Contents

Chapter 1 Introduction ........................................................................................................... 6
  1.1 Versions ......................................................................................................................... 6

Chapter 2 User Interface ......................................................................................................... 8
  2.1 Creating, saving, and opening a configuration ................................................................. 8
    2.1.1 Creating a new configuration ....................................................................................... 8
    2.1.2 Saving a configuration ............................................................................................... 8
    2.1.3 Importing sources ..................................................................................................... 8
      2.1.3.1 Importing Board/Kit Configuration ..................................................................... 9
      2.1.3.2 Importing configuration ..................................................................................... 10
      2.1.3.3 Importing registers ......................................................................................... 10
    2.1.4 Restoring configuration from source code ............................................................... 13
  2.2 Toolbar ......................................................................................................................... 13
    2.2.1 Eclipse project selection ............................................................................................ 14
    2.2.2 Config Tools Overview ............................................................................................. 14
    2.2.3 Show Problems View ............................................................................................... 14
    2.2.4 Update code ............................................................................................................ 14
    2.2.5 Functional groups .................................................................................................... 17
      2.2.5.1 Functional group properties .............................................................................. 17
    2.2.6 Undo/Redo actions ................................................................................................. 18
    2.2.7 Selecting the tools ................................................................................................. 19
  2.3 Status bar ....................................................................................................................... 19
  2.4 Preferences .................................................................................................................... 19
  2.5 Configuration preferences ............................................................................................. 22
  2.6 Problems view ............................................................................................................... 24
  2.7 Registers view ............................................................................................................... 25
  2.8 Log view ....................................................................................................................... 27
  2.9 Config tools overview ................................................................................................... 27
  2.10 Config Tools snippets ................................................................................................. 29

Chapter 3 Pins Tool ................................................................................................................. 30
  3.1 Pins routing principle ..................................................................................................... 30
    3.1.1 Beginning with peripheral selection ......................................................................... 30
    3.1.2 Beginning with pin/internal signal selection ............................................................. 31
    3.1.3 Routing of peripheral signals .................................................................................. 31
  3.2 Example workflow ........................................................................................................ 36
  3.3 User interface ................................................................................................................ 39
    3.3.1 Pins view ................................................................................................................. 40
    3.3.2 Package ................................................................................................................... 41
    3.3.3 Peripheral Signals view .......................................................................................... 43
      3.3.3.1 Filtering in the Pins and Peripheral Signals views .............................................. 45
    3.3.4 Routing Details view ............................................................................................. 46
      3.3.4.1 Labels and identifiers ....................................................................................... 47
    3.3.5 Expansion Header ................................................................................................... 48
      3.3.5.1 Expansion Board ............................................................................................ 53
    3.3.6 Functions ............................................................................................................... 55
    3.3.7 Highlighting and color coding ................................................................................. 56
  3.4 Errors and warnings ...................................................................................................... 58
    3.4.1 Incomplete routing ................................................................................................. 58
Chapter 7 Trusted Execution Environment Tool................................................................. 95
  7.1 AHB with security extension-enabled devices................................................................. 96
    7.1.1 User Memory Regions view..................................................................................... 96
    7.1.2 Security Access Configuration view....................................................................... 98
      7.1.2.1 SAU .............................................................................................................. 98
      7.1.2.2 Interrupts...................................................................................................... 99
      7.1.2.3 Secure/Non-secure MPU............................................................................. 100
      7.1.2.4 MPC........................................................................................................... 102
      7.1.2.5 Masters/Slaves............................................................................................ 103
      7.1.2.6 Pins............................................................................................................. 104
      7.1.2.7 Miscellaneous............................................................................................. 107
    7.1.3 Memory attribution map......................................................................................... 108
      7.1.3.1 Core 0........................................................................................................ 108
      7.1.3.2 Other masters............................................................................................. 109
    7.1.4 Access Overview................................................................................................... 111
    7.1.5 Code generation................................................................................................... 112
  7.2 RDC-enabled devices............................................................................................... 113
    7.2.1 User Memory Regions view................................................................................. 113
      7.2.1.1 Access templates......................................................................................... 113
    7.2.2 Security Access Configuration view................................................................... 114
      7.2.2.1 RDC........................................................................................................... 114
        7.2.2.1.1 RDC Masters....................................................................................... 114
        7.2.2.1.2 Memory Regions............................................................................... 116
        7.2.2.1.3 Peripherals....................................................................................... 117
      7.2.2.2 XRDC2 Domains view................................................................................ 118
        7.2.2.2.1 MPU............................................................................................... 118
        7.2.2.2.2 Domains........................................................................................... 120
        7.2.2.2.3 Masters............................................................................................ 120
        7.2.2.2.4 Peripherals....................................................................................... 122
        7.2.2.2.5 Memory Regions............................................................................... 124
        7.2.2.2.6 Memory Slots.................................................................................. 125
      7.2.2.3 Miscellaneous............................................................................................. 126
    7.2.3 Memory Attribution Map...................................................................................... 126
    7.2.4 Access Overview................................................................................................... 128
    7.2.5 Domains Overview.............................................................................................. 129
    7.2.6 Code generation................................................................................................... 130

Chapter 8 Advanced Features......................................................................................... 132
  8.1 Exporting the Pins table............................................................................................ 132
  8.2 Tools advanced configuration................................................................................... 133
  8.3 Generating HTML reports....................................................................................... 133
  8.4 Exporting sources.................................................................................................... 133
  8.5 Exporting registers.................................................................................................. 134
  8.6 Command line execution.......................................................................................... 134
    8.6.1 Command Line execution - Pins Tool............................................................. 137
    8.6.2 Command Line execution - Clocks Tool......................................................... 138
    8.6.3 Command Line execution - Peripherals Tool.................................................. 139
    8.6.4 Command Line execution - TEE Tool............................................................. 140
  8.7 Managing data and working offline.......................................................................... 141
    8.7.1 Working offline.................................................................................................. 142
    8.7.2 Downloading data............................................................................................ 142
    8.7.3 Exporting data.................................................................................................. 142
    8.7.4 Importing data.................................................................................................. 143
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>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pins Tool</td>
<td>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.</td>
</tr>
<tr>
<td>Clocks Tool</td>
<td>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.</td>
</tr>
<tr>
<td>Peripherals Tool</td>
<td>Enable you to configure the initialization for the MCUXpresso SDK drivers.</td>
</tr>
<tr>
<td>Device Configuration Tool</td>
<td>Enables you to generate a Device Configuration Data (DCD) image using the format and constrains specified in the Boot ROM reference manual.</td>
</tr>
<tr>
<td>TEE (Trusted Execution Environment) Tool</td>
<td>Enables you to configure security policies of memory areas, bus masters, and peripherals, in order to isolate and safeguard sensitive areas of your application.</td>
</tr>
</tbody>
</table>

1.1 Versions

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

NOTE

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

TIP

To use the desktop tool in the offline mode, create a configuration for the given processor while online. The tool will then store the processors locally in the user folder and enable faster access and offline use. Otherwise, it is possible to download and export the data using the Export menu.
Figure 1. Desktop version of Pins tool
Chapter 2
User Interface

2.1 Creating, saving, and opening a configuration

In this context, configuration stands for common tools settings stored in an MEX (Microcontrollers Export Configuration) file. This file contains settings of all available tools. The folder with the saved MEX file must contain exactly one project file to be able to parse the toolchain project.

2.1.1 Creating a new configuration

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

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

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

2.1.2 Saving a configuration

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

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

2.1.3 Importing sources

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

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.
3. Click **Next**.

4. On the next page, click **Browse** to specify the location of the source file.

5. Select the source file that you wish to import and click **Open**.

6. On the next page, select which functional groups to import (based on tools) by selecting the checkbox in the left column.

7. Define how to import the functional groups by selecting one of the two available options in the dropdown menu in the right column:
   - **Rename** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, it is automatically renamed to the indexed one. For example, if `BOARD_InitPins` exists in the configuration then the imported function is renamed to `BOARD_InitPins1`.
   - **Overwrite** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, then the existing one is replaced with the imported one.

8. Click **Finish**.

---

**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 Board/Kit Configuration

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

To import board/kit configuration, do the following:

1. In the **Menu bar**, select **File > Import...**.
2. In the **Import** wizard, select **MCUXpresso Config Tools > Import Board/Kit Configuration**.

3. Click **Next**.

4. On the next page, select the board/kit variant from the dropdown menu.

5. Select which functional groups to import (based on tools) by selecting the checkbox in the left column.

6. Define how to import the functional groups by selecting one of the two available options in the dropdown menu in the right column:
   - **Rename** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, it is automatically renamed to the indexed one. For example, if `BOARD_InitPins` exists in the configuration then the imported function is renamed to `BOARD_InitPins1`.
   - **Overwrite** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, then the existing one is replaced with the imported one.

7. Click **Finish**.

### 2.1.3.2 Importing configuration

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

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

2. In the **Import** wizard, select **MCUXpresso Config Tools > Import configuration (*.mex)**.

3. Click **Next**.

4. On the next page, click **Browse** to specify the location of the registers file.

5. Select the MEX file that you wish to import and click **Open**.

6. On the next page, select which functional groups to import (based on tools) by selecting the checkbox in the left column.

7. Define how to import the functional groups by selecting one of the two available options in the dropdown menu in the right column:
   - **Rename** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, it is automatically renamed to the indexed one. For example, if `BOARD_InitPins` exists in the configuration then the imported function is renamed to `BOARD_InitPins1`.
   - **Overwrite** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, then the existing one is replaced with the imported one.

8. Click **Finish**.

### 2.1.3.3 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.
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.
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.

