## Contents

### Chapter 1 Introduction

1.1 Versions.......................................................................................................................... 5

### Chapter 2 User Interface

2.1 Creating, saving, and opening a configuration................................................................. 7
   2.1.1 Creating a new configuration..................................................................................... 7
   2.1.2 Saving a configuration............................................................................................. 7
   2.1.3 Importing sources.................................................................................................. 7
       2.1.3.1 Importing configuration.................................................................................. 8
       2.1.3.2 Importing registers....................................................................................... 8
   2.1.4 Restoring configuration from source code............................................................... 10
2.2 Toolbar................................................................................................................................ 11
   2.2.1 Eclipse project selection......................................................................................... 11
   2.2.2 Config Tools Overview........................................................................................... 11
   2.2.3 Show Problems View............................................................................................. 11
   2.2.4 Update code.......................................................................................................... 12
   2.2.5 Functional groups.................................................................................................. 14
       2.2.5.1 Functional group properties......................................................................... 14
   2.2.6 Undo/Redo actions................................................................................................ 15
   2.2.7 Selecting the tools................................................................................................ 15
2.3 Status bar.......................................................................................................................... 15
2.4 Preferences....................................................................................................................... 16
2.5 Configuration preferences............................................................................................... 17
2.6 Problems view.................................................................................................................. 18
2.7 Registers view................................................................................................................. 19
2.8 Log view........................................................................................................................... 21
2.9 Config tools overview..................................................................................................... 21

### Chapter 3 Pins Tool

3.1 Pins routing principle..................................................................................................... 23
   3.1.1 Beginning with peripheral selection..................................................................... 23
   3.1.2 Beginning with pin/internal signal selection......................................................... 24
3.2 Workflow.......................................................................................................................... 24
3.3 Example usage................................................................................................................ 25
3.4 User interface.................................................................................................................. 27
   3.4.1 Functions............................................................................................................... 28
   3.4.2 Package................................................................................................................. 29
   3.4.3 Routed Pins view................................................................................................. 31
       3.4.3.1 View controls.............................................................................................. 32
       3.4.3.2 Filtering routed pins................................................................................... 33
   3.4.4 Peripheral Signals view....................................................................................... 33
   3.4.5 Pins view.............................................................................................................. 35
   3.4.6 Labels and identifiers......................................................................................... 36
   3.4.7 Filtering in the Pins and Peripheral Signals views.............................................. 37
   3.4.8 Highlighting and color coding............................................................................ 38
3.5 Errors and warnings...................................................................................................... 40
   3.5.1 Incomplete routing............................................................................................... 40
3.6 Code generation.............................................................................................................. 41
### Chapter 4 Clocks Tool

- **4.1 Features**
- **4.2 User interface overview**
- **4.3 Clock configuration**
- **4.4 Global settings**
- **4.5 Clock sources**
- **4.6 Setting states and markers**
- **4.7 Frequency settings**
  - **4.7.1 Pop-up menu commands**
  - **4.7.2 Frequency precision**
- **4.8 Dependency arrows**
- **4.9 Details view**
- **4.10 Clock diagram**
  - **4.10.1 Mouse actions in diagram**
  - **4.10.2 Color and line styles**
  - **4.10.3 Clock model structure**
- **4.11 Clocks menu**
- **4.12 Troubleshooting problems**
- **4.13 Code generation**
  - **4.13.1 Working with the code**
- **4.14 Clock Consumers view**

### Chapter 5 Peripherals Tool

- **5.1 Features**
- **5.2 Basic Terms and Definitions**
- **5.3 Workflow**
- **5.4 User interface overview**
- **5.5 Common toolbar**
- **5.6 Documentation view**
- **5.7 Peripherals view**
- **5.8 Components view**
- **5.9 Settings Editor**
  - **5.9.1 Quick selections**
  - **5.9.2 Settings**
  - **5.9.3 Settings Editor header**
- **5.10 SEMC Validation tool**
  - **5.10.1 Validation view**
- **5.11 Problems**
- **5.12 Code generation**

### Chapter 6 Device Configuration Tool

- **6.1 Device Configuration Data (DCD) view**
- **6.1.1 Device Configuration Data (DCD) view actions**
- **6.2 Code generation**

### Chapter 7 Trusted Execution Environment Tool

- **7.1 User Memory Regions**
- **7.2 Security access configuration view**
  - **7.2.1 Masters/Slaves**
  - **7.2.2 MPC**
  - **7.2.3 Interrupts**
  - **7.2.4 Pins**
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 utilization 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 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 and can be used in both web and desktop versions.

It's important there is exactly one project file in the directory where the MEX is located to be able parse the toolchain project (either IAR EW project file EWP, or MDK uVision UVPROJX, or ARM GCC CMakeLists.txt).

2.1.1 Creating a new configuration

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

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

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

2.1.2 Saving a configuration

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

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

2.1.3 Importing sources

To import source code files:

1. Select File > Import... from the Main Menu.
2. Select MCUXpresso Config Tools>Import Source.
3. Click Next.

4. You can select one or more C files to import using the **Browse** button in the **Import Pins Source Files** window.

5. Select how to import the files:
   
   - **Rename** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, it is automatically renamed to the indexed one. For example, if `BOARD_InitPins` already exists in the configuration then the imported function is renamed to `BOARD_InitPins1`.
   
   - **Overwrite** – All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, then the existing one is replaced with the imported one.

6. Click Finish.

**NOTE**

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

### 2.1.3.1 Importing configuration

### 2.1.3.2 Importing registers

You can import register configuration from a processor memory dump.

**NOTE**

Currently, register configuration can be imported into the Clocks Tool only.
To import register configuration, do the following:

1. Select File>Import…>Import Registers from the Main Menu. Alternatively, click the Import Registers Configuration button in the Registers view, or drag-and-drop the memory dump file anywhere in the Registers view.
2. In the Import Registers wizard, specify the location of the registers configuration. If you want a new functional group to be created, select the option, and specify the functional group name.

3. Click Finish.

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

2.1.4 Restoring configuration from source code

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

The following is an example of the settings information in the generated source code.
If this information is not corrupted, it’s possible to re-import the clock settings into the tool using the following steps.

1. Select File > Import....
2. Select Clocks Tool / Import Source Files.
3. Click Next.
4. Click Browse.
5. Navigate and select the clock_config.c file previously produced by the Clocks Tool.
6. If the settings parse successfully, clock configurations are added into the current global configuration.

2.2 Toolbar
The toolbar is located on the top of the window and includes buttons/menus of frequently used actions.

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. See Config Tools Overview for more information.

