NXP Semiconductors
User's Guide

Document identifier: MCUXIDECTUG
Rev. 0, 1/2021

MCUXpresso Config Tools User's Guide (IDE)

Contents

Chapter 1 Introduction 5
1.1 Versions 5

Chapter 2 User Interface 7
2.1 Creating, saving, and opening a configuration 7
2.1.1 Creating a new configuration 7
2.1.2 Saving a configuration 7
2.1.3 Importing sources 7
2.1.3.1 Importing configuration 8
2.1.3.2 Importing registers 9
2.1.4 Restoring configuration from source code 11
2.2 Toolbar 11
2.2.1 Eclipse project selection 12
2.2.2 Config Tools Overview 12
2.2.3 Show Problems View 12
2.2.4 Update code 12
2.2.5 Functional groups 15
2.2.5.1 Functional group properties 15
2.2.6 Undo/Redo actions 16
2.2.7 Selecting the tools 16
2.3 Status bar 16
2.4 Preferences 17
2.5 Configuration preferences 18
2.6 Problems view 19
2.7 Registers view 20
2.8 Log view 22
2.9 Config tools overview 22

Chapter 3 Pins Tool 24
3.1 Pins routing principle 24
3.1.1 Beginning with peripheral selection 24
3.1.2 Beginning with pin/internal signal selection 25
3.1.3 Routing of peripheral signals 25
3.2 Example workflow 30
3.3 User interface 33
3.3.1 Pins view 34
3.3.2 Package view 35
3.3.3 Peripheral Signals view 37
3.3.3.1 Filtering in the Pins and Peripheral Signals views 39
3.3.4 Routing Details view 40
3.3.4.1 Labels and identifiers 41
3.3.5 Expansion Header 42
3.3.5.1 Expansion Board 46
3.3.6 Highlighting and color coding 48
3.4 Errors and warnings 50
3.4.1 Incomplete routing 50
3.5 Code generation 51
3.6 Using pins definitions in code 52

Chapter 4 Clocks Tool 54
4.1 Features 54
4.2 User interface 54
4.3 Clock configuration 55
4.4 Global settings 55
4.5 Clock sources 55
4.6 Setting states and markers 56
4.7 Frequency settings 56
4.7.1 Pop-up menu commands 57
4.7.2 Frequency precision 57
4.8 Dependency arrows 57
4.9 Details view 58
4.10 Clocks diagram 59
4.10.1 Mouse actions in diagram 59
4.10.2 Color and line styles 60
4.10.3 Clock model structure 61
4.11 Clocks menu 62
4.12 Troubleshooting problems 62
4.13 Code generation 63
4.13.1 Working with the code 64
4.14 Clock Consumers view 64

Chapter 5 Peripherals Tool 66
5.1 Features 66
5.2 Basic terms and definitions 66
5.3 Workflow 66
5.4 User interface 67
5.4.1 Toolbar (Peripherals) 68
5.4.1.1 Object Missing 68
5.4.2 Components view 68
5.4.3 Peripherals view 71
5.4.4 Settings Editor 71
5.4.4.1 Quick selections 72
5.4.4.2 Settings 72
5.4.4.3 Settings Editor header 74
5.4.5 Documentation view 75
5.5 SEMC Validation tool 76
5.5.1 Validation view 76
5.6 Problems 77
5.7 Code generation 77

Chapter 6 Device Configuration Tool 80
6.1 Device Configuration Data (DCD) view 80
6.1.1 Device Configuration Data (DCD) view actions 80
6.2 Code generation 82

Chapter 7 Trusted Execution Environment Tool 83
7.1 AHB with security extension-enabled devices 84
7.1.1 User Memory Regions view 84
7.1.2 Security Access Configuration view 88
7.1.2.1 SAU 88
7.1.2.2 Interrupts 89

7.1.2.3 Secure/Non-secure MPU....90
7.1.2.4 MPC....92
7.1.2.5 Masters/Slaves....93
7.1.2.6 Pins....95
7.1.2.7 Miscellaneous....97
7.1.3 Memory attribution map....97
7.1.3.1 Core 0....97
7.1.3.2 Other masters....98
7.1.4 Access Overview....100
7.1.5 Code generation....101
7.2 RDC-enabled devices....102
7.2.1 User Memory Regions view....102
7.2.1.1 Access templates....102
7.2.2 Security Access Configuration view....103
7.2.2.1 RDC....103
7.2.2.1.1 RDC Masters....103
7.2.2.1.2 Memory Regions....105
7.2.2.1.3 Peripherals....106
7.2.2.2 XRDC2 Domains view....107
7.2.2.2.1 MPU....107
7.2.2.2.2 Domains....109
7.2.2.2.3 Masters....109
7.2.2.2.4 Peripherals....111
7.2.2.2.5 Memory Regions....113
7.2.2.2.6 Memory Slots....114
7.2.2.3 Miscellaneous....115
7.2.3 Memory Attribution Map....116
7.2.4 Access Overview....118
7.2.5 Domains Overview....119
7.2.6 Code generation....120

Chapter 8 Advanced Features....122
8.1 Switching the processor....122
8.2 Exporting the Pins table....123
8.3 Tools advanced configuration....124
8.4 Generating HTML reports....124
8.5 Exporting sources....124
8.6 Exporting registers....126
8.7 Command line execution....126
8.7.1 Command line execution - Pins tool....128
8.7.2 Command line execution - Clocks Tool....129
8.7.3 Command line execution - Peripherals tool....130
8.7.4 Command line execution - Project Cloner....131
8.8 Managing data and working offline....132
8.8.1 Working offline....132
8.8.2 Downloading data....132
8.8.3 Exporting data....133
8.8.4 Importing data....133
8.8.5 Updating data....133

Chapter 9 Support....135

Chapter 1
Introduction

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

Table 1. MCUXpresso Config Tools

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

1.1 Versions

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

NOTE

The desktop version of the tool contacts the NXP server and fetches the list of the available processors. Once used, the processors data is retrieved on demand.

TIP

To use the desktop tool in the offline mode, create a configuration for the given processor while online. The tool will then store the processors locally in the user folder and enable faster access and offline use. Otherwise, it is possible to download and export the data using the **Export** menu.

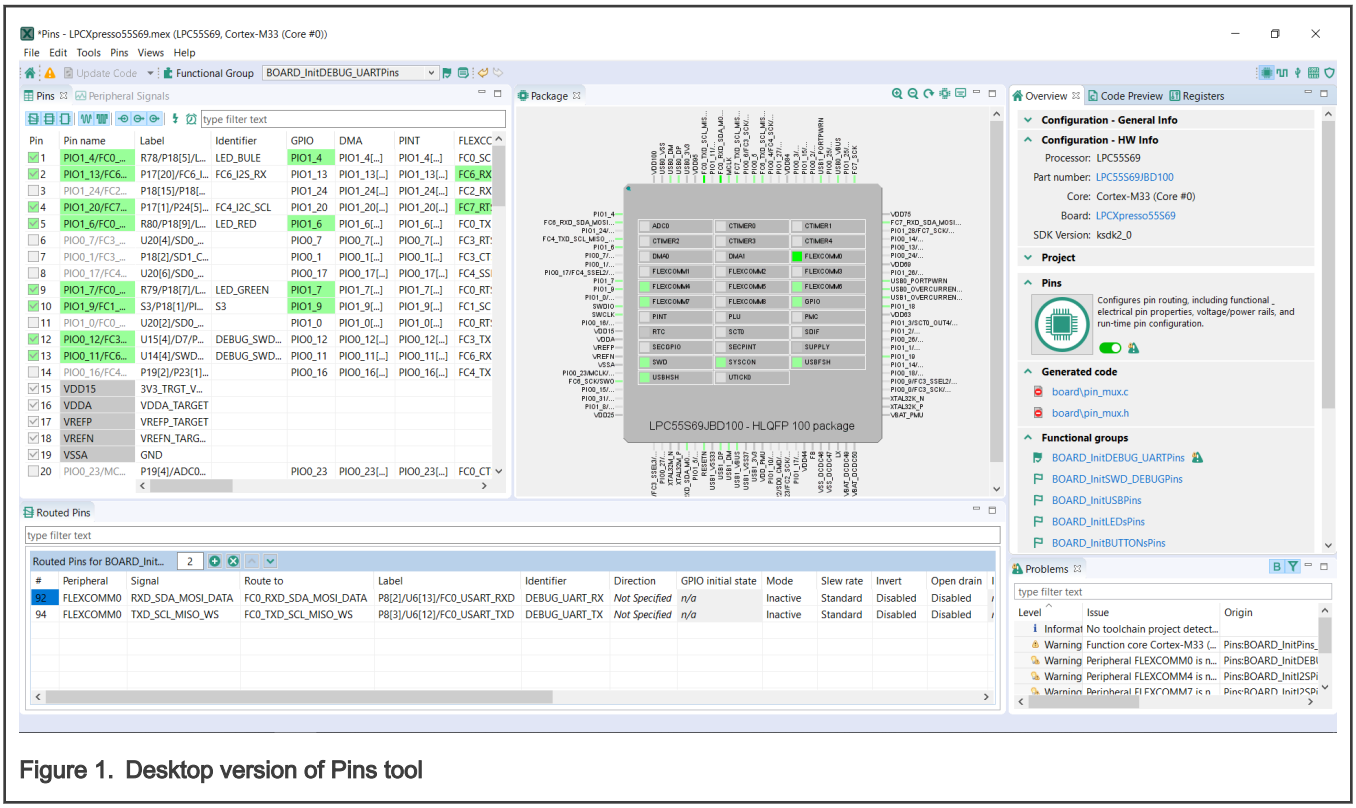


Figure 1. Desktop version of Pins tool

Chapter 2
User Interface

2.1 Creating, saving, and opening a configuration

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

The folder with the saved MEX file must contain exactly one project file to be able to parse the toolchain project. The file type depends on the toolchain of the project and can be one of the following:

Table 2. Supported toolchain project files

Toolchain	Project file
IAR EW	EWP
MDK µVision	UVPROJX
ARM GCC	CMakeLists.txt

2.1.1 Creating a new configuration

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

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

NOTE

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

2.1.2 Saving a configuration

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

NOTE

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

2.1.3 Importing sources

To import source code files, do the following:

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

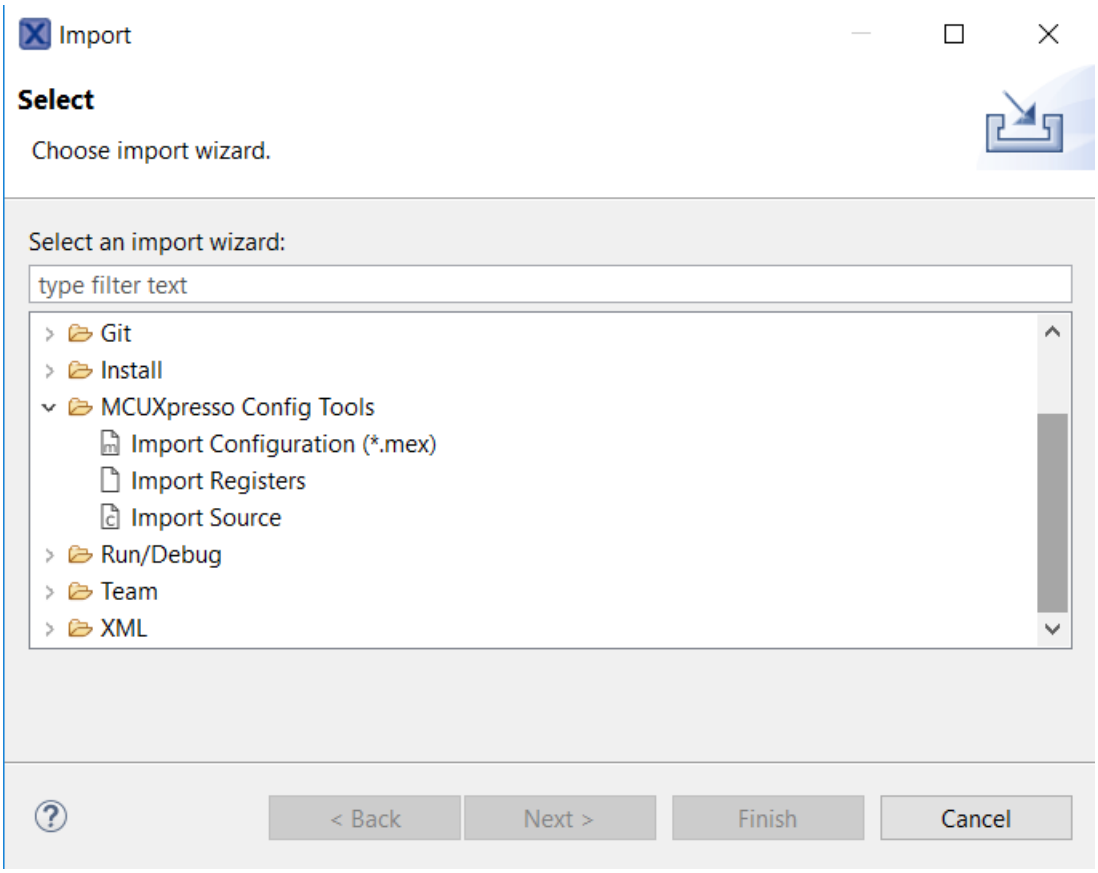


Figure 2. Import Source wizard

3. Click **Next**.
4. On the next page, click **Browse** to specify the location of the source file.
5. Select the source file 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 already exists in the configuration then the imported function is renamed to BOARD_InitPins1.
   - **Overwrite** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, then the existing one is replaced with the imported one.
8. Click **Finish**.

NOTE

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

2.1.3.1 Importing configuration

2.1.3.2 Importing registers

You can import register configuration from a processor memory dump.

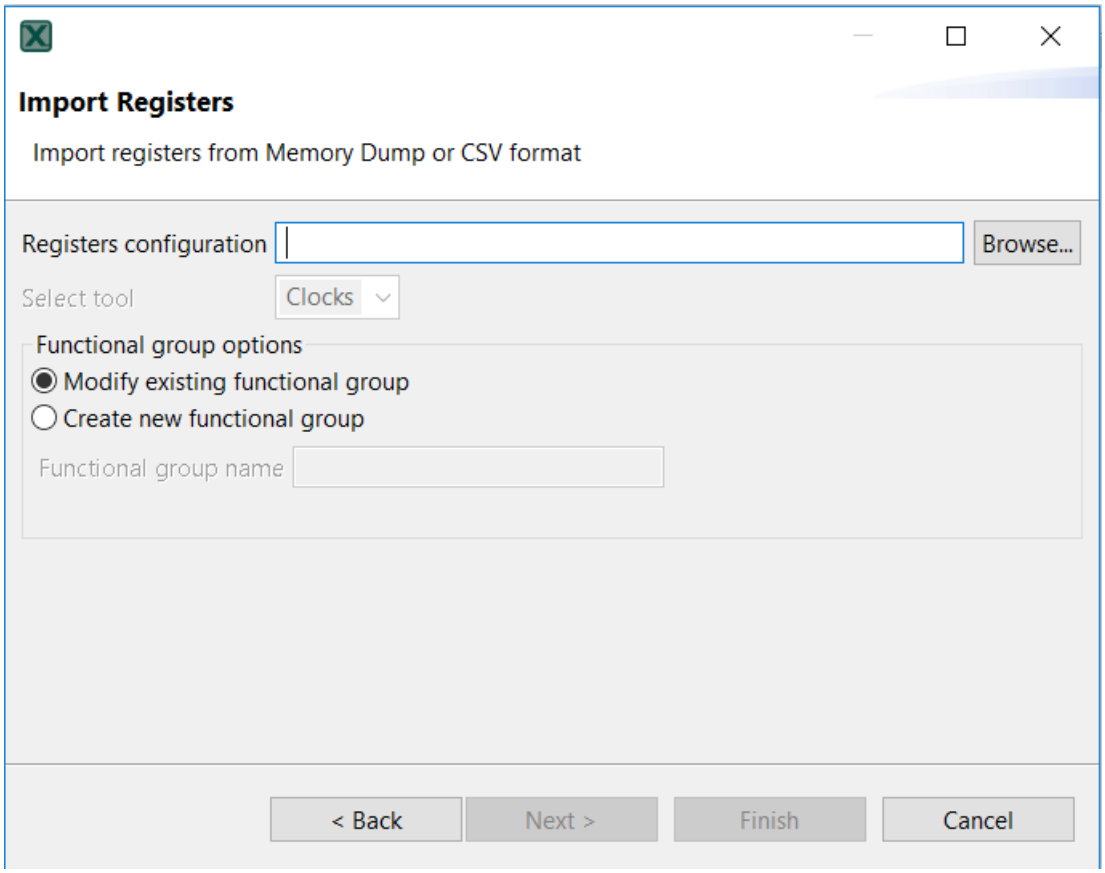
NOTE

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

NOTE

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

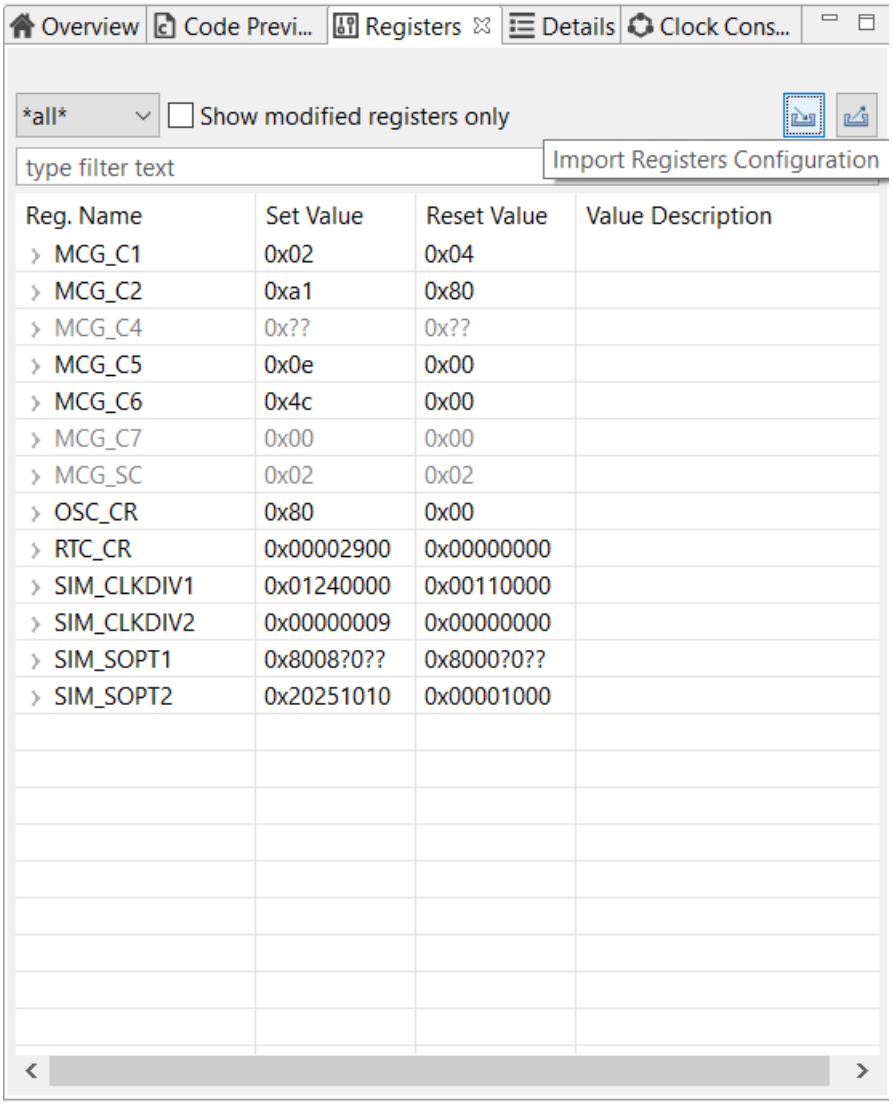
Figure 3. Import Registers



To import register configuration, do the following:

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

Figure 4. Import Registers Configuration



2. In the **Import** wizard, select **MCUXpresso Config Tools > Import Registers**.
3. Click **Next**.
4. On the next page, click **Browse** to specify the location of the registers configuration.
5. Select the registers file you wish to import, and click **OK**.
6. By default, the imported register configuration will overwrite the existing functional group. If you want a new functional group to be created instead, select the **Create new functional group** option button, and specify the functional group name.
7. Click **Finish**.

NOTE

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

2.1.4 Restoring configuration from source code

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

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

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

Figure 5. Setting Information in the source code

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

1. In the **Menu bar**, select **File > Import...**.
2. From the list, select **MCUXpresso Config Tools > Clocks Tool > Import Source Files**.
3. Click **Next**.
4. Click **Browse**.
5. Navigate and select the *clock_config.c* file previously produced by the Clocks tool.
6. If the settings parse successfully, clock configurations are added into the current global configuration.

2.2 Toolbar

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

Table 3. 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.

Table continues on the next page...

Table 3. Toolbar (continued)

Item	Description
Functional group selection	Select functional group. Functional group in the Peripherals tool represents a group of peripherals that are initialized as a group. The tool generates a C function for each function group that contains the initialization code.
Call from default initialization	Set the current functional group to be initialized by the default initialization function.
Functional group properties	Open the **Functional group properties** dialog to modify name and other properties of the function group.
Tool selection	Display icons of individual tools. Use them to switch between tools.
Undo/Redo	Undo/Redo last action.

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

2.2.1 Eclipse project selection

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

2.2.2 Config Tools Overview

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

2.2.3 Show Problems View

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

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

2.2.4 Update code

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

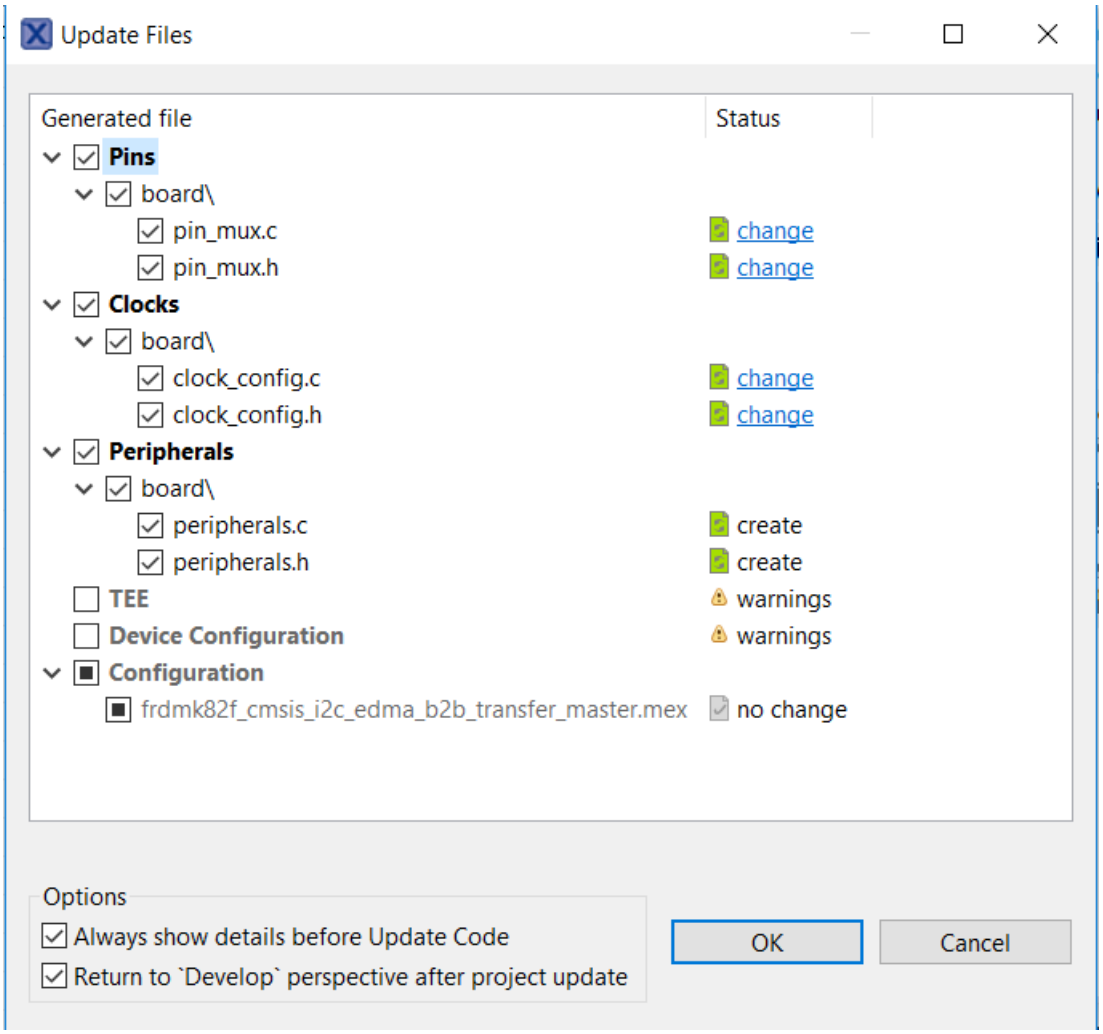


Figure 6. Update Files window

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

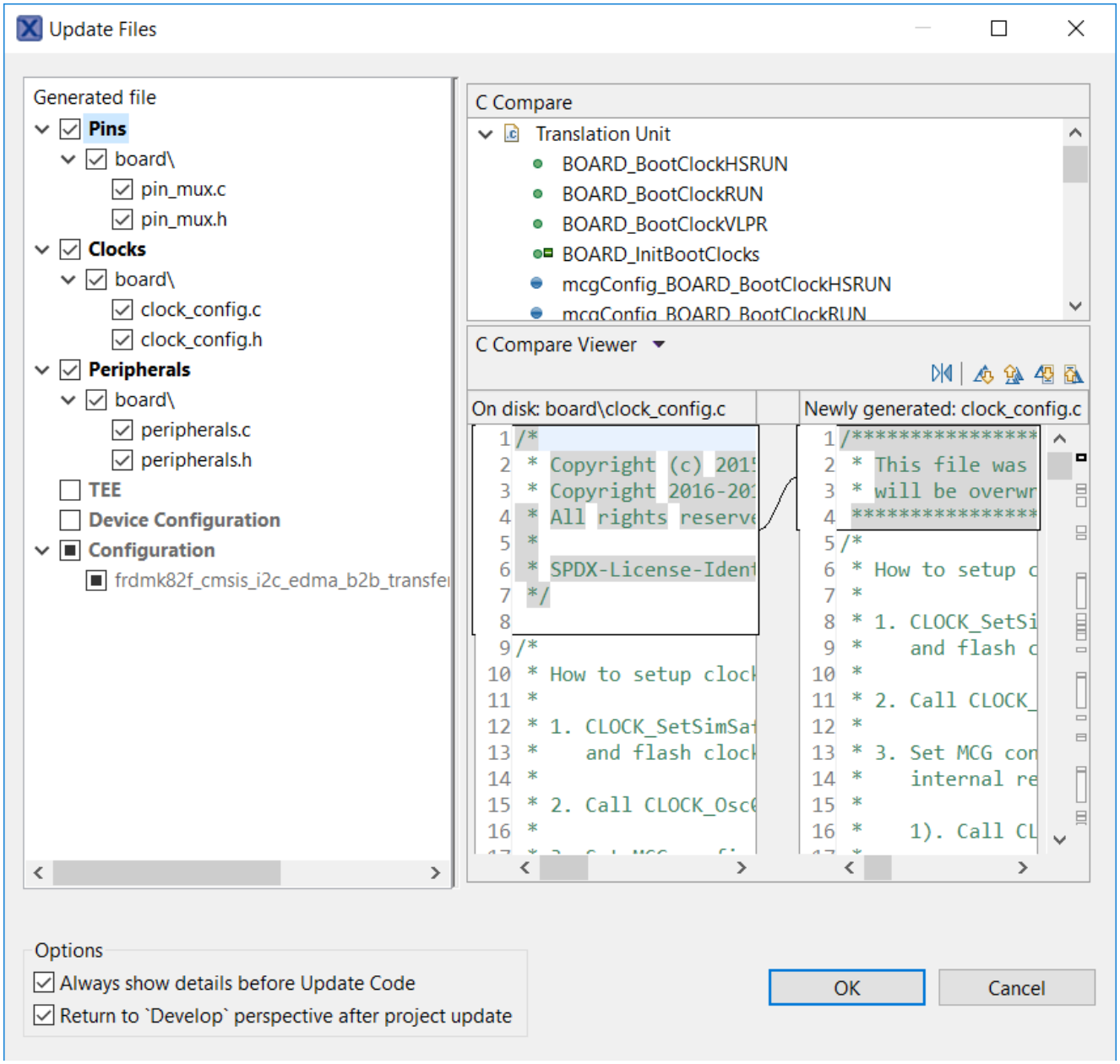


Figure 7. Show differences

To update the project without opening the **Update Files** 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**.

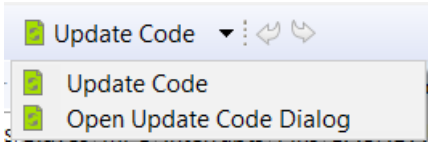


Figure 8. Update Code dropdown menu

NOTE

The generated code is always overwritten.

NOTE

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

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

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

2.2.5 Functional groups

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

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

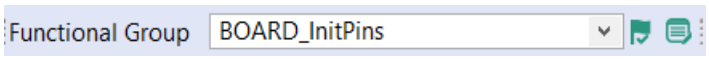


Figure 9. Functional groups

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

Table 4. Functional Groups

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

NOTE

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

2.2.5.1 Functional group properties

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

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

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

NOTE

Not all processors support this option.

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

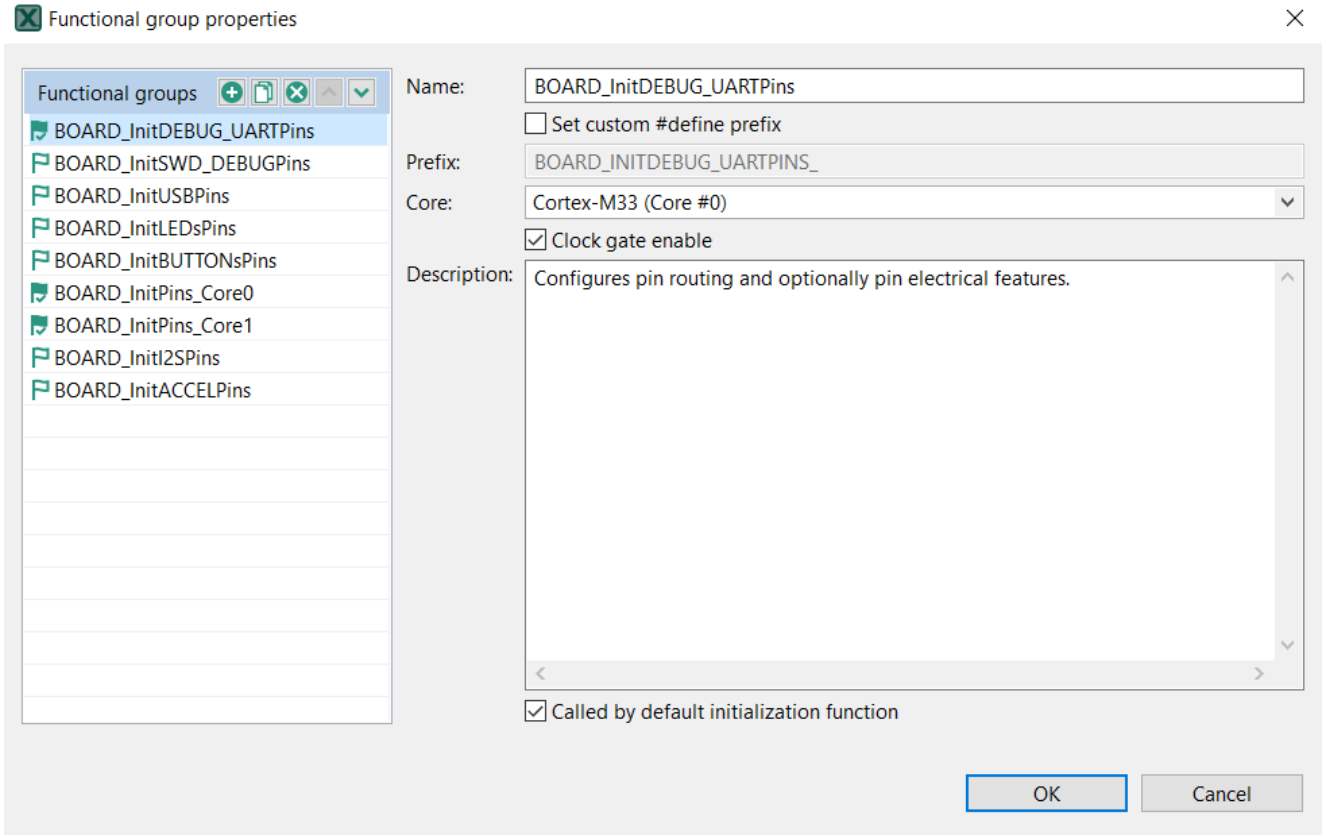


Figure 10. Functional group properties for the Pins tool

2.2.6 Undo/Redo actions

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

Table 5. Undo/reto actions

Icon	Description
	Cancels the previous action
	Cancels the previous undo action

2.2.7 Selecting the tools

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

2.3 Status bar

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

2.4 Preferences

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

NOTE

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

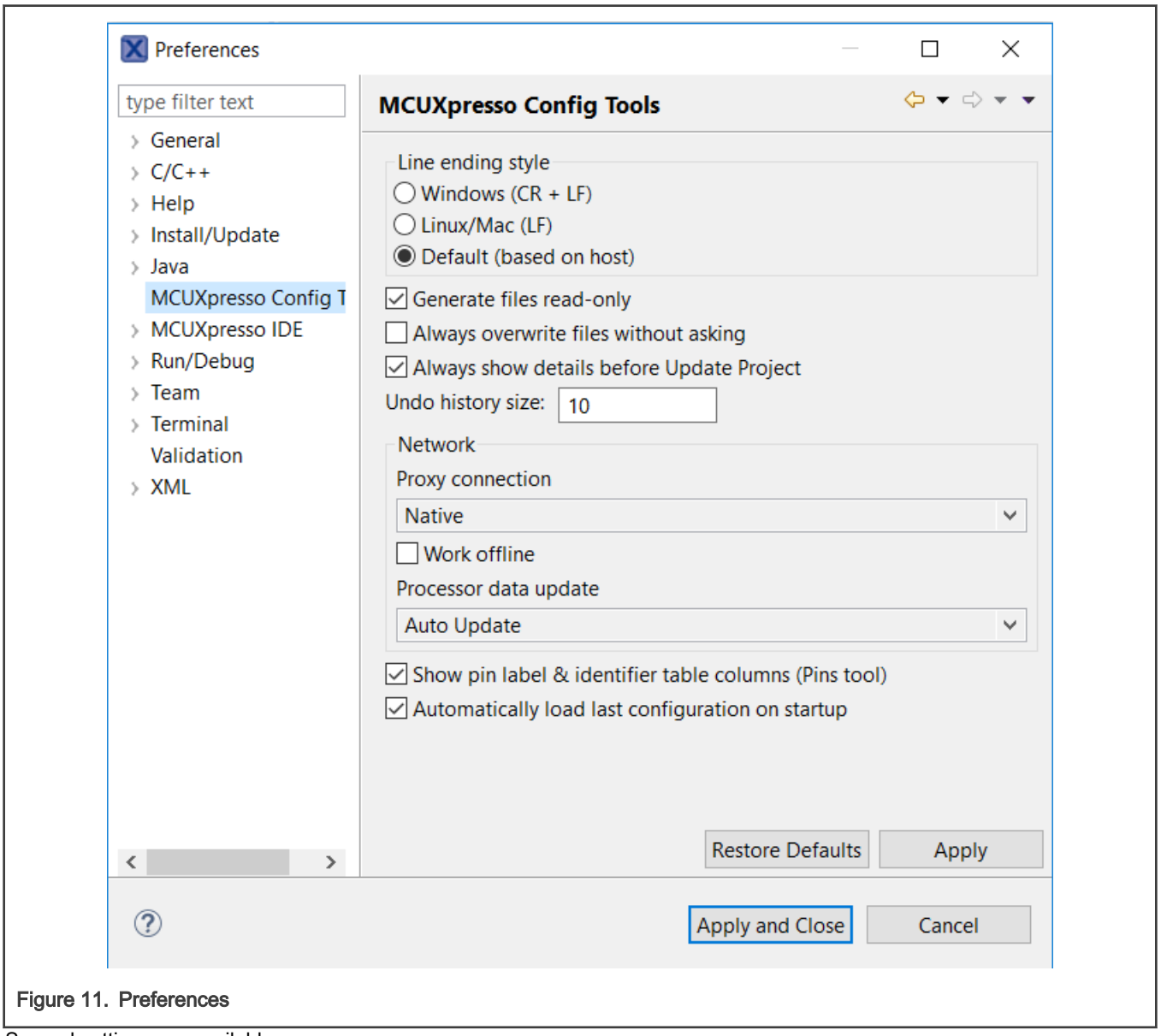


Figure 11. Preferences

Several settings are available.

Table 6. Preferences

Item	Description
Line ending style	Select between **Windows (CR + LF)**, **Linux/Mac (LF)**, or **Default (based on host)**.
Generate files read-only	Prevent modifying the source files unintentionally. Generated source files are marked as read-only.

Table continues on the next page...

Table 6. Preferences (continued)

Item	Description
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 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.
Show pin label & identifier table columns (Pins tool)	Select to show the pin label and the label identifier in the relevant views.
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.

2.5 Configuration preferences

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

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

Figure 12. Configuration Preferences

Several preferences are available.

Table 7. Configuration Preferences

Item	Description
Validate boot init only	Validate tools' dependencies only against 'boot init' function group.When selected, dependencies from all functional groups of all tools must be satisfied in the functional groups marked for default initialization. Clearing this option hides warnings in case the user is using complex scenarios with alternating functional groups within the application code.
Generate YAML	Generate YAML into C sources files.
Generate extended information into header file	Generate extended information into the header file. For projects created in earlier MCUXpresso versions, this option is selected by default.
Custom source file copyright header	Add a custom copyright header to generated source files that don't already contain copyright.
Generate code only for registers that are different from the after-reset state	Generate code only for registers that are different from the after-reset state. For projects created in earlier MCUXpresso versions, this option is selected by default.

WARNING

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

2.6 Problems view

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

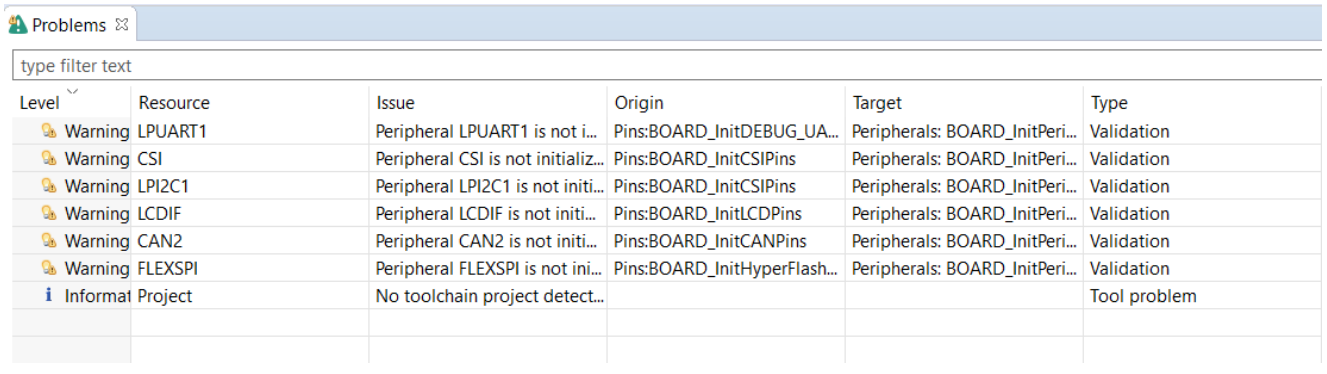


Figure 13. Problems view

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

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

Table 8. 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, and so on.
Issue	Description of the problem.
Origin	Information on the dependency source.
Target	Tool that handles the dependency and its resolution.
Type	Type of the problem. It's either the validation checking dependencies between tools, or a single tool issue.

