IMXUG

User Guide for Config Tools for i.MX

Rev. 19.0 — 17 September 2026 **User guide**

Document information

Information	Content
Keywords	MCUXpresso Config Tools, i.MX
Abstract	The Config Tools for i.MX is part of MCUXpresso Config Tools, a suite of evaluation and configuration tools that help users from initial evaluation to production software development. Config Tools for i.MX is an easy-to-use way to configure the pins and DDR of the i.MX processor devices. The software, in general, enables you to create, inspect, change, and modify any aspect of the pin configuration and muxing of the device. It also allows you to configure and validate DDR settings. This document describes the basic components of the Config Tools for i.MX and lists the steps to configure and use them to configure both pins and DDR.

1 Introduction

The Config Tools for i.MX is part of MCUXpresso Config Tools, a suite of evaluation and configuration tools that help users from initial evaluation to production software development. Config Tools for i.MX is an easy-to-use way to configure the pins and DDR of the i.MX processor devices. The software enables you to create, inspect, and modify any aspect of the pin configuration and muxing of the device. It also allows you to configure and validate DDR settings. This document describes the basic components of the Config Tools for i.MX and lists the steps to configure and use them to configure both pins and DDR.

Note: Only the standalone desktop version is currently available for i.MX processors.

1.1 Features

The Config Tools for i.MX consists of the following tools: Pins, Clocks, Peripherals, TEE, DDR, SERDES, and PBL.

The Pins tool is designed for:

- Configuration of pin routing/muxing
- Managing different functions used for routing initialization
- Configuration of pin functional/electrical properties
- Generation of code for routing and functional/electrical properties

The DDR tool is designed for:

- Configuration of DDR controllers
- Validation of DDR configuration

The Pins tool can be used to define routing of pins for target device/board. The tool configuration may be shared using the stored configuration in the MEX file or by using the generated C or DTSI (optional) snippet files (via Import/Export or via copy-paste of the generated source).

Note: The Pins tool generates code for routing the pin to the peripheral, but not for the configuration of the peripheral. Some peripherals might need additional configuration of the pin to assign a function or channel. For example, for some ADC peripherals, the routing provides a connection between the pin and the ADC peripheral. You can then assign the ADC channel from within the ADC peripheral.

The DDR tool allows you to view and configure basic DDR attributes, such as memory type, frequency, number of channels and others and test the DDR configuration by a variety of tests. After you have specified the connection type, you can choose scenarios, tests to run in these scenarios, and view the test results, logs, and summary.

1.2 Versions

For i.MX, the tool is referred to as Config Tools for i.MX and is available as a desktop application only. The tool contacts the NXP server and fetches the list of the available processors. Once used, the processors data is retrieved on demand. 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.

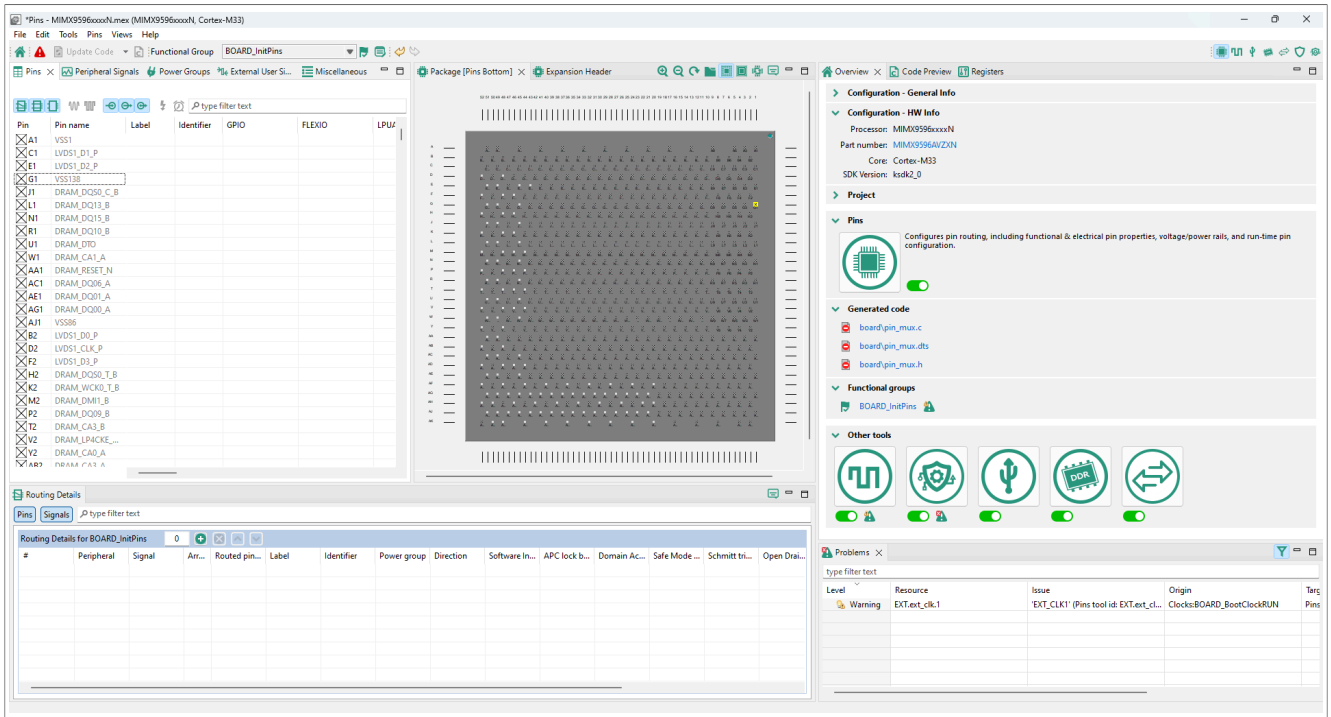


Figure 1. Config Tools for i.MX

1.3 Tools localization

Tools are available in English and Simplified Chinese only.

The locale of Tools automatically copies the global settings of your computer.

To set the locale manually, add the following parameter to the command line:

```
tools.exe -nl zh
```

You can also set the locale in the tools.ini file by adding the following line:

```
-Duser.language=zh
```

Note: Setting your system locale to Chinese automatically launches the tool with localized Chinese menu items, tooltips, and help. Delete the *[home_dir]/.nxp* folder after switching languages because some menu items may be cached.

2 User Interface

2.1 Start development wizard

Upon starting Config Tools, you are automatically welcomed by a startup wizard. With this wizard, you can create a configuration or open an existing one.

Note: To skip the wizard on subsequent startups, select the **Always open last configuration** checkbox below the **Open an existing configuration** option. You can also perform the same action by selecting the **Automatically open previously used configuration** checkbox in **Preferences**.

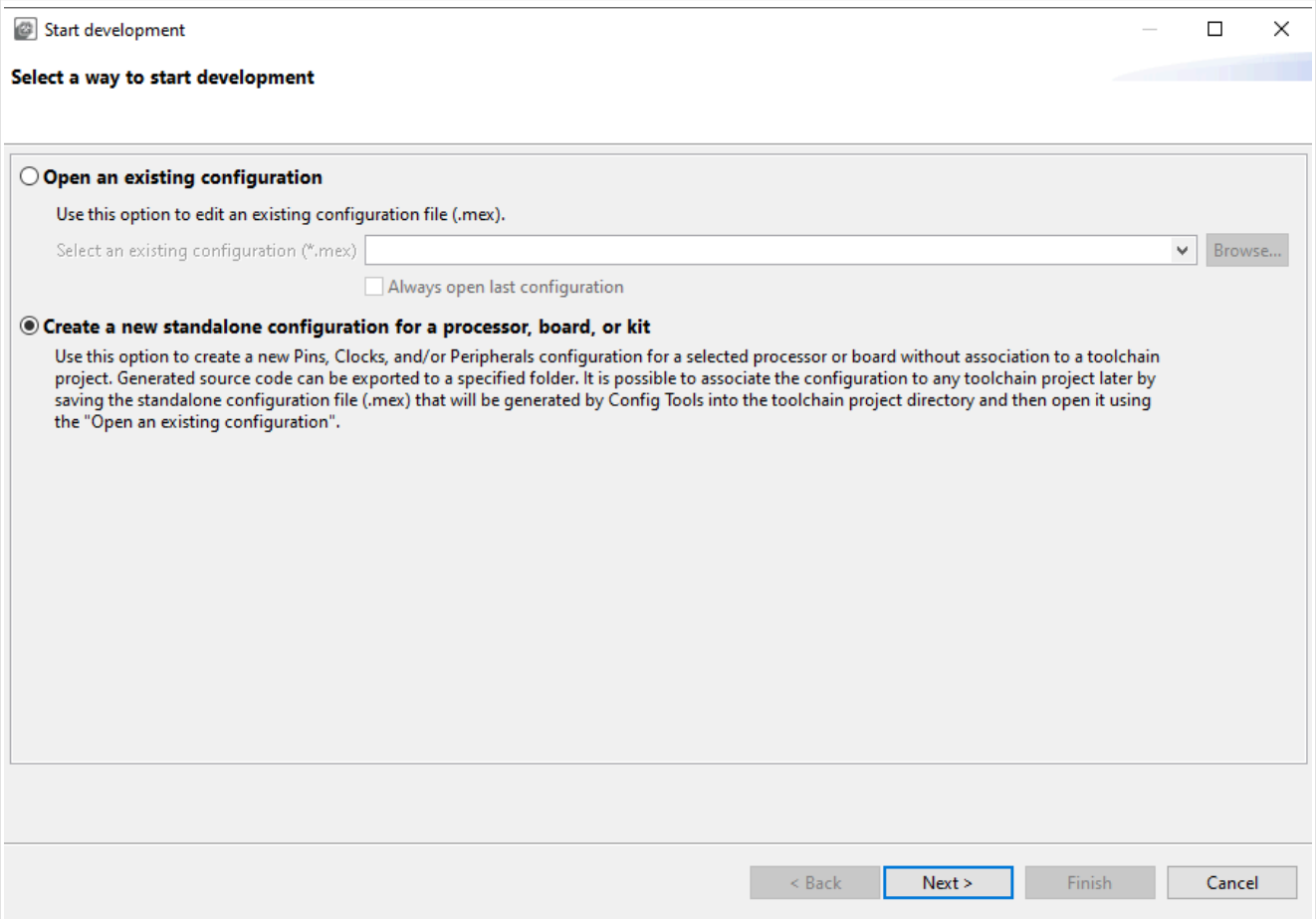


Figure 2. Start development wizard

Note: The content of this wizard is similar to the wizard that you open by selecting **File > New** in the **Menu bar**.

2.2 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.

2.2.1 Creating a new configuration

You can create a configuration from the **Start development** wizard or by selecting **File > New** from the **Menu bar**.

If you start creating your development for any NXP board or kit, we recommend you start with an example to create a configuration for a board or a kit. Such a configuration contains board-specific settings. If you select a processor, the configuration is empty.

After the new configuration is created, you can continue by importing an existing configuration from an MEX file. This is useful if you already have a configuration available or if you want to reuse a previous configuration. To import an existing configuration from an MEX file, select **File > Import… > Import configuration (*.mex)** from the **Menu bar**.

2.2.1.1 Creating a new standalone configuration

You can create a new configuration that is not part of any toolchain project.

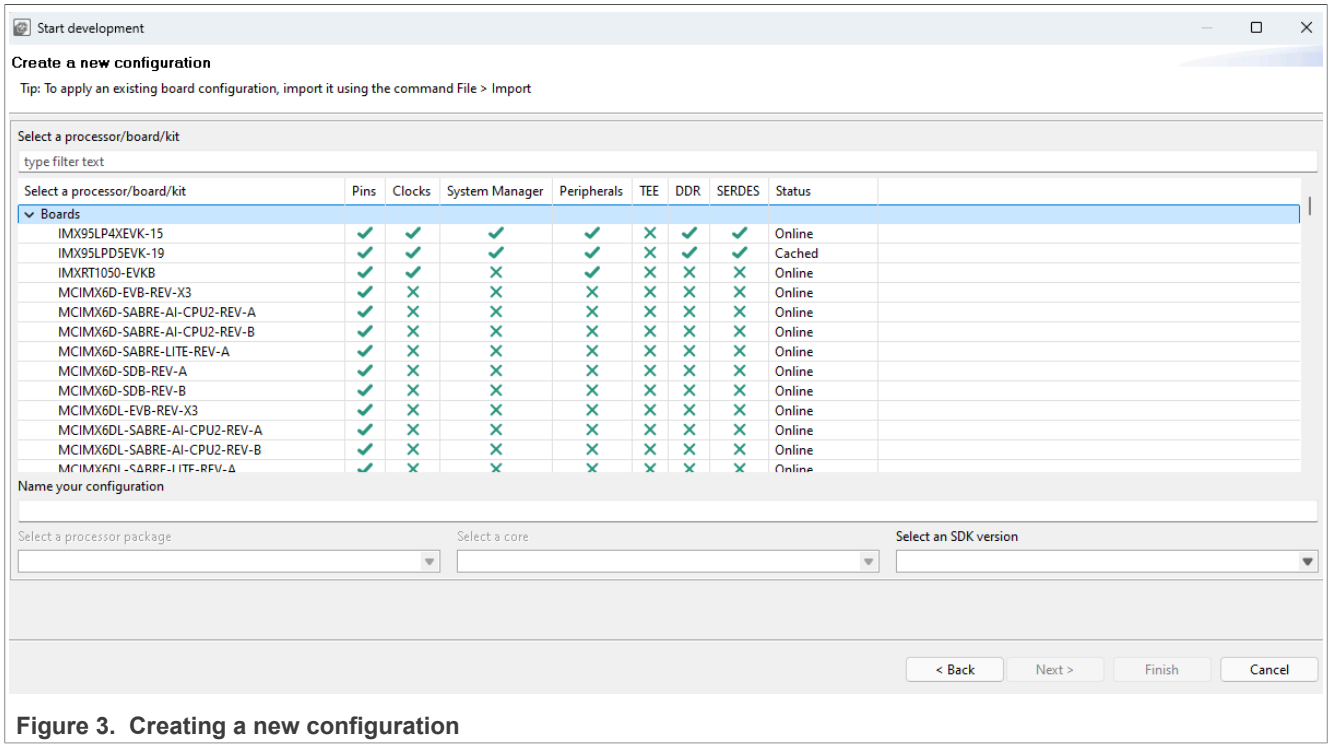


Figure 3. Creating a new configuration

To create a standalone configuration, do the following:

1. In the **Start development** wizard select **Create a new standalone configuration for processor, board, or kit**. Alternatively, in the **Menu bar**, select **File > New**.
2. Click **Next**.
3. Select the processor, board, or kit from the list.
   Note: If you are working offline, you only see locally saved options. For more information, see the Working offline section.
4. Name your configuration. Optionally, you can select processor package, core, and SDK version.
5. Click **Finish**.

2.2.2 Saving a configuration

To save your configuration for future use, select **File > Save** from the **Menu bar**.

To save a back-up of your configuration, do the following:

1. In the **Menu bar**, select **File > Save Copy As**.
2. In the dialog, specify the name and destination of the configuration.
3. Click **Save**.
   The folder with the saved MEX file must contain exactly one project file to parse the toolchain project.

2.2.3 Opening an existing configuration

To open an existing configuration, do the following:

1. In the **Start development** wizard, select **Open an existing configuration**. Alternatively, in the **Menu bar**, select **File > Open**.
2. Click **Browse** to navigate to your configuration file.
3. Select the configuration file and click **Open**.
4. Optionally, select **Always open last configuration** to skip the **Start development** wizard and load the last-saved configuration by default.

2.2.4 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 select a processor whose data is marked as preliminary, Config Tools for i.MX indicates this in two ways:

- The **Start development** wizard shows a note informing you that the selected processor uses a preliminary version of the processor data.
- The **Problems** view displays 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.

2.2.5 User templates

You can export and store the current configuration as a user template for later use as a reference configuration file.

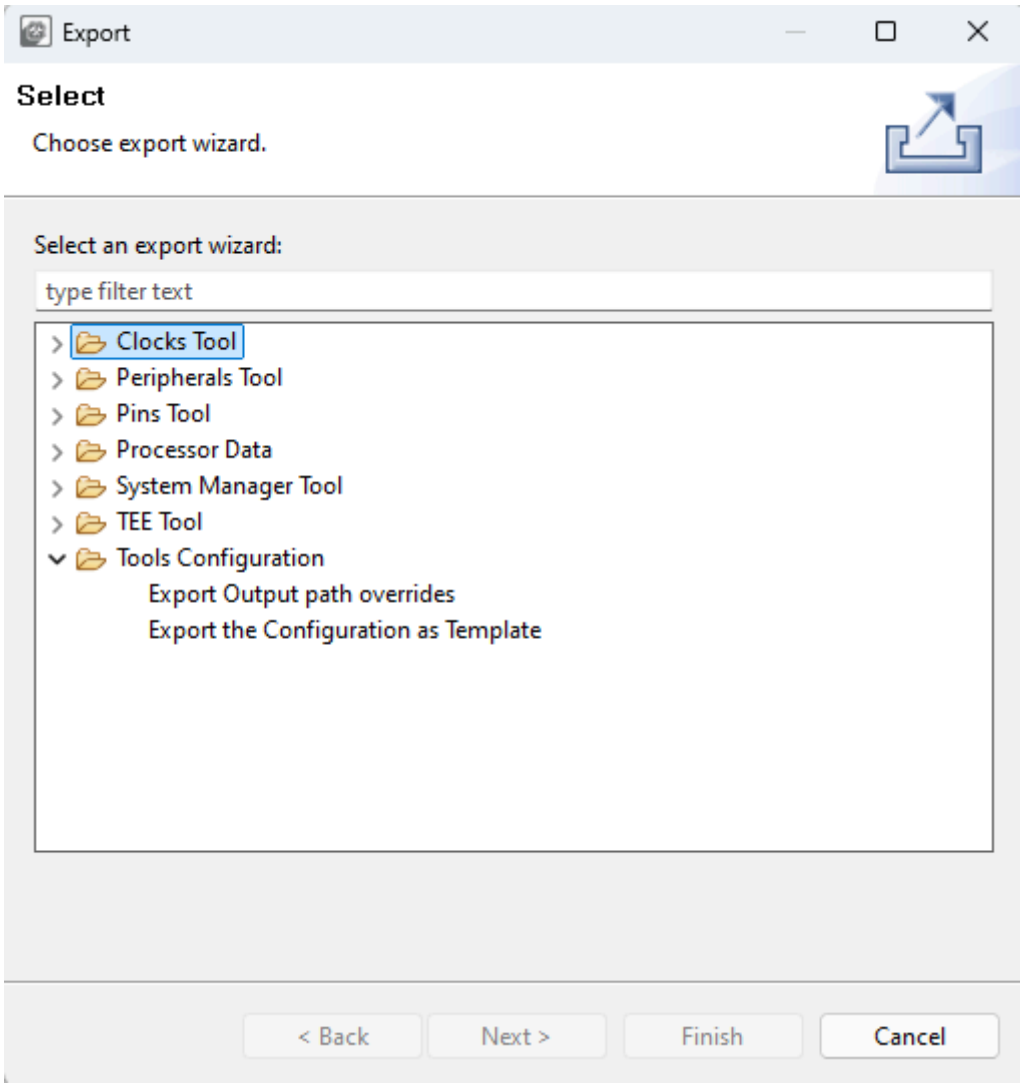


Figure 4. Export template

You can also define custom labels for pins or identifier prefixes for `#define` in generated code.

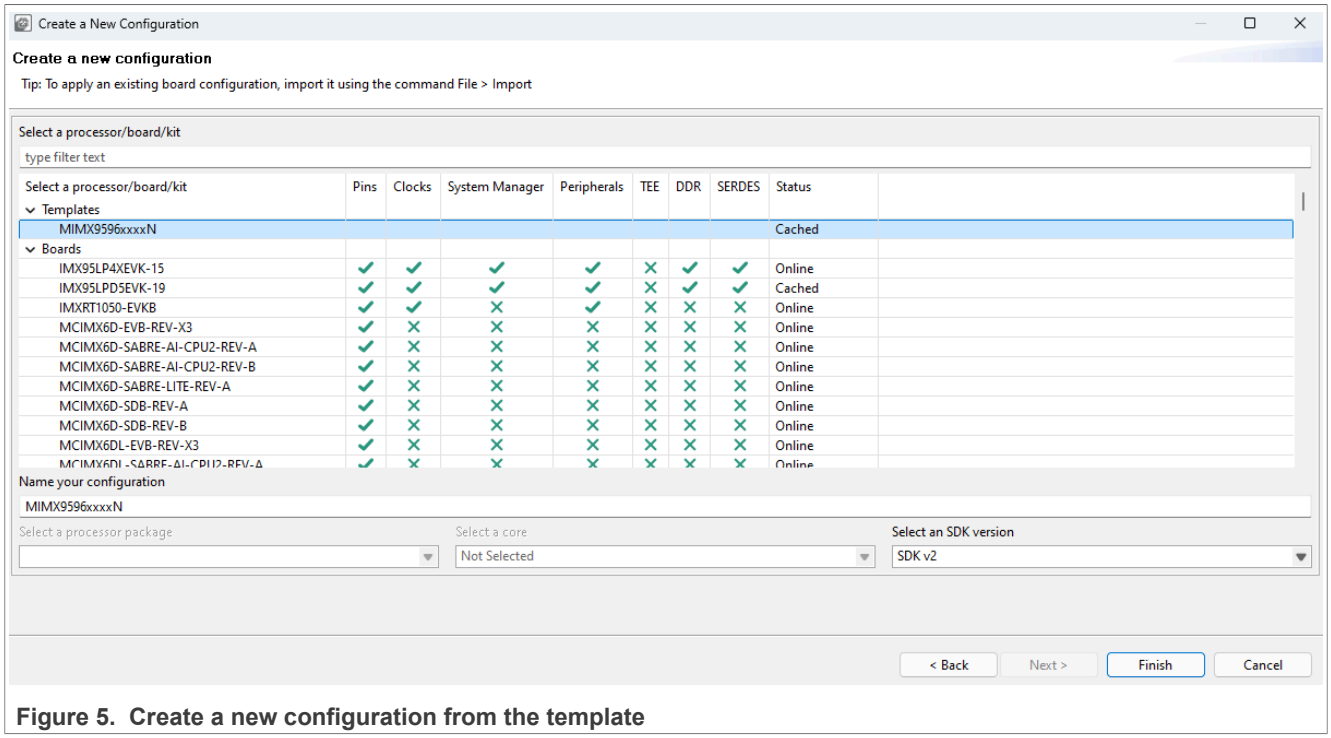


Figure 5. Create a new configuration from the template

Note: The templates are stored at the following location on your local hard disk: `{$user}/.nxp/{tools_folder}/{version}/templates`.

2.2.6 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**.

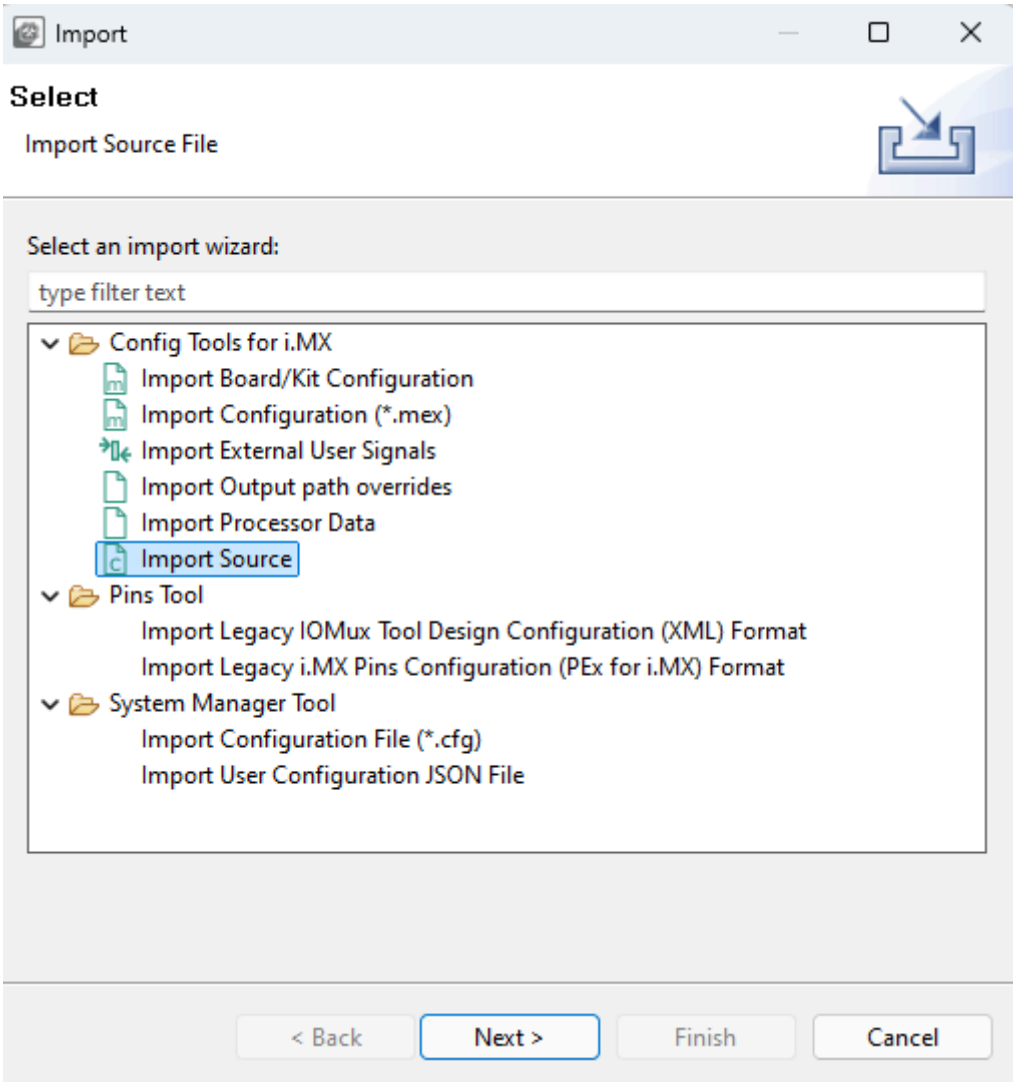


Figure 6. Import Source wizard

3. Click **Next**.
4. On the next page, click **Browse** to specify the location of the source file.
5. Select the source file 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**.

2.2.6.1 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 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**.

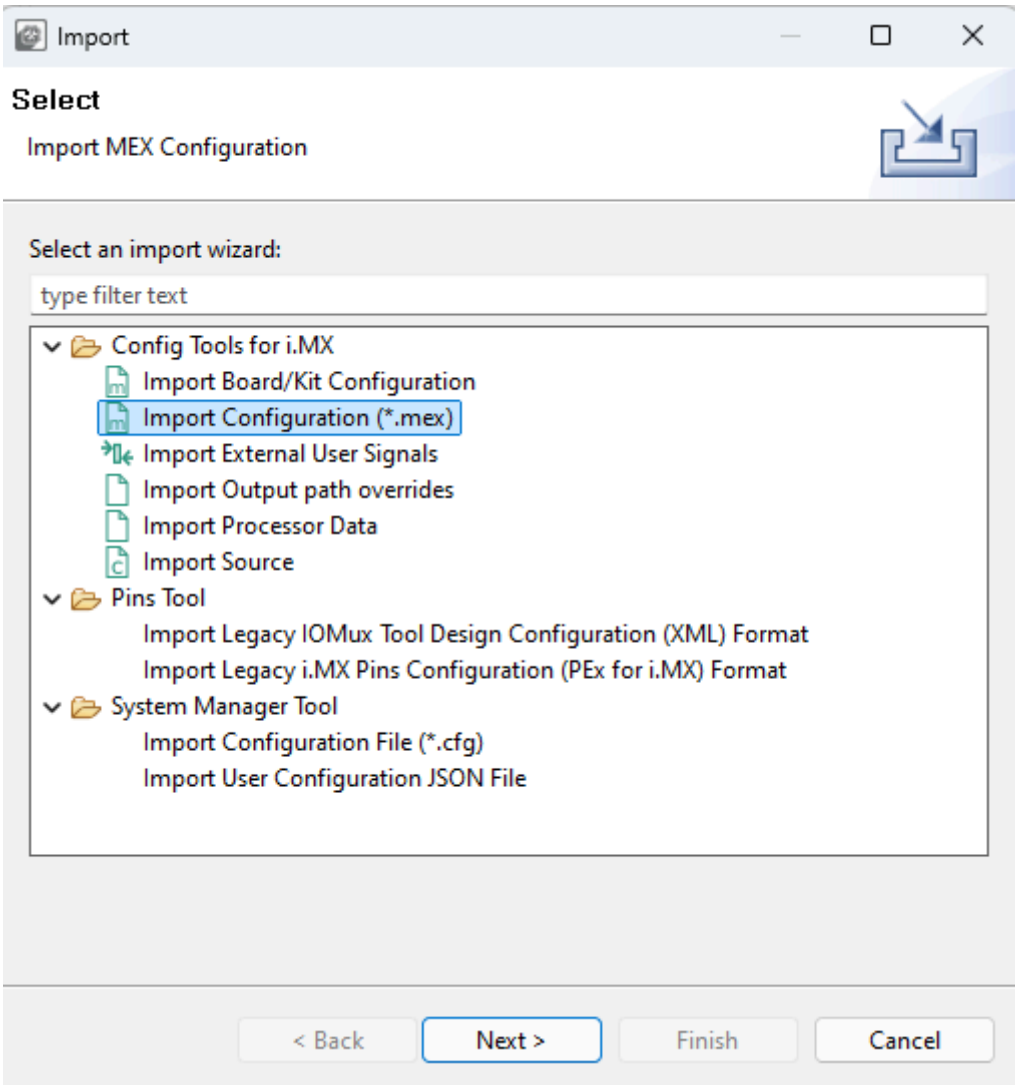


Figure 7. Import Wizard

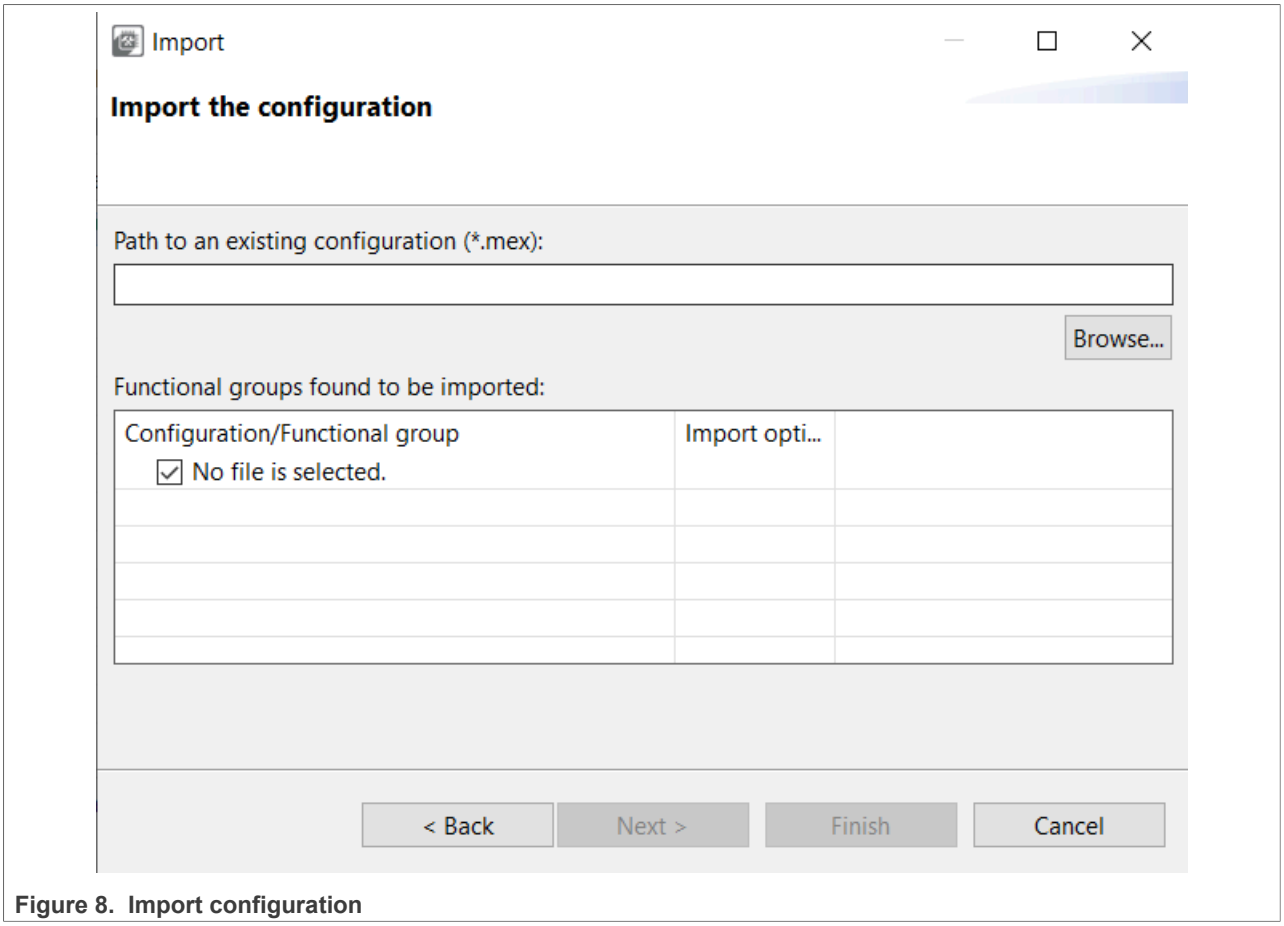


Figure 8. Import configuration

2.2.6.2 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 **Config Tools for i.MX > 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**.

2.2.7 Exporting sources

It is possible to 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**.

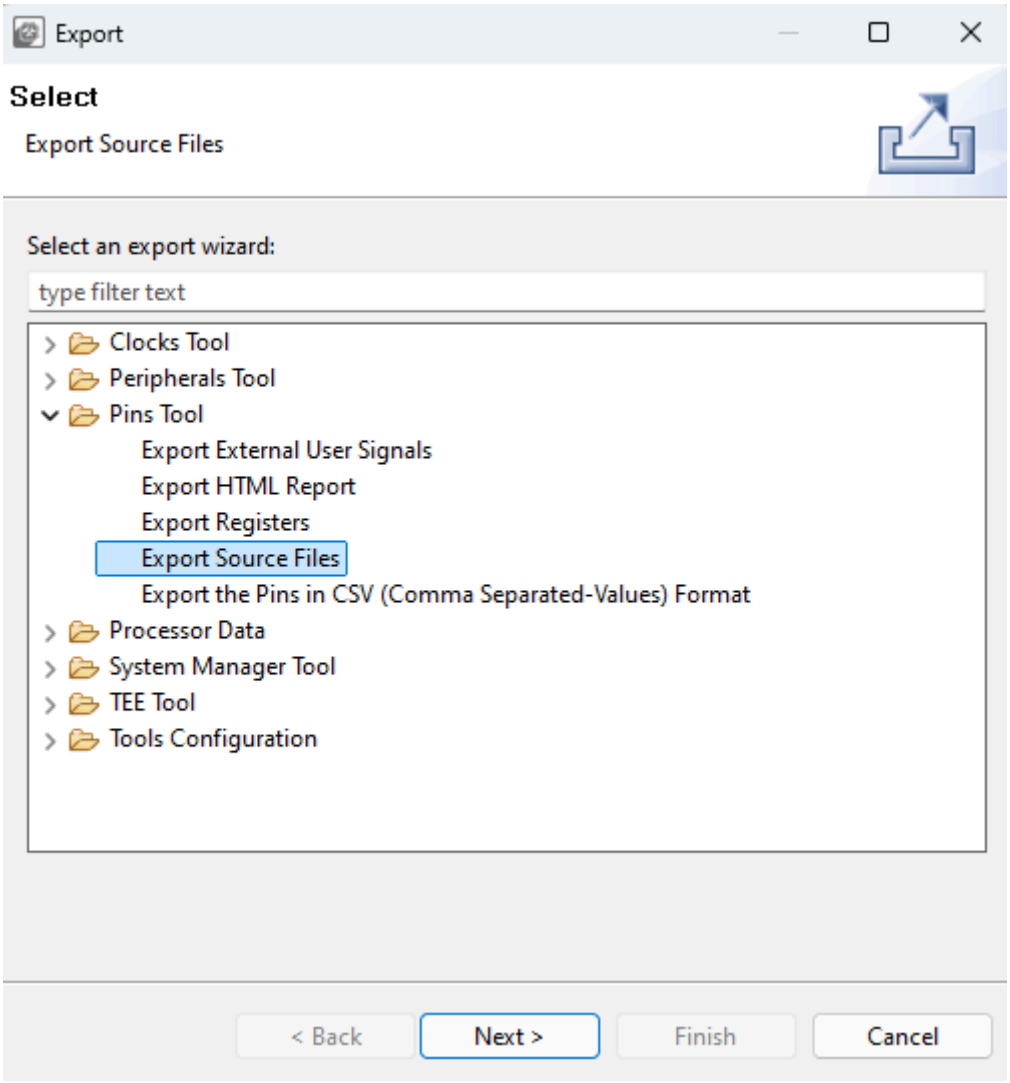


Figure 9. Export wizard

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

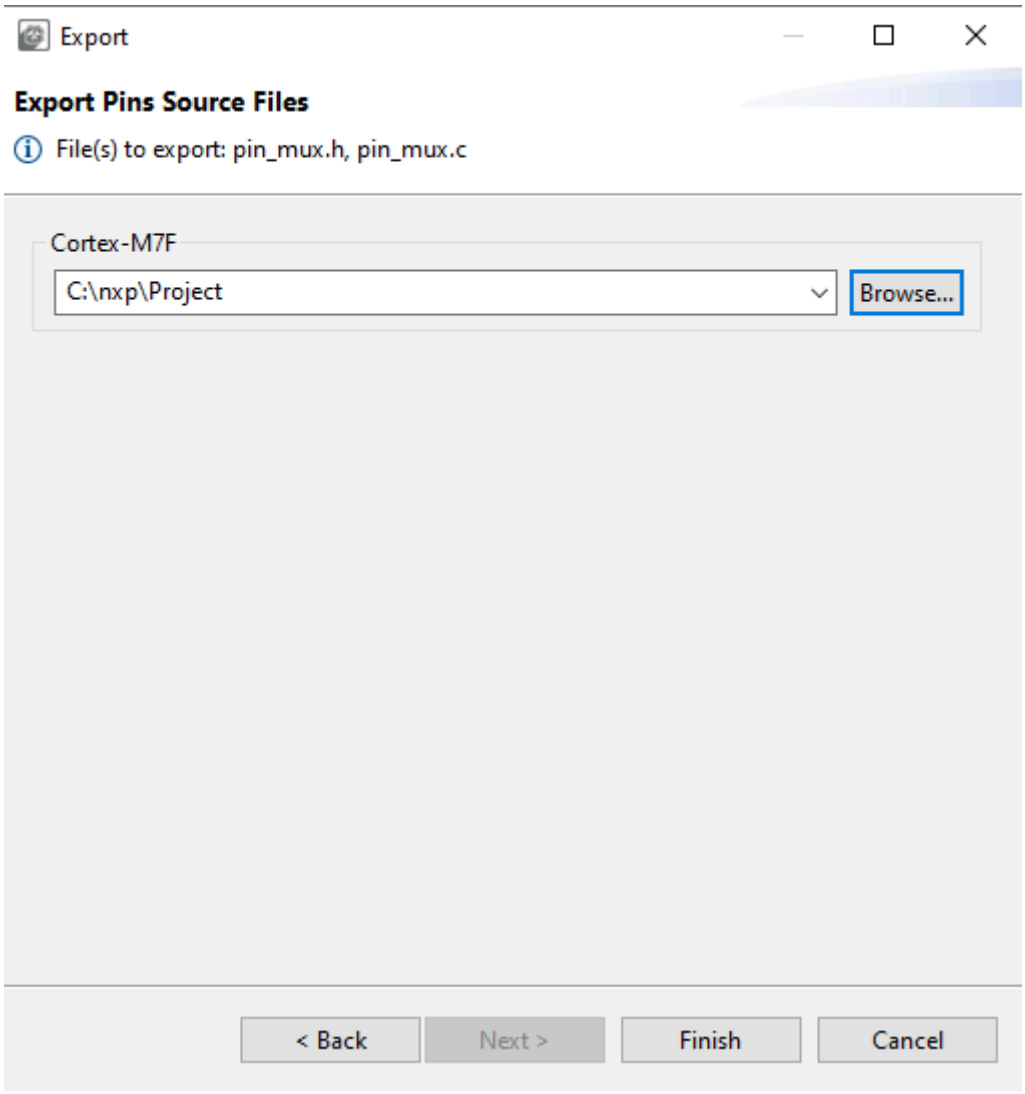


Figure 10. Export sources - Select target folder

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

2.3 Menu bar

The **Menu bar** contains six menus: **File, Edit, Tools, Views, Help, and** a **tool-specific menu**.

The **File** menu contains file management items.

Table 1. Menu bar

Menu item	Description
New…	Create a configuration. For more information, see the Creating, saving, and opening a configuration section.
Open	Open a configuration from an MEX file.
Save	Save the current configuration.

Table 1. Menu bar...*continued*

Menu item	Description
Save Copy As…	Create a backup copy of the current configuration.
Switch processor	Switch to a different processor. For more information, see the Switching processor section.
Switch package	Switch to a different processor package. For more information, see the Switching processor section.
Select Core	Select a processor core for further configuration.
Data Manager	Manage local data. For more information, see the Managing data and working offline section.
Import…	Import settings from source files. For more information, see the Advanced Features section.
Export…	Export source files and other tool information. For more information, see the Advanced Features section.
Exit	Exit the application. If there are any unsaved changes, you are prompted to save the changes.

The **Edit** menu contains basic editing actions as well as items modifying the appearance and behavior of the whole framework.

Table 2. Edit menu

Menu item	Description
Open Update Code Dialog	Update code after configuration change. For more information, see the Update code section.
Undo (…)	Cancel a previous action. The action to be undone is always appended.
Redo (…)	Cancel a previous undo action. The action to be redone is always appended.
Copy	Copy the selected text to the clipboard.
Select All	Select the whole text in the current field/view.
Call from default initialization function	Set the currently selected functional group to be called from the default initialization function.
Functional Group Properties	Edit functional group properties.
Preferences	Edit preferences. For more information, see the Preferences section.
Configuration Preferences	Edit configuration preferences. For more information, see the Configuration Preferences section.

The **Tools** menu lists all the tools available in the tools framework. Use this menu to switch between the tools.

The **Tool-specific** menu contains items tailor-made for individual tools. Only items relevant to the currently active tool are displayed. The menu name copies the name of the currently active tool.

Table 3. Tool-specific menu

Item	Description
Functional Groups	Edit functional group properties.

Table 3. Tool-specific menu...*continued*

Item	Description
Automatic Routing	Resolves routing issues. Opens the **Automatic Routing** dialog, which displays routing issues that have been resolved and the ones that require manual correction.
Apply Expansion Board	Apply an expansion board to an already created expansion header.
Create the Default Routing	Open a dialog for the creation of a new functional group containing the after-reset state of pins and internal signals.
Refresh	Refresh both the generated code and the whole GUI.
Reset to Board Defaults	Reset the configuration of the Board/Kit defaults.
Reset to Processor Defaults	Reset the configuration of the processor defaults.

The **Views** menu contains a tool-specific list of available views. Select a view from the list to open it. Select an already opened view to highlight it. Choose **Reset views** to reset the current tool perspective to its default state. The **Help** menu contains assistance and general information-related items.

Table 4. Views menu

Item	Description
Contents	Display the User Guide.
Quick Start guide	Open a PDF file of the Quick Start guide.
Release Notes	Display release notes of the installed version.
Community	Display web pages of the product-related community forums.
Processor Information	Display web pages containing information about the currently used processor.
Kit/Board Information	Display web pages containing information about the currently used board or kit.
Check for updates	Check for a newer version of the product. If a new version is available, you are prompted to confirm and perform the update.
Open Cheat Sheet	Display a cheat sheet to help with using the tools. You can also load a cheat sheet from a file, or from a URL.
About	Display general product information.

2.4 Toolbar

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

Table 5. 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 to update generated peripheral initialization code directly within the specified toolchain project.

Table 5. Toolbar...*continued*

Item	Description
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 can contain additional items depending on the selected tool. See the chapters dedicated to individual tools for more information.

2.4.1 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.

Table 6. Config tools overview

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

Table 7. Additional information

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.

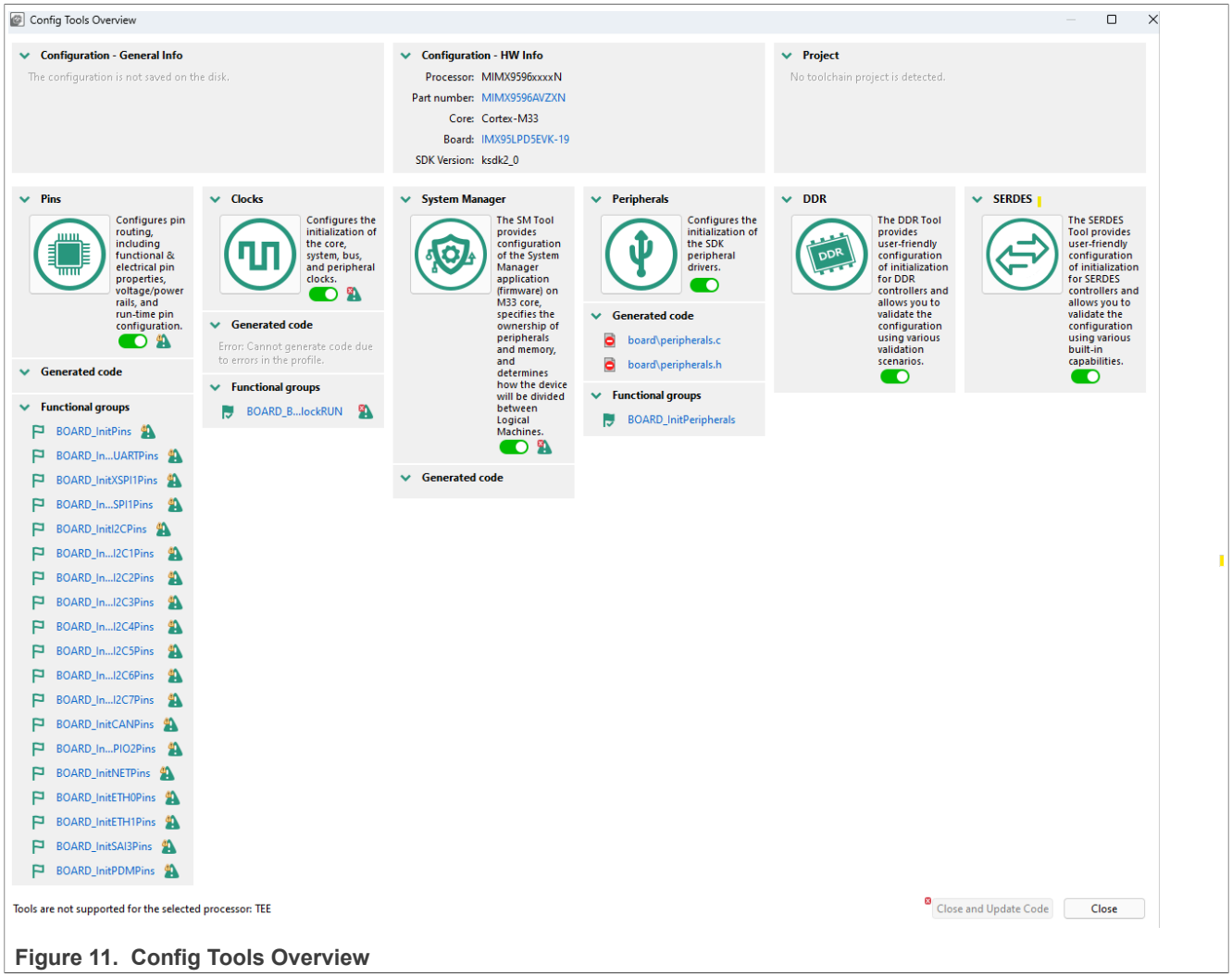


Figure 11. Config Tools Overview

Note: Unsupported tools are not displayed in the overview.

2.4.2 Update code

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**.

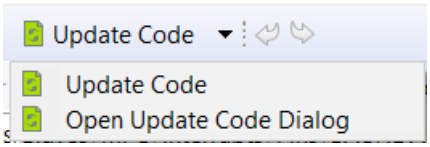


Figure 12. Update Code dropdown menu

Note: The generated code is always overwritten.

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)

2.4.3 Functional groups

Every **Pins** configuration can contain several functional groups.

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

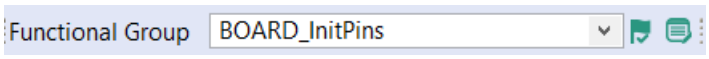


Figure 13. Functional groups

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

Table 8. Configuring functional groups buttons

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.

2.4.3.1 Functional group properties

In the **Functional Group Properties** window, you can configure several options for functions and code generation. Each setting is applicable for the selected function. You can specify a generated function name, select a 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 `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** - The prefix is used for identifiers, constants, and functions related to the functional group 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.

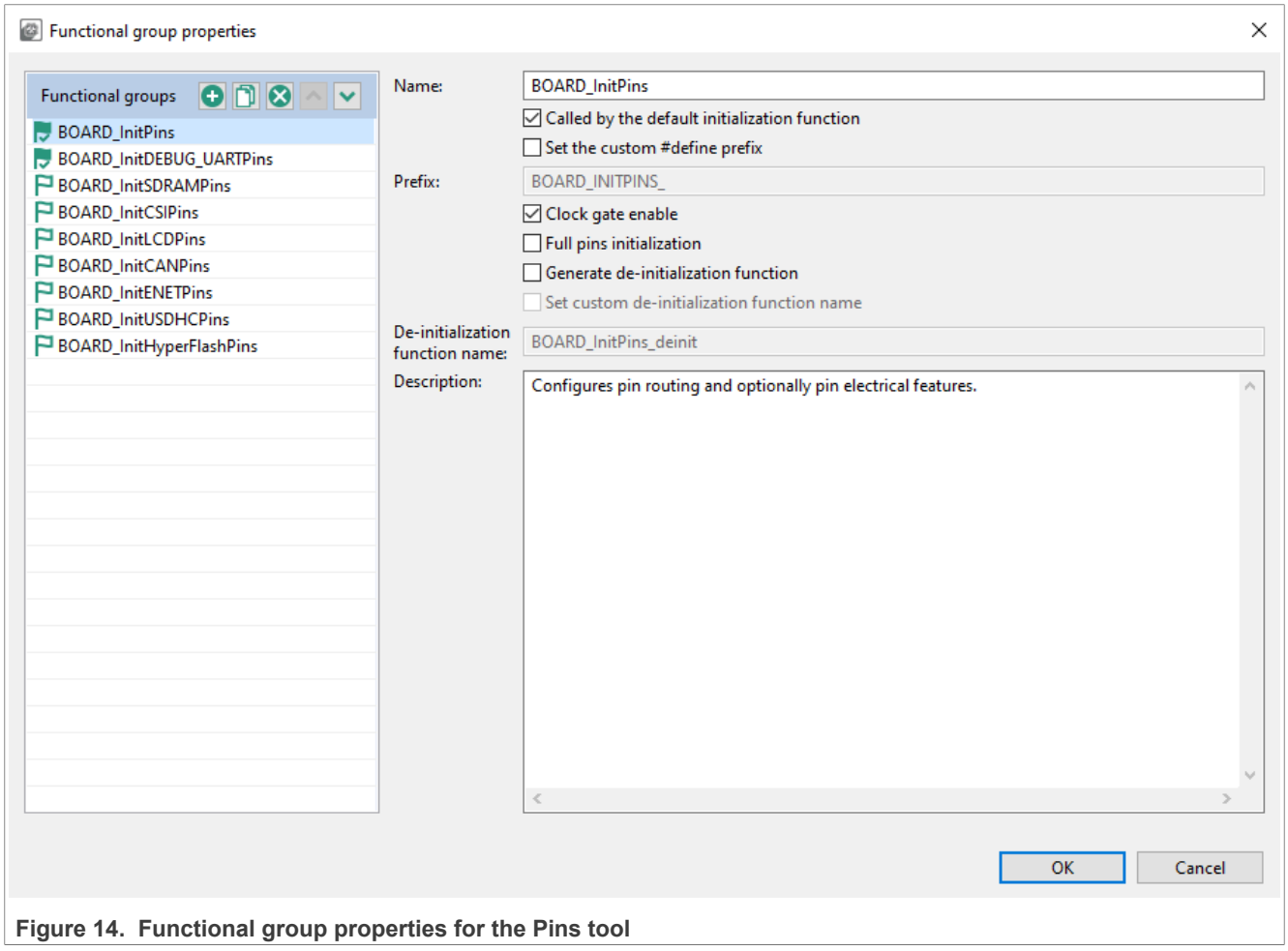


Figure 14. Functional group properties for the Pins tool

2.4.4 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**.

Table 9. Undo/Redo actions

Icon	Description
	Cancels the previous action
	Cancels the previous undo action

2.5 Preferences

To configure preferences in the **Preferences** dialog, select **Edit>Preferences** from the **Menu bar**.

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

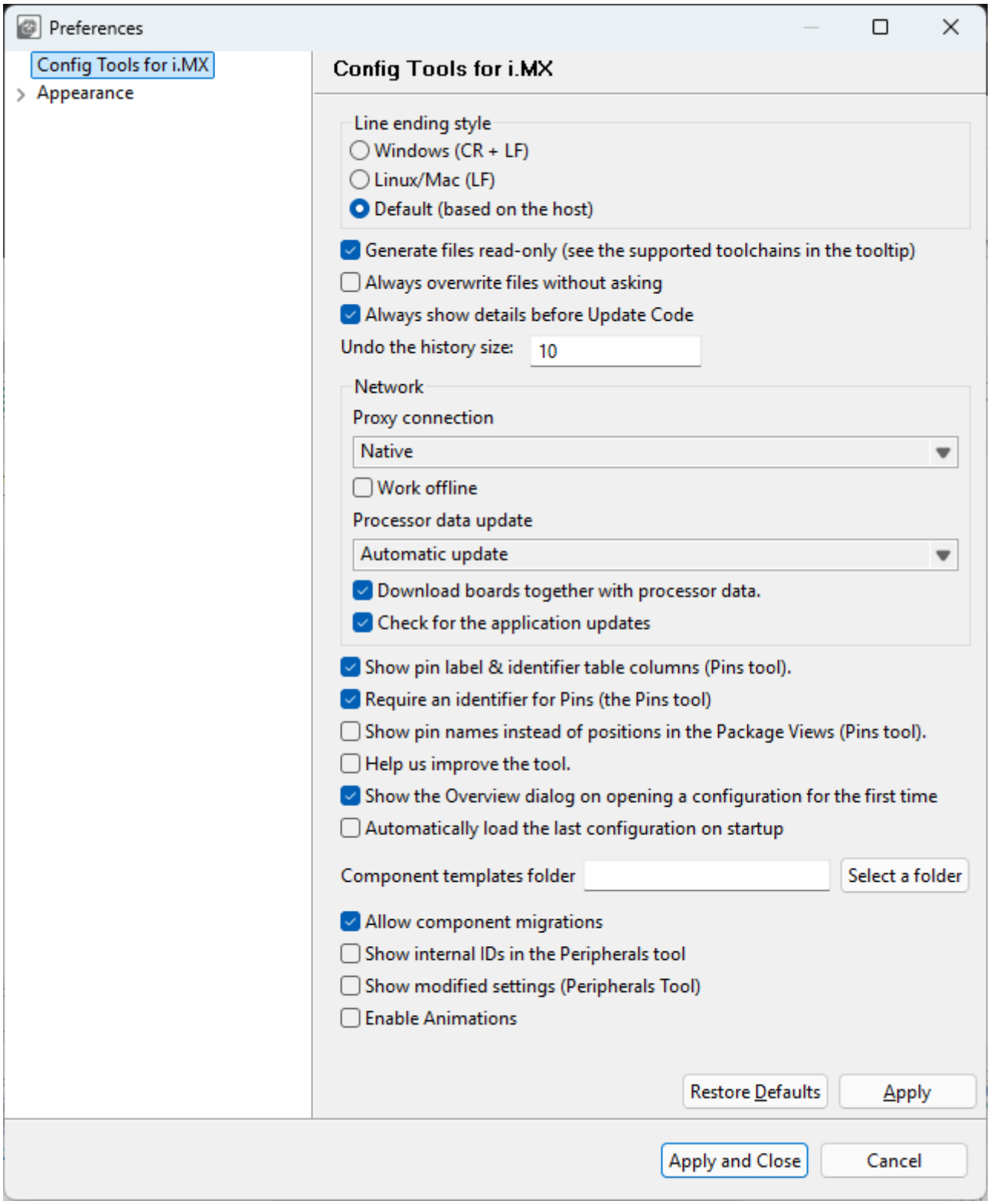


Figure 15. Preferences

Several settings are available.

Table 10. 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.
Proxy connection	- **Direct** - Connect directly and avoid a proxy connection. - **Native** - Use system proxy configuration for network connection. **Note:** The proxy settings are copied from operating system settings. In case of error, you can specify proxy information in the tools.ini file, located in the <install_dir>/bin/ folder. Make sure that the file contains the following lines: - Djava.net.useSystemProxies=true (already present by default) - Dhttp.proxyHost=<somecompany.proxy.net> - Dhttp.proxyPort=80 **Note:** Authentication is not supported.
Work Offline	Disable both the connection to NXP cloud and the download of processor/board/kit data.
Download boards together with processor data	When enabled, automatically downloads the board data for all boards associated with the selected processor whenever processor data is downloaded.
Processor data update	Select from the following options:- **Auto Update** - Update the processor data automatically. - **Manual** - Update processor data after confirmation. - **Disabled** - Disable processor data update.
Check for application updates	Check for application updates on a weekly basis.
Show pin label & identifier table columns (Pins Tool)	Select to show the pin label and the label identifier in the relevant views.
Require an identifier for Pins (Pins Tool)	Controls generation of pins “Identifier” related warnings. With this preference enabled, warnings will be generated for bidirectional signals that have no Identifier set.
Help us improve the tool	Send device-configuration and tool-use information to NXP. Sending this information helps fix issues and improve the tools.
Show Overview window on opening configuration for the first time	Open the Overview dialog on opening configuration for the first time.
Automatically load last configuration on startup	Avoid the startup window and load the last used configuration instead.
Enable animations	Enables animations in the user interface, such as smoother scrolling or opening a drop-down menu.

2.6 Configuration preferences

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

To configure the preferences related to the configuration, select **Edit > Configuration Preferences** from the main menu.

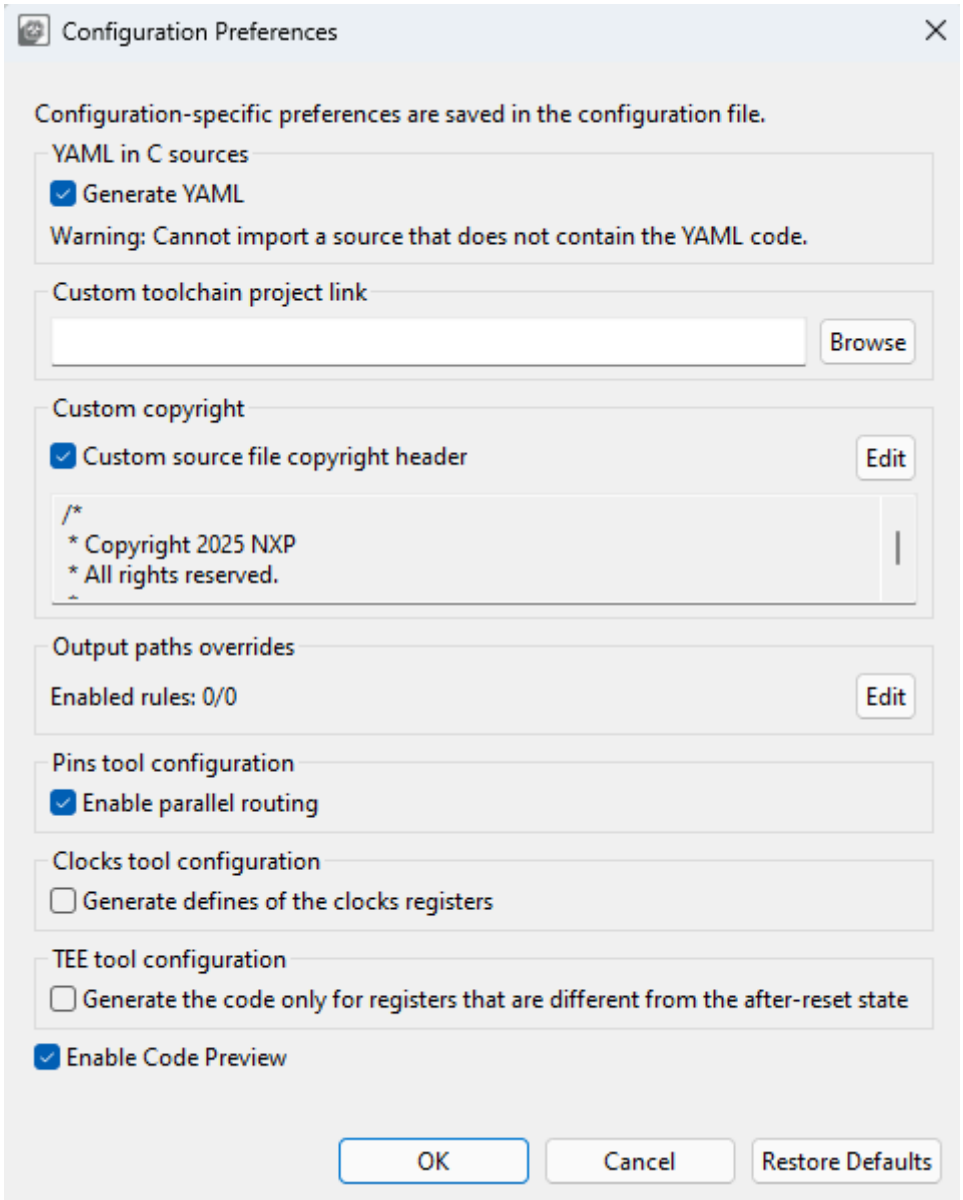


Figure 16. Configuration preferences

The following preferences are available:

Table 11. Configuration preferences

Item	Description
Generate YAML	Select to generate YAML into C source files.
Custom toolchain project link	Select to set the path to the toolchain project folder, otherwise the default path in the same folder as the configuration is used. An absolute path or a relative path to a saved configuration (MEX) can be used. Only for standalone Config Tools.

Table 11. Configuration preferences...*continued*

Item	Description
Custom source file copyright header	Select to add a custom copyright header to generated source files that do not already contain copyright.
Output path overrides	Rules that override the output file paths generated by the tools. They are applied in the **Update code** and **Exports** commands. A special dialog allows you to edit them.
Generate code only for registers that are different from the after-reset state	Select to generate code for registers that are different from the after-reset state.
Enable Code Preview	Controls how the code is generated. When this preference is enabled, code generation is performed automatically after every change in the configuration and the Code Preview is updated accordingly. When this preference is disabled, code generation is stopped, a warning message is displayed in the Code Preview window, and the action can be manually triggered by using one of the available options: - By pressing the “generate code” link highlighted in the warning message from the Code Preview window. - By pressing the **Update Code** button in the toolbar, where code update is preceded by code generation.

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

2.7 Problems view

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

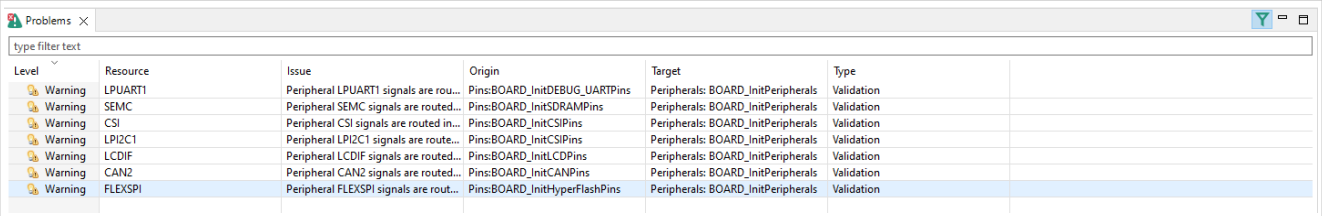


Figure 17. Problems view

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:

Table 12. Problems

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: either a validation check for dependencies between tools, or a single-tool issue.

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

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

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

Table 13. Problems view buttons

Button	Description
	Filters messages in the **Problems** view. If selected, only problems for the active tool are displayed. See Configuration preferences section for details.

2.8 Registers view

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

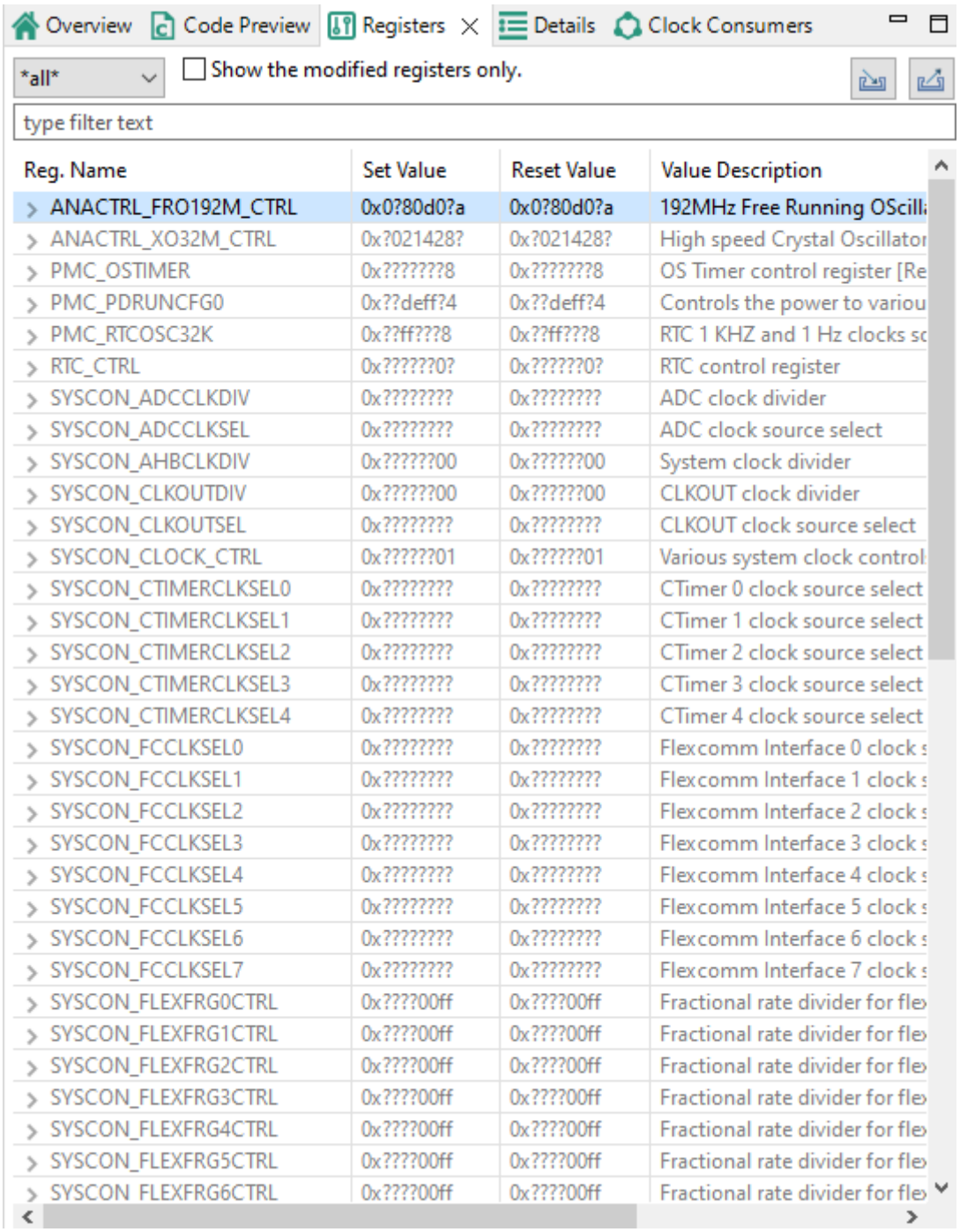


Figure 18. Registers view

The **Registers view** contains several items.

Table 14. Registers view

Item	Description
Peripheral filter drop-down list	List the registers only for the selected peripheral. Select **all** to list registers for all the peripherals.

Table 14. Registers view...*continued*

Item	Description
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.

Table 15. Color highlighting styles

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.

2.9 Log view

The **Log** view shows user-specific information about 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.

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

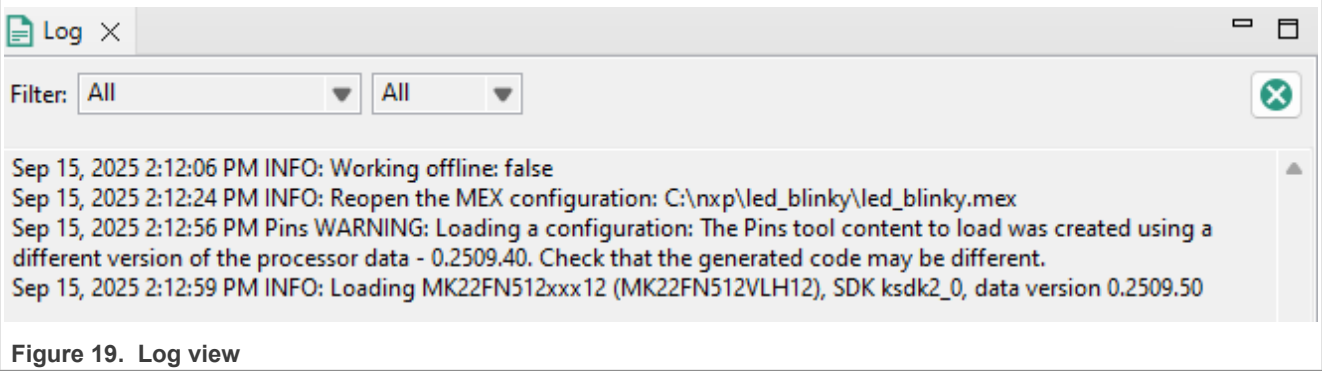


Figure 19. Log view

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.

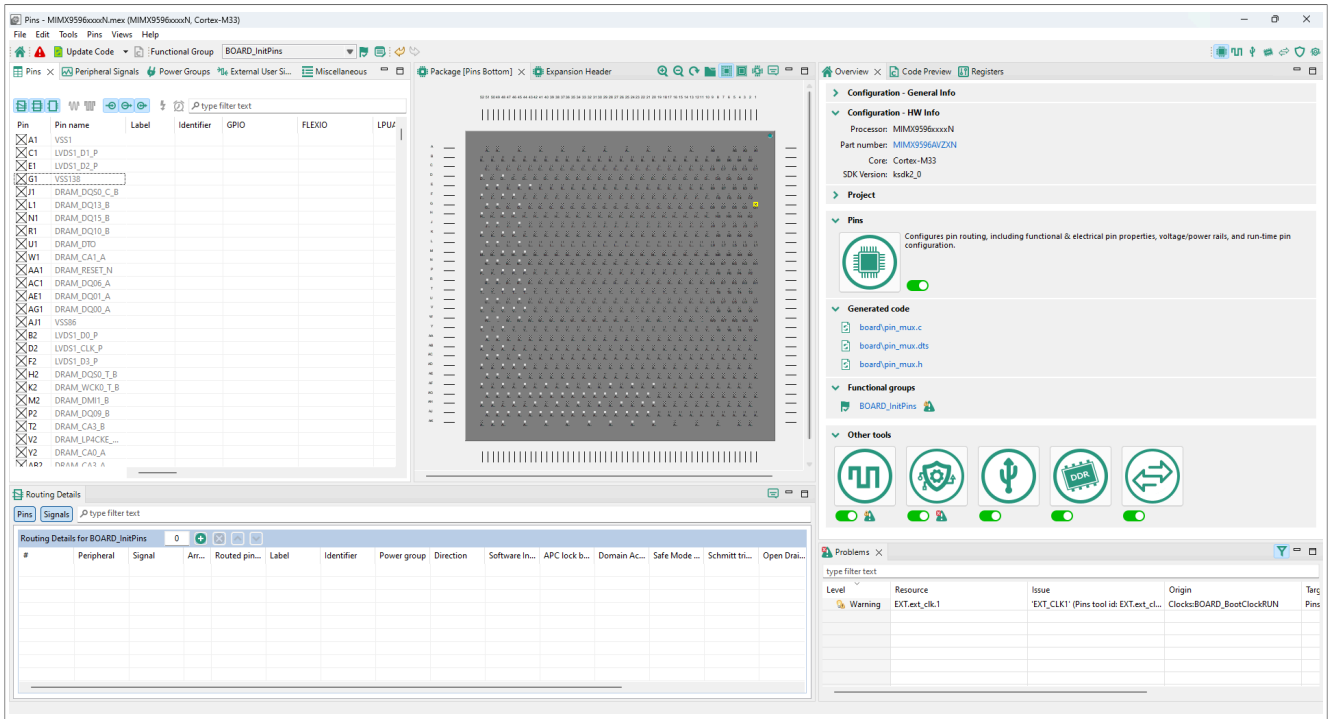


Figure 20. Pins tool

3.1 Pins routing principle

The Pins tool is designed to configure routing peripheral signals either to pins or to internal 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 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:
   - An exclamation mark and default text color indicate that the item selection can cause a register conflict, or the item does not support the selected signal.
   - An exclamation mark and gray text color indicate that the item cannot be routed to the selected pin/internal signal. The item is available for a different pin/internal signal that uses the same signal.
     Note: In the **Pins** view and the **Package** view, you can configure only pins and not internal signals.

3.1.2 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.

