad configuration available in a configuration where each pin is associated with the signal name and the function.

The table is empty when a new configuration is created, which means no pin is configured. Each row represents the 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.

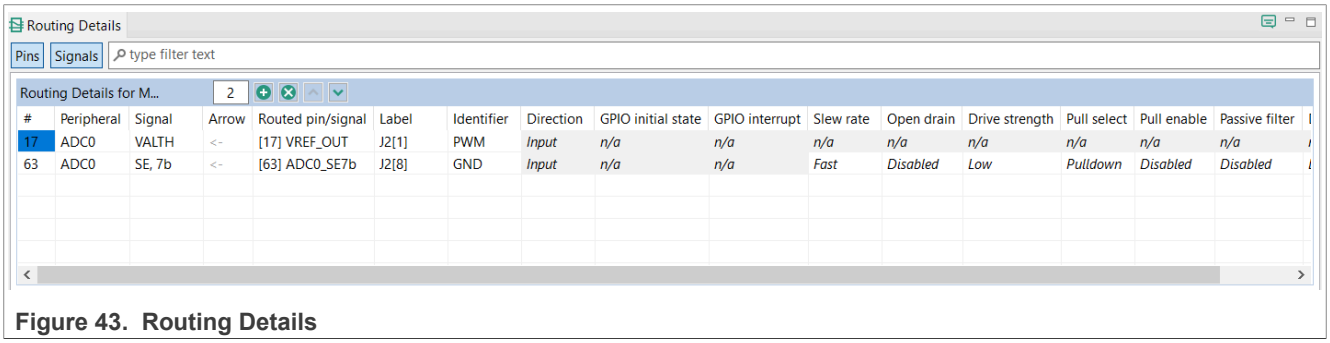


Figure 43. Routing Details

Add a row with the **Add new row** button in the view toolbar.

Configure the pin/signal by selecting the **Peripheral** first, then the required **Signal**, and finally, the pin to **Route to**.

Use the columns in the right side of the table to configure the electrical features.

You can also use the **Pins** and **Peripheral Signals** views to route pins and peripheral signals and view/modify the configuration in the **Routing Details** view. If the feature is not supported, *n/a* is displayed.

To highlight peripheral/pin information in the **Package** and **Pins** views, right-click the row and select **Highlight**.

To filter rows, type the text or the search phrase in the filter area in the view toolbar.

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

To display pins or signals only, use the **Pins** and **Signals** buttons in the view toolbar.

To add a row to the end of table, click the **Add new row** button.

To remove the selected row, click the **Delete the selected row** button.

To delete a specific row or insert a new row at a given position, right-click and use the dropdown list commands.

To add a specific number of rows, enter the number in the field.

To clear the table, type 0.

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

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

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

The gray background indicates read-only items.

3.3.4.1 Properties configured in Routing Details view

The properties of pins and internal signals that can be configured are shown in dedicated columns. The properties include:

- Common properties available for all pins and internal signals
  - **#** - Package pin number/coordinate
  - **Peripheral** - Name of the selected peripheral module
  - **Signal** - Name of the selected peripheral signal/signal function
  - **Arrow** - Arrow indicating input, output, input/output, or not specified direction of the signal
  - **Routed pin/signal** - Name of the pin or internal signal
  - **Label** - Pin label with max length of 128 characters; By submitting an empty label the identifier is deleted as well
  - **Identifier** - Pin identifier used for #define code generation
  - **Direction** - Pin direction
- General-Purpose Input Output (GPIO) properties available only for GPIO pins
  - **GPIO initial state** - GPIO output initial state. It is available only if pin direction is set to output.
  - **GPIO interrupt** - Configuration of interrupt request for the pin. It is available only if the pin direction is set to input.
- Processor-specific properties
  - Properties that may differ for different processors, for example configuration of pull ups, open drain, drive strength, slew rate, power groups

Tip: Click the **Routing Details Legend** button in the top-right corner of the view to display a dialog explaining all the properties and their values.

Generation of initialization code

The Pins tool generates initialization code only for pins and internal signals configured in the Routing Details view. Generally, initialization code is generated always if it is necessary for proper routing of the pin or internal signal to selected peripheral signal. On the other hand, the code related to the direction, GPIO functionality or processor-specific properties is automatically optimized out in the following cases:

- The user does not explicitly select a value of a property. It is signalized by the italic font of the value. It is the default state after adding a pin or internal signal to the Routing Details view. The displayed value shows either the value set by another configuration of the same pin in the same function or in a function called from the default initialization function. When there is no such configuration, the displayed value matches the after-reset state. To generate the code even if the value is the same as the default value, select the value from the drop-down menu: the value is shown with normal font and code is generated. To return to the default value select “No init” from the drop-down menu: the value is again shown with italic font and no code is generated.

- A “Not specified” value is selected in case of the direction property.

Additionally, it is possible to force generation of full initialization code of all properties using “Full pins initialization option” in Functional groups. If this option is enabled, it is not possible to use the “No init” value from the drop-down menu and the initialization code is generated unless the “Not specified” value is selected (for direction only). For more information about the “Full pins initialization option” option, refer to the Functional group properties section in this documentation.

Validation of routing

The Pins tool automatically performs validation of the selected properties. An error is shown for a property in the Routing Details view in the following cases:

-The value of the property conflicts with the value of another property. Typically, conflicts arise when two pins or internal signals configure the same register and bitfield with different initialization values. To resolve the conflict, the configuration of one of the pins or internal signals must be changed.

- There is no after-reset value specified for the property. In this case the error indicates that the user has to select the value of the property or “Not specified” to indicate that the property can be ignored by the tool.
- The property has to be set by the user. In this case, the “No init” or “Not specified” are not available and the user must decide what value will be used. A typical use case is when routing of the internal signal would not be complete without such selection or when it is necessary to confirm by the user the intended usage of the pin or signal.

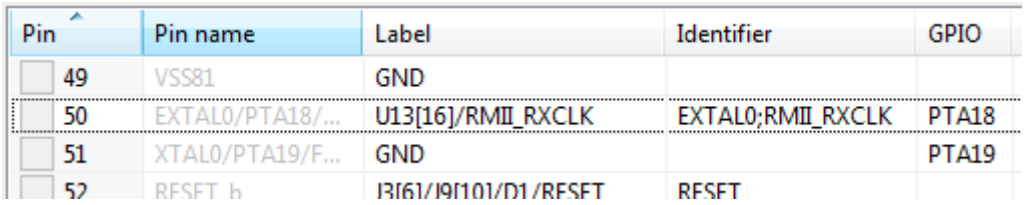
3.3.4.2 Labels and identifiers

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

Boards and kits have predefined labels. However, it is also possible to define a pin label listed in the **Pins** and **Routing Details** views.

To set or update the **Labels and Identifier** column visibility, select **Window > Preferences > MCUXpresso Config Tools** from the **Menu bar**, and select the **Show pin label > identifier table columns (Pins tool)** checkbox.

The pin identifier is used to generate the `#define` in the pin_mux.h file. However, it is an optional parameter. If the parameter is not defined, the code for `#define` is not generated. Also, 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 **Routing Details** 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	RESET_b	J3[6]/J9[10]/D1/RESET	RESET	

Figure 44. Pin identifier

In this case, it is possible to select from values if the pin is routed. See Routing Details

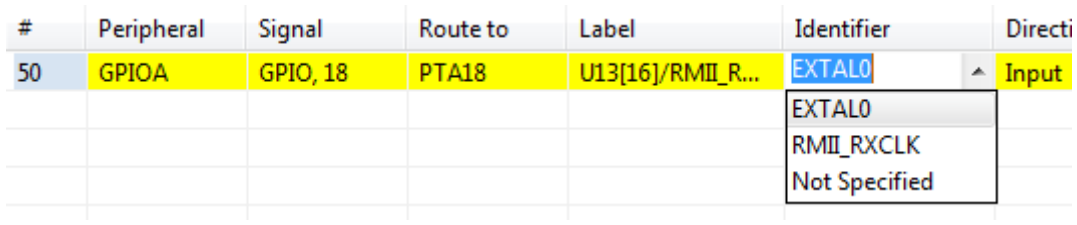


Figure 45. Identifier in Routing Details table

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 in Labels and identifiers.

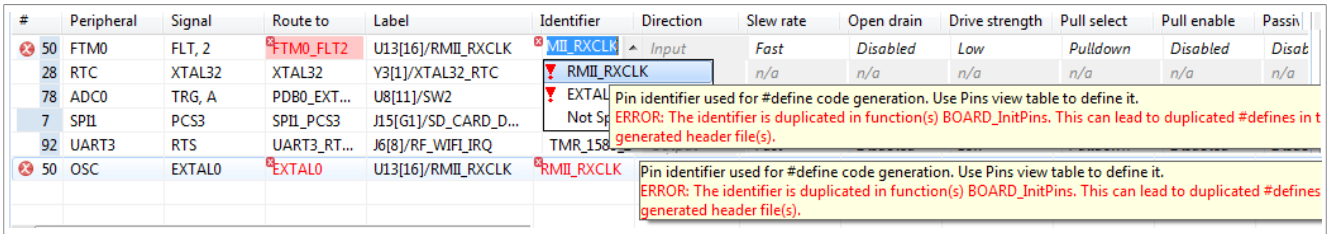


Figure 46. Identifier errors

By typing text into the Identifier column, you can define a new identifier for the pin. It is automatically used only in the selected routing. Available identifiers can be revised in a dialog invoked from the context menu or in the Pins view.

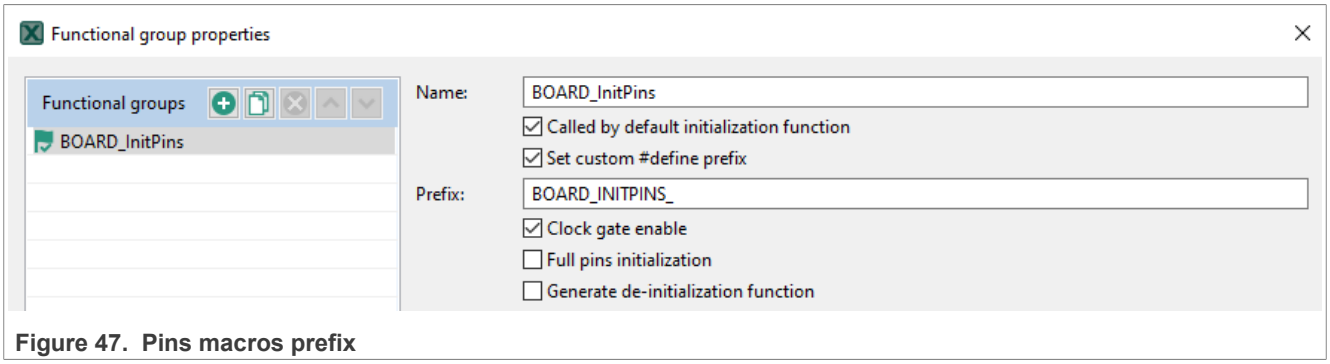


Figure 47. Pins macros prefix

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 a 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.

3.3.5 Expansion header

In the **Expansion Header** view, you can add and modify an expansion header configuration, map the connectors, and route the pin signals. You can also import and apply an expansion board to the header.

Certain boards, such as LPCXpresso55S69, come with preconfigured expansion headers.

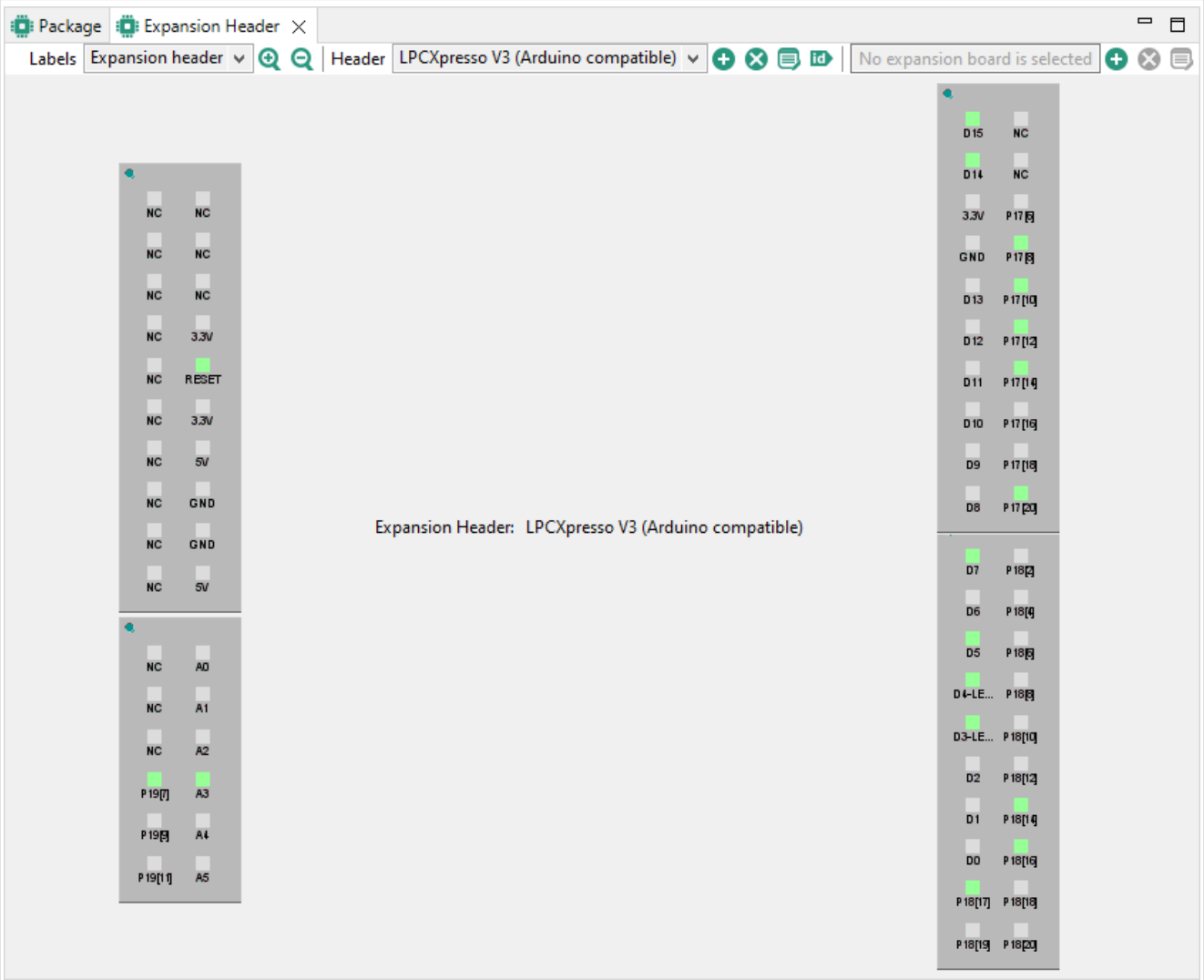


Figure 48. Expansion header

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 **Window > Show view > Expansion Header** from the **Main menu**.
2. Add a header by selecting the **Add** button in the view toolbar.
3. In the **Add New Expansion Header** window, select the **Header type** from the drop-down list.

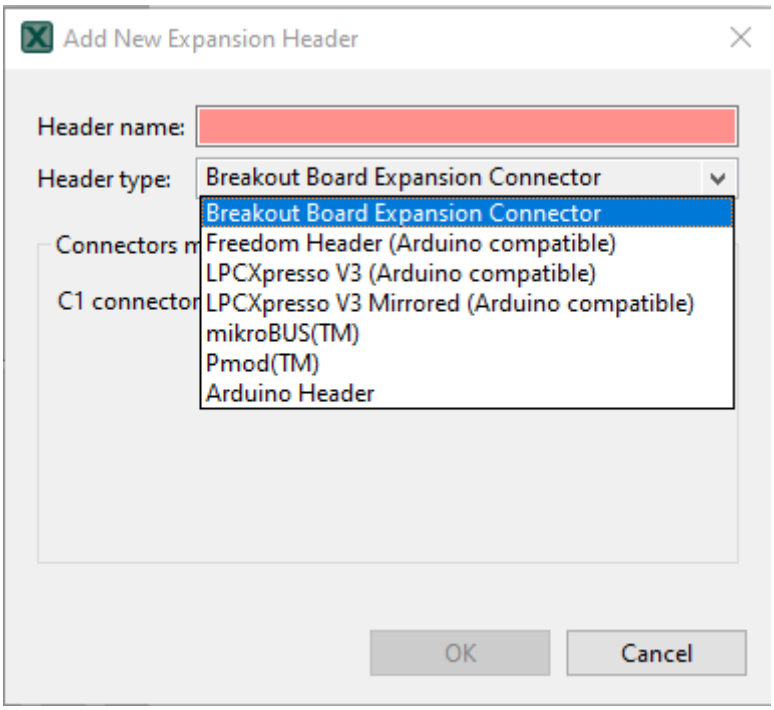


Figure 49. Adding new expansion header

4. Name the header and map the connectors.

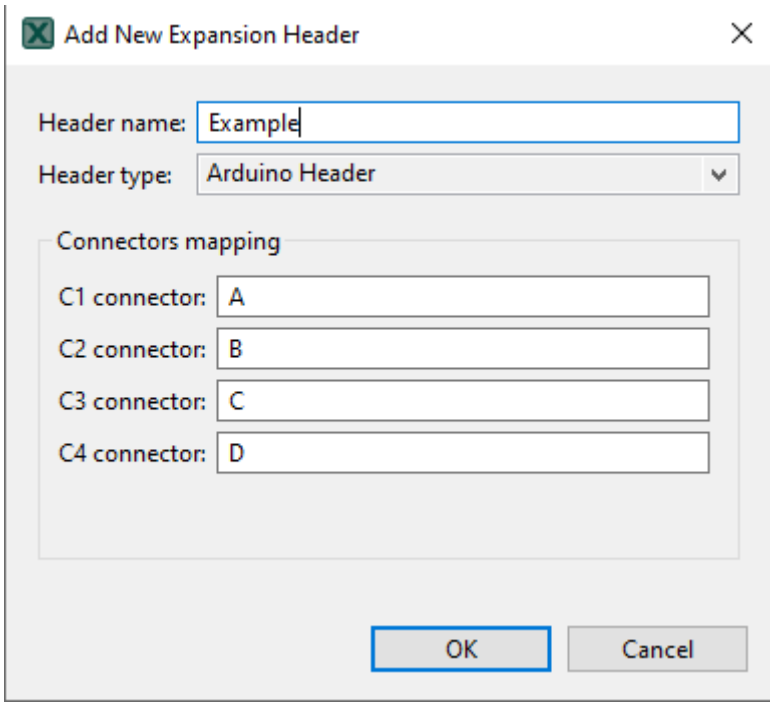


Figure 50. Adding new expansion header

5. Select **OK**.
   Expansion Header view now displays the connector layout. You can point your cursor over the pins to display additional information. Right-click the pin to display a shortcut menu of additional options.

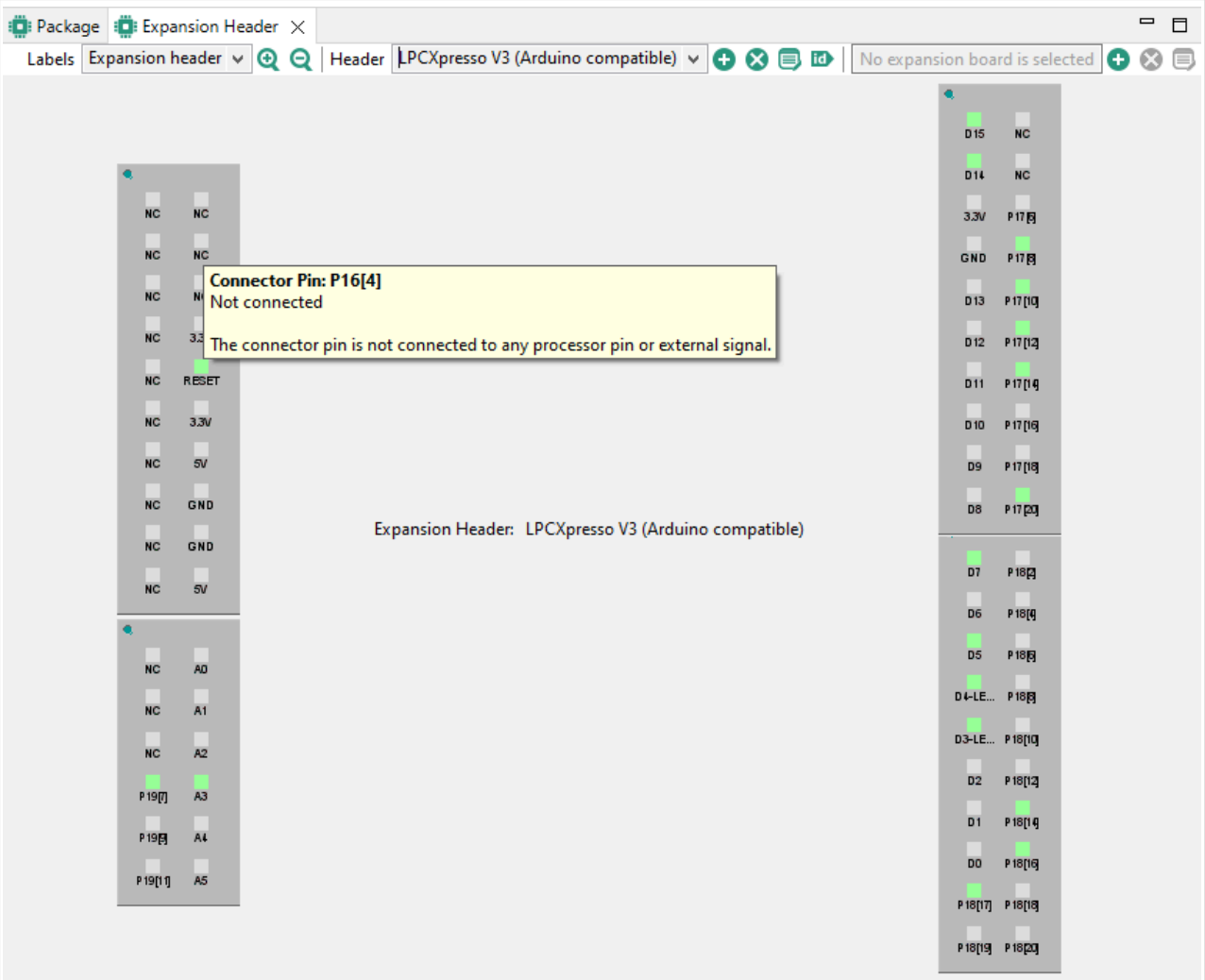


Figure 51. Expansion header

6. To map the header pin to processor pin, right-click the header pin and select **Connect**.
7. In the **Connector Pin** dialog, select the processor pin/external signal from the list and click **OK**.

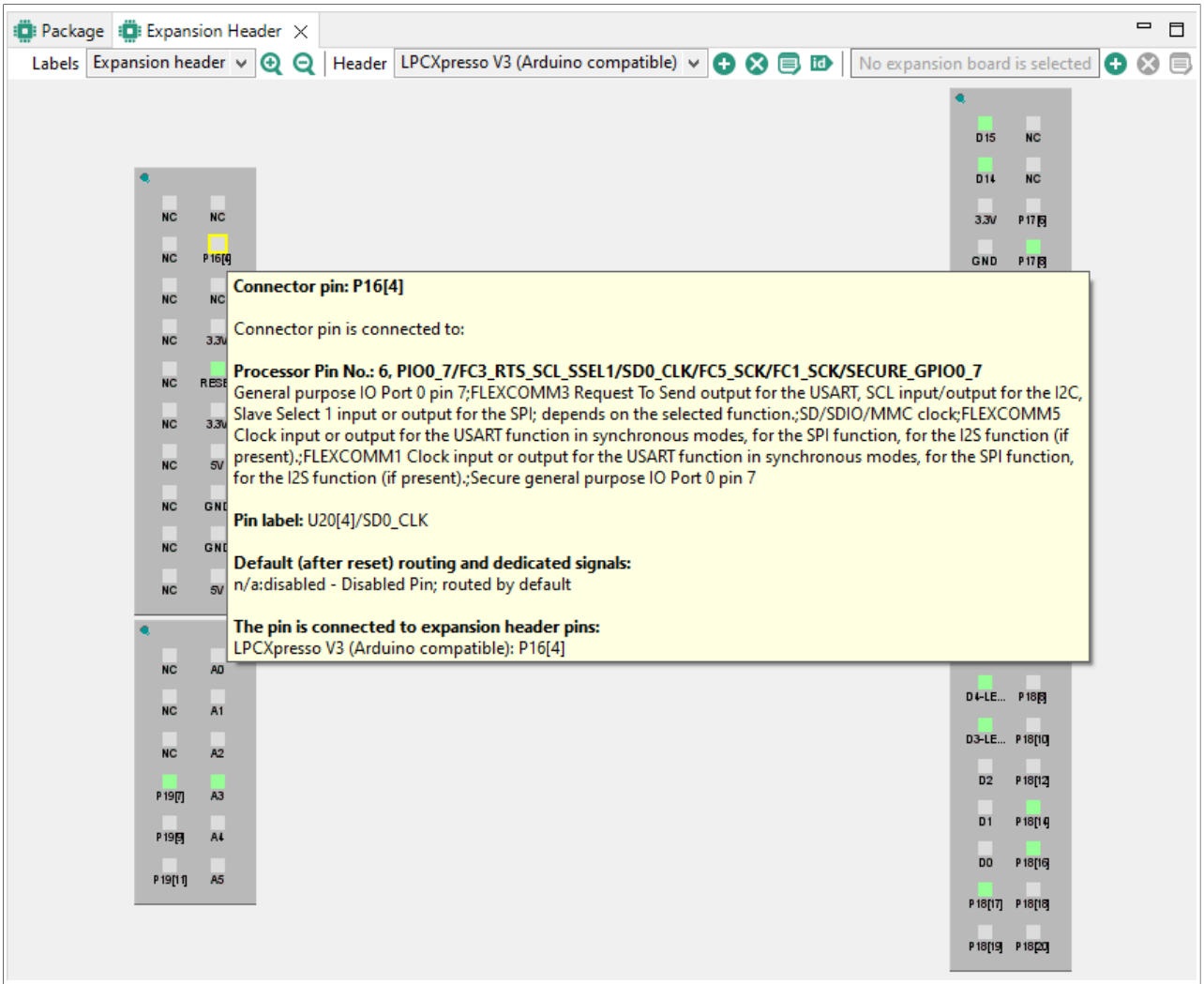


Figure 52. Connected pin

8. To route the pin, right-click the header pin and select **Route**.
9. In the **Pin** dialog, select the signal from the list and click **OK**.
   The connector pin is now routed.

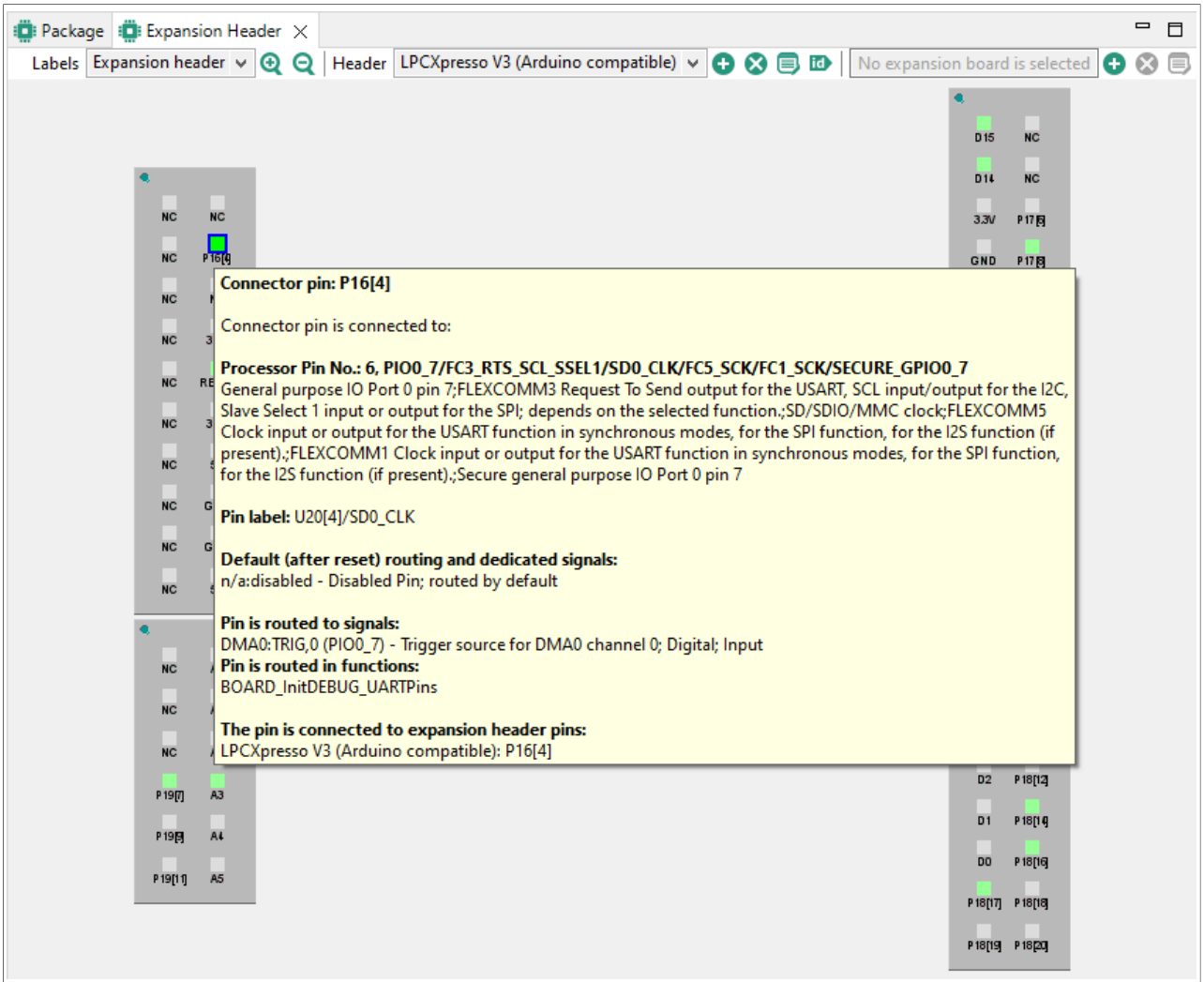


Figure 53. Routed pin

You can create more than one expansion header configuration. Switch between the configurations in the view's drop-down list.
To highlight the pin/routing configuration in the **Pins** and **Routing Details** views, right-click the connector pin and select **Highlight**.
Modify the configuration parameters at any time by selecting the **Edit** button. Information in the **Pins** view is updated automatically. Pin connections between the header and the processor and their labels can be locked to prevent any modifications.
Remove a configuration by selecting the **Remove** button.
Use the **Label** drop-down list to switch between display information for header, board, and routing.
The **ID** button allows setting expansion header pin labels as processor pin identifiers. Upon user selection, the new identifiers can be also explicitly selected.

3.3.5.1 Expansion Board

In the **Expansion Header** view, you can also apply an expansion board to an already created expansion header. The expansion board configuration can be imported into Pins tool in the form of an XML file. Based on the chosen processor, the tool will then recommend adequate routing.

Note: Only a single expansion board can be configured per expansion header.

1. In the **Expansion Header** view, click the **Apply expansion board to the selected header**. Alternatively, select **Pins > Apply expansion board** from the **Menu bar**.
2. In the **Apply expansion board** dialog, click **Browse** to locate the XML file with expansion board information and click **OK**.

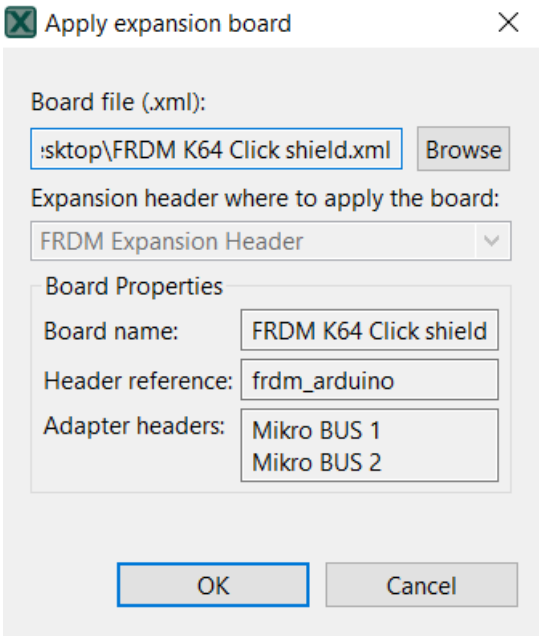


Figure 54. Apply expansion board

3. Click **OK** to apply the expansion board.
4. On the next page, choose if you want to create a new functional group for the expansion board, or modify an existing functional group. In the latter case, use the dropdown list to select from available functional groups.
5. In the **Expansion Board Routing** table, inspect the suggested routing of expansion board pins. If you want to change the route of a pin, click the pin cell in the **Route** column and select the signal in the **Connector pin** dialog and click **Done**.

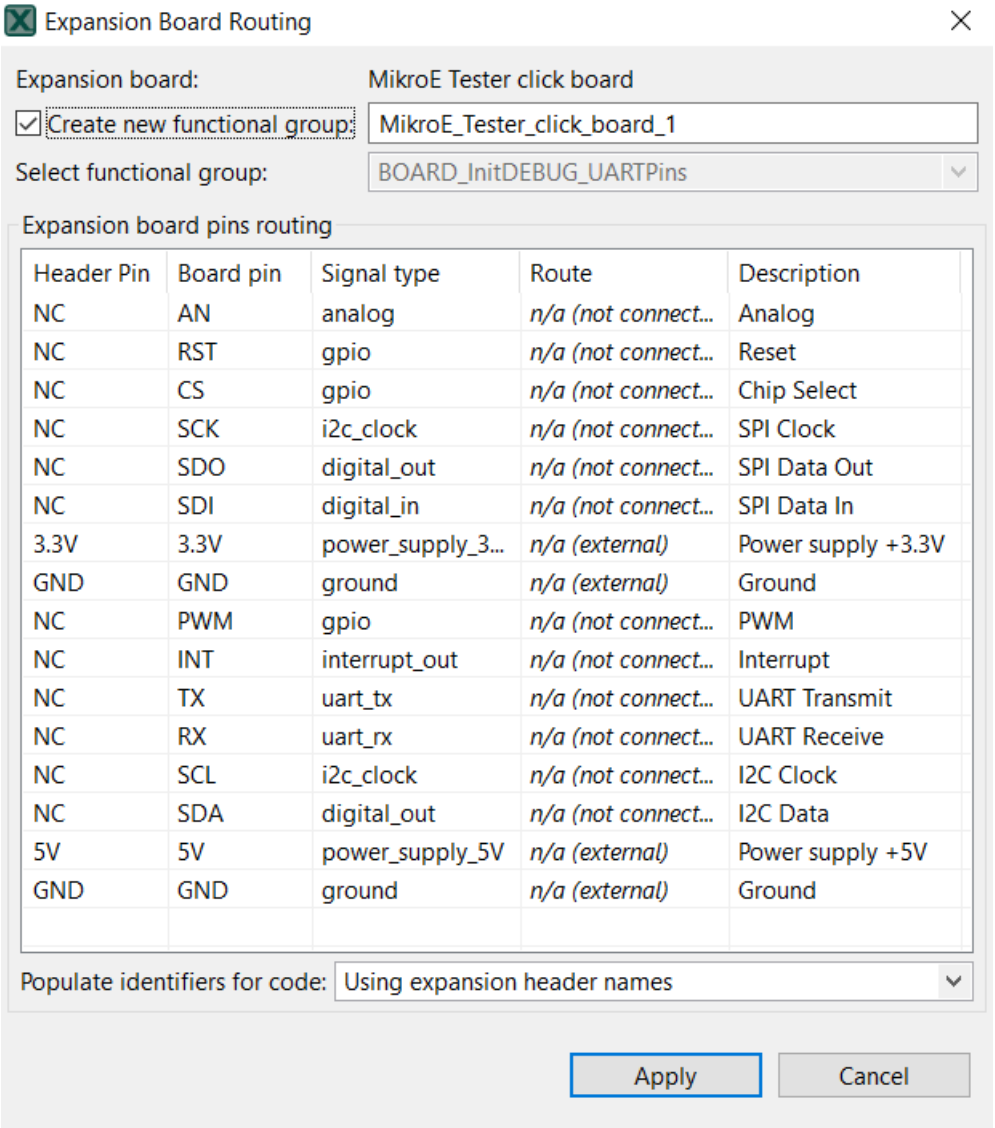


Figure 55. Expansion board routing

6. Choose how you want to populate identifiers for code. Following options are available:
   - Expansion header names
   - Expansion board names
   - None
7. Click **Apply** to apply the settings.
   You can change the expansion board signal routing at any time by clicking the **Configure routing for expansion board** button in the **Expansion Header** view.

3.3.6 Miscellaneous view

This view contains Generate extended information into the header file (previously located in Configuration preferences) and possible processor specific settings.

When the Generate extended information into the header file option is selected, the extended information is generated into the header file. For projects created in earlier MCUXpresso versions, this option is selected by default.

When option Automatically select property value for a new routing is set, the after-reset state is explicitly set for every electrical property of a new routing.

3.3.7 Functions

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

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

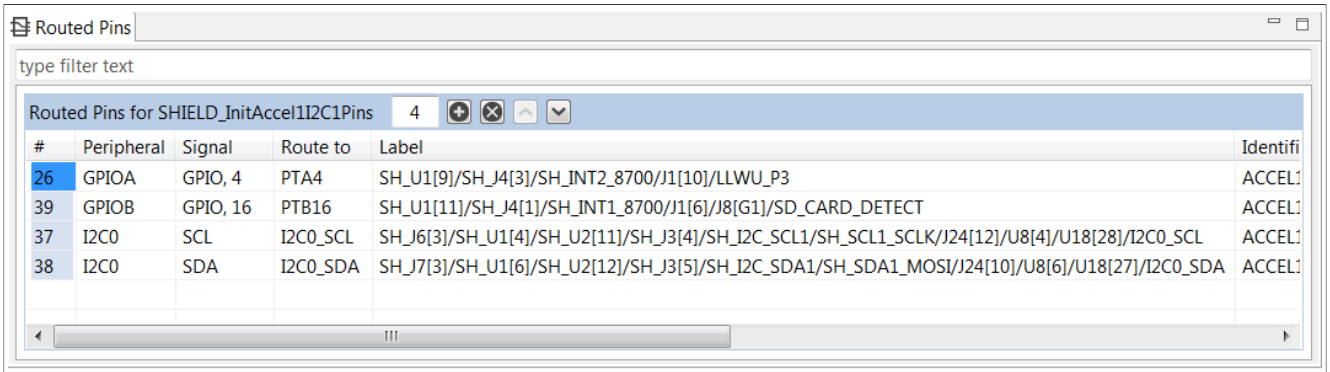


Figure 56. Routed Pins view

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.