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

NOTE

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

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

Table 9. Filter buttons

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

2.7 Registers view

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

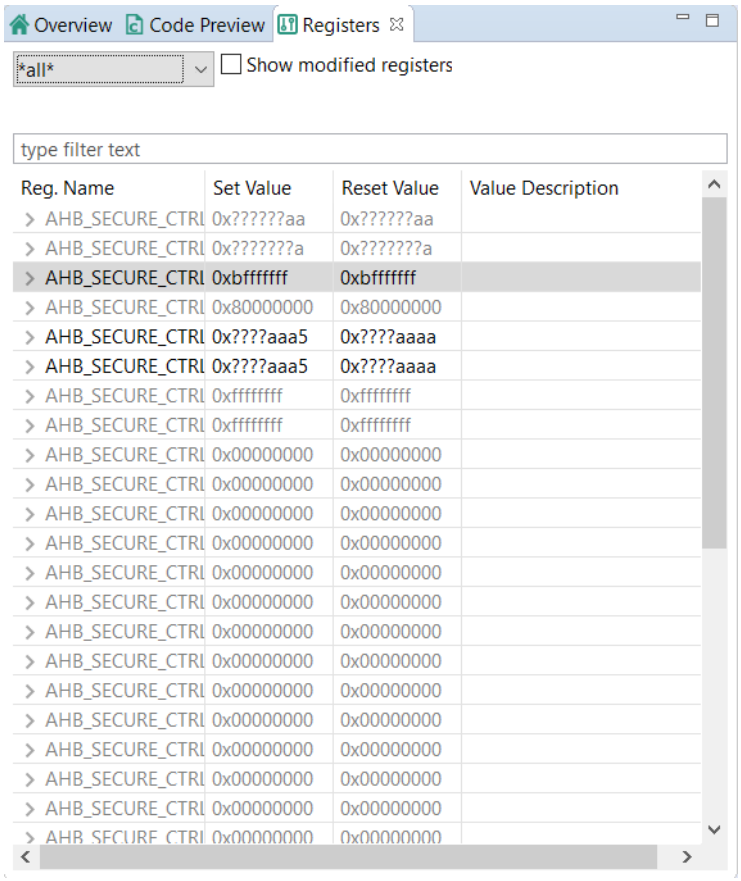


Figure 14. Registers view

The **Registers view** contains several items.

Table 10. 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 chekbox	Hide the registers that are left in their after-reset state or are not configured.
Text filter	Filter content by text.

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

Table 11. Color codes

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

NOTE

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

2.8 Log view

The **Log** view shows user-specific information about MCUXpresso Config Tools operations. The **Log** view can show up to 100 records across all tools in chronological order.

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

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

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

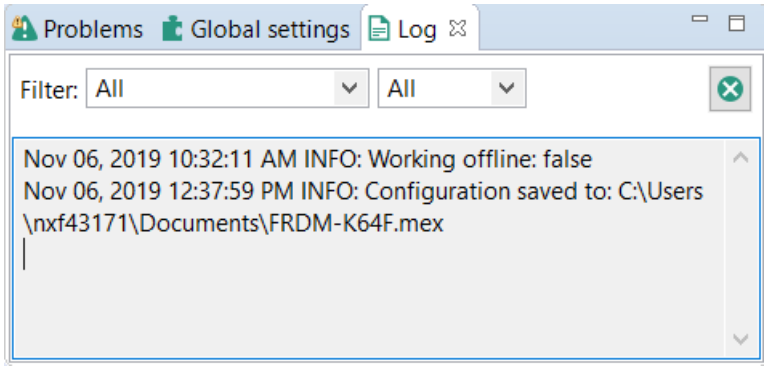


Figure 15. Log view

2.9 Config tools overview

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

Config Tools Overview contains several items.

Table 12. Config Tools Overview

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

NOTE

If you have disabled a tool and want to reopen it, click the tool icon in the upper right corner or select it from the Main Menu. The **Config Tools Overview** opens automatically.

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

Table 13. 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.

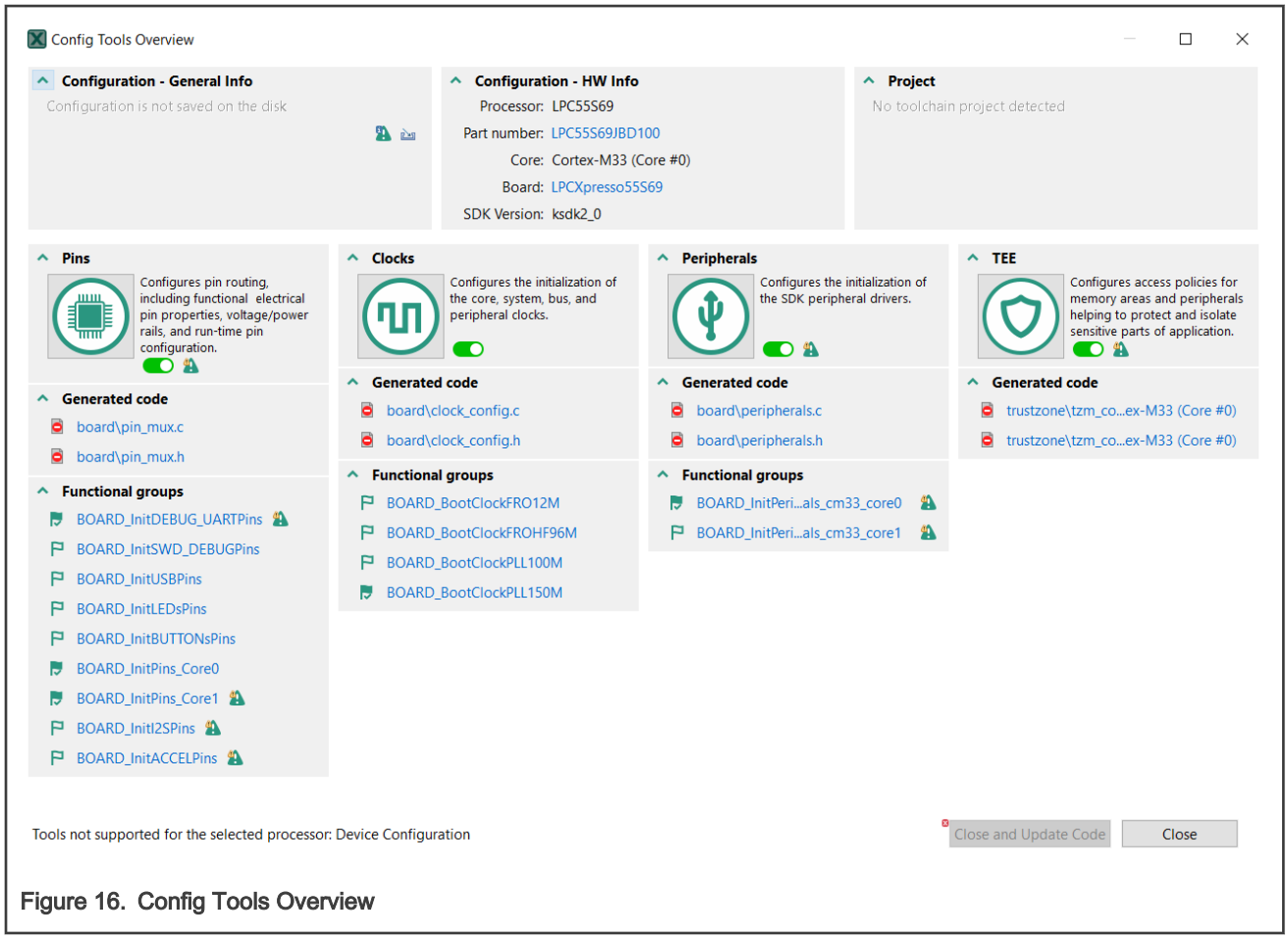
- 

Figure 16. Config Tools Overview

NOTE

Unsupported tools are not displayed in the overview.

Chapter 3
Pins Tool

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

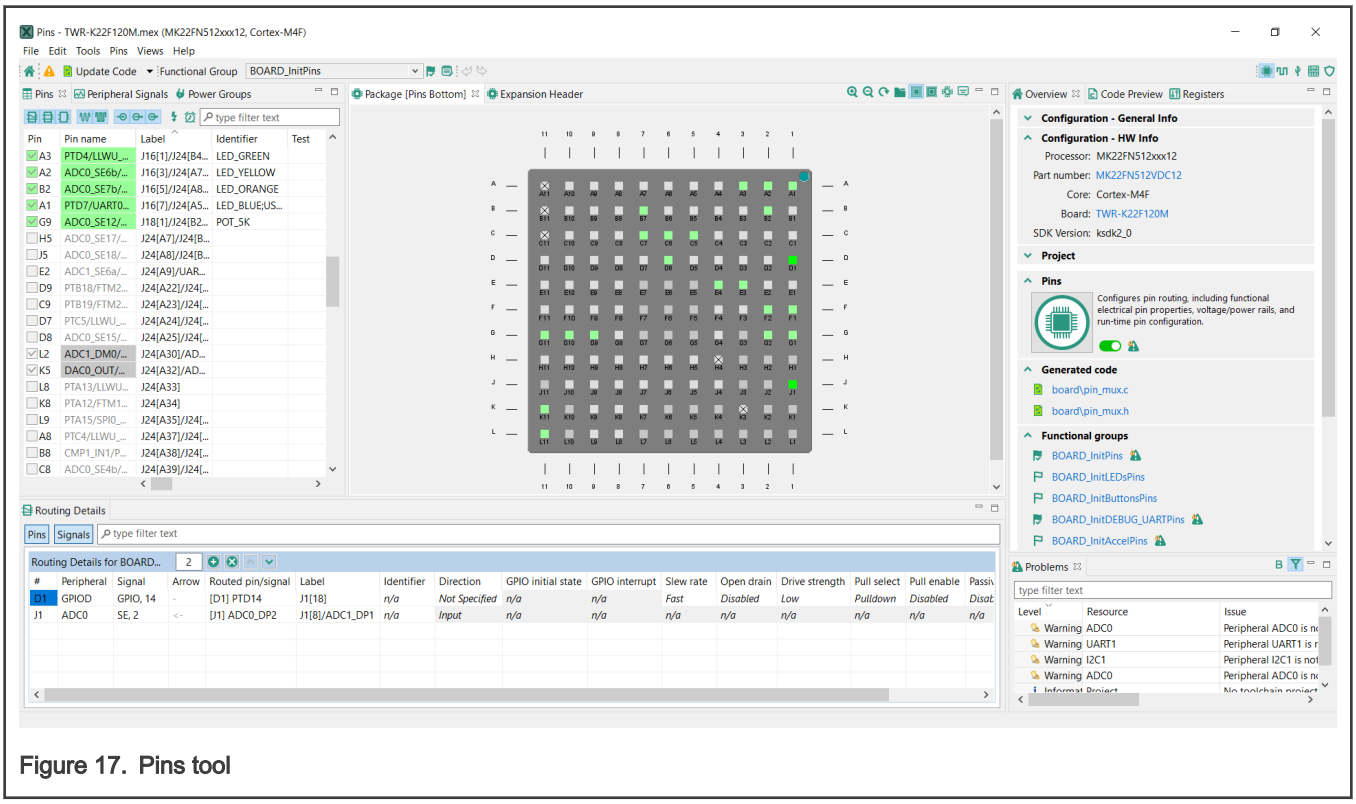


Figure 17. Pins tool

3.1 Pins routing principle

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

Internal signal is an interconnection node which peripheral signals can be connected to (without any pin interaction). Connecting two peripheral signals to internal signal makes an interconnection of these two peripheral signals.

This routing configuration can be done in the following views:

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

Following two sections describe the two methods you can use to define the routing path.

3.1.1 Beginning with peripheral selection

You can select the peripheral in the **Routing Details** view and the **Peripheral Signals** view.

1. Select the **Peripheral**.
2. In **Routing Details** view, select one of the available **Signals** or expand the peripheral in **Peripheral Signals** view.
3. Selected the desired pin/internal signal.

Items (pins/internal signals) in the **Routed pin/signal** column in the **Routing Details** view have following decorators:

- Exclamation mark and default text color indicates that such item selection causes a register conflict or the item cannot be routed to the selected peripheral signal (some other peripheral signal can be).
- Exclamation mark and gray text color indicates that the item cannot be routed to any signal of the selected peripheral. The item is available for different peripheral using the same signal.

NOTE

Route to field in **Routing details** view contains items that are connectable to the selected signal (without its channel if applicable). So when selected signal is “GPIO, 6” then the **Routed pin/signal** provides items connectable to “GPIO”.

3.1.2 Beginning with pin/internal signal selection

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

1. Select the pin/internal signal (**Routed pin/signal**).
2. Select one of the available **Peripherals**. In the **Pins** view, see all available peripherals/signals by selecting the checkbox in the first column or scroll down to the required peripheral type.
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 indicates that such item selection can cause a register conflict or the item does not support selected signal.
   - Exclamation mark and gray text color indicates that the item cannot be routed to the selected pin/internal signal. The item is available for different pin/internal signal using the same signal.

   NOTE

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

3.1.3 Routing of peripheral signals

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

Three types of peripheral signal routing are available:

1. Routing the signal from the output of an internal peripheral (A) into the input of another internal peripheral (B)

   The signal leads from the output of one internal peripheral (A) to the input node of another internal peripheral (B). In other words, 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 new row in the **Routing Details** view.

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

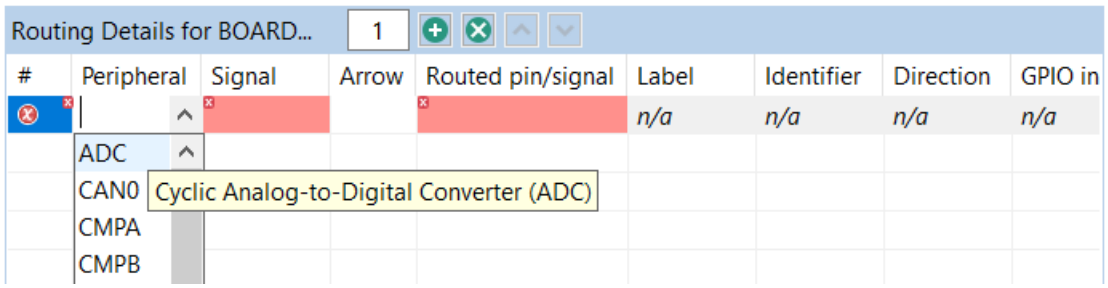


Figure 18. Selecting the peripheral (B)

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

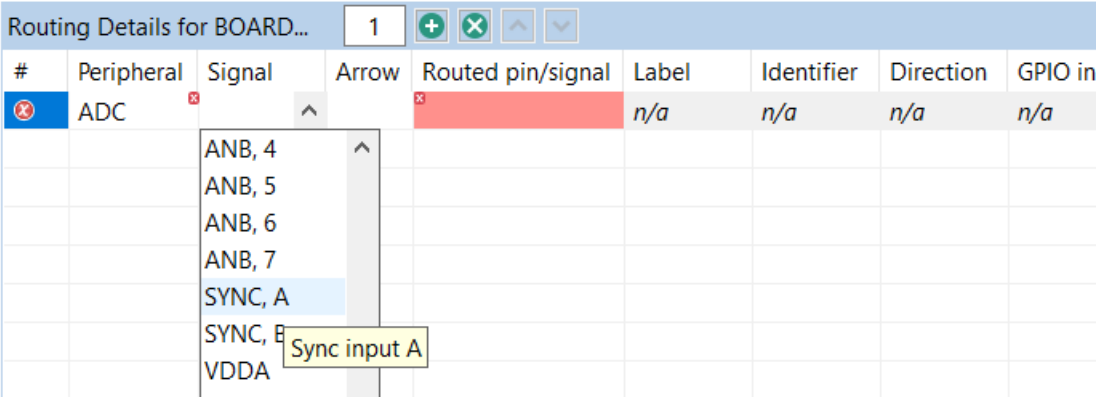


Figure 19. Selecting the input node (B)

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

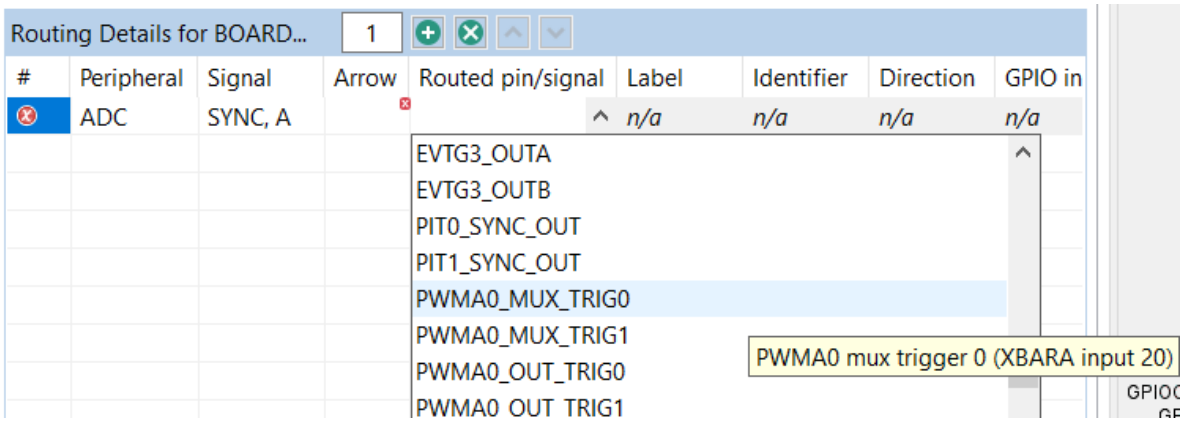
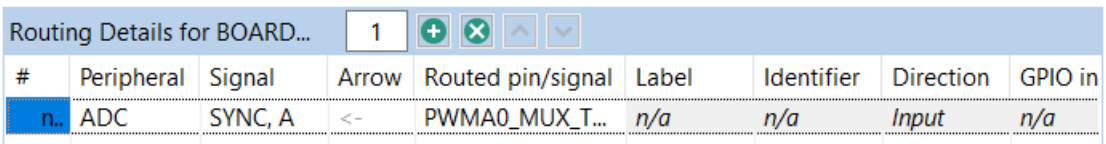


Figure 20. Selecting the output signal

Once the configuration is done, the row will look like this:

Figure 21. Result



NOTE

It's necessary to select the ADC peripheral where the signal leads to (input in ADC). It's 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 new row in the **Routing Details** view.

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

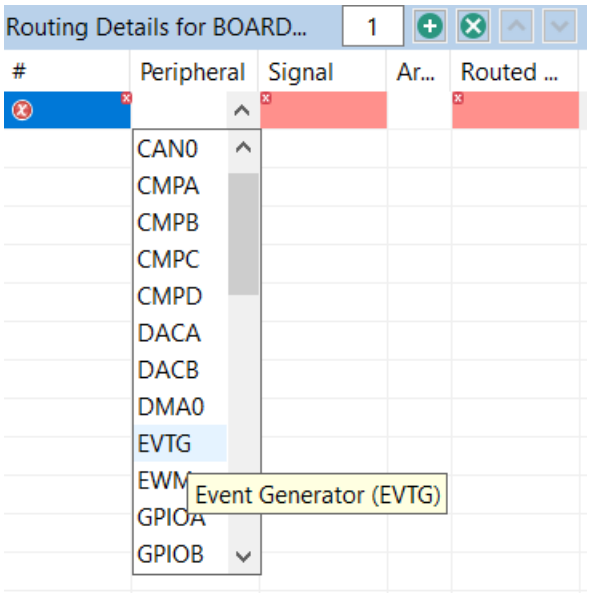


Figure 22. Selecting the peripheral (B)>

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

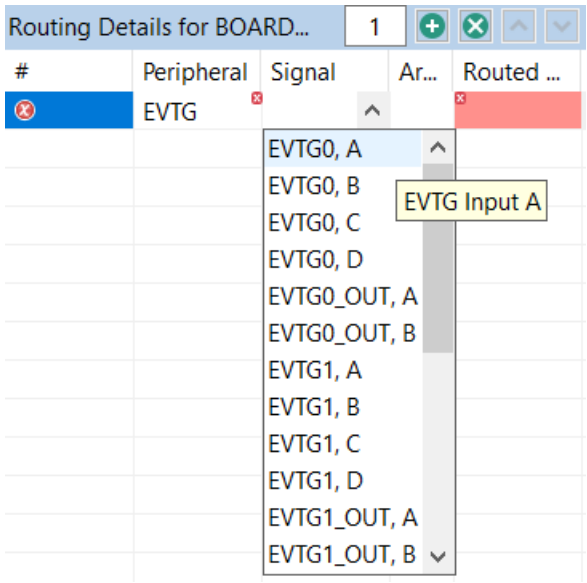


Figure 23. Selecting the input node (B)

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

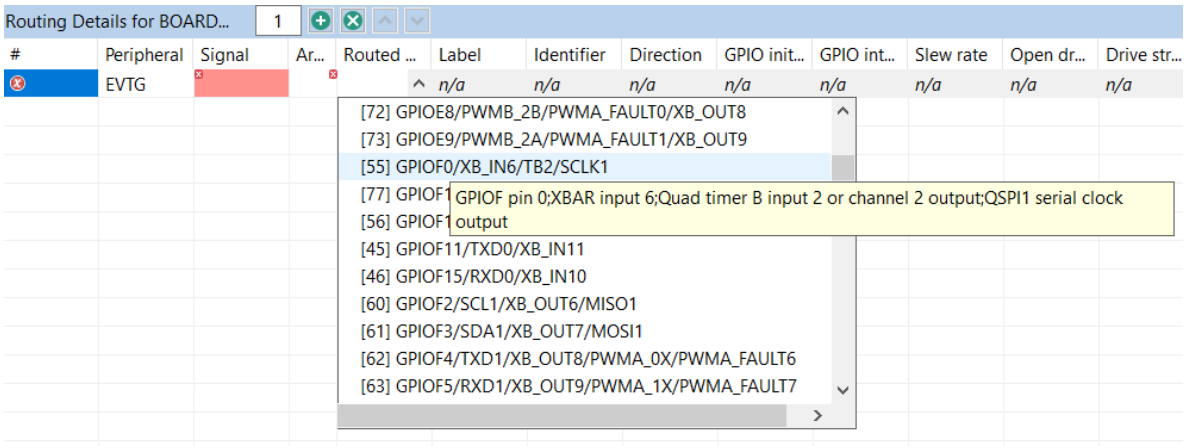


Figure 24. Selecting the pin

Once the configuration is done, the row will look like this:

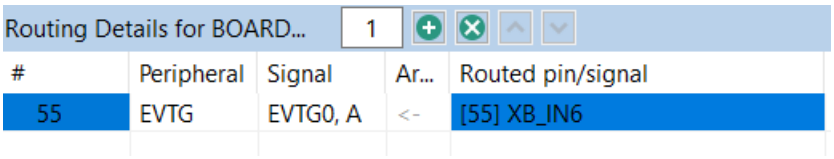


Figure 25. Result

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 new row in the **Routing Details** view.

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

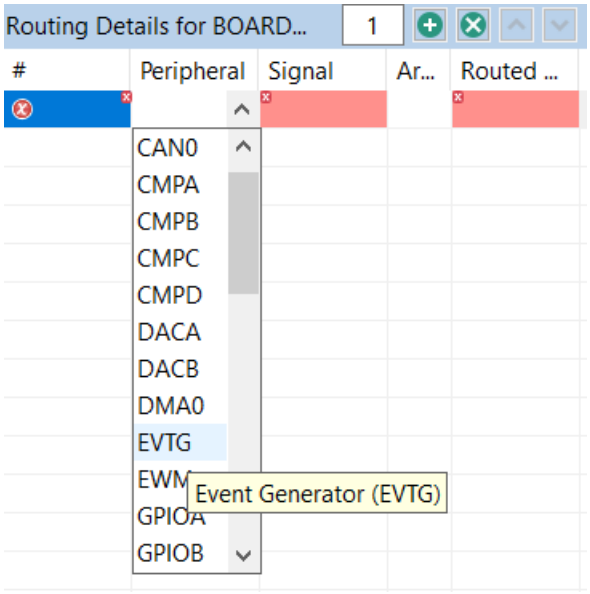


Figure 26. Selecting the peripheral (A)

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

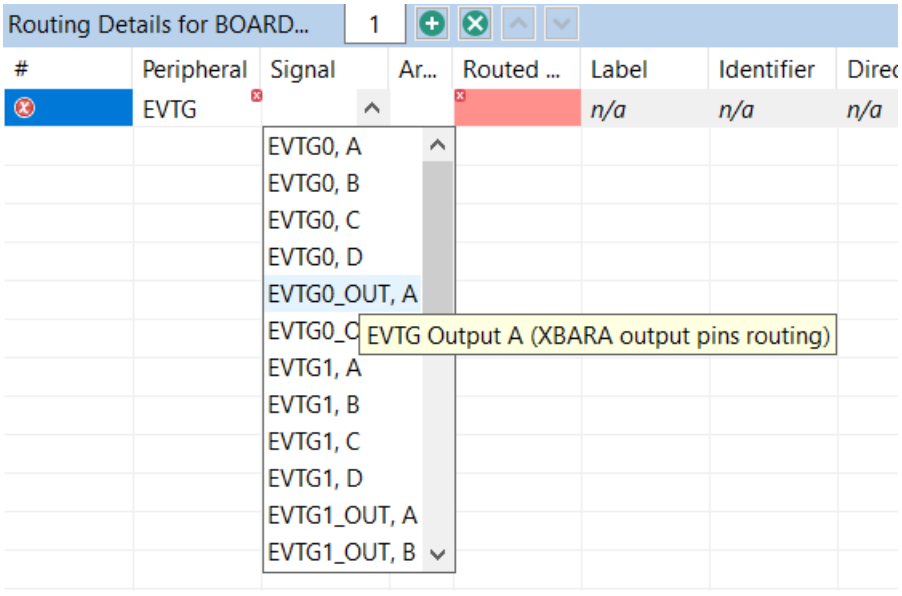


Figure 27. Selecting the output signal (A)

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

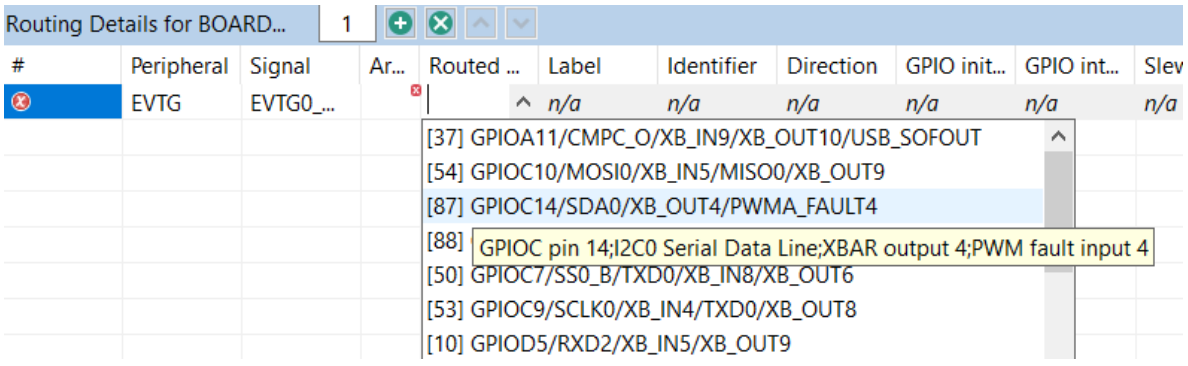


Figure 28. Selecting the pin

Once the configuration is done, the row will look like this:

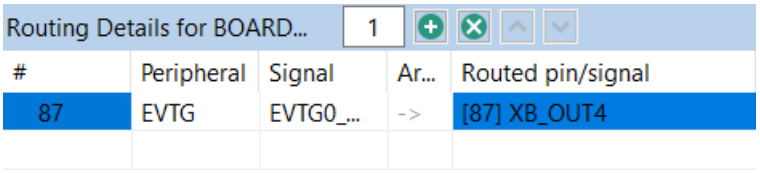


Figure 29. Result

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 workflow

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

In this example, three pins **(UART3_RX, UART3_TX and PTB20)** on a board are configured.

You can use the generated files with the application code.

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

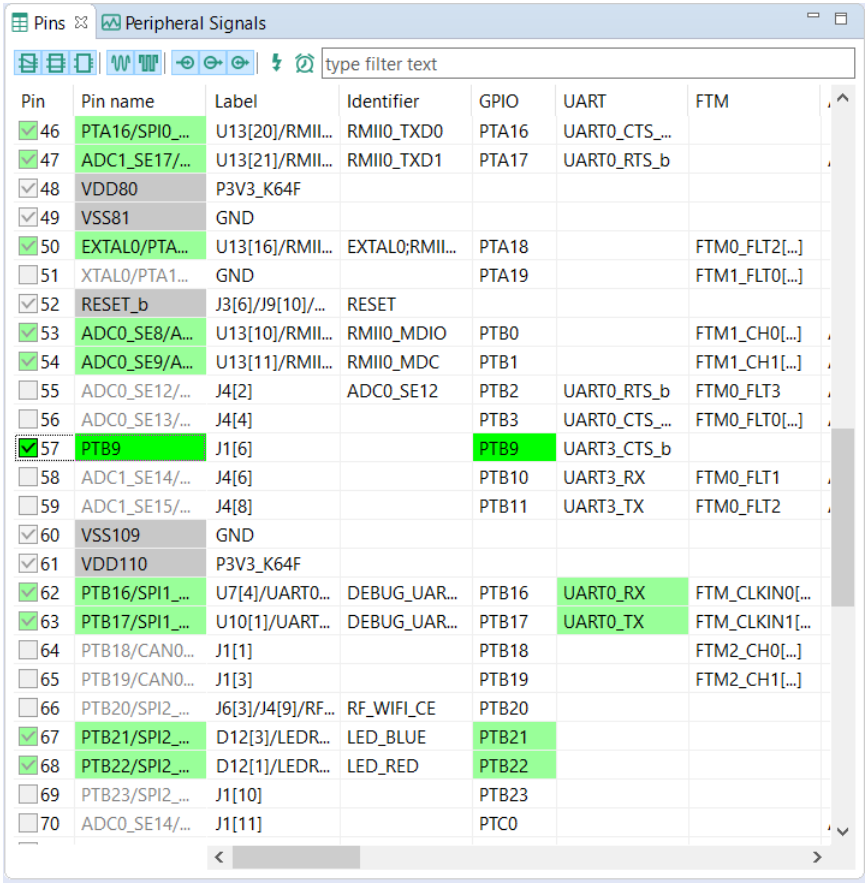


Figure 30. Configuring signals in the Pins view

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

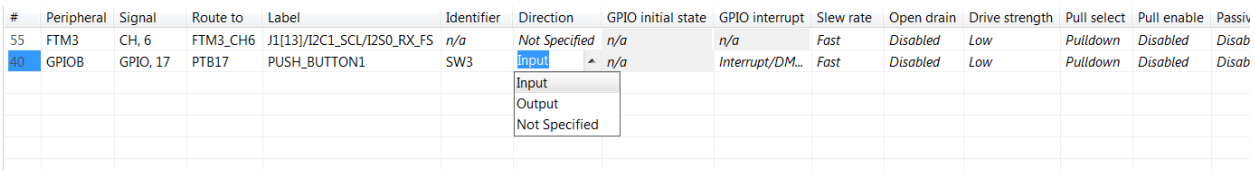


Figure 31. Selecting direction

NOTE

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

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

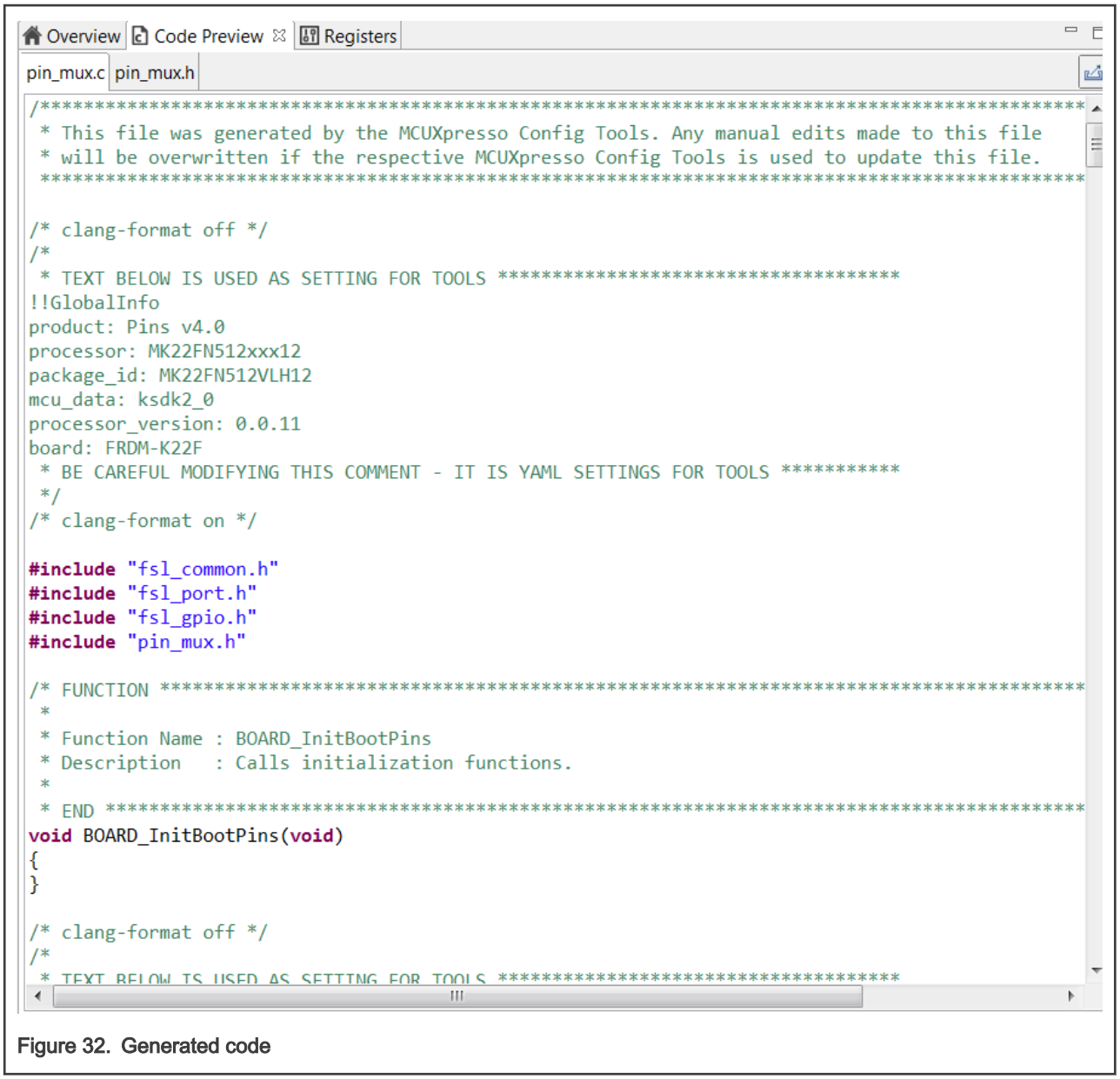


Figure 32. Generated code

4. You can now copy-paste the content of the source(s) to your application and IDE. Alternatively, you can export the generated files or update the code with the **Update Code** button in **Toolbar**. To export the files, select **File > Export** (in the desktop version) or select the menu **Pins > Export** menu (in the Web version). In the **Export** dialog expand the tree control for the tool you want to export sources for and select the **Export Source Files** option. **Export**, select the **Export Source Files** option.

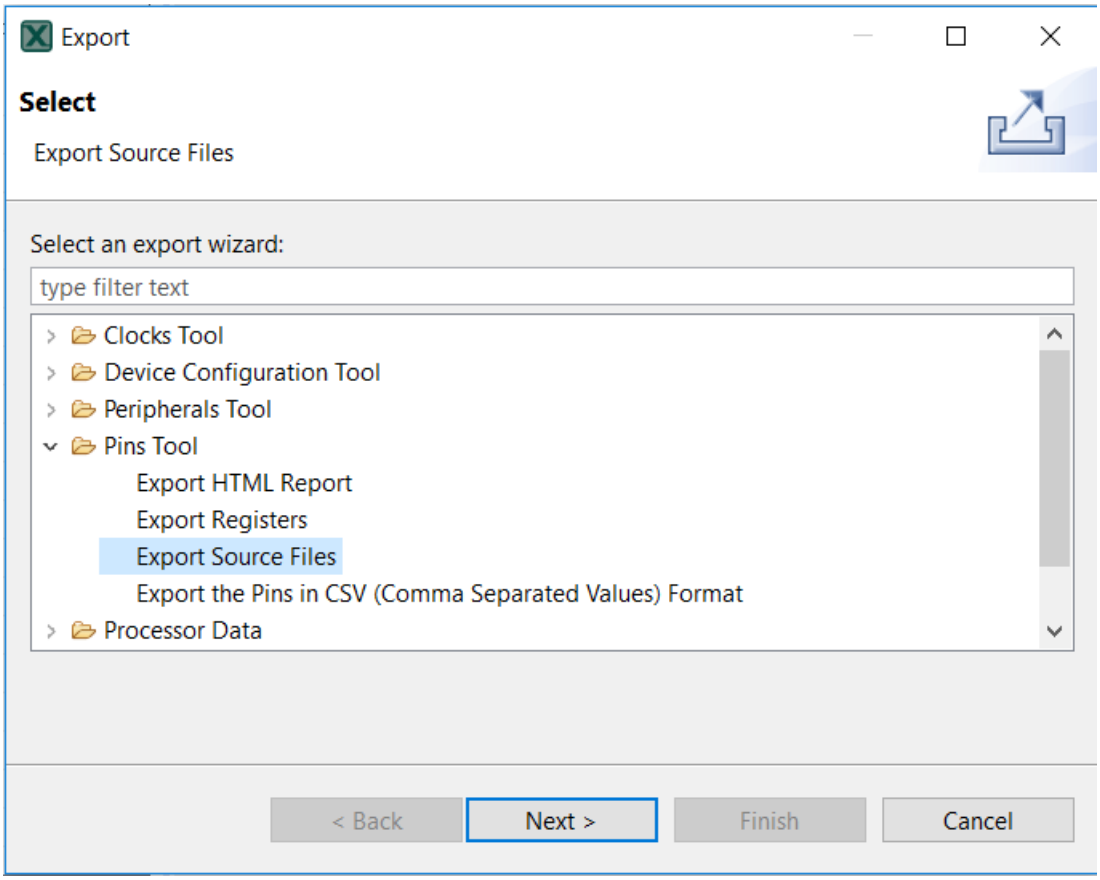


Figure 33. Export

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

3.3 User interface

The Pins tool consists of several views.

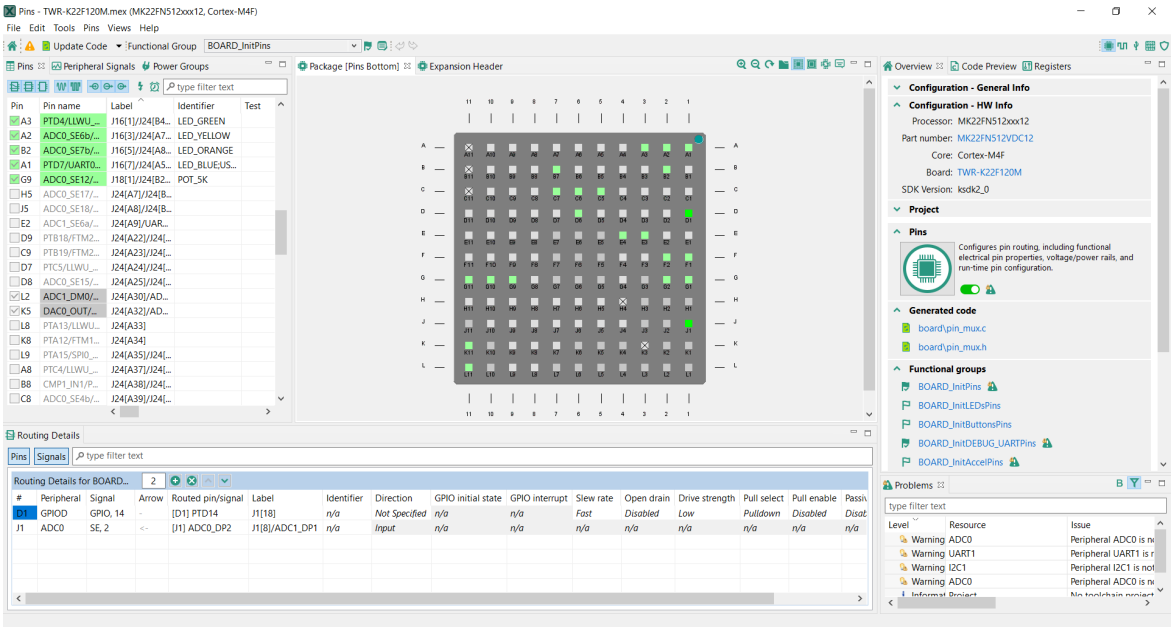


Figure 34. Pins tool user interface

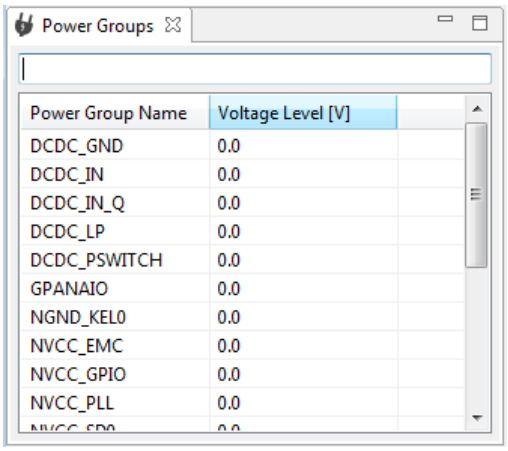


Figure 35. Selecting power group

NOTE

Power Groups are not supported for all processors.