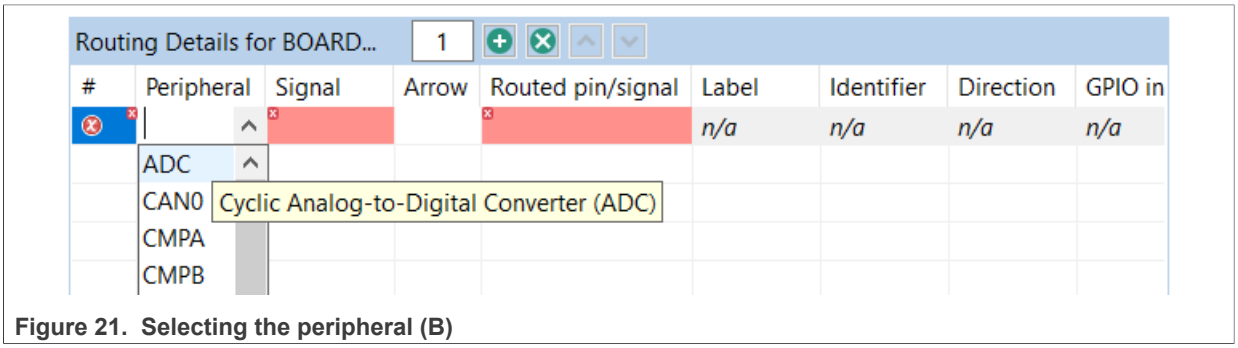


Figure 21. Selecting the peripheral (B)

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

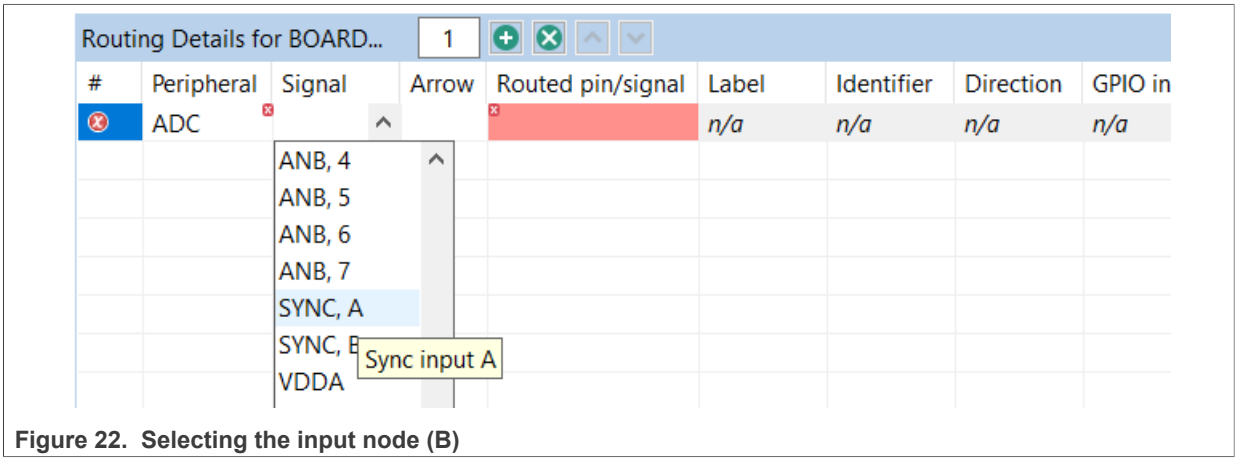


Figure 22. Selecting the input node (B)

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

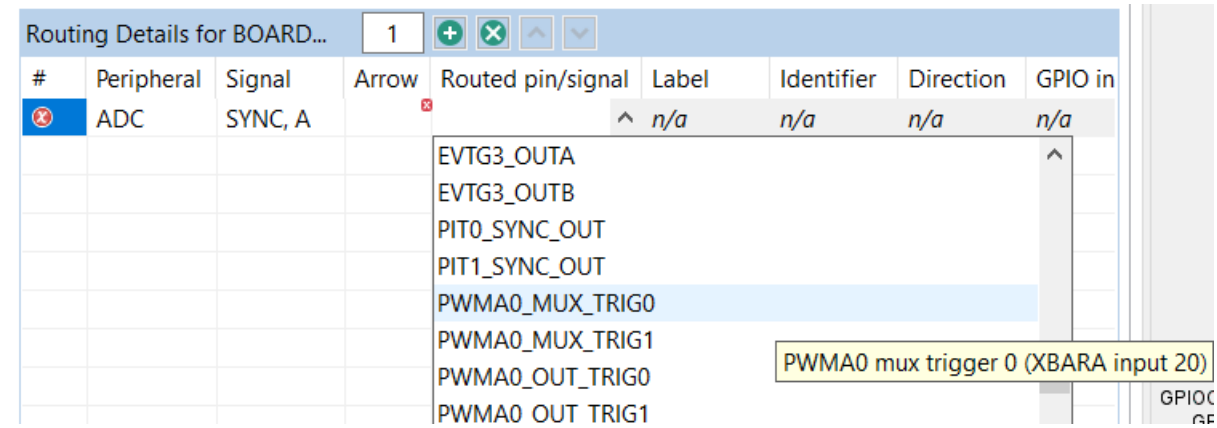


Figure 23. Selecting the output signal

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

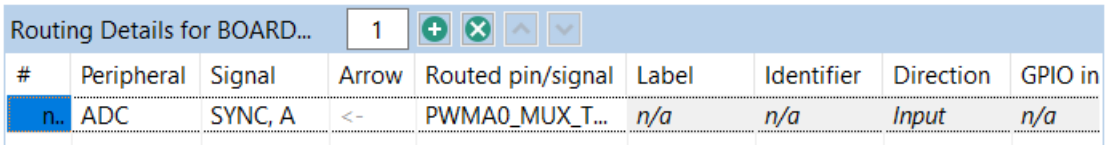


Figure 24. Result

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 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 an example):

   a. Add a row in the **Routing Details** view.
   b. Select peripheral B from the drop-down list in the **Peripheral** column.

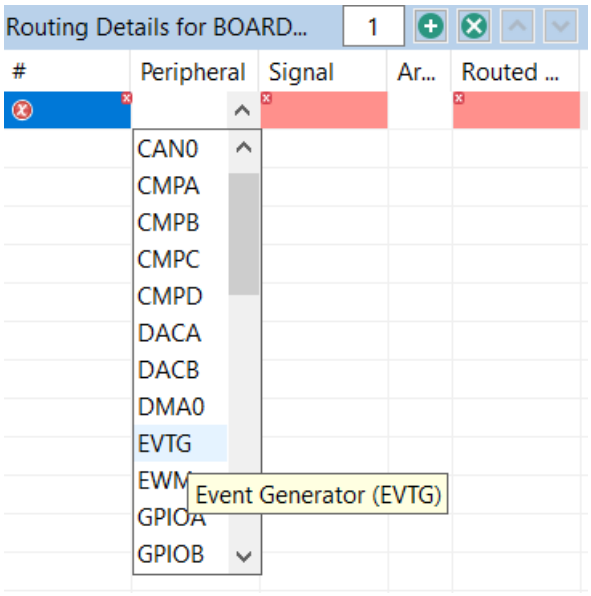


Figure 25. Selecting the peripheral (B)

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

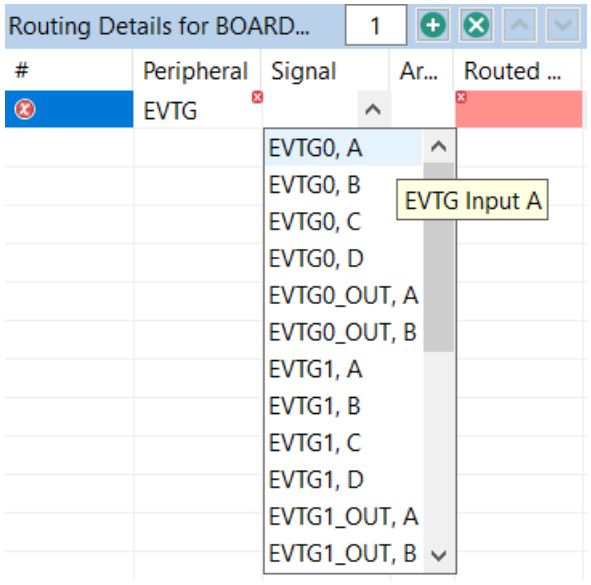


Figure 26. Selecting the input node (B)

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

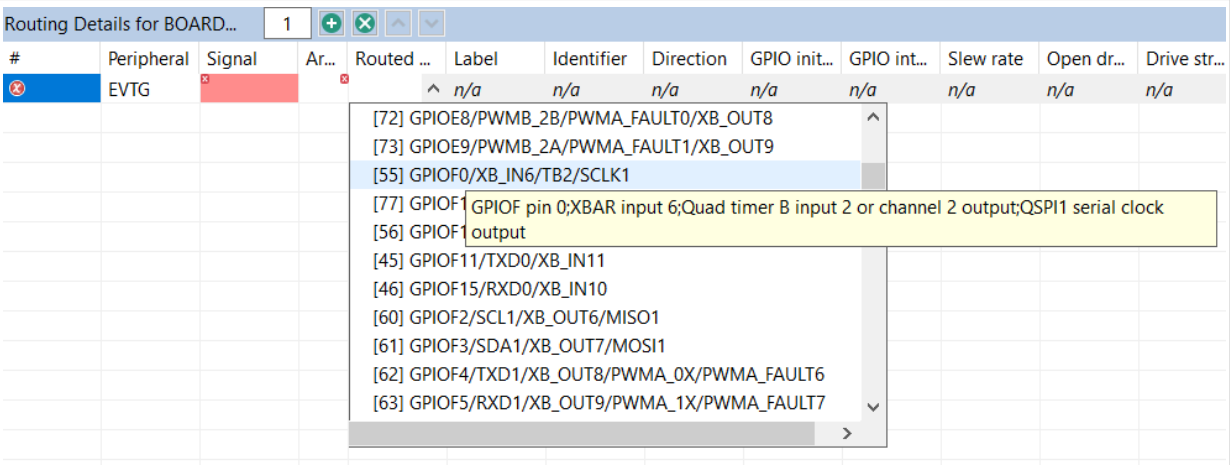


Figure 27. Selecting the pin

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

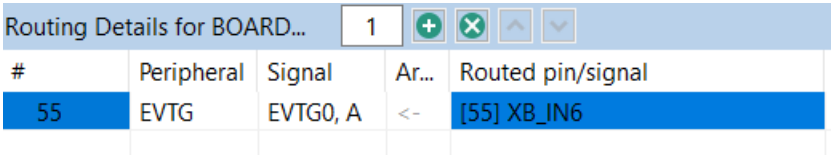


Figure 28. Result

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

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

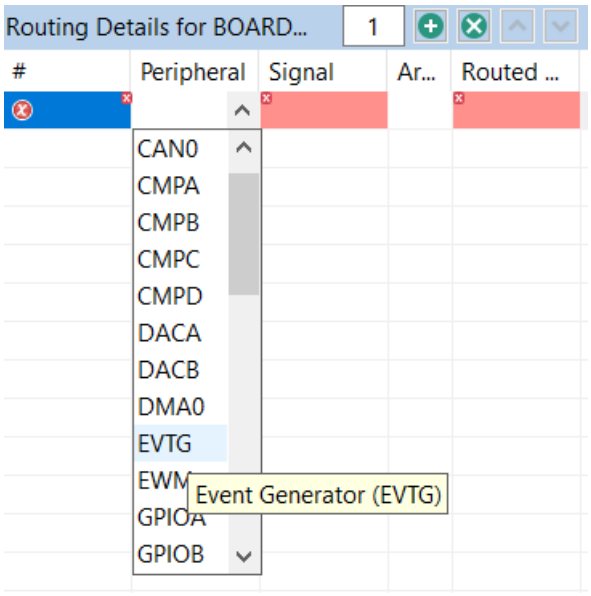


Figure 29. Selecting the peripheral (A)

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

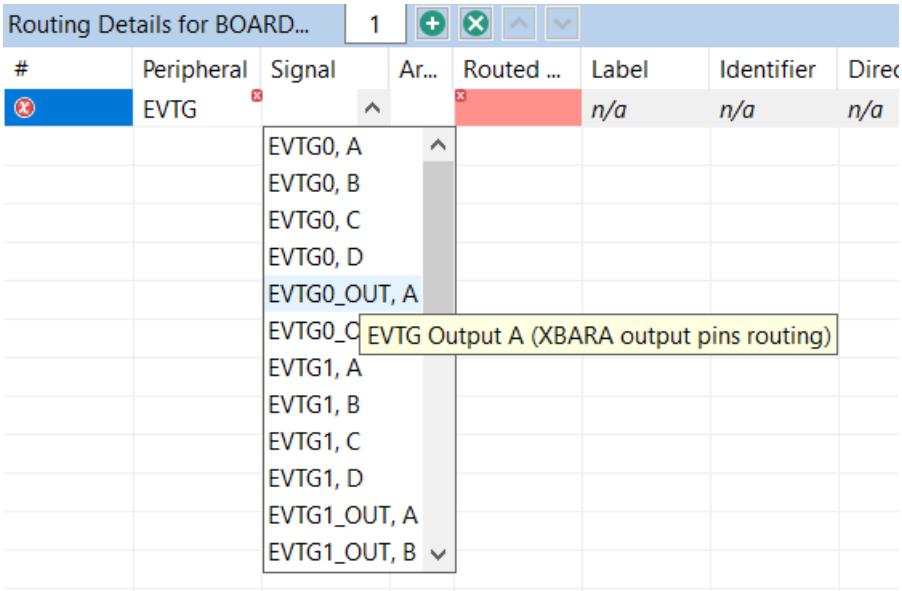


Figure 30. Selecting the output signal (A)

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

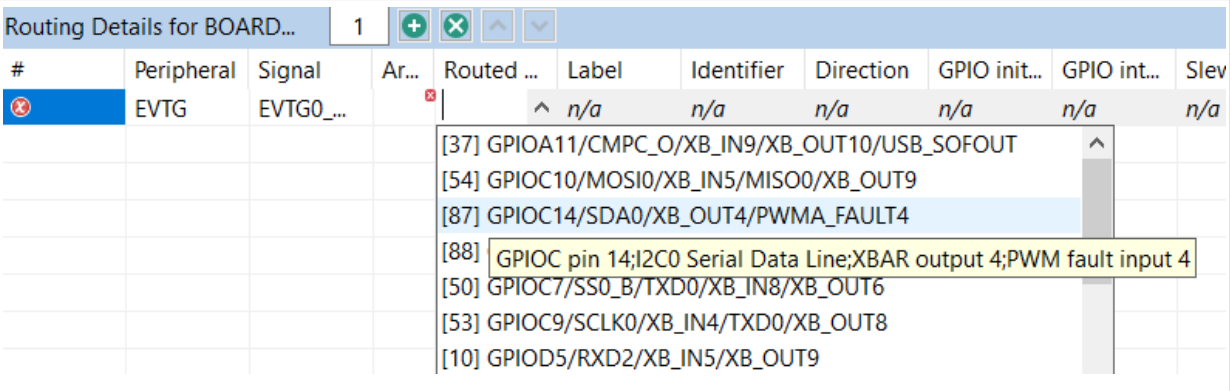


Figure 31. Selecting the pin

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

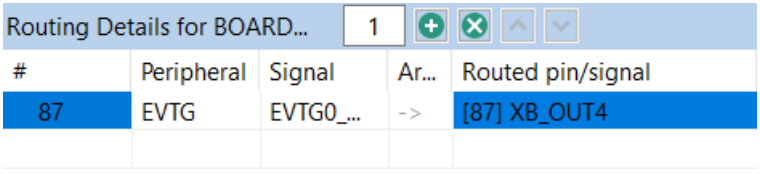


Figure 32. Result

Note: In this example, GPIOC14 is multiplexed with XB_OUT4, SDA of I2C0 and fault4 of eFlexPWMA. In this case, the tool automatically configures 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.

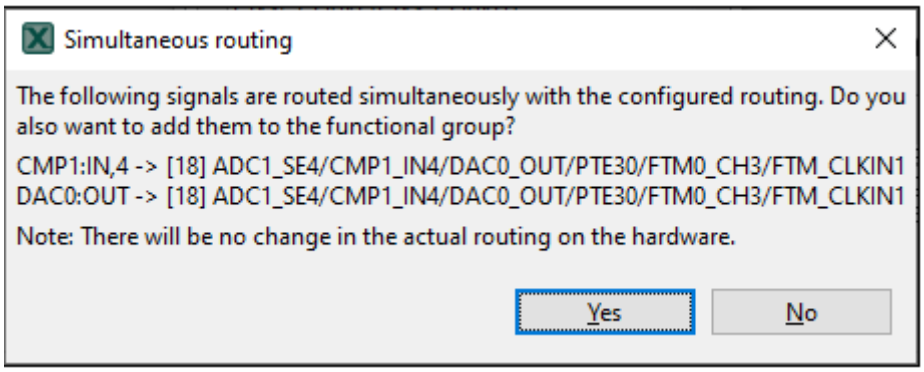


Figure 33. Simultaneous routing

3.2 Example usage

This section lists the steps to create an example pin configuration for use in a user project.

In this example, three pins (**UART4_TX**, **UART4_RX** and **GPIO_2**) routed on an MCIMX6Q-SDB-REV-B board are reconfigured to match changed (for example, customer modified) board design which is using UART4 pins instead of UART3 ones and must re-route and/or adjust electrical properties for a red LED pin, so then the tool generates files with application modifying the default board configuration.

1. Create new configuration for MCIMX6Q-SDB-REV-B board.

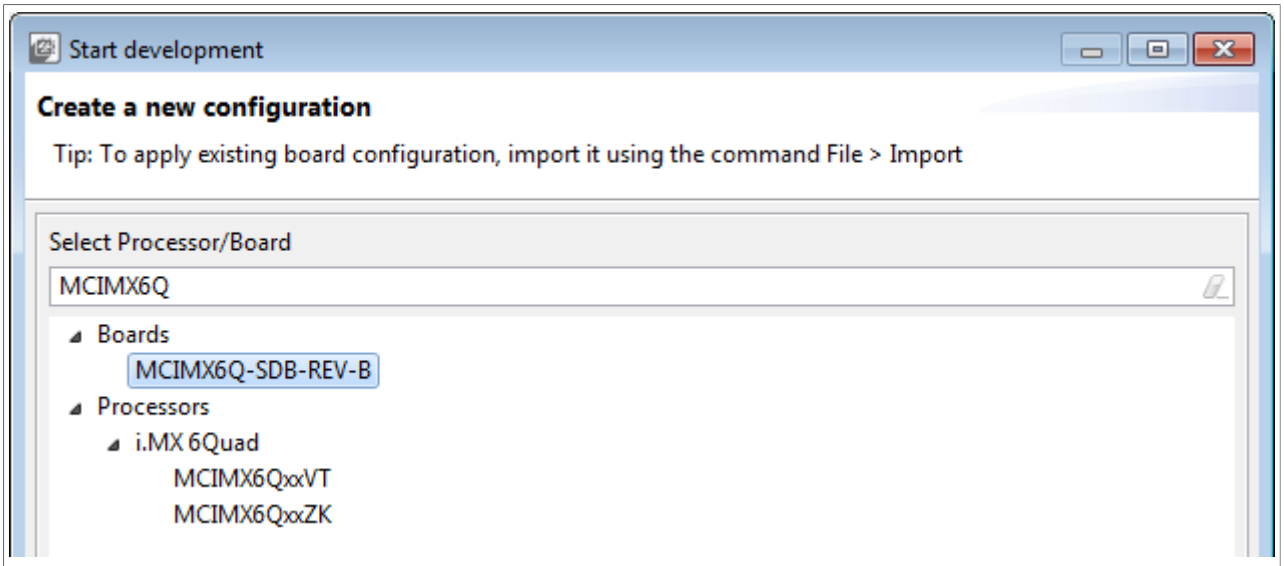


Figure 34. Create a new configuration

2. After the tool opens, select the functional group ‘init_uart_pins’ in the toolbar drop-down to show **Routed Pins** for the created configuration.

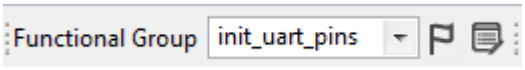


Figure 35. Select function

3. To change the original RX/TX pins routing from UART3 to UART4, you must change the configuration in the **Peripheral** and **Route to** columns of the **Routed Pins** view for the init_uart_pins selection and change the pins configuration to re-route the RX/TX pins.

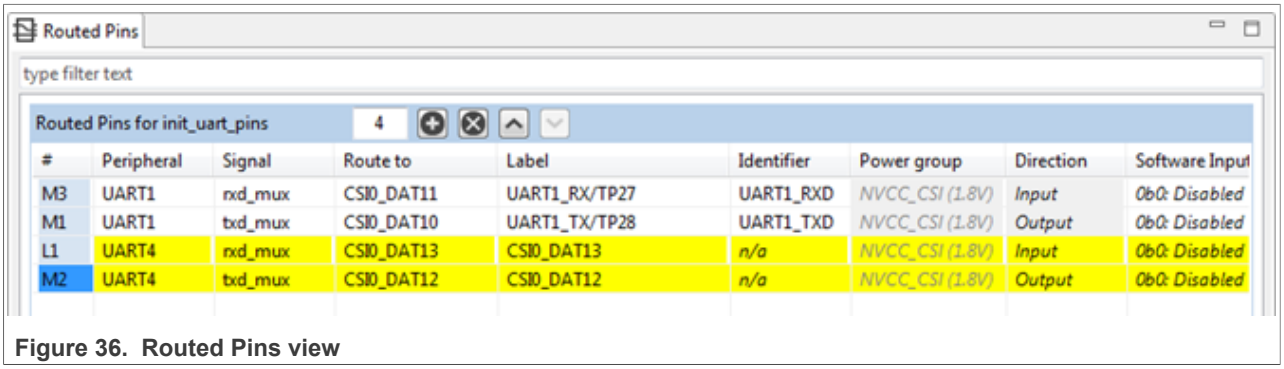


Figure 36. Routed Pins view

4. You can also adjust electrical properties configuration for these pins on the right side of the table in specific property column selection.
5. To change the configuration of the red LED pin routed originally to GPIO_2 pad, select the functional group ‘init_gpio_pins’.

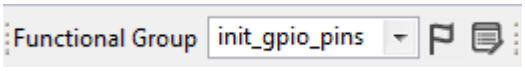


Figure 37. Select function

6. Then use filtering in the **Routed Pins** view to search configuration for “USR_DEF_RED_LED” to display simplified table content to easily modify current GPIO_2 pin routing selection.

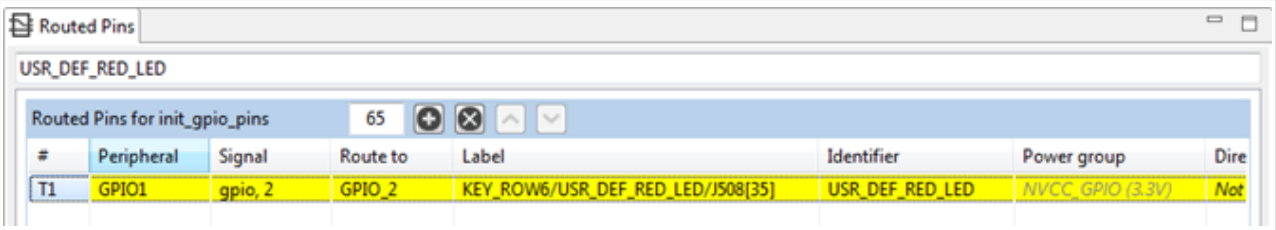


Figure 38. Routed Pins view

7. You can then adjust either electrical properties or re-route the pin to a different one if required in your modified design.
8. The Pins Tool automatically generates the source code of imx6q-board.dtsi, pin_mux.c, and pin_mux.h in the **Code Preview** view on the right.

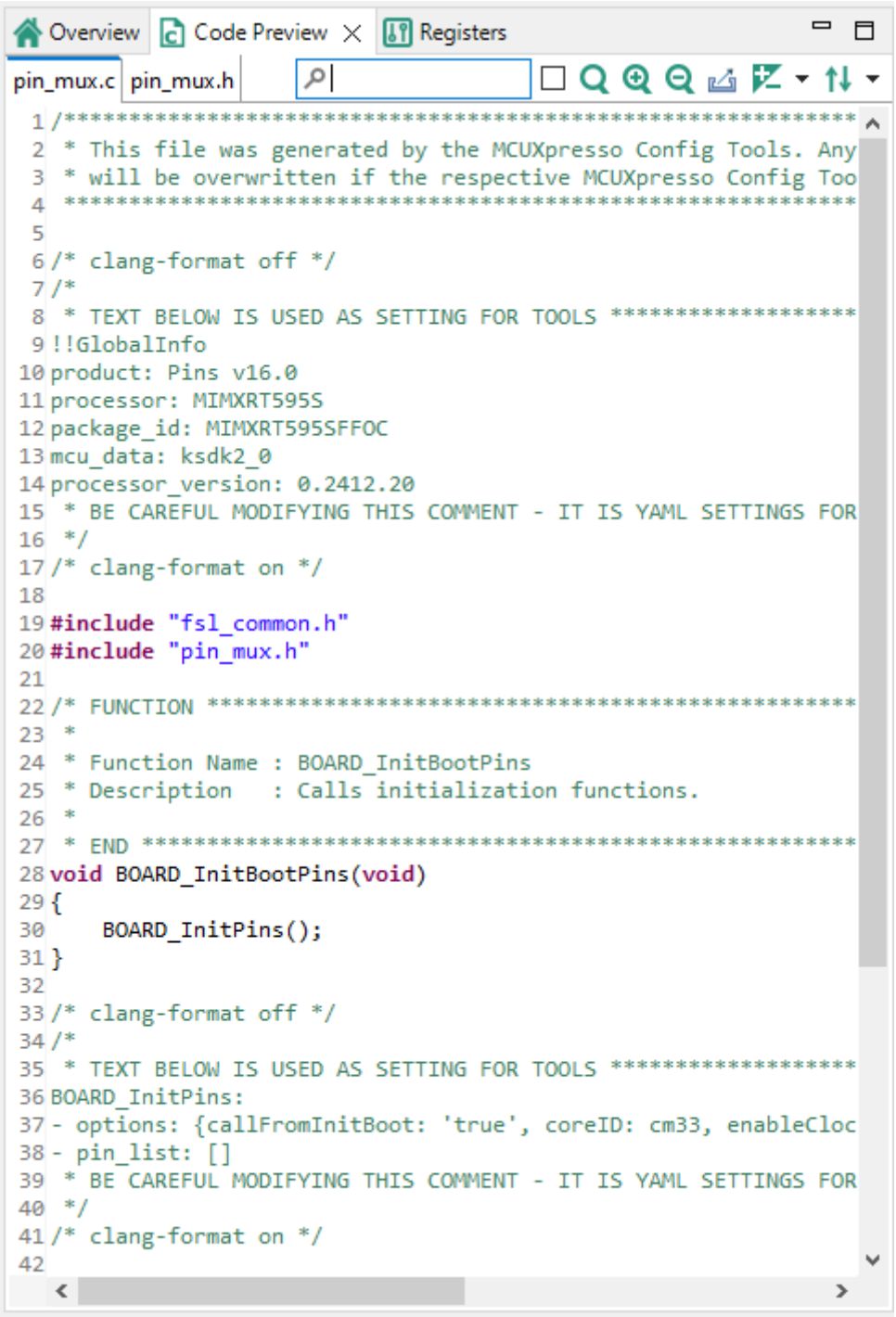


Figure 39. Generated sources

9. You can now copy-paste the content of the source(s) to your application or IDE. Alternatively, you can export the generated files. To export the files, click Export button on the right up corner of **Code Preview** view or select the menu **File > Export**, in the **Export**dialog expand the tree control for the Pins Tool and select the **Export Source Files** option.
   Note: Tool generated board-oriented device tree (DTS) DTSI file is only a snippet and not a full device tree file content. There are just basic device tree elements, initial skeleton, and processor-specific “pinfunc.h”

includes together with functional groups of fsl, pins = >…> content definitions which provide the initial IOMUXC module configuration according to the tool UI defined pin routing and functional configurations. Content itself must be manually merged together with existing Linux BSP device tree file(s) to apply the tool generated pins configuration. This tool also does not generate nor export processor-specific “pinfunc.h” file that is containing definition of all supported DTS pin functional configuration macros. This file is not purposely integrated within the tool output because it is a part of separate Linux BSP support package deliverables.

3.3 User interface

The Pins tool consists of several views.

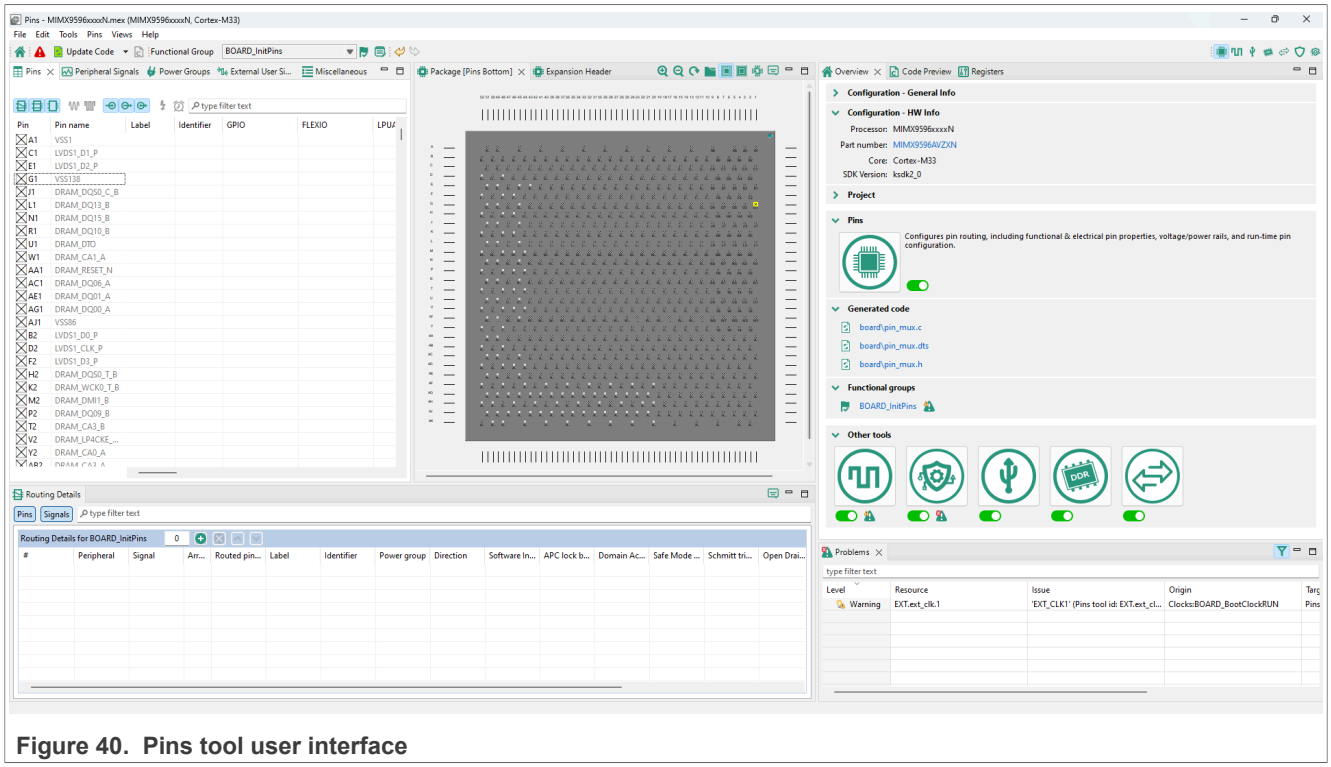
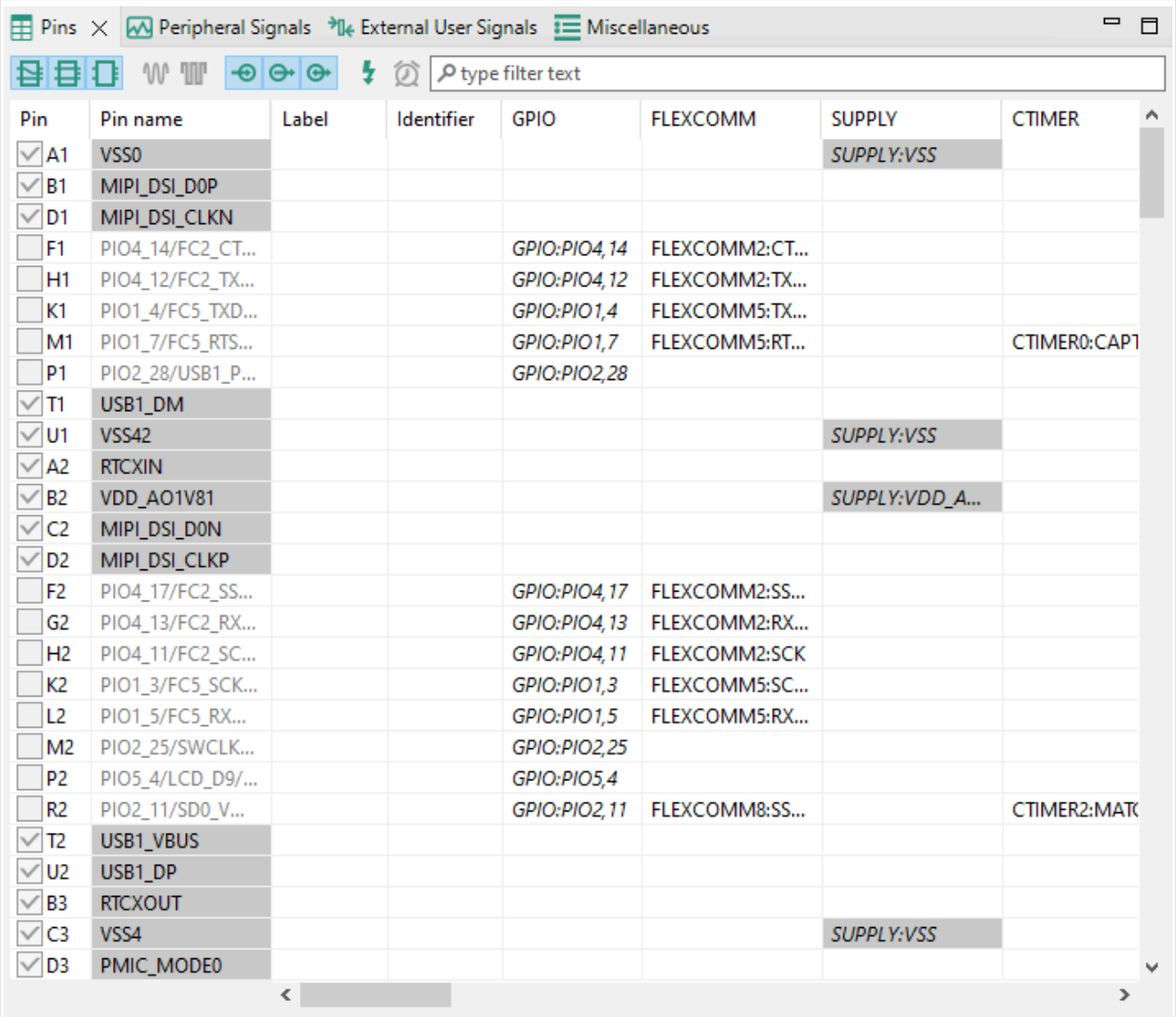


Figure 40. Pins tool user interface

3.3.1 Pins view

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



Pin	Pin name	Label	Identifier	GPIO	FLEXCOMM	SUPPLY	CTIMER
A1	VSS0					SUPPLY:VSS	
B1	MIPI_DSI_D0P						
D1	MIPI_DSI_CLKN						
F1	PIO4_14/FC2_CT...			GPIO:PIO4,14	FLEXCOMM2:CT...		
H1	PIO4_12/FC2_TX...			GPIO:PIO4,12	FLEXCOMM2:TX...		
K1	PIO1_4/FC5_TXD...			GPIO:PIO1,4	FLEXCOMM5:TX...		
M1	PIO1_7/FC5_RTS...			GPIO:PIO1,7	FLEXCOMM5:RT...		CTIMER0:CAPT
P1	PIO2_28/USB1_P...			GPIO:PIO2,28			
T1	USB1_DM						
U1	VSS42					SUPPLY:VSS	
A2	RTCXIN						
B2	VDD_AO1V81					SUPPLY:VDD_A...	
C2	MIPI_DSI_D0N						
D2	MIPI_DSI_CLKP						
F2	PIO4_17/FC2_SS...			GPIO:PIO4,17	FLEXCOMM2:SS...		
G2	PIO4_13/FC2_RX...			GPIO:PIO4,13	FLEXCOMM2:RX...		
H2	PIO4_11/FC2_SC...			GPIO:PIO4,11	FLEXCOMM2:SCK		
K2	PIO1_3/FC5_SCK...			GPIO:PIO1,3	FLEXCOMM5:SC...		
L2	PIO1_5/FC5_RX...			GPIO:PIO1,5	FLEXCOMM5:RX...		
M2	PIO2_25/SWCLK...			GPIO:PIO2,25			
P2	PIO5_4/LCD_D9/...			GPIO:PIO5,4			
R2	PIO2_11/SD0_V...			GPIO:PIO2,11	FLEXCOMM8:SS...		CTIMER2:MAT(
T2	USB1_VBUS						
U2	USB1_DP						
B3	RTCXOUT						
C3	VSS4					SUPPLY:VSS	
D3	PMIC_MODE0						

Figure 41. 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 changes 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.

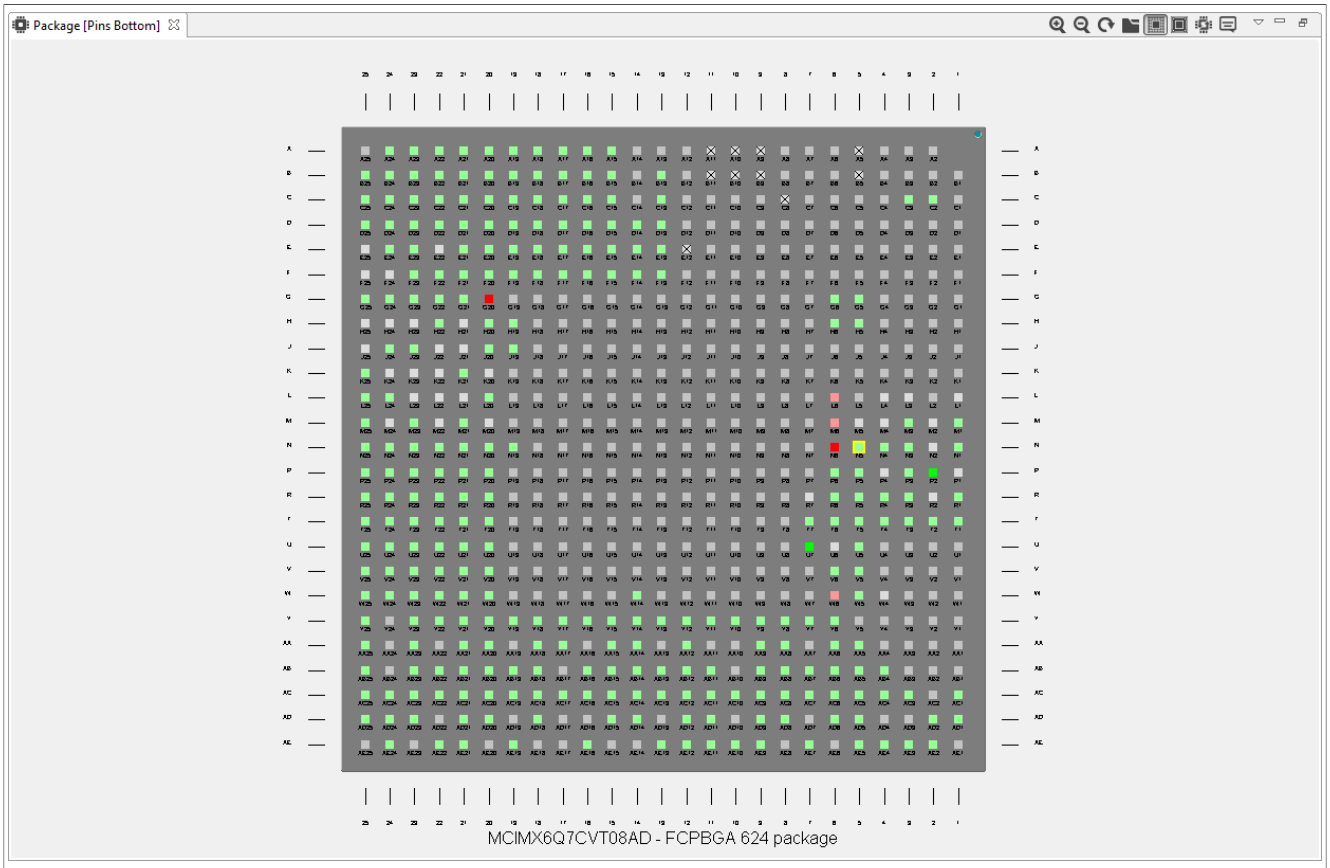


Figure 42. Processor package

This view shows the package overview with pin locations. In the center are the peripherals.

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:

Table 16. Toolbar icons

Icon	Description
	Zoom in package image.
	Zoom out package image.
	Rotate package image.
	Show pins as seen from the bottom. This option is available on BGA packages only.
	Show pins as seen 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** icon launches **Switch package for the Processor**.

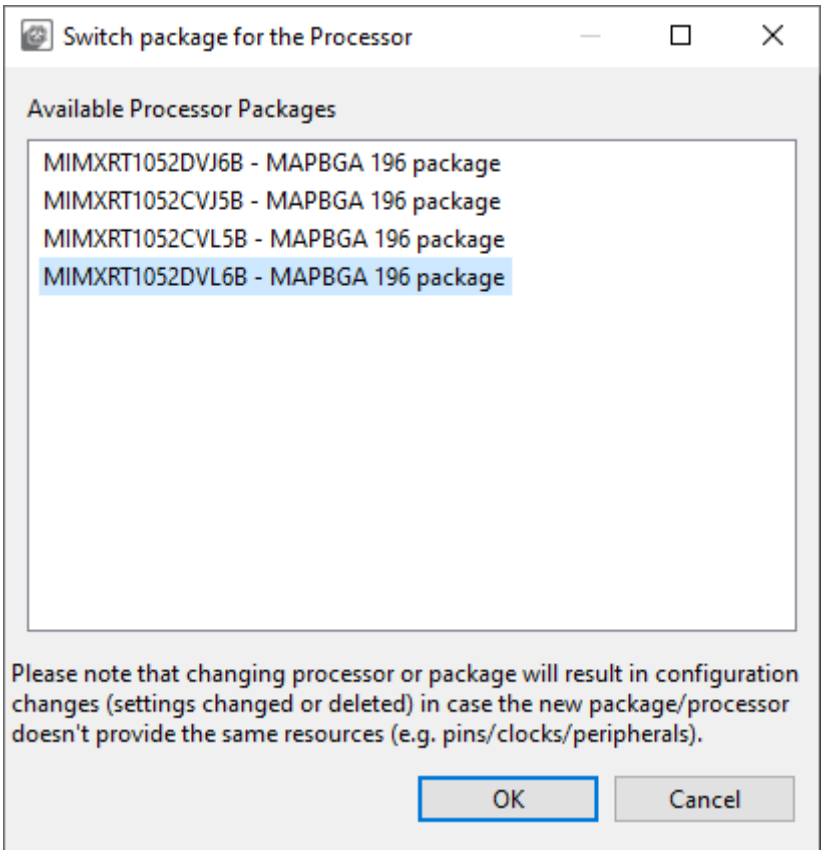


Figure 43. Switch package

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.

Table 17. Peripheral Signals view

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

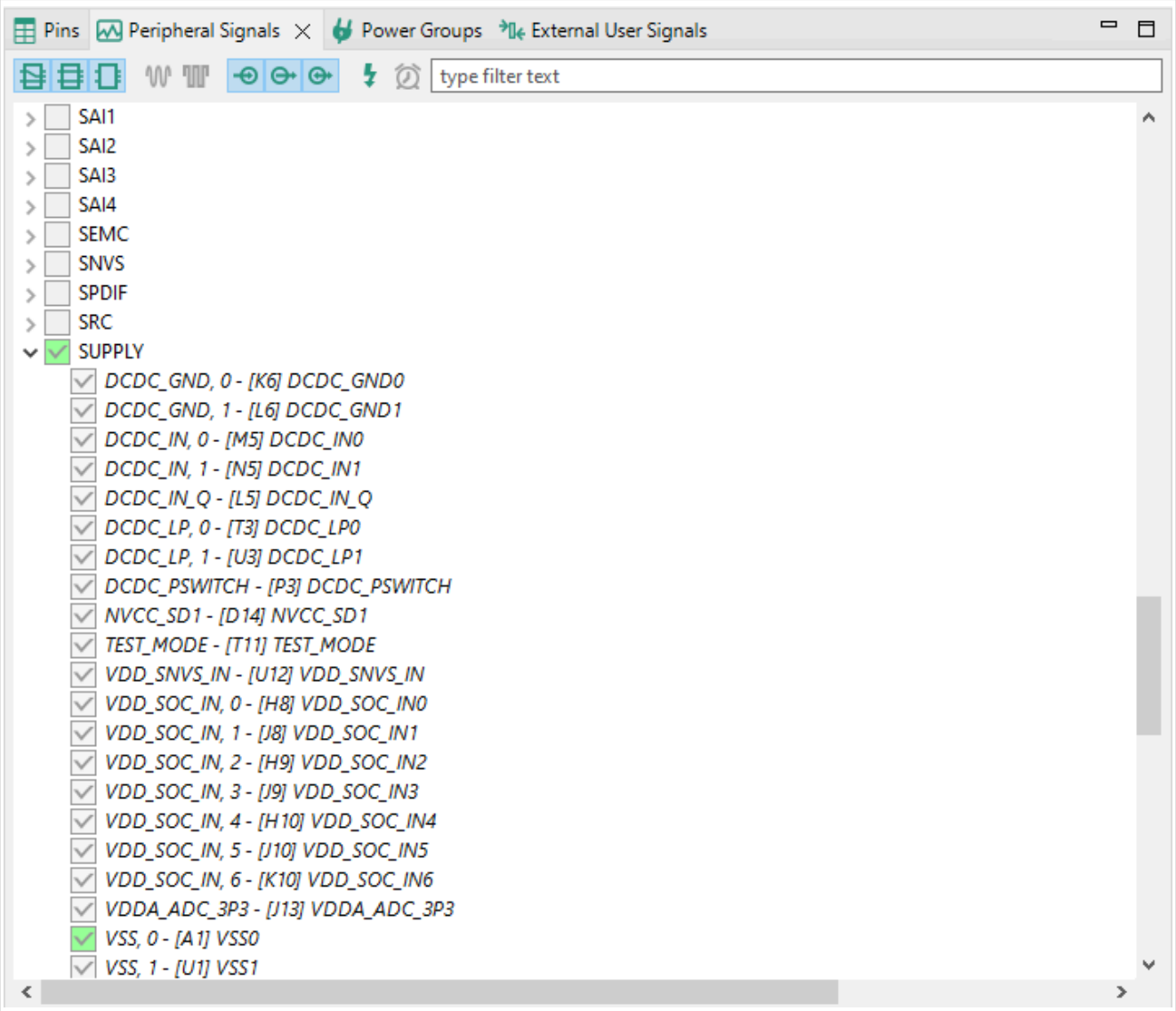


Figure 44. Peripheral Signals view

Use the checkbox to route/unroute the pins.

To highlight the pin/routing configuration for 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.

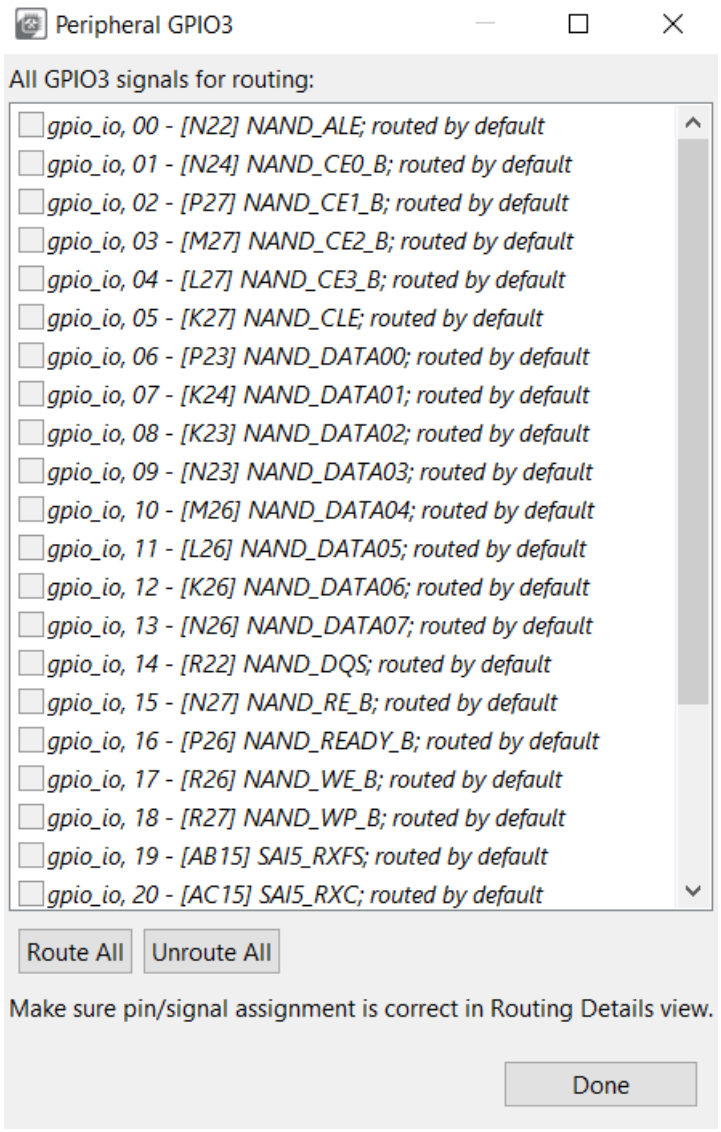


Figure 45. Select signals dialog

3.3.3.1 Filtering in the Pins and Peripheral Signals views

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

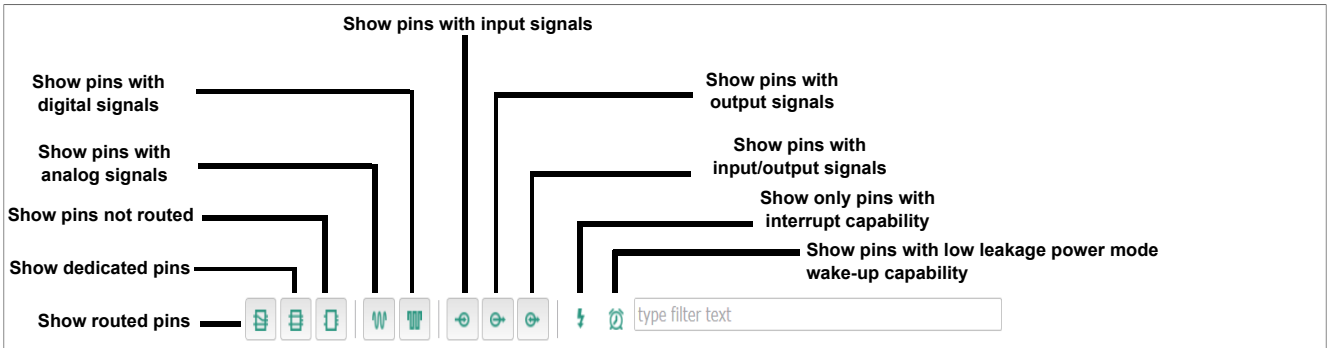


Figure 46. Filtering controls

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

3.3.3.2 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 already routed.

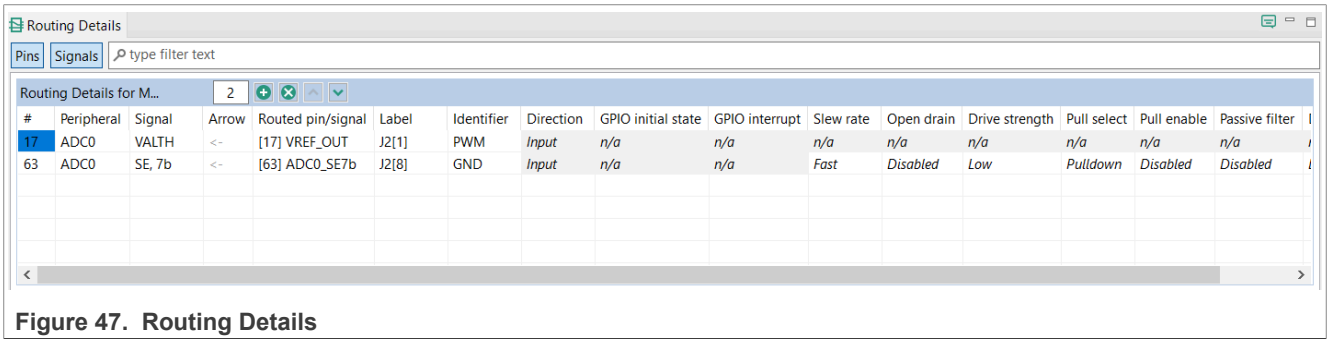


Figure 47. Routing Details

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, the tool also searches full pin names and displays 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.

3.3.3.3 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 a maximum length of 128 characters; submitting an empty label also deletes the identifier
  - **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. Available only if the pin direction is set to output.
  - **GPIO interrupt** - Configuration of interrupt request for the pin. 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 if it is necessary for proper routing of the pin or internal signal to the 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. Italic font indicates this state. 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, 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 is in conflict with the value of other property. Typically conflicts arise when two pins or internal signals configures same register and bitfield with different initialization value. To resolve the conflict, change the configuration of one of the pins or internal signals.
- 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 tool can ignore the property.
- The user must set the property. 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.

3.3.3.4 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** columns visibility, select **Edit > Preferences** from the **Menu bar**, and select the **Show pin label & identifier table columns (Pins tool)** checkbox.

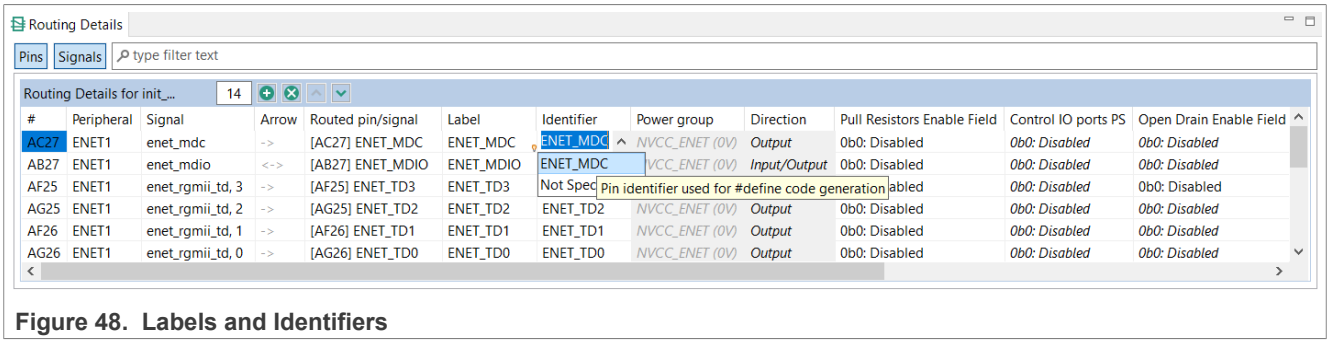


Figure 48. Labels and Identifiers

The pin identifier generates 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.

Figure 49. Pin identifier

In this case, you can select from available values if the pin is routed. See the **Identifier** column in the **Routing Details** view.

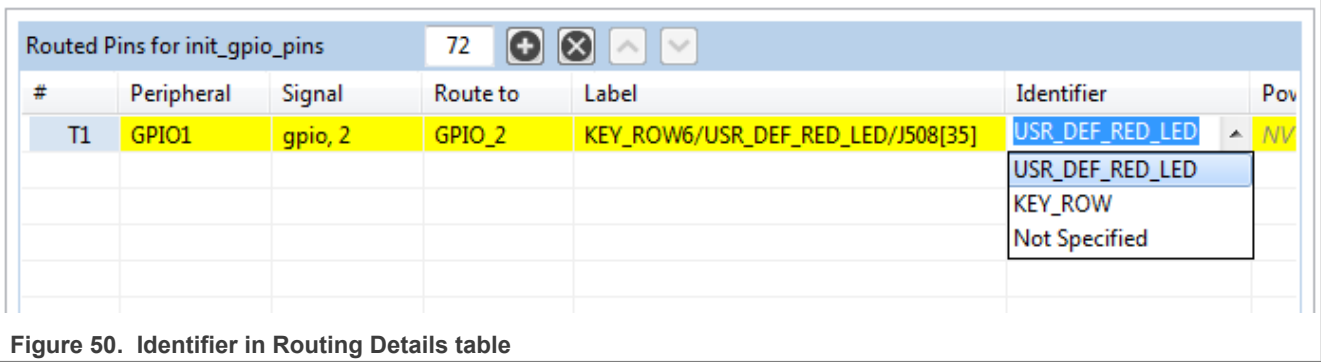


Figure 50. Identifier in Routing Details table

A check verifies whether the generated defines are duplicated in the `pin_mux.h` file. These duplications are indicated in the identifier column as errors.

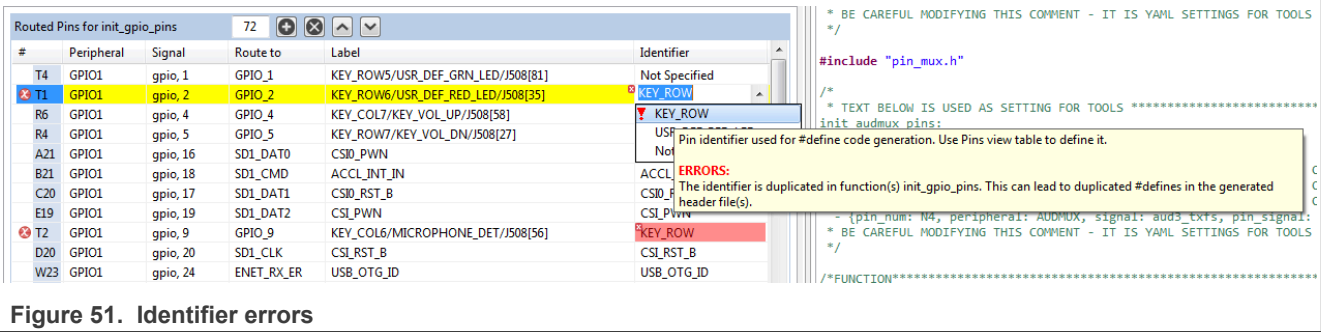


Figure 51. Identifier errors

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

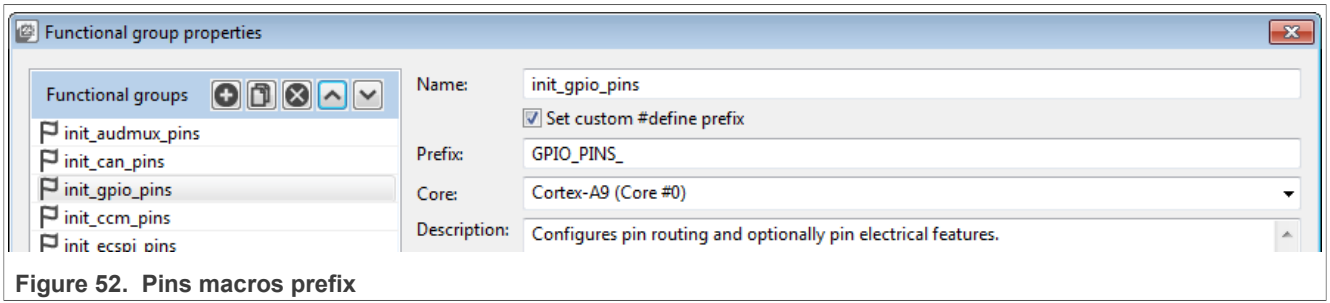


Figure 52. Pins macros prefix

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

3.3.4 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.

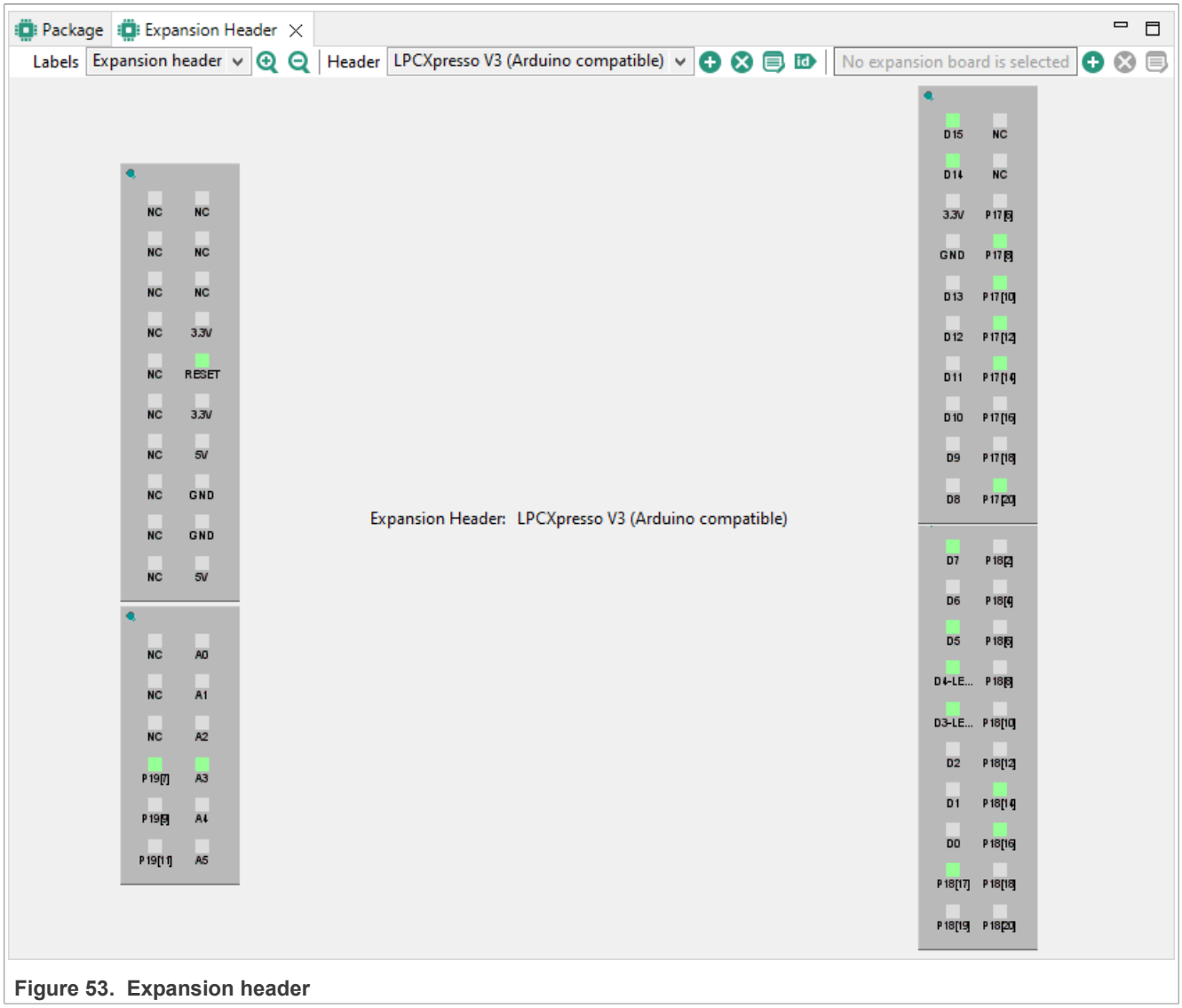


Figure 53. Expansion header

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

1. Open the view by selecting **Views > Expansion Header** from the **Toolbar**.
2. Add a 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.

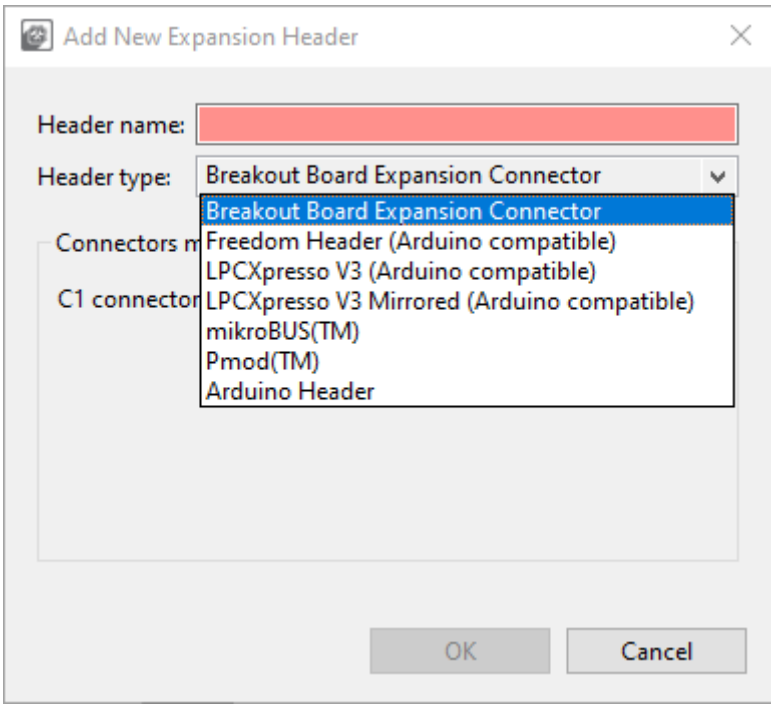


Figure 54. Adding new expansion header

4. Name the header and map the connectors.

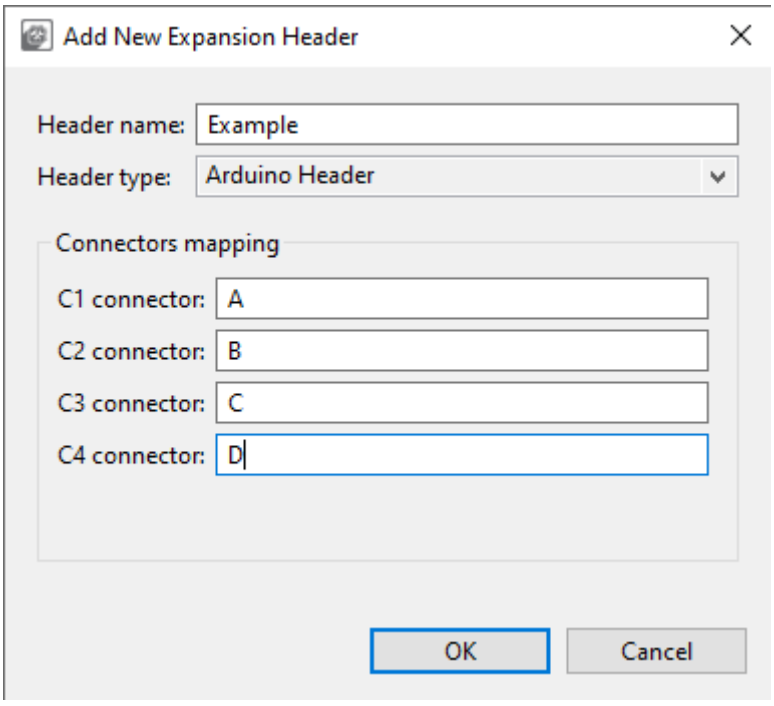


Figure 55. Adding new expansion header

5. Select **OK**.
   Expansion Header view now displays the connector layout. Hover over a pin to display additional information. Right-click the pin to display a shortcut menu of additional options.

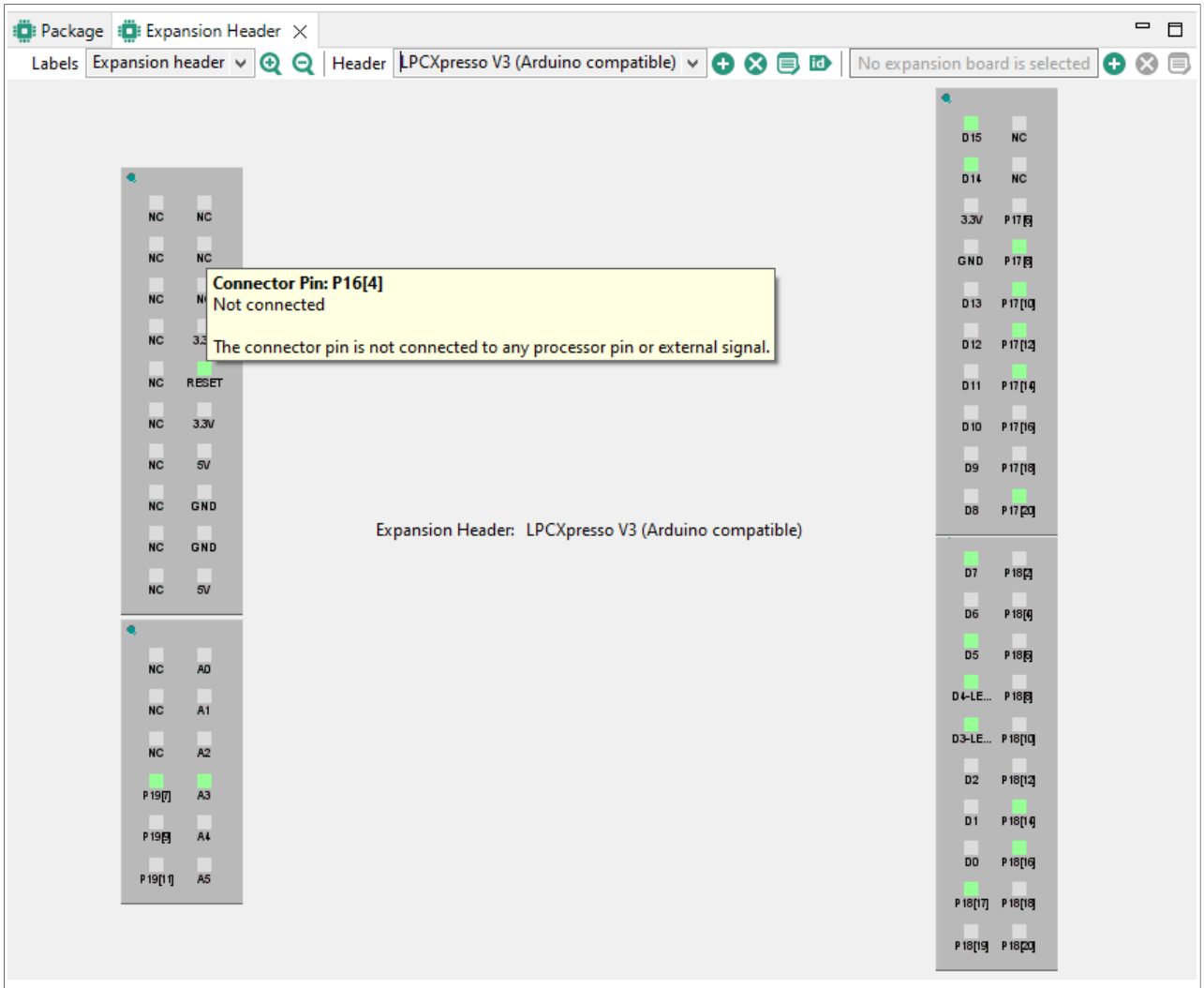


Figure 56. Expansion header

6. To map the header pin to a 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**.

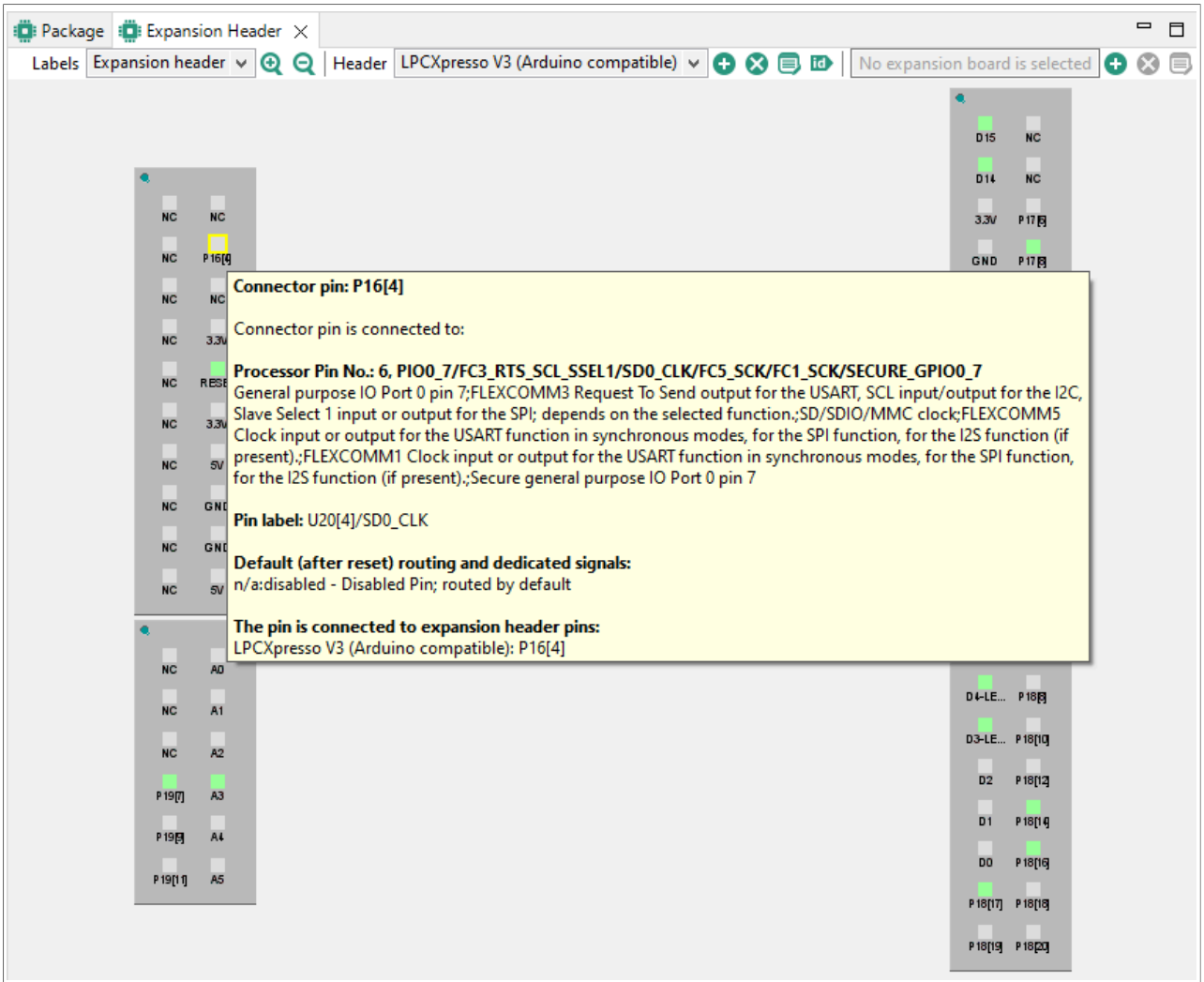


Figure 57. Connected pin

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.