---

**Figure 4. Import Registers Configuration**

<table>
<thead>
<tr>
<th>Reg. Name</th>
<th>Set Value</th>
<th>Reset Value</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANACTRL_FRO192M_CT</td>
<td>0x0780d07a</td>
<td>0x0780d07a</td>
<td>192MHz Free Run</td>
</tr>
<tr>
<td>ANACTRL_XO32M_CTRL</td>
<td>0x?121428?</td>
<td>0x?021428?</td>
<td>High speed Crystal</td>
</tr>
<tr>
<td>PMC_OSTIMER</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>OS Timer control</td>
</tr>
<tr>
<td>PMC_PDRUNCFG0</td>
<td>0x????????4</td>
<td>0x????????4</td>
<td>Controls the power</td>
</tr>
<tr>
<td>PMC_RTCOSC32K</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>RTC 1 KHz and 1 MHz</td>
</tr>
<tr>
<td>RTC_CTRL</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>RTC control register</td>
</tr>
<tr>
<td>SYSCON_ADCCLKDIV</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>ADC clock divider</td>
</tr>
<tr>
<td>SYSCON_ADCCLKSEL</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>ADC clock source</td>
</tr>
<tr>
<td>SYSCON_AHBCLKDIV</td>
<td>0x????????0</td>
<td>0x????????0</td>
<td>System clock divide</td>
</tr>
<tr>
<td>SYSCON_CLKOUTDIV</td>
<td>0x????????0</td>
<td>0x????????0</td>
<td>CLKOUT clock divide</td>
</tr>
<tr>
<td>SYSCON_CLKOUTSEL</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>CLKOUT clock source</td>
</tr>
<tr>
<td>SYSCON_CLOCK_CTRL</td>
<td>0x????????21</td>
<td>0x????????101</td>
<td>Various system clock</td>
</tr>
<tr>
<td>SYSCON_CTIMERCLKSEL</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>CTimer 0 clock select</td>
</tr>
<tr>
<td>SYSCON_CTIMERCLKSEL</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>CTimer 1 clock select</td>
</tr>
<tr>
<td>SYSCON_CTIMERCLKSEL</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>CTimer 2 clock select</td>
</tr>
<tr>
<td>SYSCON_CTIMERCLKSEL</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>CTimer 3 clock select</td>
</tr>
<tr>
<td>SYSCON_CTIMERCLKSEL</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>CTimer 4 clock select</td>
</tr>
<tr>
<td>SYSCON_FCCCLKSEL0</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>Flexcomm Interface</td>
</tr>
<tr>
<td>SYSCON_FCCCLKSEL1</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>Flexcomm Interface</td>
</tr>
<tr>
<td>SYSCON_FCCCLKSEL2</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>Flexcomm Interface</td>
</tr>
<tr>
<td>SYSCON_FCCCLKSEL3</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>Flexcomm Interface</td>
</tr>
<tr>
<td>SYSCON_FCCCLKSEL4</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>Flexcomm Interface</td>
</tr>
<tr>
<td>SYSCON_FCCCLKSEL5</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>Flexcomm Interface</td>
</tr>
<tr>
<td>SYSCON_FCCCLKSEL6</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>Flexcomm Interface</td>
</tr>
<tr>
<td>SYSCON_FCCCLKSEL7</td>
<td>0x?????????</td>
<td>0x?????????</td>
<td>Flexcomm Interface</td>
</tr>
<tr>
<td>SYSCON_FLEXFRG0CTRL</td>
<td>0x?????0ff</td>
<td>0x?????0ff</td>
<td>Fractional rate divide</td>
</tr>
<tr>
<td>SYSCON_FLEXFRG1CTRL</td>
<td>0x?????0ff</td>
<td>0x?????0ff</td>
<td>Fractional rate divide</td>
</tr>
<tr>
<td>SYSCON_FLEXFRG2CTRL</td>
<td>0x?????0ff</td>
<td>0x?????0ff</td>
<td>Fractional rate divide</td>
</tr>
<tr>
<td>SYSCON_FLEXFRG3CTRL</td>
<td>0x?????0ff</td>
<td>0x?????0ff</td>
<td>Fractional rate divide</td>
</tr>
<tr>
<td>SYSCON_FLEXFRG4CTRL</td>
<td>0x?????0ff</td>
<td>0x?????0ff</td>
<td>Fractional rate divide</td>
</tr>
<tr>
<td>SYSCON_FLEXFRG5CTRL</td>
<td>0x?????0ff</td>
<td>0x?????0ff</td>
<td>Fractional rate divide</td>
</tr>
</tbody>
</table>