3.3.1 Pins view

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

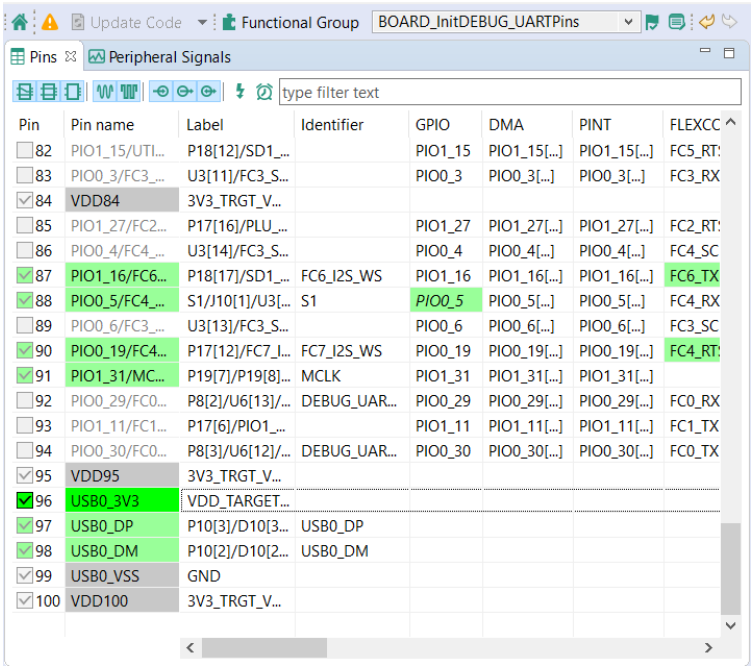


Figure 36. Pins table view

This view shows the list of all the pins available on a given device. The **Pin name** column shows the default name of the pin, or if the pin is routed. The pin name is changed to show appropriate function for selected peripheral if routed. The next columns of the table shows peripherals and pin name(s) on given peripheral. Peripherals with few items are cumulated in the last column.

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

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

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

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 view

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

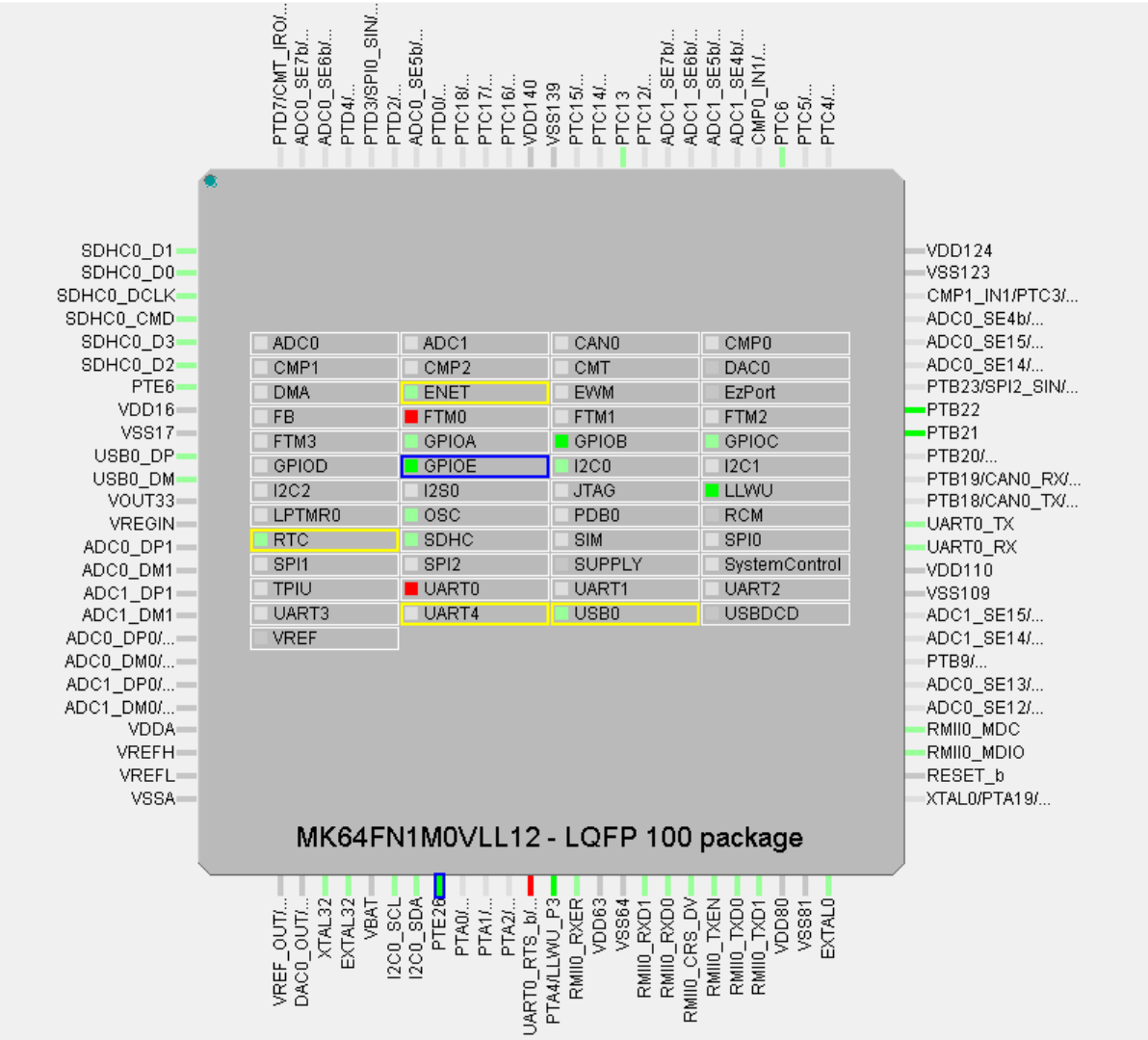


Figure 37. Processor package

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 14. 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.

NOTE

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

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

3.3.3 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 15. Status codes

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

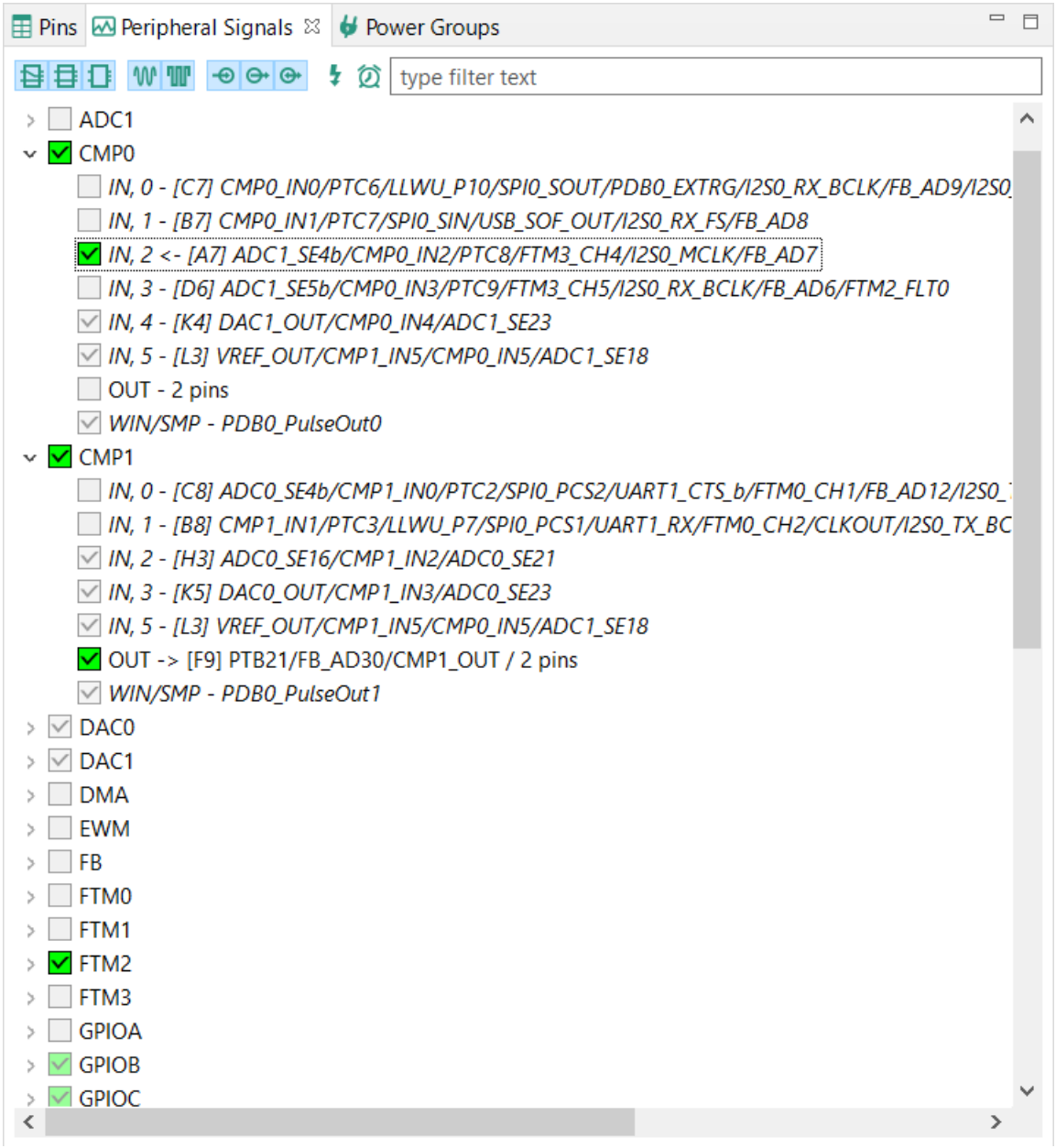


Figure 38. Peripheral Signals view

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.

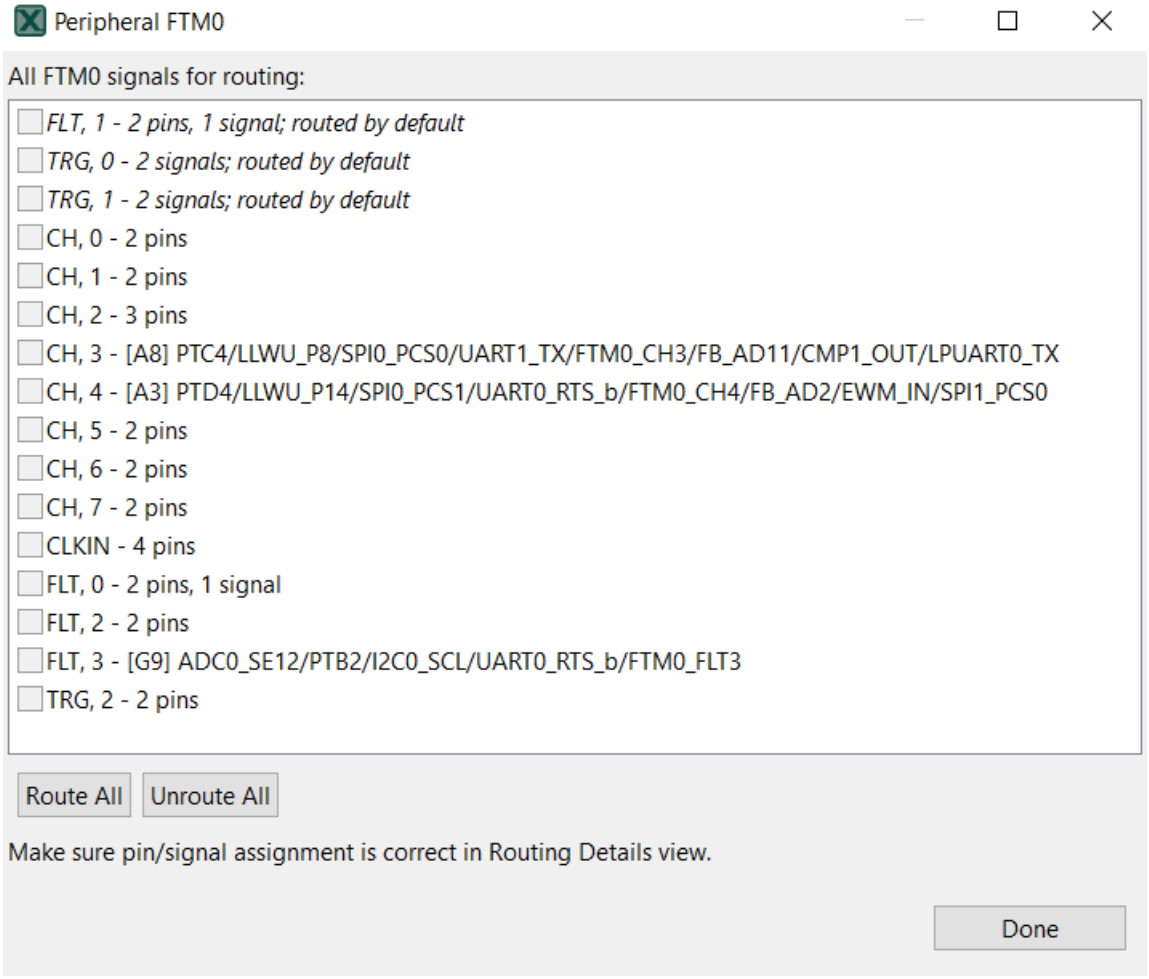


Figure 39. Select signals dialog

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.

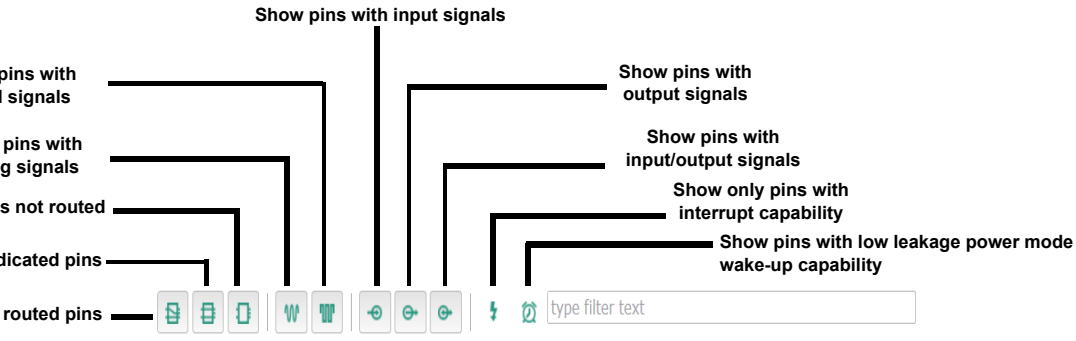


Figure 40. Filtering Controls

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.

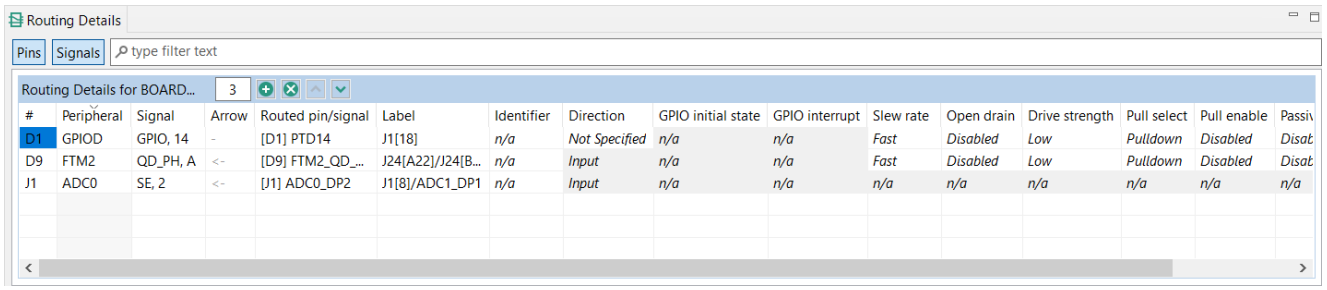


Figure 41. Routing Details

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

To remove the selected row, click on 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

- 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's 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.

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

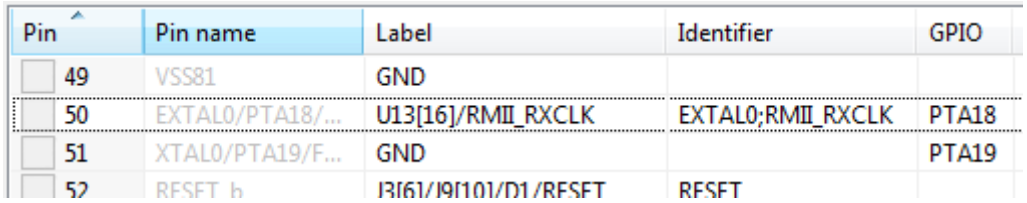


Figure 42. Pin identifier

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

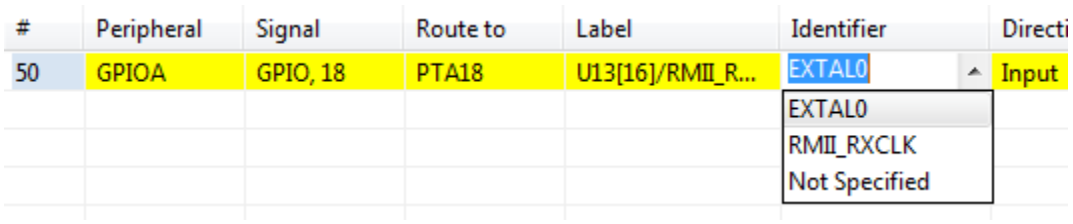


Figure 43. Identifier in Routing Details table

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.

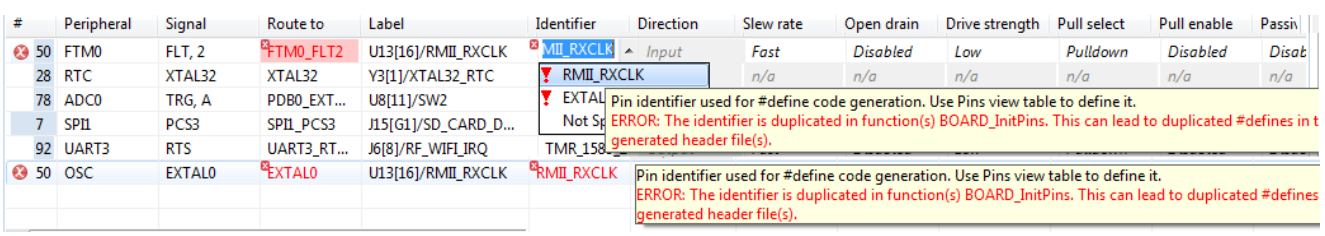


Figure 44. Identifier errors

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 should be used in the source code.

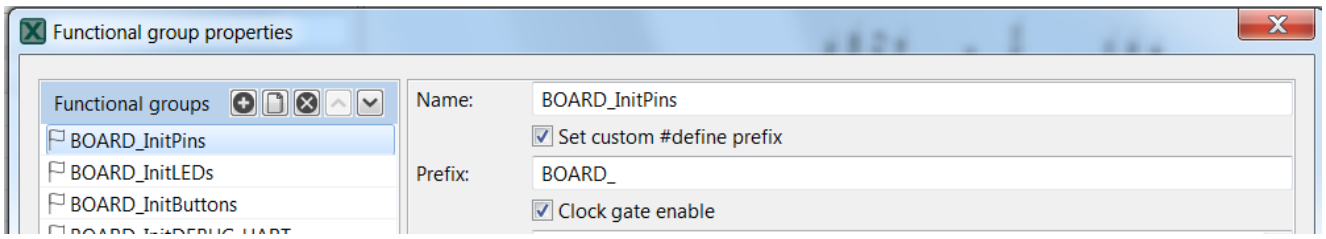


Figure 45. Pins macros prefix

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

3.3.5 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.

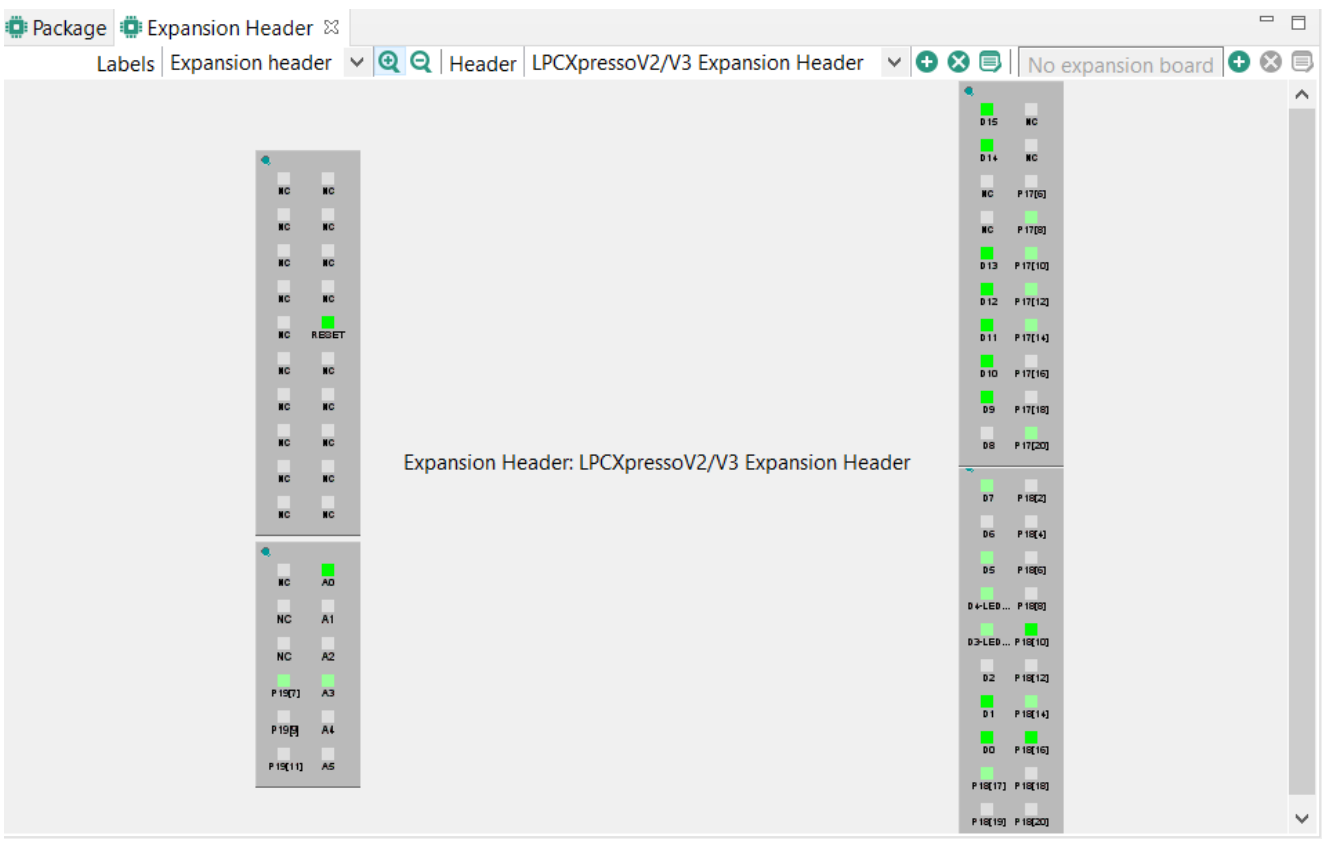


Figure 46. Expansion header

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

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

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

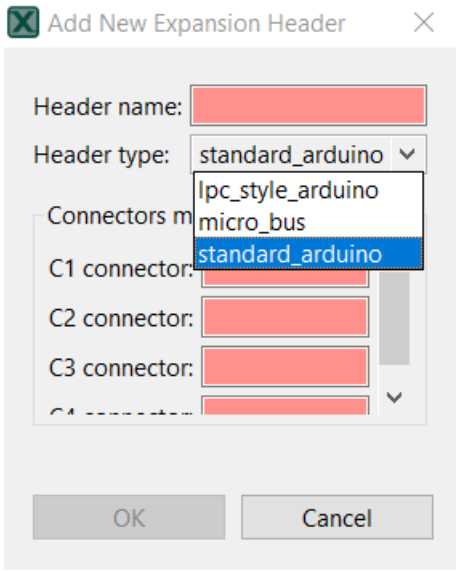


Figure 47. Adding new expansion header

4. Name the header and map the connectors.

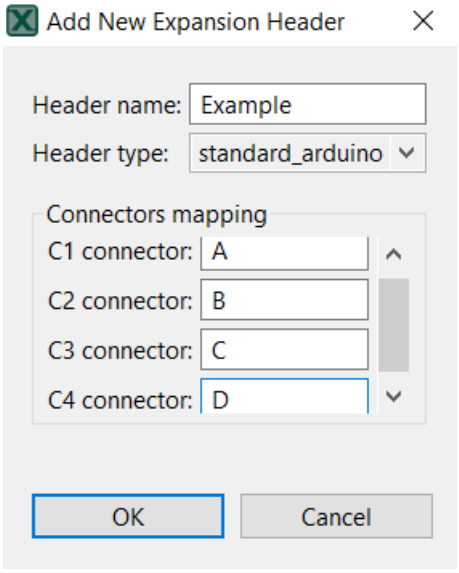


Figure 48. Adding new expansion header

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.

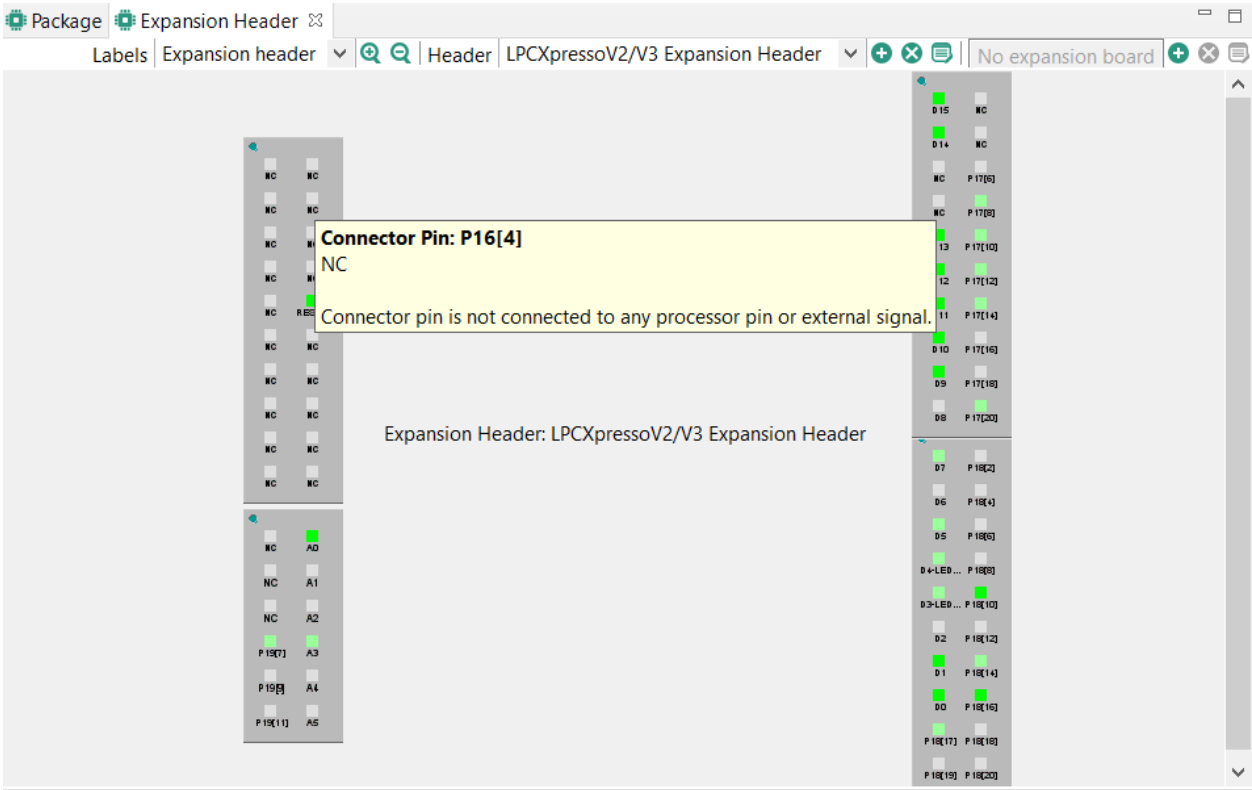


Figure 49. Expansion header

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**.

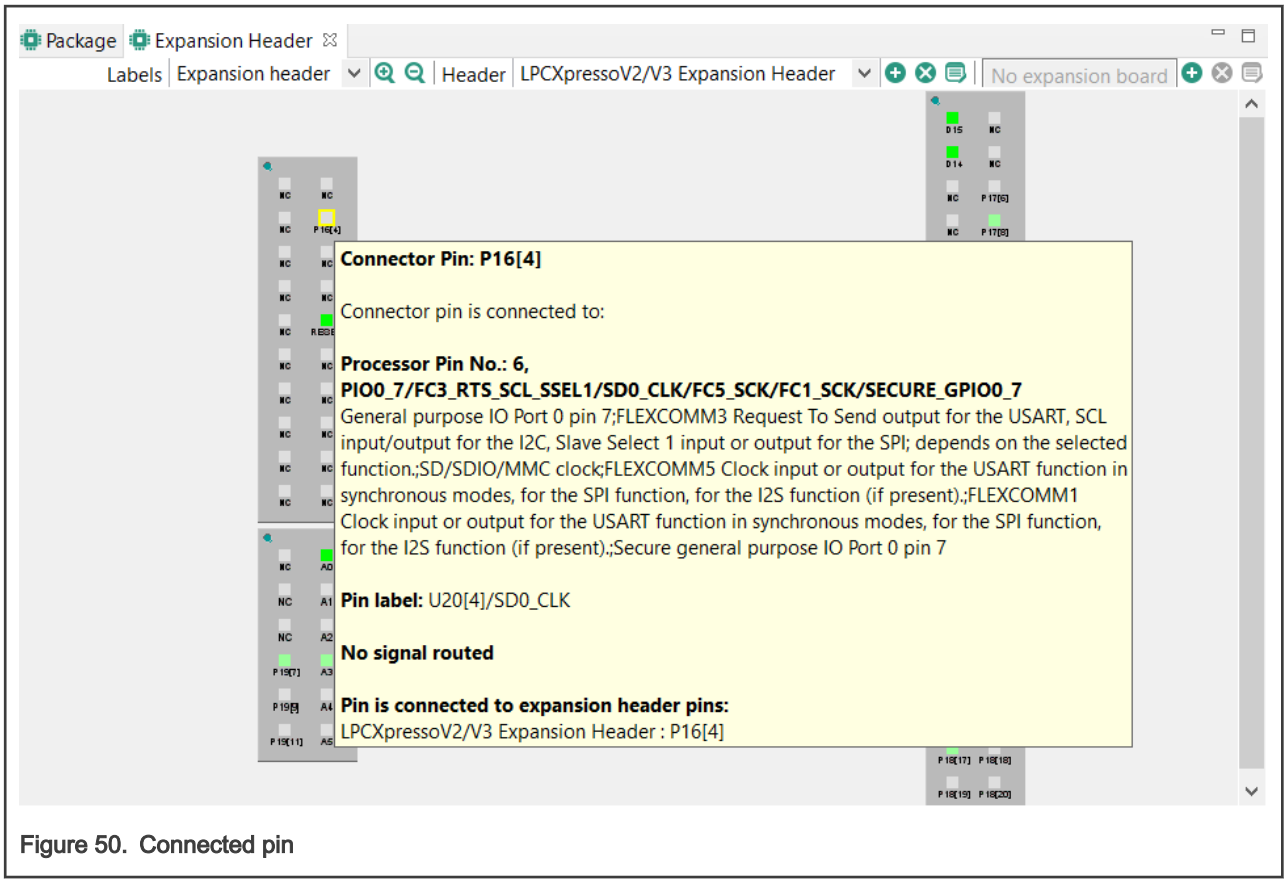


Figure 50. Connected pin

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.

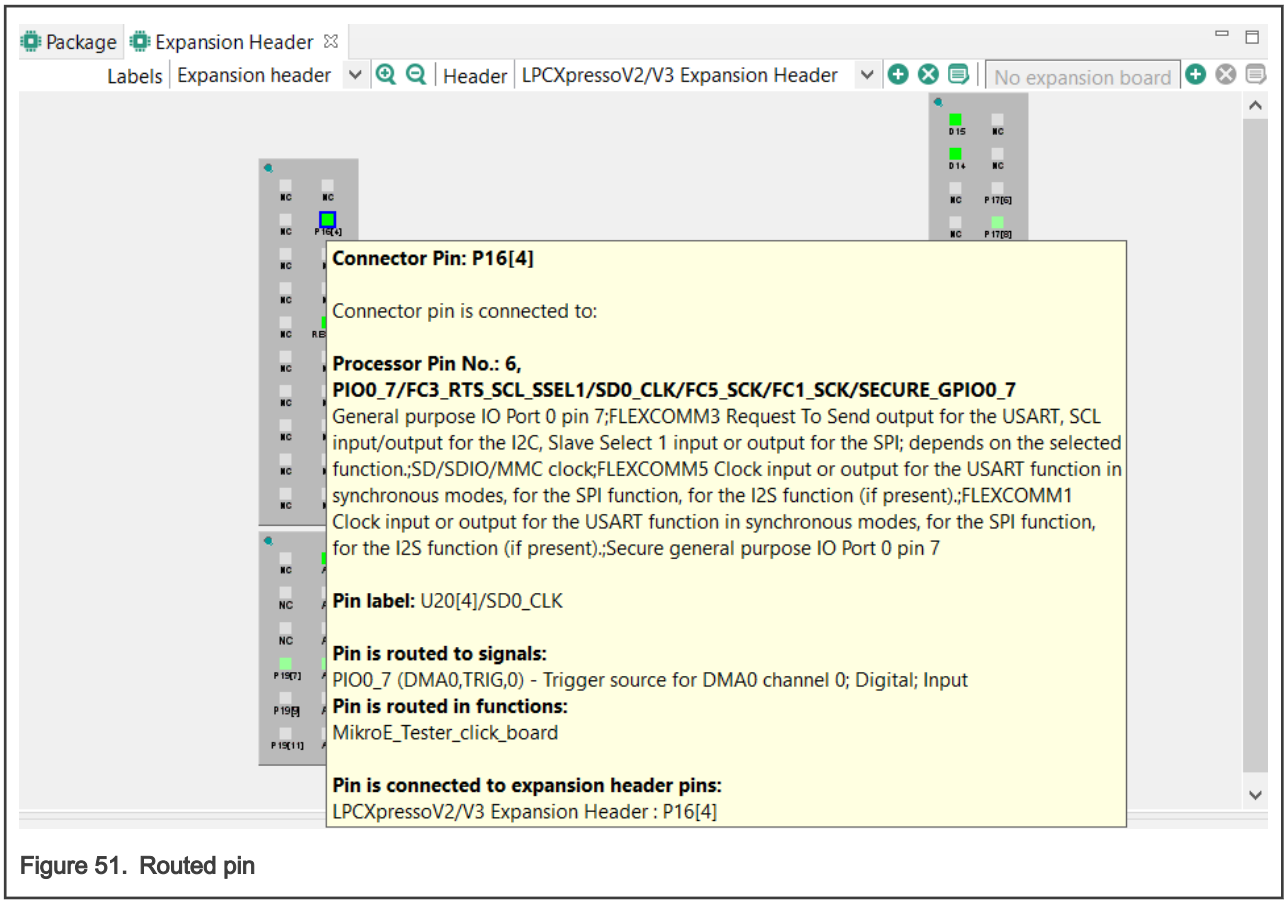


Figure 51. Routed pin

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.

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

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

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**.

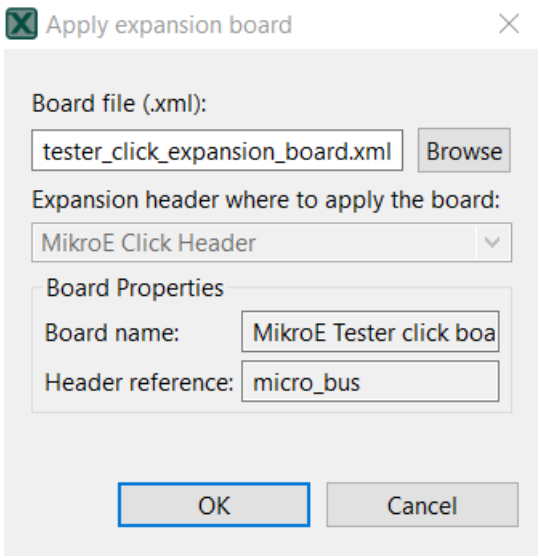


Figure 52. Apply expansion board

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**.

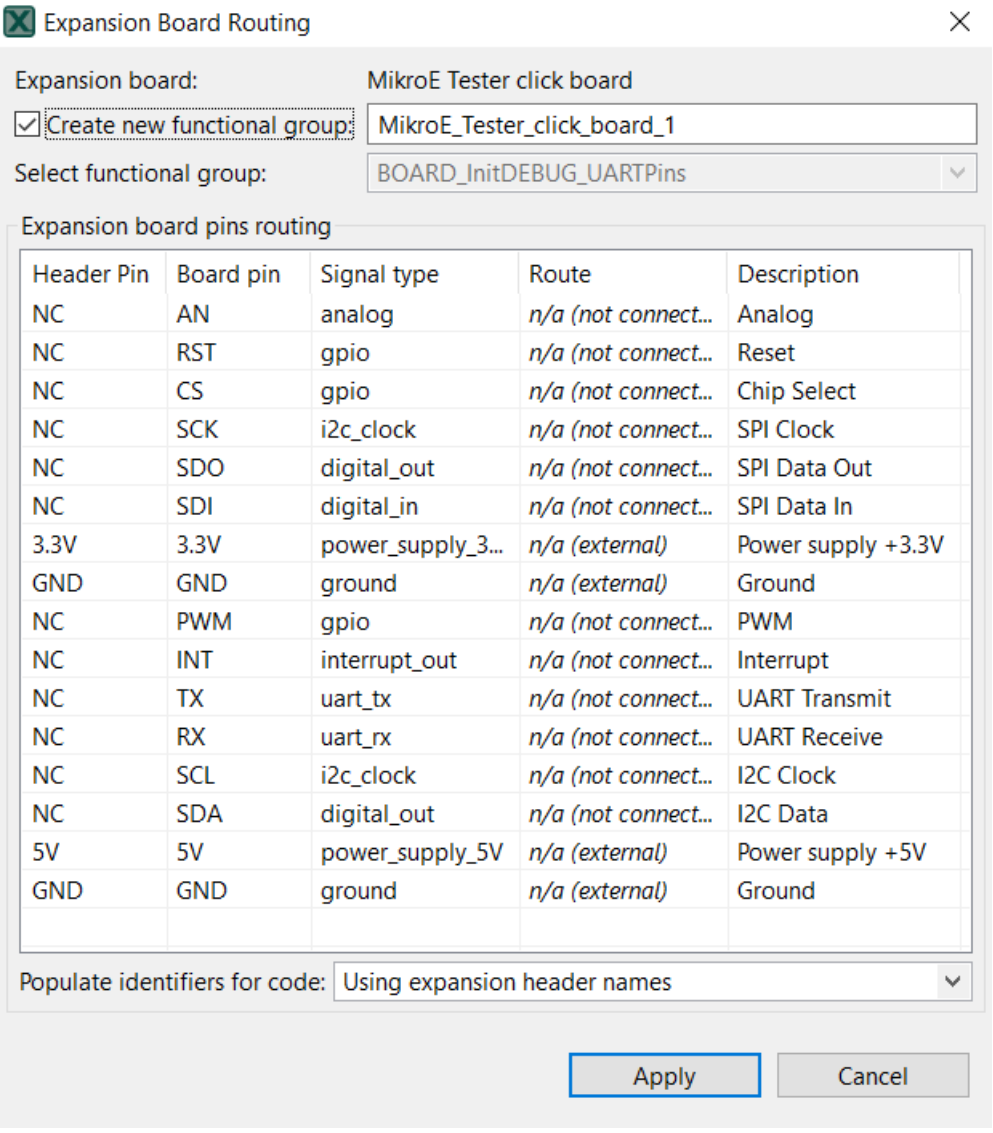


Figure 53. Expansion board routing

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 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 grey indicates that the pin is available for mux, but is not muxed or used.
- Dark gray indicates that the pin/peripheral is dedicated. It is routed by default and has no impact on generated code.