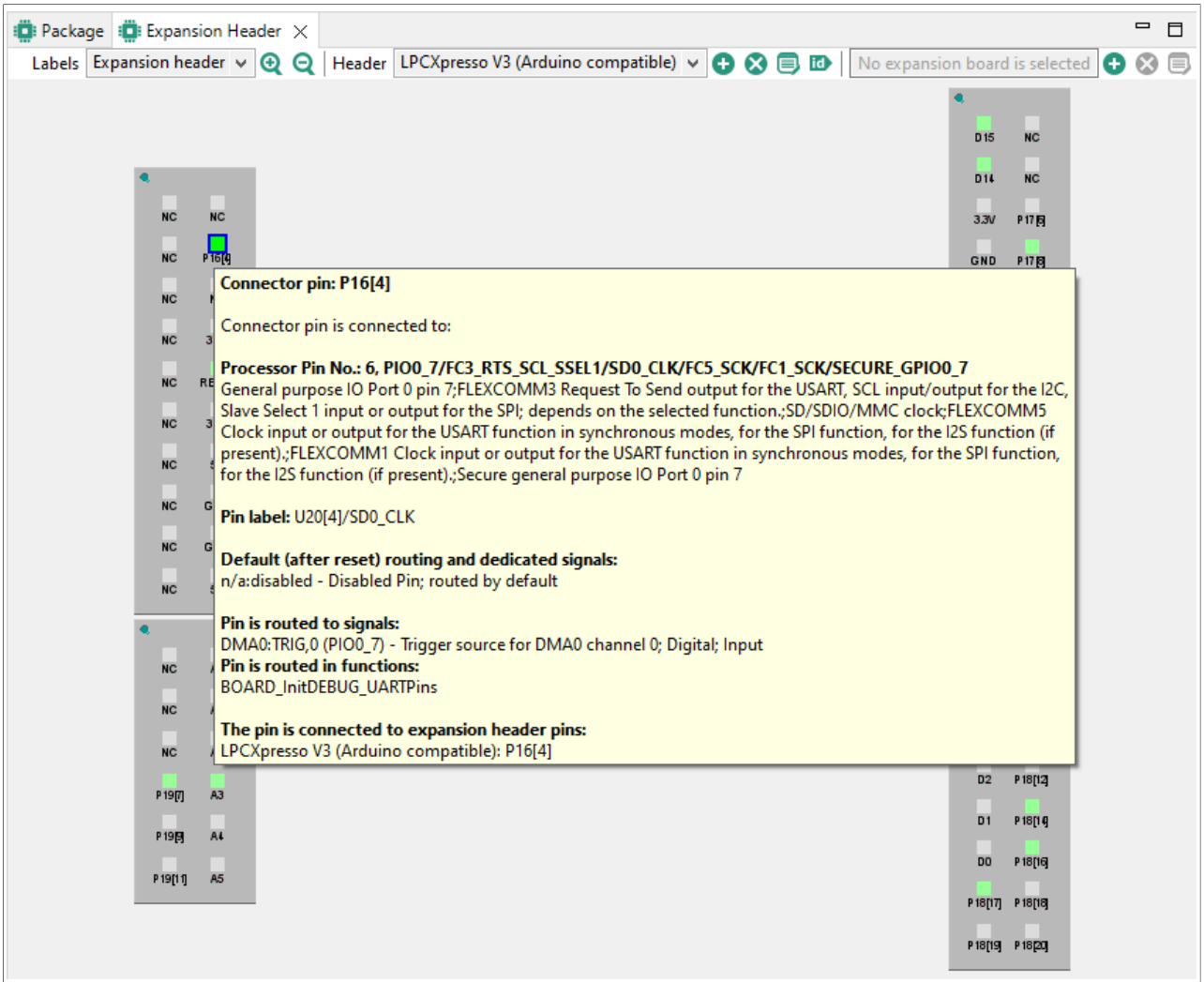


Figure 58. Routed pin

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 updates automatically. Pin connections between the header and the processor, and their labels, can be locked to prevent 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.

3.3.4.1 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 then recommends 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**.

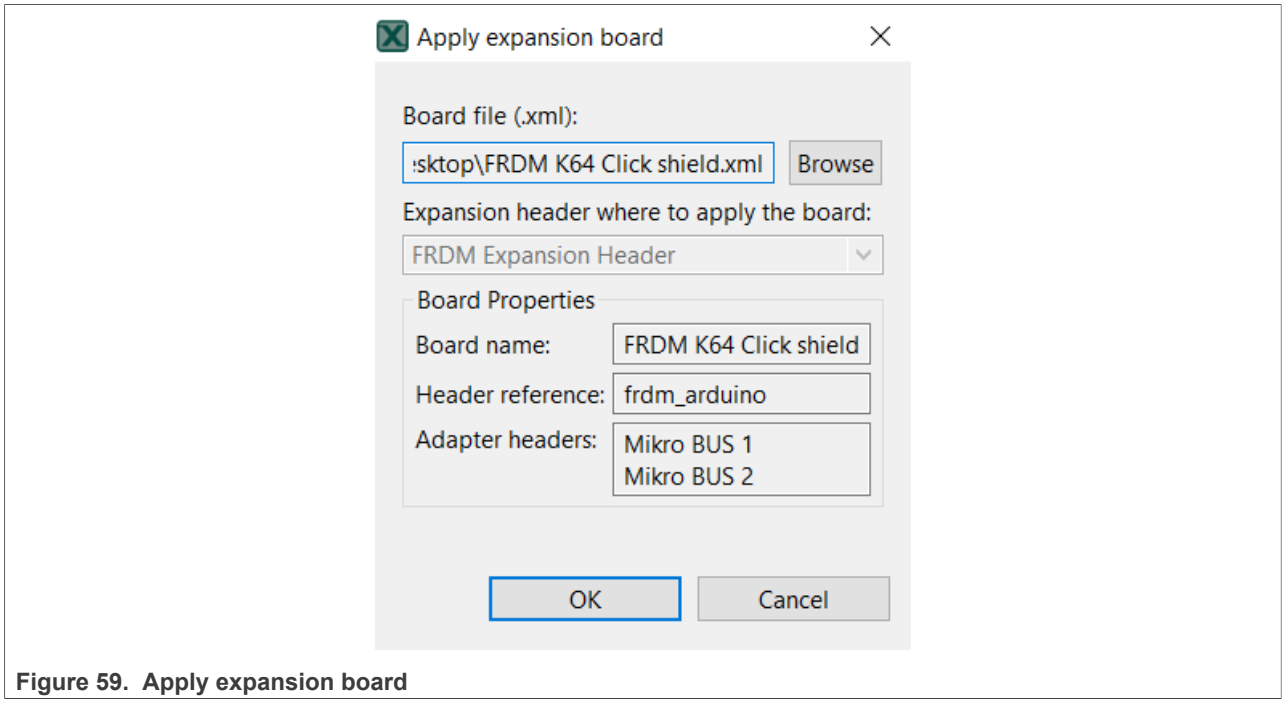


Figure 59. Apply expansion board

3. Click **OK** to apply the expansion board.
4. On the next page, choose whether to create a new functional group for the expansion board or modify an existing one. If modifying, 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. To change the route of a pin, click the pin cell in the **Route** column, select the signal in the **Connector pin** dialog, and click **Done**.

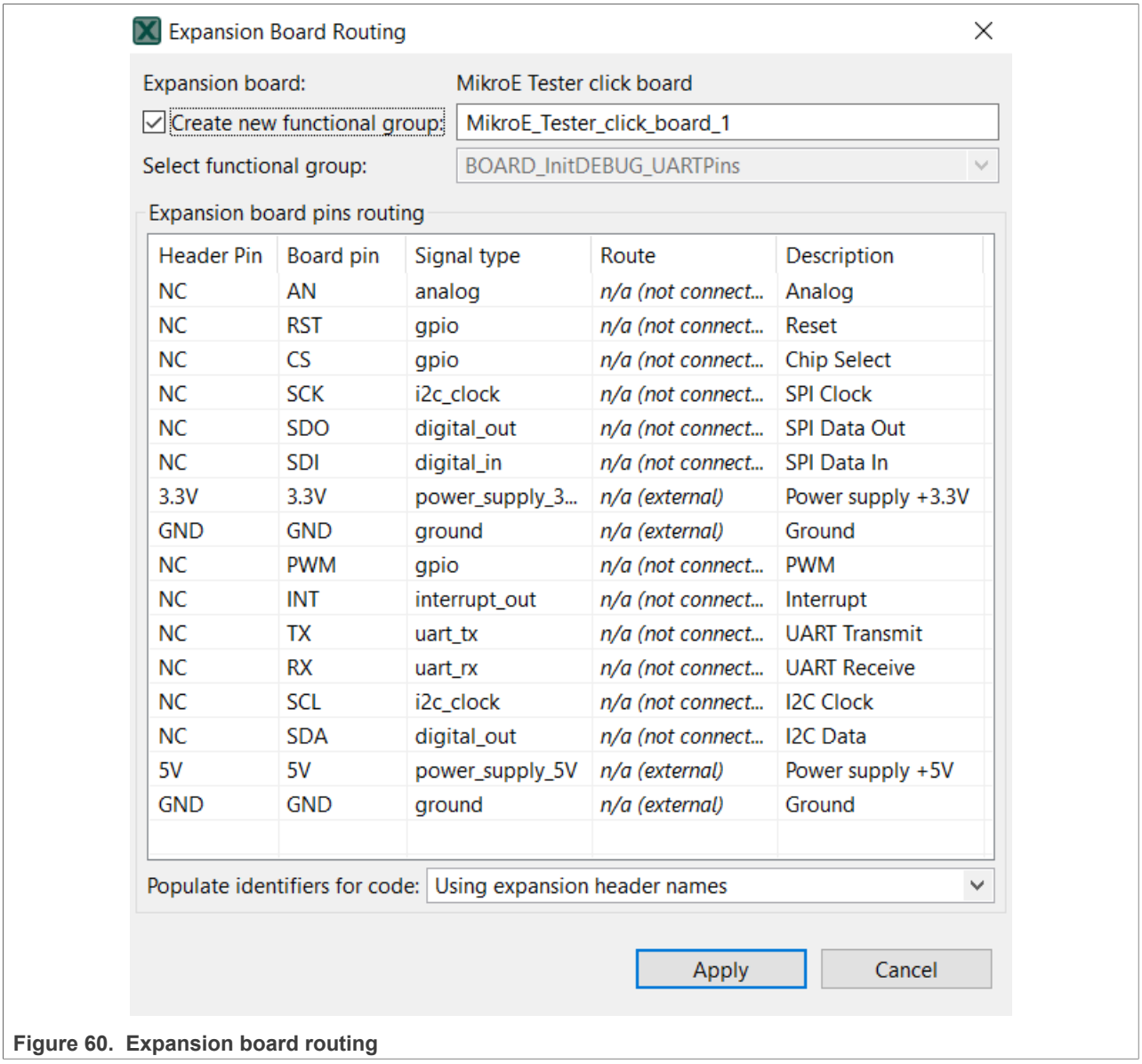


Figure 60. Expansion board routing

6. Choose how to populate identifiers for code. The 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.

3.3.5 Power groups

If your processor supports power groups, an additional tab appears next to **Pins** and **Peripheral Signals**.

Note: This feature is not supported for all devices.

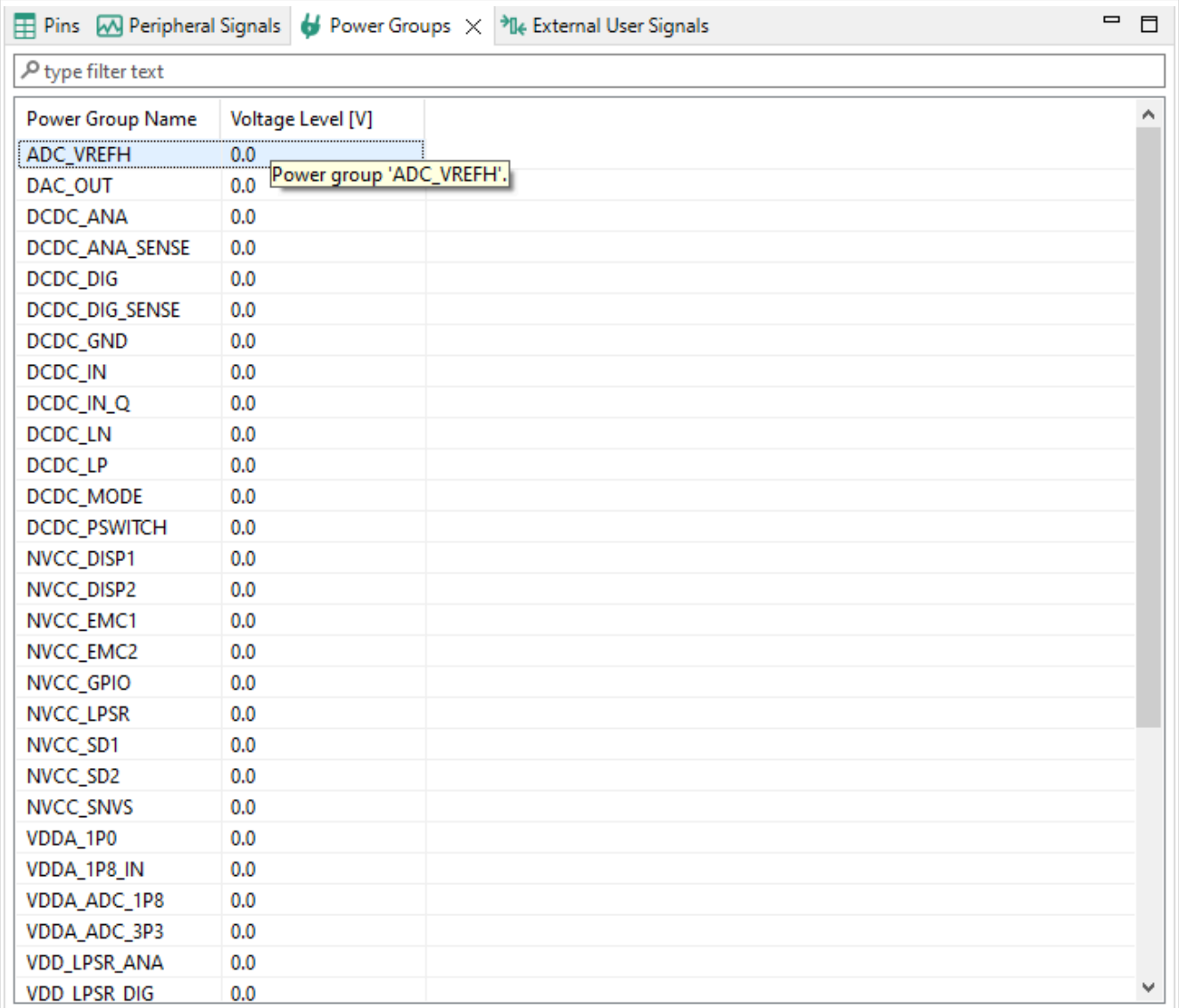


Figure 61. Selecting power group

3.3.6 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.

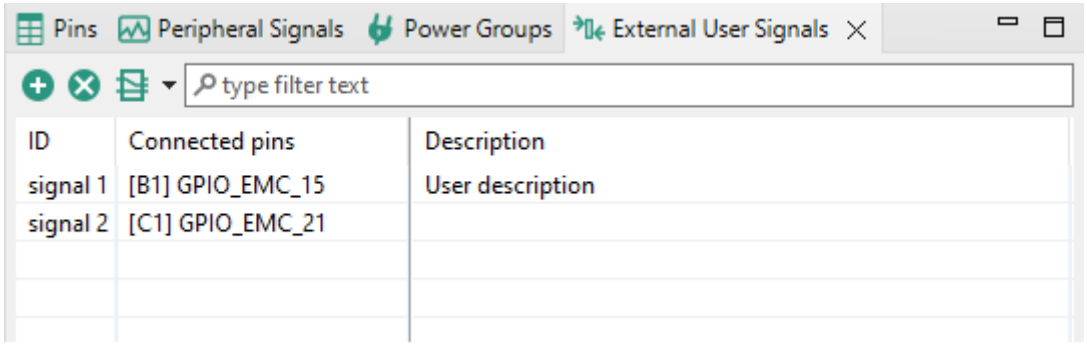


Figure 62. External User Signals view

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.

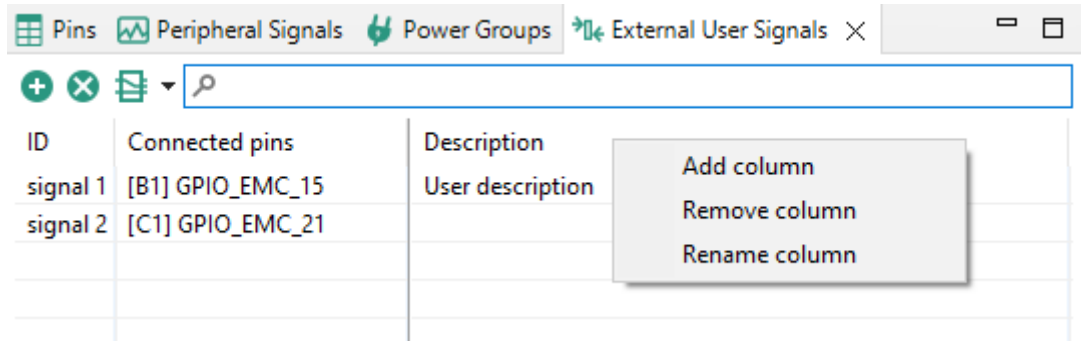


Figure 63. External User Signals header context menu

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

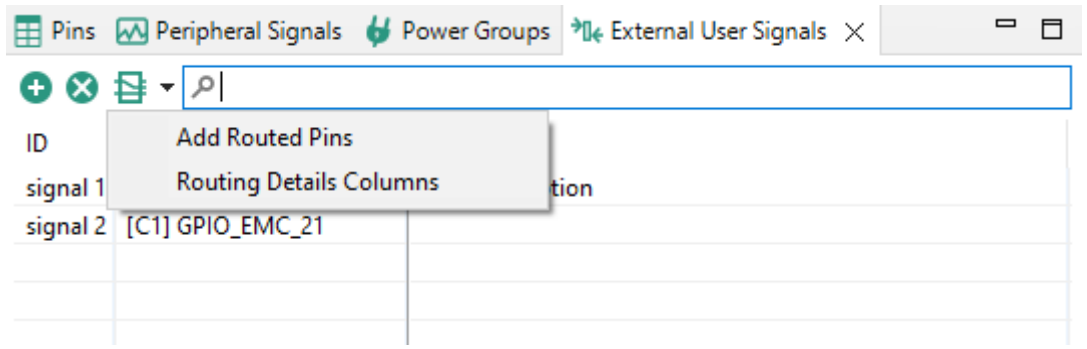


Figure 64. External User Signals view

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 are displayed in the table. Those columns are read-only and they always reflect actual values in the **Routing Details** view for all functional groups.

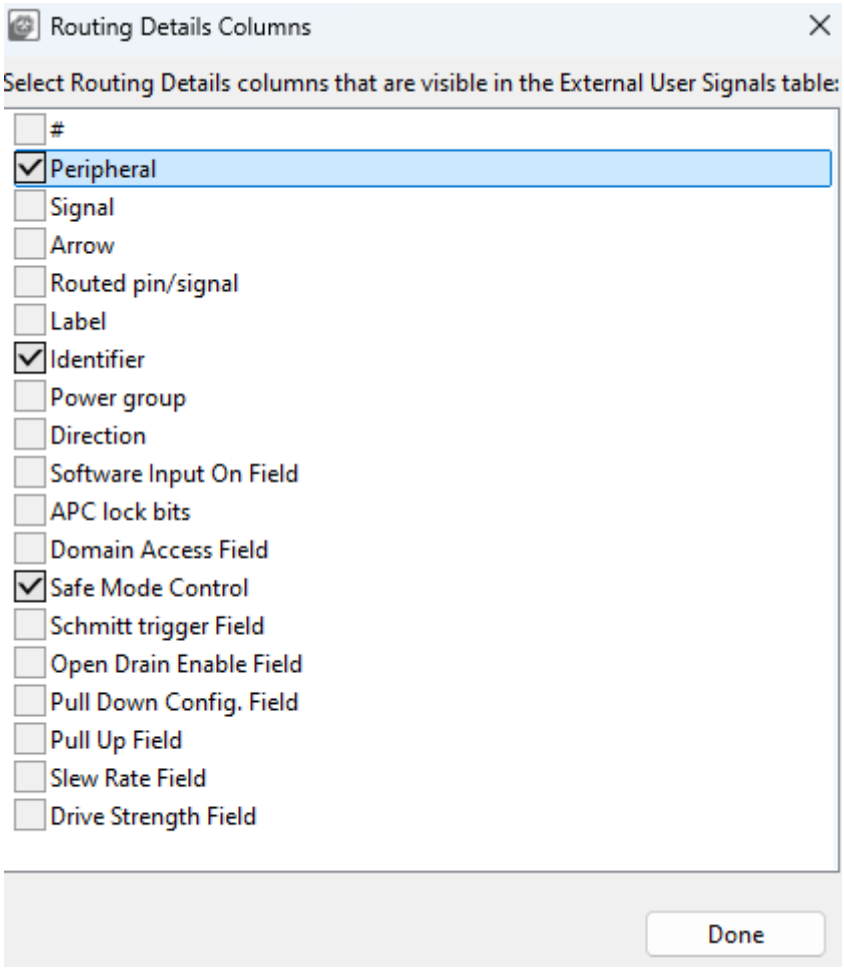


Figure 65. Routing Details columns dialog

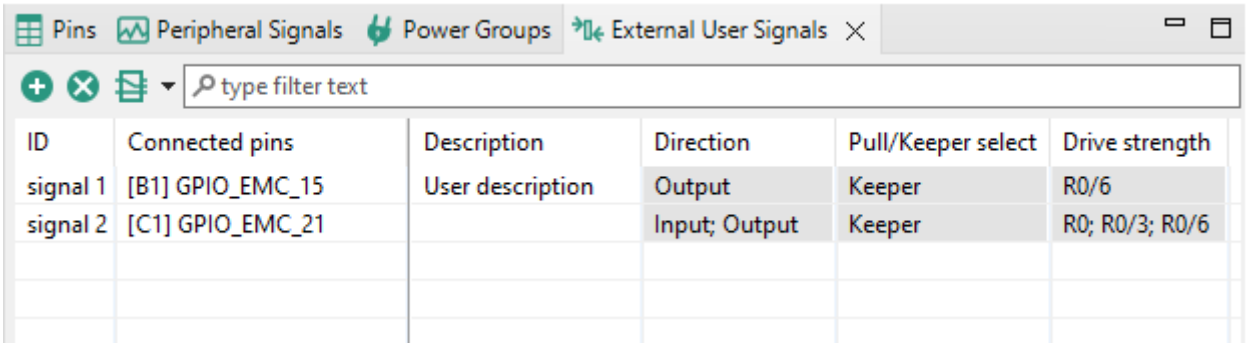


Figure 66. Routing Details table

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

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

3.3.8 Functions

Functions group a set of routed pins and generate code for the configuration in a function that the application can then call.

The tool allows you to create multiple functions to configure pin muxing.

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 peripherals for the selected function. Pins and peripherals for other functions appear with light transparency and cannot be configured until you switch to that function.

3.3.9 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.

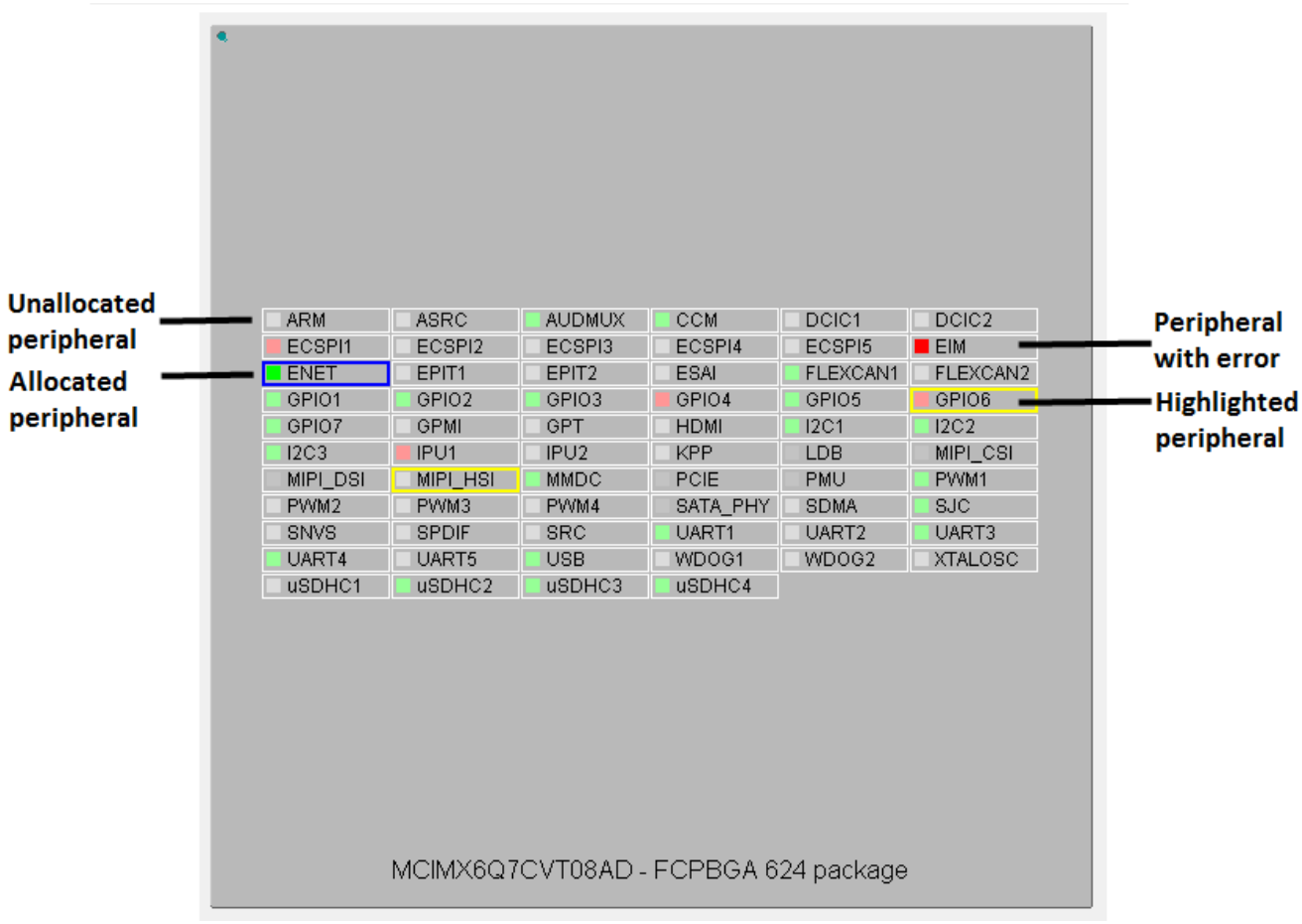


Figure 67. Highlighting and color coding

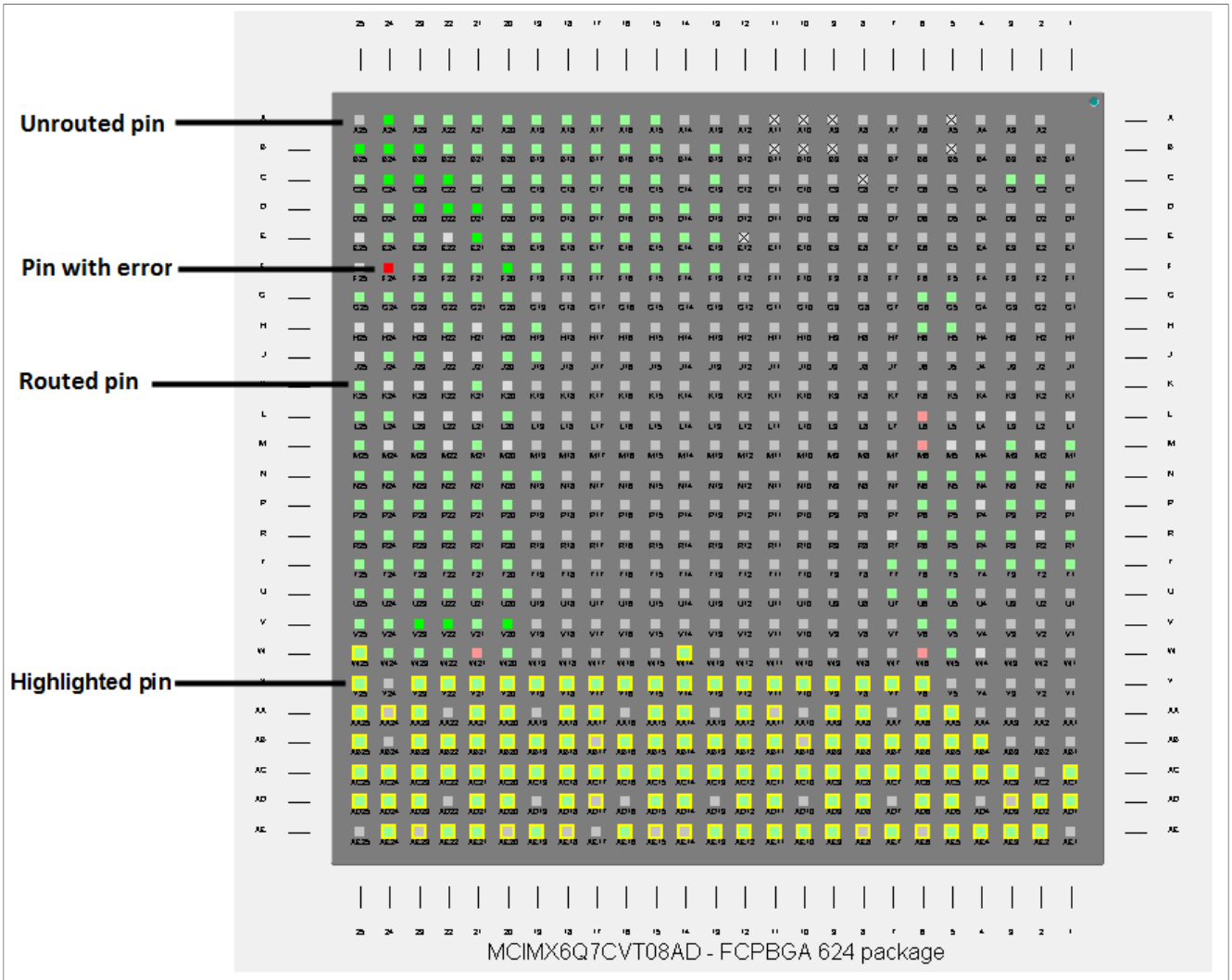


Figure 68. BGA package on pins side

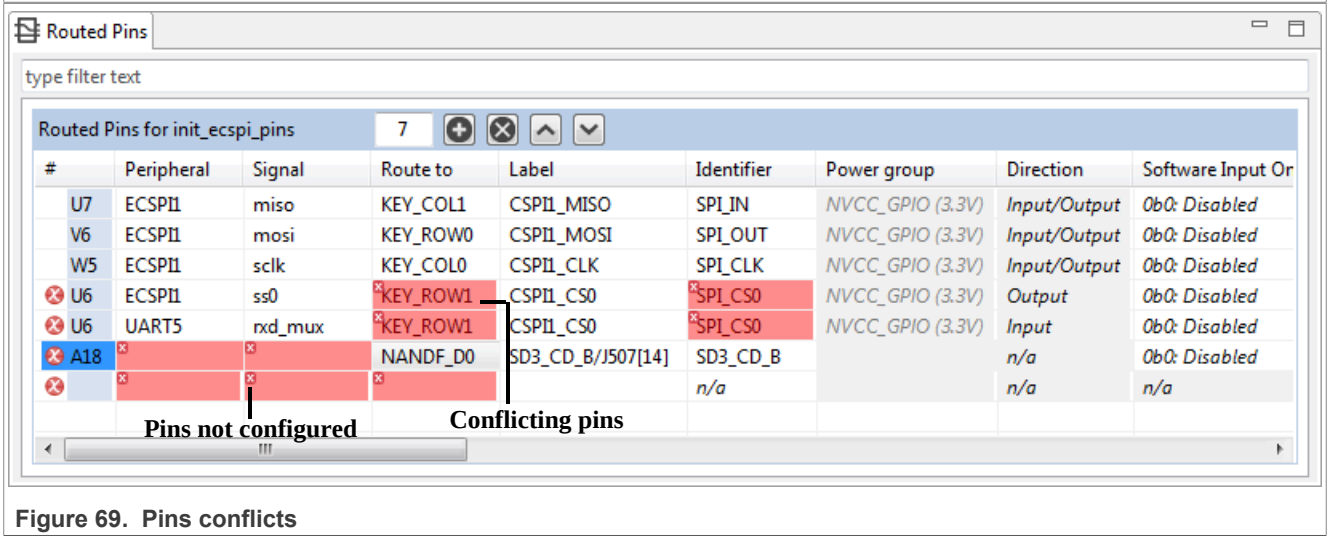


Figure 69. Pins conflicts

#	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 70. 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 behave as described above.

3.4 Errors and warnings

The Pins Tool checks for any conflict in the routing and also for errors in the configuration. The Pins Tool checks routing conflicts 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.

Routed Pins

type filter text

Routed Pins for BOARD_Init... 6

#	Peripheral	Signal	Route to	Label	Identifier	Direction	GPIO initial state	Mode	Slew rate	Invert	Open drain
3..	FLEXCOMM0	RXD_SDA_MOSI_DATA	FC0_RXD_SDA_MOSI_DATA	P17[17]/P24[1]/PIO1_5_GPIO...	n/a	Not Specified	n/a	Inactive	Standard	Disabled	Disabled
5	FLEXCOMM0	TXD_SCL_MISO_WS	FC0_TXD_SCL_MISO_WS	R80/P18[9]/LEDB/PWM_ARD	LED_RED	Not Specified	n/a	Inactive	Standard	Disabled	Disabled
1..	PMC	VREFINPUT_1	VDDA	VDDA_TARGET	n/a	Input	n/a	n/a	n/a	n/a	n/a
9..	USBFSH	USB_VDD	USB0_3V3	VDD_TARGET_3.3	n/a	Input	n/a	n/a	n/a	n/a	n/a
2..	CTIMER3	CAPTURE, 3	CT_INP10	U14[12]/SWO_TRGT	Not Specified	Input	n/a	Inactive	Standard	Disabled	Disabled
1..	CTIMER3	CAPTURE, 3	CT_INP4	P19[2]/P23[1]/ADC0_N	n/a	Input	n/a	Inactive	Standard	Disabled	Disabled

Figure 71. 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.

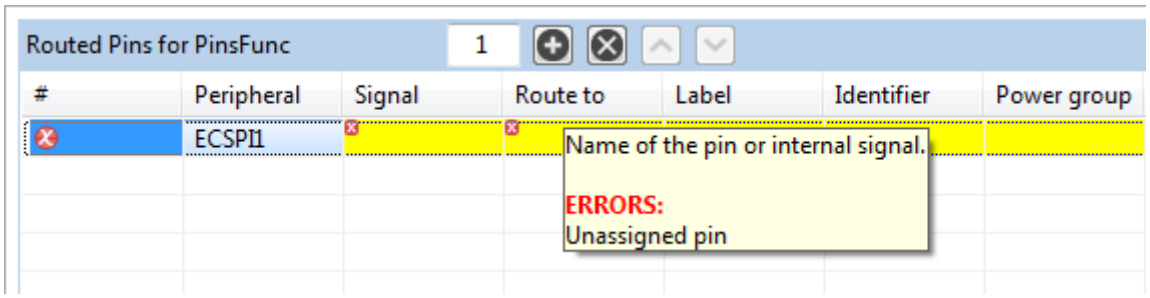


Figure 72. Incomplete routing

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.

3.4.2 Power groups voltage level conflicts

The Pins tool provides information about possible voltage level conflicts when the peripheral signals routed pins are configured from different power groups and the power groups have different voltage level value set in **Power groups** view.

In case of a potential voltage level conflict, a warning is displayed - a useful feature for hardware board designers.

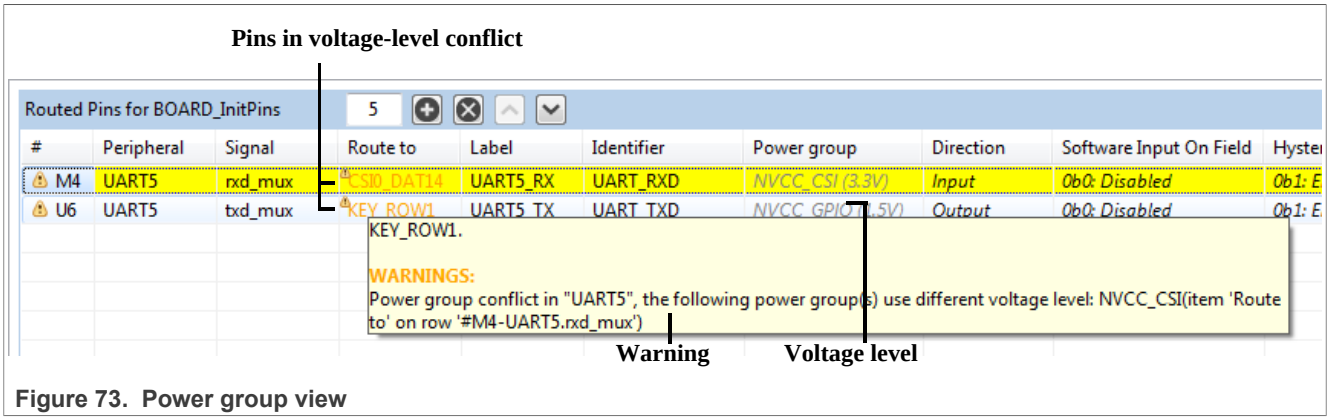


Figure 73. Power group view

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 in 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 **Set viewing style for source differences**. You can also disable highlighting altogether from the same dropdown menu. The Copy, Search, Zoom-in, Zoom-out, and Export source features are available in the **Code Preview** view. You can also invoke the search with 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.

Note: Code Preview view contains the **Export** button and possibility of exporting sources - link to export wizard - **Pins Tool > Export Source Files** option-allowing export sources per used cores in multi-core enabled configuration.

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 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. If you change the pin where the signal is routed, the application still refers to the proper pin.

For example, when *LED_RED* is specified as an identifier of a pin routed to *PTB22*, the following defines are generated in 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 the function group prefix and the pin identifier. For more details, see the Functional groups and Labels and identifiers sections.

To write to this GPIO pin in an application using the SDK driver (fsl_gpio.h), use the following code, which refers 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 the after-reset state. This option is specific to each **Functional group**, allowing you 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, you can create a new functional group that routes default signals to pins and internal signals. The functionality is available in **Pins > Create Default Routing**. There, you 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 is set. The electrical properties of pins are set to their after-reset state.

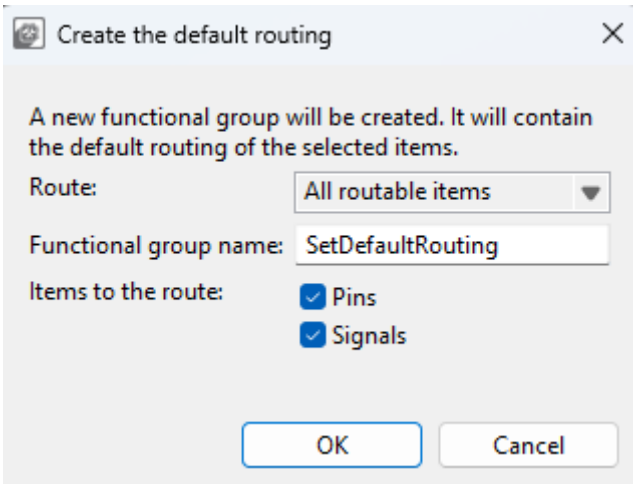


Figure 74. Create the Default Routing

4 Clocks Tool

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 elements settings and calculate the resulting output clock frequencies.
- 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.
- Find clock elements settings that fulfill given requirements for outputs.
- Register values define generation of C.

4.2 User interface overview

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

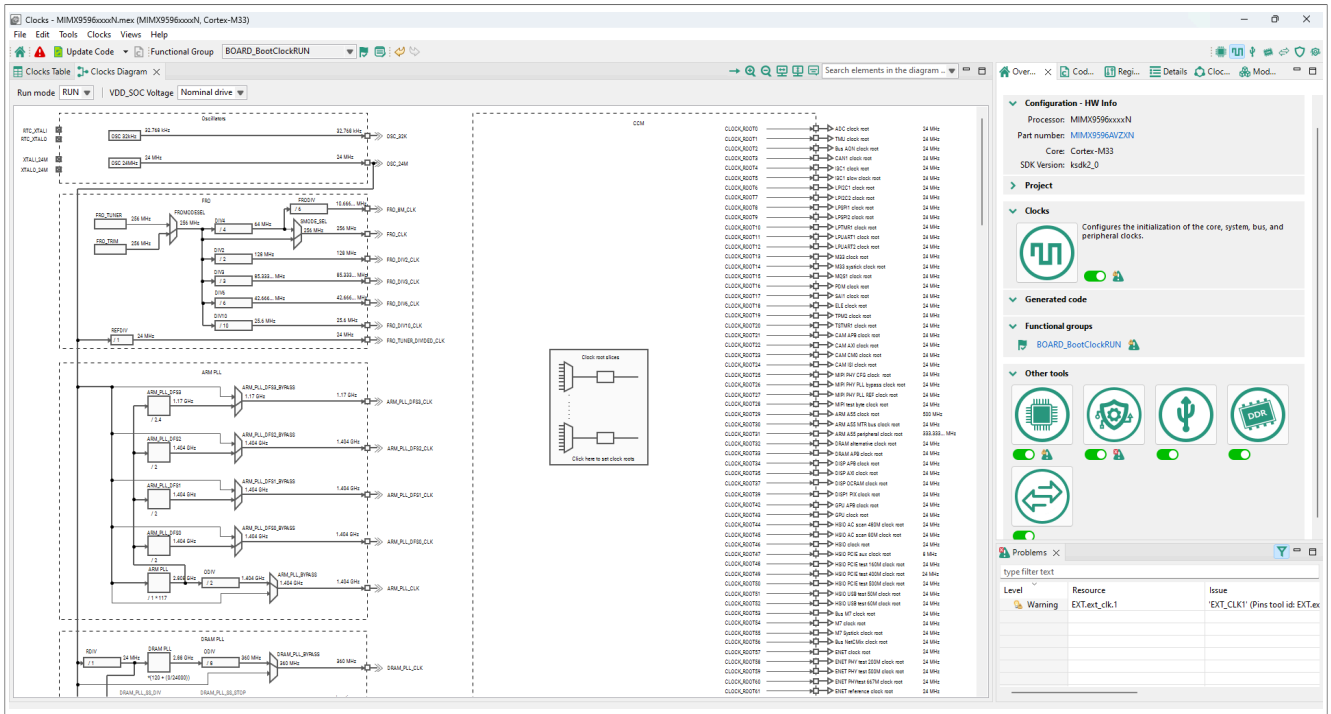


Figure 75. User interface

4.2.1 Details view

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

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

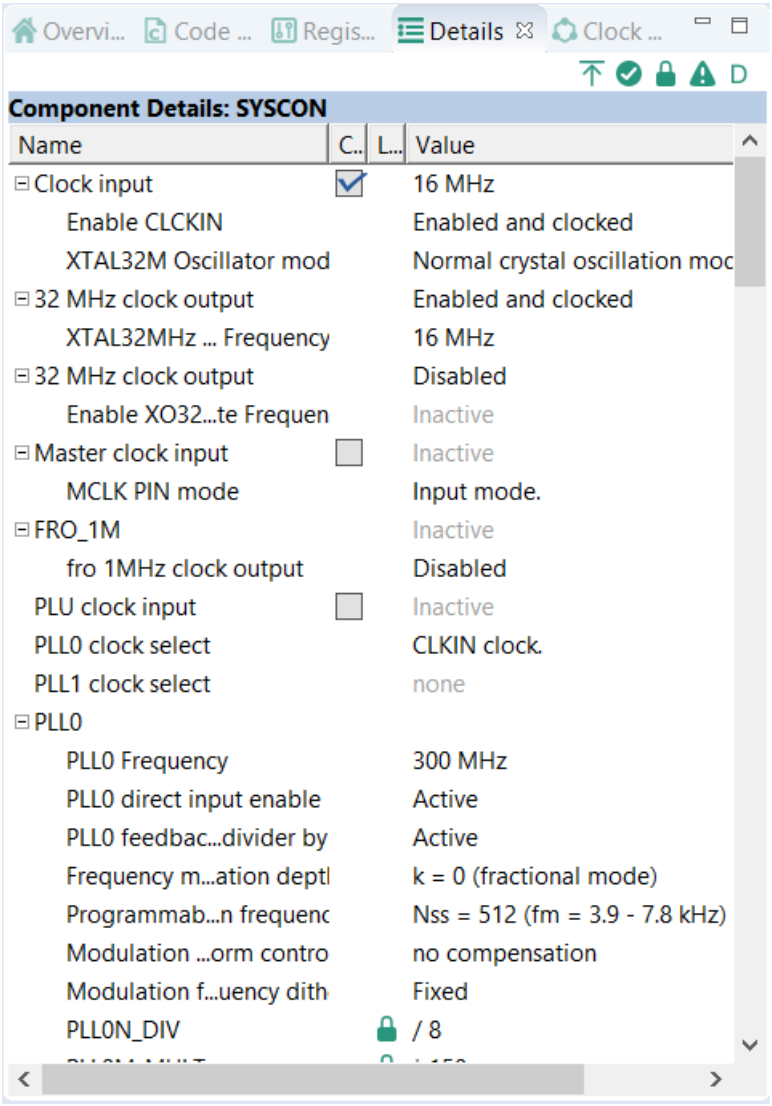


Figure 76. Details view

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 and highlight it 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.

4.2.2 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.

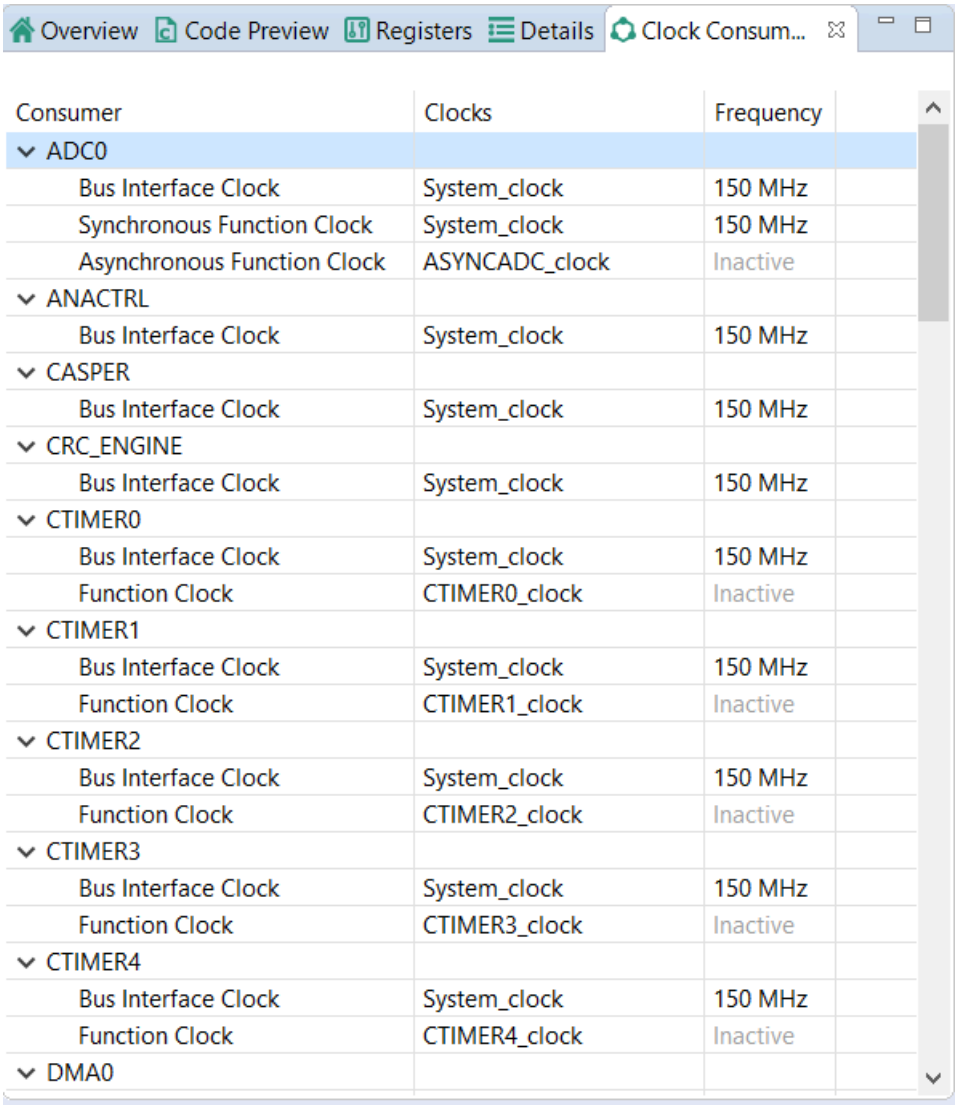


Figure 77. Clock Consumers view

4.2.3 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.

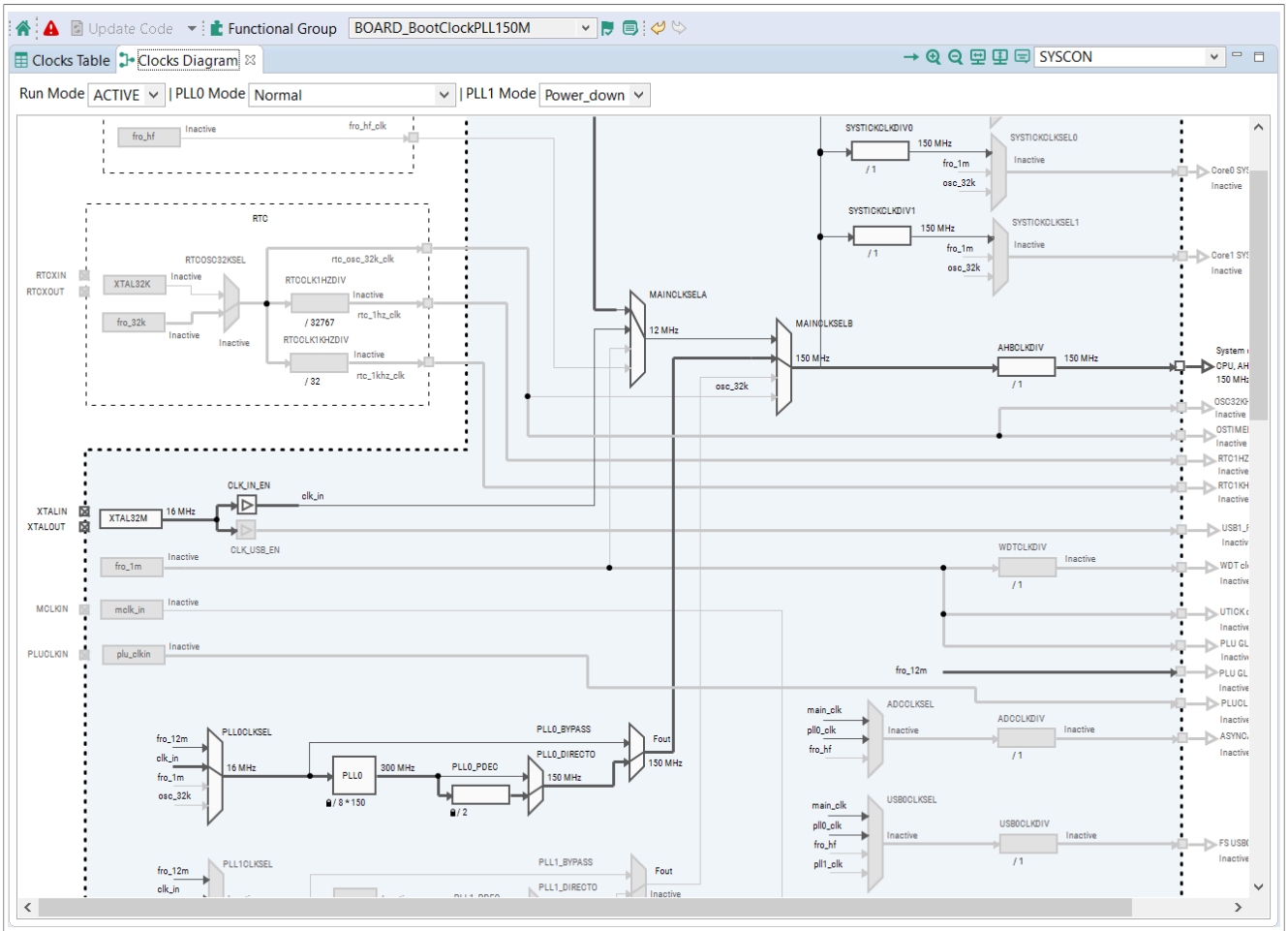


Figure 78. Clocks diagram

4.2.3.1 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.
- **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.

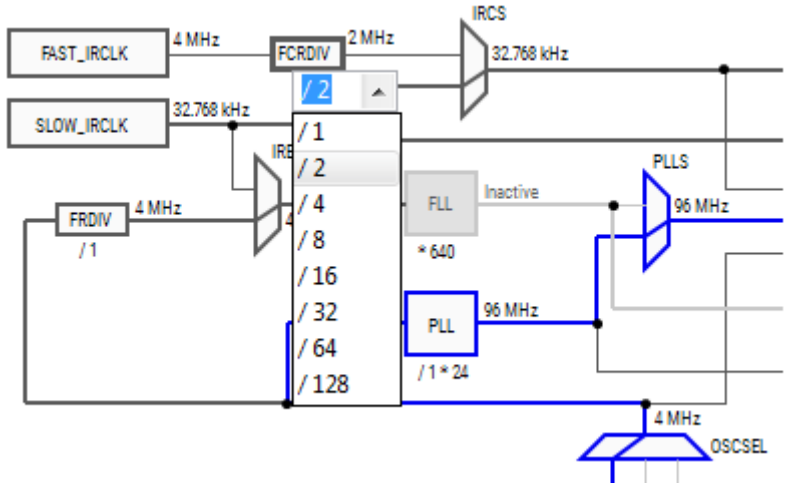


Figure 79. Clocks mouse actions in diagram

- **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.

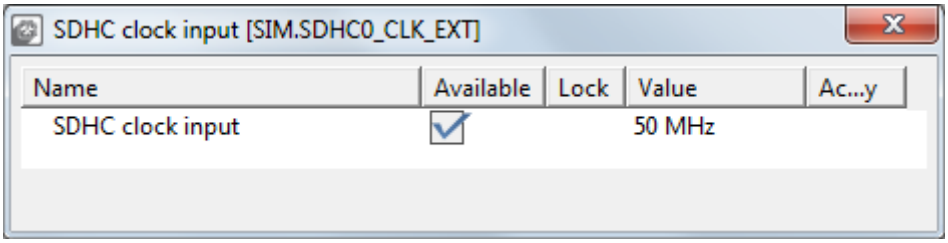


Figure 80. Floating view

4.2.3.2 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.

4.2.3.3 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 81. Clock source

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

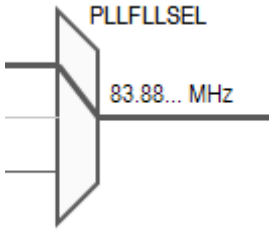


Figure 82. Clocks selector

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

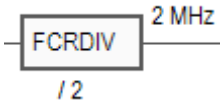


Figure 83. Prescaler

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

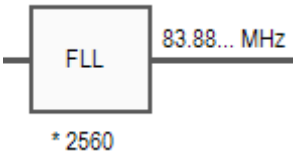


Figure 84. Frequency Locked Loop

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

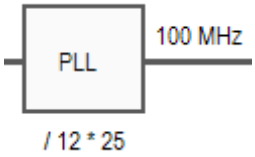


Figure 85. Phase Locked Loop

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

Figure 86. Clock gate

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

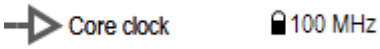


Figure 87. Clock output

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

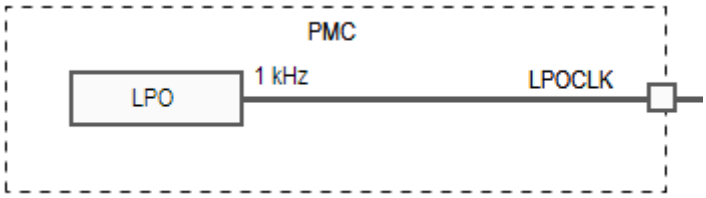


Figure 88. Clock component

- **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.

4.3 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.

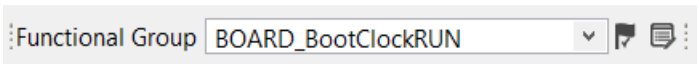


Figure 89. Default clock configuration

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.

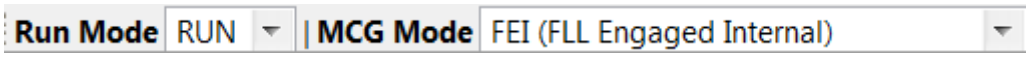


Figure 90. Global settings

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.

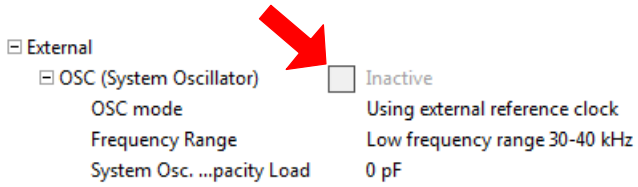


Figure 91. External clock source configuration

Note: Some clock sources remain inactive even though the checkbox is checked. This is because clock source functionality depends on other settings, such as power mode or additional enable/disable options. Hover over 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 setting rows in the settings tables (clock sources, output, details, or individual).

Table 18. 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 a setting can be locked, it is automatically locked when you change the value. To add/remove the lock manually, use the pop-up menu

Table 18. Setting states and markers...*continued*

State/Style/Marker	Icon	Description
		command **Lock/Unlock**. **Note:** The clock element settings that cannot be automatically adjusted by the tool keep their value as is and do not allow locking. 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 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 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 92. Setting the core clock frequency

If the tool has reached the required frequency, it appears locked and is displayed as follows:

Figure 93. Tool attains the required frequency

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

Figure 94. Tool encounters problem

The frequency value in square brackets [ ] indicates the value the tool uses in calculations, not the requested value.

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 the lock on the frequency, enabling the tool to change any valid value that satisfies all other requirements, limits, and constraints.
- **Find Near Valid Value** - Finds a valid frequency near the specified value when the tool cannot reach the requested frequency.
- **Advanced resolver for {Clock output}** - Invokes more advanced search for the valid settings that fulfills the requirements. It may take significant time so a progress dialog is shown. If the resolver is not successful, the user is informed about it. 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.

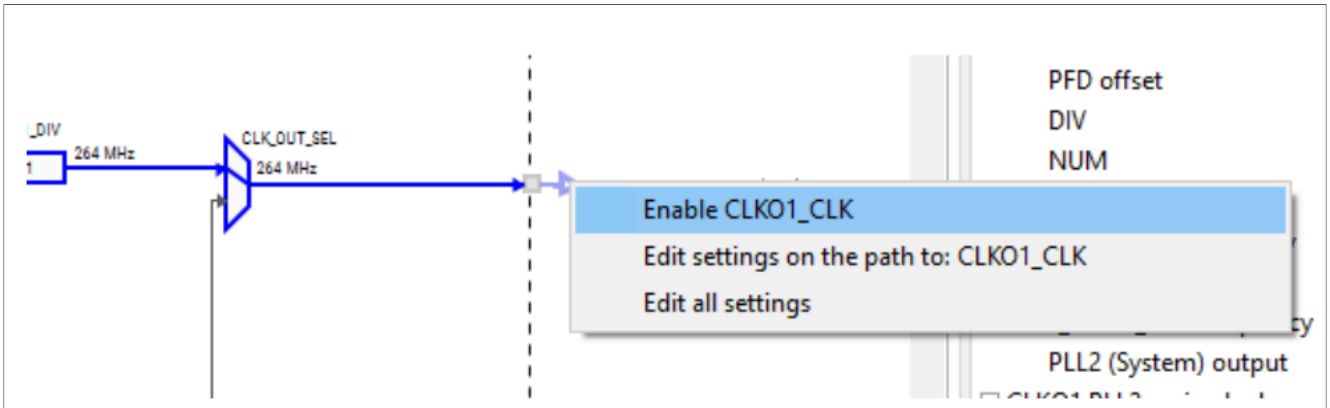


Figure 95. Pop-up commands for outputs in the clocks table view

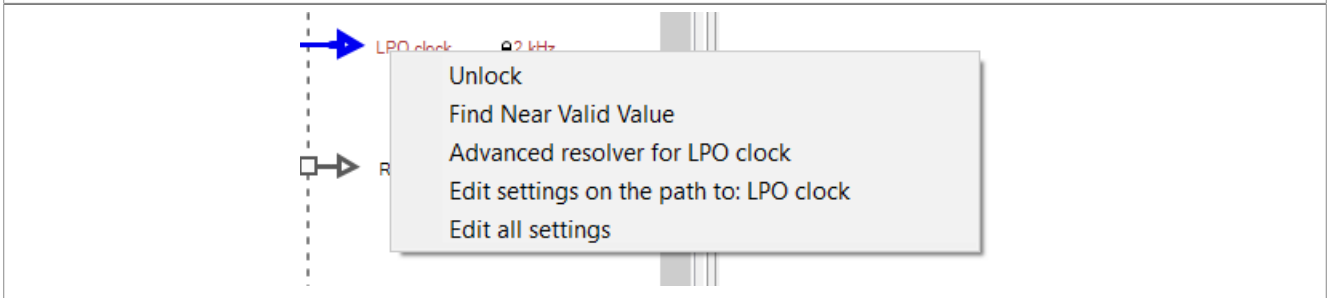


Figure 96. Pop-up commands for outputs in the clocks diagram

4.7.1.1 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 can be individually adjusted by clicking the value.

Figure 97. 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 using the signal from the same clock source. They identify dependencies and influences when the clock source or elements on a shared clock path change.

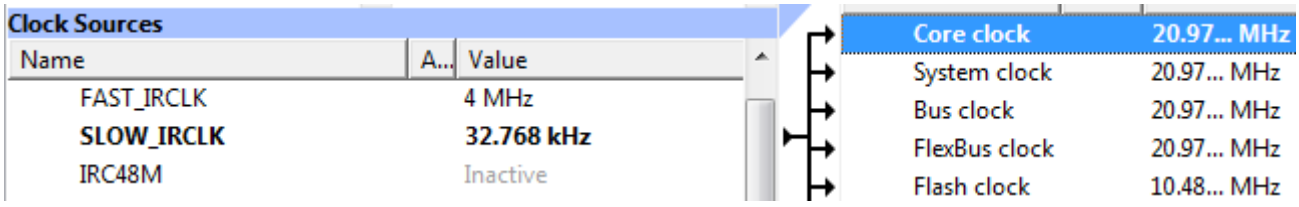


Figure 98. Dependency arrows

4.9 Modular clock initialization

Some MCUs support clock initialization per a clock module. In this case, there is a generated function (`InitClockModule`) that can be used to initialize clock modules separately. This function is also used by the generated initialization function for the current functional group.

The usage of the module initialization function can be configured independently for each module via the **Modular Initialization** view.

4.9.1 Modular Initialization view

This view is displayed only for MCUs that support modular initialization, as shown in Modular Initialization view.

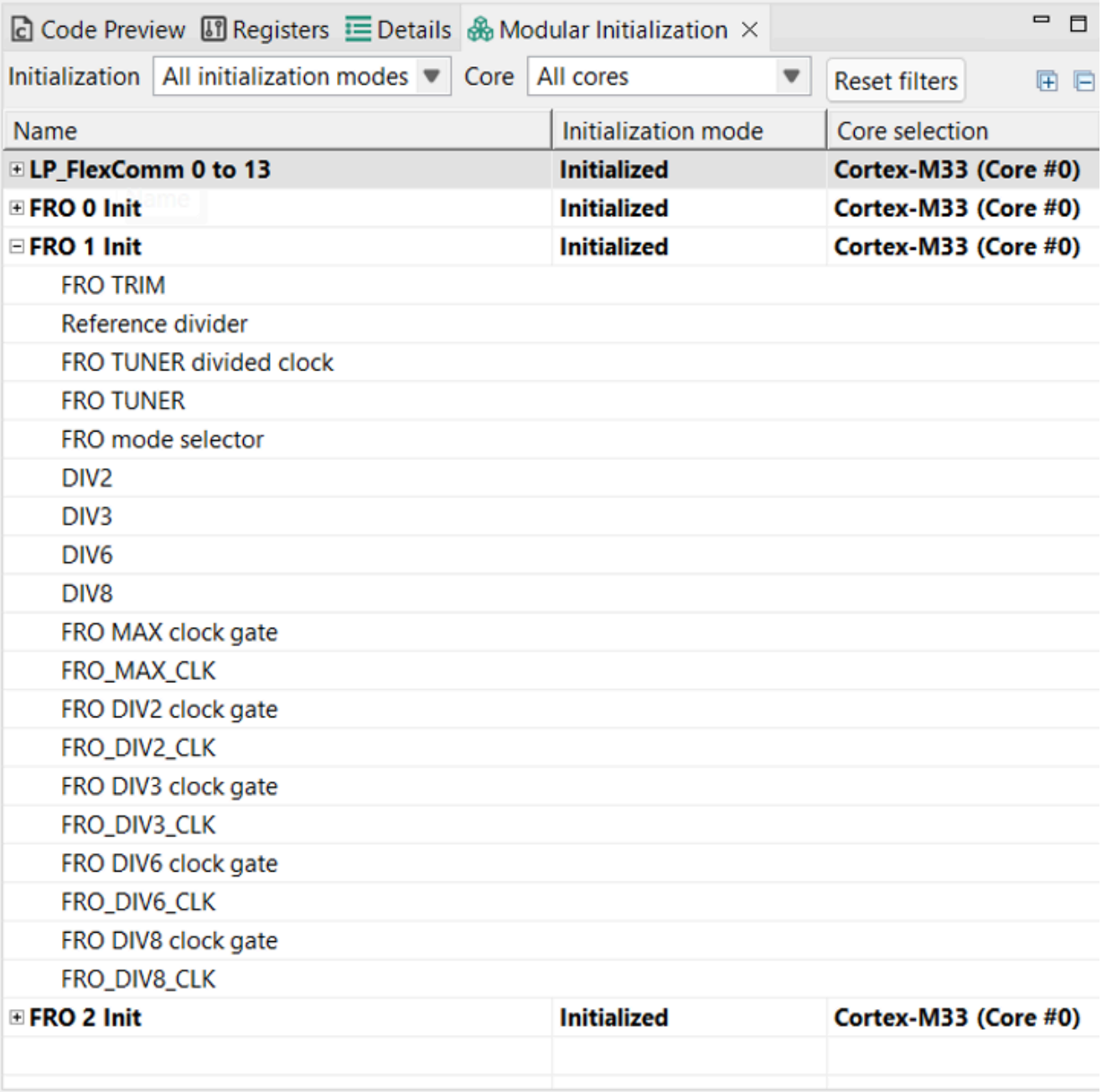


Figure 99. Modular Initialization view

The view contains a table with an expandable list of all clock initialization modules on the MCU in the current functional group. When you expand the modules using the expand/collapse icon on the left, a list of all clock elements that are parts of the module appears. Double-click to see the **Details** view, where the double-clicked element is focused and selected. This selection propagates to the clock diagram and other views.

The table has the following columns:

- **Name** is the name of the clock module.
- **Initialization mode** shows how the initialization of the clock module is handled. The following values are available:
  - **Initialized**: the code of the initialization function for the module is generated and the module initialization function is called from the functional group initialization function.
  - **Runtime**: the code of the module is generated, but it is not called from the initialization function for this functional group. Call the module initialization function (`InitClockModule`) for this module if needed during the runtime.
  - **Custom**: the initialization code for that module is not generated. The user must handle its initialization completely.
- **Core selection** selects the core that is to handle the initialization of the clock module. As the clocks tool generates independent files for each code, this influences in which file the code will be generated.

4.9.2 Details view

For supported MCUs, the **Details** view shows an additional Init Module column. See Details view extended with Init Module column for details.

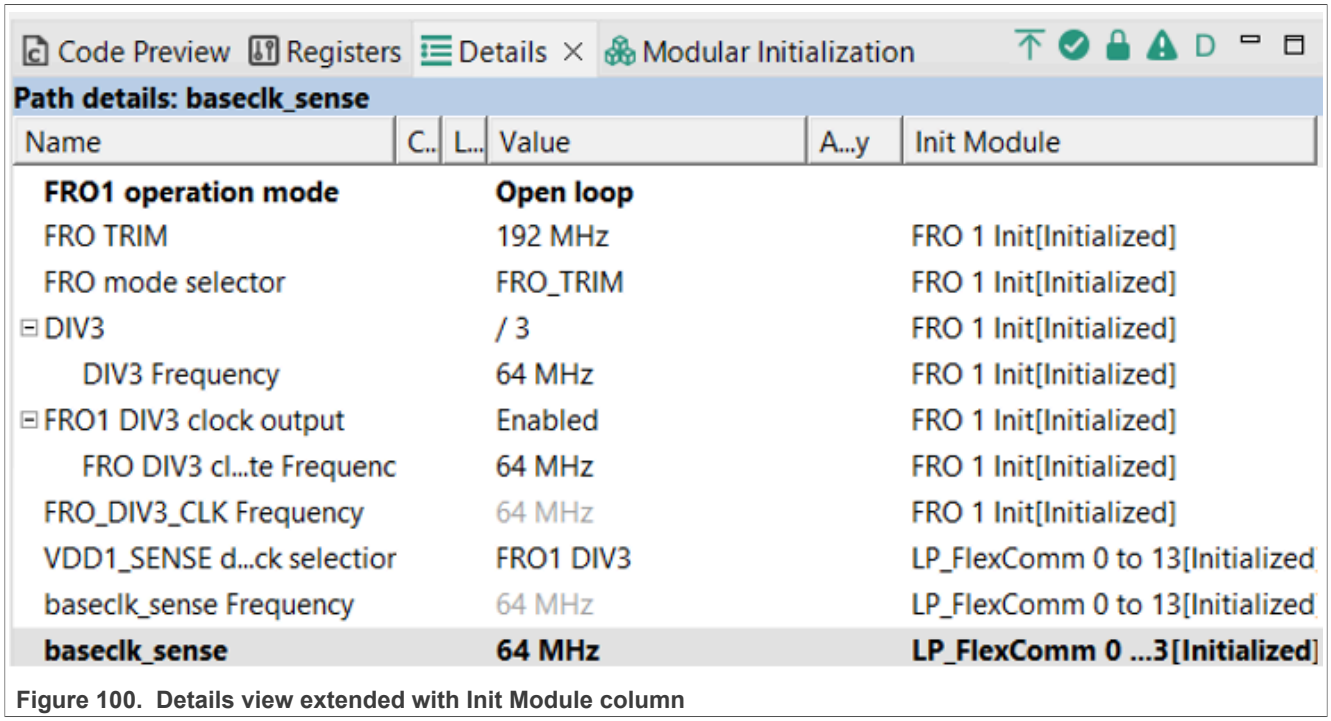


Figure 100. Details view extended with Init Module column

The Init Module column displays the clock initialization module of the clock element in the corresponding row if the element belongs to a module. The module name is followed by the value of the initialization mode in square brackets. When the clock output is selected, all initialization modules that affect the clock element on the path to the corresponding clock output are visible.

Double-click on an initialization module to switch to the **Modular Initialization** view where the module is selected and expanded.

4.9.3 Clock diagram

To alert the user of the potential of non-initialized clock output, the tool shows a yellow marker next to the affected clock output in the Clock Diagram. This marker highlights the clock elements on the path to the marked clock output belonging to a clock module in a non-default initialization mode (runtime or custom). The tooltip

of that clock output displays information about the initialization modules that are in the nondefault initialization state. This feature can be seen in Yellow marker in clock diagram informing about nondefault initialization mode.

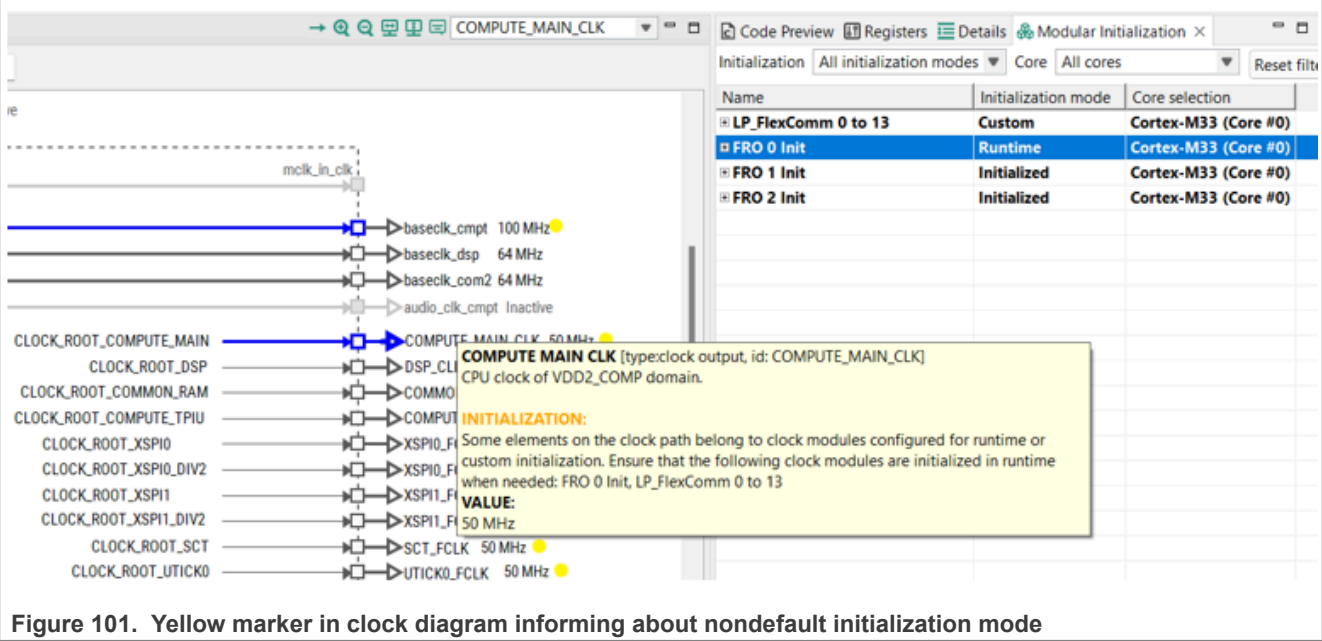


Figure 101. Yellow marker in clock diagram informing about nondefault initialization mode

4.10 Troubleshooting problems

Problems or conflicts may 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 one or more locked frequency constraints exist for which the tool cannot find a valid setting to 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 only want to enable a clock output, use the pop-up menu command **Enable**, which automatically finds settings that provide a valid frequency on the clock output.
4. If the required clock output value cannot be satisfied, use the pop-up menu command “Advanced resolver for {clock output}”.
5. If you can accept 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. Remove locks by selecting **Clocks > Unlock All Settings.** If too many changes are required and conflicting, reset the model to the default values and start from the beginning. To reset, select **Clocks > Reset to processor defaults**.

