IMXUG

User Guide for Config Tools for i.MX

Rev. 7 — 10 January 2024 **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, 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.

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 Pins, TEE, DDR, SERDES, PBL tools.

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, in general, 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 function or channel. For example, for some ADC the routing provide connection between 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.

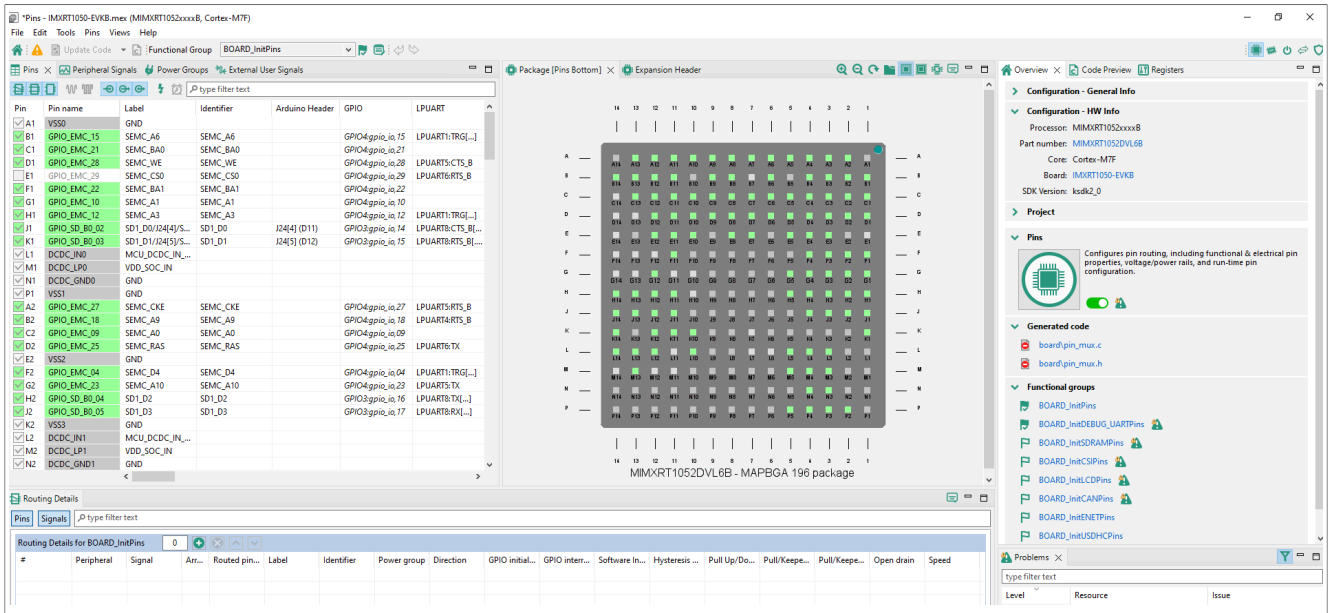


Figure 1. Config Tools for i.MX

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, tool tips, and help. You may need to 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****.*

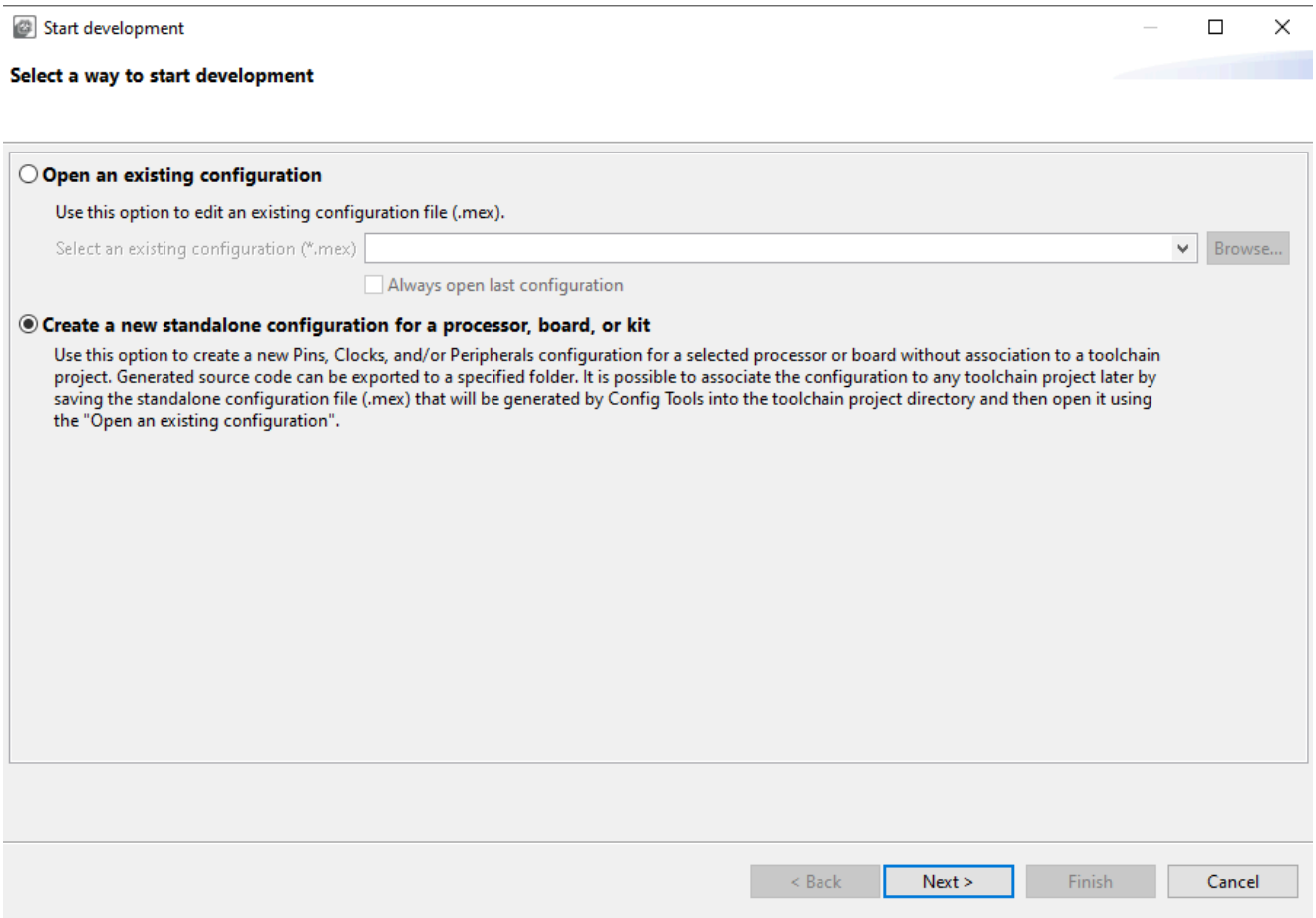


Figure 2. Start development wizard

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 recommended you start with example to create a configuration for a board or a kit. Such configuration contains board-specific settings. If you select a processor, the configuration will be empty.

After the new configuration is created, you can continue by importing an existing configuration from an MEX file. It 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.

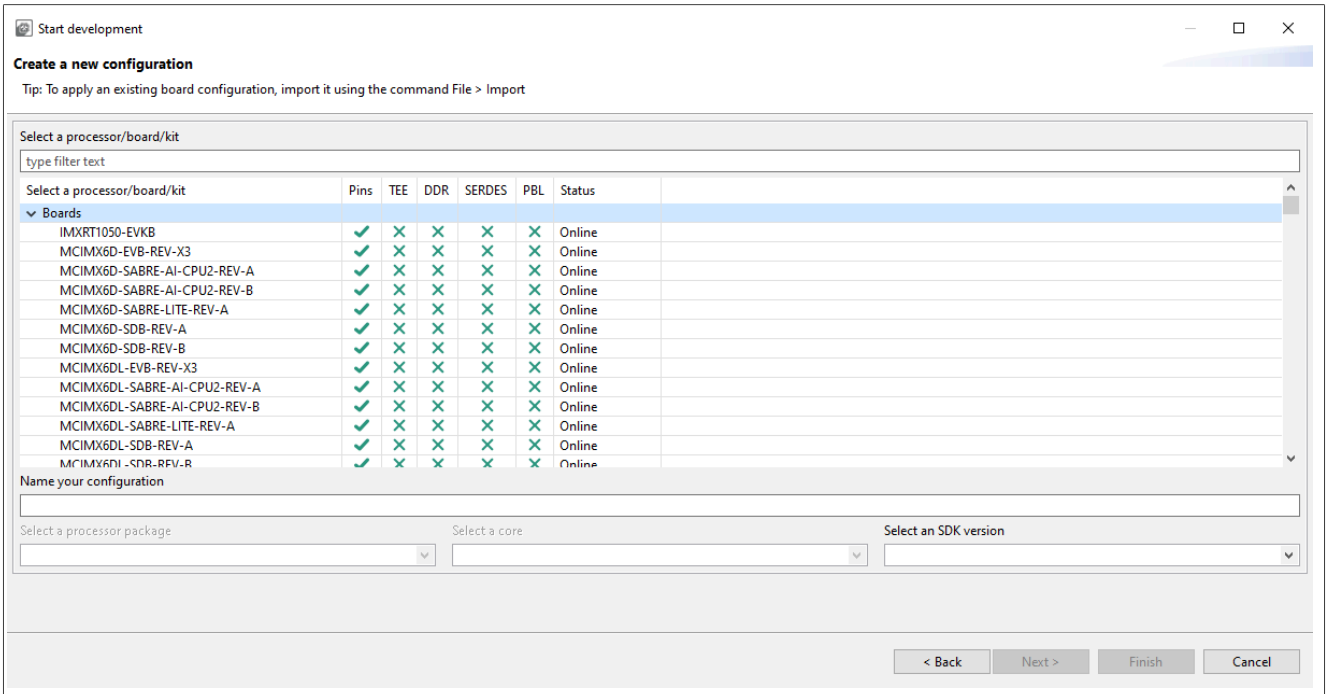


Figure 3. Creating a new configuration

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 will 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 be able 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 User templates

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

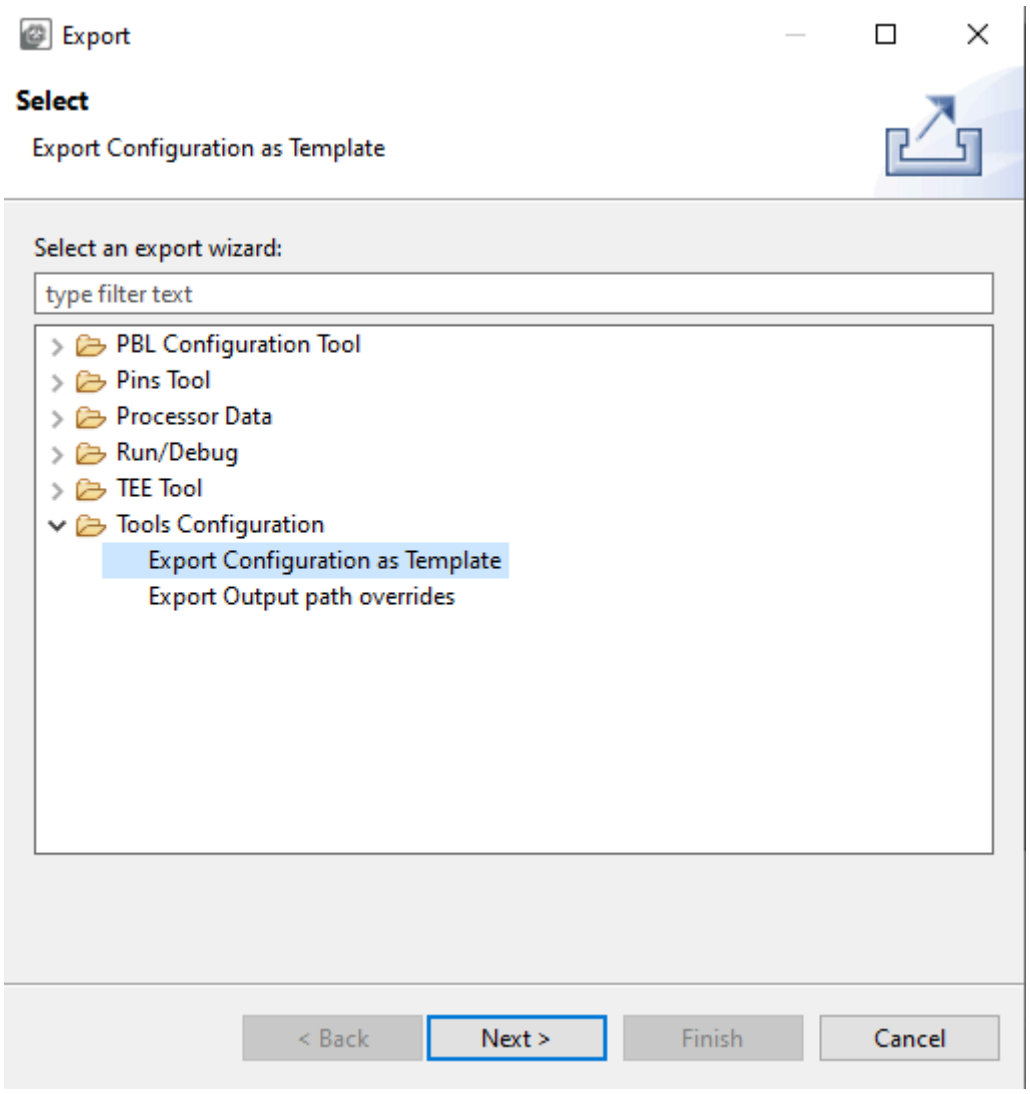


Figure 4. Export template

The exported template is available in the **New Configuration** wizard and can be used to create a configuration. You can also define custom labels for pins or identifiers prefixes for `#define` in generated code. You can export the configuration by selecting, in the **Menu bar**, **File** > **Export** > **Tools Configuration > Export Configuration as Template**.

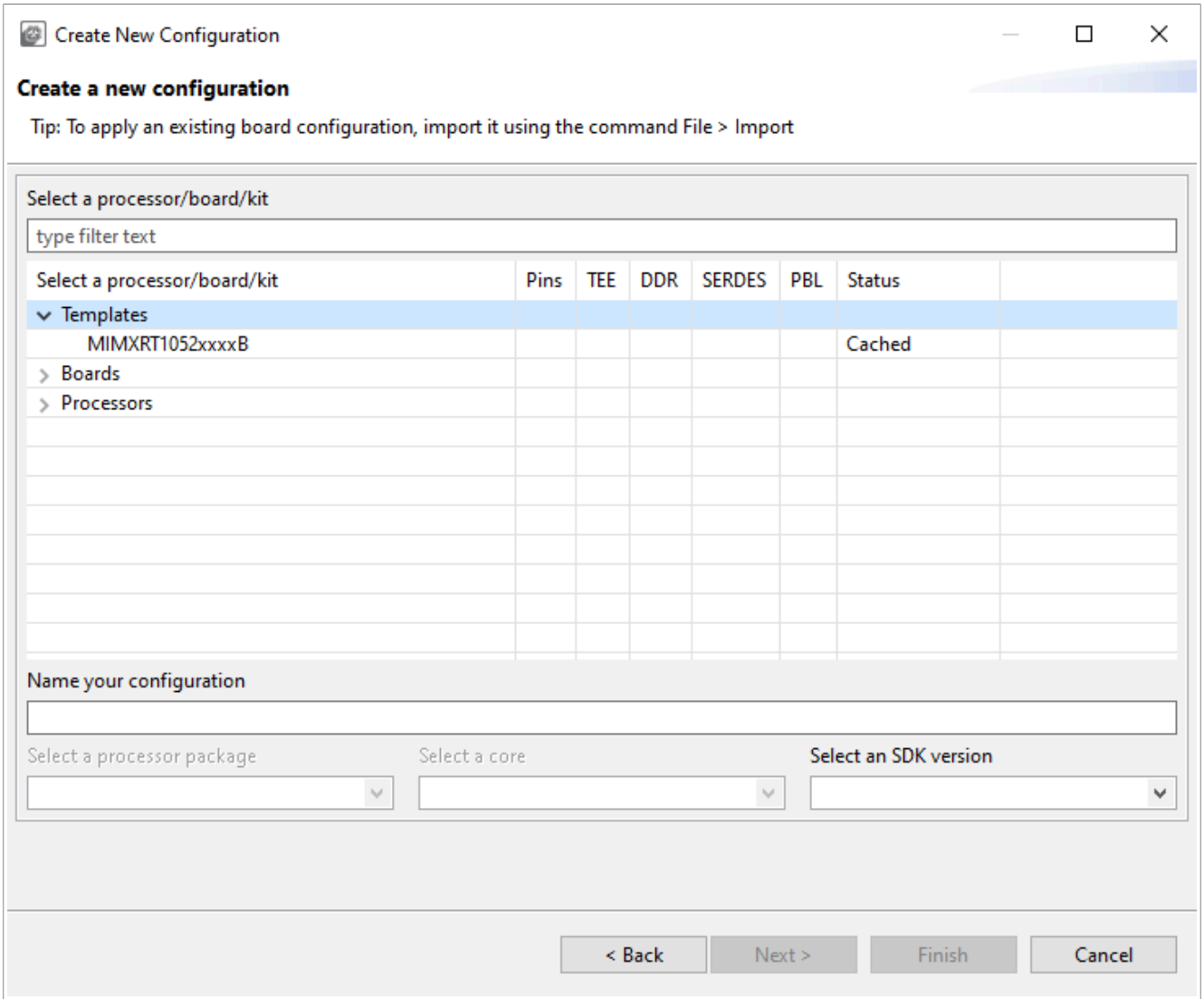


Figure 5. Create a new configuration from the template

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

2.2.5 Importing sources

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

Note: *You can import only C 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**.

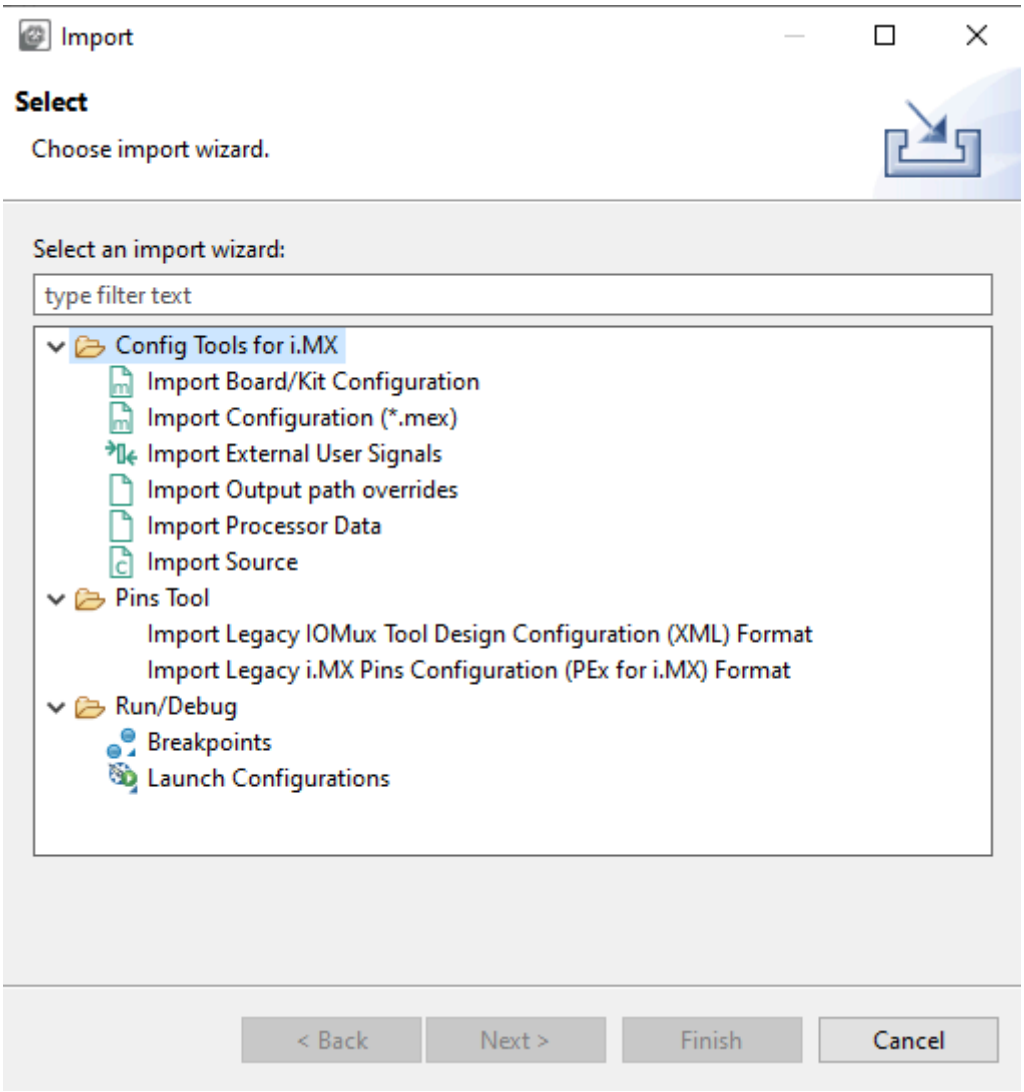


Figure 6. Import Source wizard

3. Click **Next**.
4. On the next page, click **Browse** to specify the location of the source file.
5. Select the source file that you wish to import and click **Open**.
6. On the next page, select which functional groups to import (based on tools) by selecting the checkbox in the left column.
7. Define how to import the functional groups by selecting one of the two available options in the dropdown menu in the right column:
   - **Rename** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, it is automatically renamed to the indexed one. For example, if `BOARD_InitPins` exists in the configuration then the imported function is renamed to `BOARD_InitPins1` .
   - **Overwrite** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, then the existing one is replaced with the imported one.
8. Click **Finish**.

2.2.5.1 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 that you wish to import and click **Open**.
6. On the next page, select which functional groups to import (based on tools) by selecting the checkbox in the left column.
7. Define how to import the functional groups by selecting one of the two available options in the dropdown menu in the right column:
   - **Rename** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, it is automatically renamed to the indexed one. For example, if `BOARD_InitPins` exists in the configuration then the imported function is renamed to `BOARD_InitPins1`.
   - **Overwrite** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, then the existing one is replaced with the imported one.
8. Click **Finish**.

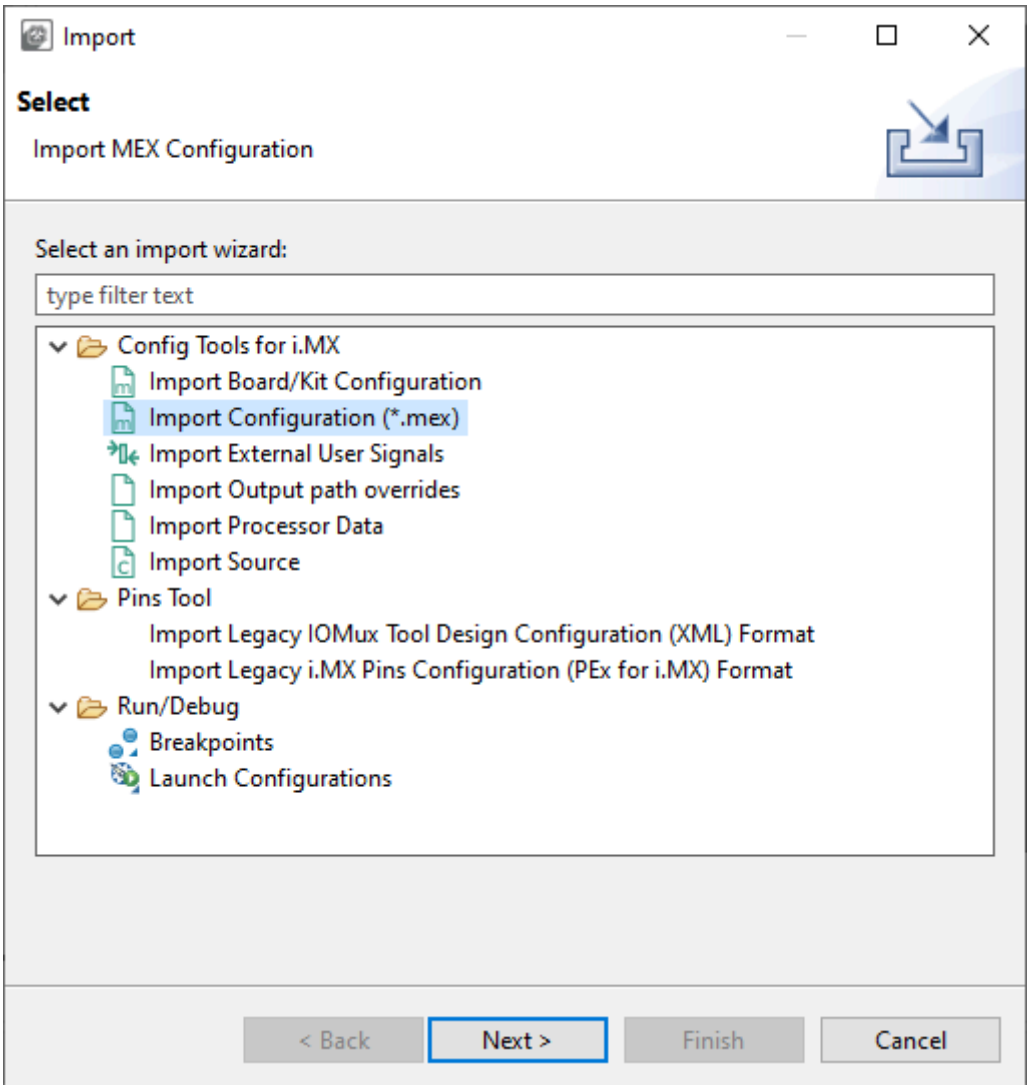


Figure 7. Import Wizard

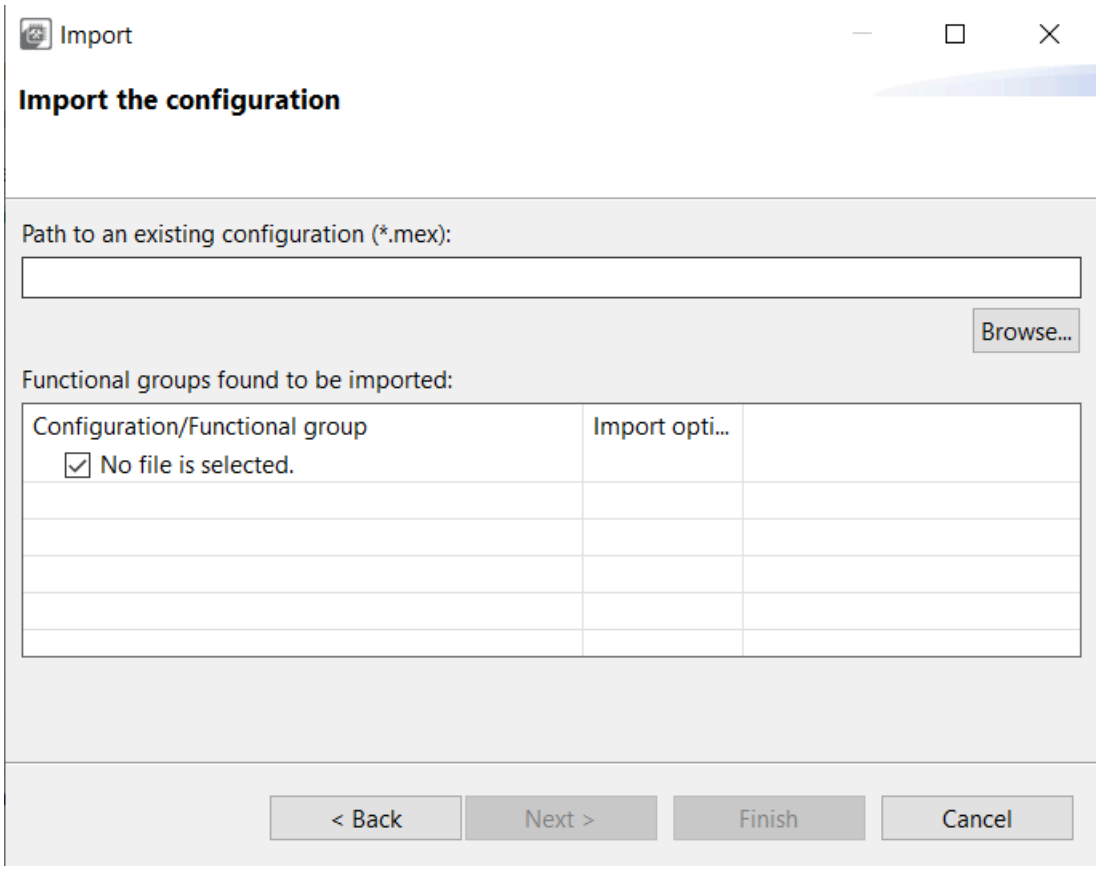


Figure 8. Import configuration

2.2.5.2 Importing Board/Kit Configuration

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

To import board/kit configuration, do the following:

1. In the **Menu bar**, select **File > Import...>**.
2. In the **Import** wizard, select **MCUXpresso Config Tools** > **Import Board/Kit Configuration**.
3. Click **Next**.
4. On the next page, select the board/kit variant from the dropdown menu.
5. Select which functional groups to import (based on tools) by selecting the checkbox in the left column.
6. Define how to import the functional groups by selecting one of the two available options in the dropdown menu in the right column:
   - **Rename** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, it is automatically renamed to the indexed one. For example, if BOARD_InitPins exists in the configuration then the imported function is renamed to BOARD_InitPins1.
   - **Overwrite** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, then the existing one is replaced with the imported one.
7. Click **Finish**.

2.2.6 Exporting sources

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

To launch the Export wizard:

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

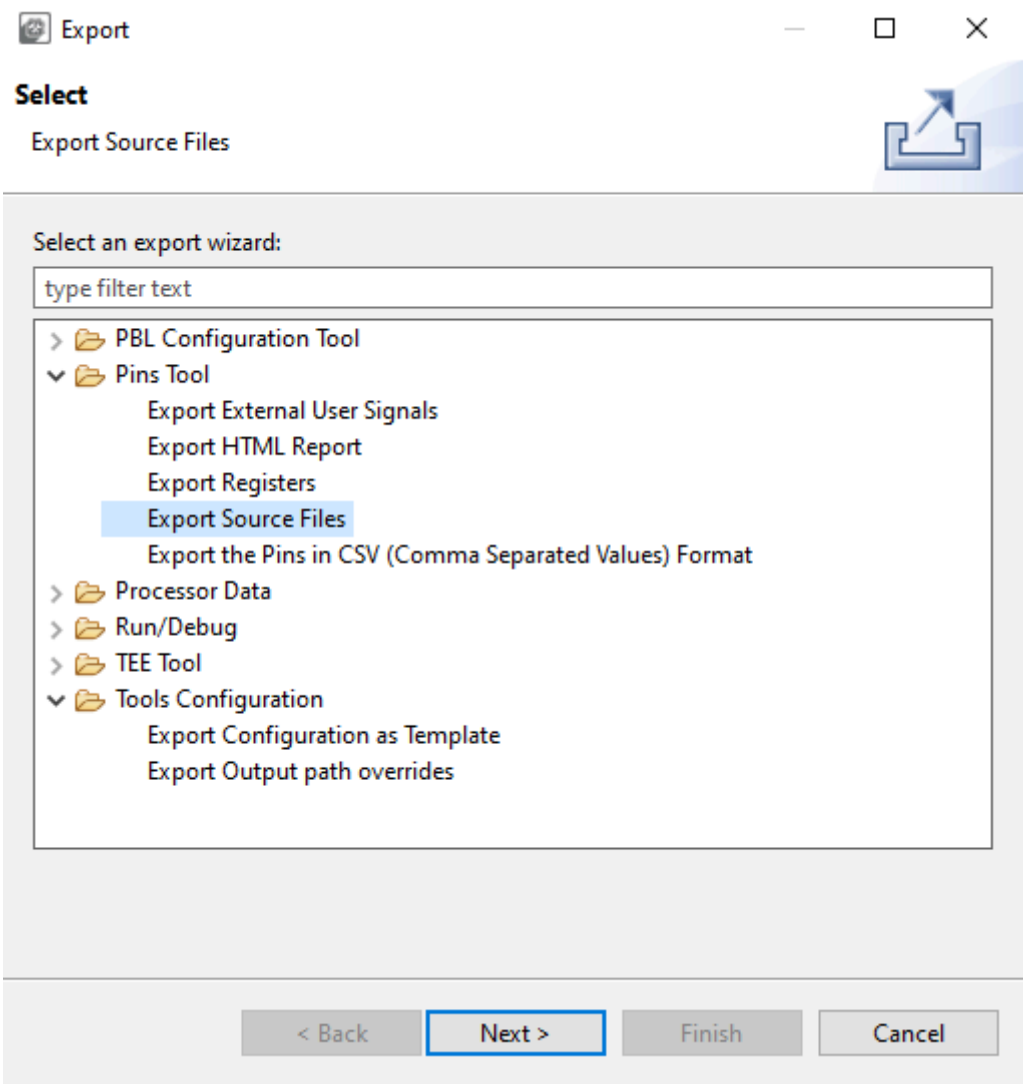


Figure 9. Export wizard

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

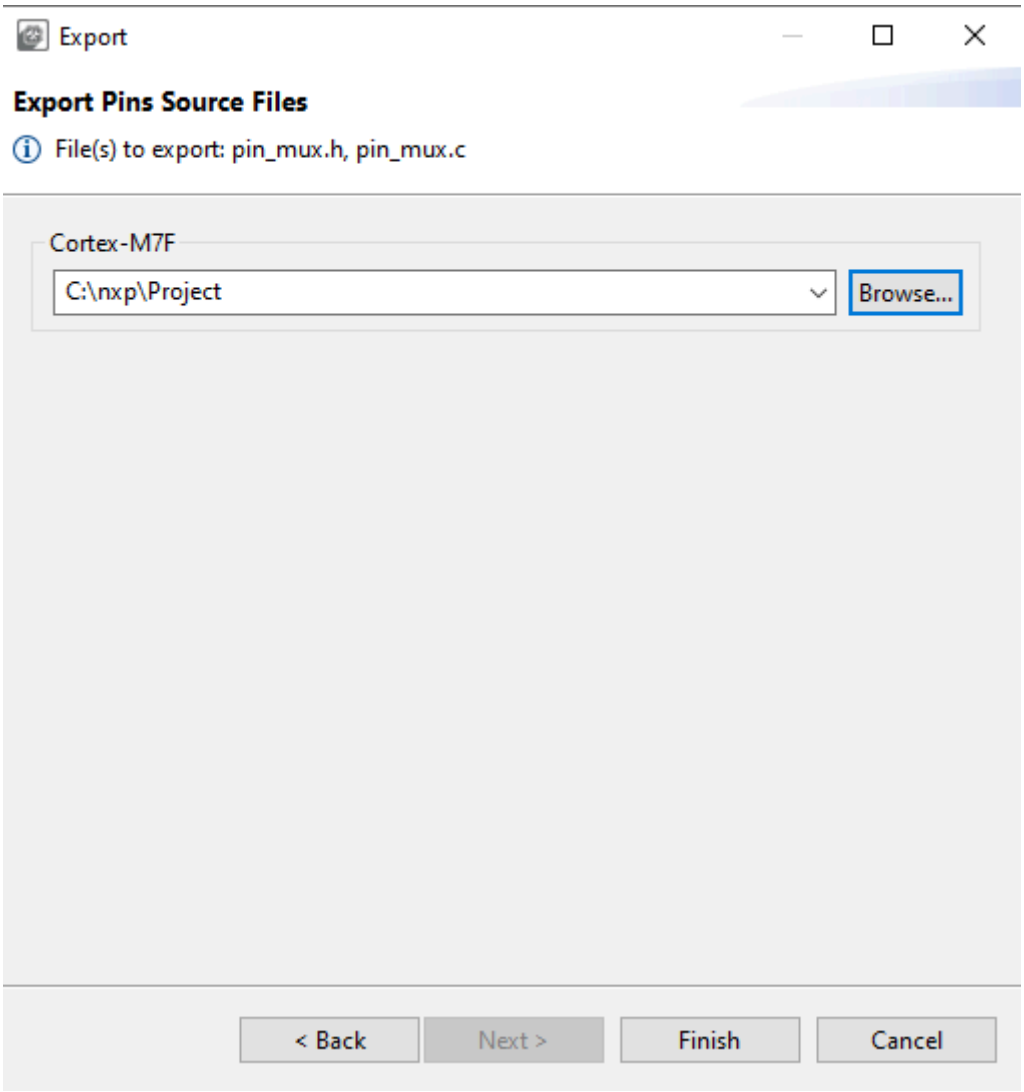


Figure 10. Export sources – Select target folder

5. In case of 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. File menu

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.
Save Copy As...	Create a backup copy of the current configuration.

Table 1. File menu...*continued*

Menu item	Description
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 Section 2.4.2 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. Pins menu

Item	Description
Functional Groups	Edit functional group properties.
Automatic Routing	Attempt to resolve 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.

Table 3. Pins menu*...continued*

Item	Description
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's 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. Help 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 on 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 allowing you to update generated peripheral initialization code directly within specified toolchain project.
Generate Code	Regenerate source code when "Enable Code Preview" preference is disabled.
Functional group selection	Select functional group. Functional group in the Peripherals tool represents a group of peripherals that are initialized as a group. The tool generates a C function for each function group that contains the initialization code.
Call from default initialization	Set the current functional group to be initialized by the default initialization function.
Functional group properties	Open the **Functional group properties** dialog to modify name and other properties of the function group.

Table 5. Toolbar*...continued*

Item	Description
Tool selection	Display icons of individual tools. Use them to switch between tools.
Undo/Redo	Undo/Redo last action.

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

2.4.1 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/DDR/SERDES/PBL/TEE	Displays basic information about **Pins, DDR, SERDES, PBL, TEE** tools.

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

Table 7. Config Tools Overview

Item	Description
Generated code	Contains the list of source-code files. Click the links to open the files in the **Code Preview** view.
Functional groups	Contains the list of the currently active functional groups. To select the groups in the **Functional groups** tab in the toolbar, select the relevant links.

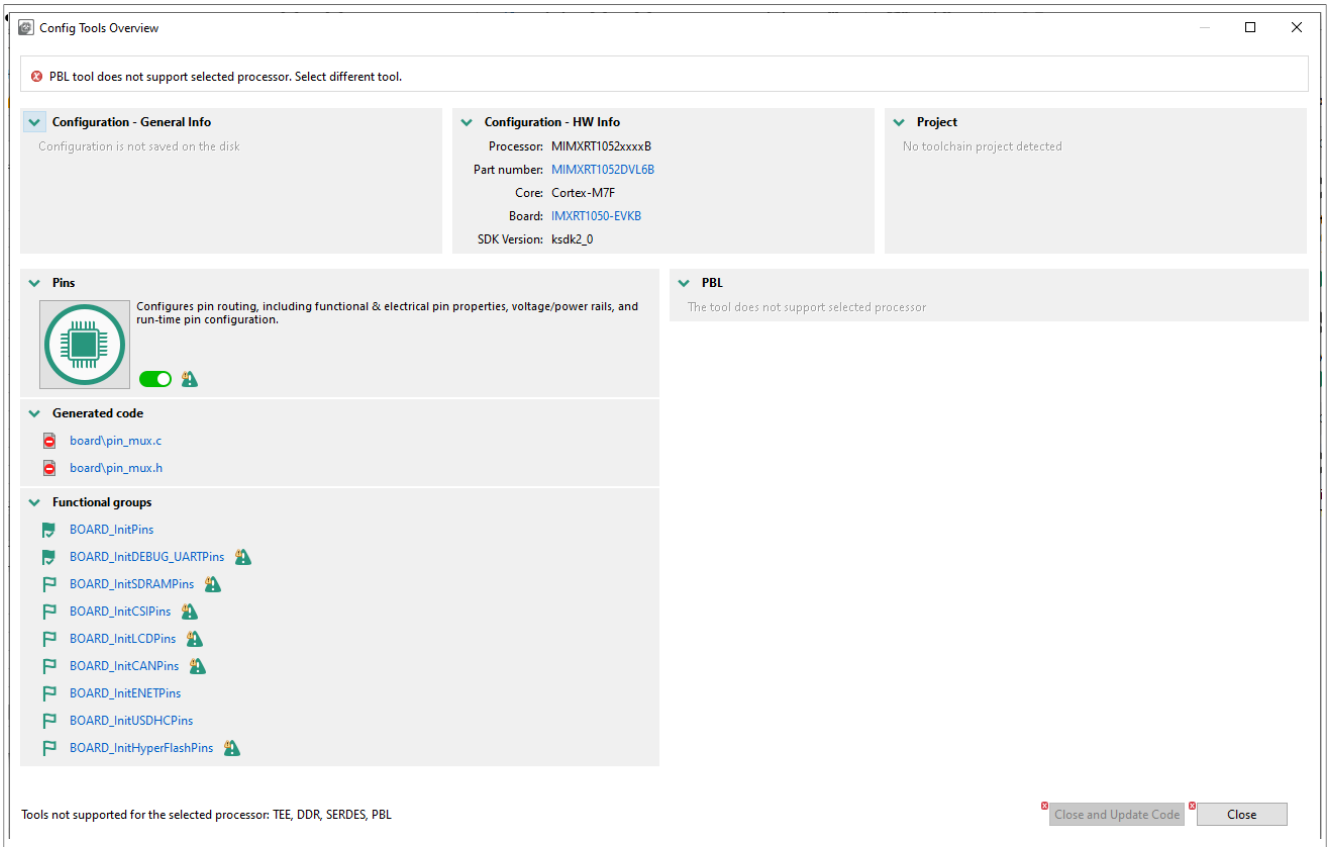


Figure 11. Config Tools Overview

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**.

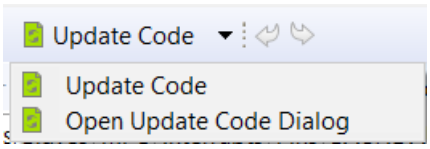


Figure 12. Update Code dropdown menu

Note: *The generated code is always overwritten.*

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

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

2.4.3 Functional groups

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

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

Figure 13. Functional groups

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

Table 8. Functional Groups

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

Note:

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

2.4.3.1 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 generated function name, select core (for multicore processors only) that is affecting the generated source code, or write function description (this description is generated in the C file). You can also add, copy, and remove functional groups as needed.

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

Functional group properties are specific for individual Config Tools:

The Pins tool:

- **Set custom #define prefix** - If this property is set, the specific custom prefix is used for macros generated into the `pin_mux.h`. Otherwise the name of the functional group is used as the prefix.
- **Prefix** - The custom prefix string. If it is empty, no prefix is used.
- **Clocks gate enable** - If this property is enabled, the clock gate is enabled in the generated code. The clock gate is needed for access to the peripherals, so have it enabled elsewhere.
- **Core** (for multicore processors only) - Selects the core that is used for executing this function.
- **Full pins initialization** - If this property is set, all features of the pins are fully initialized in the generated function even if matches the after-reset state of the processor. If it is not set, the value may be "not specified" or "Reset (...)" that means no code is generated and after-reset state is expected.
- **De-initialization function** - If this feature is set, an additional function that sets all pins and peripheral signals in this functional group to their after-reset state is generated. The new function has a suffix `_deinit` by default.
- **Set custom de-initialization function name** - Allows specifying a user-defined name of the de-initialization function.