3.3.8 Highlighting and color coding

You can 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 gray 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.

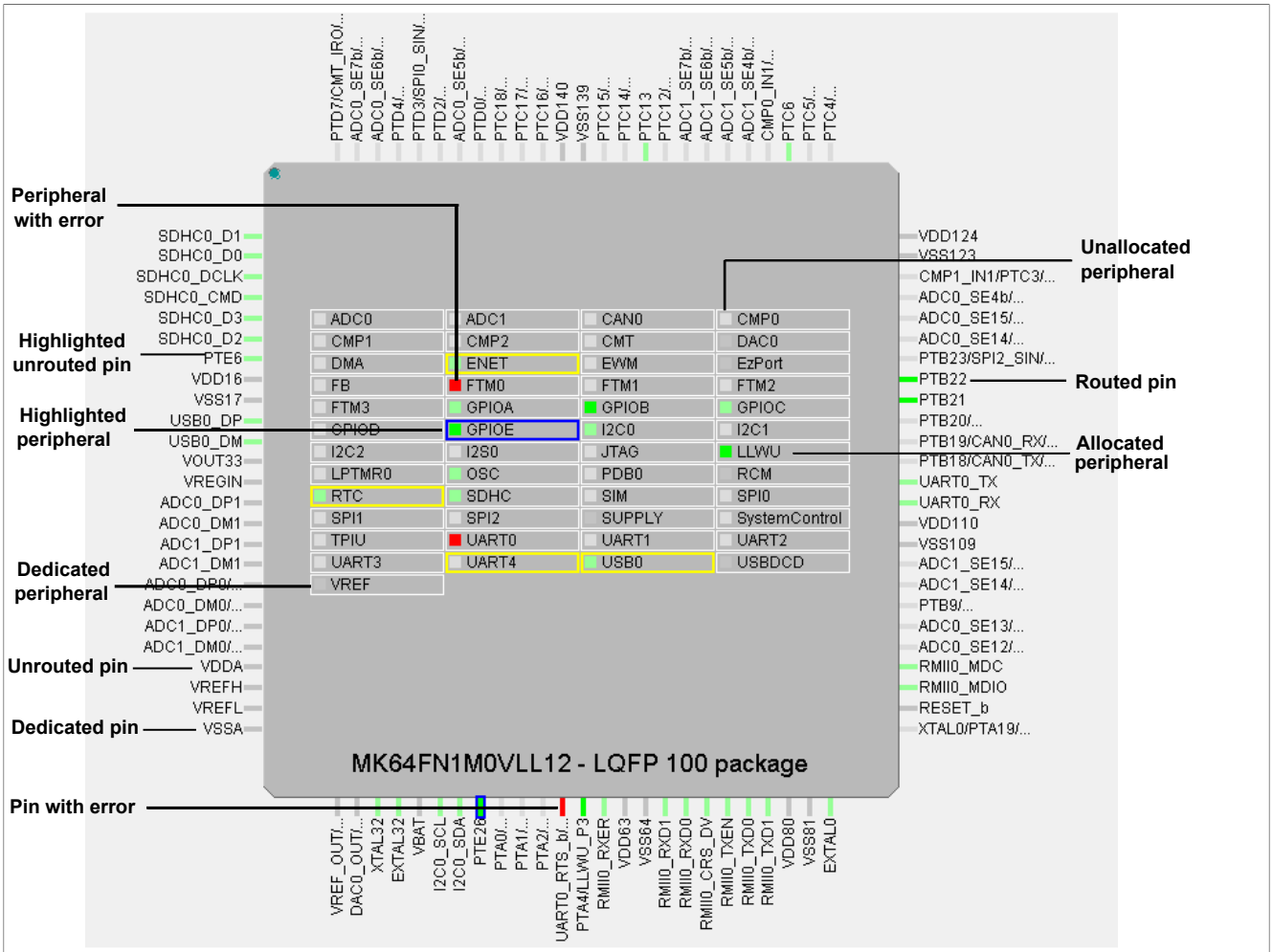


Figure 57. Highlighting and color coding

#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Open drain	Drive strenc
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 58. 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 59. Warnings

- **Package** view
  - Click 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 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.

3.3.8.1 External User Signals view

This view allows the user to define a custom description of the signals. An External User Signal has a defined unique ID within the table, pins to which it is connected, and any amount of additional text information. All of it can be customized.

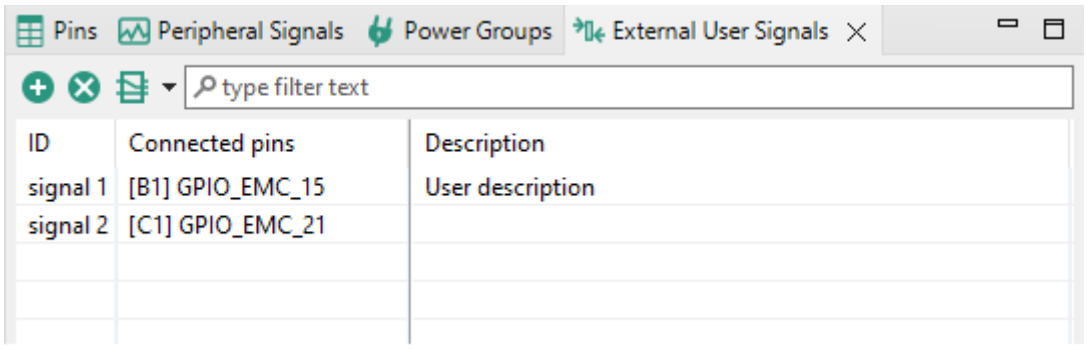


Figure 60. External User Signals view

Connecting to a pin(s) can be done from a context menu of the selected signal. Multiple pins can be connected to the signal as well as multiple signals can be connected to the pin. When some signals are defined, the External User Signals column is added to the **Pins** view. The connection between pins and signals can be also done from there.

Additional columns can be specified using the table header context menu.

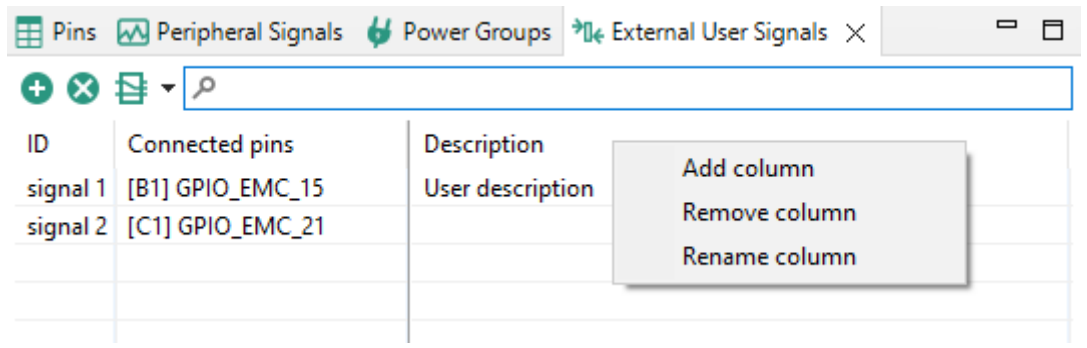


Figure 61. External User Signals header context menu

The button with the **Routing Details** view icon can be used to add routed pins to the table or to display columns from **Routing Details** view in the **External User Signals** view.

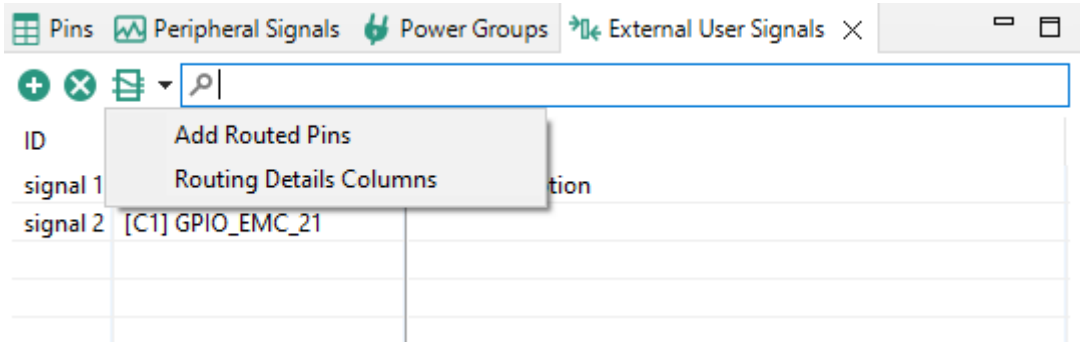


Figure 62. External User Signals header context menu

The **Add Routed Pins** option collects routed pins from all functional groups and adds them to the table when they are not already present. A new signal is created for each added pin.

Select Routing Details Columns to open a dialog with Routing Details columns. The selected columns will be displayed in the table. Those columns are read-only and they always reflect actual values in the **Routing Details** view for all functional groups.

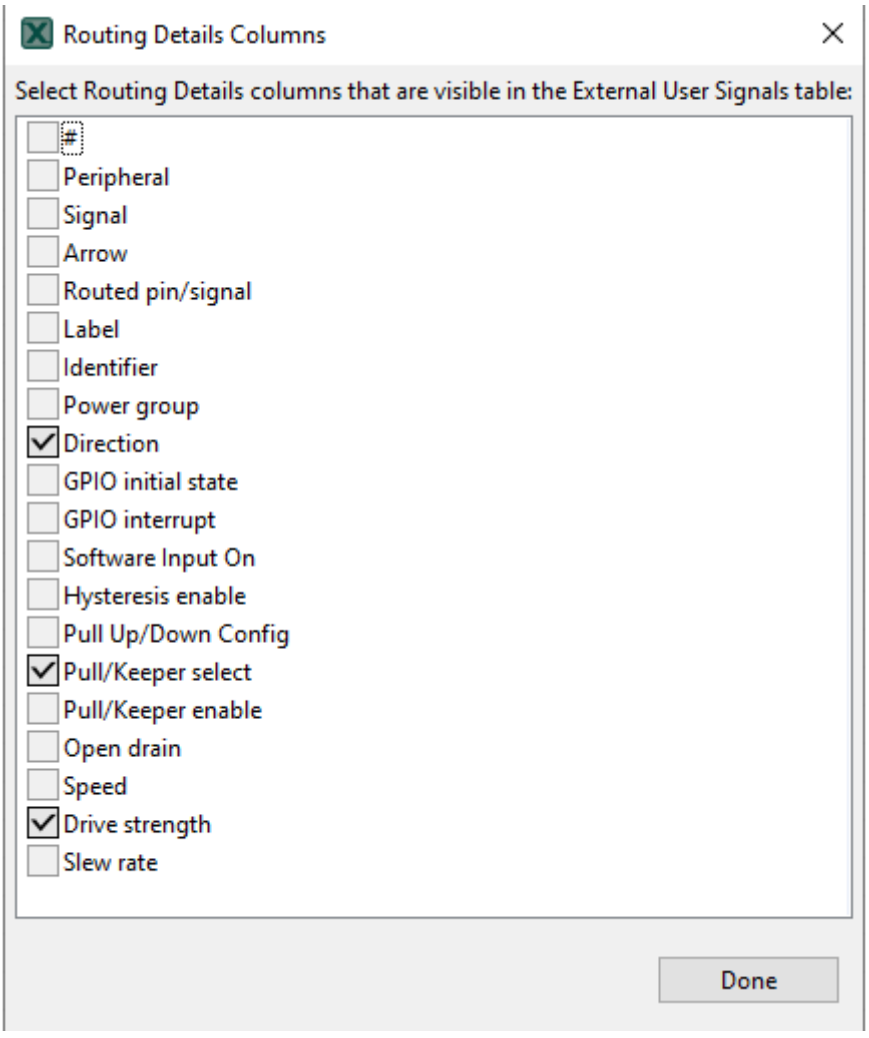


Figure 63. Routing Details columns dialog

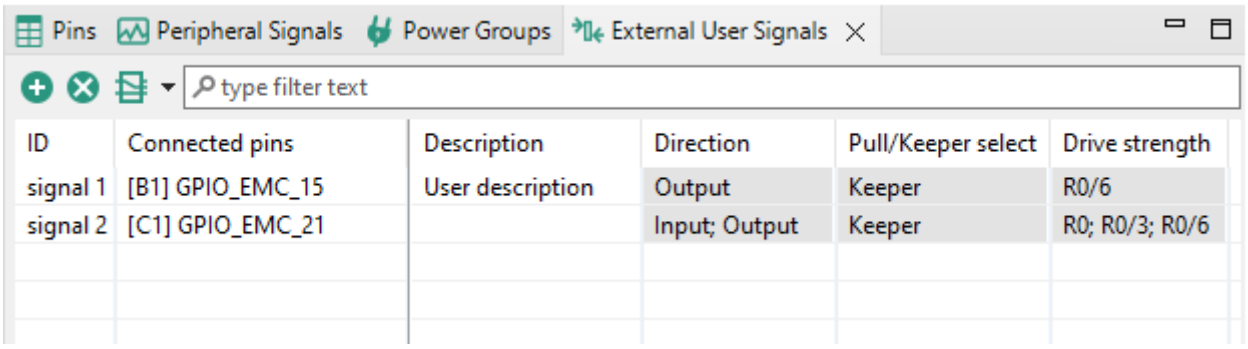


Figure 64. Routing Details table

When needed, External User Signals can be also exported to CSV and then imported to another configuration. Merging of signals is not supported so when some signals are defined for the configuration, they are replaced by imported signals.

3.4 Errors and warnings

The Pins Tool checks for any conflict in the routing and also for errors in the configuration. Routing conflicts are checked across all **INIT** functions (default initialization functions). It is possible to configure different routing of one pin in different functions (not INIT functions) to allow dynamic pins routing reconfiguration.

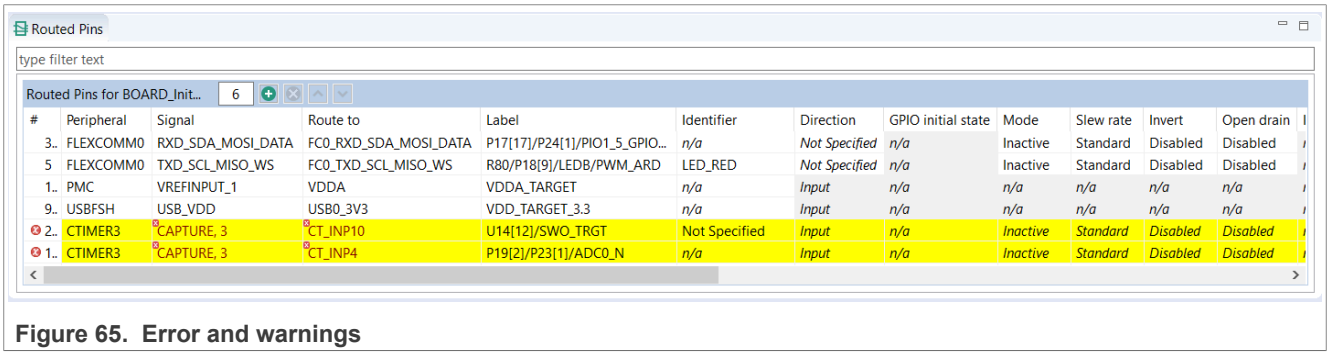


Figure 65. Error and warnings

If an error or warning is encountered, the conflict in the **Routing Details** 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 last two rows in the figure above show the peripheral/signal where the erroneous configuration occurs. The detailed error/warning message appears as a tooltip.

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

3.4.1 Incomplete routing

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

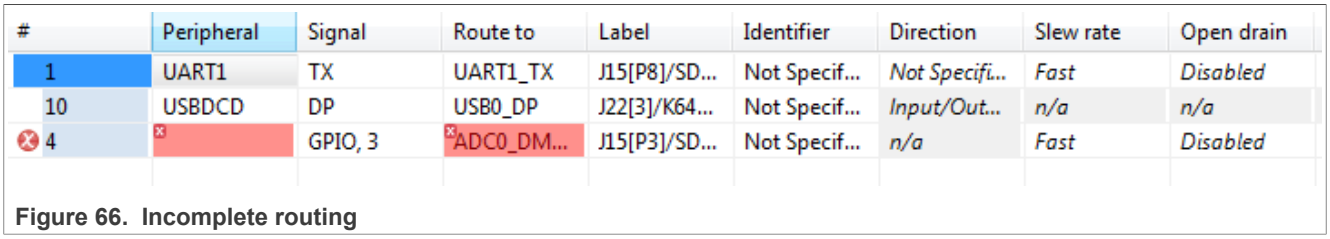


Figure 66. Incomplete routing

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.

3.5 Code generation

If the settings are correct and no error is reported, the code generation engine instantly regenerates 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. Such features as Copy, Search, Zoom-in, Zoom-out, and Export source are available in the **Code Preview** view. The search can also be invoked by CTRL+F or from the context 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.

pin_mux.c@Cortex-M33 (Core #0) pin_mux.h@Cortex-M33 (Core #0)

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

/* clang-format off */
/*
 * TEXT BELOW IS USED AS SETTING FOR TOOLS *************************************
!!GlobalInfo
product: Pins v14.0
processor: LPC55S69
package_id: LPC55S69JBD100
mcu_data: ksdk2_0
processor_version: 0.14.3
 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS ***********
 */
/* clang-format on */

#include "fsl_common.h"
#include "pin_mux.h"

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

/* clang-format off */
/*
 * TEXT BELOW IS USED AS SETTING FOR TOOLS *************************************
BOARD_InitPins:
- options: {callFromInitBoot: 'true', coreID: cm33_core0, enableClock: 'true'}
- pin_list: []
 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS ***********
 */
/* clang-format on */

/* FUNCTION ********************************************************************
 *
 * Function Name : BOARD_InitPins
```

Figure 67. 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.

3.6 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 the case 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);
```

3.7 Full initialization of pins

In some cases, the default values are not reliable, as there may be code running before the application that modifies the pin configuration (for example, a bootloader). The option **Full initialization of pins** ensures that the initialization is fully done even for items that use after-reset state. This option is specific for each **Functional group** allowing to force full initialization of routing. **Full initialization of pins** is not enabled by default. When enabled, the electrical properties of existing routing are changed. The values in italics are changed to explicit values corresponding with them. When the option is disabled, the pins tool removes initialization of values that match the “No init” state.

3.8 Create Default Routing

If necessary, it is possible to create a new functional group that will route default signals to pins and internal signals. The functionality is available in **Pins -> Create Default Routing**. There the user can select:

- Whether all pins and signals will be routed, or only the ones that are not routed in other functional groups.
- The name of the new functional group.
- Whether the routing is created for pins and/or internal signals.

In the created functional group, the Full initialization function of the pins feature will be set. The electrical properties of pins will be set to their after-reset state.

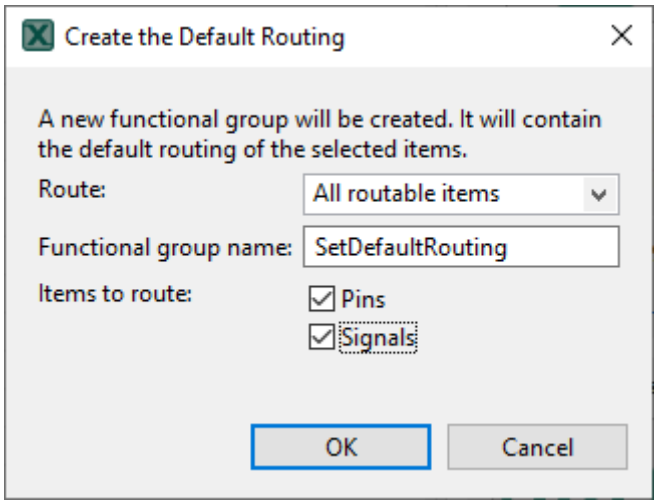


Figure 68. Create the Default Routing

4 Clocks Tool

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

4.1 Features

The Clocks tool allows you to perform various actions related to the Clock initialization, among them the following:

- Inspect and modify element configurations on the clock path from the clock source up to the core/peripherals.
- Validate clock element settings and calculate the resulting output clock frequencies.
- Generate a configuration code using the SDK.
- Modify the settings and provide 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.
- Find clock elements settings that fulfill given requirements for outputs.
- Fully integrated into the tools framework along with other tools.
- Show configuration problems in the **Problems** view and guide the user toward resolution.
- Register values define generation of C.

4.2 User interface overview

The **Clocks** tool is integrated and runs within the MCUXpresso Config Tools framework.

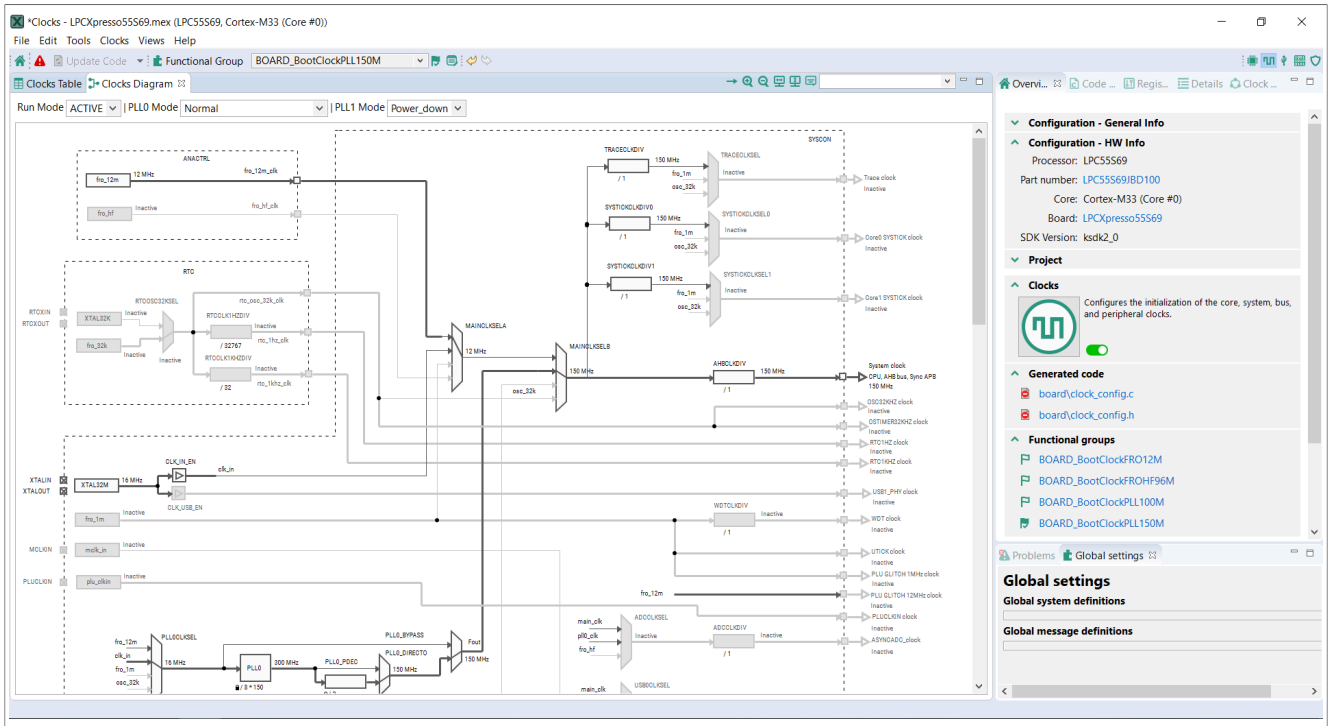


Figure 69. User interface

4.3 Clock configuration

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

There can be one or more clock configurations handled by the Clocks tool. The default clock configuration is created with the name "*BOARD_BootClockRUN*". Multiple configurations mean that 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 70. Default clock configuration

Note: The code generation engine of the tool generates a 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 is recommended to set them first. Global settings can be modified in **Clock Table**, **Clock Diagram**, and **Details** views, and the **Functional group properties** dialog.

Note: Global settings can be changed at any time.

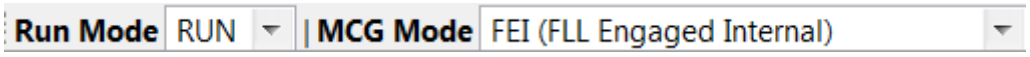


Figure 71. Global settings

4.5 Clock sources

The **Clock Sources** table is 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.

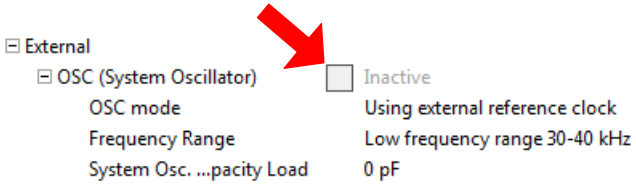


Figure 72. External clock source configuration

Note: Some clock sources remain inactive even though the checkbox is checked. It 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).

4.6.1 Table Setting states and markers

Table 15. Setting states and markers

State/Style/Marker	Icon	Description
Error marker		Indicates that there is an error in the settings or something related to it. See the tooltip of the setting for details.
Warning marker		Indicates that there is a warning in the settings or something related to it. See the tooltip of the setting for details.
Lock icon		Indicates that the settings (that may be automatically adjusted by the tool) are locked to prevent any automatic adjustment. If the setting can be locked, they are automatically locked when you change the value. To add/remove the lock manually, use the pop-up menu command **Lock/Unlock**.**Note:** The clock element settings that cannot be automatically adjusted by the tool keep their value as is and do not allow

Table 15. Setting states and markers...*continued*

State/Style/Marker	Icon	Description
		locking. They are: clock sources, clock selectors, and configuration elements.
Yellow background	100 MHz	Indicates that the field is directly or indirectly changed by the previous user action.
Gray text	FCTRIM	Indicates that the value of a 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 shows the text “inactive” instead of a 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 recalculates 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 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 is 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 the frequency you would like to set, change it, and press **Enter**.

Figure 73. Setting the core clock frequency

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

Figure 74. Tool attains the required frequency

If the tool cannot reach the required frequency or another problem occurs, it is displayed as follows:

Figure 75. 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

To access the menu, right-click on the clock output in the clocks view or in the diagram.

- **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.
- **Advanced resolver for {Clock output}** - Invokes a more advanced search for valid settings that fulfill the requirements. It may take significant time, so a progress dialog is shown. If the resolver is not successful, the user is informed. This command can alter clock selectors and modify various other clock settings on the clock path. If the result is not satisfactory, use the `UNDO` command to return to the original state.
- **Edit settings of: {element}** - Invokes the floating view with the settings for a single element.
- **Edit all settings** - Invokes the floating view with all the settings for an element.
- **Edit settings on the path to: {clock output}** - Invokes the floating view with the settings for all elements on the clock path leading to the selected clock output.

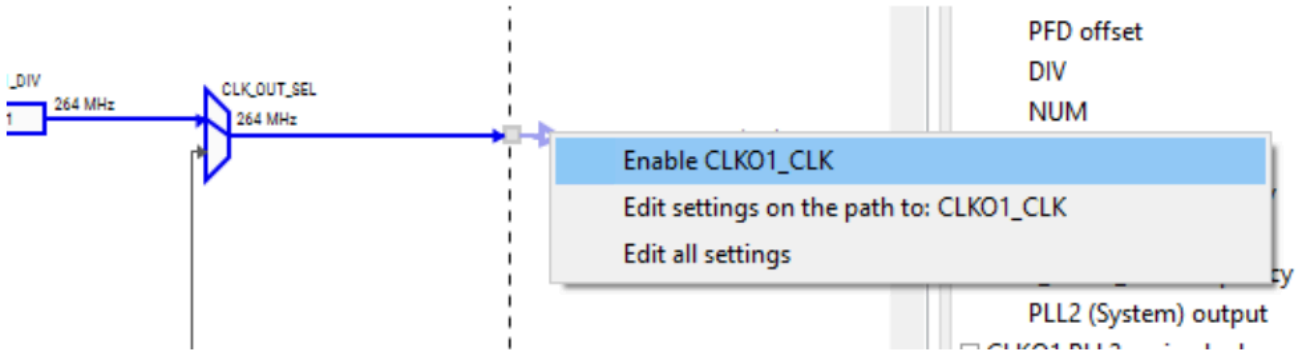


Figure 76. Pop-up commands for outputs in the clocks table view

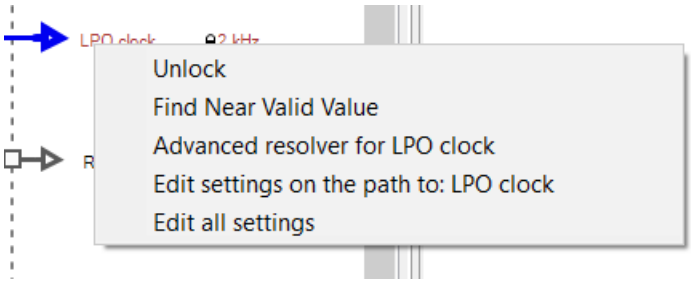


Figure 77. Pop-up commands for outputs in the clocks diagram

4.7.2 Frequency precision

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

Figure 78. Frequency precision

4.8 Dependency arrows

In the **Clocks 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 to all outputs that use the signal from the same clock source. They identify the dependencies and influences when there is a change in the clock source or elements on a shared clock path.

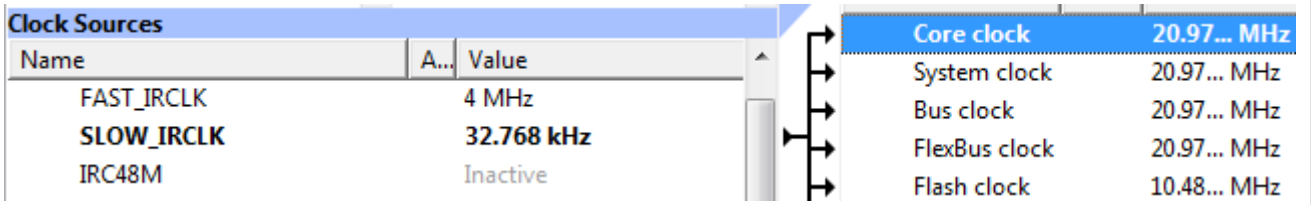


Figure 79. Dependency arrows

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**.

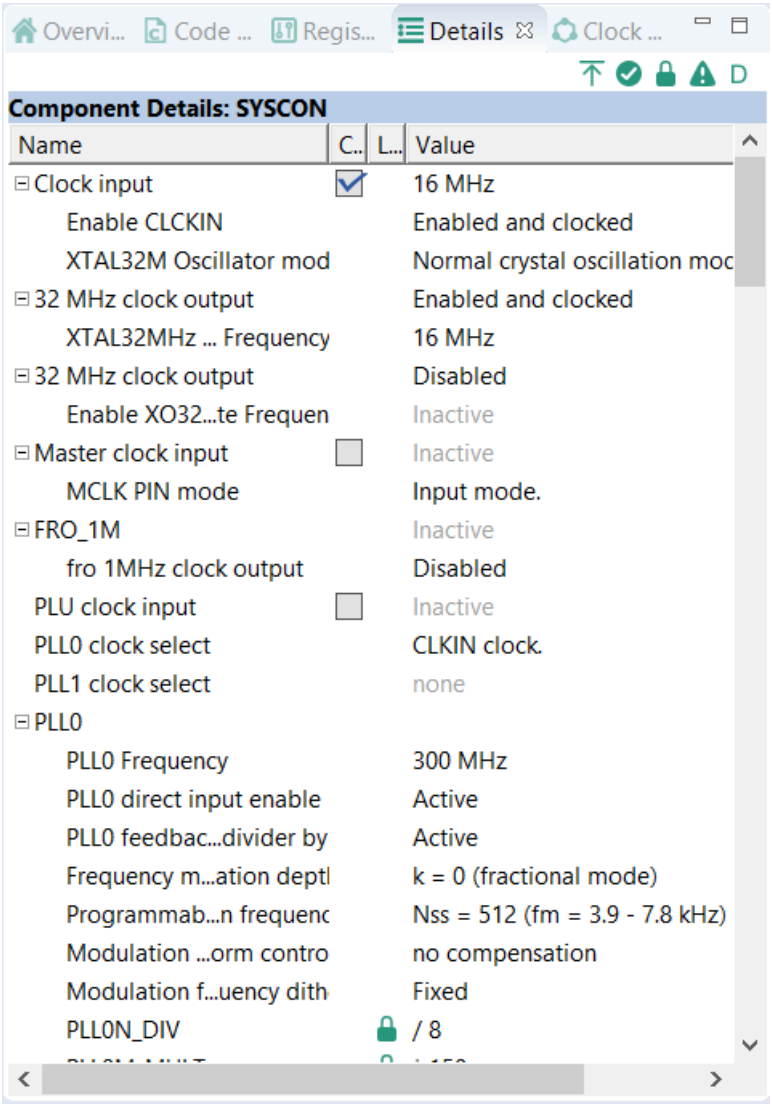


Figure 80. Details view

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 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 Clocks diagram

The clocks 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 is 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.

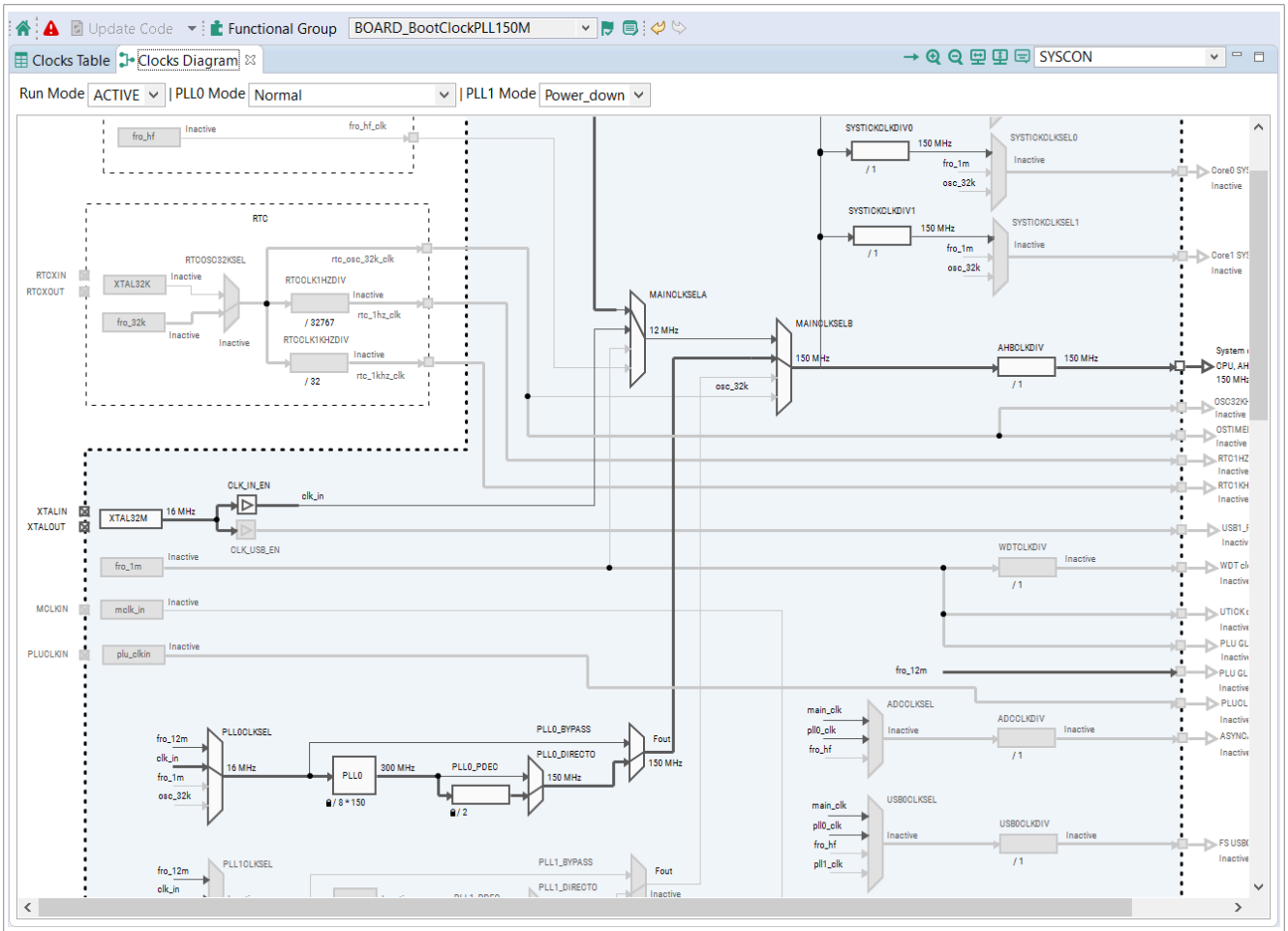


Figure 81. Clocks diagram

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 the element** to show its settings in the **Details** view (force to open the view if closed or not visible).
- **Single-click on the element** to show its settings in the **Details** view.
- **Single-click on a selected Clock source** to display a dropdown menu for enabling or disabling the source.
- **Single-click on a selected Clock selector** to display selector input options.

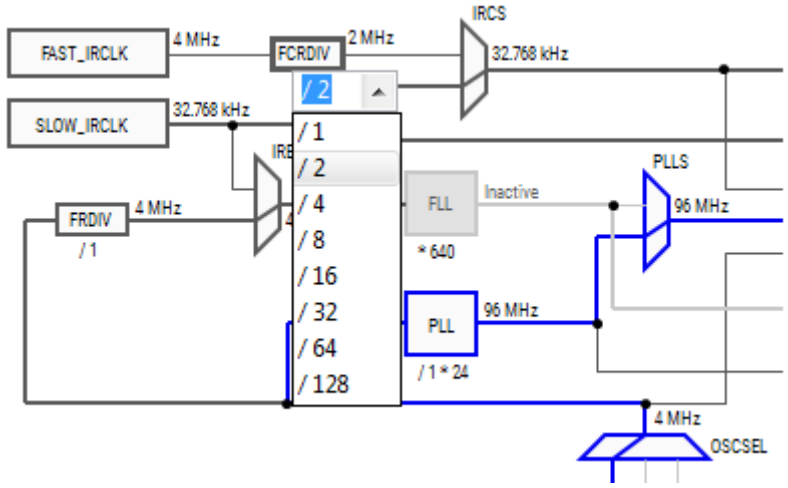


Figure 82. Clocks mouse actions in diagram

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

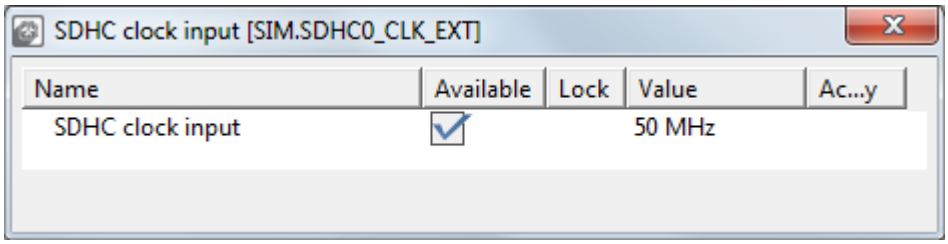


Figure 83. Floating view

4.10.2 Color and line styles

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

The color and line styles can indicate:

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

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

4.10.3 Clock model structure

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

The following are the clock model elements.