**Creating, saving, and opening a configuration**
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

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

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

```cpp
/* 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 is possible to reimport the clock settings into the tool using the following steps.

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

2.2 Toolbar

The toolbar is on the top of the window and includes buttons/menus of frequently used actions common to all tools. See the following sections for more information.
<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config Tools Overview</strong></td>
<td>Open the <strong>Overview</strong> dialog with information about currently used tools.</td>
</tr>
<tr>
<td><strong>Show Problems View</strong></td>
<td>Open the <strong>Problems</strong> view.</td>
</tr>
<tr>
<td><strong>Update Code</strong></td>
<td>Open the update dialog allowing you to update generated peripheral initialization code directly within specified toolchain project.</td>
</tr>
<tr>
<td><strong>Generate Code</strong></td>
<td>Regenerate source code when “Enable Code Preview” preference is disabled.</td>
</tr>
<tr>
<td><strong>Functional group selection</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Call from default initialization</strong></td>
<td>Set the current functional group to be initialized by the default initialization function.</td>
</tr>
<tr>
<td><strong>Functional group properties</strong></td>
<td>Open the <strong>Functional group properties</strong> dialog to modify name and other properties of the function group.</td>
</tr>
<tr>
<td><strong>Tool selection</strong></td>
<td>Display icons of individual tools. Use them to switch between tools.</td>
</tr>
<tr>
<td><strong>Undo/Redo</strong></td>
<td>Undo/Redo last action.</td>
</tr>
</tbody>
</table>

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 \textit{change} link.
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 7. Show differences

Figure 8. Update Code dropdown menu

NOTE
The generated code is always overwritten.
The **Update Code** action is enabled under following conditions:

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

### 2.2.5 Functional groups

Every **Pins/Clocks/Peripherals/TEE** 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](image)

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

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Toggle &quot;Called from default initialization function&quot; feature (in source code)</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Opens the <strong>Functional group properties</strong> window</td>
</tr>
</tbody>
</table>

### 2.2.5.1 Functional group properties

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

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

Functional group properties are specific for individual Config Tools:

The Pins tool:

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

Clocks tool:

• **Set custom #define prefix** - If this property is set, the custom prefix is used for macros define in `pin_mux.h`. Otherwise the name of the functional group is used as the prefix.

• **Prefix** - The custom prefix string. If it is empty, no prefix is used.

• **Other settings** - The processor-specific settings are specific for each processor. See the tooltips for details.

Peripherals tool:

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

TEE tool:

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

![Functional group properties](image)

**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**.
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.
Several settings are available.
## Table 5. Preferences

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Line ending style</strong></td>
<td>Select between Windows (CR + LF), Linux/Mac (LF), or Default (based on host).</td>
</tr>
<tr>
<td><strong>Generate files read-only</strong></td>
<td>Prevent modifying the source files unintentionally. Generated source files are marked as read-only.</td>
</tr>
<tr>
<td><strong>Generate source folder</strong></td>
<td>At build time, automatically create a folder including source files.</td>
</tr>
<tr>
<td><strong>Create empty configuration if no yaml is available</strong></td>
<td>Generates a configuration even if no yaml is present.</td>
</tr>
<tr>
<td><strong>Always overwrite files without asking</strong></td>
<td>Update existing files automatically, without prompting.</td>
</tr>
<tr>
<td><strong>Always show details before Update Code</strong></td>
<td>Review changes before the project is updated.</td>
</tr>
<tr>
<td><strong>Undo history size</strong></td>
<td>Enter the maximum number of steps that can be undone. Enter 0 to disable.</td>
</tr>
<tr>
<td><strong>Proxy connection</strong></td>
<td><strong>Direct</strong> – Connect directly and avoid a proxy connection.</td>
</tr>
<tr>
<td></td>
<td><strong>Native</strong> – Use system proxy configuration for network connection.</td>
</tr>
<tr>
<td><strong>NOTE</strong></td>
<td>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 <code>&lt;install_dir&gt;/bin/</code> folder. Make sure that the file contains the following lines:</td>
</tr>
<tr>
<td></td>
<td>• Djava.net.useSystemProxies=true (already present by default)</td>
</tr>
<tr>
<td></td>
<td>• Dhttp.proxyHost=&lt;somecompany.proxy.net&gt;</td>
</tr>
<tr>
<td></td>
<td>• Dhttp.proxyPort=80</td>
</tr>
<tr>
<td><strong>NOTE</strong></td>
<td>Authentication is not supported.</td>
</tr>
<tr>
<td><strong>Work Offline</strong></td>
<td>Disable both the connection to NXP cloud and the download of processor/board/kit data.</td>
</tr>
<tr>
<td><strong>Processor data update</strong></td>
<td>Select from the following options:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Auto Update</strong> – Update the processor data automatically.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Manual</strong> – Update processor data after confirmation.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Disabled</strong> – Disable processor data update.</td>
</tr>
<tr>
<td><strong>Check for application updates</strong></td>
<td>Check for application updates on a weekly basis</td>
</tr>
<tr>
<td><strong>Show pin label &amp; identifier table columns (Pins tool)</strong></td>
<td>Select to show the pin label and the label identifier in the relevant views.</td>
</tr>
</tbody>
</table>

*Table continues on the next page...*
Table 5. Preferences (continued)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Require Identifier for Pins (Pins Tool)</td>
<td>Controls generation of pins “Identifier” related warnings. With this preference enabled, warnings will be generated for bidirectional signals that have no Identifier set.</td>
</tr>
<tr>
<td>Show Overview window on opening configuration for the first time</td>
<td>Open the Overview dialog on opening configuration for the first time.</td>
</tr>
<tr>
<td>Automatically load last configuration on startup</td>
<td>Avoid the startup window and load the last used configuration instead.</td>
</tr>
</tbody>
</table>
| Enable Code Preview                                    | Controls how the code is generated. When this preference is enabled, code generation is performed automatically after every change in the configuration and the Code Preview is updated accordingly. When this preference is disabled, code generation is stopped, warning message is displayed in Code Preview window, and the action can be manually triggered by using one of the available options:  
  • By pressing the "generate code" link highlighted in the warning message from the Code Preview window.  
  • By pressing the Update Code button in the toolbar, where code update is preceded by code generation. |
| Component template folder                              | The path to the folder with component templates. Keep empty to use the default path. The default path is to folder component_templates in data of the Config tools. |
| Allow component migrations                             | When a configuration associated with a toolchain project is open, the peripheral tool automatically checks if the configuration components match the project and suggests a migration if they are not. |
| Enable animations                                       | Enables animations in the user interface, such as smoother scrolling or opening a drop-down menu.                                               |

2.5 Configuration preferences

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

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

Table 6. Configuration Preferences

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate boot init only</td>
<td>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.</td>
</tr>
<tr>
<td>Generate YAML</td>
<td>Generate YAML into C sources files.</td>
</tr>
<tr>
<td>Custom source file copyright header</td>
<td>Add a custom copyright header to generated source files that do not already contain copyright.</td>
</tr>
<tr>
<td>Generate extended information into header file</td>
<td>Generate extended information into the header file. For projects created in earlier MCUXpresso versions, this option is selected by default.</td>
</tr>
<tr>
<td>Generate defines of clock registers</td>
<td>This feature is disabled by default (Edit-&gt;Configuration Preferences). The new registers.h file with registers defines is generated in the Code Preview tab. The custom prefix can be defined in the Functional group properties.</td>
</tr>
</tbody>
</table>

Table continues on the next page...
Table 6. Configuration Preferences (continued)

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

**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.

![Problems view](image)

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

![Problems view](image)

**Table 7. Problems view**

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

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

Table 8. Filter buttons

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Button]</td>
<td>Enables the Validate boot init only preference. See Configuration preferences section for details.</td>
</tr>
<tr>
<td>![Button]</td>
<td>Filters messages in the Problems view. If selected, only problems for the active tool are displayed. See Configuration preferences section for details.</td>
</tr>
</tbody>
</table>

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.
The Registers view contains several items.

Table 9. Registers

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peripheral filter drop-down list</td>
<td>List the registers only for the selected peripheral. Select all to list registers for all the peripherals.</td>
</tr>
<tr>
<td>Show modified registers only checkbox</td>
<td>Hide the registers that are left in their after-reset state or are not configured.</td>
</tr>
<tr>
<td>Text filter</td>
<td>Filter content by text.</td>
</tr>
</tbody>
</table>

The following table lists the color highlighting styles used in the Registers view.
Table 10. Color codes

<table>
<thead>
<tr>
<th>Color</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yellow background</td>
<td>Indicates that the bitfield has been affected by the last change made in the tool.</td>
</tr>
<tr>
<td>Gray text color</td>
<td>Indicates that the bitfield is not edited and the value is the after-reset value.</td>
</tr>
<tr>
<td>Black text</td>
<td>Indicates the bit-fields that the tool modifies.</td>
</tr>
</tbody>
</table>

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

## 2.8 Log view

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

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

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

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

![Log view](image)

**Figure 15. Log view**

## 2.9 Config tools overview

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

**Config Tools Overview** contains several items.
Table 11. Config Tools Overview

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration – General Info</td>
<td>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 <strong>Import additional settings into current configuration</strong> button.</td>
</tr>
<tr>
<td>Configuration – HW Info</td>
<td>Displays the processor, part number, core, and SDK-version information of the current configuration.</td>
</tr>
<tr>
<td>Project</td>
<td>Displays toolchain project information.</td>
</tr>
<tr>
<td>Pins/Clocks/Peripherals/TEE/Device Config</td>
<td>Displays basic information about the <strong>Pins</strong>, <strong>Clocks</strong>, <strong>Peripherals</strong>, <strong>TEE</strong>, and <strong>Device Configuration</strong> tools.</td>
</tr>
</tbody>
</table>

**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 12. Config Tools Overview

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generated code</td>
<td>Contains the list of source-code files. Click the links to open the files in the <strong>Code Preview</strong> view.</td>
</tr>
<tr>
<td>Functional groups</td>
<td>Contains the list of the currently active functional groups. To select the groups in the <strong>Functional groups</strong> tab in the toolbar, select the relevant links.</td>
</tr>
</tbody>
</table>
2.10 Config Tools snippets

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

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

![Pins tool](image)

### 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 that 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".

![Image](image.png)

**Figure 18. Defining routing path**

### 3.1.2 Beginning with pin/internal signal selection

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

1. Select the pin/internal signal (**Routed pin/signal**).
2. Select one of the available **Peripherals**.
3. For the selected peripheral, select one of the available **Signals**.

Items in **Peripheral** column in **Routing Details** view have the following symbols:

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

### NOTE

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

### 3.1.3 Routing of peripheral signals

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

Three types of peripheral signal routing are available:

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

![Figure 19. Selecting the peripheral (B)](image1)

<table>
<thead>
<tr>
<th>#</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Arrow</th>
<th>Routed pin/signal</th>
<th>Label</th>
<th>Identifier</th>
<th>Direction</th>
<th>GPIO in</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ADC</td>
<td></td>
<td></td>
<td></td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>CAN0</td>
<td></td>
<td></td>
<td></td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>CMPA</td>
<td></td>
<td></td>
<td></td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>CMPB</td>
<td></td>
<td></td>
<td></td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

![Figure 20. Selecting the input node (B)](image2)

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

![Figure 21. Selecting the output signal](image3)

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

![Figure 22. Result](image4)

It is necessary to select the ADC peripheral where the signal leads to (input in ADC). It is a limitation of the Pins tool that the signal is not listed for the PWM peripheral (output). Notice the direction of the signal in the Arrow column.
2. Routing the signal from a pin on the package to internal peripheral input signal through an inter-peripheral crossbar

**NOTE**

Only if a crossbar switch is present.

The signal leads from a pin on the package (XB_IN) connected through an inter-peripheral crossbar, to an internal peripheral (B) input node. In other words, the signal leads from XB_IN to B (XB_IN > B). To configure a signal in this way, perform the following steps (routing pin 55 using XB_IN6 to EVTG0 input A (XB_IN6 > EVTG0) used as example):

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

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

![Selecting the peripheral (B)](image)

Figure 23. Selecting the peripheral (B)>

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

![Selecting the input node (B)](image)

Figure 24. Selecting the input node (B)

d. Select the XB_IN pin from the drop-down list in the **Routed pin/signal** column.
Once the configuration is done, the row looks like this:

<table>
<thead>
<tr>
<th>#</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Ar...</th>
<th>Routed pin/signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>55</td>
<td>EVTG</td>
<td>EVTG0, A</td>
<td>-</td>
<td>[55] XB_IN6</td>
</tr>
</tbody>
</table>

Figure 26. Result

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

3. Routing the signal from internal peripheral (A) output to a pin via inter-peripheral crossbar

   **NOTE**
   Only if a crossbar switch is present.

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

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

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

![Figure 27. Selecting the peripheral (A)](image)

![Routing Details for BOARD...](image)

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Signal</th>
<th>Ar...</th>
<th>Routed...</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVTG0, A</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG0, B</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG0, C</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG0, D</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG0_OUT, A</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG0_OUT, B</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG1, A</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG1, B</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG1, C</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG1, D</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG1_OUT, A</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG1_OUT, B</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
</tbody>
</table>

Figure 28. Selecting the output signal (A)

![Routing Details for BOARD...](image)

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Signal</th>
<th>Ar...</th>
<th>Routed...</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVTG0, A</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG0, B</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG0, C</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG0, D</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG0_OUT, A</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG0_OUT, B</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG1, A</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG1, B</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG1, C</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG1, D</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG1_OUT, A</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
<tr>
<td>EVTG1_OUT, B</td>
<td></td>
<td></td>
<td>n/a</td>
</tr>
</tbody>
</table>

Figure 27. Selecting the peripheral (A)

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

![Routing Details for BOARD...](image)
Once the configuration is done, the row looks like this:

<table>
<thead>
<tr>
<th>#</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Ar...</th>
<th>Routed...</th>
<th>Label</th>
<th>Identifier</th>
<th>Direction</th>
<th>GPIO int.</th>
<th>GPIO ini.</th>
<th>Siel...</th>
</tr>
</thead>
<tbody>
<tr>
<td>87</td>
<td>EVTG</td>
<td>EVTG0...</td>
<td></td>
<td></td>
<td>-&gt;</td>
<td>[87] XB_OUT4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 30. Result

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

3.2 Example workflow

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

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

You can use the generated files with the application code.

1. In the Pins view on the left, select the UART3_RX and TX signals. For it, you can click into the cells to make them ‘green’.
2. In the Routing Details view, select the Output direction for the TX and PTB20 signals.

![Figure 32. Selecting direction](image)

---

**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.
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 that you want to export sources for and select the **Export Source Files** option. **Export**, select the **Export Source Files** option.
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.
3.3.1 Pins view

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

NOTE
Power Groups are not supported for all processors.
This view shows the list of all the pins available on a given device. The **Pin name** column shows the default name of the pin, or if the pin is routed. The next columns are optional. They are **Label**, **Identifier**, and **Expansion header connections** (one column for each expansion header). The pin name is changed to show appropriate function for selected peripheral if routed. The next column of the table shows peripherals and signals and pin name(s) on given peripheral. Peripherals with few items are cumulated in the last column.

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

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

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

Select a row to open a drop-down list that offers the following options:

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

**TIP**

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

### 3.3.2 Package

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

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Zoom in" /></td>
<td>Zoom in package image.</td>
</tr>
<tr>
<td><img src="image2.png" alt="Zoom out" /></td>
<td>Zoom out package image.</td>
</tr>
<tr>
<td><img src="image3.png" alt="Rotate" /></td>
<td>Rotate package image.</td>
</tr>
<tr>
<td><img src="image4.png" alt="Bottom" /></td>
<td>Show pins as you can see it from the bottom. This option is available on BGA packages only.</td>
</tr>
<tr>
<td><img src="image5.png" alt="Top" /></td>
<td>Show pins as you can see it from the top. This option is available on BGA packages only.</td>
</tr>
<tr>
<td><img src="image6.png" alt="Resources" /></td>
<td>Show resources. This option is available on BGA packages only.</td>
</tr>
<tr>
<td><img src="image7.png" alt="Switch" /></td>
<td>Switch package.</td>
</tr>
<tr>
<td><img src="image8.png" alt="Legend" /></td>
<td>Package legend.</td>
</tr>
<tr>
<td><img src="image9.png" alt="Select" /></td>
<td>Select the information displayed as pin labels. This option is not available on BGA packages.</td>
</tr>
</tbody>
</table>

**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 14. Status codes

<table>
<thead>
<tr>
<th>Color code</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image10.png" alt="Error" /></td>
<td>Error</td>
</tr>
<tr>
<td><img src="image11.png" alt="Configured" /></td>
<td>Configured</td>
</tr>
<tr>
<td><img src="image12.png" alt="Not configured" /></td>
<td>Not configured</td>
</tr>
<tr>
<td><img src="image13.png" alt="Warning" /></td>
<td>Warning</td>
</tr>
<tr>
<td><img src="image14.png" alt="Dedicated" /></td>
<td>Dedicated: Device is routed by default and has no impact on the generated code.</td>
</tr>
</tbody>
</table>
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.
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.

![Filtering Controls](image)

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 42. Routing Details](image)

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

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

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

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

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

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

NOTE
When you enter the search text, it also searches the text in the full pin names displays rows that contain the search text.

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

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

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

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

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

To clear the table, type 0.

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

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

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

The gray background indicates read-only items.

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

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

**Figure 43. Routing Details Legend**

<table>
<thead>
<tr>
<th>Settings related to hardware features of pins are generally offering:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>Reset</strong> - default (after reset) value - no code will be generated unless it is combined with some other initialization</td>
</tr>
<tr>
<td>- <strong>No init</strong> - Property will not be initialized - no conflict validations will be performed and no code will be generated unless it is combined with some other initialization</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Columns and values descriptions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>#</strong> - Package pin number/coordinate</td>
</tr>
<tr>
<td>- <strong>Peripheral</strong> - Name of the selected peripheral module</td>
</tr>
<tr>
<td>- <strong>Signal</strong> - Name of the selected peripheral signal/signal function</td>
</tr>
<tr>
<td>- <strong>Arrow</strong> - Arrow indicating direction of the signal</td>
</tr>
<tr>
<td>- <strong>Routed pin/signal</strong> - Name of the pin or internal signal</td>
</tr>
<tr>
<td>- <strong>Label</strong> - Pin label with max length of 128 characters: By submitting empty label the identifier is deleted as well</td>
</tr>
<tr>
<td>- <strong>Identifier</strong> - Pin identifier used for #define code generation</td>
</tr>
<tr>
<td>- <strong>Direction</strong> - Pin direction</td>
</tr>
<tr>
<td>- <strong>GPIO initial state</strong> - GPIO output initial state</td>
</tr>
<tr>
<td>a) <strong>Logical 0</strong></td>
</tr>
<tr>
<td>b) <strong>Logical 1</strong></td>
</tr>
<tr>
<td>- <strong>GPIO interrupt</strong> - It allows configuration of interrupt/DMA request for the pin. To enable interrupt event (DMA transfer) from the pin, interrupt</td>
</tr>
</tbody>
</table>

- The value shown using italic indicates the after-reset value. The real value may be different from the after reset value, if configured in other functions.

Use the drop-down menu to select the required value.

- If you select the same value as the after-reset value, the tool will always generate code to set this feature.

Use the drop-down "Reset" value to reset the value to its after-reset state.

- If an item does not support reset to after reset value, the **Reset** menu is not available.

- The first row shows pin number or coordinate on BGA package.

### 3.3.4.1 Labels and identifiers

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

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

To setup the **Labels and Identifier** columns visibility, select **Window > Preferences** > **MCUXpresso Config tools** from the **Menu bar**, and select the **Show pin label & identifier table columns (Pins tool)** checkbox.

The pin identifier is used to generate the **#define** in the pin_mux.h file. However, it is an optional parameter. If the parameter is not defined, the code for **#define** is not generated. 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.
In this case, it is possible to select from values if the pin is routed. See Routing Details.

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.

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

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

### 3.3.5 Expansion Header

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

Certain boards, such as LPCXpresso55S69, come with preconfigured expansion headers.
The expansion header is not automatically preset for every supported device. If the header is not preconfigured, follow these steps to create and modify an expansion header configuration:

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

![Figure 49. Adding new expansion header](image)

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.
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**.
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.
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.

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.
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**.
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 Functions

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

The tool allows to creates multiple functions that can be used to configure pin muxing.
The usage of pins is indicated by 50% opacity in Pins, Peripheral Signals, and Package views. Each function can define a set of routed pins or re-configure already routed pins.

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

### 3.3.7 Highlighting and color coding

You can easily identify routed pins/peripherals in the package using highlighting. By default, the current selection (pin/peripheral) is highlighted in the Package view.

- The pin/peripheral is highlighted by yellow border around it in the Package view. If the highlighted pin/peripheral is selected, then it has a blue border around it.
- Red indicates that the pin has an error.
- Green indicates that the pin is muxed or used.
- Light gray indicates that the pin is available for mux, but is not muxed or used.
- Dark gray indicates that the pin/peripheral is dedicated. It is routed by default and has no impact on generated code.
Figure 57. Highlighting and color coding

<table>
<thead>
<tr>
<th>#</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Route to</th>
<th>Label</th>
<th>Identifier</th>
<th>Direction</th>
<th>Slow rate</th>
<th>Open drain</th>
<th>Drive strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>CMP0</td>
<td>IN_0</td>
<td>ADC1</td>
<td>ADC1_SE4</td>
<td>J1[7]</td>
<td>n/a</td>
<td>Fast</td>
<td>Disabled</td>
<td>Low</td>
</tr>
<tr>
<td>26</td>
<td>CMP1</td>
<td>IN_1</td>
<td>VREF_OUT</td>
<td>VREF_OUT</td>
<td>J2[9]</td>
<td>n/a</td>
<td>Fast</td>
<td>Disabled</td>
<td>Low</td>
</tr>
<tr>
<td>6</td>
<td>UART3</td>
<td>RX</td>
<td>UART3_RX</td>
<td>J13[1]</td>
<td>Not Specified</td>
<td>Input</td>
<td>Fast</td>
<td>Disabled</td>
<td>Low</td>
</tr>
<tr>
<td>6</td>
<td>FTM0</td>
<td>CH_0</td>
<td>FTM0_CH0</td>
<td>J1[5]</td>
<td>Not Specified</td>
<td>Not Specified</td>
<td>Slow</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Figure 58. Pins conflicts

<table>
<thead>
<tr>
<th>#</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Route to</th>
<th>Label</th>
<th>Identifier</th>
<th>Direction</th>
<th>Slow rate</th>
<th>Open drain</th>
<th>Drive strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>GPIOE</td>
<td>GPIOC 3</td>
<td>PTI25</td>
<td>J12[1]/J12[4]/LEDGRB_GREEN</td>
<td>Not Specified</td>
<td>Input</td>
<td>Slow</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>71</td>
<td>FTM0</td>
<td>CH_0</td>
<td>FTM0_CH0</td>
<td>J1[5]</td>
<td>n/a</td>
<td>Output</td>
<td>Fast</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Figure 59. Warnings

- Package view
— Click the peripheral or use the pop-up menu to highlight peripherals:
  ◦ and all allocated pins (to selected peripheral).
  ◦ or all available pins if nothing is allocated yet.
— Click the pin or use the pop-up menu to highlight the pin and the peripherals.
— Click outside the package to cancel the highlight.

• **Peripherals / Pins** view
  — The peripheral and pin behaves as described above.

### 3.4 Errors and warnings

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

If an error or warning is encountered, the conflict in the **Routing Details** view is represented in the first column of the row and the error/warning is indicated in the cell, where the conflict was created. The last two rows in the figure above show the peripheral/signal where the erroneous configuration occurs. The detailed error/warning message appears as a tooltip.

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

#### 3.4.1 Incomplete routing

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

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

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

**NOTE**

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

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

Code Preview automatically highlights differences between the current and immediately preceding iteration of the code. You can choose between two modes of highlighting by clicking 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 62. Generated code

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