Clocks tool:

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

Peripherals tool:

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

TEE tool:

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

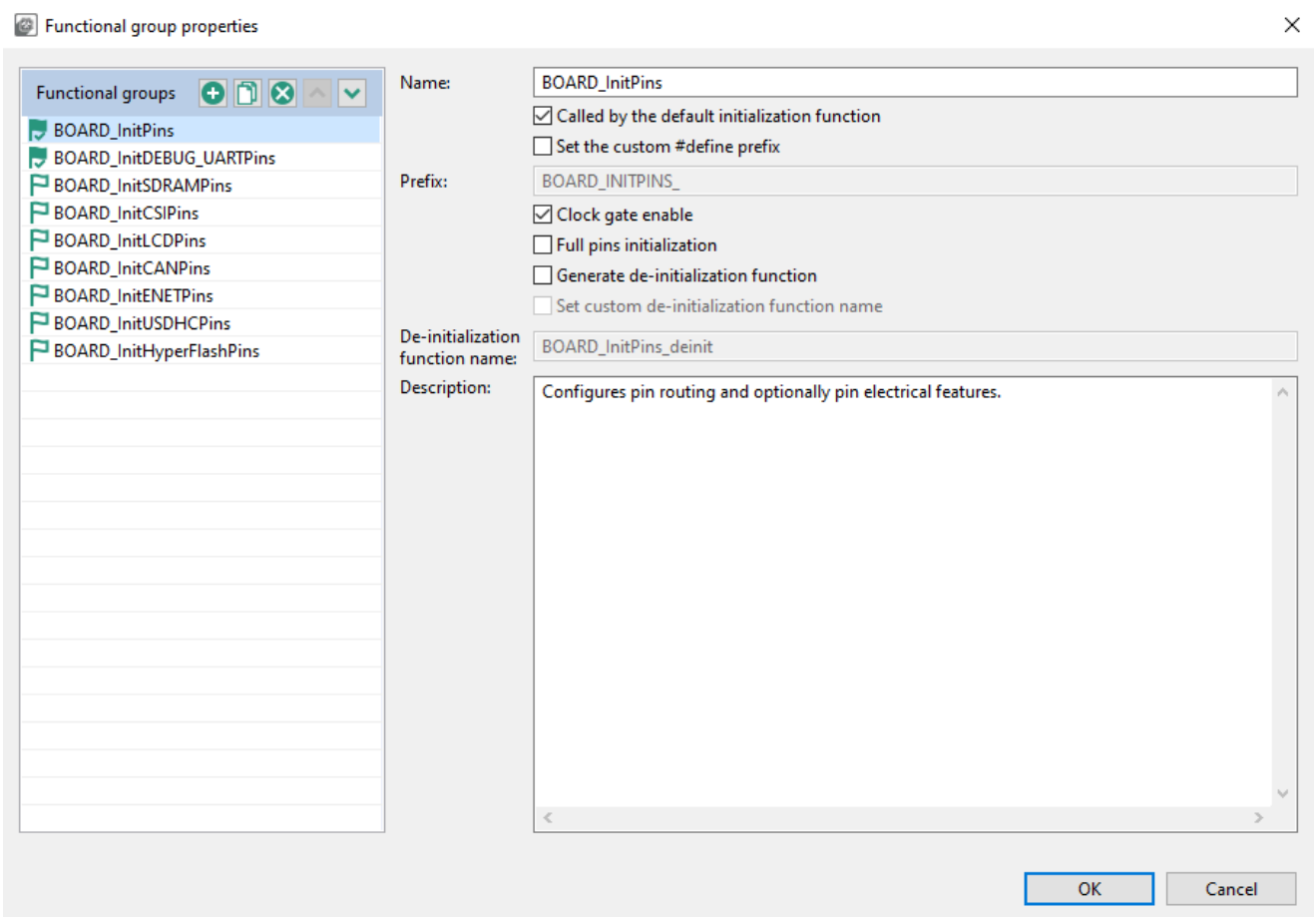


Figure 14. Functional group properties for the Pins tool

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/reto 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.*

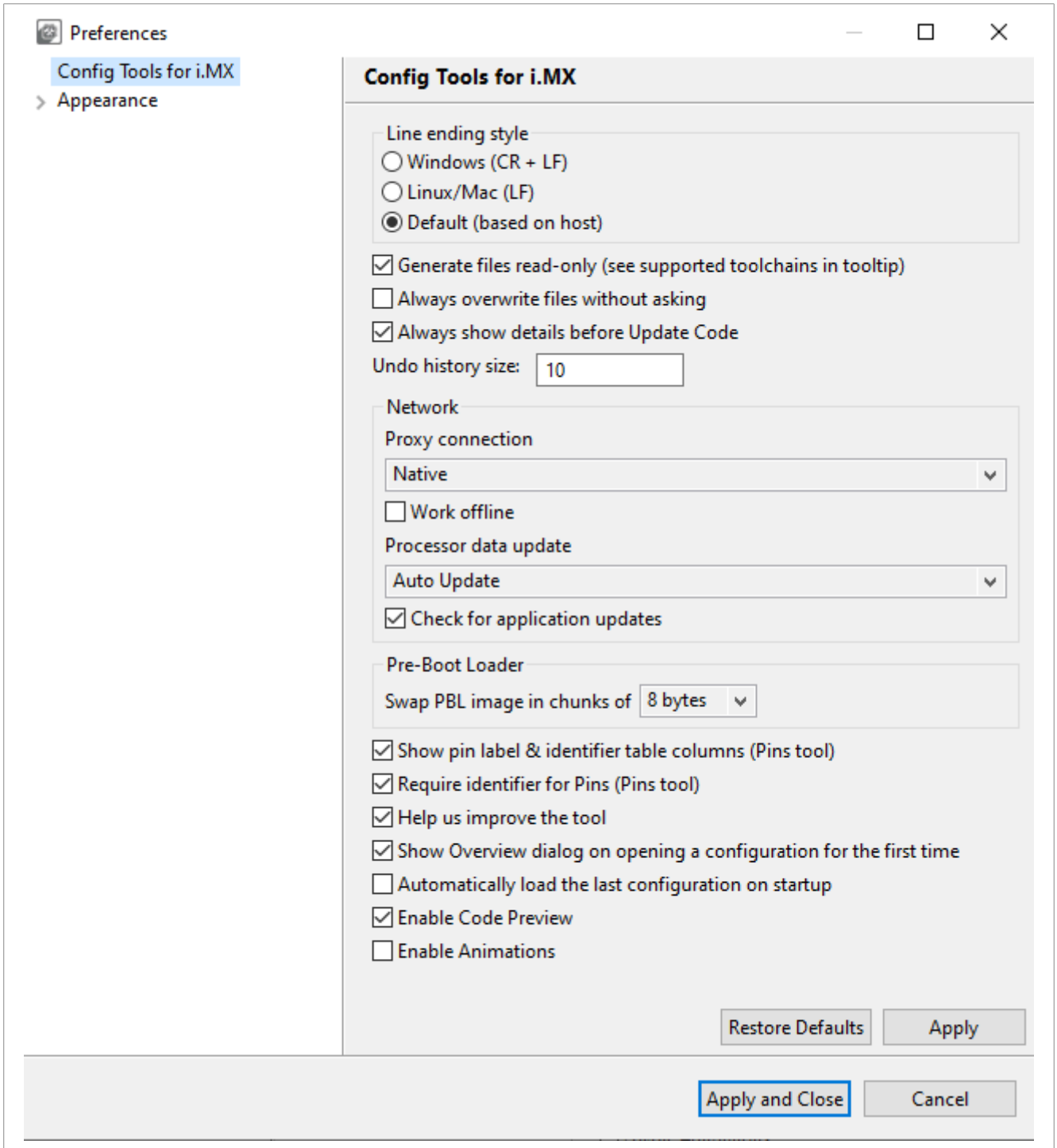


Figure 15. Preferences

Several settings are available.

Table 10. Preferences

Item	Description
Line ending style	Select between **Windows (CR + LF)**, **(LF)**, or **Default (based on host)**.
Generate source folder	At build time, automatically create a folder including source files.
Create empty configuration if no yaml is available	Generates a configuration even if no yaml is present.
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.
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 to NXP 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 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, warning message is displayed

Table 10. Preferences...*continued*

Item	Description
	in 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.
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.

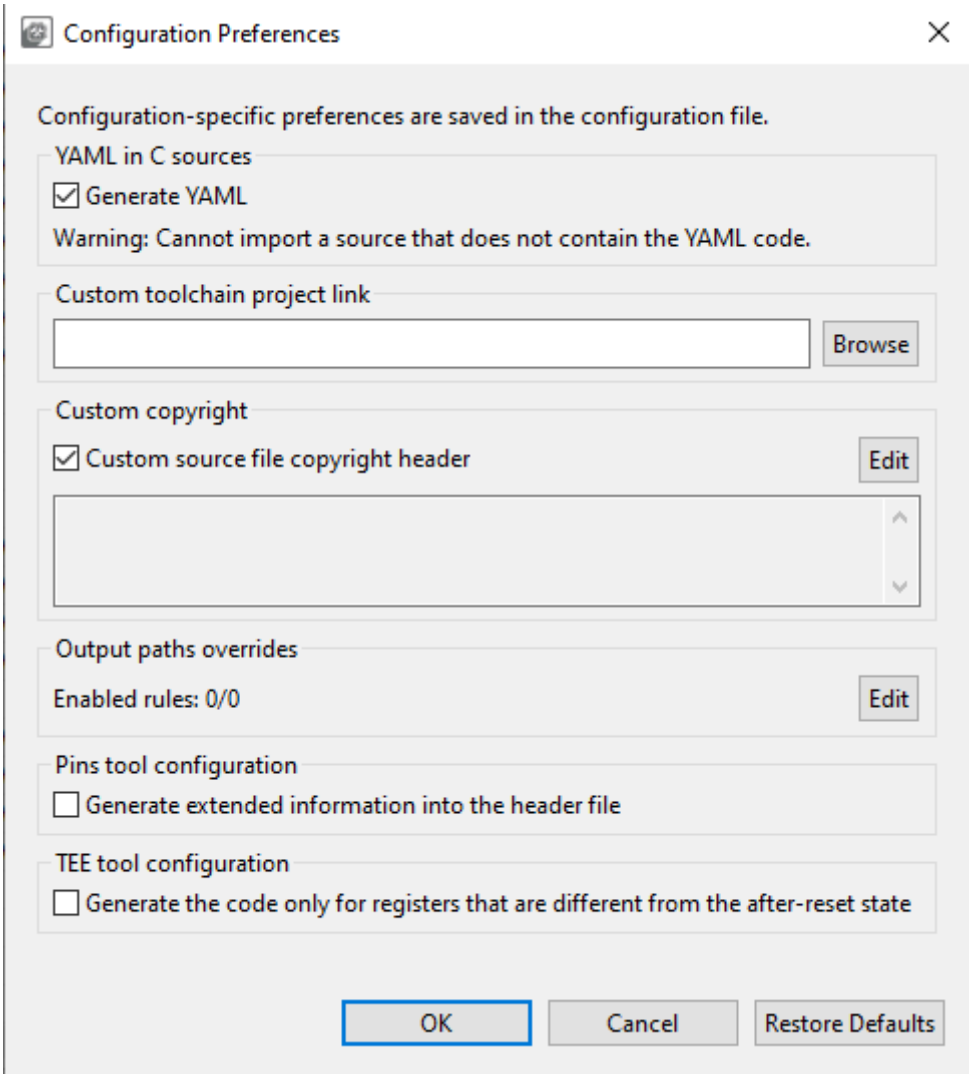


Figure 16. Configuration preferences

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.
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 are used to override the path of the output files are generated by the tools. They are applied in the **Update code** and **Exports** commands. A special dialog allows editing.
Generate extended information into header file	Select to generate extended information into the header file. For projects created in earlier MCUXpresso versions, this option is selected by default.
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.

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.

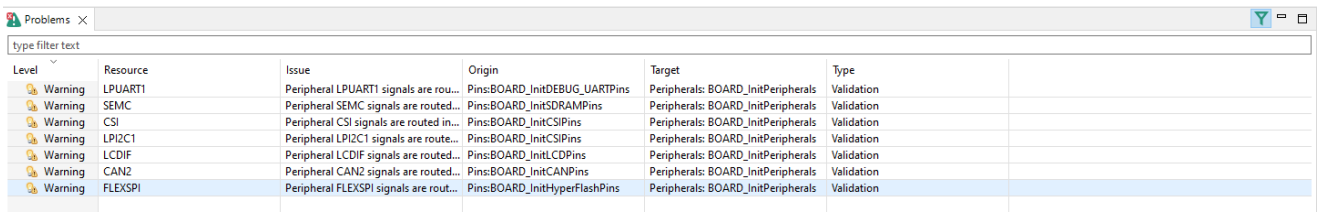


Figure 17. Problems view

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 view

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

Every issue comes with a context menu accessible by right-clicking the table row. Use this menu to access information about the problem or to apply a quick fix where applicable. You can also copy 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. Filter 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 state that is in the registers by default after the reset. The values of the registers are displayed in the hexadecimal and binary form. If the value of the register (or bit) is not defined, an interrogation mark "?" is displayed instead of the value.

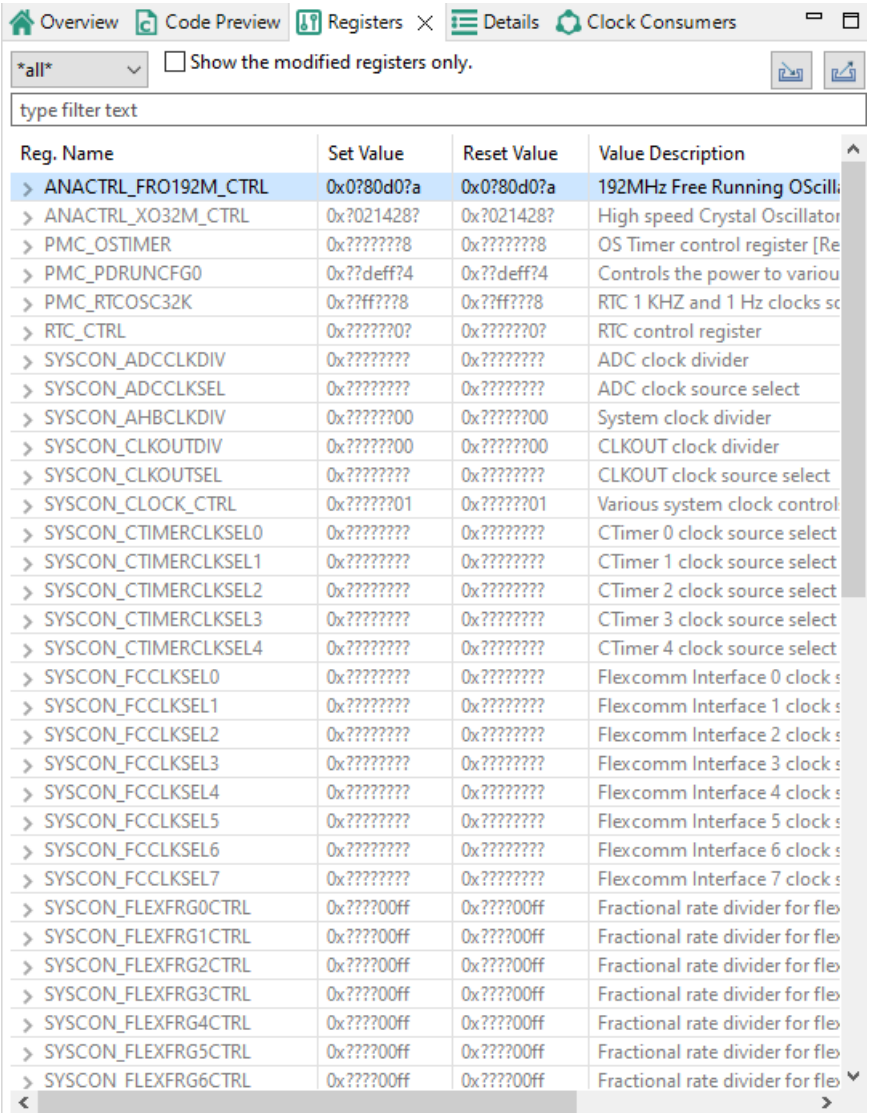


Figure 18. Registers view

The **Registers view** contains several items.

Table 14. Registers

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

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

Table 15. Color codes

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

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

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.

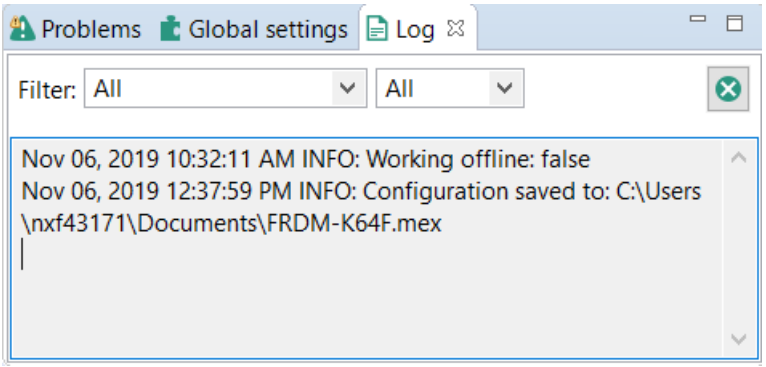


Figure 19. Log view

3 Pins Tool

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

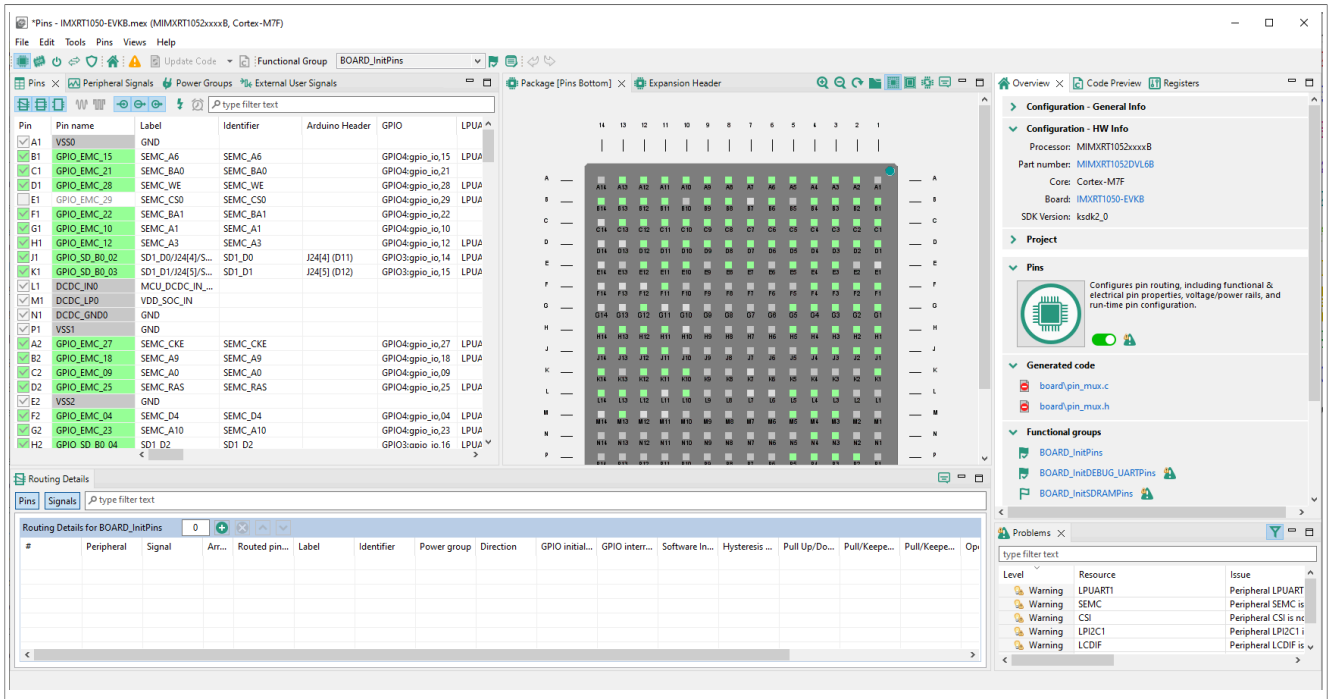


Figure 20. Pins tool

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**

Following two sections describe the two methods that 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:
   - Exclamation mark and default text color indicate that such item selection can cause a register conflict or the item does not support selected signal.
   - Exclamation mark and gray text color indicate that the item cannot be routed to the selected pin/internal signal. The item is available for different pin/internal signal using the same signal.
   Note: *In the **Pins** view and the **Package** view, you can configure only pins and not internal signals.*

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

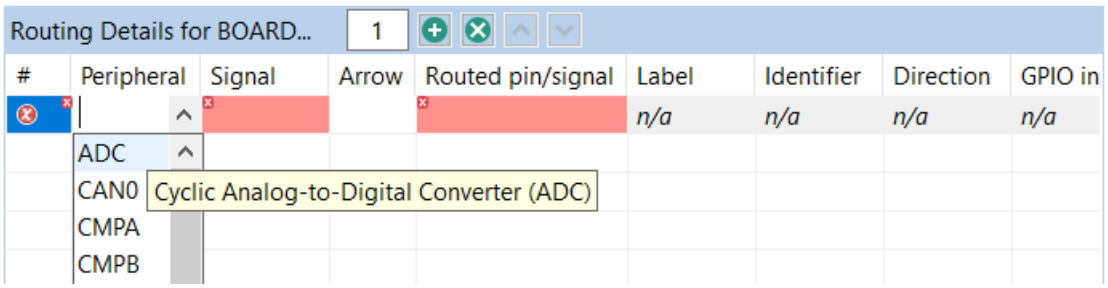


Figure 21. Selecting the peripheral (B)

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

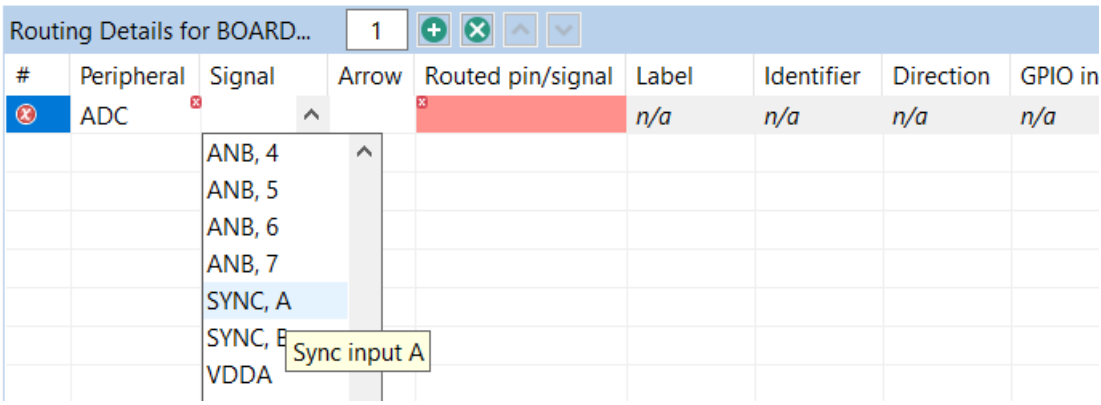


Figure 22. Selecting the input node (B)

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

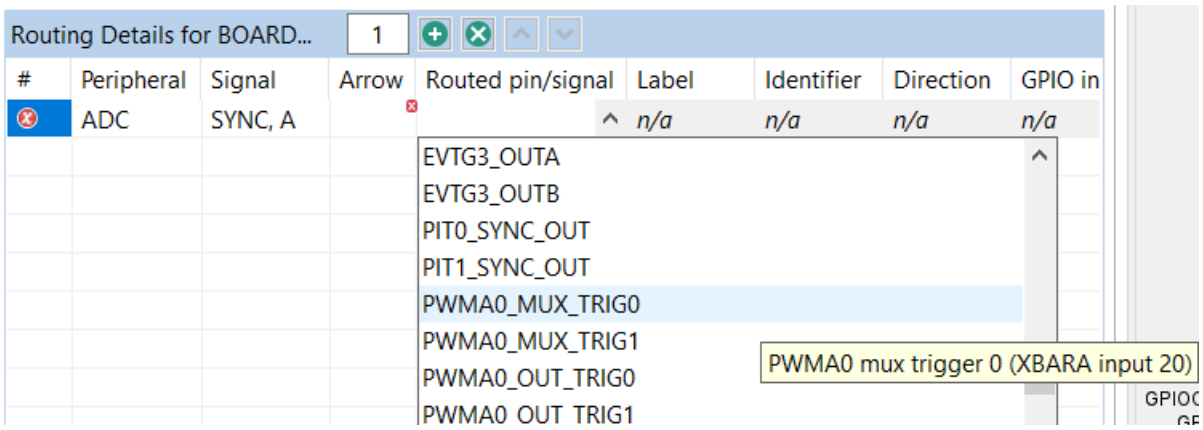


Figure 23. Selecting the output signal

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 example):

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

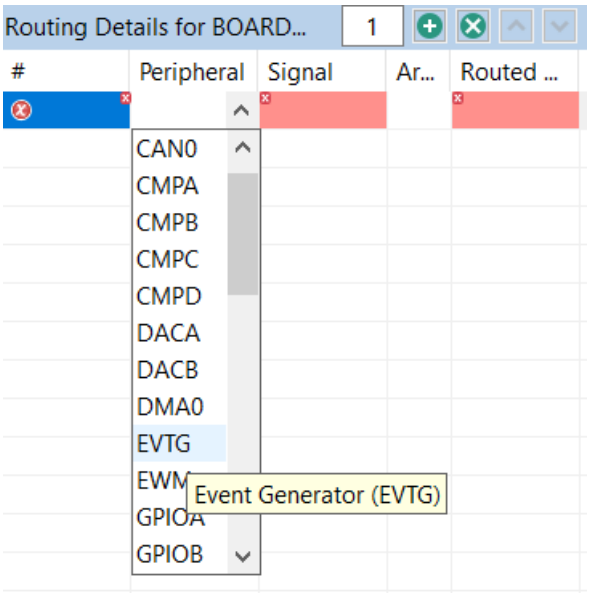


Figure 25. Selecting the peripheral (B)>

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

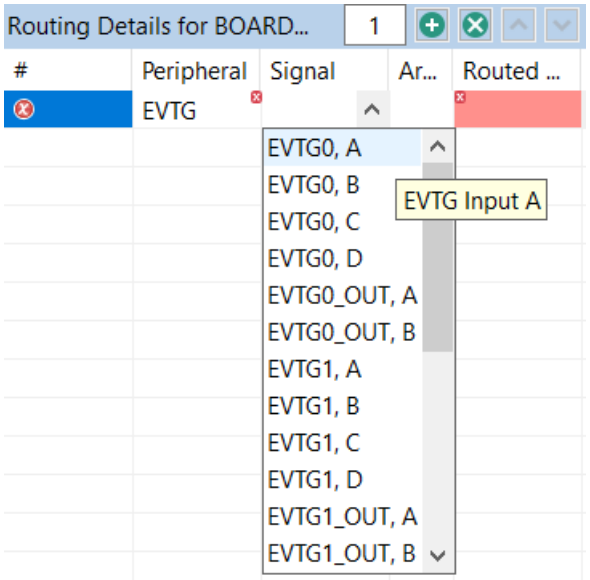


Figure 26. Selecting the input node (B)

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

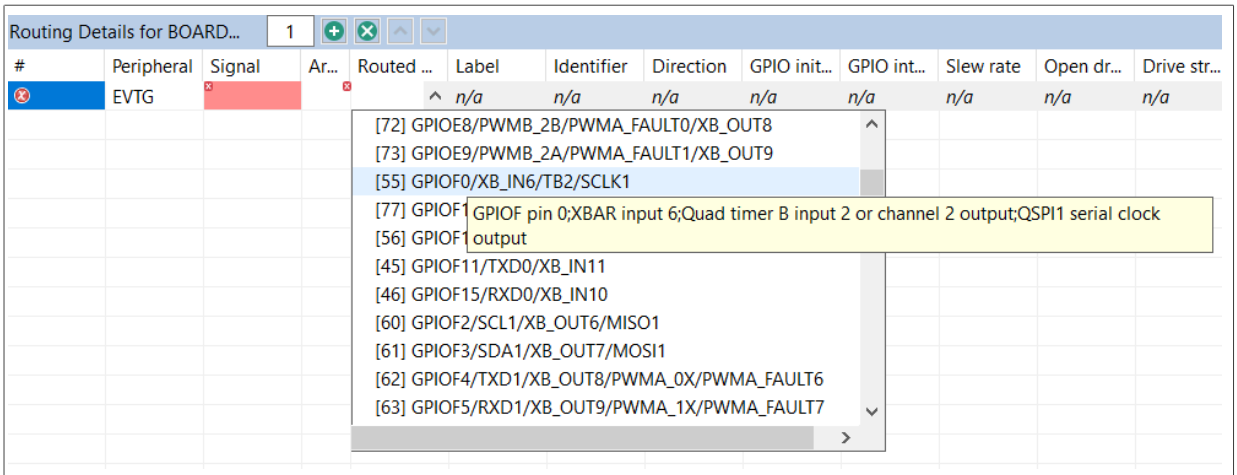


Figure 27. Selecting the pin

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

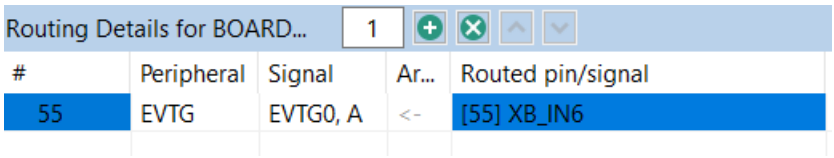


Figure 28. Result

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

3. Routing the signal from 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.

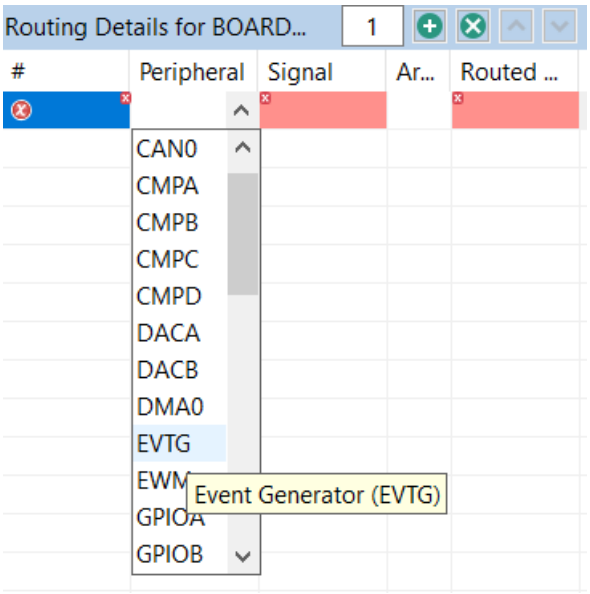


Figure 29. Selecting the peripheral (A)

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

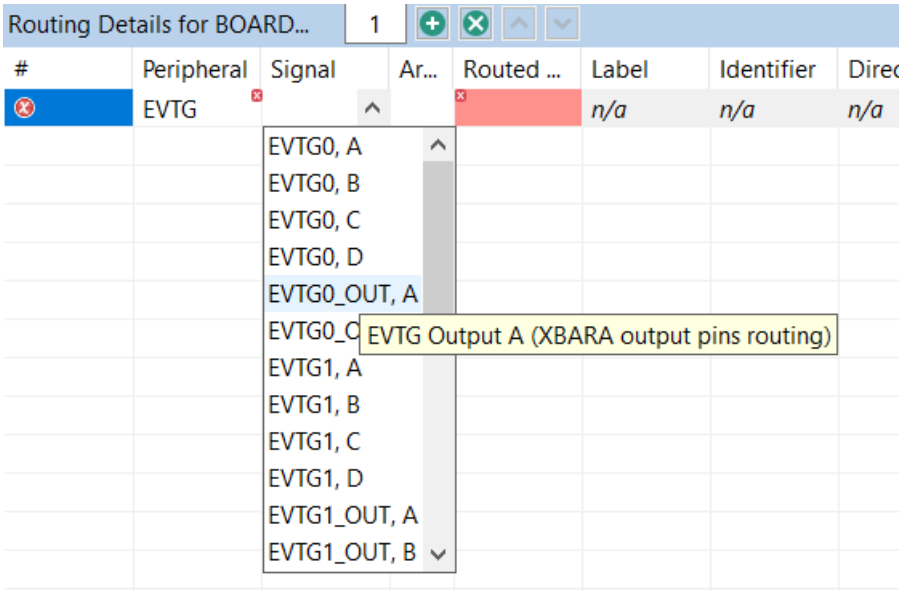


Figure 30. Selecting the output signal (A)

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

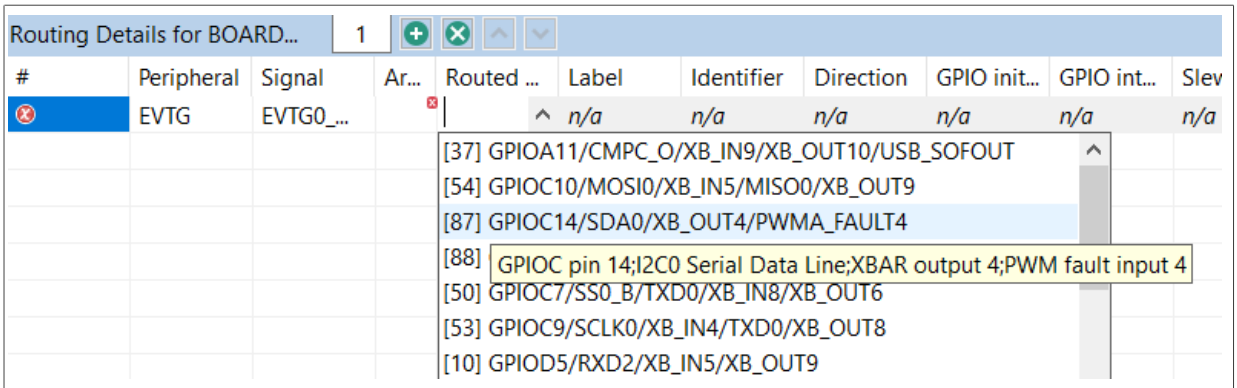


Figure 31. Selecting the pin

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

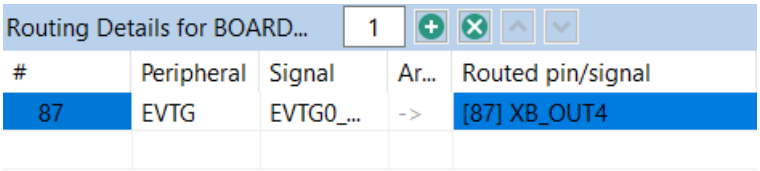


Figure 32. Result

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

3.2 Example usage

This section lists the steps to create an example pin configuration, which can then be used 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 generated files with application modifying the default board configuration.

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

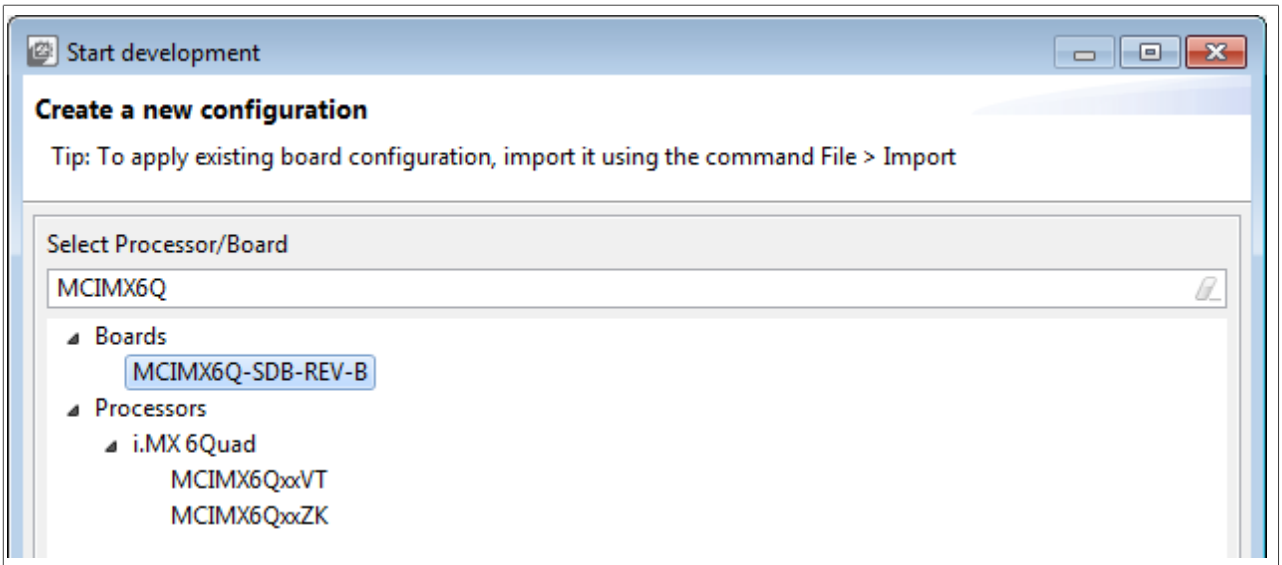


Figure 33. Create a new configuration

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

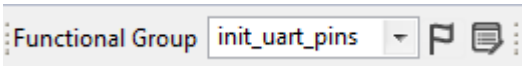


Figure 34. Select function

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

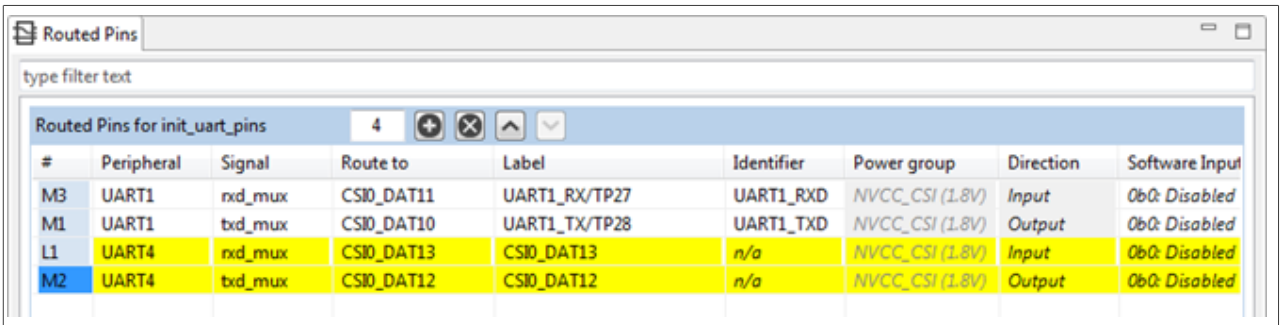


Figure 35. Routed Pins view

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 configuration of red LED pin routed originally to GPIO_2 pad, you must to select functional group 'init_gpio_pins'.

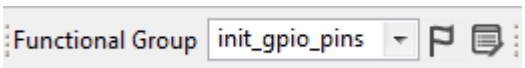


Figure 36. Select function

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 37. 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.

Code Preview

imx6q-board.dtsi@Cortex-A9 (Core #0) | pin_mux.c@Cortex-A9 (Core #0) | pin_mux.h@Cortex-A9 (Core #0)

```
/*
 * TEXT BELOW IS USED AS SETTING FOR TOOLS ************************************
!!GlobalInfo
product: Pins v4.1
processor: MCIMX6QxxVT
package_id: MCIMX6Q7CVT08AD
mcu_data: i_mx_1_0
processor_version: 3.0.0
board: MCIMX6Q-SDB-REV-B
power_domains: {PCIE_VPH: '2.5', SATA_VPH: '2.5', NVCC_SD3: '3.3', NVCC_NANDF: '3.3'
  NVCC_EIM0: '3.3', NVCC_MIPI: '2.5', NVCC_EIM1: '3.3', HDMI_VPH: '2.5', NVCC_EIM2:
  NVCC_ENET: '3.3', NVCC_DRAM: '1.5'}
 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS ***********
 */

/dts-v1/;

#include "skeleton.dtsi"
#include "imx6q-pinfunc.h"

/ {
    model = "Freescale i.MX 6Quad User Board";
    compatible = "fsl,imx6q-board", "fsl,imx6q";

    soc {
        #address-cells = <1>;
        #size-cells = <1>;

        iomuxc: iomuxc@020e0000 {
            compatible = "fsl,imx6q-iomuxc";
            reg = <0x020e0000 0x4000>;
        };
    };
};

&iomuxc {
    pinctrl-names = "default";
```

Figure 38. 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) in order 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.

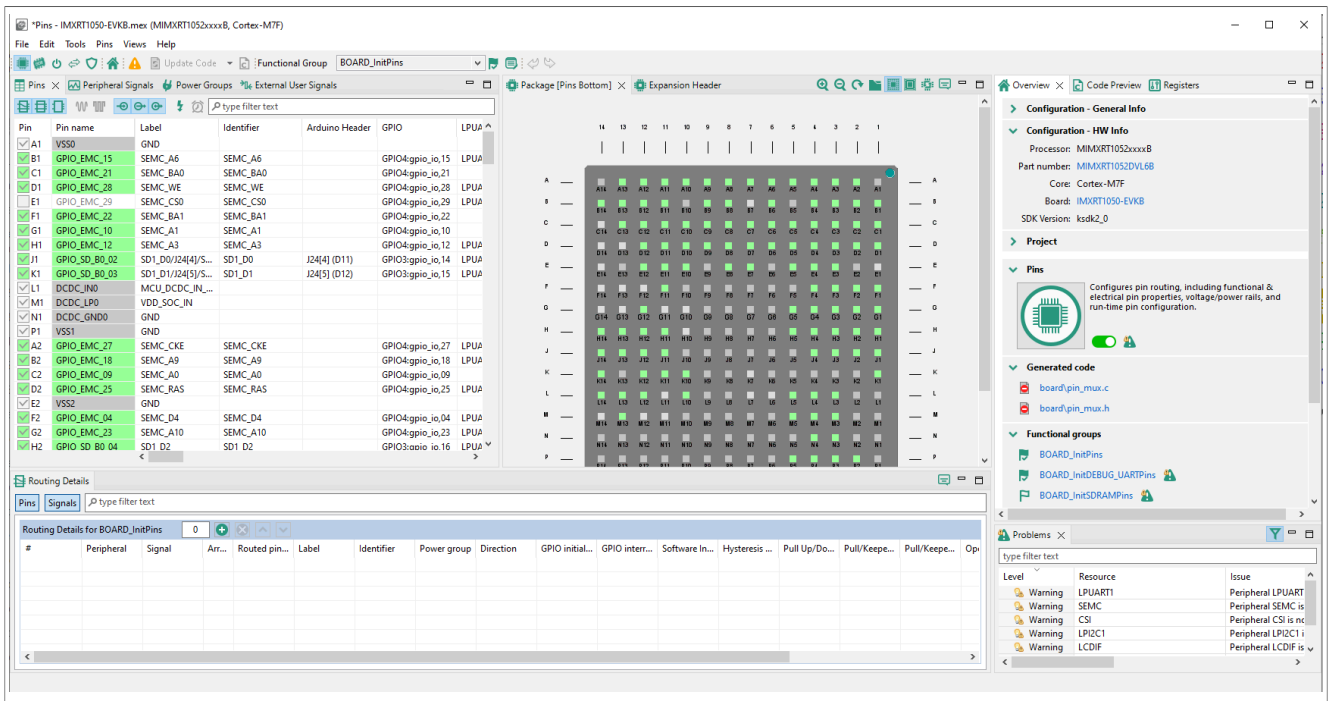


Figure 39. Pins tool user interface

3.3.1 Pins view

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

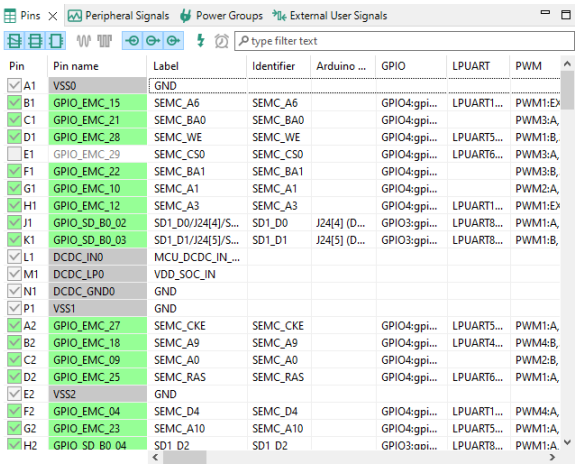


Figure 40. Pins table view

This view shows the list of all the pins available on a given device. The **Pin name** column shows the default name of the pin, or if the pin is routed. The next columns are optional. They are **Label**, **Identifier, External**

User Signals and **Expansion header connections** (One column for each expansion header). The pin name is changed to show appropriate function for selected peripheral if routed. The next column of the table shows peripherals and signals and pin name(s) on given peripheral. Peripherals with few items are cumulated in the last column.