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.

![Update Files window](image)

To inspect the code difference between the versions, click the change link.
To update the project without opening the Update Files window, clear the Always show details before Update Code option.

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

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** 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.png)

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>![flag]</td>
<td>Toggle &quot;Called from default initialization function&quot; feature (in source code)</td>
</tr>
<tr>
<td>![folder]</td>
<td>Opens the <strong>Functional group properties</strong> window</td>
</tr>
</tbody>
</table>

**NOTE**

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

#### 2.2.5.1 Functional group properties

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

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

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

  **NOTE**

  Not all processors support this option.

- **Called from default initialization function** - Enable to call the function is called from the default initialization function.
- **Clock gate enable**
2.2.6 Undo/Redo actions

You can reverse your actions by clicking the following buttons:

Table 3. Undo/reto options

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Cancels the previous action</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Cancels the previous undo action</td>
</tr>
</tbody>
</table>

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, select Window>Preferences>MCUXpresso Config Tools from the Main Menu. The Preferences window will open.

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

In this window you can set the following:

- **Line ending style** – Select between Windows (CR + LF), Linux/Mac (LF), or Default (based on host).
- **Generate files read-only** – Prevents modifying the source files unintentionally. Generated source files are marked as read-only.
- **Generate source folder** - At build time, automatically creates a folder including source files.
- **Create empty configuration if no yaml is available** - Will generate a configuration even if no yaml is present.
- **Always overwrite files without asking** – Select to update existing files automatically, without prompting.

Figure 11. Preferences window
• **Always show details before Update Code** – Select to review changes before the project is updated.

• **Undo history size** – Enter the number of steps you want to undo. Enter 0 to disable.

• **Proxy connection**
  - **Direct** – Select to connect directly and avoid a proxy connection.
  - **Native** – Select to use system proxy configuration for network connection.

  **NOTE**
  The proxy settings are copied from operating system settings. In case of error, you can specify proxy information in the tools.ini file, located in the `<install_dir>/bin/` folder. Make sure the file contains the following lines:
  - `Djava.net.useSystemProxies=true` (already present by default)
  - `Dhttp.proxyHost=<somecompany.proxy.net>`
  - `Dhttp.proxyPort=80`

  **NOTE**
  Authentication is not supported.

• **Work Offline** – Select to disable both the connection to NXP cloud and the download of processor/board/kit data.

• **Processor data update** – Select from the following options:
  - **Auto Update** – Select to update the processor data automatically.
  - **Manual** – Select to be update processor data after confirmation.
  - **Disabled** – Select to disable processor data update.

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

• **Automatically load last configuration on startup** – Select to avoid the startup window and load the last used configuration instead.

### 2.5 Configuration preferences

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

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

• **Validate boot init only** – Select to validate tools dependencies only against ‘boot init’ function group. When selected, dependencies from all functional groups of all tools must be satisfied in the functional groups marked for default initialization. Clearing this option hides warnings in case the user is using complex scenarios with alternating functional groups within the application code.

• **Generate YAML** – Select to generate YAML into C sources files.

• **Generate extended information into header file** – Select to generate extended information into the header file. For projects created in earlier MCUXpresso versions, this option is selected by default.

• **Custom source file copyright header** – Select to add a custom copyright header to generated source files that don’t already contain copyright.

• **Generate code only for registers that are different from the after-reset state** – Select to generate code only for registers that are different from the after-reset state. For projects created in earlier MCUXpresso versions, this option is selected by default.

**WARNING**

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

### 2.6 Problems view

This view displays issues in individual tools and in the inter-dependencies between the tools.
To open the **Problems view**, click the **Show Problems view** button, or select **Views > Problems**.

The table contains the following information:

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

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

**NOTE**

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

**Filter buttons**

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

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

### 2.7 Registers view

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

- **Peripheral filter** dropdown list – Use to list the registers only for the selected peripheral. Select **all** to list registers for all the peripherals.

- **Show modified registers only** checkbox – Select this option to hide the registers that are left in their after-reset state or are not configured.

- **Text filter** – Use to filter content by text.

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

### Table 4. Color codes

<table>
<thead>
<tr>
<th>Color</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yellow background</td>
<td>Indicates that the bit-field has been affected by the last change made in the tool.</td>
</tr>
<tr>
<td>Gray text color</td>
<td>Indicates the bit-field 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 the progress of the tools. The Log view can show up to 100 records across all tools in the chronological order.

Each record consists of the timestamp, the name of the tool responsible for the record, the severity level, and the actual message. If no tool name is specified, the records is created by the shared functionality.

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

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

![Log view](image)

2.9 Config tools overview

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

Config Tools Overview contains the following options:

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

*NOTE*

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

- To enable/disable the tools, click the toggle button. You can navigate to the tools by clicking their icons. The following information about the tools is also available:
  - **Generated code** – Contains the list of source-code files. Click the links to open the files in the Code Preview view.
  - **Functional groups** – Contains the list of the currently active functional groups. To select the groups in the Functional groups tab in the toolbar, select the relevant links.
Figure 16. Config Tools Overview

NOTE

Unsupported tools are not displayed in the overview.
Chapter 3
Pins Tool

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

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 either of these views:

- Pins
- Peripheral Signals
- Package
- Routed Pins

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

3.1 Pins routing principle

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

This routing configuration can be done in either of these views:

1. Select the Peripheral.
2. In Routed Pins 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 Route to column in the Routed Pins 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 Routed Pins view contains items that are connectable to the selected signal (without its channel if applicable). So when selected signal is “GPIO, 6” then the Route to provides items connectable to “GPIO”.

NOTE
In the Package view there is no possibility to select pin/internal signal when a peripheral signal is connectable to more pins/internal signals.

3.1.2 Beginning with pin/internal signal selection

You can select a pin or an internal signal in the Routed Pins view.

1. Select the pin/internal signal (Route to).
2. Select one of the available Peripheral. In the Pins view, see all available peripherals/signals by clicking on the checkbox in the first column or scroll the columns to the required peripheral type.
3. For the selected peripheral, select one of the available Signals.

Items in Peripheral column in Routed Pins view have following symbols:

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

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

3.2 Workflow

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

1. In the Pins view on the left find a pin and peripheral signal in the table and configure the routing by clicking on the signal cell.

   NOTE
   This routing configuration can be similarly done in other views Peripheral Signals, Package, Routed Pins.