4.11 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 appears 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 102. Code Preview

4.11.1 Working with the code

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

To transfer the code into your project, 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 tool can generate the defines with these values into a new file registers.h. This option is enabled by default (**Edit->Configuration Preferences**). For more information, see section Configuration Preferences.

5 Peripherals Tool

This chapter provides general information about the Peripherals Tool.

5.1 Features

The Peripherals tool features:

- Configuration of initialization for SDK drivers
- User-friendly user interface allowing to inspect and modify settings
- Smart configuration component selection along the SDK drivers used in toolchain project
- Instant validation of basic constraints and problems in configuration
- Generation of initialization source code using SDK function calls
- Multiple functional-group support for initialization alternatives
- Configuration problems are shown in the Problems view and marked with decorators in other views
- Integration in 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 19. 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).

Table 19. Terms and Definitions...*continued*

Term	Definition
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 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** to use and configure individual settings.
   Note: The component mode selection may affect the appearance of some settings. Therefore, always select the mode first.
5. 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.
6. Click the **Update Code** button in the toolbar. Alternatively, export the source code by selecting **File > Export…** from the **Menu bar**.
   Note: To export the source code, also click the **Export** button in the **Code Preview** view.
7. Save settings in MEX format (used for all settings of all tools) by selecting **File > Save** from the **Menu bar**.

5.4 User interface overview

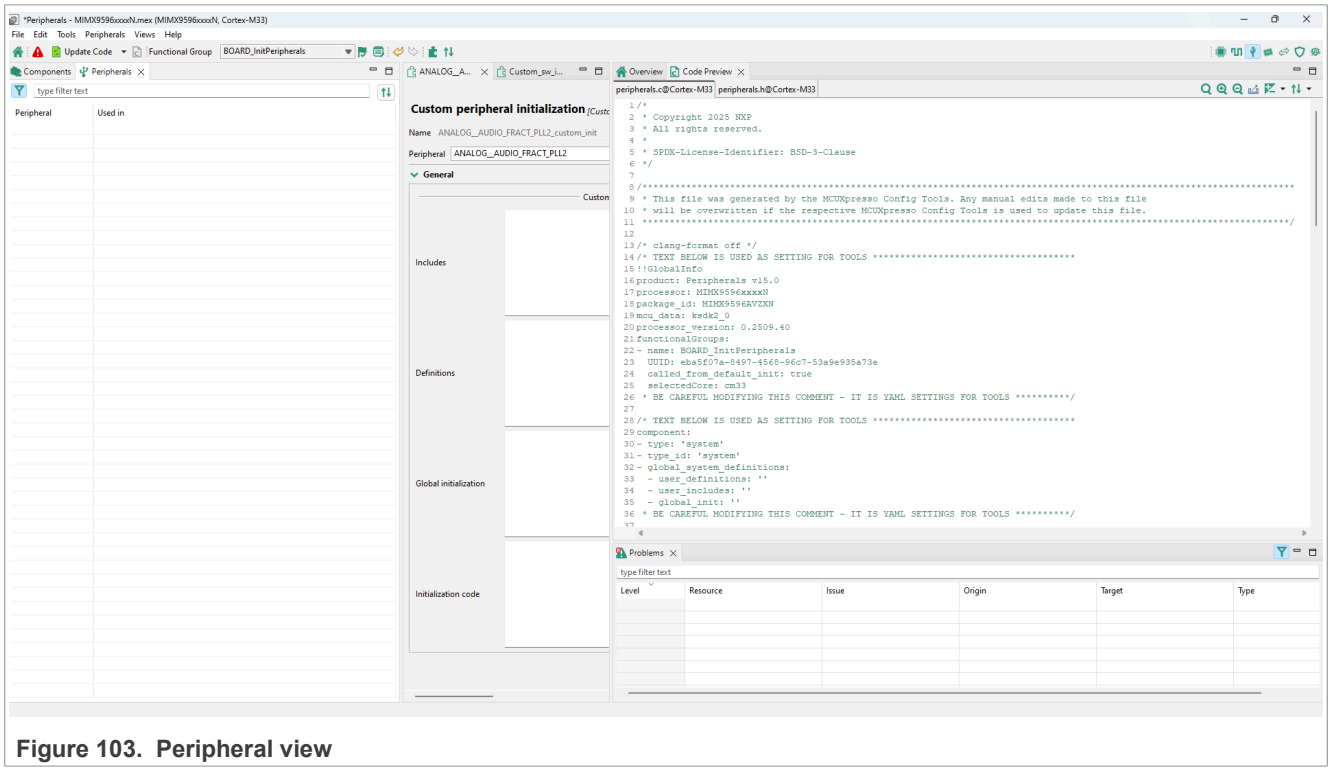


Figure 103. Peripheral view

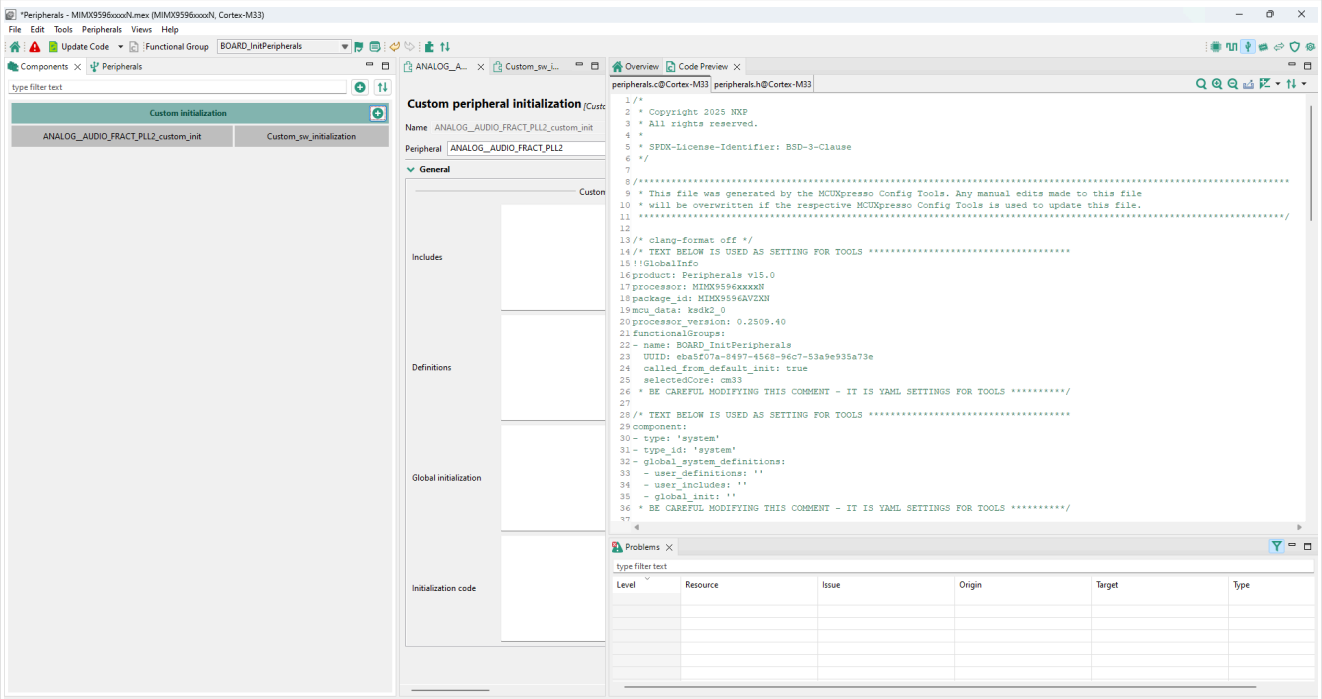


Figure 104. Components view

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:

Table 20. Common 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 chapter.

5.4.1.1 Initialization order dialog

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

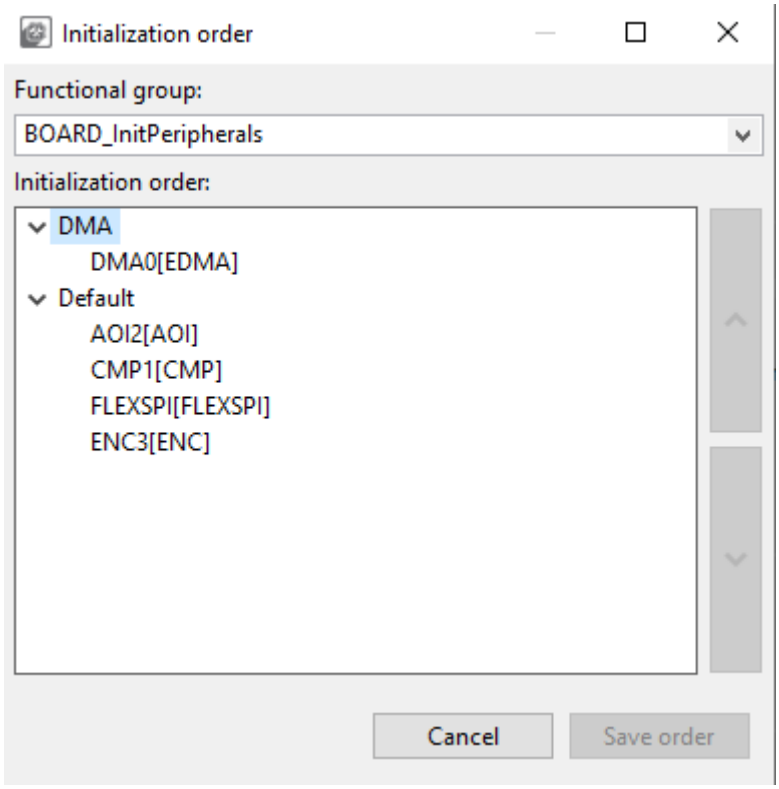


Figure 105. Initialization order dialog

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.

Table 21. Components view

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.

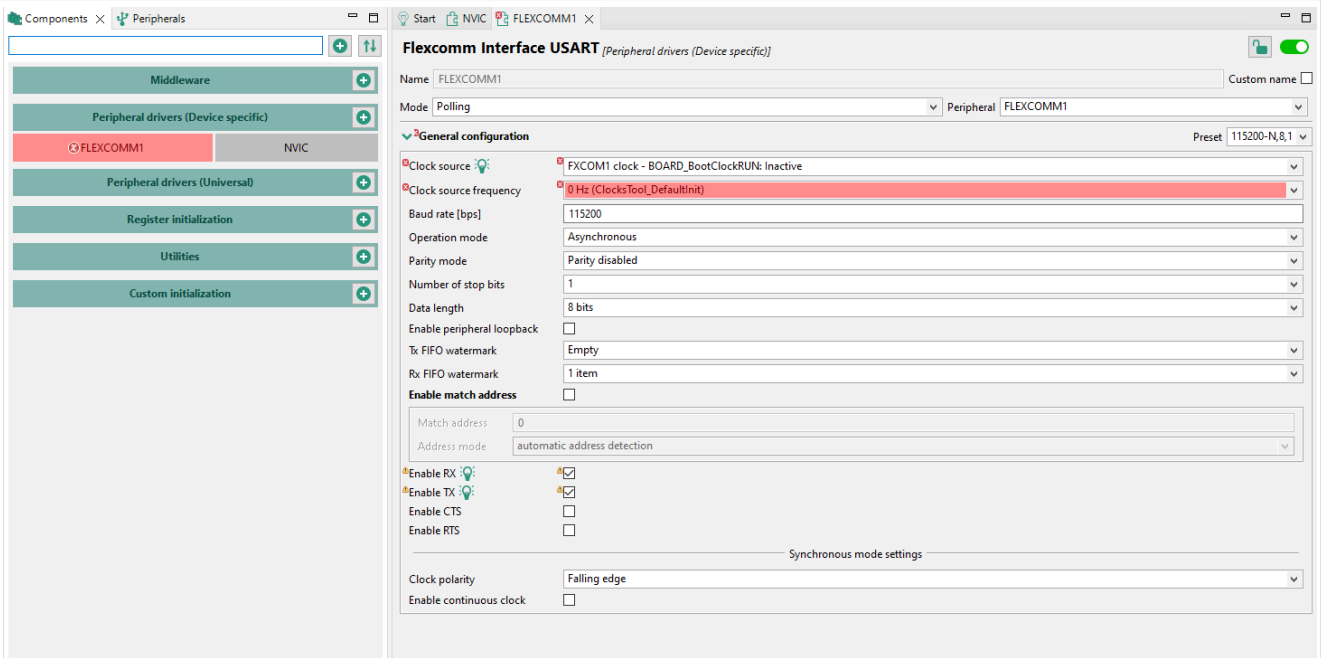


Figure 106. Components view

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

Table 22. 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 **Middleware/Peripheral drivers/Other** categories to add new components in them directly.
Filter configuration components by name	Type a text string to filter configuration component names in the search bar.

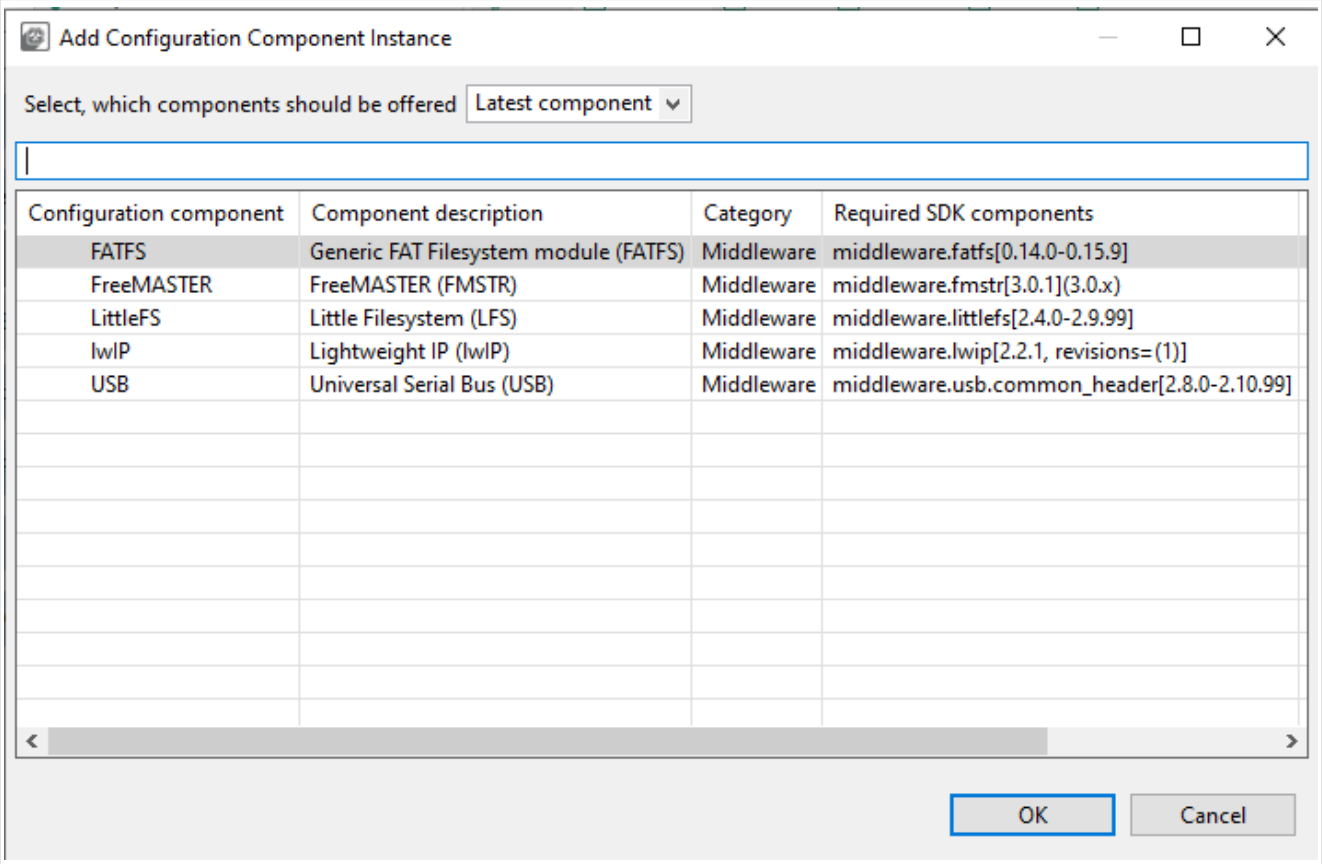


Figure 107. Add component dialog

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

Table 23. Shortcut menu

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 has no 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 23. Shortcut menu...*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. The first column contains the 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 generated code. The name of the component instance 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.

Table 24. Shortcut menu

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 lets you select which instance 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.
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 are shown 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 saved 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 you hover the mouse cursor over 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.

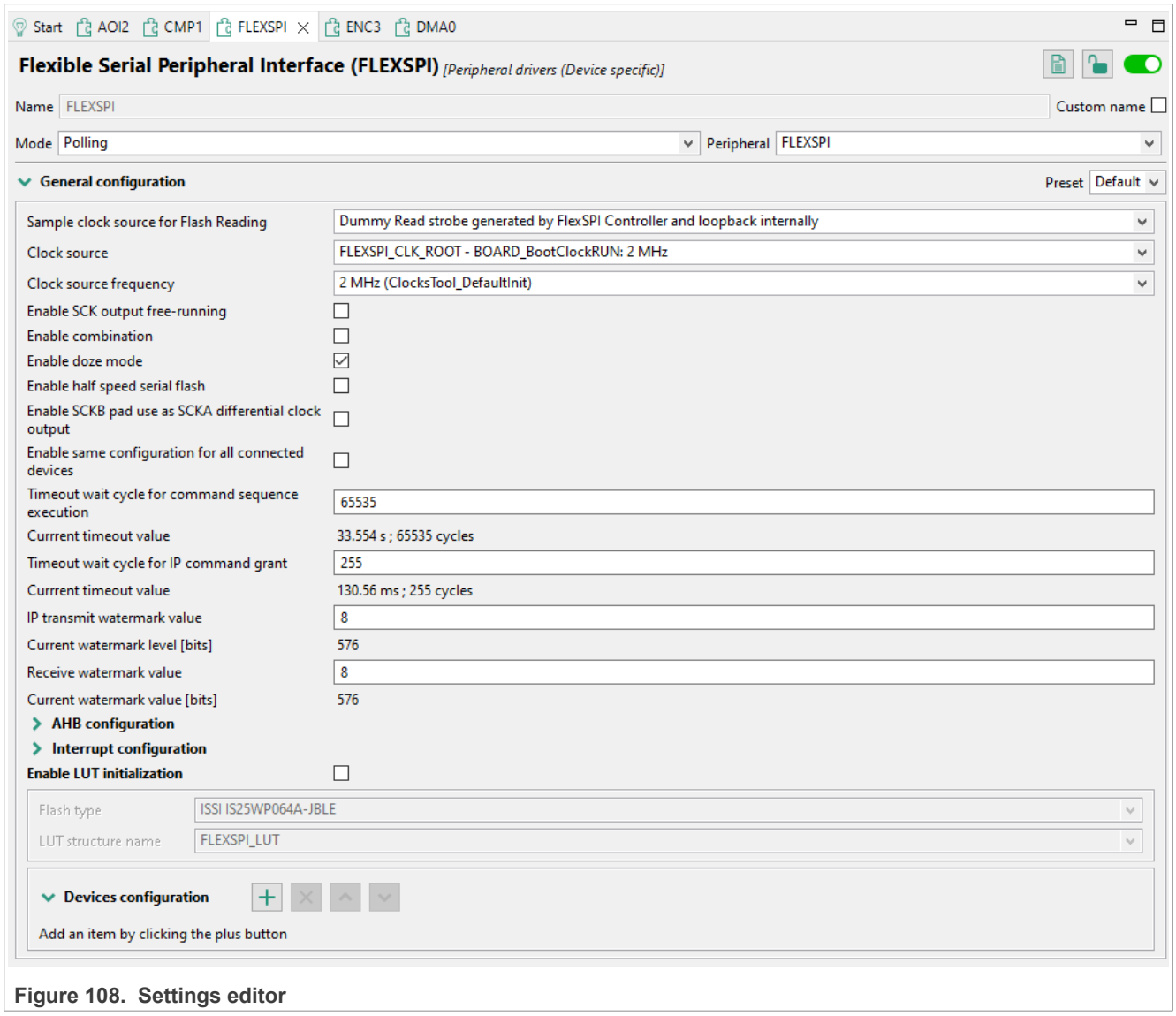


Figure 108. Settings editor

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 109. Settings Editor header

Table 25. 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 is still generated.
Enable/disable component instance switch	Use the switch to enable or disable the selected component instance. Disabling the instance does not remove it from the tool configuration, but prevents 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. Use these presets to quickly apply the desired combination of settings or return to the default state.

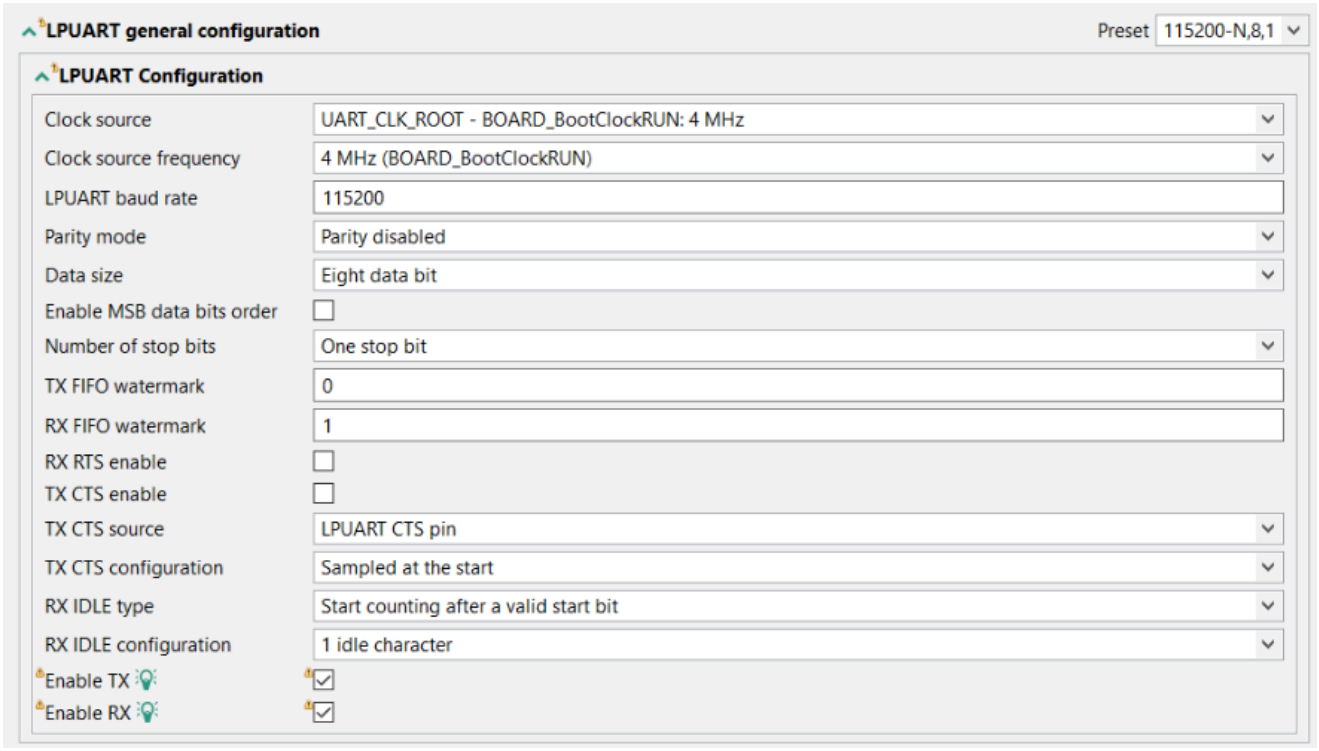


Figure 110. Quick selection example

5.4.4.3 Settings

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

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

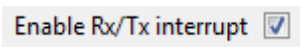


Figure 111. Boolean setting example

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

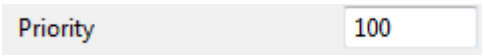


Figure 112. Integer/Float setting example

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

Figure 113. String setting example

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

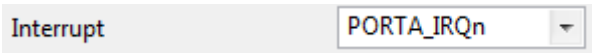


Figure 114. Enumeration setting example

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

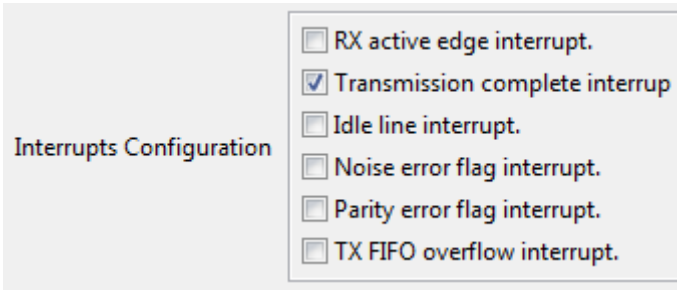


Figure 115. Set setting example

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

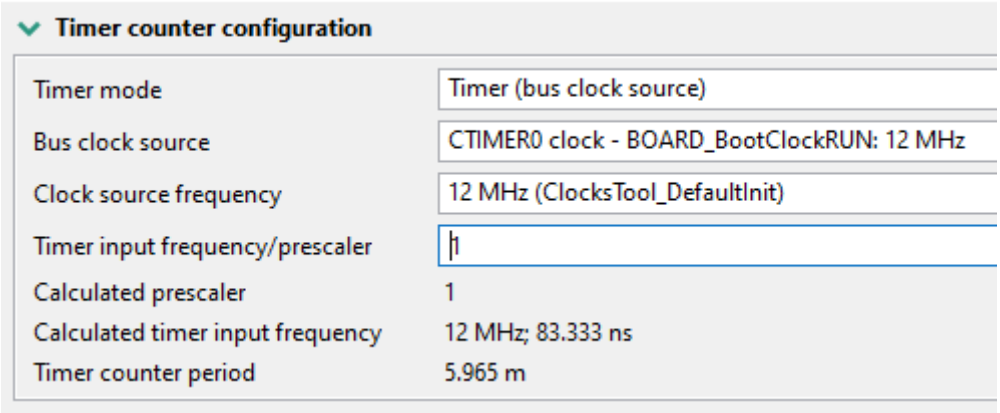


Figure 116. Structure setting example

- **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.

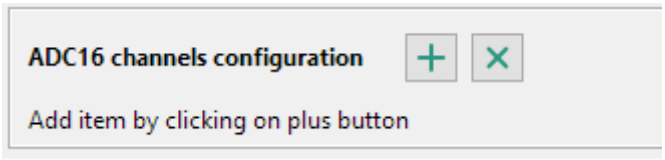


Figure 117. Array

The '+' button adds a new item at the end of the 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.

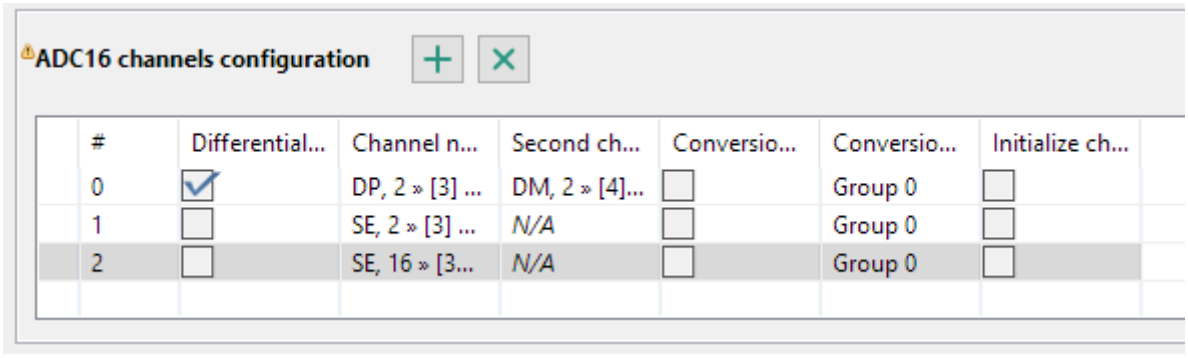


Figure 118. Array setting example

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

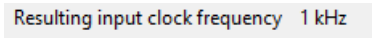


Figure 119. Info setting example

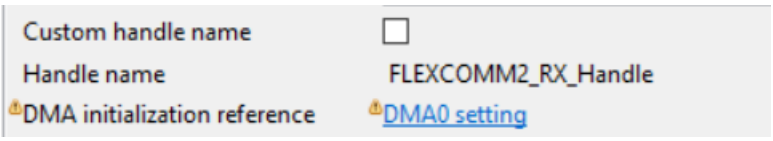


Figure 120. Info setting as link example

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

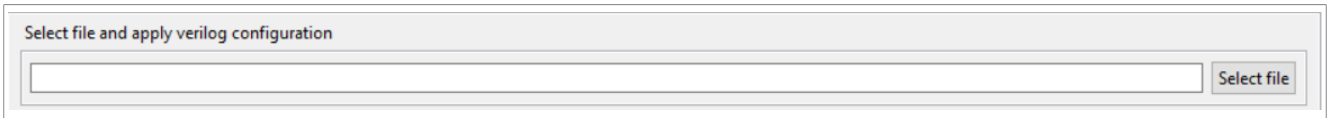


Figure 121. File setting

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.

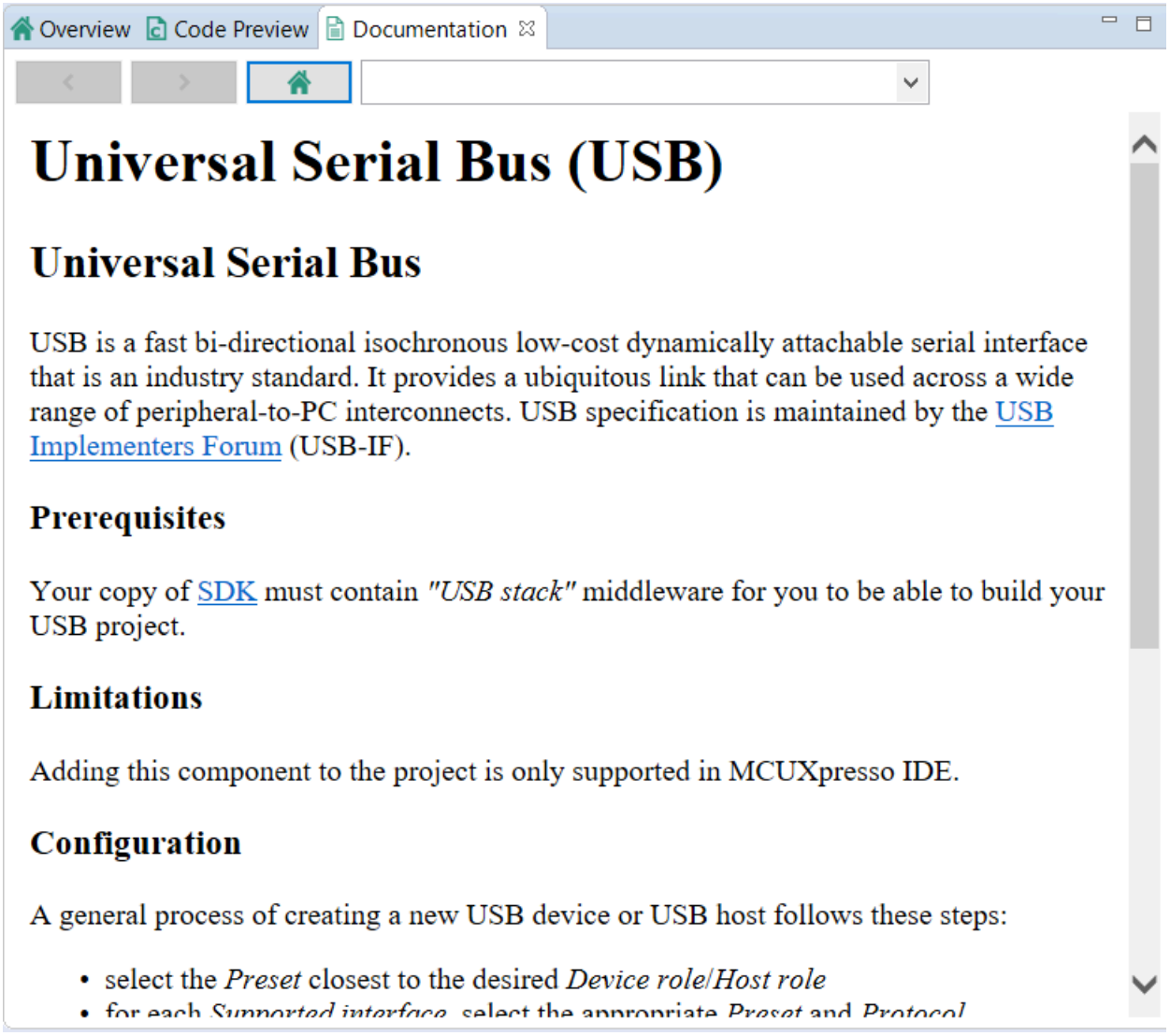


Figure 122. Documentation view

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 the 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:

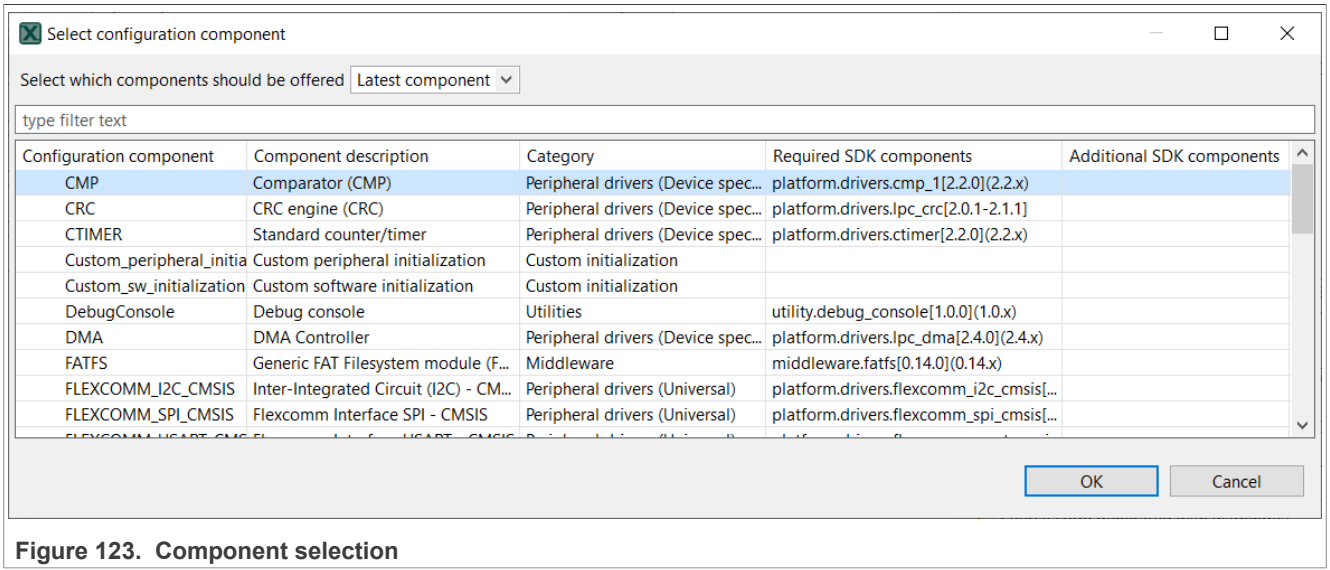


Figure 123. Component selection

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

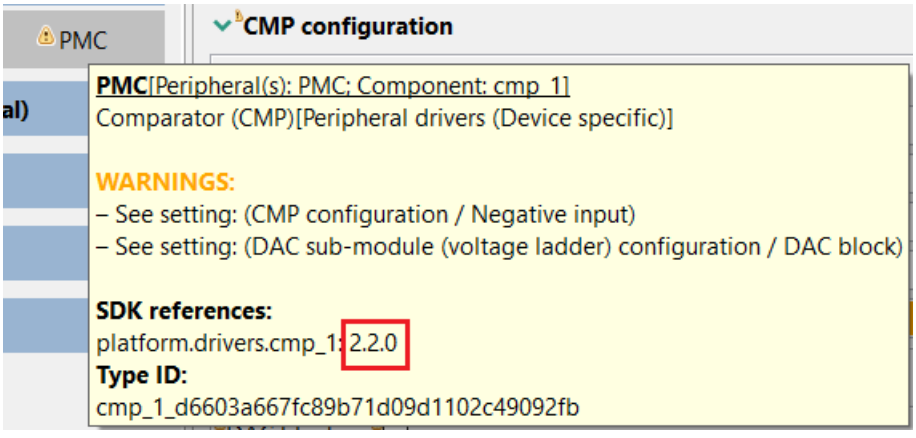


Figure 124. Tooltip in the Component view

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. The migration can also be launched manually in the peripherals tool using the menu **Peripherals > Migrate to other component versions.**

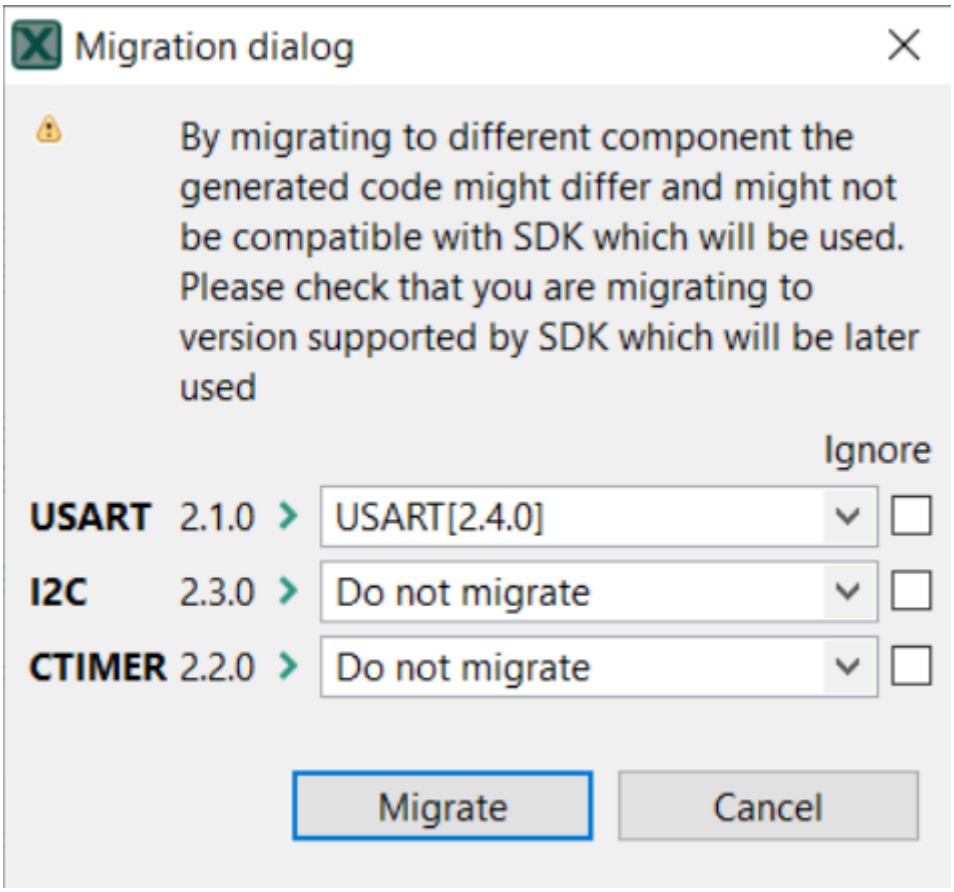


Figure 125. Migration dialog

Each row in the dialog corresponds to one configuration component that can be migrated to another version. The dialog above displays the current version specified in the current configuration component and a combo box allowing you to select 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 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 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.

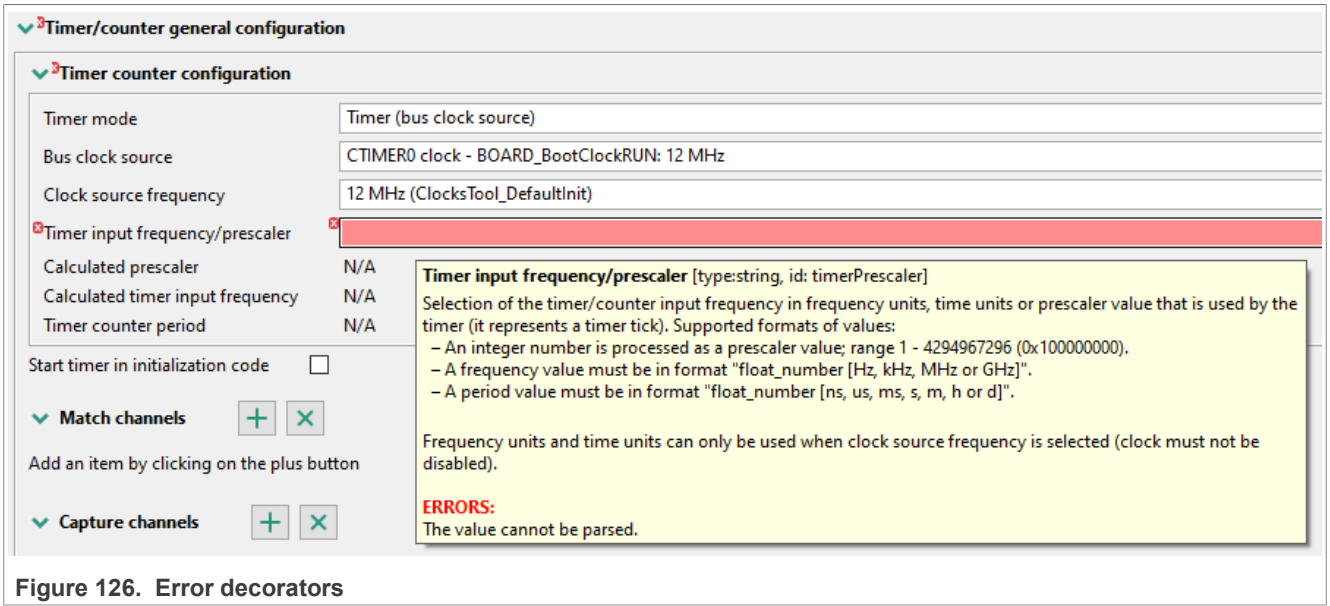


Figure 126. Error decorators

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

Figure 127. Quick fix 1

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

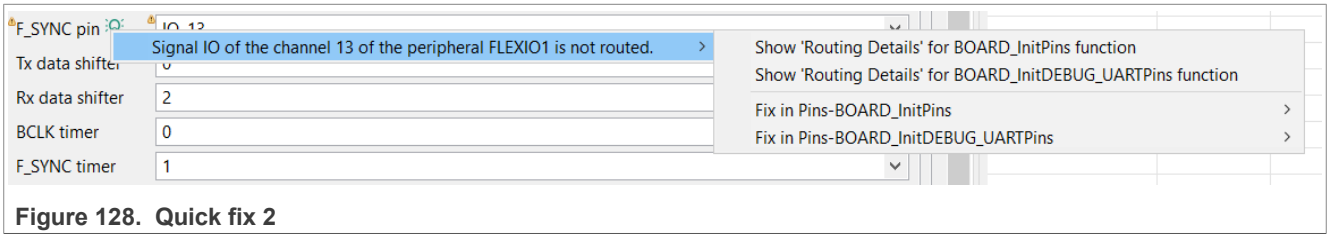


Figure 128. Quick fix 2

The problem can be quickly fixed even when reported from a table cell as the context menu contains the list of possible quick fixes (see register initialized SCTimer, for example LPC54114 **Resources > Outputs setting**).

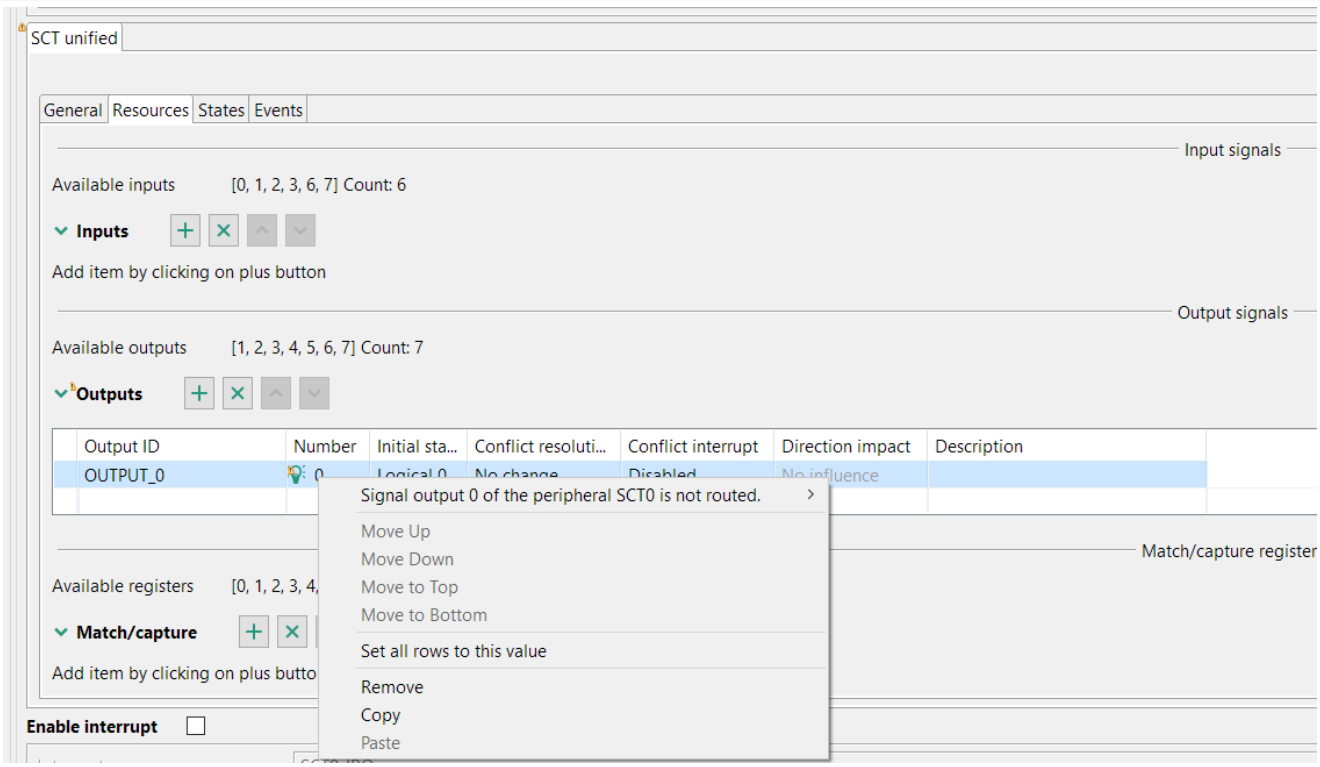


Figure 129. Quick fix 3

5.6 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 **Peripherals** tool.

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

The **Peripherals** tool produces the following C files:

- peripherals.c
- peripherals.h

Note: For multicore processors, peripherals.c/.h are generated for each core, containing functional groups associated with that core. The core association 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. It is prefixed by the “prefix” property if defined in the functional group dialog. These functions include execution of all assigned component 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 130. Code Preview

6 DDR Tool

This section introduces the DDR configuration and validation tool, which is an embedded component of Config Tools for i.MX.

The DDR tool provides two main functionalities: configuration and validation.

Supported devices are indicated in the new project configuration page.

Note: DDR tool is provided “as is” to aid customer capabilities of evaluating, debugging, and optimizing their designs. The results, or any part thereof, provided by the tool cannot be under any circumstances seen as a substitute for the traditional validation and compliance methods, which still need to be performed to declare compliance of the designs with the respective JEDEC standards.

6.1 Create a new DDR tool project

To use the DDR tool, first create a new project.

To create a new DDR tool project, follow these steps:

1. Open the **Config tools for i.MX**.
2. Choose **Create a new standalone configuration for processor, board, or kit** and click **Next**.
3. From **Processors**, choose one of the devices with DDR tool support and click **Finish.**
4. To open the DDR tool view, click the **DDR** tool icon.

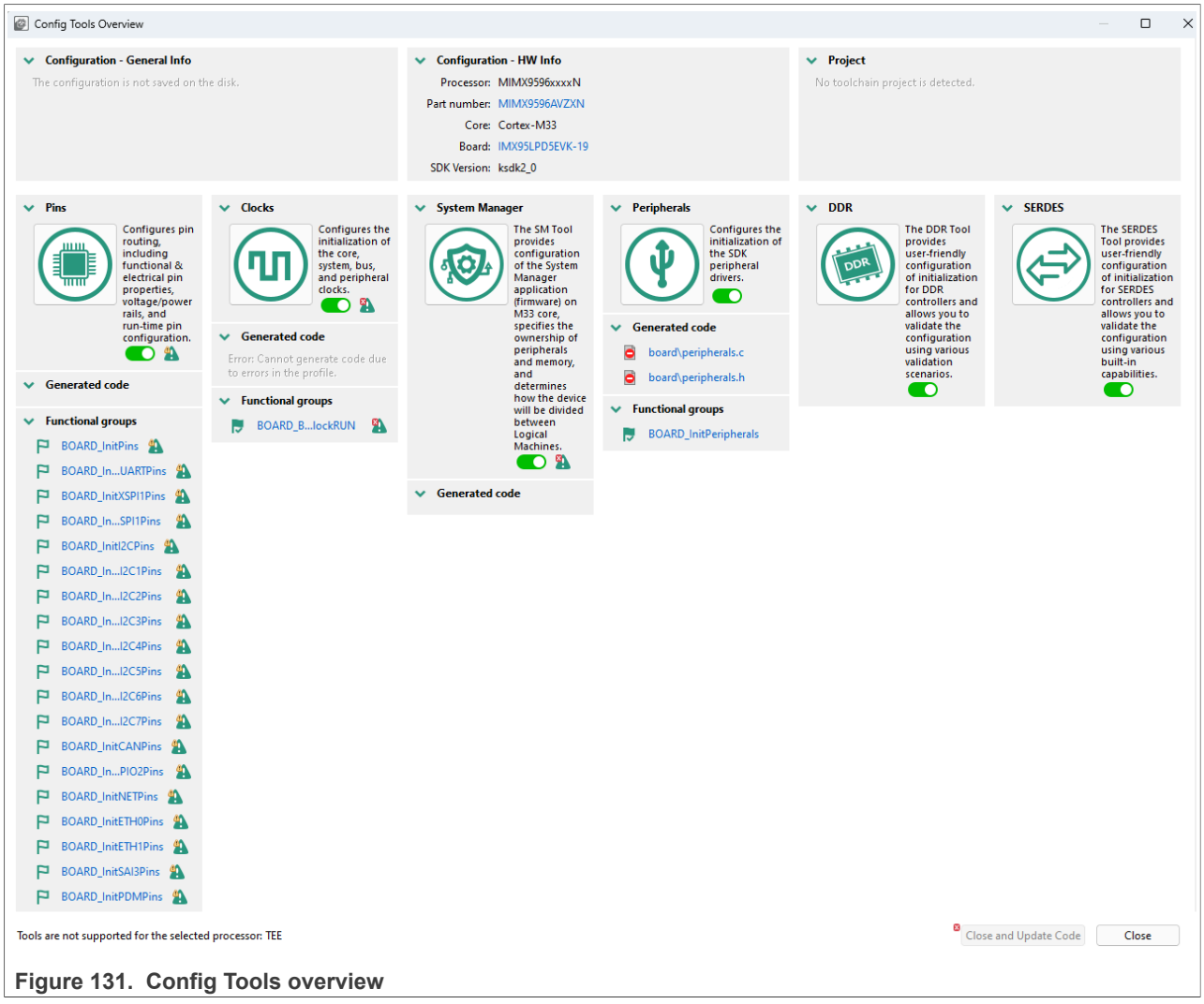


Figure 131. Config Tools overview

5. To use the DDR tool, accept the Disclaimer.

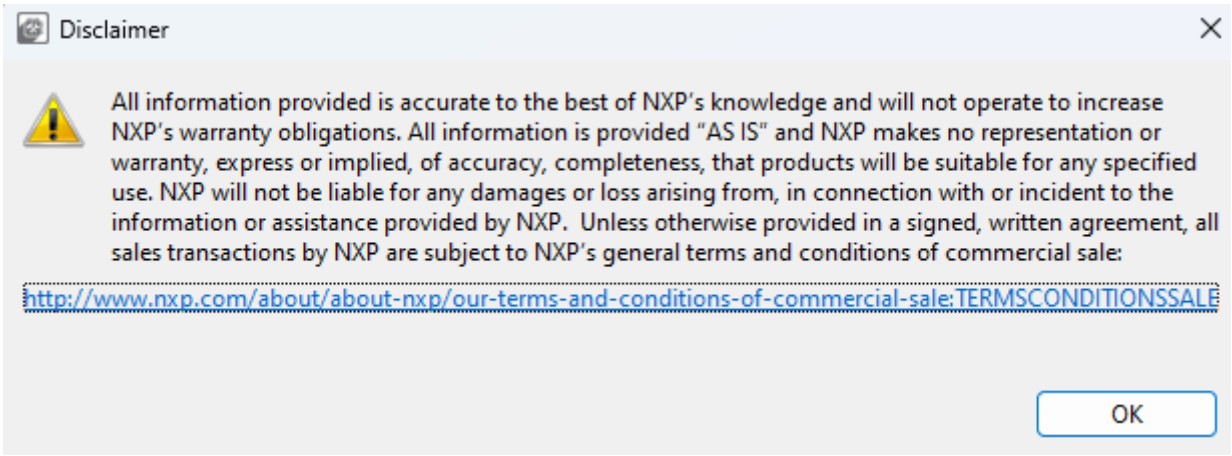


Figure 132. Disclaimer

6.2 DDR configuration

The DDR configuration provides a user-friendly graphical interface to configure the DDR interface and other associated subsystems. You can use it to change the DDR controller and PHY configuration when a different memory module is used to the configuration and to optimize the parameters associated with signal integrity.

6.2.1 Import initialization script

Import initialization script allows you to load the initialization script provided by the **Register Programming Aid (RPA)** tool and bypass the UI configuration.

1. To import the RPA initialization script, use the **Import initialization script** button and browse for the desired *.ds file.

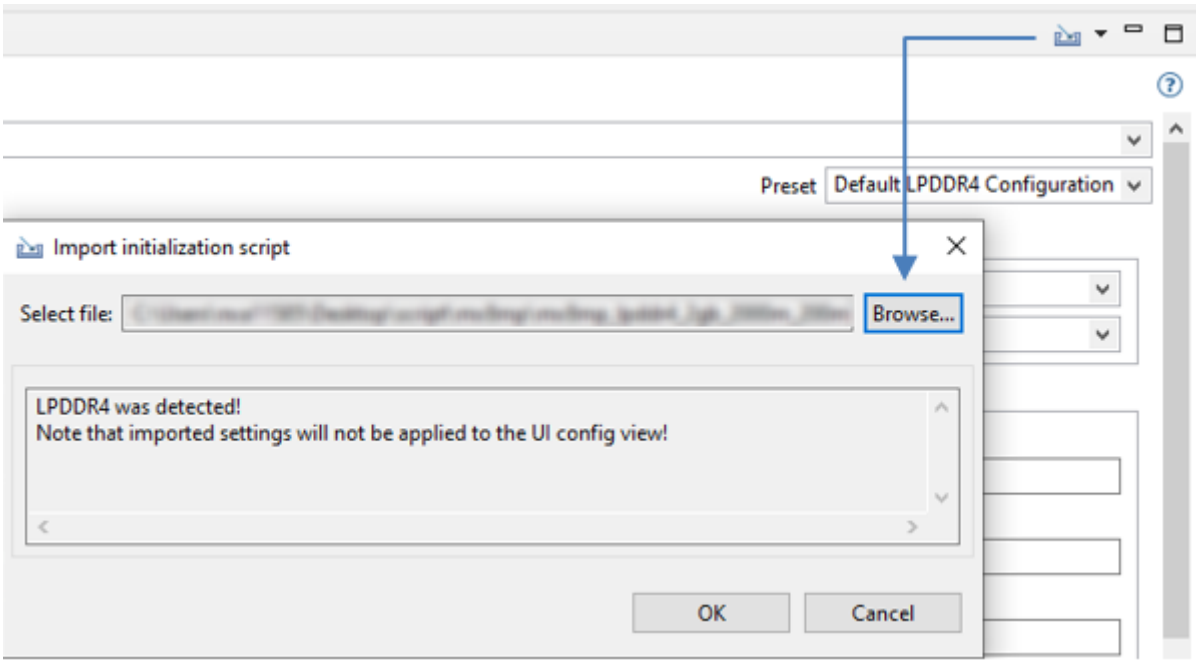


Figure 133. Import initialization script

2. To load the *.ds file and disable the UI configuration interface, press OK.

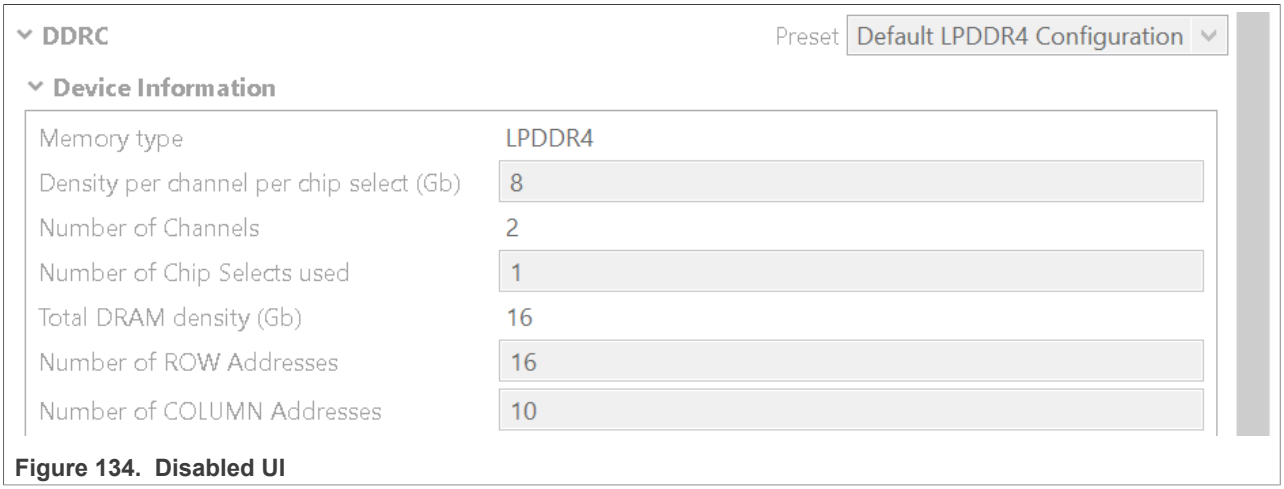


Figure 134. Disabled UI

3. The contents of the imported *.ds file are shown in **Code Preview**, *ddr_config.ds*.

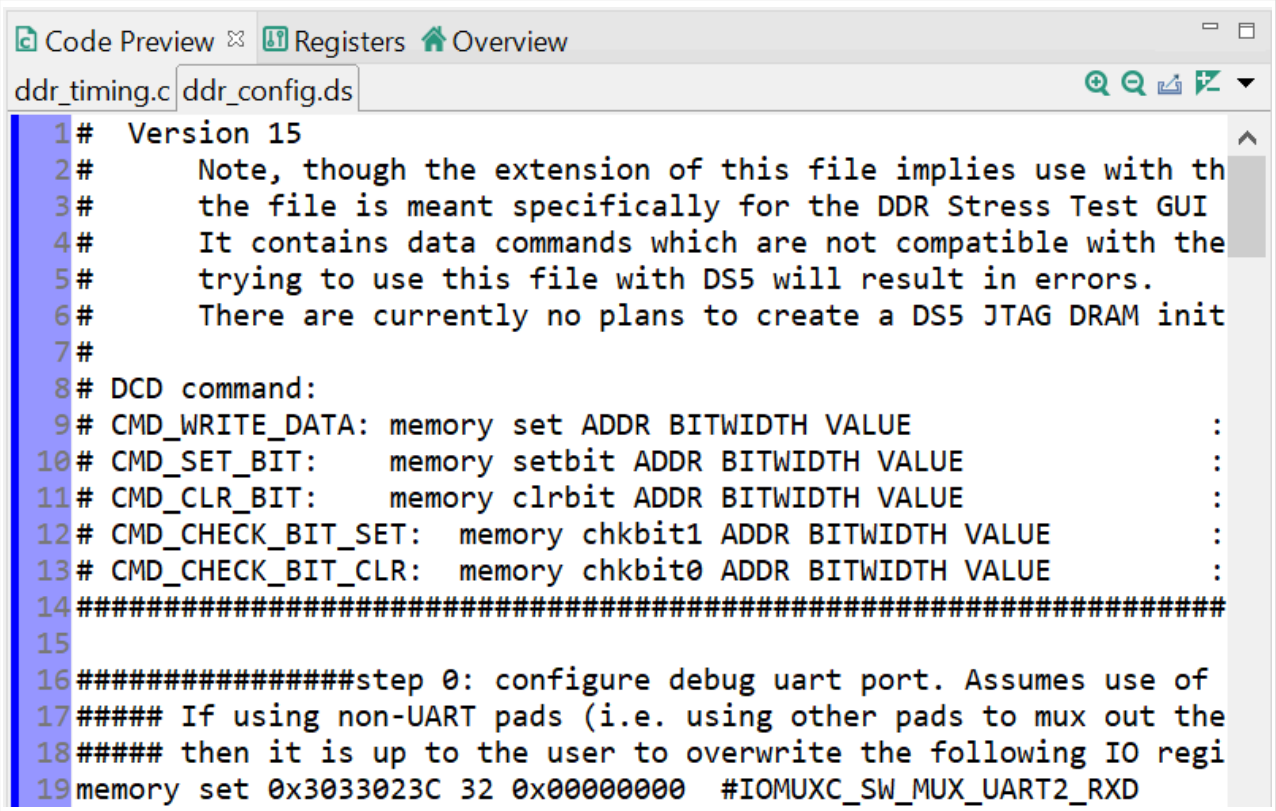


Figure 135. RPA script content

6.2.2 Select custom System manager

For processors with the System manager, the DDR tool supports the default EVK board. For custom boards, select the System manager for that specific board to run memory tests.

To select a custom System manager, use the **Select System Manager** button, select **Use Custom System Manager** and browse for the custom System manager binary.

Note: Modify the SM configuration `configs/customer_board.cfg`

1. In the A55 non-secure section:
   a. Make the LM owner of the `DDR_CTRL, DDR_PHY, SRC`, and `SYSCTR_CTL` resources

```
# Resources
…
DDR_CTRL OWNER
DDR_PHY OWNER
SRC OWNER
SYSCTR_CTL OWNER
```

   b. Grant the full DDR access to the LM:

```
# Memory EXEC, begin=0x080000000, end=<highest DDR address>
```

2. Ensure that UART1 is dedicated to the A55 LM, as the A55 test code uses this UART. This is the default in the NXP EVK SM configuration (SM uses UART2), but customers may have changed this.

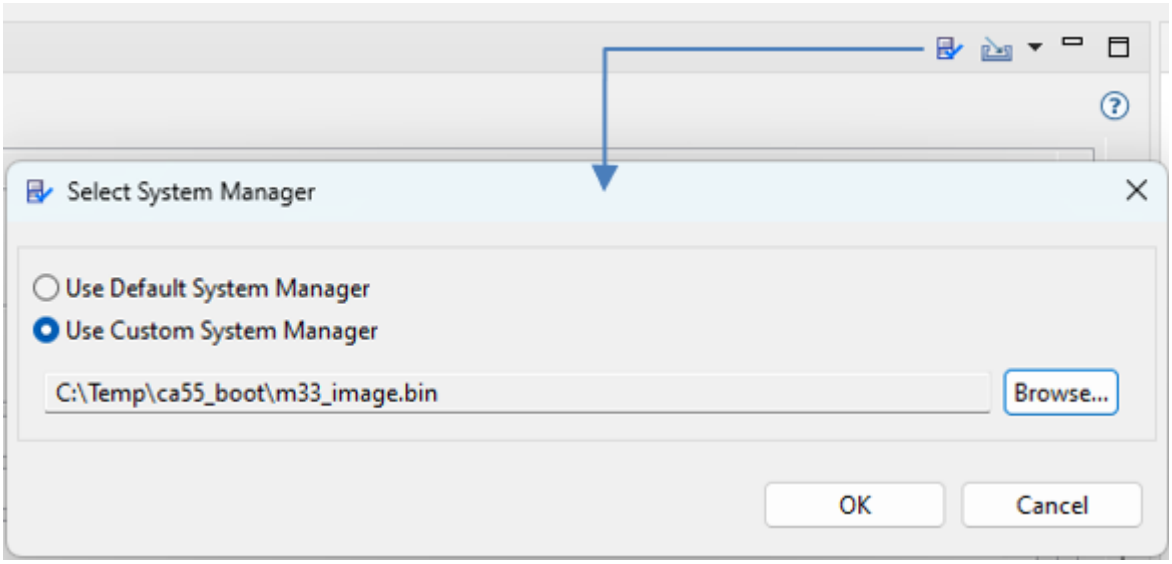


Figure 136. Select System manager

To revert to EVK default System manager, select the **Use Default System Manager** option.

6.2.3 Enable manual configuration

To switch back to UI configuration, press **Enable manual config**.

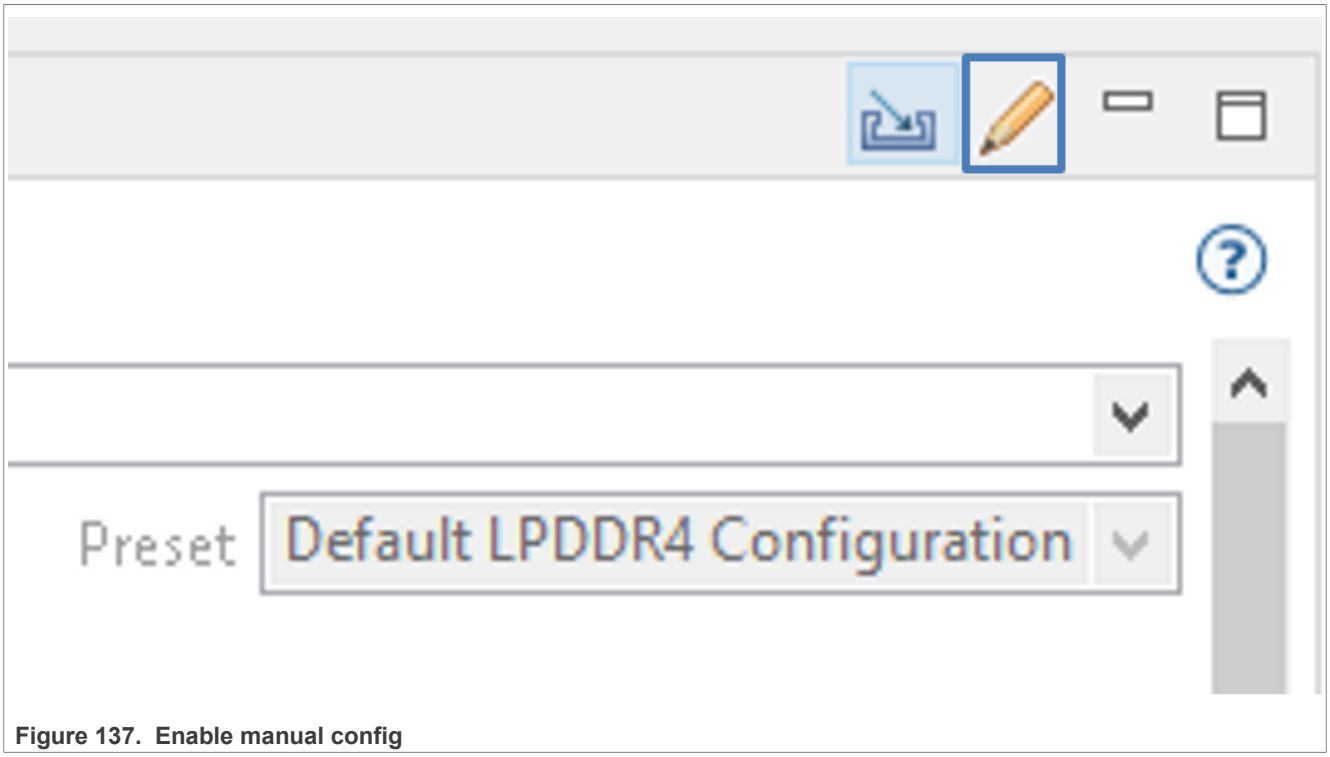


Figure 137. Enable manual config

6.2.4 UI configuration

The UI configuration allows you to manually change Device Information, PHY options, or design-specific configuration. Two modes are available: **Basic** and **Advanced**.

Note: Advanced mode is only recommended for experienced users.

Basic mode allows you to configure the parameters that are design-dependent.

1. Device information

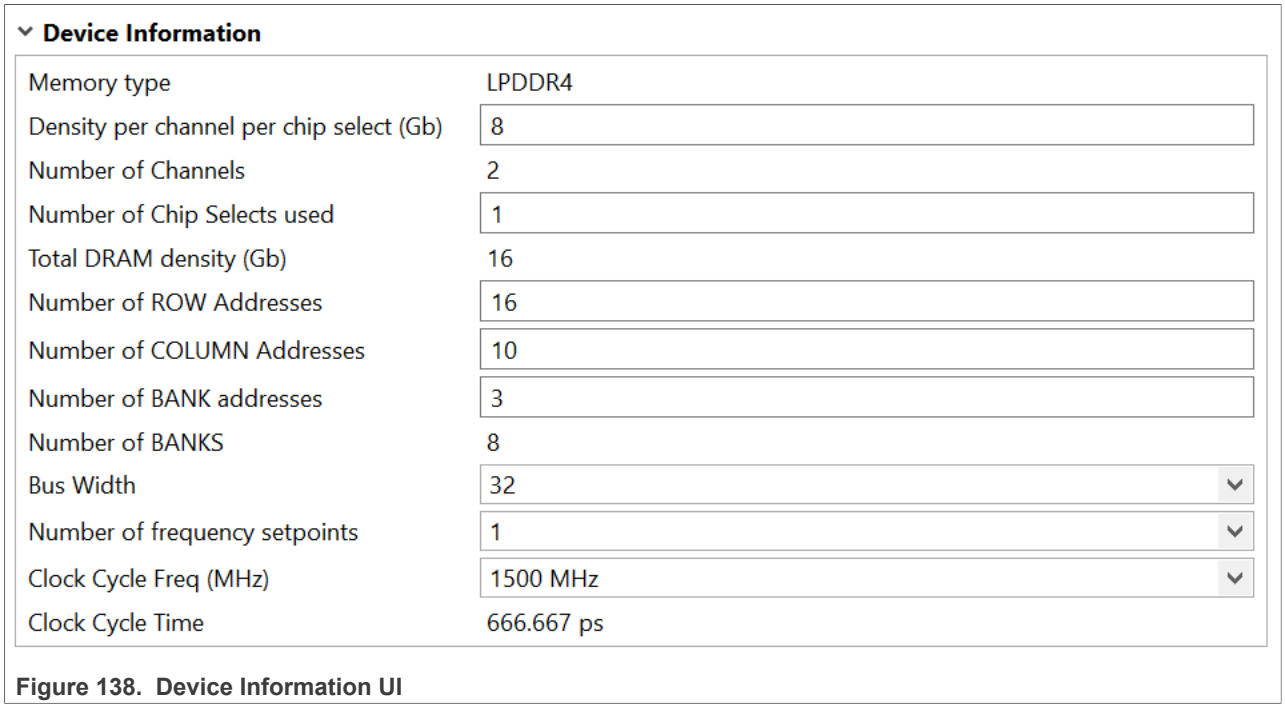


Figure 138. Device Information UI

2. Board data bus configuration for LPDDR4

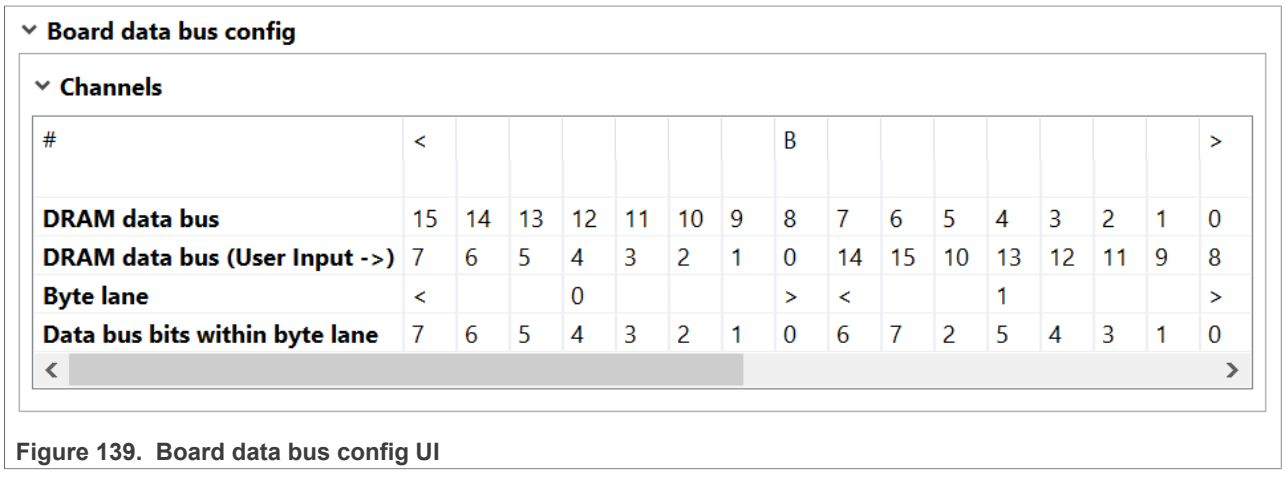


Figure 139. Board data bus config UI

3. UART port selection

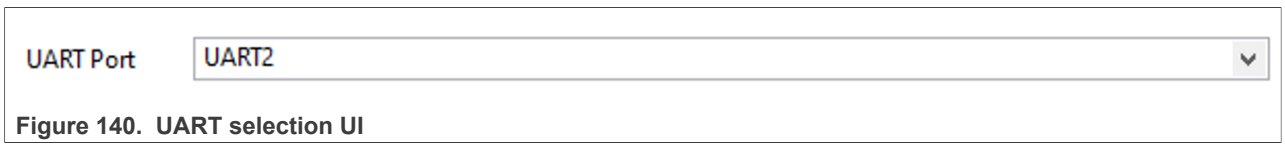


Figure 140. UART selection UI

4. DBI selection (for LPDDR4)

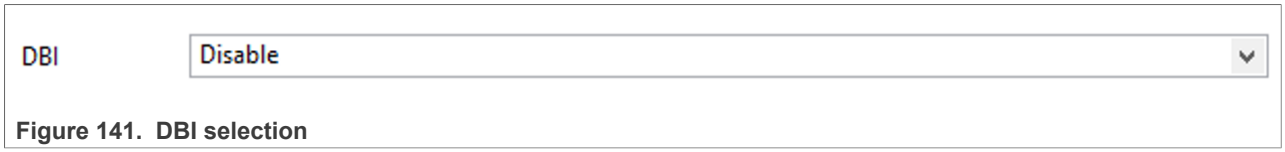


Figure 141. DBI selection

5. Inline ECC configuration (for devices with Inline ECC support) allows you to select the following parameters:
   a. Enable/Disable Inline ECC

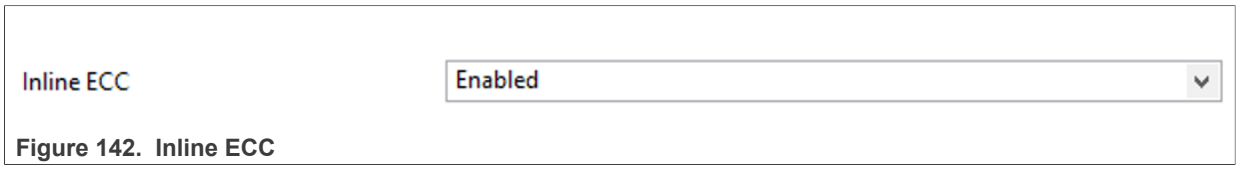


Figure 142. Inline ECC

   b. ECC region granularity

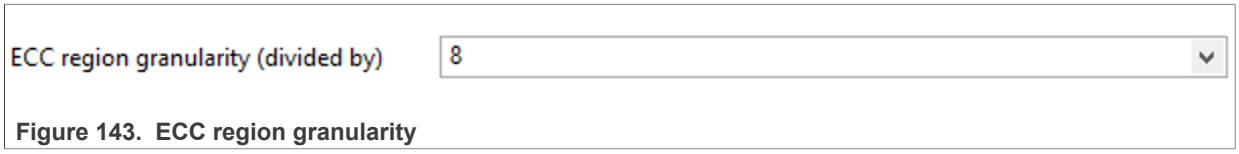


Figure 143. ECC region granularity

   c. ECC protection configuration for each Main Memory Region
   The **ECC config binary/non-binary aligned** section summarizes the ECC configuration overview.