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

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

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

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.

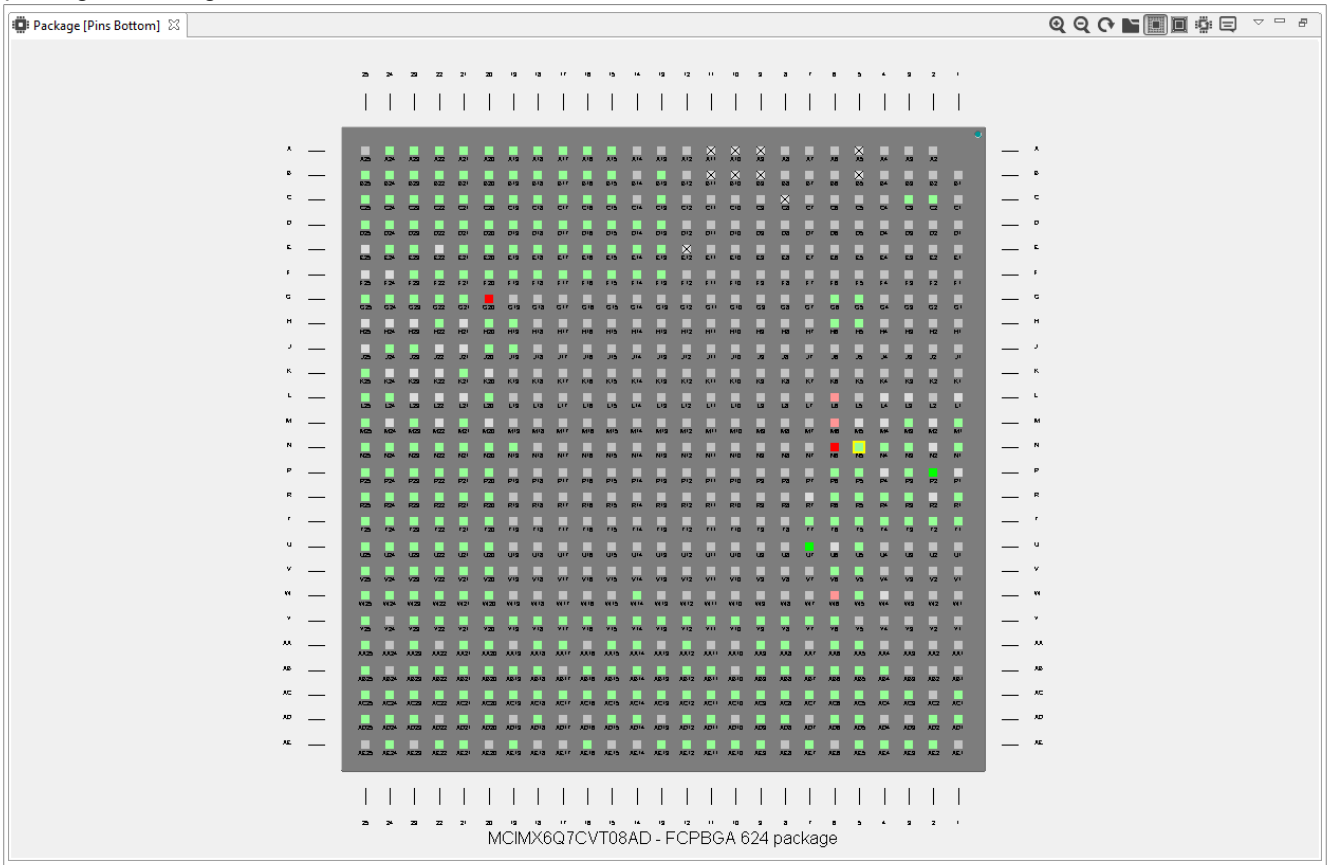


Figure 41. Processor package

This view shows package overview with pins location. 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 options

Icon	Description
	Zoom in package image.
	Zoom out package image.
	Rotate package image.
	Show pins as you can see it from the bottom. This option is available on BGA packages only.
	Show pins as you can see it from the top. This option is available on BGA packages only.
	Show resources. This option is available on BGA packages only.
	Switch package.
	Package legend.
	Select the information displayed as pin labels. This option is not available on BGA packages.

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

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

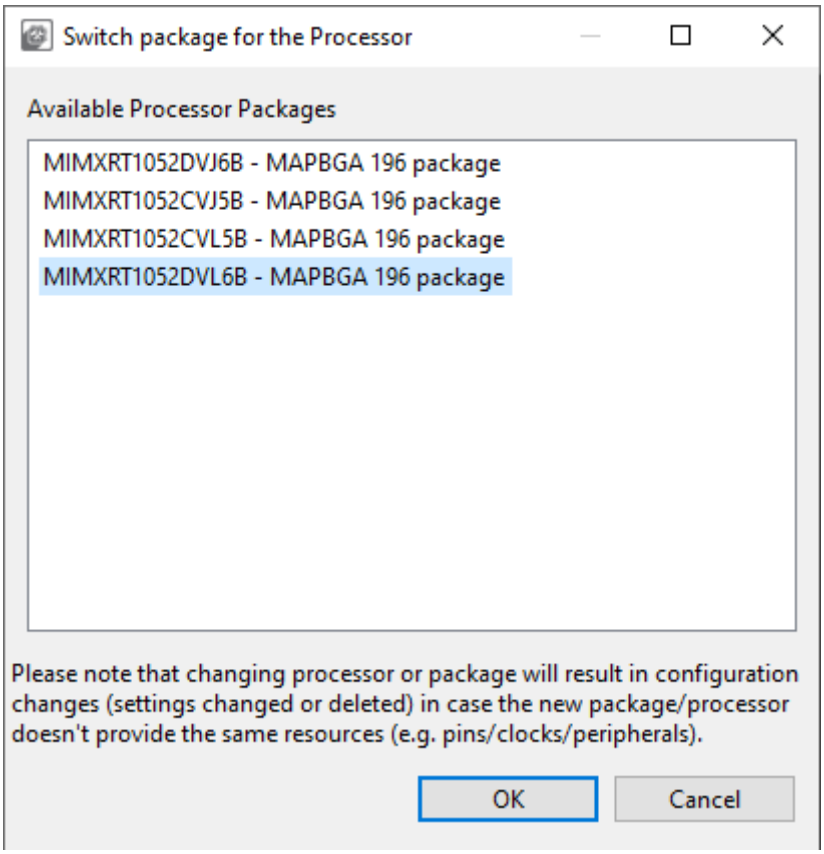


Figure 42. Switch package

The **Switch package for the Processor** window shows list of available processor packages, showing 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. Status codes

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

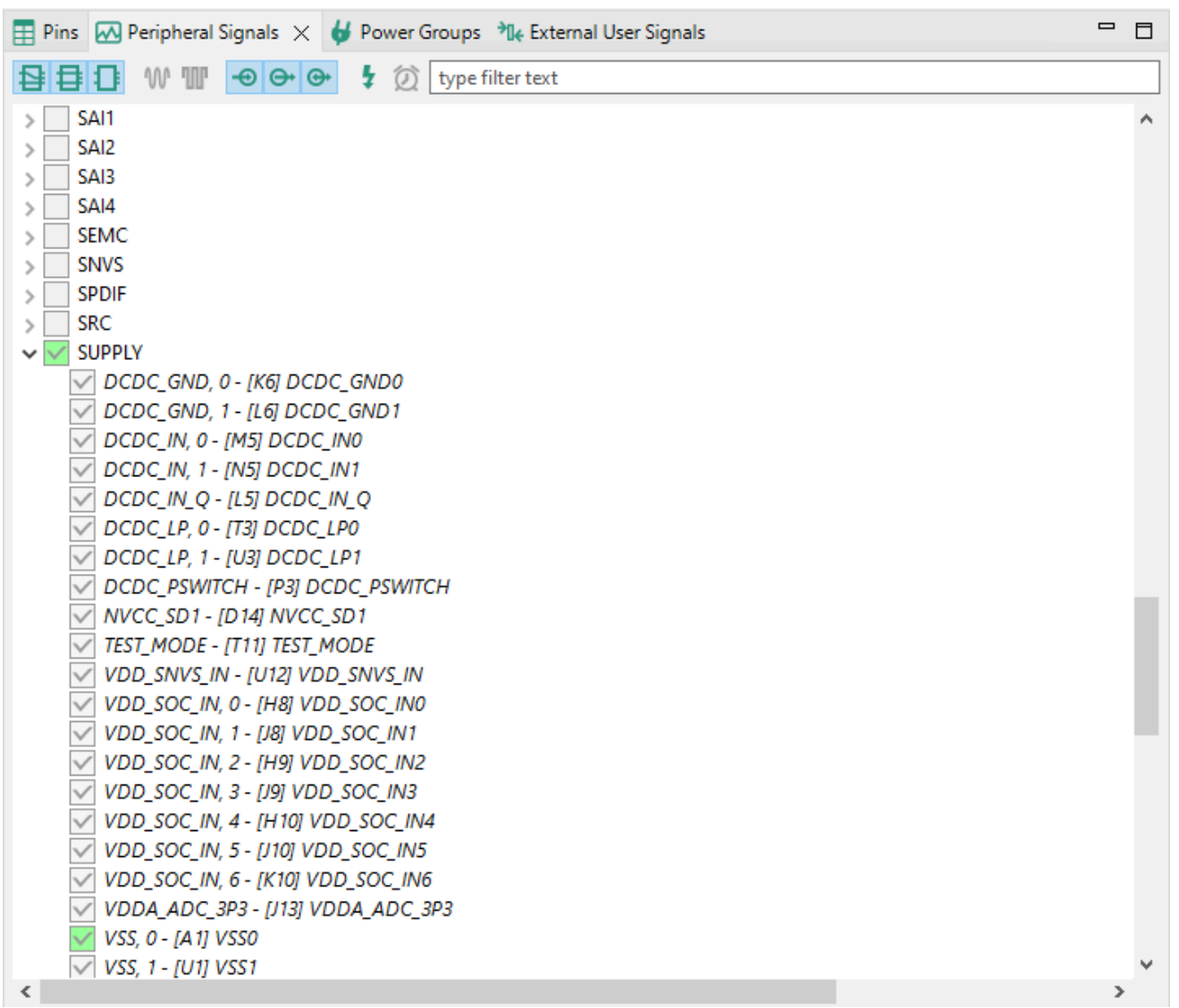


Figure 43. Peripheral Signals view

Use the checkbox to route/unroute the pins.

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

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

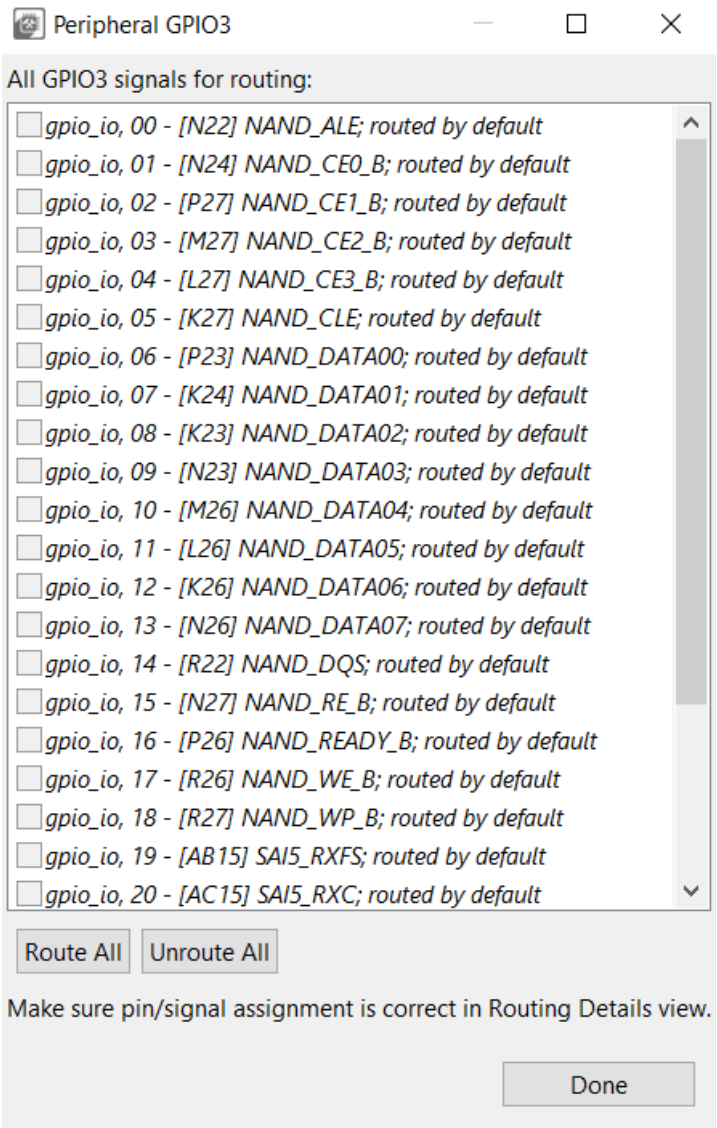


Figure 44. Select signals dialog

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.

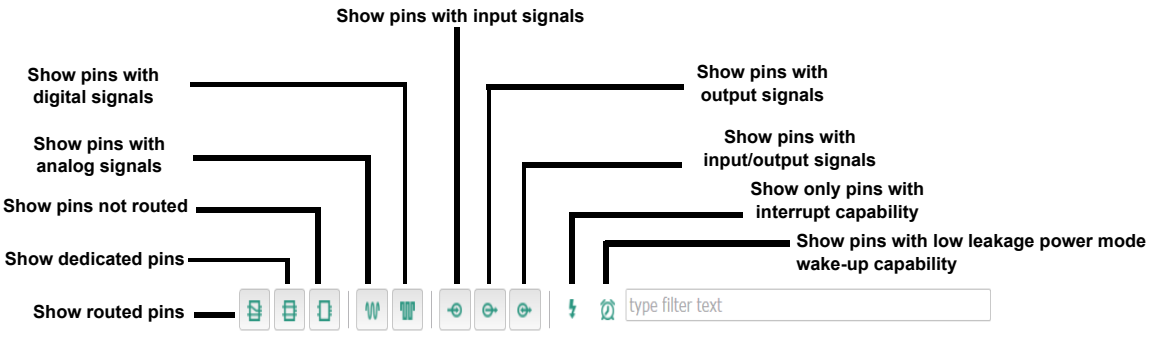


Figure 45. Filtering Controls

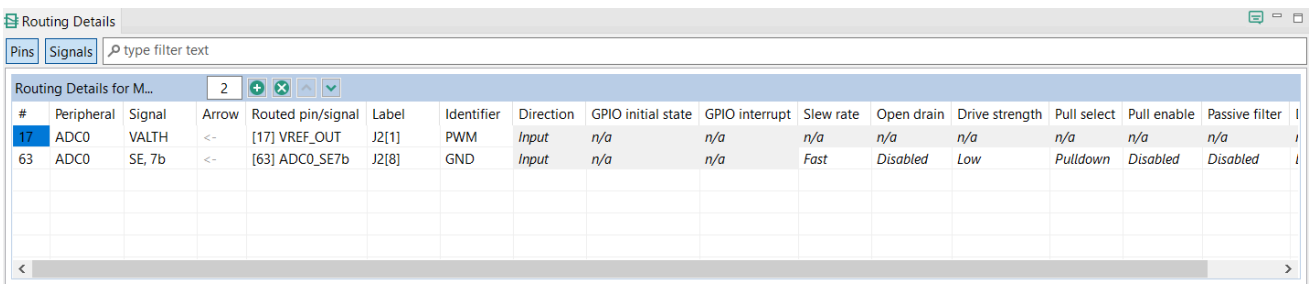
Type any text to search across the table/tree. It will search for the pins/peripheral signals containing the specified text. You can also use wildcards "*" and "?" to help you filter results you want. Use "space" to search for multiple strings at the same time.

3.3.4 Routing Details view

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

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

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



#	Peripheral	Signal	Arrow	Routed pin/signal	Label	Identifier	Direction	GPIO initial state	GPIO interrupt	Slew rate	Open drain	Drive strength	Pull select	Pull enable	Passive filter
17	ADC0	VALTH	<-	[17] VREF_OUT	J2[1]	PWM	*Input*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*
63	ADC0	SE, 7b	<-	[63] ADC0_SE7b	J2[8]	GND	*Input*	*n/a*	*n/a*	*Fast*	*Disabled*	*Low*	*Pulldown*	*Disabled*	*Disabled*

Figure 46. Routing Details

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

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

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

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

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

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

Note: *When you enter the search text, it also searches the text in the full pin names 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.

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

Tip:

- *Click the **Routing Details Legend** button in top right corner of the view to display a dialog explaining the fields.*

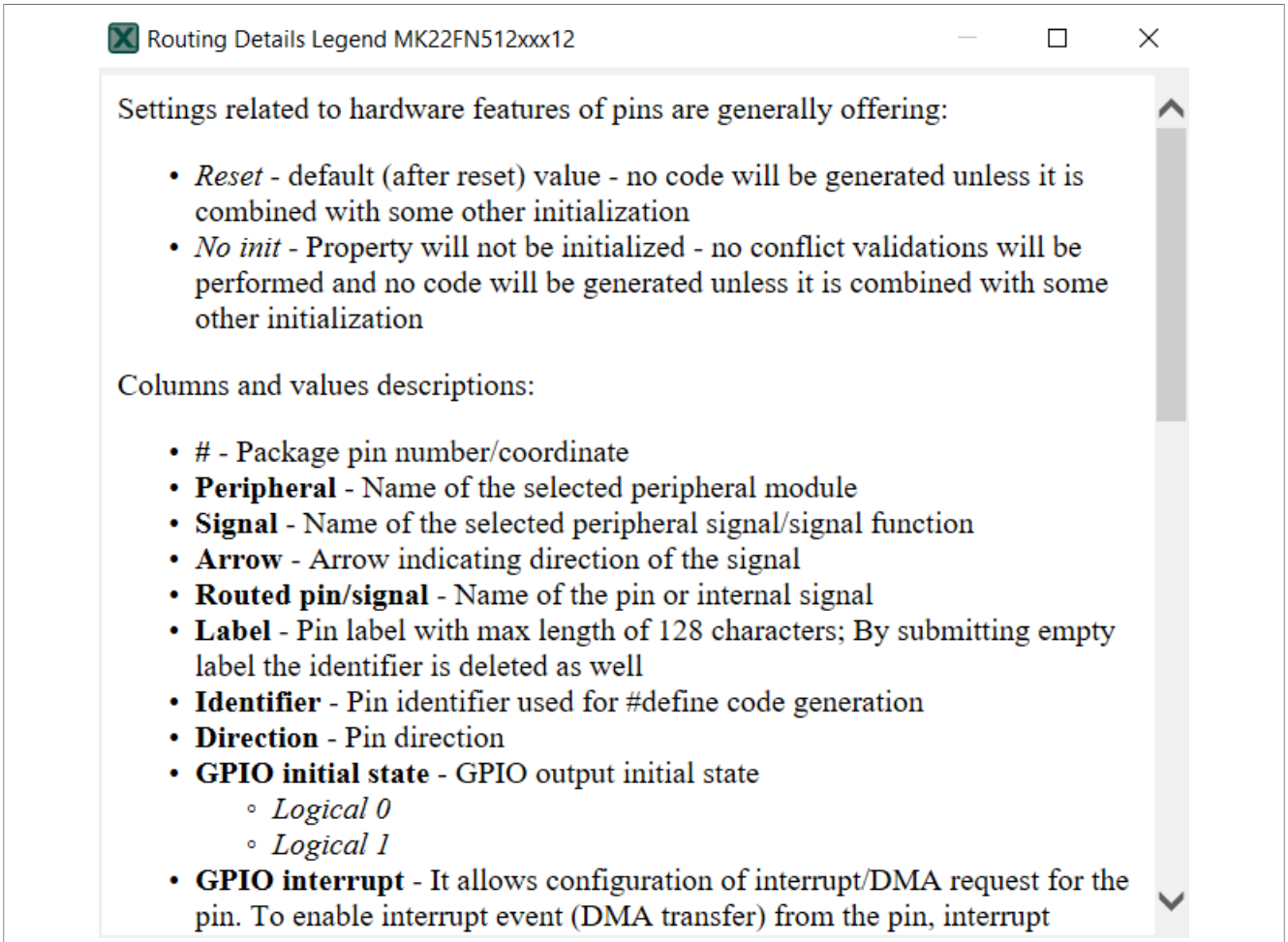


Figure 47. Routing Details Legend

- *The value shown using italic indicates the after-reset value. The real value may be different from the after reset value, if configured in other functions.*
  Use the drop-down menu to select the required value.
- *If you select the same value as the after-reset value, the tool will always generate code to set this feature.*
  Use the drop-down "Reset" value to reset the value to its after-reset state.
- *If an item does not support reset to after reset value, the **Reset** menu is not available.*
- *The first row shows pin number or coordinate on BGA package.*

3.3.4.1 Labels and identifiers

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

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

To set\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.

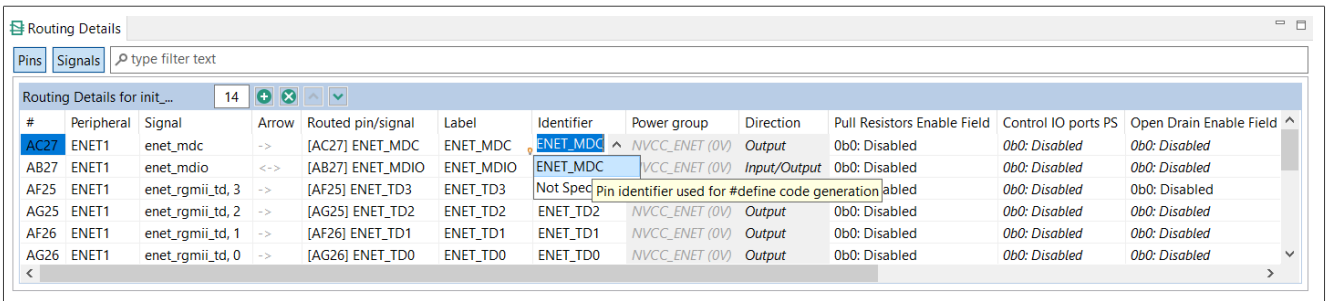


Figure 48. Labels and Identifiers

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

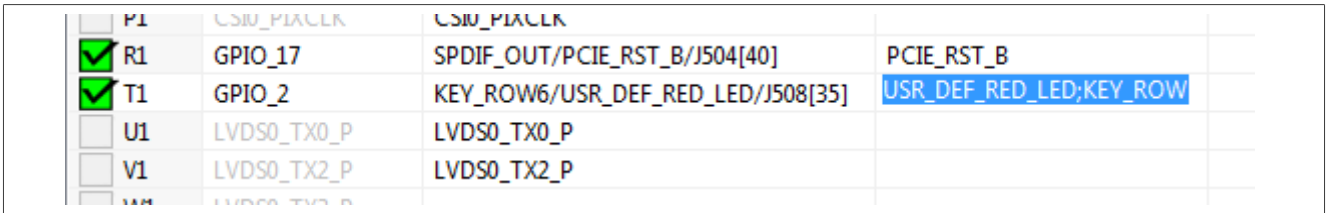


Figure 49. Pin identifier

In this case, it is possible to select from values if the pin is routed. See the **Identifier** column in the **Routing Details** view.

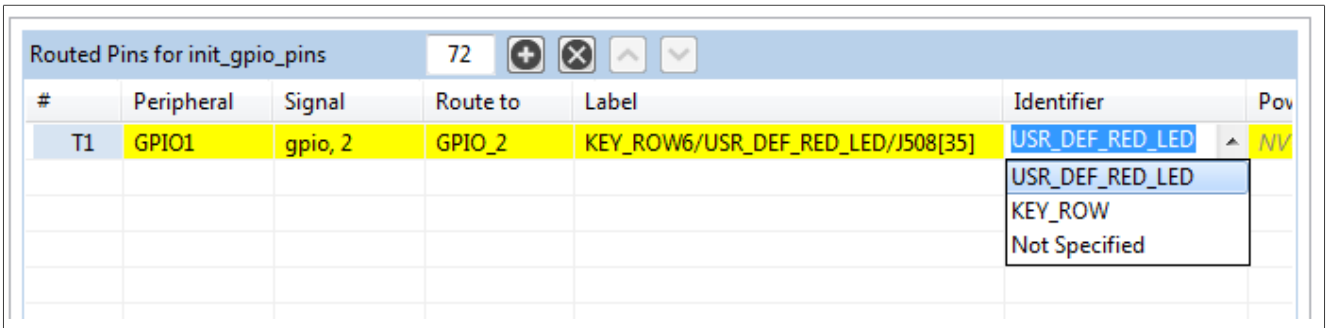


Figure 50. Identifier in Routing Details table

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

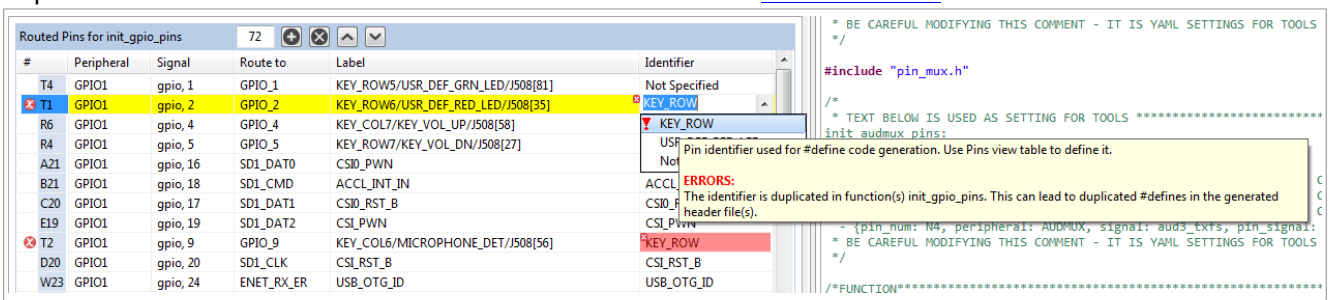


Figure 51. Identifier errors

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

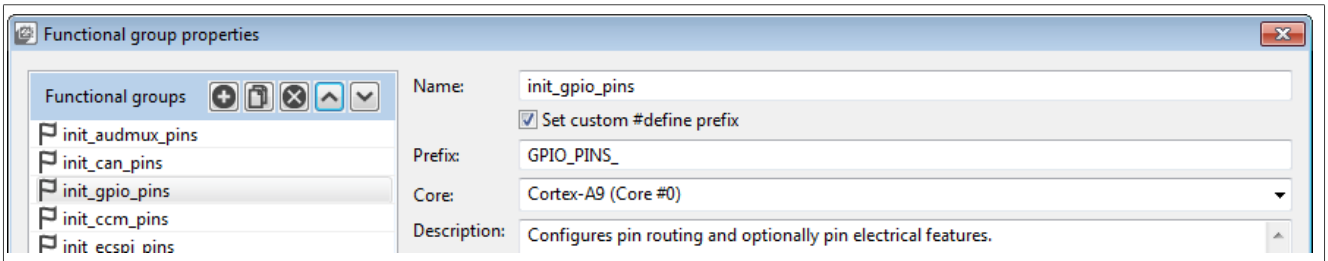


Figure 52. Pins macros prefix

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

3.3.5 Expansion Header

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

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

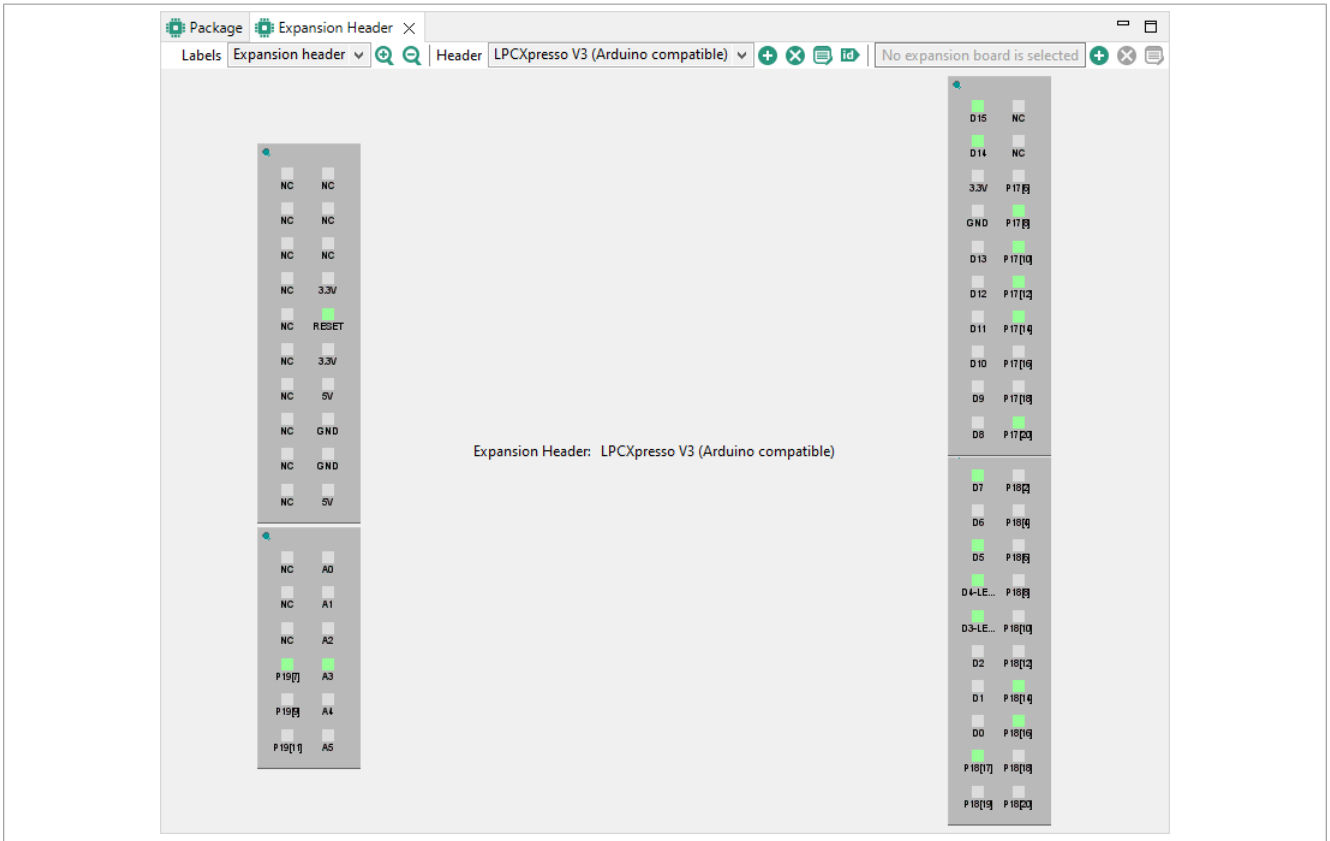


Figure 53. Expansion header

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.

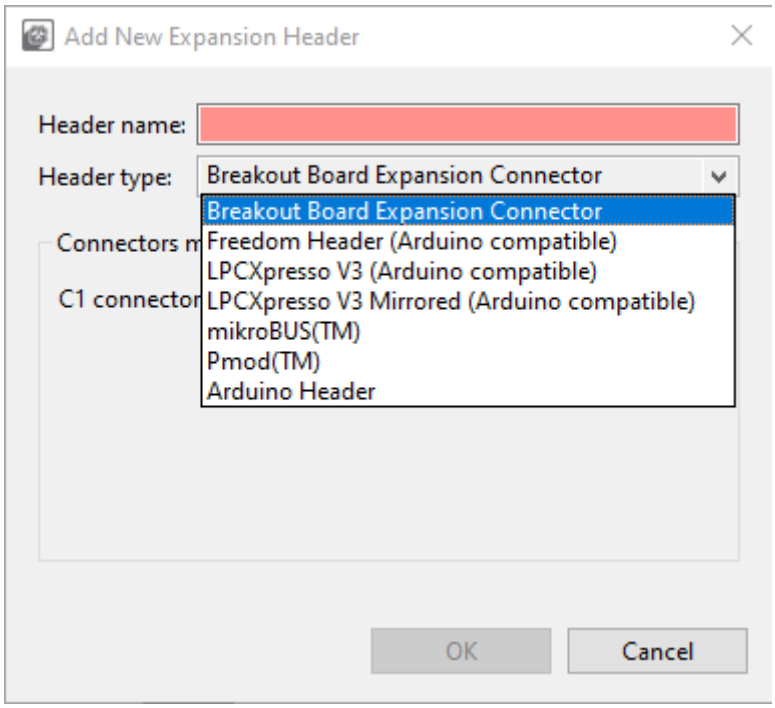


Figure 54. Adding new expansion header

4. Name the header and map the connectors.

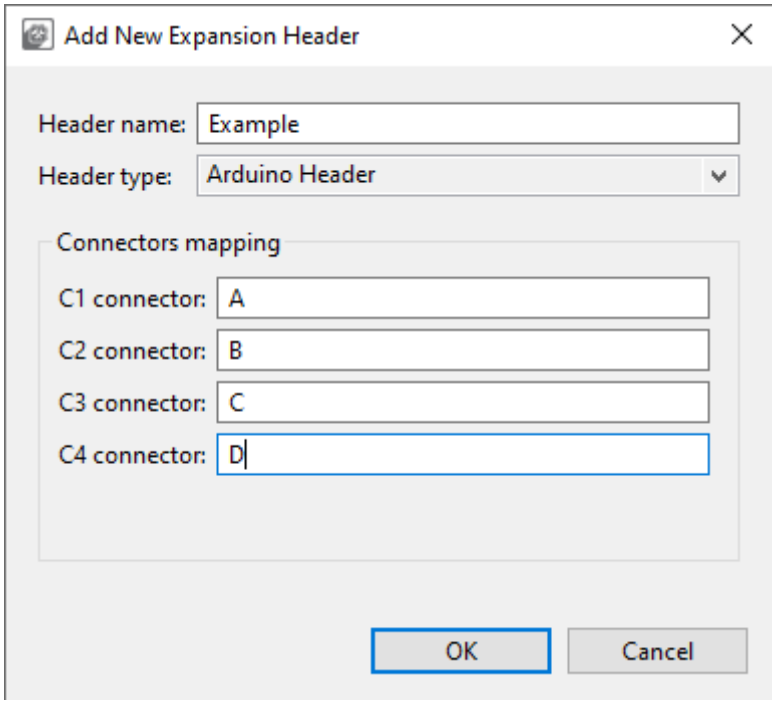


Figure 55. Adding new expansion header

5. Select **OK**.

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

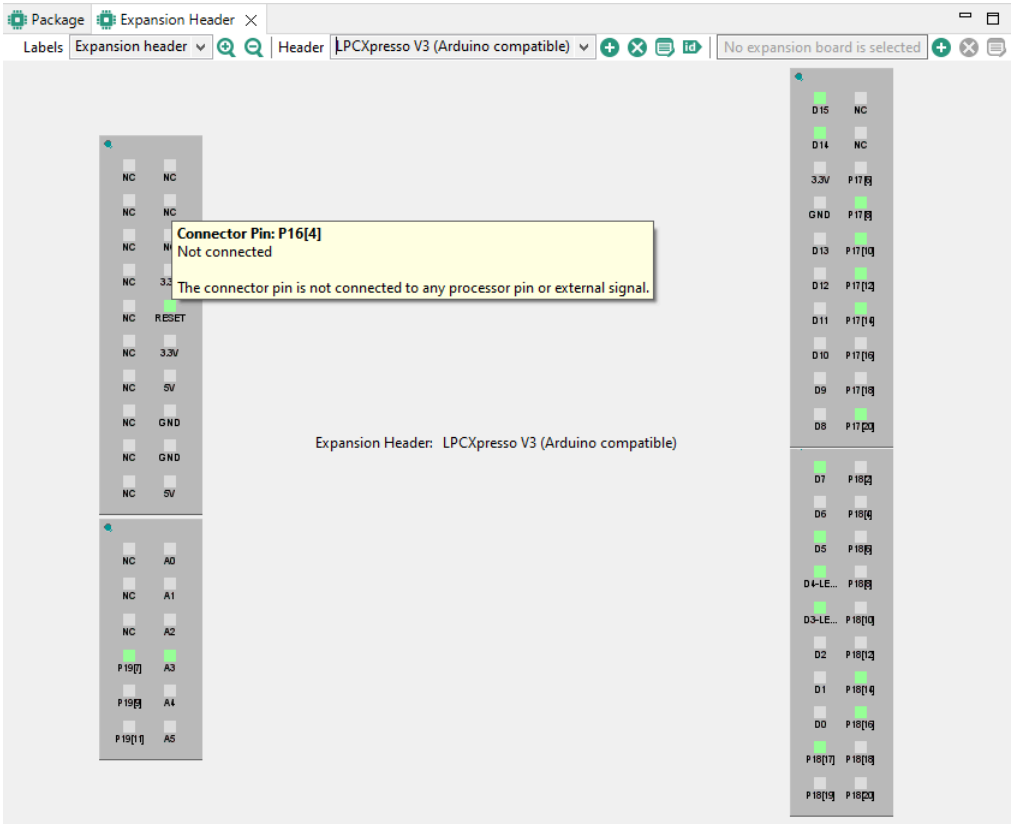


Figure 56. Expansion header

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

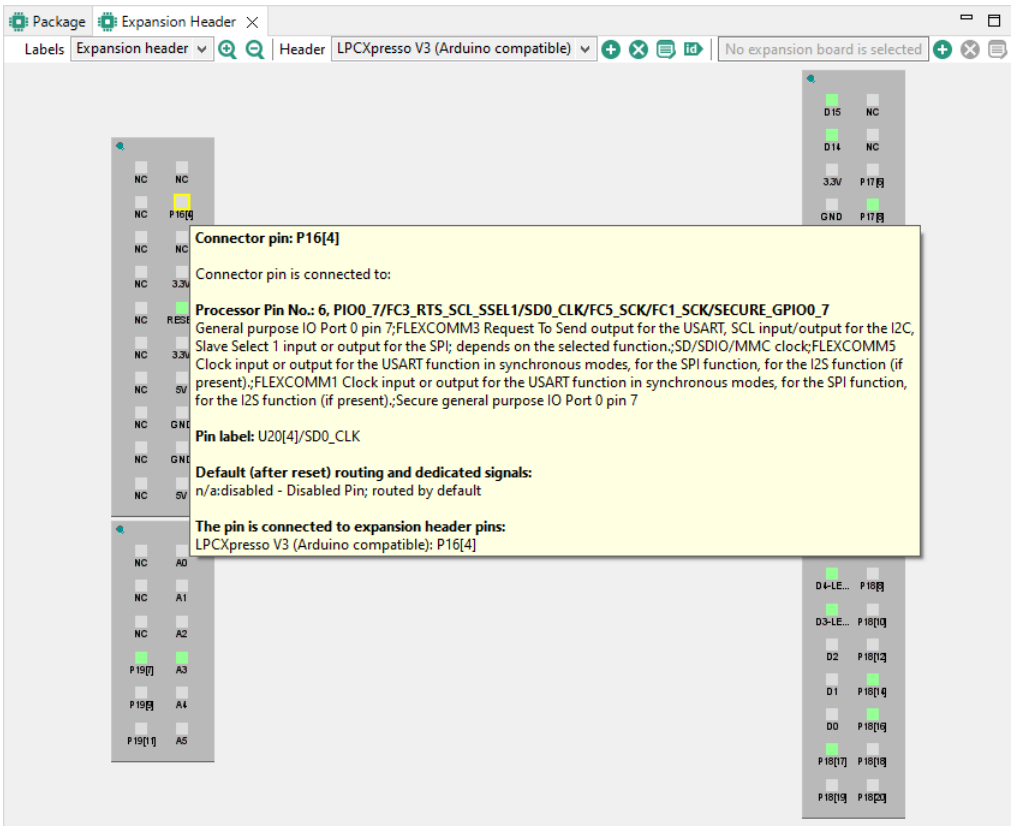


Figure 57. Connected pin

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.

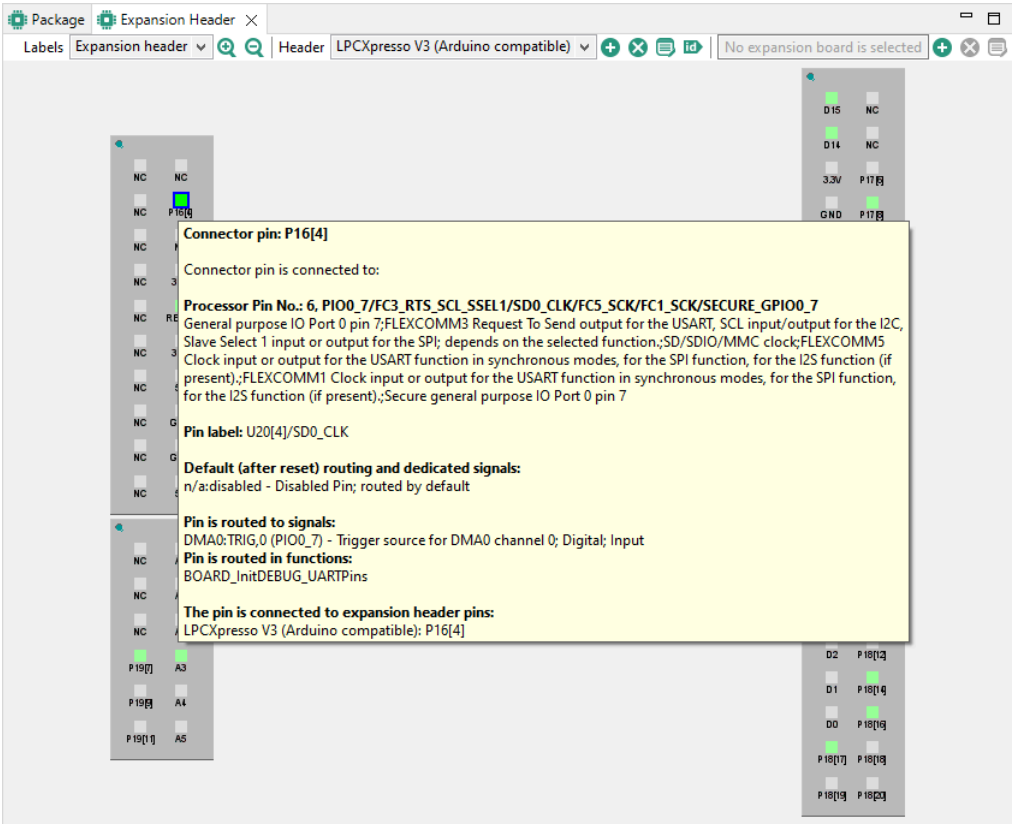


Figure 58. Routed pin

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 is updated automatically. Pin connections between the header and the processor and their labels can be locked to prevent any modifications.