2. Optionally, configure the electrical properties in the Routed pins view in the middle by selecting required state.
3. Open the **Code Preview** view to inspect the source code.
4. Click the **Update Code** button in the **Toolbar** to update the code.

### 3.3 Example usage

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

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

You can use the generated files with the application code.

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

![Configure Signals in Pins View](image)

**Figure 19.** Configure Signals in Pins View

2. In the middle view, called the **Routed Pins** view, select the **Output** direction for the **TX** and **PTB20** signals.

![Select Direction](image)

**Figure 20.** Select Direction
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 the Code Preview view.

Figure 21. Generated code

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

### 3.4 User interface

Pins Tool consists of several views.
3.4.1 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.

### 3.4.2 Package

The view displaying the processor package appears in the middle of the Pins tool window. The processor package shows an overall overview of the package including resource allocation.
This view shows Package overview with pins location. In the center are the peripherals.

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>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Q]</td>
<td>Zoom in package image.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Zoom out package image." /></td>
<td>Zoom out package image.</td>
</tr>
<tr>
<td><img src="image" alt="Rotate package image." /></td>
<td>Rotate package image.</td>
</tr>
<tr>
<td><img src="image" alt="Show pins as you can see it from the bottom. This option is available on BGA packages only." /></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="image" alt="Show pins as you can see it from the top. This option is available on BGA packages only." /></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="image" alt="Show resources. This option is available on BGA packages only." /></td>
<td>Show resources. This option is available on BGA packages only.</td>
</tr>
<tr>
<td><img src="image" alt="Switch package." /></td>
<td>Switch package.</td>
</tr>
<tr>
<td><img src="image" alt="Package legend" /></td>
<td>Package legend</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.

### 3.4.3 Routed Pins view

The **Routed Pins** view displays a list of routed pins and allows further configuration. This view also allows the configuration of the electrical properties of pins and displays all the pins. 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 the 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. Use the table drop down menu to configure the pin. To configure pins, start from left to right – select the peripheral first, then the required signal, and finally, the routed pin.

See the right part of the table to configure the electrical features.

If the feature is not supported, n/a is displayed.

![Routed Pins view](image)

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

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

### 3.4.3.1 View controls

The following figure illustrates the **Routed pins** view controls.

![Routed Pins view controls](image)

**Figure 28. View controls**

**Add / remove rows:**
- To add a new row to the end of table, click on the [+ ] button.
- To remove the selected row, click on the [x] button.
- To delete a specific row or insert a new row at a given position, right-click and use the pop-up menu commands.

**Add a specific number of rows or clear the table:**
- To add a specific number of rows, specify the exact number of rows.
- To clear the table, type 0.

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

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

**Copy-paste rows:**
To copy the row, right-click any cell in the row and choose **Copy**. You can later paste the copied row into the **Routed Pins** view of another functional group or configuration by right-clicking the table and choosing **Paste**.
3.4.3.2 Filtering routed pins

The following image illustrates the filter area of the Routed Pins view.

![Filter area](image)

Figure 29. Filter area

To instantly filter rows, type the text or the search phrase in the filter area (type filter text).

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

3.4.4 Peripheral Signals view

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

<table>
<thead>
<tr>
<th>Color code</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td>Error</td>
</tr>
<tr>
<td>✓</td>
<td>Configured</td>
</tr>
<tr>
<td></td>
<td>Not configured</td>
</tr>
<tr>
<td>✓</td>
<td>Warning</td>
</tr>
<tr>
<td>✓</td>
<td>Dedicated: Device is routed by default and has no impact on the generated code.</td>
</tr>
</tbody>
</table>

Table 6. Status codes
Figure 30. Peripheral Signals view

Use the checkbox to route/unroute the selected pins.

To route/unroute multiple pins, click the peripheral and select the options in the Select signals window.
3.4.5 Pins view

The Pins view shows all the pins in a table format.
Figure 32. Pins table view

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

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

Every routed pin appears in the Routed pins table.

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

You can double-right-click a row to open a dropdown menu 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.

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

### 3.4.6 Labels and identifiers

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

Boards and kits have pre-defined labels. However, it’s also possible to define a pin label listed in the Pins and Routed Pins views.

To set/update the Labels and Identifier columns visibility, select Edit > Preferences.

The pin identifier is used to generate the `#define` in the pin_mux.h file. However, it’s an optional parameter. If the parameter is not defined, the code for `#define` is not generated. Additionally, you can define multiple identifiers, using the `;` character as a separator. You can also set the identifier by typing it directly into the cell in the Identifier column in the Pins and Routed Pins views.
In this case it's possible to select from values if the pin is routed. See Routed pins table.

![Figure 33. Pin Identifier](image)

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

![Figure 34. Identifier in Routed Pins table](image)

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

![Figure 35. Identifier errors](image)

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.

### 3.4.7 Filtering in the Pins and Peripheral Signals views

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

### 3.4.8 Highlighting and color coding

It's possible to easily identify routed pins/periherals in the package using highlighting. By default, the current selection (pin/peripheral) is highlighted in the package view.

- The pin/peripheral is highlighted by yellow border around it in the Package view. If the highlighted pin/peripheral is selected then it has a blue border around it.
- Red indicates that the pin has an error.
- Green indicates that the pin is muxed or used.
- Light grey indicates that the pin is available for mux, but is not muxed or used.
- Dark gray indicates that the pin/peripheral is dedicated. It is routed by default and has no impact on generated code.
• Package view
  — Click on the peripheral or use the pop-up menu to highlight peripherals:
3.5 Errors and warnings

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

The first two rows in the figure above show the peripheral/signal where the erroneous configuration occurs. The fourth row shows the warning on the unconfigured identifier while specifying a direction. The detailed error/warning message appears as a tooltip.

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

3.5.1 Incomplete routing

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

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.
3.6 Code generation

If the settings are correct and no error is reported, the code generation engine instantly re-generates the source code. You can view the resulting code 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 multicore, 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.

![Generated code](image)

**Figure 43. Generated code**

YAML configuration contains configuration of each pin. It stores only non-default values.
TIP
For multicore processors, it will generate source files for each core. If processor is supported by SDK, it can
generate BOARD_InitBootPins function call from main by default. You can specify "Call from BOARD_InitBootPins"
for each function, in order to generate appropriate function call.
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

Following are the features of the Clock Tool:

- Inspects and modifies element configurations on the clock path from the clock source up to the core/peripherals.
- Validates clock elements settings and calculates the resulting output clock frequencies.
- Generates a configuration code using the SDK.
- Modifies 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.
- Helps to find clock elements settings that fulfills given requirements for outputs.
- Fully integrated in tools framework along with other tools.
- Shows configuration problems in Problems view and guides the user for the resolution.