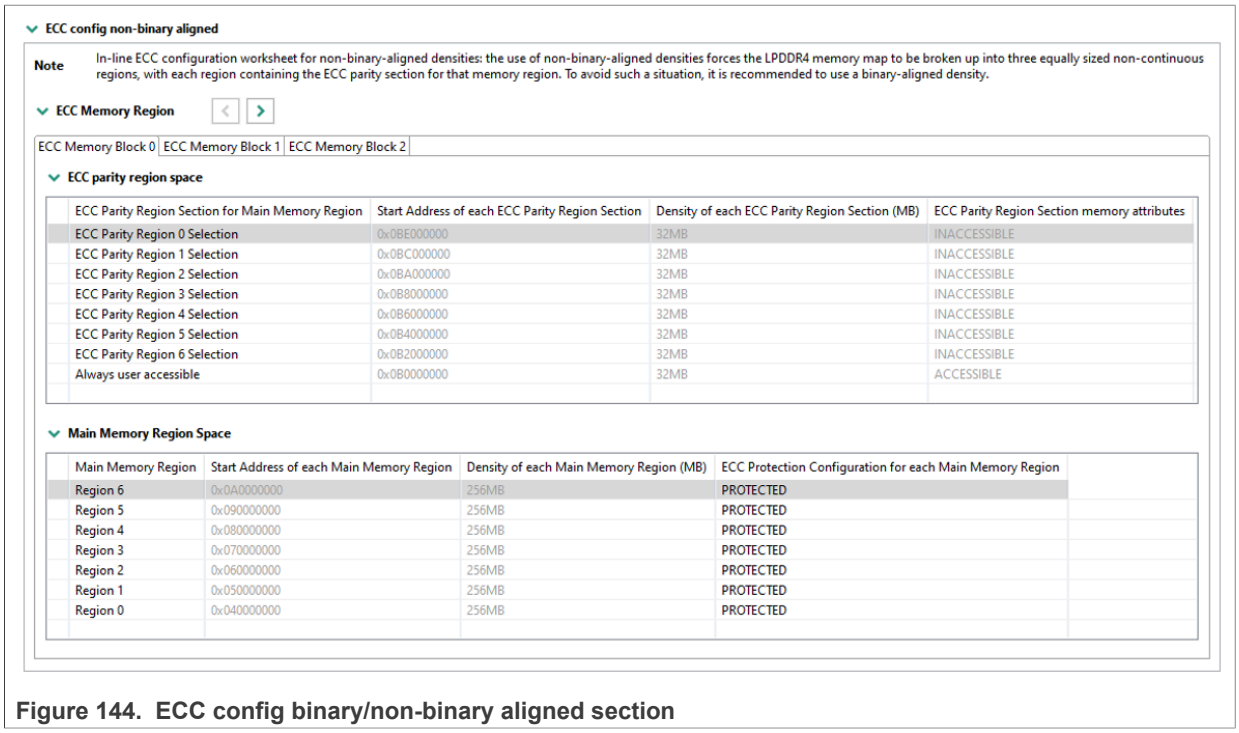


Figure 144. ECC config binary/non-binary aligned section

Advanced mode allows you to configure additional parameters.

1. The firmware version is the one officially supported by the BSP, but the **DDR tool** allows you to select multiple versions.

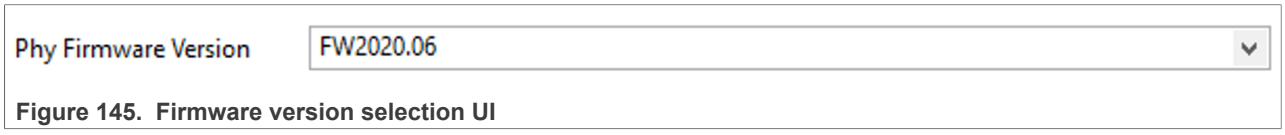


Figure 145. Firmware version selection UI

Note: Use only the Firmware version for your specific SoC and BSP GA version. Not all Firmware versions are supported for each SOC.

2. PHY log level selection

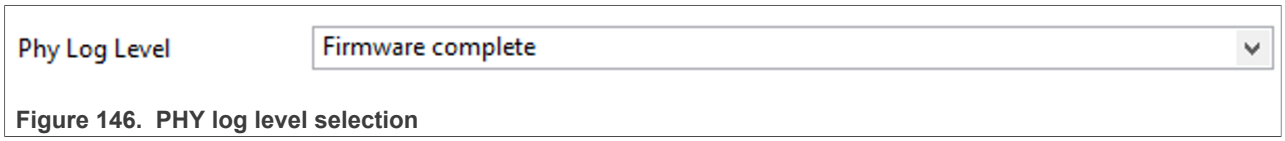


Figure 146. PHY log level selection

3. IOMUX configuration

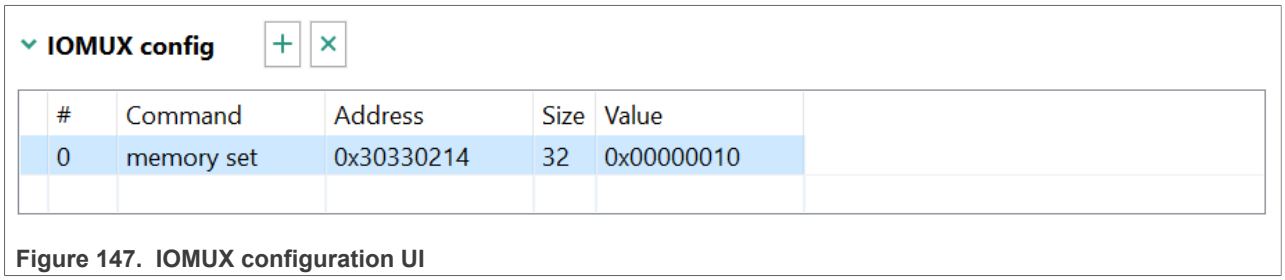


Figure 147. IOMUX configuration UI

4. PMIC configuration sequence
Note: If the Custom PMIC initialization option is enabled, the available commands are: `pmic_set` : most significant byte is the register, followed by the value `pmic_cfg`: most significant byte is the I2C bus, followed by the I2C address.

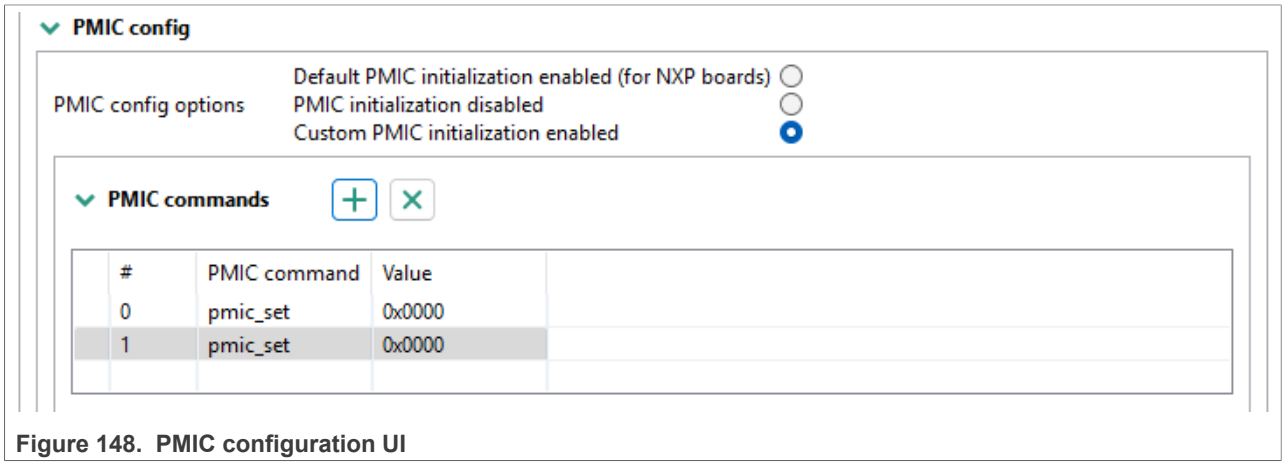


Figure 148. PMIC configuration UI

5. Custom configuration

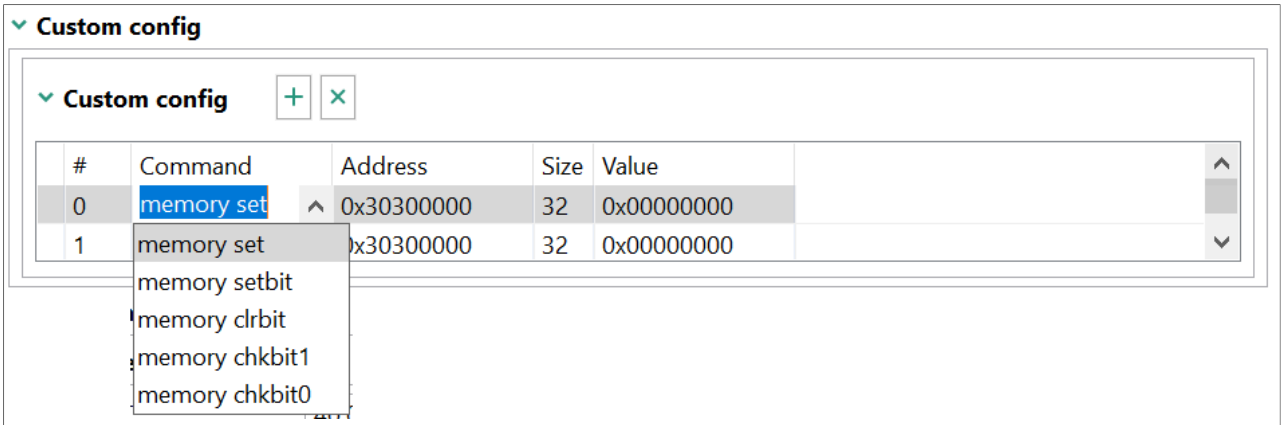


Figure 149. Custom configuration UI

Note: Any write to an incorrect address may cause unexpected behavior.

6. DQ ODT and DS configuration

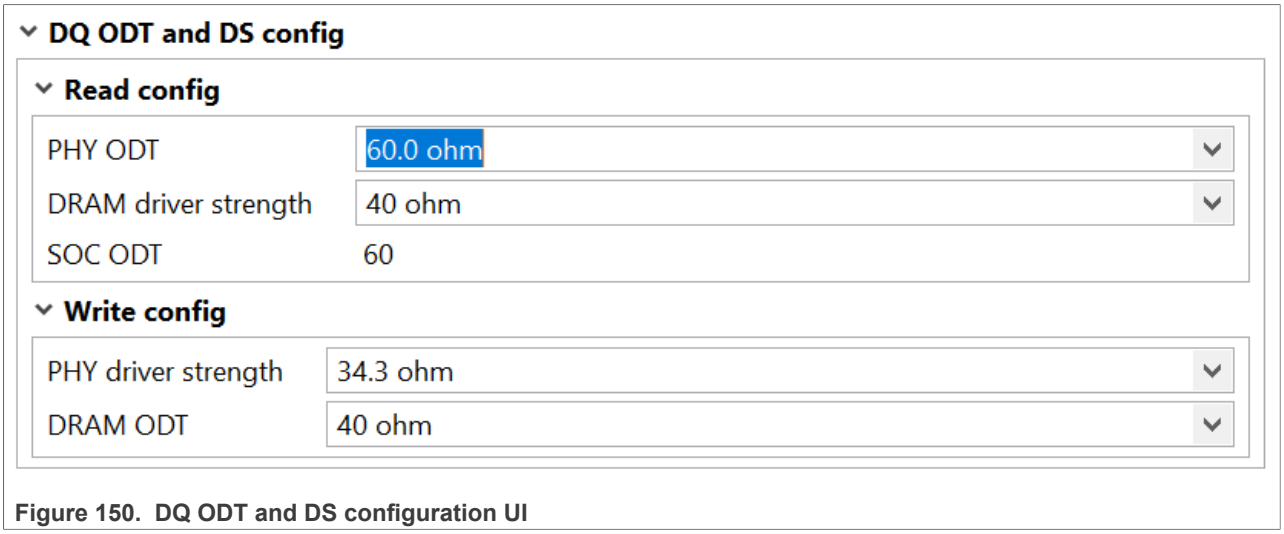


Figure 150. DQ ODT and DS configuration UI

7. CA ODT and DS configuration - Informational Only

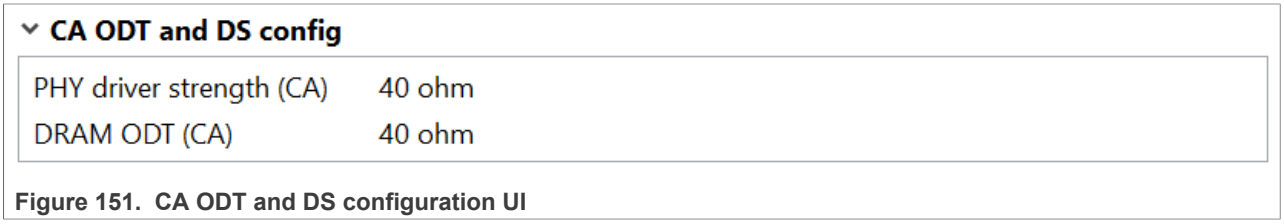


Figure 151. CA ODT and DS configuration UI

6.2.5 Multi-Vendor Initiative

Preset configurations for memory devices from other vendors that have been tested by NXP can be made available by selecting **Include configurations for memories tested by NXP**. They then appear in the **Preset** drop list.

Note: These preset configurations are only applicable to the EVK board on which the device was tested. Custom designs may require additional changes.

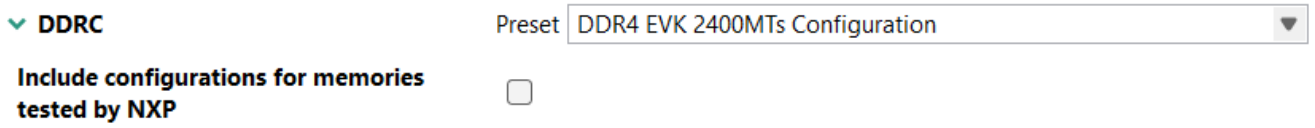


Figure 152. Checkbox for enabling MVI Presets

6.2.6 Code generation

You can generate the configuration as C code in the **Code Preview View**, which can be used by the U-Boot SPL driver.

Any change in the GUI triggers code generation; the update is highlighted in the **Code Preview**.

lpddr5_config.ds | lpddr5_timing.c

```
1  /*
2   * Copyright 2025 NXP
3   *
4   * SPDX-License-Identifier: BSD-3-Clause
5   *
6   * Code generated with DDR Tool v3.8.0_0.7-f3abba53
7   * DDR PHY FW2024.09
8   * Chip revision: B0
9   * Part number: NXP LPDDR5 EVK 19x19 board's default
10  */
11
12 #include "ddr.h"
13
14 /* Initialize DDRC registers */
15 static struct ddrc_cfg_param ddr_ddrc_cfg[] = {
16     {0x5e080110, 0x41114001U},
17     {0x5e080000, 0x3ffU},
18     {0x5e080008, 0x0U},
19     {0x5e080010, 0x4U},
20     {0x5e080014, 0x80000U},
21     {0x5e080020, 0x80000000U},
22     {0x5e080024, 0x80007ffU},
```

Figure 153. Code Preview

Save files from **Code Preview** to disk using the **DDR tool Export Wizard**.

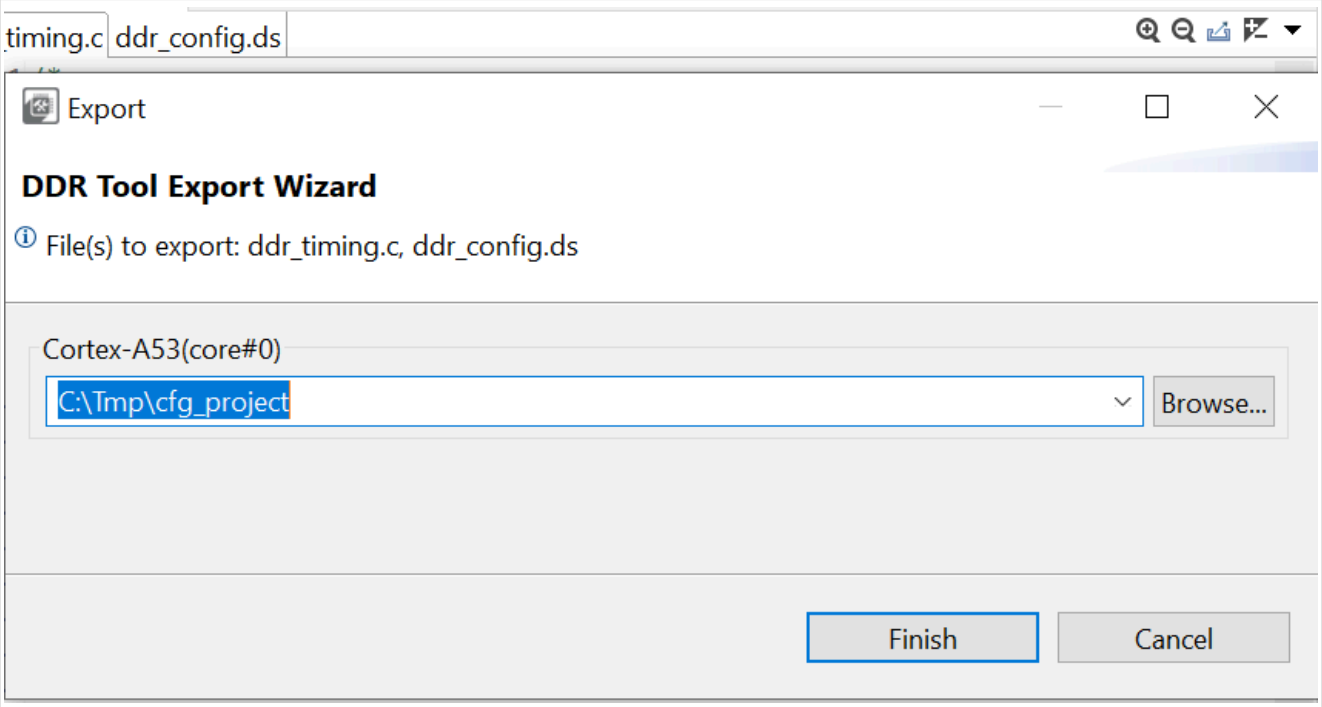


Figure 154. Export generated files

6.3 DDR validation

The DDR validation uses different scenarios to assess DDR performance, by downloading a test image to the processor's internal RAM.

DDR validation assesses stability of the DDR interface on the board in a non-OS environment.

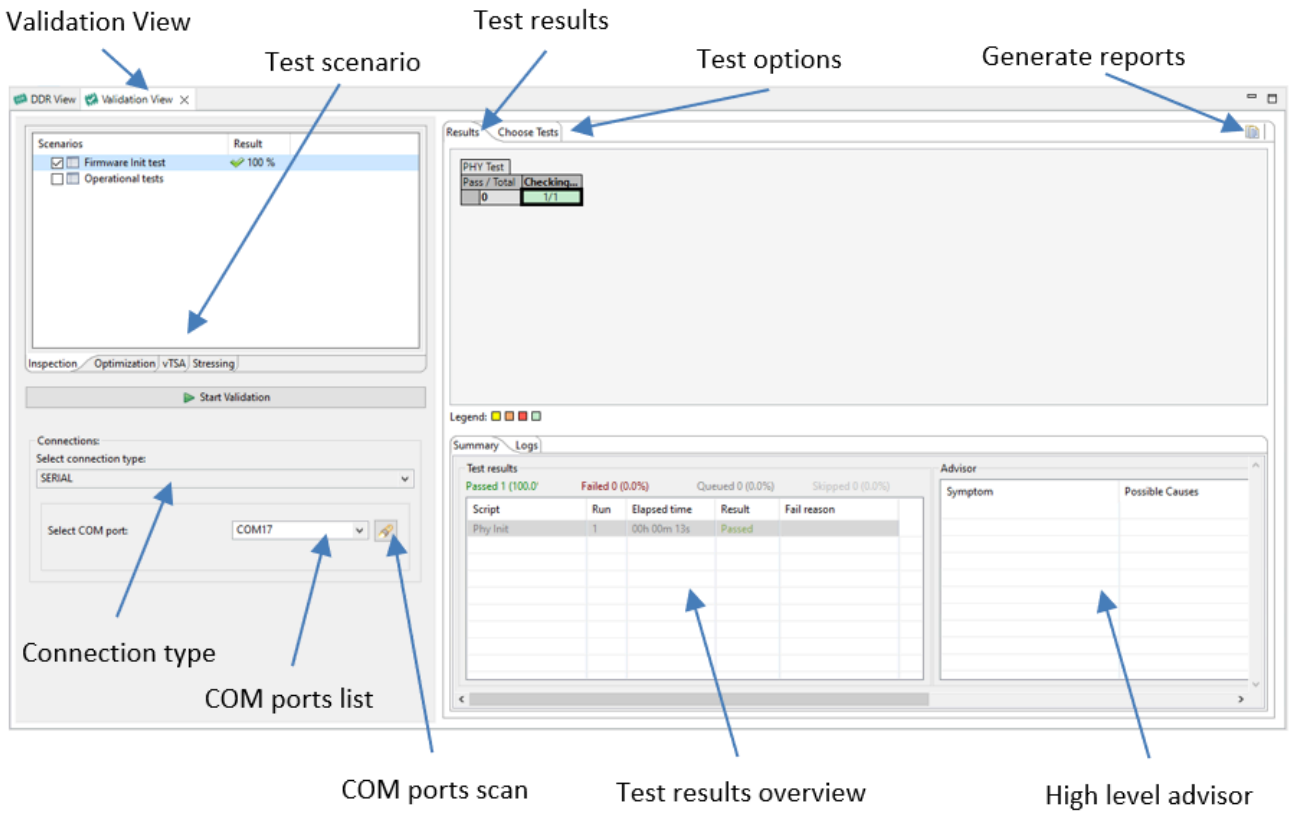


Figure 155. DDR validation

6.3.1 Connection

This section describes connection to boards of various types.

6.3.1.1 Boards with Serial Download mode/Manufacture mode

To connect to a board, do the following:

1. Configure the board to boot in the Serial Download mode/Manufacture mode and power up the board.
2. Connect a UART cable from the host computer to the UART of the A-core on the board.
3. Connect a USB cable from the host computer to the USB port on the board that is used by the Serial Download mode. An “HID-compliant device” or a “USB Input Device” is shown in the Windows Device Manager.

Note: Connect the USB cable directly to the host PC USB port, not through a USB hub.

With the board connected to the host computer, search the UART ports using **COM port scan**. The COM port drop-down list is populated with all available UART ports.

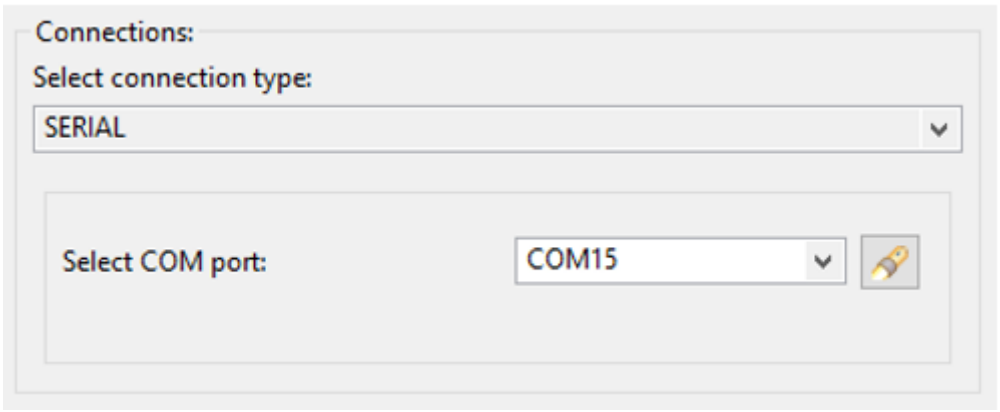


Figure 156. COM port selection

Choose the correct UART that is used as the A-core debug UART port.

6.3.1.2 Boards with JTAG connection available

1. Connect the JTAG probe (only the CWTAP probe is supported) to the board.
2. Select between USB or Ethernet connection.

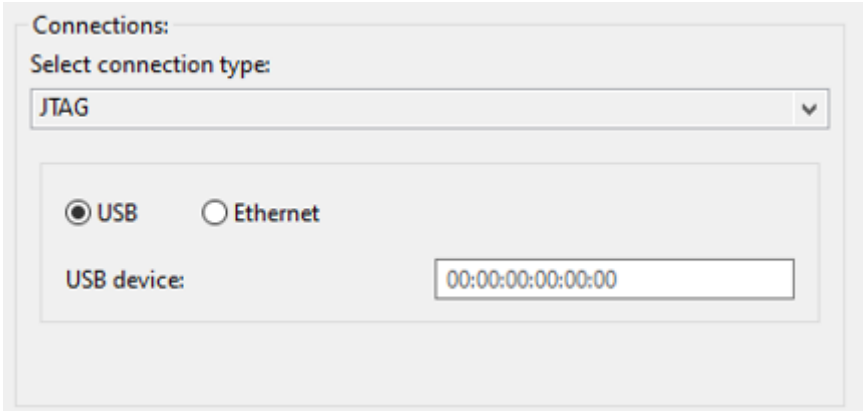


Figure 157. Connection type selection

6.3.2 Test scenarios

Once the DDR configuration and board connection are set up, execute different **Test scenarios**. Customize each test by setting parameters in **Test options**.

Depending on the test and options selected, the execution time may differ. By default, a 90-second timeout is set to ensure that the test finishes if an issue occurs. To change the default value, edit the **Timeout (seconds)** option:

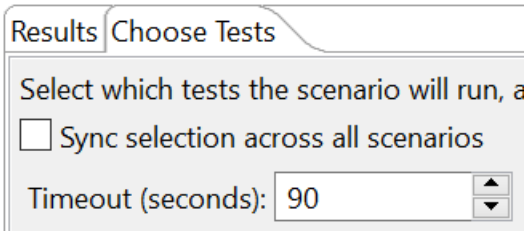


Figure 158. Timeout selection

When the initial timeout expires, provide input to continue or stop the test execution.

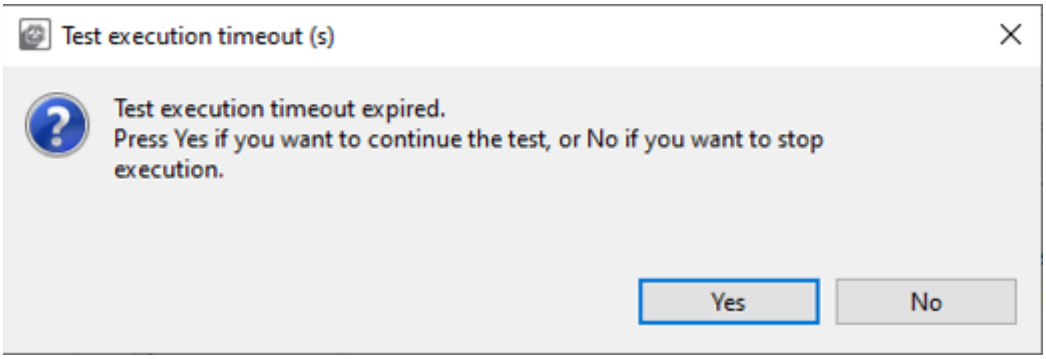


Figure 159. Test execution timeout(s)

To start test execution, press the "**Start Validation**" button. You can check the status of the running test from the **Logs** console. By default, the log level is set to **ERROR**. Additional log-level options are available, with different output in the console:

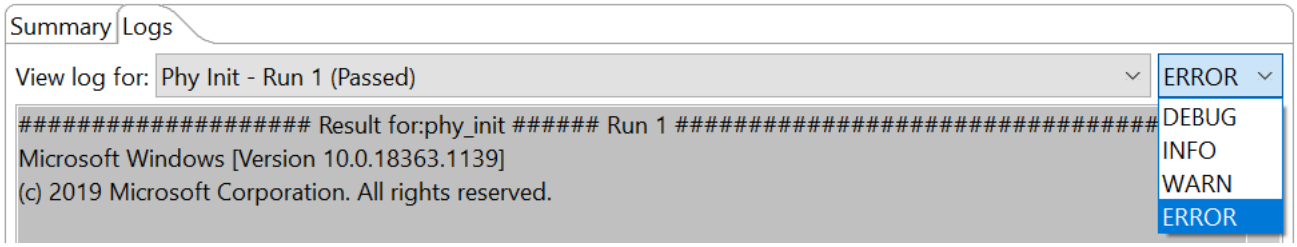


Figure 160. Log level selection

At the end of the test, the PASS/FAIL status is displayed in "**Results**". The test summary is displayed in "**Summary**".

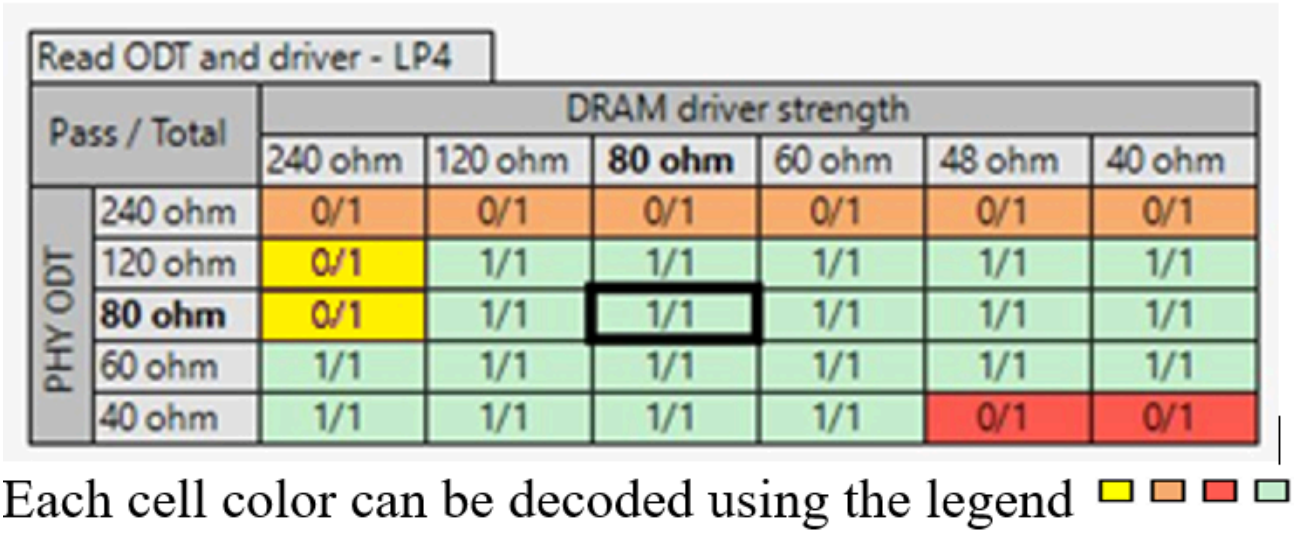


Figure 161. Test status

- Yellow is for *Test failed*
- Orange is for *Configuration error*
- Red is for *Target connection error or exception in the script*
- Green is for *Test passed*

If a test fails, the **Summary** view in the **Advisor** section displays **Symptom** and **Possible Causes** to provide high-level guidance on the debug process when looking for possible DDR subsystem issues.

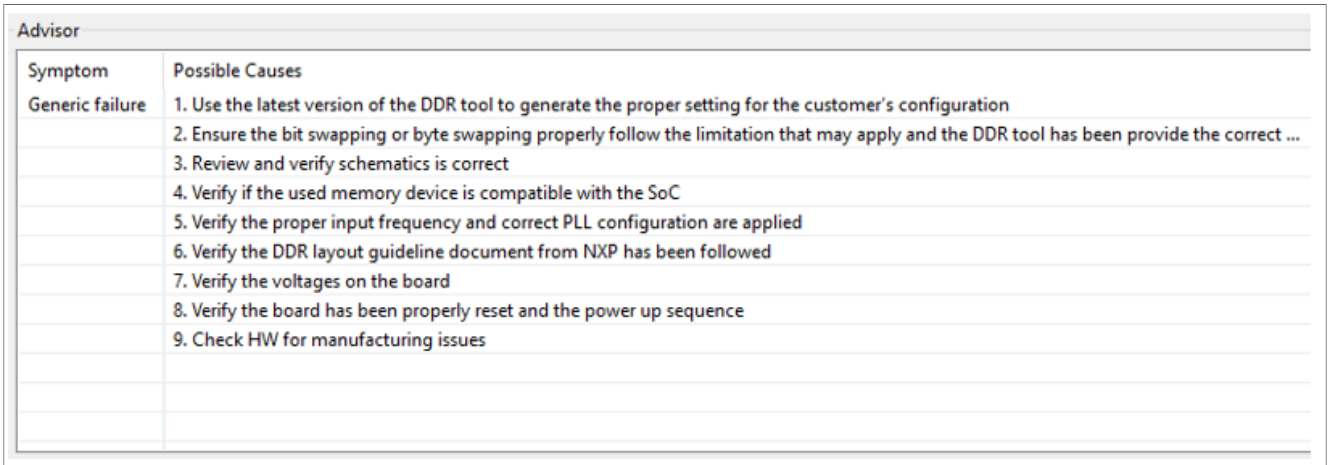


Figure 162. Advisor section

The DDR tool offers several test scenarios that can be split into Inspection, Optimization, vTSA, and Stressing.

6.3.2.1 Inspection

Inspection shows the status of the DDR Controller and DDR PHY configuration, by executing the following tests:

1. *Firmware Init* - executes the DDR PHY training to check the DDR PHY configuration.
2. *Operational* - performs a basic memory access test by running **Write-Read-Compare/ Walking Ones/ Walking Zeros** tests. Options such as Start Address, Size, Enable DDR Memory cache, and Access mode/ Pattern are available for each test.

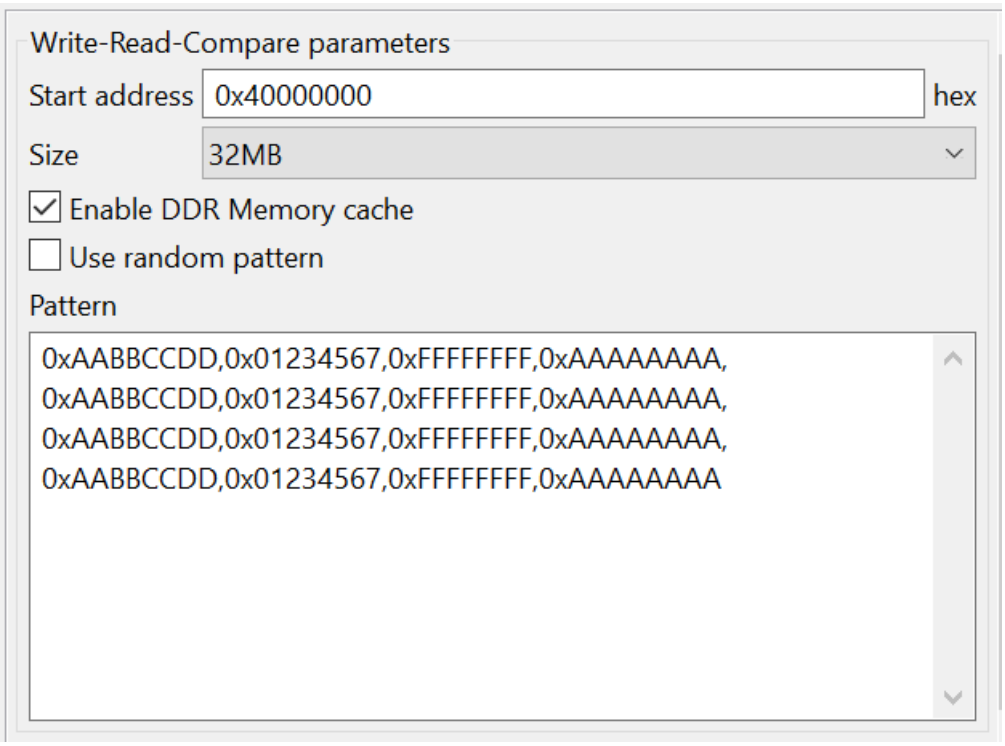


Figure 163. Test options

3. *ECC Regions test* - tests each ECC region with 1-bit error injection to verify the region's ECC capability (protected or unprotected).
   Note: The test is possible only for devices with inline ECC support.

6.3.2.2 Optimization

DQ ODT and driver strength tests sweep the DQ IO configurations to create board-specific Driver Strength vs. ODT PASS/FAIL map for the Reads and the Writes.

Note: Optimization is not available when UI configuration is bypassed by RPA initialization script import.

Read ODT and driver

Pass / Total		DRAM driver strength					
		240 ohm	120 ohm	80 ohm	60 ohm	48 ohm	40 ohm
PHY ODT	**240 ohm**	0/1	0/1	0/1	0/1	0/1	0/1
	120 ohm	0/1	1/1	1/1	1/1	1/1	1/1
	80 ohm	0/1	1/1	1/1	1/1	1/1	1/1
	60 ohm	1/1	1/1	1/1	1/1	1/1	1/1
	40 ohm	1/1	1/1	1/1	1/1	1/1	1/1

Figure 164. DQ ODT and DS map

For passing cells (Green cells), the option *Apply current selection in DDR configuration* (right-click on the cell) is enabled. It sets the respective Driver Strength and ODT value into the configuration for use in other scenarios.

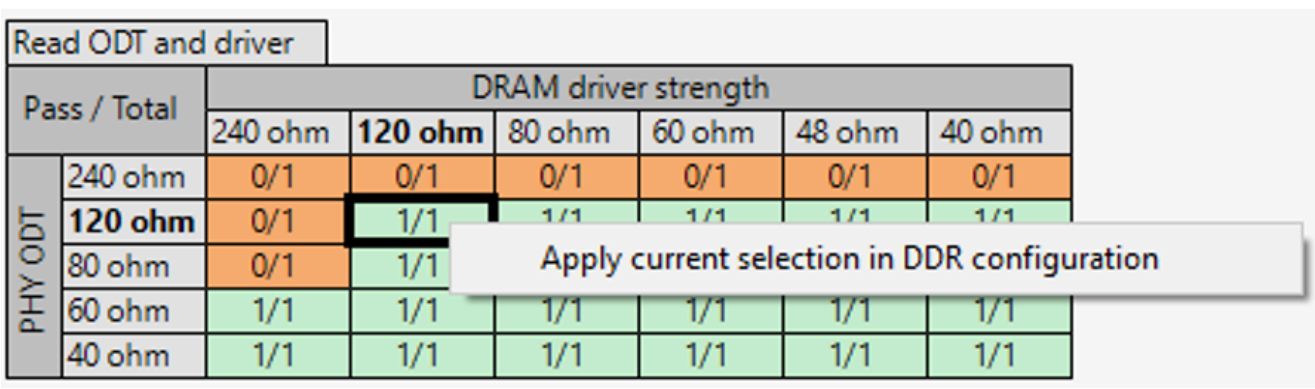


Figure 165. Apply DQ ODT and DS configuration

Note: You can use the Driver Strength vs. ODT map as one of the criteria when deriving optimal ODT/Driver Strength values. This map cannot serve as all-encompassing output to make this determination.

Note: NXP strongly recommends using the default ODT and Drive strength values that are tested and validated as part of the GA BSP. To ensure that your design adheres to the board layout requirements, refer to the i.MX 8M Hardware Developer's Guide.

Vref for 1D optimization test sweeps the PHY Vref and/or DRAM Vref to determine the values needed for PHY training to pass when PHY training fails, and to determine the trained values after the 2D training.

When the test passes, *Apply current selection in DDR configuration* option is enabled. It sets the respective PHY Vref and DRAM Vref values into the configuration.

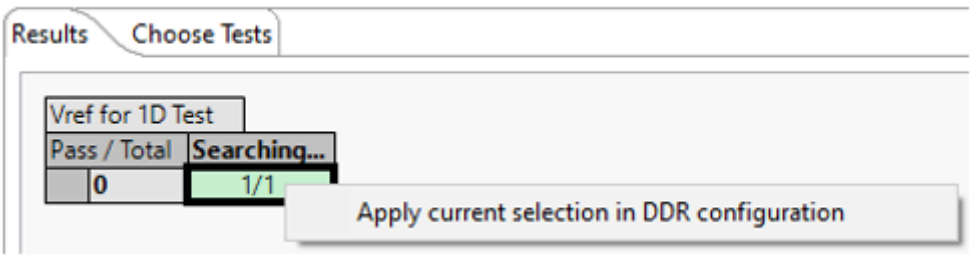


Figure 166. Apply current selection in DDR configuration

Vref for the CA optimization test executes only 1D training and reads the VrefCA after the training is complete.

When the test passes, the **Apply the current selection in the DDR configuration** option is enabled. It sets the VrefCA value in the configuration.

6.3.2.3 vTSA

vTSA performs Virtual Timing Signal Analysis by running tests to determine margins of the DDR subsystem.

Diag Write Margin/ Diag Read Margin tests create virtual data eye diagrams for each DQ lane.

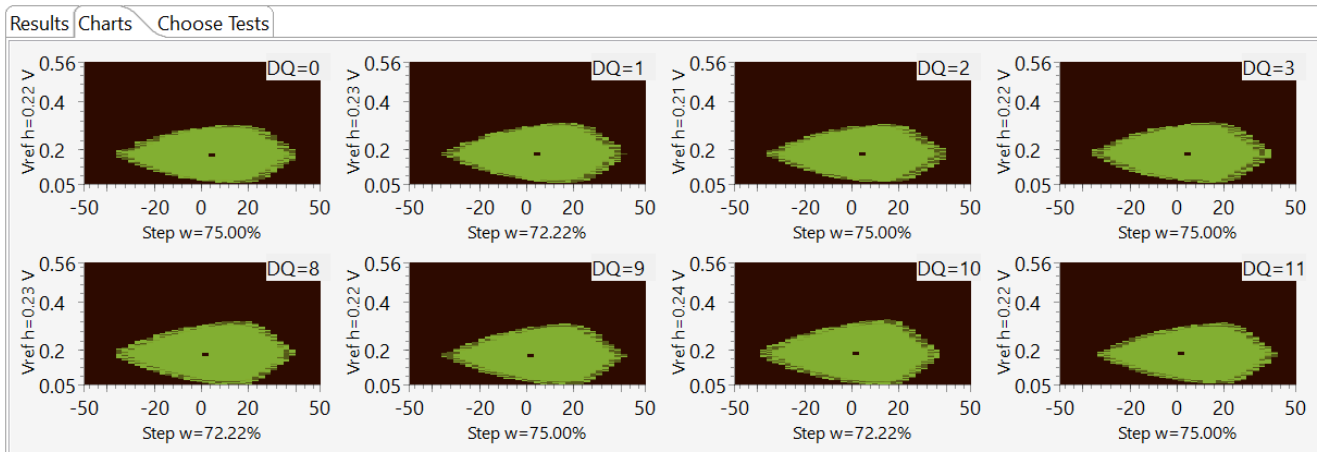


Figure 167. Diagnostic data eyes

CA bus signals test creates post-training diagrams of the 6 CA control lines for each channel and CS.

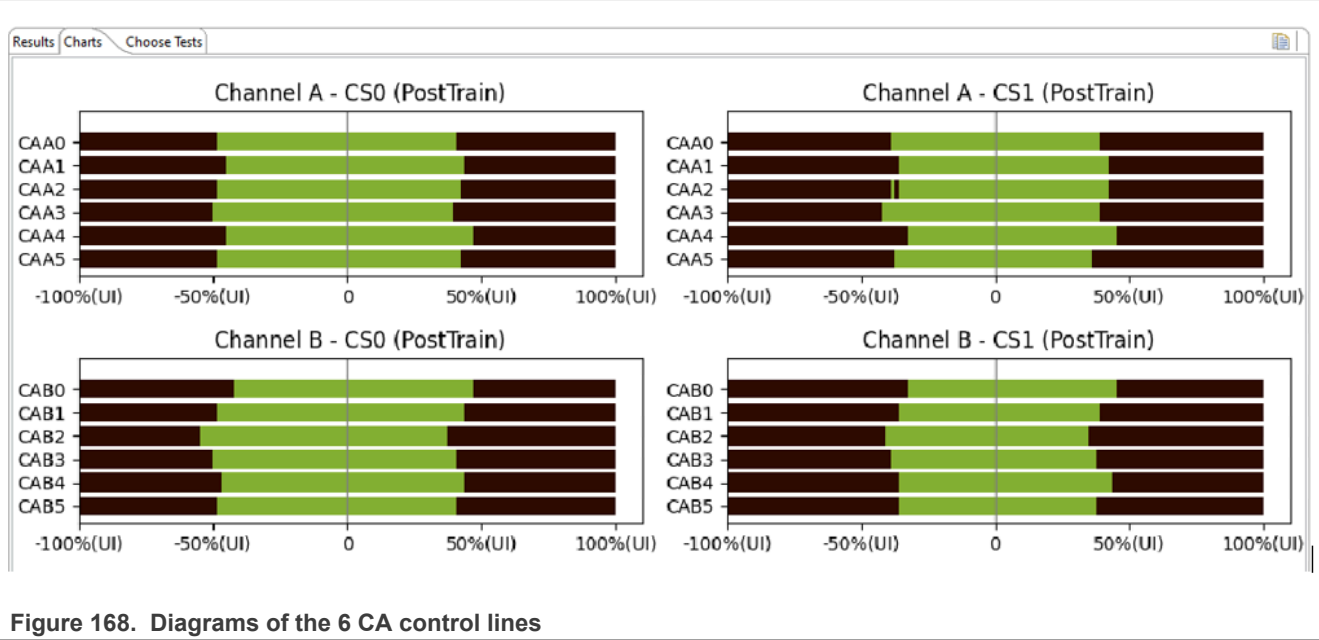


Figure 168. Diagrams of the 6 CA control lines

CA eye test creates post-training eyes of the 6 CA control lines for each channel and CS.

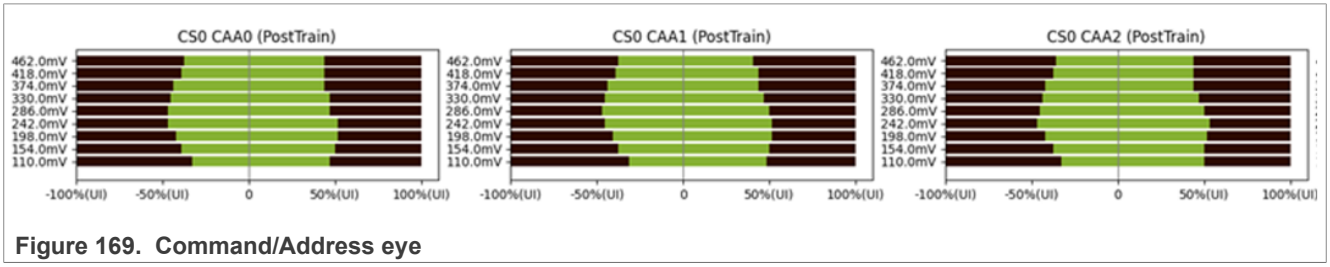


Figure 169. Command/Address eye

Note: Details about vTSA are provided in the FAQ section.

6.3.2.4 Stressing

To test the stability of the DDR configuration more extensively, you can use the *Stressing* scenario, with its suite of tests that covers different situations.

Two ways of running *Stress* tests are available:

1. Single run runs the test suite one time with different options selected (Size, Enable DDR Memory cache, Stop on fail). In case of failure, you can check the status of each test in the suite in the **Logs** console, with Log level set to **INFO**

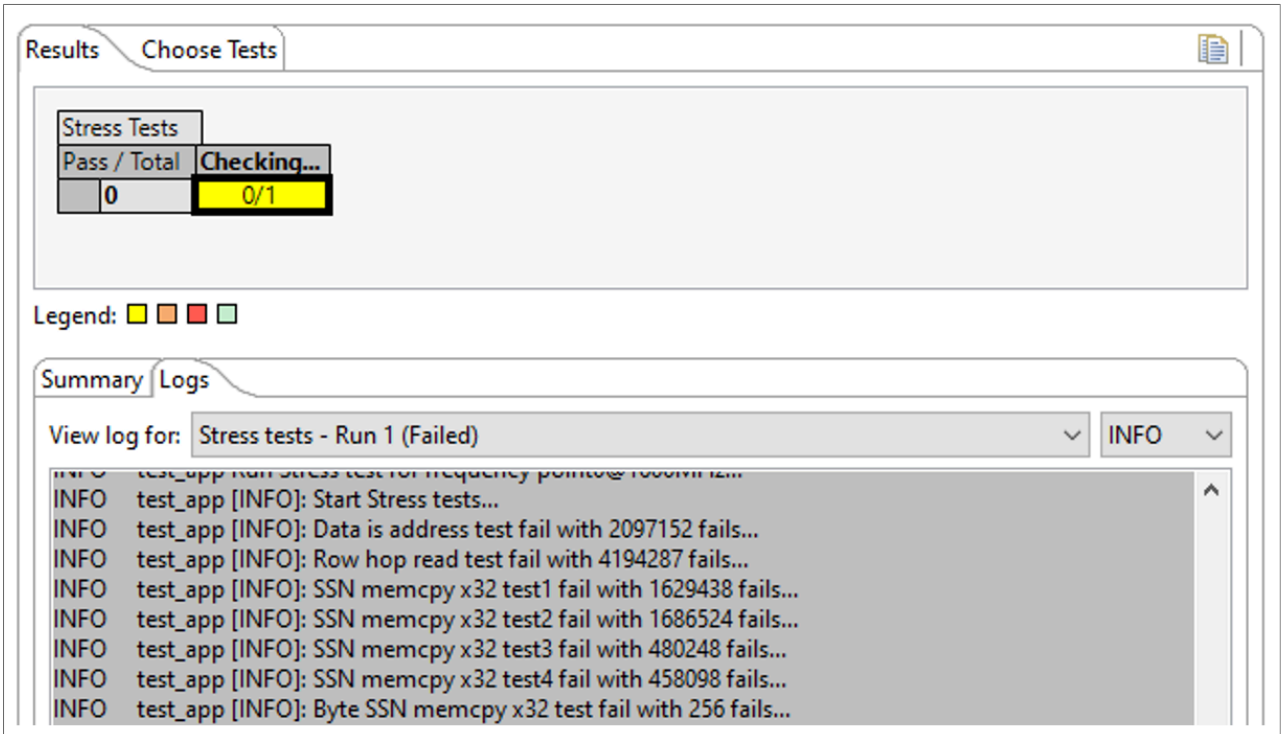


Figure 170. Stress tests

2. Test duration runs the test suite for a selected time. This is suitable for overnight tests.

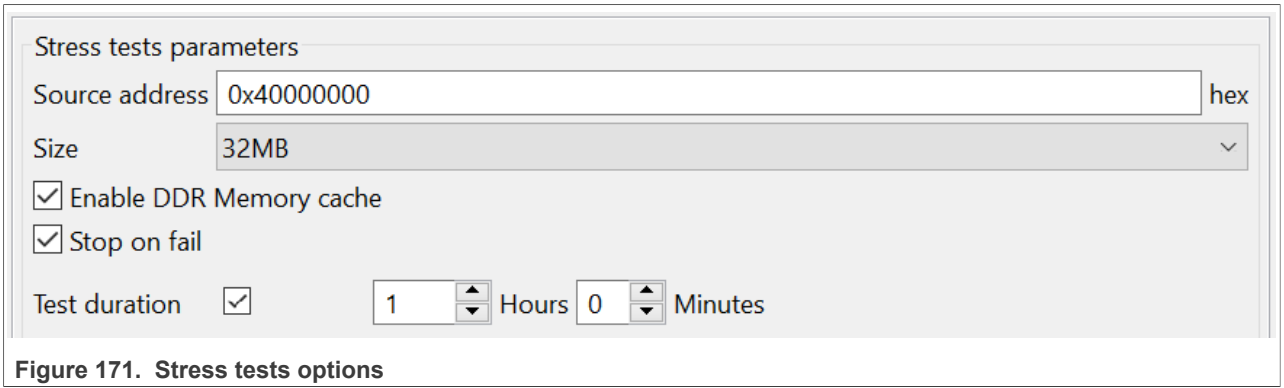


Figure 171. Stress tests options

In the **Logs** console, you can monitor test execution and see the number of iterations and the duration.

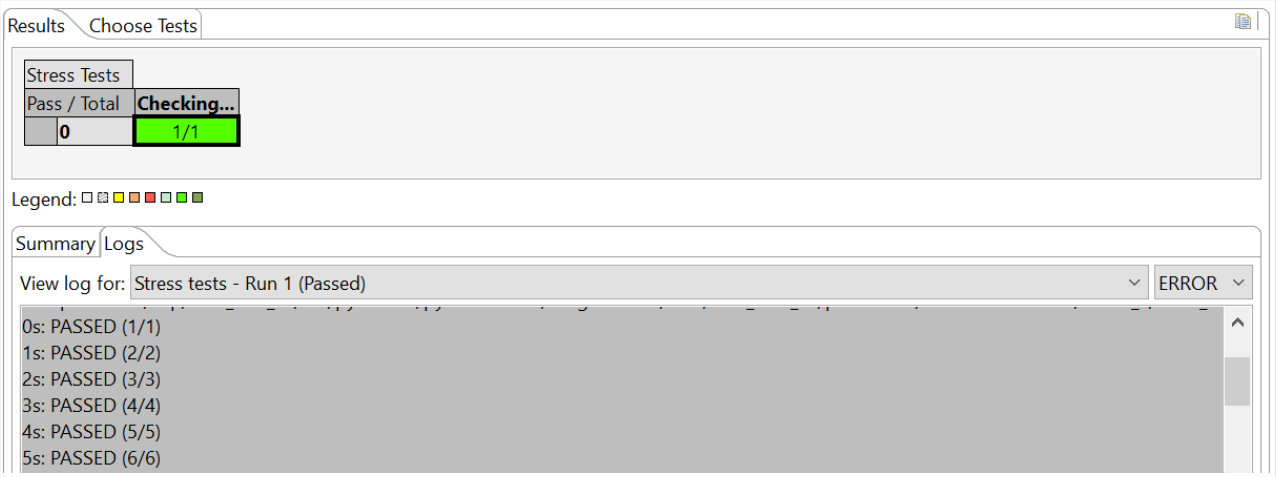


Figure 172. Stress tests results

Note: Ensure the **Timeout (seconds)** setting is higher than the **Test duration** setting, otherwise the test ends with timeout.

6.4 FAQ

1. What does vTSA mean?
   a. vTSA is an abbreviation for Virtual Timing Signal Analysis.
   b. A “virtual” TSA uses the memory controller itself to test margins without test equipment. “Virtual” does not mean simulation!
   c. Memory controllers can alter timings, voltage references, termination settings, and so on, for both incoming and outgoing signals.
   d. “Training” is a process when the memory controller sweeps these parameters and finds the configuration with the most margin for operation.
   e. A vTSA simply logs this information for output, which provides insight into the signaling margin of the system without the need for test equipment.
   f. Initialization and calibration settings can be dumped to a file for analysis as well.
2. What is the vTSA output?
   a. Virtual Timing Signal Analysis(vTSA) provides write and read data eye diagrams virtually by running a series of write/read transactions as opposed to the hardware method of using a high-speed oscilloscope to perform manual physical TSA (pTSA) measurements.
   b. Therefore, vTSA output approximates the actual write/read eyes.
   c. Expect some variation between trained values of delay lines and VREF in comparison to the vTSA report of these values.
   d. vTSA only reports the values it detects in the “widest” part of the reported eye, which may itself vary from run-to-run.
   e. The key takeaway from using this tool is to display the virtual write/read eyes to convince the user of the robustness of their design.

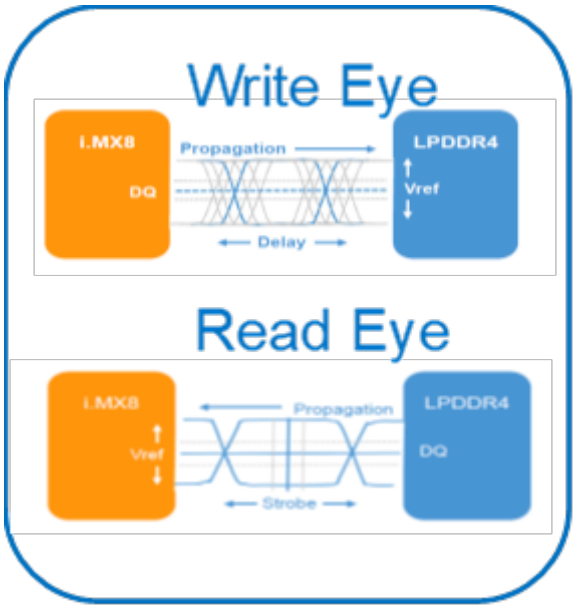


Figure 173. Read eye

3. Need more information about vTSA?
   a. The vTSA tool is an approximation of an actual pTSA. Thus, you may note some variation between the trained delay line value and the vTSA “mid” value. This variation can be up to ~20 ps. Verify that the trained delay line value has ample margin within the data eye.
   b. For the trained VREF value, the LPDDR4 device Mode Register 14 (MR14), which holds the trained VREF value, applies to all byte lanes. Per JEDEC, MR14 is not per byte lane; it applies to all byte lanes.
   c. For a board that follows the NXP DDR layout guidelines, there should be plenty of margin around the trained VREF value. See the guidelines in the respective NXP Hardware Developer Guide.
4. How is a data eye margin generated?
   a. There are several delay steps available to shift each DQ and DQS.
   b. As DQ crosses the unit interval, from zero-step delay to 1 unit interval step delay, each step is tested with a write-read-compare test to determine pass or fail.
   c. The DQ traverse of the unit interval is repeated for all available VREF steps.
   d. Delay steps generate a line, and repeating the lines at each VREF step generates data eye margin.
   e. The crossing of DQS signal with trained VREF and delay step is placed in the generated data eye margin.
   f. Each **passing** dot in the margin eye already meets the setup, hold, and voltage requirement.

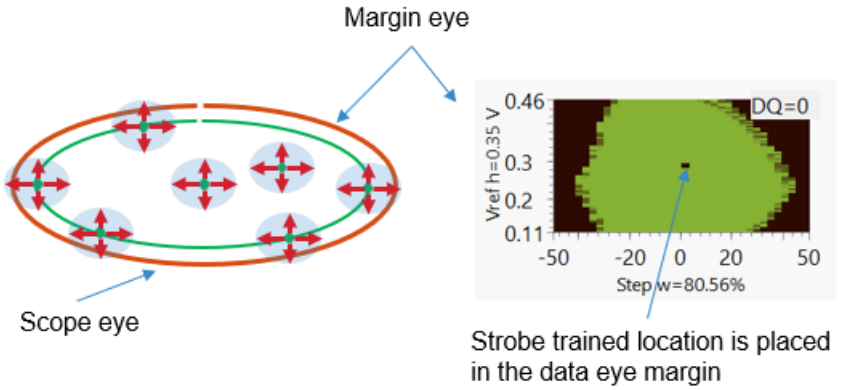


Figure 174. Data eye margin generation

5. What represents the information next to the data eye?
   a. Unit interval = 1/data rate; for example, at 3200MT/s data rate the unit interval = 312.5 ps
   b. The x-axis displays the time. It is one unit interval in percentage. -50 % to +50 %
   c. The x-axis data eye margin width opening is displayed as the percentage of one unit interval. For example: Step w=80.56 % of UI
   d. The y-axis displays the voltage.
   e. The y-axis open data eye margin height/amplitude opening is displayed in voltage. Ex: Vref h=0.35 V

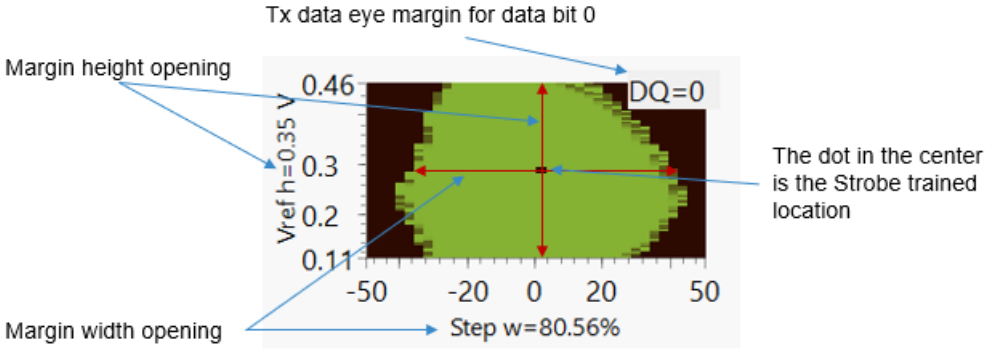


Figure 175. Data eye information

6. How much margin is considered good?
   a. To determine the required margin mask, you must do the following:
      - Optimize the DDR interface
      - Once settings are optimized, generate the worst-case data eye margin using the worst-case conditions (temperature, voltage, frequency, pattern) for a customer board DDR interface.
      - Use DDR training to optimize/center the strobe to the eye margin
   b. A green pixel in the data eye margin indicates a passing cell. It means for that green pixel the setup and hold time as well as the VIHLac/dc are satisfactory.

c. Any additional green pixels around the strobe location in the data eye margin are additional margin available to DDR for that DQ.

7. Why is the trained Vref sometimes not in the exact center of the eye?
   a. VREF training selects a value that satisfies all byte lanes' passing VREF window and programs it into the LPDDR4 MR14 register. For a single byte lane, the trained VREF value may not appear centered in the data eye, but it provides the best margin for that lane while satisfying the other byte lanes.
8. How to check that the wrong UART is selected?
   Set the Log Level to DEBUG and check the messages from console

```
File "C:\nxp\i.MX_CFG_v9\bin\python38\serial\serialwin32.py", line 62, in open
  raise SerialException("could not open port {!r}: {!r}".format(self.portstr, ctypes.WinError()))
serial.serialutil.SerialException: could not open port 'COM4': PermissionError(13, 'Access is denied.', None, 5)
```

Figure 176. Wrong UART selected - message 1

or

```
Traceback (most recent call last):
  File "C:\ProgramData\NXP\mcu_data_v9\processors\MIMX8MM4xxxKZ\ksdk2_0\mem_validation\ddrc\scripts\common\base_test.py", line 224,
    assert self.is_waiting_for_input()
AssertionError
```

Figure 177. Wrong UART selected - message 2

9. How to proceed in case of test timeout?
   a. In case of test timeout, the below pop-up window appears

Figure 178. Test timeout pop-up

   b. Increase the timeout (second) option and rerun the test. If the following error appears, power off the board, unplug the UART cable, power on the board, then plug in the UART cable.

```
Traceback (most recent call last):
  File "C:\ProgramData\NXP\mcu_data_v9\processors\MIMX8MM4xxxKZ\ksdk2_0\mem_validation\ddrc\scripts\common\base_test.py", line 224,
    assert self.is_waiting_for_input()
AssertionError
```

Figure 179. UART connection error

7 Trusted Execution Environment Tool

In the **Trusted Execution Environment**, or **TEE** tool, you can configure security policies of memory areas, bus masters, and peripherals 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, enable the relevant enable secure check option in the **Miscellaneous** subview of the **Security Access Configuration** view.

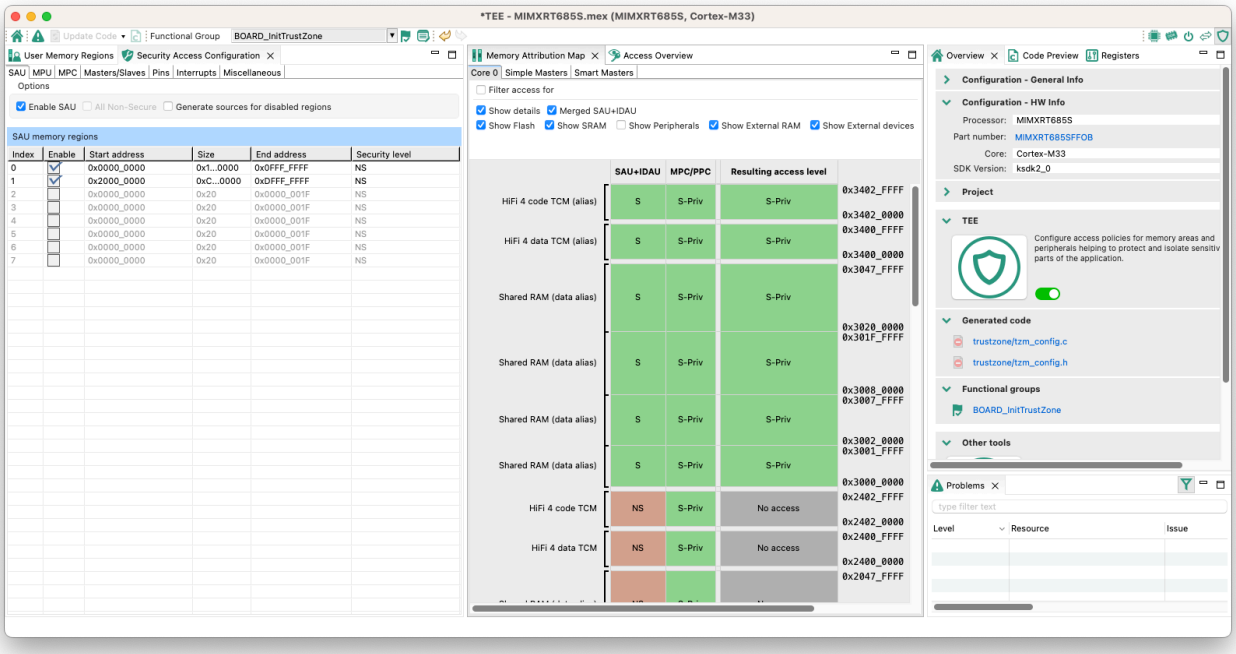


Figure 180. TEE tool user interface (SAU+IDAU)

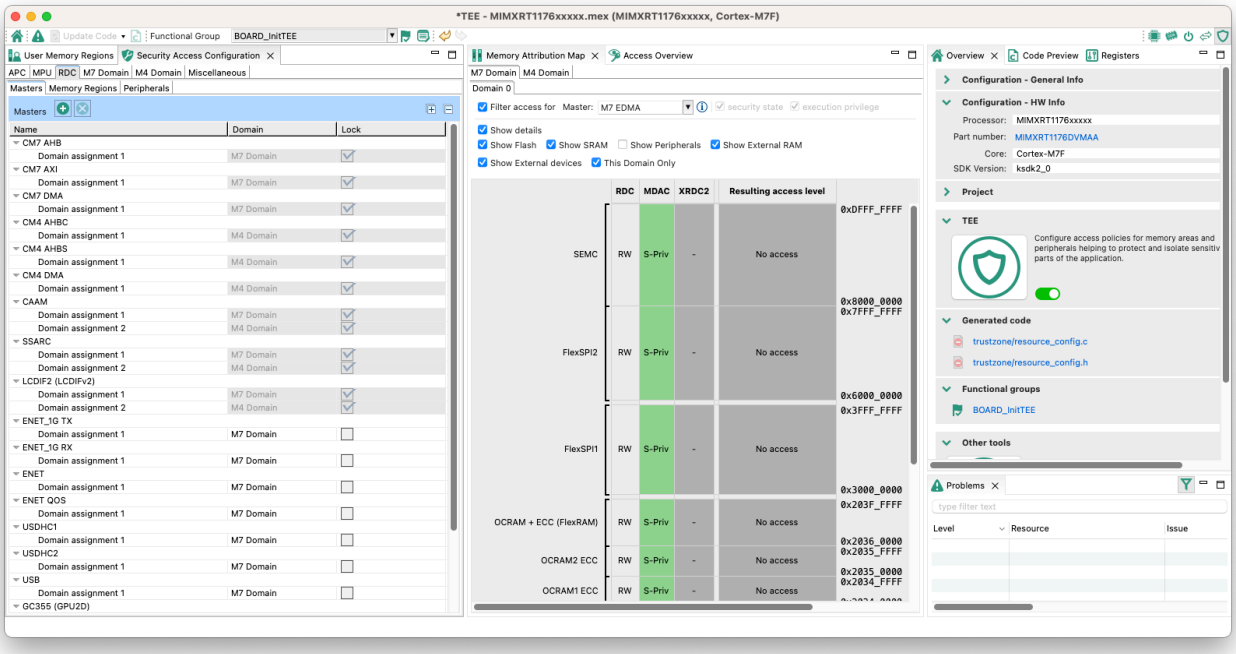


Figure 181. TEE tool user interface (RDC)

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, and specify their address, size, security level, and 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 or 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. If the issue is easily fixed, 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.

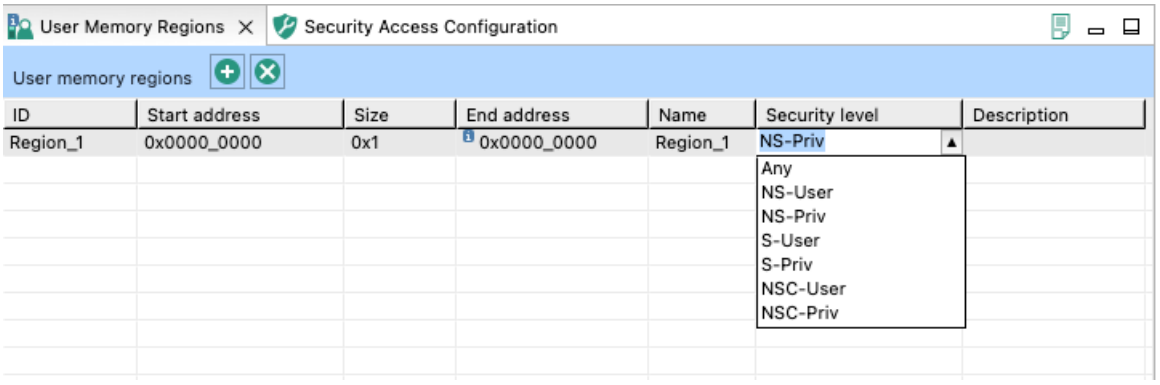


Figure 182. User Memory Regions

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 SAU memory regions to non-secure access level by selecting **All Non-Secure**.

Note: This option is only available when SAU is disabled.

To generate code for disabled memory regions, select the **Generate sources for disabled regions** option.