Remove a configuration by selecting the **Remove** button.

Use the **Label** drop-down list to switch between display information for header, board, and routing.

The **ID** button allows setting expansion header pin labels as processor pin identifiers. Upon user selection, the new identifiers can be also explicitly selected.

3.3.5.1 Expansion Board

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

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

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

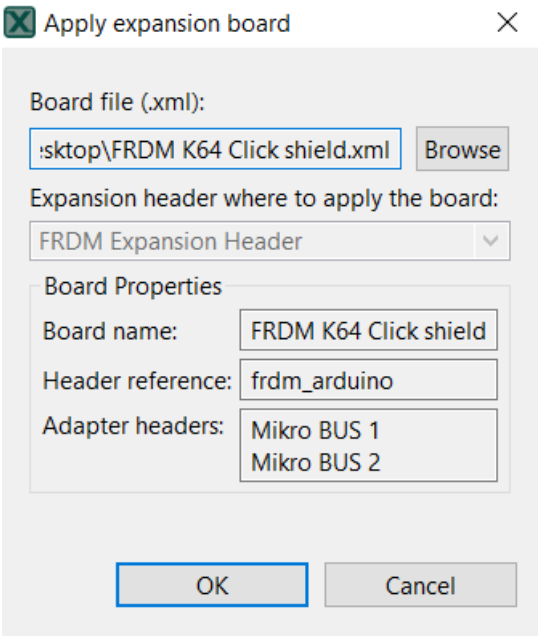


Figure 59. Apply expansion board

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

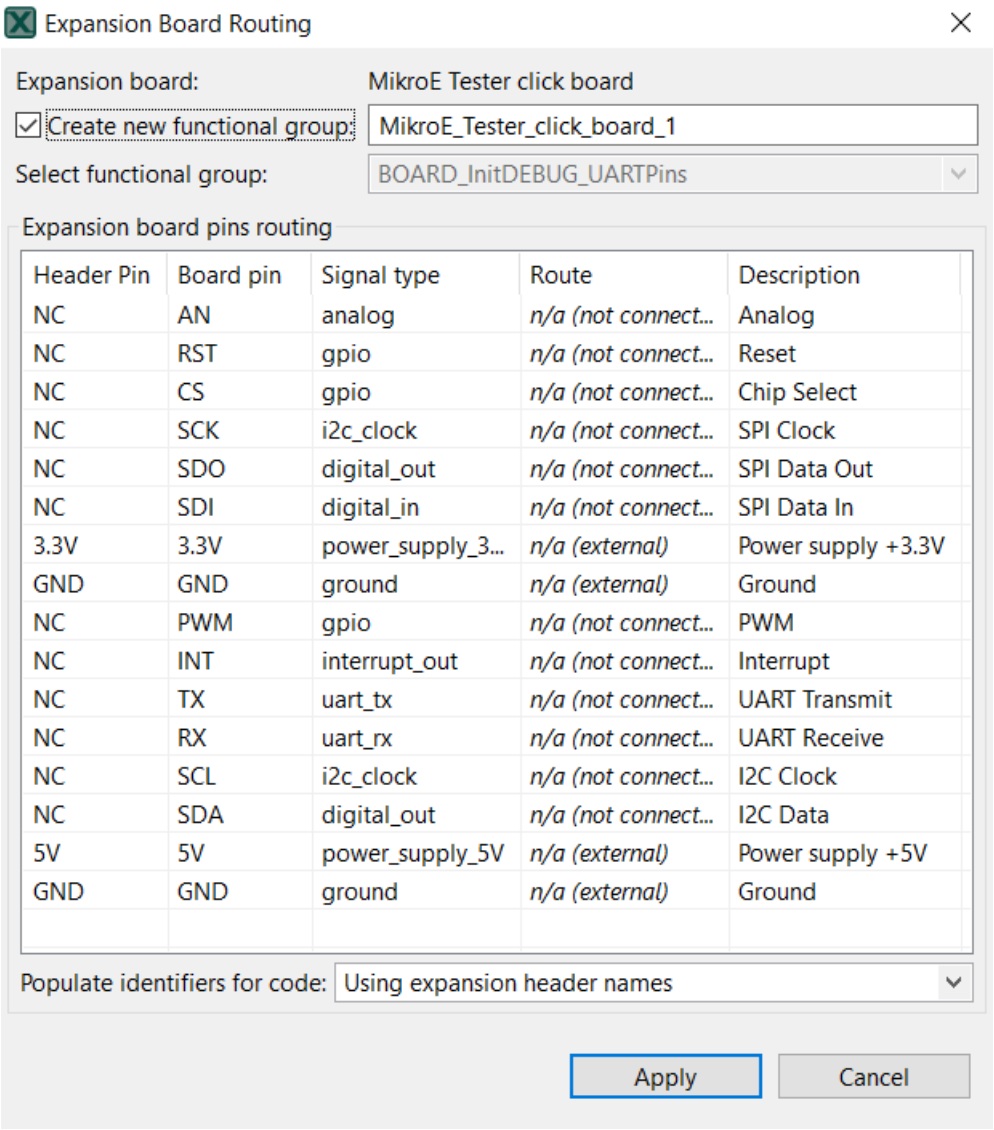


Figure 60. Expansion board routing

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

3.3.6 Power groups

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

Note: *This feature is not supported for all devices.*

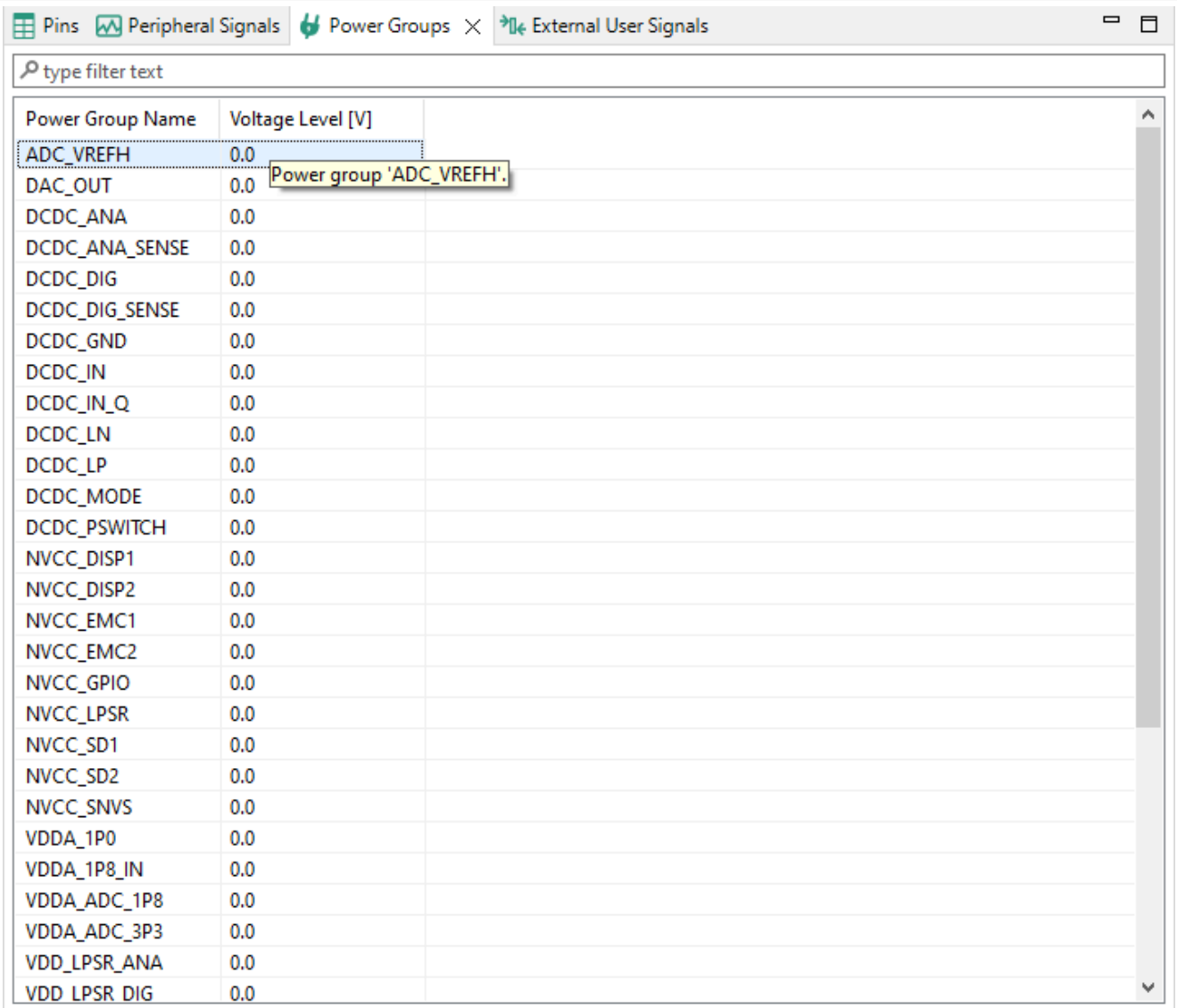


Figure 61. Selecting power group

3.3.7 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.

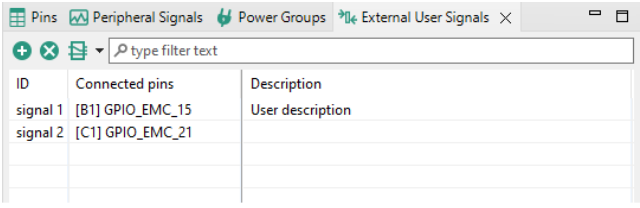


Figure 62. External User Signals view

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.

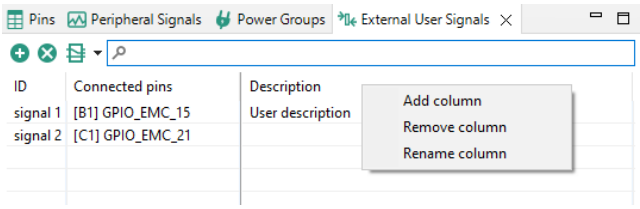


Figure 63. External User Signals header context menu

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

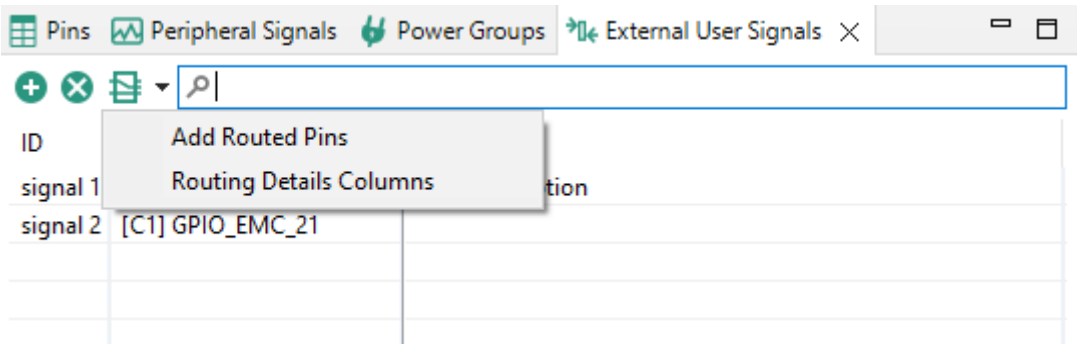


Figure 64. External User Signals view

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

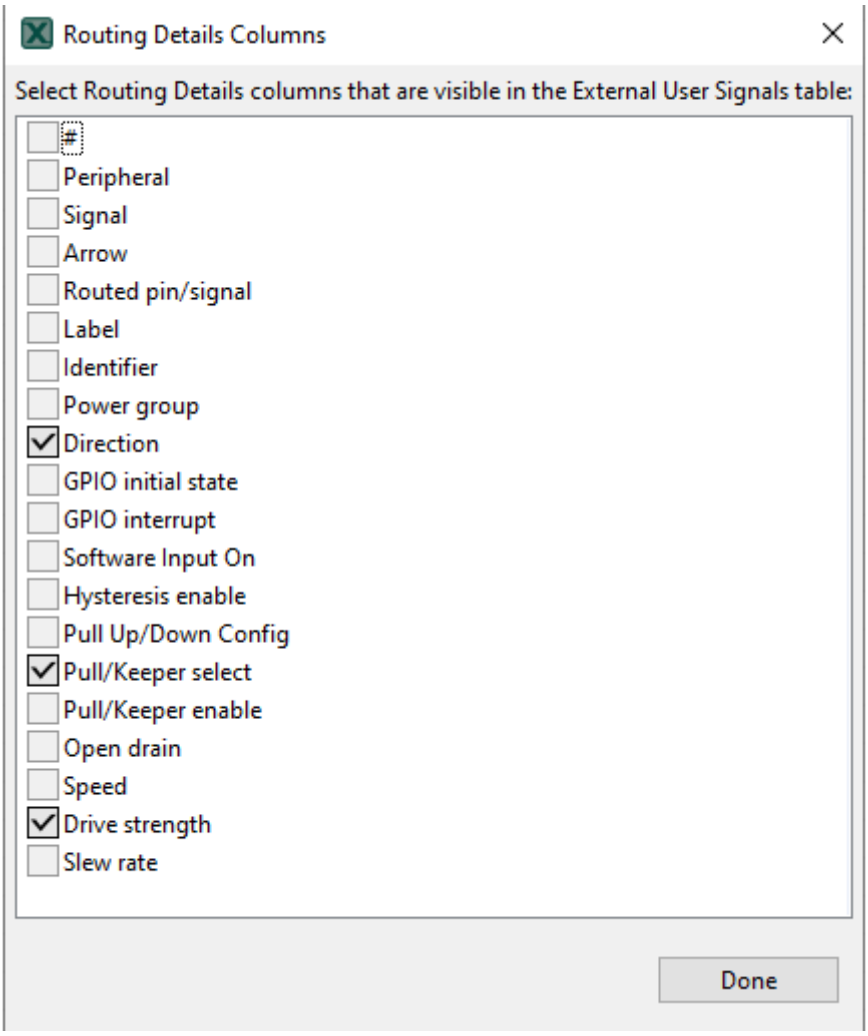
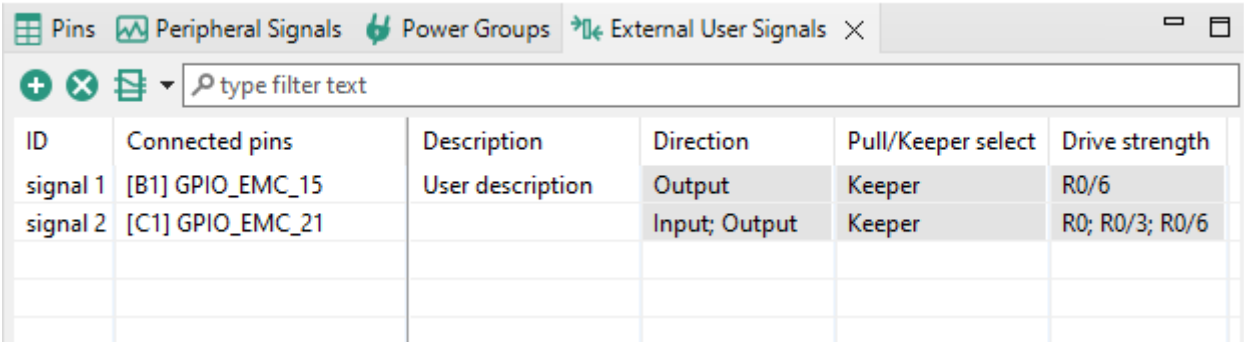


Figure 65. Routing Details columns dialog



ID	Connected pins	Description	Direction	Pull/Keeper select	Drive strength
signal 1	[B1] GPIO_EMC_15	User description	Output	Keeper	R0/6
signal 2	[C1] GPIO_EMC_21		Input; Output	Keeper	R0; R0/3; R0/6

Figure 66. Routing Details table

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

3.3.8 Functions

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

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

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

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

3.3.9 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.

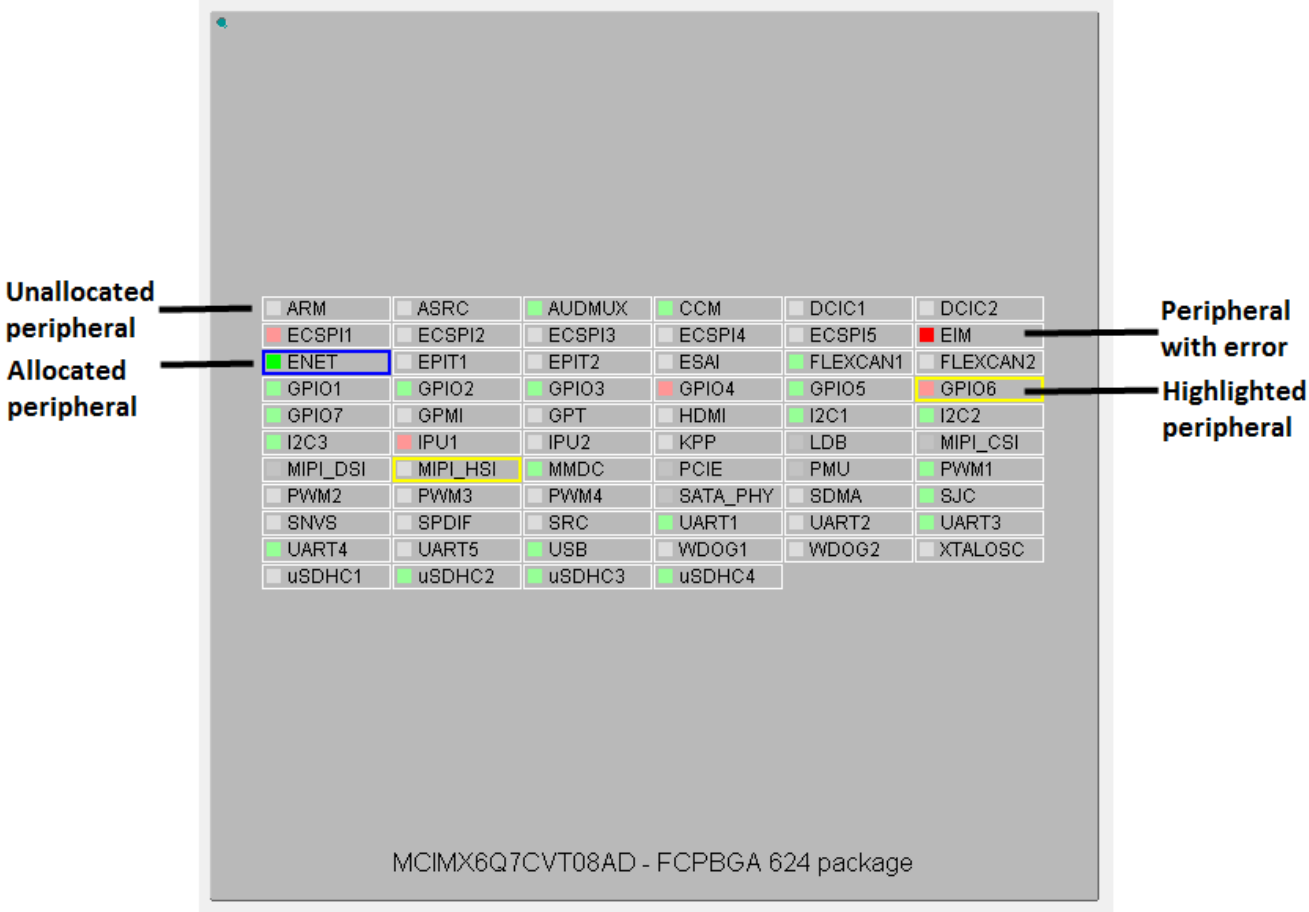


Figure 67. Highlighting and color coding

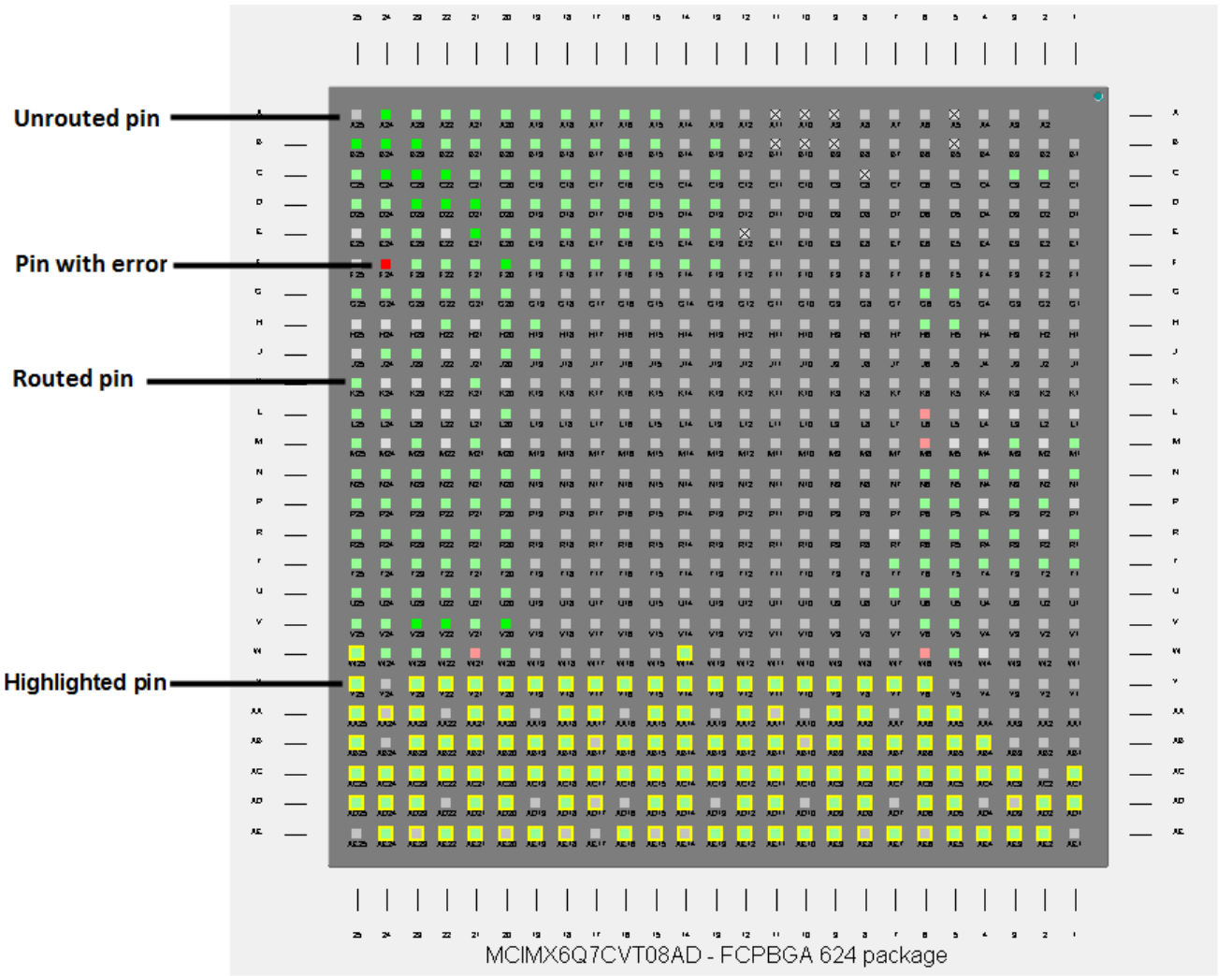


Figure 68. BGA package on pins side

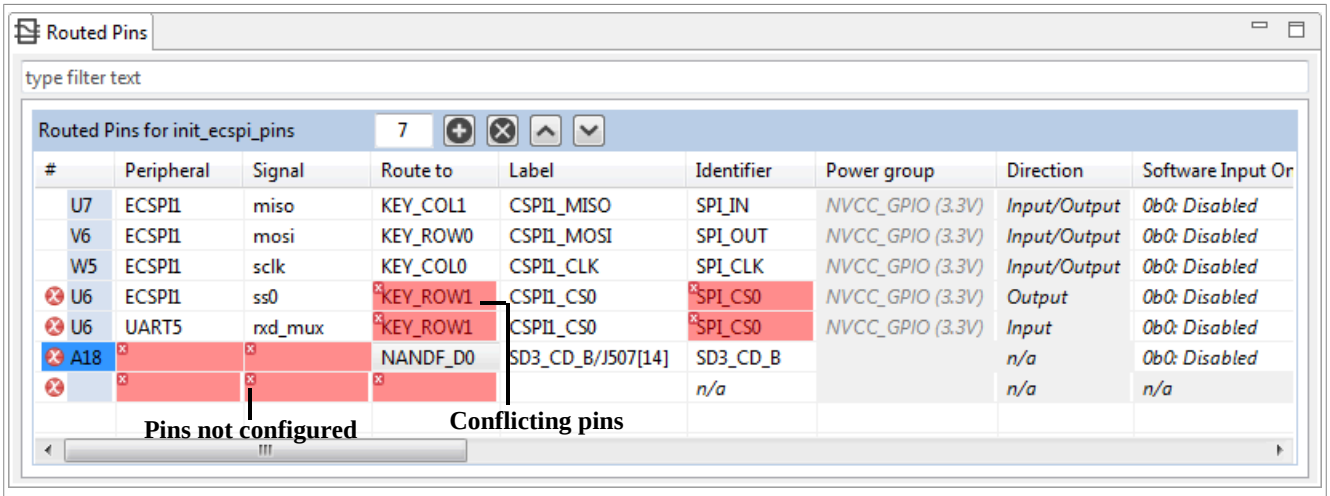


Figure 69. Pins conflicts

#	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 behaves 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. Routing conflicts are checked across all **INIT** functions (default initialization functions). It is possible to configure different routing of one pin in different functions (not INIT functions) to allow dynamic pins routing reconfiguration.

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.

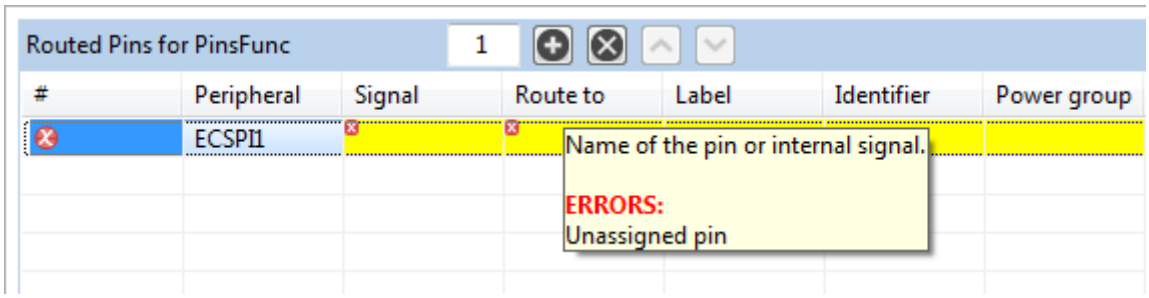


Figure 72. Incomplete routing

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 a 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.

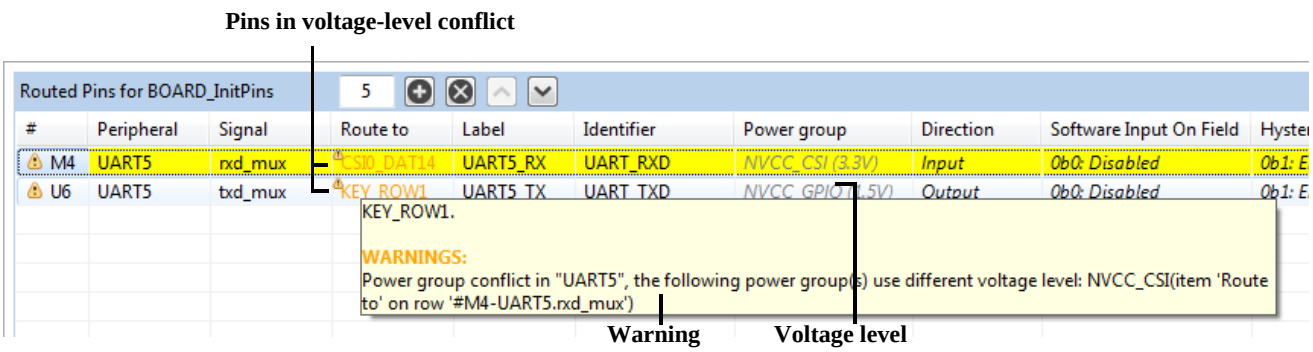


Figure 73. Power group view

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 the **Code Preview** view of the **Pins** tool.

Code Preview automatically highlights differences between the current and immediately preceding iteration of the code. You can choose between two modes of highlighting by clicking the **Set viewing style for source differences**. You can also disable highlighting altogether from the same dropdown menu. Such features as Copy, Search, Zoom-in, Zoom-out, and Export source are available in the **Code Preview** view. The search can also be invoked by CTRL+F or from the context menu.

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

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

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

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, in order to generate appropriate function call.*

3.6 Using pins definitions in code

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

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

```
#define BOARD_LED_RED_GPIO GPIOB /*!<@brief GPIO device name: GPIOB */
#define BOARD_LED_RED_PORT PORTB /*!<@brief PORT device name: PORTB *'/
#define BOARD_LED_RED_PIN 22U /*!<@brief PORTB pin index: 22 */
```

The name of the define is composed from function group prefix and pin identifier. For more details, see Section 2.4.3 and Section 3.3.4.1 sections.

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

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

3.7 Full initialization of pins

In some cases, the default values are not reliable, as there may be code running before the application that modifies the pin configuration (for example, a bootloader). The option **Full initialization of pins** ensures that the initialization is fully done even for items that use after-reset state. This option is specific for each **Functional group** allowing to force full initialization of routing. **Full initialization of pins** is not enabled by default. When enabled, the electrical properties of existing routing are changed. The "Reset" values are changed to explicit values corresponding with them. When the option is disabled, the pins tool changes the values that are matching after-reset state to the "Reset" values.

3.8 Create Default Routing

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

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

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

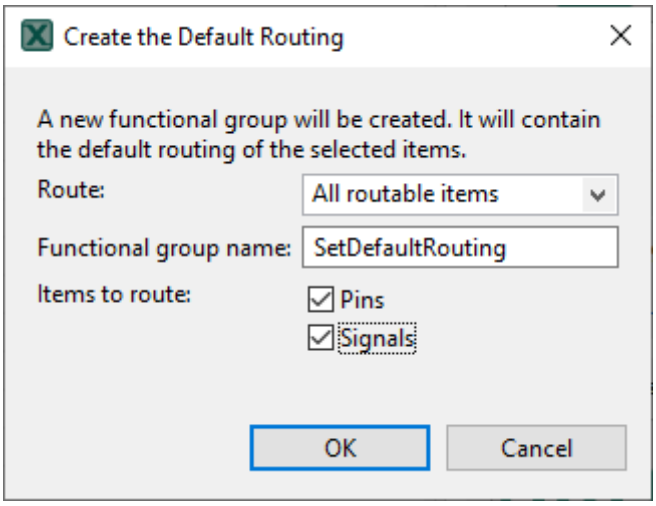


Figure 74. Create the Default Routing

4 DDR Tool

This section introduces the DDR configuration and validation tool, which is an embedded componen 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.*

4.1 Create a new DDR tool project

To use the DDR tool, you first must 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.

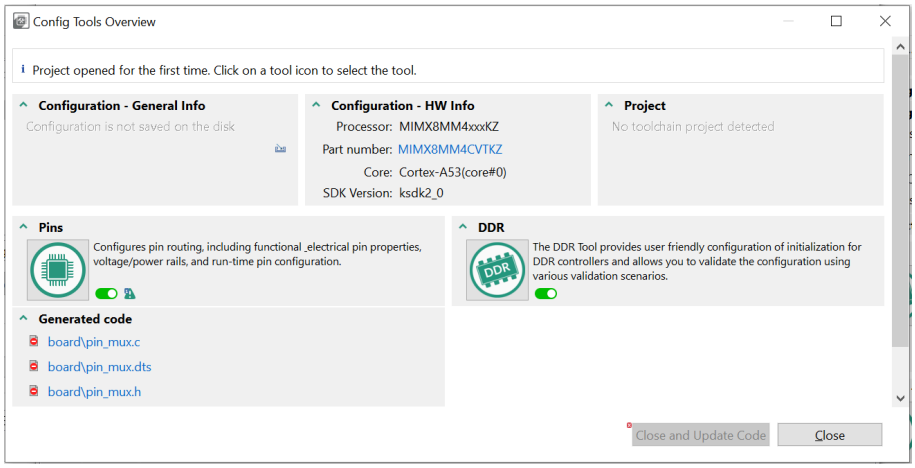


Figure 75. Config Tools overview

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

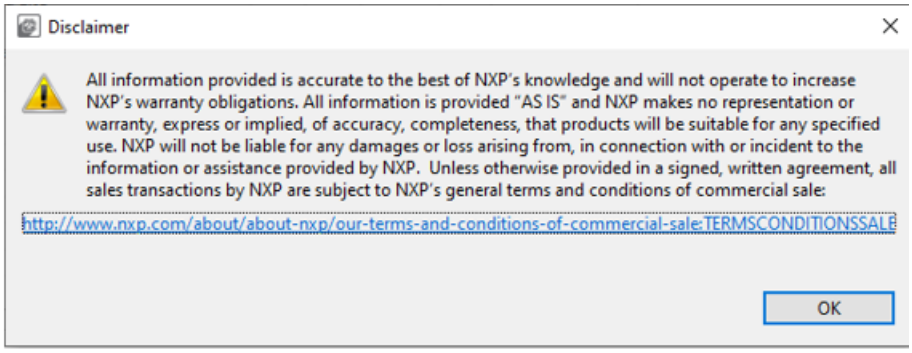


Figure 76. Disclaimer

4.2 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.

4.2.1 Import initialization script

Import initialization script allows loading the initialization script provided by the **Register Programming Aid (RPA)** tool and bypassing the UI configuration. To obtain the latest RPAs, refer to the following link on NPX community.

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

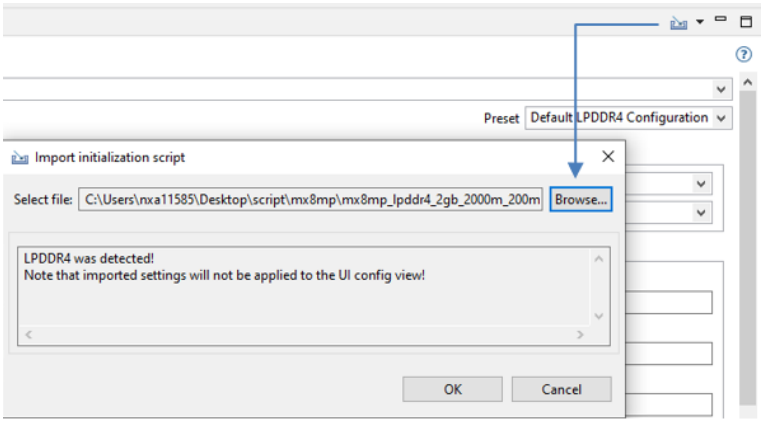


Figure 77. Import initialization script

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

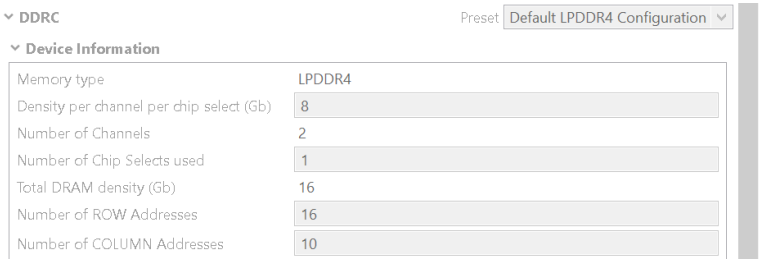


Figure 78. Disabled UI

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

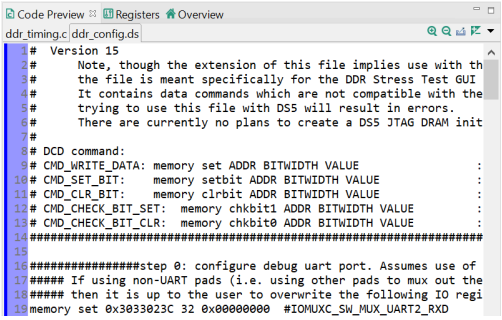


Figure 79. RPA script content

4.2.2 Import from target

Import from target allows loading the DDR initialization of an already configured working target and bypassing the UI configuration. Use this option only for devices with JTAG connection available.

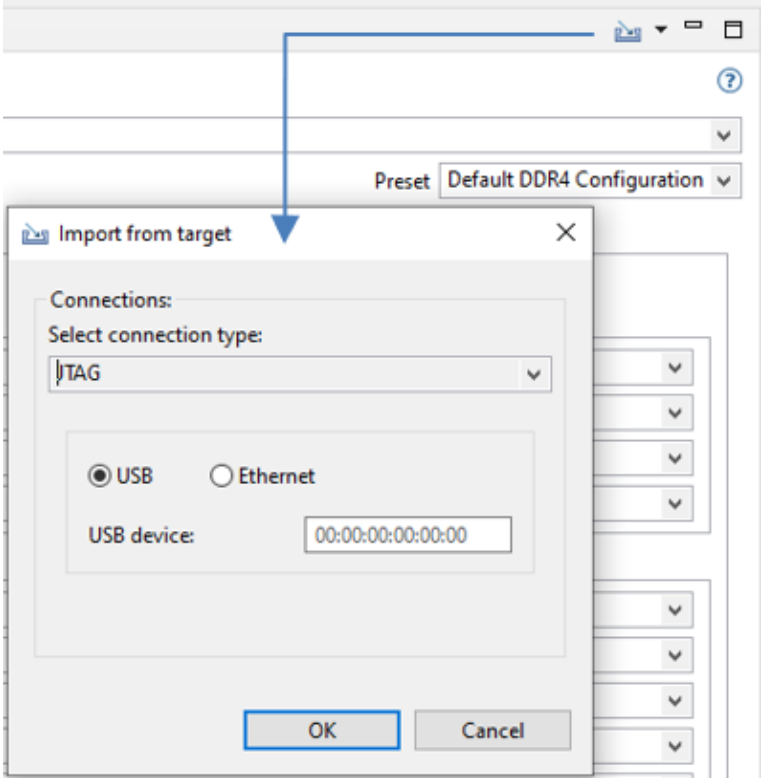


Figure 80. Import from target

4.2.3 Enable manual configuration

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

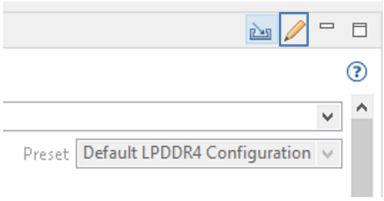


Figure 81. Enable manual config

4.2.4 UI configuration

The UI configuration allows you to change manually Device Information, PHY options, or Design-specific configuration. There are two modes 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

Device Information

Memory type	LPDDR4
Density per channel per chip select (Gb)	8
Number of Channels	2
Number of Chip Selects used	1
Total DRAM density (Gb)	16
Number of ROW Addresses	16
Number of COLUMN Addresses	10
Number of BANK addresses	3
Number of BANKS	8
Bus Width	32
Number of frequency setpoints	1
Clock Cycle Freq (MHz)	1500 MHz
Clock Cycle Time	666.667 ps

Figure 82. Device Information UI

2. Board data bus configuration for LPDDR4

Board data bus config

Channels

#	<							B								>
DRAM data bus	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRAM data bus (User Input ->)	7	6	5	4	3	2	1	0	14	15	10	13	12	11	9	8
Byte lane	<			0				>	<			1				>
Data bus bits within byte lane	7	6	5	4	3	2	1	0	6	7	2	5	4	3	1	0

Figure 83. Board data bus config UI

3. UART port selection

UART Port: UART2

Figure 84. UART selection UI

4. DBI selection (for LPDDR4)

DBI: Disable

Figure 85. DBI selection

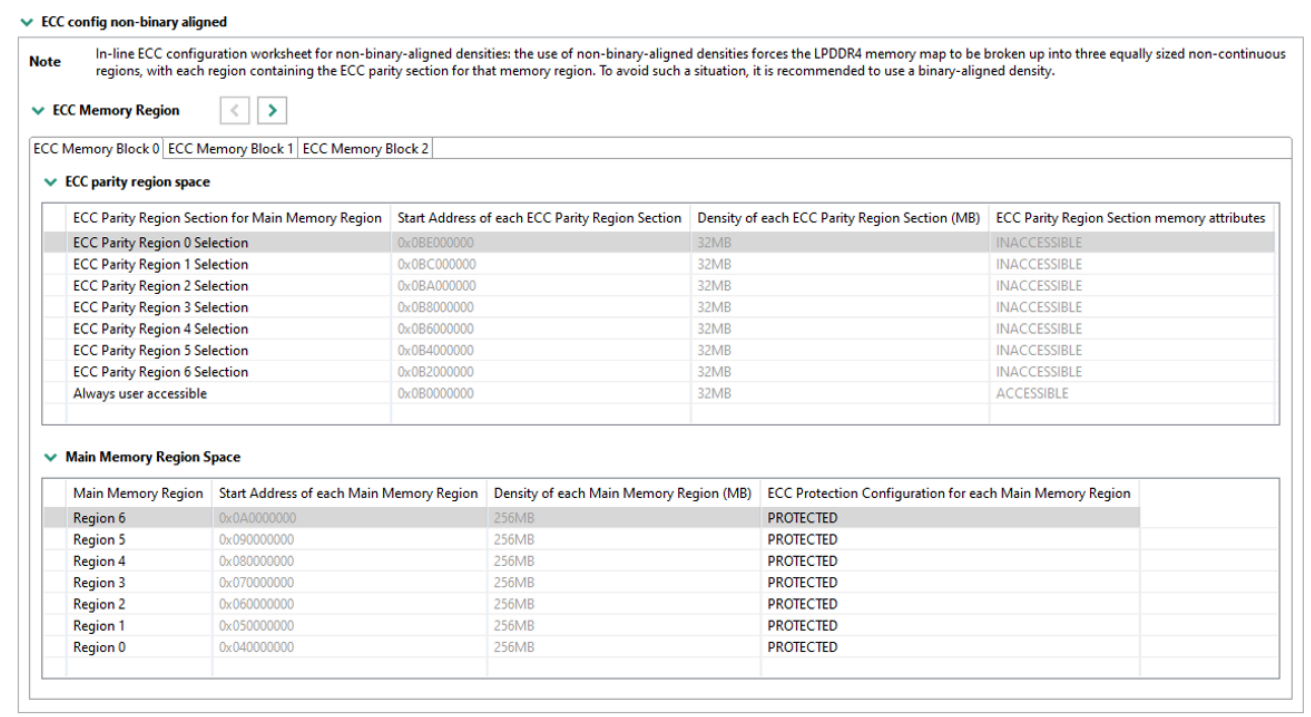
5. Inline ECC configuration (for devices with Inline ECC support) allows selection of the following parameters:
   a. Enable/Disable Inline ECC

Figure 86. Inline ECC

b. ECC region granularity

Figure 87. ECC region granularity

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



ECC Parity Region Section for Main Memory Region	Start Address of each ECC Parity Region Section	Density of each ECC Parity Region Section (MB)	ECC Parity Region Section memory attributes
ECC Parity Region 0 Selection	0x0BE000000	32MB	INACCESSIBLE
ECC Parity Region 1 Selection	0x0BC000000	32MB	INACCESSIBLE
ECC Parity Region 2 Selection	0x0BA000000	32MB	INACCESSIBLE
ECC Parity Region 3 Selection	0x0B8000000	32MB	INACCESSIBLE
ECC Parity Region 4 Selection	0x0B6000000	32MB	INACCESSIBLE
ECC Parity Region 5 Selection	0x0B4000000	32MB	INACCESSIBLE
ECC Parity Region 6 Selection	0x0B2000000	32MB	INACCESSIBLE
Always user accessible	0x0B0000000	32MB	ACCESSIBLE

Main Memory Region	Start Address of each Main Memory Region	Density of each Main Memory Region (MB)	ECC Protection Configuration for each Main Memory Region
Region 6	0x0A0000000	256MB	PROTECTED
Region 5	0x090000000	256MB	PROTECTED
Region 4	0x080000000	256MB	PROTECTED
Region 3	0x070000000	256MB	PROTECTED
Region 2	0x060000000	256MB	PROTECTED
Region 1	0x050000000	256MB	PROTECTED
Region 0	0x040000000	256MB	PROTECTED

Figure 88. 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** offers the possibility to select between multiple versions.