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

Figure 84. Clock source

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

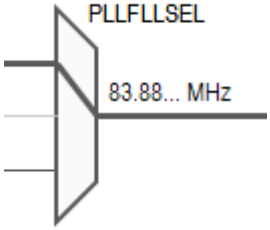


Figure 85. Clocks selector

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

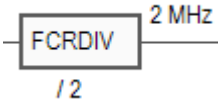


Figure 86. Prescaler

- **Frequency Locked Loop (FLL) -** Multiplies an input frequency by a given factor.

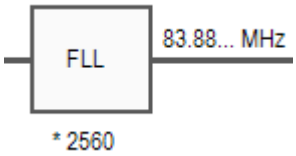


Figure 87. Frequency Locked Loop

- **Phase Locked Loop (PLL) -** Contains a pre-divider and can therefore divide or multiply by a given value.

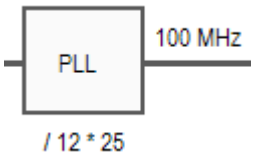


Figure 88. Phase Locked Loop

- **Clock gate -** Stops the propagation of an incoming signal.

Figure 89. Clock gate

- **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.

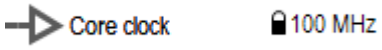


Figure 90. Clock output

- **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.

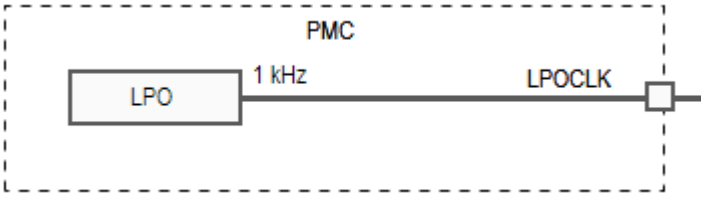


Figure 91. Clock component

- **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

Options related to the **Clocks** tool can be found in the **Clocks** menu in the **Menu bar**.

4.11.1 Table Clocks menu

Table 16. Clocks menu

Menu item	Description
Functional groups	Open the **Functional group properties** dialog.
Refresh	Refresh each clocks configuration with explicit invocation of code generation.
Reset To Board Defaults	Reset 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 is 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. **Requirements 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 setting and satisfy those requirements.
2. **Invalid settings or requirements:** [*element list*] - Indicates that the value of a setting 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. Start with only one locked clock output frequency and let the tool find and calculate other ones. After you are successful, you can add more.
2. Go through the locked outputs (if there are any) and verify the requirements (there can possibly be typos in the required frequency, wrong units, and so on).
3. If you seek only to enable some clock output, try to use pop-up the menu command **Enable** that tries to automatically find settings providing any valid frequency on clock output.
4. If the required clock output value cannot be satisfied, try using the pop-up menu command **Advanced resolver for {clock output}**.
5. If you want a frequency value close to the requested value but want to keep the clock selectors and clock sources unchanged, right-click and from the pop-up menu select **Clock output > Find near value**.
6. If the problems still persist, find the elements and settings with marked errors in the diagram or tables and see the details in the tooltip.
7. If you cannot reach the values you need, use the diagram view to see the elements on the clock path leading to the clock output you want to set. Check and adjust the settings of these elements manually in the **Details** view.
8. Try to remove locks by selecting **Clocks > Unlock All Settings.** In case too many changes are required and conflicting, 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 is possible, the pop-up is extended by:
  - **Remove lock** - Removes the lock from erroneous element.
  - **Find Near value** - Finds the nearest value.
  - **Enable**- If the clock output is disabled, tries to find settings that provide valid frequency on the clock output.
  - **Advanced resolver** - Invoked advanced resolver that tries to find suitable settings to achieve the required frequency. For more information, see the Advance resolver in the pop-up menu commands.