```c
#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*:

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

3.7 Full initialization of pins

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

3.8 Create Default Routing

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

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

In the created functional group, the Full initialization function of the pins feature will be set. The electrical properties of pins will be set to their after-reset state.
Figure 63. Create the Default Routing
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 fulfill 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.
• Register values define generation of C.

4.2 User interface overview
The Clocks tool is integrated and runs within the MCUXpresso Config Tools framework.
4.3 Clock configuration

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

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

**NOTE**

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

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

**4.4 Global settings**

Global settings, such as Run Mode and MCG mode, influence the entire clock system. It is recommended to set them first. Global settings can be modified in Clock Table, Clock Diagram, and Details views, and the Functional group properties dialog.
Global settings can be changed at any time.

Figure 66. Global settings

4.5 Clock sources

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

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

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

Figure 67. External clock source configuration

NOTE

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

4.6 Setting states and markers

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

Table 15. Setting states and markers

<table>
<thead>
<tr>
<th>State/Style/Marker</th>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error marker</td>
<td>🔴</td>
<td>Indicates that there is an error in the settings or something related to it. See the tooltip of the setting for details.</td>
</tr>
<tr>
<td>Warning marker</td>
<td>🔴</td>
<td>Indicates that there is a warning in the settings or something related to it. See the tooltip of the setting for details.</td>
</tr>
<tr>
<td>Lock icon</td>
<td>🔒</td>
<td>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.</td>
</tr>
</tbody>
</table>

Table continues on the next page...
Table 15. Setting states and markers (continued)

<table>
<thead>
<tr>
<th>State/Style/Marker</th>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
<td>The clock element settings that cannot be automatically adjusted by the tool keep their value as is and do not allow locking. They are: clock sources, clock selectors, and configuration elements.</td>
</tr>
<tr>
<td>Yellow background</td>
<td><img src="image" alt="100 MHz" /></td>
<td>Indicates that the field is directly or indirectly changed by the previous user action.</td>
</tr>
<tr>
<td>Gray text</td>
<td><img src="image" alt="FCTRIM" /></td>
<td>Indicates that the value of setting does not actively influence the clock. It is disabled or relates to an inactive clock element. For example, on the clock path following the unavailable clock source or disabled element. The frequency signal also shows the text “inactive” instead of frequency. The value is also gray when the value is read-only. In such a state, it is not possible to modify the value.</td>
</tr>
</tbody>
</table>

### 4.7 Frequency settings

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

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

- **Frequency** – Indicates that a clock signal is active and the output is fed with the shown frequency. The tool automatically chooses the appropriate frequency units. In case the number is too long or has more than three decimal places, it is shortened and only two decimal places are shown, followed by an ellipsis (‘…’), indicating that the number is longer.

- **“Inactive” text** – Indicates that no clock signal flows into the clock output or is disabled due to some setting.

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

![Core clock](image)

Figure 68. Setting the core clock frequency

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

![Core clock](image)

Figure 69. Tool attains the required frequency

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

![Core clock](image)

Figure 70. Tool encounters problem

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

---

**NOTE**

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

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

- **Lock/Unlock** – Removes a lock on the frequency which enables the tool to change any valid value that satisfies all other requirements, limits, and constraints.

- **Find Near Valid Value** – Tries to find a valid frequency that lies near the specified value, in case the tool failed in reaching the requested frequency.

- **Advanced resolver for (Clock output)** - Invokes more advanced search for the valid settings that fulfills the requirements. It may take significant time so a progress dialog is shown. If the resolver is not successful, the user is informed about it. This command can alter clock selectors and modify various other clock settings on the clock path. If the result is not satisfactory, use the **UNDO** command to return to the original state.

- **Edit settings of: (element)** – Invokes the floating view with the settings for a single element.

- **Edit all settings** – Invokes the floating view with all the settings for an element.

- **Edit settings on the path to: (clock output)** – Invokes the floating view with the settings for all elements on the clock path leading to the selected clock output.

![Image 83x350 to 529x499](image)

Figure 71. Pop-up commands for outputs in the clocks table view

- **Unlock**
  - **Find Near Valid Value**
  - **Advanced resolver for LPO clock**
  - **Edit settings on the path to: LPO clock**
  - **Edit all settings**

![Image 135x190 to 477x318](image)

Figure 72. Pop-up commands for outputs in the clocks diagram

4.7.2 Frequency precision

For locked frequency settings (where the user requests a specific value) the frequency precision value is also displayed. By default, the value is 0.1 % but can be individually adjusted by clicking the value.
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. It identifies the dependencies and the influences when there is a change in the clock source or elements on a shared clock path.

![Clock Sources Table](image)

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

- **Display clock-element information** - Point the mouse cursor at the clock element to display general clock-element information.

- **View the clock-element in Clocks Diagram or Clocks Table** - Left-click on a clock element to highlight it in the Clocks Diagram or Clocks Table views, depending on which is currently active.

- **View detailed clock-element information** - Double-click a clock element to display element details, as well as highlight the element in Clocks Diagram or Clocks Table, depending on which is currently active. You can also view element details by clicking the Open in new window button in the upper right corner of the Details view.

- **Modify clock-element settings** - Left-click in the Value column to change clock element value, such as frequency, or select an option from the dropdown menu.

- **Lock/unlock clock elements** - Right-click on a clock element to lock/unlock the element.

- **Filter for active/locked/erroneous clock elements** - Use the buttons in the upper-right corner of the Details view to filter for active/locked/erroneous clock elements, or to remove all current filters.
4.10 Clocks diagram

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

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

![Clocks diagram](image)

Figure 76. Clocks diagram

4.10.1 Mouse actions in diagram

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

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

- **Edit settings of: {element}** – Invokes the floating view with the settings for a single element.
- **Edit all settings** – Invokes the floating view with all the settings for an element.
- **Edit settings on the path to: {clock output}** – Invokes the floating view with the settings for all elements on the clock path leading to the selected clock output.

4.10.2 Color and line styles

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

The color and line styles can indicate:

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

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

4.10.3 Clock model structure

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

The following are the clock model elements.
• **Clock source** – Produces a clock signal of a specified frequency. If it is an external clock source, it can have one or more related pins.

![Clock source diagram](image)

Figure 79. Clock source

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

![Clocks selector diagram](image)

Figure 80. Clocks selector

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

![Prescaler diagram](image)

Figure 81. Prescaler

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

![Frequency Locked Loop diagram](image)

Figure 82. Frequency Locked Loop

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

![Phase Locked Loop diagram](image)

Figure 83. Phase Locked Loop

• **Clock gate** – Stops the propagation of incoming signal.

![Clock gate diagram](image)

Figure 84. Clock gate
• **Clock output** – Marks the clock signal output that has some name and can be further used by the peripherals or other parts of the processor. You can put a lock and/or frequency request.

![Clock output](image1)

**Figure 85. 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.

![Clock component](image2)

**Figure 86. 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>
<thead>
<tr>
<th>Menu item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional groups</td>
<td>Open the Functional group properties dialog.</td>
</tr>
<tr>
<td>Refresh</td>
<td>Refresh each clocks configuration with explicit invocation of code generation.</td>
</tr>
<tr>
<td>Reset To Board Defaults</td>
<td>Reset the clock model to board defaults.</td>
</tr>
<tr>
<td>Reset To Processor Defaults</td>
<td>Resets the clock model to processor defaults.</td>
</tr>
<tr>
<td>Unlock All Settings</td>
<td>Unlocks all locks in all settings.</td>
</tr>
<tr>
<td>Unlock Settings on the Active Path</td>
<td>Unlocks all locks in the settings that are on the active path.</td>
</tr>
</tbody>
</table>

### 4.12 Troubleshooting problems

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

You may encounter any of the following problems:

1. **Requirements not satisfiable**: Indicates that there are one or more locked frequency or frequency constraints for which the tool is not able to find a valid setting and satisfy those requirements.