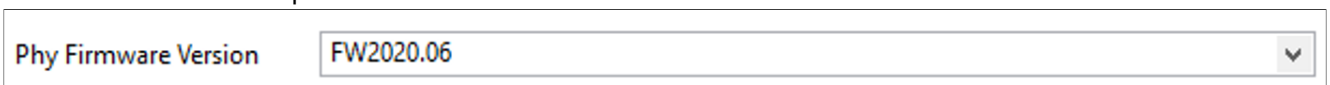


Figure 89. Firmware version selection UI

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 90. PHY log level selection

3. IOMUX configuration

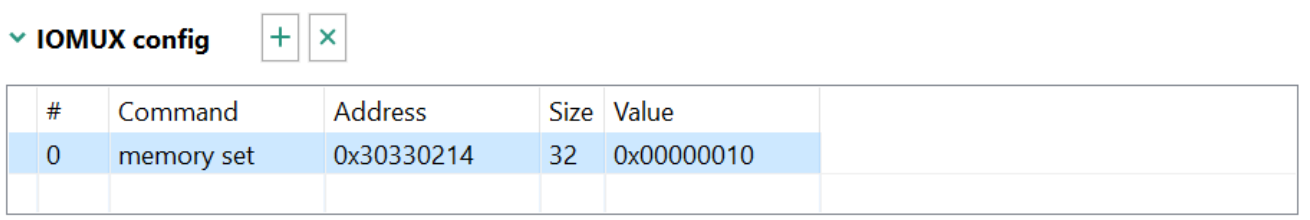


Figure 91. IOMUX configuration UI

4. PMIC configuration sequence

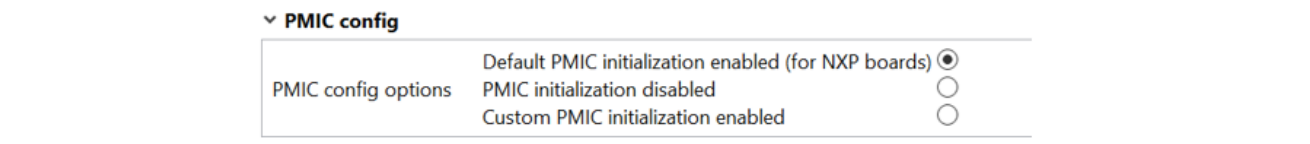


Figure 92. PMIC configuration UI

5. Custom configuration

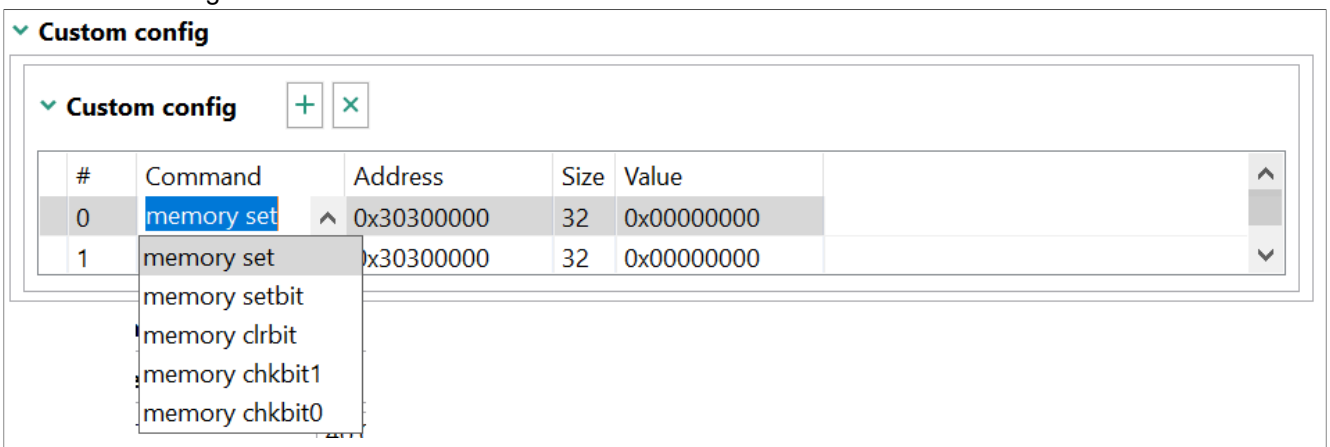


Figure 93. Custom configuration UI

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

6. DQ ODT and DS configuration

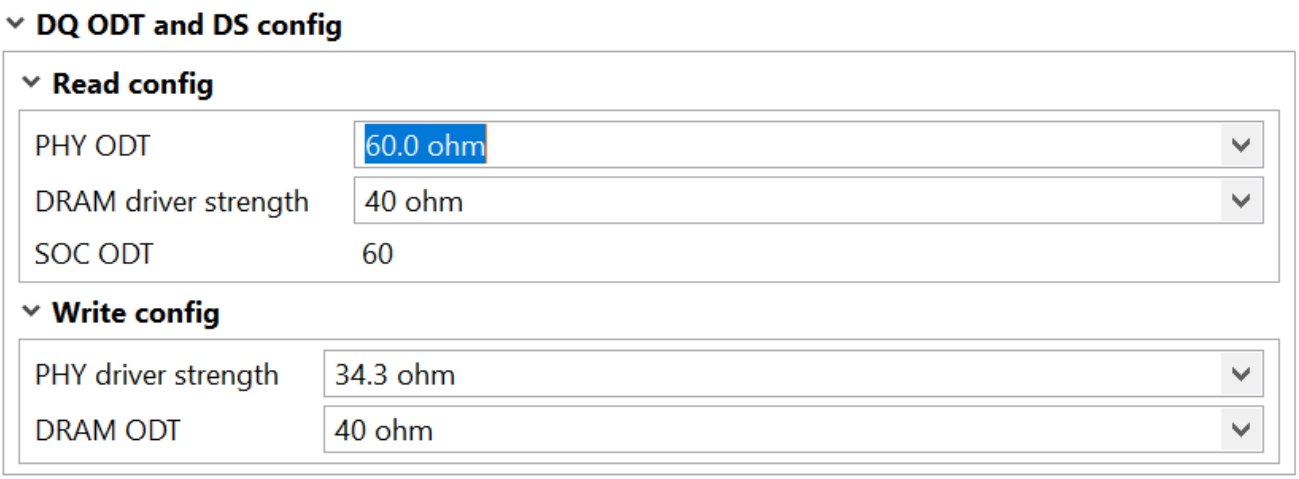


Figure 94. DQ ODT and DS configuration UI

7. CA ODT and DS configuration – Informational Only

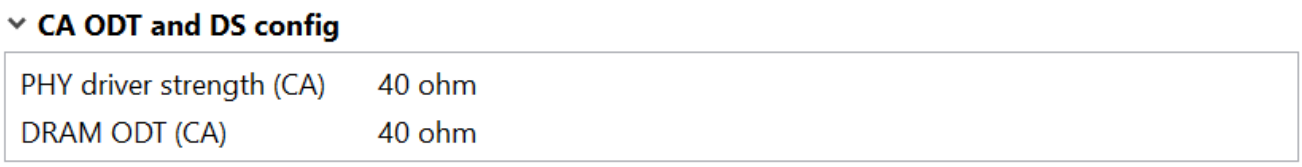


Figure 95. CA ODT and DS configuration UI

4.2.5 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.

You can trigger code generation by any change in the GUI, it is highlighted in the **Code Preview**.

Figure 96. Code Preview

You can save files from **Code Preview** on the disk by using **DDR tool Export Wizard**.

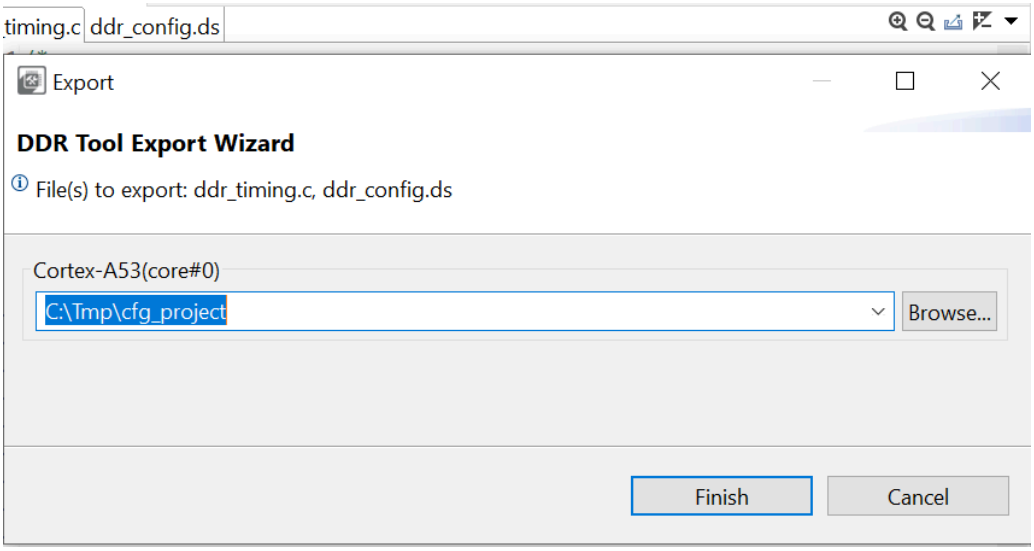


Figure 97. Export generated files

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

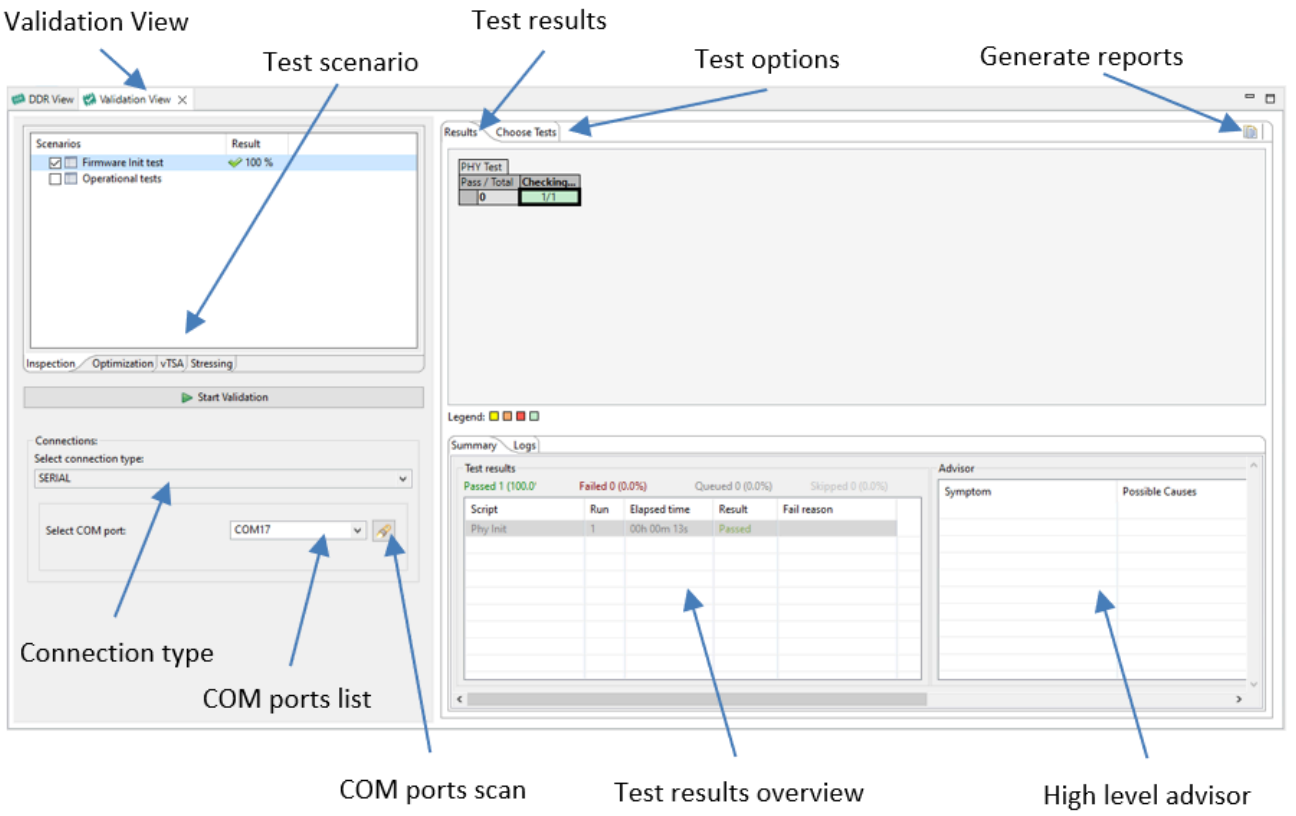


Figure 98. DDR validation

4.3.1 Connection

This section describes connection to boards of various types.

4.3.1.1 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.

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

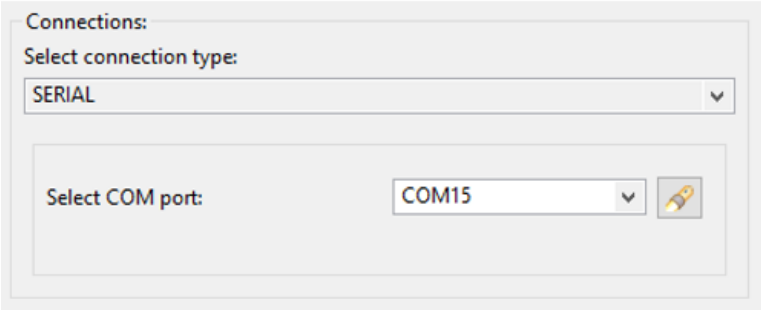


Figure 99. COM port selection

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

4.3.1.2 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.

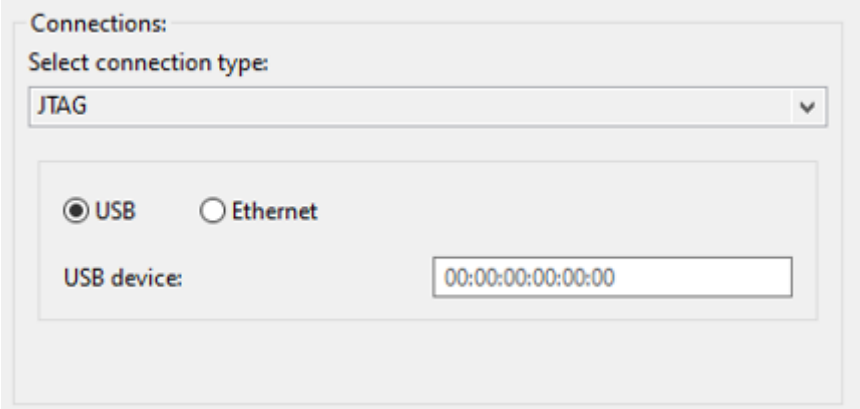


Figure 100. Connection type selection

4.3.2 Test scenarios

Once the DDR configuration and board connection are set up, you can execute different **Test scenarios**. You can customize each test by setting the parameters from **Test options**.

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

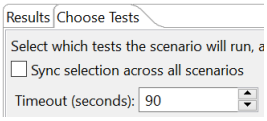


Figure 101. Timeout selection

When initial timeout expires, provide the input to continue or not the test execution.

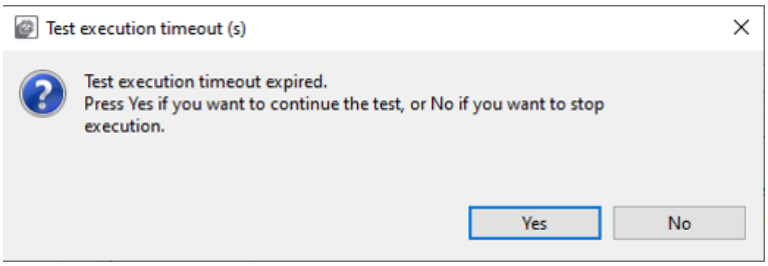


Figure 102. Test execution timeout(s)

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:

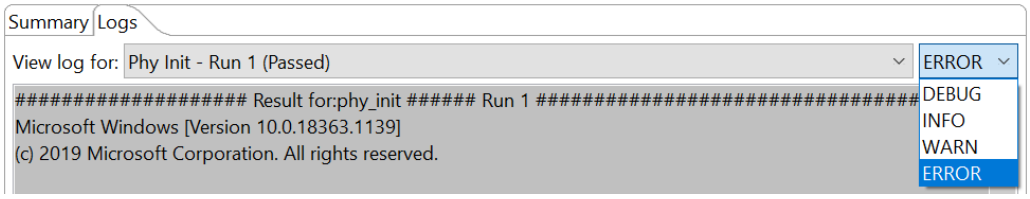
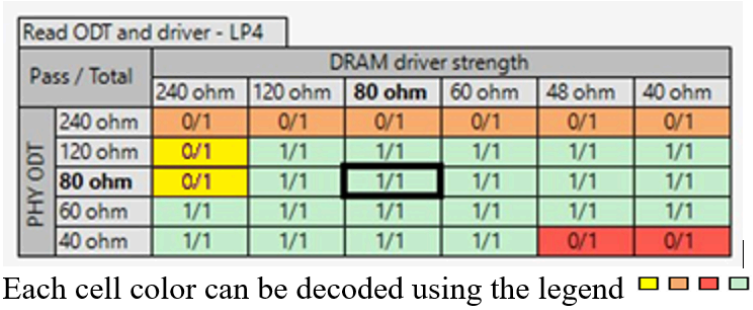


Figure 103. Log level selection

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



Read ODT and driver - LP4

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	0/1	0/1

Each cell color can be decoded using the legend

Figure 104. 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.

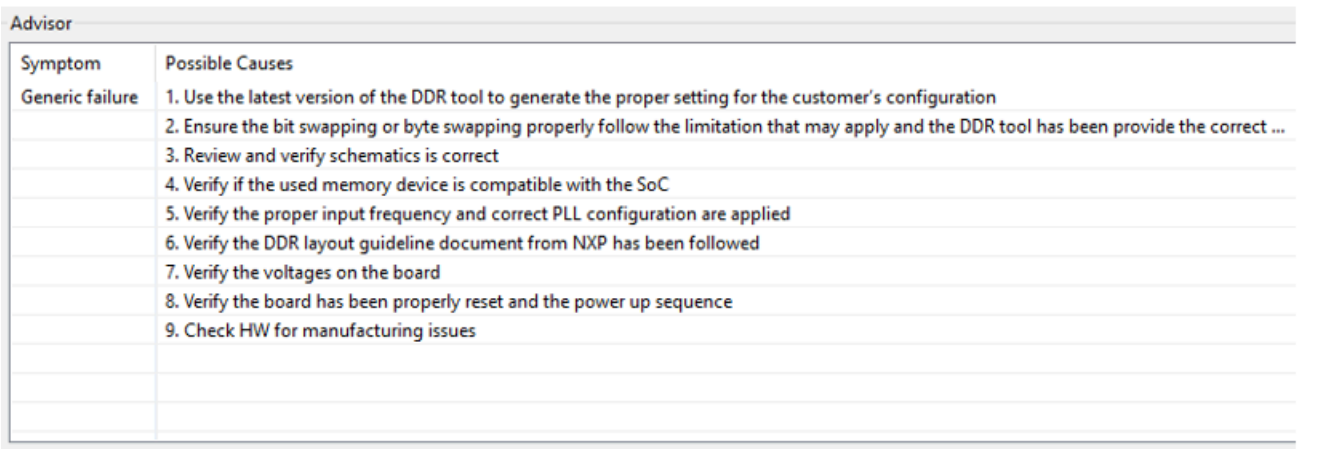


Figure 105. Advisor section

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

4.3.2.1 Inspection

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

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

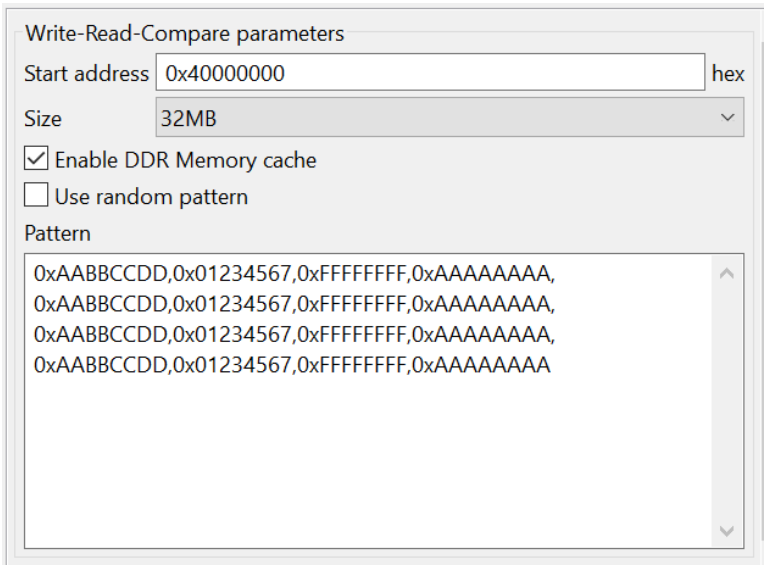


Figure 106. Test options

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 text is possible only for devices with inline ECC support.*

4.3.2.2 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 107. 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.

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

Apply current selection in DDR configuration

Figure 108. 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-comprising 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 our GA BSP. To ensure that your design adheres to the board layout requirements, refer to the device i.MX 8M Hardware Developer's Guide.*

Vref for 1D optimization test sweeps the PHY Vref and/or DRAM Vref to determine the values for the PHY training to pass in case 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.

Results | Choose Tests

Vref for 1D Test

Pass / Total	Searching...
0	1/1

Apply current selection in DDR configuration

Figure 109. 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, which sets the VrefCA value into the configuration, is enabled.

4.3.2.3 vTSA

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

Diag Write Margin/ Diag Read Margin tests creates virtual data eye diagram for each DQ lanes.

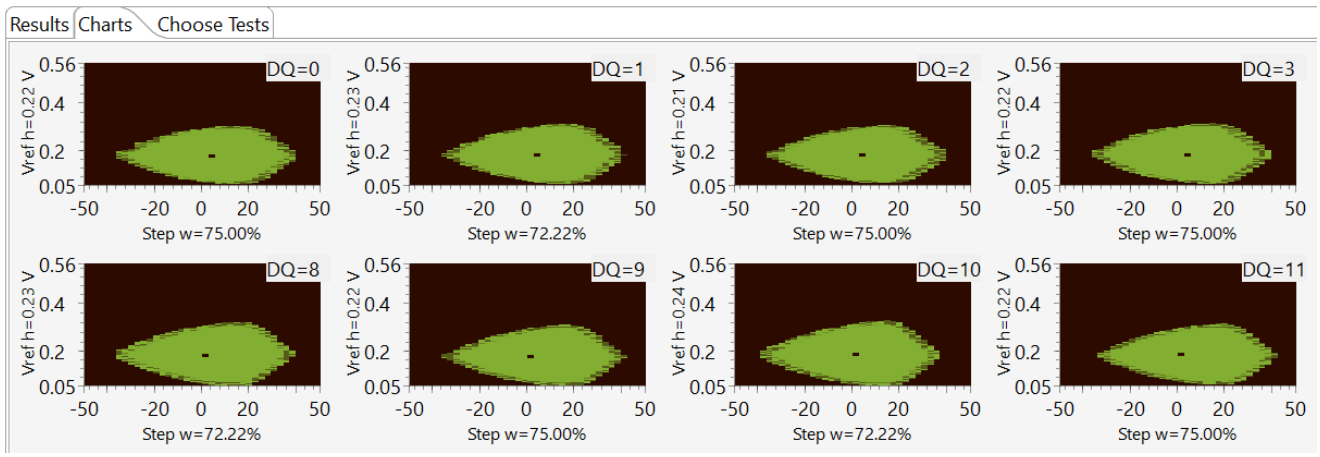


Figure 110. Diagnostic data eyes

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

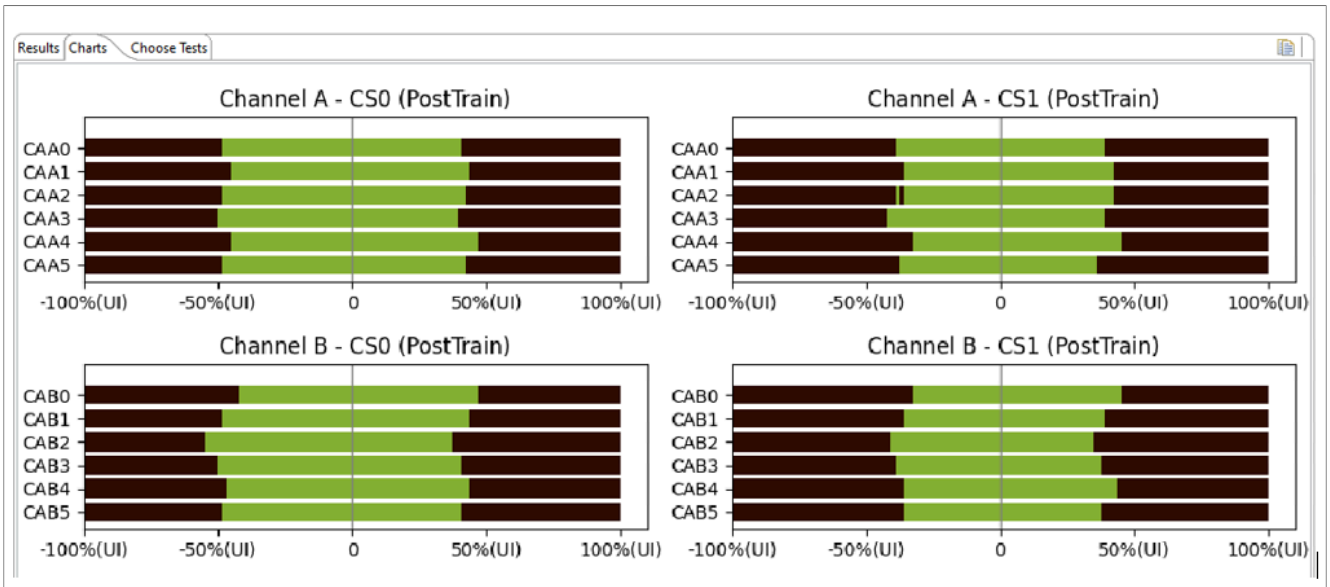


Figure 111. Diagrams of the 6 CA control lines

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

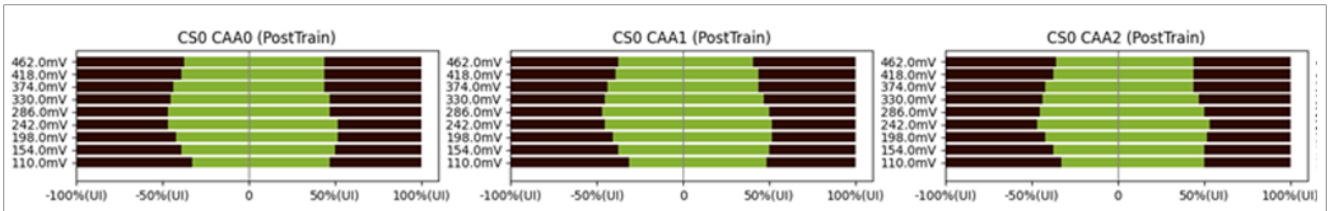


Figure 112. Command/Address eye

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

4.3.2.4 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**

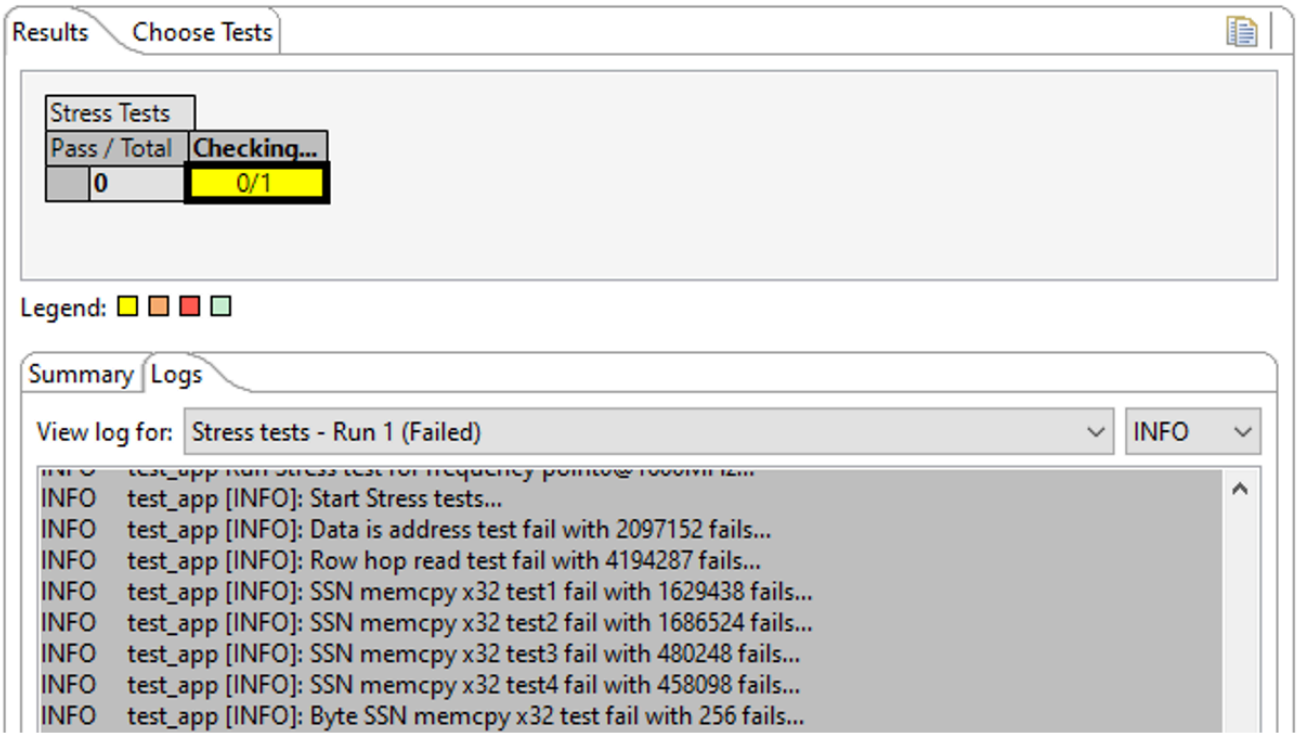


Figure 113. Stress tests

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

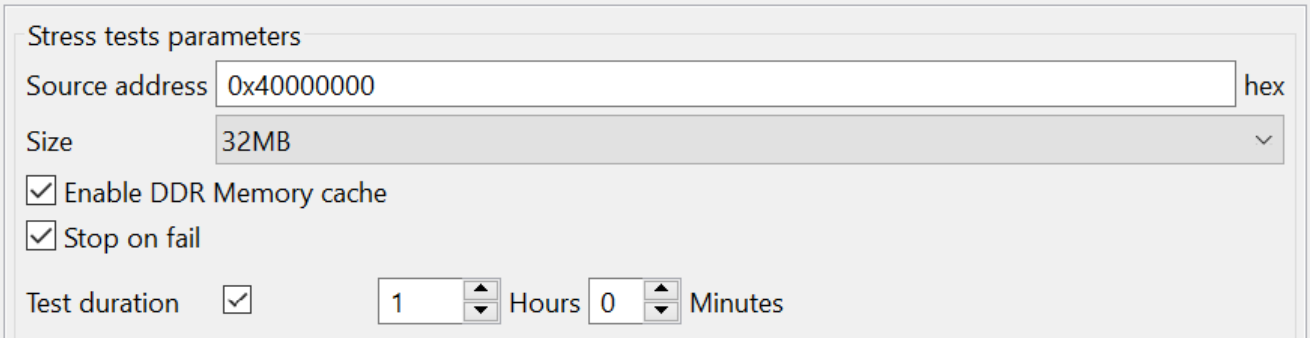


Figure 114. Stress tests options

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

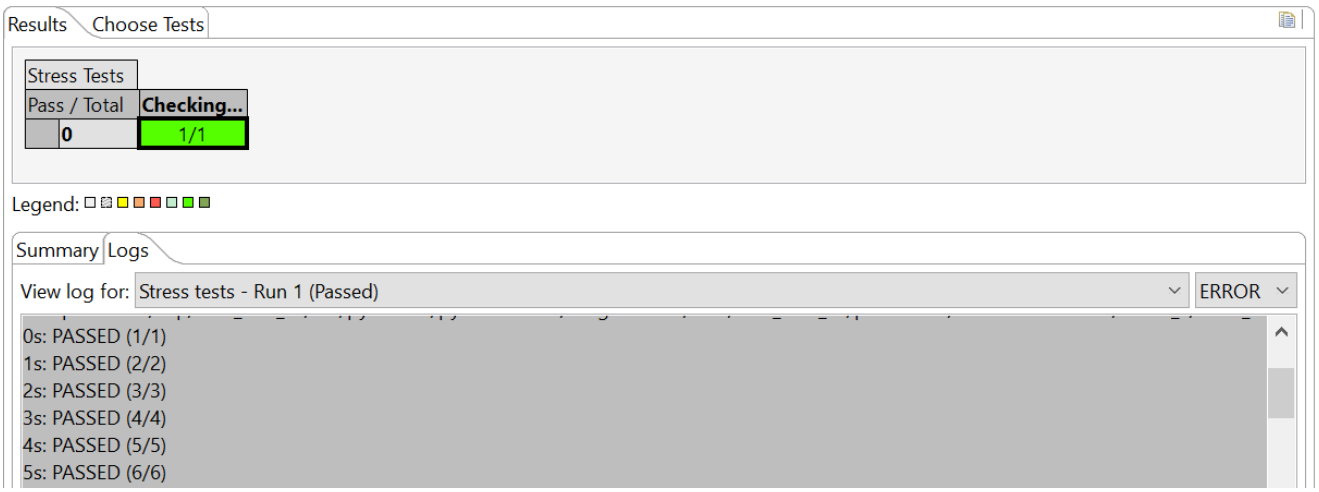
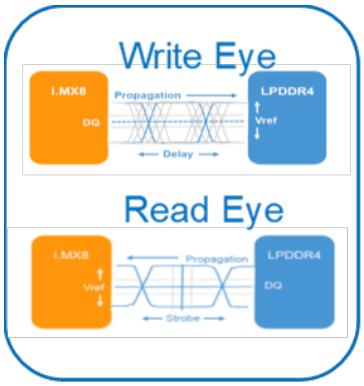


Figure 115. Stress tests results

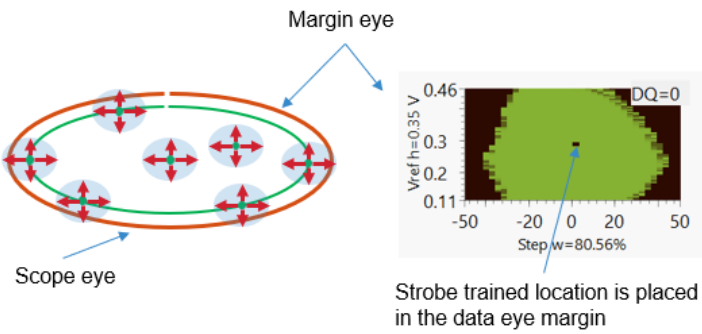
Note: *Make sure the* ***Timeout (seconds)*** *setting is higher than the* ***Test duration*** *setting, otherwise the test ends with timeout.*

4.4 FAQ

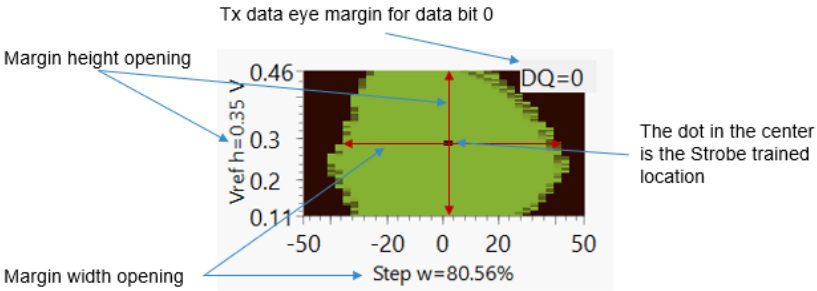
1. What does vTSA mean?
   a. vTSA is an abbreviation for Virtual Timing Signal Analysis.
   b. A “virtual” TSA uses the memory controller itself to test margins without test equipment. “Virtual” does not mean simulation!
   c. Memory controllers have the ability the alter timings, voltage references, 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. This being the case, vTSA output itself approximates the actual write/read eyes.
   c. You should 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. 

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. It is observed and expected that this variation may be up to ~20 ps. The key takeaway from the generated eye diagrams should be focused on verifying the ample margin of the trained delay line value 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. In other words per JEDEC, there is not an MR14 per byte lane and instead, MR14 applies to all byte lanes.
   c. For a board that follows the NXP DDR layout guidelines, there should be plenty of margin around the VREF trained value. You can find the guidelines in the respective NXP Hardware Developer Guide.
4. How a data eye margin is 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 internal 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.



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.

6. How much margin is considered as good?
   a. To determine the required margin mask, you must do the following:
      - Optimize The DDR interface
      - Settings have been optimized, generate the worst-case data eye margin using the worst-case conditions (temperature, voltage, frequency, pattern) for a customer board DDR interface.
      - Use the DDR training optimizing/centering 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. The VREF training must select a value that corresponds to all of the byte lanes' passing VREF window and then program this value into the LPDDR4 MR14 register. It means that for a one-byte lane, though the trained VREF value may not seem to be in the exact center of the data eye, it is selected to provide the best possible margin for this byte lane along with satisfying the other byte lanes.
8. How to check that 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)
```

   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
```

9. How to proceed in case of test timeout?

   a. 

   b.

```
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
```

5 Trusted Execution Environment Tool

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

You can set security policies of different parts of your application in the **Security Access Configuration** and its 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: *In order for your configuration to come into effect, make sure you have enabled the relevant enable secure check option in the* ***Miscellaneous*** *subview of the* ***Security Access Configuration*** *view.*

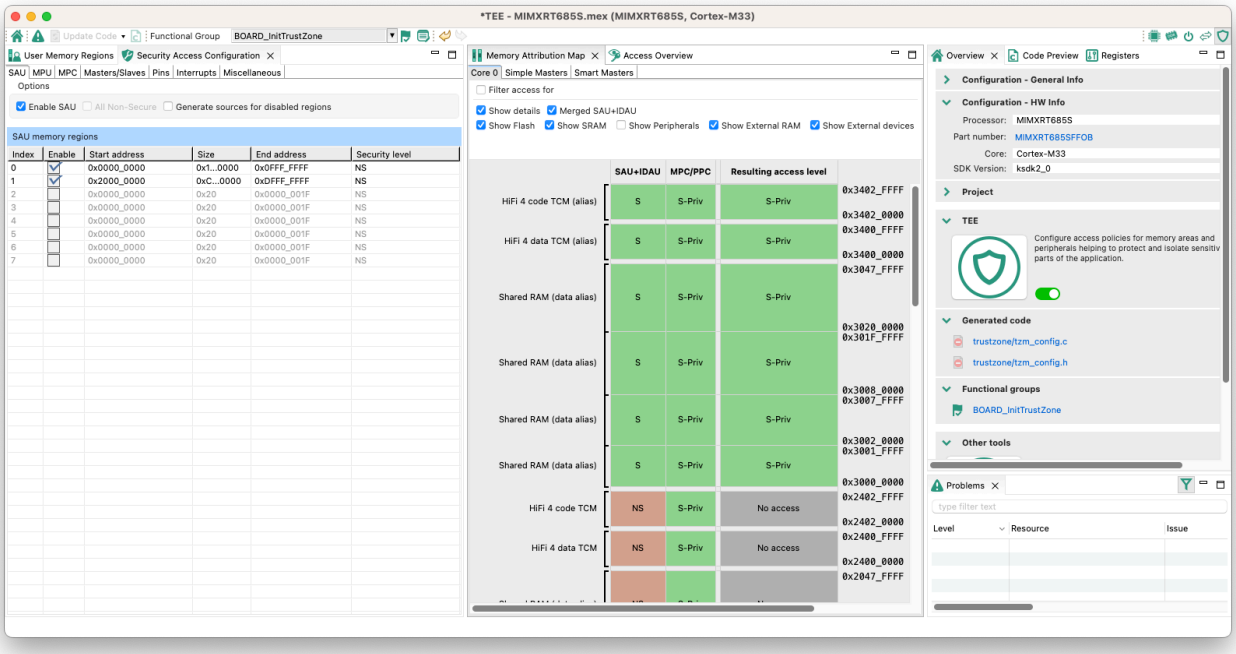


Figure 116. TEE tool user interface (SAU+IDAU)

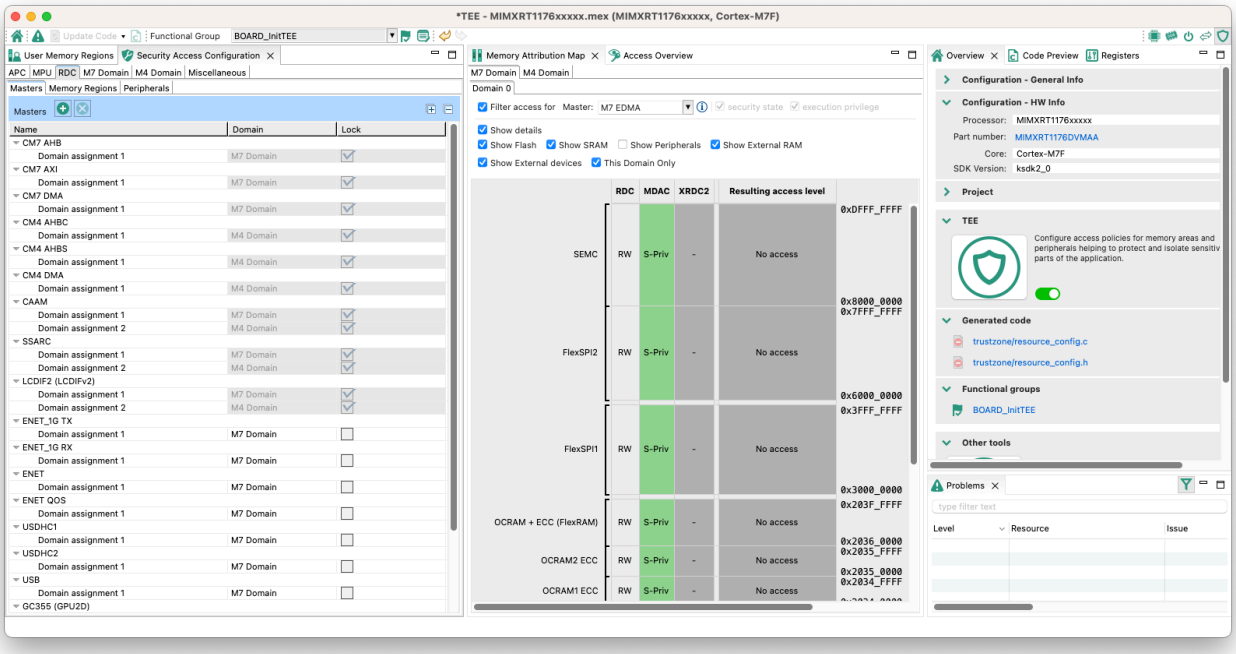


Figure 117. TEE tool user interface (RDC)

5.1 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 security extensions TrustZone-M with AHBSC.