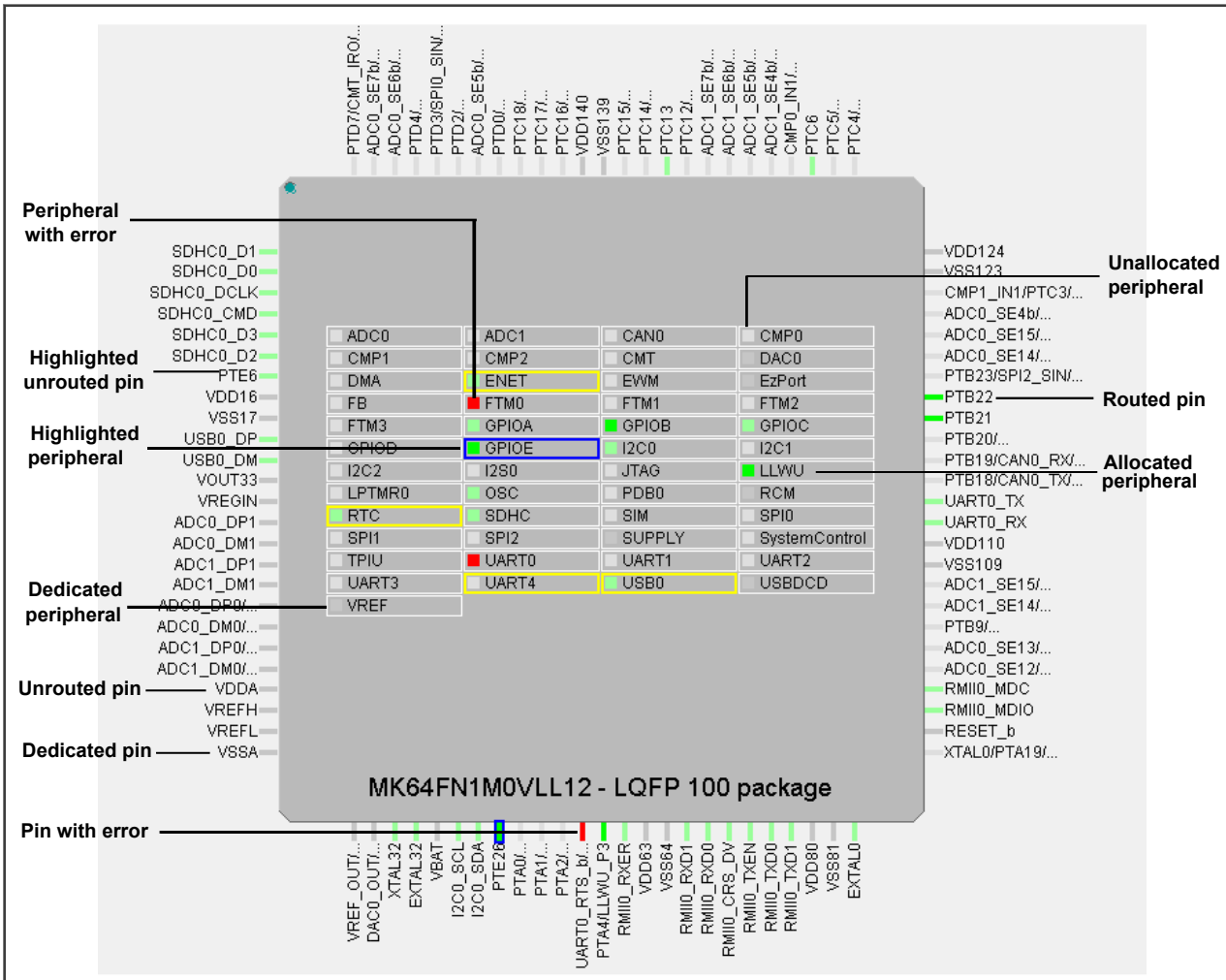
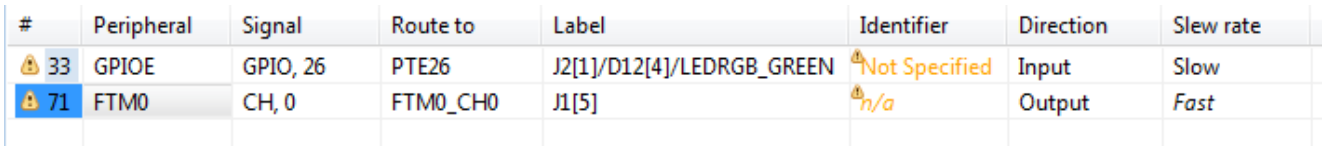


Figure 54. Highlighting and color coding

#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Open drain	Drive strenc
80	CMP0	IN, 0	ADC1_SE4...	J1[7]	n/a	n/a	Fast	Disabled	Low
26	CMP1	IN, 1	VREF_OUT...	J2[17]	n/a	n/a	n/a	n/a	n/a
4	GPIOE	GPIO, 3	PTE3	J15[P3]/SDHC0_C...	Not Specified	Not Specifi...	Fast	Disabled	Low
	CMP1	IN, 0			n/a	n/a	n/a	n/a	n/a
6	UART3	RX	UART3_RX	J15[P1]/SDHC0_D2	Not Specified	Input	Fast	Disabled	Low
6	FTM3	CH, 0	FTM3_CH0	J15[P1]/SDHC0_D2	Not Specified	Not Specifi...	Slow	Disabled	Low
					n/a	n/a	n/a	n/a	n/a

Figure 55. Pins conflicts



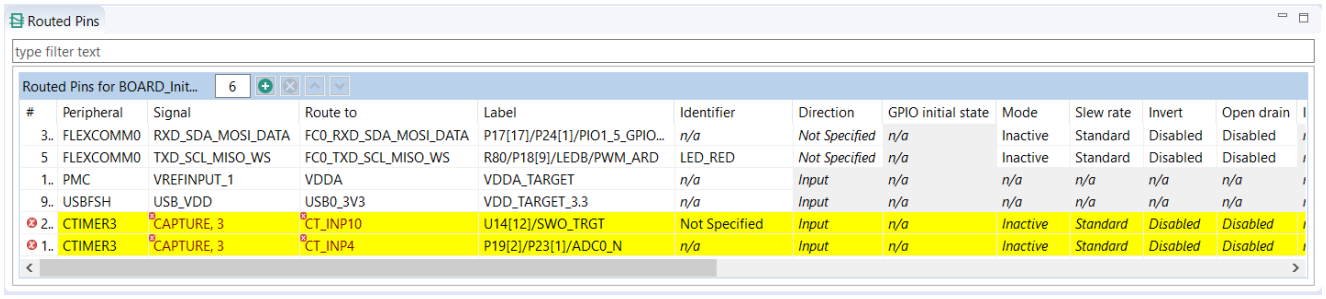
#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate
33	GPIOE	GPIO, 26	PTE26	J2[1]/D12[4]/LEDRGB_GREEN	Not Specified	Input	Slow
71	FTM0	CH, 0	FTM0_CH0	J1[5]	n/a	Output	Fast

Figure 56. Warnings

- **Package** view
  - Click on the peripheral or use the pop-up menu to highlight peripherals:
    - and all allocated pins (to selected peripheral).
    - or all available pins if nothing is allocated yet.
  - Click on the pin or use the pop-up menu to highlight the pin and the peripherals.
  - Click outside the package to cancel the highlight.
- **Peripherals / Pins** view
  - The peripheral and pin behaves as described above image.

3.4 Errors and warnings

The Pins Tool checks for any conflict in the routing and also for errors in the configuration. Routing conflicts are checked only for the selected function. It is possible to configure different routing of one pin in different functions to allow dynamic pins routing re-configuration.



#	Peripheral	Signal	Route to	Label	Identifier	Direction	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 57. 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 first two rows in the figure above show the peripheral/signal where the erroneous configuration occurs. The fourth row shows the warning on the unconfigured identifier while specifying a direction. The detailed error/warning message appears as a tooltip.

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

3.4.1 Incomplete routing

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

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

Figure 58. Incomplete routing

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

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

NOTE

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

3.5 Code generation

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

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

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

NOTE

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

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

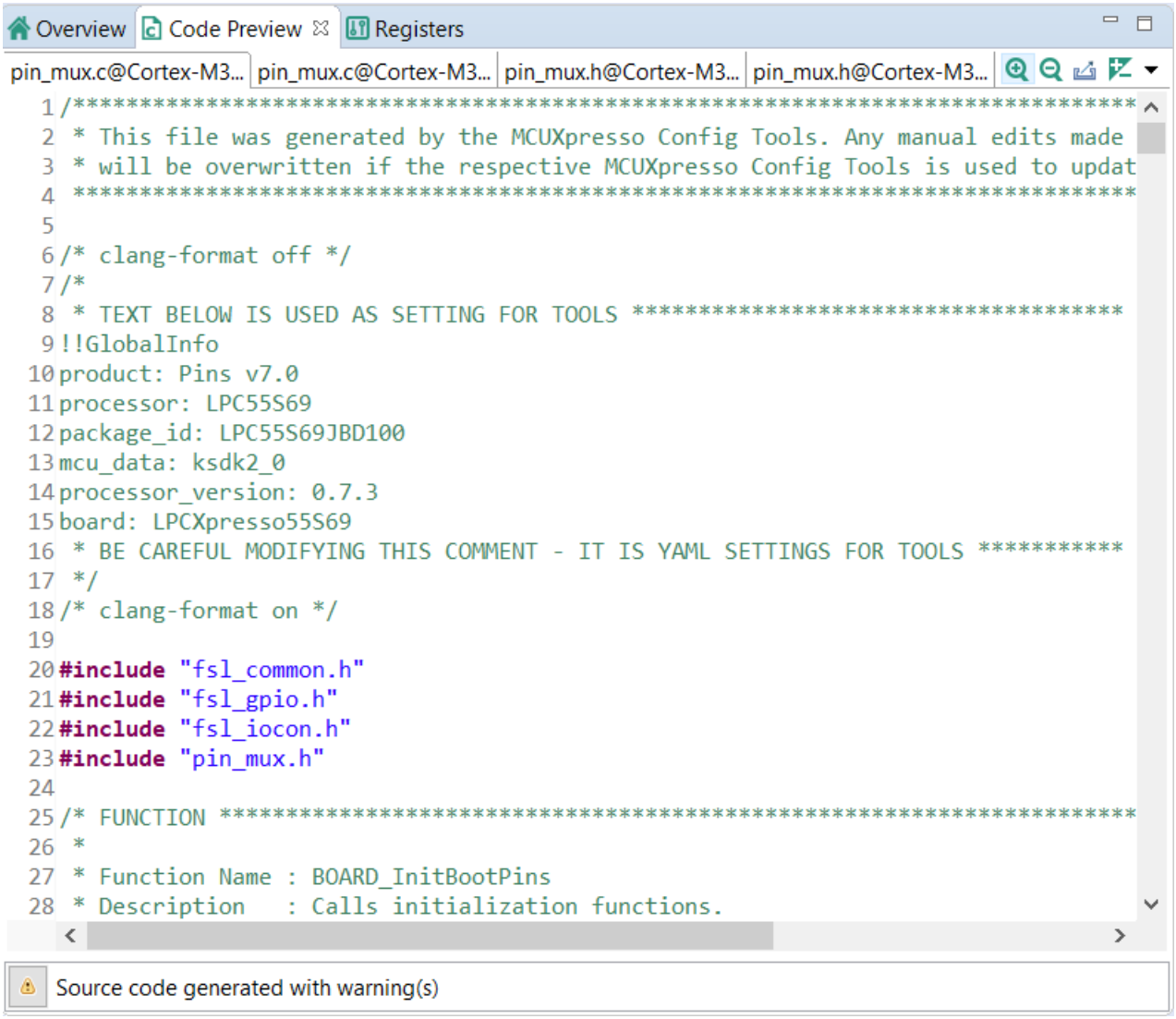


Figure 59. Generated code

YAML configuration contains configuration of each pin. It stores only non-default values.

TIP

For multicore processors, it will generate source files for each core. If processor is supported by SDK, it can generate BOARD_InitBootPins function call from main by default. You can specify "Call from BOARD_InitBootPins" for each function, in order to generate appropriate function call.

3.6 Using pins definitions in code

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

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

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

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

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

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

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

Chapter 4
Clocks Tool

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

4.1 Features

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

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

4.2 User interface

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

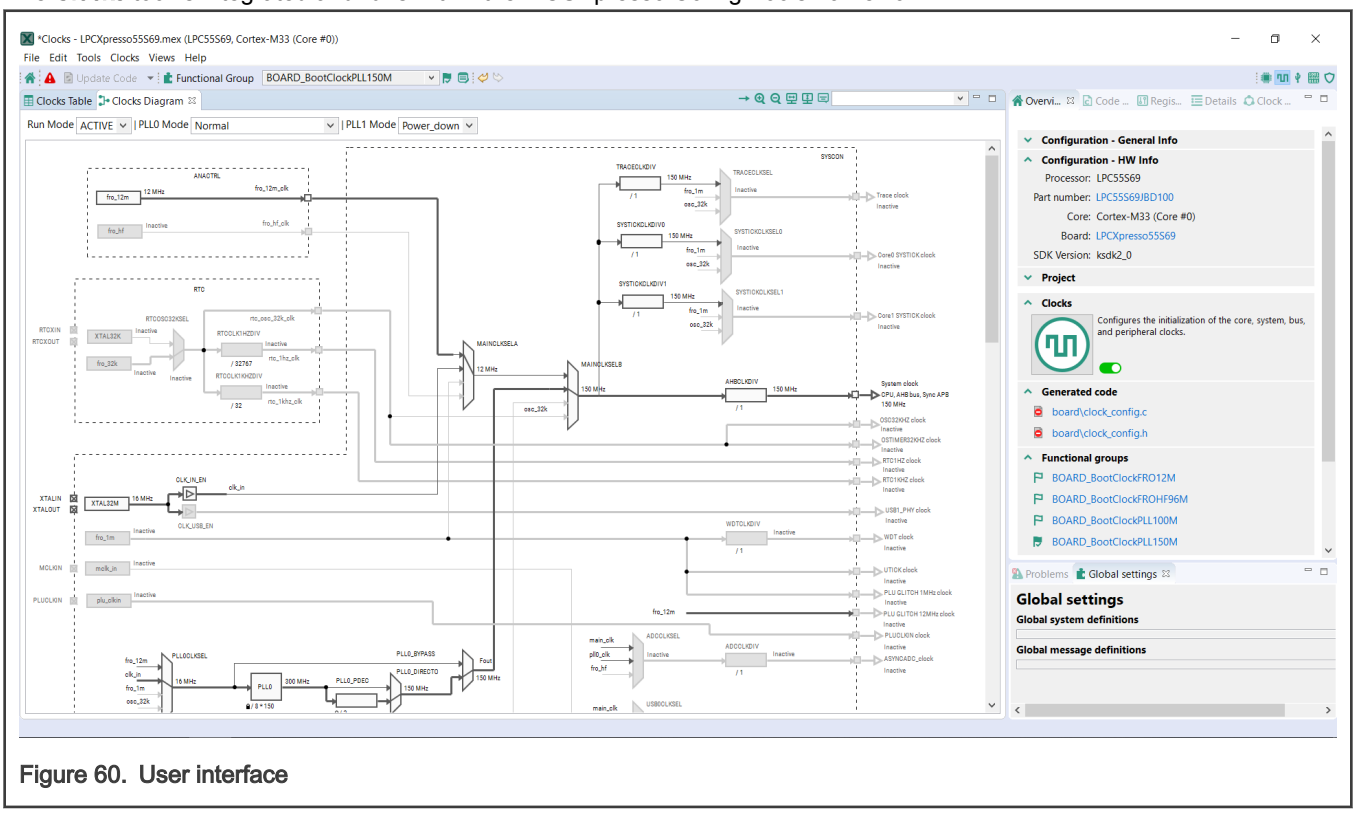


Figure 60. User interface

4.3 Clock configuration

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

There can be one or more clock configurations handled by the Clocks tool. The default clock configuration is created with the name "*BOARD_BootClockRUN*". Multiple configurations means multiple options are available for the processor initialization.

NOTE

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

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

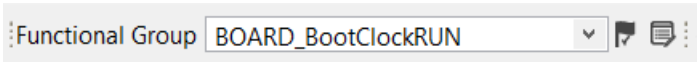


Figure 61. Default clock configuration

NOTE

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

4.4 Global settings

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

NOTE

Global settings can be changed at any time.

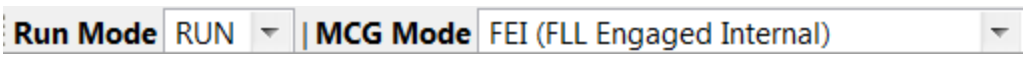


Figure 62. Global settings

4.5 Clock sources

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

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

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

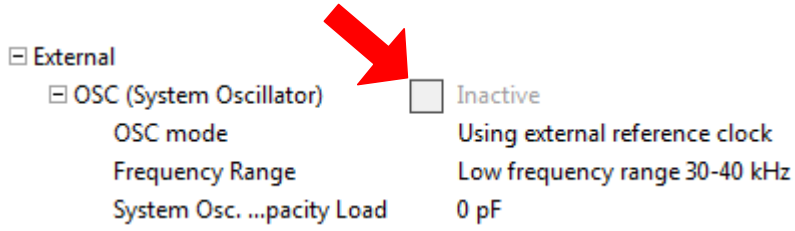


Figure 63. External clock source configuration

NOTE

Some clock sources remain inactive even though the checkbox is checked. This is because the clock sources functionality depends on other settings like power mode or additional enable/disable setting options. You can hover the cursor on the setting to see a tooltip with information on the element and possible limitations/options.

4.6 Setting states and markers

The following states, styles, and markers reflect the information shown in the settings' rows in the settings tables (clock sources, output, details or individual).

Table 16. Setting states and markers

State/Style/ Marker	Icon	Description
Error marker		Indicates that there is an error in the settings or something related to it. See the tooltip of the setting for details.
Warning marker		Indicates that there is a warning in the settings or something related to it. See the tooltip of the setting for details.
Lock icon		Indicates that the settings (that may be automatically adjusted by the tool) are locked to prevent any automatic adjustment. If the setting can be locked, they are automatically locked when you change the value. To add/remove the lock manually, use the pop-up menu command **Lock/Unlock**. **NOTE** The clock element settings that cannot be automatically adjusted by the tool keep their value as is and do not allow locking. These are: clock sources, clock selectors and configuration elements.
Yellow background		Indicates that the field is directly or indirectly changed by the previous user action.
Gray text		Indicates that the value of setting does not actively influence the clock. It is disabled or relates to an inactive clock element. For example, on the clock path following the unavailable clock source or disabled element. The frequency signal also show the text "inactive" instead of frequency. The value is also gray when the value is read-only. In such a state it is not possible to modify the value.

4.7 Frequency settings

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

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

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

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

Figure 64. Setting the core clock frequency

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

Figure 65. Tool attains the required frequency

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

Figure 66. Tool encounters problem

The frequency value in square brackets [ ] indicates the value that the tool is actually using in the calculations instead of the value that has been requested.

NOTE

You can edit or set requirements only for the clock source and the output frequencies. The other values can be adjusted only when no error is reported.

4.7.1 Pop-up menu commands

- **Lock/Unlock** – Removes a lock on the frequency which enables the tool to change any valid value that satisfies all other requirements, limits, and constraints.
- **Find Near Valid Value** – Tries to find a valid frequency that lies near the specified value, in case the tool failed in reaching the requested frequency.

Figure 67. Pop-up menu commands

4.7.2 Frequency precision

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

Figure 68. Frequency precision

4.8 Dependency arrows

In the **Clocks Table** view, the area between the clock sources and the clock output contains arrows directing the clock source to outputs. The arrows lead from the current clock source used for the selected output into all outputs that are using the signal from the same clock source. This identifies the dependencies and the influences when there is change in the clock source or elements on a shared clock path.

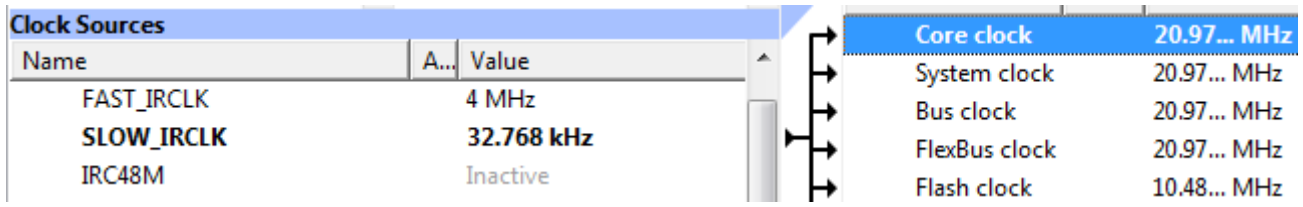


Figure 69. Dependency arrows

4.9 Details view

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

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

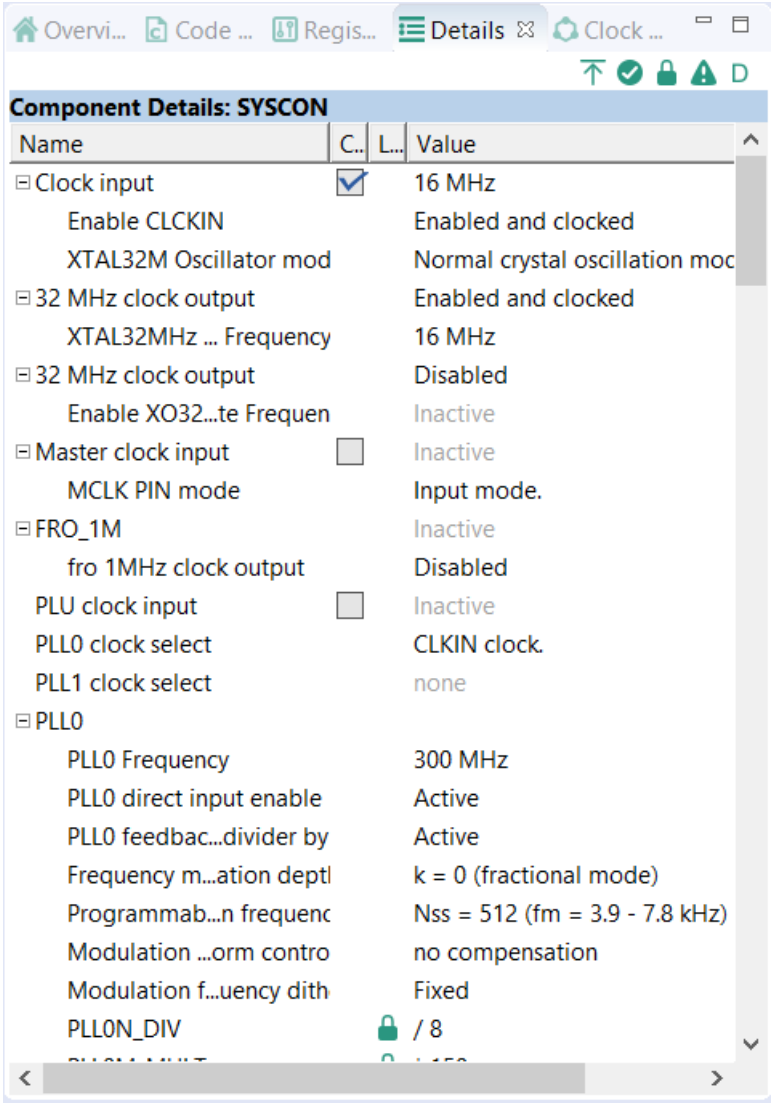


Figure 70. Details view

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

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

4.10 Clocks diagram

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

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

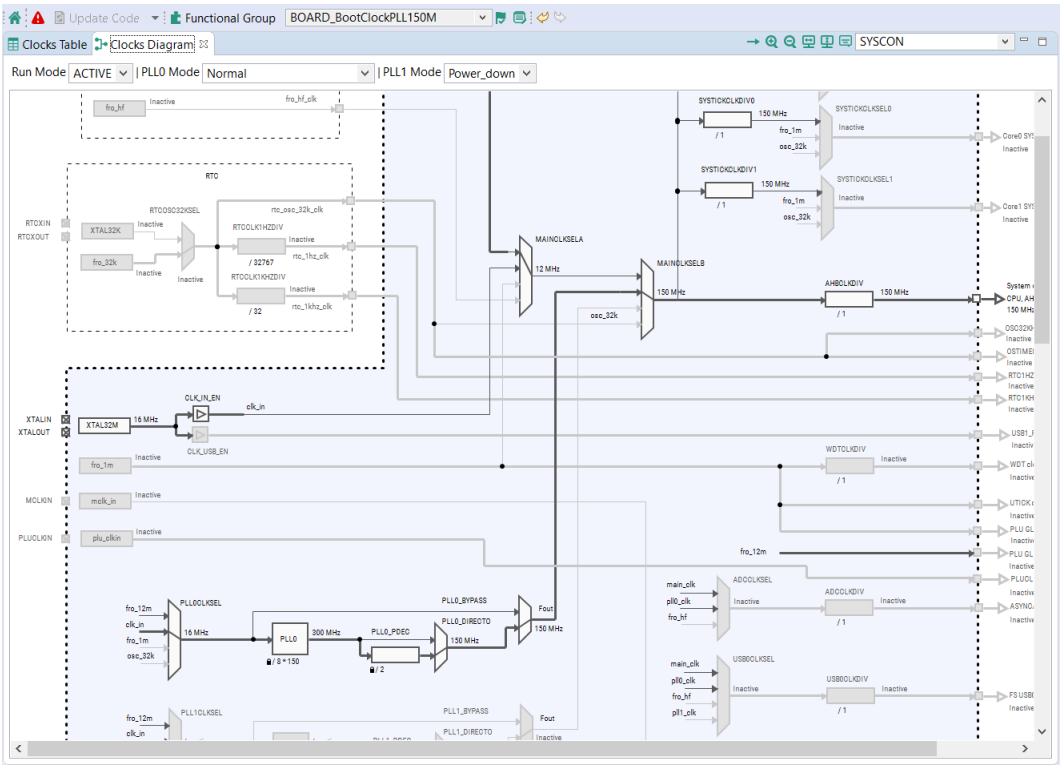


Figure 71. Clocks diagram

4.10.1 Mouse actions in diagram

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

- **Position the mouse cursor on the element** to see the tooltip with the information on the clock element such as status, description, output frequency, constraints, and enable/disable conditions.

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

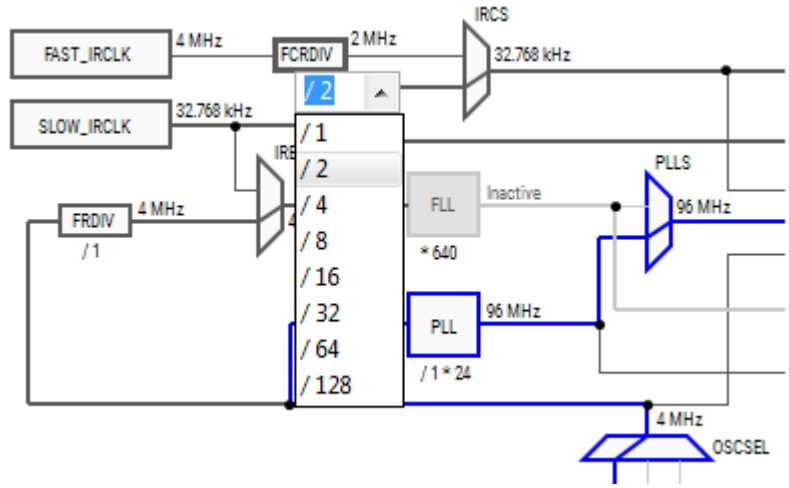


Figure 72. Clocks mouse actions in diagram

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

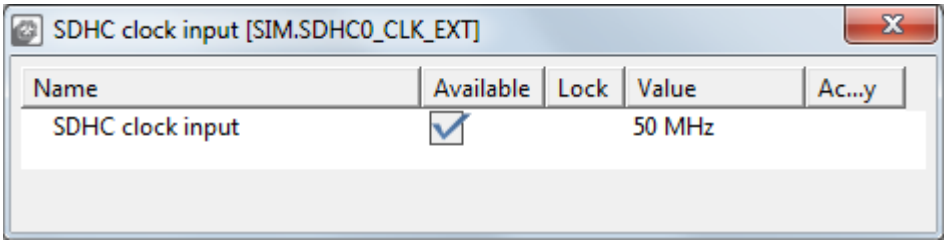


Figure 73. Floating view

4.10.2 Color and line styles

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

The color and line styles can indicate:

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

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

4.10.3 Clock model structure

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

The following are the clock model elements.

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

Figure 74. Clock source

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

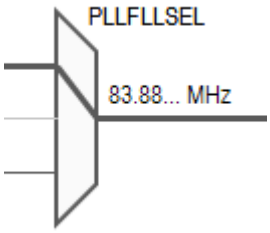


Figure 75. Clocks selector

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

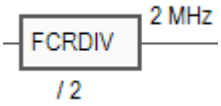


Figure 76. Prescaler

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

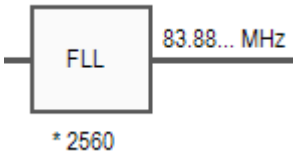


Figure 77. Frequency Locked Loop

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

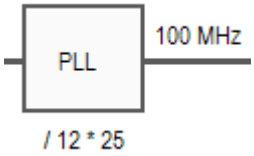


Figure 78. Phase Locaked Loop

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

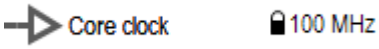


Figure 79. Clock output

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

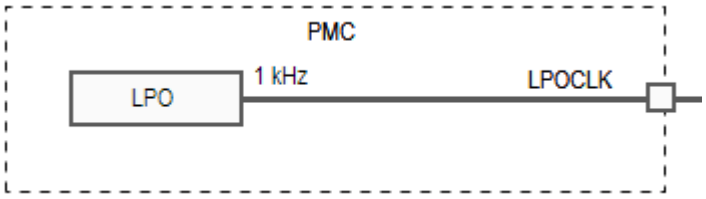


Figure 80. Clock component

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

4.11 Clocks menu

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

Table 17. Clocks menu

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

4.12 Troubleshooting problems

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

You may encounter any of the following problems:

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

The following are some tips to troubleshoot encountered problems.

1. Find the elements and settings with marked errors in the diagram or tables and see the details in the tooltip.
2. Start with only one locked frequency and let the tool find and calculate other ones. After you are successful you can add more.
3. Go through the locked outputs, if there are any, and verify the requirements (possible errors in the required frequency, wrong units, and so on).
4. If you are OK to have a near around of the requested value, right-click and from the pop-up menu select **Clock output > Find near value**.
5. If you cannot reach the values you need, see the clock paths leading to the clock output you want to adjust and check the selectors if it's possible to switch to another source of clock.
6. Try to remove locks by selecting **Clocks > Unlock All Settings**. In case many changes are required, you can simply reset the model to the default values and start from the beginning. To reset, select **Clocks > Reset to processor defaults**.

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

- **Show problem** - Shows the problem in the **Clocks Diagram** view If one the solutions are possible then the pop up is extended by:
  - **Remove lock** - Removes the lock from erroneous element.
  - **Find Near value** - Finds the nearest value.

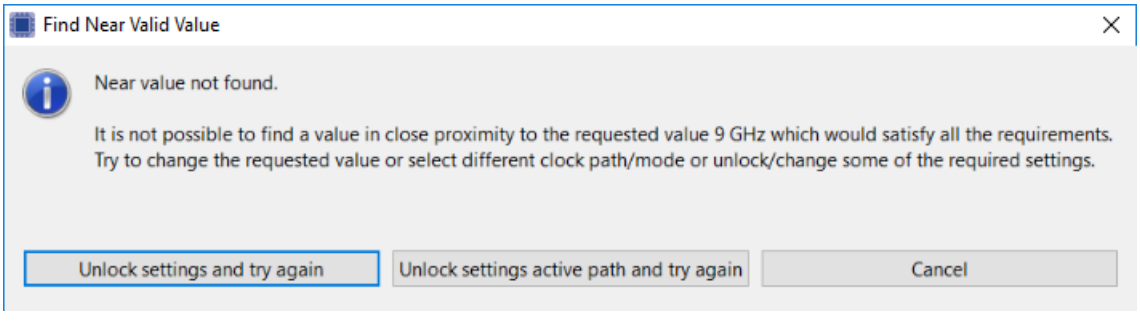


Figure 81. Find Near Value Dialog

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

4.13 Code generation

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

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

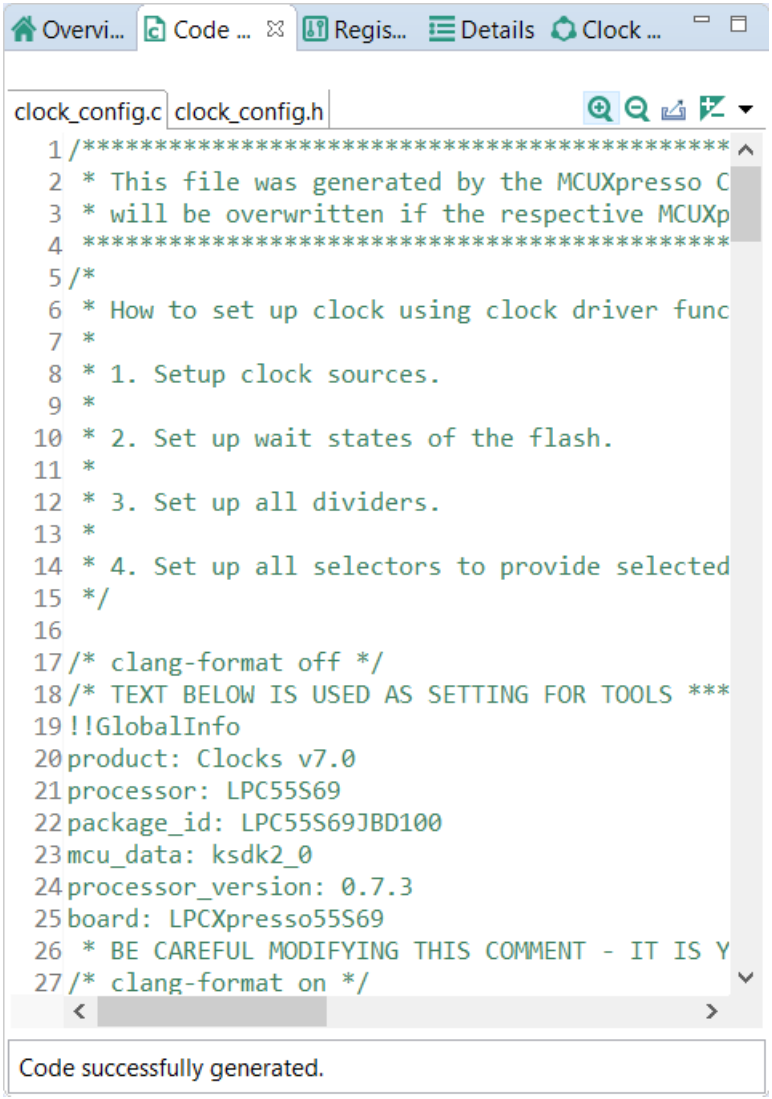


Figure 82. Code Preview

4.13.1 Working with the code

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

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

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

4.14 Clock Consumers view

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

NOTE

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

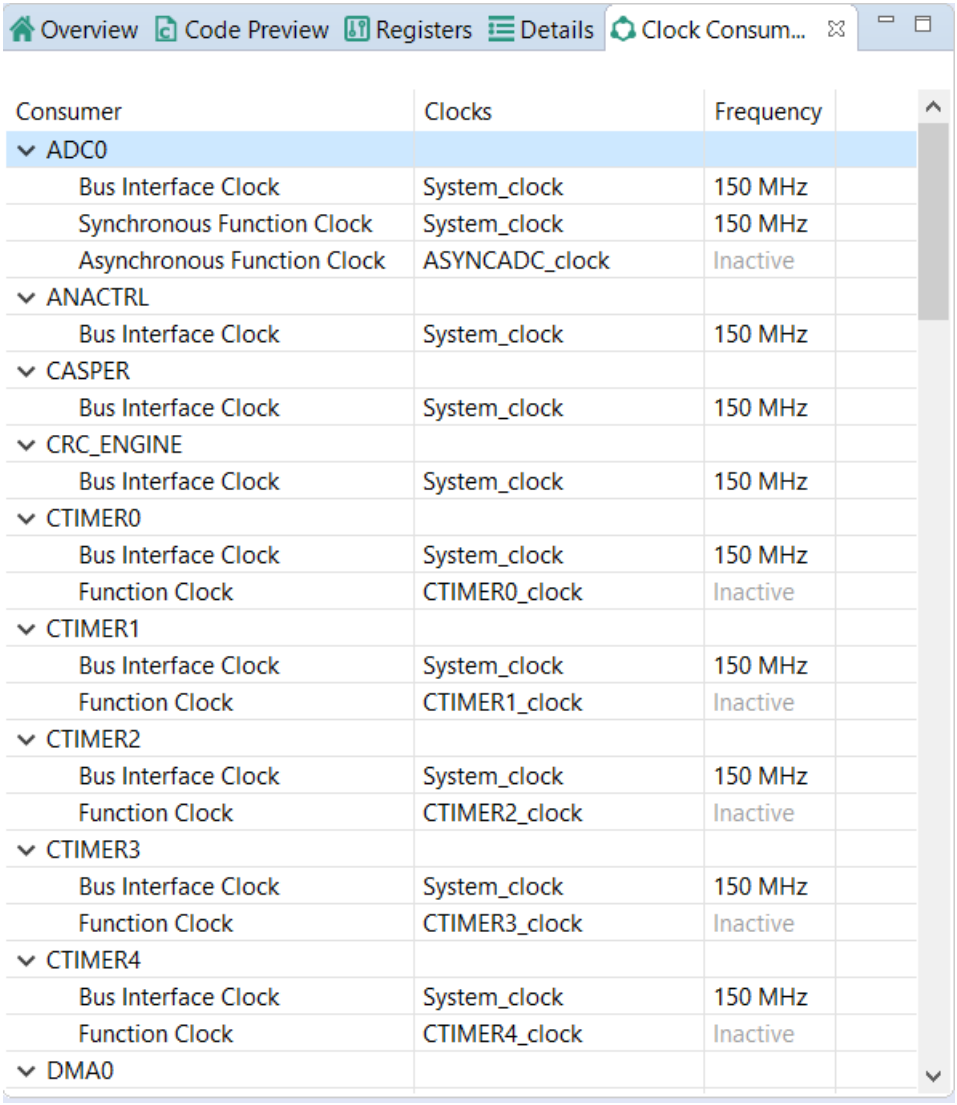


Figure 83. Clock Consumers view

Chapter 5
Peripherals Tool

5.1 Features

The Peripherals tool features:

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

5.2 Basic terms and definitions

Table 18. Terms and definitions

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

5.3 Workflow

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

1. In the **Peripherals** view, select the peripheral instance you would like to configure (use the checkbox).
2. In case more components are available for use by the peripheral, the **Select component** dialog appears. The dialog displays the list of suitable configuration components for the selected peripheral matching the SDK driver for the selected processor.
3. Select the component you want to use and click **OK**.

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

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

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

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

6. You can use the **Update Code** button from the toolbar. Alternatively, you can export the source code by selecting **File>Export...** from the **Menu bar**.