2. **Invalid settings or requirements**: [element list] – Indicates that the value of a setting is not valid. For example, the current state of settings is beyond the acceptable range.

The following are some tips to troubleshoot encountered problems:

1. Start with only one locked clock output frequency and let the tool find and calculate other ones. After you are successful, you can add more.
2. Go through the locked outputs (if there are any) and verify the requirements (there can possibly be typos in the required frequency, wrong units, and so on).

3. If you seek only to enable some clock output, try to use pop-up the menu command **Enable** that tries to automatically find settings providing any valid frequency on clock output.

4. If the required clock output value cannot be satisfied try to use the pop-up menu command "Advanced resolver for (clock output)."

5. If you are OK to have a near frequency value around of the requested value but would like to keep the clock selectors and clock sources unchanged, right-click and from the pop-up menu select **Clock output > Find near value**.

6. If the problems still persist, find the elements and settings with marked errors in the diagram or tables and see the details in the tooltip.

7. If you cannot reach the values you need, use the diagram view to see the elements on clock path leading to the clock output you want to have set. Try to check and adjust the settings of these elements manually in the Details view.

8. Try to remove locks by selecting **Clocks > Unlock All Settings.** In case too many changes are required and conflicting, you can simply reset the model to the default values and start from the beginning. To reset, select **Clocks > Reset to processor defaults**.

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

- **Show problem** - Shows the problem in the **Clocks Diagram** view.

  If one 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.
  - **Enable** - If the clock output is disabled, tries to find settings that provide valid frequency on the clock output.
  - **Advanced resolver** - Invoked advanced resolver that tries to find suitable settings to achieve the required frequency. For more information, see the Advance resolver in the **pop-up menu commands**.

Find Near Value Dialog

![Find Near Valid Value](image)

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

### 4.13 Code generation

If the settings are correct and no error is reported, the tool’s code generation engine instantly regenerates the source code. The resulting code is found in the **Code Preview** view.
Code Preview automatically highlights differences between the current and immediately preceding iteration of the code. You can choose between two modes of highlighting by clicking the Set viewing style for source differences. You can also disable highlighting altogether from the same dropdown menu.

4.13.1 Working with the code

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

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

• Click Update Code in the toolbar.
• Copy the content using the COPY command, either by pressing the CTRL+C keys or the pop-up menu after the whole text is selected.
• Use export command.
• Click the Export button in Code Preview view.
If you need access to values of registers calculated by the tool, the defines with these values can be generated into new file registers.h. It can be enabled by default (Edit->Configuration Preferences). For more information, see section Configuration Preferences.

4.14 Clock Consumers view

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

**NOTE**

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

![Figure 88. Clock Consumers view](image-url)

---

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

4.14 Clock Consumers view

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

**NOTE**

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

![Figure 88. Clock Consumers view](image-url)
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, FREEMaster, LwIP)
- The settings can be automatically migrated to a different SDK component version
- Support of code snippets

5.2 Basic Terms and Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional group</td>
<td>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.</td>
</tr>
<tr>
<td>Peripheral instance</td>
<td>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.</td>
</tr>
<tr>
<td>Configuration component</td>
<td>Provides user interface for configuring SDK software component (for example, peripheral driver) and generates code for its initialization.</td>
</tr>
<tr>
<td>Component instance</td>
<td>Configuration component can have multiple instances with different settings. (for example, for each peripheral instance like UART0, UART1).</td>
</tr>
<tr>
<td>Component mode</td>
<td>Specific use case of the component instance (for example, TRANSFER mode of DSPI, or interrupt-based mode of communication).</td>
</tr>
</tbody>
</table>

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

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

3. Select the component that you want to use and click OK.

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

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

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

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

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

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

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 overview

![Peripheral view](image)

**Figure 89. Peripheral view**
5.4.1 Common toolbar (Peripherals)

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global settings</td>
<td>Open a tab aggregating global settings of all configuration sets.</td>
</tr>
<tr>
<td>Initialization order</td>
<td>Open a dialog for customization of peripheral initialization order.</td>
</tr>
</tbody>
</table>

**NOTE**

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

5.4.1.1 Initialization order dialog

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

5.4.2 Components view

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

The view highlights configuration components based on their status.

<table>
<thead>
<tr>
<th>Status</th>
<th>Color highlighting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Light gray.</td>
</tr>
<tr>
<td>Enabled/with warning</td>
<td>Light gray with the alert symbol.</td>
</tr>
<tr>
<td>Enabled/with error</td>
<td>Red with the error symbol.</td>
</tr>
<tr>
<td>Disabled</td>
<td>Dark gray.</td>
</tr>
</tbody>
</table>
In the **Components** view, you can perform several actions.

**Table 20. Components view actions**

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

*Table continues on the next page...*
Table 20. Components view actions (continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter configuration components by name</td>
<td>Type a text string to filter configuration component names in the search bar.</td>
</tr>
</tbody>
</table>

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

Table 21. Shortcut menu options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Open the configuration component in the Settings Editor.</td>
</tr>
<tr>
<td>Open in another view</td>
<td>Duplicates the configuration component in the Settings Editor.</td>
</tr>
<tr>
<td>Enable/Disable</td>
<td>Enable/Disable the configuration component.</td>
</tr>
<tr>
<td>Edit comment</td>
<td>Create/Edit custom notes for the configuration component.</td>
</tr>
<tr>
<td>Lock/Unlock editing of component instance</td>
<td>Lock/Unlock the editing of the component instance.</td>
</tr>
<tr>
<td>Save to use case library</td>
<td>Create a template from the configuration component.</td>
</tr>
<tr>
<td>Documentation</td>
<td>Display the documentation of the configuration component, if available.</td>
</tr>
<tr>
<td>Remove</td>
<td>Remove the component from configuration.</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Migrate</th>
<th>Migrate the component to a different component type or to a component with a newer driver version.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move to</td>
<td>Choose from available functional groups to move the configuration component to.</td>
</tr>
<tr>
<td>Copy to</td>
<td>Choose from available functional groups to copy the configuration component to.</td>
</tr>
</tbody>
</table>
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 unchecking the checkbox.

Double-click 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>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Open the component instance in the <strong>Settings Editor</strong>.</td>
</tr>
<tr>
<td>Open in another view</td>
<td>Duplicate the component instance in the <strong>Settings Editor</strong>.</td>
</tr>
<tr>
<td>Enable/Disable</td>
<td>Enable/Disable the component instance.</td>
</tr>
<tr>
<td>Add component instance</td>
<td>Add a component instance to the peripheral.</td>
</tr>
<tr>
<td>Initialized in user code</td>
<td>Mark the peripheral as configured by user code (available on not configured peripherals).</td>
</tr>
<tr>
<td>Edit comment</td>
<td>Create/Edit custom notes for the component instance.</td>
</tr>
<tr>
<td>Lock/Unlock editing of component instance</td>
<td>Lock/Unlock the editing of the component instance.</td>
</tr>
<tr>
<td>Save to use case library</td>
<td>Create a template from the component instance.</td>
</tr>
<tr>
<td>Documentation</td>
<td>Display the documentation of the component instance, if available.</td>
</tr>
<tr>
<td>Remove</td>
<td>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.</td>
</tr>
<tr>
<td>Migrate</td>
<td>Migrate the component to a different component type or to a component with a newer driver version.</td>
</tr>
<tr>
<td>Move to</td>
<td>Choose from available functional groups to move the component instance to.</td>
</tr>
<tr>
<td>Copy to</td>
<td>Choose from available functional groups to copy the component instance to.</td>
</tr>
</tbody>
</table>

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.

![Quick selection example](image)

**Figure 93.** 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).
  ![Enable Rx/Tx interrupt](image)
  
  **Figure 94.** Boolean setting example

- **Integer, Float** – Integer or float number.
  ![Priority](image)
  
  **Figure 95.** Integer/Float setting example

- **String** – Textual input. More than a single line can be supported.
  ![Handler name](image)
  
  **Figure 96.** String setting example

- **Enumeration** – Selection of one item from list of values.
• **Set** – List of values, multiple of them can be selected.

![Figure 97. Enumeration setting example](image)

- RX active edge interrupt.
- Transmission complete interrupt
- Idle line interrupt.
- Noise error flag interrupt.
- Parity error flag interrupt.
- TX FIFO overflow interrupt.

![Figure 98. Set setting example](image)

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

![Figure 99. Structure setting example](image)

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

![Figure 100. Array](image)

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

**NOTE**

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

<table>
<thead>
<tr>
<th>Header item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Displays the configuration component title.</td>
</tr>
<tr>
<td>Name</td>
<td>Displays the component instance name. This name is used in the generated code</td>
</tr>
<tr>
<td></td>
<td>in constants and function identifiers and is derived from the peripheral name.</td>
</tr>
<tr>
<td></td>
<td>You can change it at any time by clicking the Custom name button and editing</td>
</tr>
<tr>
<td></td>
<td>the field.</td>
</tr>
<tr>
<td>Mode</td>
<td>Displays the required usage for the component instance and influences</td>
</tr>
<tr>
<td></td>
<td>available settings. Use the dropdown menu to change the mode (where applicable).</td>
</tr>
<tr>
<td>Peripheral</td>
<td>Displays the name of the peripheral to be associated with the component instance. Use the dropdown menu to change it.</td>
</tr>
<tr>
<td>Documentation</td>
<td>Click the button to view configuration component-specific documentation in the Documentation view. Not all configuration components are documented, therefore not all setting headers contain the Documentation icon.</td>
</tr>
<tr>
<td>Lock editing</td>
<td>Click the button to lock/unlock component editing. Source code will still be generated.</td>
</tr>
<tr>
<td>Enable/disable component instance switch</td>
<td>Use the switch to enable or disable selected component instance. By disabling the instance, you don’t remove it from the tools configuration, but prevent its inclusion in the generated code.</td>
</tr>
</tbody>
</table>
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.

Universal Serial Bus (USB)

Universal Serial Bus

USB is a fast bi-directional isochronous low-cost dynamically attachable serial interface that is an industry standard. It provides a ubiquitous link that can be used across a wide range of peripheral-to-PC interconnects. USB specification is maintained by the USB Implementers Forum (USB-IF).

Prerequisites

Your copy of SDK must contain "USB stack" middleware for you to be able to build your USB project.

Limitations

Adding this component to the project is only supported in MCUxpresso IDE.

Configuration

A general process of creating a new USB device or USB host follows these steps:

• select the Preset closest to the desired Device role/Host role
• for each Supported interface, select the appropriate Preset and Protocol

You can open the Documentation view in several ways:

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

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

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

To create a component use case, do the following:

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

5.4.7 Component Migration to a different version

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

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

![Component selection dialog](image)

Figure 105. Component selection

It is also visible in the tooltip in the Component view:
You can update the SDK components in the toolchain/IDE project.

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

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

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

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

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

After you confirm the dialog by selecting "Migrate", the component is replaced by the component matching the selected version of the SDK component. The settings are migrated to the corresponding settings in the new version of the component, where it is possible.
If the new version of the component contains some new settings, these settings are filled with the default values. Check manually if the components are set properly.

5.5 Problems

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

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

![Error decorators](image1)

Figure 108. Error decorators

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

![Quick fix 1](image2)

Figure 109. Quick fix 1

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

![Quick fix 2](image3)

Figure 110. Quick fix 2

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

![Quick fix 3](image4)

Figure 111. Quick fix 3
5.6 Code generation

If the settings are correct and no error is reported, the tool’s code generation engine instantly regenerates the source code. You can view the resulting code in the Code Preview view of the Peripherals tool.