4.2 User interface overview

The Clocks tool is integrated and runs with the MCUXpresso Config Tools framework. For documentation on the common interface and menu items, see the Config Tools User Interface chapter.
4.3 Clock configuration

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

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

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

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

Figure 44. User interface

Figure 45. Default clock configuration

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

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

NOTE
Global settings can be changed at any time.

Figure 46. Global settings

4.5 Clock sources

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

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

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

Figure 47. External clock source configuration

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

4.6 Setting states and markers

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

Table 7. 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</td>
</tr>
</tbody>
</table>

Table continues on the next page...
Table 7. 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></td>
<td>when you change the value. To add/remove the lock manually, use the pop-up menu command <strong>Lock/Unlock</strong>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>NOTE</strong> The clock element settings that cannot be automatically adjusted by the tool keep their value as is and do not allow locking. These are: clock sources, clock selectors and configuration elements.</td>
</tr>
<tr>
<td>Yellow background</td>
<td>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>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 show the text “inactive” instead of frequency. The value is also gray when the value is read-only. In such a state it is not possible to modify the value.</td>
</tr>
</tbody>
</table>

4.7 Frequency settings

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

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

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

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

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

![Figure 48. Setting the core clock frequency](image)

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

![Figure 49. Tool attains the required frequency](image)

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

![Figure 50. Tool encounters problem](image)

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

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

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

![Pop-up menu commands](Image)

Figure 51. Pop-up menu commands

4.7.2 Frequency precision

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Lock</th>
<th>Value</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core clock</td>
<td></td>
<td>21 MHz [20.97... MHz]</td>
<td>±5%</td>
</tr>
</tbody>
</table>

Figure 52. Frequency precision

4.8 Dependency arrows

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

![Dependency arrows](Image)

Figure 53. Dependency arrows

4.9 Details view

The **Details** view displays and allows you to change clock-element settings information. The information is also updated in real-time based on any changes in the **Clocks Diagram** and **Clocks Table**.
In the **Details** view, you can perform the following actions:

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

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

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

![Clock diagram](image)

**Figure 55. Clock diagram**

4.10.1 Mouse actions in diagram

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

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

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

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

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

4.10.3 Clock model structure

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

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

  ![Clock source diagram](image)

  **Figure 58. Clock source**

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

  ![Clocks selector diagram](image)

  **Figure 59. Clocks selector**

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

  ![Prescaler diagram](image)

  **Figure 60. Prescaler**

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

  ![Frequency Locked Loop diagram](image)

  **Figure 61. Frequency Locked Loop**

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

  ![Phase Locked Loop diagram](image)

  **Figure 62. Phase Locked Loop**

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

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

  ![Clock output diagram](image)

  **Figure 63. Clock output**
Clocks Tool

- **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 diagram](image)

**Figure 64. 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

Commands related to the Clocks tool can be found in the Clocks menu and include the following commands:

- **Functional groups** – Opens the Functional group properties dialog.
- **Refresh** – Refreshes each clocks configuration with explicit invocation of code generation.
- **Reset To Board Defaults** – Resets the clock model to board defaults.
- **Reset To Processor Defaults** – Resets the clock model to processor defaults.
- **Unlock All Settings** – Unlocks all locks in all settings.

### 4.12 Troubleshooting problems

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

You may encounter any of the following problems:

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

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

The following are some tips to troubleshoot encountered problems.

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

- **Show problem** - Shows the problem in the **Clocks Diagram** view. If one of 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.

![Find Near Value Dialog](image)

**Figure 65. Find Near Value Dialog**

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

### 4.13 Code generation

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

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

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

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

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

4.14 Clock Consumers view

The Clock Consumers view provides an overview of peripheral instances. It also provides information on clock-clock instance pairing. This view is not editable and is for information only.
Information about which peripherals are consuming which output clock is available in the clock output tooltip.

<table>
<thead>
<tr>
<th>Consumer</th>
<th>Clocks</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC0</td>
<td>System_clock</td>
<td>150 MHz</td>
</tr>
<tr>
<td></td>
<td>Synchronous Function Clock</td>
<td>System_clock</td>
</tr>
<tr>
<td></td>
<td>Asynchronous Function Clock</td>
<td>ASYNCADC_clock</td>
</tr>
<tr>
<td>ANACTRL</td>
<td>Bus Interface Clock</td>
<td>System_clock</td>
</tr>
<tr>
<td>CASPER</td>
<td>Bus Interface Clock</td>
<td>System_clock</td>
</tr>
<tr>
<td>CRC_ENGINE</td>
<td>Bus Interface Clock</td>
<td>System_clock</td>
</tr>
<tr>
<td>CTIMER0</td>
<td>Bus Interface Clock</td>
<td>System_clock</td>
</tr>
<tr>
<td></td>
<td>Function Clock</td>
<td>CTIMER0_clock</td>
</tr>
<tr>
<td>CTIMER1</td>
<td>Bus Interface Clock</td>
<td>System_clock</td>
</tr>
<tr>
<td></td>
<td>Function Clock</td>
<td>CTIMER1_clock</td>
</tr>
<tr>
<td>CTIMER2</td>
<td>Bus Interface Clock</td>
<td>System_clock</td>
</tr>
<tr>
<td></td>
<td>Function Clock</td>
<td>CTIMER2_clock</td>
</tr>
<tr>
<td>CTIMER3</td>
<td>Bus Interface Clock</td>
<td>System_clock</td>
</tr>
<tr>
<td></td>
<td>Function Clock</td>
<td>CTIMER3_clock</td>
</tr>
<tr>
<td>CTIMER4</td>
<td>Bus Interface Clock</td>
<td>System_clock</td>
</tr>
<tr>
<td></td>
<td>Function Clock</td>
<td>CTIMER4_clock</td>
</tr>
<tr>
<td>DMA0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 67. Clock Consumers view
Chapter 5
Peripherals Tool

5.1 Features

The Peripherals Tool features:

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

5.2 Basic Terms and Definitions

The following are the basic terms and definitions used in the chapter:

- Functional group - represents a group of peripherals that are initialized as a group. The tool generates a C function for each functional group that contains the initialization code for the peripheral instances in this group. Only one functional group can be selected as default initialization, the others are treated as alternatives that are not initialized by default.

- Peripheral instance – occurrence of a peripheral (device) of specific type. For example, UART peripheral has three instances on the selected processor, so there are UART0, UART1 and UART2 devices.