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

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

5.4 User interface

Figure 84. User interface

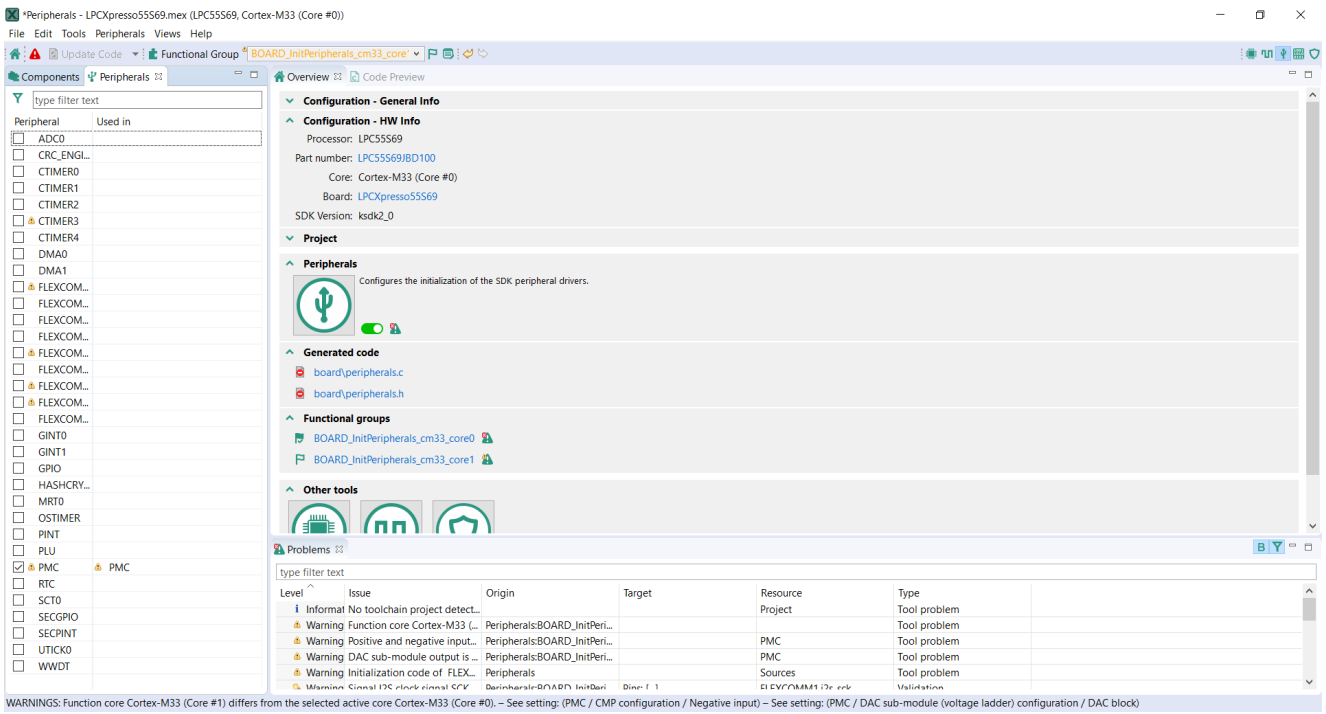
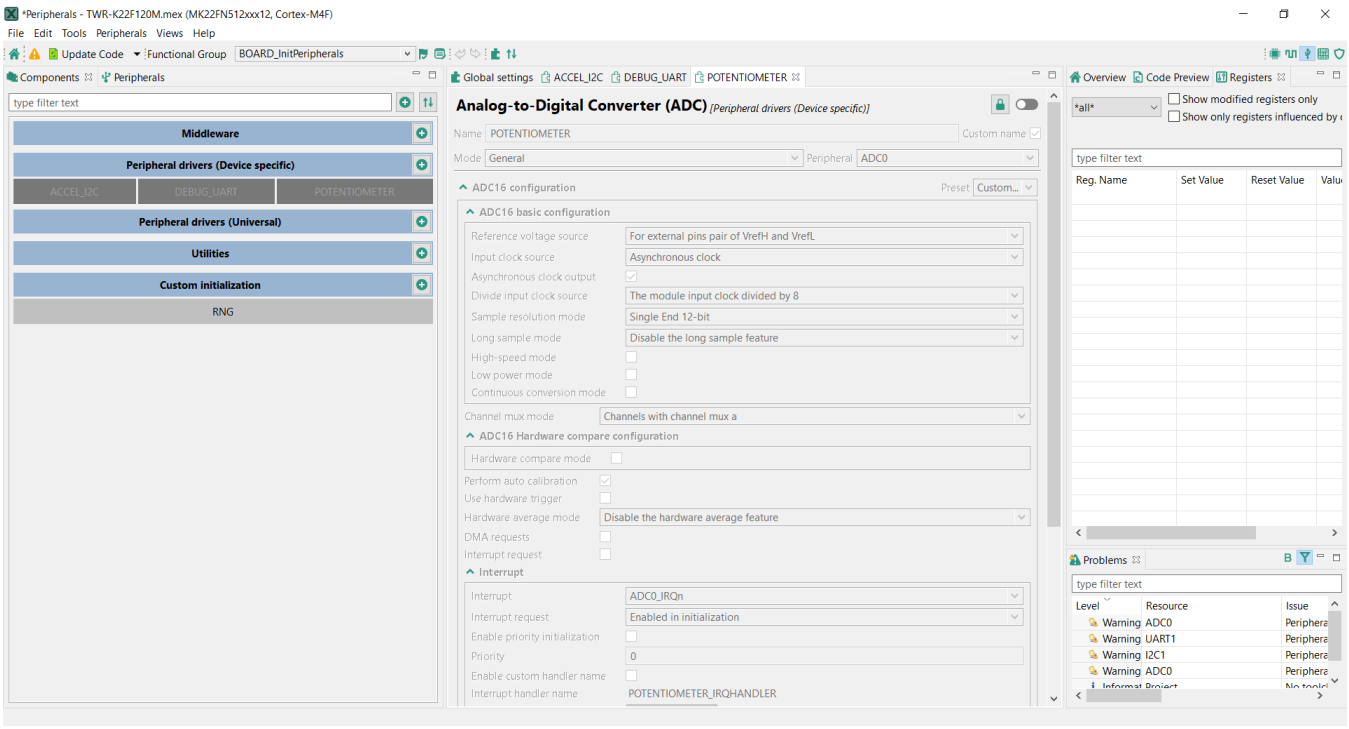


Figure 85. Peripheral view

Figure 86. Components view



5.4.1 Toolbar (Peripherals)

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

Table 19. Toolbar

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

NOTE

For details on other items, refer to the Toolbar chapter.

5.4.1.1 Object Missing

This object is not available in the repository.

5.4.2 Components view

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

The view highlights configuration components based on their status.

Table 20. Component status

Status	Color highlighting
Enabled	Light gray.
Enabled/with warning	Light gray with the alert symbol.

Table continues on the next page...

Table 20. Component status (continued)

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

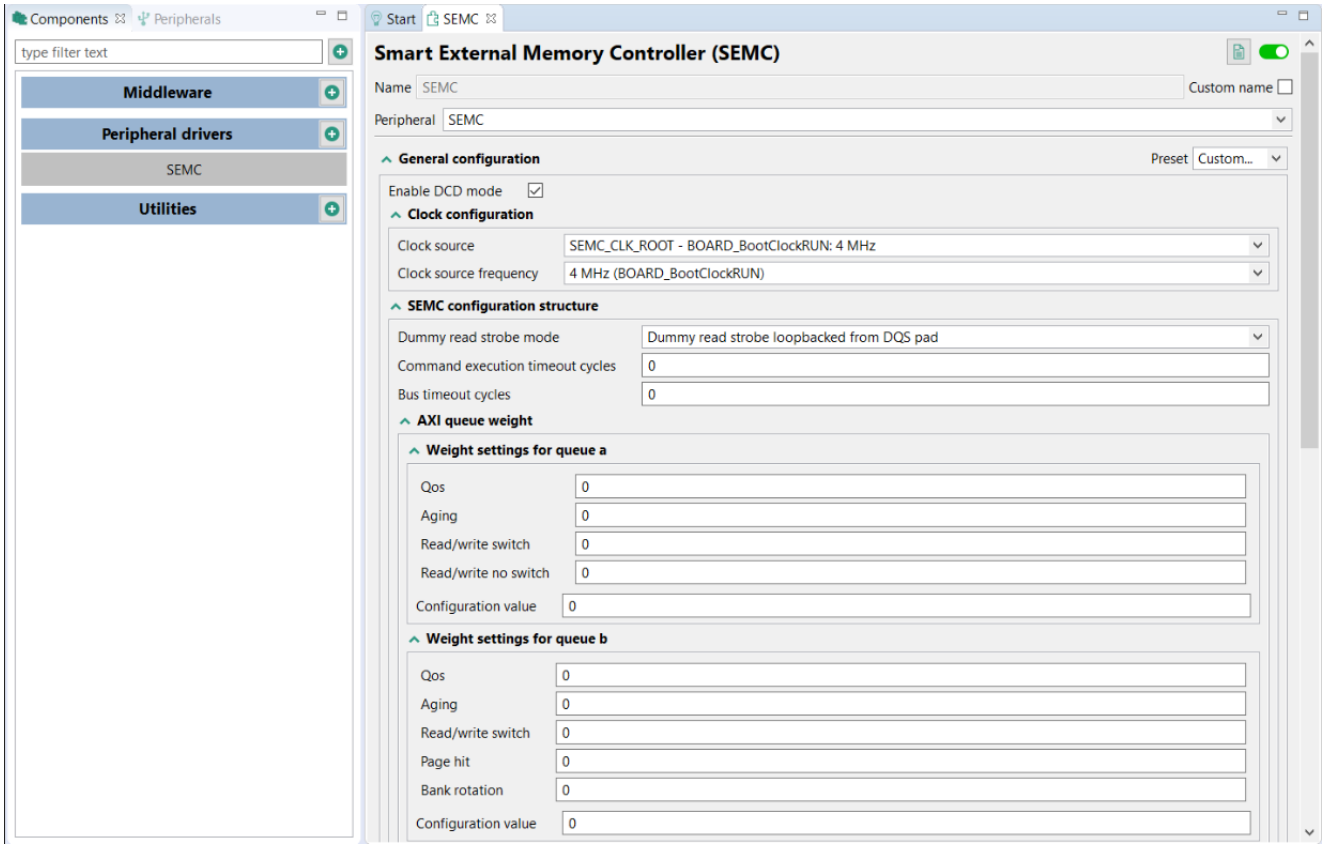


Figure 87. Components view

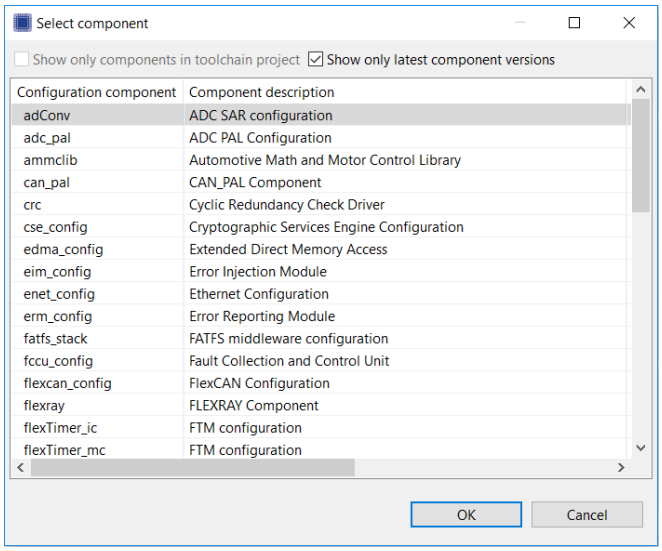
In the **Components** view, you can perform several actions.

Table 21. Components view actions

Display configuration-component information	Point the mouse cursor at the configuration component to display general configuration-component information.
Open the Settings Editor of the configuration component	Left-click the configuration component to open its **Settings Editor**.
Add new configuration components	Left-click the **+** button and select from the list to add a new component. In the **Select component** dialog, you can filter the list to show only toolchain-project-relevant, or latest version components. You can also click the **+** buttons next to **Middleware/Peripheral drivers/Other** categories to add new components in them directly.

Table continues on the next page...

Table 21. Components view actions (continued)

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

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

Table 22. Shortcut menu options

Option	Description
Open	Open the configuration component in the **Settings Editor**.
Open in another view	Duplicates the configuration component in the **Settings Editor**.
Edit comment	Create/Edit custom notes for the configuration component.
Lock/Unlock editing of component instance	Lock/Unlock the editing of the component instance.
Documentation	Display the documentation of the configuration component, if available.
Remove	Remove the component from configuration. **NOTE** If the component has any global settings, a dialog appears prompting you to confirm the removal. If the component doesn't have any global settings, the component is deleted after removing the last instance.
Migrate	Migrate the component to a different component type or to a component with a newer driver version.
Enable/Disable	Enable/Disable the configuration component
Move to	Choose from available functional groups to move the configuration component to.
Copy to	Choose from available functional groups to copy the configuration component to.

5.4.3 Peripherals view

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

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

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

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

Disable a component instance by deselecting it.

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

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

Table 23. Shortcut menu options

Option	Description
Open	Open the component instance in the **Settings Editor**.
Open in another view	Duplicate the component instance in the **Settings Editor**.
Add	Add a component instance to the peripheral.
Edit comment	Create/Edit custom notes for the component instance.
Lock/Unlock editing of component instance	Lock/Unlock the editing of the component instance.
Documentation	Display the documentation of the component instance, if available.
Remove	Remove the component instance from configuration. If more instances are in use, a confirmation window will allow you to select which instance you want to remove.
Migrate	Migrate the component to a different component type or to a component with a newer driver version.
Enable/Disable	Enable/Disable the component instance.
Move to	Choose from available functional groups to move the component instance to.
Copy to	Choose from available functional groups to copy the component instance to.
Initialized in user code	Mark the peripheral as configured by user code (available on not configured peripherals).

5.4.4 Settings Editor

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

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

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

5.4.4.1 Quick selections

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

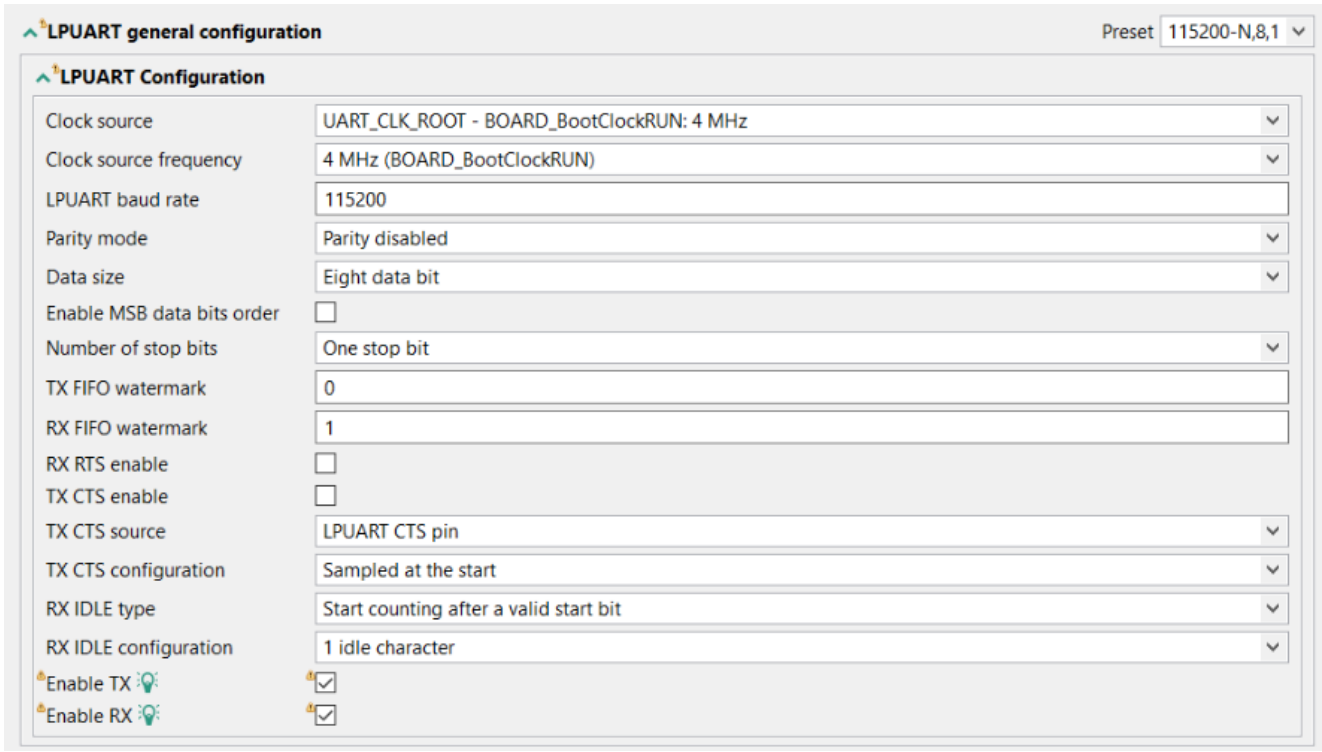


Figure 88. Quick selection example

5.4.4.2 Settings

Following setting types are available in the **Settings Editor**

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

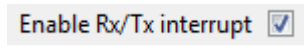


Figure 89. Boolean setting example

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

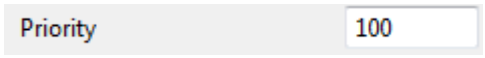


Figure 90. Integer/Float setting example

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

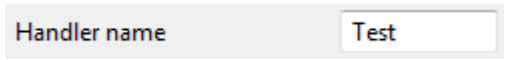


Figure 91. String setting example

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

Figure 92. Enumeration setting example

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

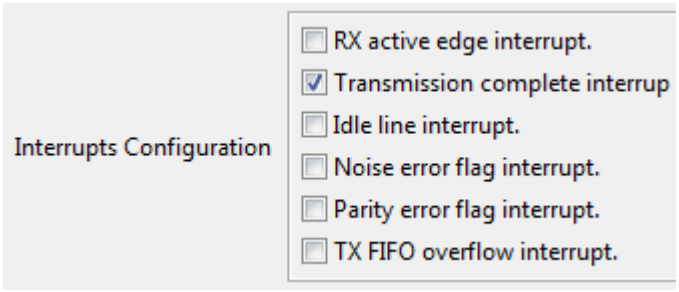


Figure 93. Set setting example

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

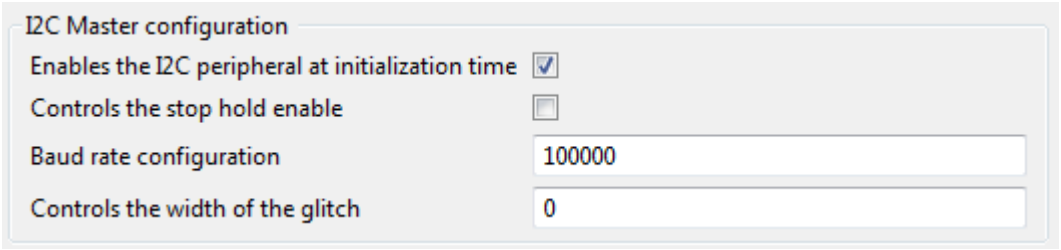


Figure 94. Structure setting example

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

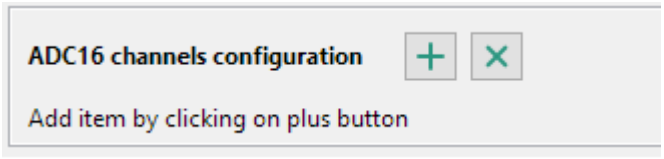


Figure 95. Array

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

NOTE

System clipboard is not used for this purpose.

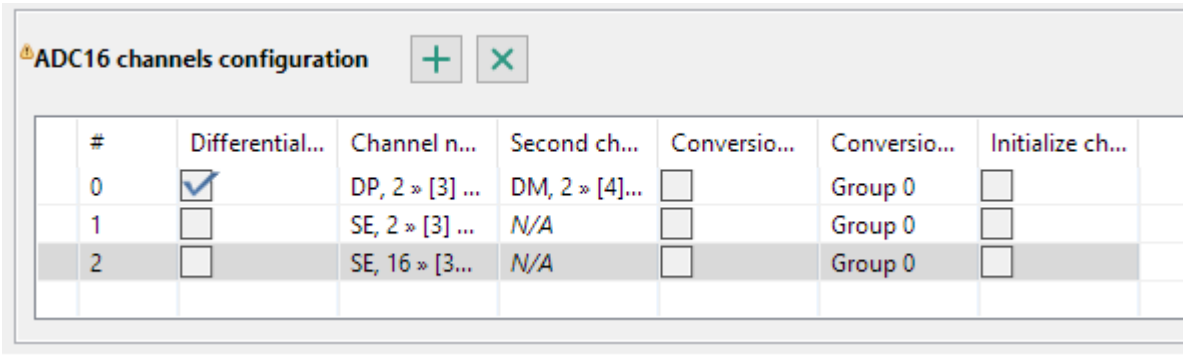


Figure 96. Array setting example

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

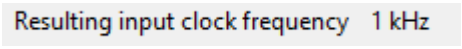


Figure 97. Array setting example

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

5.4.4.3 Settings Editor header

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

Figure 98. Settings Editor header

Table 24. Settings Editor header

Header item	Description
Description	Displays the configuration component title.
Name	Displays the component instance name. This name is used in the generated code in constants and function identifiers and is derived from the peripheral name. You can change it at any time by clicking the **Custom name** button and editing the field.
Mode	Displays the required usage for the component instance and influences available settings. Use the dropdown menu to change the mode (where applicable).
Peripheral	Displays the name of the peripheral to be associated with the component instance. Use the dropdown menu to change it.
Documentation	Click the button to view configuration component-specific documentation in the **Documentation** view. Note that not all configuration components are documented, therefore not all setting headers contain the **Documentation** icon.

Table continues on the next page...

Table 24. Settings Editor header (continued)

Header item	Description
Lock editing	Click the button to lock/unlock component editing. Source code will still be generated.
Enable/disable component instance switch	Use the switch to enable or disable selected component instance. Note that by disabling the instance, you don't remove it from the tools configuration, but prevent its inclusion in the generated code.

5.4.5 Documentation view

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

NOTE

Not all components might have this option enabled.

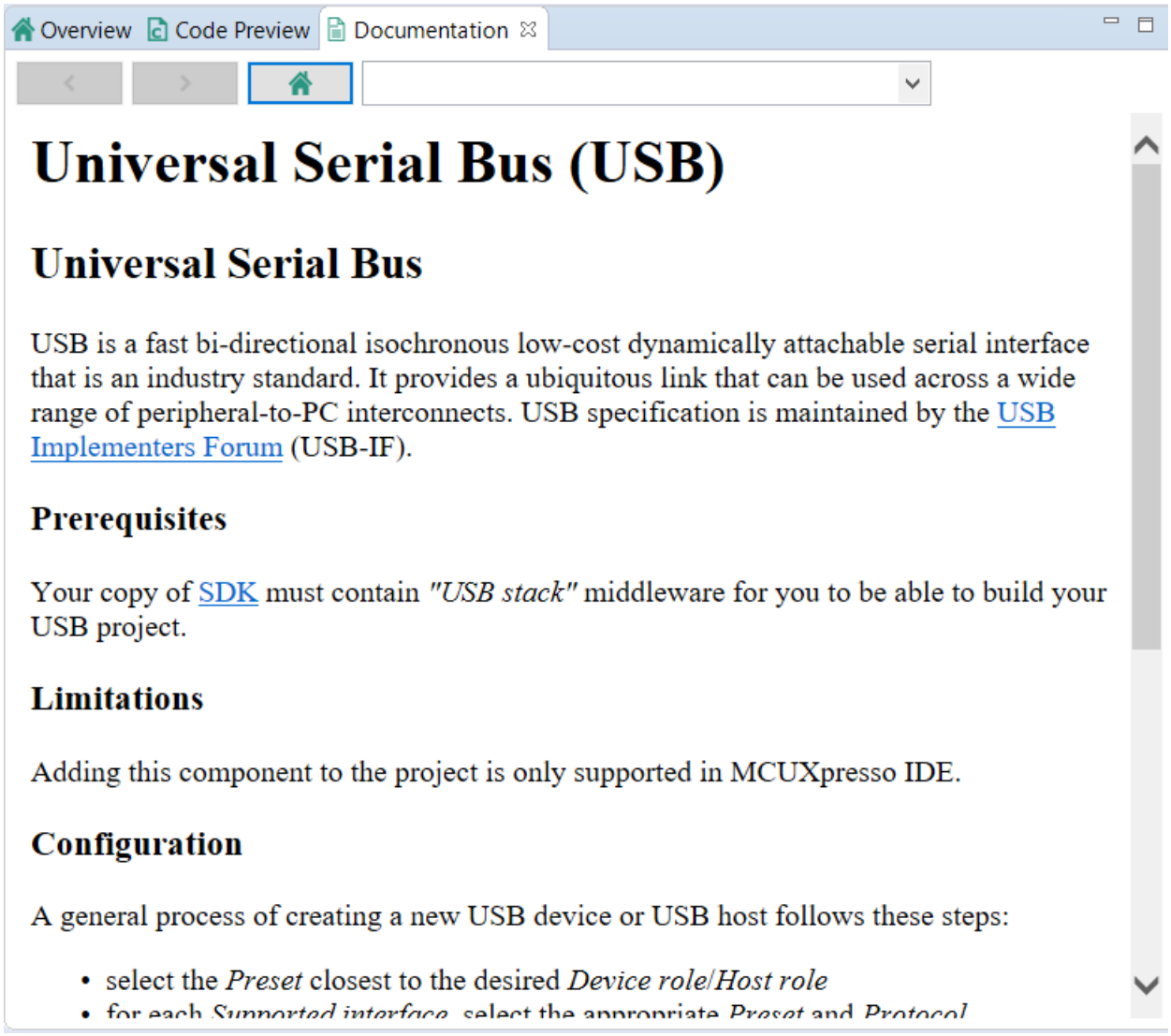


Figure 99. Documentation view

You can open the **Documentation** view in several ways:

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

5.5 SEMC Validation tool

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

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

NOTE

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

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

5.5.1 Validation view

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

To run validation tests, do the following:

1. Select the correct connection type and COM port in the **Connections** area. Alternatively, scan for available COM ports by clicking the **Scan for available COM ports** button.
2. Choose a scenario you wish to test in the **Scenarios** sub-view.
    a. To test the Phy configuration:
        i. Select **Firmware Init** > **Firmware Init test**.
    b. To perform a basic memory check by running Write-Read-Compare/Walking Ones/Walking Zeros:
        i. Select **Operational** > **Operational tests**.
        ii. Select **Choose Tests**.
        iii. Select the tests to perform and set the test parameters.
    c. To run stress tests:
        i. Select **Stress Tests** > **Stress tests**.
        ii. Select **Choose Tests** and set the test parameters.
    d. To run diags tests:
        i. Select **Diags** and choose between **Diags margin state/Diag Write Margin/Diag Read Margin**.
        ii. Select **Choose Tests** and set the test parameters.
3. To start validation, click the **Start Validation** button.
4. Observe the results in real-time in the **Results** sub-view.

5. Inspect the results in the **Summary** and **Logs** sub-views.
6. View the U-boot-compatible code in **Core Preview**. You can export the generated code for importing into U-boot by selecting the **Export** button.
7. Download DDRV reports using the **Configure and generate DDRV reports** button.

5.6 Problems

The tool validates the settings and problems and errors are reported in the Problems view.

If there is an error related to the setting or component an error decorator is shown next to the element containing an error.

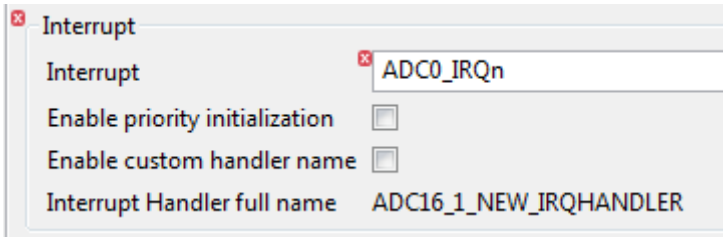


Figure 100. Error decorators

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

Figure 101. Quick fix 1

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

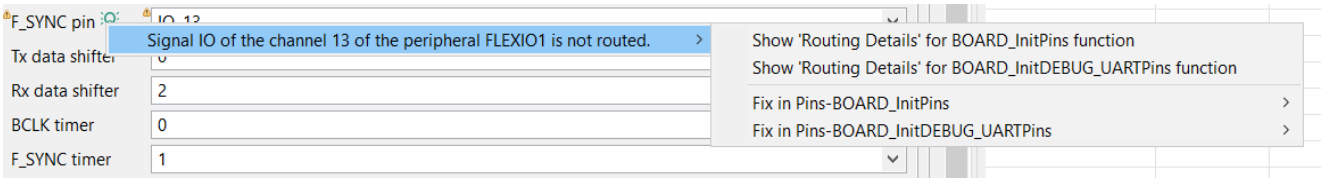


Figure 102. Quick fix 2

5.7 Code generation

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

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

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

- peripherals.c
- peripherals.h

NOTE

For multicore processors the peripherals.c/.h are generated for each core, containing functional groups associated with that core. This can be configured in functional group properties.

NOTE

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

These files contain initialization code for peripherals produced by selected configuration components including:

- Constants and functions declaration in header file.
- Initialized configuration structures variables (constants).
- Global variables for the user application that are used in the initialization. For example, handles and buffers.
- Initialization function for each configuration component.
- Initialization function for each functional group. The name of the function is the same as the functional group name. These functions include execution of all assigned components' initialization functions.
- Default initialization function containing call to the function initializing the selected functional group of peripherals.

NOTE

The prefixes of the global definitions (defines, constants, variables and functions) can be configured in the Properties of the functional group.

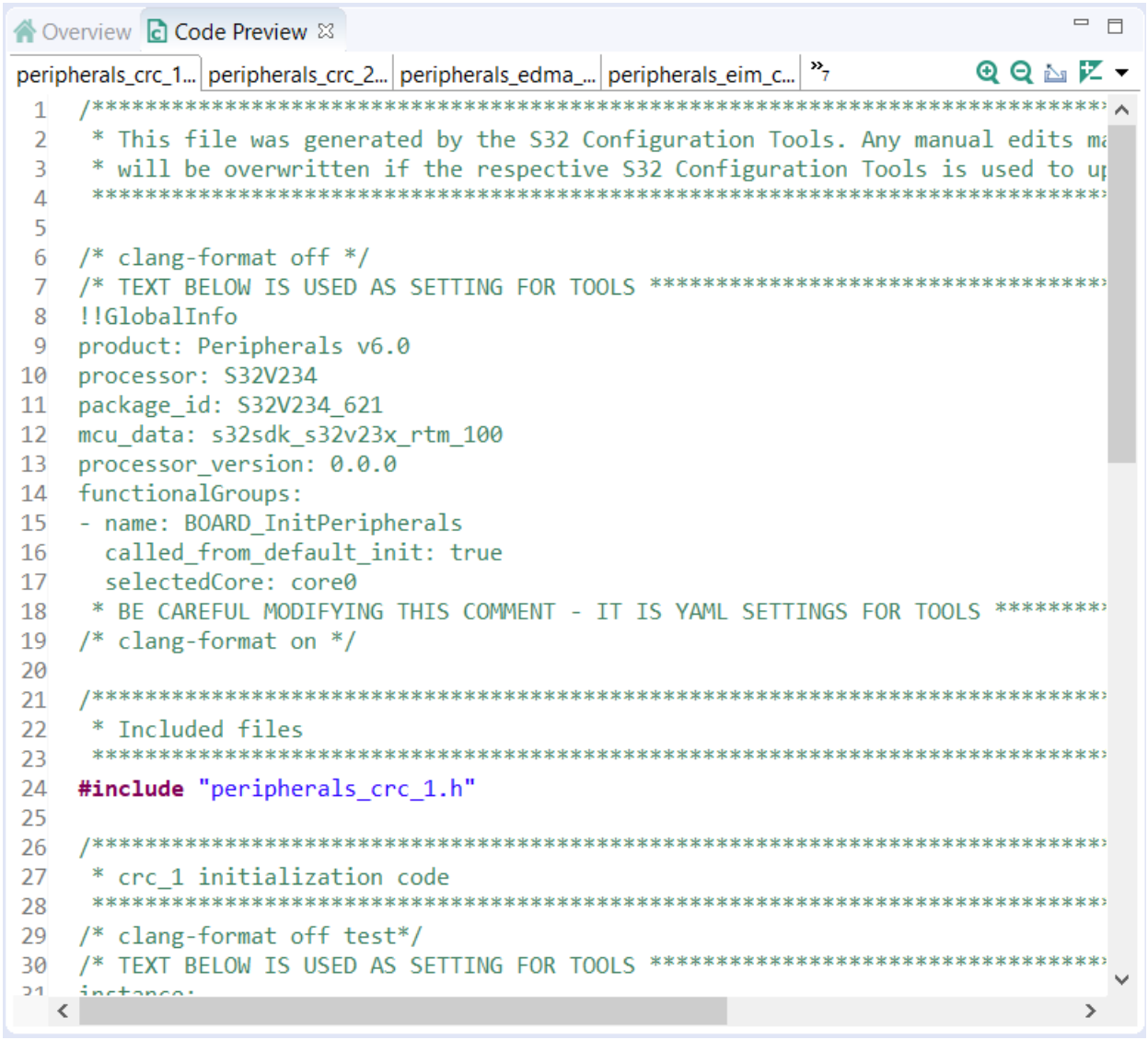


Figure 103. Code Preview

Chapter 6
Device Configuration Tool

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

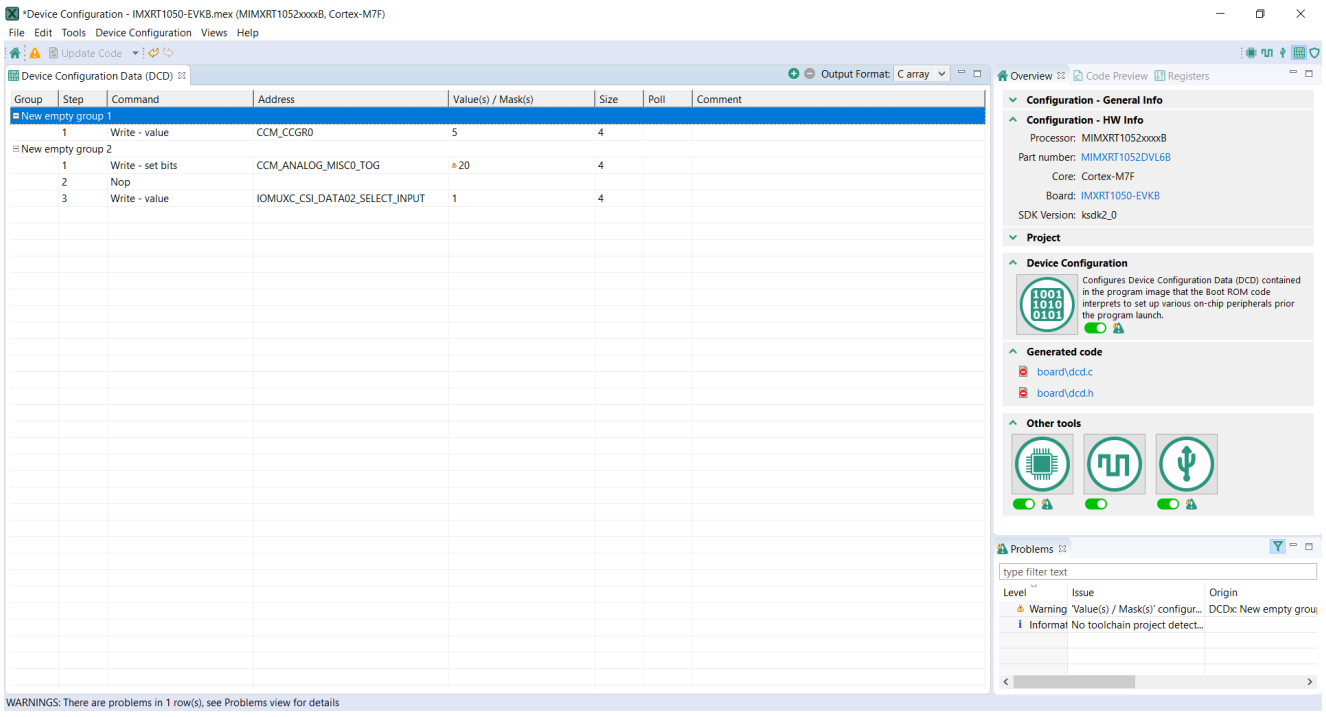


Figure 104. Device Configuration tool

6.1 Device Configuration Data (DCD) view

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

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

6.1.1 Device Configuration Data (DCD) view actions

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

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

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

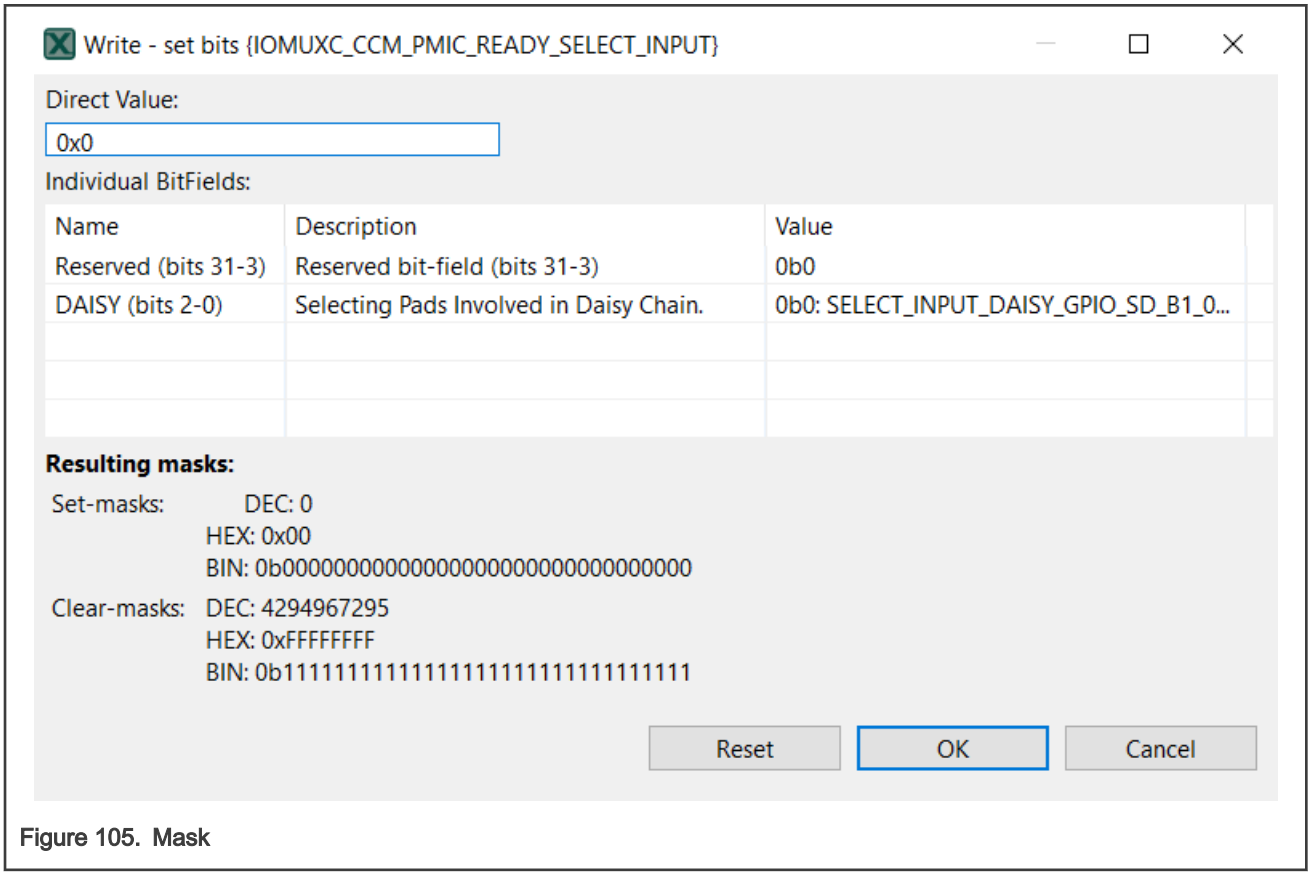


Figure 105. Mask

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

NOTE

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

Basic cell selection shortcuts are applicable.

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

6.2 Code generation

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

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

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

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

- dcd.c
- dcd.h

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

- dcd.bin

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

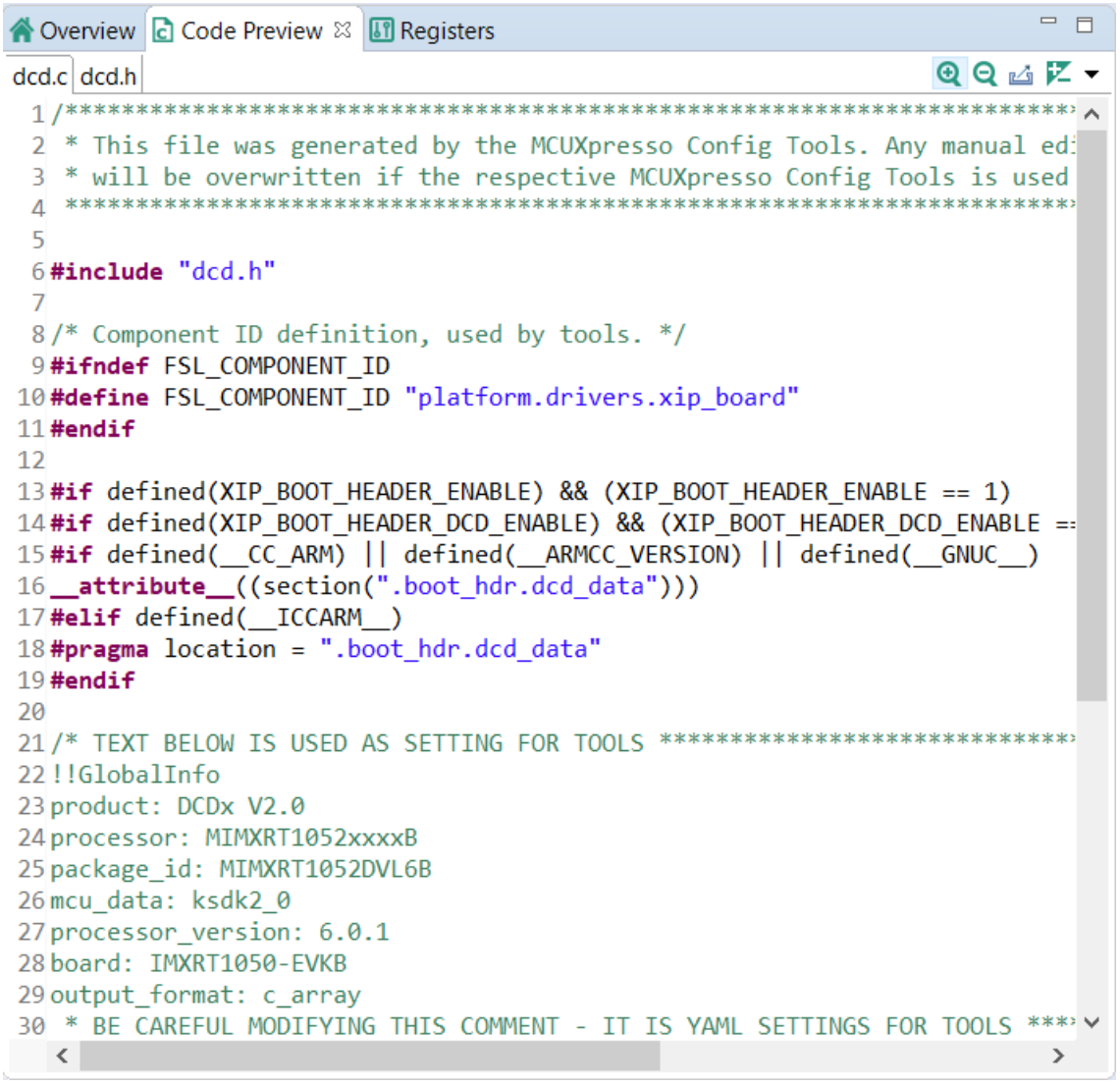


Figure 106. Code Preview

Chapter 7
Trusted Execution Environment Tool

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

You can set security policies of different parts of your application in the **Security Access Configuration** and its sub-views, and review these policies in the **Memory Attribution Map**, **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** sub-view of the **Security Access Configuration** view.