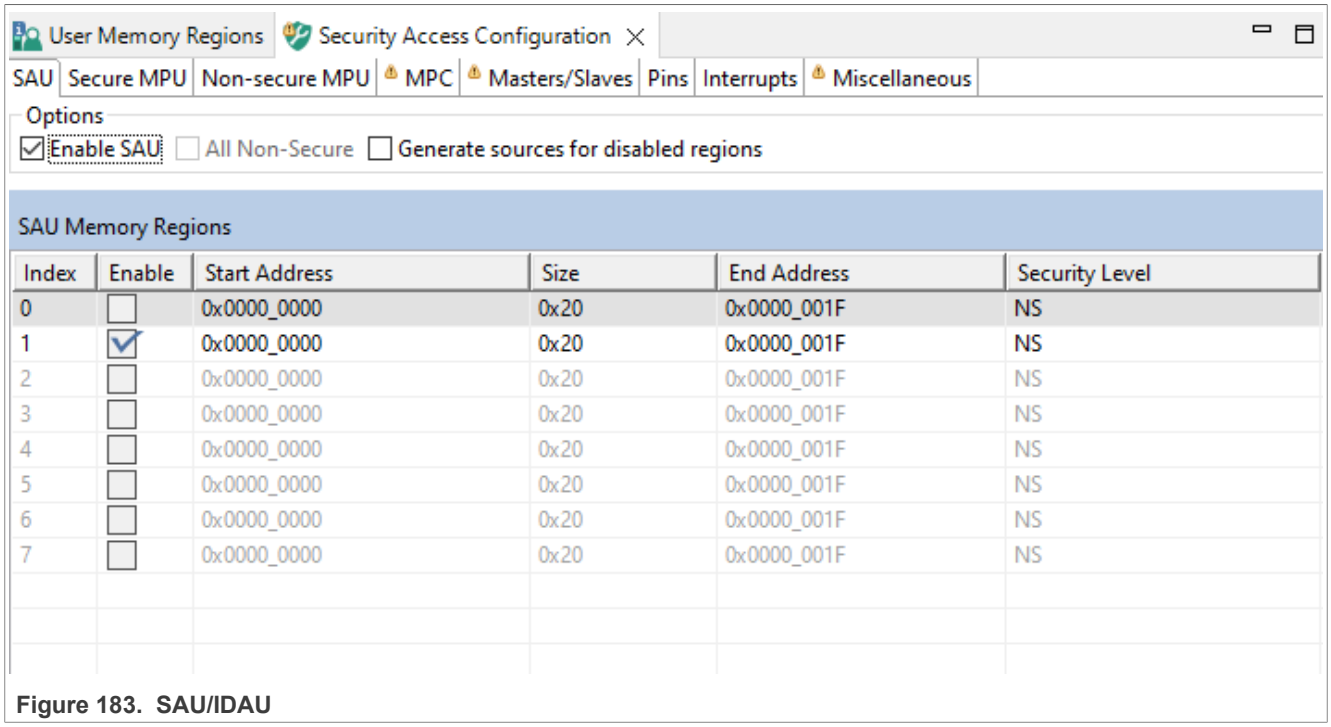
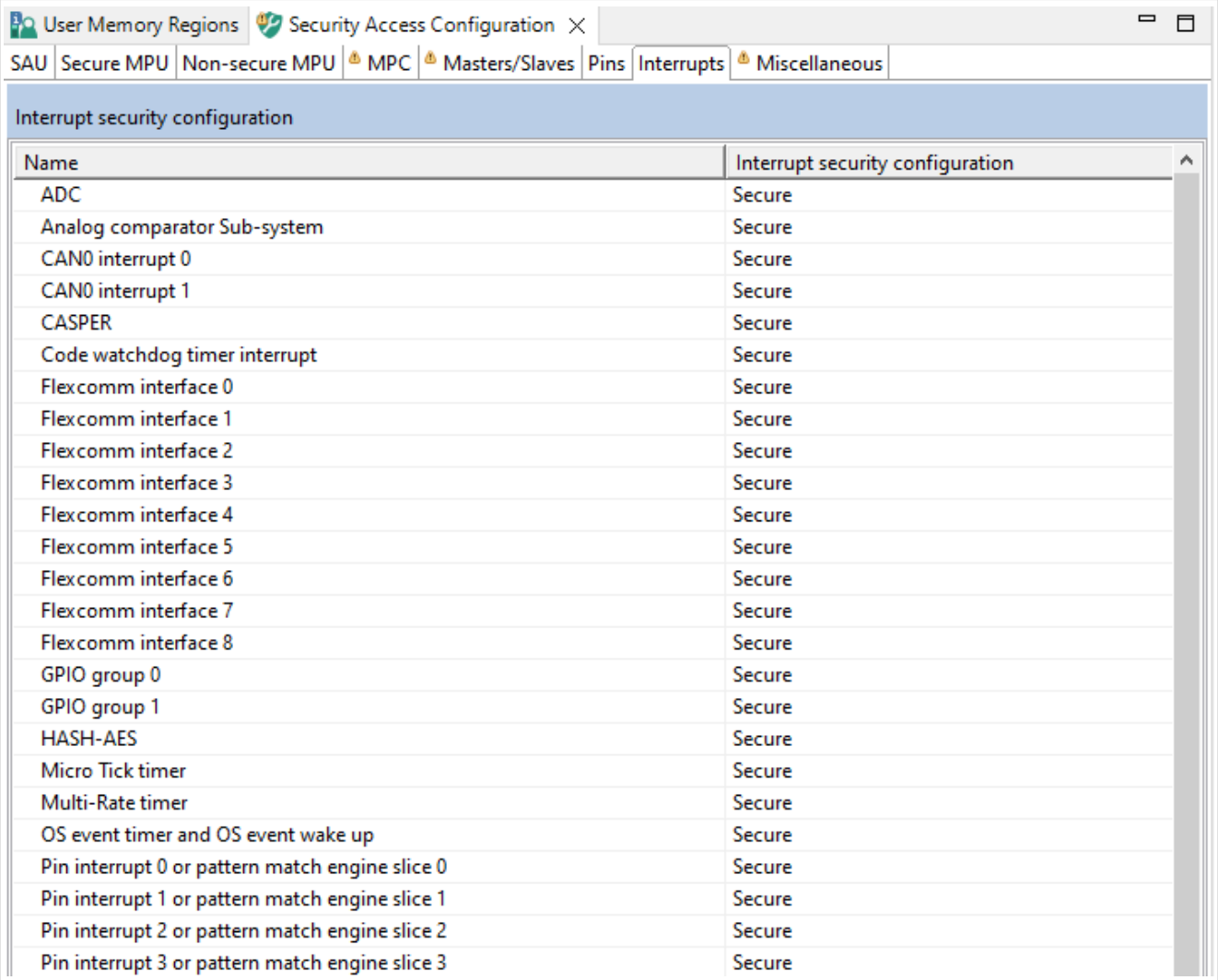


Figure 183. SAU/IDAU

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 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. 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 184. Interrupts

7.1.2.3 Secure/Non-secure MPU

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

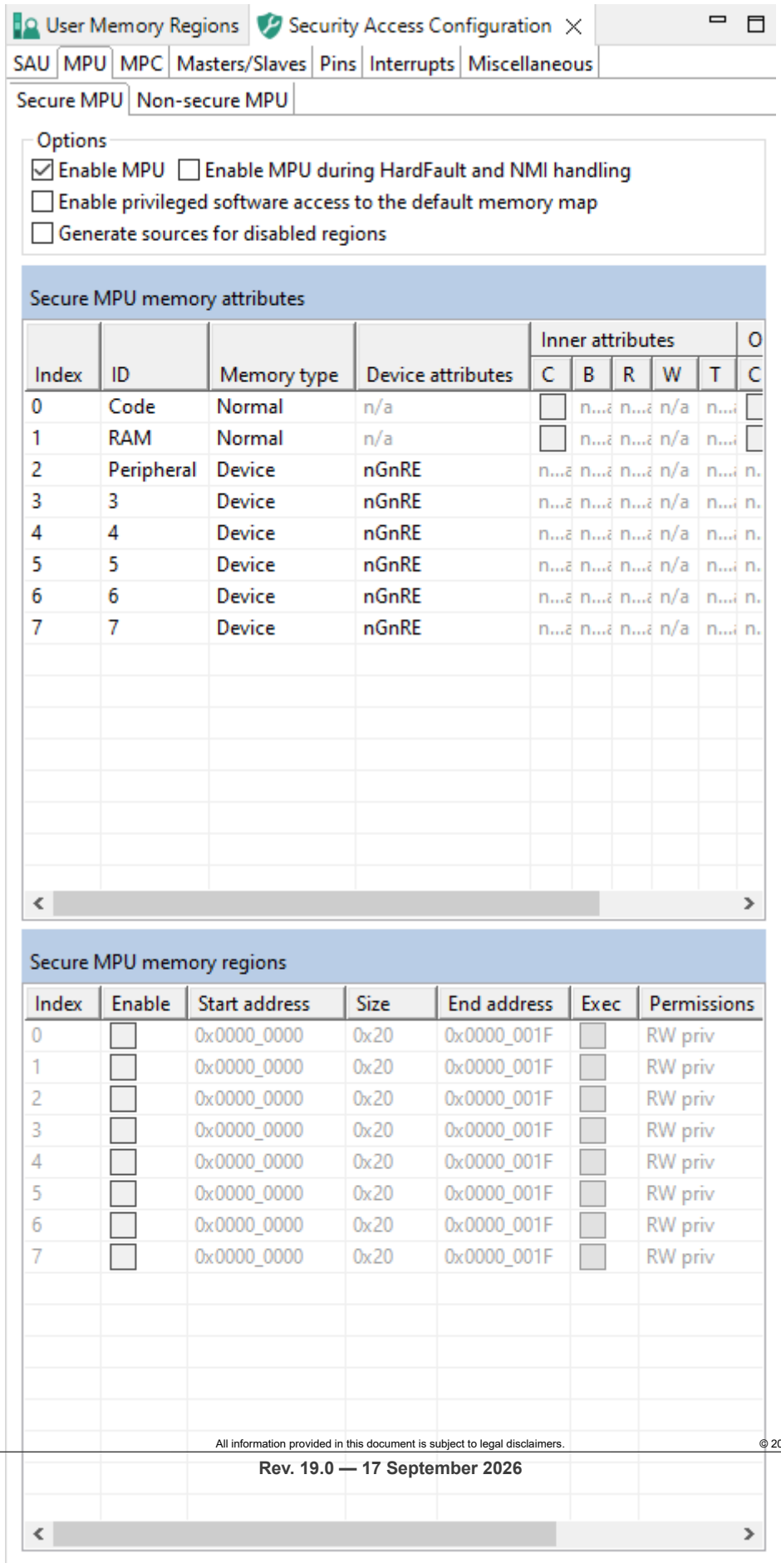


Figure 185. MPU

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 to allow the region 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

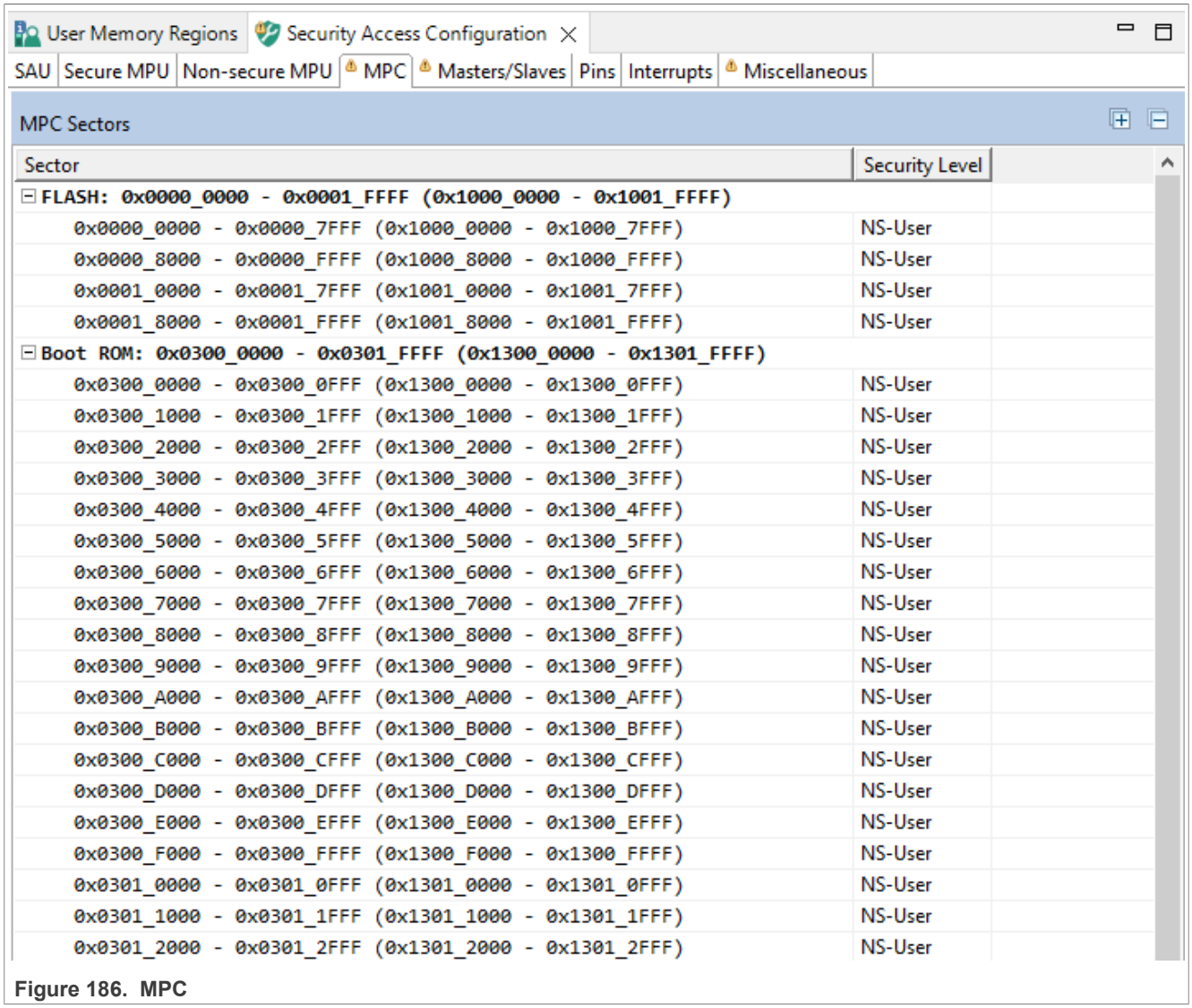


Figure 186. MPC

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

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

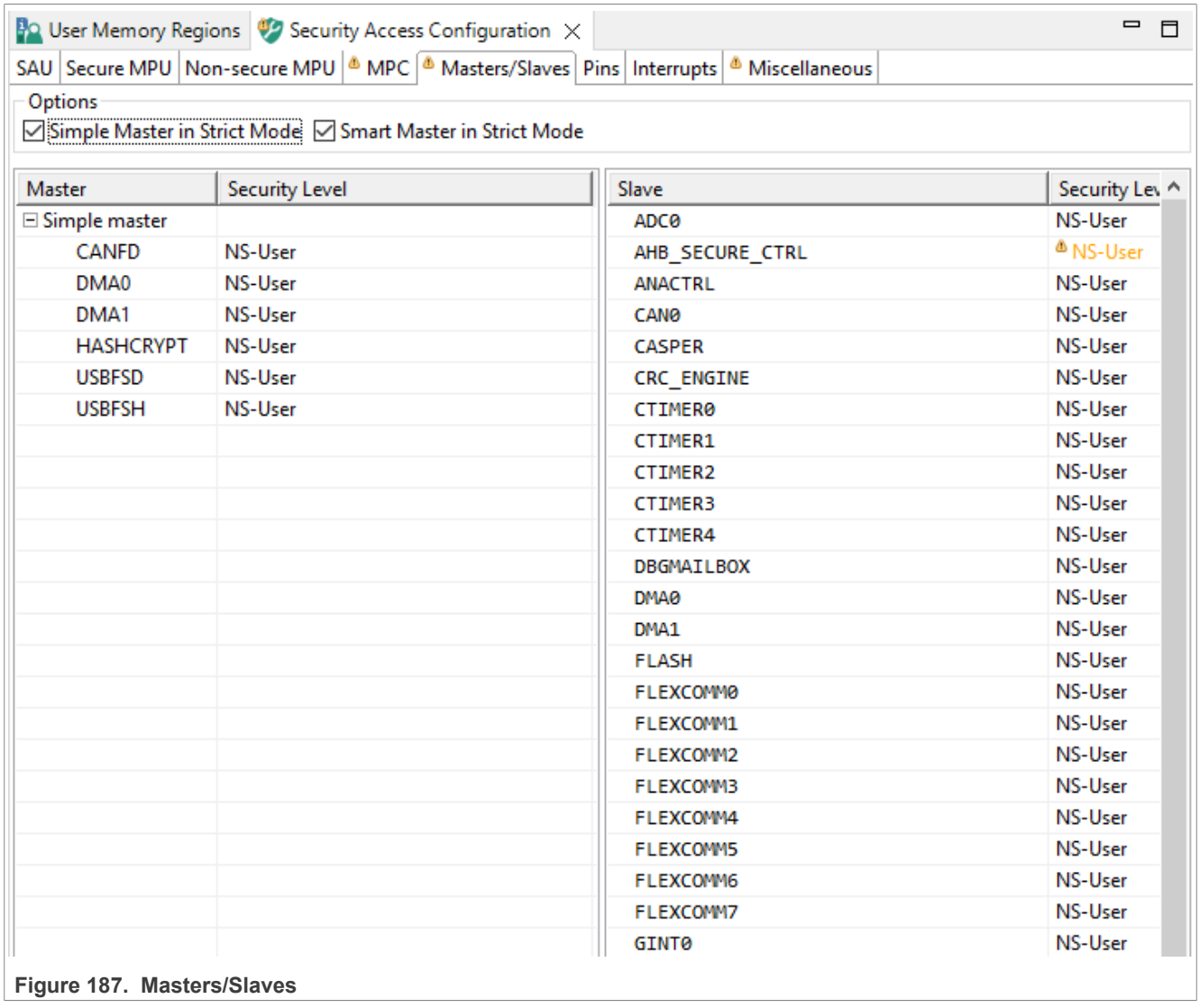


Figure 187. Masters/Slaves

7.1.2.6 Pins

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

All pins' reading GPIO state is set to **Allow** by default. To change the reading GPIO state of a pin, left-click the **Reading GPIO state** cell 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. To expand the selection, use **Shift+Up/Down** after selecting the row.

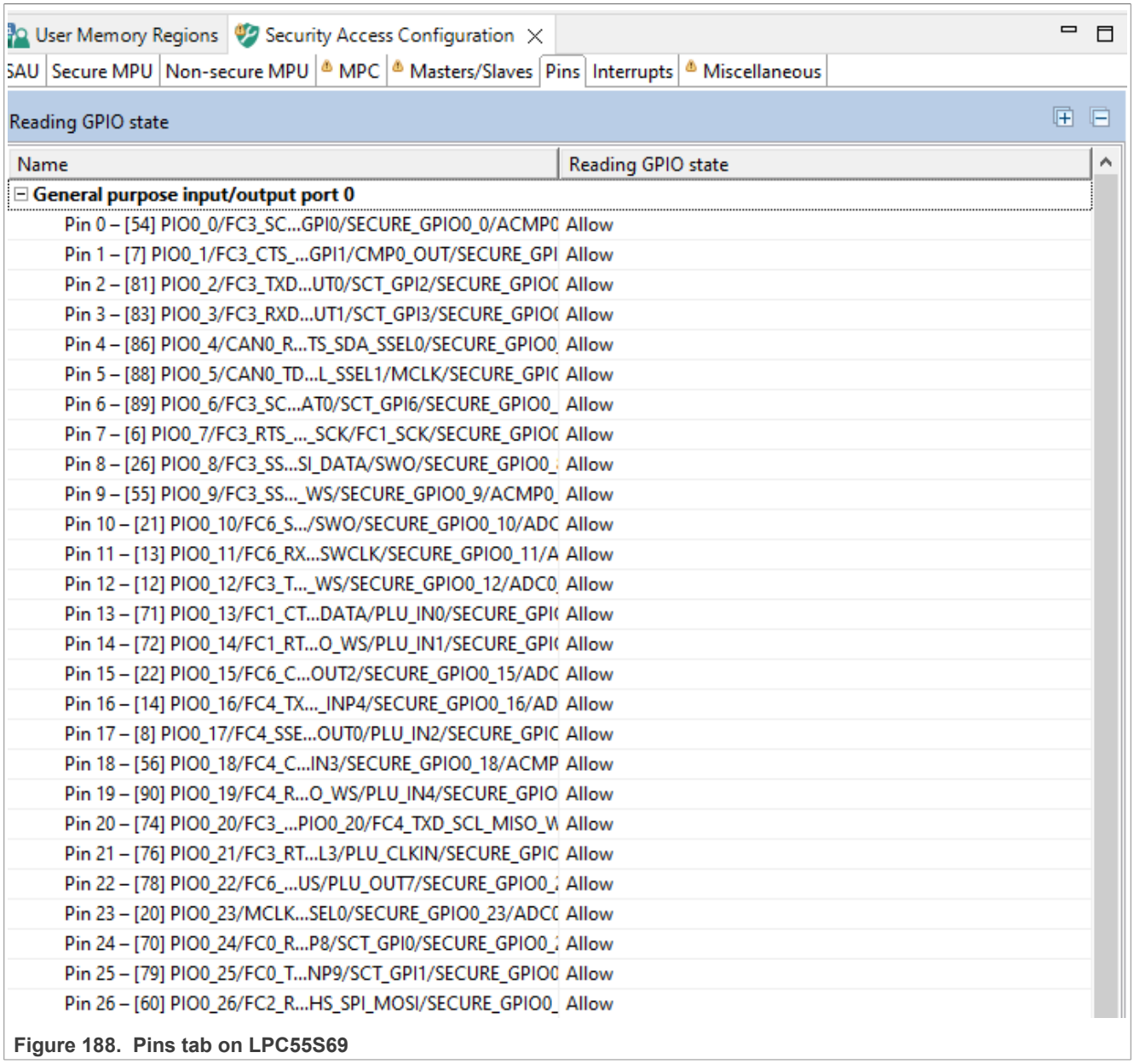


Figure 188. Pins tab on LPC55S69

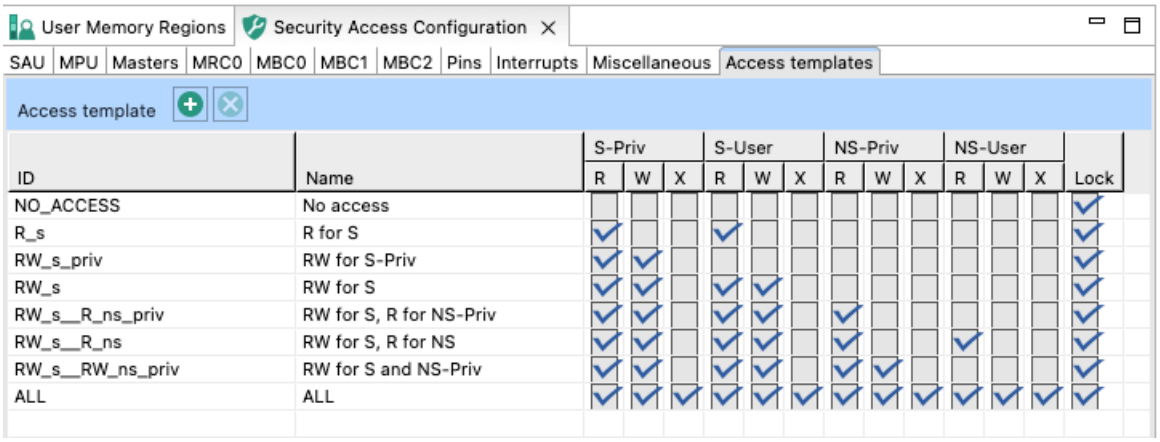


Figure 189. Global Access Templates

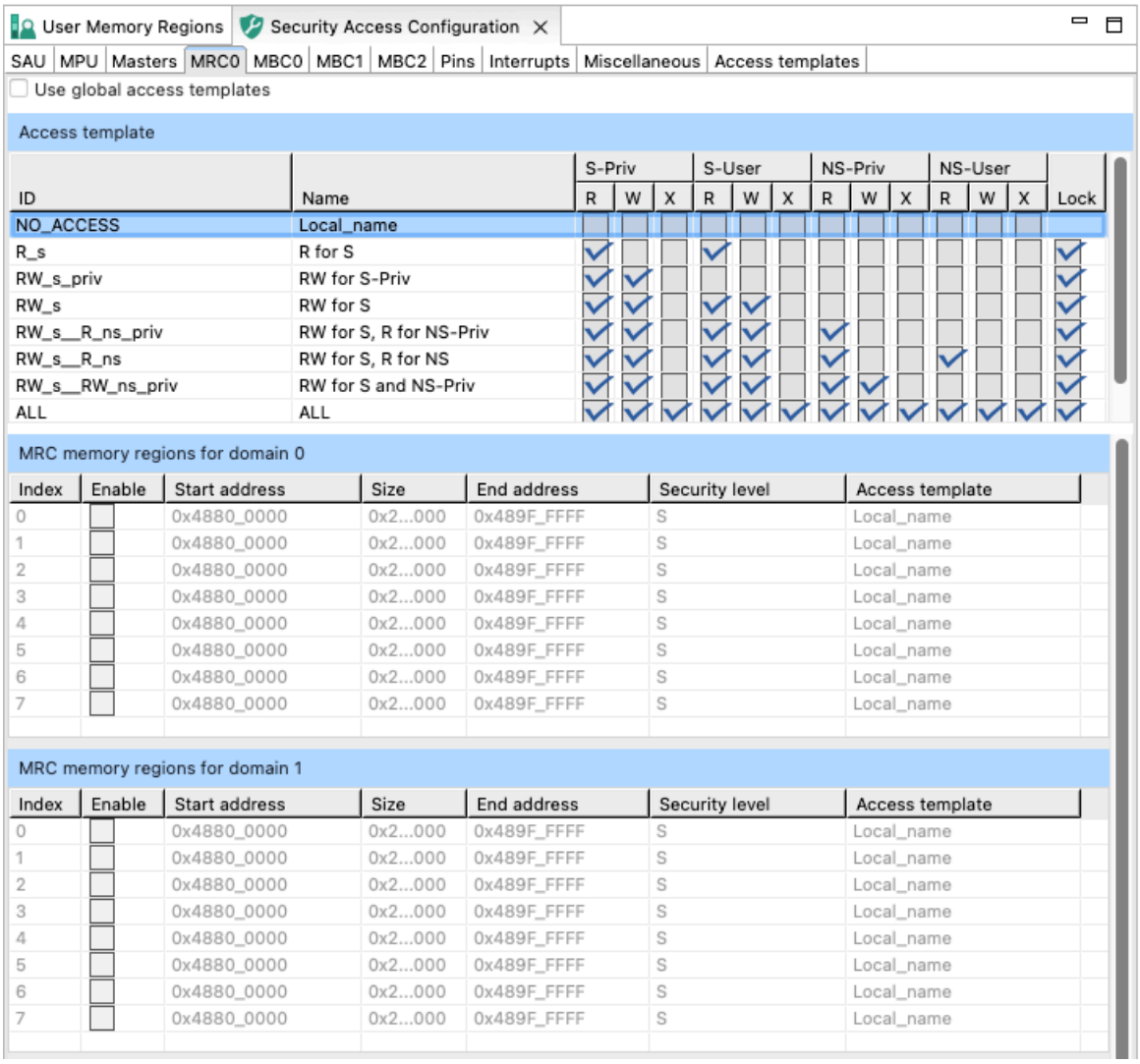


Figure 190. Local access templates

7.1.2.7 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 the configuration in the **Security Access Configuration** view. Point your cursor over individual options to display a tooltip explaining the function of each option.

A togglable checkbox enables or disables code generation for the entire group. When the group is disabled, code generation for that group is suspended, the generation options within it cannot be edited, and all option configurations revert to their default values (either reset values or default values).

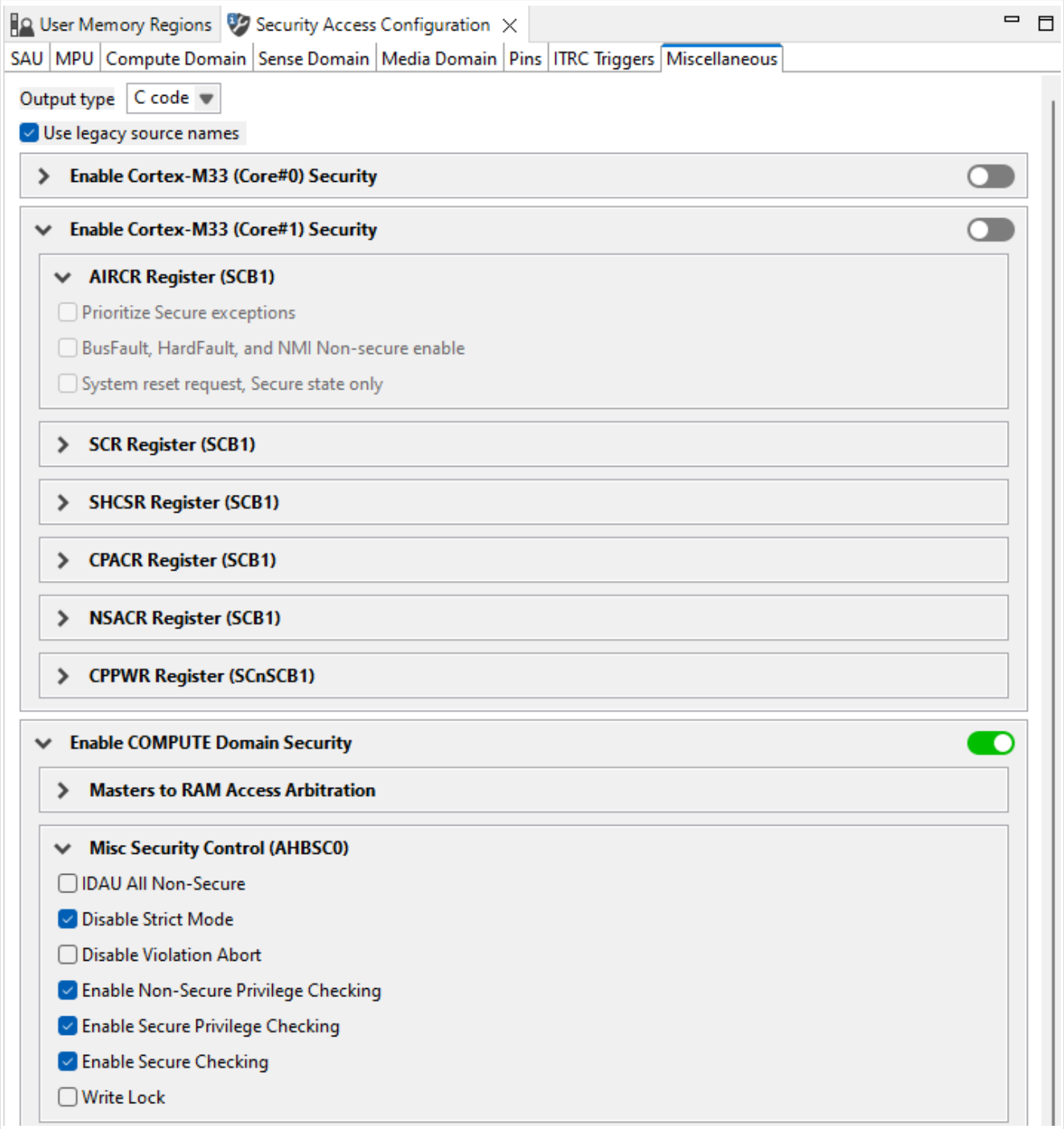


Figure 191. Miscellaneous

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 the 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**.

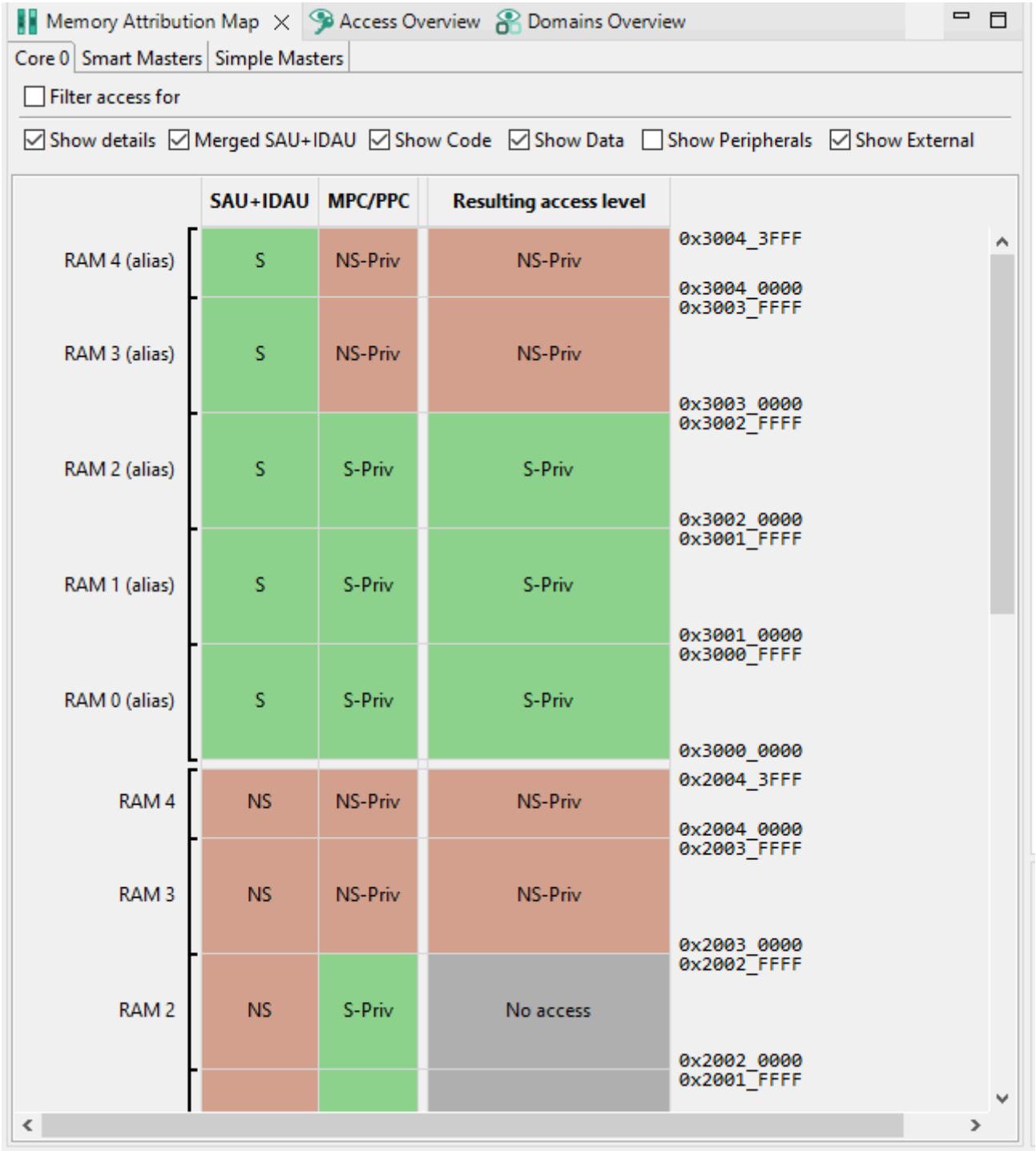


Figure 192. Core 0

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**.

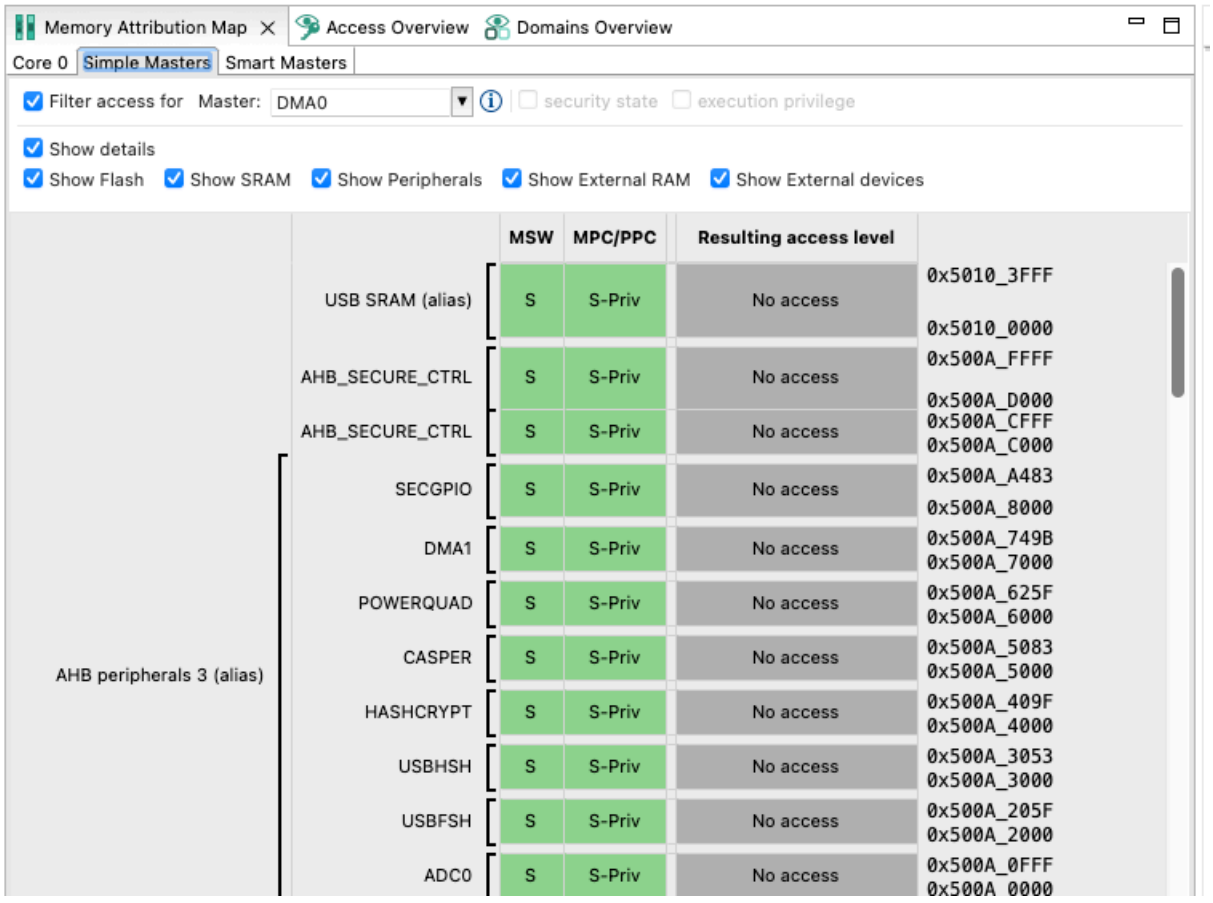


Figure 193. Simple/Smart masters

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 using the button on the right-hand side of the toolbar.

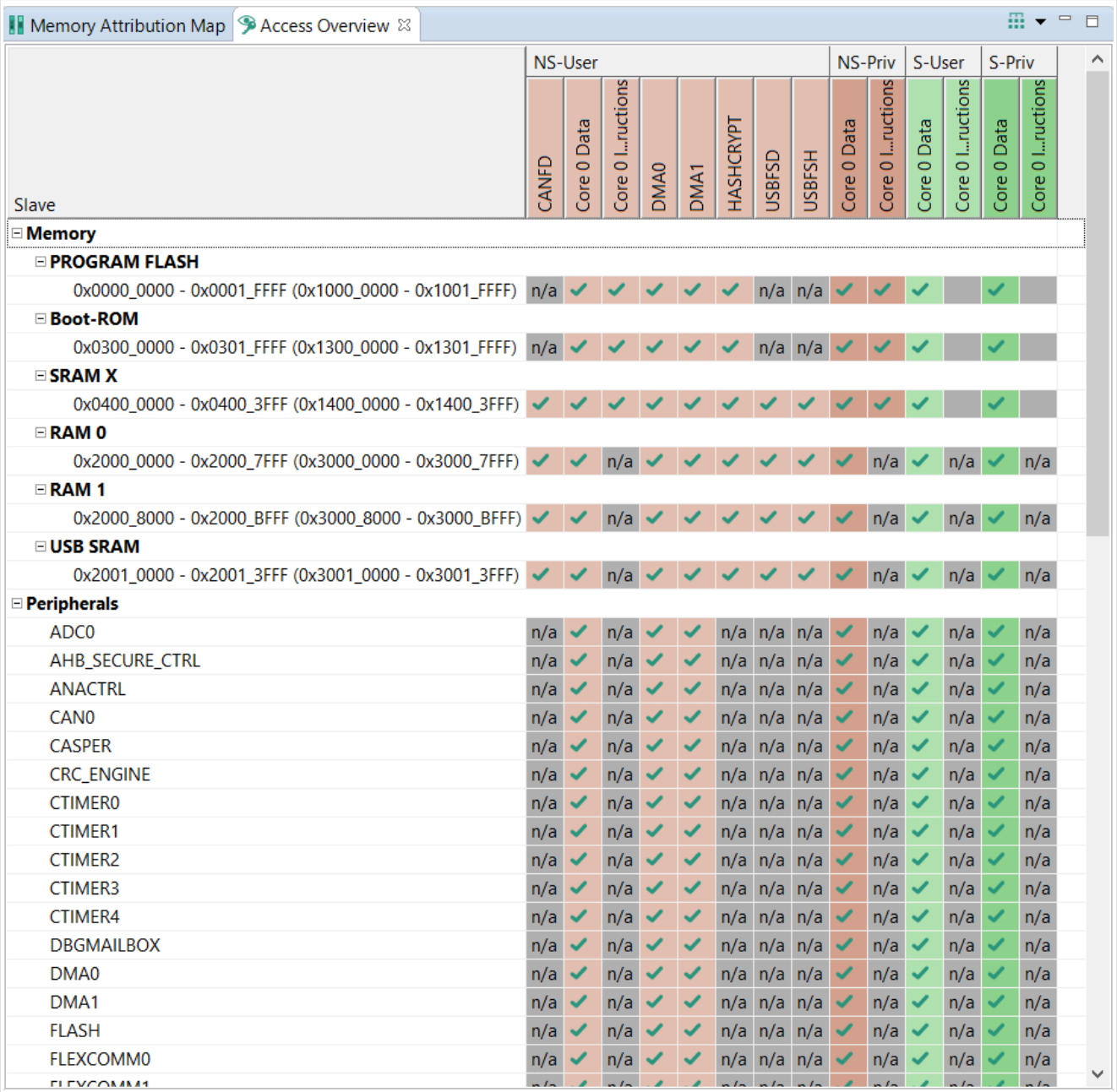


Figure 194. Access Overview

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 or 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), XRDC2 (eXtended Resource Controller 2), and TrustZone-M with TRDC.

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
- Kinetis W
  - KW45B41Z
  - KW45B410
  - KW47B42Z
  - KW47B420
- i.MX RT
  - MIMXRT1181
  - MIMXRT1182
  - MIMXRT1187
  - MIMXRT1189
- MCXW
  - MCXW716A
  - MCXW716C
- i.MX 91
  - MIMX930x
  - MIMX931x
  - MIMX933x
  - MIMX935x

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, and specify their address, size, security level, and description. You can then fix any errors in the settings using the **Problems** view.

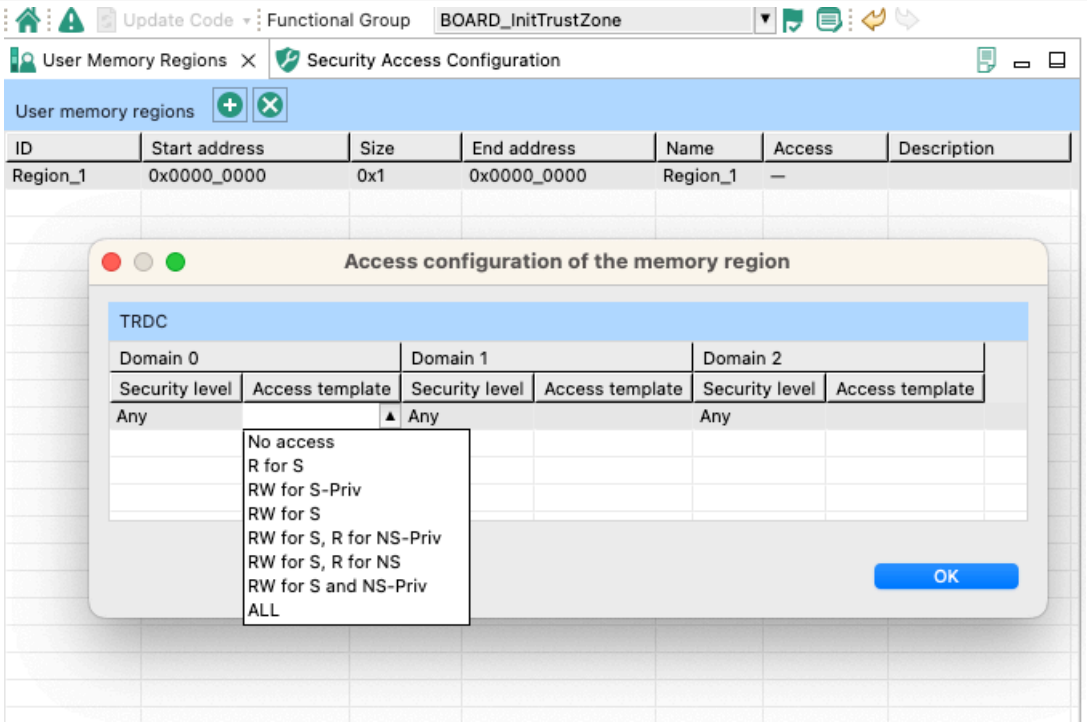


Figure 195. User Memory Regions

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

Enter or change the memory region parameters by clicking the row 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 is easily fixed, right-click the cell to display a dropdown list of solutions.

Remove the memory region by selecting the table row and clicking the **Remove selected memory region(s)** button in the view 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: First specify the number of domains in **M4 Domain/M7 Domain > Domains**.

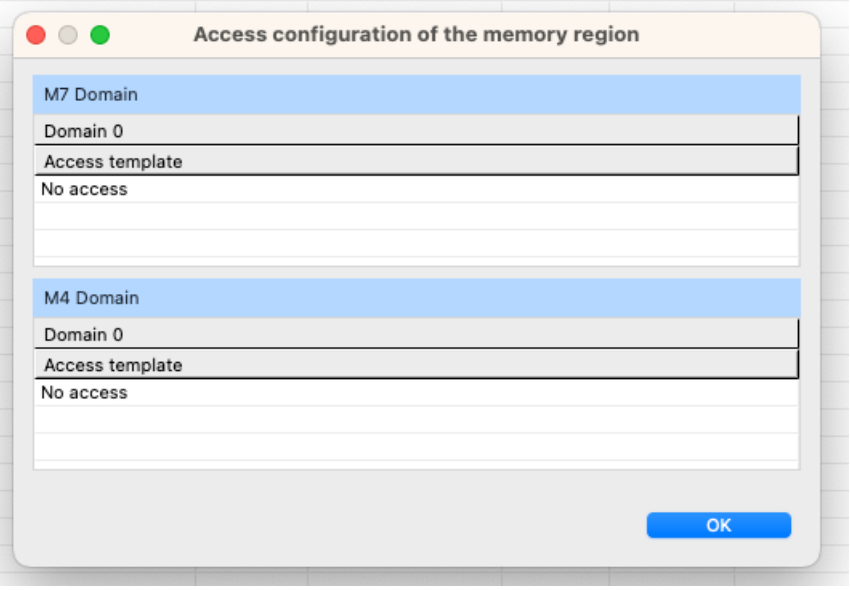


Figure 196. Access template

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 several 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.

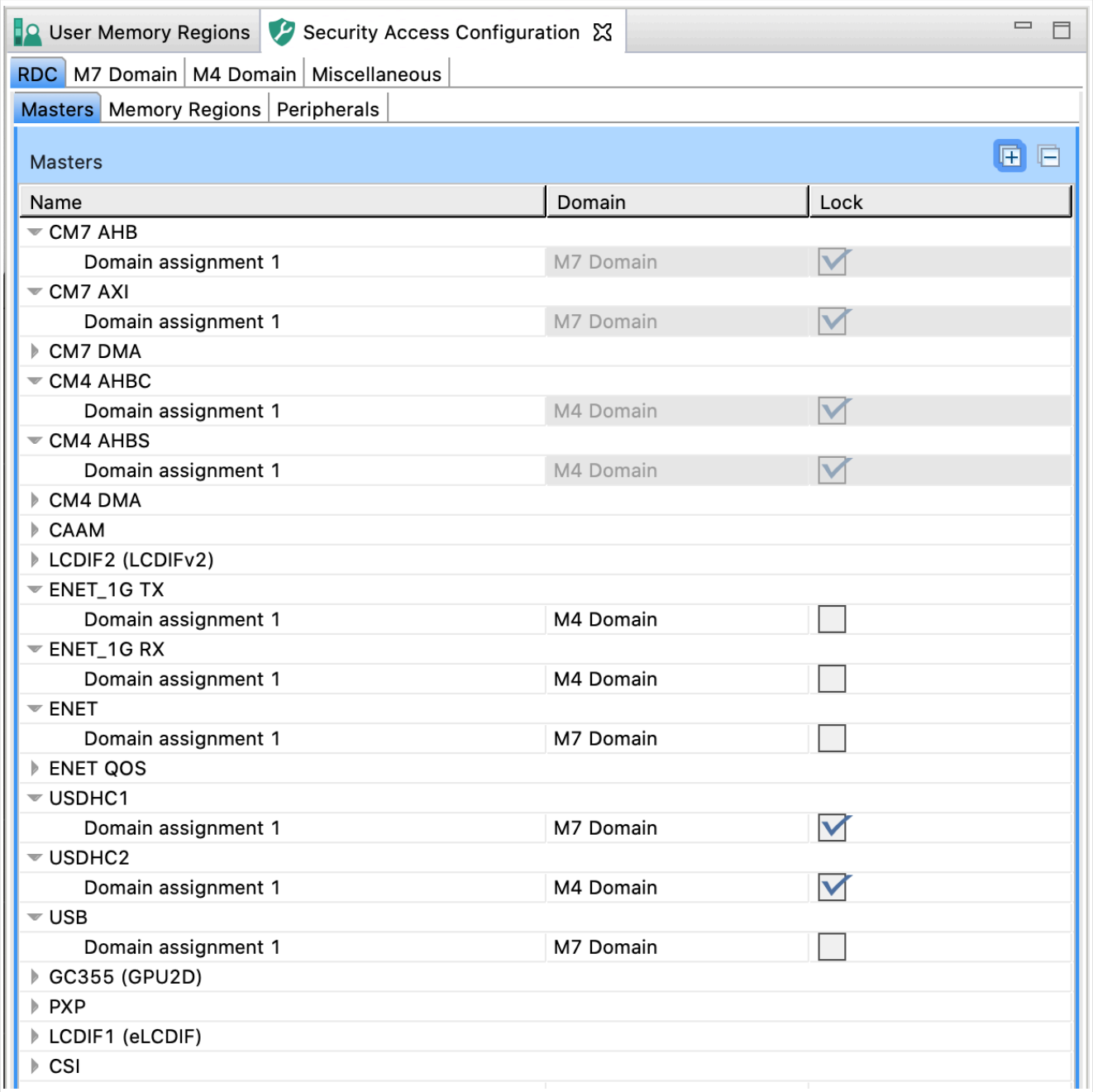


Figure 197. RDC Masters

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.

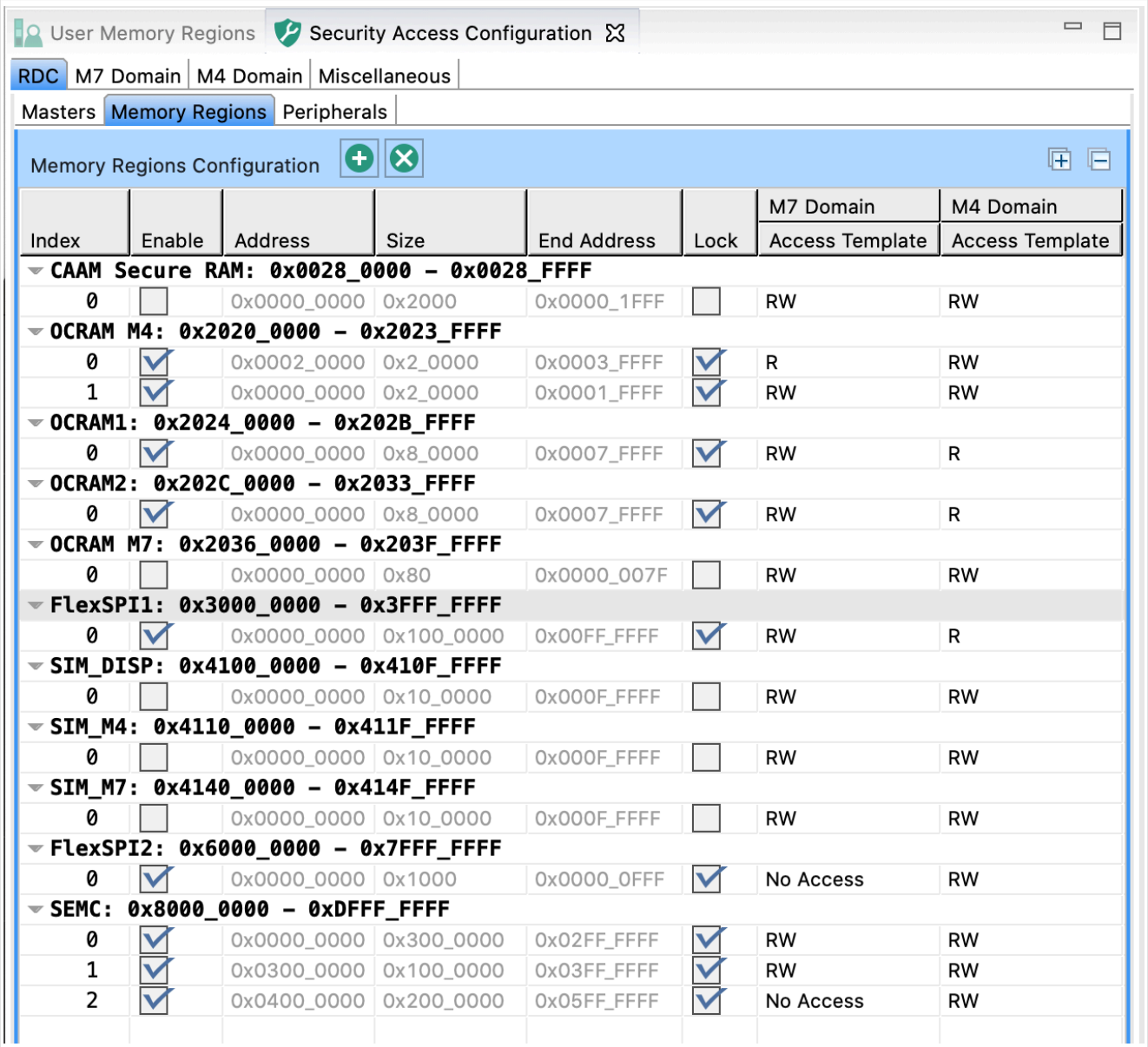
Alternatively, you can select the options by right-clicking the master and using the dropdown list.

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.

The 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 198. Memory Regions

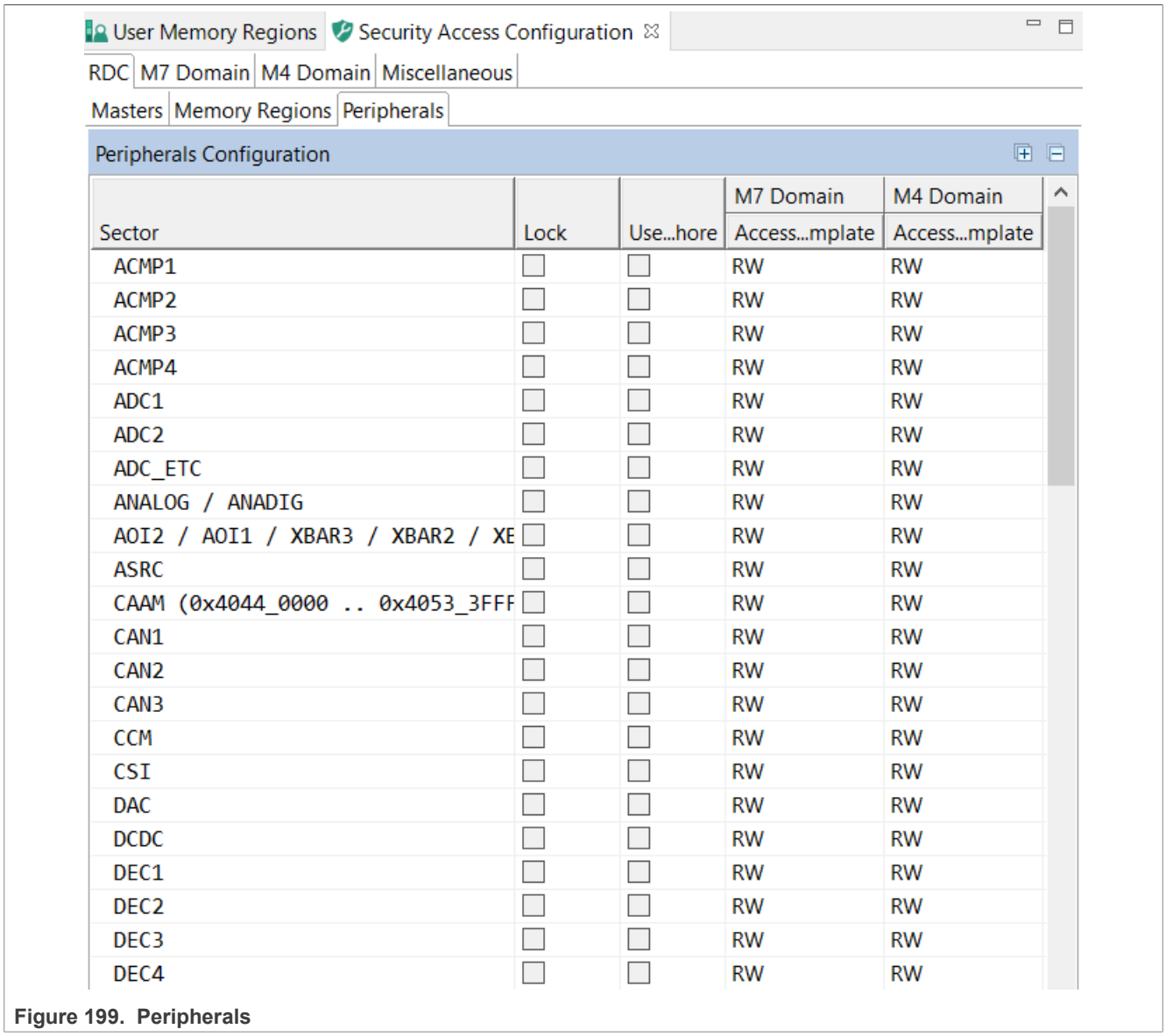
Use the **Memory Regions Configuration** table to enable and configure MRC slaves:

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 199. 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 it obtains a semaphore. While the domain holds the semaphore, 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. Enable it by selecting the **Enable MPU** option.

Note: Not every device supports MPU.

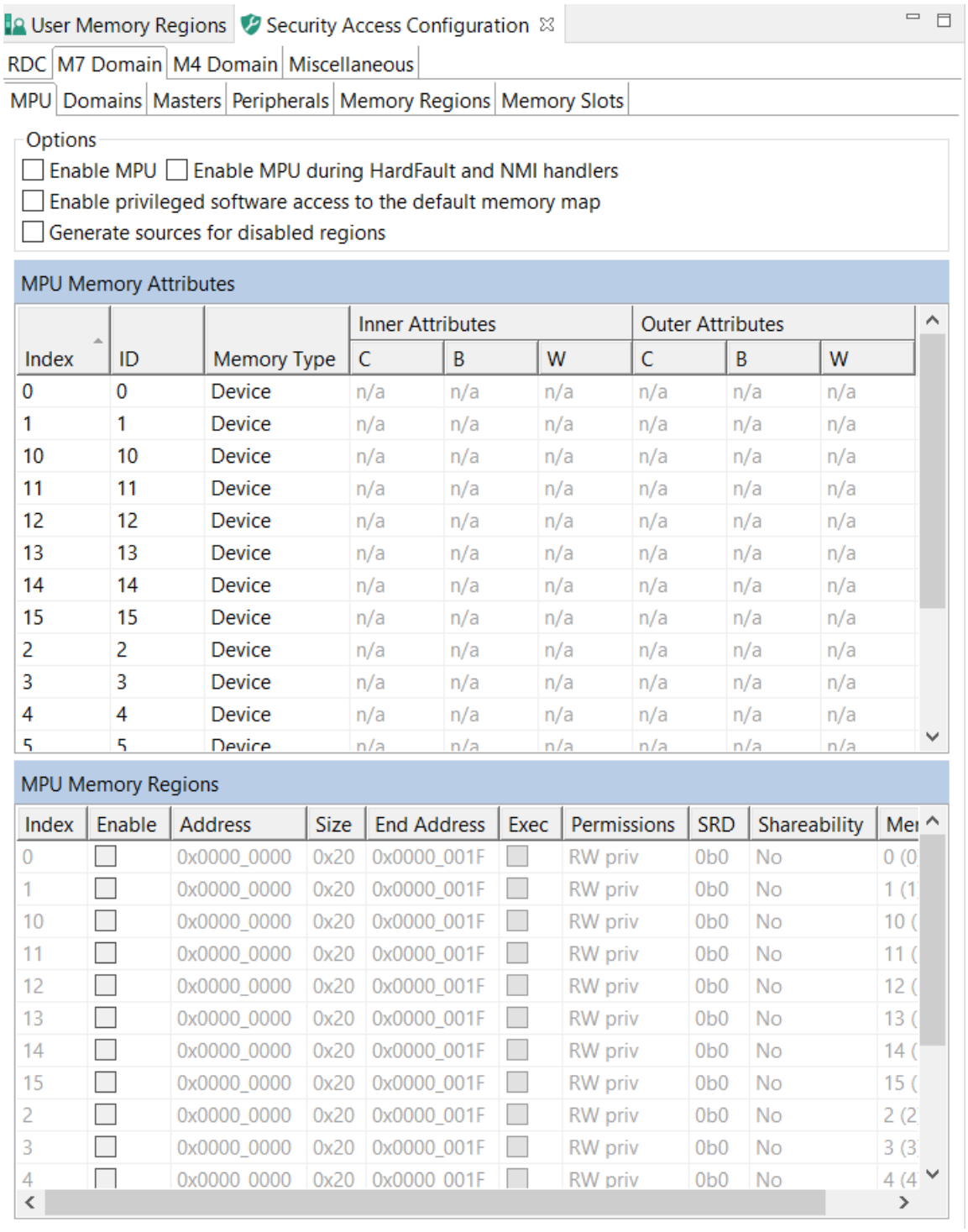


Figure 200. MPU

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 to allow the region 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.

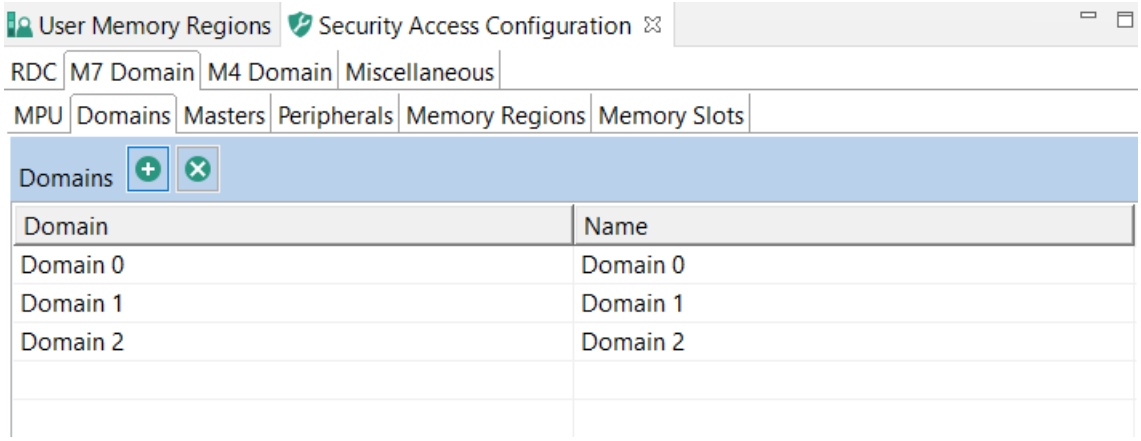


Figure 201. Domains

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.

7.2.2.2.3 Masters

In the **Masters** subview, you can view, add/remove, and 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.

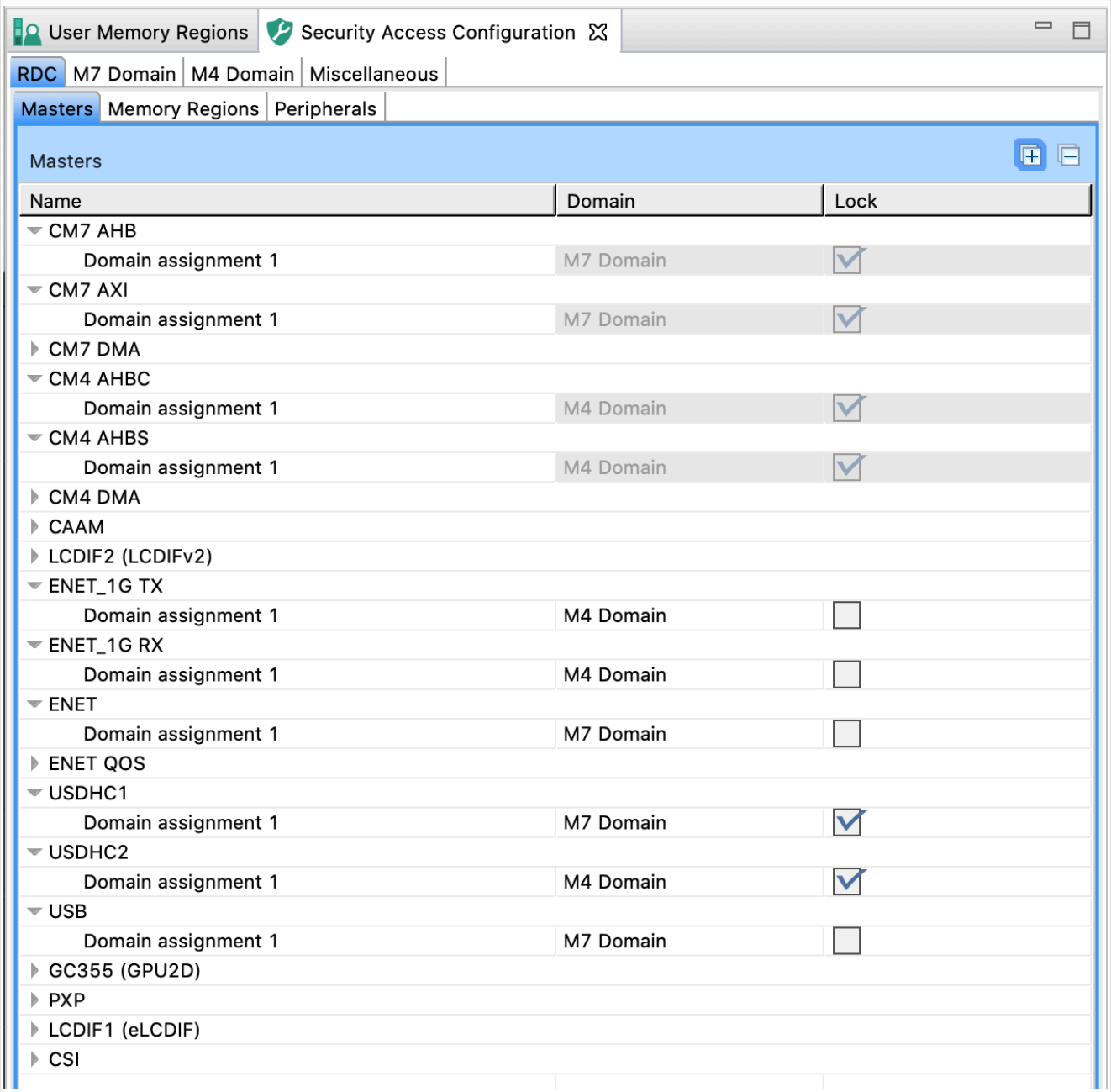


Figure 202. Masters

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.

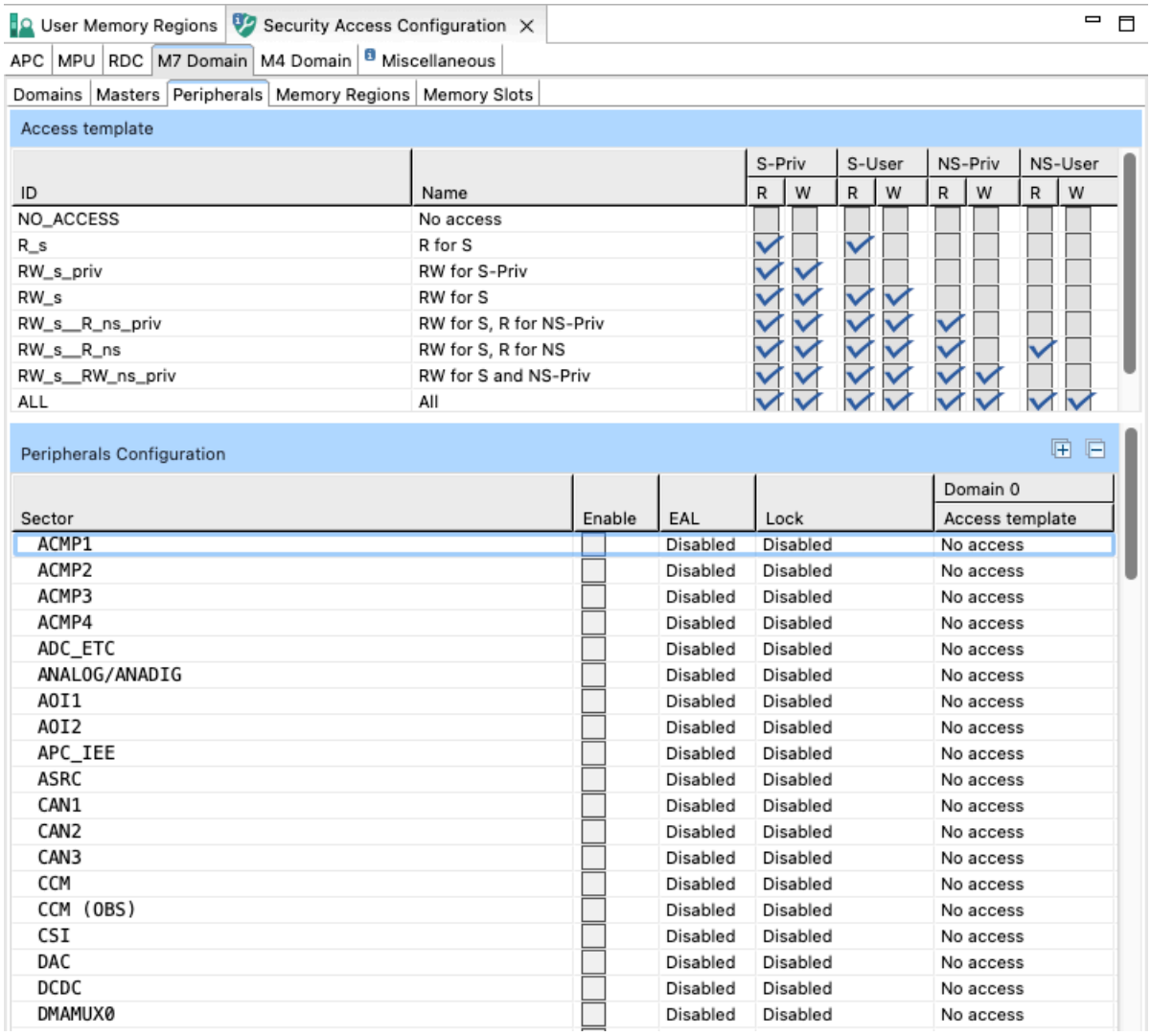


Figure 203. Peripherals

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, right-click the master and select the options from 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.

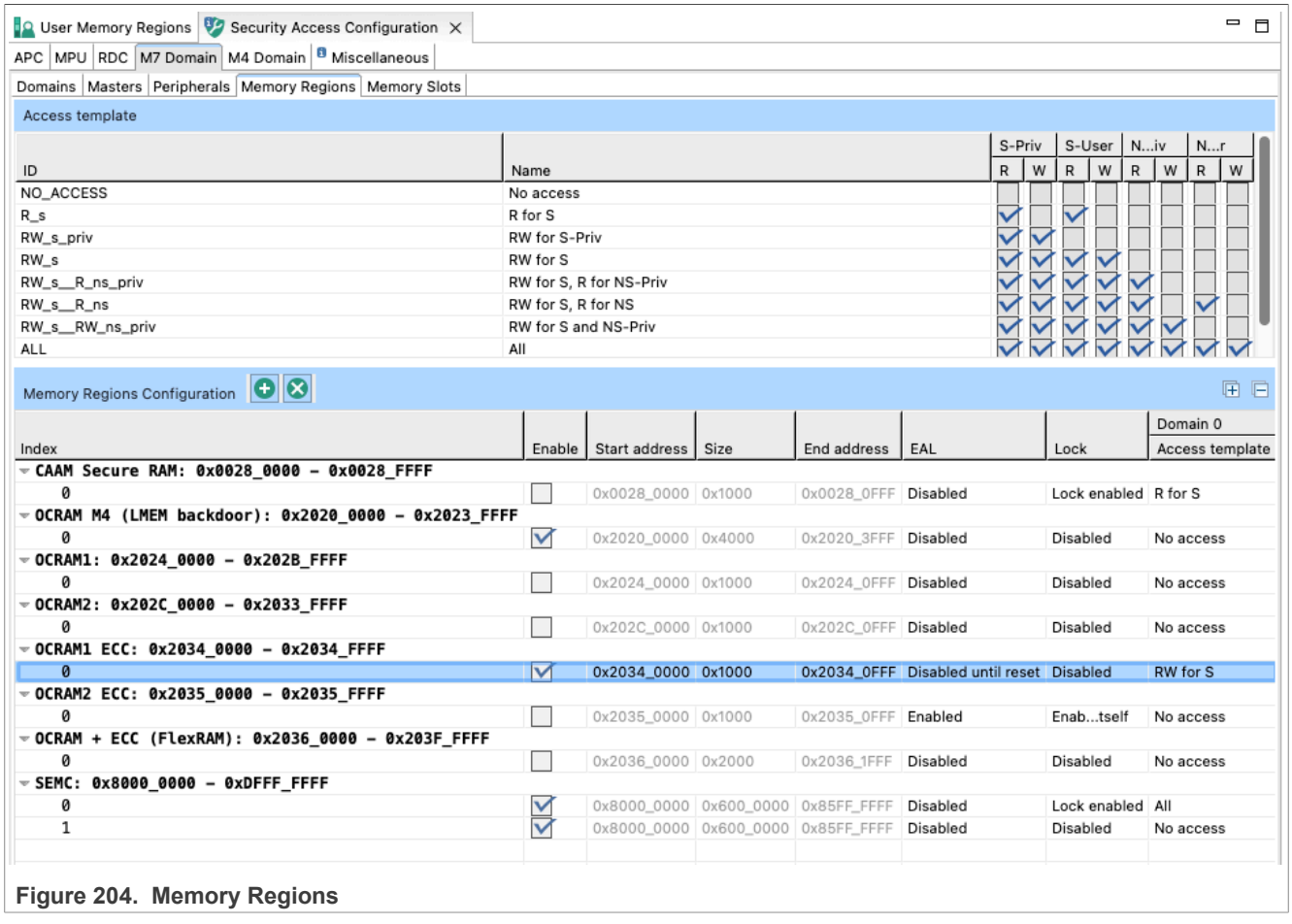


Figure 204. Memory Regions

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, right-click the master and select the options from 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.