- Configuration component – provides user interface for configuring SDK software component (for example, peripheral driver) and generates code for its initialization.

- Component instance – configuration component can have multiple instances with different settings. (for example, for each peripheral instance like UART0, UART1).

- Component mode – specific use-case of the component instance (for example, TRANSFER mode of DSPI, or interrupt-based mode of communication).

5.3 Workflow

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

1. In the Peripherals view, select the peripheral instance you would like to configure (use the checkbox).

2. In case more components are available for use by the peripheral, the Select component dialog appears. The Select component dialog shows the list of suitable configuration components for the selected peripheral matching the SDK driver for the selected processor.

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

4. In the settings editor that automatically opens, select the Component mode that you would like to use and configure individual settings.
The selection of the component mode may impact appearance of some settings. Therefore, the selection of the mode should be always the first step.

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

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

6. You can use Update Code command from the toolbar. If not, you can export the source code by selecting File>Export... from the Main Menu.

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

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

5.4 User interface overview

![User interface](image)

Figure 68. User interface

5.5 Common toolbar

The common toolbar provides access to commands and selections that are available in context of all MCUXpresso Config Tools. It offers the following items:

- **Config Tools Overview** - Opens the Overview with information about currently-used tools.
- **Show Problems View** - Opens the Problems view.
- **Update Code** - Opens update dialog allowing you to update generated peripheral initialization code directly within specified toolchain project.
- **Global settings** - Opens a tab aggregating global settings of all configuration sets.
Peripheral Tool

- **Functional group selection** – 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.

- Function group related icons
  - **Call from default initialization** – sets the current functional group to be initialized by the default initialization function.
  - **Functional group properties** – opens the **Functional group properties** dialog to modify name and other properties of the function group

- **Tool switching icons** – Contains icons of individual tools. Click these icons to switch the currently visible tool.

- **Undo/Redo** - Allows you to undo/redo last actions.

  **NOTE**
  For details on other commands, refer Toolbar

5.6 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 three different 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.

5.7 Peripherals view

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

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

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

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

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

Right-click the row to open a context menu. The context menu allows you to:

- **Open** - Open the component instance in the Settings Editor (if enabled). In case more instances are enabled for the peripheral, you need to choose between them.
- **Add a component instance** - Add a component instance to the peripheral.
- **Documentation** - Open the Documentation view (if applicable).
- **Remove** - Remove the component instance (if more instances are in use, a confirmation window will allow you to select which instance you want to remove).
- **Migrate** - Migrate the component to a different component type or to a component with a newer driver version.
- **Edit comment** - Add/edit custom notes for the component instance.
- **Enable/Disable** - Enable/Disable the component instance. In case more instances are enabled for the peripheral, you need to choose between them.
- **Move to** - Move the component instance to a different Functional Group.
- **Copy to** - Copy the component instance to another Functional Group.

### 5.8 Components view

The components view shows a list of configuration components, sorted by type into **Middleware/Peripheral drivers/Other**. The view displays configuration components differently based on their status:

- **Enabled** - Highlights the configuration component in light gray.
- **Enabled/with warning** - Highlights the configuration component in light gray with the alert symbol.
- **Enabled/with error** - Highlights the configuration component in red with the error symbol.
- **Disabled** - Highlights the configuration component in dark gray.
In the Components view, you can perform the following actions:

- **Display configuration-component information** - Point the mouse cursor at the configuration component to display general configuration-component information.

- **Open the Settings Editor of the configuration component** - Left-click the configuration component to open its Settings Editor.

- **View Configuration component documentation** - Right-click the configuration component and choose Documentation from the dropdown menu to view configuration-component documentation. If the configuration component isn’t documented, the option is highlighted in gray.

- **Remove the component from configuration** - Right-click the configuration component and choose Remove from the dropdown menu.

**NOTE**

If the component has any global settings, a dialog appears prompting you to confirm the removal. If the component doesn’t have any global settings, the component is deleted after removing the last instance.

- **Migrate** - Right-click the configuration component and choose Migrate to migrate the component to a different component type or to a component with a newer driver version.

- **Edit comment** - Right-click the configuration component and choose Edit comment to add/edit custom notes for that component.

- **Enable/disable the configuration component** - Right-click the configuration component and choose Enable or Disable to enable/disable the configuration component.

- **Move the configuration component to another functional group** - Right-click the configuration component and choose Move to to select from a list of functional groups to move the configuration component to.

- **Copy the configuration component to another functional group** - Right-click the configuration component and choose Copy to to select from a list of functional groups to copy the configuration component to.
• Add new configuration components - Left-click the + button and choose from the list to add a new component. You can filter the list to show only toolchain-project-relevant, or latest version components. You can also click the + buttons next to Middleware/Peripheral drivers/Other categories to add new components in them directly.

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

5.9 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.

5.9.1 Quick selections

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

![UART Configuration](image)

Figure 71. Quick selection example

5.9.2 Settings

The following settings occur in the editor.

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

![Enable Rx/Tx interrupt](image)

Figure 72. Boolean setting example

• Integer, Float – Integer or float number.
Figure 73. Integer/Float setting example

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

Figure 74. String setting example

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

Figure 75. Enumeration setting example

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

Figure 76. Set setting example

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

Figure 77.Structure setting example

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

Figure 78. Array

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

NOTE

System clipboard is not used for this purpose.
5.9.3 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).

**Settings Editor** header contains the following:

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

5.10 SEMC Validation tool

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

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

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

### 5.10.1 Validation view

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

To run validation tests, do the following:

1. Choose a scenario you wish to test in the **Scenarios** sub-view.
2. Choose the test type you want to run and the number of test repetitions in the **Choose Tests** sub-view. You can choose between the following tests:
   - DMA
   - Write-Read Compare
   - Walking Ones
   - Walking Zeros
3. Specify test parameters, such as start address, size, and pattern.
4. Start validation by clicking the **Start Validation** button in the **Scenarios** sub-view.
5. Observe the results in real-time in the **Results** sub-view.
6. Inspect the results in the **Summary** and **Logs** sub-views.