Currently, following devices of this type are supported:

- LPC55Sxx
  - LPC55S69, LPC55S66
  - LPC55S16, LPC55S14
  - LPC55S06, LPC55S04
- RT6xx, RT5xx
  - MIMXRT685S, MIMXRT633S
  - MIMXRT595S, MIMXRT555S, MIMXRT533S1q
    Note: *Pre-production only.*

5.1.1 User Memory Regions view

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

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

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

- **NS-User** - Non-secure user

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

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

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

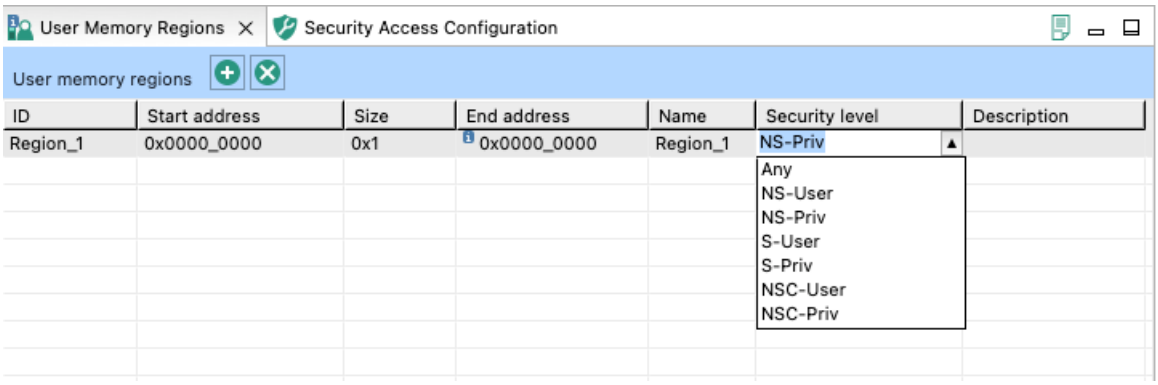


Figure 118. User Memory Regions

5.1.2 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.

5.1.2.1 SAU

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

When enabled, you can set up SAU memory regions, specify their start and size or end address, and specify their access level. SAU automatically sets the entire memory space to a Secure access level when disabled. When enabled, SAU deems every uncovered (that is, unconfigured) memory region as Secure, so only NS or NSC can be selected for a covered (configured) memory region.

You can choose between two access levels:

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

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

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

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

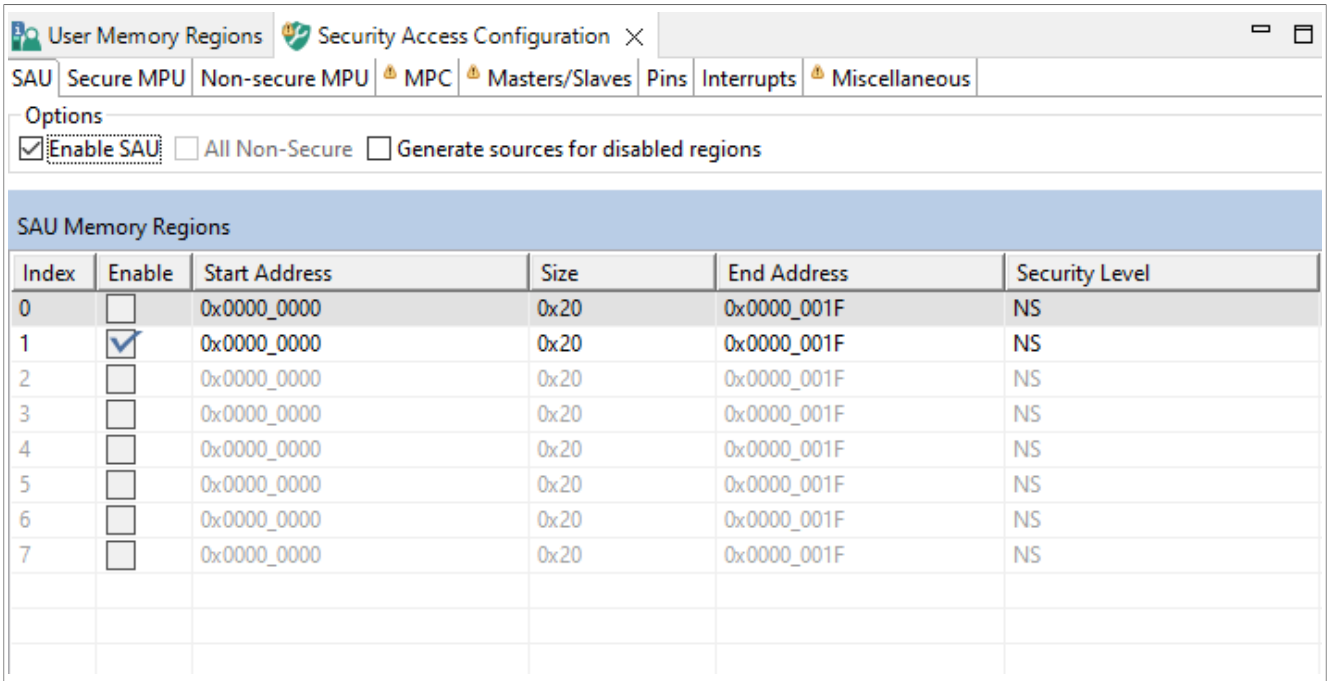


Figure 119. SAU/IDAU

5.1.2.2 Interrupts

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

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

User Memory Regions | Security Access Configuration

SAU | Secure MPU | Non-secure MPU | MPC | Masters/Slaves | Pins | Interrupts | Miscellaneous

Interrupt security configuration

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 120. Interrupts

5.1.2.3 Secure/Non-secure MPU

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

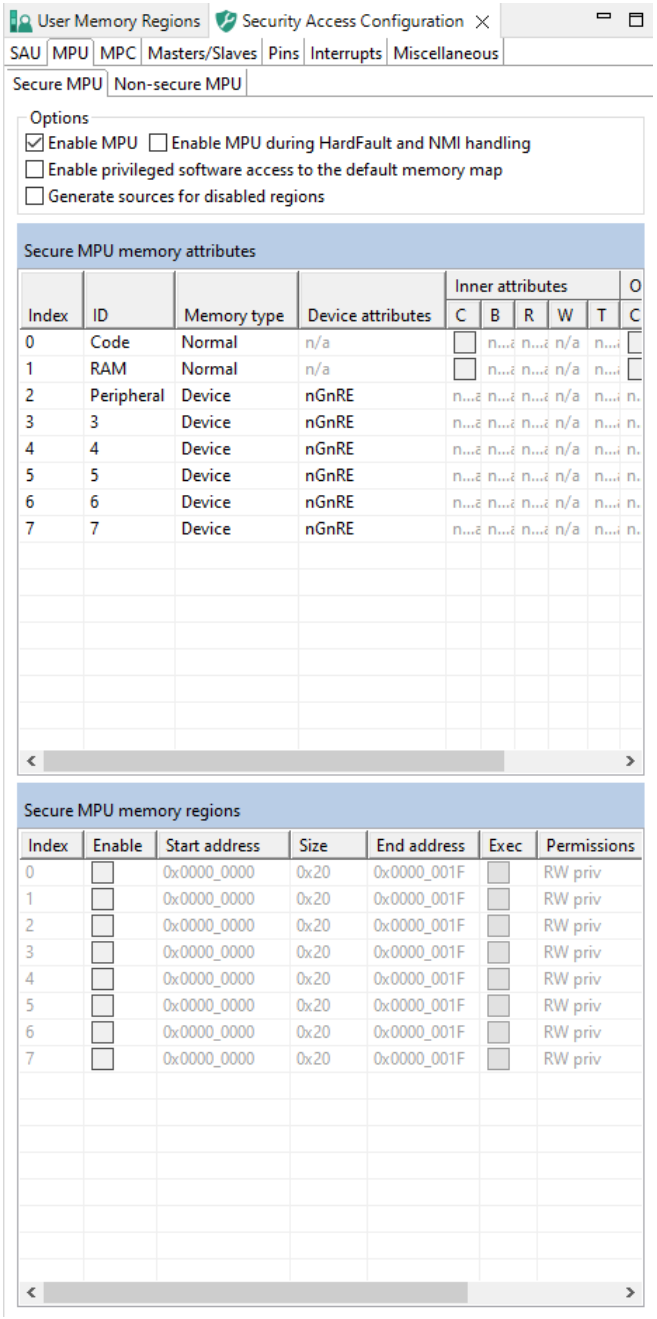


Figure 121. MPU

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

Note: *Not every device supports MPU.*

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

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

1. **Enable** the region.
2. Specify the **Address**.
3. Specify either the **Size** or the **End Address**.

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

5.1.2.4 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

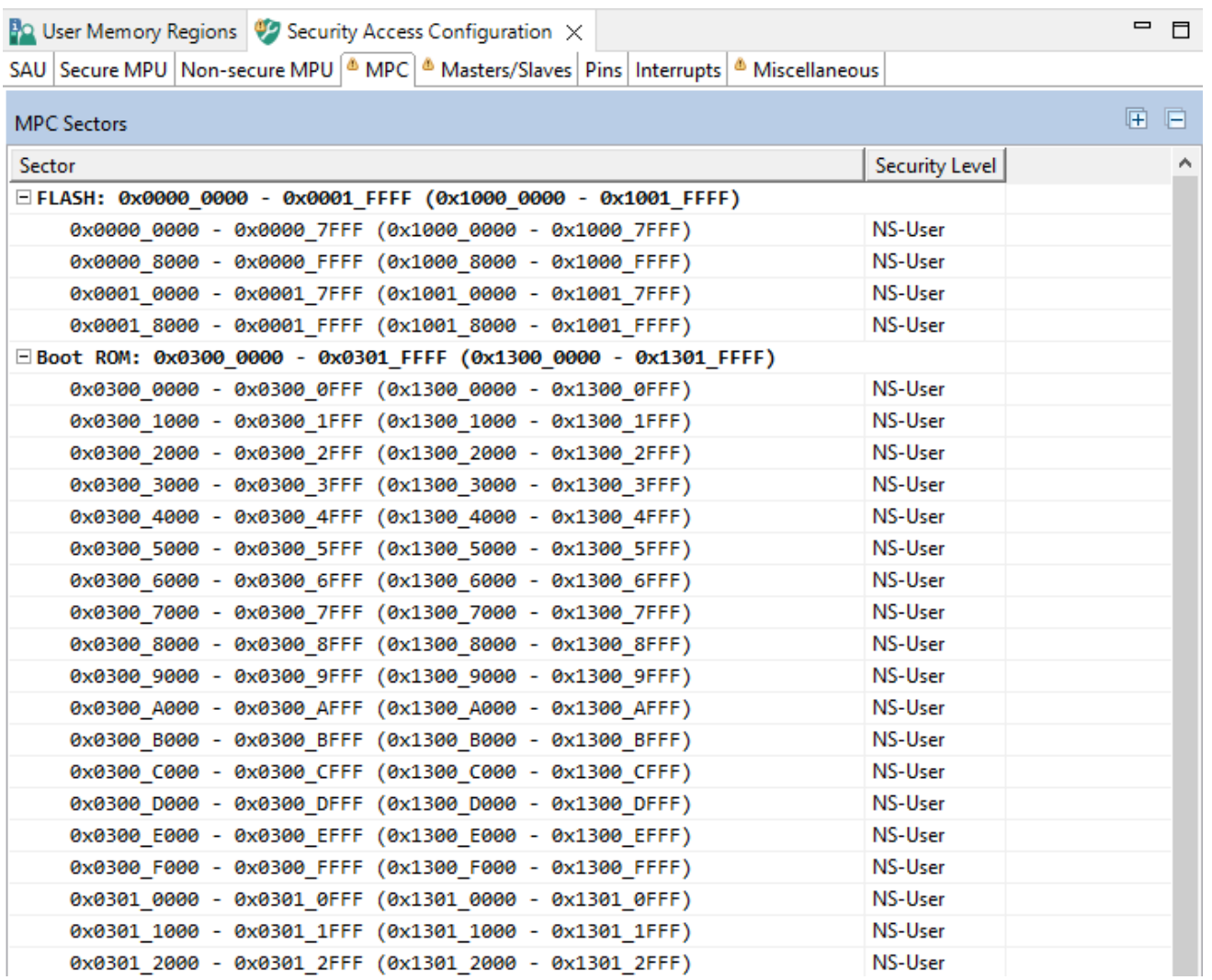


Figure 122. MPC

5.1.2.5 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.*

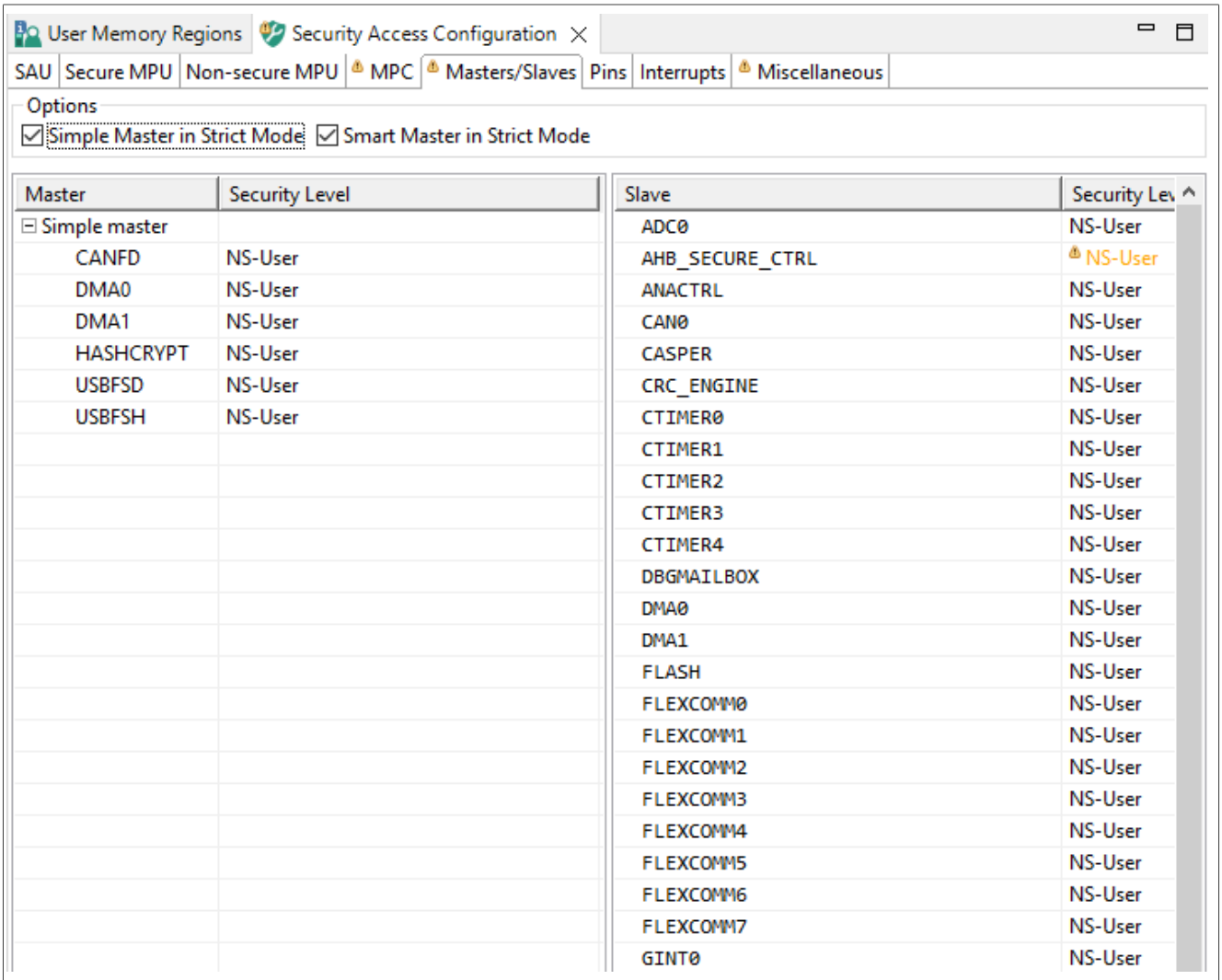


Figure 123. Masters/Slaves

5.1.2.6 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. If you want to change the pins reading GPIO state, left-click the **Reading GPIO state** cell of the pin and choose from the dropdown menu. Alternatively, right-click the pin's **Name** cell and choose the reading GPIO state from the context menu. To select multiple entries, use the **Ctrl+Left-click** shortcut, then right-click the selected area for the context menu. Alternatively, you can use **Shift +Up/Down** after selecting the row to expand the selection.

User Memory Regions | Security Access Configuration

SAU | Secure MPU | Non-secure MPU | MPC | Masters/Slaves | Pins | Interrupts | Miscellaneous

Reading GPIO state

Name	Reading GPIO state
General purpose input/output port 0	
Pin 0 – [54] PIO0_0/FC3_SC...GPI0/SECURE_GPIO0_0/ACMP0	Allow
Pin 1 – [7] PIO0_1/FC3_CTS_...GPI1/CMP0_OUT/SECURE_GPI	Allow
Pin 2 – [81] PIO0_2/FC3_TXD...UT0/SCT_GPI2/SECURE_GPIO0	Allow
Pin 3 – [83] PIO0_3/FC3_RXD...UT1/SCT_GPI3/SECURE_GPIO0	Allow
Pin 4 – [86] PIO0_4/CAN0_R...TS_SDA_SSEL0/SECURE_GPIO0	Allow
Pin 5 – [88] PIO0_5/CAN0_TD...L_SSEL1/MCLK/SECURE_GPIC	Allow
Pin 6 – [89] PIO0_6/FC3_SC...AT0/SCT_GPI6/SECURE_GPIO0_	Allow
Pin 7 – [6] PIO0_7/FC3_RTS_..._SCK/FC1_SCK/SECURE_GPIO0	Allow
Pin 8 – [26] PIO0_8/FC3_SS...SI_DATA/SWO/SECURE_GPIO0_	Allow
Pin 9 – [55] PIO0_9/FC3_SS..._WS/SECURE_GPIO0_9/ACMP0_	Allow
Pin 10 – [21] PIO0_10/FC6_S.../SWO/SECURE_GPIO0_10/ADC	Allow
Pin 11 – [13] PIO0_11/FC6_RX...SWCLK/SECURE_GPIO0_11/A	Allow
Pin 12 – [12] PIO0_12/FC3_T..._WS/SECURE_GPIO0_12/ADC0	Allow
Pin 13 – [71] PIO0_13/FC1_CT...DATA/PLU_IN0/SECURE_GPIC	Allow
Pin 14 – [72] PIO0_14/FC1_RT...O_WS/PLU_IN1/SECURE_GPIC	Allow
Pin 15 – [22] PIO0_15/FC6_C...OUT2/SECURE_GPIO0_15/ADC	Allow
Pin 16 – [14] PIO0_16/FC4_TX..._INP4/SECURE_GPIO0_16/AD	Allow
Pin 17 – [8] PIO0_17/FC4_SSE...OUT0/PLU_IN2/SECURE_GPIC	Allow
Pin 18 – [56] PIO0_18/FC4_C...IN3/SECURE_GPIO0_18/ACMP	Allow
Pin 19 – [90] PIO0_19/FC4_R...O_WS/PLU_IN4/SECURE_GPIO	Allow
Pin 20 – [74] PIO0_20/FC3_...PIO0_20/FC4_TXD_SCL_MISO_W	Allow
Pin 21 – [76] PIO0_21/FC3_RT...L3/PLU_CLKIN/SECURE_GPIC	Allow
Pin 22 – [78] PIO0_22/FC6_...US/PLU_OUT7/SECURE_GPIO0_	Allow
Pin 23 – [20] PIO0_23/MCLK...SEL0/SECURE_GPIO0_23/ADC0	Allow
Pin 24 – [70] PIO0_24/FC0_R...P8/SCT_GPI0/SECURE_GPIO0_	Allow
Pin 25 – [79] PIO0_25/FC0_T...NP9/SCT_GPI1/SECURE_GPIO0	Allow
Pin 26 – [60] PIO0_26/FC2_R...HS_SPI_MOSI/SECURE_GPIO0_	Allow

Figure 124. Pins tab on LPC55S69

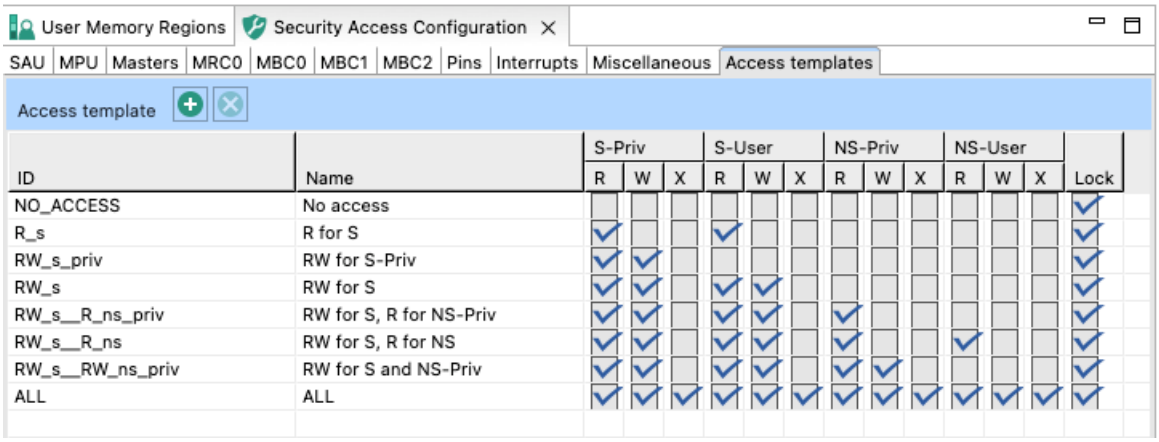


Figure 125. Global Access Templates

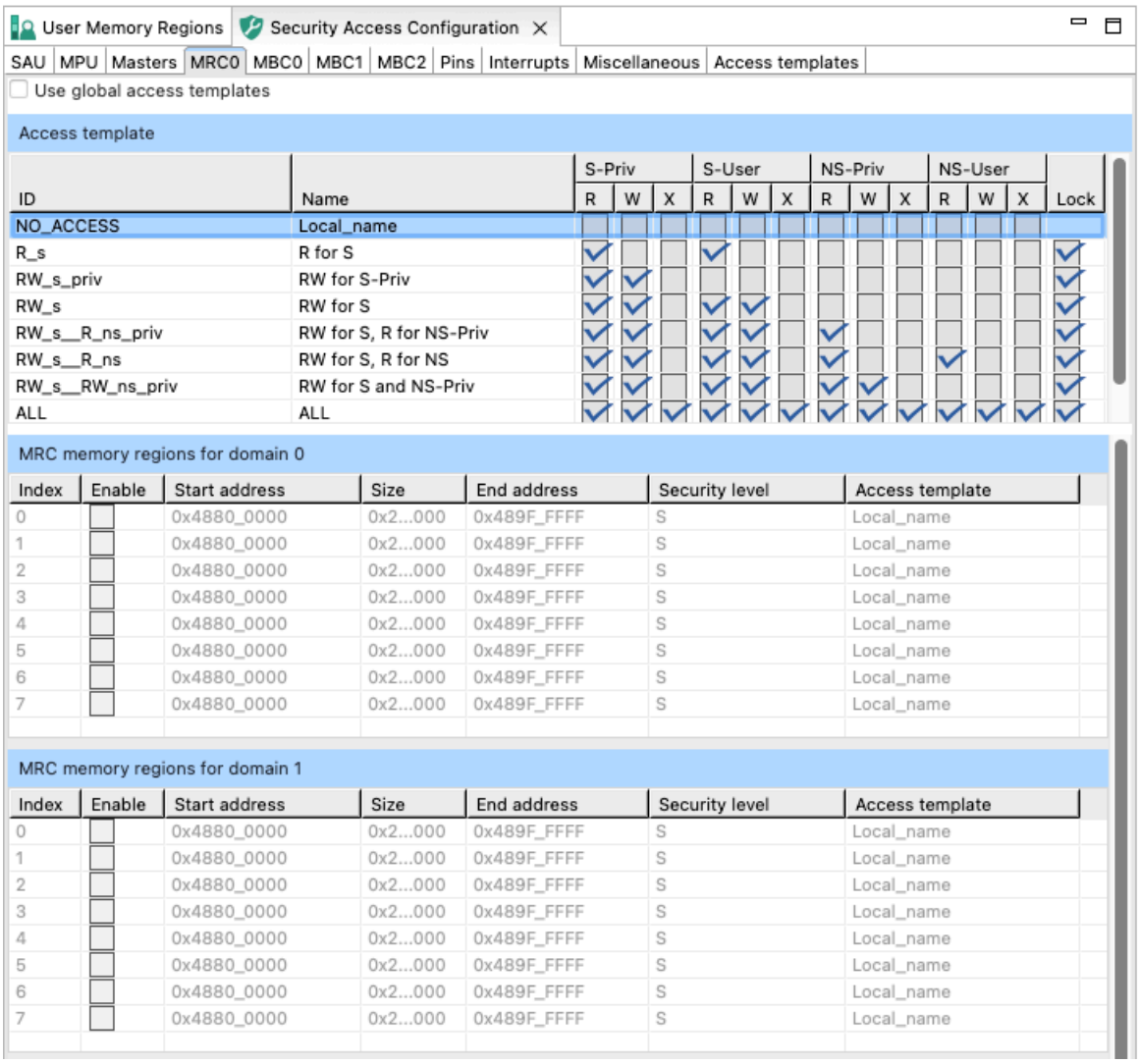


Figure 126. Local access templates

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

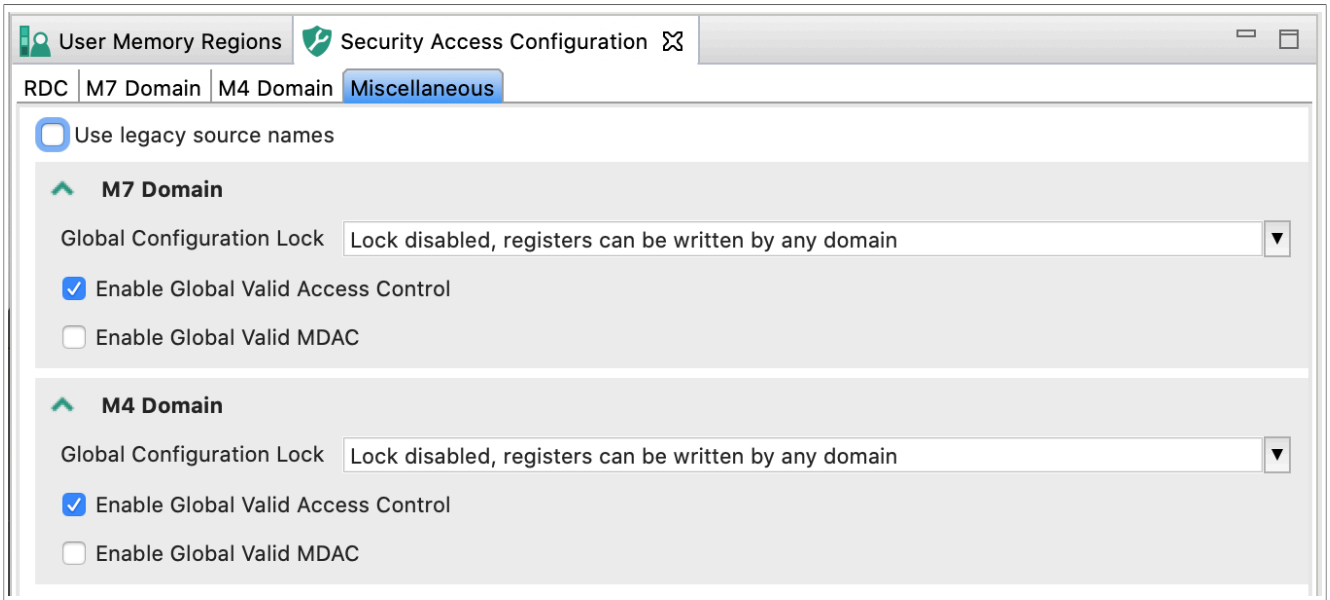


Figure 127. Miscellaneous (RDC+XRDC2)

5.1.3 Memory attribution map

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

5.1.3.1 Core 0

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

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

To set the display options, do the following:

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

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

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

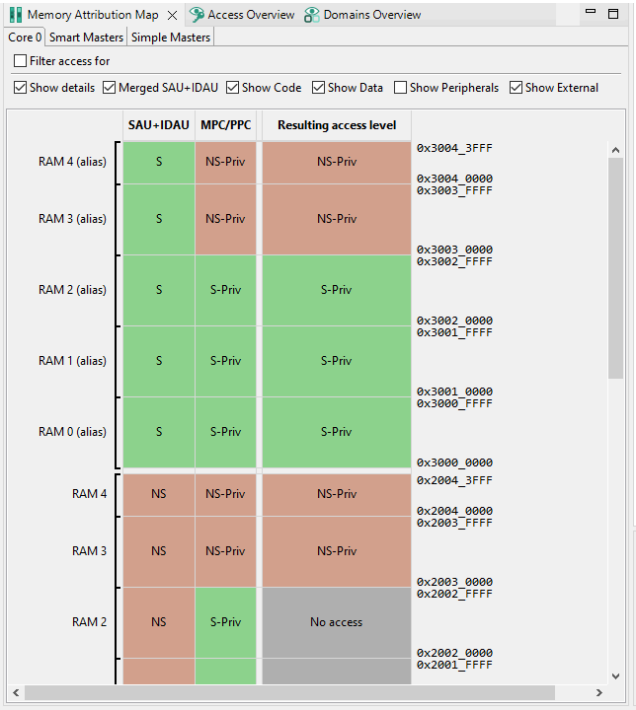


Figure 128. Core 0

5.1.3.2 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**.

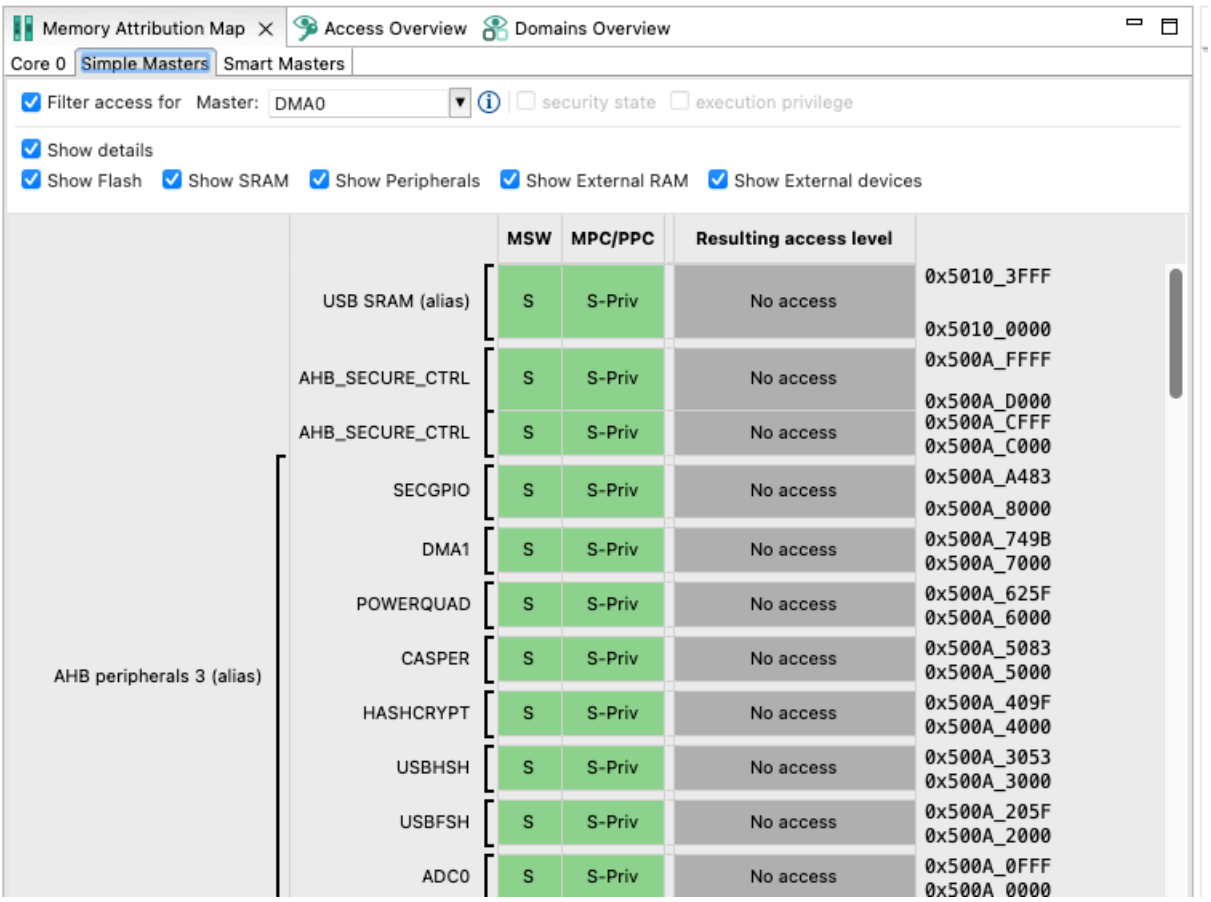


Figure 129. Simple/Smart masters

5.1.4 Access Overview

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

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

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

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

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

Memory Attribution Map | Access Overview

Slave	NS-User								NS-Priv		S-User		S-Priv	
	CANFD	Core 0 Data	Core 0 I...ructions	DMA0	DMA1	HASHCRYPT	USBFSD	USBFSH	Core 0 Data	Core 0 I...ructions	Core 0 Data	Core 0 I...ructions	Core 0 Data	Core 0 I...ructions
Memory														
PROGRAM FLASH														
0x0000_0000 - 0x0001_FFFF (0x1000_0000 - 0x1001_FFFF)	n/a	✓	✓	✓	✓	✓	n/a	n/a	✓	✓	✓		✓	
Boot-ROM														
0x0300_0000 - 0x0301_FFFF (0x1300_0000 - 0x1301_FFFF)	n/a	✓	✓	✓	✓	✓	n/a	n/a	✓	✓	✓		✓	
SRAM X														
0x0400_0000 - 0x0400_3FFF (0x1400_0000 - 0x1400_3FFF)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	
RAM 0														
0x2000_0000 - 0x2000_7FFF (0x3000_0000 - 0x3000_7FFF)	✓	✓	n/a	✓	✓	✓	✓	✓	✓	n/a	✓	n/a	✓	n/a
RAM 1														
0x2000_8000 - 0x2000_BFFF (0x3000_8000 - 0x3000_BFFF)	✓	✓	n/a	✓	✓	✓	✓	✓	✓	n/a	✓	n/a	✓	n/a
USB SRAM														
0x2001_0000 - 0x2001_3FFF (0x3001_0000 - 0x3001_3FFF)	✓	✓	n/a	✓	✓	✓	✓	✓	✓	n/a	✓	n/a	✓	n/a
Peripherals														
ADC0	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a
AHB_SECURE_CTRL	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a
ANACTRL	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a
CAN0	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a
CASPER	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a
CRC_ENGINE	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a
CTIMER0	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a
CTIMER1	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a
CTIMER2	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a
CTIMER3	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a
CTIMER4	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a
DBGMAILBOX	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a
DMA0	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a
DMA1	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a
FLASH	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a
FLEXCOMM0	n/a	✓	n/a	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	✓	n/a

Figure 130. Access Overview

5.1.5 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 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.

5.2 RDC-enabled devices

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

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

Currently, following devices of this type are supported:

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

5.2.1 User Memory Regions view

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

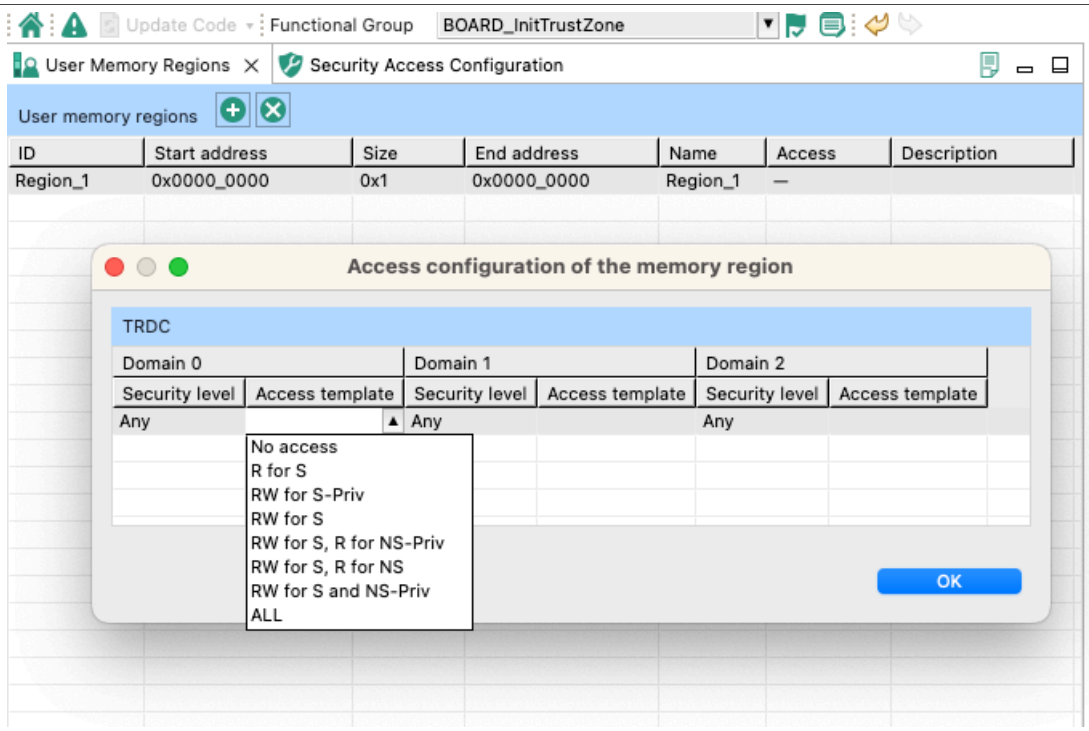


Figure 131. User Memory Regions

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

Enter/change the memory region's parameters by clicking the row's cells.

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

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

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

5.2.1.1 Access templates

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

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

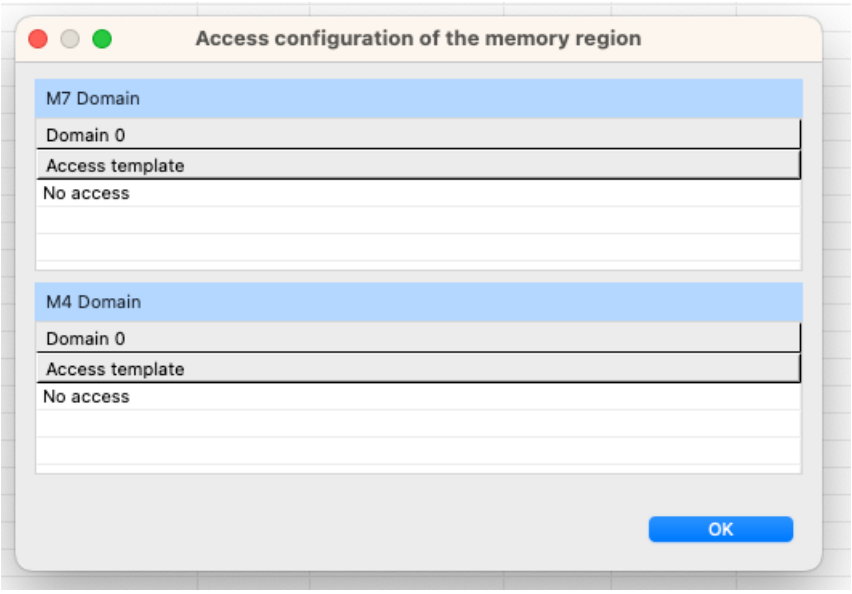


Figure 132. Access template

Select access template by clicking the topmost cell of 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.