Code Preview automatically highlights differences between the current and immediately preceding iteration of the code. You can choose between two 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. It can be configured in functional group properties.

NOTE

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

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

- Constants and 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.
Code generation

Figure 112. 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.

![Device Configuration tool](image)

**Figure 113. Device Configuration tool**

6.1 Device Configuration Data (DCD) view

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

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

6.1.1 Device Configuration Data (DCD) view actions

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

- **Create a new command group** - Right-click the table and choose Add Group from the context menu.
- **Re/Name a command group** - Left-click the command group cell and enter the required name.
- **Disable a command group** - Right-click the command group row and choose Disable Group from the context menu.
- **Remove a command group** - Right-click the command group row and choose Remove Group from the context menu.
- **Collapse all command groups** - Right-click the table and choose Collapse All Groups from the context menu.
- **Expand all command groups** - Right-click the table and choose Expand All Groups from the context menu.
- **Add a command to a group** - Right-click the table and choose Add Command from the context menu. Alternatively, click the Add Command button in the tool's toolbar.
• Specify command type - Left-click the row’s Command cell and choose from the dropdown menu.

• Specify register address for a command - Left-click the row’s Address cell and choose from the dropdown menu.

• Specify a value or a mask for a command - Left-click the row’s Value(s) / Mask(s) cell to open the mask window. Enter the value into the field and select OK. Alternatively, select Cancel to cancel the operation, or Reset to reset the value.

![Mask Window](image)

Figure 114. Mask

• Specify the size of write/read data for a command - Left-click the row’s Size cell and choose from the dropdown menu.

• Specify the number of polls of a command - Left-click the row’s Poll cell and enter the required value.

• Add a comment to a command - Left-click the row’s Comment cell.

• Remove a command - Right-click the command row and choose Remove Command from the context menu. Alternatively, click the Remove Command button in the tool’s toolbar.

• Cut a command - Right-click the command row and choose Cut from the context menu.

• Copy a command - Right-click the command row and choose Copy from the context menu.

• Paste a command - Right-click the command row and choose Paste from the context menu.

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

Basic cell selection shortcuts are applicable.

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

6.2 Code generation

If the settings are correct and no error is reported, the code generation engine instantly regenerates the source code. You can view the resulting code 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.
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 subviews, and review these policies in the Memory Attribution Map, Access Overview and Domains Overview views. Use the User Memory Regions view to create a convenient overview of memory regions and their security levels.

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

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

Figure 116. TEE tool user interface (SAU+IDAU)
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 security extensions AHBSC and TZ+TRDC. Currently, following devices of this type are supported:

- LPC55Sxx
  - LPC55S69, LPC55S66
  - LPC55S16, LPC55S14
  - LPC55S06, LPC55S04
- KW45
- 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.

![User Memory Regions](image)

**Figure 118. User Memory Regions**

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

---

**NOTE**

After the import, you might have to correct some of the security levels manually.
7.1.2 Security Access Configuration view

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

7.1.2.1 SAU

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

When enabled, you can set up SAU memory regions, specify their start and size or end address, and specify their security level. SAU automatically sets the entire memory space to a Secure security 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.
In the Interrupts subview, you can set security designation for device’s peripheral interrupts. In case if the processor contains more than a single core or processing unit, additional Handling by Core tables might appear. In these tables, you can specify if the interrupts coming from the peripheral can be handled by the core or processing unit.

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

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

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

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

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

### 7.1.2.4 MPC

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

Set the memory sector security level by left-clicking the relevant cell in the **Security level** column and choosing from the dropdown list. Alternatively, you can right-click the relevant cell in the **Sector** column and choose the security level from the context menu.

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

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

- **NS-User** - Non-secure user
- **NS-Priv** - Non-secure privileged
- **S-User** - Secure user
- **S-Priv** - Secure privileged
In the Masters/Slaves subview, you can configure security levels for bus masters and slaves.

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

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

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

You can further specify the interrelation between master and slave security levels by selecting the following options:
• **Simple Master in Strict Mode** - Select to allow simple bus master to read and write on same level only. De-select to allow to read and write on same and lower level.

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

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

![User Memory Regions](image)

<table>
<thead>
<tr>
<th>Options</th>
<th>Security Access Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>[✓] Simple Master in Strict Mode</td>
<td>[✓] Smart Master in Strict Mode</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Master</th>
<th>Security Level</th>
<th>Slave</th>
<th>Security Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANFD</td>
<td>NS-User</td>
<td>ADC0</td>
<td>NS-User</td>
</tr>
<tr>
<td>DMA0</td>
<td>NS-User</td>
<td>AHB_SECURE_CTRL</td>
<td>NS-User</td>
</tr>
<tr>
<td>DMA1</td>
<td>NS-User</td>
<td>ANACTRL</td>
<td>NS-User</td>
</tr>
<tr>
<td>HASHCRYPT</td>
<td>NS-User</td>
<td>CAN0</td>
<td>NS-User</td>
</tr>
<tr>
<td>USBFSI</td>
<td>NS-User</td>
<td>CASPER</td>
<td>NS-User</td>
</tr>
<tr>
<td>USBFSH</td>
<td>NS-User</td>
<td>CRC_ENGINE</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CTIMER0</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CTIMER1</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CTIMER2</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CTIMER3</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CTIMER4</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBGMAILBOX</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DMA0</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DMA1</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLASH</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLEXCONN0</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLEXCONN1</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLEXCONN2</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLEXCONN3</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLEXCONN4</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLEXCONN5</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLEXCONN6</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLEXCONN7</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GINT0</td>
<td>NS-User</td>
</tr>
</tbody>
</table>

Figure 124. Masters/Slaves

### 7.1.2.6 Pins

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

All pins' reading GPIO state is set to **Allow** by default. If you want to change the pins reading GPIO state, left-click the **Reading GPIO state** cell of the pin and choose from the dropdown menu. Alternatively, right-click the pin's **Name** cell and choose the reading GPIO state from the context menu. To select multiple entries, use the **Ctrl+Left-click** shortcut, then right-click the selected area for the context menu. Alternatively, you can use **Shift+Up/Down** after selecting the row to expand the selection.
### General purpose input/output port 0

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-54</td>
<td>PIO0_0/FC3_SC...GPIO/SECURE_GPIO0/0/ACMP0_Allow</td>
</tr>
<tr>
<td>1-7</td>
<td>PIO0_1/FC3_CTS...GPIO/CMPO_OUT/SECURE_GPIO_Allow</td>
</tr>
<tr>
<td>2-81</td>
<td>PIO0_2/FC3_TXD...UT0/SCT_GPIO/SECURE_GPIO0_Allow</td>
</tr>
<tr>
<td>3-83</td>
<td>PIO0_3/FC3_RXD...UT1/SCT_GPIO/SECURE_GPIO0_Allow</td>
</tr>
<tr>
<td>4-86</td>
<td>PIO0_4/CAN0_R...TS_SDA_SSEL0/SECURE_GPIO0_Allow</td>
</tr>
<tr>
<td>5-88</td>
<td>PIO0_5/CAN0_TD...L_SSEL1/MCLK/SECURE_GPIO0_Allow</td>
</tr>
<tr>
<td>6-89</td>
<td>PIO0_6/FC3_SC...A10/SCT_GPIO/SECURE_GPIO0_Allow</td>
</tr>
<tr>
<td>7-6</td>
<td>PIO0_7/FC3_RTS...SCK/FC1_SCK/SECURE_GPIO0_Allow</td>
</tr>
<tr>
<td>8-26</td>
<td>PIO0_8/FC3_SS...SI_DATA/SWO/SECURE_GPIO0_Allow</td>
</tr>
<tr>
<td>9-55</td>
<td>PIO0_9/FC3_SS...WS/SECURE_GPIO0/9/ACMP0_Allow</td>
</tr>
<tr>
<td>10-21</td>
<td>PIO1_10/FC6_S.../SWO/SECURE_GPIO10/10/ADC_Allow</td>
</tr>
<tr>
<td>11-13</td>
<td>PIO1_11/FC6_RX...SWCLK/SECURE_GPIO11/11/A_Allow</td>
</tr>
<tr>
<td>12-12</td>
<td>PIO1_12/FC3_T...WS/SECURE_GPIO12/12/ADC0_Allow</td>
</tr>
<tr>
<td>13-71</td>
<td>PIO1_13/FC1_CT...DATA/PLU_IN0/SECURE_GPIO1_Allow</td>
</tr>
<tr>
<td>14-72</td>
<td>PIO1_14/FC1_RT...O_WS/PLU_IN1/SECURE_GPIO1_Allow</td>
</tr>
<tr>
<td>15-22</td>
<td>PIO1_15/FC6_C...OUTZ/SECURE_GPIO15/15/ADC_Allow</td>
</tr>
<tr>
<td>16-14</td>
<td>PIO1_16/FC4_TX...INP4/SECURE_GPIO16/16/AD_Allow</td>
</tr>
<tr>
<td>17-8</td>
<td>PIO1_17/FC4_SSE...OUTI/PLU_IN2/SECURE_GPIO1_Allow</td>
</tr>
<tr>
<td>18-56</td>
<td>PIO1_18/FC4_C...IN3/SECURE_GPIO18/18/ACMP_Allow</td>
</tr>
<tr>
<td>19-90</td>
<td>PIO1_19/FC4_R...O_WS/PLU_IN4/SECURE_GPIO0_Allow</td>
</tr>
<tr>
<td>20-74</td>
<td>PIO1_20/FC3...PIO0_20/FC4_TXD_SCL_MISO_VI_Allow</td>
</tr>
<tr>
<td>21-76</td>
<td>PIO1_21/FC3_RT...L3/PLU_CLKIN/SECURE_GPIO1_Allow</td>
</tr>
<tr>
<td>22-78</td>
<td>PIO1_22/FC6...US/PLU_OUT7/SECURE_GPIO0_Allow</td>
</tr>
<tr>
<td>23-20</td>
<td>PIO1_23/MCLK...SEL0/SECURE_GPIO0/23/ADC0_Allow</td>
</tr>
<tr>
<td>24-70</td>
<td>PIO1_24/FC0_R...P8/SCT_GPIO/SECURE_GPIO0_Allow</td>
</tr>
<tr>
<td>25-79</td>
<td>PIO1_25/FC0_T...N9P/SCT_GPIO/SECURE_GPIO0_Allow</td>
</tr>
<tr>
<td>26-60</td>
<td>PIO1_26/FC2_R...HS_SPI_MOSI/SECURE_GPIO0_Allow</td>
</tr>
</tbody>
</table>

Figure 125. Pins
Figure 126. Pins tab on KW45

Figure 127. Global Access Templates
7.1.2.7 Miscellaneous

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

Figure 129. Miscellaneous (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 subview, you can review security levels set for Core 0 to the code, data, and peripherals memory regions. The table is read-only.