### 5.11 Problems

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

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

**Figure 82. Error decorators**

### 5.12 Code generation

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

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

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

- peripherals.c
- peripherals.h

---

**NOTE**

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

---

**NOTE**

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

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

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

---

**NOTE**

The prefixes of the global definitions (defines, constants, variables and functions) can be configured in the Properties of the functional group.
Figure 83. Code Preview

```c
/**
 * This file was generated by the S32 Configuration Tools. Any manual edits made
 * will be overwritten if the respective S32 Configuration Tools is used to update
 * the code.
 */

// clang-format off

!!GlobalInfo

product: Peripherals v6.0
processor: S32V234
package_id: S32V234_621
mcu_data: s32sdk_s32v23x_rtm_100
processor_version: 0.0.0
functionalGroups:
- name: BOARD_InitPeripherals
  called_from_default_init: true
  selectedCore: core0
  * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS ********
  /* clang-format on */

#include "peripherals_crc_1.h"

/*
 * crc_1 initialization code
 */

/* clang-format off test*/

/*
 * TEXT BELOW IS USED AS SETTING FOR TOOLS ***********/
```
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.

6.1 Device Configuration Data (DCD) view

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

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

6.1.1 Device Configuration Data (DCD) view actions

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

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

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

6.2 Code generation

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

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

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

The code in a C array format is generated in two files:
• dcd.c
• dcd.h

The code in a binary format is generated in a single file:
• dcd.bin

To change the code format, choose the required option from the dropdown menu in the Device Configuration Data (DCD) view.
Figure 85. Code Preview

```c
#include "dcd.h"

/* Component ID definition, used by tools. */

#ifndef FSL_COMPONENT_ID
#define FSL_COMPONENT_ID "platform.drivers.xip_board"
#endif

#if defined(XIP_BOOT_HEADER_ENABLE) && (XIP_BOOT_HEADER_ENABLE == 1)
#elif defined(XIP_BOOT_HEADER_DCD_ENABLE) && (XIP_BOOT_HEADER_DCD_ENABLE == 1)
#endif

__attribute__((section(".boot_hdr.dcd_data")))

#pragma location = ".boot_hdr.dcd_data"

/* TEXT BELOW IS USED AS SETTING FOR TOOLS *****************************/

#include <stdio.h>  // Include standard input/output library
#include <stdlib.h> // Include standard library

int main() {  // Define main function
    printf("Hello, World!
");  // Print message to standard output
    return 0;  // Return success code
}
```
Chapter 7
Trusted Execution Environment Tool

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

You can set security policies of different parts of your application in the Security access configuration and its sub-views, and review these policies in the Memory map and Access overview views. Use the User Memory Regions view to create a convenient overview of memory regions and their security levels.

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

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

7.1 User Memory Regions

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
Trusted Execution Environment Tool

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

Remove the memory region by selecting the table row and clicking the **Remove selected memory region(s)** button in the view's header.
Figure 87. User Memory Regions
You can import memory region configuration from other IDE projects by clicking the **Import memory regions configuration from the IDE project(s)** button in the view toolbar. Select the project you want from the list to import its memory regions settings into your current project.

**NOTE**

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

### 7.2.1 Masters/Slaves

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

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

### Masters/Slaves

<table>
<thead>
<tr>
<th>Master</th>
<th>Security level</th>
<th>Slave</th>
<th>Sec._el</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple master</td>
<td></td>
<td>ADC0</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AHB_SECURE_CTRL</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ANACTRL</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CASPER</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></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>FLEXCOMM0</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLEXCOMM1</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLEXCOMM2</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLEXCOMM3</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLEXCOMM4</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLEXCOMM5</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLEXCOMM6</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FLEXCOMM7</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GINT0</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GINT1</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPIO</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HASHCRYPT</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INPUTMUX</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IOCIN</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAILBOX</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRT0</td>
<td>NS-User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OSTMER</td>
<td>NS-User</td>
</tr>
</tbody>
</table>

Figure 90. Masters/Slaves
7.2.2 MPC

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

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

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

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

Figure 91. MPC

7.2.3 Interrupts

In the Interrupts sub-view, you can set security designation for device's peripheral interrupts. In case that the processor contains more than a single core or processing unit, additional Handling by Core tables might appear. In these tables, you can specify if the interrupts coming from the peripheral can be handled by the core or processing unit.
All interrupts are set to **Secure** by default. If you want to change the interrupt source's security designation, left-click the **Secure** cell of the interrupt and choose from the dropdown menu. Alternatively, right-click the interrupt's **Name** cell and choose the security designation from the context menu. To select multiple entries, use the **Ctrl+Left-click** shortcut, then right-click the selected area for the context menu.

<table>
<thead>
<tr>
<th>Name</th>
<th>Interrupt handling by Core1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Allow</td>
</tr>
<tr>
<td>Analog comparator and Capacitive Touch</td>
<td>Allow</td>
</tr>
<tr>
<td>CASPER</td>
<td>Allow</td>
</tr>
<tr>
<td>Flexcomm interface 0</td>
<td>Allow</td>
</tr>
<tr>
<td>Flexcomm interface 1</td>
<td>Allow</td>
</tr>
<tr>
<td>Flexcomm interface 2</td>
<td>Allow</td>
</tr>
<tr>
<td>Flexcomm interface 3</td>
<td>Allow</td>
</tr>
<tr>
<td>Flexcomm interface 4</td>
<td>Allow</td>
</tr>
<tr>
<td>Flexcomm interface 5</td>
<td>Allow</td>
</tr>
<tr>
<td>Flexcomm interface 6</td>
<td>Allow</td>
</tr>
<tr>
<td>Flexcomm interface 7</td>
<td>Allow</td>
</tr>
<tr>
<td>GPIO group 0</td>
<td>Allow</td>
</tr>
<tr>
<td>GPIO group 1</td>
<td>Allow</td>
</tr>
<tr>
<td>HASH-AES</td>
<td>Allow</td>
</tr>
<tr>
<td>High speed SPI</td>
<td>Allow</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Interrupt security configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Secure</td>
</tr>
<tr>
<td>Analog comparator and Capacitive Touch</td>
<td>Secure</td>
</tr>
<tr>
<td>CASPER</td>
<td>Secure</td>
</tr>
<tr>
<td>Flexcomm interface 0</td>
<td>Secure</td>
</tr>
<tr>
<td>Flexcomm interface 1</td>
<td>Secure</td>
</tr>
<tr>
<td>Flexcomm interface 2</td>
<td>Secure</td>
</tr>
<tr>
<td>Flexcomm interface 3</td>
<td>Secure</td>
</tr>
<tr>
<td>Flexcomm interface 4</td>
<td>Secure</td>
</tr>
<tr>
<td>Flexcomm interface 5</td>
<td>Secure</td>
</tr>
<tr>
<td>Flexcomm interface 6</td>
<td>Secure</td>
</tr>
<tr>
<td>Flexcomm interface 7</td>
<td>Secure</td>
</tr>
<tr>
<td>GPIO group 0</td>
<td>Secure</td>
</tr>
<tr>
<td>GPIO group 1</td>
<td>Secure</td>
</tr>
<tr>
<td>HASH-AES</td>
<td>Secure</td>
</tr>
<tr>
<td>High speed SPI</td>
<td>Secure</td>
</tr>
</tbody>
</table>