5.2.2 Security Access Configuration view

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

5.2.2.1 RDC

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

5.2.2.1.1 RDC Masters

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

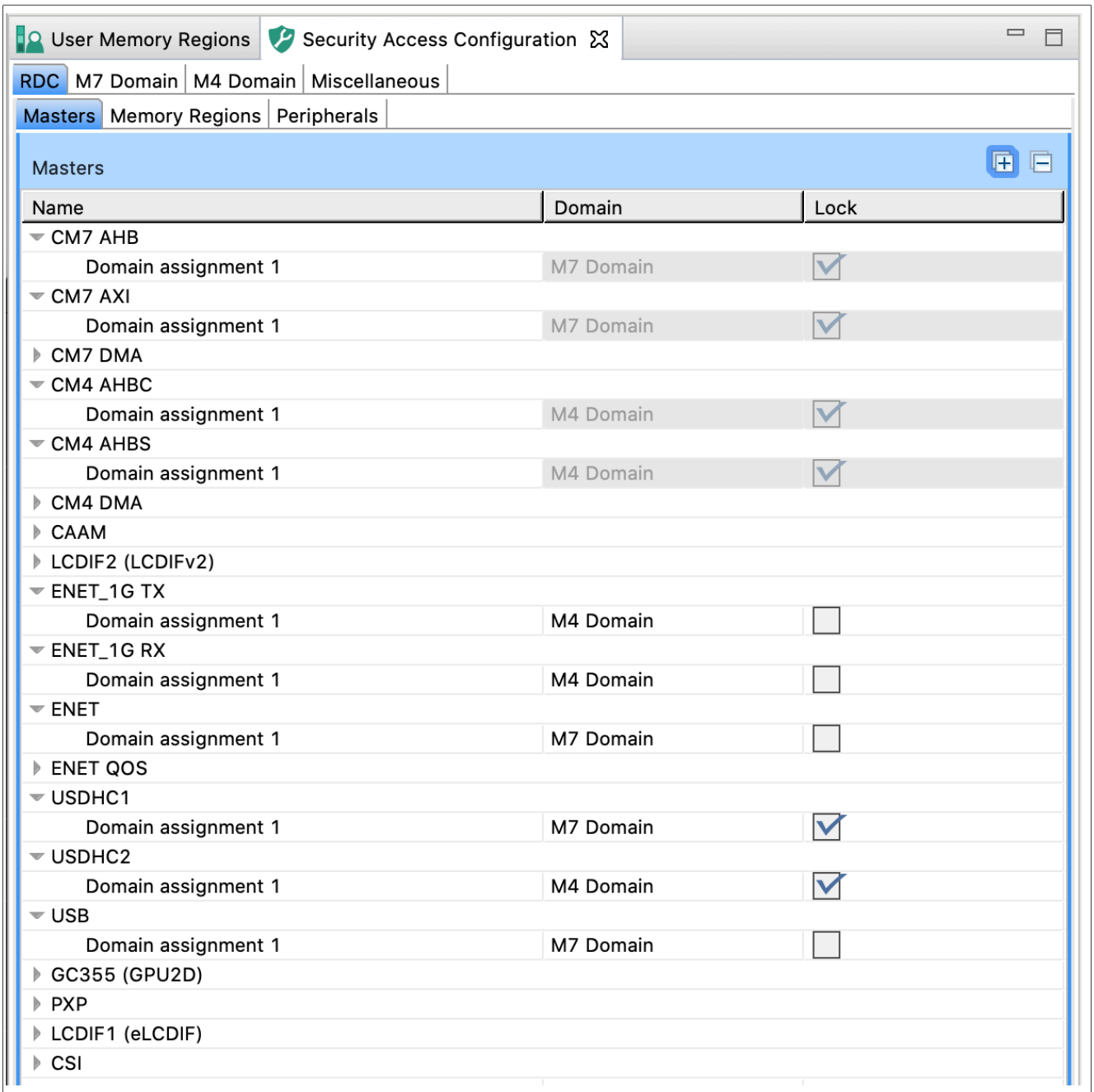


Figure 133. RDC Masters

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.

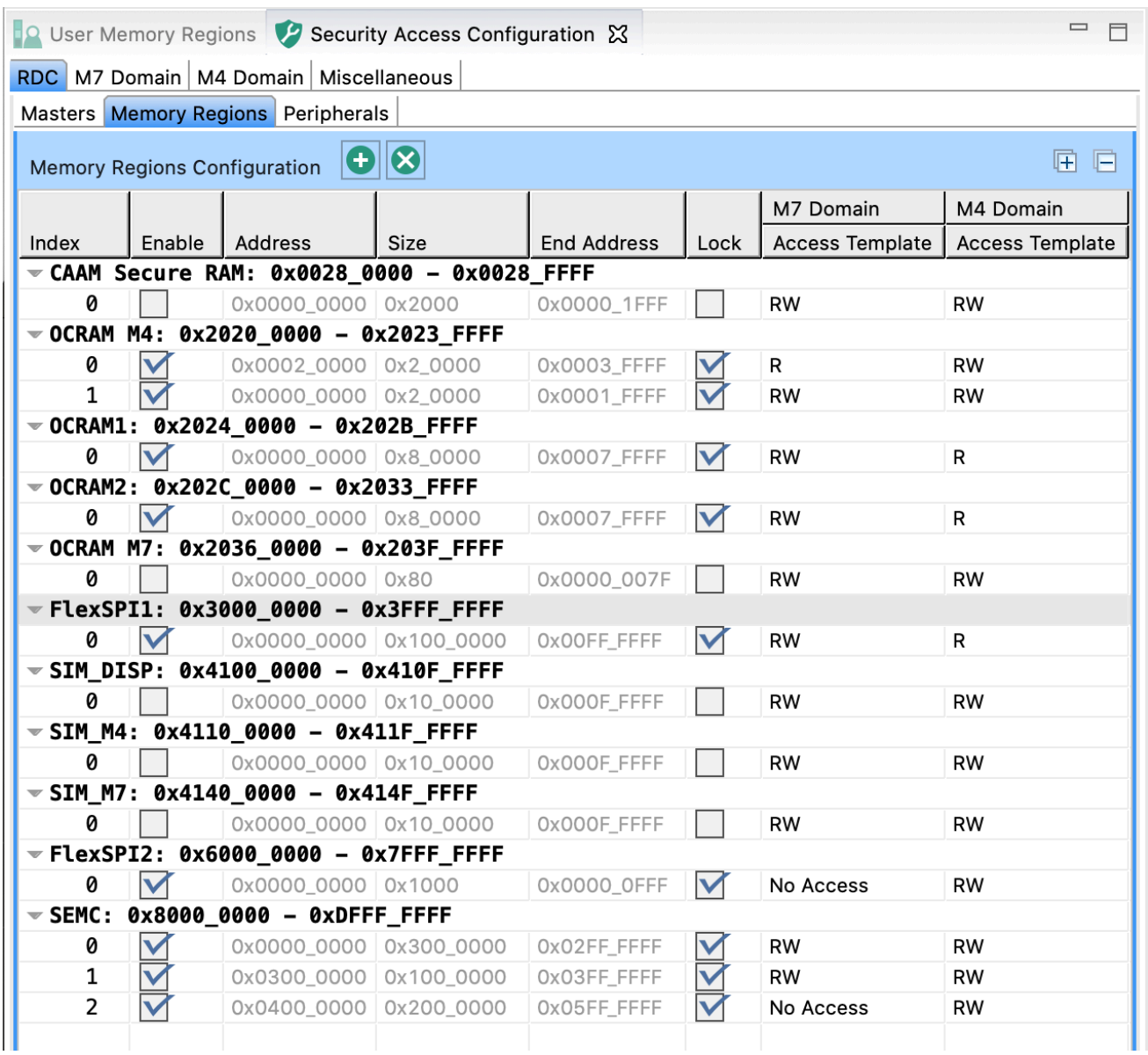
Alternatively, you can select the options by right-clicking the master and using the dropdown list.

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

5.2.2.1.2 Memory Regions

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

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



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

Figure 134. Memory Regions

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

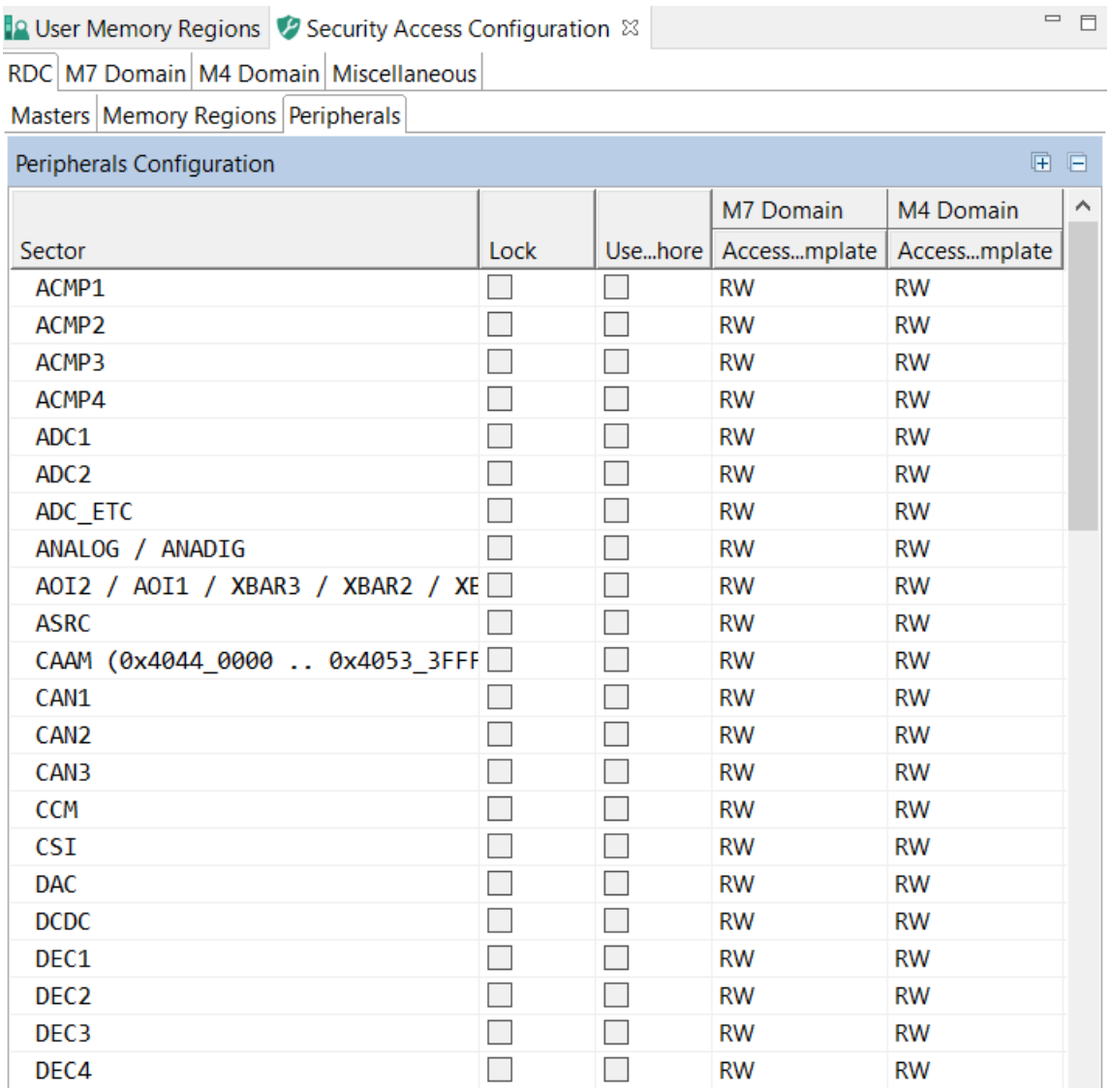
1. **Enable** the region.
2. Specify the **Address**.
3. Specify either the **Size** or the **End Address**.
4. Optional: **Lock** the settings to prevent further register modifications.

5. Set the **Access Template** for available domains.

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

5.2.2.1.3 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 135. Peripherals

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

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

3. Set the **Access Template** for available domains.

5.2.2.2 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.

5.2.2.2.1 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, and enforces access rules to memory, and supports the standard ARMv7 Protected Memory System Architecture model.

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

Note: *Not every device supports MPU.*

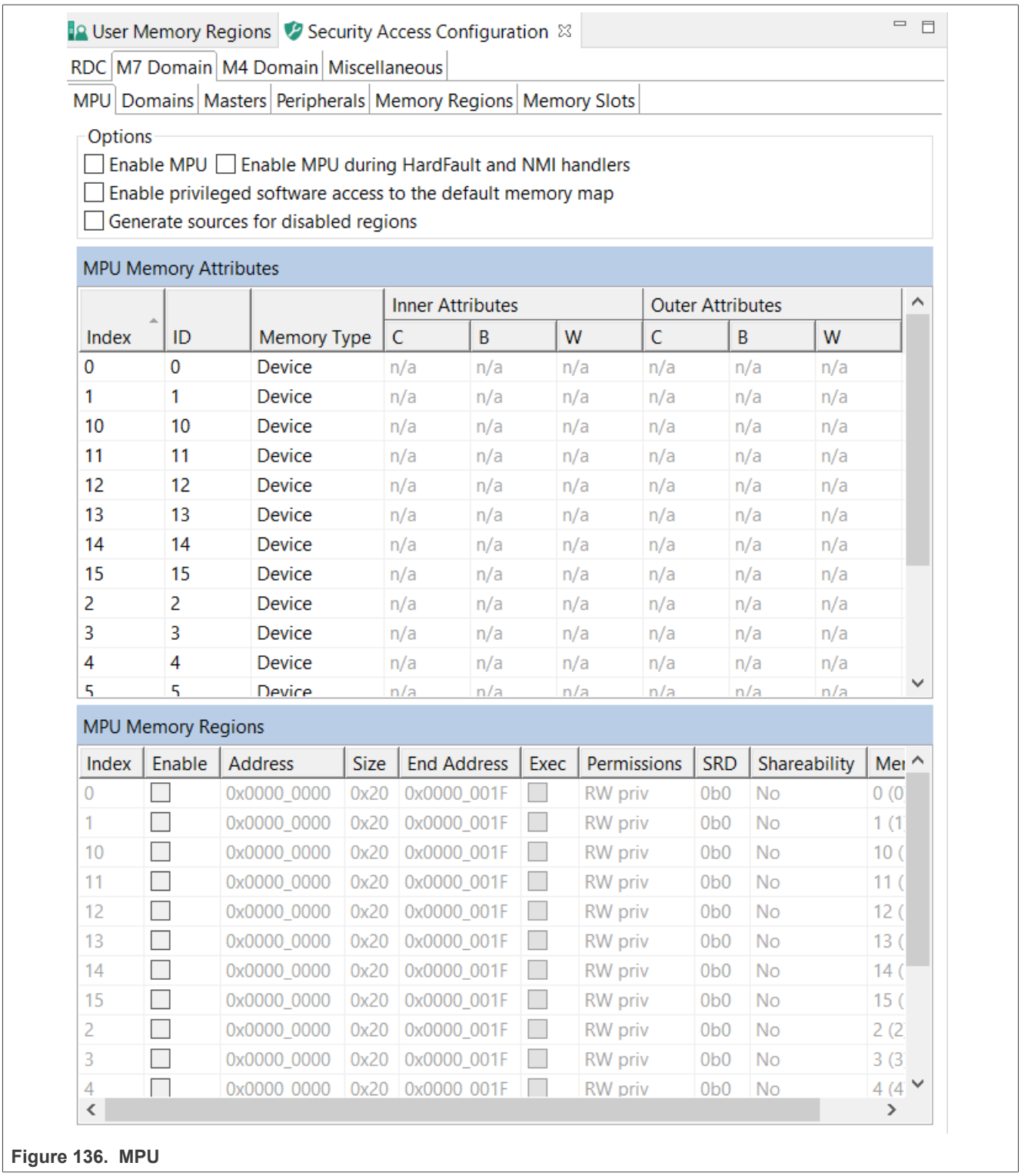


Figure 136. MPU

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

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

1. **Enable** the region.

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

5.2.2.2.2 Domains

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

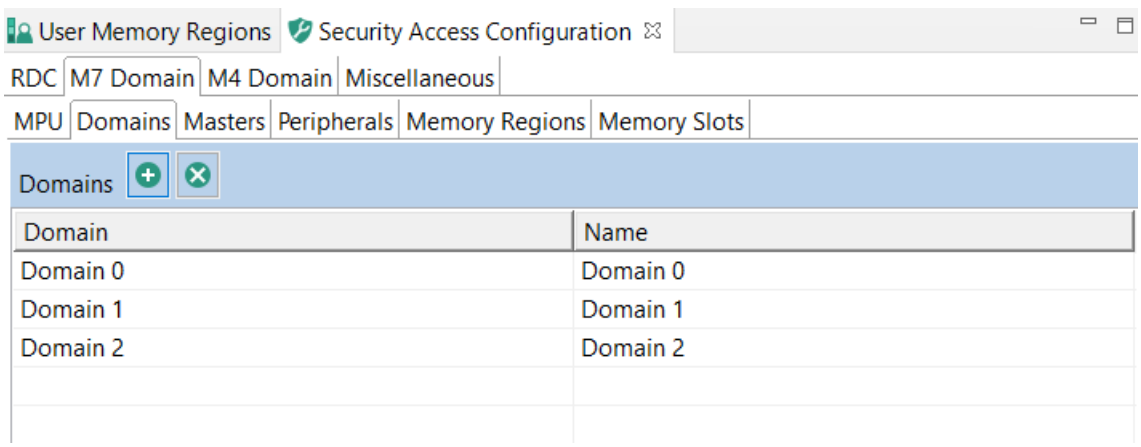


Figure 137. Domains

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.

5.2.2.2.3 Masters

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

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

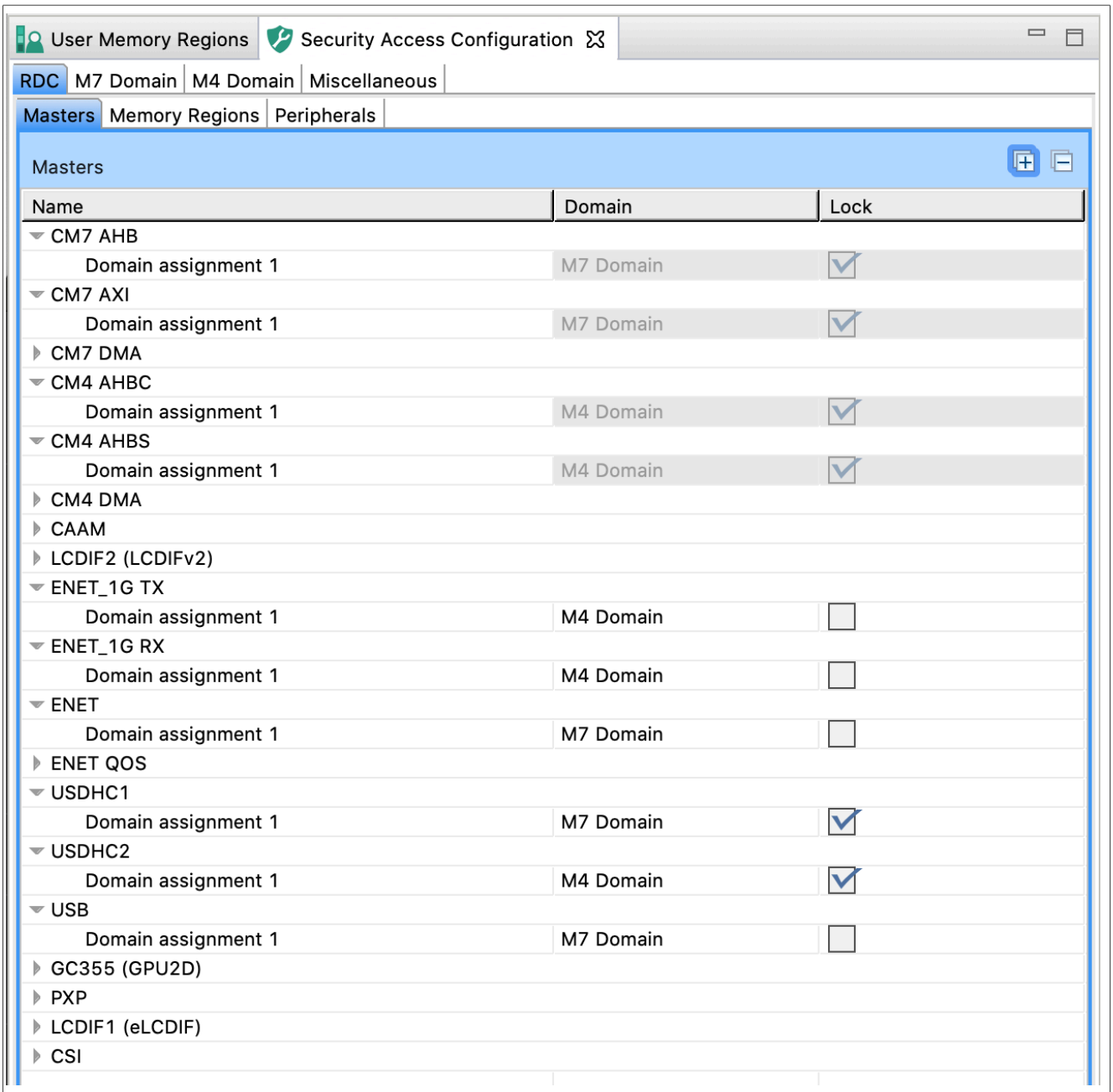


Figure 138. Masters

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.

5.2.2.2.4 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.

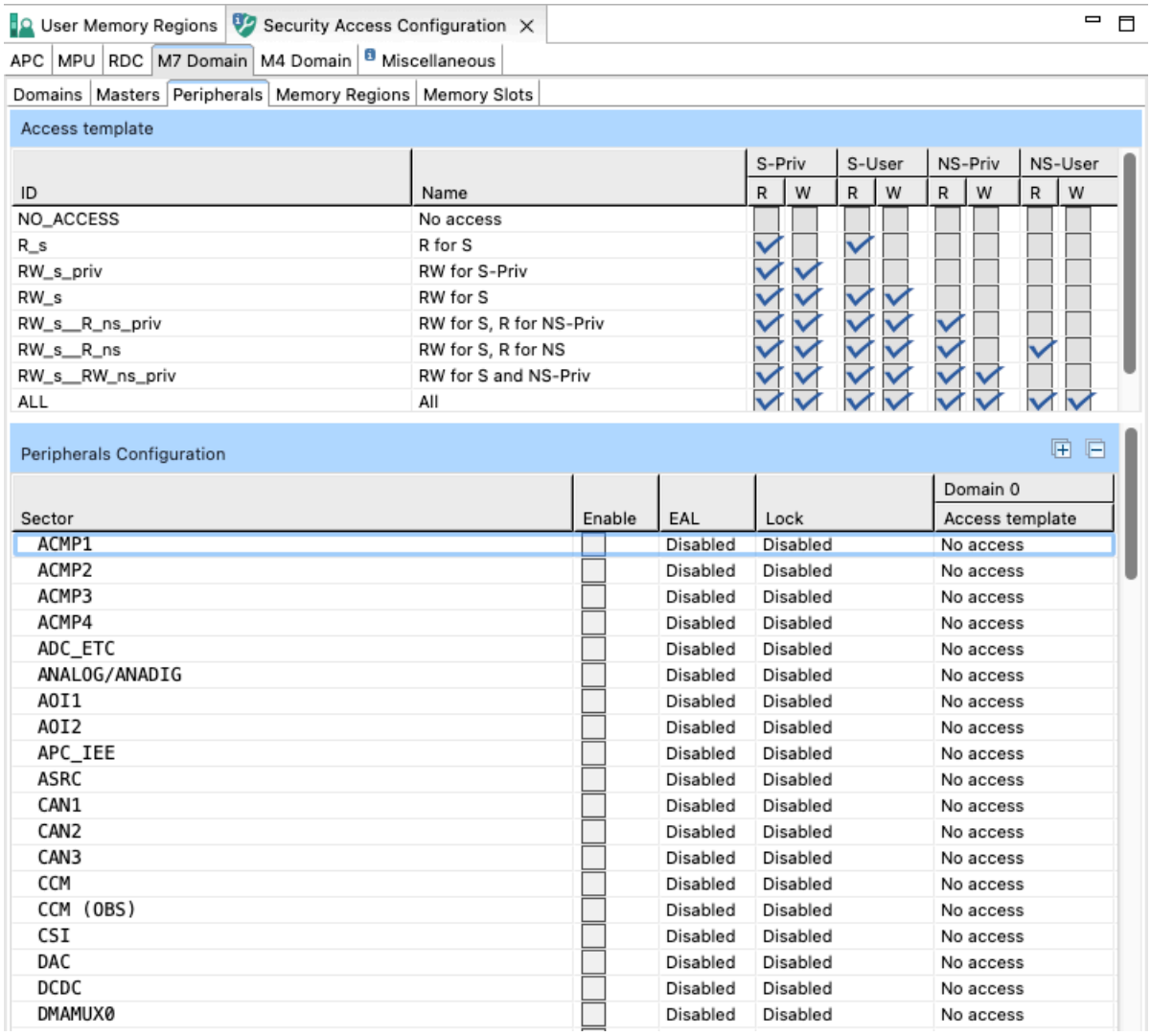


Figure 139. Peripherals

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

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

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

5.2.2.2.5 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.

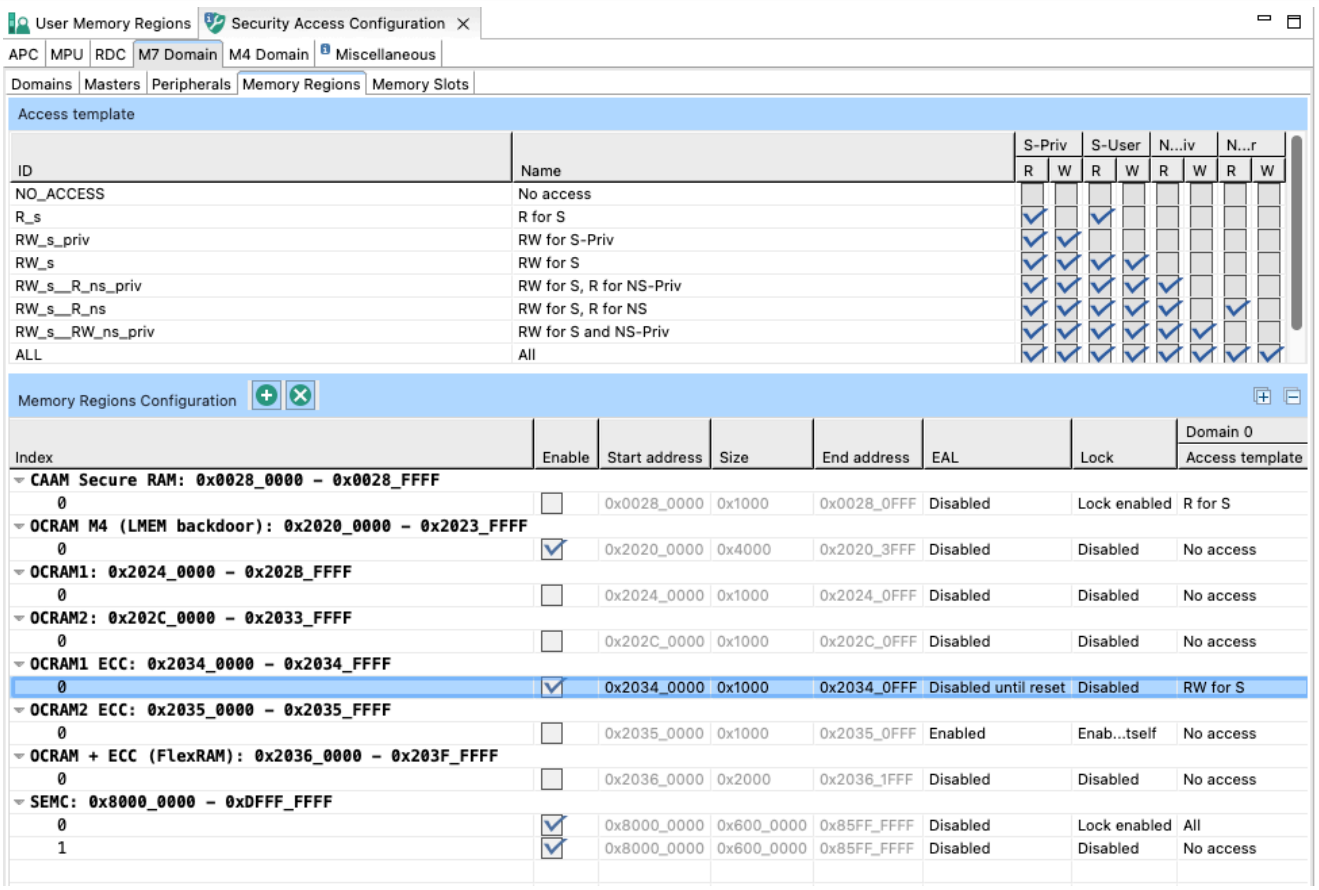


Figure 140. Memory Regions

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

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

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

5.2.2.2.6 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.

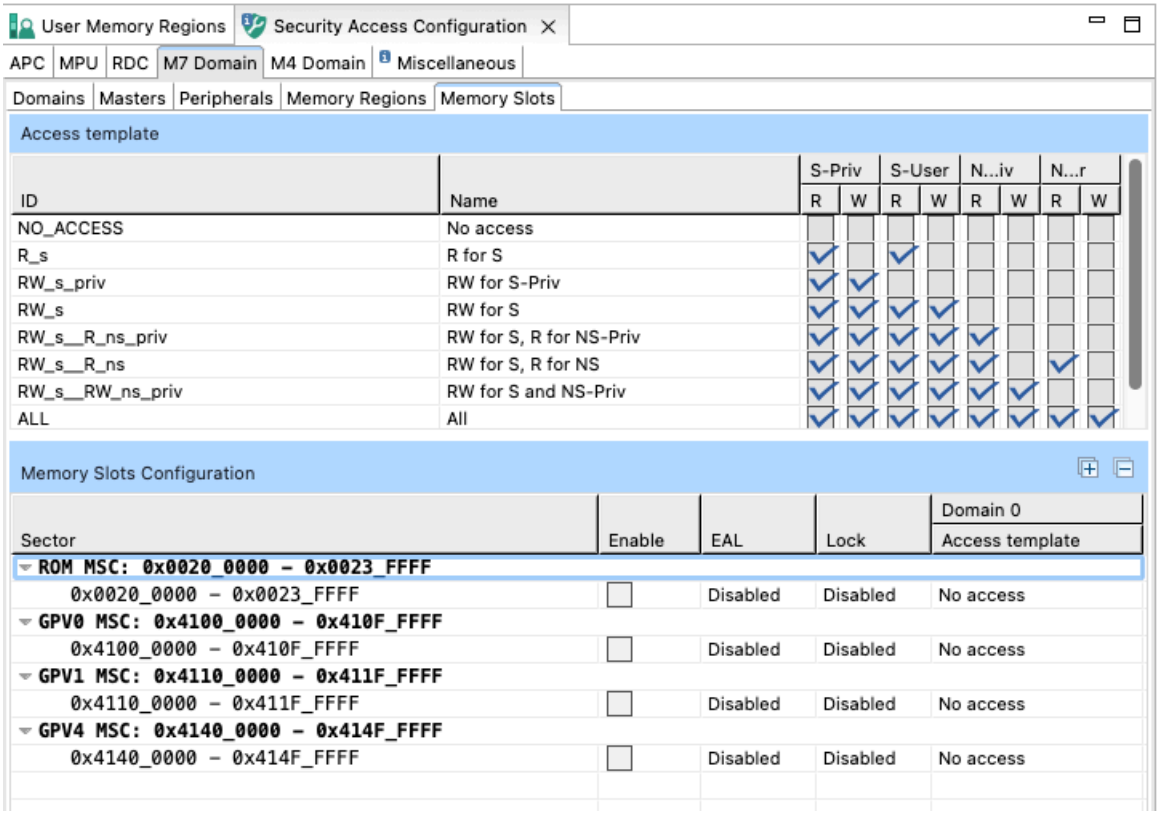


Figure 141. Memory Slots

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.

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

5.2.2.3.1 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 Reference Manual (link to be provided).

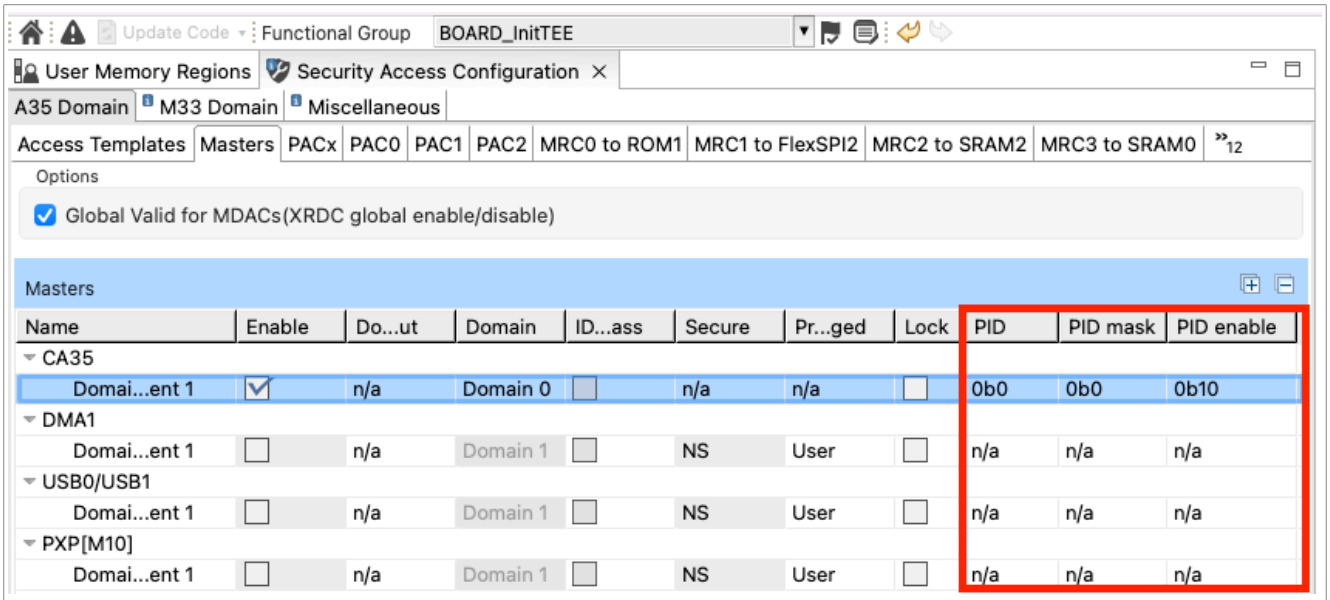


Figure 142. XtRDC master tab

5.2.2.3.2 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, the user can 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.

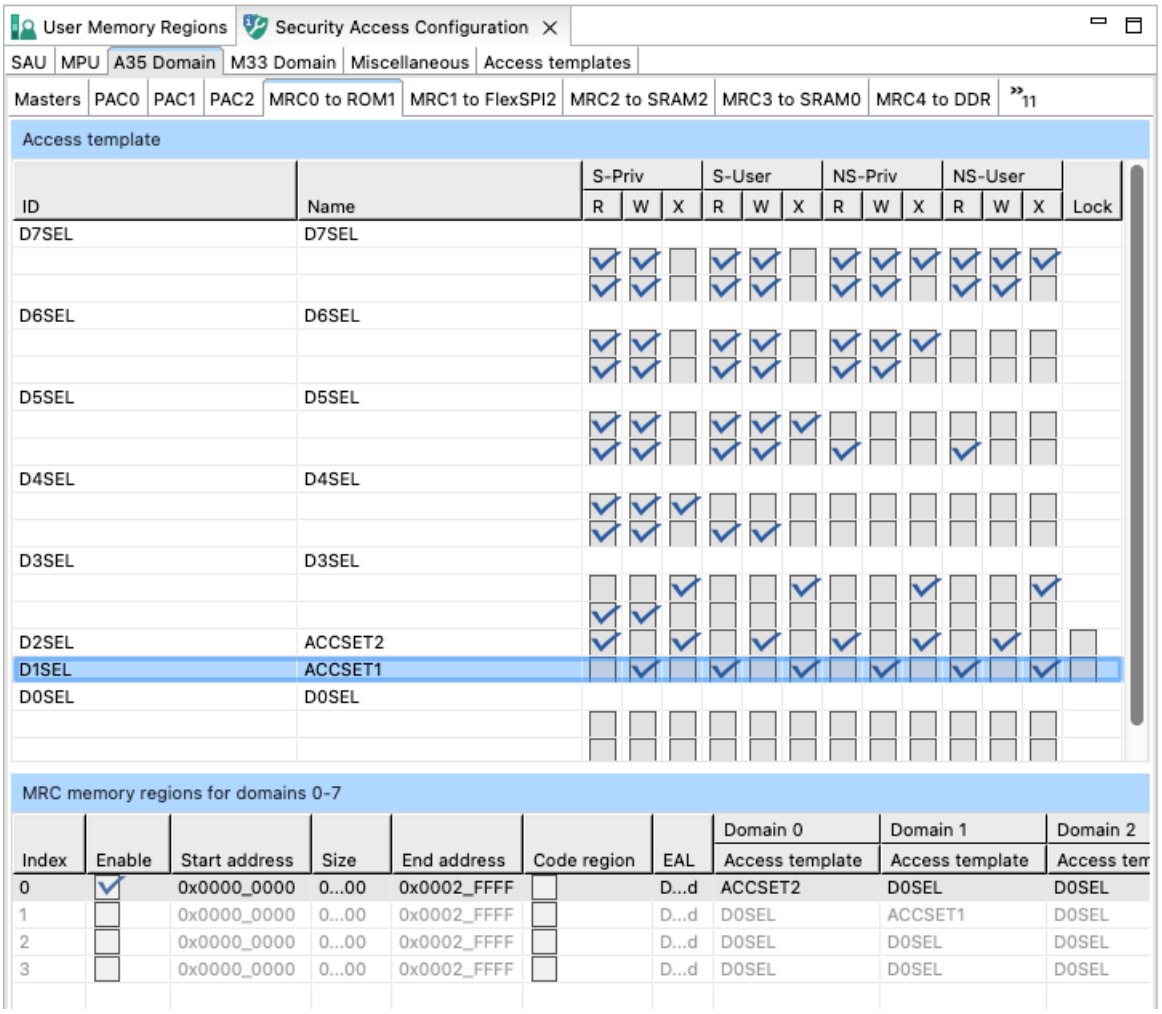


Figure 143. XtRDC MRC tab

5.2.2.3.3 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.

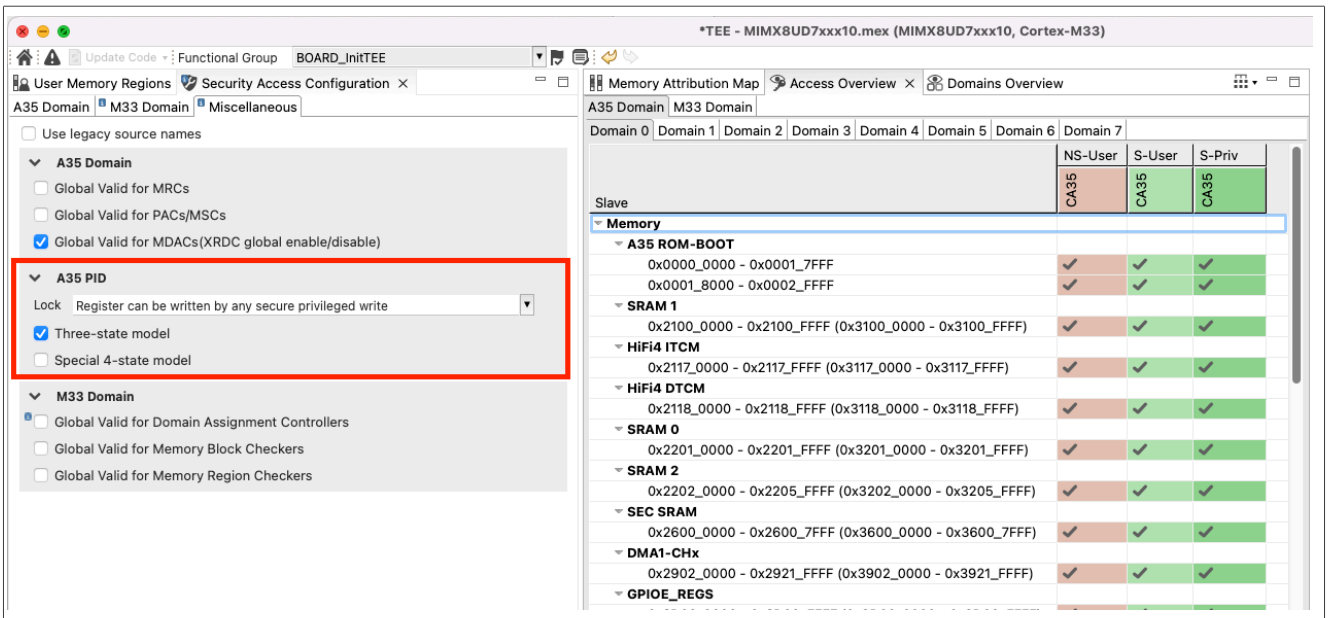


Figure 144. Access modes

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

5.2.2.4.1 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.

5.2.2.4.2 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.

5.2.2.4.3 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.

5.2.2.4.4 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.

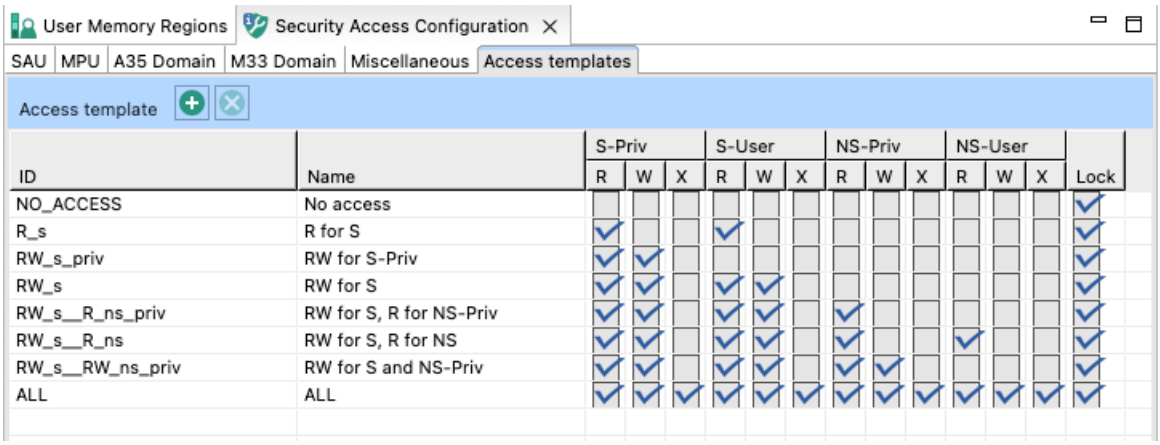


Figure 145. Access templates

5.2.2.4.5 MRC

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

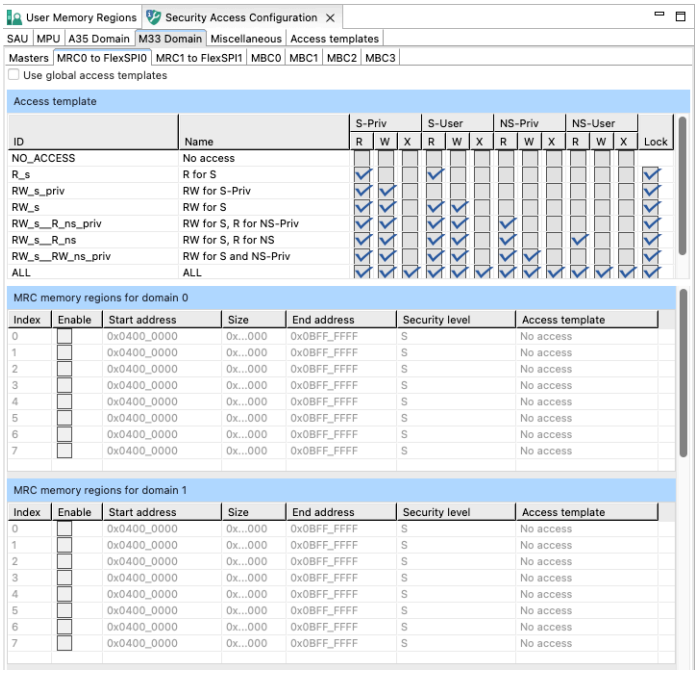


Figure 146. MRC

5.2.2.4.6 MBC

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

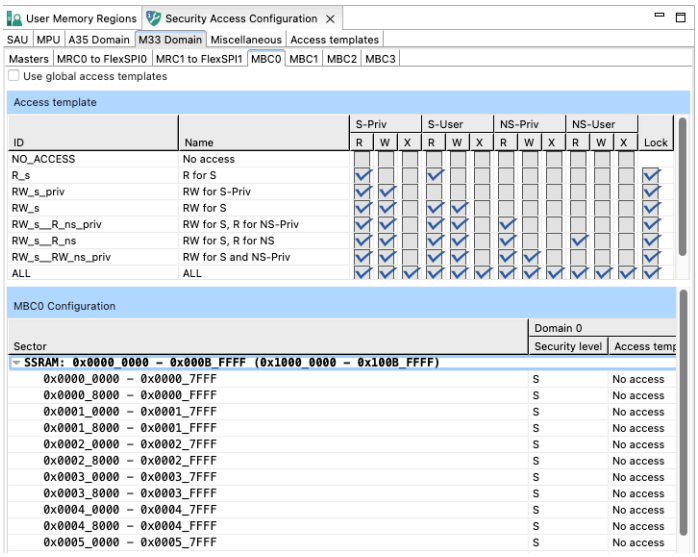


Figure 147. MBC

5.2.2.5 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.