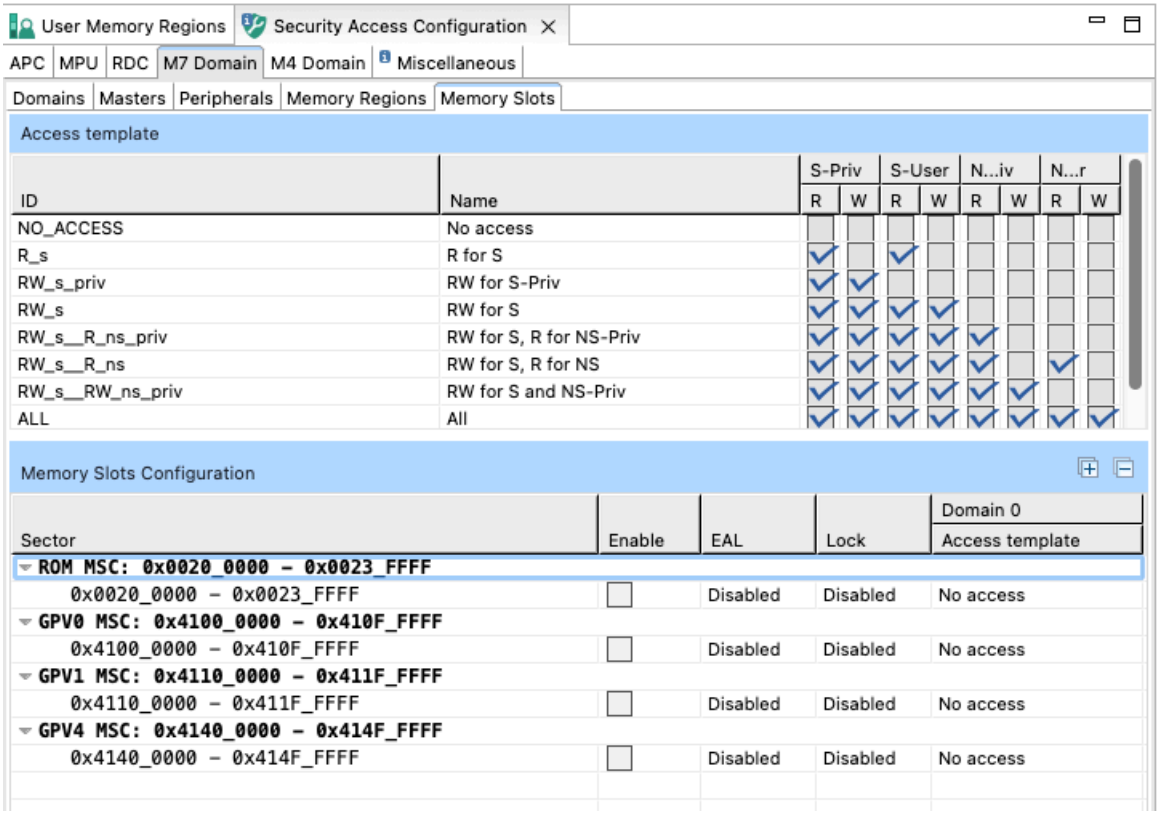


Figure 205. Memory Slots

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 XRDC (eXtended Trusted Resource Domain Controller) on Cortex-A35 in i.MX8 ULP

7.2.2.3.1 Masters

XRDC masters are similar to TRDC masters. In addition, the following features are supported:

- **PID (Process Identifier)** is combined with the PIDM field to determine the domain hit.

- **PIDM (PID Mask)** provides a masking capability so that multiple process identifiers can be included as part of the domain hit determination. If a bit in the PIDM is set, the corresponding bit of the PID is ignored in the comparison.
- **PID enable** provides the ability to include inclusive or exclusive sets of masked PID values. Allowed values are 00b, 01b, 10b, and 11b. For more info, see the corresponding Reference Manual.

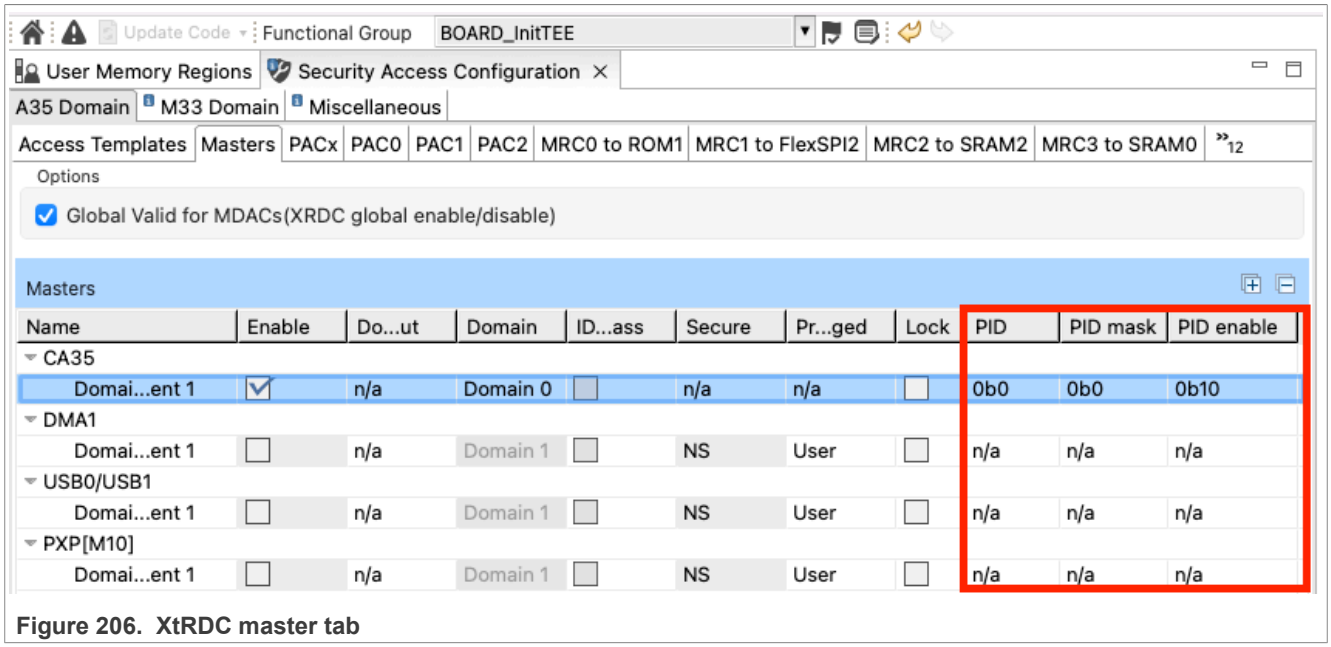


Figure 206. XtRDC master tab

7.2.2.3.2 MRC

MRC on XRDC is similar to MRC on TRDC. There are several minor differences:

1. There is only one instance of the memory regions table because address ranges are shared across all domains. For each memory region, specify an access template for each domain.
2. The code region specifies which templates would be used (0= data, 1 = code). The templates are now hybrid. It means that there are two templates for the same ID and name - the first row is for the data region and the second row is for the code region. These templates, which have the lock field, can be edited by clicking the desired access box.

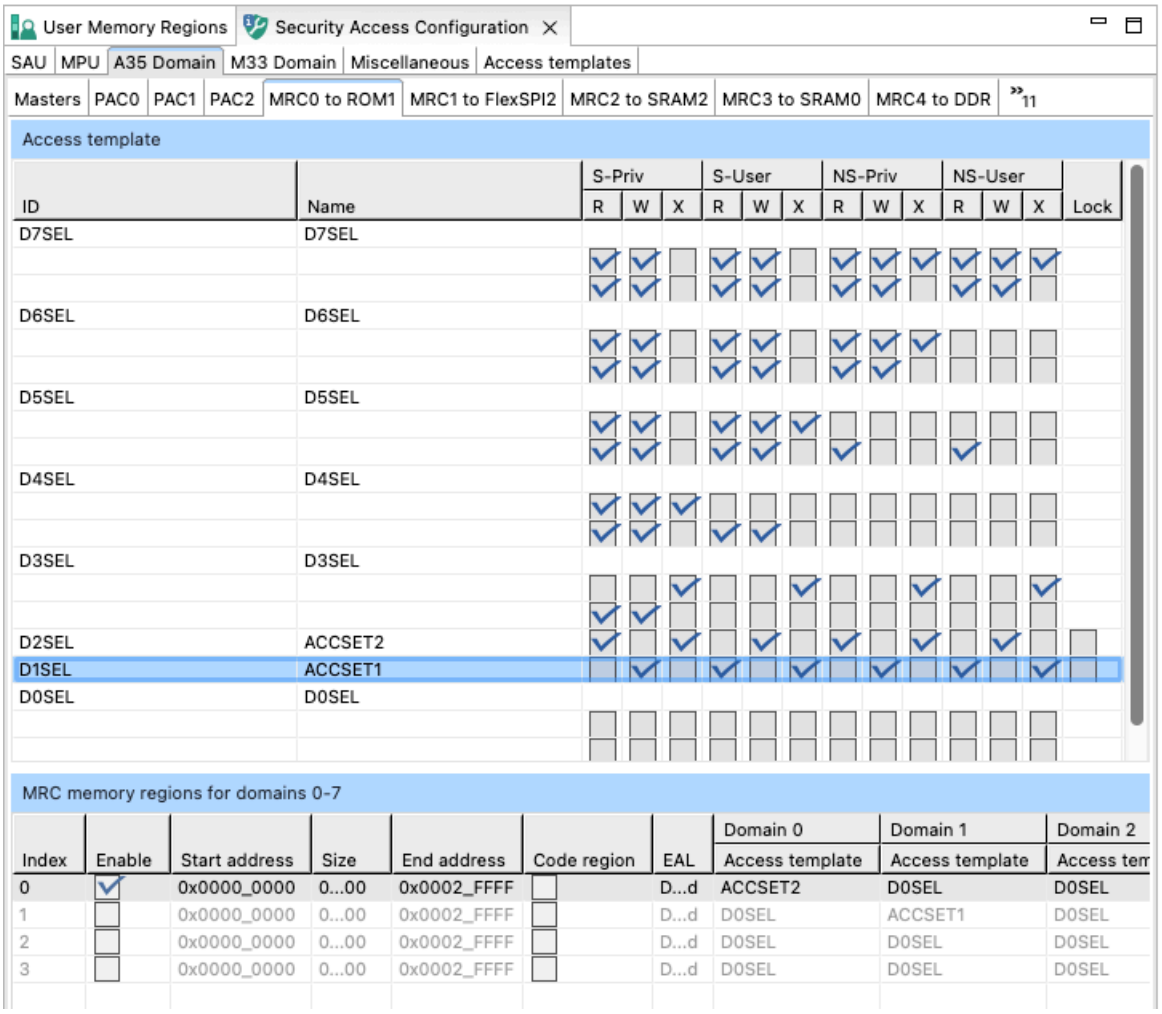


Figure 207. XtRDC MRC tab

7.2.2.3.3 Access control modes

There are two modes that can be enabled for PID.

For processors only supporting TSM, the Three-State Model (SecurePriv, SecureUser, NonsecureUser), the nonsecure[n] output signal from the MDAC submodule is forced to zero while in privileged mode to enable precise state transitions between the user and privileged modes. When SP4SM, the Special 4-State Model, is enabled, the MDAC does not use the MDA[DIDS,DID] fields. The MDAC tracks the current access level and generates specific domainIDs for specific access levels.

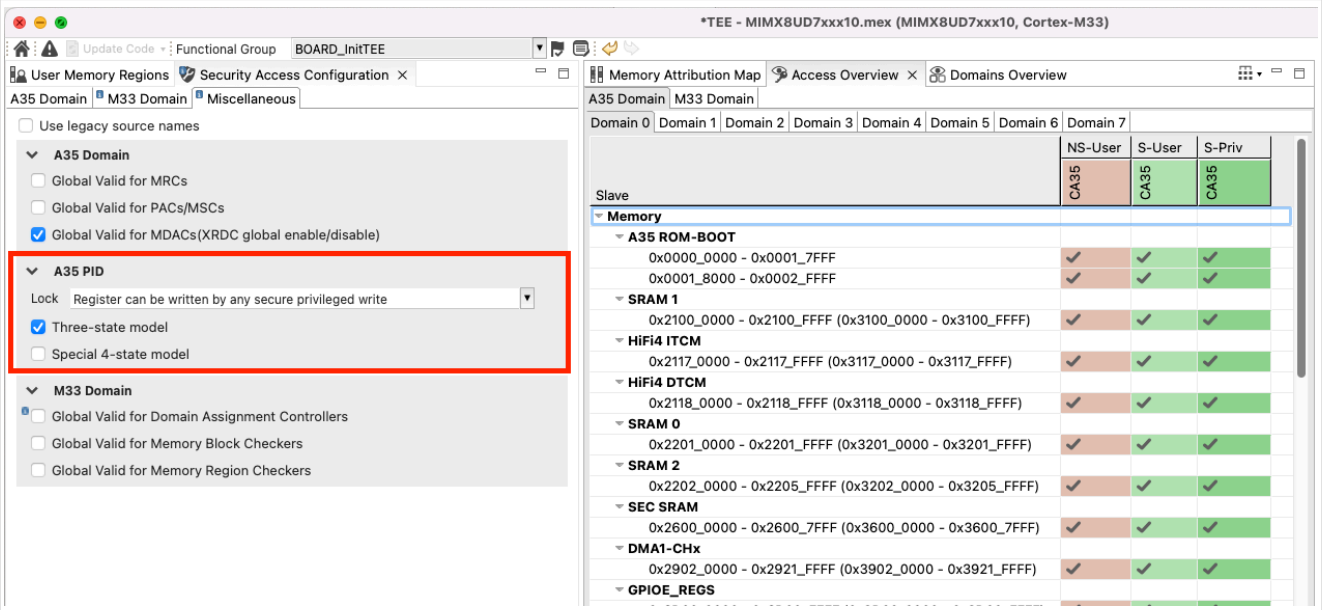


Figure 208. Access modes

7.2.2.4 Trusted Resource Domain Controller on Cortex-M33 in i.MX8 ULP and KW45 (TRDC)

The Trusted Resource Domain Controller (TRDC) provides comprehensive security management for Cortex-M33 based devices in i.MX8 ULP and KW45 processors. TRDC enables fine-grained access control through domain-based resource allocation, where chip resources are assigned to processing domains identified by unique domain identifiers (DIDs).

The TRDC configuration includes Memory Protection Unit (MPU) setup with Secure/Non-Secure register banks, domain management for resource assignment, master configuration with domain ID control, and access template management supporting both global (RDC-wide, editable) and local (checker-specific, immutable) templates. Memory access control is enforced through Memory Region Controller (MRC) for configurable memory regions and Memory Block Checker (MBC) for fixed memory blocks, providing comprehensive protection for both memory spaces and peripherals across different security domains.

7.2.2.4.1 MPU

This MPU is identical to other MPUs with Cortex-M33 (for example, LPC55S) or other cores based on the Armv8-M architecture or above with Secure/Non-Secure register banks.

7.2.2.4.2 Domains

The domains are similar to RDC/XRDC2/XRDC: assignment of chip resources to processing “domains”, where a unique domain identifier (domainID, DID) is assigned to each processing domain. The number of supported DIDs is typically the number of CPUs plus one.

7.2.2.4.3 Masters

Masters are similar to Masters in XRDC2 on MIMXRT117x. The user can also choose the domain ID input or ID bypass depending on the master type.

7.2.2.4.4 Access templates

Access templates are similar to patterns in XRDC2 on MIXRT117x. The main difference is as follows: you can switch between “global” (for the entire RDC, used by all checkers, and editable) and “local” (specific to the checker and immutable) templates; meanwhile access templates in XRDC2 are always validator-dependent and editable.

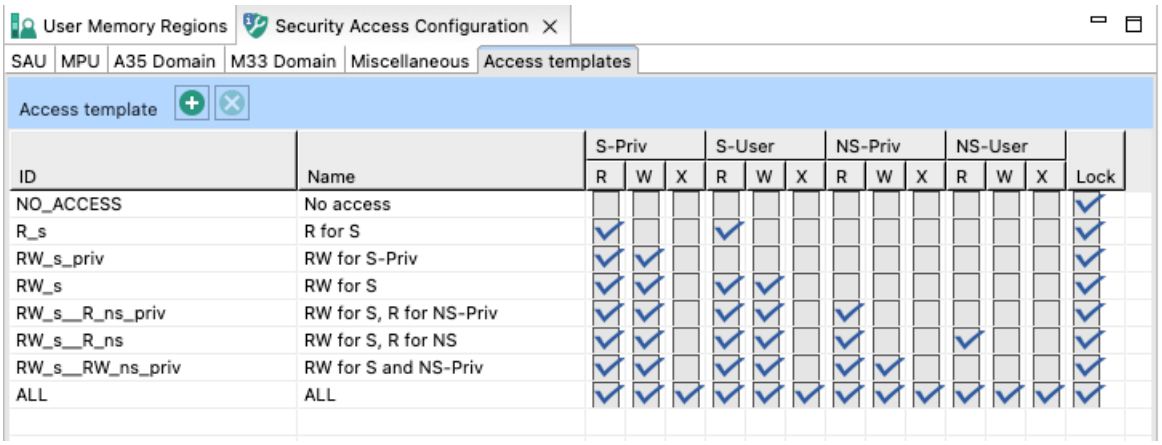


Figure 209. Access templates

7.2.2.4.5 MRC

MRC on TRDC is similar to MRC (Memory Regions) in XRDC2.

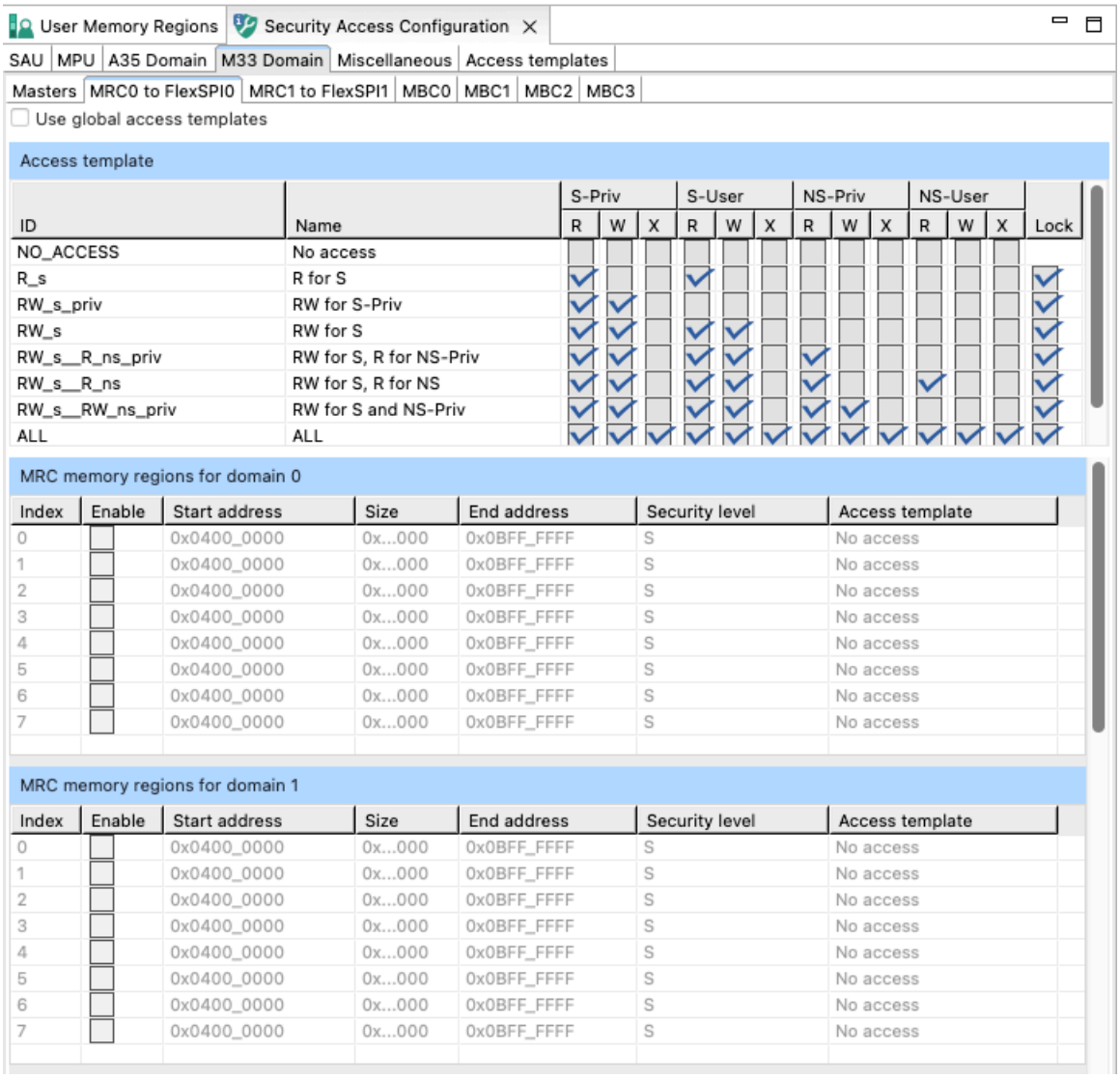


Figure 210. MRC

7.2.2.4.6 MBC

MBC in TRDC is similar to MSC (Memory Slots) in XRDC2 and MSC in XRDC.

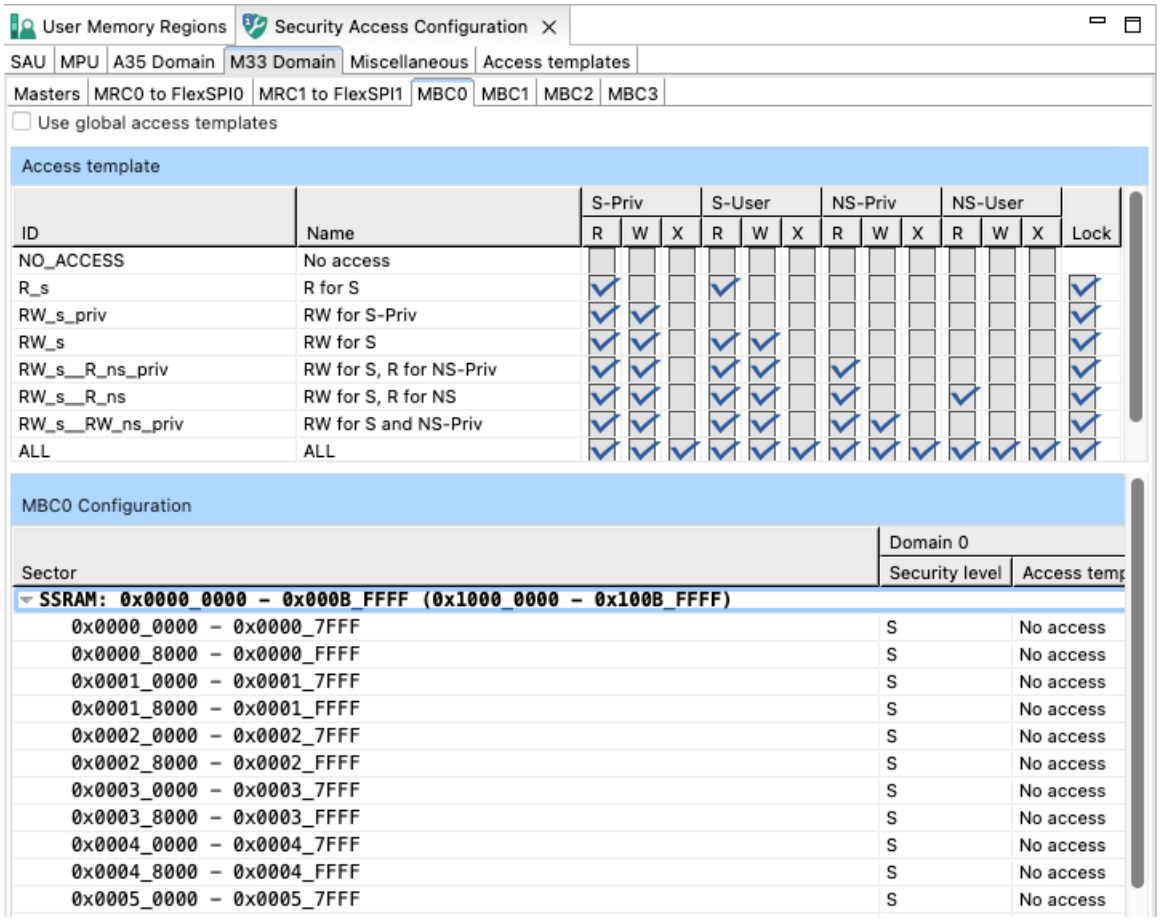


Figure 211. MBC

7.2.2.5 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.

A togglable checkbox enables or disables the code generation for the entire group. When the group is disabled, the code generation for that group is suspended, the generation options within it cannot be edited, and all option configurations revert to their default values (either reset values or default values).

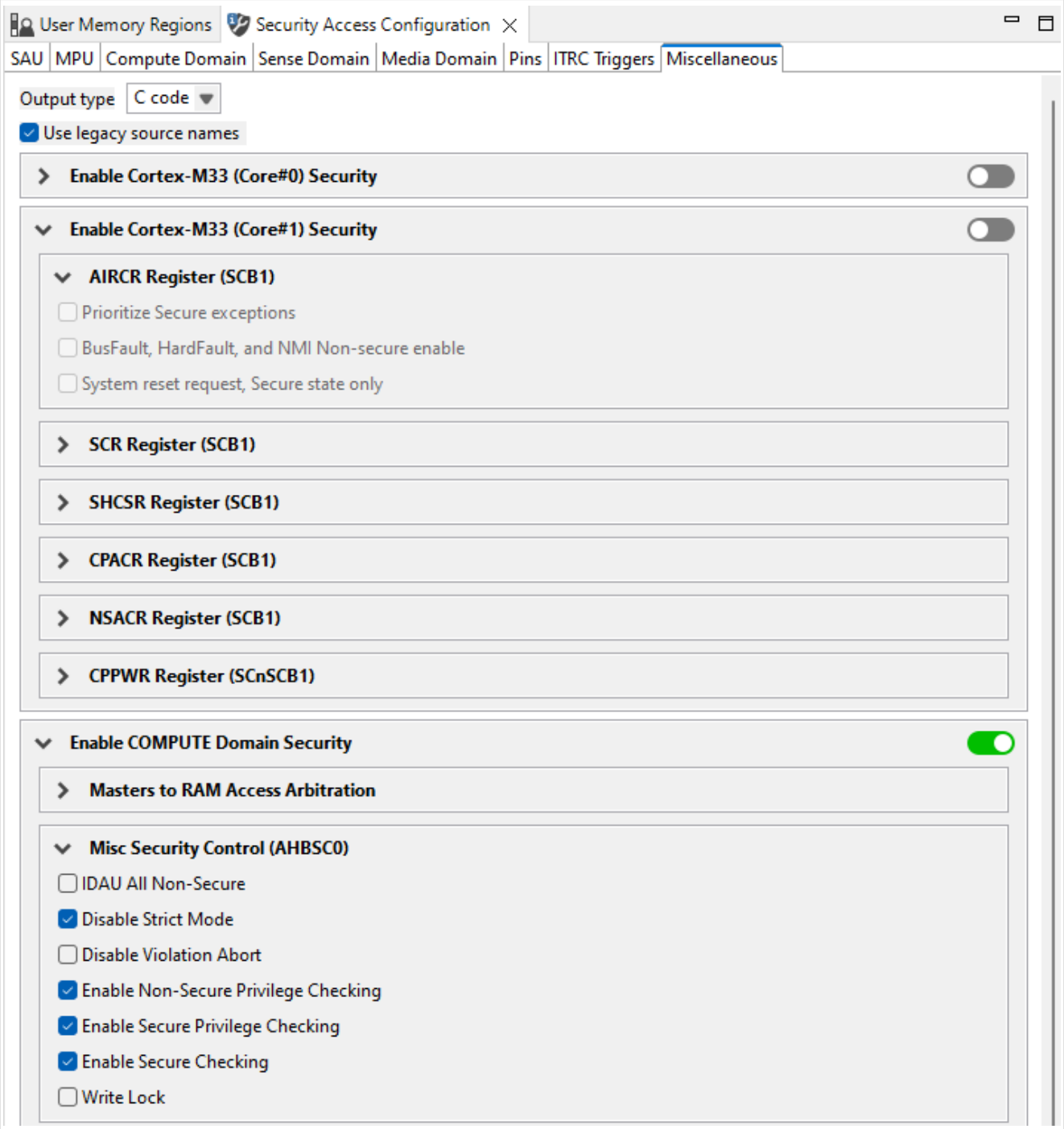


Figure 212. Miscellaneous

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.

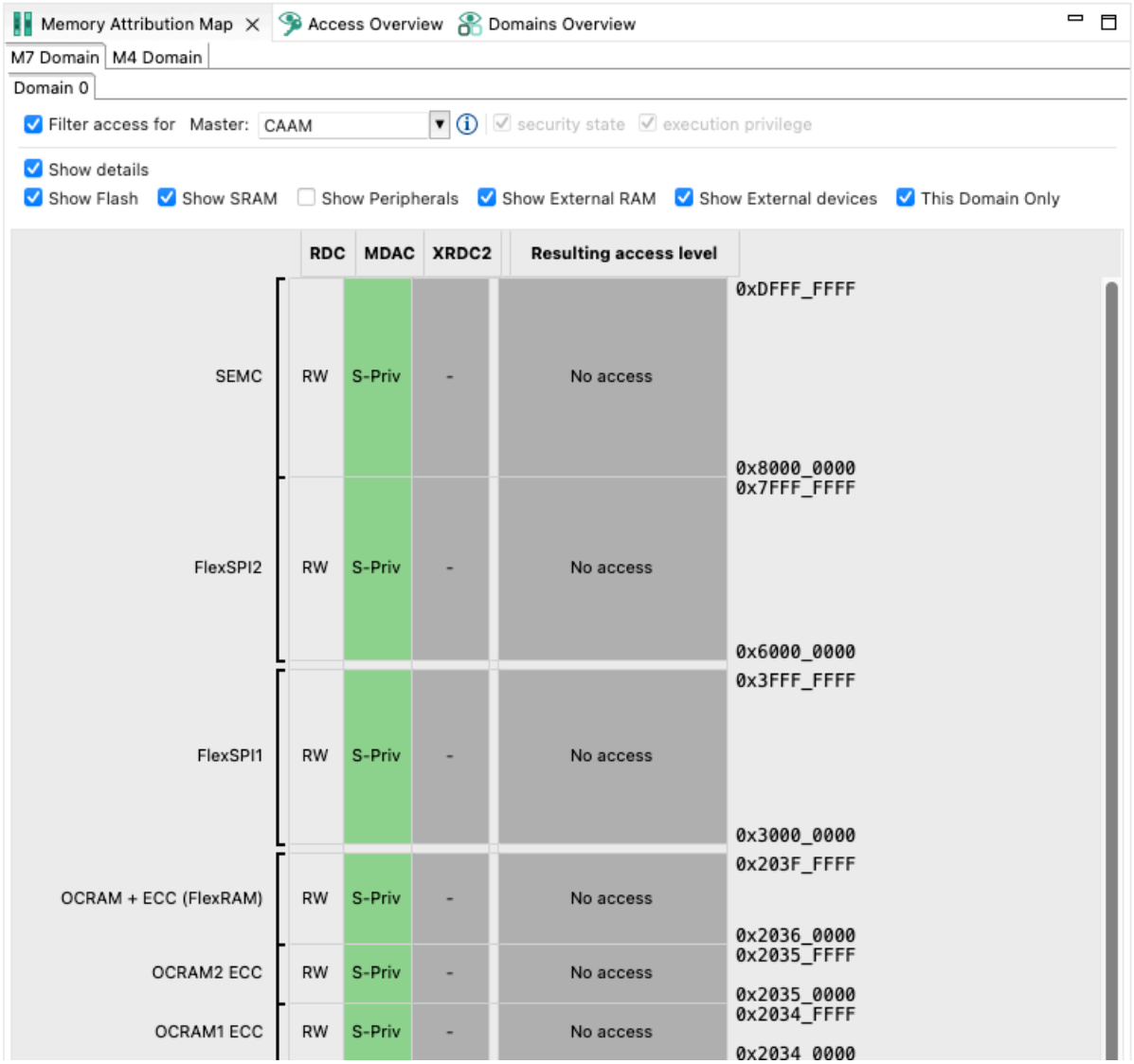


Figure 213. Memory Attribution Map

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 the 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.

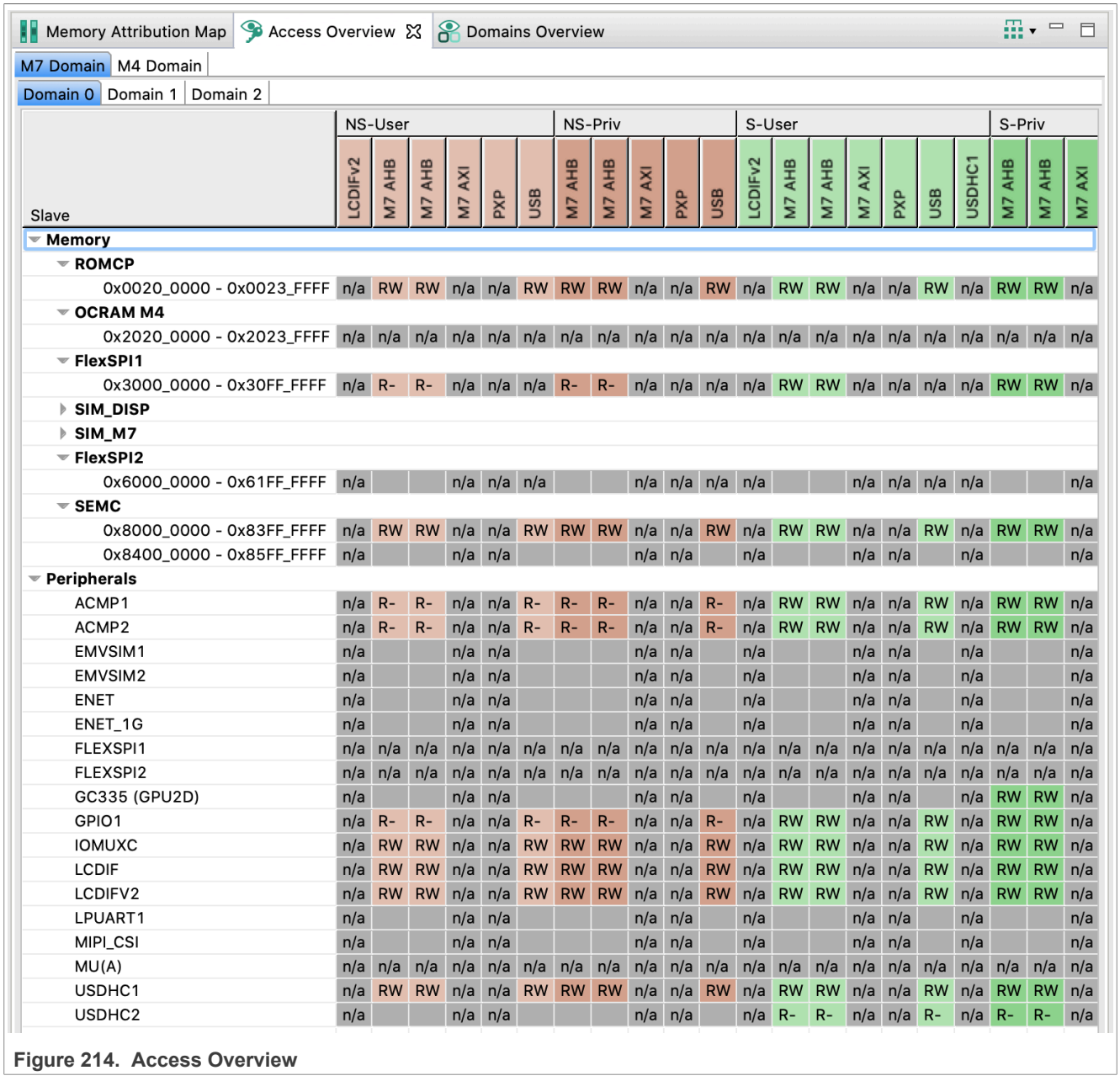
Double-click the cell to open the pertinent settings in **Security Access Configuration**.

7.2.4 Access Overview

In **Access Overview**, you can review the security policies you set in the **Security Access Configuration** view. The view is divided into subviews displaying the 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.



Slave	NS-User						NS-Priv					S-User							S-Priv		
	LCDIFv2	M7 AHB	M7 AHB	M7 AXI	PXP	USB	M7 AHB	M7 AHB	M7 AXI	PXP	USB	LCDIFv2	M7 AHB	M7 AHB	M7 AXI	PXP	USB	USDHC1	M7 AHB	M7 AHB	M7 AXI
Memory																					
ROMCP																					
0x0020_0000 - 0x0023_FFFF	n/a	RW	RW	n/a	n/a	RW	RW	RW	n/a	n/a	RW	n/a	RW	RW	n/a	n/a	RW	n/a	RW	RW	n/a
OCRAM M4																					
0x2020_0000 - 0x2023_FFFF	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
FlexSPI1																					
0x3000_0000 - 0x30FF_FFFF	n/a	R-	R-	n/a	n/a	n/a	R-	R-	n/a	n/a	n/a	n/a	RW	RW	n/a	n/a	n/a	n/a	RW	RW	n/a
SIM_DISP																					
SIM_M7																					
FlexSPI2																					
0x6000_0000 - 0x61FF_FFFF	n/a			n/a	n/a	n/a			n/a	n/a	n/a	n/a			n/a	n/a	n/a	n/a			n/a
SEMC																					
0x8000_0000 - 0x83FF_FFFF	n/a	RW	RW	n/a	n/a	RW	RW	RW	n/a	n/a	RW	n/a	RW	RW	n/a	n/a	RW	n/a	RW	RW	n/a
0x8400_0000 - 0x85FF_FFFF	n/a			n/a	n/a				n/a	n/a		n/a			n/a	n/a		n/a			n/a
Peripherals																					
ACMP1	n/a	R-	R-	n/a	n/a	R-	R-	R-	n/a	n/a	R-	n/a	RW	RW	n/a	n/a	RW	n/a	RW	RW	n/a
ACMP2	n/a	R-	R-	n/a	n/a	R-	R-	R-	n/a	n/a	R-	n/a	RW	RW	n/a	n/a	RW	n/a	RW	RW	n/a
EMVSIM1	n/a			n/a	n/a				n/a	n/a		n/a			n/a	n/a		n/a			n/a
EMVSIM2	n/a			n/a	n/a				n/a	n/a		n/a			n/a	n/a		n/a			n/a
ENET	n/a			n/a	n/a				n/a	n/a		n/a			n/a	n/a		n/a			n/a
ENET_1G	n/a			n/a	n/a				n/a	n/a		n/a			n/a	n/a		n/a			n/a
FLEXSPI1	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
FLEXSPI2	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
GC335 (GPU2D)	n/a			n/a	n/a				n/a	n/a		n/a			n/a	n/a		n/a	RW	RW	n/a
GPIO1	n/a	R-	R-	n/a	n/a	R-	R-	R-	n/a	n/a	R-	n/a	RW	RW	n/a	n/a	RW	n/a	RW	RW	n/a
IOMUXC	n/a	RW	RW	n/a	n/a	RW	RW	RW	n/a	n/a	RW	n/a	RW	RW	n/a	n/a	RW	n/a	RW	RW	n/a
LCDIF	n/a	RW	RW	n/a	n/a	RW	RW	RW	n/a	n/a	RW	n/a	RW	RW	n/a	n/a	RW	n/a	RW	RW	n/a
LCDIFV2	n/a	RW	RW	n/a	n/a	RW	RW	RW	n/a	n/a	RW	n/a	RW	RW	n/a	n/a	RW	n/a	RW	RW	n/a
LPUART1	n/a			n/a	n/a				n/a	n/a		n/a			n/a	n/a		n/a			n/a
MIPI_CSI	n/a			n/a	n/a				n/a	n/a		n/a			n/a	n/a		n/a			n/a
MU(A)	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
USDHC1	n/a	RW	RW	n/a	n/a	RW	RW	RW	n/a	n/a	RW	n/a	RW	RW	n/a	n/a	RW	n/a	RW	RW	n/a
USDHC2	n/a			n/a	n/a				n/a	n/a		n/a	R-	R-	n/a	n/a	R-	n/a	R-	R-	n/a

Figure 214. 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 215. 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 **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 or TRDC with security extension-enabled devices support ROM preset as well as C code. To generate code in the ROM preset, select the option in the **Miscellaneous** subview.

8 SerDes Tool

This section provides information about the SerDes Tool.

8.1 Introduction

This section introduces the SerDes (Serializer/Deserializer) tool, an embedded component of Config Tools for i.MX that supports the LX2, LA, and i.MX 95 processor families.

The SerDes tool provides two main functionalities: configuration and validation.

Note: The SerDes tool is provided as-is to aid customers in evaluating, debugging, and optimizing their designs. The results, or any part thereof, provided by the tool cannot under any circumstances be seen as a substitute for the traditional validation and compliance methods, which must be performed to declare compliance of the designs with the respective IP standards.

8.2 Create a SerDes tool project

To use the SerDes tool, create a project.

To create a SerDes tool project, follow these steps:

1. Open the **Config tools for i.MX**.
2. Choose **Create a new standalone configuration for a processor, board, or kit** and click **Next**.
3. From **Processors**, choose one of the devices with SerDes tool support and click **Finish.**
4. Click the **SerDes** tool icon to open the **SerDes tool** view

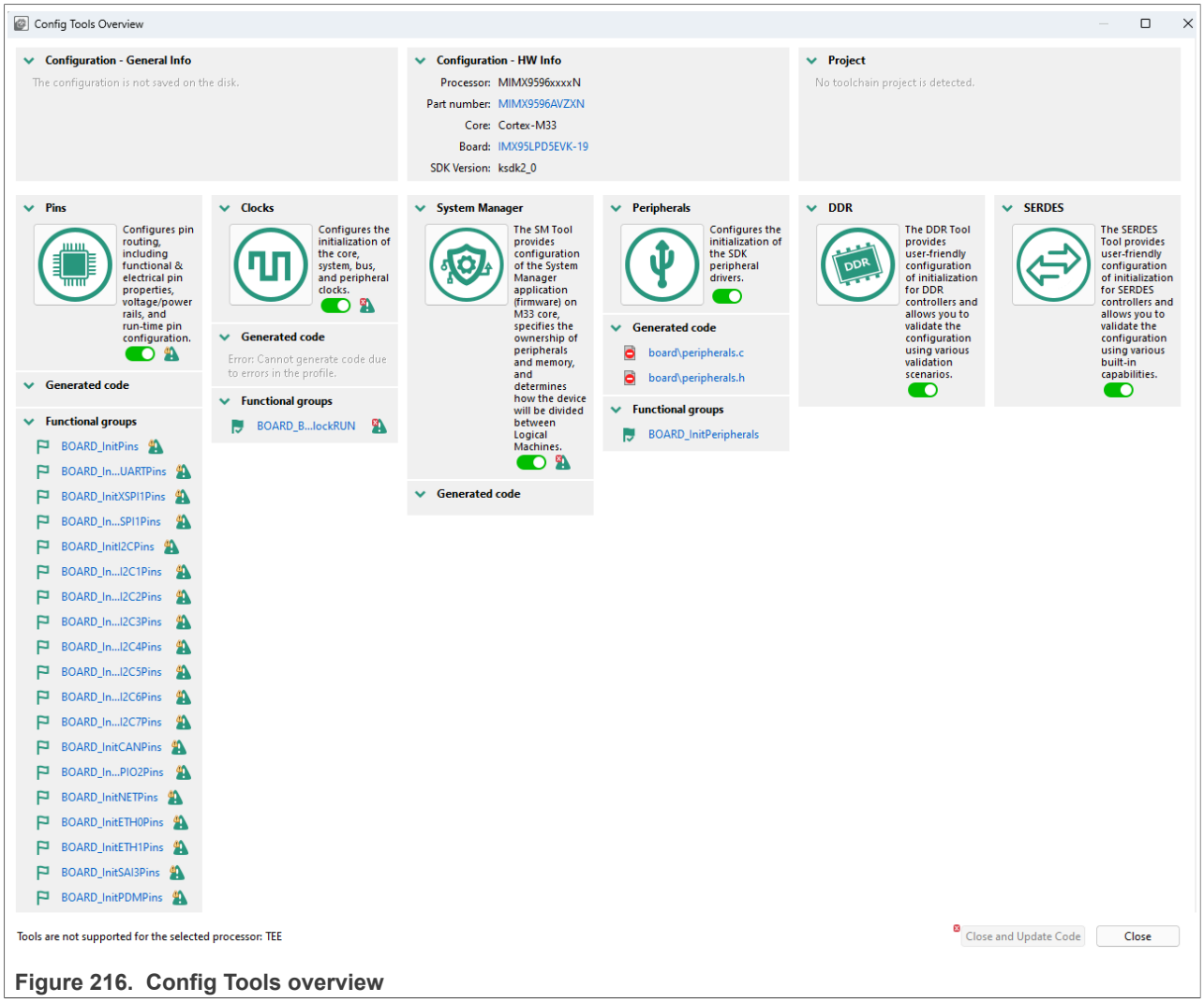


Figure 216. Config Tools overview

To use the SerDes tool, accept the Disclaimer.

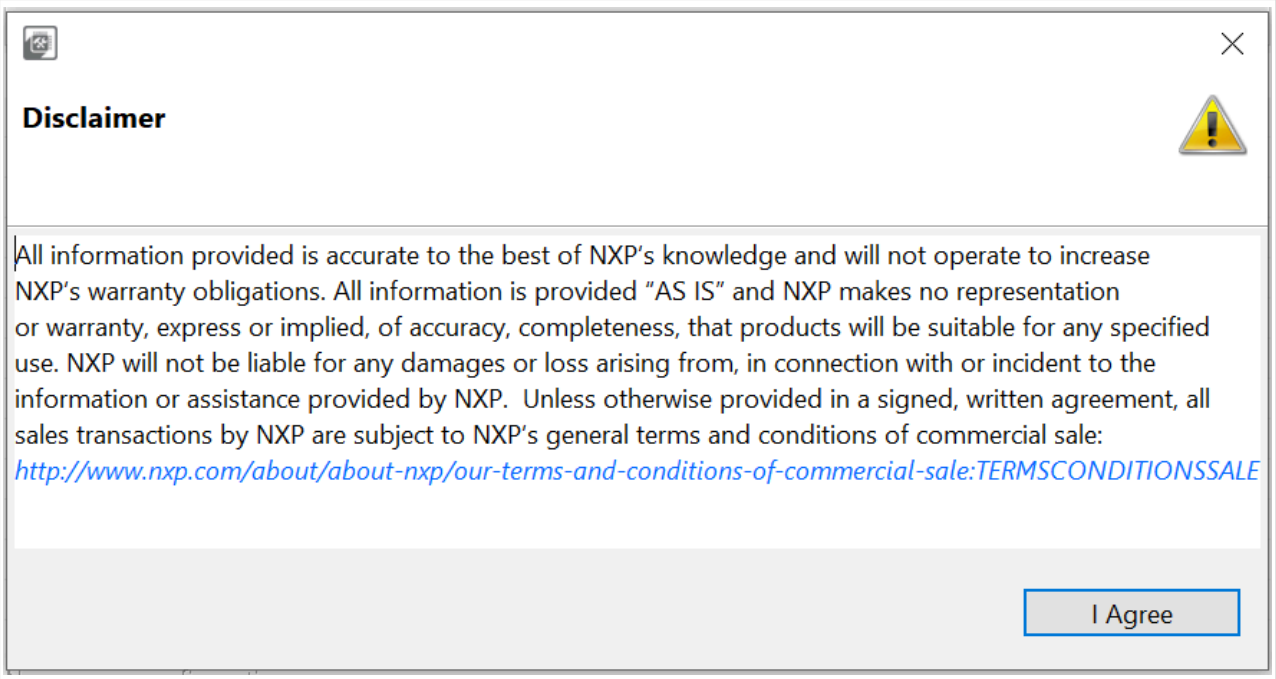


Figure 217. Disclaimer

8.3 SerDes configuration

The SerDes configuration provides a user-friendly graphical interface to configure the SerDes interface and other associated sub-systems. It can be used to change the SerDes configuration when a different protocol is used and to optimize the parameters associated with signal integrity.

An SoC can have several SerDes modules. Each SerDes module is organized into entities called lanes. The number of lanes varies with the device type.

Configuring SerDes means effectively configuring its lanes and involves:

- Protocol and speed - define what kind of traffic and at which speed the lane can operate with
- PLL used by the lane
- Transmit and receive equalization and electrical parameters. These are more in-depth configuration parameters that can affect the quality of the transmission on the lane

8.3.1 Lane configuration

The Lane Configuration is provided in the **SERDES** view and gives access to various advanced SerDes configuration options, which are related to the transmitter and receiver, electrical and equalization parameters.

The Lane Configuration panel shows the configuration for a single lane at a time. To configure a lane, click a specific tab (each tab represents a SerDes protocol) and then configure it in the Board Config.

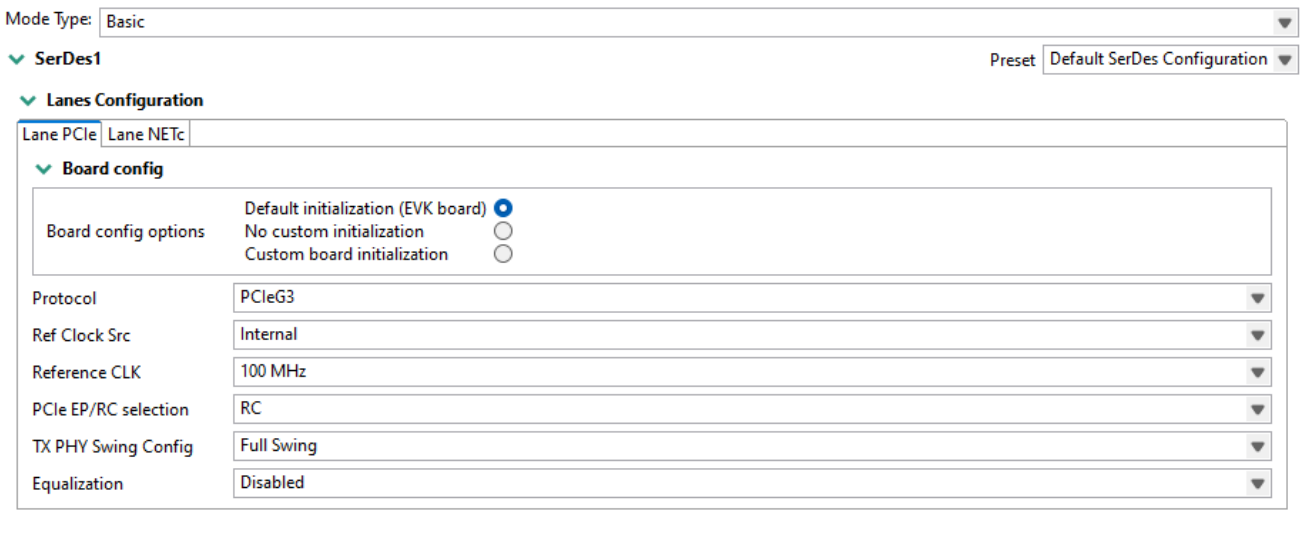


Figure 218. Lane Configuration panel

Any change in the Lane Configuration is highlighted in the **Register** view for a quick SerDes register inspection.

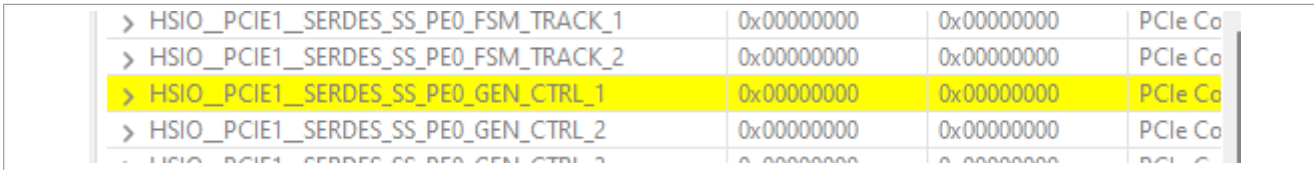


Figure 219. Register view

8.4 SerDes validation

After a SerDes configuration is defined (protocol/speed, lane receive and transmit, and PLL allocation), verify whether it performs reliably under the intended traffic type.

The validation relies on the SerDes hardware block built-in, programmable test capabilities that are used to verify the block under different conditions. The validation programs the built-in testing capabilities and then runs a series of tests.

The SerDes validation uses different scenarios to assess SerDes performance by downloading a test image to the processor's internal RAM using the Serial Download mode. Results are sent to the SerDes tool via UART.

SerDes validation can help to assess the stability of the SerDes interface on the board in a non-OS environment.

8.4.1 Connection

To connect to a board, the following prerequisites are required:

1. Configure the board to boot in the Serial download mode/Manufacture mode and power up the board.
2. Connect a UART cable from the host computer to the UART of the A-core on the board.
3. Connect a USB cable from the host computer to the USB port on the board that is used by the Serial download mode. A HID-compliant device or USB Input Device will be shown in the Windows Device Manager.

Note: Connect the USB cable directly to the host PC USB port, not through a USB hub.

With the board connected to the host computer, search for UART ports using **COM port scan**. The COM port drop-down list is populated with all available UART ports.

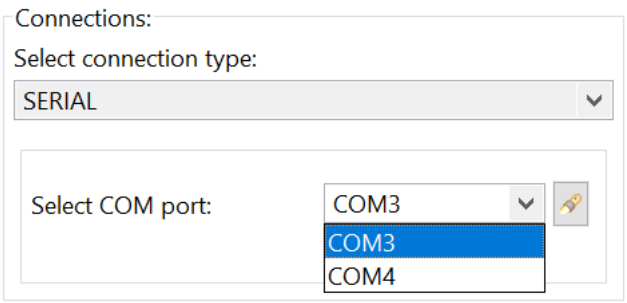


Figure 220. COM port drop-down list

Choose the correct UART that is used as the A-core debug UART port.

8.4.2 Test scenarios

Once the SerDes configuration and board connection are set up, different **Test scenarios** can be executed. Each test can be customized by setting the parameters from **Test options**.

Depending on the test and options selected, the execution time may differ. By default, a 90-second timeout is set to ensure that the test finishes if an issue occurs. To change the default value, edit the Timeout (seconds) option:

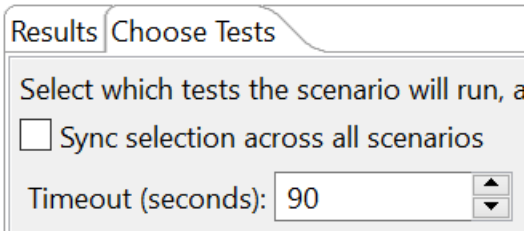


Figure 221. Timeout option

To start test execution, press the **Start Validation** button. Check the status of the running test from the **Logs** console. By default, the log level is set to **INFO**. Additional log level options are available with different output in the console:

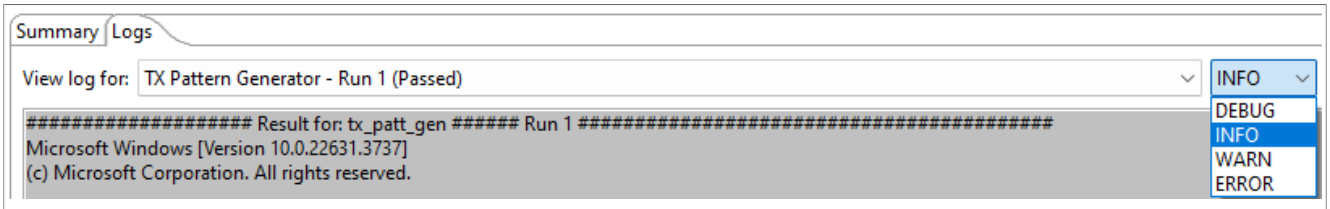


Figure 222. Additional log level options

At the end of the test, the status is displayed in **Results** with the test summary in **Summary**.

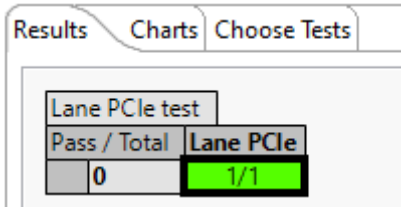


Figure 223. Test summary

Each cell color can be decoded using the Legend (see the figure below).

- Yellow is for *Test failed*
- Orange is for *Configuration error*
- Red is for *Target connection error or exception in the script*
- Green is for *Test passed*

Figure 224. Legend

The SerDes tool offers several test scenarios for each available lane.

8.4.2.1 TX pattern generator

Running the TX pattern generator test drives the SerDes module to generate a specific pattern (set up by the user) on the TX side of the lane that is being verified.

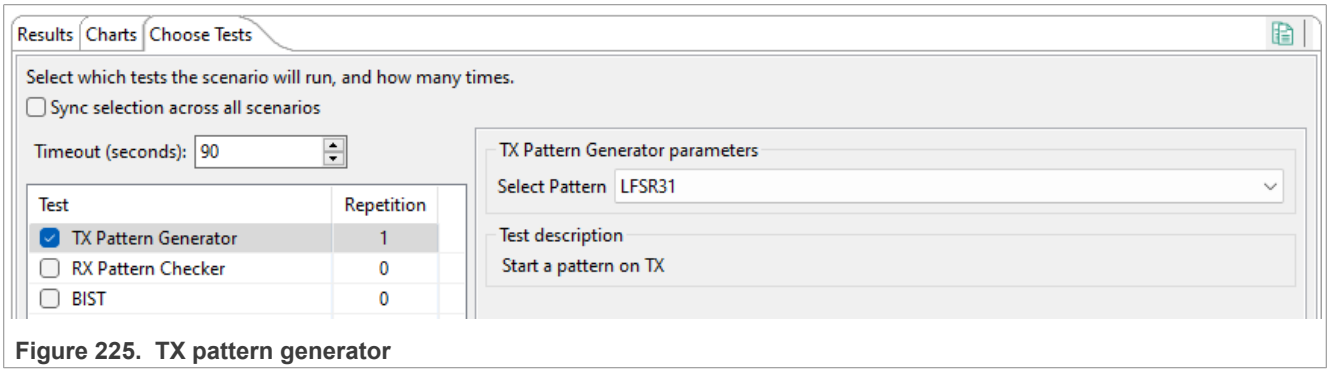


Figure 225. TX pattern generator

The TX pattern generator test can be used in an environment with two devices connected in a way that:

- One device starts transmitting data with the TX pattern generation test.
- The other device verifies the received data with the BIST test or the RX Pattern Checker in External mode.
- The TX pattern generator test continues to run until the board is restarted.

8.4.2.2 RX pattern checker

The RX pattern checker verifies the pattern received on the RX side.

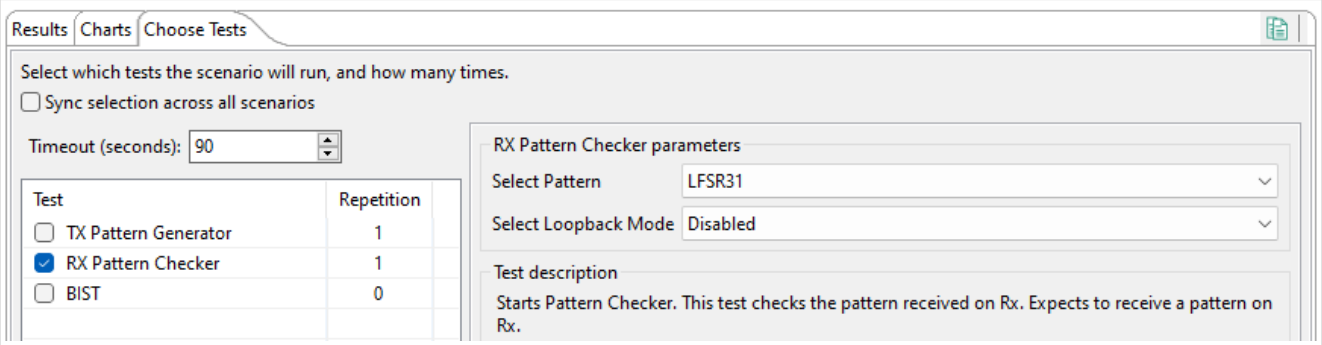


Figure 226. RX pattern checker

8.4.2.3 BIST

Built-in self-test (BIST) refers to a quick self-evaluation of a hardware block.

Running a BIST test involves the following steps:

1. Choose a pattern to be generated and how the traffic flows through the lane (that is, Loopback mode: Disabled, Internal, External).
2. Choose the number of **Insert Error**. The BIST output must contain the same number of generated errors as the number of errors inserted. This is a quick on-the-fly method to detect if some bits were lost.
3. Choose the **Time to wait**. This represents how much traffic is generated and analyzed by the SerDes lane. The time can widely vary from seconds to days, having a direct impact on when BIST results are available.

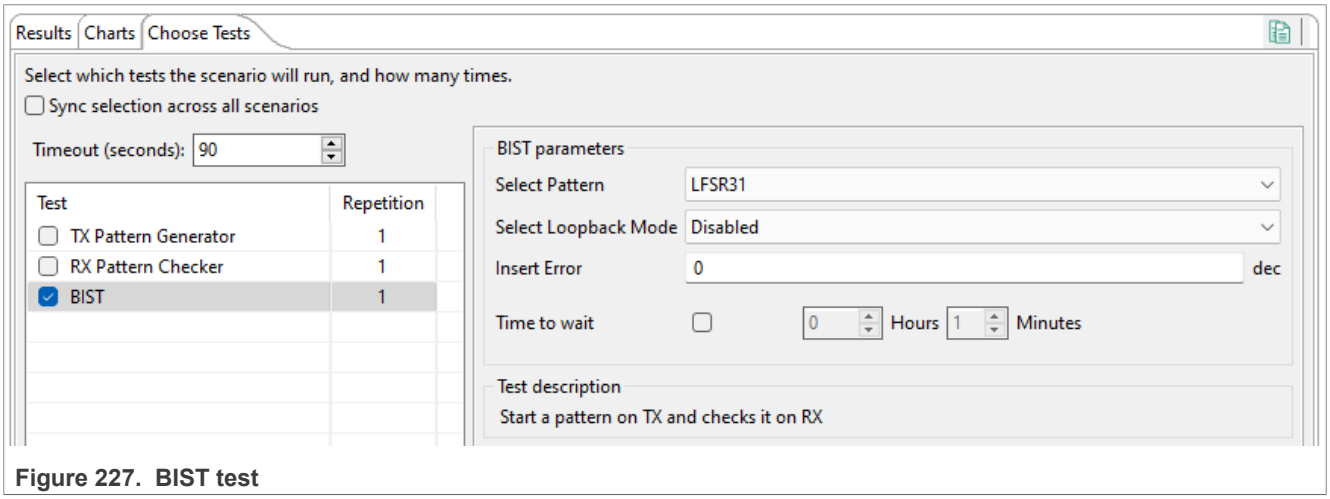


Figure 227. BIST test

9 System Manager Configuration Tool

The System Manager Configuration tool allows you to configure a project with the System Manager (SM) application on i.MX 9 series processors.

System Manager is a low-level system function that runs on a System Control Processor (SCP) and supports isolation and management of power domains, clocks, resets, sensors, and pins on complex application processors. It typically executes on Cortex-M processor architectures and delivers a comprehensive abstraction layer for the underlying hardware functionality. The primary purpose of SM is to allow isolation between software running on different cores in the SoC. SM maintains exclusive access to critical resources—such as power, clocks, reset, and PMIC—while providing an RPC interface to those clients (agents), enabling access control, arbitration, and aggregation policies for shared critical resources.

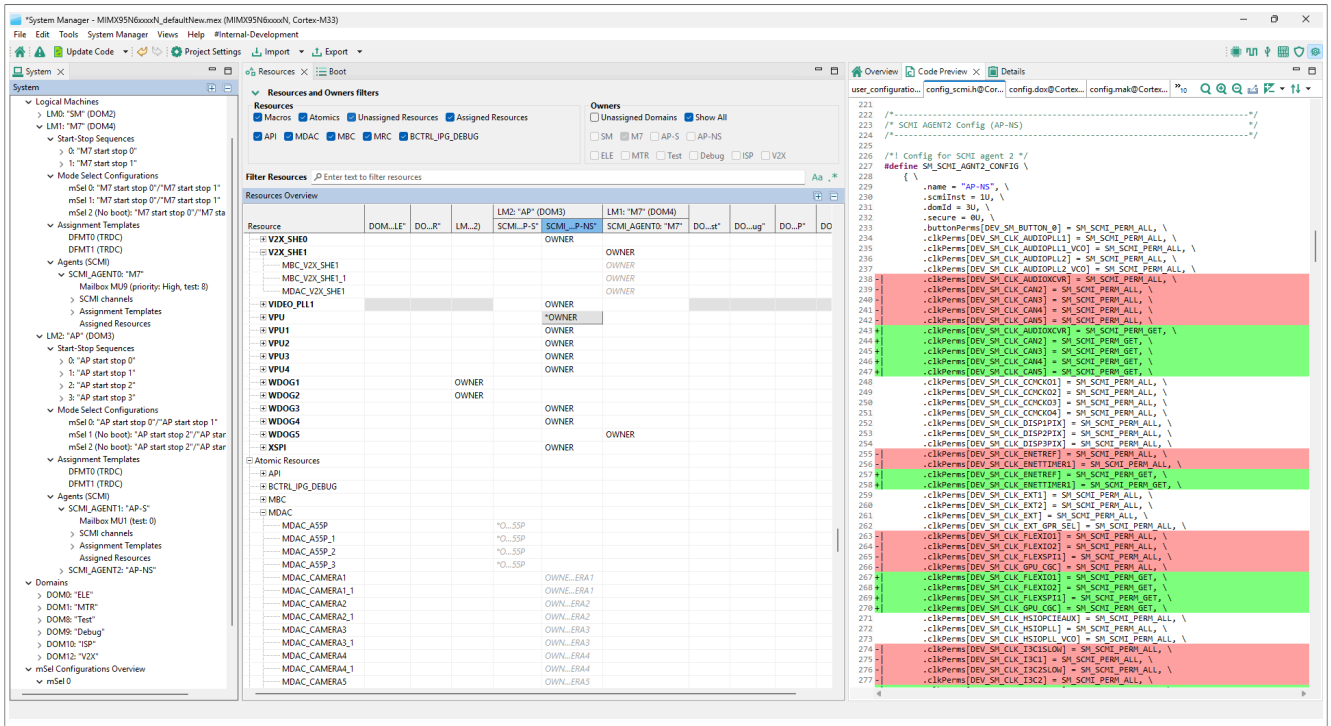


Figure 228. System Manager tool

9.1 SM configuration

Configuration of SM determines how the device (SoC) will be divided between Logical Machines (LM) and specifies the ownership of peripherals and memory. This ownership is used to determine API access rights to resources (clocks, pins, and so on) and CPU access rights to peripherals and memory (for example, RDC configuration). SM determines which LM will boot at POR and what RPC API functions the agents on those LM can call.

NXP provides the SM along with example board configurations. The configuration specifies the associated device and board (in most cases, a new configuration must be created for a customer product).

9.2 SM project settings and files description

The SM tool loads and processes data from a specified location of an SM Firmware project (for example, https://github.com/nxp-imx/imx-sm) by using the **Project Settings** wizard available in the System Manager toolbar.

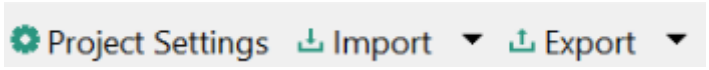


Figure 229. System Manager toolbar - Project Settings wizard and Import/Export options

Create an empty configuration based on the available board and device configuration (this option is suitable for developers of new or unsupported boards) or import an existing SM project configuration from CFG (for example, `mx95evk.cfg` from an SM FW repository) or a JSON file. Problems that occur during the configuration import are displayed in the report dialog that can be exported. In case of any issues, review the report and adjust the configuration accordingly.

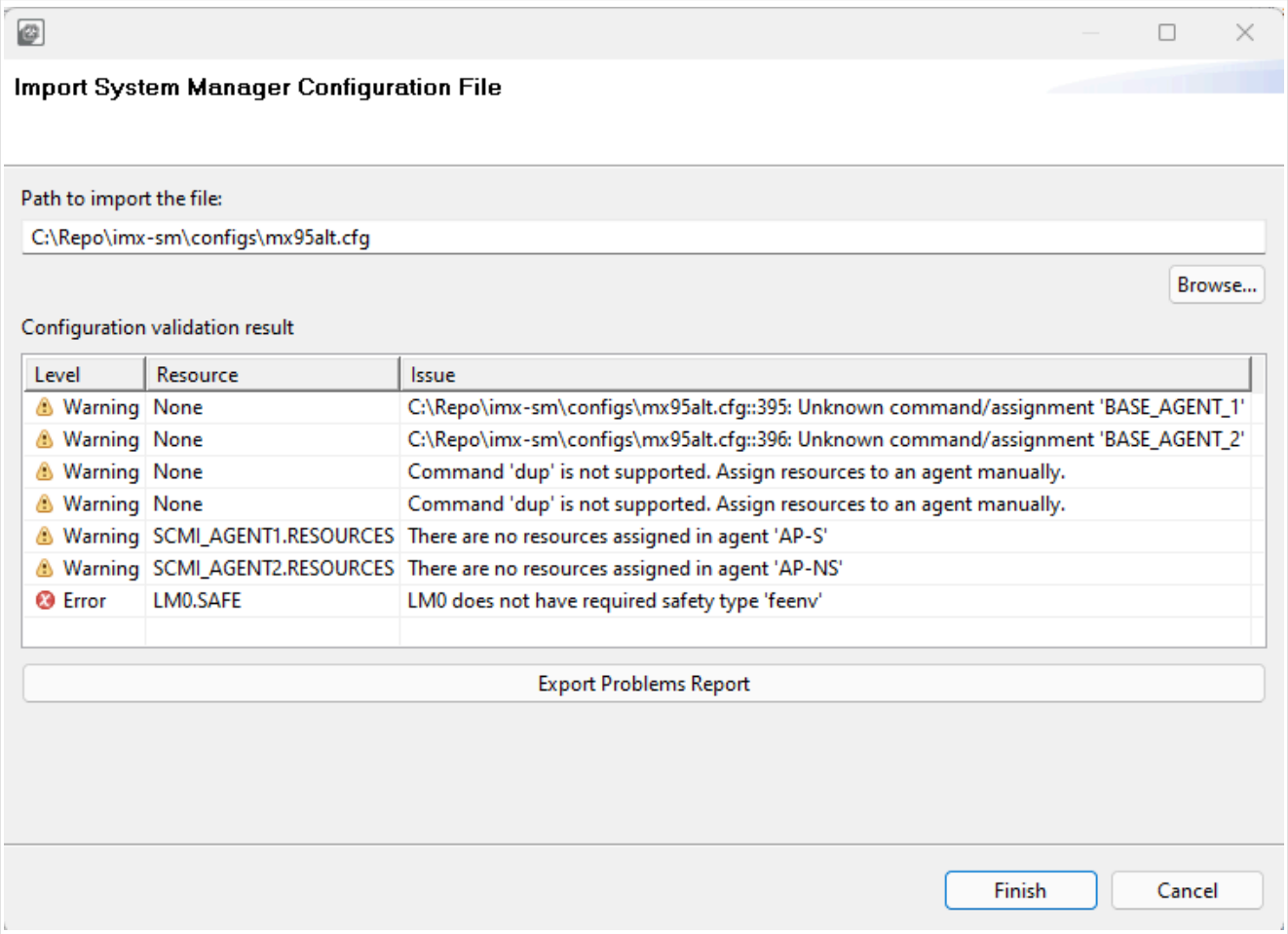


Figure 230. Problems reporting during import

The SM tool generates SM configuration source files that can be exported to the project location. The tool generates source files of the following types:

- Doxygen DOX file (`config.dox`)
- MAK file (`config.mak`)
- Configuration header files (for example, `config_scmi.h`, `config_trdc_h`)

Apart from that, the SM tool can convert CFG files to the JSON format in the form of Resource Database JSON files and a user configuration JSON file. These files can be exported by using the export dialog.

- `atomic_resources.json` contains available device and board specific atomic resources that can be configured by SM FW for a selected chip (pins, clocks, power domains, security peripheral regions, controls, faults).
- `macro_resources.json` contains predefined macro resources defined by SM FW. A macro resource defines a group of atomic resources that describe a larger logical cluster of HW resources. For example, macro resource LPUART1 can specify a power domain, clock, or memory block checker—a peripheral instance that can be assigned to any of the Resource Owners specified by HW and SM FW: Domain (DOM), Logical Machine (LM), or Agent.
- `chip_data.json` contains chip-specific information needed for SM FW configuration, such as security peripheral description.
- `error_log.json` contains configuration problems that occur during input files parsing or output generation.
- `user_configuration.json` contains the current SM project configuration state.