Figure 92. Interrupts
7.2.4 Pins

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

All pins' reading GPIO state is set to Allow by default. If you want to change the pins reading GPIO state, left-click the Reading GPIO state cell of the pin and choose from the dropdown menu. Alternatively, right-click the pin's Name cell and choose the reading GPIO state from the context menu. To select multiple entries, use the Ctrl+Left-click shortcut, then right-click the selected area for the context menu.
7.2.5 SAU

In the SAU sub-view, you can enable and configure SAU (Security attribution unit).
When enabled, you can set up SAU memory regions, specify their start and size or end address, and specify their security level. SAU automatically sets the entire memory space to a secure severity level when disabled. It also sets the entire memory space to a secure security level when enabled but without set memory regions.

You can choose between two security levels:

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

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

**NOTE**

This option is only available when SAU is disabled.

You can also decide to generate code even for disabled memory regions by selecting the option **Generate sources for disabled regions**.
In the Miscellaneous sub-view, you can set various configuration options. The list of these options depends on processor data, and varies greatly. All the options influence your register settings, and can be inspected in the Register view. Only some of the options directly influence configuration you have made in the Security access configuration view. Point your cursor over individual options to display a tooltip explaining the function of each option.
### Memory attribution map

In the Memory attribution map, you can view security levels set for memory regions. This view is read-only.
7.3.1 Core 0

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

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

To set the display options, do the following:

1. Click the **Filter access for** checkbox to enable filtering options.
2. Select the master security access you want to review by choosing from the **Master** dropdown menu.
3. Optionally, set the security level of the selected master by choosing from the **Security mode** dropdown menu. This setting has no effect on the configuration.
4. Optionally, customize the output by de-selecting the **Show details** and **Merged SAU+IDAU** options.
5. Optionally, filter displayed memory regions in the **Filter** area.

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

Double-click the cell to open the pertinent settings in **Security access configuration**.
In the **Other masters** sub-view, you can review security attributes of memory in relation to access rights by master other than Core 0. The table is read-only.

To set the display options, do the following:

### 7.3.2 Other masters

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

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

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

Double-click the cell to open the pertinent settings in **Security access configuration**.
7.4 Access Overview

In Access Overview, you can review security policies you have set in Security access configuration view.

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

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

Figure 97. Other masters
Point your cursor at an entry to display a tooltip with information about the entry. You can group the displayed information by security or by masters by using the button on the right-hand side of the toolbar.

Figure 98. Access Overview
7.5 Code Generation

If the settings are correct and no error is reported, the code generation engine instantly re-generates the source code. You can view the resulting code the Code Preview view of the 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.
Chapter 8
Advanced Features

8.1 Switching the processor

You can switch the processor or the package of the current configuration to a different one. However, switching to a completely different processor may lead to problems, such as inaccessible pin routing or unsatisfiable clock-output frequency. In that case, it's necessary to fix the problem manually. For example, go to the Pins Routing table and re-configure all pins which report an error or conflict. Alternatively, you may need to change the required frequencies on Clock output.

Select File > Switch processor menu to change the processor in the selected configuration.

![Switch processor](image)

Select File > Switch package to change the package of the current processor.

Select File > Switch package to change the package of the current processor.
8.2 Exporting the Pins table

To export the Pins table, do the following:

1. Select File > Export from the main menu.
2. In the Export dialog, select the Export the Pins in CSV (Comma Separated Values) Format option.
3. Click Next.
4. Select the folder and specify the file name to which you want to export.
5. The exported file contains content of the current Pins view table, plus 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.

**Figure 102. Aligning to block**

### 8.3 Tools advanced configuration

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

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

### 8.4 Generating HTML report

Select Export > Pins/Clocks/Peripherals Tool > Export HTML Report to generate the report.

### 8.5 Exporting sources

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

To launch the Export wizard:

1. Select File > Export from the Main Menu.
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.

### 8.6 Exporting registers

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

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

8.7 Command line execution

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

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

Notes regarding command line execution:

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

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

The following commands are supported in the framework:

<table>
<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>Force language</td>
<td>-nl {lang}</td>
<td>Force set language</td>
<td>Removal of the '.nxp' folder from home directory is recommended, as some text might be cached</td>
<td>-nl zh</td>
</tr>
</tbody>
</table>

Table continues on the next page...
<table>
<thead>
<tr>
<th>Command</th>
<th>Option</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show console</td>
<td>-consoleLog</td>
<td>Log output is also sent to Java’s System.out (typically back to the command shell if any)</td>
<td>None</td>
</tr>
<tr>
<td>Select MCU</td>
<td>-MCU</td>
<td>MCU to be selected by framework</td>
<td>Requires -SDKversion command -MCU MK64FX512xxx12</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 -SDKversion test_ksdk2_0</td>
</tr>
<tr>
<td>Select part number</td>
<td>-PartNum</td>
<td>Select specific package of the MCU</td>
<td>Requires -MCU and -SDKversion commands -PartNum MK64FX512VLL12</td>
</tr>
<tr>
<td>Configuration name</td>
<td>-ConfigName</td>
<td>Name of newly created configuration - used in export</td>
<td>Name is used when new configuration is created by -MCU and -SDKversion commands -ConfigName &quot;MyConfig&quot;</td>
</tr>
<tr>
<td>Select tool</td>
<td>-HeadlessTool</td>
<td>Select a tool that should be run in headless mode</td>
<td>None</td>
</tr>
<tr>
<td>Load configuration</td>
<td>-Load</td>
<td>Load existing configuration from (*.mex) file</td>
<td>None</td>
</tr>
<tr>
<td>Export Mex</td>
<td>-ExportMEX</td>
<td>Export .mex configuration file after tools run Argument is expected as a folder name</td>
<td>None</td>
</tr>
<tr>
<td>Export all generated files</td>
<td>-ExportAll</td>
<td>Export generated files (with source code and so on. Code is regenerated before export)</td>
<td>Requires -HeadlessTool command -HeadlessTool Pins -ExportAll C:/exports/generated</td>
</tr>
<tr>
<td>Create new configuration by importing toolchain project</td>
<td>-ImportProject {path}</td>
<td>Creates new configuration by importing toolchain project Parameter is path to the root of the toolchain project</td>
<td>Requires -HeadlessTool command -HeadlessTool Pins -ImportProject c:\test \myproject</td>
</tr>
<tr>
<td>Specify SDK path</td>
<td>-SDKpath</td>
<td>Specify absolute path to the root directory of the SDK package. @since v3.0</td>
<td>-SDKpath c:\nxp \SDK_2.0_MKL43Z256 xxx4</td>
</tr>
</tbody>
</table>
8.7.1 Command line execution - Pins Tool