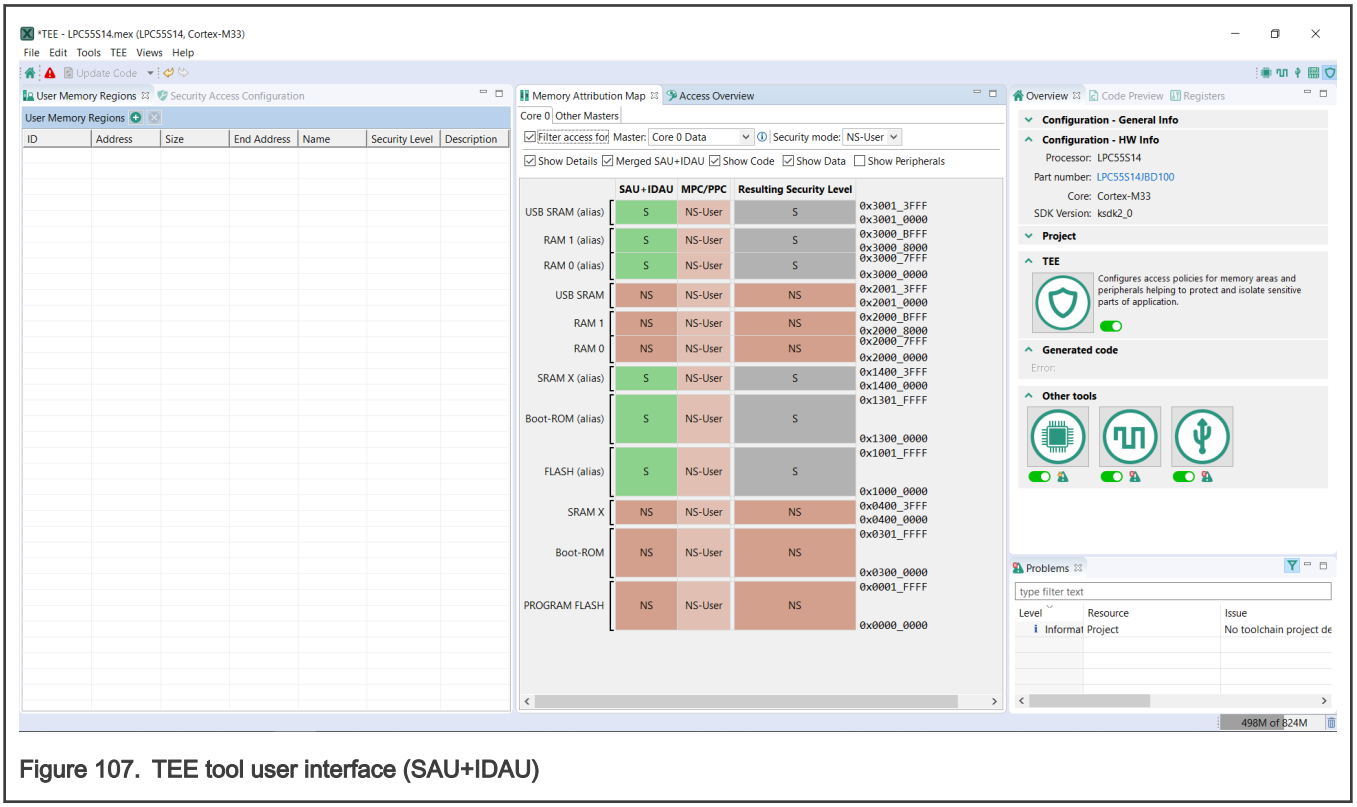


Figure 107. TEE tool user interface (SAU+IDAU)

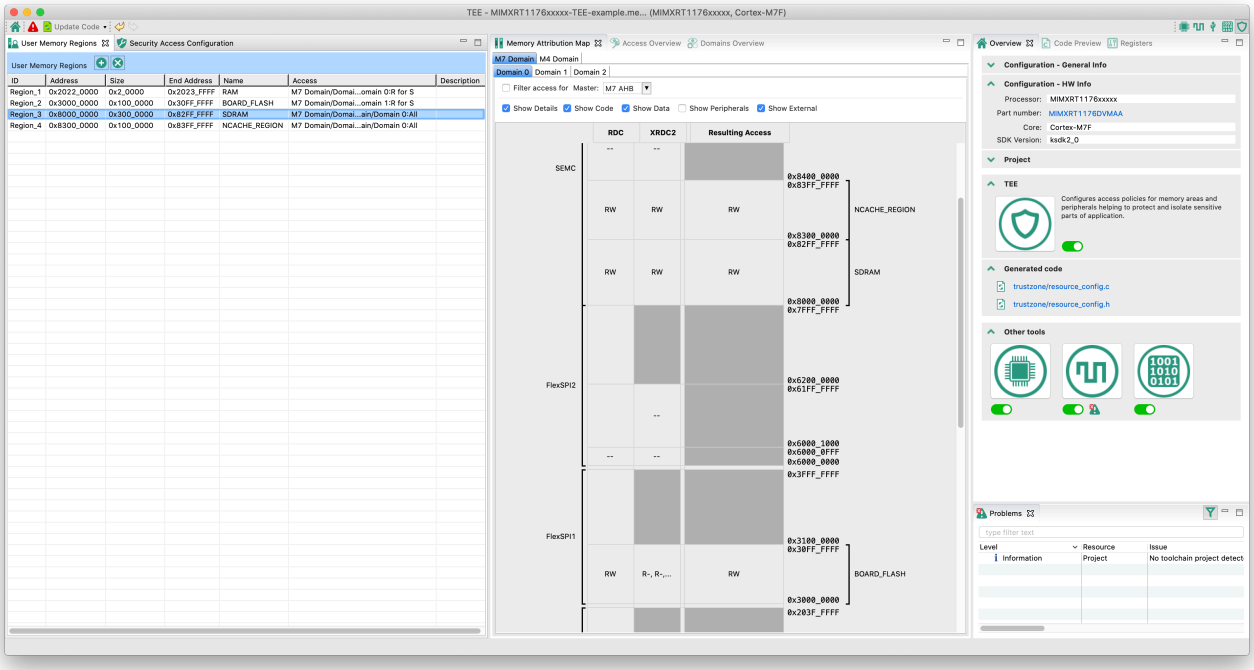


Figure 108. TEE tool user interface (RDC)

7.1 AHB 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 ARM TrustZone and AHB with security extension

Currently, following devices of this type are supported:

- LPC55Sxx
  - LPC55S69, LPC55S66
  - LPC55S16, LPC55S14
  - LPC55S06, LPC55S04
- RT6xx, RT5xx
  - MIMXRT685S, MIMXRT633S
  - MIMXRT595S, MIMXRT555S, MIMXRT533S

NOTE

Pre-production only.

7.1.1 User Memory Regions view

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

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

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

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

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

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

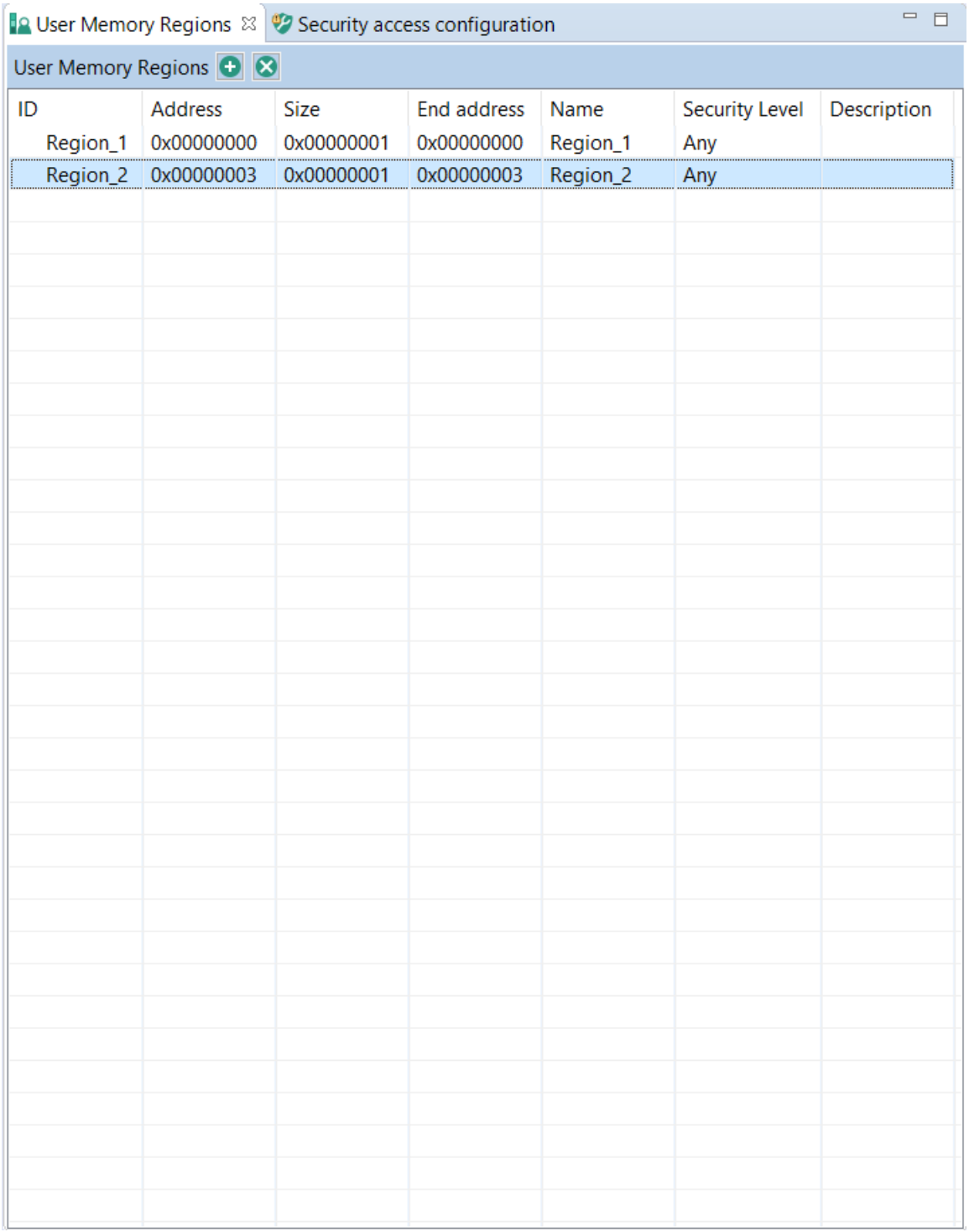


Figure 109. User Memory Regions

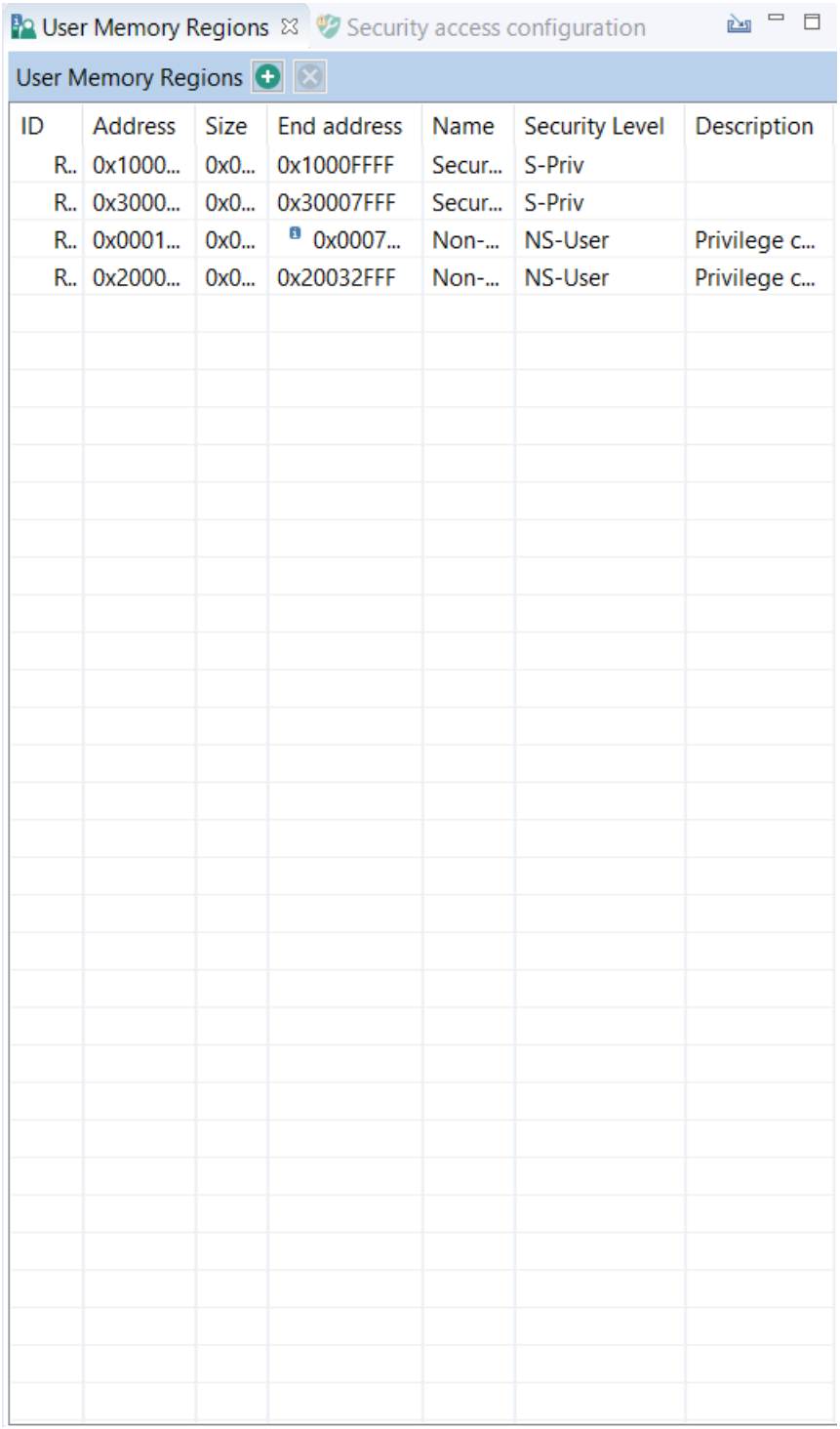


Figure 110. User Memory Regions

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

NOTE

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

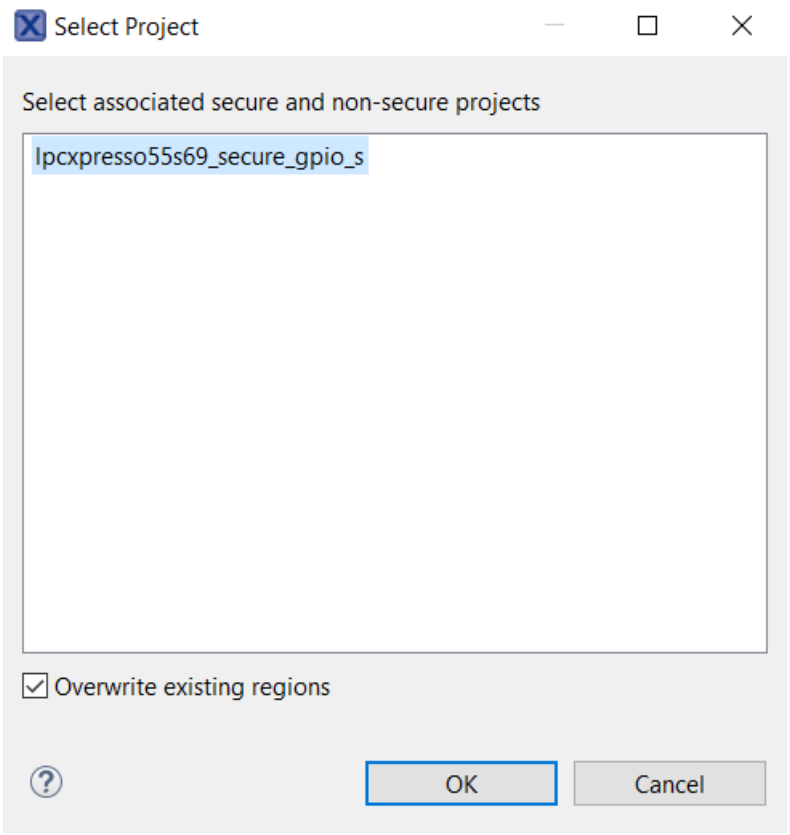


Figure 111. Select Project

7.1.2 Security Access Configuration view

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

7.1.2.1 SAU

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

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

You can choose between two security levels:

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

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

NOTE

This option is only available when SAU is disabled.

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

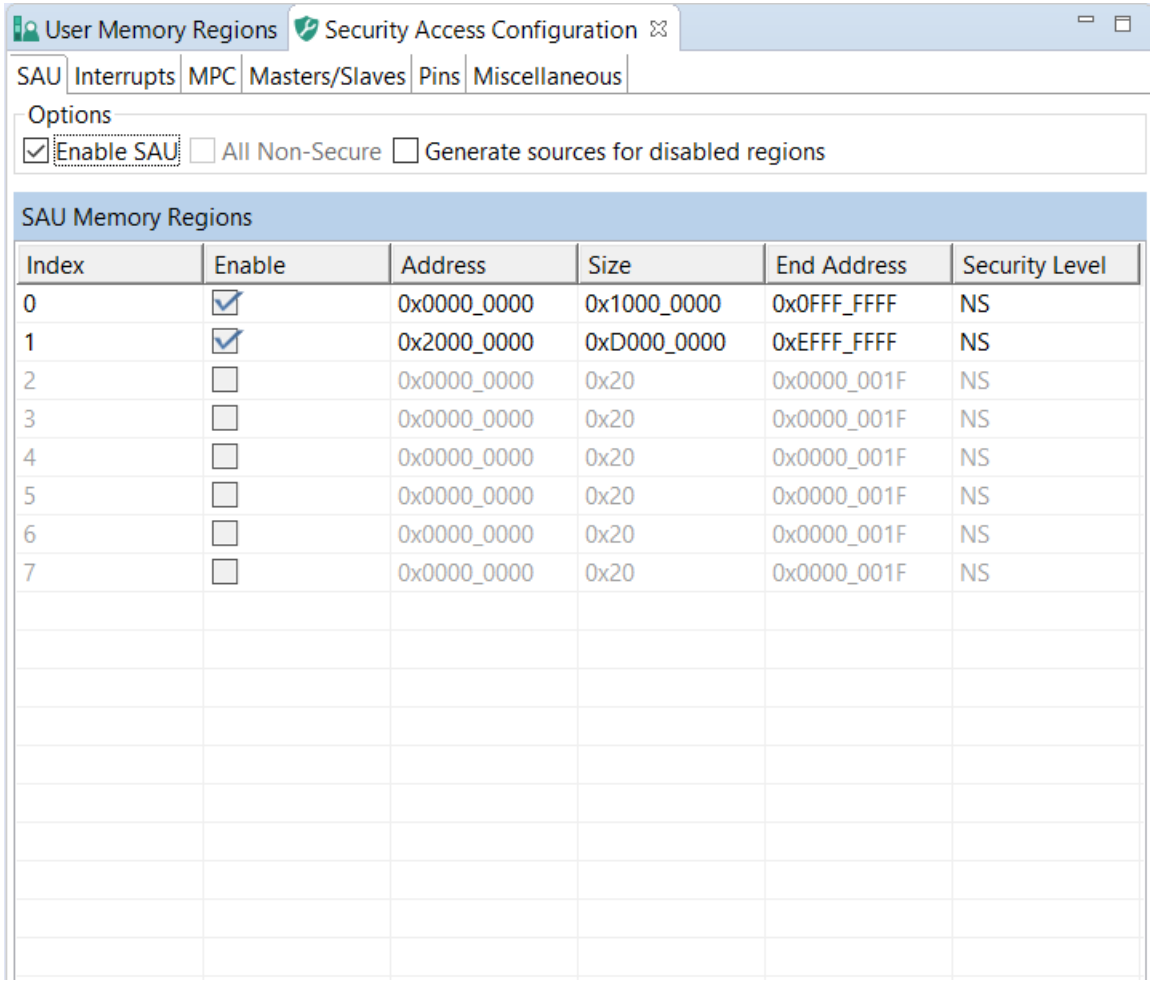


Figure 112. SAU/IDAU

7.1.2.2 Interrupts

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

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

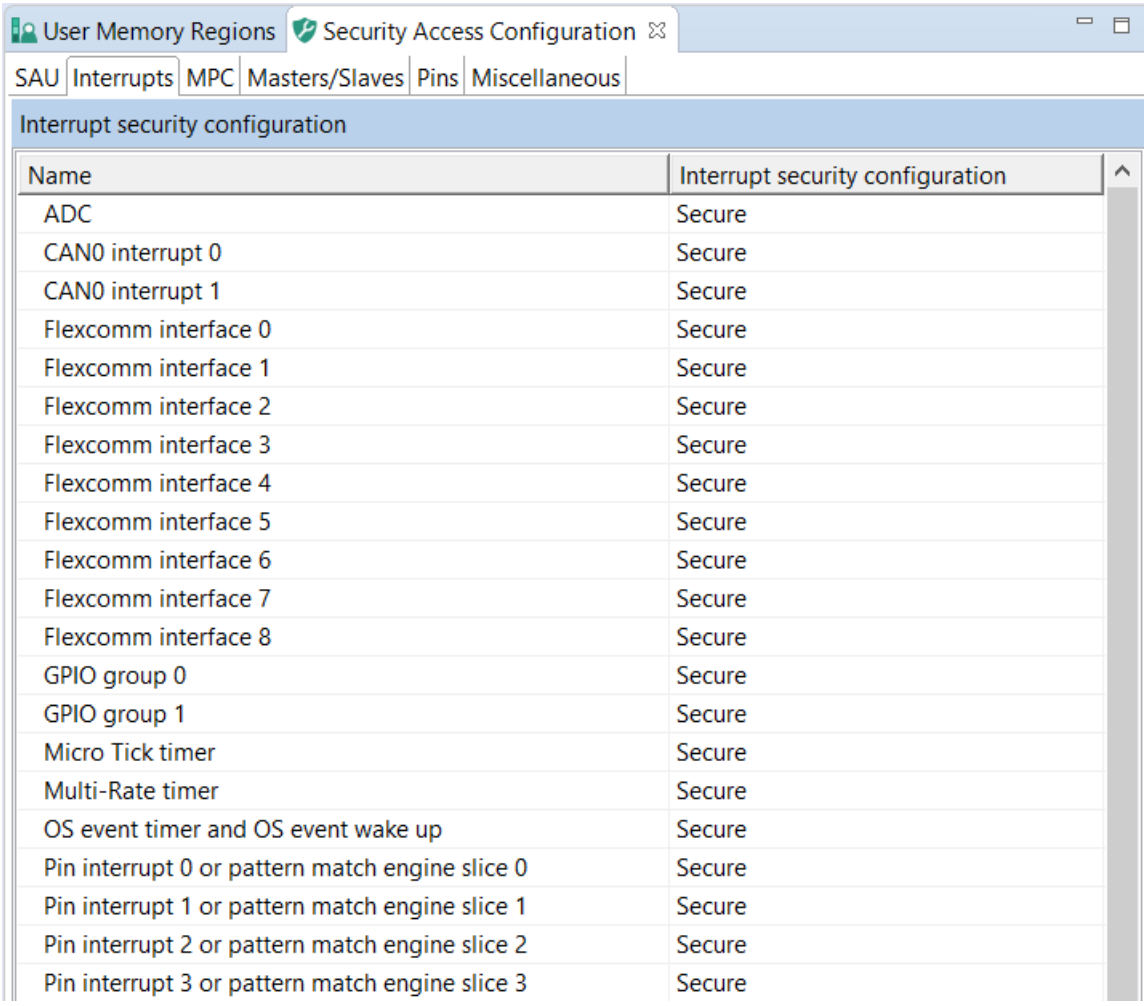


Figure 113. Interrupts

7.1.2.3 Secure/Non-secure MPU

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

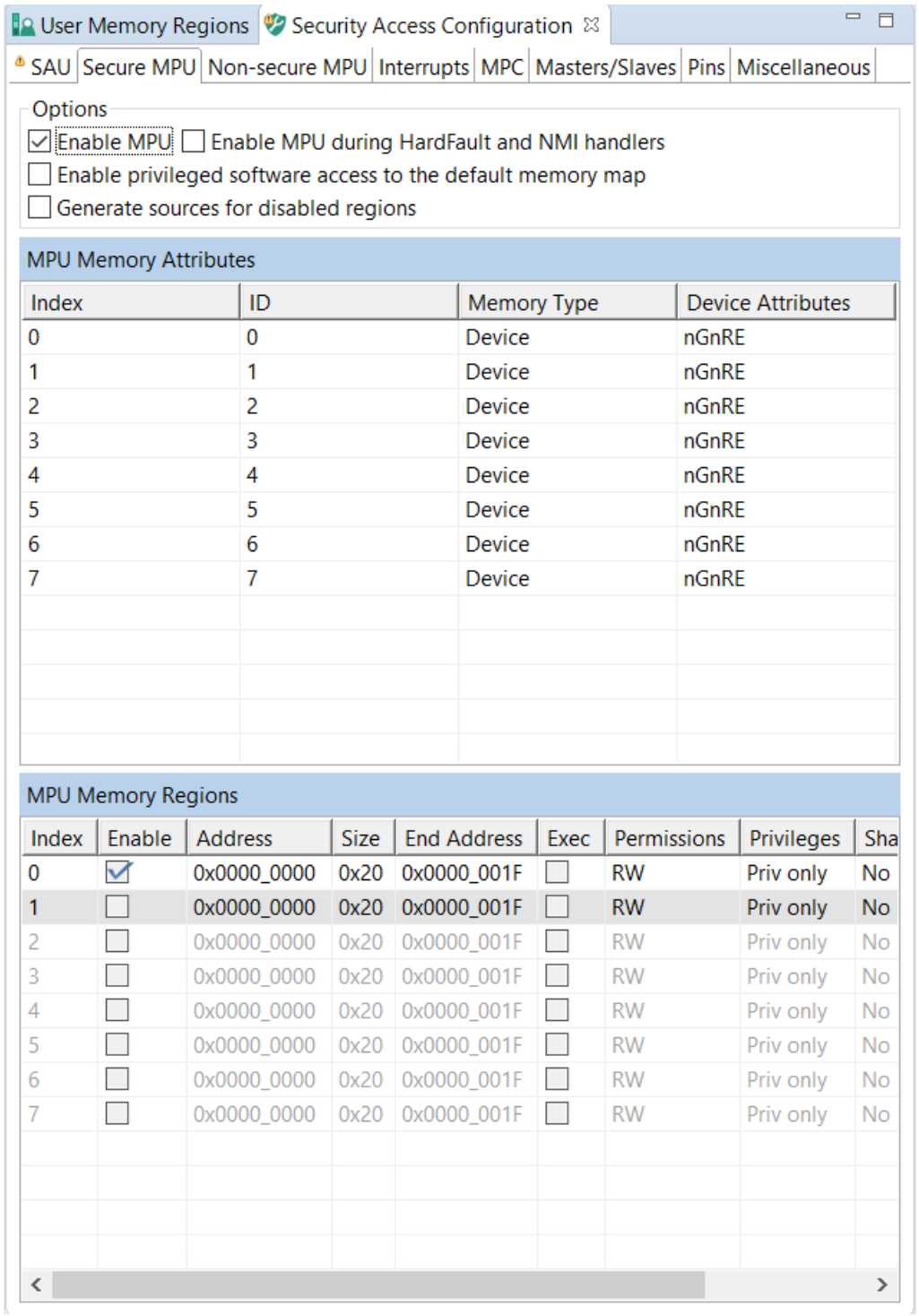


Figure 114. MPU

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

NOTE

Not every device supports MPU.

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

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

1. **Enable** the region.
2. Specify the **Address**.
3. Specify either the **Size** or the **End Address**.
4. Set the **Exec** option if you want the region to be able to run code.
5. Set the **Permissions** (Read Only or Read/Write).
6. Set the **Privileges**.

> **NOTE**
>
> Privileged access can be set by default for all memory regions not handled by MPU by selecting the **Enable privileged software access to the default memory map** option.

7. Set the **Shareability**, or the caching options.
8. Allocate one of the sets from the **MPU Memory Attributes** table in **Mem.Attr.**. Sets can be allocated to more than one region.

7.1.2.4 MPC

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

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

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

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

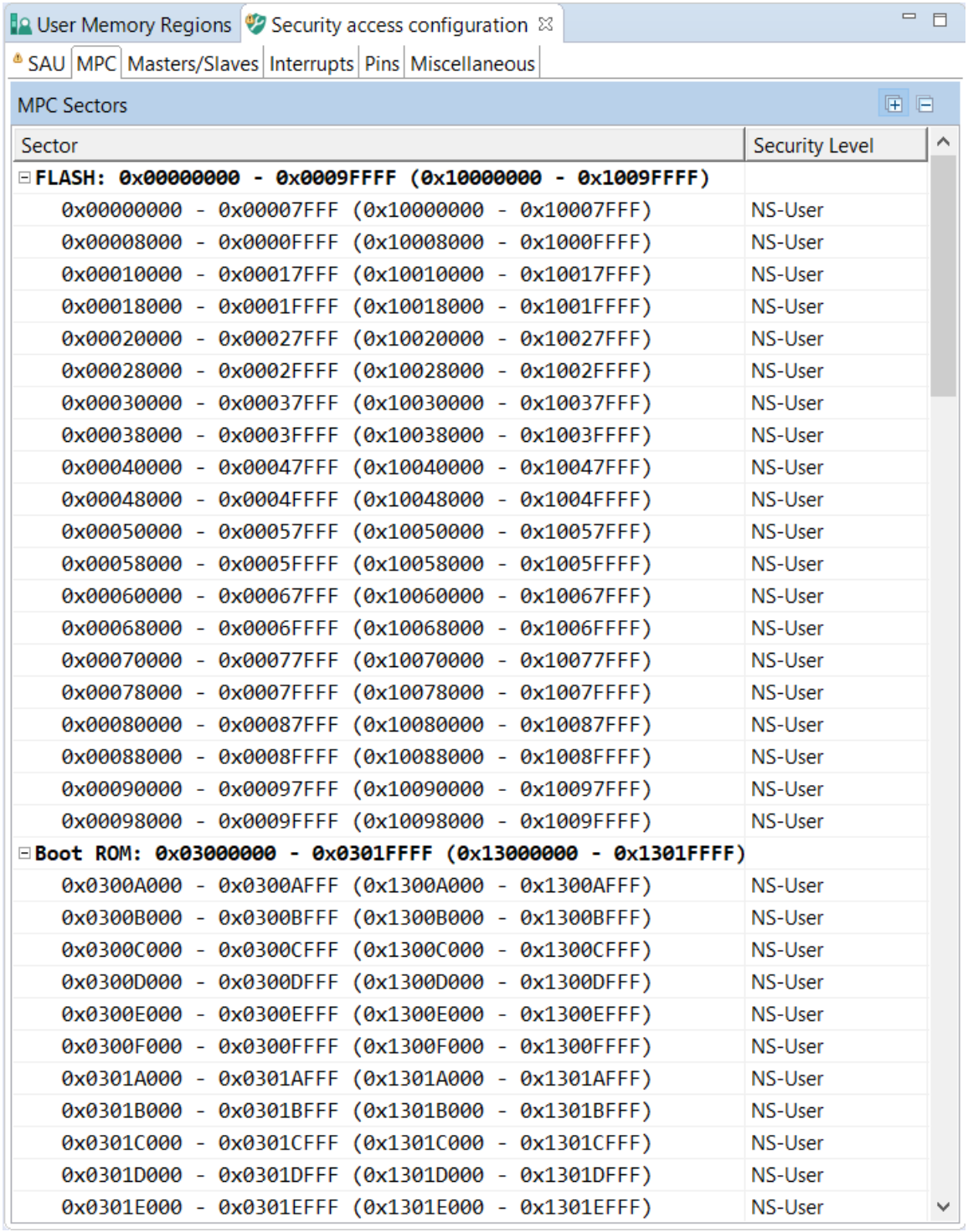


Figure 115. MPC

7.1.2.5 Masters/Slaves

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

Set the bus master/slave security level by left-clicking the relevant cell in the **Security level** column and choosing from the dropdown list. Alternatively, you can right-click the relevant cell in the **Master** and **Slave** column and choose from the security

level from the context menu. To select multiple entries, use the **Ctrl+Left-click** shortcut, then right-click the selected area for the context menu.

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

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

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

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

NOTE

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

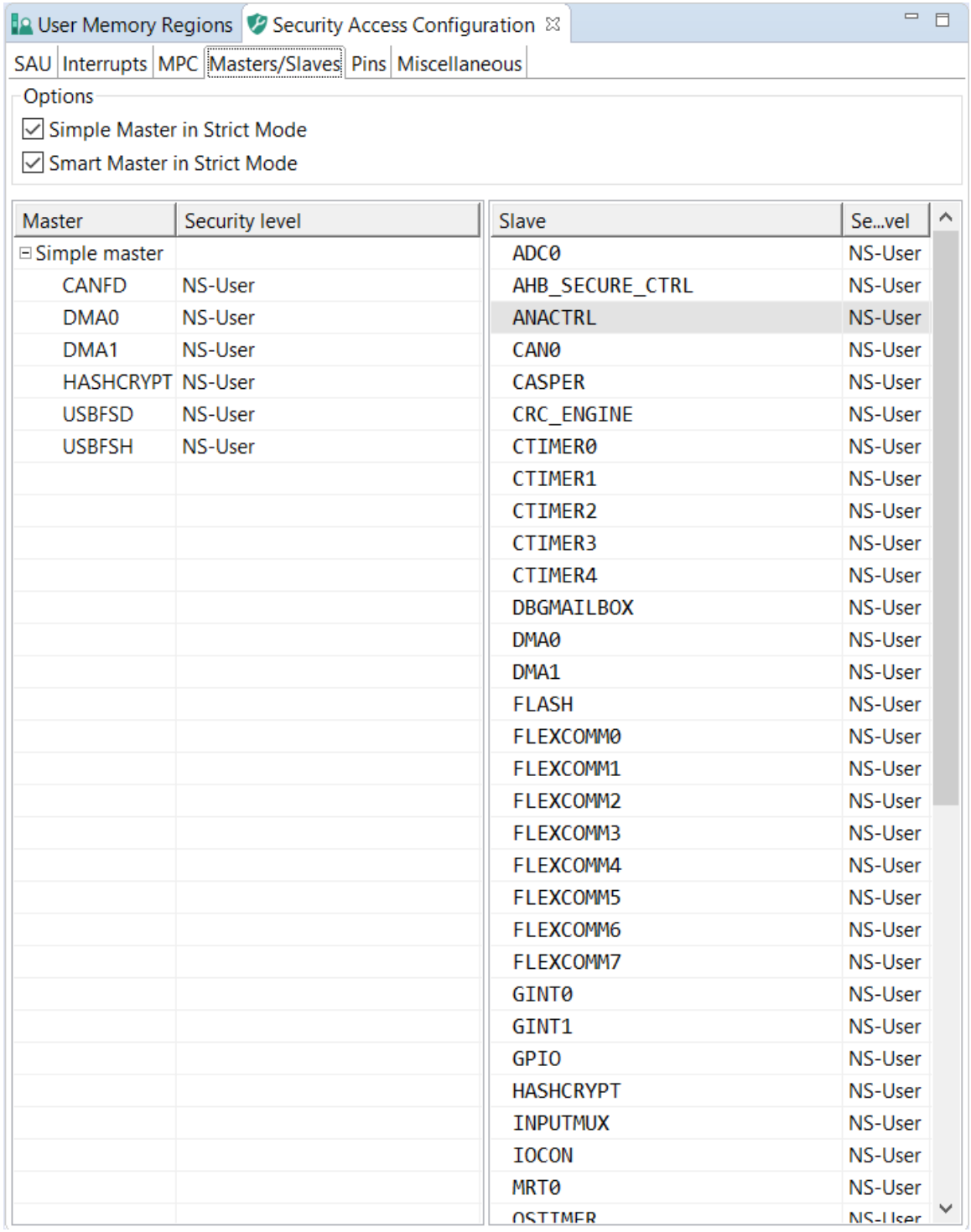


Figure 116. Masters/Slaves

7.1.2.6 Pins

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

All pins' reading GPIO state is set to **Allow** by default. If you want to change the pins reading GPIO state, left-click the **Reading GPIO state** cell of the pin and choose from the dropdown menu. Alternatively, right-click the pin's **Name** cell and choose the

reading GPIO state from the context menu. To select multiple entries, use the **Ctrl+Left-click** shortcut, then right-click the selected area for the context menu.