The Access by Master table displays MSW or SAU+IDAU, MPC (Memory Protection Checker) security level, and Resulting 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 that 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.
### 7.1.3.2 Other masters

In the **Other Masters** subview, 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:

#### Table: Memory Attribution Map

<table>
<thead>
<tr>
<th>Memory Alias</th>
<th>SAU/IDAU</th>
<th>MPC/PPC</th>
<th>Resulting Security Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB SRAM (alias)</td>
<td>S</td>
<td>NS-User</td>
<td>S</td>
</tr>
<tr>
<td>RAM 1 (alias)</td>
<td>S</td>
<td>NS-User</td>
<td>S</td>
</tr>
<tr>
<td>RAM 0 (alias)</td>
<td>S</td>
<td>NS-User</td>
<td>S</td>
</tr>
<tr>
<td>USB SRAM</td>
<td>NS</td>
<td>NS-User</td>
<td>NS</td>
</tr>
<tr>
<td>RAM 1</td>
<td>NS</td>
<td>NS-User</td>
<td>NS</td>
</tr>
<tr>
<td>RAM 0</td>
<td>NS</td>
<td>NS-User</td>
<td>NS</td>
</tr>
<tr>
<td>SRAM X (alias)</td>
<td>S</td>
<td>NS-User</td>
<td>S</td>
</tr>
<tr>
<td>Boot-ROM (alias)</td>
<td>S</td>
<td>NS-User</td>
<td>S</td>
</tr>
<tr>
<td>FLASH (alias)</td>
<td>S</td>
<td>NS-User</td>
<td>S</td>
</tr>
<tr>
<td>SRAM X</td>
<td>NS</td>
<td>NS-User</td>
<td>NS</td>
</tr>
<tr>
<td>Boot-ROM</td>
<td>NS</td>
<td>NS-User</td>
<td>NS</td>
</tr>
<tr>
<td>PROGRAM FLASH</td>
<td>NS</td>
<td>NS-User</td>
<td>NS</td>
</tr>
</tbody>
</table>
1. Click the Filter access for checkbox to enable filtering options.
2. Select the master type security access that you want to review by choosing from the Master dropdown menu.
3. Optionally, customize the output by de-selecting the Show Details, Show Code, Show Data, 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.
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 right-hand side of the toolbar.

![Memory Attribution Map](image)

**Figure 132. Access Overview**

### 7.1.5 Code generation

If the settings are correct and no error is reported, the code generation engine instantly regenerates the source code. You can view the resulting code the **Code Preview** view of the 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 subview.

7.2 RDC-enabled devices

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

This section describes the features and appearance of the tool devices 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.

![User Memory Regions](image)

Figure 133. 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.
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** subview, you can assign masters to domains and specify access rules for slaves for each domain.

**7.2.2.1.1 RDC Masters**

In the **RDC Masters** subview, you can view available bus masters, allocate them to available domains (cores), and lock/unlock the allocation.
Allocate a master to a domain by clicking the cell in the Domain column in the Masters table and selecting the domain from the dropdown list.

Select the Lock checkbox to prevent further register modifications.

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

**NOTE**

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

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

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

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.

![Memory Regions Configuration Table](image)

---

**Figure 136. Memory Regions**

---

MCUXpresso Config Tools User's Guide v.12 (IDE), Rev. 3, 30 June 2022

NXP Semiconductors
5. Set the **Access Template** for available domains. Alternatively, you can select the options by right-clicking the master and using the dropdown list.

### 7.2.2.1.3 Peripherals

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

![Peripherals Configuration](image)

**Figure 137. Peripherals**

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

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

**NOTE**

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

3. Set the **Access Template** for available domains.
7.2.2.2 XRDC2 Domains view

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

7.2.2.2.1 MPU

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

The MPU enforces privilege rules, separates processes, 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.
Use the MPU Memory Attributes table to name and configure MPU memory attribute sets. Click the cells of the Memory Type and Inner/Outer Attributes columns to display the available options.

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

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

### 7.2.2.2.2 Domains

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

![Domains subview](image)

**Figure 139. Domains**

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

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

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

### 7.2.2.2.3 Masters

In the **Masters** subview, you can 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.
To add a new domain assignment:

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

**NOTE**

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

**NOTE**

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

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

**7.2.2.2.4 Peripherals**

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

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

The **Access Template** table displays the ID and name of all access templates available for the PAC on the selected device. The information is data driven and display-only.
Use the Peripherals Configuration table to configure access for a peripheral:

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

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

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

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

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

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

Figure 142. Memory Regions

![Memory Regions Configuration Table]

<table>
<thead>
<tr>
<th>Index</th>
<th>Enable</th>
<th>Address</th>
<th>Size</th>
<th>End Address</th>
<th>Lock</th>
<th>Access Template</th>
<th>Domain 0</th>
<th>Domain 1</th>
<th>Domain 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0x0028_0000</td>
<td>0x0000_0100</td>
<td>0x0000_0FFF</td>
<td>Unlocked</td>
<td>No access</td>
<td>No access</td>
<td>No access</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>0x2020_0000</td>
<td>0x2000_0200</td>
<td>0x0001_0FFF</td>
<td>Unlocked</td>
<td>R for S</td>
<td>No access</td>
<td>No access</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0x0002_0000</td>
<td>0x0000_0200</td>
<td>0x0003_0FFF</td>
<td>Unlocked</td>
<td>R for S</td>
<td>R for S</td>
<td>R for S</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>0x2034_0000</td>
<td>0x2030_0100</td>
<td>0x0000_0FFF</td>
<td>Unlocked</td>
<td>No access</td>
<td>No access</td>
<td>No access</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>0x2035_0000</td>
<td>0x2034_0100</td>
<td>0x0000_0FFF</td>
<td>Unlocked</td>
<td>No access</td>
<td>No access</td>
<td>No access</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>0x2036_0000</td>
<td>0x2035_0100</td>
<td>0x0000_0FFF</td>
<td>Unlocked</td>
<td>No access</td>
<td>No access</td>
<td>No access</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>0x0000_0000</td>
<td>0x0200_0000</td>
<td>0x0000_1FFF</td>
<td>Unlocked</td>
<td>No access</td>
<td>No access</td>
<td>No access</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>0x3000_0000</td>
<td>0x3000_0100</td>
<td>0x00FF_0FFF</td>
<td>Unlocked</td>
<td>RW for S, R for NS</td>
<td>All</td>
<td>No access</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>0x6000_0000</td>
<td>0x7F00_0200</td>
<td>0x01FF_0FFF</td>
<td>Unlocked</td>
<td>No access</td>
<td>No access</td>
<td>No access</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>0x8000_0000</td>
<td>0x0DFF_0FFF</td>
<td>0x03FF_0FFF</td>
<td>Unlocked</td>
<td>All</td>
<td>All</td>
<td>No access</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0x0400_0000</td>
<td>0x2000_0000</td>
<td>0x05FF_0FFF</td>
<td>Unlocked</td>
<td>No access</td>
<td>No access</td>
<td>No access</td>
<td></td>
</tr>
</tbody>
</table>
1. Select the **Enable** checkbox.
2. Specify the **Start Address**.
3. Specify either **Size** or **End Address**.
4. Set the **Lock** to the desired state.
5. Set the **Access Template** for all listed domains.

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

### 7.2.2.2.6 Memory Slots

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

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

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

![Memory Slots](image)

**Figure 143. Memory Slots**
Use the **Memory Slots Configuration** table to configure access for a memory space:

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

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

### 7.2.2.3 Miscellaneous

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

![User Memory Regions](image)

**Figure 144. Miscellaneous (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.
To set the display options, do the following:

1. Click the **Filter access for** checkbox to enable filtering options.
2. Select the master that you want to review by choosing from the Master dropdown menu.

3. Optionally, set the security 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.
Figure 146. Access Overview

Point your cursor at an entry to display a tooltip with information about the entry.
You can group the displayed information by security or by masters by using the button on the right-hand side of the toolbar.

### 7.2.5 Domains Overview

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

Point your cursor over the cells to display a tooltip with information about the security level combination.
If the settings are correct and no error is reported, the code generation engine instantly regenerates the source code. You can view the resulting code the Code Preview view of the 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** subview.
8.1 Exporting the Pins table

To export the Pins table, do the following:

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

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

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

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

8.3 Generating HTML reports

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

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

8.4 Exporting sources

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.

3. Click Next.
4. Select the target folder where you want to store the generated files.
5. In case of multicore processors, select the cores you want to export.
6. Click Finish.

8.5 Exporting registers

You can export the content of tool-modified 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.

8.6 Command line execution

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

On error application exits:

- Tools v4.1 and older:
  - with ‘123321’ error code. The reason should be logged.
- Tools v5.0 and newer:
  - 1 when parameter is missing
  - 2 when tool error occurs

You can chain commands in CLI.

Notes regarding command-line execution:

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

```
-HeadlessTool Clocks -MCU MK64FX512xxx12 -SDKVersion ksdk2_0 -ExportSrc C:/exports/src -HeadlessTool
Pins -MCU MK64FX512xxx12 -SDKVersion ksdk2_0 -ExportSrc C:/exports/src -HeadlessTool Peripherals -MCU
MK64FX512xxx12 -SDKVersion kdk2_0 -ExportSrc C:/exports/src
```

**NOTE**

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

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

Recommended usage:

• For manual one time usage, include the -updateData parameter on the CLI.
• For multiple executions, for example, continuous integration set-up you job:
  — Use the command with -updateData, which updates possibly outdated data.
  — Use all other commands in the package without this parameter:

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

The following commands are supported in the framework:
<table>
<thead>
<tr>
<th>Command name</th>
<th>Definition and parameters</th>
<th>Description</th>
<th>Restriction</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version of the product</td>
<td>-version</td>
<td>Shows the build version of the product into the stdout and continues with parsing other parameters. (since 6.0)</td>
<td>-version</td>
<td>-version</td>
</tr>
<tr>
<td>Force language</td>
<td>-nl {lang}</td>
<td>Forces set language {lang} is in ISO-639-1 standard</td>
<td>Removal of the '.nxp' folder from home directory is recommended, as some text might be cached Only 'zh' and 'en' are supported</td>
<td>-nl zh</td>
</tr>
<tr>
<td>Show console</td>
<td>-consoleLog</td>
<td>Logs output is also sent to Java's System.out (typically back to the command shell if any)</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Select MCU</td>
<td>-MCU</td>
<td>MCU to be selected by framework Changes the processor in the result configuration of the previous chain</td>
<td>Requires -SDKversion command</td>
<td>-MCU MK64FX512xxx12</td>
</tr>
<tr>
<td>Select Board</td>
<td>-Kit</td>
<td>Board to be selected by framework (MCU is automatically selected too) (since 6.0)</td>
<td>Requires -SDKversion command</td>
<td>-Board FRDM-K22F - SDKversion ksdk2_0</td>
</tr>
<tr>
<td>Select Kit</td>
<td>-SDKversion</td>
<td>Kit to be selected by framework (MCU and board is automatically selected too)(since 6.0)</td>
<td>Requires -SDKversion command</td>
<td>-Kit FRDM-K22F- AGM01 -SDKversion ksdk2_0</td>
</tr>
<tr>
<td>Select SDK version</td>
<td>-SDKversion</td>
<td>Version of the MCU to be selected by framework</td>
<td>Requires -MCU command</td>
<td>-SDKversion test_ksdk2_0</td>
</tr>
<tr>
<td>Select part number</td>
<td>-PartNum</td>
<td>Selects specific package of the MCU</td>
<td>Requires -MCU and -SDKversion commands</td>
<td>-PartNum MK64FX512VLL12</td>
</tr>
<tr>
<td>Configuration name</td>
<td>-ConfigName</td>
<td>Name of newly created configuration - used in export Name is used when new configuration is created by -MCU and -SDKversion commands</td>
<td>-ConfigName &quot;MyConfig&quot;</td>
<td></td>
</tr>
<tr>
<td>Select tool</td>
<td>-HeadlessTool</td>
<td>Selects a tool that should be run in headless mode</td>
<td>None</td>
<td>-HeadlessTool Clocks</td>
</tr>
<tr>
<td>Load configuration</td>
<td>-Load</td>
<td>Loads existing configuration from (*.mex) file</td>
<td>None</td>
<td>-Load C:/conf/conf.mex</td>
</tr>
</tbody>
</table>