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

Table 9. 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>Import C files</td>
<td>-ImportC</td>
<td>Import .c files into configuration Importing is done after loading mex and before generating outputs</td>
<td>Requires -HeadlessTool Pins</td>
<td>-HeadlessTool Pins - ImportC C:/imports/file1.c C:/imports/file2.c</td>
</tr>
<tr>
<td>Import DTSI files</td>
<td>-ImportDTSI</td>
<td>Import .dtsi files into configuration Importing is done after loading mex and before generating outputs</td>
<td>Requires -HeadlessTool Pins</td>
<td>-HeadlessTool Pins - ImportDTSI C:/imports/file1.dtsi C:/imports/file2.dtsi</td>
</tr>
<tr>
<td>Export all generated files</td>
<td>-ExportAll</td>
<td>Export generated files (with source code etc.) Code will be regenerated before export Includes -ExportSrc,-ExportCSV,-ExportHTML and in framework -ExportMEX Argument is expected as a folder name</td>
<td>Requires -HeadlessTool Pins</td>
<td>-HeadlessTool Pins - ExportAll C:/exports/generated</td>
</tr>
<tr>
<td>Export Source files</td>
<td>-ExportSrc</td>
<td>Export generated source files. Code will be regenerated before export Argument is expected as a folder name</td>
<td>Requires -HeadlessTool Pins</td>
<td>-HeadlessTool Pins - ExportSrc C:/exports/src</td>
</tr>
<tr>
<td>Export CSV file</td>
<td>-ExportCSV</td>
<td>Export generated csv file. Code will be regenerated before export Argument is expected as a folder name</td>
<td>Requires -HeadlessTool Pins</td>
<td>-HeadlessTool Pins - ExportSrc C:/exports/src</td>
</tr>
<tr>
<td>Export HTML report file</td>
<td>-ExportHTML</td>
<td>Export generated html report file. Code will be regenerated before export Argument is expected as a folder name</td>
<td>Requires -HeadlessTool Pins</td>
<td>-HeadlessTool Pins - ExportHTML C:/exports/html</td>
</tr>
</tbody>
</table>

*Table continues on the next page...*
Table 9. 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>Export registers</td>
<td>- ExportRegisters</td>
<td>Export registers tab into folder. Code will be regenerated before export</td>
<td>Requires -HeadlessTool Pins</td>
<td>-HeadlessTool Pins - ExportRegisters C:/exports/regs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Argument is expected as a folder name</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8.7.2 Command line execution - Clocks Tool

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

Table 10. 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>Import C files</td>
<td>-ImportC</td>
<td>Import .c files into 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>Export generated files (with source code and so on. Code is regenerated before export)</td>
<td>Requires -HeadlessTool Clocks</td>
<td>-ExportAll C:/exports/generated</td>
</tr>
<tr>
<td>Export Source files</td>
<td>-ExportSrc</td>
<td>Export generated source files. Code will be regenerated before export</td>
<td>Requires -HeadlessTool Clocks</td>
<td>-ExportSrc C:/exports/src</td>
</tr>
</tbody>
</table>
8.7.3 Command line execution - Peripherals Tool

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

Table 11. 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>Export all generated files</td>
<td>-ExportAll</td>
<td>Export 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>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>Export generated source files.</td>
<td>Requires -HeadlessTool Peripherals</td>
<td>-HeadlessTool Peripherals -ExportSrc C:/exports/src</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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>

* for internal commands, internal plugin must be installed into production application

8.7.4 Command line execution - Project Cloner

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

Table 12. Commands supported in Project Cloner

<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>Specify SDK path</td>
<td>-SDKpath {path}</td>
<td>Specify absolute path to the root directory of the SDK package</td>
<td>@since v3.0</td>
<td>-SDKpath c:\nxp \SDK_2.0_MKL43ZZ256 xxx4</td>
</tr>
</tbody>
</table>

Table continues on the next page...
### Table 12. Commands supported in Project Cloner (continued)

<table>
<thead>
<tr>
<th>Clone SDK example project</th>
<th>-PG_clone {board} {example} {toolchain} {wrkspc} {prjName}</th>
<th>Clones specified SDK example projecte under new name</th>
<th>Requires - HeadlessTool PrjCloner and -SDKpath (path) @since v3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1. {board} - subdirectory of the board in SDK package</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. {example} - relative path from board sub-dir and name of the example, for example demo_apps/ hello_world; use '/' as a path separator</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. {toolchain} - id of the toolchain to create project (see toolchains - toolchain - id)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. {wrkspc} - absolute path where new project shall be created, e.g. projects workspace</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5. {prjName} - name of the new project</td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

```bash
PG_clone twrk64f120m demo_apps/hello kds c:\tmp exmpl
```

---

### 8.8 Managing data and working offline

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

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

8.8.2 Downloading data

You can download required processor data with Data Manager.

**NOTE**

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

To download processor data, do the following:

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

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

8.8.3 Exporting data

You can export downloaded processor data with Data Manager in a ZIP format.

To export data, do the following:

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

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

---

**NOTE**

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

---

### 8.8.4 Importing data

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

To import data, do the following:

1. Select **Config Tools>Data Manager** from the **Main Menu**.
2. In the **Data Manager**, select **Import**.
3. In the **Import Processor Data** window, specify the location of the ZIP file you want to import.
4. Choose the data to import by selecting the checkbox.
5. 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**.

---

**NOTE**

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

---

### 8.8.5 Updating data

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

---

**NOTE**

If you select the relevant option in **Window > Preferences > MXUXpresso Config Tools** in the **Main Menu**, 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. Select **Config Tools>Data Manager** from the **Main Menu**.
2. In the **Data Manager**, filter outdated data by clicking **Select outdated**.
3. Click **Update / Download** and confirm.
   
   You can always check versions of your data by clicking **Cached only** and comparing version information in the **Local Version** and **Remote Version** columns.

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

---

**NOTE**

Doesn't affect user templates.
Chapter 9
Support

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