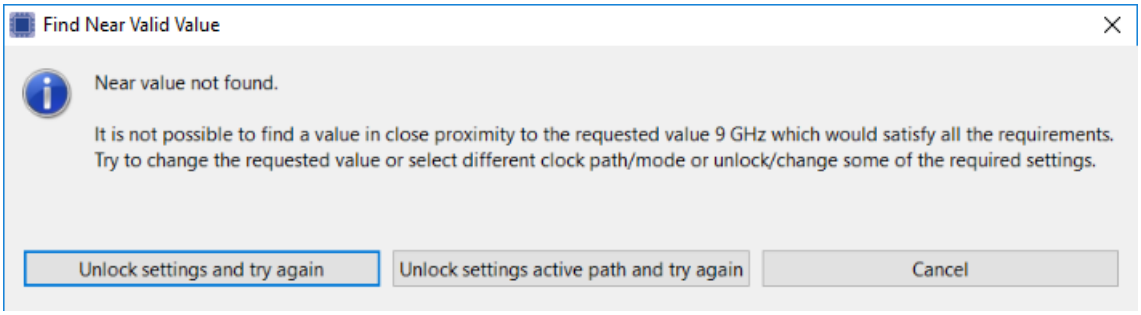


Figure 92. Find Near Value dialog

```
- **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 is 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 regenerates 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. Such features as Copy, Search, Zoom-in, Zoom-out, and Export source are available in the **Code Preview** view. The search can also be invoked by CTRL+F or from the context menu.

Figure 93. Code Preview

4.13.1 Working with the code

The generated code aligns with the SDK. To use the code with the SDK project, transfer the code into your project structure.

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

- Click **Update Code** in the toolbar.

- 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.
  If you need access to values of registers calculated by the tool, the defines with these values can be generated into a new file registers.h. This can be enabled by default (**Edit > Configuration Preferences**). For more information, see Configuration Preferences.

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.

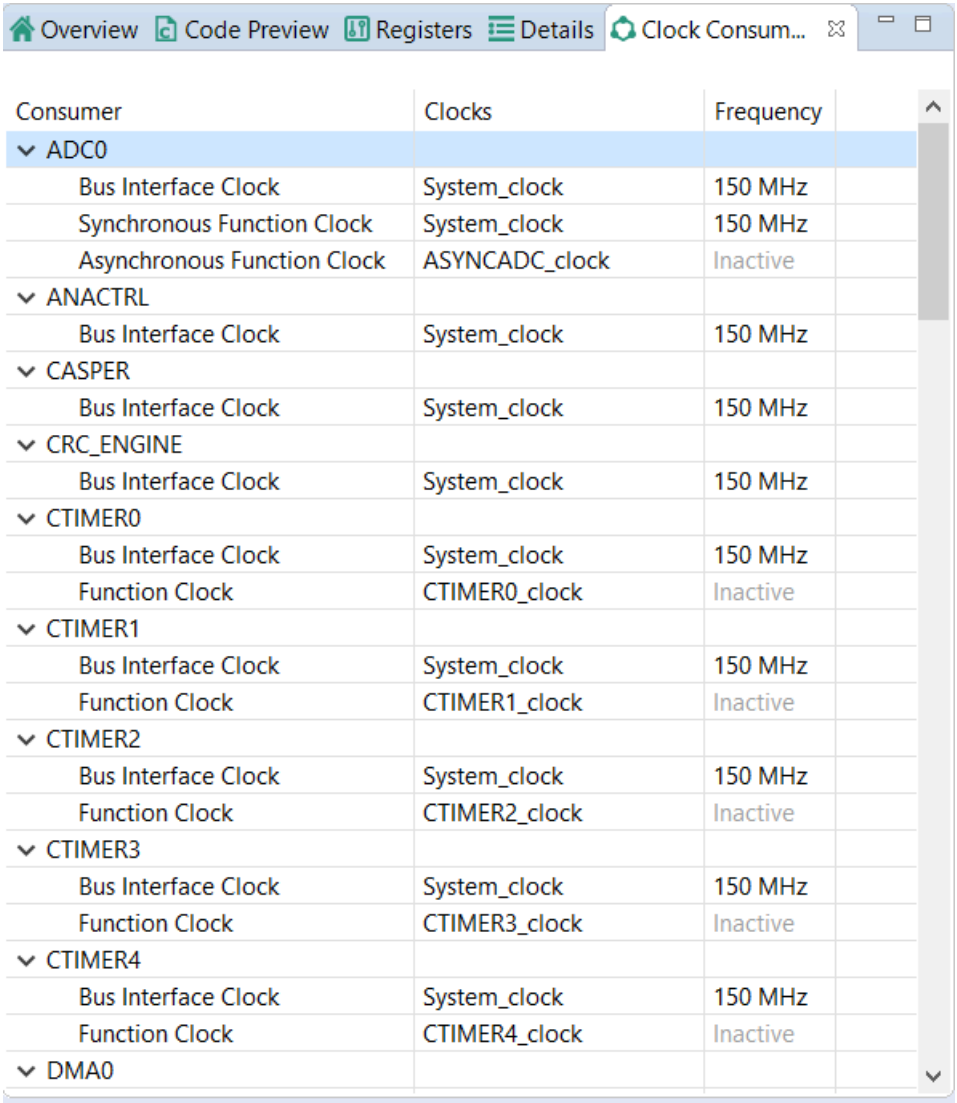


Figure 94. Clock Consumers view

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 based on the SDK drivers used in the 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 the Problems view and marked with decorators in other views
- Integration into the MCUXpresso Config Tools framework along with other tools
- Middleware configuration support (USB, FREEMaster, LwIP)
- The settings can be automatically migrated to a different SDK component version
- Support of code snippets

5.2 Basic terms and definitions

Table 17. Basic terms and definitions

Term	Definition
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 dialog displays the list of suitable configuration components for the selected peripheral matching the SDK driver for the selected processor.
3. Select the component that you want to use and click **OK**.

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

Note: The component mode selection may affect the appearance of some settings. Therefore, always select the mode first.

1. Open the **Code Preview** and see the output source code.

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

1. You can use the **Update Code** button from the toolbar. Alternatively, you can export the source code by selecting **File > Export…** from the **Menu bar**.
   Note: To export the source code, you can also click the **Export** button in the **Code Preview** view.
2. Settings can be saved in a MEX format (used for all settings of all tools) by selecting **File > Save** from the **Menu bar**.

5.4 User interface overview

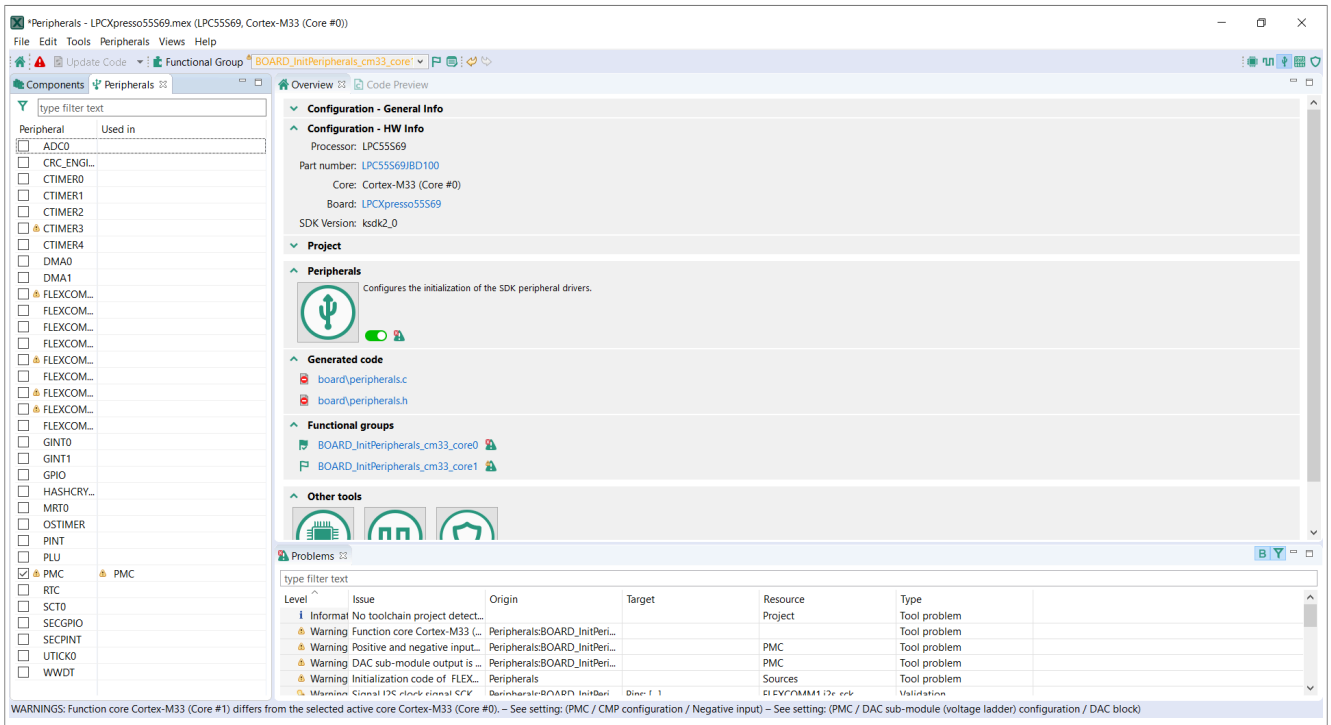


Figure 95. Peripheral view

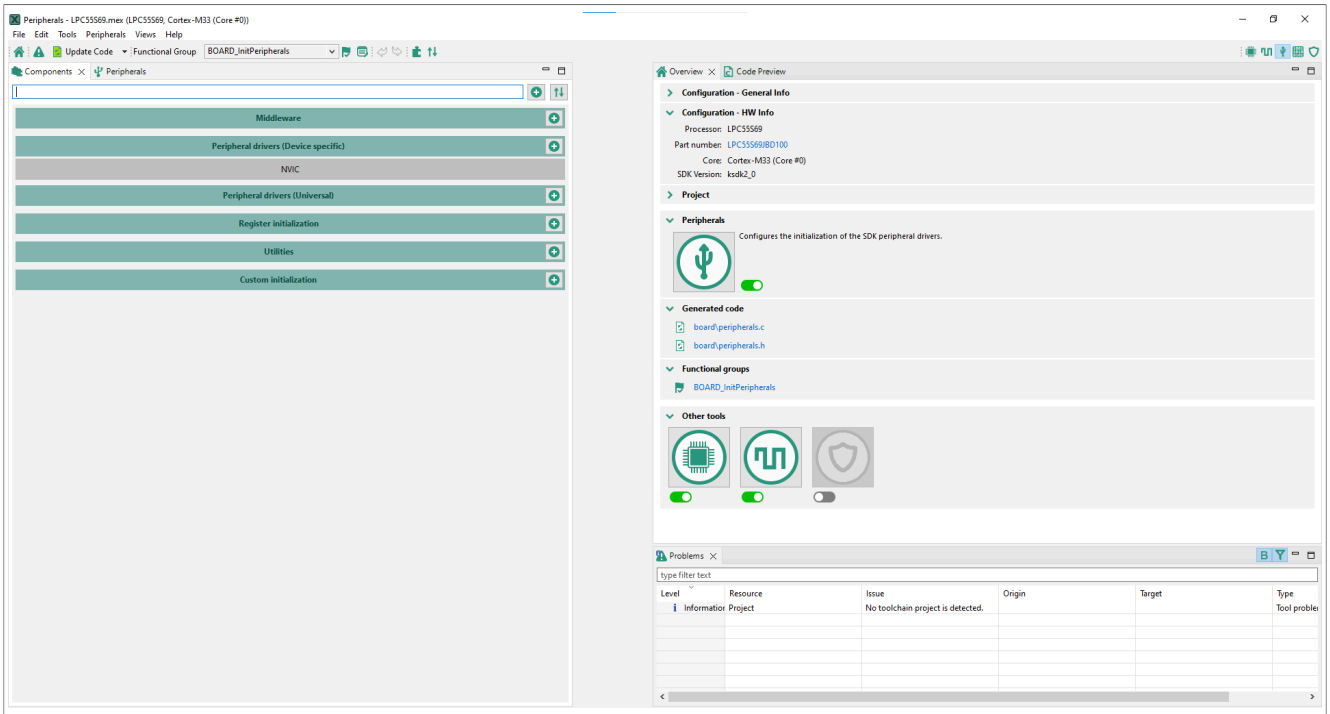


Figure 96. Components view

5.4.1 Common toolbar (Peripherals)

In addition to general items available to all tools, the **Toolbar** of the **Peripherals** tool contains two additional items:

5.4.1.1 Table Toolbar

Table 18. Toolbar

Item	Description
Global settings	Open a tab aggregating global settings of all configuration sets.
Initialization order	Open a dialog for customization of peripheral initialization order.

Note: For details on other items, refer to the Toolbar subsection.

5.4.1.2 Initialization order dialog

In the **Initialization order** dialog, you can customize the initialization order of peripherals within selected functional groups.

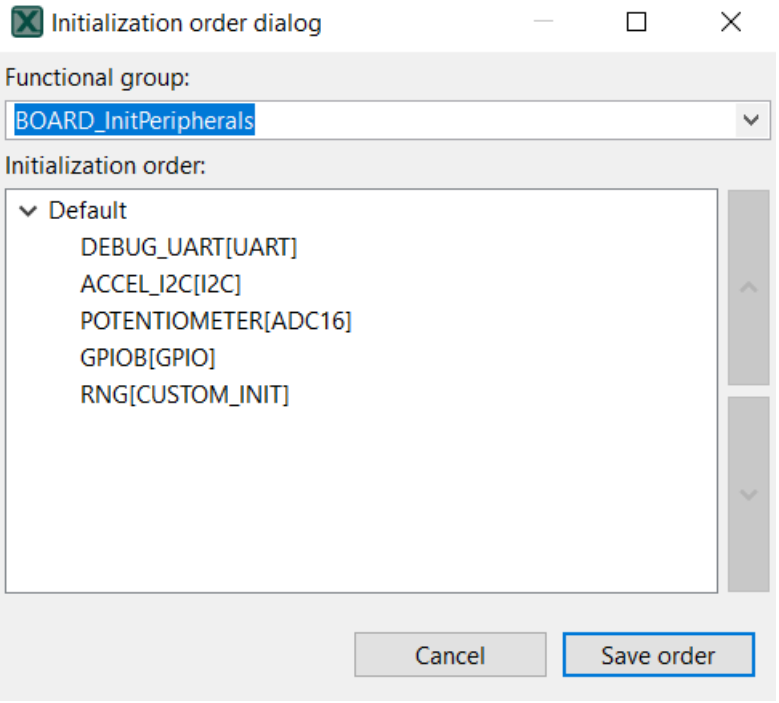


Figure 97. Initialization order dialog

1. Select the functional group you want to modify using the **Functional group** dropdown list.
2. In the **Initialization order** list, use the up and down arrows to adjust the sequence of initialization.
3. Click **Save order** to save your settings, or **Cancel** to close the dialog without changes.

5.4.2 Components view

The components view shows a list of configuration components, sorted by category into groups such as Middleware, Peripheral drivers and others.

The view highlights configuration components based on their status.

5.4.2.1 Table Component status

Table 19. Component status

Status	Color highlighting
Enabled	Light gray.
Enabled/with warning	Light gray with the alert symbol.
Enabled/with error	Red with the error symbol.
Disabled	Dark gray.

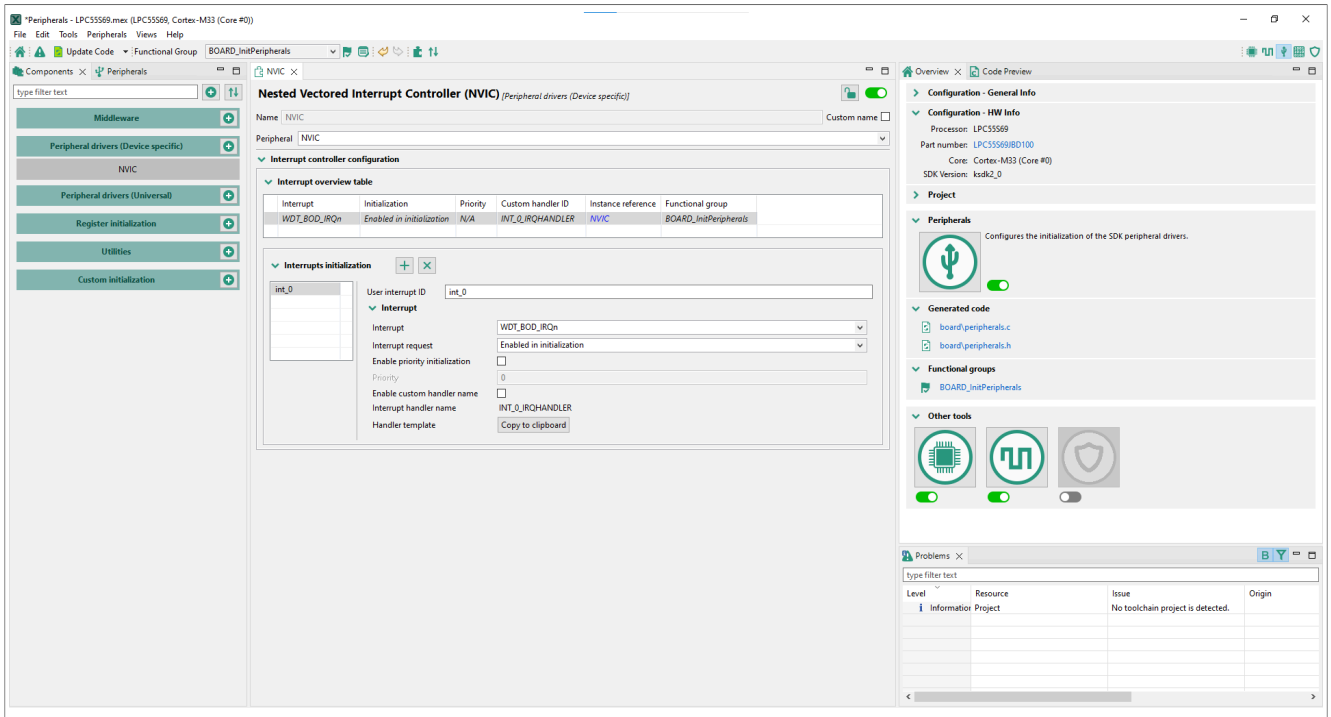


Figure 98. Components view

In the **Components** view, you can perform several actions.

5.4.2.2 Table Components view actions

Table 20. Components view actions

Option	Description
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**.
Add new configuration components	Left-click the **+** button and select from the list to add a component. In the **Select component** dialog, you can filter the list to show only toolchain-project-relevant or latest-version components. You can also click the **+** buttons next to the **Middleware**, **Peripheral drivers**, or **Other** categories to add new components directly, as shown in the figure below the table.
Filter configuration components by name	Type a text string to filter configuration component names in the search bar.

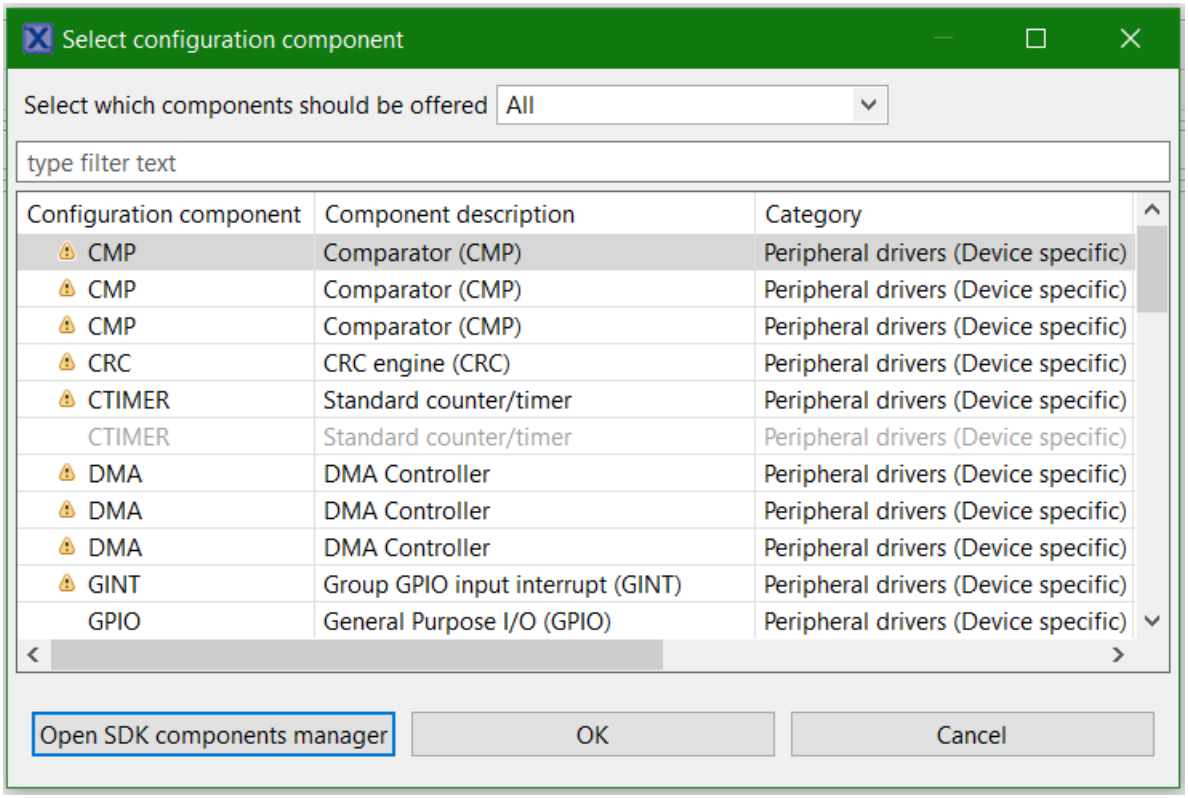


Figure 99. Add new configuration components

Right-click the configuration component to open a shortcut menu. Several options are available in the shortcut menu.

5.4.2.3 Table Shortcut menu options

Table 21. Shortcut menu options

Option	Description
Open	Open the configuration component in the **Settings Editor**.
Open in another view	Duplicates the configuration component in the **Settings Editor**.
Enable/Disable	Enable/Disable the configuration component.
Edit comment	Create/Edit custom notes for the configuration component.
Lock/Unlock editing of component instance	Lock/Unlock the editing of the component instance.
Save to use case library	Create a template from the configuration component.
Documentation	Display the documentation of the configuration component, if available.
Remove	Remove the component from configuration.**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	Migrate the component to a different component type or to a component with a newer driver version.

Table 21. Shortcut menu options...*continued*

Option	Description
Move to	Choose from available functional groups to move the configuration component to.
Copy to	Choose from available functional groups to copy the configuration component to.

5.4.3 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.

The second column contains the name of the component instance handling the peripheral. This name is customizable in the Settings Editor and is used in the generated code. The component instance name cannot 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 unchecking the checkbox.

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

Right-click the peripheral to open a shortcut menu. Several options are available in the shortcut menu.

5.4.3.1 Table Shortcut menu options

Table 22. Shortcut menu options

Option	Description
Open	Open the component instance in the **Settings Editor**.
Open in another view	Duplicate the component instance in the **Settings Editor**.
Enable/Disable	Enable/Disable the component instance.
Add component instance	Add a component instance to the peripheral.
Initialized in user code	Mark the peripheral as configured by user code (available on not configured peripherals).
Edit comment	Create/Edit custom notes for the component instance.
Lock/Unlock editing of component instance	Lock/Unlock the editing of the component instance.
Save to use case library	Create a template from the component instance.
Documentation	Display the documentation of the component instance, if available.
Remove	Remove the component instance from configuration. 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.
Move to	Choose from available functional groups to move the component instance to.

Table 22. Shortcut menu options...*continued*

Option	Description
Copy to	Choose from available functional groups to copy the component instance to.

5.4.4 Settings Editor

You can edit peripheral component settings in the **Settings Editor**. Open editors appear in the central area of the screen, each with its own tab. Multiple editors can be open at the same time. Changes made in the editor are immediately applied and retained even if the Settings Editor is closed. Disabled settings are highlighted in gray. If a component instance is disabled, all settings are highlighted in gray. Tooltips appear for all enabled settings when the mouse cursor is placed on a setting.

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.

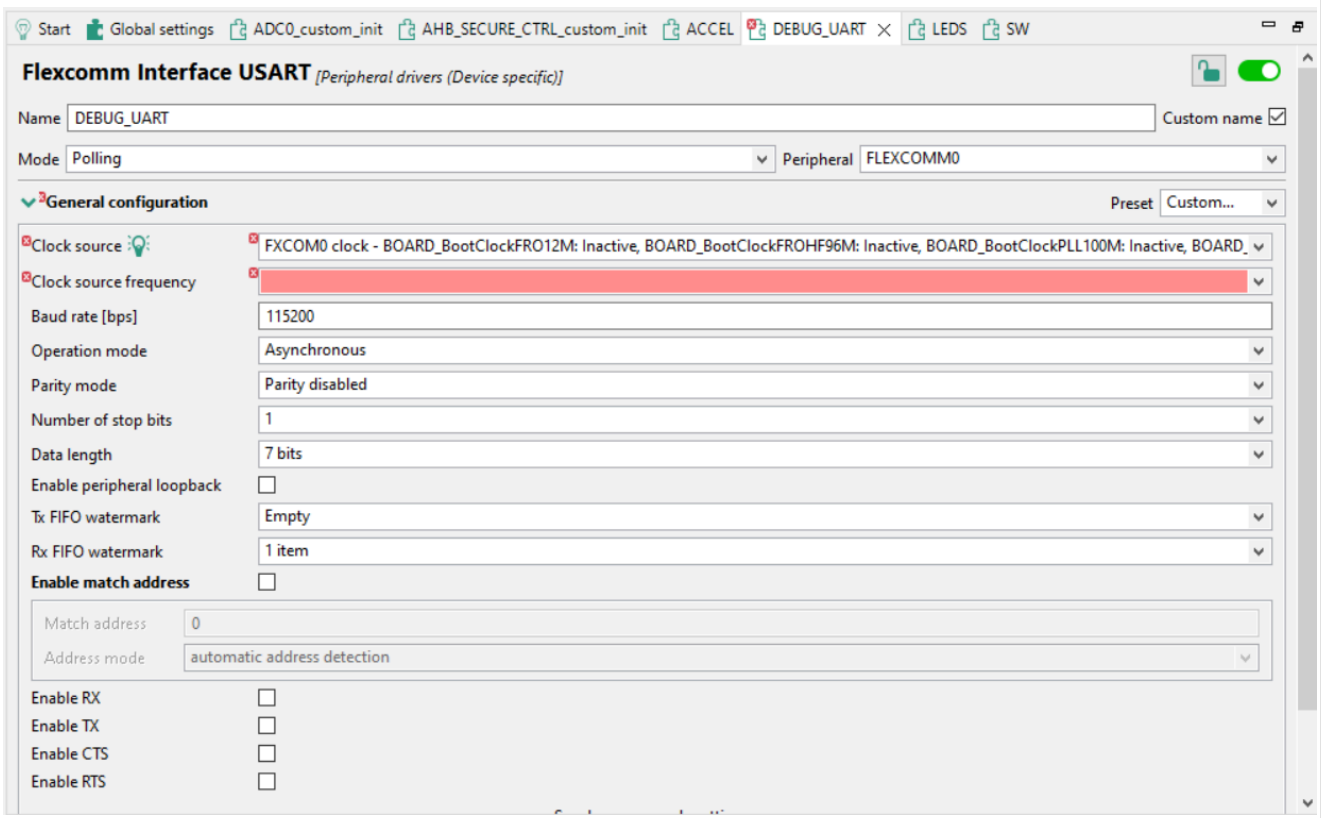


Figure 100. Settings editor

5.4.4.1 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 101. Settings Editor header

5.4.4.1.1 Table Settings Editor header

Table 23. Settings Editor header

Header item	Description
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.
Documentation	Click the button to view configuration component-specific documentation in the **Documentation** view. Not all configuration components are documented, therefore not all setting headers contain the **Documentation** icon.
Lock editing	Click the button to lock/unlock component editing. Source code will still be generated.
Enable/disable component instance switch	Use the switch to enable or disable selected component instance. By disabling the instance, you don't remove it from the tools configuration, but prevent its inclusion in the generated code.

5.4.4.2 Presets

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

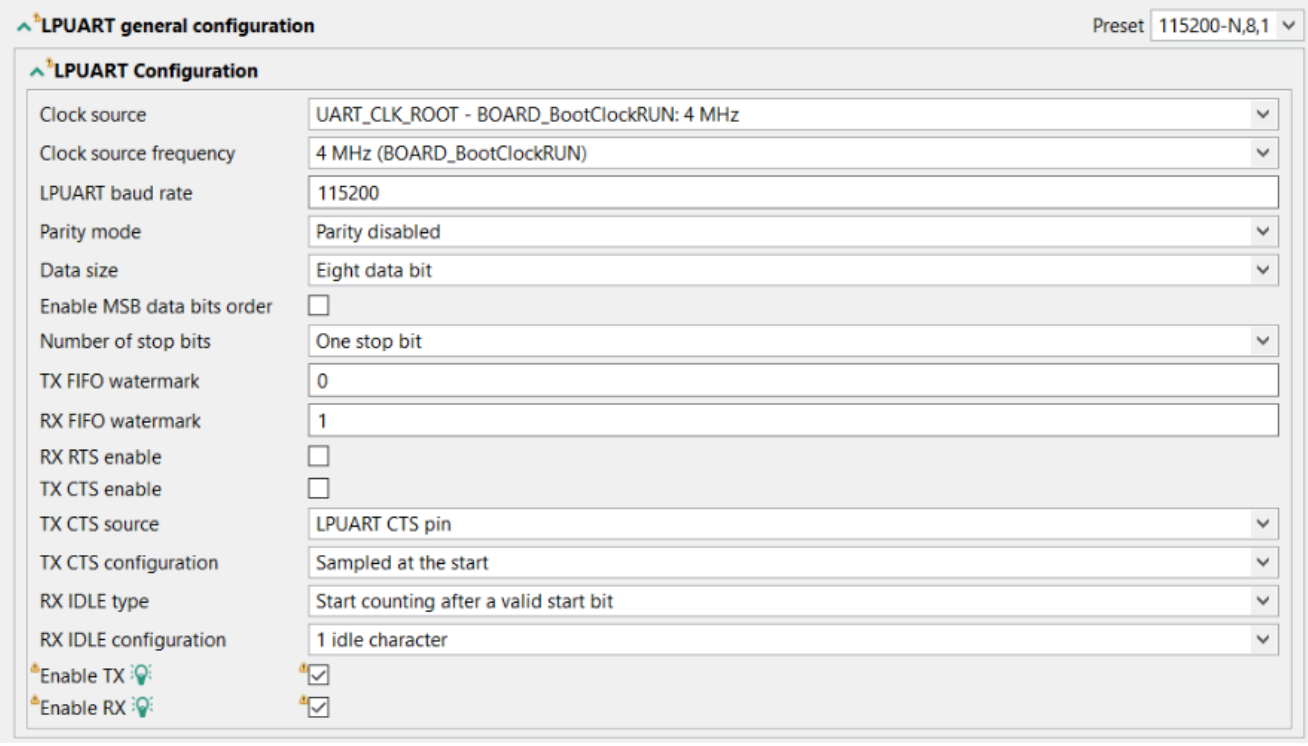


Figure 102. Quick selection example

5.4.4.3 Settings

The following setting types are available in the **Settings Editor** :

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

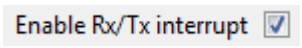


Figure 103. Boolean setting example

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

Figure 104. Integer/Float setting example

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

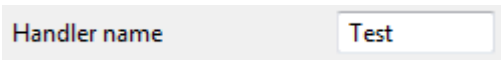


Figure 105. String setting example

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

Figure 106. Enumeration setting example

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

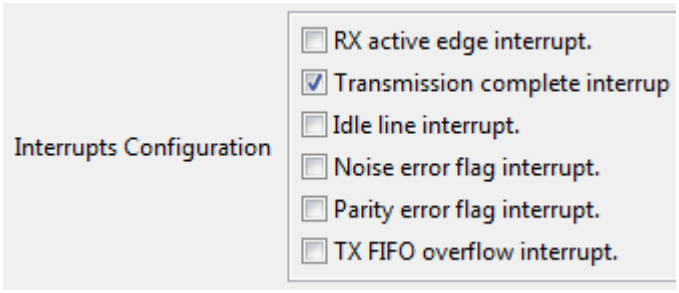


Figure 107. Set setting example

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

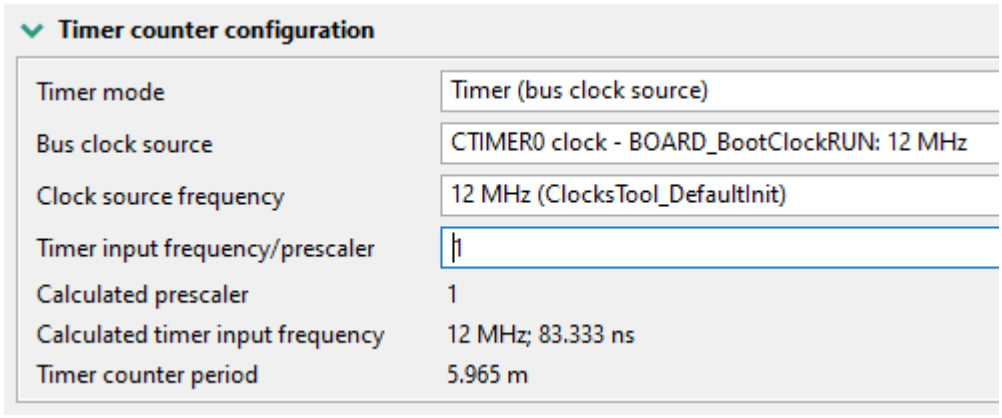


Figure 108. Structure setting example

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

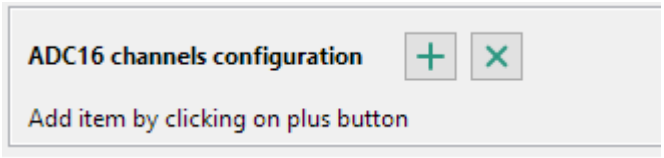


Figure 109. Array

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: The array can be copied and pasted to another configuration, including in the second running instance of Config Tools.

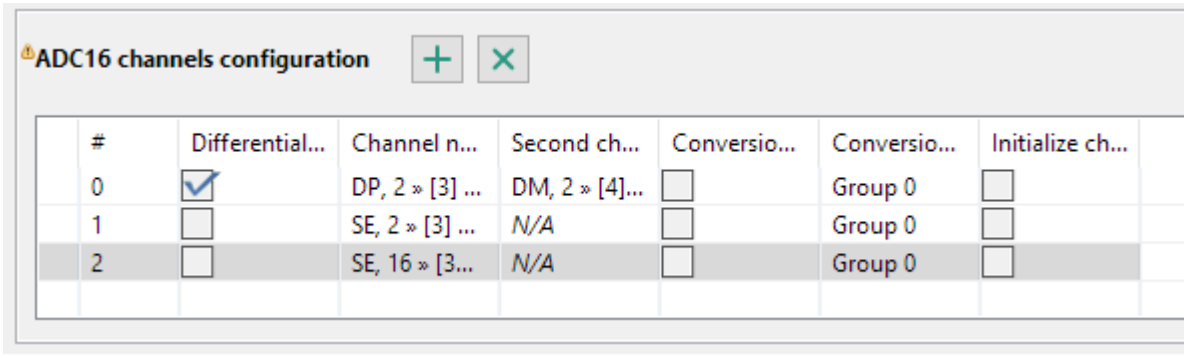


Figure 110. Array setting example

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

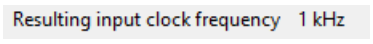


Figure 111. Info setting example

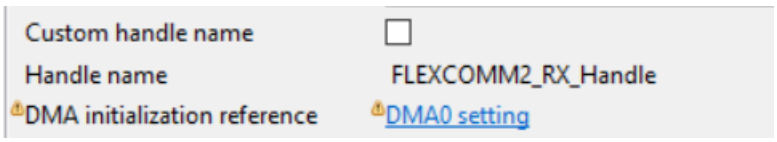


Figure 112. Info setting as link example

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

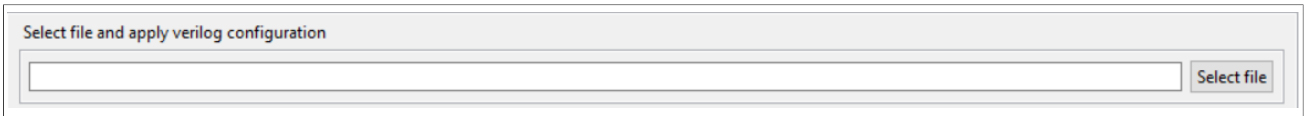


Figure 113. File setting

5.4.5 Documentation view

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

Note: Not all components might have this option enabled.

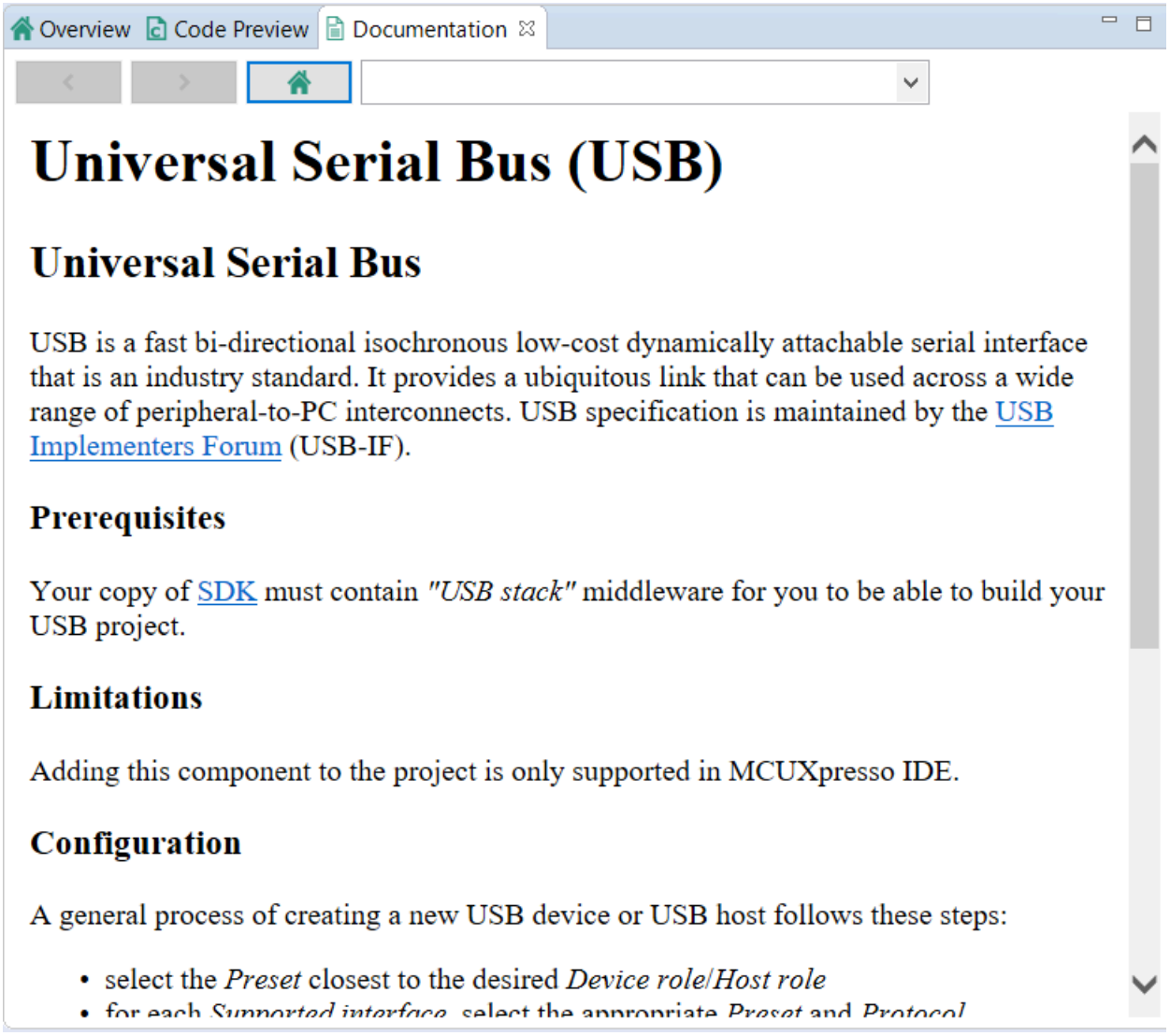


Figure 114. Documentation view

You can open the **Documentation** view in several 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.
- In the **Settings Editor**, click the question mark next to the settings label.

5.4.6 Component use case library

In Peripherals tool, you can save, edit, and import/export component use cases for future use. Use cases are saved in a MEX format and can be viewed and modified in the **Component use case library**. The library displays all created/imported use cases by component type.

To open the **Component use case library**, select **Peripherals > Component use case library** from the **Menu bar**.

To create a component use case, do the following:

1. Right-click an entry in the **Peripherals** or **Components** views.
2. Select **Save to use case library** from the context menu.
3. Enter the name and description in the **Use case detail** dialog.
4. Click **OK**.

5.4.7 Component Migration to a different version

Configuration components that generate configuration code for SDK components often require specific versions (or range of versions) of one or more SDK components.

The SDK component and its version that is expected for the proper function of the configuration component are visible when components are added to the selection dialog:

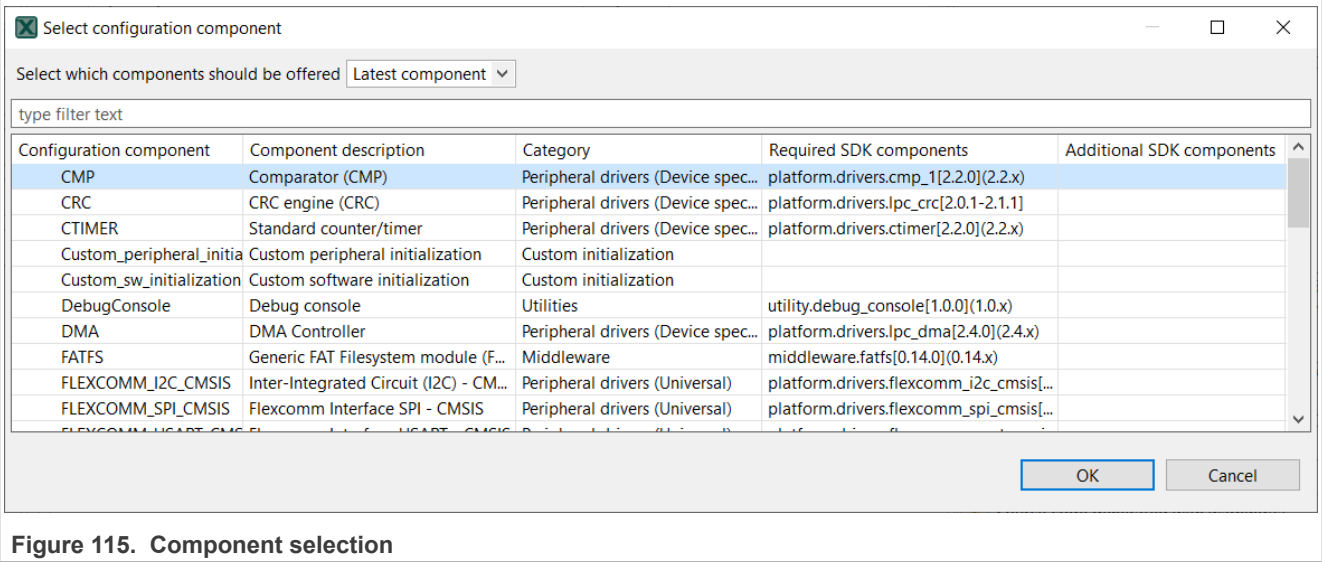


Figure 115. Component selection

It is also visible in the tooltip in the Component view:

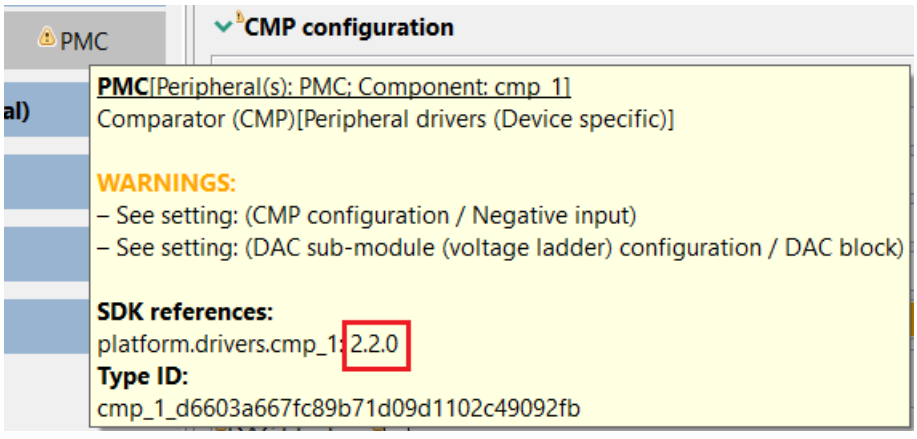


Figure 116. Tooltip in the Component view

You can update the SDK components in the toolchain/IDE project.

When a configuration is open and the SDK component versions in the toolchain project do not match the version referenced in the configuration component, automatic migration is offered in the **Component migration** dialog. Migration can also be launched manually in the Peripherals tool using the **Peripherals > Migrate to other component versions** menu option.

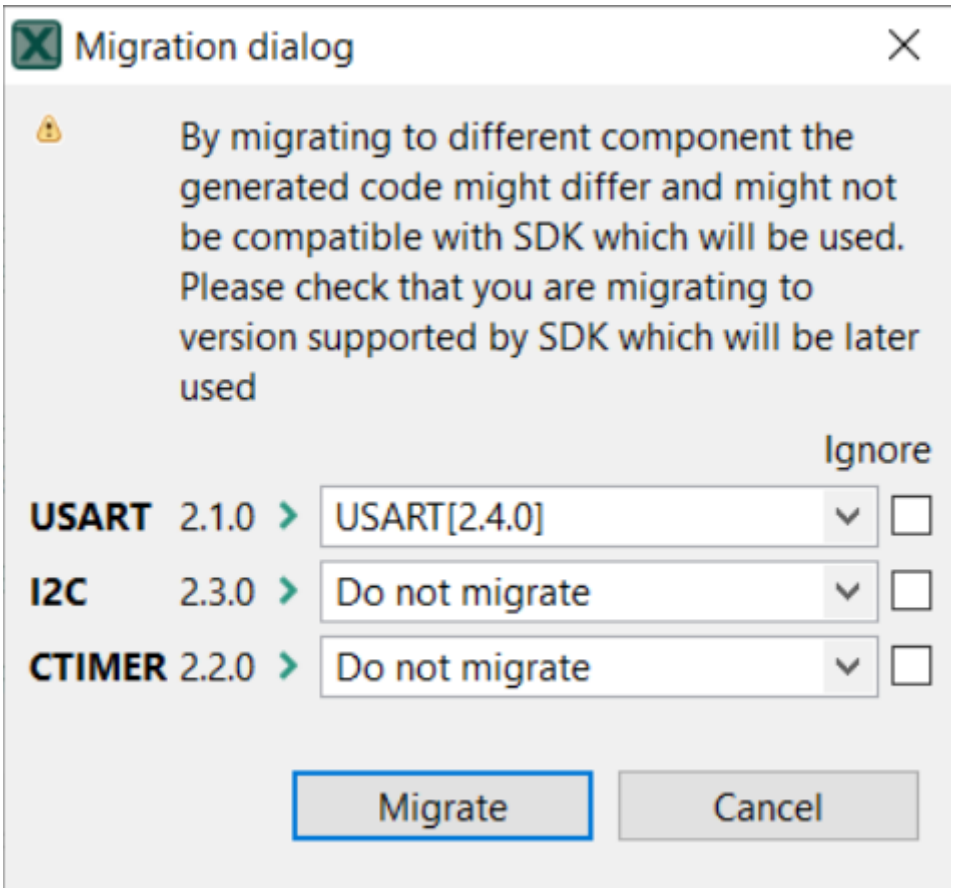


Figure 117. Migration dialog

Each row in the dialog corresponds to one configuration component that can be migrated to another version. The dialog displays the current version specified in the configuration component and a combo box for selecting the new version that replaces the current one.

In standalone Config Tools, it is possible to migrate settings to any available version. In IDE/toolchain project mode, the combo box contains only the component with the version matching the SDK component currently used in the project.

The default selection in the toolchain-less configuration is “Do not migrate”. In the toolchain configuration, the default selection is the only version to which the migration can be performed. If “Do not migrate” is selected, no changes are made to the particular component.

The **Ignore** checkbox prevents the component from the migration during next check.

After you confirm the dialog by selecting “**Migrate**”, the component is replaced by the component matching the selected version of the SDK component. The settings are migrated to the corresponding settings in the new version of the component, where it is possible.

If the new version of the component contains some new settings, these settings are filled with the default values. Check manually if the components are set properly.

The **Migration result** dialog is a dialog with the summary of the migration. It automatically opens after migration is successfully completed. This dialog contains the summary of events with various impact, that happened during the migration. It also contains a link, that opens the generated detail report in the HTML format, and a button that copies the path to the report into the system clipboard.

The generated Migration report contains the date and information about the configuration location, MCU, toolchain project in its header. It helps to identify to what configuration it is related. Below that is a table, that contains rows for each migration. A short description of the migration is provided. Each row has two columns where the first contains the path to the migrated setting and the second contains the description of the migration. Each row can have differently colored text, that indicates the importance of the message.

5.5 Memory 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 and FCB peripherals and can be used after selecting these peripherals 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.

If the settings are valid, click the **Sync with DCD** button to synchronize the memory settings with the **DCD** tool. Manually update the existing SDRAM configuration in the DCD tool with the one generated by clicking **Apply to DCD**. If a configuration is not defined in the DCD tool, the configuration generated by **Apply to DCD** does not work as-is; add an additional configuration in the Clocks or Pins tools.

5.5.1 Validation view

Use the **Validation** view to run validation scenarios for your memory settings and analyze 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 that you wish to test in the **Scenarios** subview.

- To verify the configuration, select one of the available tests.

1. To start validation, click the **Start Validation** button.
2. Observe the results in real time in the **Results** subview.
3. Inspect the results in the **Summary** and **Logs** subviews.