Table continues on the next page...
<table>
<thead>
<tr>
<th>Command name</th>
<th>Definition and parameters</th>
<th>Description</th>
<th>Restriction</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export Mex</td>
<td>-ExportMEX</td>
<td>Exports .mex configuration file after tools run</td>
<td>None</td>
<td>-MCU xxx - SDKversion xxx - ExportMEX C:/exports/my_config_folder</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Argument is expected as a folder name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Export all generated files</td>
<td>-ExportAll</td>
<td>Exports generated files (with source code and so on. Code is regenerated before export)</td>
<td>Requires -HeadlessTool command</td>
<td>-HeadlessTool Pins -ExportAll C:/exports/generated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Includes -ExportSrc and in framework -ExportMEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Argument is expected as a folder name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generate source files</td>
<td>-CustomCopyright</td>
<td>File content is inserted as a copyright file header comment into generated source files (.c, .h, .dts, .dti), that doesn't contain copyright</td>
<td>Requires -HeadlessTool command</td>
<td>-CustomCopyright c:\test\copyright.txt</td>
</tr>
<tr>
<td>with custom copyright</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Override the output path</td>
<td>-OutputPathOverrides</td>
<td>Path to the file with rules, that will be used to override output paths of the generated file. Empty list of rules removes the setted rules.</td>
<td></td>
<td>-OutputPathOverriderules c:\test\outputPathOverrideRules.yaml</td>
</tr>
<tr>
<td>of the generated files</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update locally downloaded data</td>
<td>-updateData</td>
<td>Downloads data for already locally downloaded data if they have an update.</td>
<td></td>
<td>-updateData</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8.6.1 Command Line execution - Pins Tool

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

Table 25. Commands supported in Pins

<table>
<thead>
<tr>
<th>Command name</th>
<th>Definition and parameters</th>
<th>Description</th>
<th>Restriction</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable tool</td>
<td>-Enable</td>
<td>Enables the tool if it is disabled in the current configuration</td>
<td>Requires -HeadlessTool Pins</td>
<td>-HeadlessTool Pins -Enable</td>
</tr>
<tr>
<td>Import C files</td>
<td>-ImportC</td>
<td>Imports .c files into configuration</td>
<td>Requires -HeadlessTool Pins</td>
<td>-HeadlessTool Pins -ImportC C:/imports/file1.c C:/imports/file2.c</td>
</tr>
</tbody>
</table>

Table continues on the next page...
### Table 25. Commands supported in Pins (continued)

<table>
<thead>
<tr>
<th>Command name</th>
<th>Definition and parameters</th>
<th>Description</th>
<th>Restriction</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Importing is done after loading mex and before generating outputs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Export all generated files (to simplify all exports commands to one command)</td>
<td>-ExportAll</td>
<td>Exports 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</td>
<td>Requires -HeadlessTool Pins -ExportAll C:/exports/generated</td>
<td></td>
</tr>
<tr>
<td>Export Source files</td>
<td>-ExportSrc</td>
<td>Exports generated source files. The code will be regenerated before export Argument is expected as a folder name</td>
<td>Requires -HeadlessTool Pins -ExportSrc C:/exports/src</td>
<td></td>
</tr>
<tr>
<td>Export CSV file</td>
<td>-ExportCSV</td>
<td>Exports generated csv file. The code will be regenerated before export Argument is expected as a folder name</td>
<td>Requires -HeadlessTool Pins -ExportCSV C:/exports/src</td>
<td></td>
</tr>
<tr>
<td>Export HTML report file</td>
<td>-ExportHTML</td>
<td>Exports generated html report file. The code will be regenerated before export Argument is expected as a folder name</td>
<td>Requires -HeadlessTool Pins -ExportHTML C:/exports/html</td>
<td></td>
</tr>
<tr>
<td>Export registers</td>
<td>-ExportRegisters</td>
<td>Exports registers tab into folder. The code will be regenerated before export Argument is expected as a folder name</td>
<td>Requires -HeadlessTool Pins -ExportRegisters C:/exports/regs</td>
<td></td>
</tr>
</tbody>
</table>

### 8.6.2 Command Line execution - Clocks Tool

This section describes the Command Line Interface (CLI) commands supported by the Clocks Tool.
Table 26. Commands supported in Clocks

<table>
<thead>
<tr>
<th>Command name</th>
<th>Definition and parameters</th>
<th>Description</th>
<th>Restriction</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable tool</td>
<td>-Enable</td>
<td>Enables tool if it is disabled in the current configuration</td>
<td>Requires - HeadlessTool Clocks</td>
<td>-HeadlessTool Clocks - Enable</td>
</tr>
<tr>
<td>Import C files</td>
<td>-ImportC</td>
<td>Imports .c files into the configuration</td>
<td>Requires - HeadlessTool Clocks</td>
<td>-ImportC C:/imports/file1.c C:/imports/file2.c</td>
</tr>
<tr>
<td>Export all generated files</td>
<td>-ExportAll</td>
<td>Exports generated files (with source code and all the available export objects). Code will be regenerated before export. Includes -ExportSrc and in framework - ExportMEX Argument is expected as a folder name</td>
<td>Requires - HeadlessTool Clocks</td>
<td>-ExportAll C:/exports/generated</td>
</tr>
<tr>
<td>Export Source files</td>
<td>-ExportSrc</td>
<td>Exports generated source files. Code will be regenerated before export Argument is expected as a folder name</td>
<td>Requires - HeadlessTool Clocks</td>
<td>-ExportSrc C:/exports/src</td>
</tr>
<tr>
<td>Export HTML report file</td>
<td>-ExportHTML</td>
<td>Exports generated html report file. Code will be regenerated before export Argument is expected as a folder name</td>
<td>Requires - HeadlessTool Clocks</td>
<td>-ExportHTML C:/exports/html</td>
</tr>
</tbody>
</table>

8.6.3 Command Line execution - Peripherals Tool

This section describes the Command Line Interface (CLI) commands supported by the Peripherals Tool.
Table 27. Commands supported in Peripherals Tool

<table>
<thead>
<tr>
<th>Command name</th>
<th>Definition and parameters</th>
<th>Description</th>
<th>Restriction</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable tool</td>
<td>-Enable</td>
<td>Enables the tool if it is disabled in the current configuration</td>
<td>Requires -HeadlessTool Peripherals</td>
<td>-HeadlessTool Peripherals -Enable</td>
</tr>
<tr>
<td>Import C files</td>
<td>-ImportC</td>
<td>Imports .c files into configuration</td>
<td>Requires -HeadlessTool Peripherals</td>
<td>-HeadlessTool Peripherals -ImportC C:/imports/file1.c C:/imports/file2.c</td>
</tr>
<tr>
<td>Export all generated files</td>
<td>-ExportAll</td>
<td>Exports generated files (with source code etc.)</td>
<td>Requires -HeadlessTool Peripherals</td>
<td>-HeadlessTool Peripherals -ExportAll C:/exports/generated</td>
</tr>
<tr>
<td>(to simplify all exports commands to one command)</td>
<td></td>
<td>The code will be regenerated before export</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Includes -ExportSrc, -ExportHTML and in framework -ExportMEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Argument is expected to be a folder</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Export Source files</td>
<td>-ExportSrc</td>
<td>Exports generated source files</td>
<td>Requires -HeadlessTool Peripherals</td>
<td>-HeadlessTool Peripherals -ExportSrc C:/exports/src</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The code will be regenerated before export</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Argument is expected to be a folder</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The code will be regenerated before export</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Argument is expected to be a folder</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8.6.4 Command Line execution - TEE Tool

This section describes the Command Line Interface (CLI) commands supported in the TEE Tool.
Table 28. Commands supported in TEE Tool

<table>
<thead>
<tr>
<th>Command name</th>
<th>Definition and parameters</th>
<th>Description</th>
<th>Restriction</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable tool</td>
<td>-Enable</td>
<td>Enables the tool if it is disabled in the current configuration</td>
<td>Requires - HeadlessTool TEE</td>
<td>-HeadlessTool TEE - Enable</td>
</tr>
<tr>
<td>Export all generated files</td>
<td>-ExportAll</td>
<td>Exports generated files (with source code etc.)</td>
<td>Requires - HeadlessTool TEE</td>
<td>-HeadlessTool TEE - ExportAll C:/exports/generate</td>
</tr>
<tr>
<td>Export Source files</td>
<td>-ExportSrc</td>
<td>Exports generated source files</td>
<td>Requires - HeadlessTool TEE</td>
<td>-HeadlessTool TEE - ExportSrc C:/exports/src</td>
</tr>
</tbody>
</table>

8.7 Managing data and working offline

With the Data Manager, you can download, import, and export processor data. This feature is especially useful if you want to make the best out of the tools while staying offline.
Figure 152. Data Manager

8.7.1 Working offline

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

8.7.2 Downloading data

You can download required processor data with Data Manager.

NOTE

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

To download processor data, do the following:

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

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

8.7.3 Exporting data

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

To export data, do the following:

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

5. Click **Finish**.

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

**NOTE**

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

### 8.7.4 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 that you want to import and click **OK**.
5. Choose the data to import by selecting the checkbox in the table.
6. Click **Finish**.

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

**NOTE**

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

### 8.7.5 Updating data

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

**NOTE**

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

**NOTE**

Internet connection is required for data update.

To update cached data, do the following:

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

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

You can clean all cached data by selecting **Clean cached**. It removes all processor, board, kit, and component data, as well as SDK info files from your computer.
### 8.8 Output path overrides

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

**NOTE**

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

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

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

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

**Example content of the output path override yaml file:**

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

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

```yaml
outputPathOverrides: []
```
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.
Chapter 10
Revision history

Table 29. Revision history

<table>
<thead>
<tr>
<th>Revision number</th>
<th>Date</th>
<th>Substantive changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>27 April 2020</td>
<td>Initial release</td>
</tr>
<tr>
<td>1</td>
<td>01 July 2021</td>
<td>Minor changes</td>
</tr>
<tr>
<td>2</td>
<td>22 December 2021</td>
<td>New features are added, screenshots are updated.</td>
</tr>
<tr>
<td>3</td>
<td>30 June 2022</td>
<td>Updated for v.12</td>
</tr>
</tbody>
</table>
Legal information

Definitions

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

Disclaimers

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

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

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

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

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

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

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

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

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

**Export control** — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.
Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

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

Evaluation products — This product is provided on an “as is” and “with all faults” basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out of the use or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer’s exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer.

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

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

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

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

Trademarks

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

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

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mall, 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.

Airfast — is a trademark of NXP B.V.

Bluetooth — the Bluetooth wordmark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NXP Semiconductors is under license.

Cadence — the Cadence logo, and the other Cadence marks found at www.cadence.com/go/trademarks are trademarks or registered trademarks of Cadence Design Systems, Inc. All rights reserved worldwide.

CodeWarrior — is a trademark of NXP B.V.

ColdFire — is a trademark of NXP B.V.

ColdFire+ — is a trademark of NXP B.V.

EdgeLock — is a trademark of NXP B.V.

EdgeScale — is a trademark of NXP B.V.

EdgeVerse — is a trademark of NXP B.V.

eIQ — is a trademark of NXP B.V.

FeliCa — is a trademark of Sony Corporation.

Freescale — is a trademark of NXP B.V.

HitAG — is a trademark of NXP B.V.