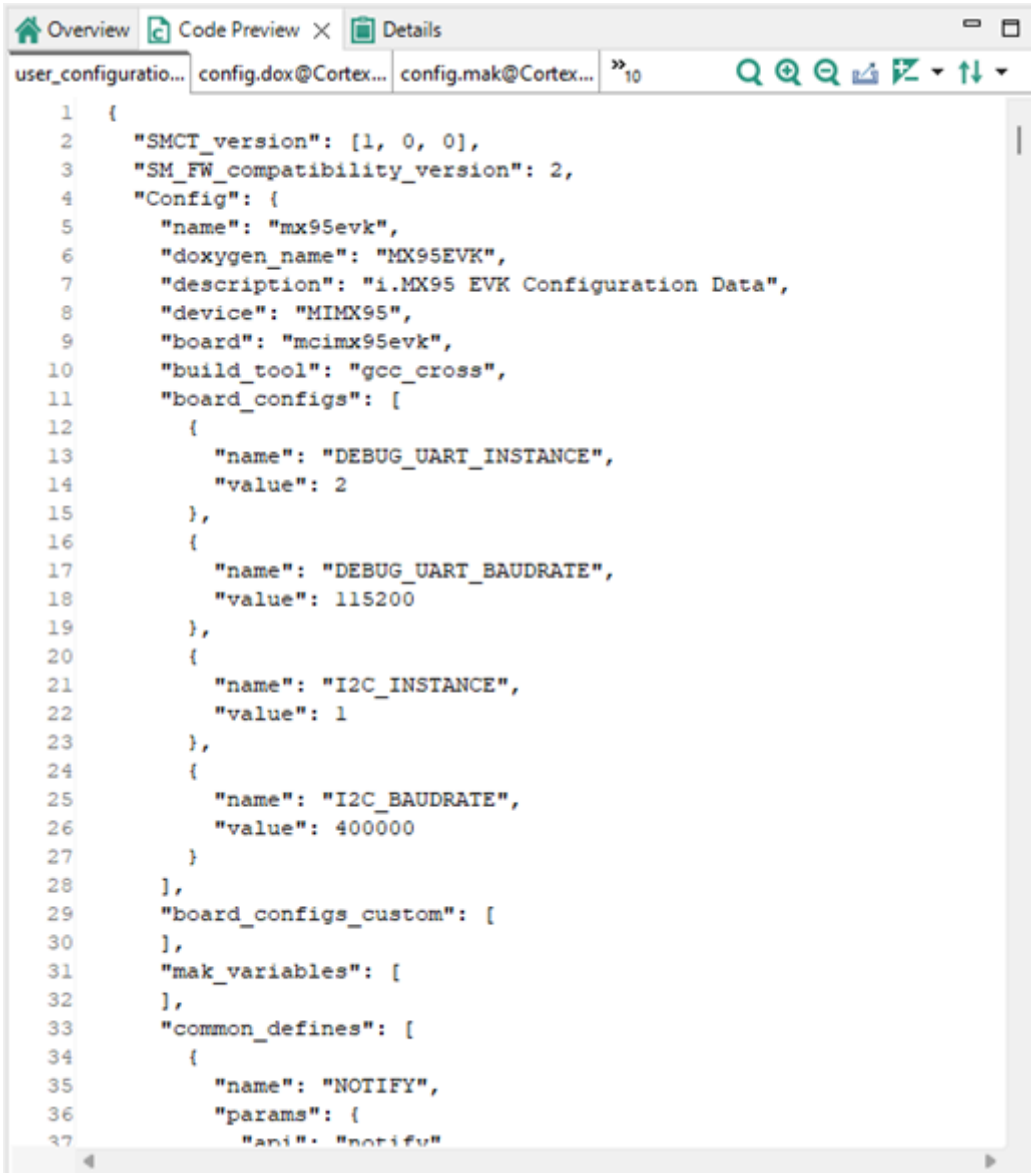


Figure 231. Generated user_configuration.json file

9.3 User interface overview

The SM tool consists of several views available to configure the SM project and to generate expected SM FW source files.

9.3.1 System view

The System view shows an overview of configurable elements and their important details of the System Manager project in the form of a project navigation tree. The System view allows you to configure Resource Owner specific settings, such as RPC protocol, Transport, Mailbox, and Assignment Templates (definition of a group of parameters that will be assigned to a Resource Owner). These are the main categories that can be configured.

- Project Configuration - SM project configuration name and description, compiler Mak file configuration

- Board Configuration - Board specific settings, SM debug options, Global Assignment Templates definition, creation of custom User Macro Resources
- Logical Machines - LM specific settings (for example, RPC), Mode Select Configurations from the LM perspective, Start-Stop Sequences configuration, Assignment Templates definition
  - Agents - SCMI Agent specific settings (for example, Mailbox), Assignment Templates definition
    - RPC Channels - Channel settings (for example, transport)
- Domains - DOM specific settings, Assignment Templates definition
- mSel Configurations Overview - Mode Select Configurations from the mSel perspective

To configure a generic System Manager project setting, for example, Project Mak file or debug features, use the System view tree to select the desired node. Every selection of the tree node shows available configuration settings in the configuration Details view. To expand/collapse all child nodes of the active selection, use the **Expand/Collapse** buttons on the System view header.

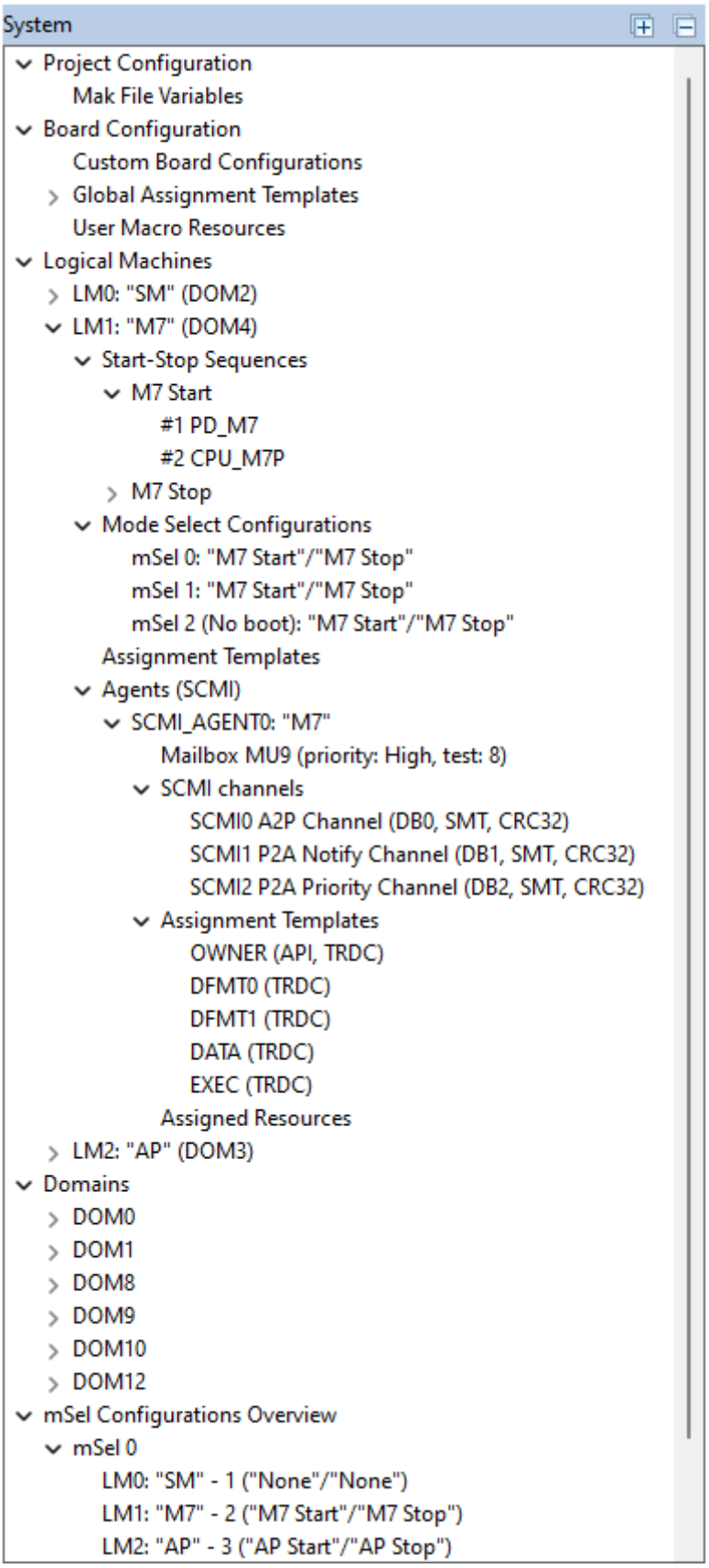


Figure 232. System View

9.3.2 Details view configuration

The Details view displays SM element settings and allows you to change them.

The information is dynamically updated to reflect modifications in the System, Boot, or Resources views.

In the Details view, the following actions can be performed:

- To display the element information, point the mouse cursor at the SM element.
- Inspect the SM element in the System, Boot, or Resources views.
- Modify the SM element settings by left-clicking the element value. There are the following types of settings:
  - Textbox - Left-click and type the value.
  - Dropdown - Left-click and select one of the offered values.
  - Checkbox - Left-click to set (enable) the element.
  - Button - Left-click to trigger an action or move to a specific configuration section.
  - Information - This setting type cannot be modified and is used to inform about a specified value.
- Add an element to the table by left-clicking the plus button.
- Remove an element from the table by left-clicking the cross button.
- Navigate to the parent setting via the Navigation bar on the top of the **Details** view.

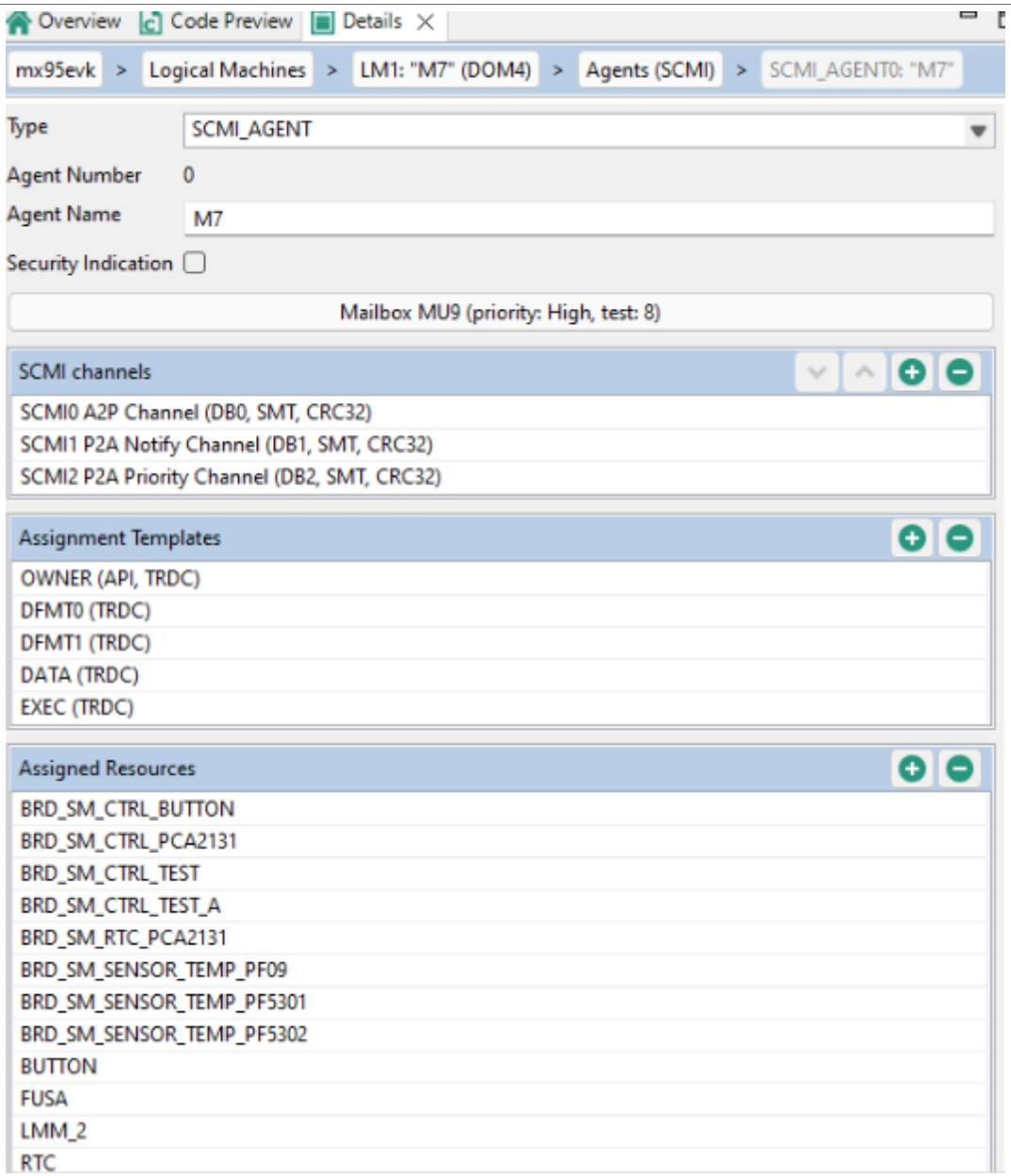


Figure 233. Configuration details view of the selected agent

9.3.3 Resources view

The **Resources** view shows an overview of resources assigned to Resource Owners and is used as a way of navigation to show the details of the selected resource assignment. The assignment is usually set by the Assignment Template that represents assigned parameters to the resource. After an Assignment Template is set to a resource, it is possible to configure additional available parameters related to the resource in the **Configuration Details** view.

The top of the **Resources** view contains various filters that can be used to specify the overview intention. Filter Resource Owners using specified filters and Resources using the search bar or specified filters.

Access control takes two forms, API access via an agent's MU and RDC access for a DID. Each LM is a unique DID and can have multiple agents, which all have the same DID.

To assign a resource to a Resource Owner (LM, SCMI_AGENT, DOM) use the **Resources** view.

- *Assign and configure Resource parameters* - Select the cell with the corresponding Resource (row) and the Resource Owner (column) in the **Resources** view. Click the **Assign** button in the configuration **Details** view.
- *Assign a resource with pre-configured Assignment Templates* - Using the **System** and **Details** views configure the Assignment Templates for the desired Resource owner - Set the name (identifier) and Resource parameters to be configured (parameters that do not relate to the assigning Resource are ignored).
- *Assign a resource with pre-configured mandatory (required) Assignment Templates DFMT0 and DFMT1* - Resources that configure Atomic Resource of the type MDAC require mandatory Assignment Templates DFMT0 for CPU and DFMT1 for non-CPU bus-masters to apply. The required templates are assigned automatically.

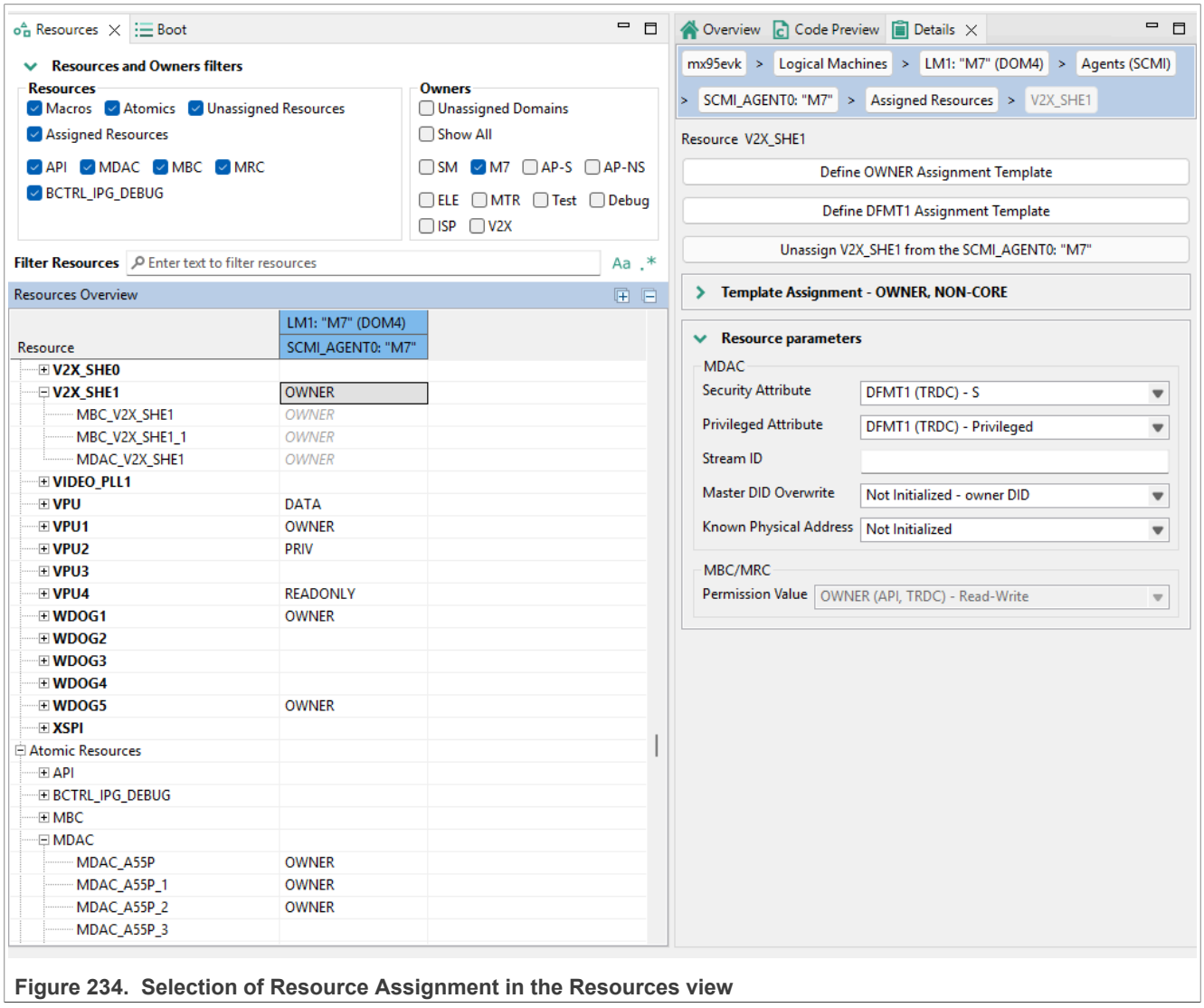


Figure 234. Selection of Resource Assignment in the Resources view

- *Create a new memory sector* - Select the corresponding **Resource** (row) with the <MEM> suffix and the **Resource Owner** (column) in the **Resources View**. Double-click the desired cell and select **Create a new memory sector**. Then configure the memory sector parameters (Start Address, Size or End Address, and Permissions) in the **Details View**.
- *Assign an existing memory sector to a **Resource owner*** - Expand the **Resource** (row) with the <MEM> suffix node, select the memory sector, and double-click the cell to select the template or assign the memory sector.

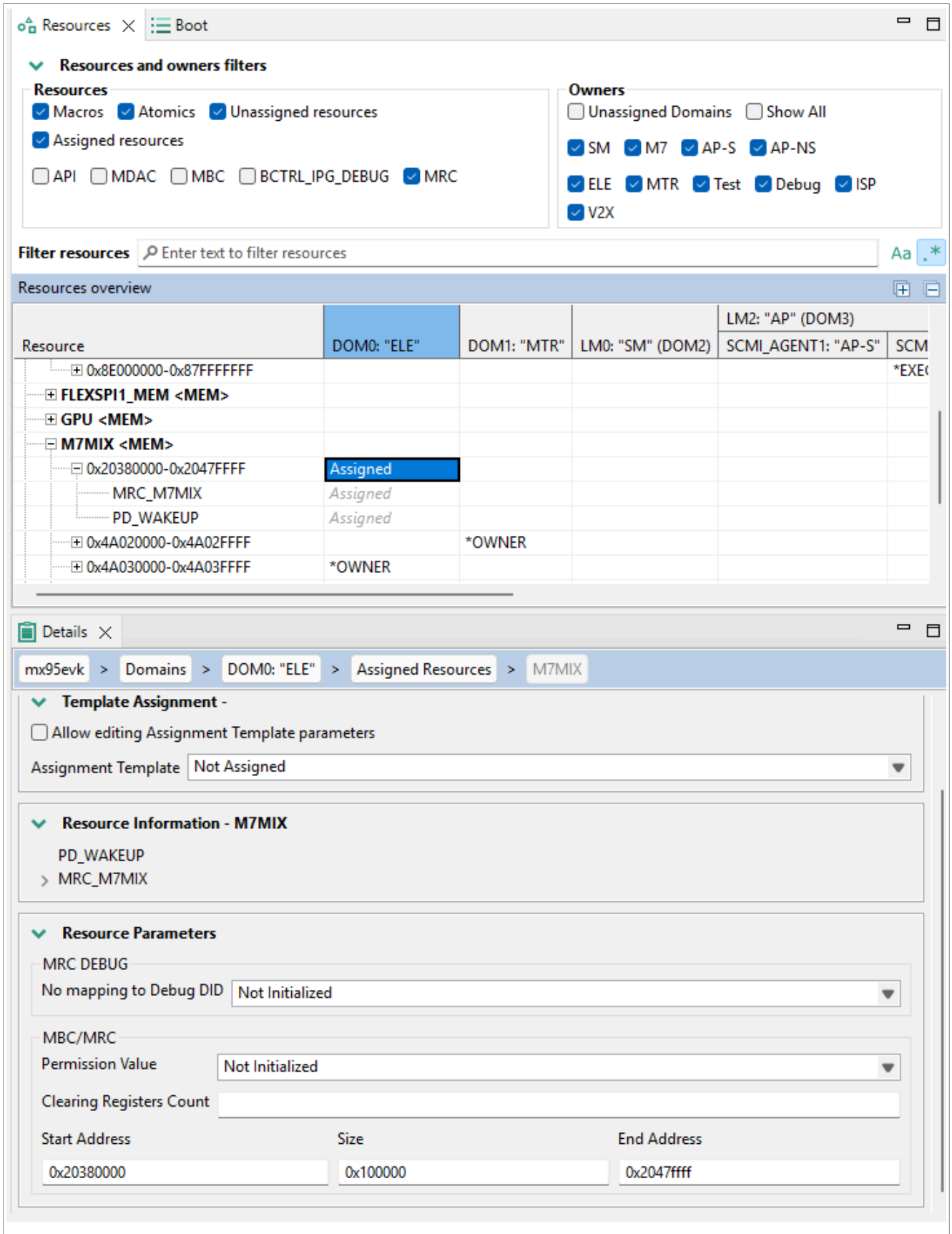


Figure 235. Example of Memory resource assignment - M7MIX<0x20380000-0x2047ffff> to ELE Domain

- To modify parameters of the resource assignment, select the resource and use the **Details** view:
  - Use the Template Assignment section to select and see configured parameters assigned by the Assignment Template. Modify the Assignment Template parameters by selecting the checkbox to enable editing.
    Note: Updating the selected Assignment Template affects all Resources that are assigned to this template.
  - To configure additional parameters to the selected Assignment template, select the settings below the Assignment Template selection.
    Note: The parameters that are not available are not related to the Resource. The parameters that are grayed are configured by the Assignment Template and cannot be overwritten. To change a grayed setting value, create a new Assignment Template, choose another, or update the selected one.
- To remove the Resource assignment, select the cell with the corresponding Resource (row) and the Resource Owner (column) in the **Resources** view. Then click the **Unassign** button in the configuration **Details** view.
- To find a desired Resource or to adjust the overview table, use the *Filter* or the search bar. Combine the selection of Resources and Owners filters with the search bar filtering. The search bar supports case-sensitive and regular expression search. Enable it by clicking the buttons on the right side of the search bar.
  Note: Macro Resources can be expanded in the **Resources** view to see all Atomic Resources that are included in the resource assignment.

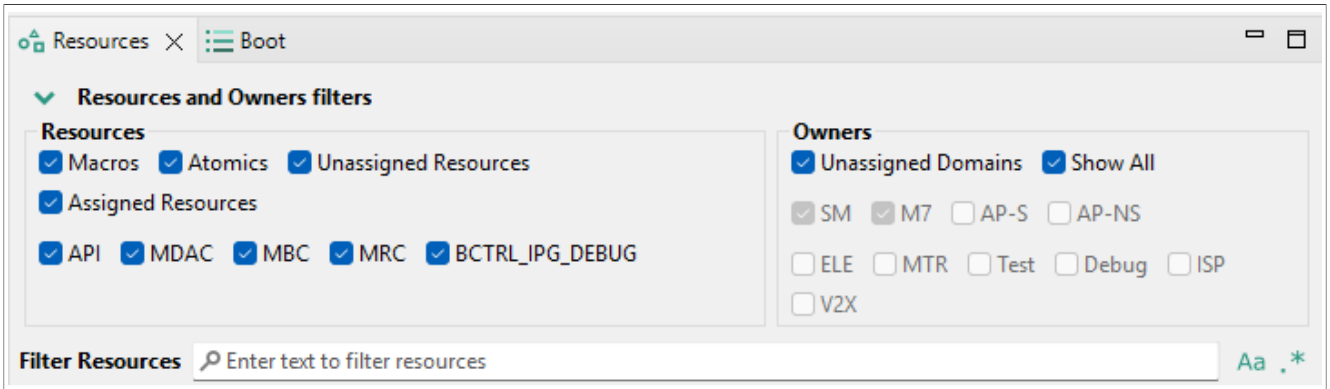


Figure 236. Filtering a resource

9.3.4 Boot view

The **Boot** view shows an overview of an independent lifecycle execution of LMs and configuration of boot Mode Select (mSel) options. The following options can be configured:

- Add or remove a new mSel mode from the System Manager configuration by clicking the **Add/Remove** buttons in the header of the **Boot** view.
- In the **Boot** view, the mSel table lists logical machines in the boot order. If a machine's boot is disabled, it stays in its position but its index changes to 0 and the remaining machines are renumbered accordingly. To reorder items, select one and click the **Up** or **Down** button.
- In the **Details** view of the selected Logical Machine, the Relative time to the time the SM starts to boot Logical machines can be configured.
- In the **Boot** view, each LM can have multiple boot configurations with different start and stop sequences. It can be determined if the LM is booted when the SM boots in every mSel section.
  - Configuration of the "bootskip" flag skips an error due to no image in the boot container.
  - Select the Start and Stop commands to execute when an LM is booted or shut down (sequences).
    - To add, remove, or modify a Start-Stop Sequence, select the LM to be affected from the desired mSel section and use the **Details** view.
    - Use plus/minus buttons to add or remove a Start-Stop Sequence
    - Double-click the Start-Stop Sequence to open its configuration details to be modified
      - Modify the index (identifier) and user name of the Start-Stop Sequence

– Use plus/minus buttons to add or remove Resource Commands to be called in the sequence.
– Double-click the Resource Command to open its configuration details to be modified (selection of Resource Command, optional arguments, and unit test flag)

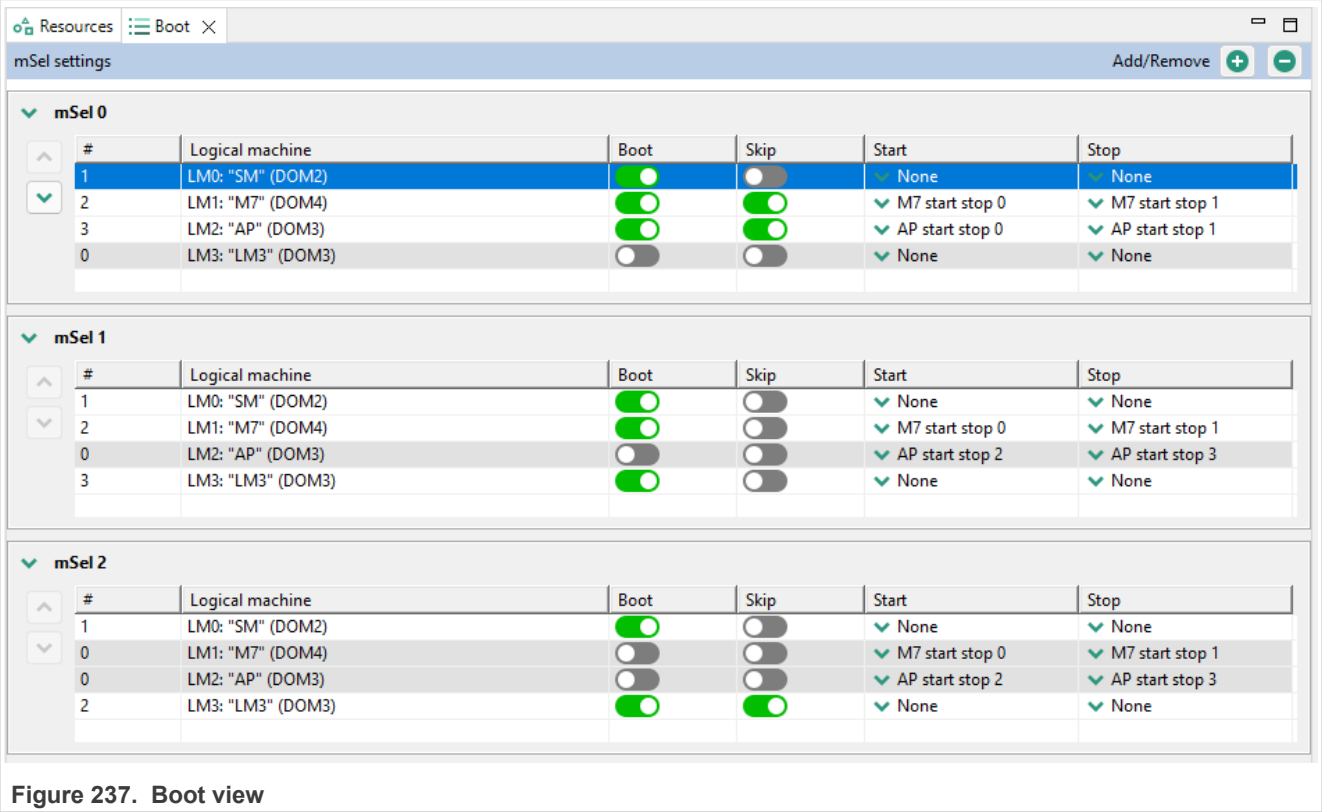


Figure 237. Boot view

9.4 Code generation

If the settings are correct and no error is reported, the code generation engine instantly regenerates the source code. The resulting code appears in the **Code preview** view of the System Manager tool.

Code preview automatically highlights differences between the current and immediately preceding iteration of the code. It is possible to choose between two modes of highlighting by clicking the Set viewing style for source differences. Highlighting can be disabled 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 238. Highlighted change of the code

9.5 Use cases workflow

This section describes how to configure specific System Manager parts or functions. The typical use case of the tool starts with importing the existing configuration. Creating an empty configuration suits the development of a new board or processor not yet supported by the official System Manager Firmware release.

1. After selecting a supported processor, import an existing configuration of the SM project in CFG format using the **Project settings** wizard.
2. To configure a generic System Manager project setting, for example Project Mak file or debug features, use the tree in the **System** view to select the desired element. Every selection of the tree element shows you available configuration settings in the Configuration **Details** view.
3. To configure a setting related to booting of the Logical Machines, use the **Boot** view.
   a. To add or remove the mSel configuration, use the plus/minus buttons in the left pane of the **Boot** view.
   b. In the desired mSel section configure parameters for each Logical Machine:
      i. Select LM and use **Details** view to configure Boot related parameters and add or modify Start/Stop sequences of Resource Command calls.
      ii. Enable the Boot parameter so that the index of the Logical Machine corresponds to the actual boot order (index 0 means the LM does not boot).
      iii. Enable the optional “Skip” parameter to skip the booting error message and select the Start and Stop sequences.
4. Use the **Resources** view to assign a Resource to a Resource Owner (every Resource Owner is related to a specific TRDC domain).
   a. To find a desired Resource or to adjust the overview table, use the Filter or Search bar.
   b. To add a new assignment, double-click on the desired Macro or the Atomic resource (row) related to the Owner (column) and assign a resource or a template.
   c. To remove an existing assignment, select the assignment and click the **Unassign** button in the **Details** view.
   d. To modify an assignment, click the resource and use **Details** view.
5. To adjust the overview of the Resource assignments, use the filter or search bar in the **Resources** view.

10 Advanced Features

This section describes advanced features.

10.1 Switching the processor

You can switch the processor or the package of the current configuration to a different one. However, switching to a completely different processor may lead to various issues, such as inaccessible pin routing or unsatisfiable clock-output frequency. In that case, fix the problem manually. For example, go to the **Routing Details** view and reconfigure all pins that report an error or conflict. Alternatively, you may need to change the required frequencies on clock output.

To change the processor in the selected configuration, select **File > Switch processor** from the **Menu bar**.

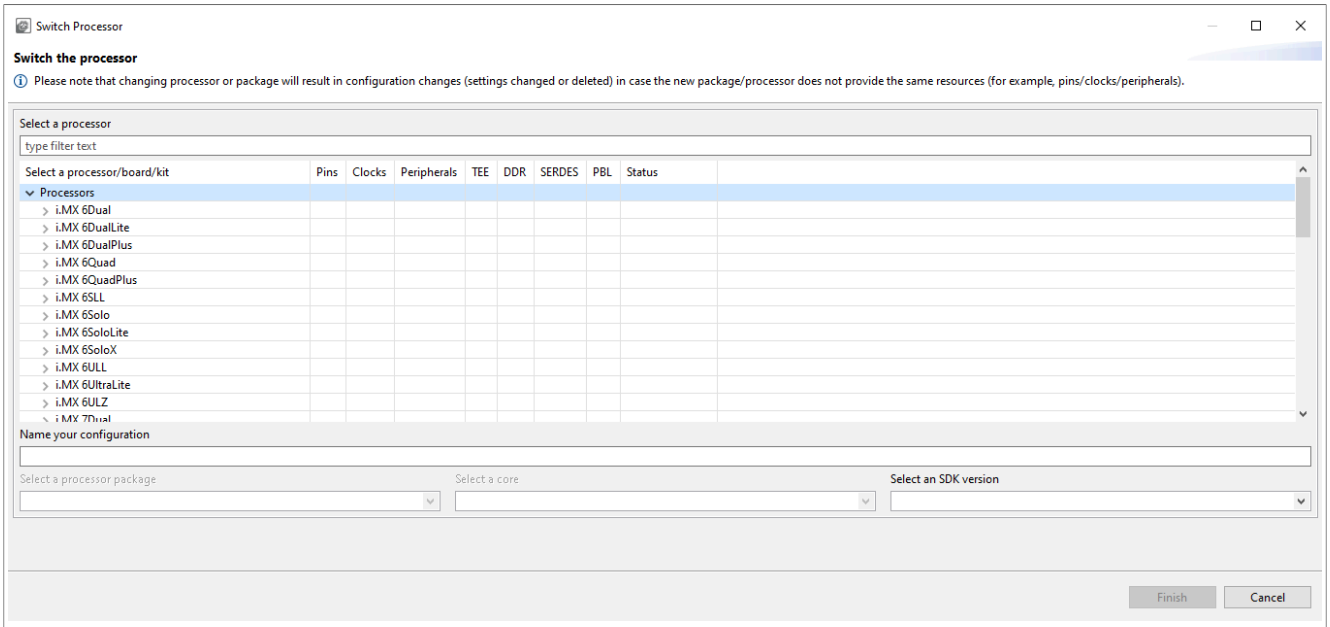


Figure 239. Switch processor

To change the package of the currently selected processor, select **File > Switch package** from the **Menu bar**.

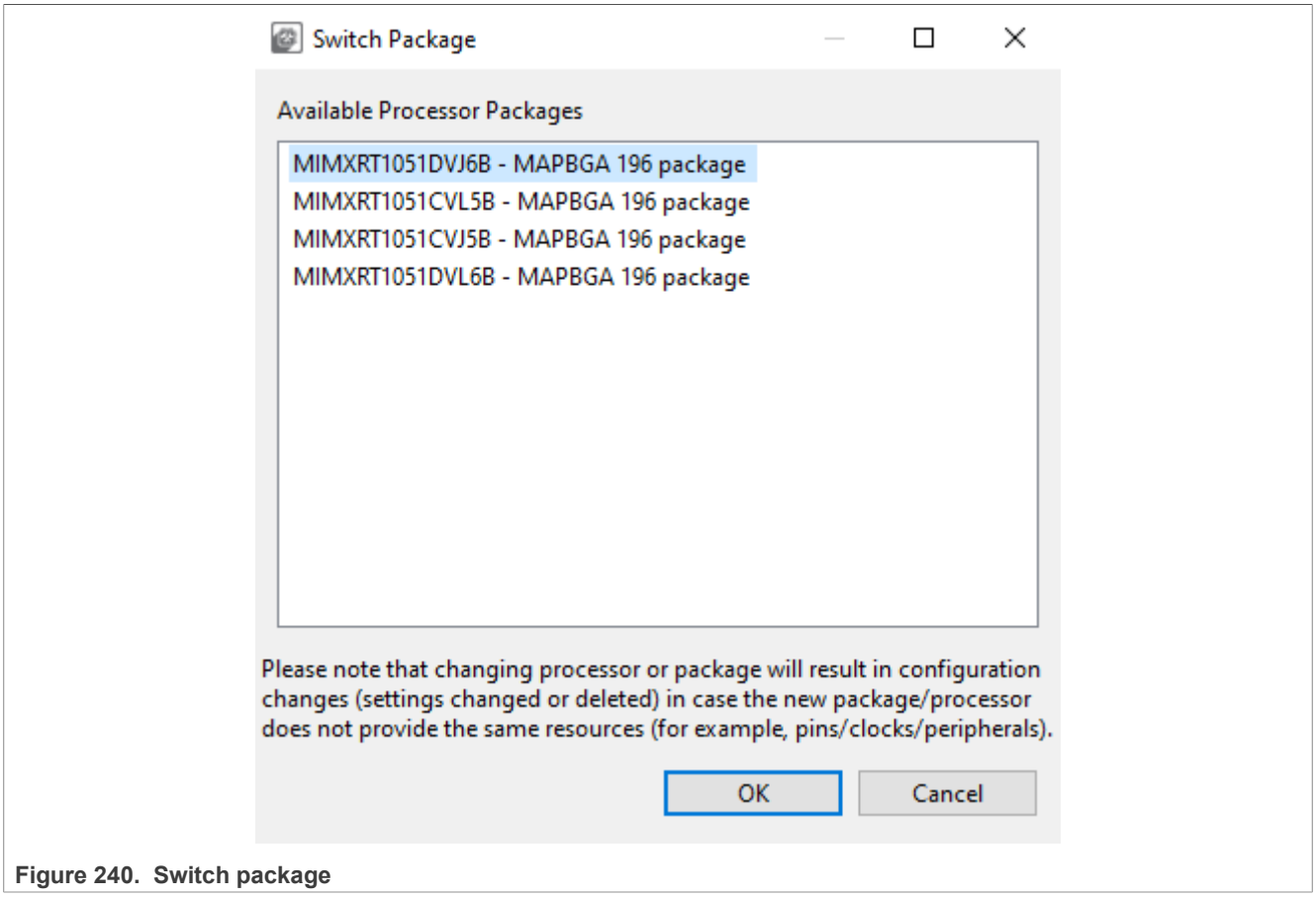


Figure 240. Switch package

10.2 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 241. 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 242. Aligning to block

10.3 Tools advanced configuration

Use the tools.ini file to configure the processor data directory location. Define the “com.nxp.mcudata.dir” property to set the data directory location.

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

10.4 Generating HTML reports

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

To generate the HTML report, select **Export > Pins > Export HTML Report**.

10.5 Exporting sources

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**.

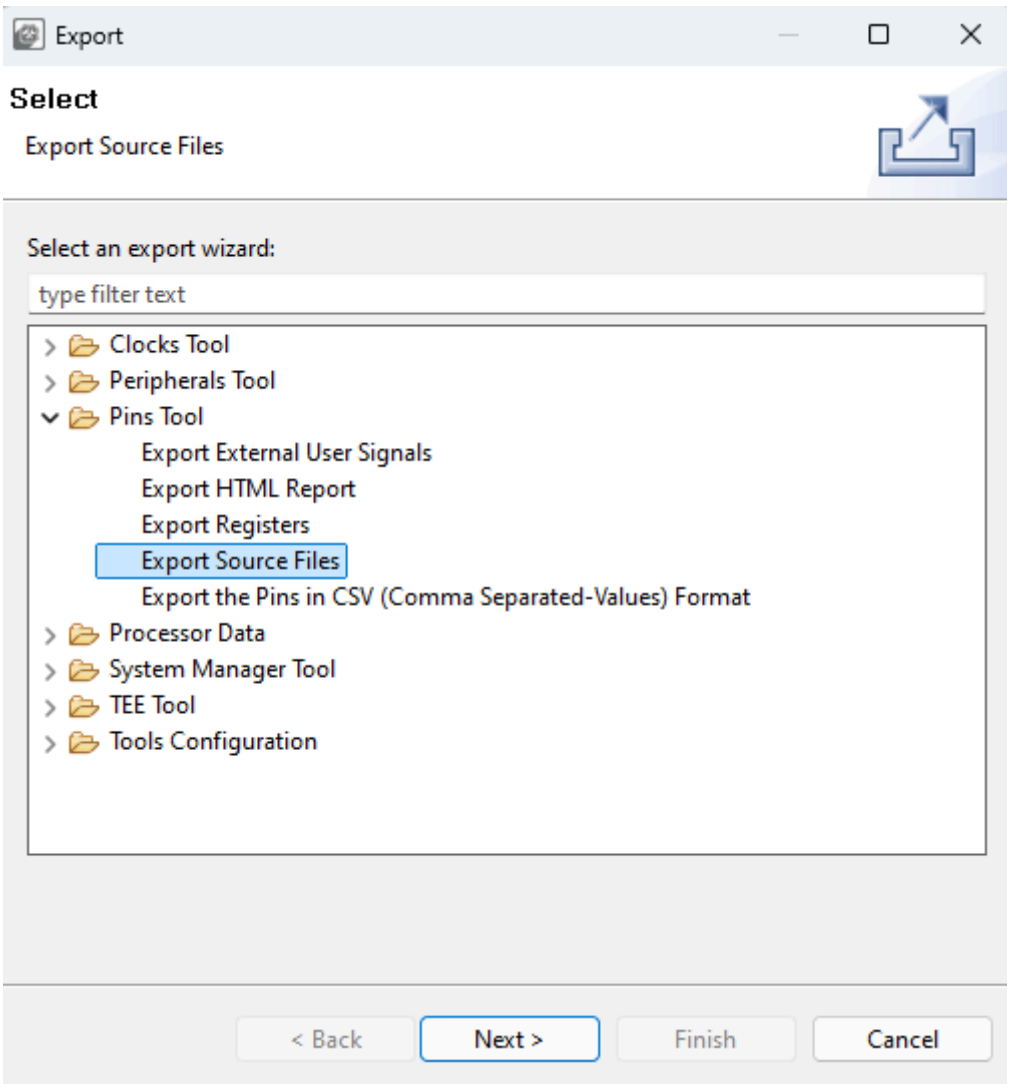


Figure 243. Export wizard

1. Click **Next**.
2. Select the target folder where you want to store the generated files.

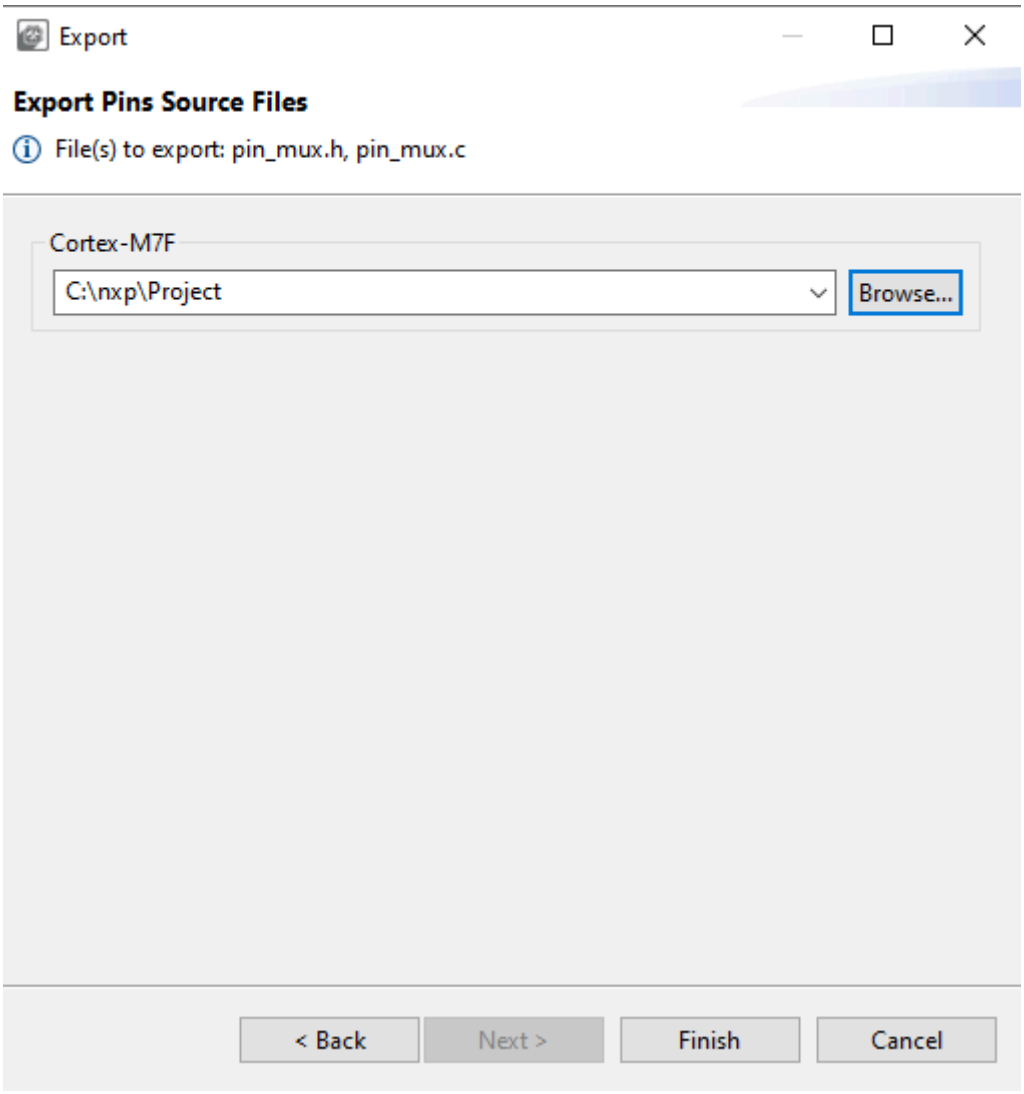


Figure 244. Export sources - Select target folder

1. For multicore processors, select the cores to export.
2. Click **Finish**.

10.6 Exporting registers

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

To export registers, follow these steps:

1. Select **File > Export** from the main menu.
2. Select the **Pins Tool > Export Registers** option.

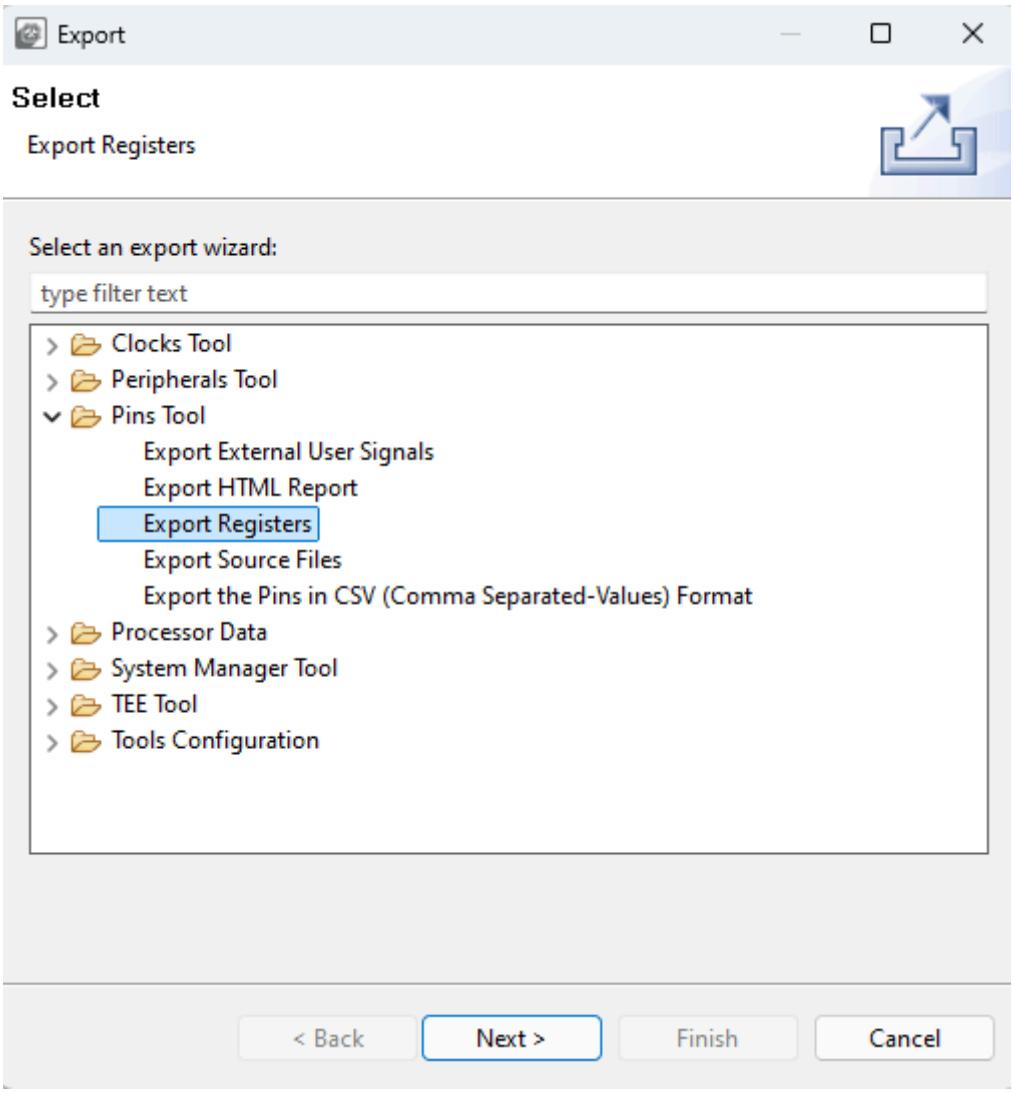


Figure 245. Export registers

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

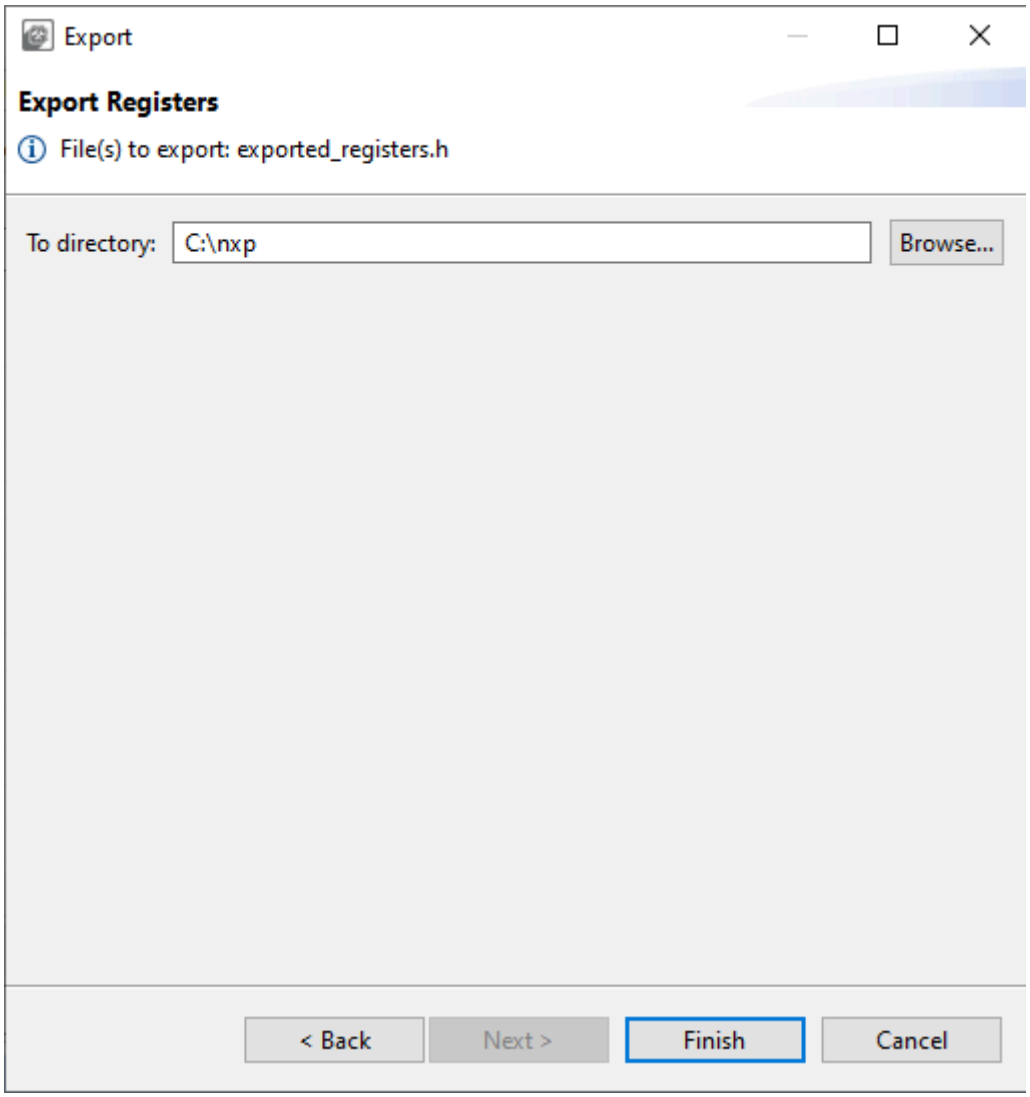


Figure 246. Export registers directory

1. Click **Finish**.

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

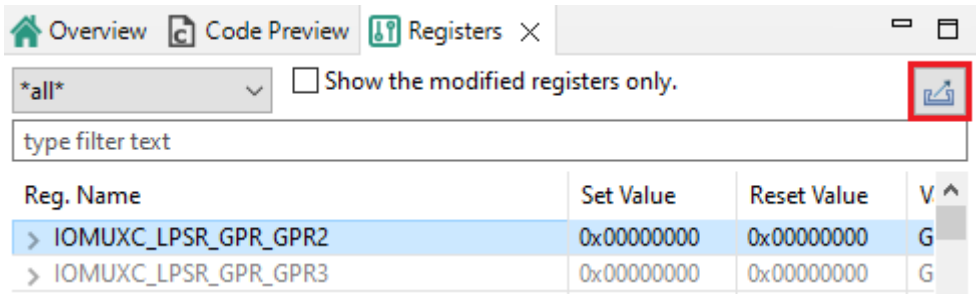


Figure 247. Export registers to CSV

10.7 Managing data and working offline

With the **Data Manager**, you can download, import, and export processor data. This feature is especially useful when working offline.

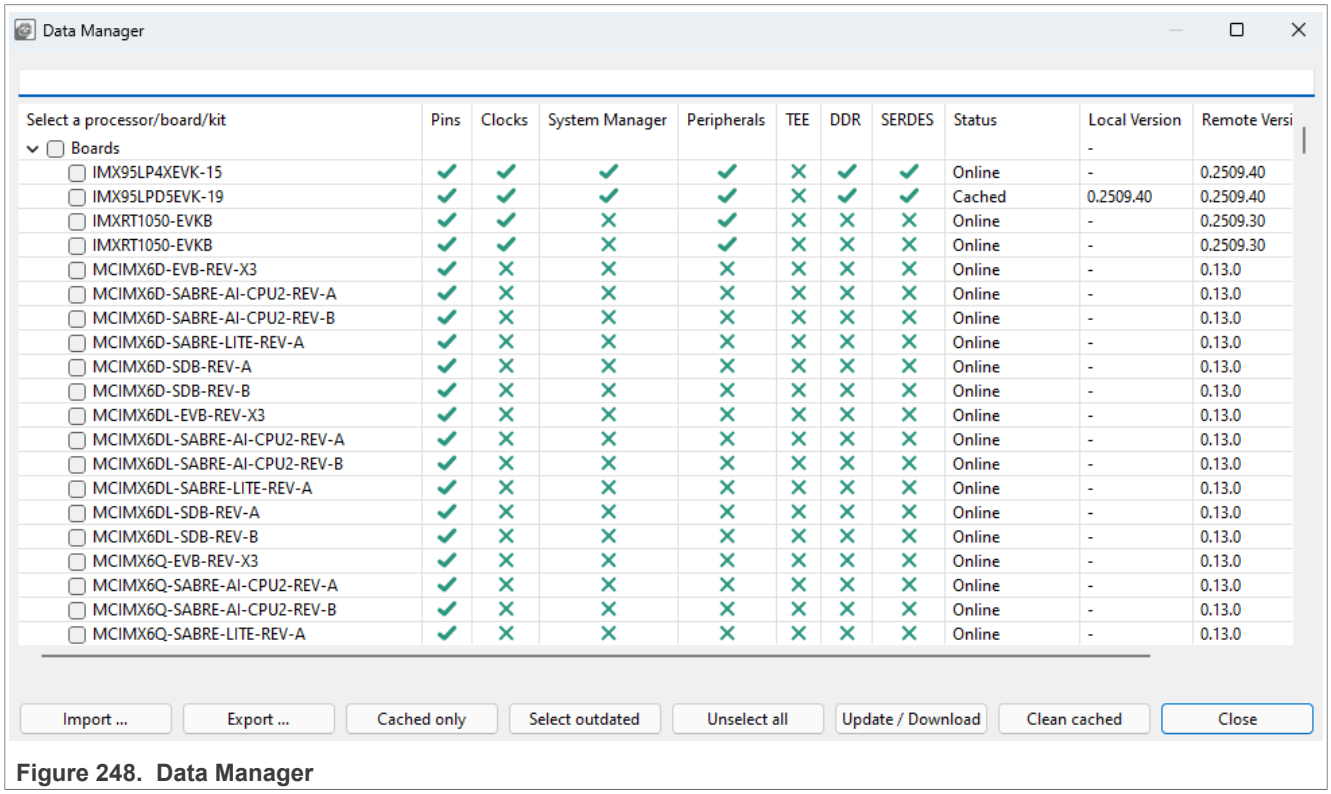


Figure 248. Data Manager

10.7.1 Working offline

You can create a new configuration even without access to the Internet by working with cached processor data. To do so, download processor-specific data before going offline, or import data downloaded and exported from an online computer.

To work offline, select **Edit > Preferences > Work offline** from the **Menu bar**.

10.7.2 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 **File > 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**.

You can now close your Internet connection and work with the data by selecting **File > New… > Create new standalone configuration for processor, board, or kit** in the **Start development** wizard.

10.7.3 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.

10.7.4 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 **File > 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**.
   The tool saves the data to 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**.

10.7.5 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 **File > 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 > Import Processor Data** from the **Menu bar**.

10.7.6 Updating data

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

Note: If you select the relevant option in **Edit > Preferences** in the **Menu bar**, data is 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 **File** > **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.

10.8 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 is 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. Reference substring groups with placeholders $1, $2, and so on.

Export the output path override rules using the wizard to a YAML file. The structure of the YAML file mirrors 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: [
]
```

10.9 Import pins configuration from legacy tools project

The Pins tool from **Config Tools for i.MX** imports pin configuration from existing legacy tool projects with the following prerequisites.

- Before importing any pins configuration from the legacy tools project, download the required i.MX processor pins tool data to a local directory.
- Before importing, create a new configuration for the respective i.MX processor of the given package variant to minimize manual correction of the imported pins configuration.

10.9.1 Importing from an IO Mux Tool design configuration file

You can import an existing pin configuration from an IO Mux Tool Design Configuration project file (XML) that was used to keep pin routing configuration within a legacy IO Mux Tool. The import wizard is used to import pin routing configuration from existing project XML file compatible with IO Mux Tool version v3.4.0.3.

To import from an IO Mux Tool design configuration file, do the following:

1. Select **File > Import** from the Main Menu.
2. Select the **Import Legacy IOMux Tool Design Configuration (XML) Format** option.
3. Click **Next**.
4. Select the source file (XML) to import by using the **Browse** button in the **Import** dialog.

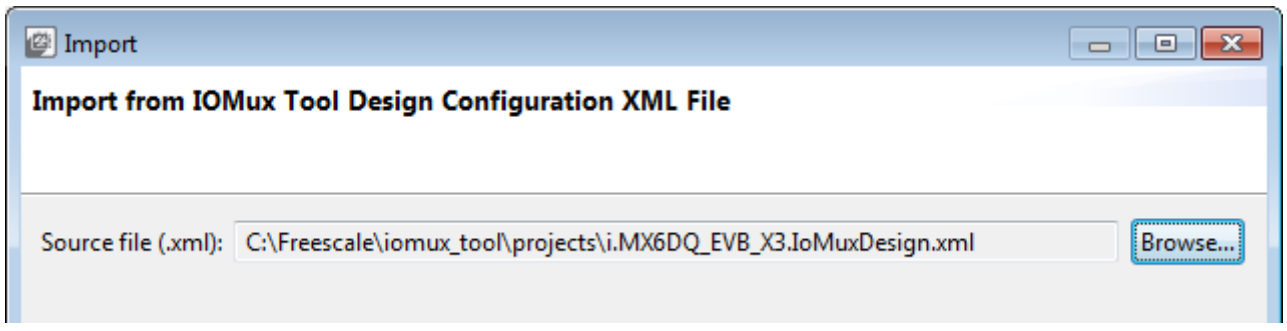


Figure 249. Import from IOMux Tool Design Configuration XML File

1. Click **Finish**.
   The selected source file is processed and the existing pin configuration for peripheral routing is imported for each peripheral to the Pins tool. A new function is created for each peripheral instance with all configured pins using the function name “configure_<peripheral name>_pins” and added into the **Routed Pins** view.

10.9.2 Importing from a Processor Expert project

You can import the existing pin configuration from a Processor Expert (PEx) project file (PE). The PE file is the main project file for legacy i.MX pin mux component and is available within the Processor Expert for i.MX tool.

Use the **Import** wizard to import legacy i.MX pin configuration from the existing PEx project file.

To import from a Processor Expert project, do the following:

1. Select **File > Import** from the main menu.
2. Select the **Import Legacy i.MX Pins Configuration (PEx for i.MX) Format** option.
3. Click **Next**.
4. Select the source file (.pe) to import using the **Browse** button in the **Import Pins Configuration from Processor Expert Project** dialog.

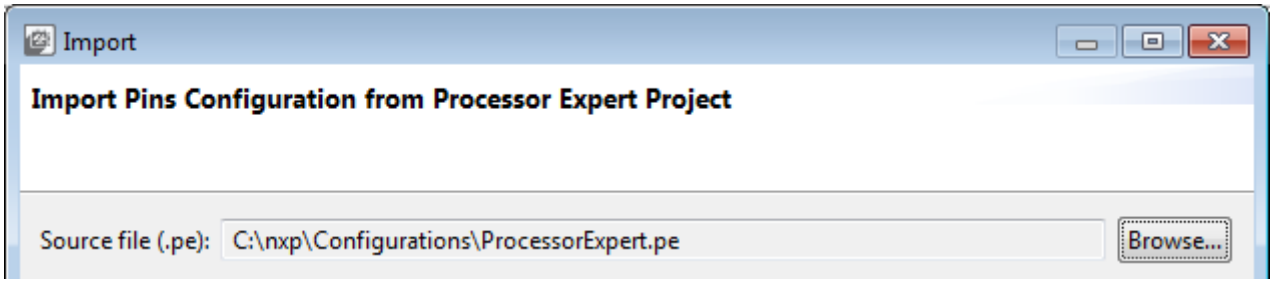


Figure 250. Import Pins Configuration from Processor Expert Project

1. Click **Finish**.
   The selected source file is processed and the existing pin configuration for peripheral routing is imported for each peripheral to the Pins Tool. A new function is created for each peripheral instance with all configured pins using the function name "`configure_<peripheral name>_pins`" and is added into the routed pins.

10.10 Command-line execution

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

On error 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:

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 existstools.exe -updateData -consoleLog -HeadlessTool Pins@rem now runs tools many timestools.exe -consoleLog -HeadlessTool Pins -Load some.mex -ExportAll c:/directorytools.exe -consoleLog -HeadlessTool Pins -Load other.mex -ExportAll c:/other_directory@rem and so on.`

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

Table 26. 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)		-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 cachedOnly '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	
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 frameworkChanges the processor in the result configuration of the previous chain	Requires the `-HeadlessTool`and `-SDKversion` commands.	-HeadlessTool Pins -MCU MK64 FX512xxx12 -SDKversion ksdk2_ 0-HeadlessTool Pins -Load C:/conf/conf. mex -SDKVersion ksdk2_0 -MCU MK64 FN1M0xxx12 -Export

Table 26. Commands supported in the framework*...continued*

Command name	Definition and parameters	Description	Restriction	Example
				MEX C:/conf/MK64.mex
Select core	`-Core`	Select core for the configuration	Requires the `-HeadlessTool, -MCU, -SDKversion` commands	-HeadlessTool Pins -Core core0
Select board	`-Board`	Board to be selected by framework (MCU is automatically selected too)(since 6.0)	Requires the `-HeadlessTool` and `-SDKversion` commands.	-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 `-HeadlessTool` and `-SDKversion` commands.	-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 a specific package of the MCUChanges the package in result configuration of the previous chain, see the command about `"-MCU"`	Requires the `-MCU` and `-SDKversion` commands	-HeadlessTool Pins -MCU MK64FX512xxx12 -SDKversion ksdk2_0 -PartNum MK64FX512VLL12
Configuration name	`-ConfigName`	Name of newly created configuration - used in exportRename the configuration	Requires the `-HeadlessTool` command	-HeadlessTool Pins -MCU MK64FX512xxx12 -SDKversion ksdk2_0 -Config Name “MyConfig”-HeadlessTool Pins -ConfigName “My RenameConfig”
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
Load configuration	`-Load`	Loads the existing configuration from a `*.mex` file	None	-Load C:/conf/conf.mex
Export Mex	`-ExportMEX`	Exports the`.mex` configuration file after tools runThe argument is expected to be a	Requires the `-HeadlessTool` command	-HeadlessTool Pins -MCU xxx -SDKversion xxx -ExportMEX C:/exports/my_config_folder-HeadlessTool Pins -

Table 26. Commands supported in the framework*...continued*

Command name	Definition and parameters	Description	Restriction	Example
		folder or path to specify the `.mex` file		ExportMEX C:/exports/my_config_folder/my.mex
Export all generated files	`-ExportAll`	Exports all generated files (with source code, and so on). Code is regenerated before exportIncludes `-ExportSrc`and in framework – `ExportMEX`The Argument is expected to be a folder name.	Requires the – `HeadlessTool` command	-HeadlessTool Pins -ExportAll C:/exports/generated
Create a new configuration by importing a toolchain project	`-ImportProject {path}`	Creates a new configuration by importing the toolchain project. The parameter is a path to the root of the toolchain project.	Requires the `-Headless Tool`command	-HeadlessTool PrjCloner -Import Project c:\test\myproject-Headless Tool Peripherals -ImportProject c:\test\myproject-HeadlessTool Prj Cloner -ImportProject c:\test\myproject\myproject.cbuild-gen-idx.yml-HeadlessTool Peripherals -Import Project c:\test\myproject -sdkPath c:\test\sdk_dir
Create a new configuration from a toolchain project	`-CreateFrom Project {path}`	Creates a new configuration (memory only) by importing a toolchain project based on the found `.mex` or YAML info. It does not do any changes on the disk (mex and update code), validates the configuration. The parameter is a path to the root of the toolchain project. Since v16: allows setting directly the toolchain project file path (for example, `*.cbuild-gen-idx.yml, *.uvprojx, *.ewp, CMakeLists.txt`) allows creating/opening configuration directly in UI from the toolchain project	None	-HeadlessTool Prj Cloner -Create FromProjectc:\test\myproject-Headless Tool Peripherals -CreateFromProject c:\test\myproject-Create FromProject c:\test\myproject\myproject.cbuild-gen-idx.yml

Table 26. Commands supported in the framework*...continued*

Command name	Definition and parameters	Description	Restriction	Example
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 `-Headless Tool`command	-HeadlessTool Pins -CustomCopyright c:\test\copyright.txt
Override the output path of the generated files	`-OutputPath Overrides`	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 `-Headless Tool`command	-HeadlessTool Pins -OutputPathOverrides c:\test\outputPath OverrideRules.yaml
Batch processing	`-BatchFile`	Path to the file with commands that will be run on defined paths. For details, see: Batch processing	Requires the `-Headless Tool` command; intent without other commands.	-HeadlessTool Pins -BatchFile C:/conf/batch Commands.yaml
Project link	`-ProjectLink`	A custom path to the toolchain project folder. It can be used as the absolute path or relative against the saved configuration. Empty sting for default that is not saved in the configuration (since v14).	When the command is not used along with the `-HeadlessTool` command, the `-Load` command is required.	-HeadlessTool Pins -ProjectLink C:/project-HeadlessTool Pins -ProjectLink armgcc-HeadlessTool Pins -ProjectLink “”-Load C:/conf/conf.mex -Project Link armgcc
Update locally downloaded data	`-updateData`	Downloads data for already locally downloaded data if they have an update.	None	-updateData
Update code generated by tools, that is .c and .h (and other) source files.	`-UpdateCode`	Performs the update code operation for all tools, the same way as is the default behavior of the **Update Code** button and then clicking OK. If the configuration was modified in UI tools to generate only some sources, only these sources are generated. Select a saved configuration, for example via the `-Load` argument. The output folder is determined from the location of the saved configuration.	Requires the `-HeadlessTool` command.Requires selection of a saved configuration, for example, using the `-Load` argument.	-HeadlessTool Clocks -UpdateCode -Load C:/conf/conf.mex -Load C:/conf/conf.mex -HeadlessTool Clocks -UpdateCode **Note:** “-Load” can be both before or after “-HeadlessTool” argument, “-Update Code” must be after “-HeadlessTool”

Table 26. Commands supported in the framework*...continued*

Command name	Definition and parameters	Description	Restriction	Example
		That can be changed using the `-Project Link` argument.		

10.10.1 Command-Line execution - Pins Tool

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

Table 27. Commands supported in Pins

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	Requires -Headless Tool Pins -Enable
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,-Export CSV, -ExportHTML and in framework -Export MEX. 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

Table 27. Commands supported in Pins...*continued*

Command name	Definition and parameters	Description	Restriction	Example
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 -ExportRegisters C:/exports/regs

10.10.2 Command-line execution - Peripherals Tool

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

Table 28. 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, -Export HTML, and in the framework -Export MEX. 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 -Export HTML C:/exports/html
Migrate all Peripherals tool components to the versions used in the toolchain project	`-MigrateComponentsToToolchainVersion`	Migrates all instances of Peripherals tool components, which are not configuring the version used in	Requires the -HeadlessTool Peripherals command	-HeadlessTool Peripherals -Migrate ComponentsTo ToolchainVersion -

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

Command name	Definition and parameters	Description	Restriction	Example
		the toolchain project, to the version of SDK component used in the toolchain project. No changes are done when the toolchain is not linked or the Peripherals tool component already configures the version of the SDK component used in the toolchain project.		ExportAll C:/exports/ toolchainVersions
Migrate all Peripherals tool components to the highest supported version of the SDK component	`-MigrateComponentsToLatestVersion`	Migrates all instances of Peripherals tool components to the MCU's highest supported version of the SDK component, which the Peripherals tool component configures. No changes are done when already using the highest version available on the MCU.	Requires the -HeadlessTool Peripherals command	-HeadlessTool Peripherals -Migrate ComponentsToLatest Version -ExportAll C:/exports/toolchain Versions
Regenerate all PLU configurations	`-PLURegenerateConfigurations`	Regenerates the content of all PLU instances present in the configuration. Currently only the PLU instances in Verilog file and Verilog text modes are regenerated by new Verilog code synthesis and optimization of the model. Requires usage of mapping comment in the Verilog code.	Requires the`-HeadlessTool Peripherals` command	-Load C:/xyz. mex -Headless Tool Peripherals -PLURegenerate Configurations - Export All C:/exports/plu

10.10.3 Command-Line execution - TEE Tool

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

Table 29. 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

Table 29. Commands supported in TEE Tool*...continued*

Command name	Definition and parameters	Description	Restriction	Example
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 framework -Export MEX. The argument is expected to be a folder.	Requires the `-HeadlessTool TEE` command	-HeadlessTool TEE -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 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

10.11 Batch processing

Batch processing simplifies and speeds 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: Use this command without specifying other tool commands except "`-HeadlessTool`".

10.11.1 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.

10.11.1.1 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.

10.11.1.2 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
```

10.11.1.3 Steps list

The steps entity contains a list of steps to be processed for every listed path, similar to 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.

11 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.

12 Revision history

Table 30. Revision history

Document ID	Release date	Description
IMXUG v.19.0	17 September 2026	Updated for v.26.09
IMXUG v.18.0	24 June 2026	Updated for v.26.06
IMXUG v.17.0	25 March 2026	Updated for v.26.03
IMXUG v.16.0	16 December 2025	Updated for v.25.12
IMXUG v.15.0	18 September 2025	Updated for v.25.09
IMXUG v.14.0	20 June 2025	Updated for v.25.06
IMXUG v.13.0	17 March 2025	Updated for v.25.03
IMXUG v.12.0	15 January 2025	Updated for v.24.12
IMXUG v.11.0	24 September 2024	Updated for v.16.1
IMXUG v.10.0	1 July 2024	Updated for v.16
IMXUG v.9.0	19 April 2024	Updated for v.15.1
IMXUG v.8.0	16 February 2024	Boards with Serial Download mode/ Manufacture mode is updated.
IMXUG v.7.0	10 January 2024	Updated for v.15
IMXUG v.6.0	31 July 2023	Batch processing is added.
IMXUG v.5.0	2 January 2023	DDR Tool is updated.
IMXUG v.4.0	20 September 2022	Updated for v.12.1

Table 30. Revision history...*continued*

Document ID	Release date	Description
IMXUG v.3.0	30 June 2022	Updated for v.12
IMXUG v.2.0	22 December 2021	New features are added, screenshots are updated.
IMXUG v.1.0	01 July 2021	Minor changes
IMXUG 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