4. 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.
5. Download DDRV reports using the **Configure and generate DDRV reports** button.

5.6 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.

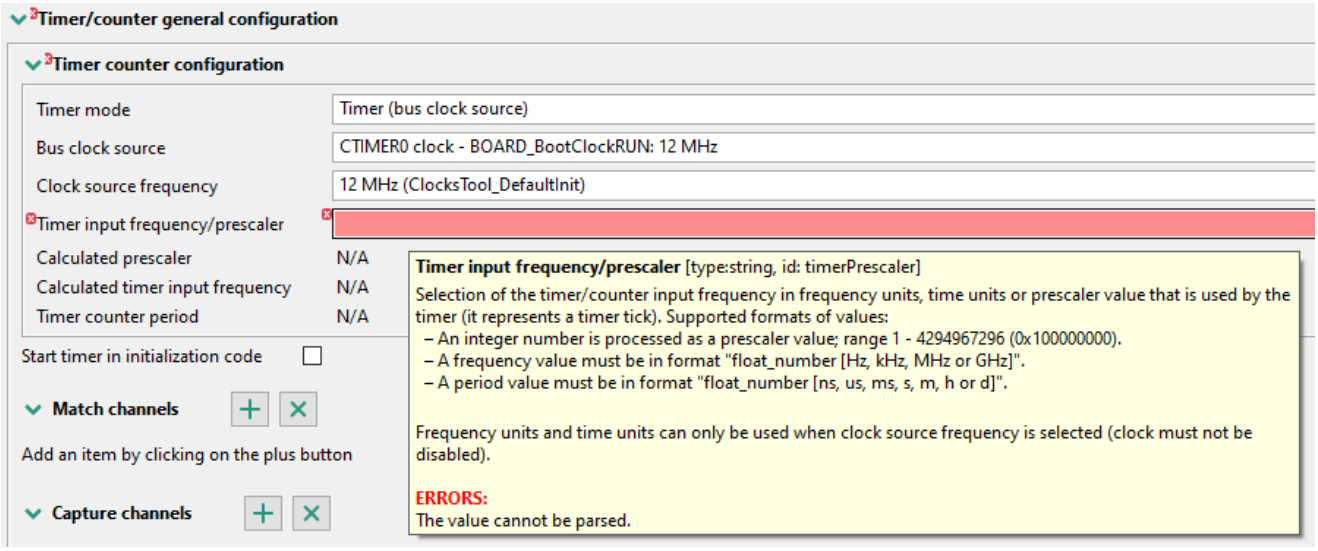


Figure 118. Error decorators

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

Figure 119. Quick fix 1

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

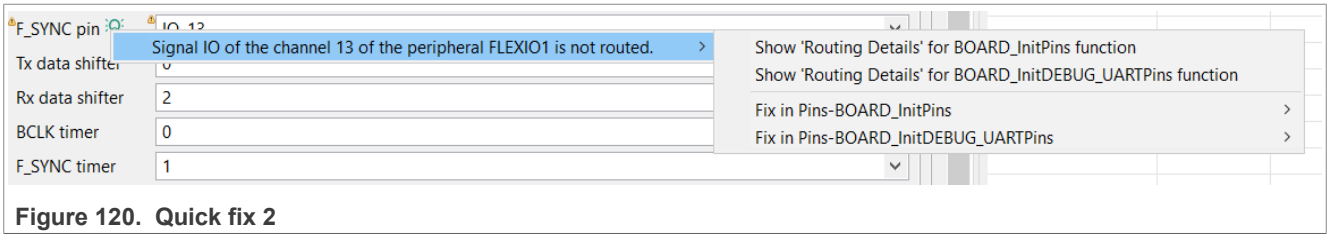


Figure 120. Quick fix 2

There is a new possibility to do quick fix from the table in the context menu after right-clicking on the cell that contains the warning/error icon (see register initialized SCTimer, for example LPC54114. **Resources->Outputs setting**).

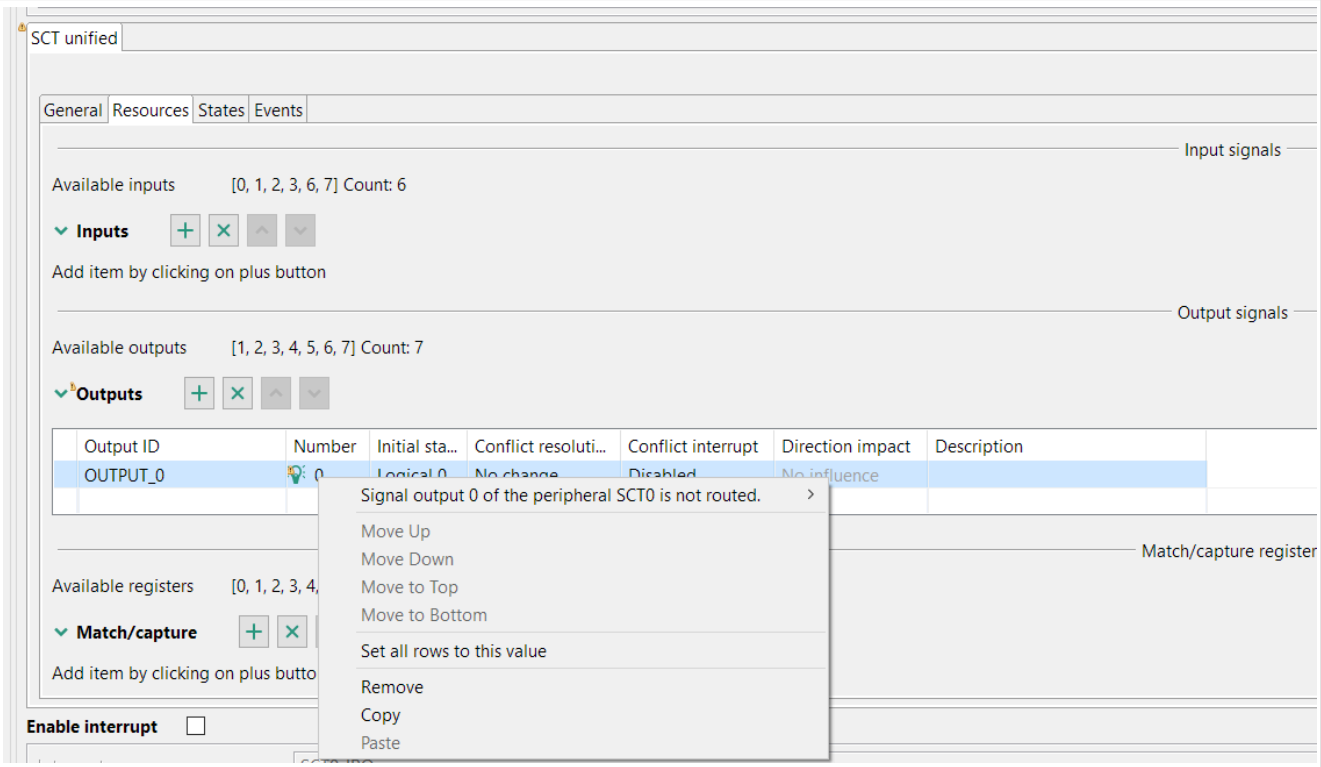


Figure 121. Quick fix 3

5.7 Code generation

If the settings are correct and no error is reported, the tool's code generation engine instantly regenerates the source code. You can view the resulting code in 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 highlighting modes by clicking **Set viewing style for source differences**, or disable highlighting altogether from the same dropdown menu. Copy, Search, Zoom-in, Zoom-out, and Export source are also available in the **Code Preview** view. Search can also be invoked by CTRL+F or from the context 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. It 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 function declarations in the 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.

Figure 122. Code Preview

6 Device Configuration Tool

The **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.

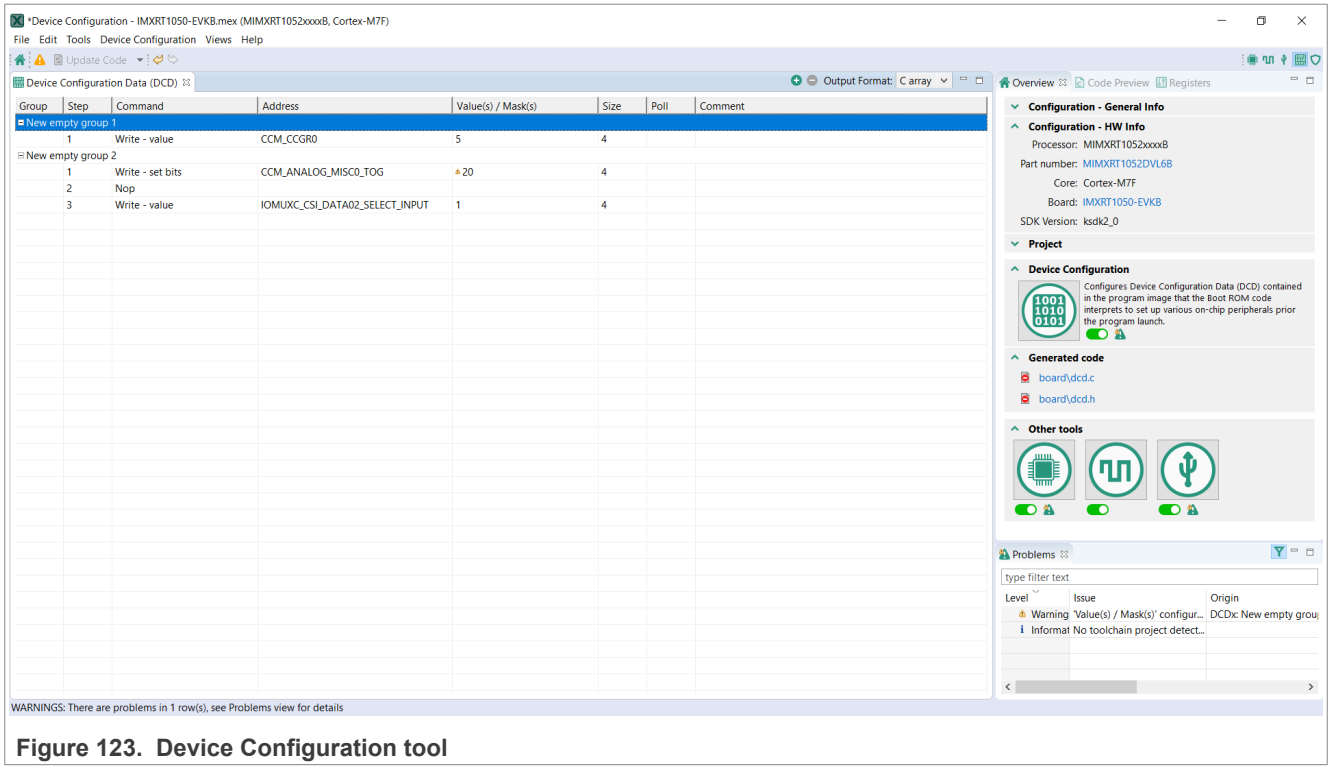


Figure 123. Device Configuration tool

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 command groups and commands and specify their parameters.

Commands in the **Device Configuration Data (DCD)** can be synchronized 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 that 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 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.

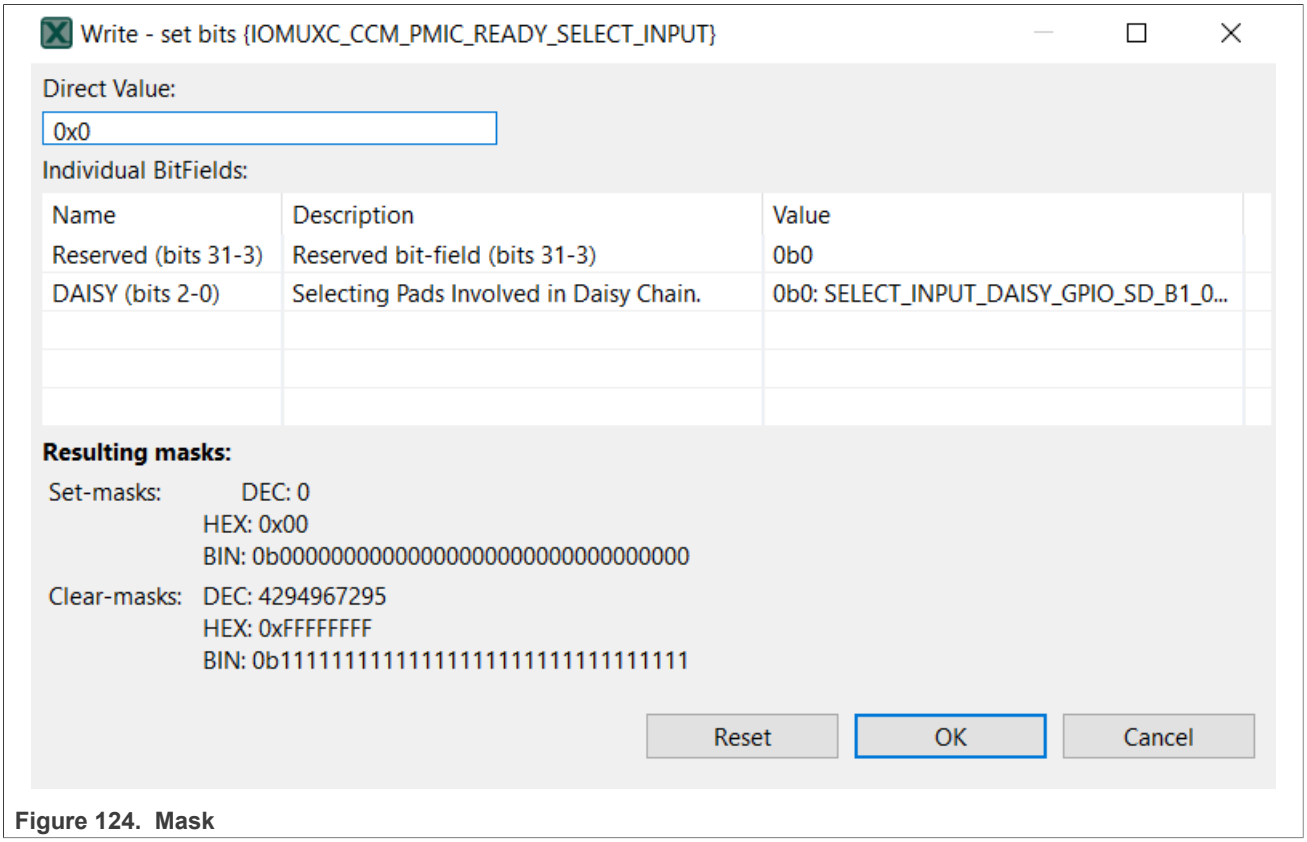


Figure 124. Mask

- **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 **Menu bar** 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 regenerates the source code. You can view the resulting code in 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. Such features as Copy, Search, Zoom-in, Zoom-out, and Export source are available in the **Code Preview** view. The search can also be invoked by CTRL+F or from the context 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.

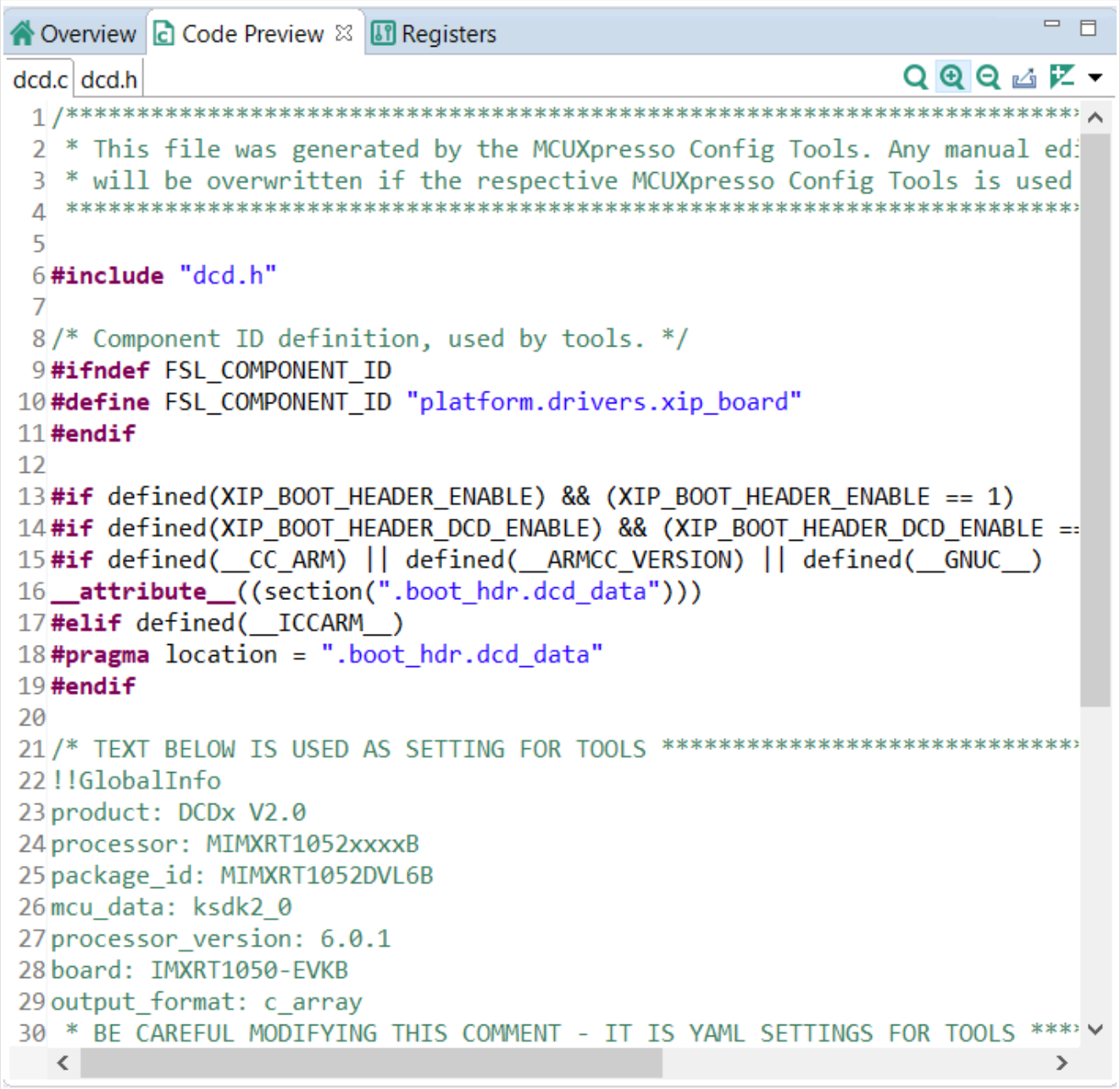


Figure 125. Code Preview

7 Trusted Execution Environment Tool

In the **Trusted Execution Environment** (**TEE**) tool, you can configure security policies of memory areas, bus masters, and peripherals 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 subviews, and review these policies in the **Memory Attribution Map**, **Access Overview** and **Domains 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: For your configuration to take effect, make sure you have enabled the relevant secure check option in the **Miscellaneous** subview of the **Security Access Configuration** view.

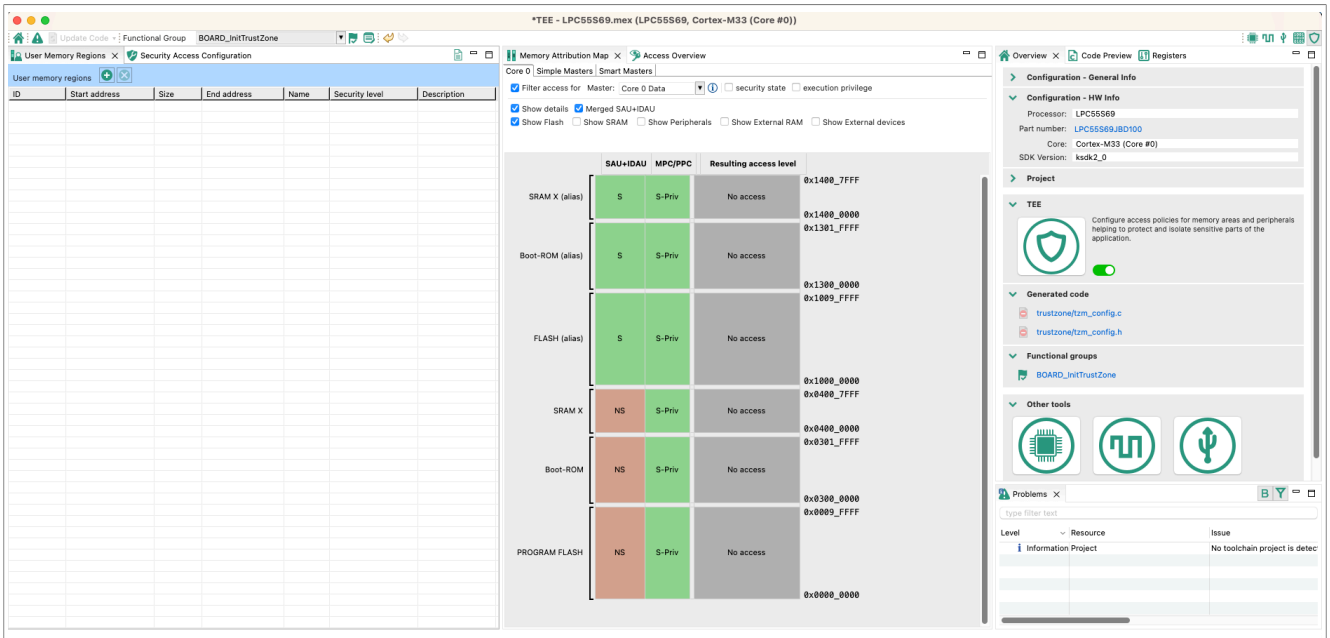


Figure 126. TEE tool user interface (TrustZone-M with AHBSC)

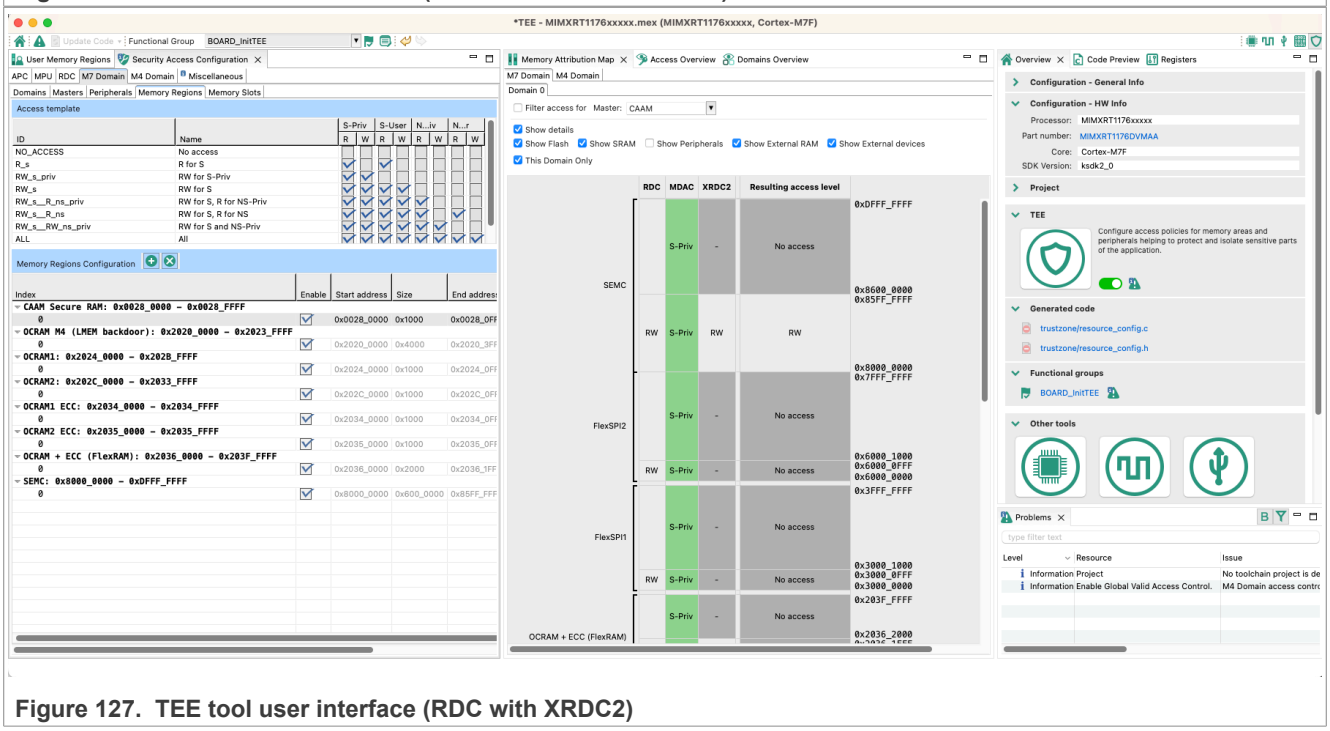


Figure 127. TEE tool user interface (RDC with XRDC2)

7.1 AHBSC with security extension-enabled devices

The features and appearance of the TEE tool are based on the security model of the loaded device.

This section describes the features and appearance of the tool for devices with a security extension TrustZone-M with AHBSC.

Currently, the following devices of this type are supported:

- LPC55Sxx
  - LPC55S69, LPC55S66
  - LPC55S16, LPC55S14, LPC55S36
  - LPC55S06, LPC55S04
- RT6xx, RT5xx, RT7xx
  - MIMXRT685S, MIMXRT633S
  - MIMXRT595S, MIMXRT555S, MIMXRT533S1
  - MIMXRT735, MIMXRT758, MIMXRT798
- MCXN
  - MCXN546, MCXN547, MCXN946, MCXN947, MCXN236, MCXN235

7.1.1 User Memory Regions view

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**

Errors in configuration are highlighted by a red icon in the relevant cell. In the case the issue is easily fixed, you can right-click the cell to display a dropdown list of offered solutions.

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

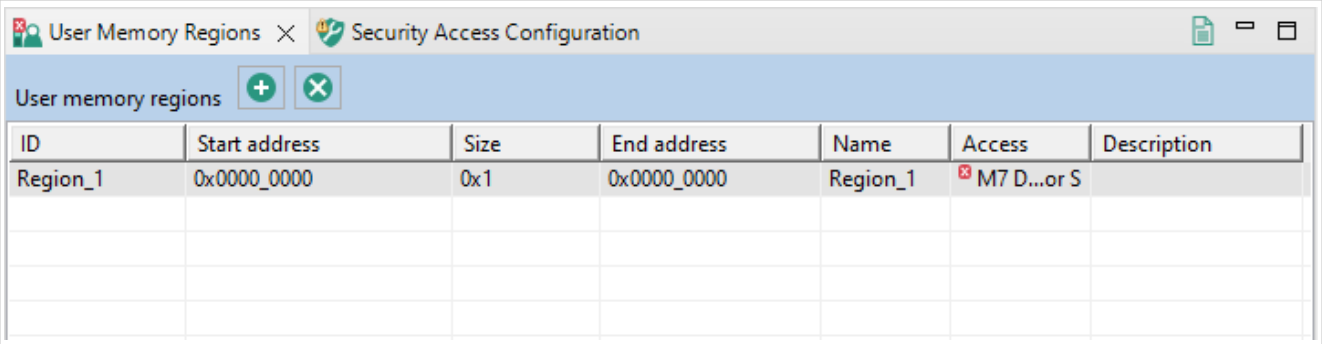


Figure 128. User Memory Regions

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 that 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.

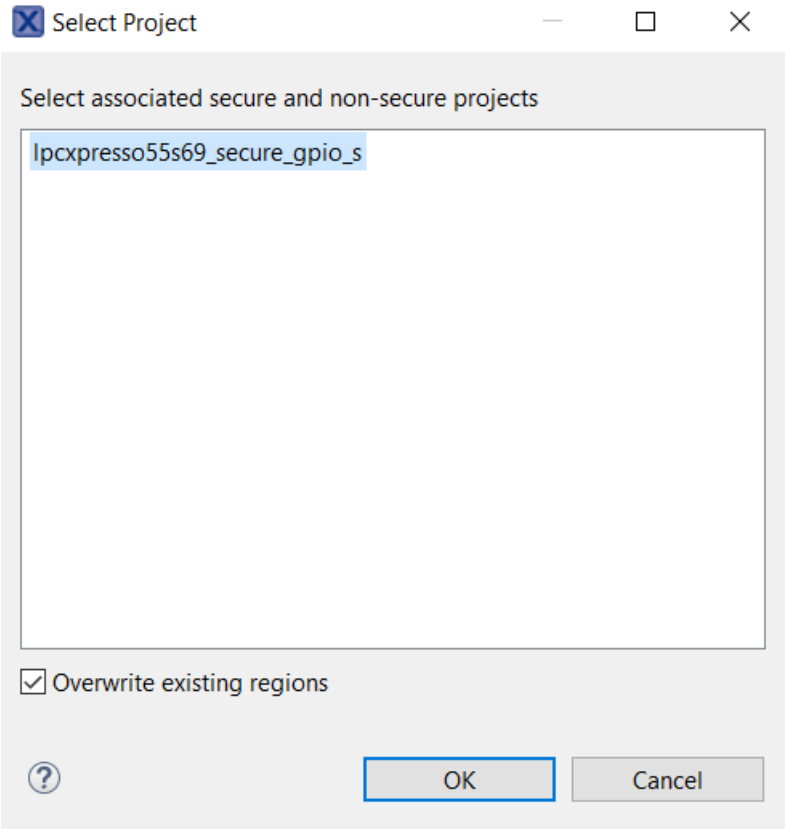


Figure 129. Select Project

7.1.2 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.

7.1.2.1 SAU

In the **SAU** subview, 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 access level. SAU automatically sets the entire memory space to a Secure access level when disabled. When enabled, SAU deems every uncovered (that is, unconfigured) memory region as Secure, so only NS or NSC can be selected for a covered (configured) memory region.

You can choose between two access levels:

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

Alternatively, you can set all the SAU memory regions to non-secure access 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**.

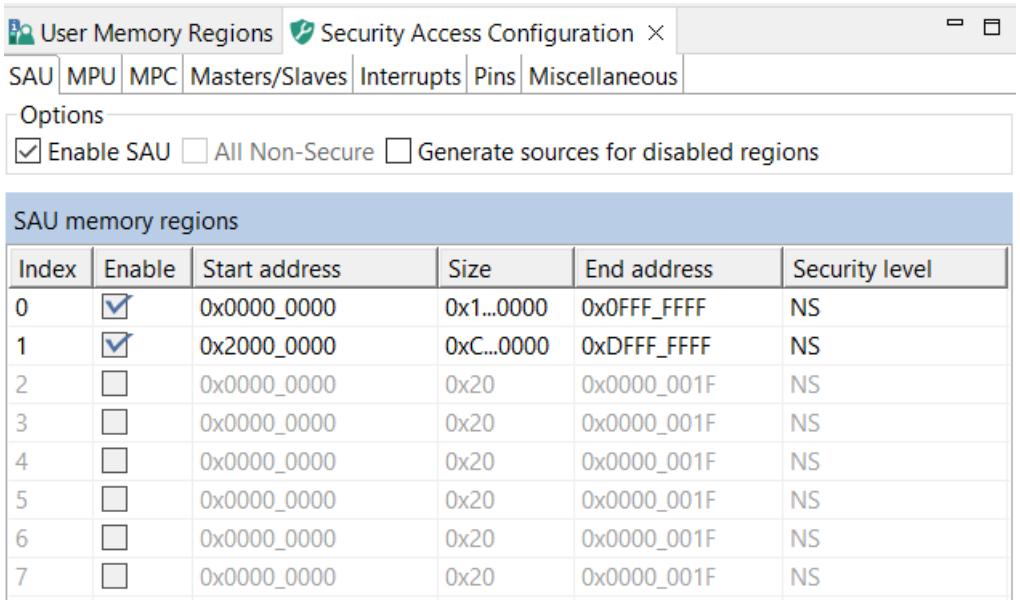
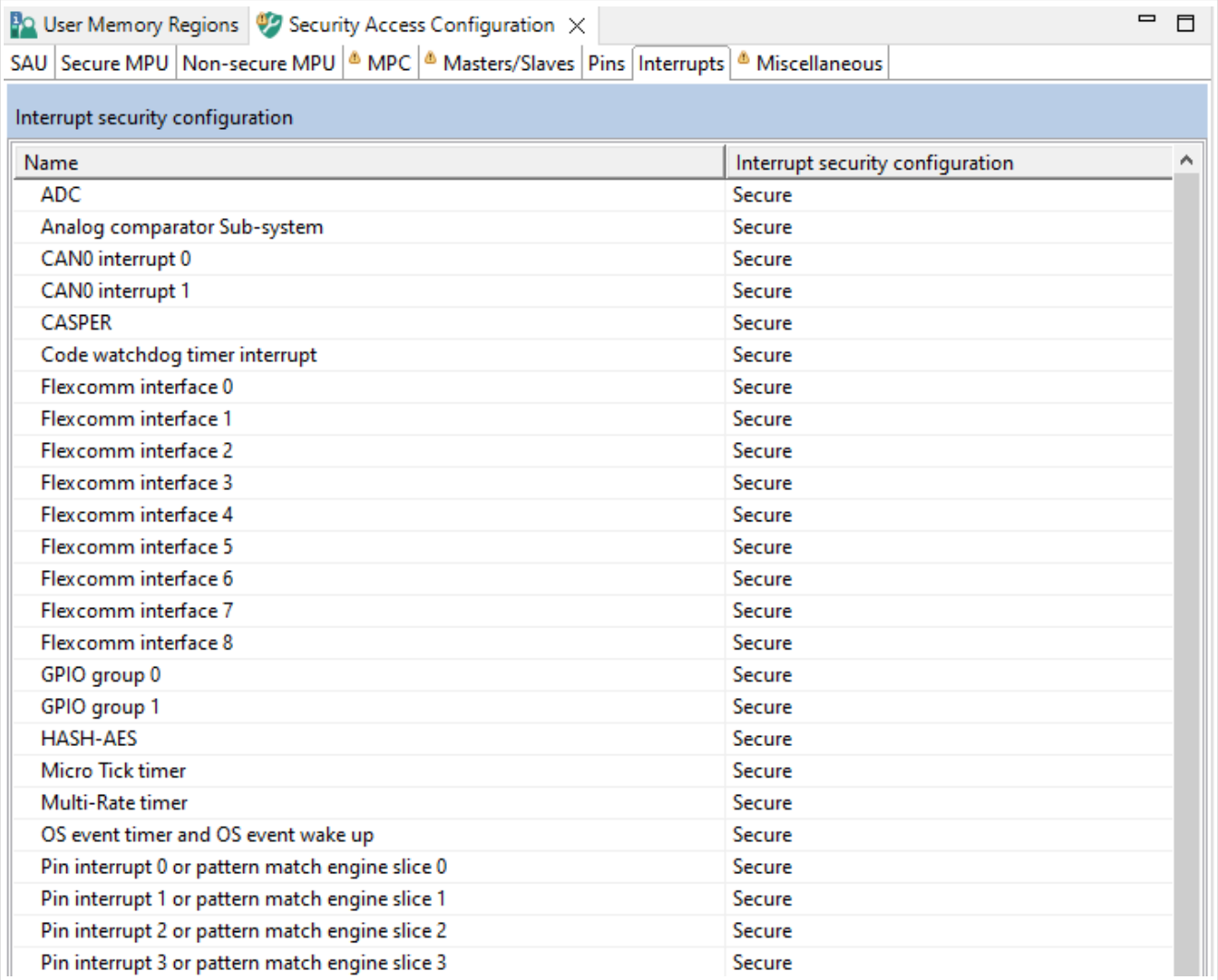


Figure 130. SAU/IDAU

7.1.2.2 Interrupts

In the **Interrupts** subview, you can set the security designation for the device's peripheral interrupts. If the processor contains more than one core or processing unit, additional **Handling by Core** tables may appear. In these tables, you can specify whether the interrupts 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. Alternatively, you can use **Shift+Up/Down** after selecting the row to expand the selection.



Name	Interrupt security configuration
ADC	Secure
Analog comparator Sub-system	Secure
CAN0 interrupt 0	Secure
CAN0 interrupt 1	Secure
CASPER	Secure
Code watchdog timer interrupt	Secure
Flexcomm interface 0	Secure
Flexcomm interface 1	Secure
Flexcomm interface 2	Secure
Flexcomm interface 3	Secure
Flexcomm interface 4	Secure
Flexcomm interface 5	Secure
Flexcomm interface 6	Secure
Flexcomm interface 7	Secure
Flexcomm interface 8	Secure
GPIO group 0	Secure
GPIO group 1	Secure
HASH-AES	Secure
Micro Tick timer	Secure
Multi-Rate timer	Secure
OS event timer and OS event wake up	Secure
Pin interrupt 0 or pattern match engine slice 0	Secure
Pin interrupt 1 or pattern match engine slice 1	Secure
Pin interrupt 2 or pattern match engine slice 2	Secure
Pin interrupt 3 or pattern match engine slice 3	Secure

Figure 131. Interrupts

7.1.2.3 Secure/Non-secure MPU

In the **Secure MPU** and **Non-secure MPU** sub-views, you can enable and configure MPU (Memory Protection Unit). You can create regions, specify their address, size, and other parameters. Use the **Secure MPU** sub-view for the configuration of the secure, and **Non-secure MPU** for the configuration of the non-secure security level.

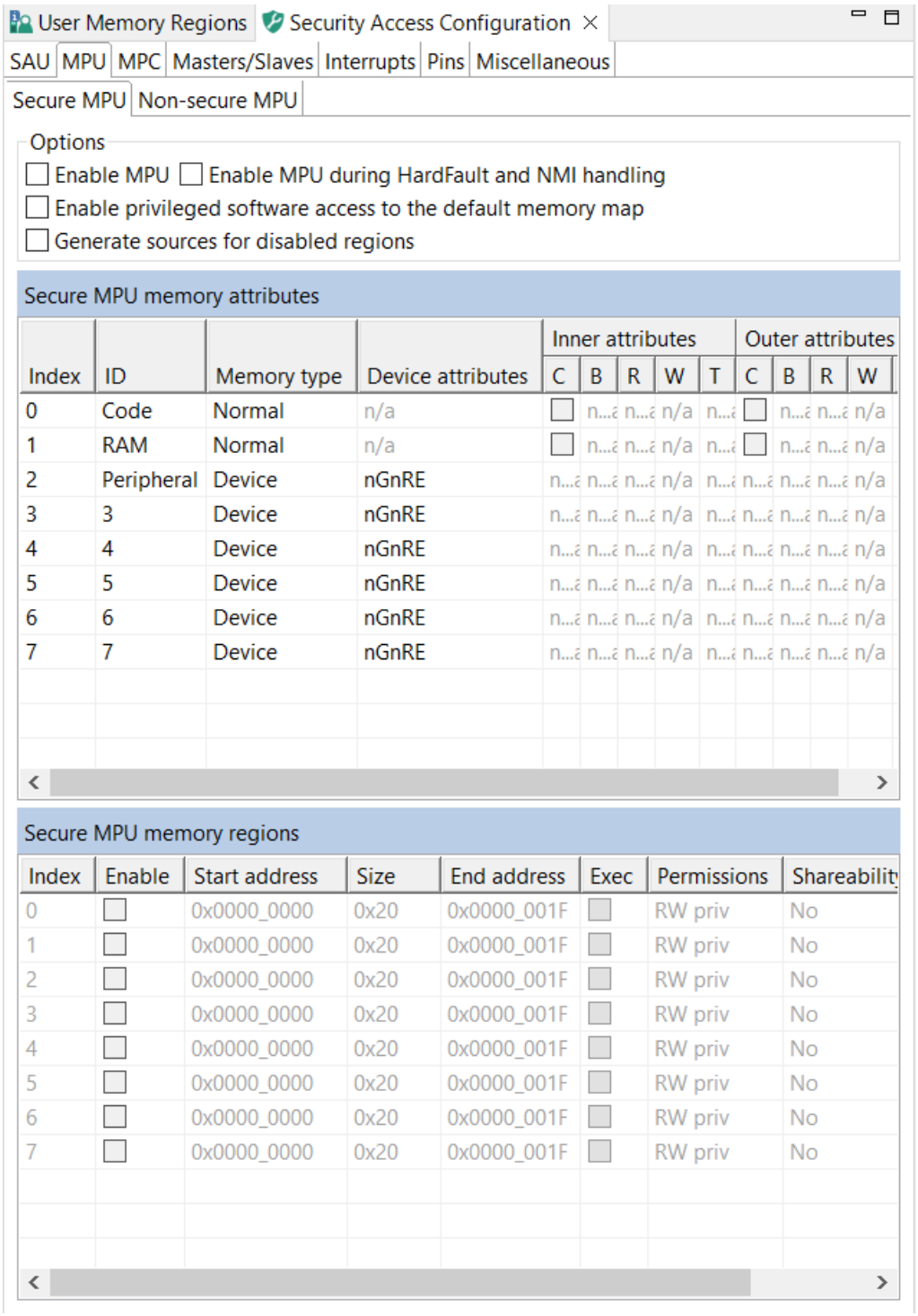


Figure 132. MPU

MPU is disabled by default and must be enabled by selecting the **Enable MPU** option.

Note: Not every device supports MPU.

Use the **MPU Memory Attributes** table to name and configure MPU memory attribute sets. Click the cells of the **Memory Type** and **Device Attributes** columns to display the available choices.

Use the **MPU Memory Regions** table to enable and configure MPU memory regions.

1. **Enable** the region.
2. Specify the **Address**.
3. Specify either the **Size** or the **End Address**.
4. Set the **Exec** option if you want the region to be able to run code.
5. Set the **Permissions** (Read Only or Read/Write).
6. Set the **Privileges**.
   Note: Privileged access can be set by default for all memory regions not handled by MPU by selecting the **Enable privileged software access to the default memory map** option.
7. Set the **Shareability**, or the caching options.
8. Allocate one of the sets from the **MPU Memory Attributes** table in **Mem.Attr.**. Sets can be allocated to more than one region.

7.1.2.4 MPC

In the **MPC** (Memory Protection Checker) subview, 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

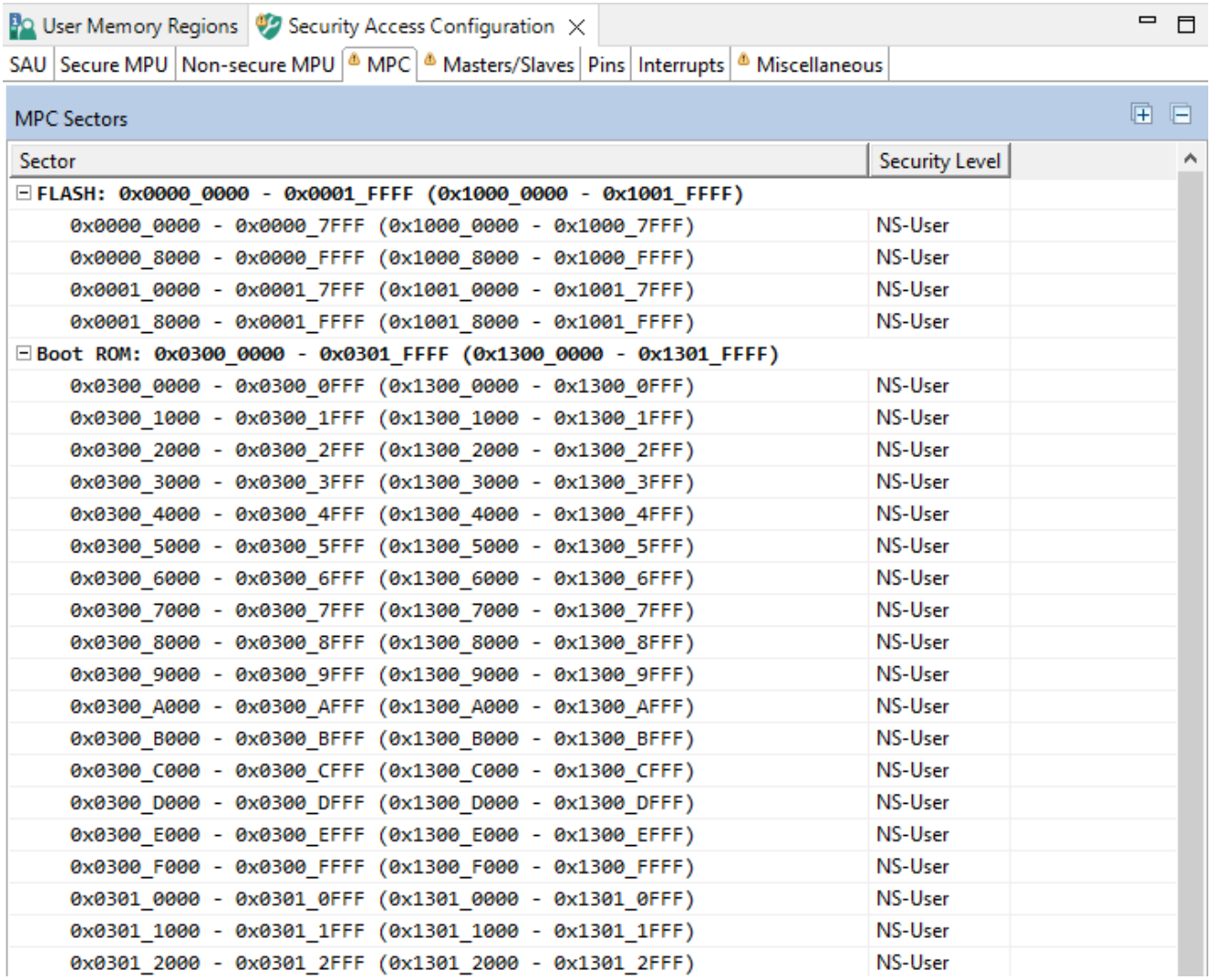


Figure 133. MPC

7.1.2.5 Masters/Slaves

In the **Masters/Slaves** subview, 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 the simple bus master to read and write at the same level only. Clear to allow reading and writing at the same and lower levels.
- **Smart Master in Strict Mode** - Select to allow the smart bus master to execute, read, and write to memory at the same level only. De-select to allow executing at the same level only, and reading and writing at the same and lower levels.

Note: The instruction-type bus master security level must equal the bus slave security level. The data and other security levels must be equal to or higher than the bus slave security level.

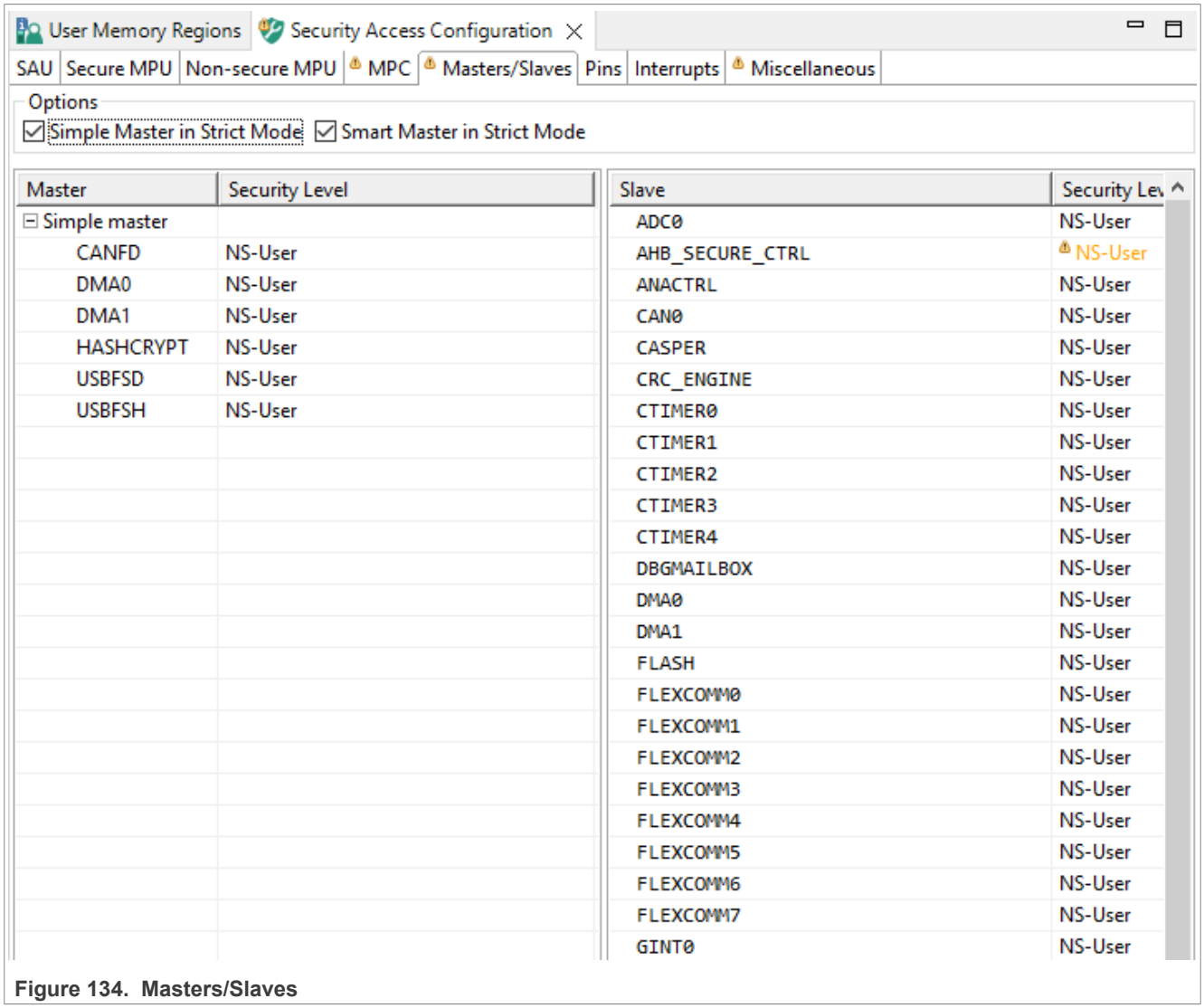


Figure 134. Masters/Slaves

7.1.2.6 Pins

In the **Pins** subview, you can specify whether 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. Alternatively, you can use **Shift +Up/Down** after selecting the row to expand the selection.

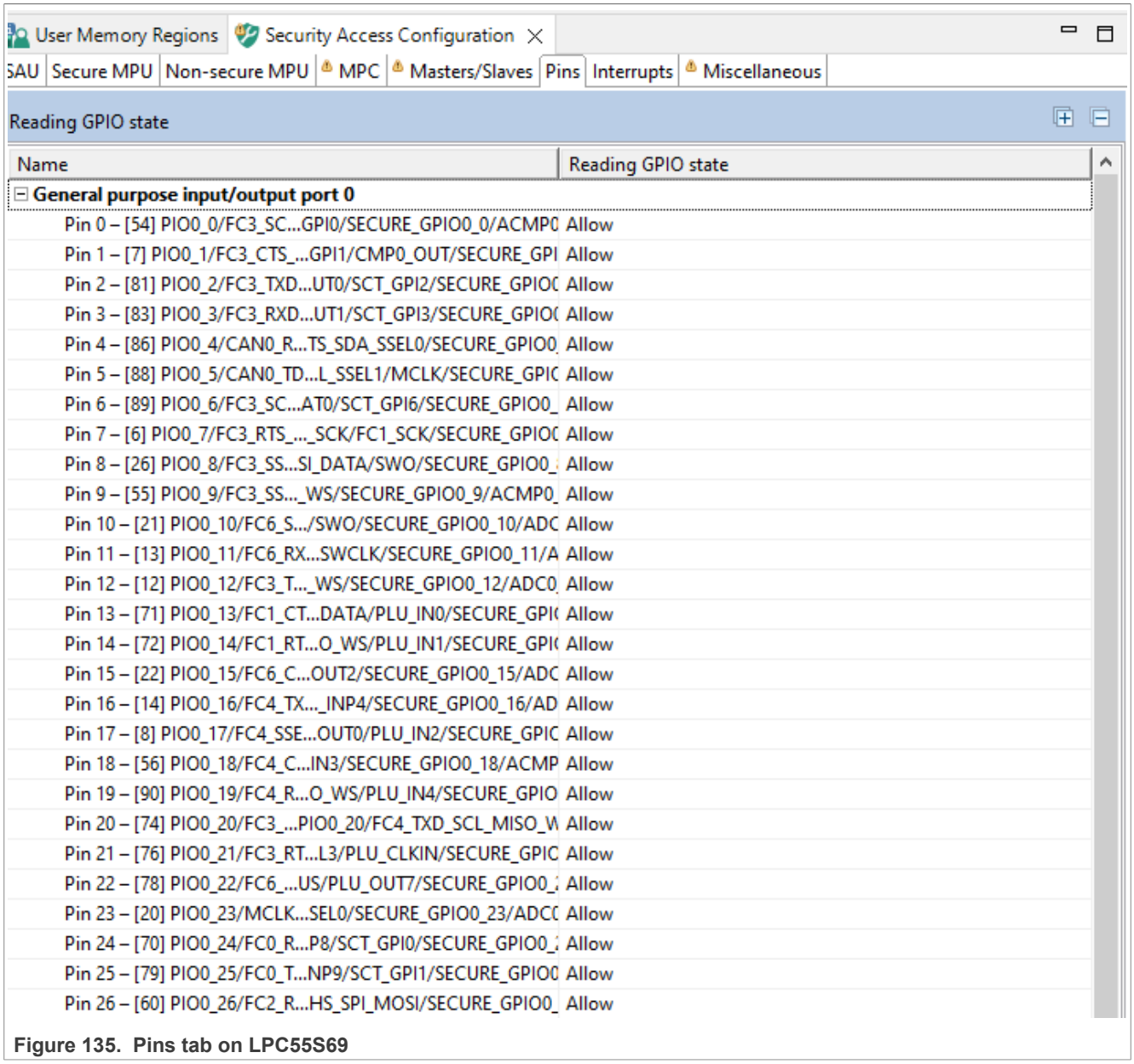


Figure 135. Pins tab on LPC55S69

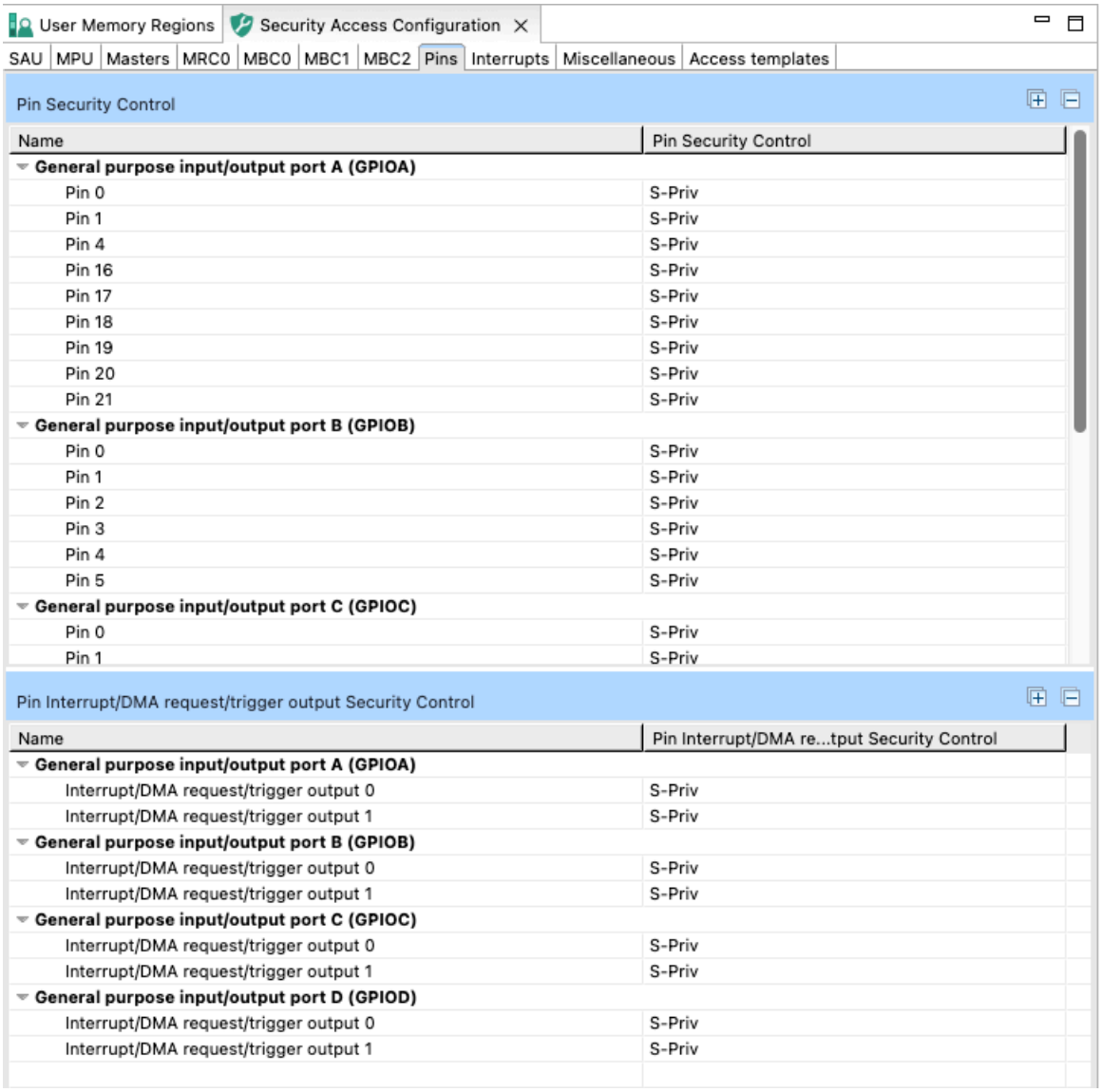


Figure 136. Pins tab on KW45

7.1.2.7 Trigger sources

In the **Triggers** subview, configure triggers of the Intrusion and Tamper Response Controller (ITRC) that provides a mechanism to configure the response action for an intrusion event detected by on-chip security sensors. The rows in the table represent input signals - explicit and implicit intrusion event detectors that generate a level and a latched signal toward the interrupt controller. Output signals are represented by columns and contain two configurable fields:

- Signal selected shows if the input signal is selected as a trigger or not.
- Writable field shows whether the input signal field is writable. Once locked, the field cannot be changed until a reset to ITRC is asserted.

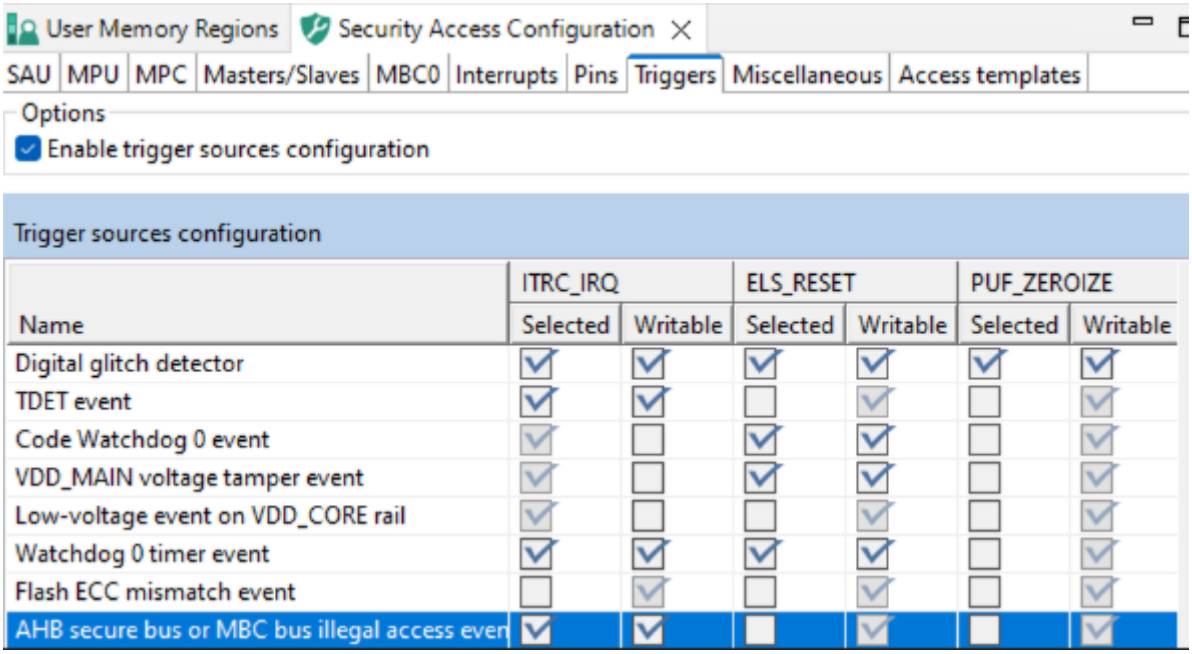


Figure 137. Triggers subview

All trigger source fields are set to “the signal is not selected” and “the signal field is writable” by default. If you want to change the trigger sources setting, left-click the **Selected/Writable** check-box.

Note: The check-box can be disabled to prevent the non-selected and non-writable states.

7.1.2.8 Miscellaneous

In the **Miscellaneous** subview, 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 that 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.

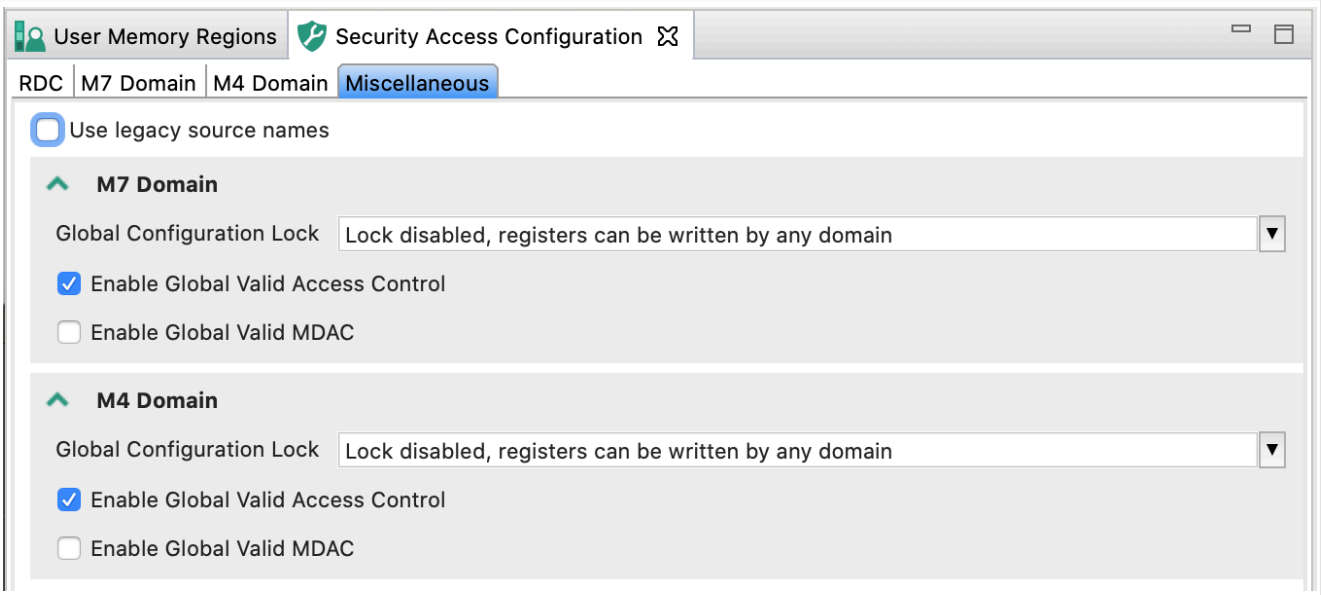


Figure 138. Miscellaneous (RDC+XRDC2)

7.1.2.8.1 TEE global options

There are several global options available for the user:

1. The **output type** list lets the user select which type they would like to generate (C code or ROM preset).
2. The **use legacy source names** checkbox lets the user switch between the current source names `resource_config.c` or `tzm_config.c` for legacy projects.
3. The **use instruction glitch resilient code for register writes** checkbox lets the user generate more resilient (instruction glitch) code of registers writes.

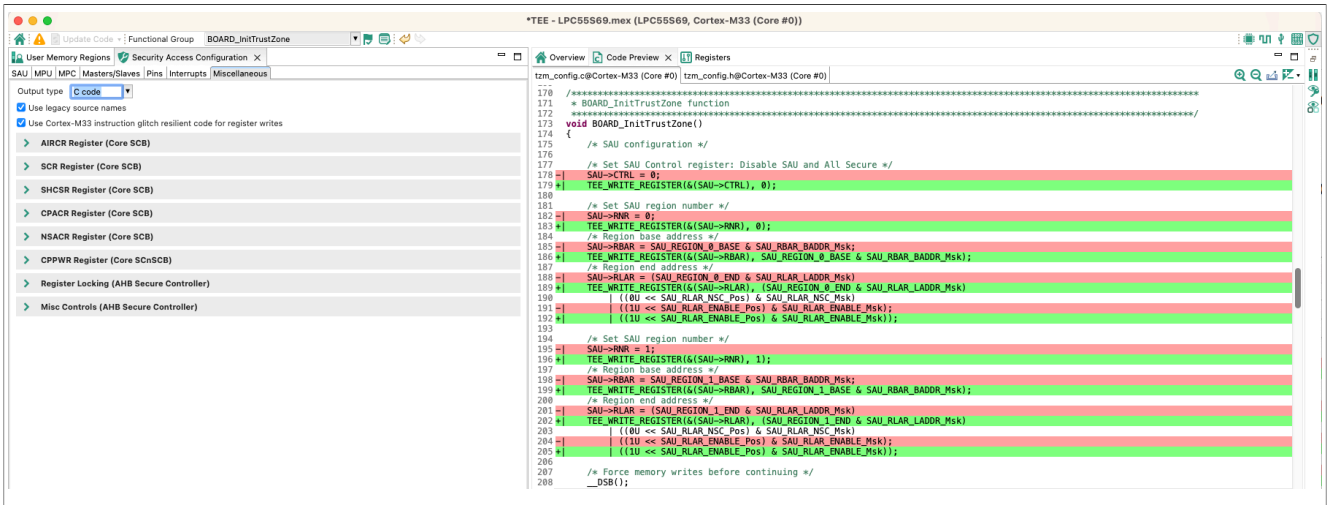


Figure 139. TEE global options

7.1.3 Memory attribution map

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

7.1.3.1 Core 0

In the **Core 0** subview, 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 access 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 that you want to review by choosing from the **Master** dropdown menu.
3. Optionally, set the security state and execution privilege check-boxes when master allows more security levels. This setting has no effect on the configuration.
4. Optionally, customize the output by de-selecting the **Show details** and **Merged SAU+IDAU** options.
5. 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**.

Figure 140. Core 0

7.1.3.2 Simple and Smart masters

In the **Simple Masters** and **Smart Masters subviews**, you can review security attributes of memory in relation to access rights by simple/smart masters. 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 that 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**, **Show Peripherals**, and **This Domain Only** 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**.

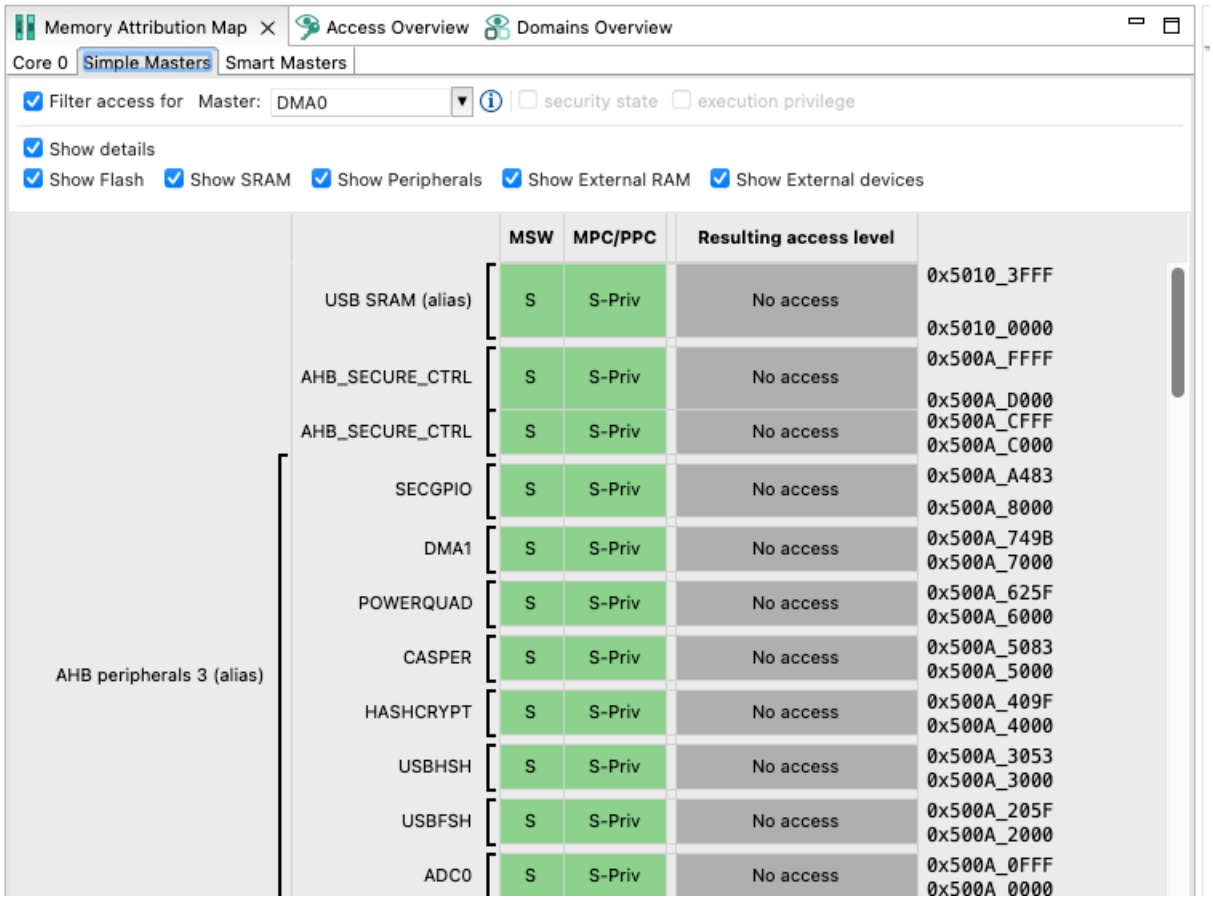


Figure 141. Simple/Smart masters

7.1.4 Access overview

In **Access Overview**, you can review the security policies you have set in the **Security Access Configuration** view. The vertical axis displays all masters, divided into color-coded groups by their security settings.

The horizontal 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.

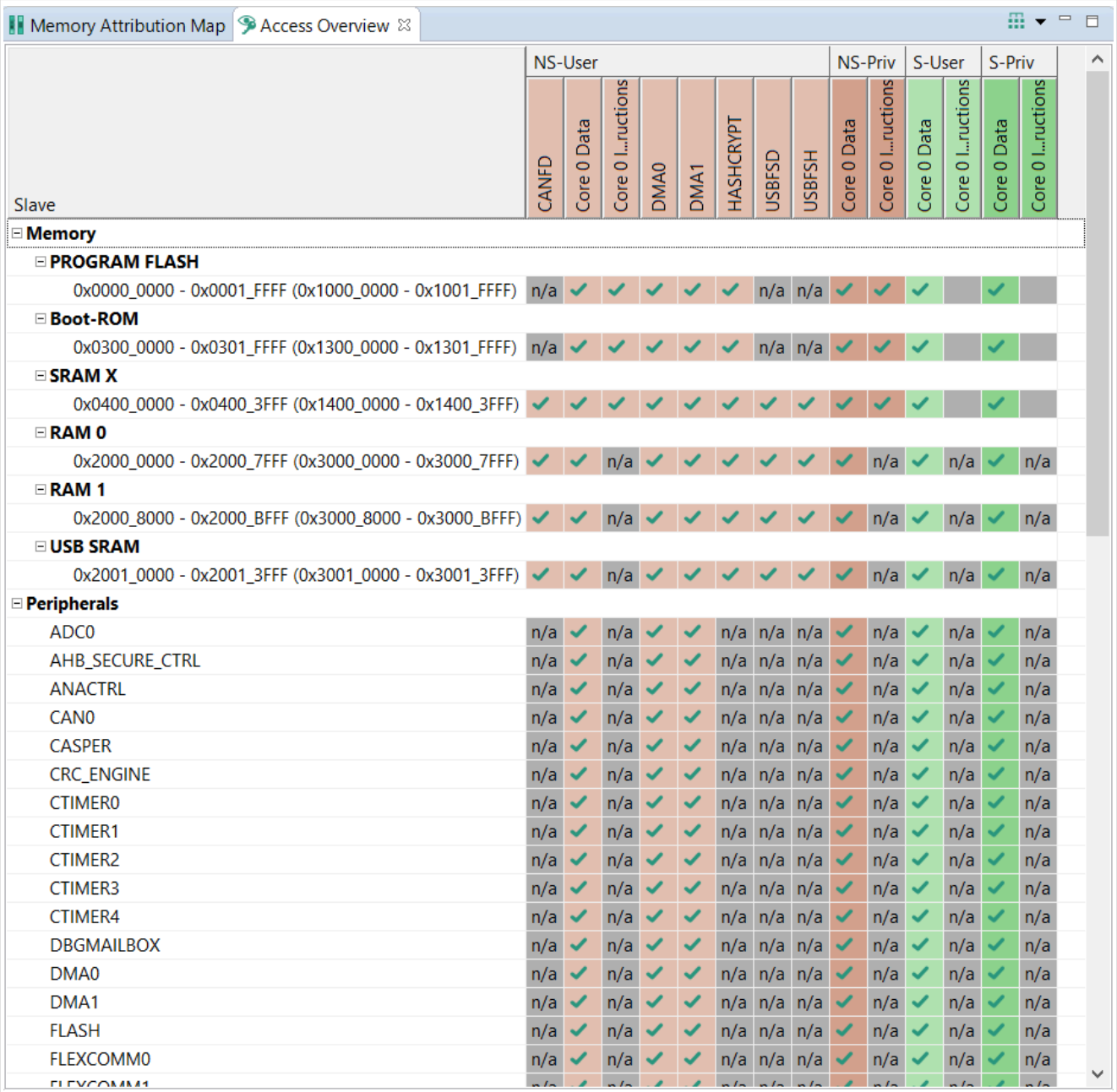


Figure 142. Access Overview

7.1.5 Code generation

If the settings are correct and no error is reported, the code generation engine regenerates the source code. You can view the resulting code in 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. Such features as Copy, Search, Zoom-in, Zoom-out, and Export source are available in the **Code Preview** view. The search can also be invoked by CTRL+F or from the context menu.

Some AHBSC and TRDC with security extension-enabled 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** subview.

7.2 RDC-enabled devices

The features and appearance of the TEE tool are based on the security model of the loaded device.

This section describes the features and appearance of the tool devices enabled with RDC (Resource Domain Controller) and XRDC2 (eXtended Resource Controller 2).

Currently, the following devices of this type are supported:

- RT1170
  - Dual core (Cortex-M7 + Cortex-M4): MIMXRT1176, MIMXRT1175, MIMXRT1173
  - Single core only (Cortex-M7): MIMXRT1172, MIMXRT1171
- KW45
- RT118x
- i.MX93

7.2.1 User Memory Regions view

In the **User Memory Regions** view, you can create and maintain a high-level configuration of memory regions and their access templates. 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.

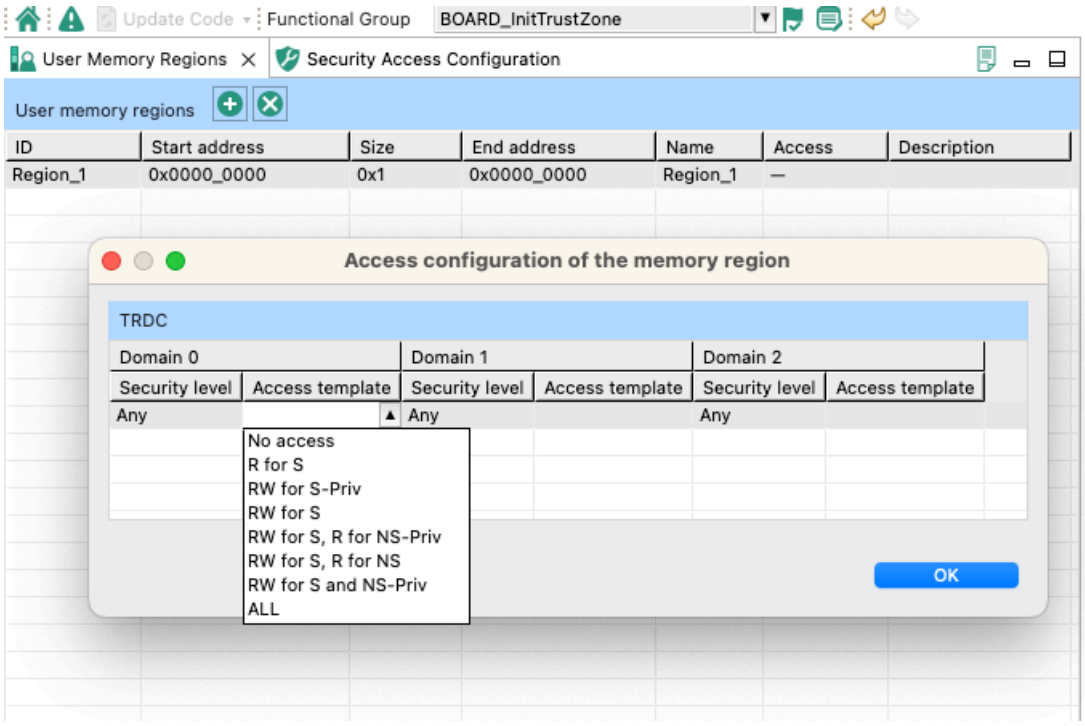


Figure 143. User Memory Regions

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.

Modify the access policy of memory regions by clicking the cell in the **Access** column. This action opens the Access templates dialog.

Errors in configuration are highlighted by a red icon in the relevant cell. If the issue can be easily fixed, you can right-click the cell to display a dropdown list of suggested solutions.

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

7.2.1.1 Access templates

In the **Access templates** dialog, you can modify access templates for device domains. The dialog displays the device RDC domains, as well as all user-created XRDC2 domains.

Note: Make sure to first specify the number of domains in the **M4 Domain/M7 Domain > Domains**.

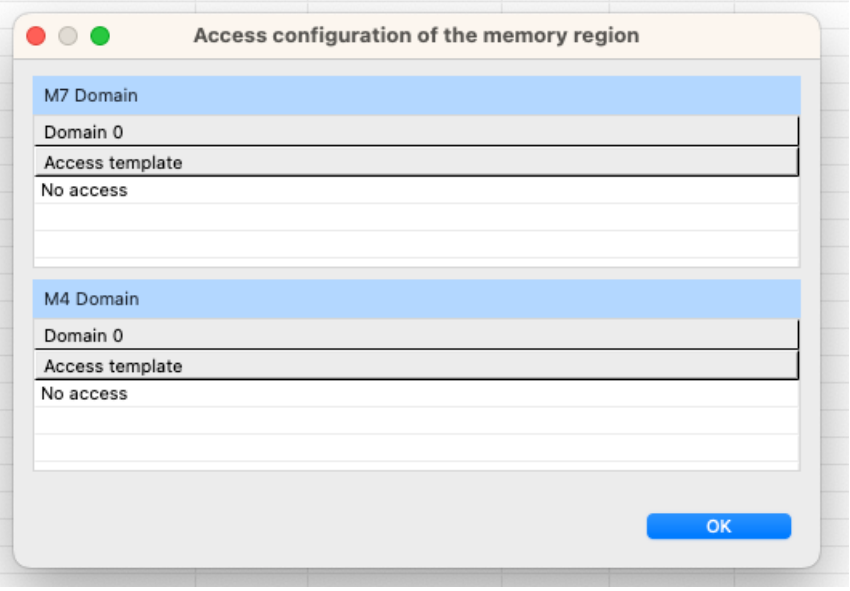


Figure 144. Access template

Select an access template by clicking the topmost cell of the domain column to open a dropdown list containing all options.

Once you have selected access templates for all domains, click **OK** to return to the **User Memory Regions** view.

7.2.2 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.

7.2.2.1 RDC

In the **RDC** subview, you can assign masters to domains and specify access rules for slaves for each domain.

7.2.2.1.1 RDC Masters

In the **RDC Masters** subview, you can view available bus masters, allocate them to available domains (cores), and lock/unlock the allocation.

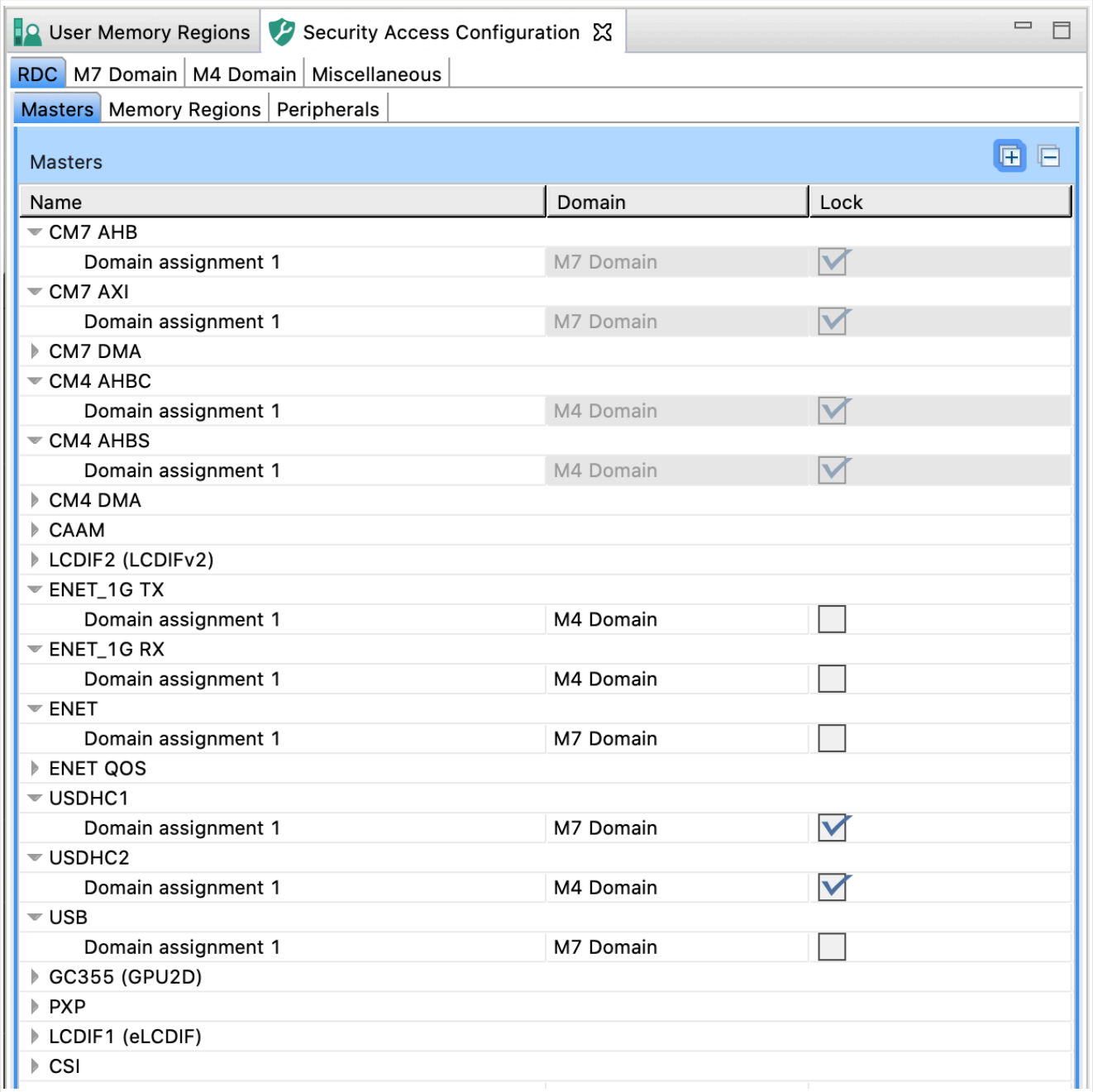


Figure 145. RDC Masters

Allocate a master to a domain by clicking the cell in the **Domain** column in the **Masters** table and selecting the domain from the dropdown list.

Select the **Lock** checkbox to prevent further register modifications.

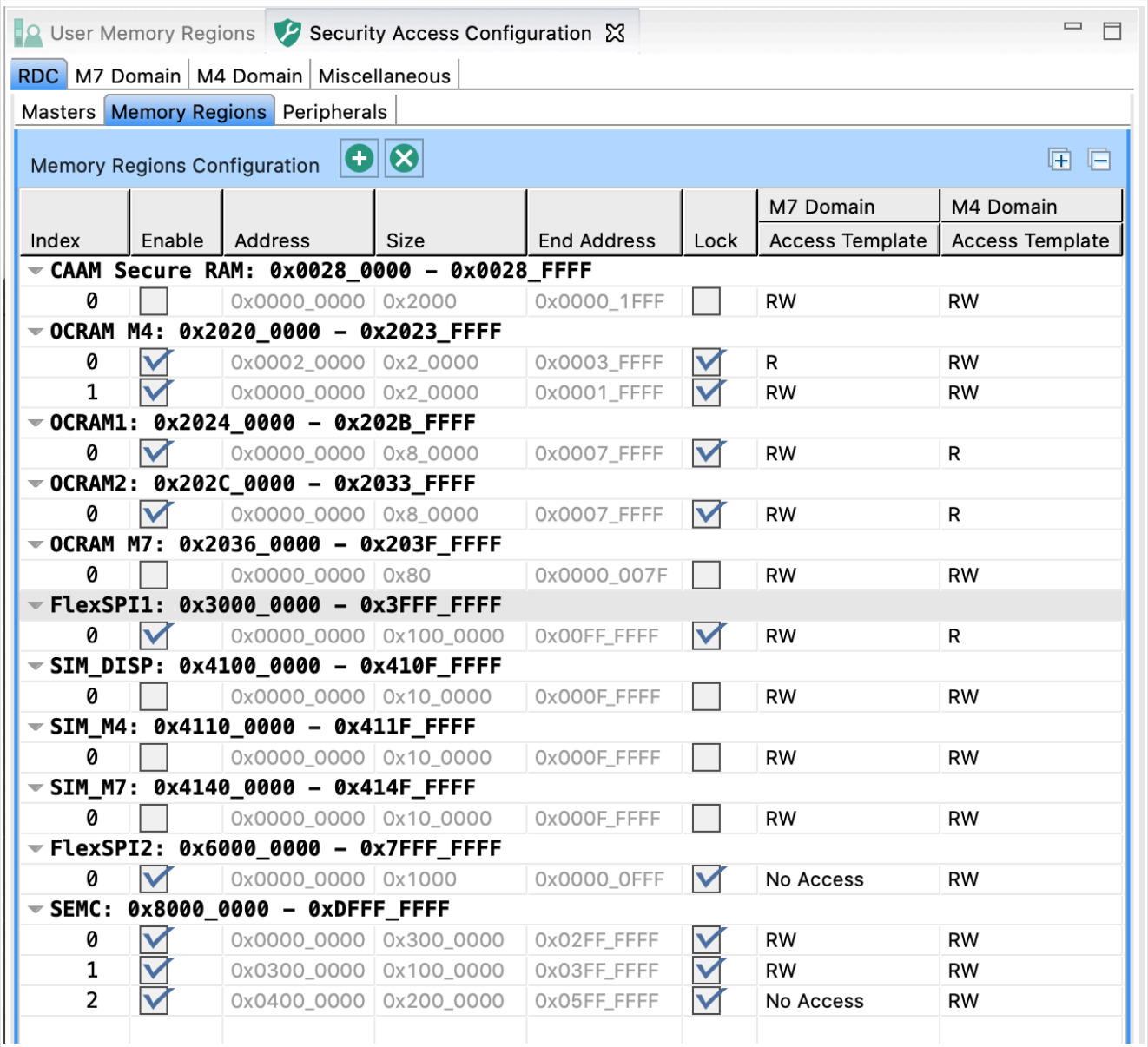
Alternatively, you can select the options by right-clicking the master and using the dropdown list.

Note: Some masters are allocated to specific domains by default and cannot be reallocated.

7.2.2.1.2 Memory Regions

In the **Memory Regions** subview, you can view, enable/disable, and configure the MRC (Memory Region Controller) bus slaves and their domain access.

Memory Region Controller implements the access controls for slave memories based on the pre-programmed Memory Region Descriptor registers.



Index	Enable	Address	Size	End Address	Lock	M7 Domain Access Template	M4 Domain Access Template
CAAM Secure RAM: 0x0028_0000 – 0x0028_FFFF							
0	☐	0x0000_0000	0x2000	0x0000_1FFF	☐	RW	RW