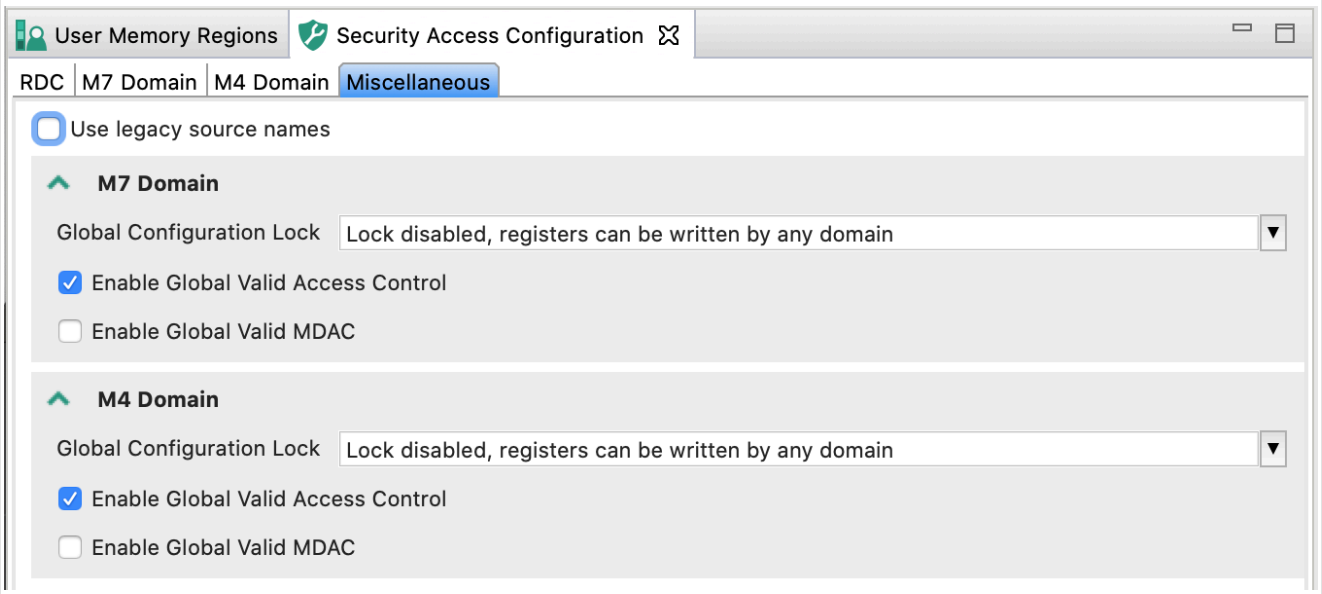


Figure 148. Miscellaneous (RDC+XRDC2)

5.2.3 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.

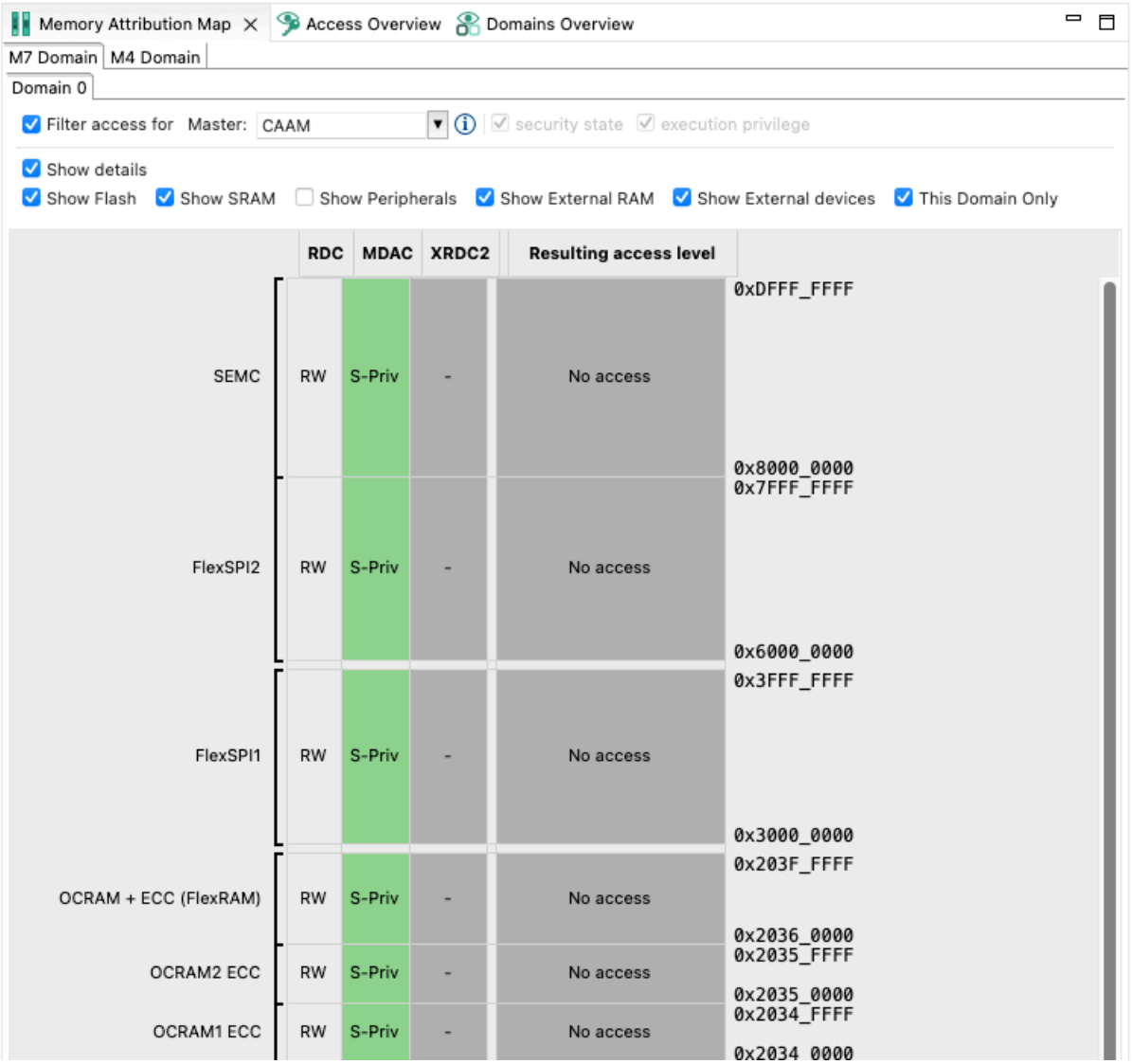


Figure 149. Memory Attribution Map

To set the display options, do the following:

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

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

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

5.2.4 Access Overview

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

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

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

Memory Attribution Map | Access Overview | Domains Overview

M7 Domain | M4 Domain

Domain 0 | Domain 1 | Domain 2

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 150. 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.

5.2.5 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 151. Domain Overview

5.2.6 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 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.

6 Advanced Features

6.1 Switching the processor

You can switch the processor or the package of the current configuration to a different one. However, switching to a completely different processor may lead to various issues, such as inaccessible pin routing or unsatisfiable clock-output frequency. In that case, it's necessary to fix the problem manually. For example, go to the **Routing Details** view and reconfigure all pins which 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**.

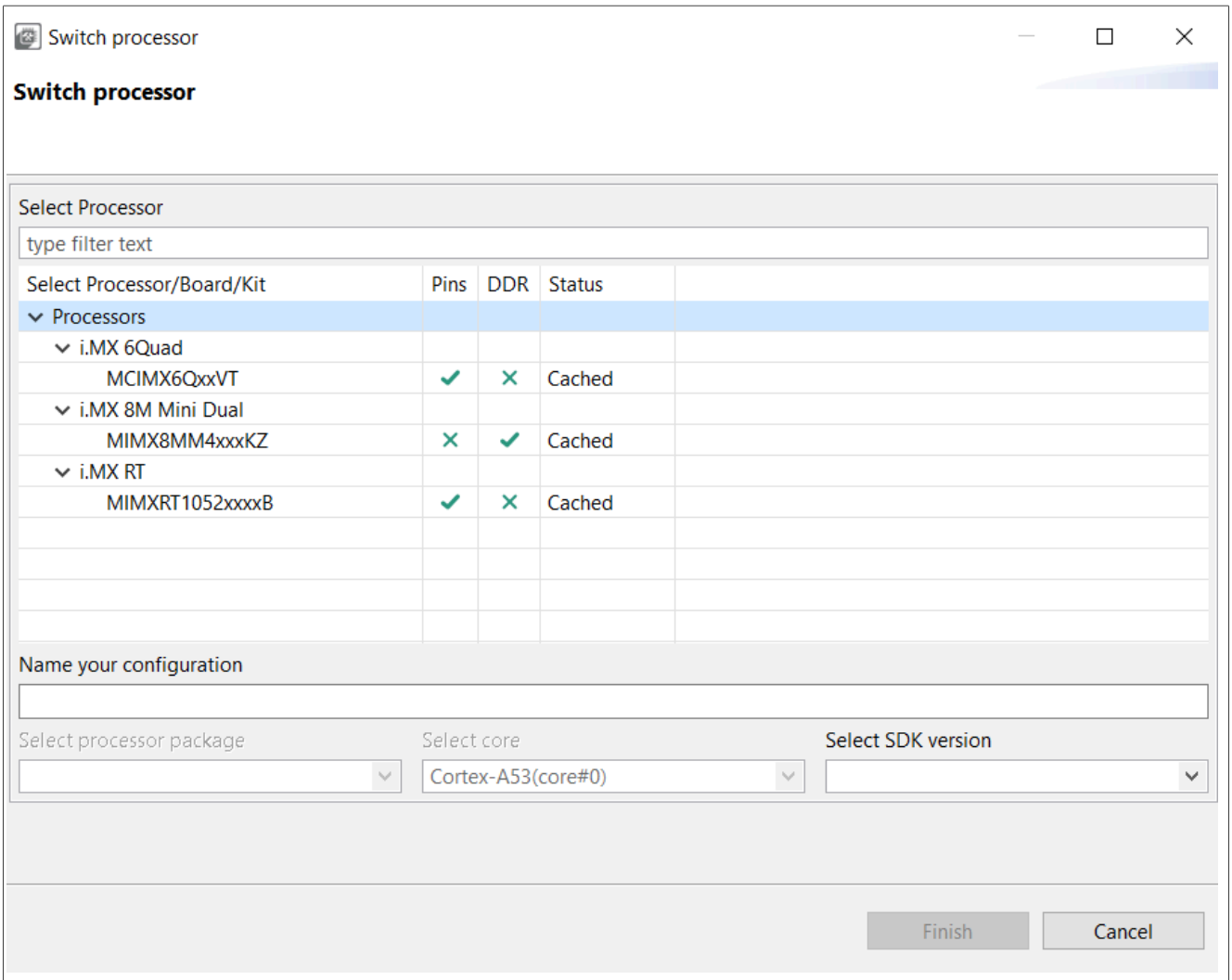


Figure 152. Switch processor

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

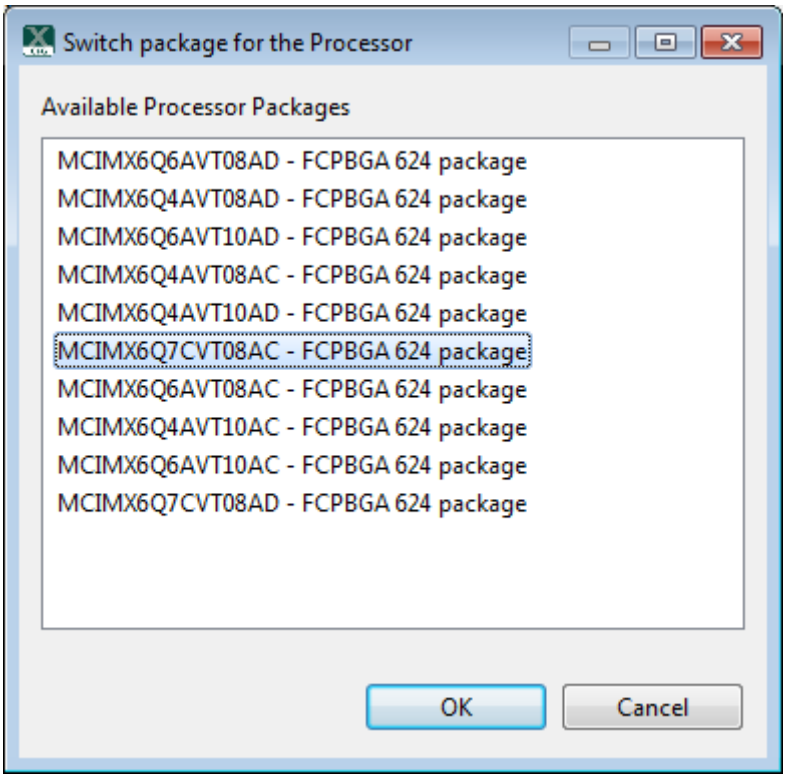


Figure 153. Switch package

6.2 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 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 154. 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 155. Aligning to block

6.3 Tools advanced configuration

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

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

6.4 Generating HTML reports

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

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

6.5 Exporting sources

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

To launch the Export wizard:

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

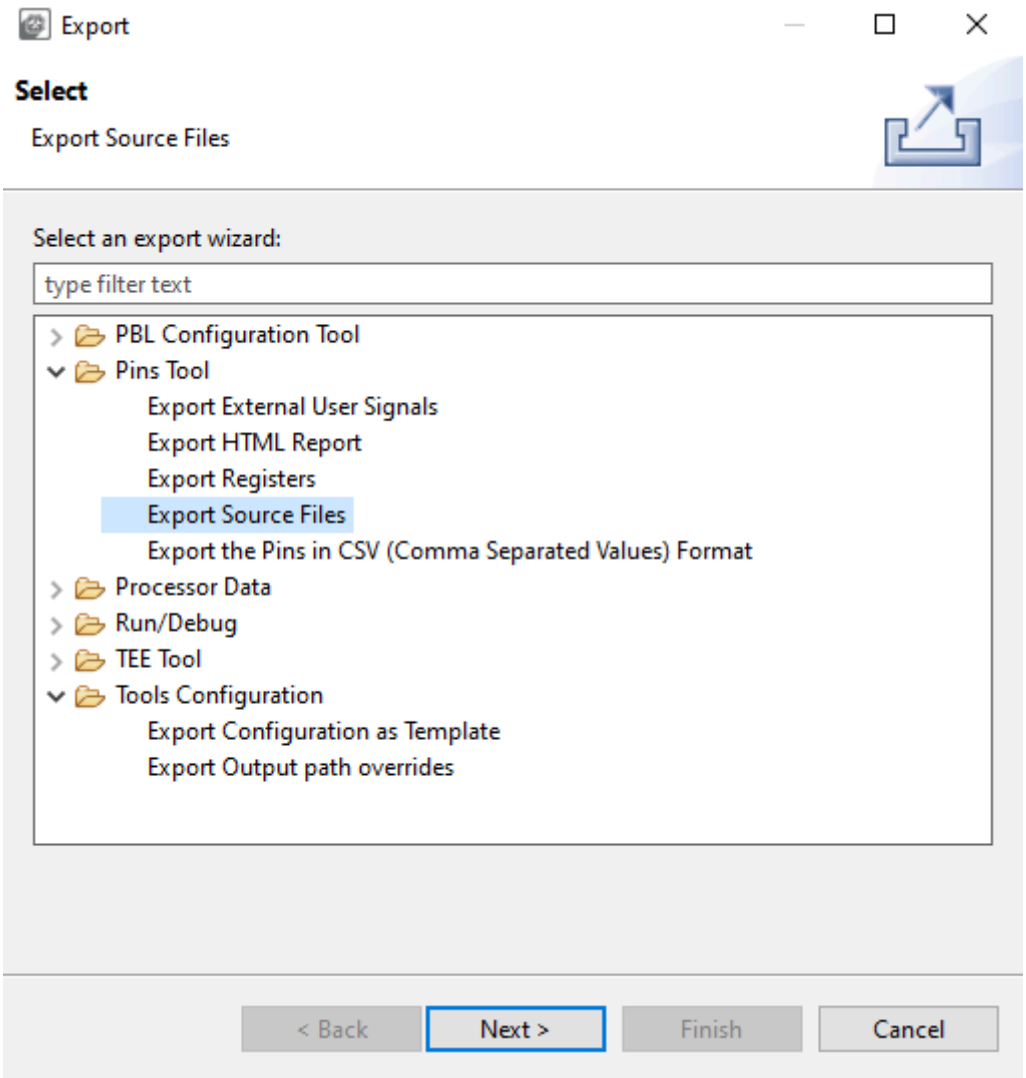


Figure 156. Export wizard

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

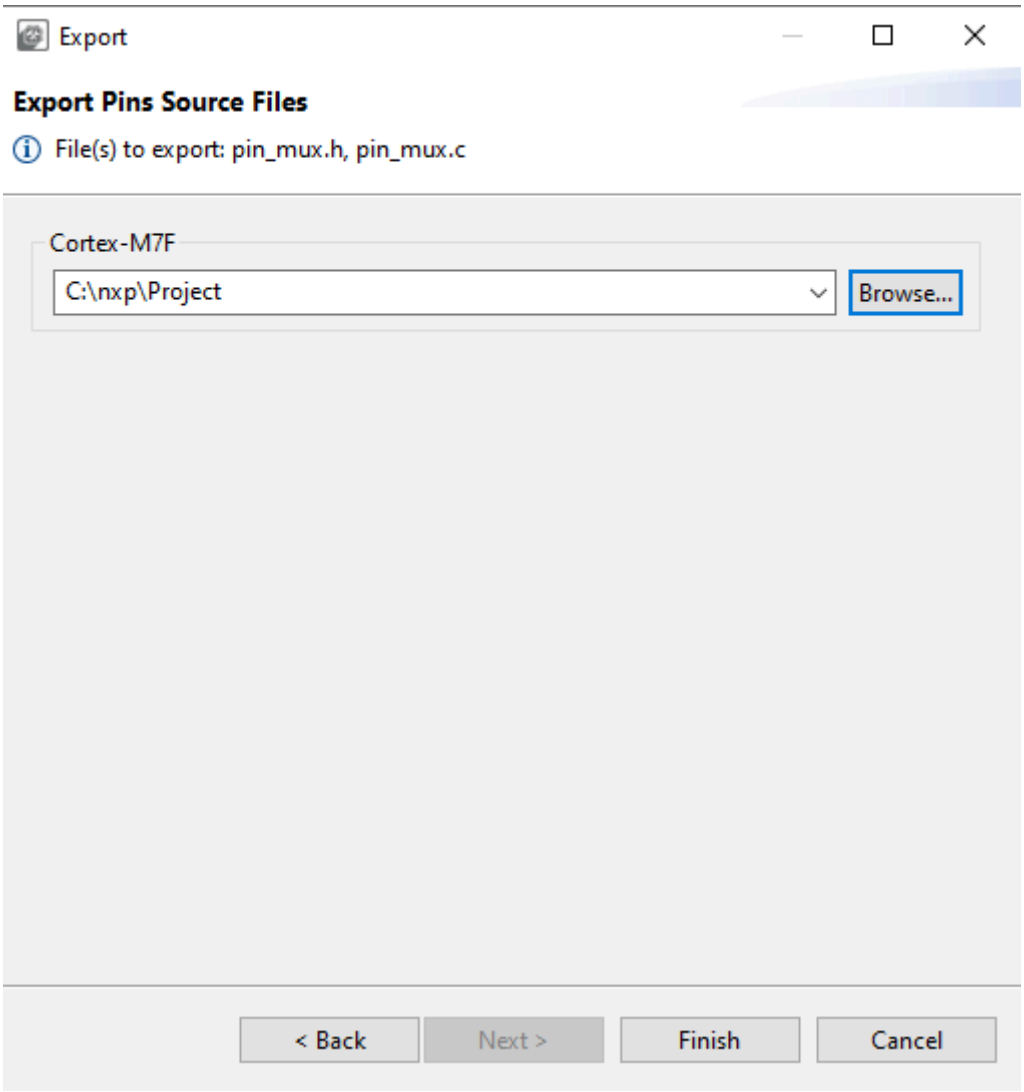


Figure 157. Export sources – Select target folder

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

6.6 Exporting registers

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

To export registers, follow these steps:

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

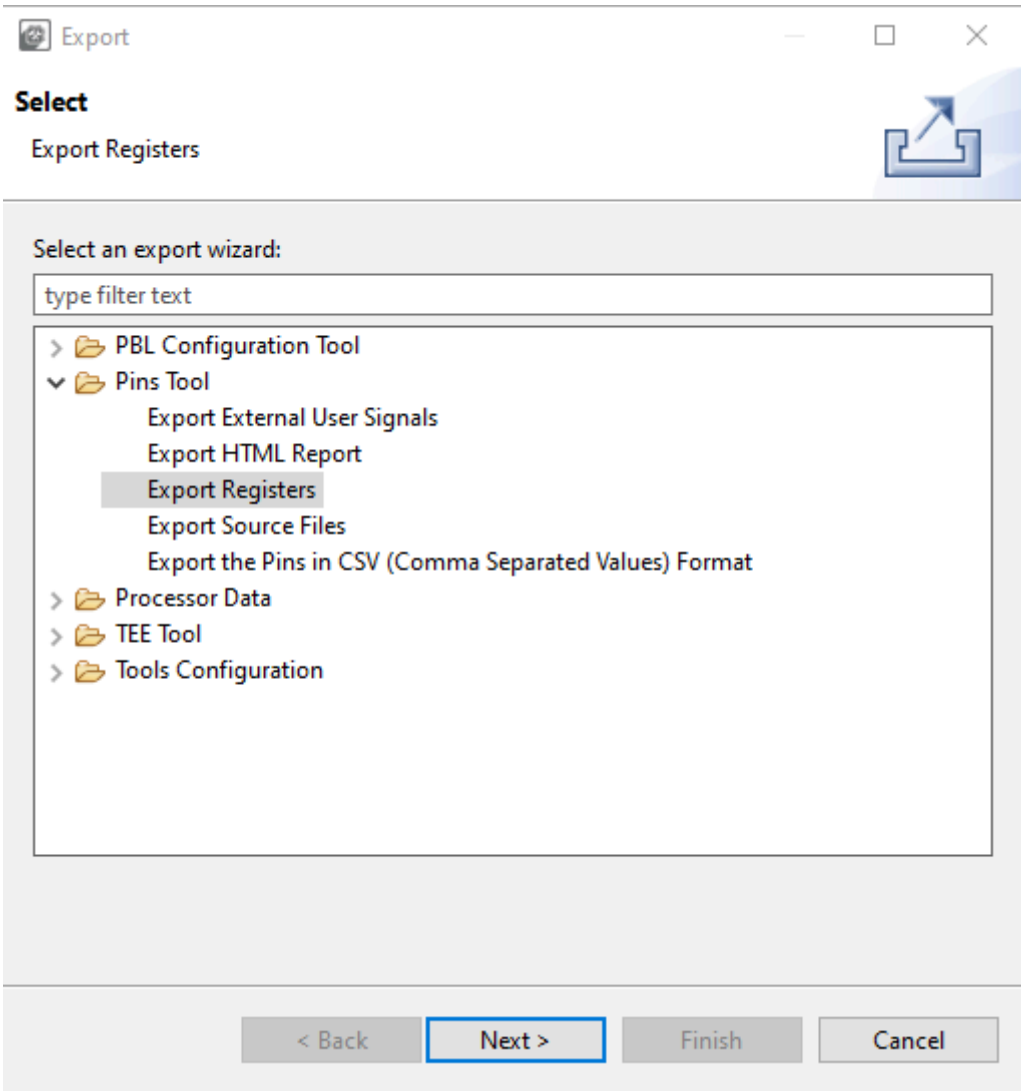


Figure 158. Export registers

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

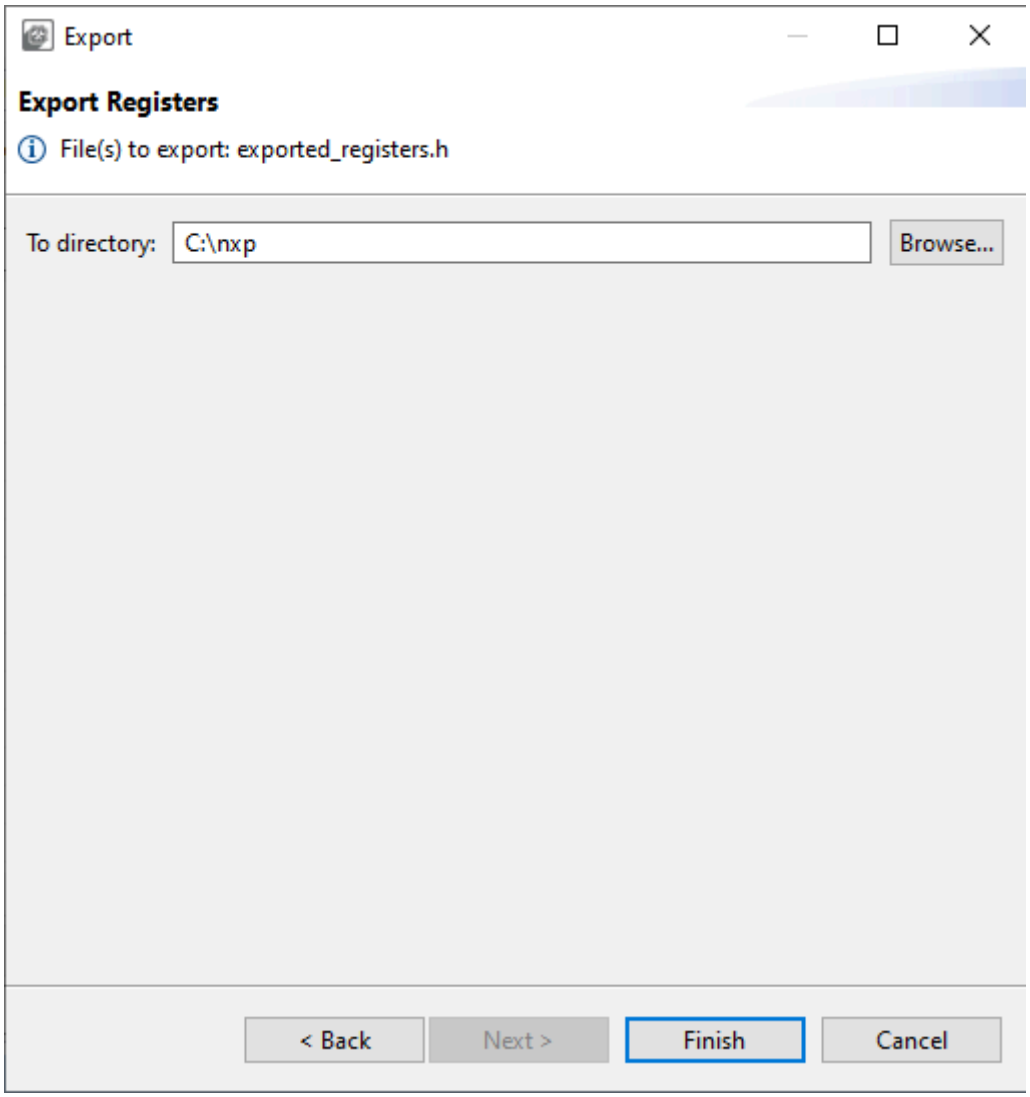


Figure 159. Export registers directory

5. Click **Finish**.

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

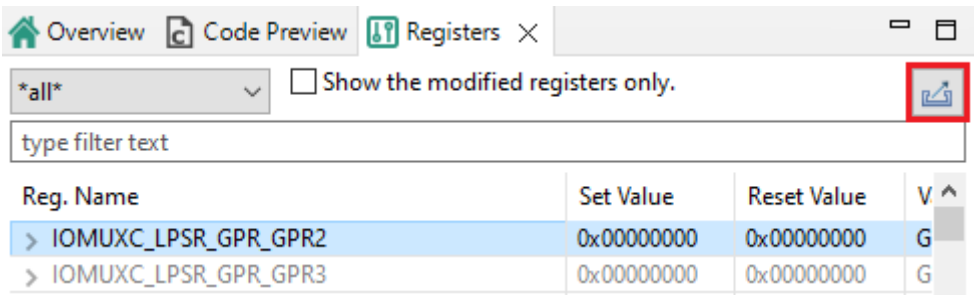


Figure 160. Export registers to CSV

6.7 Managing data and working offline

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

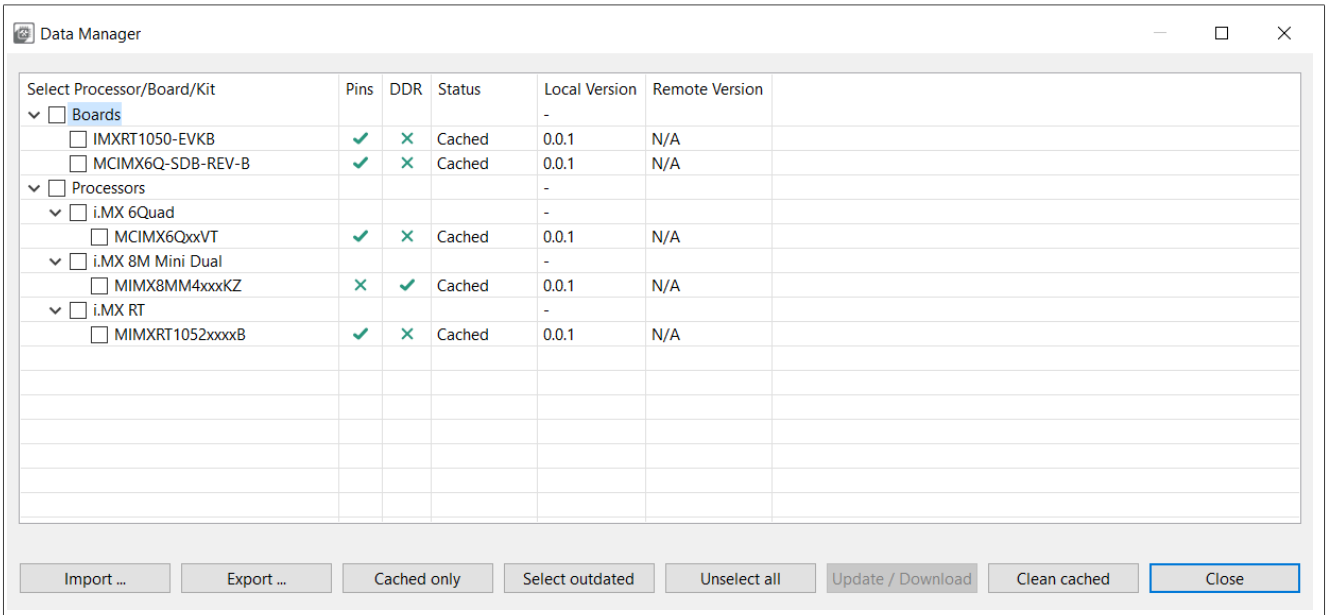


Figure 161. Data Manager

6.7.1 Working offline

You can create a new configuration even without access to the Internet by working with cached processor data. To do so you must 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**.

6.7.2 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.

6.7.3 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**,
   Data is now saved on your local computer in a ZIP format. You can physically (for example, with a USB stick) move it to an offline computer.
   Note: *You can also export downloaded data by selecting **File** > **Export** > **Processor Data** > **Export Processor Data** from the **Menu bar**.*

6.7.4 Importing data

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

To import data, do the following:

1. 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**.*

6.7.5 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 will be updated automatically or after a prompt.*

Note: *Internet connection is required for data update.*

To update cached data, do the following:

1. In **Menu bar**, select **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.*

6.8 Output path overrides

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

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

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

- Enabled – defines whether the rule will be used by the applied path or skipped.
- Description – used as a user-friendly description of the rule
- Regular expression – matches the overriding parts in the whole output path. The format is taken from the Java regular expression.
- Replacement expression – used as a replacement of all matches in the path. Substring groups can be referenced by using placeholder $1, $2 and so on.

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

Example content of the output path override yaml file:

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

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

```
outputPathOverrides: [
]
```

6.9 Import pins configuration from legacy tools project

The Pins tool from **Config Tools for i.MX** helps in importing pins configuration from the existing legacy tools 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 the need of manual correction of the imported pins configuration.

6.9.1 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.

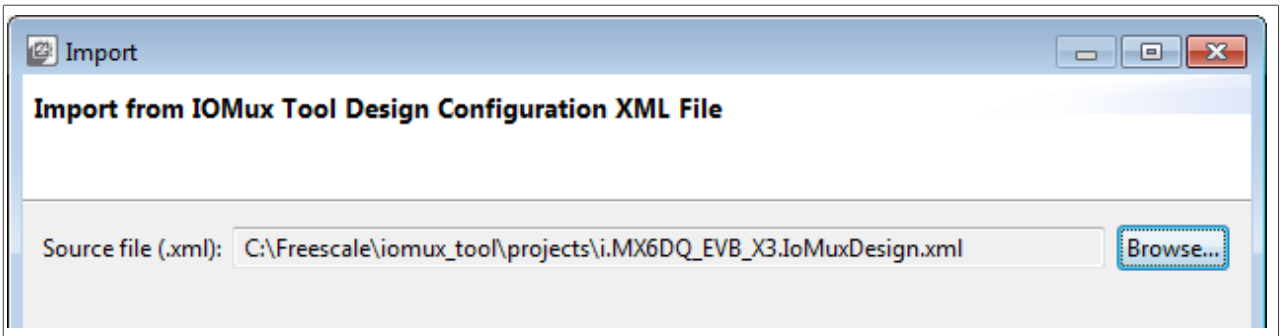


Figure 162. Import from IOMux Tool Design Configuration XML File

5. 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.

6.9.2 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.

The **Import** wizard is used 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.

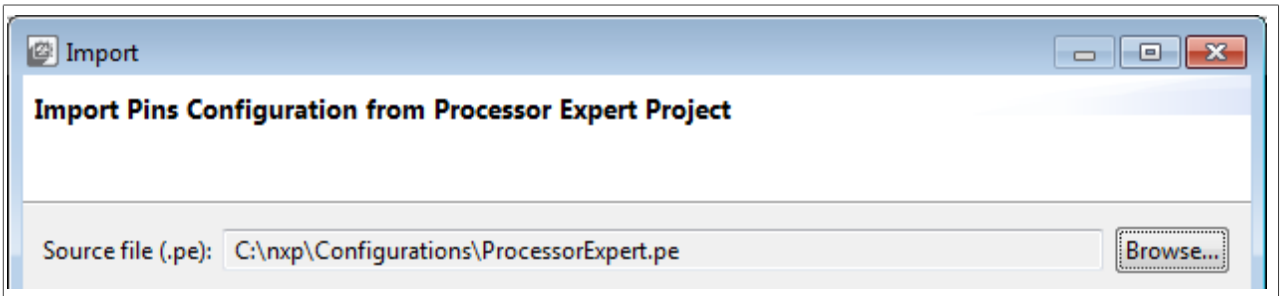


Figure 163. Import Pins Configuration from Processor Expert Project

5. 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.

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

Command example:

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

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 you job:
  - Use the first simple command with `-updateData`, which updates possibly outdated data.
  - Use all other commands in a batch run without this parameter:

```
copy /Y eclipsec.exe toolsc.exe
@rem updates all local data if newer exists
tools.exe -updateData -consoleLog -HeadlessTool Pins
@rem now runs tools many times
tools.exe -consoleLog -HeadlessTool Pins -Load some.mex -ExportAll c:/directory
tools.exe -consoleLog -HeadlessTool Pins -Load other.mex -ExportAll c:/
other_directory
@rem and so on.
```

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

Table 18. 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 cached Only 'zh' and 'en' are supported	-nl zh
Show console	`-consoleLog`	Logs output is also sent to Java's System.out (typically back to the command shell if any)	None	
Empty configuration	`-EmptyConfig`	Ensures creating a new empty configuration without any content	Requires the `-Headless Tool` command	-HeadlessTool Pins -EmptyConfig
Select MCU	`-MCU`	MCU to be selected by framework Changes the processor in the result configuration of the previous chain	Requires the `-Headless Tool` and `-SDKversion` commands.	-MCU MK64 FX512xxx12 -SDKversion ksdk2_0 -HeadlessTool Pins -Load `C:/conf/conf.mex` -SDKVersion ksdk2_0 -MCU MK64FN1 M0xxx12 -ExportMEX `C:/conf/MK64.mex`
Select core	`-Core`	Select core for the configuration		-Core core0
Select Board	`-Board`	Board to be selected by framework (MCU is automatically selected too)(since 6.0)	Requires the `-SDKversion` command	-Board FRDM-K22F -SDKversion ksdk2_0
Select Kit	`-Kit`	Kit to be selected by framework (MCU and board is automatically selected too)(since 6.0)	Requires the `-SDKversion` command	-Kit FRDM-K22F-AGM01 -SDKversion ksdk2_0
Select SDK version	`-SDKversion`	Version of the MCU to be selected by framework	Requires the `-MCU` command	-MCU MK64 FX512xxx12 -SDKversion ksdk2_0
Select part number	`-PartNum`	Selects specific package of the MCU Changes package in result configuration of	Requires the `-MCU` and `-SDKversion` commands	-MCU MK64 FX512xxx12 -SDKversion ksdk2_0 -PartNum MK64 FX512VLL12

Table 18. Commands supported in the framework...*continued*

Command name	Definition and parameters	Description	Restriction	Example
		previous chain, see the command about "-MCU"		
Configuration name	-ConfigName	Name of newly created configuration - used in export Rename the configuration	Requires the -Headless Tool command	-MCU MK64 FX512xxx12 - SDKversion ksdk2_0 -ConfigName "My Config" -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 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 run The argument is expected to be a folder or path to specify the .mex file	Requires the -Headless Tool command	-MCU xxx - SDKversion xxx -ExportMEX C:/exports/my_config_folder -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 export Includes -ExportSrc and in framework -ExportMEX The Argument is expected to be a folder name.	Requires the -Headless Tool command	-HeadlessTool Pins -ExportAll C:/exports/generated
Create a new configuration by importing a toolchain project	-ImportProject {path}	Creates a new configuration by importing toolchain project The parameter is a path to the root of the toolchain project.	Requires the -Headless Tool command	-HeadlessTool Pins -ImportProject c:\test\myproject
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	Requires the -Headless Tool command see -ImportProject	-HeadlessTool Prj Cloner -CreateFrom Project c:\test\myproject -HeadlessTool Peripherals -Create

Table 18. Commands supported in the framework...*continued*

Command name	Definition and parameters	Description	Restriction	Example
		(mex and update code), validates configuration. The parameter is a path to the root of the toolchain project.		FromProject `c:\test\myprojectsee` -ImportProject
Generate source files with custom copyright	`-Custom Copyright`	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	-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. Empty list of rules removes the set rules.	Requires the `-Headless Tool` command	-OutputPathOverrides `c:\test\output PathOverride Rules.yaml`
Batch processing	`-BatchFile`	Path to the file with commands, that will be run on defined paths. For details, see Section 6.11	Requires the `-Headless Tool` command; intent without other commands	-HeadlessTool Pins -BatchFile `C:/conf/batchCommands.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 ""
Update locally downloaded data	`-updateData`	Downloads data for already locally downloaded data if they have an update.		-updateData

6.10.1 Command-Line execution - Pins Tool

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

Table 19. 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 -HeadlessTool Pins	-HeadlessTool Pins -Enable
Import C files	`-ImportC`	Imports .c files into configuration	Requires -HeadlessTool Pins	-HeadlessTool Pins

Table 19. Commands supported in Pins...*continued*

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

6.10.2 Command-Line execution - TEE Tool

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

Table 20. 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 -Headless Tool TEE	-HeadlessTool TEE -Enable
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 -ExportMEX The argument is expected to be a folder.	Requires -HeadlessTool TEE	-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 -HeadlessTool TEE	-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 -Headless Tool TEE -Enable	-HeadlessTool TEE - Enable -Export Registers `C:/exports/regs`

6.11 Batch processing

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

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

Example:

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

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

6.11.1 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.*

6.11.1.1 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.*

6.11.1.2 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
```

6.11.1.3 Steps list

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

Each step is defined as a structure:

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

  - ${fileName} - for filename without path
- In case the “folders” list is processed:
  - ${folder} – the folder path without separator at the end

See the section Section 6.10 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.*

7 Support

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

8 Note about the source code in the document

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

Copyright 2024 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.

9 Revision history

Table 21. Revision history

Document ID	Release date	Description
IMXUG v.7	10 January 2024	Updated for v.15
IMXUG v.6	31 July 2023	Section 6.11 is added.
IMXUG v.5	2 January 2023	Section 4 is updated.
IMXUG v.4	20 September 2022	Updated for v.12.1
IMXUG v.3	30 June 2022	Updated for v.12

Table 21. Revision history*...continued*

Document ID	Release date	Description
IMXUG v.2	22 December 2021	New features are added, screenshots are updated.
IMXUG v.1	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.

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.

Amazon Web Services, AWS, the Powered by AWS logo, and FreeRTOS — are trademarks of Amazon.com, Inc. or its affiliates.

Contents

Date of release: 10 January 2024
Document identifier: IMXUG