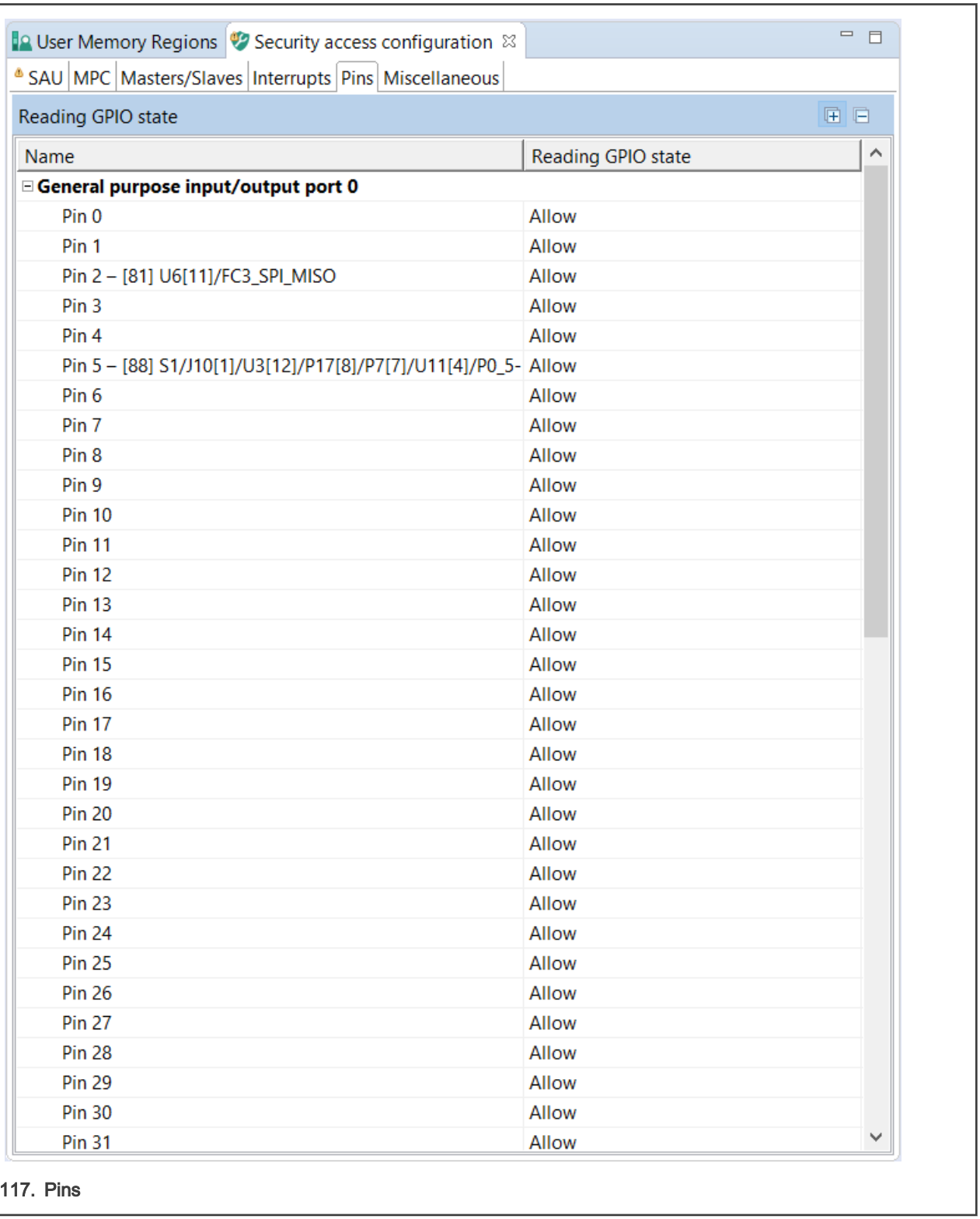


Figure 117. Pins

7.1.2.7 Miscellaneous

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

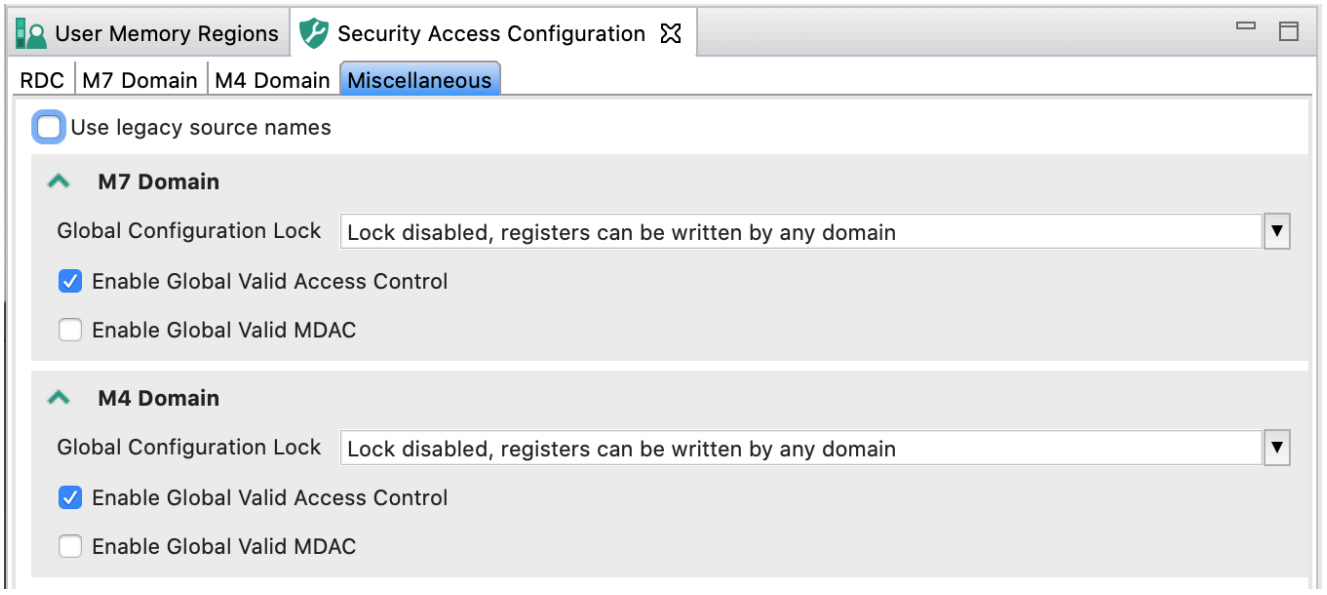


Figure 118. Miscellanous (RDC)

7.1.3 Memory attribution map

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

7.1.3.1 Core 0

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

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

To set the display options, do the following:

1. Click the **Filter access for** checkbox to enable filtering options.
2. Select the master security access you want to review by choosing from the **Master** dropdown menu.
3. Optionally, set the security level of the selected master by choosing from the **Security mode** dropdown menu. This setting has no effect on the configuration.
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 119. Core 0

7.1.3.2 Other masters

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

To set the display options, do the following:

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

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

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

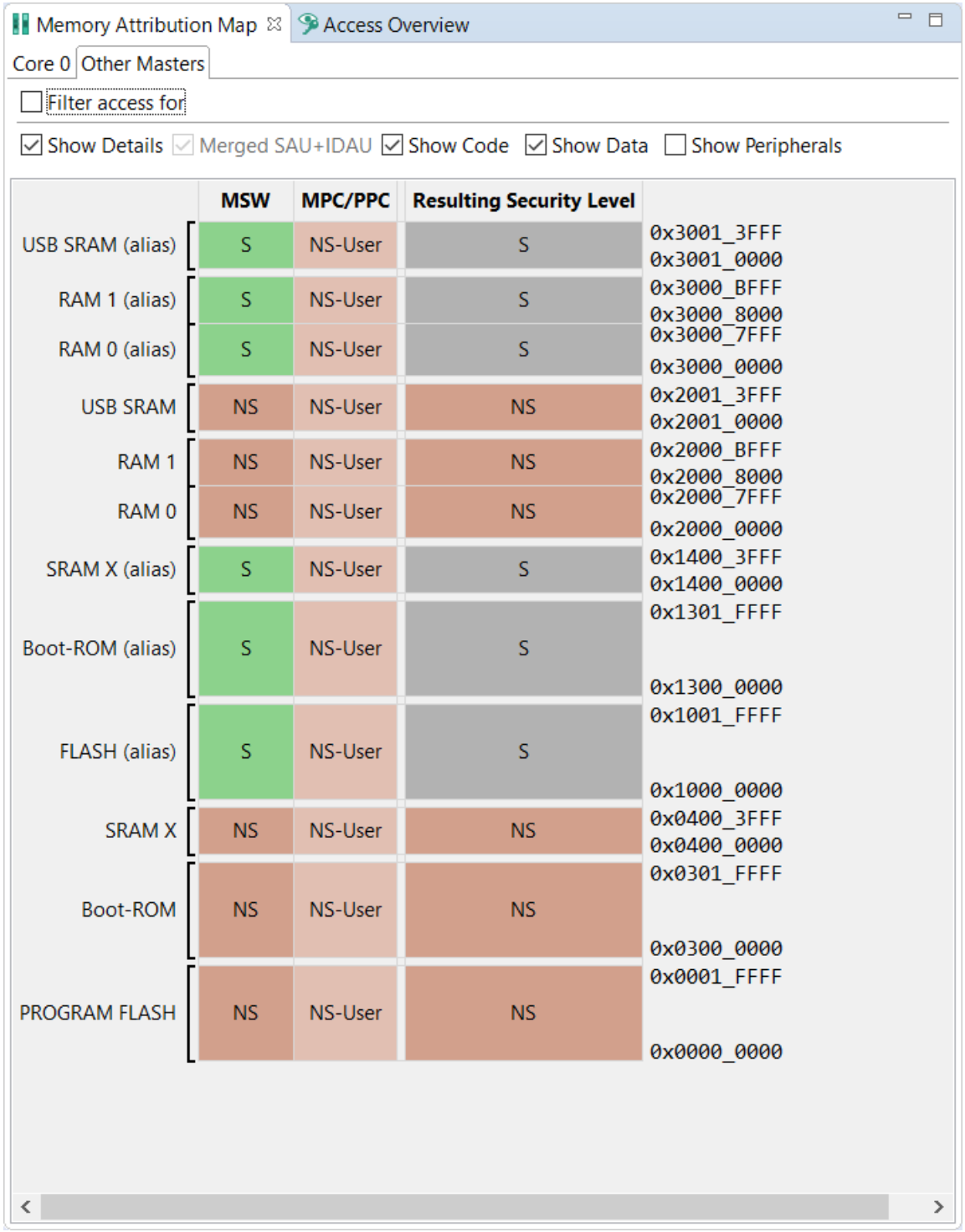


Figure 120. Other masters

7.1.4 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 righ-hand side of the toolbar.

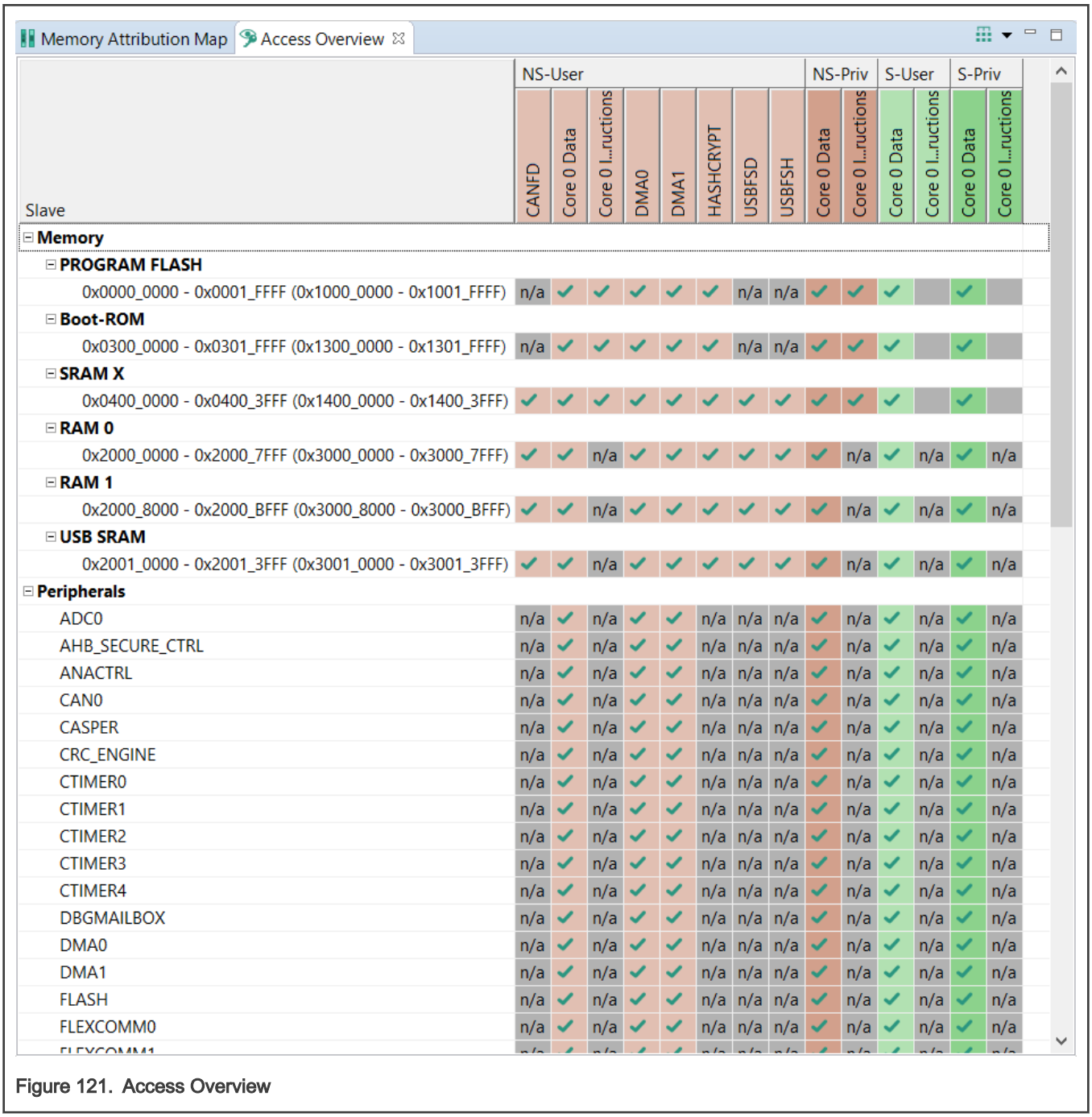


Figure 121. Access Overview

7.1.5 Code generation

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

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

Some AHB 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** sub-view.

7.2 RDC-enabled devices

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

This section describes the features and appearance of the tool devices with enabled 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

7.2.1 User Memory Regions view

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

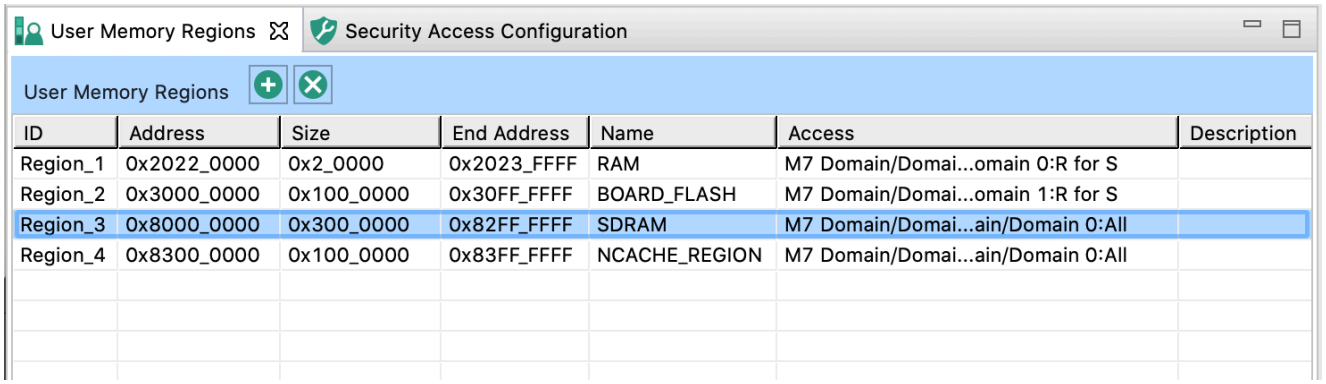


Figure 122. User Memory Regions

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.

7.2.1.1 Access templates

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

NOTE

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

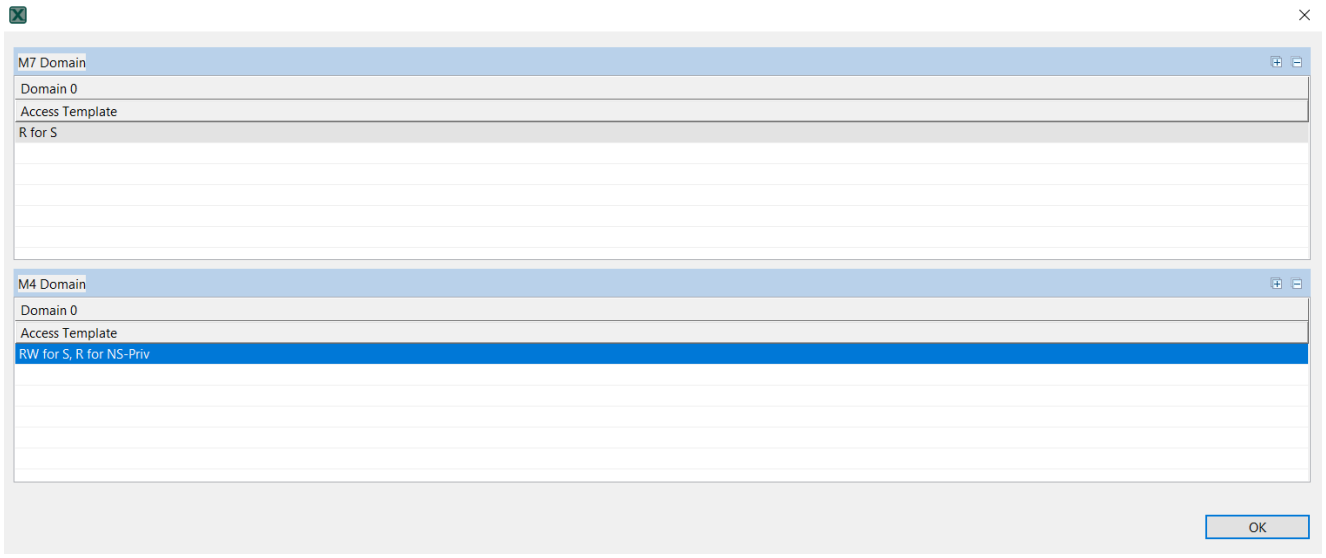


Figure 123. Access template

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.

7.2.2 Security Access Configuration view

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

7.2.2.1 RDC

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

7.2.2.1.1 RDC Masters

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

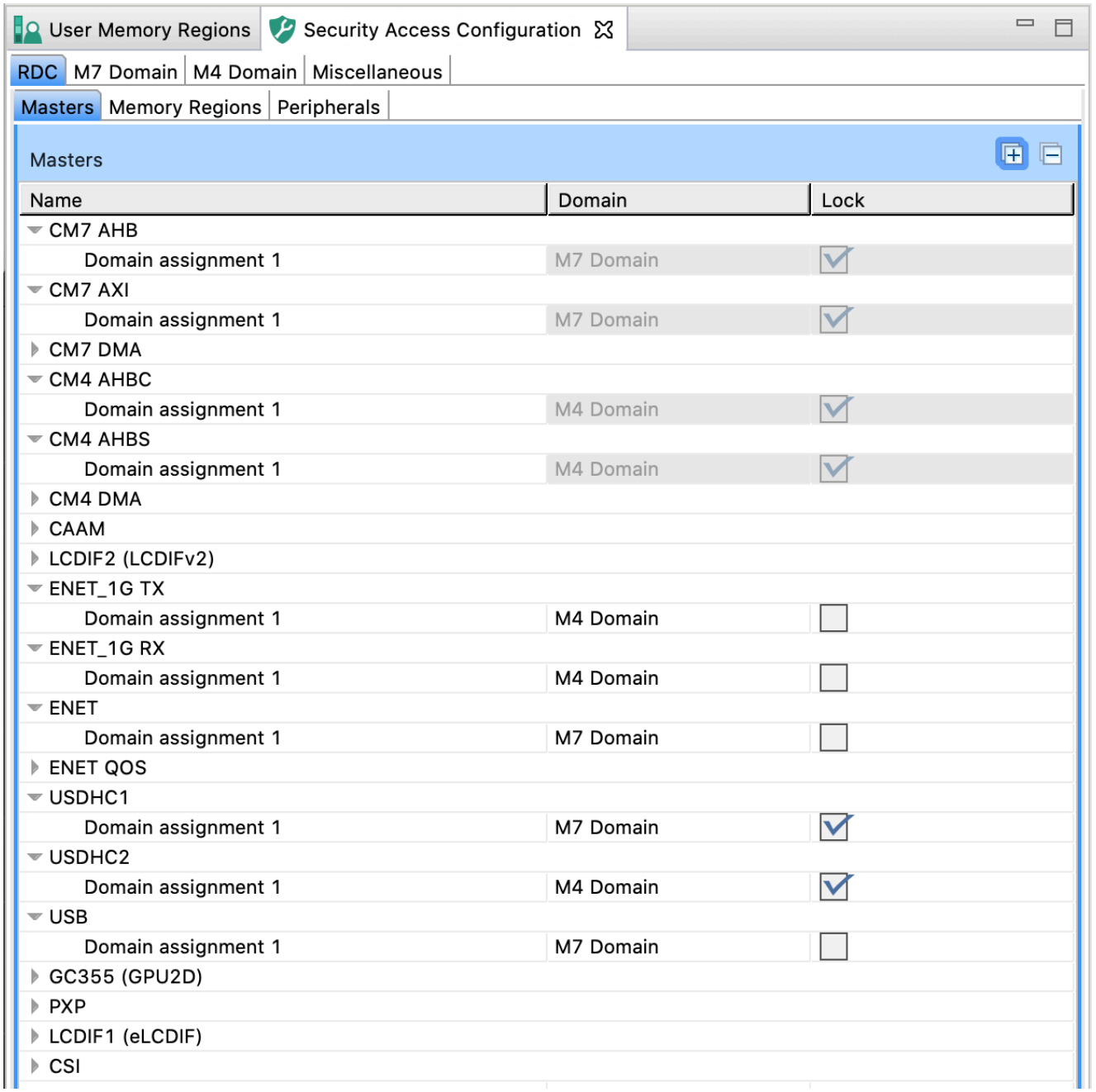


Figure 124. RDC Masters

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

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

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

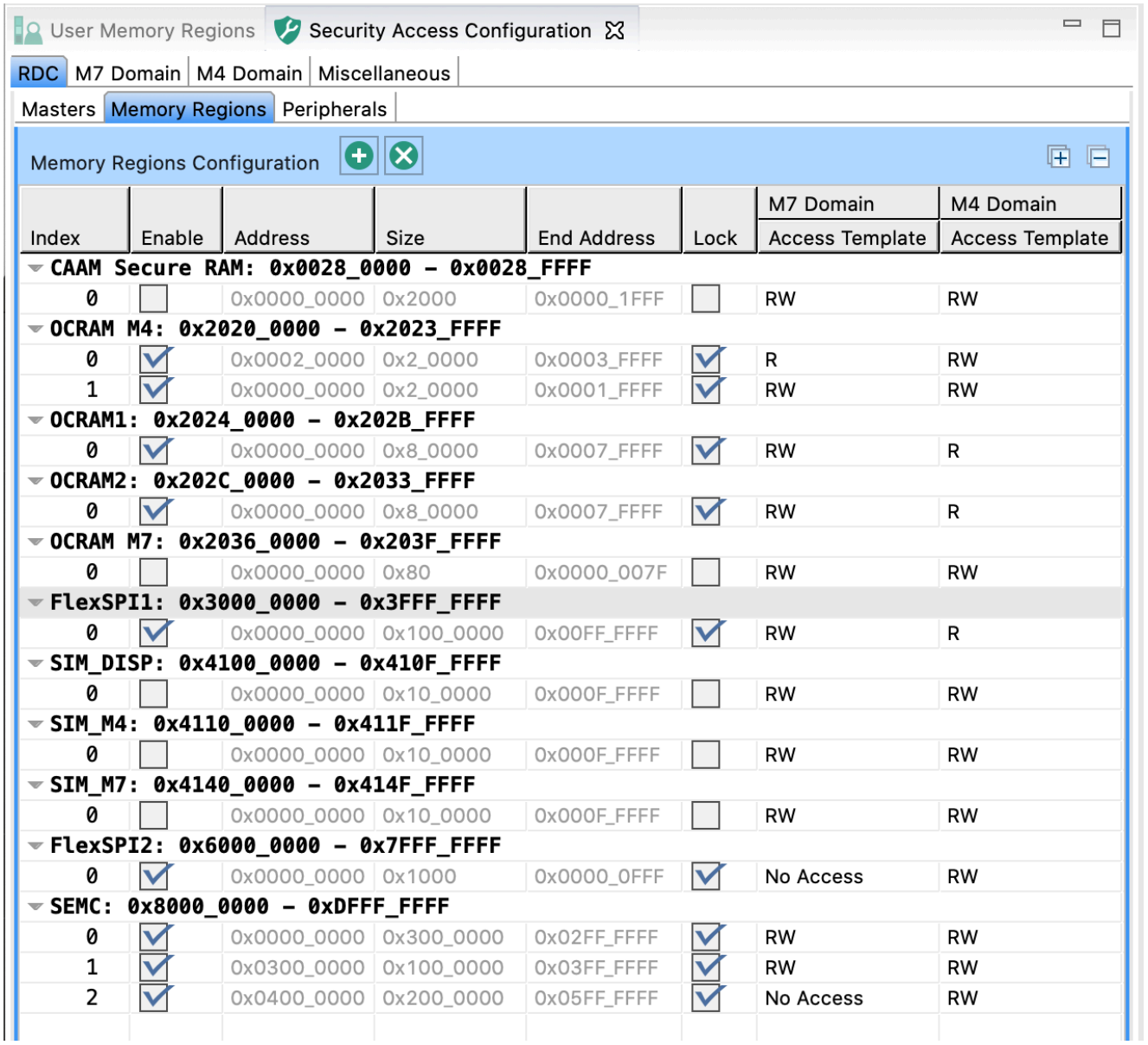
NOTE

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

7.2.2.1.2 Memory Regions

In the **Memory Regions** sub-view, 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 125. Memory Regions

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

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

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

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

7.2.2.1.3 Peripherals

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

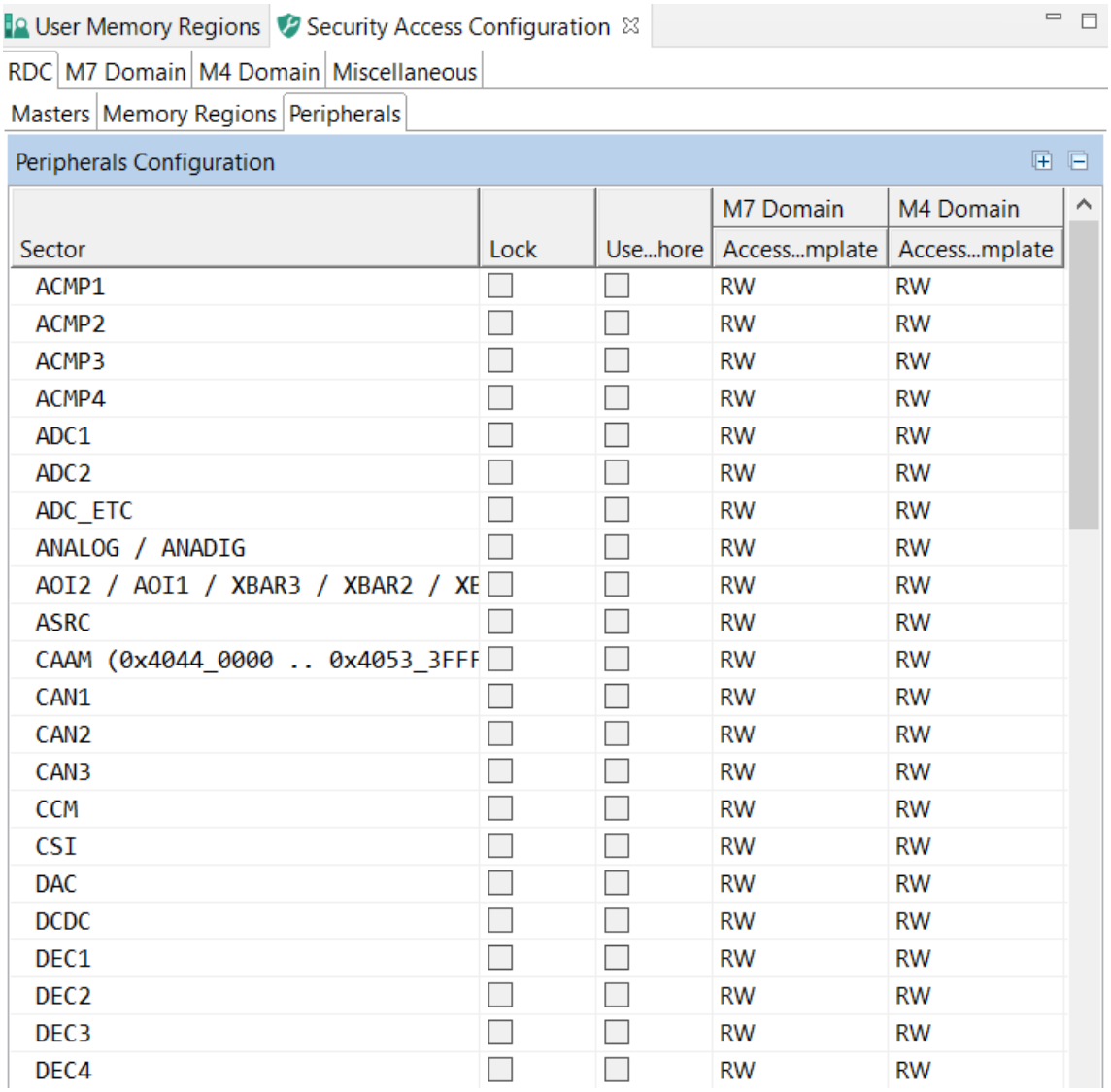


Figure 126. Peripherals

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

1. Optional: **Lock** the settings to prevent further register entries.
2. Select **Use semaphore** to enable the semaphore function for the peripheral.

> **NOTE**
>
> When enabled, the master cannot access this peripheral until obtaining a semaphore. During the time that the domain has the semaphore in possession, its bus masters have exclusive access to the peripheral.

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

7.2.2.2 XRDC2 Domains view

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

7.2.2.2.1 MPU

In the **MPU** sub-view, 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.

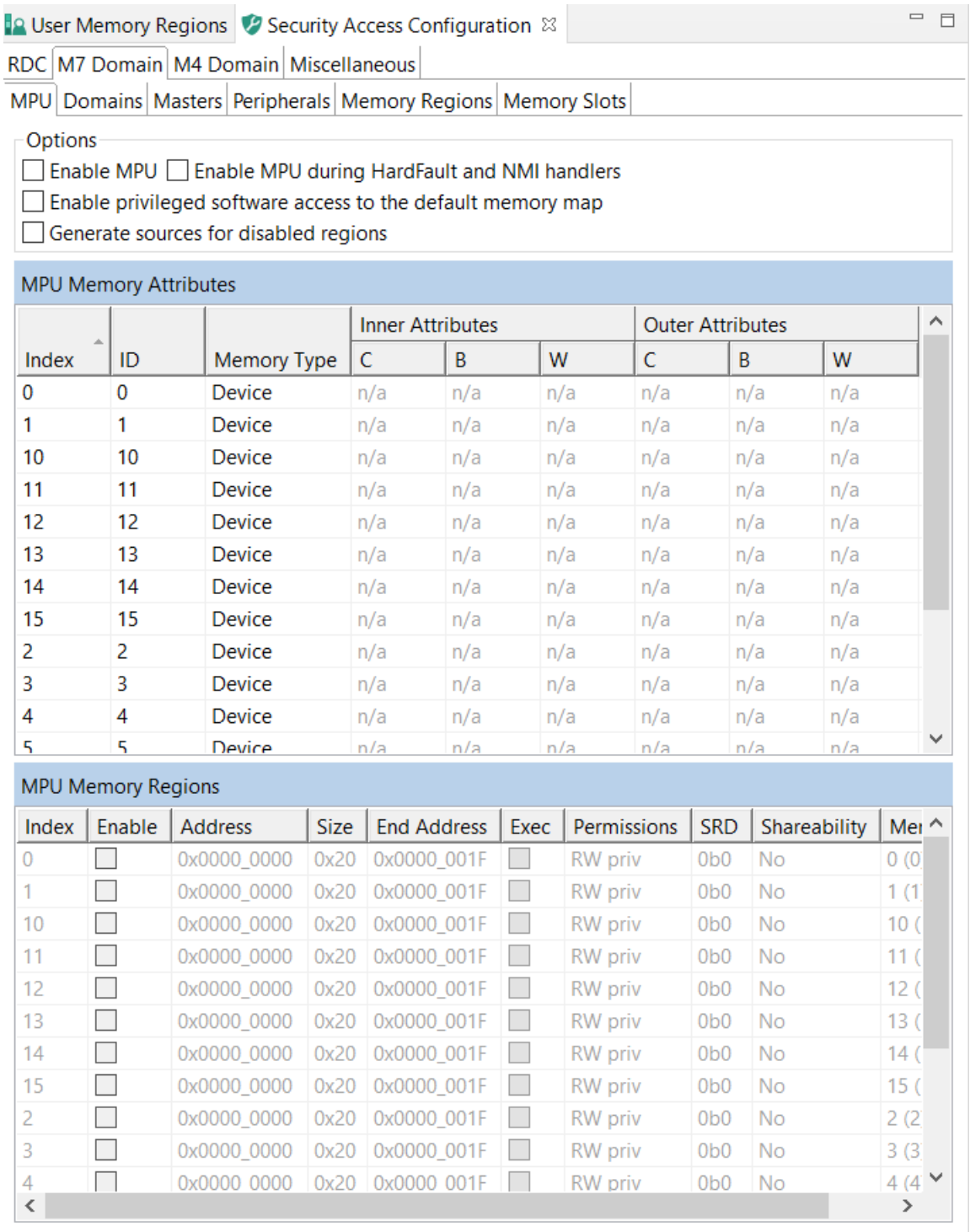


Figure 127. MPU

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

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

1. **Enable** the region.
2. Specify the **Address**.

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

7.2.2.2.2 Domains

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

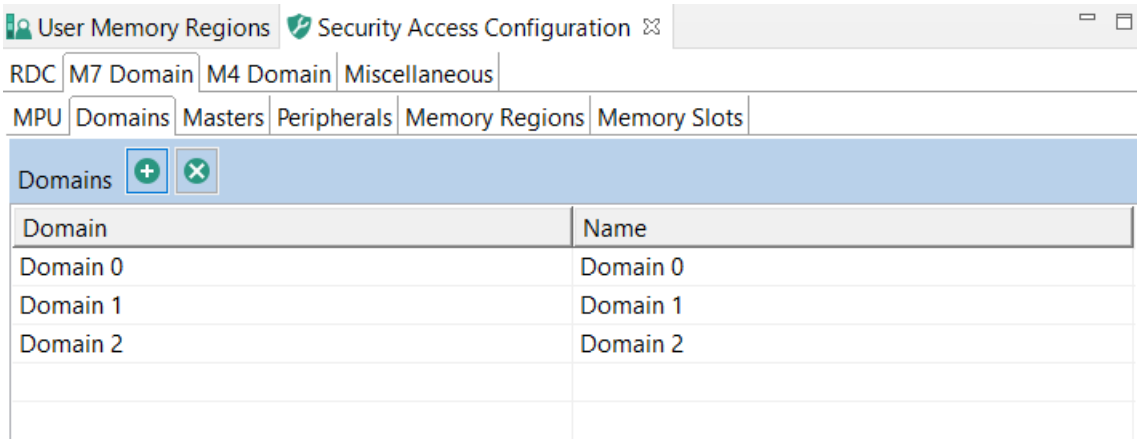


Figure 128. Domains

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

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

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

7.2.2.2.3 Masters

In the **Masters** sub-view, 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.

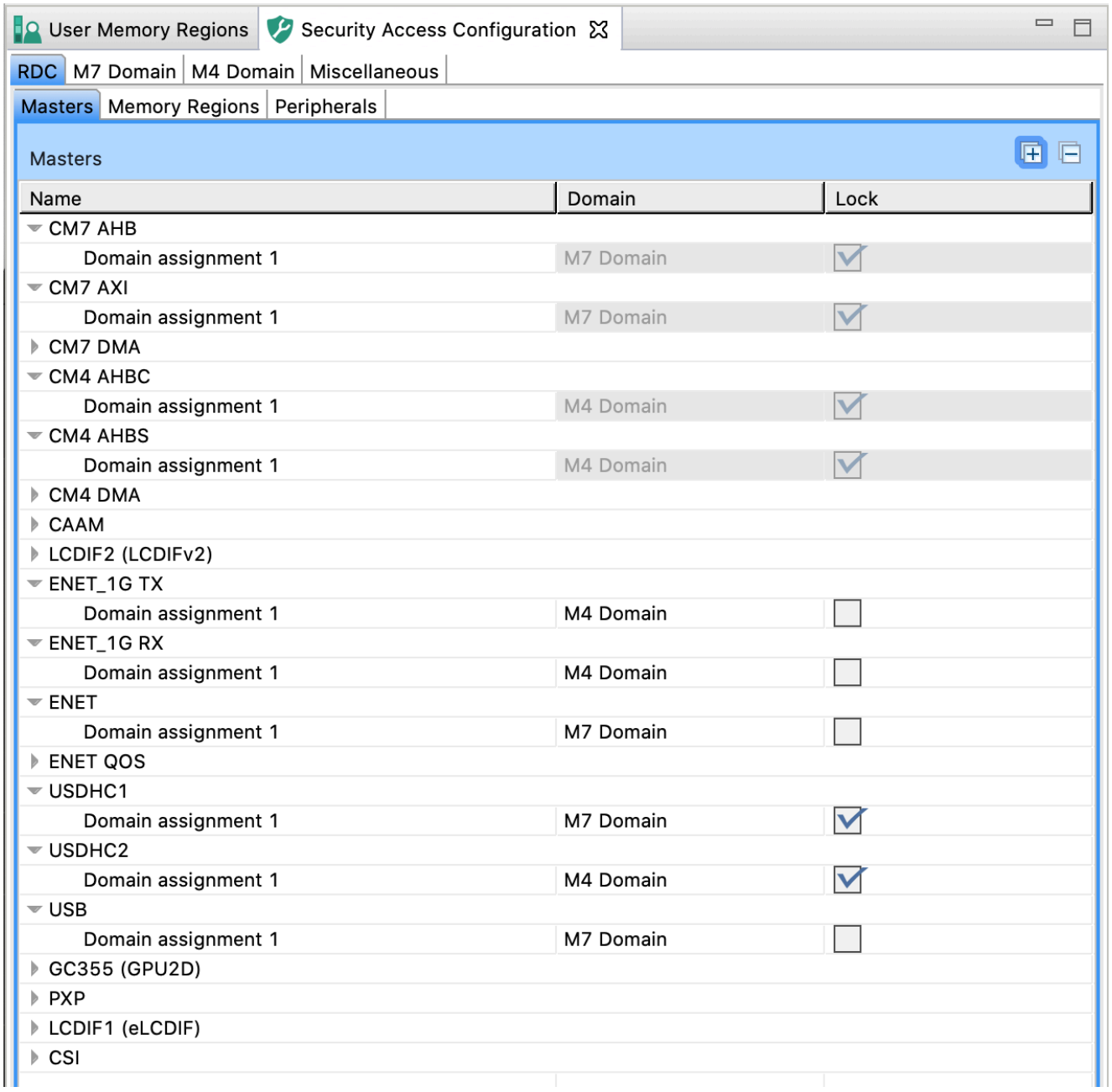


Figure 129. Masters

To add a new domain assignment:

1. Click the **Add new domain assignment for the selected master** button.
2. Select the **Enable** checkbox.
3. Enter the **Match Input** value.

NOTE

The match field specifies the reference value for the comparison with the MDAC match input. The match field width varies by MDAC instance from 0 to 16 bits. Unimplemented bits are read as 0. A size of 0 bits generates a hit on all comparisons.

4. Enter the **Mask Input** value.

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

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

7.2.2.2.4 Peripherals

In the **Peripherals** sub-view, 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.

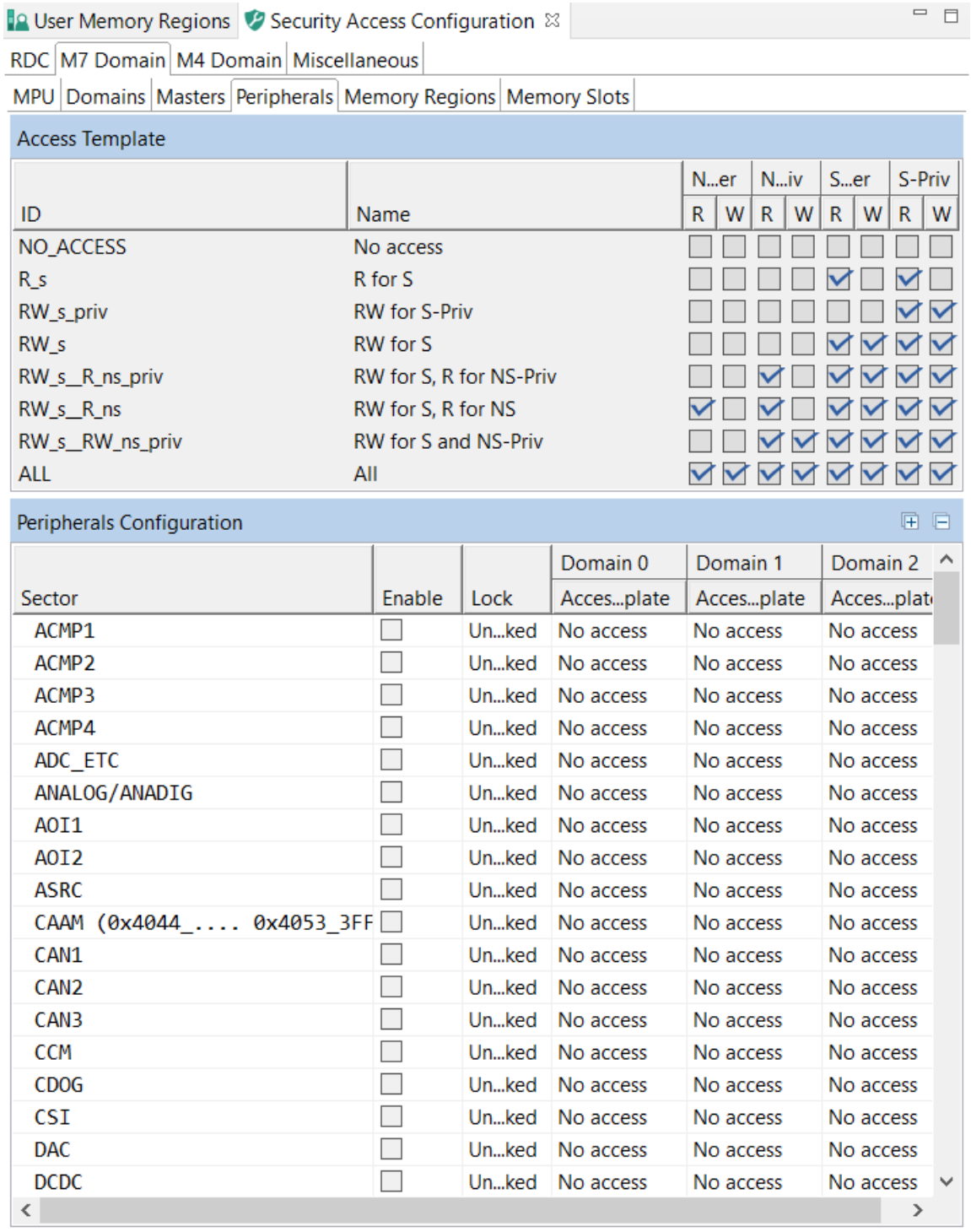


Figure 130. Peripherals

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

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

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

7.2.2.2.5 Memory Regions

In the **Memory Regions** sub-view, 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.