OCRAM M4: 0x2020_0000 – 0x2023_FFFF							
0	☑	0x0002_0000	0x2_0000	0x0003_FFFF	☑	R	RW
1	☑	0x0000_0000	0x2_0000	0x0001_FFFF	☑	RW	RW
OCRAM1: 0x2024_0000 – 0x202B_FFFF							
0	☑	0x0000_0000	0x8_0000	0x0007_FFFF	☑	RW	R
OCRAM2: 0x202C_0000 – 0x2033_FFFF							
0	☑	0x0000_0000	0x8_0000	0x0007_FFFF	☑	RW	R
OCRAM M7: 0x2036_0000 – 0x203F_FFFF							
0	☐	0x0000_0000	0x80	0x0000_007F	☐	RW	RW
FlexSPI1: 0x3000_0000 – 0x3FFF_FFFF							
0	☑	0x0000_0000	0x100_0000	0x00FF_FFFF	☑	RW	R
SIM_DISP: 0x4100_0000 – 0x410F_FFFF							
0	☐	0x0000_0000	0x10_0000	0x000F_FFFF	☐	RW	RW
SIM_M4: 0x4110_0000 – 0x411F_FFFF							
0	☐	0x0000_0000	0x10_0000	0x000F_FFFF	☐	RW	RW
SIM_M7: 0x4140_0000 – 0x414F_FFFF							
0	☐	0x0000_0000	0x10_0000	0x000F_FFFF	☐	RW	RW
FlexSPI2: 0x6000_0000 – 0x7FFF_FFFF							
0	☑	0x0000_0000	0x1000	0x0000_0FFF	☑	No Access	RW
SEMC: 0x8000_0000 – 0xDFFF_FFFF							
0	☑	0x0000_0000	0x300_0000	0x02FF_FFFF	☑	RW	RW
1	☑	0x0300_0000	0x100_0000	0x03FF_FFFF	☑	RW	RW
2	☑	0x0400_0000	0x200_0000	0x05FF_FFFF	☑	No Access	RW

Figure 146. Memory Regions

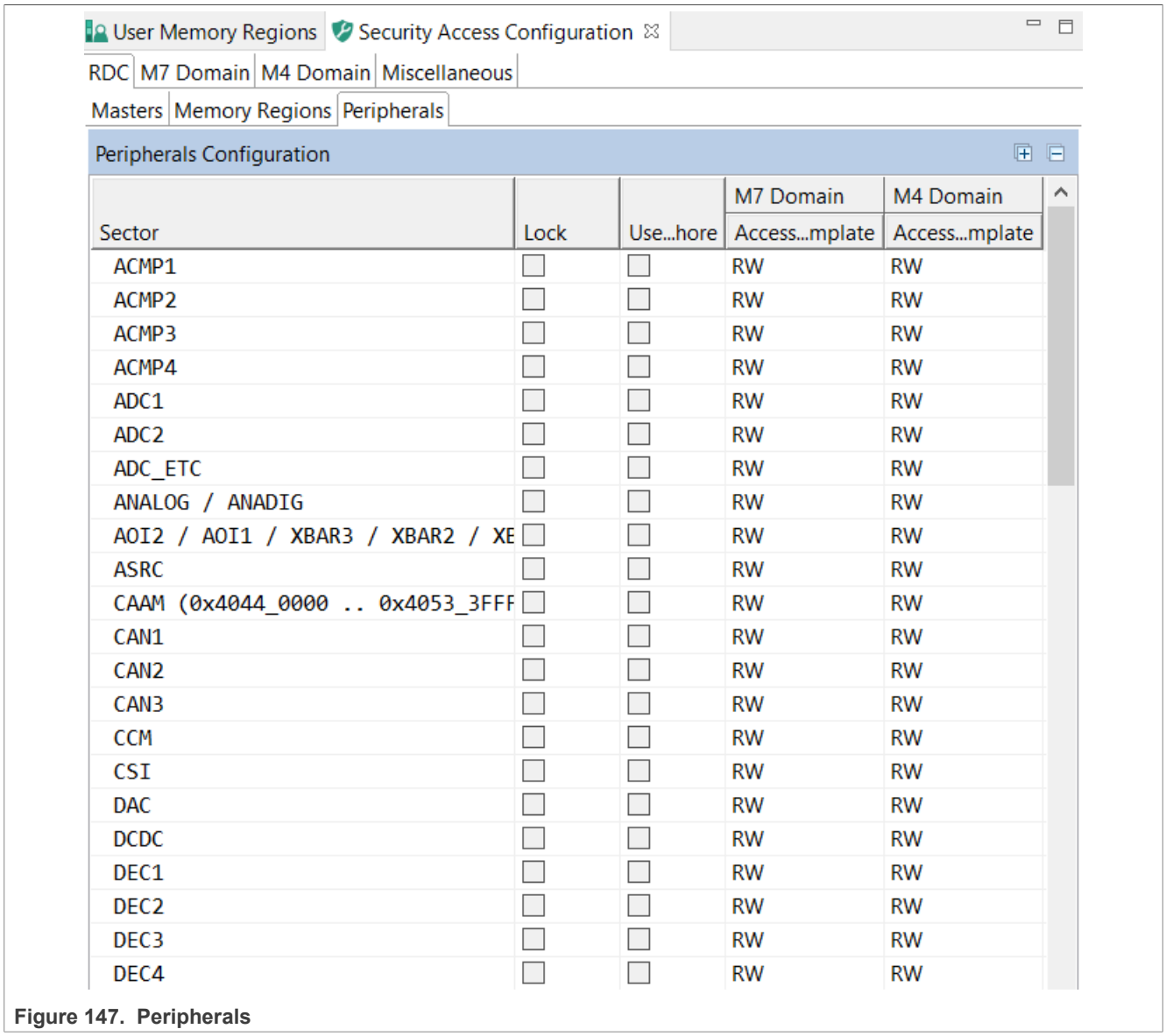
Use the **Memory Regions Configuration** table to enable and configure MRC slaves:

1. **Enable** the region.
2. Specify the **Address**.
3. Specify either the **Size** or the **End Address**.
4. Optional: **Lock** the settings to prevent further register modifications.
5. Set the **Access Template** for available domains.

Alternatively, you can select the options by right-clicking the master and using the dropdown list.

7.2.2.1.3 Peripherals

In the **Peripherals** subview, you can view and configure the PDAP (Peripheral Domain Access Permissions) for peripherals.



Sector	Lock	Use...hore	M7 Domain Access...mplate	M4 Domain Access...mplate
ACMP1	☐	☐	RW	RW
ACMP2	☐	☐	RW	RW
ACMP3	☐	☐	RW	RW
ACMP4	☐	☐	RW	RW
ADC1	☐	☐	RW	RW
ADC2	☐	☐	RW	RW
ADC_ETC	☐	☐	RW	RW
ANALOG / ANADIG	☐	☐	RW	RW
AOI2 / AOI1 / XBAR3 / XBAR2 / XE	☐	☐	RW	RW
ASRC	☐	☐	RW	RW
CAAM (0x4044_0000 .. 0x4053_3FFF	☐	☐	RW	RW
CAN1	☐	☐	RW	RW
CAN2	☐	☐	RW	RW
CAN3	☐	☐	RW	RW
CCM	☐	☐	RW	RW
CSI	☐	☐	RW	RW
DAC	☐	☐	RW	RW
DCDC	☐	☐	RW	RW
DEC1	☐	☐	RW	RW
DEC2	☐	☐	RW	RW
DEC3	☐	☐	RW	RW
DEC4	☐	☐	RW	RW

Figure 147. Peripherals

Use the **Peripherals Configuration** table to enable and configure PDAP:

1. Optional: **Lock** the settings to prevent further register entries.
2. Select **Use semaphore** to enable the semaphore function for the peripheral.
   Note: When enabled, the master cannot access this peripheral until obtaining a semaphore. During the time that the domain has the semaphore in possession, its bus masters have exclusive access to the peripheral.
3. Set the **Access Template** for available domains.

7.2.2.2 XRDC2 Domains view

In the **M7/M4 Domain** subviews, you can view and configure security policies of the XRDC2(eXtended Resource Domain Controller 2) domains. Each CPU can contain up to 16 domains.

7.2.2.2.1 MPU

In the **MPU** subview, you can enable and configure MPU (Memory Protection Unit). You can create regions, specify their address, size, and other parameters.

The MPU enforces privilege rules, separates processes, enforces access rules to memory, and supports the standard ARMv7 Protected Memory System Architecture model.

MPU is disabled by default and must be enabled by selecting the **Enable MPU** option.

Note: Not every device supports MPU.

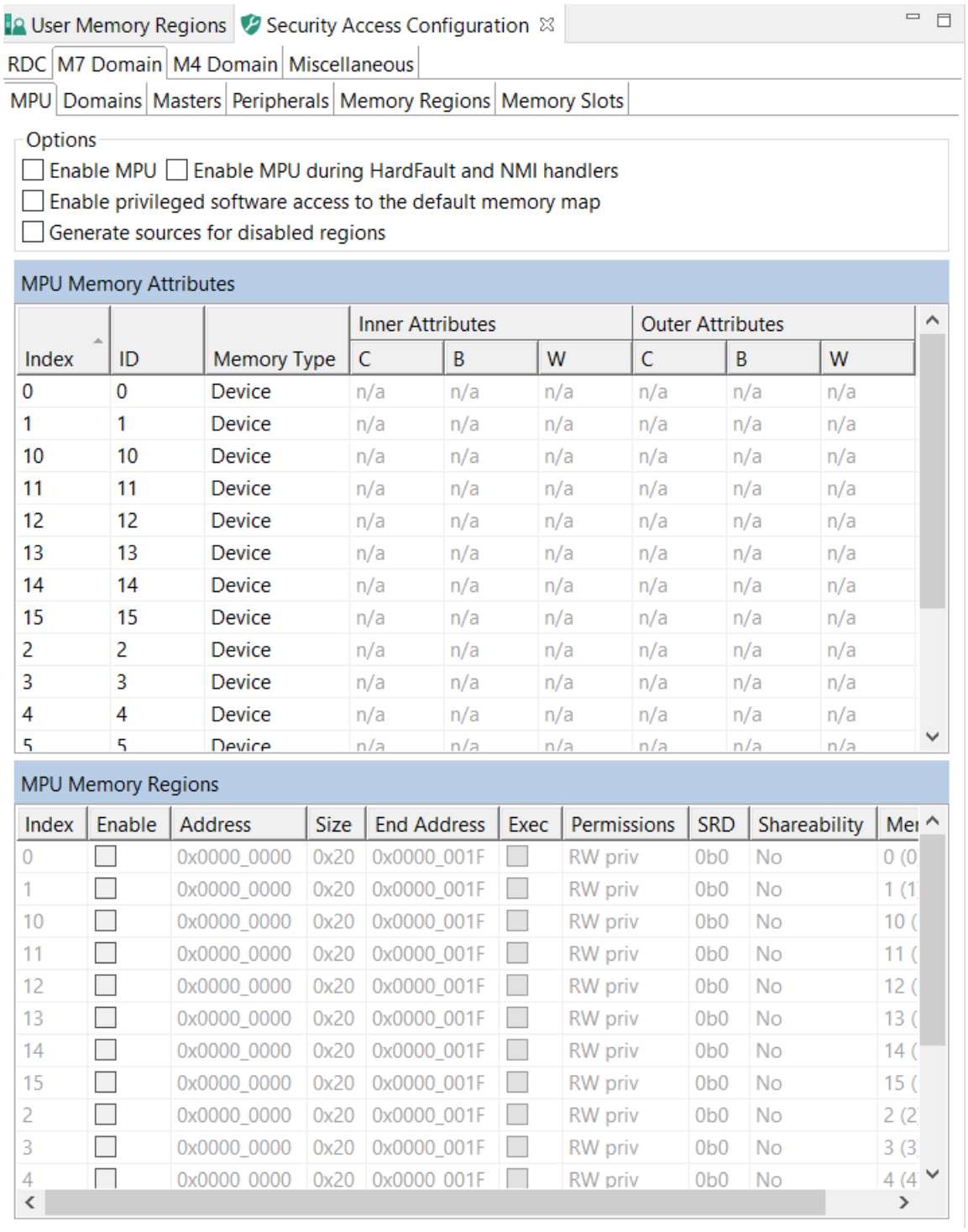


Figure 148. MPU

Use the **MPU Memory Attributes** table to name and configure MPU memory attribute sets. Click the cells of the **Memory Type** and **Inner/Outer Attributes** columns to display the available options.

Use the **MPU Memory Regions** table to enable and configure MPU memory regions.

1. **Enable** the region.

2. Specify the **Address**.
3. Specify either the **Size** or the **End Address**.
4. Set the **Exec** option if you want the region to be able to run code.
5. Set the **Permissions**.
6. Set the **SRD** (Sub Region Disable) bits.
7. Set the **Shareability**, or the caching options.

7.2.2.2.2 Domains

In the **Domains** subview, you can view, add/remove, and rename XRDC2 domains. Each CPU supports up to 16 XRDC2 domains.

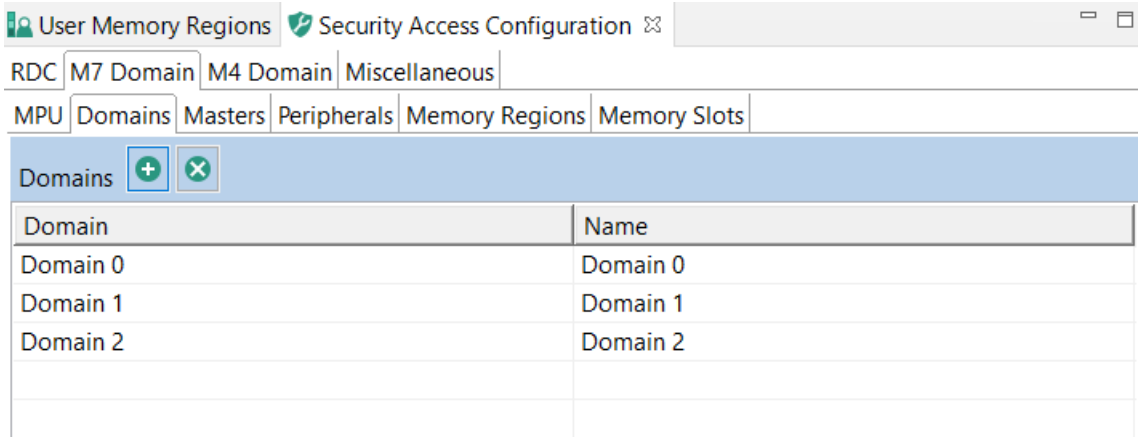


Figure 149. Domains

Add a new domain by clicking the **Add new domain** button.

Rename the domain by entering a new name in the **Name** column.

Remove a domain by clicking the **Remove last domain** button.

7.2.2.3 Masters

In the **Masters** subview, you can add/remove, view, configure XRDC2 domain assignments to available RDC masters.

Master Domain Assignment Controller (MDAC) is responsible for the generation of the DID, nonsecure and privileged attributes for every system bus transaction in the device based on pre-programmed Master Domain Assignment (MDA) registers.

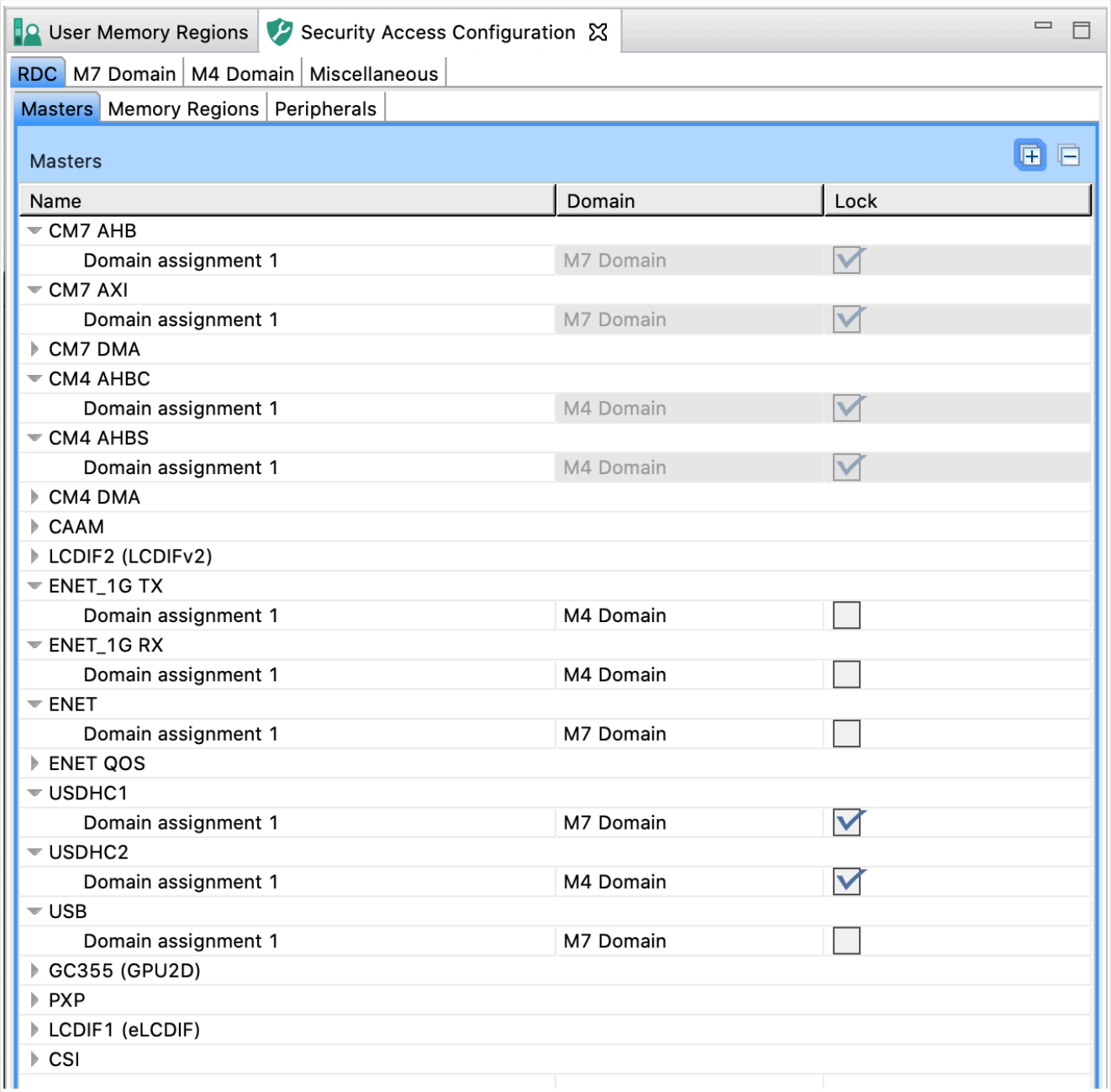


Figure 150. Masters

To add a new domain assignment:

1. Click the **Add new domain assignment for the selected master** button.
2. Select the **Enable** checkbox.
3. Enter the **Match Input** value.
   Note: The match field specifies the reference value for the comparison with the MDAC match input. The match field width varies by MDAC instance from 0 to 16 bits. Unimplemented bits are read as 0. A size of 0 bits generates a hit on all comparisons.
4. Enter the **Mask Input** value.

Note: The mask field specifies which bits are valid for the match comparison. Only bit positions in which the mask value is zero are compared. The mask field width is the same as the mask field which varies by MDAC instance from 0 to 16 bits. A mask value of all ones generates a hit on all comparisons.

5. Select the XRDC2 domain assignment from the dropdown list in the **Domain** column.
6. Select the security access type from the dropdown list in the **Secure** column.
7. Select the privileged access type from the dropdown list in the **Privileged** column.
8. Optional: select the **Lock** checkbox to prevent further register modifications.

7.2.2.2.4 Peripherals

In the **Peripherals** subview, you can view the access templates for PAC (Peripheral Access Controller) and configure access for all peripherals managed by PAC on the selected RDC domain.

The Peripheral Access Controller submodule performs access control for a set of peripherals connected to a peripheral bus bridge or integrated into a peripheral subsystem.

The **Access Template** table displays the ID and name of all access templates available for the PAC on the selected device. The information is data driven and display-only.

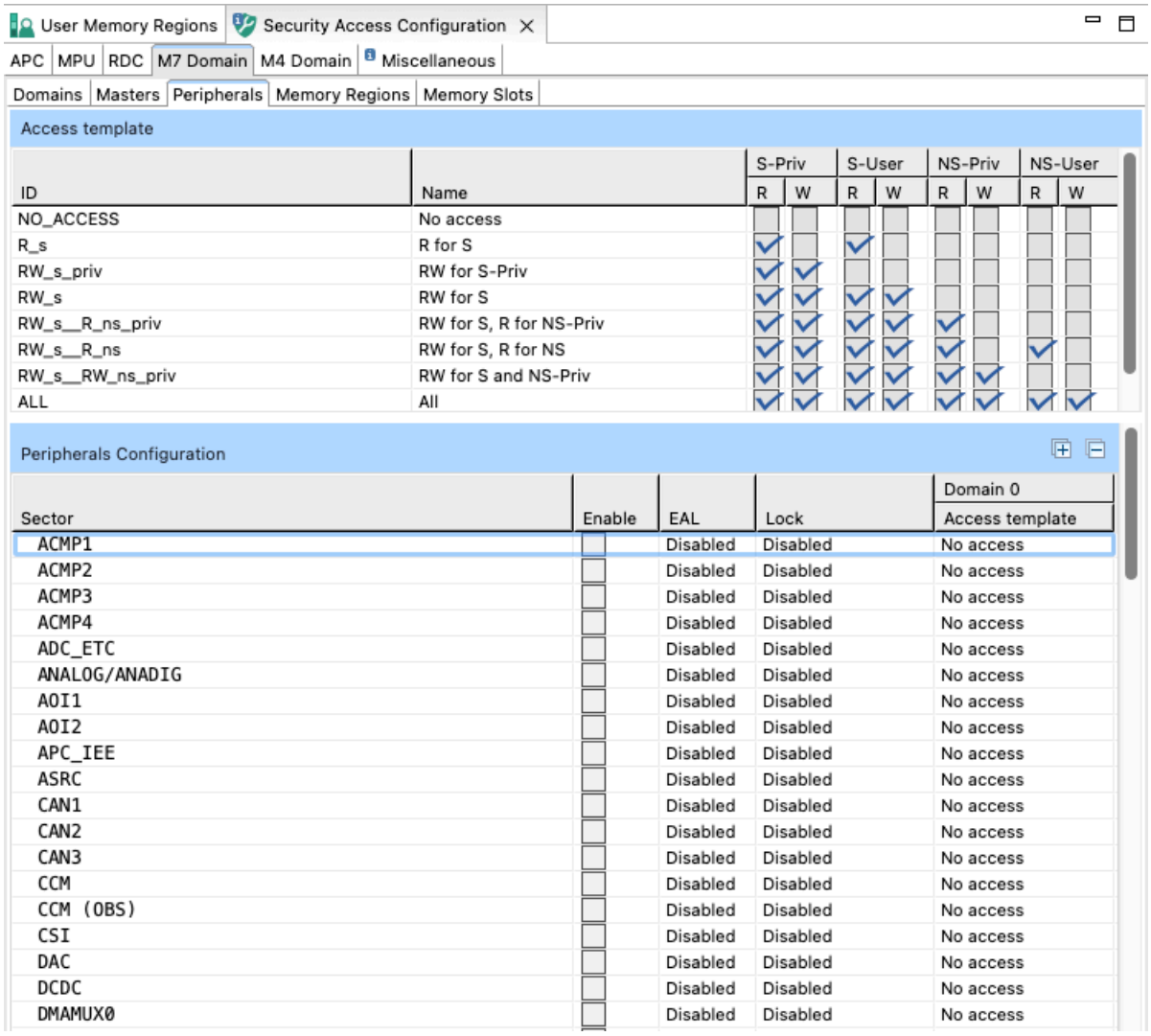


Figure 151. Peripherals

Use the **Peripherals Configuration** table to configure access for a peripheral:

1. Select the **Enable** checkbox.
2. Set the **Lock** to the desired state.
3. Set the **Access Template** for all listed domains.

Alternatively, you can select the options by right-clicking the master and using the dropdown list.

7.2.2.2.5 Memory Regions

In the **Memory Regions** subview, you can view the access templates for MRC (Memory Region Controller) and configure access for all non-peripheral memory spaces managed by MRC on the selected RDC domain.

The Memory Region Controller (MRC) provides domain-based, hardware access control for all system bus references targeted at non-peripheral memory spaces.

The **Access Template** table displays the ID and name of all access templates available for the MRC on the selected device. The information is data driven and display-only.

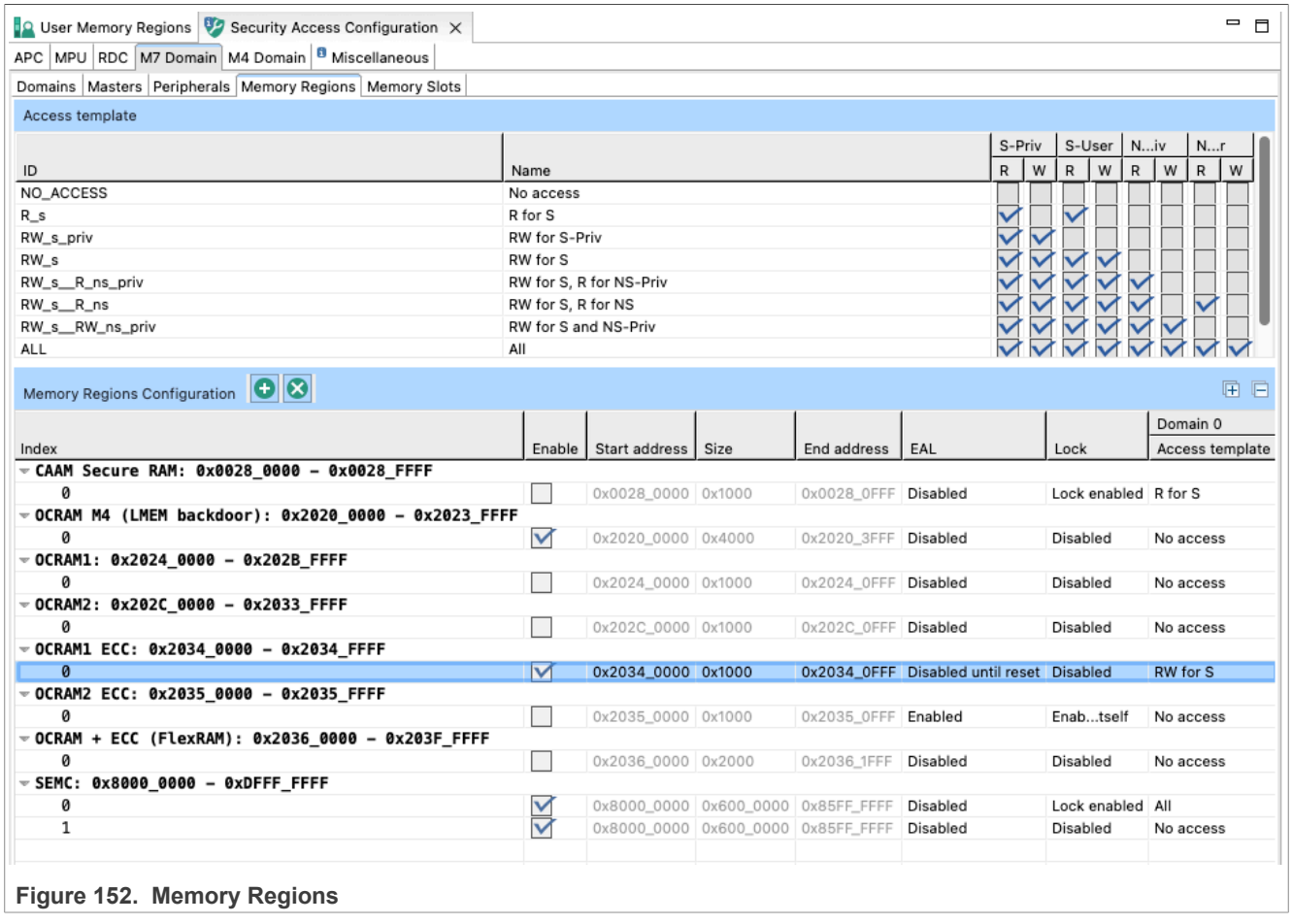


Figure 152. Memory Regions

Use the **Memory Regions Configuration** table to configure access for a non-peripheral memory space:

1. Select the **Enable** checkbox.
2. Specify the **Start Address**.
3. Specify either **Size** or **End Address**.
4. Set the **Lock** to the desired state.
5. Set the **Access Template** for all listed domains.

Alternatively, you can select the options by right-clicking the master and using the dropdown list.

7.2.2.2.6 Memory Slots

In the **Memory Slots** subview, you can view the access templates for MSC (Memory Slot Controller) and configure access for all memory spaces managed by MSC on the selected RDC domain.

The Memory Slot Controller (MSC) performs access control for a peripheral or memory space with a fixed address range.

The **Access Template** table displays the ID and name of all access templates available for the MSC on the selected device. The information is data driven and display-only.

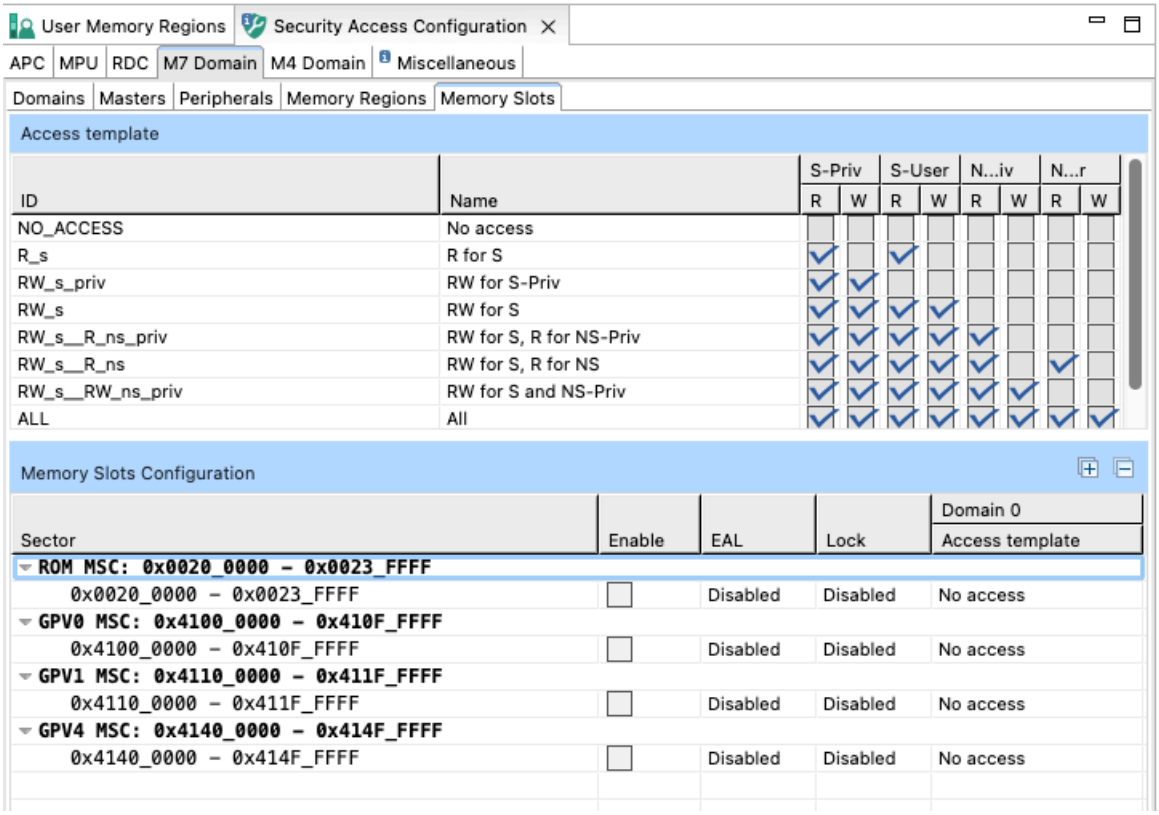


Figure 153. Memory Slots

Use the **Memory Slots Configuration** table to configure access for a memory space:

1. Select the **Enable** checkbox.
2. Set the **Lock** to the desired state.
3. Set the **Access Template** for all listed domains.

Alternatively, you can select the options by right-clicking the master and using the dropdown list.

7.2.2.3 Miscellaneous

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

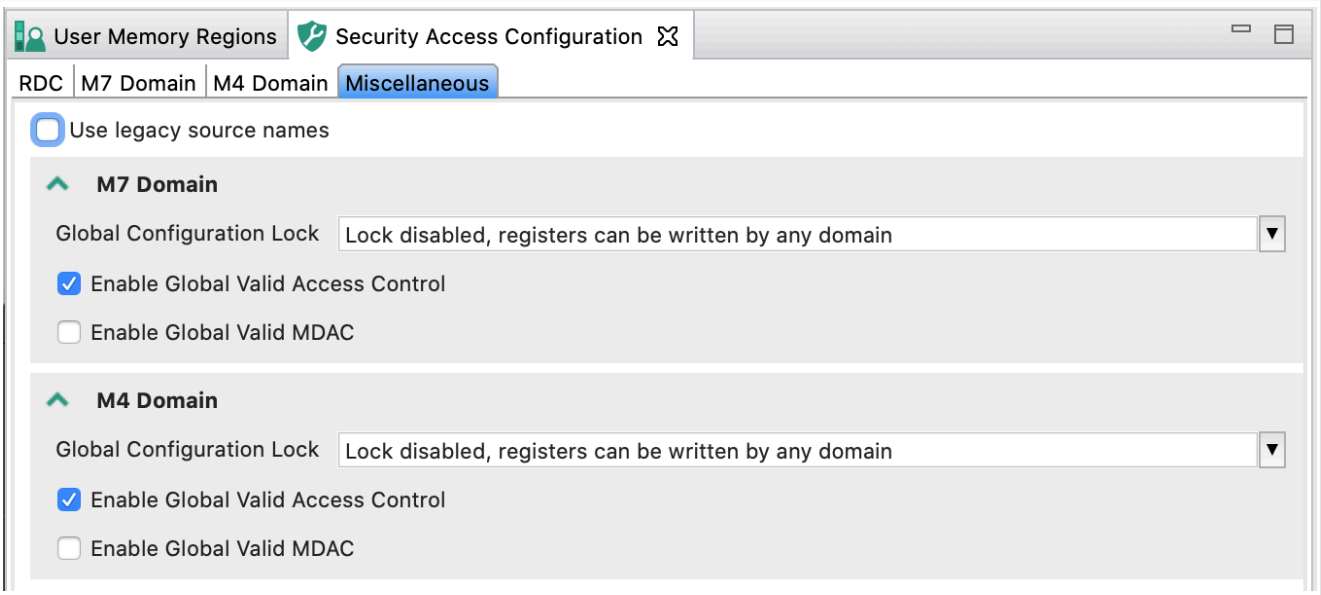


Figure 154. Miscellaneous (RDC+XRDC2)

7.2.3 Memory Attribution map

In the **Memory Attribution Map** view, you can review access levels set for all masters to the code, data, and peripherals memory regions on a domain level. The table is read-only.

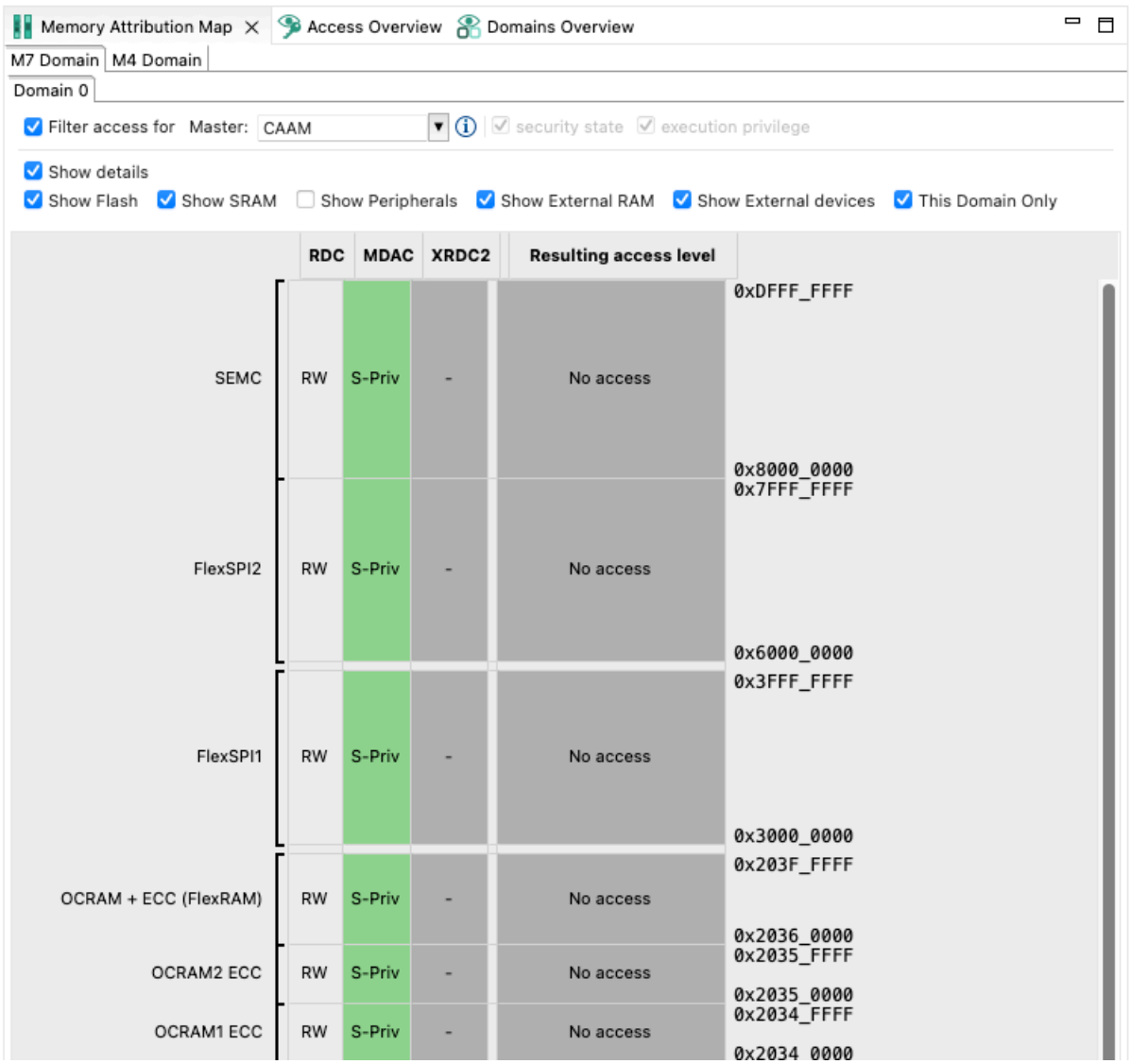


Figure 155. Memory Attribution Map

To set the display options, do the following:

1. Click the **Filter access for** checkbox to enable filtering options.
2. Select the master that you want to review by choosing from the **Master** dropdown menu.
3. Optionally, set the security state and execution privilege check-boxes when master allows more security levels. This setting has no effect on the configuration.
4. Optionally, customize the output by de-selecting the **Show Details**, **Show Flash**, **Show SRAM**, **Show Peripherals**, **Show External RAM, Show External Devices,** and **This Domain Only** options.
5. Optionally, filter displayed memory regions in the **Filter** area.

Point your cursor over the 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**.

7.2.4 Access overview

In **Access Overview**, you can review the security policies you have set in the **Security Access Configuration** view. The view is divided into subviews displaying access overview for specific XRDC2 domains.

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

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

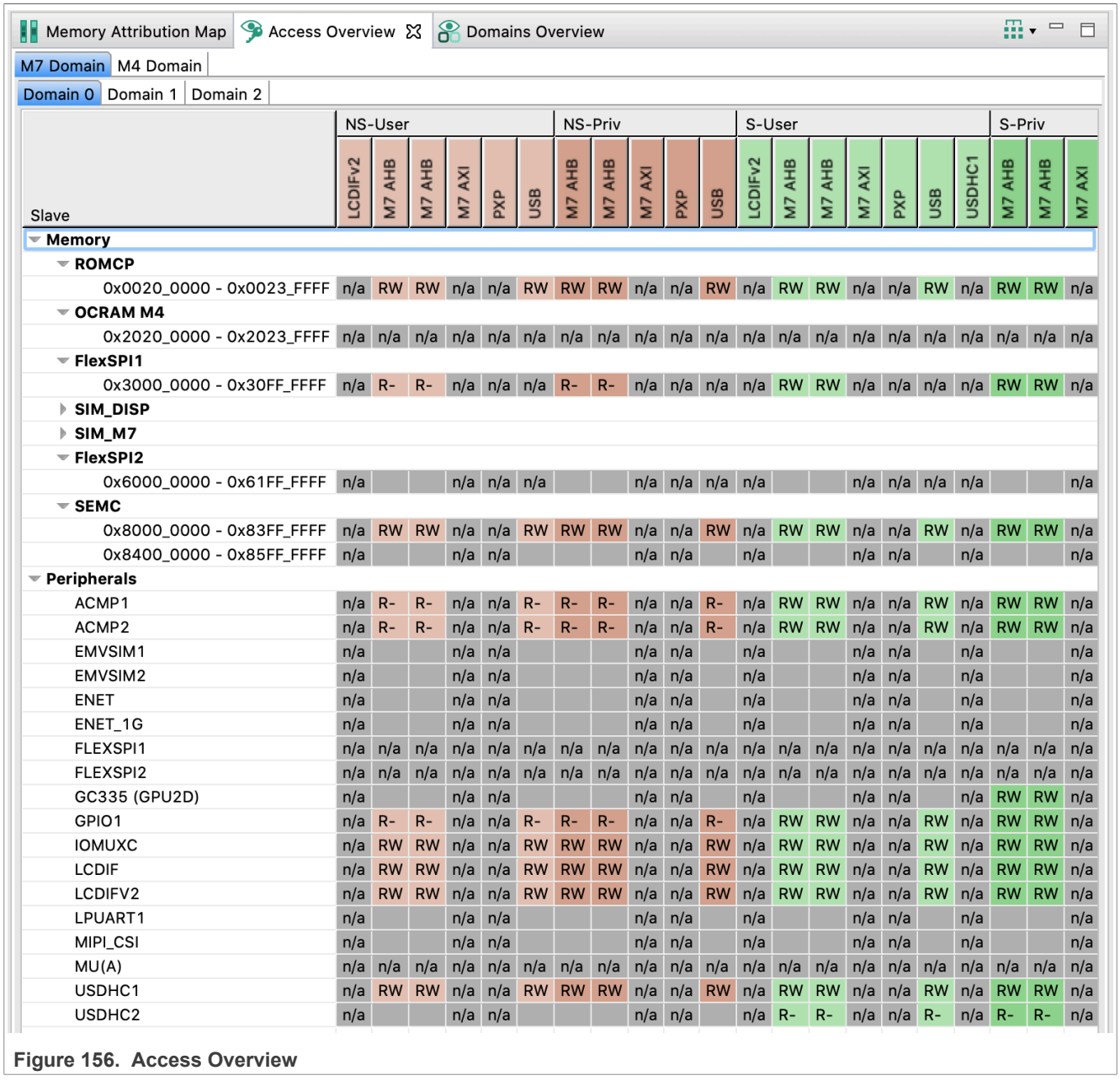


Figure 156. Access Overview

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.

7.2.5 Domains overview

In **Domains Overview**, you can review access policies of XRDC2 domains you have configured in the subviews of the **Domain** view.

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

Memory Attribution Map | Access Overview | Domains Overview

Resource	M7 Domain			M4 Domain	
	Domain 0	Domain 1	Domain 2	Domain 0	Domain 1
Masters					
CAAM		✓			
CM4 AHBC				✓	
CM4 AHBS				✓	
CM4 DMA				✓	
CM7 AHB	✓				
CM7 AXI	✓				
CM7 DMA	✓				
CSI					
ENET					
ENET QOS					
ENET_1G RX					✓
ENET_1G TX					✓
GC355 (GPU2D)			✓		
LCDIF1 (eLCDIF)			✓		
LCDIF2 (LCDIFv2)	✓				
PXP	✓				
USB	✓				
USDHC1	✓				
USDHC2				✓	
Memory					
ITCM (FlexRAM)					
ROMCP					
0x0020_0000 - 0x0023_FFFF	RW	RW			
CAAM Secure RAM					
0x0028_0000 - 0x0028_FFFF					
ITCM M4					
DTCM M4					
DTCM (FlexRAM)					
OCRAM M4					
0x2020_0000 - 0x2021_FFFF	R-			RW	RW
0x2022_0000 - 0x2023_FFFF	R-	R-		R-	
OCRAM1					
0x2024_0000 - 0x202B_FFFF	RW	RW	RW	R-	R-
OCRAM2					
0x202C_0000 - 0x2033_FFFF	RW	RW	RW	R-	R-
OCRAM1 ECC					
OCRAM2 ECC					
OCRAM M7					
FlexSPI1					
0x3000_0000 - 0x30FF_FFFF	RW	RW		R-	R-
0x3100_0000 - 0x3FFF_FFFF					

Figure 157. Domain Overview

7.2.6 Code generation

If the settings are correct and no error is reported, the code generation engine regenerates the source code. You can view the resulting code in 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. Such features as Copy, Search, Zoom-in, Zoom-out, and Export source are available in the **Code Preview** view. The search can also be invoked by CTRL+F or from the context menu.

Some AHBSC and TRDC with security extension-enabled 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** subview.

8 Advanced features

This section covers advanced capabilities of the MCUXpresso Configuration Tools that enable users to streamline their workflow and integrate the tools into automated build processes. These features are designed for users who need to go beyond the standard GUI workflow and require programmatic control, automation, or custom integration of the configuration tools.

8.1 Exporting the Pins table

To export the Pins table, do the following:

1. In the **Menu bar**, select **File > Export**.
2. In the **Export** wizard, select **Export the Pins in CSV (Comma Separated Values) Format**.
3. Click **Next**.
4. Select the folder and specify the filename to which you want to export.
5. The exported file contains the content of the Pins view table, and 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 158. Exported file content

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

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

Figure 159. Aligning to block

8.2 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.`

8.3 Generating HTML reports

You can generate an HTML report file displaying your configuration of Pins, Clocks, and Peripheral tool for future reference.

To generate the HTML report, select **Export > Pins/Clocks/Peripherals Tool > Export HTML Report**.

8.4 Exporting sources

You can export the generated source using the Export wizard.

To launch the Export wizard:

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

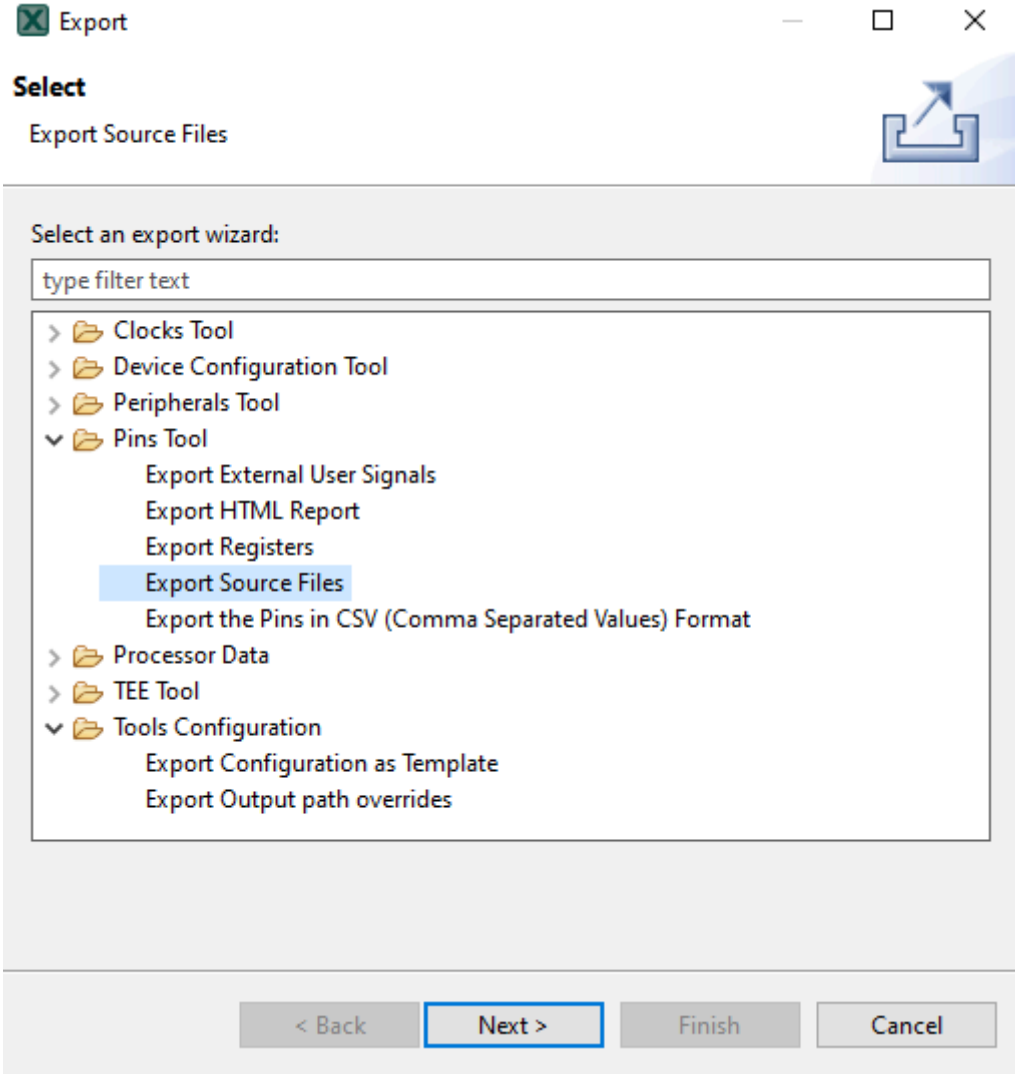


Figure 160. Export wizard

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

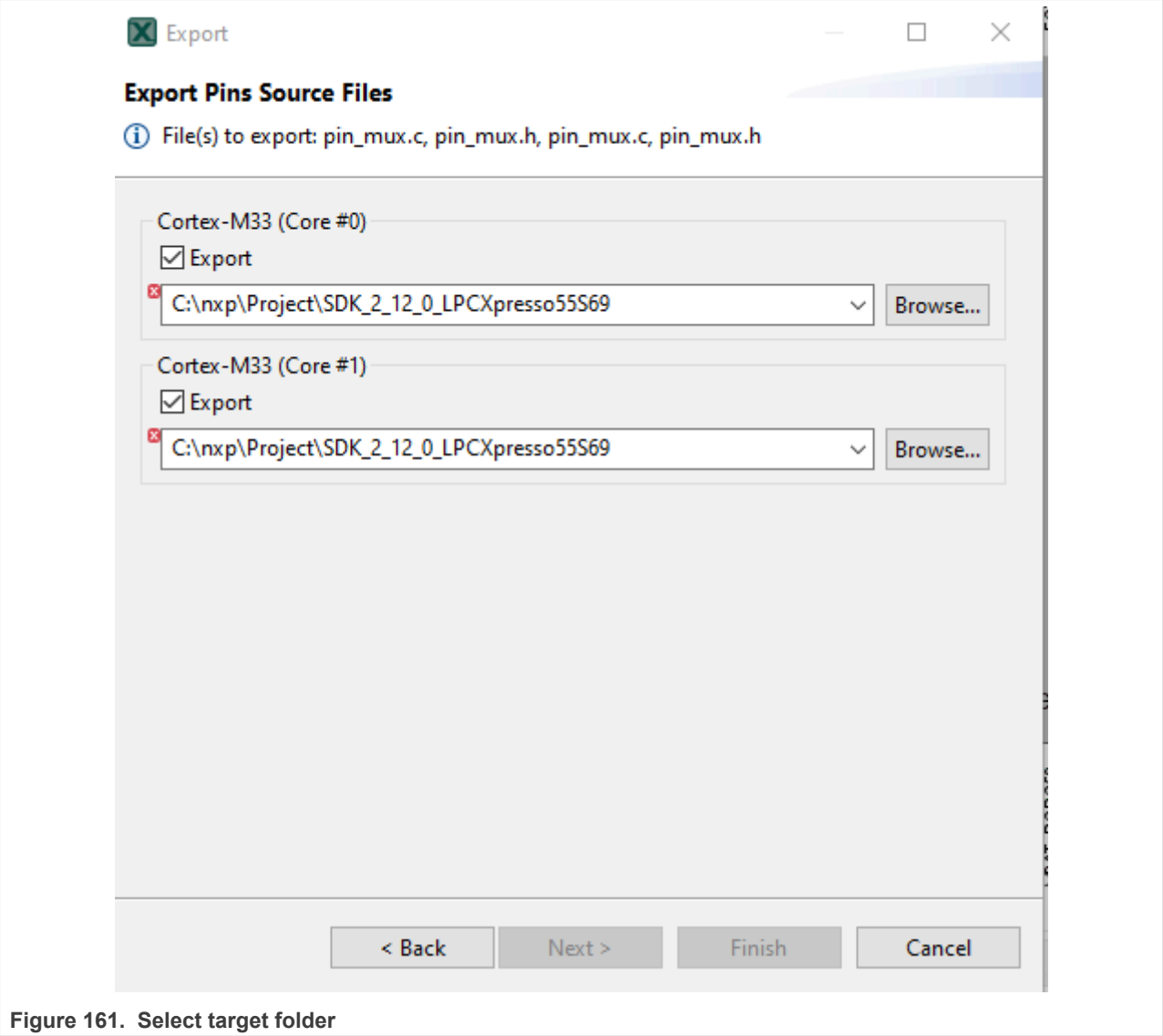


Figure 161. Select target folder

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

8.5 Exporting registers

You can export the content of tool-modified register 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.

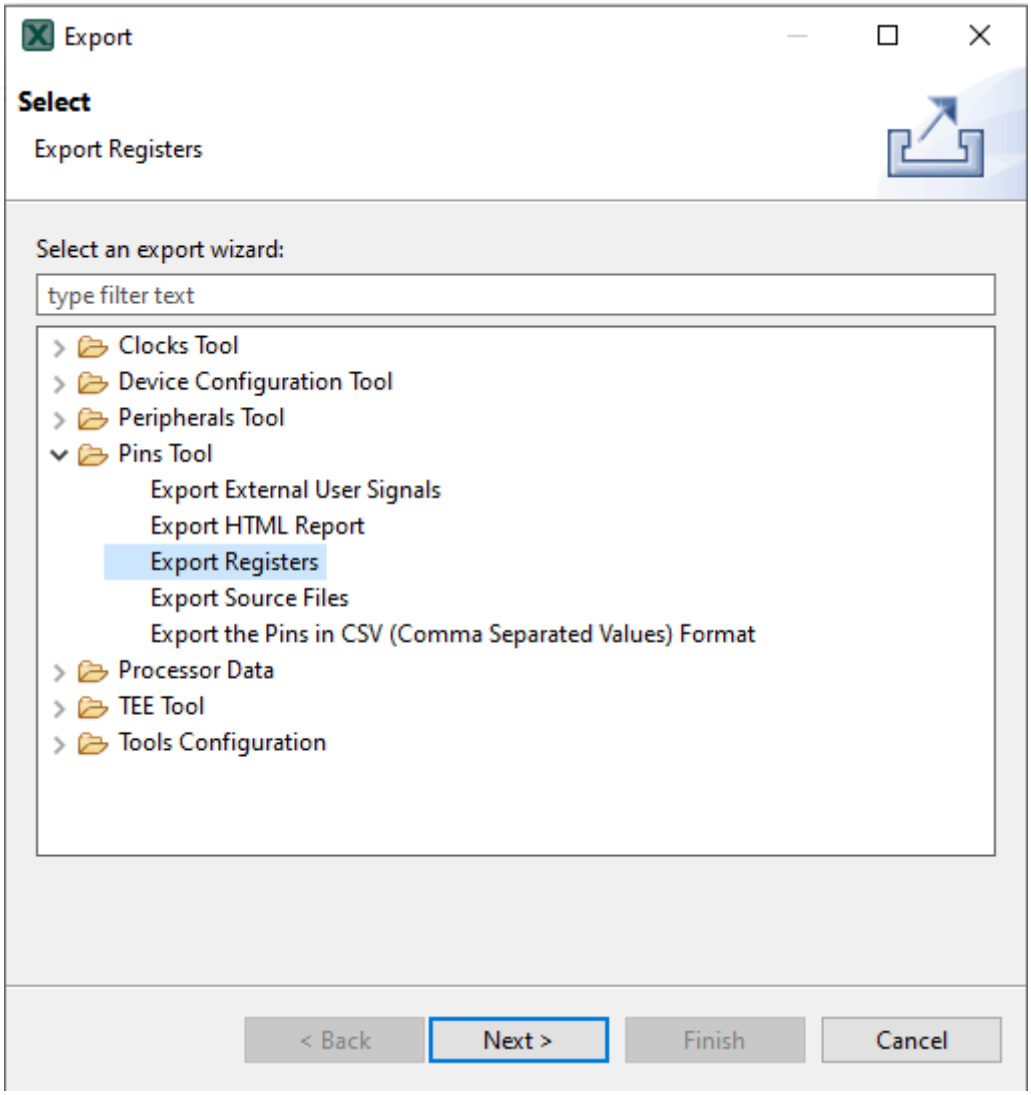


Figure 162. Export registers

3. Click **Next**.
4. Select the target file path where you want to export modified registers content.

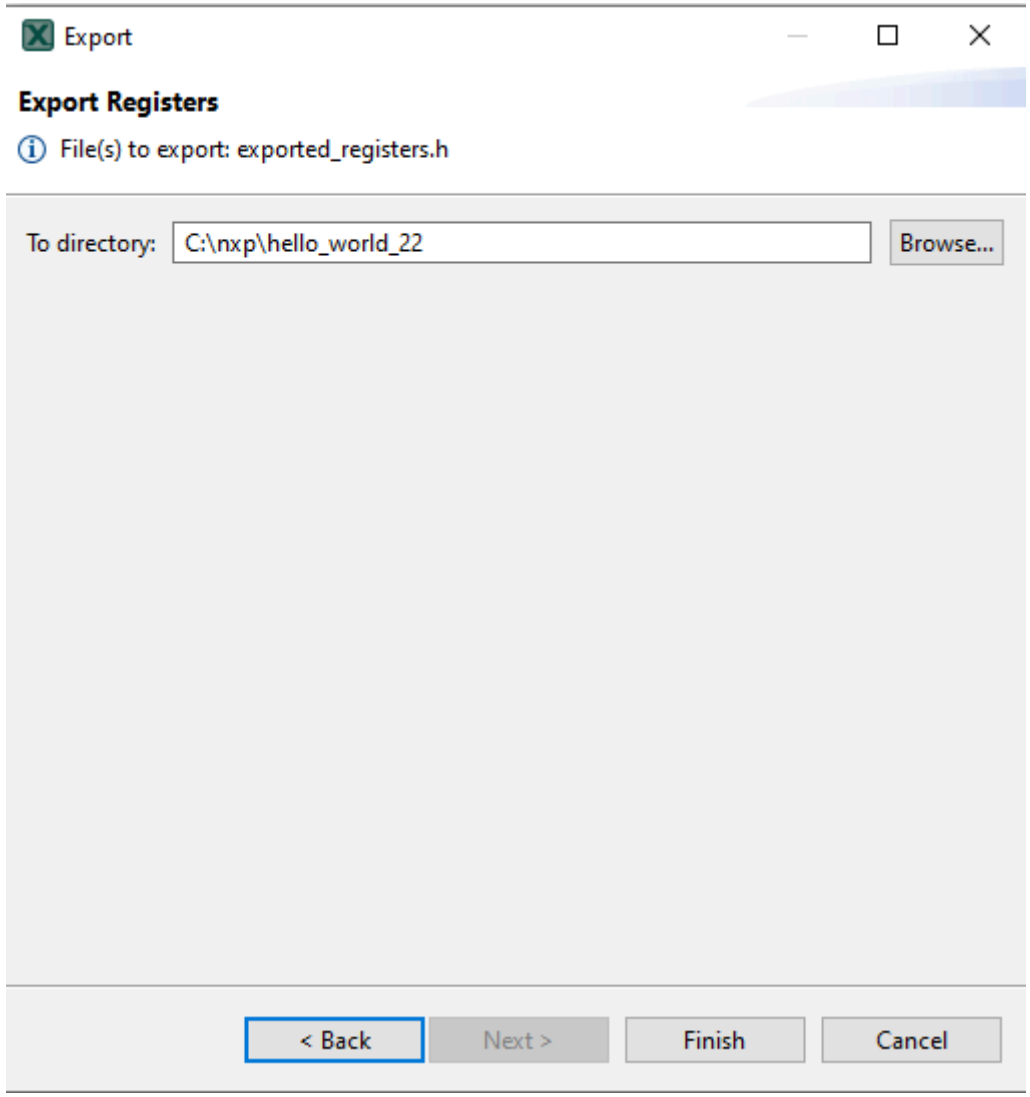


Figure 163. Export registers directory

5. Click **Finish**.

Note: The Export wizard can also be opened by clicking the **Export registers to CSV** button in the **Registers** view.

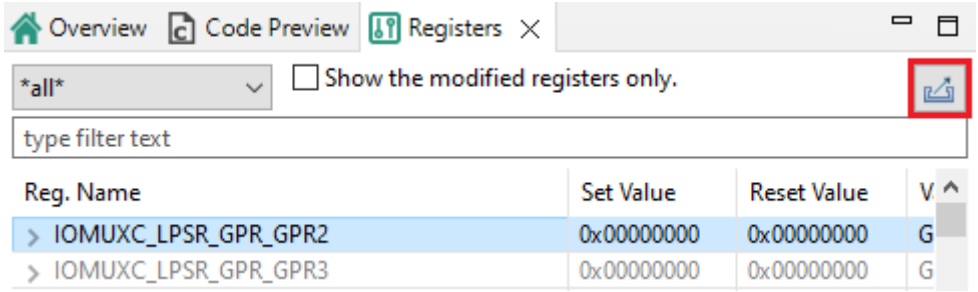


Figure 164. Export registers to CSV

8.6 Command-line execution

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

On error, the application exits:

- Tools v4.1 and older:
  - With '123321' error code. The reason should be logged.
- Tools v5.0 and newer:
  - when a parameter is missing
  - when a tool error occurs

You can chain commands in CLI.

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 the **-HeadlessTool** command and continues to the next **-HeadlessTool** command, or end. (The only exception are commands from the framework that do not need the **-HeadlessTool** command).
- Commands that do not 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 a given order.
- Commands from the framework **are not executed in a given order.**
- The following commands are not executed in a given order:
  - ImportProject
  - Export MEX
  - ExportAll
- The application can exit with the following codes when unexpected behavior occurs:
  - When a parameter is missing: 1
  - When a tool error occurs: 2

Command example:

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

Note: In **MCUXpresso IDE tools** commands can be executed only on the command line with these parameters: `-application com.nxp.swtools.framework.application [tools commands]`

Command example: `mcuxpressoide.exe -noSplash com.nxp.swtools.framework.application -HeadlessTool Clocks -MCU MK64FX512xxx12 -SDKVersion ksdk2_0 -ExportSrc C:/exports/src`

For performance reasons, when CLI is expected to be used multiple times with the same processor, the data is only loaded **if it is not already on disk**. If there is newer data on the server, it is **not updated.**

Long-running jobs share data, so they do not get updated in the middle of execution. To update local data that may have a newer version on the server, use the `-updateData` parameter.

Recommended usage:

- For manual one-time usage, include the `-updateData` parameter on the CLI.
- For multiple executions, for example, continuous integration set-up your job:
  - Use the first simple command with`-updateData`, which updates possibly outdated data.

– Use all other commands in a batch run without this parameter:
```
copy /Y eclipsec.exe toolsc.exe