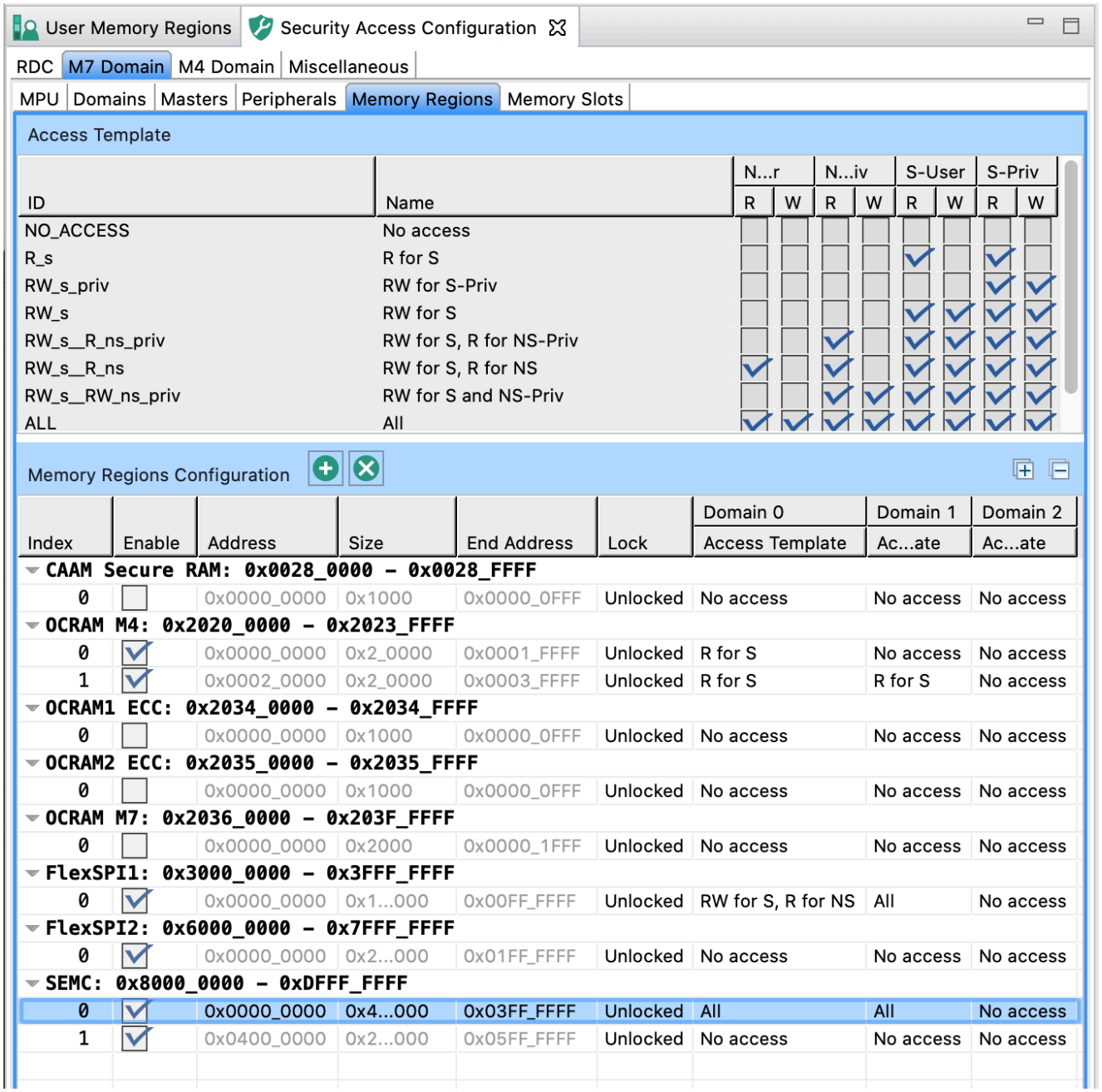


Figure 131. Memory Regions

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 Adress**.
4. Set the **Lock** to the desired state.
5. Set the **Access Template** for all listed domains.

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

7.2.2.2.6 Memory Slots

In the **Memory Slots** sub-view, 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.

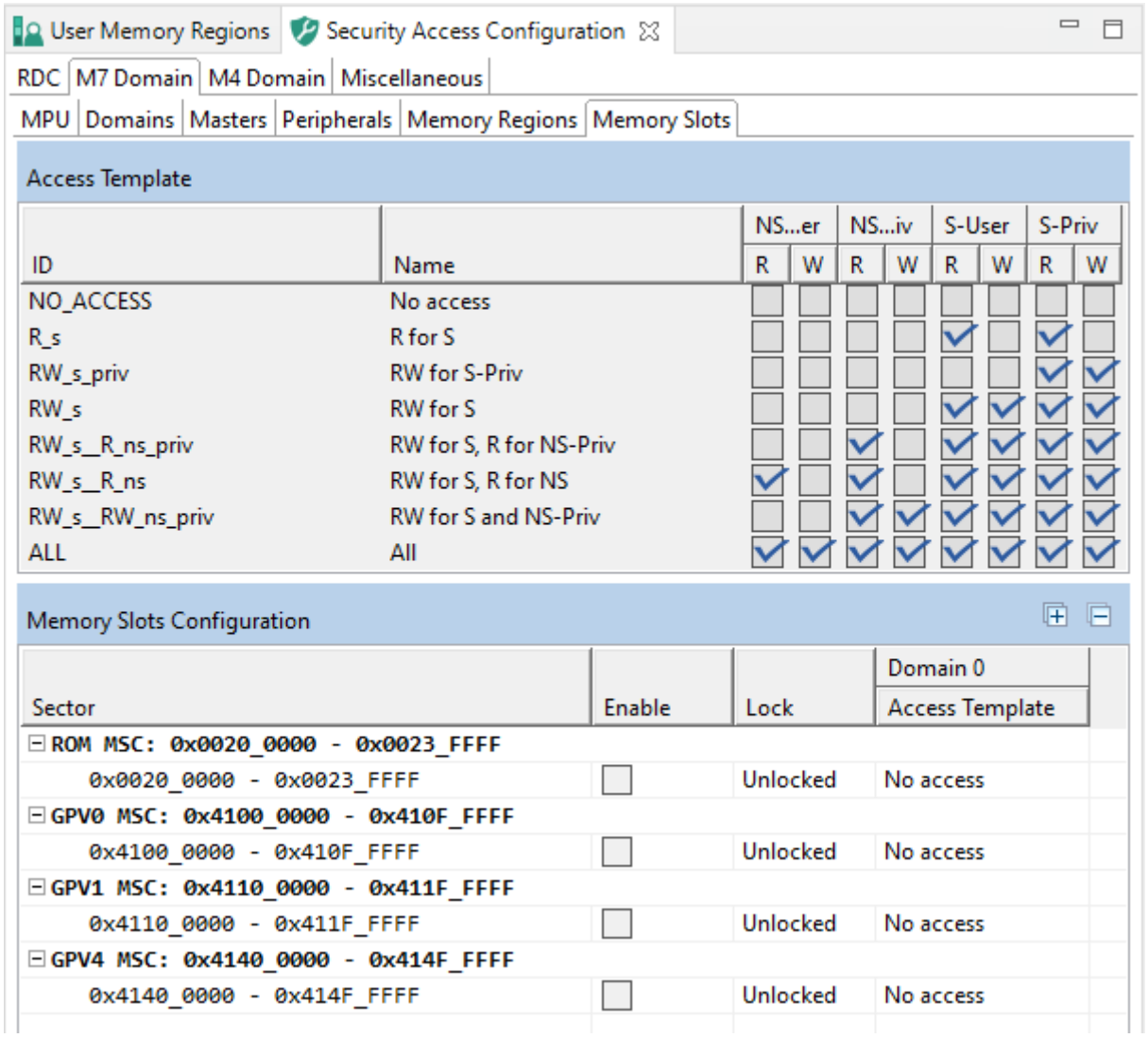


Figure 132. Memory Slots

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

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

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

7.2.2.3 Miscellaneous

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

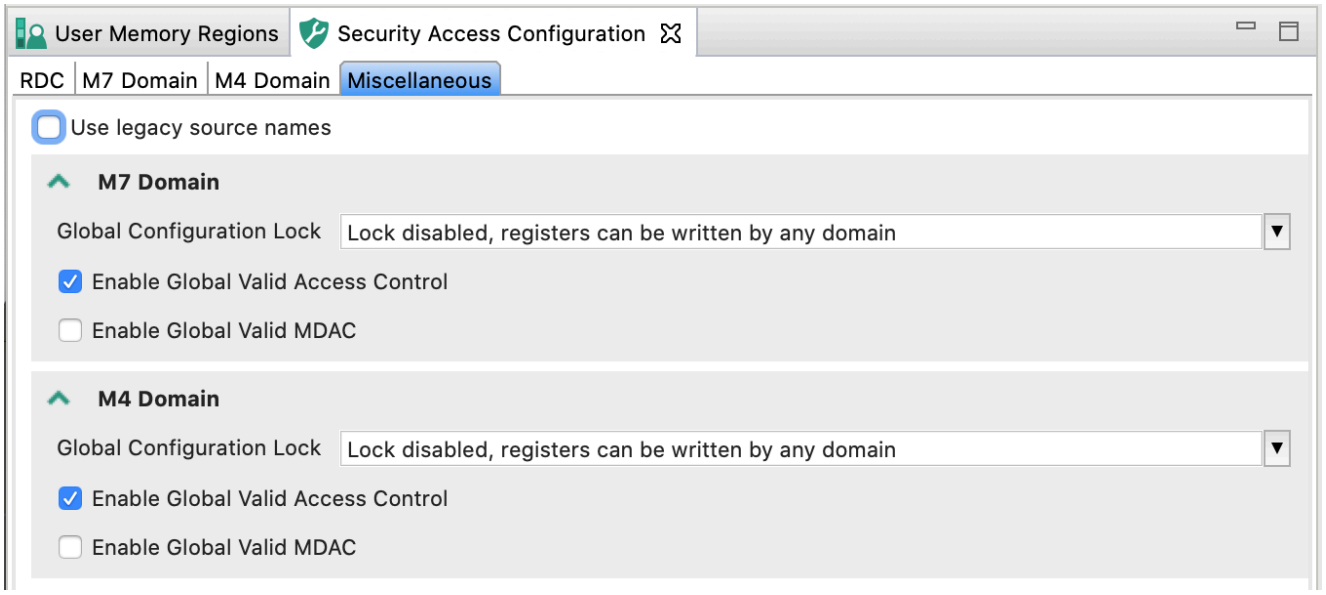


Figure 133. Miscellanous (RDC)

7.2.3 Memory Attribution Map

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

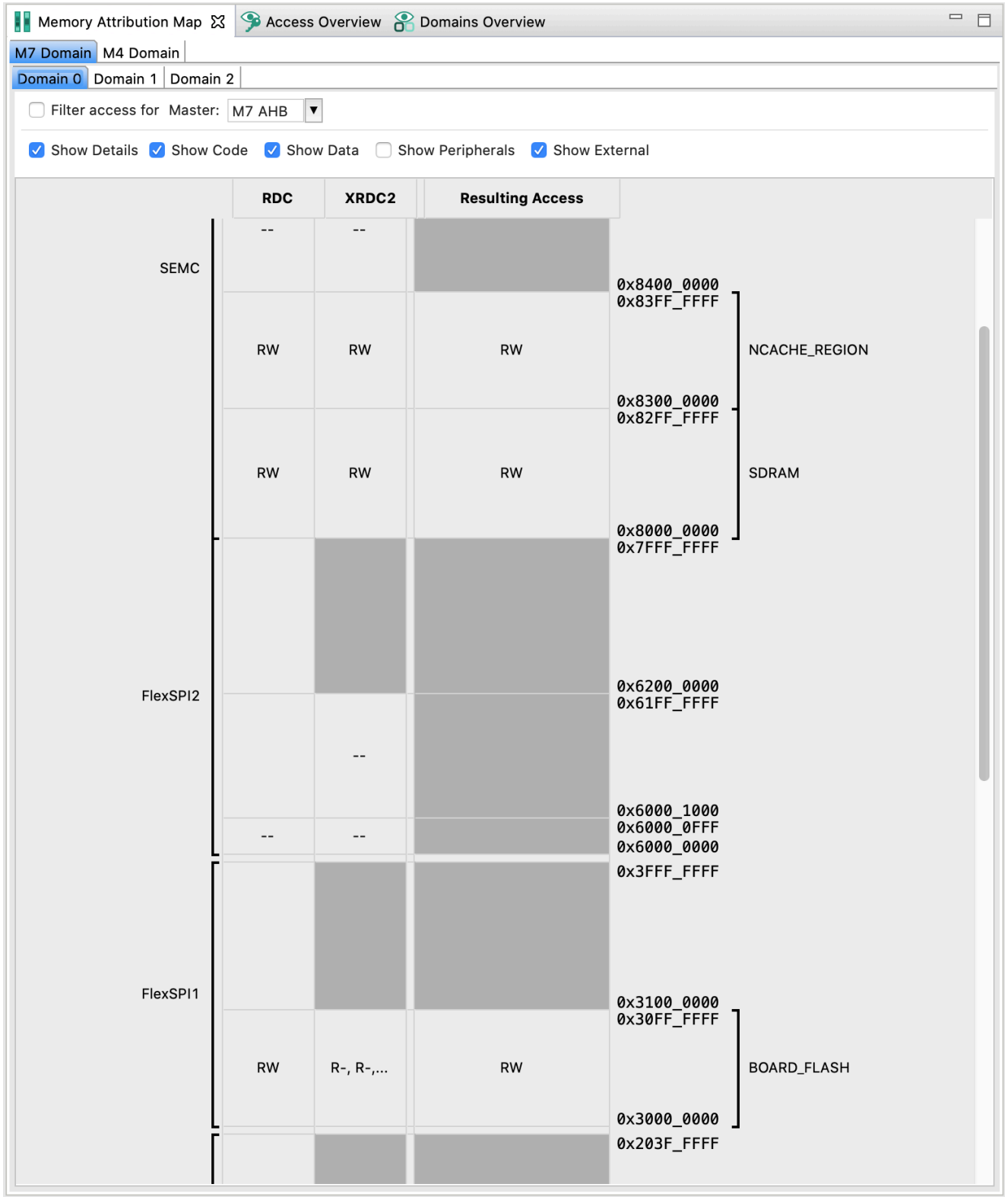


Figure 134. Memory Attribution Map

To set the display options, do the following:

1. Click the **Filter access for** checkbox to enable filtering options.

2. Select the master you want to review by choosing from the **Master** dropdown menu.
3. Optionally, set the security level of the selected master by choosing from the **Security mode** dropdown menu. This setting has no effect on the configuration.
4. Optionally, customize the output by de-selecting the **Show Details**, **Show Code**, **Show Data**, **Show Peripherals**, and **Show External** options.
5. Optionally, filter displayed memory regions in the **Filter** area.

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

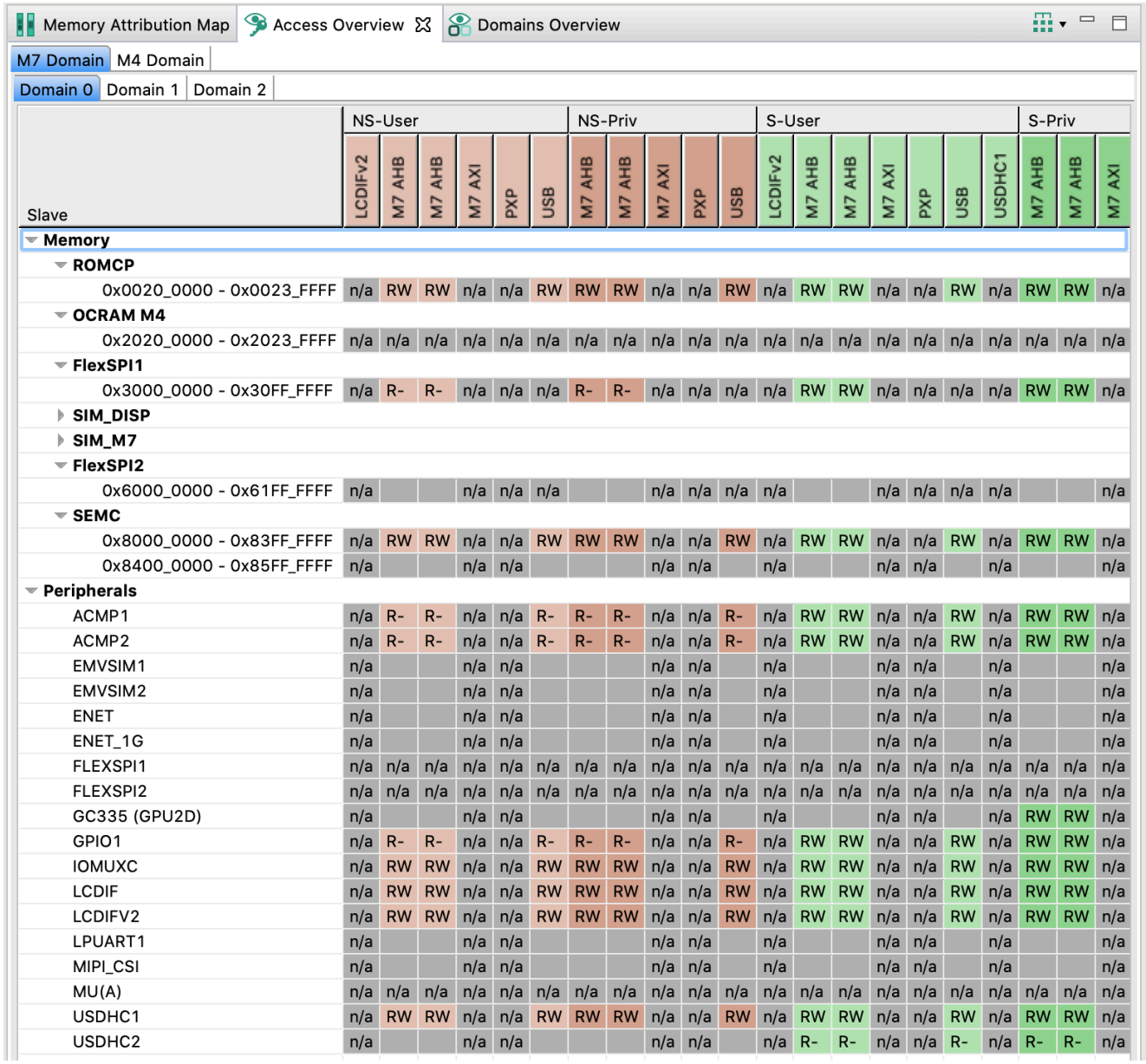
Double-click the cell to open the pertinent settings in **Security Access Configuration**.

7.2.4 Access Overview

In **Access Overview**, you can review 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.



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 135. 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 righ-hand side of the toolbar.

7.2.5 Domains Overview

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

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

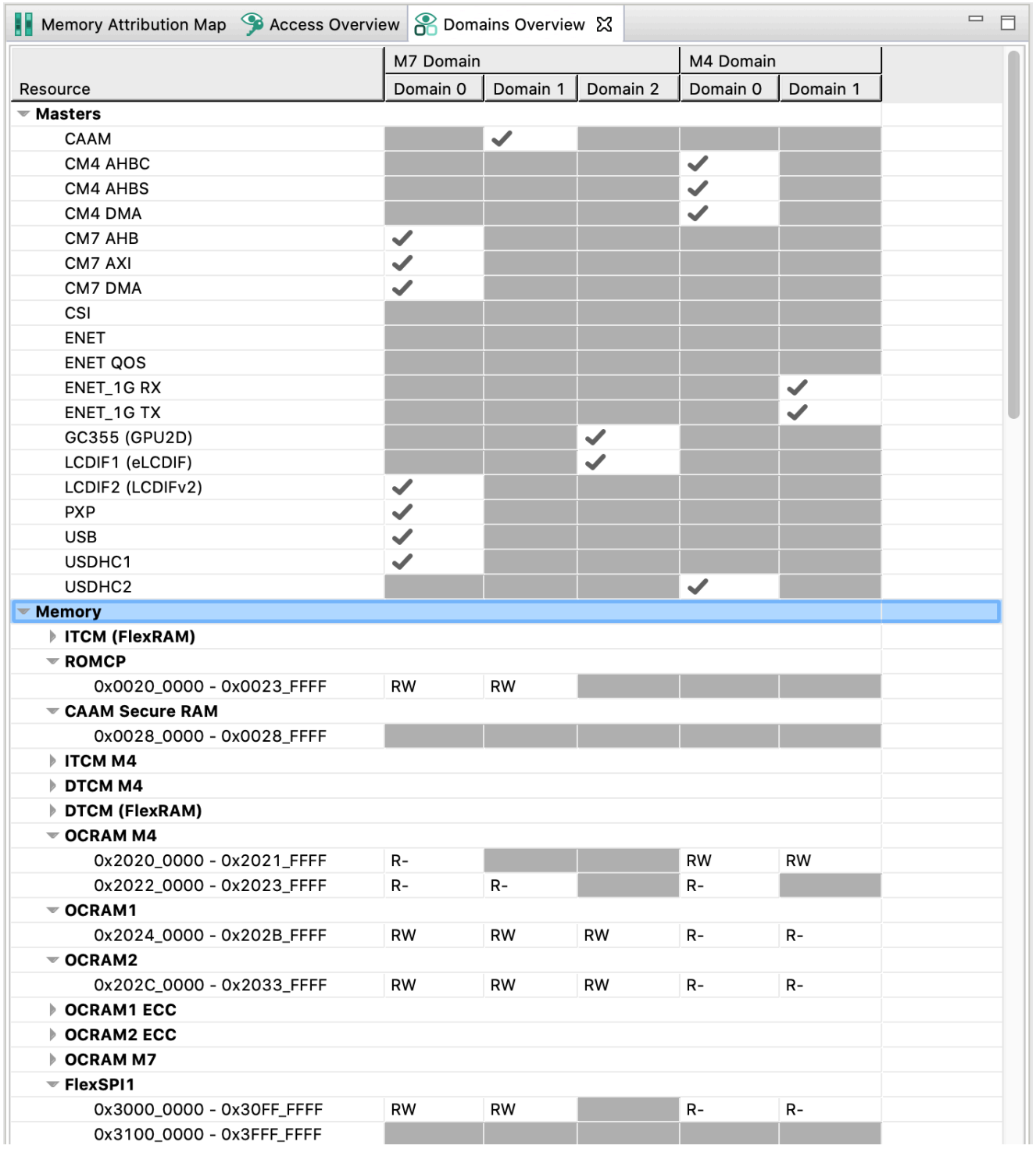


Figure 136. Domain Overview

7.2.6 Code generation

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

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

Some AHB 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** sub-view.

Chapter 8
Advanced Features

8.1 Switching the processor

You can switch the processor or the package of the current configuration to a different one. However, switching to a completely different processor may lead to 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 re-configure 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**.

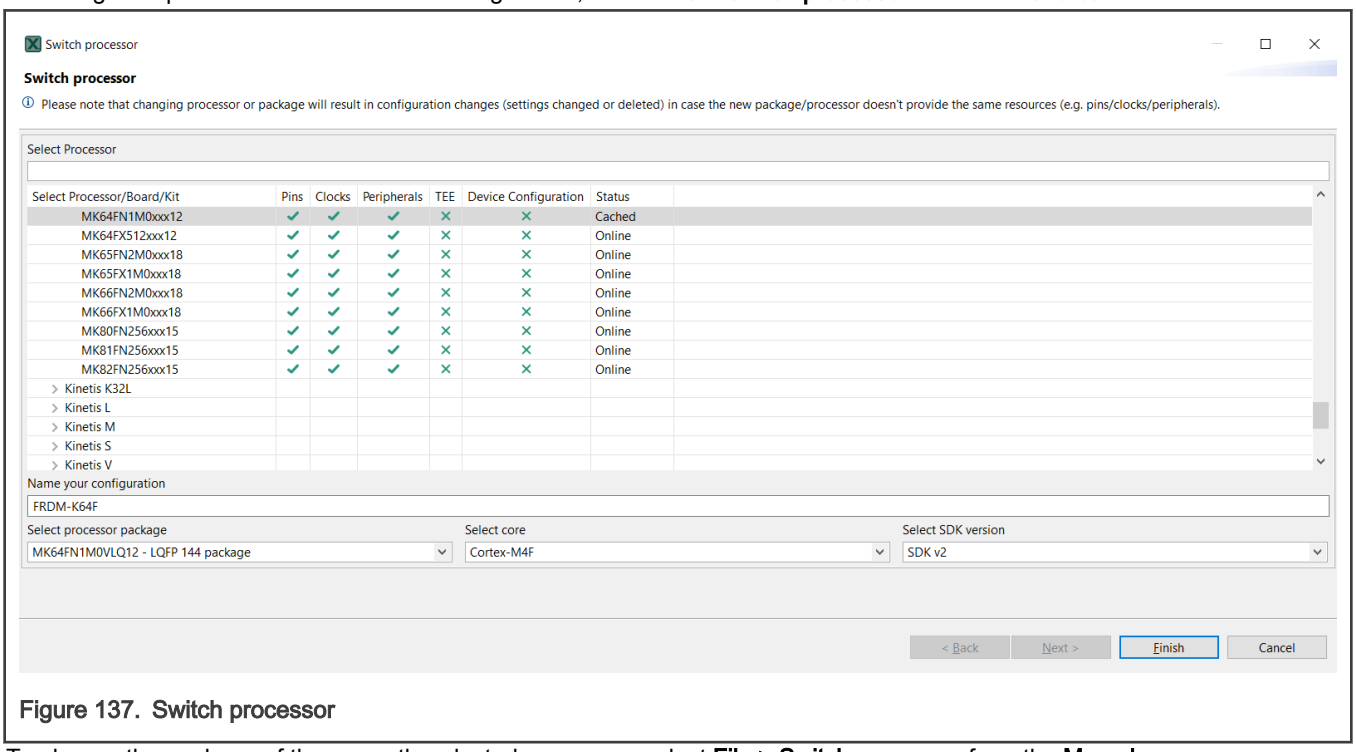


Figure 137. Switch processor

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

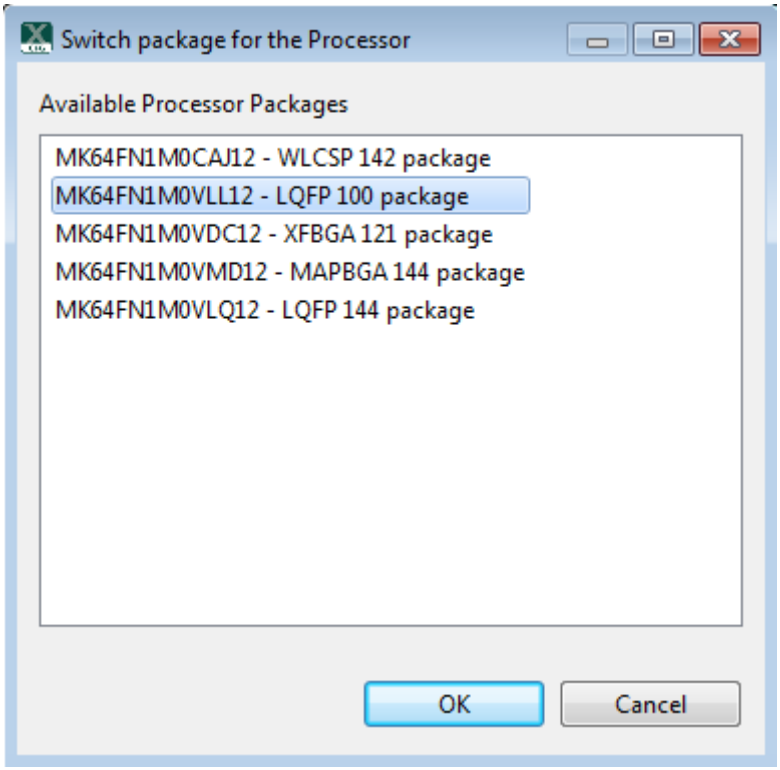


Figure 138. Switch package

8.2 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 file name 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 139. 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 140. Aligning to block

8.3 Tools advanced configuration

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

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

8.4 Generating HTML reports

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

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

8.5 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**.

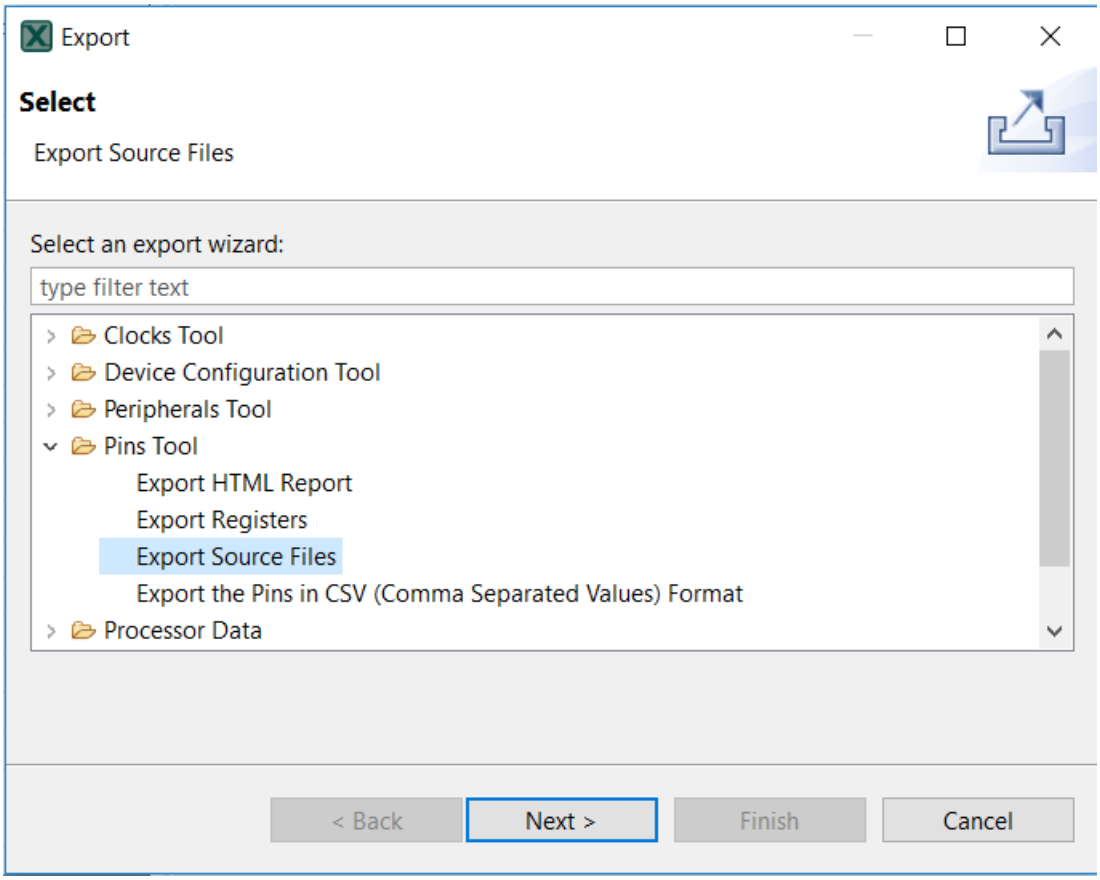


Figure 141. Export wizard

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

Figure 142. Select target folder

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

8.6 Exporting registers

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

To export registers, follow these steps:

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

8.7 Command line execution

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

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

Notes regarding command line execution:

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

Command example:

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

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

Table 25. Commands supported in the framework

Command name	Definition and parameters	Description	Restriction	Example

Table continues on the next page...

Table 25. Commands supported in the framework (continued)

Force language	-nl {lang}	Force set language {lang} is in ISO-639-1 standard	Removal of the '.nxp' folder from home directory is recommended, as some text might be cached Only 'zh' and 'en' are supported	-nl zh
Show console	-consoleLog	Log output is also sent to Java's System.out (typically back to the command shell if any)	None	
Select MCU	-MCU	MCU to be selected by framework	Requires –SDKversion command	-MCU MK64FX512xxx12
Select SDK version	-SDKversion	Version of the MCU to be selected by framework	Requires -MCU command	-SDKversion test_ksdk2_0
Select part number	-PartNum	Select specific package of the MCU	Requires -MCU and -SDKversion commands	-PartNum MK64FX512VLL12
Configuration name	-ConfigName	Name of newly created configuration - used in export	Name is used when new configuration is created by -MCU and -SDKversion commands	-ConfigName "MyConfig"
Select tool	-HeadlessTool	Select a tool that should be run in headless mode	None	-HeadlessTool Clocks
Load configuration	-Load	Load existing configuration from (*.mex) file	None	-Load C:/conf/conf.mex
Export Mex	-ExportMEX	Export .mex configuration file after tools run Argument is expected as a folder name	None	-MCU xxx -SDKversion xxx -ExportMEX C:/exports/my_config_folder
Export all generated files	-ExportAll	Export generated files (with source code and so on. Code is regenerated before export Includes -ExportSrc and in framework -ExportMEX Argument is expected as a folder name	Requires -HeadlessTool command	-HeadlessTool Pins -ExportAll C:/exports/generated
Create new configuration by importing toolchain project	-ImportProject {path}	Creates new configuration by importing toolchain project Parameter is path to the root of the toolchain project	Requires -HeadlessTool command	-HeadlessTool Pins -ImportProject c:\test\myproject

Table continues on the next page...

Table 25. Commands supported in the framework (continued)

Specify SDK path	-SDKpath {path}	Specify absolute path to the root directory of the SDK package.	@since v3.0	-SDKpath c:\nxp\SDK_2.0_MKL43Z256xxx4

8.7.1 Command line execution - Pins tool

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

Table 26. Commands supported in Pins

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

Table continues on the next page...

Table 26. Commands supported in Pins (continued)

		Code will be regenerated before export Argument is expected as a folder name		
Export HTML report file	-ExportHTML	Export generated html report file. Code will be regenerated before export Argument is expected as a folder name	Requires -HeadlessTool Pins	-HeadlessTool Pins -ExportHTML C:/exports/html
Export registers	-ExportRegisters	Export registers tab into folder. Code will be regenerated before export Argument is expected as a folder name	Requires -HeadlessTool Pins	-HeadlessTool Pins -ExportRegisters C:/exports/regs

8.7.2 Command line execution - Clocks Tool

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

Table 27. Commands supported in Clocks

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

Table continues on the next page...

Table 27. Commands supported in Clocks (continued)

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

8.7.3 Command line execution - Peripherals tool

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

Table 28. Commands supported in Peripherals Tool

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

Table continues on the next page...

Table 28. Commands supported in Peripherals Tool (continued)

		Code will be regenerated before export Argument is expected to be a folder		
Export HTML report file	-ExportHTML	Export generated html report file Code will be regenerated before export Argument is expected to be a folder	Requires -HeadlessTool Peripherals	-HeadlessTool Peripherals -ExportHTML C:/exports/html
* for internal commands, internal plugin must be installed into production application				

8.7.4 Command line execution - Project Cloner

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

Table 29. Commands supported in Project Cloner

Command name	Definition and parameters	Description	Restriction	Example
Specify SDK path	-SDKpath {path}	Specify absolute path to the root directory of the SDK package	@since v3.0	-SDKpath c:\nxp\SDK_2.0_MKL43Z256xxx4
Clone SDK example project	-PG_clone {board} {example} {toolchain} {wrkspc} {prjName}	Clones specified SDK example projecte under new name 1. {board} - subdirectory of the board in SDK package 2. {example} - relative path from board sub-dir and name of the example, for example demo_apps/hello_world; use '/' as a path separator 3. {toolchain} - id of the toolchain to create project	Requires -HeadlessTool PrjCloner and -SDKpath {path} @since v3.0	-HeadlessTool PrjCloner -SDKpath c:\nxp\SDK_2.0_MKL43Z256xxx4 -PG_clone twrk64f120m demo_apps/hello kds c:\tmp exmpl

Table continues on the next page...

Table 29. Commands supported in Project Cloner (continued)

		(see toolchains - toolchain - id) 4. {wrkspc} - absolute path where new project shall be created, e.g. projects workspace 5. {prjName} - name of the new project		

8.8 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.

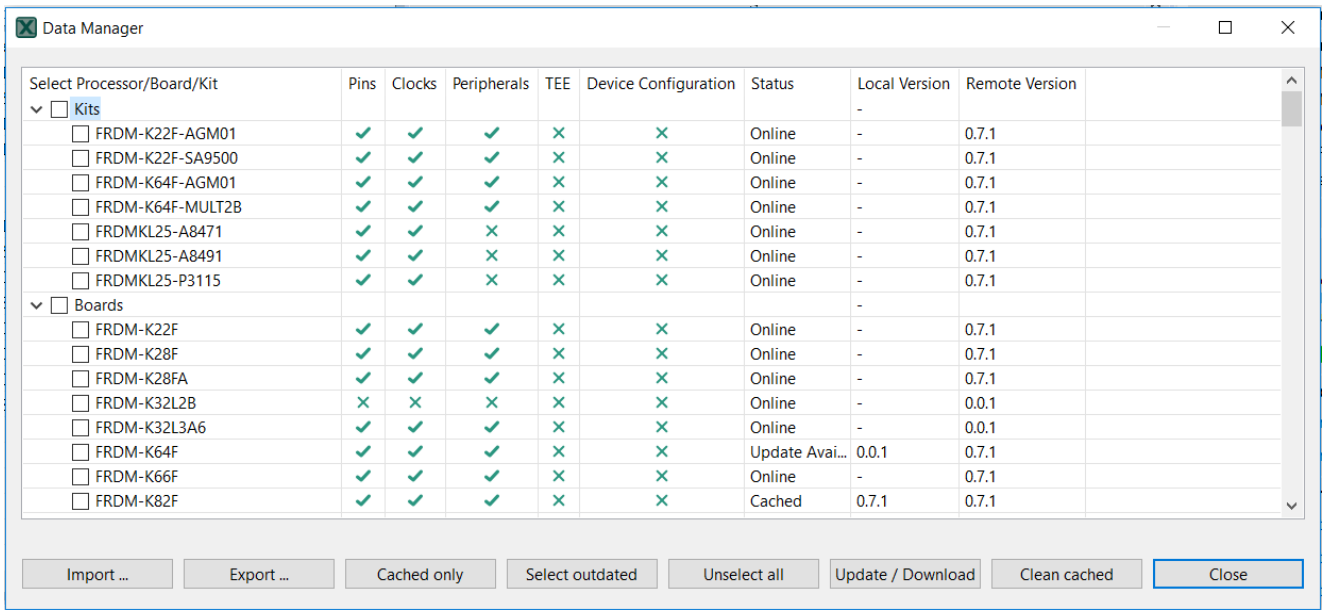


Figure 143. Data Manager

8.8.1 Working offline

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

8.8.2 Downloading data

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

NOTE

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

To download processor data, do the following:

NOTE

Internet connection is required for data download.

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

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

8.8.3 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 **Config Tools >Data Manager**.
2. In **Data Manager**, click **Export**.
3. In **Export Processor Data** window, select the processor data you want to export.
4. Click **Browse** to specify the location and name of the resulting ZIP file.
5. Click **Finish**,

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

NOTE

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

8.8.4 Importing data

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

To import data, do the following:

1. In **Menu bar**, select **Config Tools >Data Manager**.
2. In **Data Manager**, select **Import**.
3. In **Import Processor Data** dialog, click **Browse**.
4. Specify the location of the ZIP file you want to import and click **OK**.
5. Choose the data to import by selecting the checkbox in the table.
6. Click **Finish**.

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

NOTE

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

8.8.5 Updating data

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

NOTE

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

NOTE

Internet connection is required for data update.

To update cached data, do the following:

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

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

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

NOTE

Doesn't affect user templates.

Chapter 9
Support

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

How To Reach Us :

Home Page:

nxp.com

Web Support:

nxp.com/support

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

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

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

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

© NXP B.V. 2016-2021. **All rights reserved.**

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

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 1/2021

Document identifier: MCUXIDECTUG

arm