@rem updates all local data if newer exists
tools.exe -updateData -consoleLog -HeadlessTool Pins
@rem now runs tools many times
tools.exe -consoleLog -HeadlessTool Pins -Load some.mex -ExportAll c:/directory
tools.exe -consoleLog -HeadlessTool Pins -Load other.mex -ExportAll c:/
other_directory
@rem and so on.
```

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

8.6.1 Commands supported in the framework

Table 24. Commands supported in the framework

Command name	Definition and parameters	Description	Restriction	Example
Version of the product	`-version`	Shows the build version of the product into the stdout and continues with parsing other parameters. (since 6.0)	None	`-version`
Force language	`-nl {lang}`	Forces set language {lang} is in ISO-639-1 standard	Removal of the '.nxp' folder from home directory is recommended, as some text might be cached. Only 'zh' and 'en' are supported	`-nl zh`
Show console	`-consoleLog`	Logs output is also sent to Java's System. out (typically back to the command shell if any)	None	`-consoleLog`
Empty configuration	`-EmptyConfig`	Ensures creating a new empty configuration without any content	Requires the `-HeadlessTool` command	`-HeadlessTool Pins -EmptyConfig`
Select MCU	`-MCU`	MCU to be selected by framework. Changes the processor in the result configuration of the previous chain	Requires the `-HeadlessTool` and `-SDKversion` commands; without the Toolchain project.	`-HeadlessTool Pins -MCU MK64FX512xxx12 -SDKversion ksdk2_0` or `-HeadlessTool Pins -Load C:/conf/conf.mex -SDKVersion ksdk2_0 -MCU MK64FN1M0xxx12 -ExportMEX C:/conf/MK64.mex`
Select core	`-Core`	Select core for the configuration	Requires the `-HeadlessTool`, `-`	`-HeadlessTool Pins -Core core0`

Table 24. Commands supported in the framework*...continued*

Command name	Definition and parameters	Description	Restriction	Example
			`MCU`, `-SDKversion` commands.	
Select board	`-Board`	Board to be selected by framework (MCU is automatically selected too) (since 6.0)	Requires the `-SDKversion` command; without the Toolchain project.	`-HeadlessTool Pins -Board FRDM-K22F -SDKversion ksdk2_0`
Select kit	`-Kit`	Kit to be selected by framework (MCU and board is automatically selected too) (since 6.0)	Requires the `-SDKversion` command; without the Toolchain project.	`-HeadlessTool Pins -Kit FRDM-K22F-AGM01 -SDKversion ksdk2_0`
Select SDK version	`-SDKversion`	Version of the MCU to be selected by framework	Requires the `-HeadlessTool` command	`-HeadlessTool Pins -MCU MK64FX512xxx12 -SDKversion ksdk2_0`
Select part number	`-PartNum`	Selects specific package of the MCU. Changes package in result configuration of the previous chain, see the command about `-MCU`	Requires the `-MCU` and `-SDKversion` commands; without the Toolchain project.	`-HeadlessTool Pins -MCU MK64FX512xxx12 -SDKversion ksdk2_0 -PartNum MK64FX512VLL12`
Configuration name	`-ConfigName`	Name of newly created configuration - used in export. Rename the configuration	Requires the `-HeadlessTool` command	`-HeadlessTool Pins -MCU MK64FX512xxx12 -SDKversion ksdk2_0 -ConfigName "MyConfig"` or `-HeadlessTool Pins -ConfigName "MyRenameConfig"`
Select tool	`-HeadlessTool`	Selects a tool that must be run in headless mode	If the tool is disabled in the configuration, it will not be enabled by this option. Use `-Enable` if the tool must be enabled via the cmd line.	`-HeadlessTool Clocks`
Load configuration	`-Load`	Loads the existing configuration from a (*.mex) file	Without the Toolchain project.	`-Load C:/conf/conf.mex`
Export Mex	`-ExportMEX`	Exports the .mex configuration file after tools run. The argument is expected to be a folder or path to specify the .mex file	Requires the `-HeadlessTool` command	`-HeadlessTool Pins -MCU xxx -SDKversion xxx -ExportMEX C:/exports/my_config_folder` or `-HeadlessTool Pins -ExportMEX C:/exports/my_`

Table 24. Commands supported in the framework*...continued*

Command name	Definition and parameters	Description	Restriction	Example
				`config_folder/my.mex`
Export all generated files	`-ExportAll`	Exports all generated files (with source code, and so on). Code is regenerated before export. Includes `-ExportSrc` and in framework `-ExportMEX`. The Argument is expected to be a folder name.	Requires the `-HeadlessTool` command	`-HeadlessTool Pins -ExportAll C:/exports/generated`
Generate source files with custom copyright	`-CustomCopyright`	The file content is inserted as a copyright file header comment into generated source files (.c, .h, .dts, .dtsi), that does not contain copyright	Requires the `-HeadlessTool` command	`-HeadlessTool Pins -CustomCopyright c:\test\copyright.txt`
Override the output path of the generated files	`-OutputPathOverrides`	Path to the file with rules that will be used to override output paths of the generated file. An empty list of rules removes the set rules.	Requires the `-HeadlessTool` command	`-HeadlessTool Pins -OutputPathOverrides c:\test\outputPathOverrideRules.yaml`
Batch processing	`-BatchFile`	Path to the file with commands that will be run on defined paths. For details, see: Batch processing	Requires the `-HeadlessTool` command; intent without other commands.	`-HeadlessTool Pins -BatchFile C:/conf/batchCommands.yaml`
Update locally downloaded data	`-updateData`	Downloads data for already locally downloaded data if they have an update.	None	`-updateData`

8.6.2 Command-line execution - Pins Tool

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

8.6.2.1 Commands supported in Pins Tool

Table 25. Commands supported in Pins Tool

Command name	Definition and parameters	Description	Restriction	Example
Enable tool	`-Enable`	Enables the tool if it is disabled in the current configuration	Requires the `-HeadlessTool Pins` command	`-HeadlessTool Pins -Enable`

Table 25. Commands supported in Pins Tool...*continued*

Command name	Definition and parameters	Description	Restriction	Example
Import C files	`-ImportC`	Imports .c files into configuration. Importing is done after loading mex and before generating outputs	Requires the `-HeadlessTool Pins` command	`-HeadlessTool Pins -ImportC C:/imports/file1.c C:/imports/file2.c`
Export all generated files (to simplify all exports commands to one command)	`-ExportAll`	Exports generated files (with the source code, and so on). The code is regenerated before the export. Includes `-ExportSrc`, `-ExportCSV`, `-ExportHTML` and in framework `-ExportMEX`. The argument is expected to be a folder	Requires the `-HeadlessTool Pins` command	`-HeadlessTool Pins -ExportAll C:/exports/generated`
Export Source files	`-ExportSrc`	Exports generated source files. The code is regenerated before the export. The argument is expected to be a folder.	Requires the `-HeadlessTool Pins` command	`-HeadlessTool Pins -ExportSrc C:/exports/src`
Export CSV file	`-ExportCSV`	Exports the generated .csv file. The code is regenerated before export. The argument is expected to be a folder.	Requires the `-HeadlessTool Pins` command	`-HeadlessTool Pins -ExportCSV C:/exports/csv`
Export HTML report file	`-ExportHTML`	Exports the generated HTML report file. The code is regenerated before the export. The argument is expected to be a folder.	Requires the `-HeadlessTool Pins` command	`-HeadlessTool Pins -ExportHTML C:/exports/html`
Export registers	`-ExportRegisters`	Exports the registers tab into a folder. The code is regenerated before the export. The argument is expected to be a folder.	Requires the `-HeadlessTool Pins` command	`-HeadlessTool Pins -Export Registers C:/exports/regs`

8.6.3 Command-line execution - Clocks Tool

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

8.6.3.1 Commands supported in Clocks Tool

Table 26. Commands supported in Clocks Tool

Command name	Definition and parameters	Description	Restriction	Example
Enable tool	`-Enable`	Enables the tool if it is disabled in the current configuration.	Requires `-HeadlessTool Clocks`	`-HeadlessTool Clocks -Enable`
Import C files	`-ImportC`	Imports .c files into the configuration (disables other tools when importing into an empty configuration).	Requires `-HeadlessTool Clocks`	`-HeadlessTool Clocks -ImportC C:/imports/file1.c C:/imports/file2.c`
Export all generated files (to simplify all exports commands to one command)	`-ExportAll`	Exports generated files (with the source code and all the available export objects). The code is regenerated before the export. Includes `-ExportSrc` and in framework `-ExportMEX`. The argument is expected to be a folder.	Requires the `-HeadlessTool Clocks` command	`-HeadlessTool Clocks -ExportAll C:/exports/generated`
Export Source files	`-ExportSrc`	Exports generated source files. The code is regenerated before the export. The argument is expected to be a folder.	Requires the `-HeadlessTool Clocks` command	`-HeadlessTool Clocks -ExportSrc C:/exports/src`
Export HTML report file	`-ExportHTML`	Exports the generated HTML report file. The code is regenerated before the export. The argument is expected to be a folder.	Requires the `-HeadlessTool Clocks` command	`-HeadlessTool Clocks -ExportHTML C:/exports/html`
Export registers	`-ExportRegisters`	Exports the registers tab into the folder. The code is regenerated before the export. The argument is expected to be a folder.	Requires the `-HeadlessTool Clocks` command	`-HeadlessTool Clocks -ExportRegisters C:/exports/regs`

8.6.4 Command-line execution - Peripherals Tool

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

8.6.4.1 Commands supported in Peripherals Tool

Table 27. Commands supported in Peripherals Tool

Command name	Definition and parameters	Description	Restriction	Example
Enable tool	`-Enable`	Enables the tool if it is disabled in the current configuration.	Requires the `-HeadlessTool Peripherals` command	`-HeadlessTool Peripherals -Enable`
Import C files	`-ImportC`	Imports .c files into the configuration. Importing is done after loading mex and before generating outputs.	Requires the `-HeadlessTool Peripherals` command	`-HeadlessTool Peripherals -ImportC C:/imports/file1.c C:/imports/file2.c`
Export all generated files (to simplify all exports commands to one command)	`-ExportAll`	Exports generated files (with source code and so on). The code is regenerated before the export. Includes `-ExportSrc`, `-ExportHTML`, and in the framework `-ExportMEX`. The argument is expected to be a folder.	Requires the `-HeadlessTool Peripherals` command	`-HeadlessTool Peripherals -ExportAll C:/exports/generated`
Export Source files	`-ExportSrc`	Exports generated source files. The code is regenerated before the export. The argument is expected to be a folder.	Requires the `-HeadlessTool Peripherals` command	`-HeadlessTool Peripherals -ExportSrc C:/exports/src`
Export the HTML report file	`-ExportHTML`	Exports the generated HTML report file. The code is regenerated before the export. The argument is expected to be a folder.	Requires the `-HeadlessTool Peripherals` command	`-HeadlessTool Peripherals -ExportHTML C:/exports/html`

8.6.5 Command-line execution - TEE Tool

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

8.6.5.1 Commands supported in TEE Tool

Table 28. Commands supported in TEE Tool

Command name	Definition and parameters	Description	Restriction	Example
Enable tool	`-Enable`	Enables the tool if it is disabled in the current configuration	Requires the `-HeadlessTool TEE` command	`-HeadlessTool TEE -Enable`
Export all generated files (to simplify all	`-ExportAll`	Exports generated files (with source code and so on). The	Requires the `-HeadlessTool TEE` command	`-HeadlessTool TEE -ExportAll C:/exports/generated`

Table 28. Commands supported in TEE Tool*...continued*

Command name	Definition and parameters	Description	Restriction	Example
exports commands to one command)		code is regenerated before the export. Includes `-ExportSrc`, `-ExportHTML`, and in framework `-Export MEX`. The argument is expected to be a folder.		
Export Source files	`-ExportSrc`	Exports generated source files. The code is regenerated before the export. The argument is expected to be a folder.	Requires the `-HeadlessTool TEE` command	`-HeadlessTool TEE -ExportSrc C:/exports/src`
Export registers	`-ExportRegisters`	Exports the registers tab into a folder. The code is regenerated before the export. The argument is expected to be a folder.	Requires the `-HeadlessTool TEE` command	`-HeadlessTool TEE -Enable -Export Registers C:/exports/regs`

8.7 Managing data and working offline

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

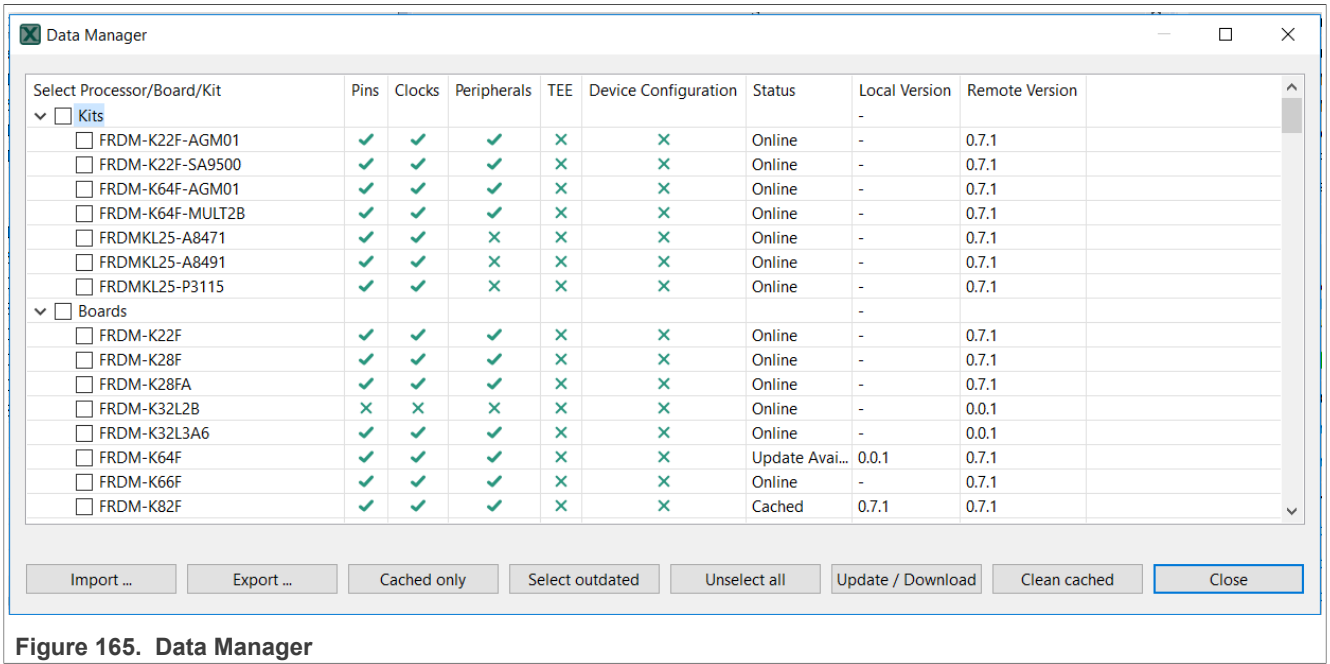


Figure 165. Data Manager

8.7.1 Working offline

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

8.7.2 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. In **Menu bar**, select **Config Tools > Data Manager**.
2. In **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**.

8.7.3 Downloading data from MCUXpresso SDK Builder

You can download required processor data from **MCUXpresso SDK Builder** website.

Note: An NXP account is required for data download from the **MCUXpresso SDK Builder** website.

To download processor, board, or kit data, do the following:

1. Go to MCUXpresso SDK Builder home page.
2. Open **Select Development Board**.
3. In the left-side menu, select **Offline data**.
4. Use the **Search for HW platform** filter field or use the tree view to find your processor, board, or kit.
5. Use the **Download Config Tools Data** button under **Actions** to download data for a specific version of the Config Tools.

You can also download processor, board, or kit data for an existing SDK Build in your **SDK Dashboard**:

1. Go to MCUXpresso SDK Builder home page.
2. Open **Access My SDK Dashboard**.
3. Select the SDK Build for which you want to download Config Tools data and click the **Download** button.
4. In the **Downloads** window, select **MCUXpresso Config Tools > Download Config Tools Data**.
5. In the next menu, use the **Download Config Tools Data** button to download data for a specific version of the Config Tools.

To import the downloaded data, see Importing data.

8.7.4 Exporting data

With **Data Manager**, you can export downloaded processor data in a ZIP format.

To export data, do the following:

1. In **Menu bar**, select **Config Tools > Data Manager**.
2. In **Data Manager**, click **Export**.
3. In **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 a 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 **Menu bar**.

8.7.5 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. In **Menu bar**, select **Config Tools > Data Manager**.
2. In **Data Manager**, select **Import**.
3. In **Import Processor Data** dialog, click **Browse**.
4. Specify the location of the ZIP file that you want to import and click **OK**.
5. Choose the data to import by selecting the checkbox in the table.
6. 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** in the **Start development** wizard.
   Note: You can also import data by selecting **File > Import > MCUXpresso Config Tools > Import Processor Data** from the **Menu bar**.

8.7.6 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 **Menu bar**, 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. In **Menu bar**, select **Config Tools** > **Data Manager**.
2. In **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**. It removes all processor, board, kit, and component data, as well as SDK info files from your computer.
   Note: This action does not affect user templates.

8.8 Output path overrides

This section contains rules that override the path, including the name, of the output files generated by the tools. The rules are applied in the Update Code, Export Wizard, and Command-Line Export commands. The rules are stored in the MEX configuration.

Note: An invalid path is logged as a warning and the original non-overridden path is used.

Rules can be edited in the Output Path Override dialog box in the configuration settings. The new rule is added to the end of the list, the removal is performed for the selected element. The rules are applied to the path in a defined order, which can be changed. The rule contains:

- Enabled - defines whether the rule will be used by the applied path or skipped.
- Description - used as a user-friendly description of the rule
- Regular expression - matches the overriding parts in the whole output path. The format is taken from the Java regular expression.

- Replacement expression - used as a replacement of all matches in the path. Substring groups can be referenced by using placeholder $1, $2 and so on.

The output path override rules can be exported using the wizard to a yaml file. The structure of the yaml file is similar to that of the dialog box.

Example content of the output path override yaml file:

```
outputPathOverrides:
-description: Rule group.h
enabled:true
regex:(bo)ar(d)(/.*\.h)
replacement: $2ar$1$3
-description:Rule2
...
```

The second way to set the rules is to replace them by overriding the output path from the yaml file using wizards or the command line. Rules are used only if all rules are valid. An empty list deletes the current rules. An empty list in the output path overrides the yaml file.

```
outputPathOverrides: [
]
```

8.9 Batch processing

Batch processing can be used to simplify and speed up processing of multiple configurations in an automated way using the command-line interface.

Batch processing is initiated by the headless command "`-BatchFile`". When the command is used, the tool operation is controlled by the specified batch file passed as an argument.

Example:

```
-HeadlessTool Pins -BatchFile C:/conf/batchCommands.yaml
```

Note: It is intended to be used without specifying other tools commands except "`-HeadlessTool`".

8.9.1 Content of the batch file

The batch file content is in YAML format and consists of the folders or files list and the list of command steps. All the steps are executed for each individual path in the given order. If the 'folders' entity is used, the 'files' cannot be used and vice versa.

Note: Paths can be defined as absolute or relative to the batch file.

8.9.1.1 Folders list

The entity "folders" contains a list of the folders to be processed.

Example of the folders list:

```
!!batch_process
folders:
- c:/ test/frdm64
- c:/_ddm/tmp/test/lpc69
steps:
- action: ImportC
  tool: Pins
```

```
  args: ${folder}/src/board/pin_mux.c
- action: ImportC
      tool: Clocks
      args: ${folder}/src/board/clock_config.c
- action: ExportAll
      tool: Clocks
      args: ${folder}/export/clocks
```

Note: The steps element description is given below.

8.9.1.2 Files list

The entity “files” specifies the list of the files to be processed.

Example of the list of files:

```
!!batch_process
files:
- c:/_ddm/tmp/test/frdm64/src/board/pin_mux.c
- c:/_ddm/tmp/test/lpc69/src/board/pin_mux.c
steps:
- action: ImportC
  args: ${file}
  tool: Pins
- action: ImportC
  tool: Clocks
  args: ${folder}/clock_config.c
```

8.9.1.3 Steps list

The steps entity contains a list of steps to be processed for every listed path in a similar way as standard headless command-line tool commands.

Each step is defined as a structure:

- Action - the name of a command-line tool command without “-” at the beginning.
- Tool - the name of the tool in the product that runs the action.
- Args - arguments of the actions, a space between the parameters is expected.
  - The following variables can be used and will be substituted with the appropriate value. In case the “files” list is processed: - `${folder}` - replaced by folder (without separator at the end) where the file is located - `${file}` - the full file path - `${fileName}` - for filename without path
  - In case the “folders” list is processed: - `${folder}` - the folder path without separator at the end

See the section Command-line execution for the list of commands and their description.

The configuration is always automatically cleaned after processing each path (as if the `-EmptyConfig` command was used). The batch file without any paths runs once and cannot use any variables.

Note: All paths on loading are converted to the path format with “/” as a separator.

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.

10 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2026 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

11 Revision history

Table 29. Revision history

Document ID	Release date	Description
MCUXIDECTUG v.18.0	17 September 2026	Updated for v.26.09
MCUXIDECTUG v.17.0	24 June 2026	Updated for v.26.06
MCUXIDECTUG v.16.0	25 March 2026	Updated for v.26.03
MCUXIDECTUG v.15.0	16 December 2025	Updated for v.25.12
MCUXIDECTUG v.14.0	18 September 2025	Updated for v.25.09
MCUXIDECTUG v.13.0	20 June 2025	Updated for v.25.06
MCUXIDECTUG v.12.0	17 March 2025	Updated for v.25.03
MCUXIDECTUG v.11.0	15 January 2025	Updated for v.24.12
MCUXIDECTUG v.10.0	24 September 2024	Updated for v.16.1
MCUXIDECTUG v.9.0	1 July 2024	Updated for v.16
MCUXIDECTUG v.8.0	19 April 2024	Updated for v.15.1
MCUXIDECTUG v.7.0	10 January 2024	Updated for v.15
MCUXIDECTUG v.6.0	31 July 2023	Updated for v.14
MCUXIDECTUG v.5.0	2 January 2023	Memory Validation tool and Validation view are added.
MCUXIDECTUG v.4.0	20 September 2022	Core 0 and Simple and Smart masters are updated.
MCUXIDECTUG v.3.0	30 June 2022	Updated for v.12

Table 29. Revision history...*continued*

Document ID	Release date	Description
MCUXIDECTUG v.2.0	22 December 2021	New features are added, screenshots are updated.
MCUXIDECTUG v.1.0	01 July 2021	Minor changes
MCUXIDECTUG v.0	27 April 2020	Initial version

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents