MCUXpresso Config Tools User's Guide (IDE)
## Contents

### Chapter 1 Introduction
1.1 Versions ................................................................. 4

### Chapter 2 Config Tools User Interface
2.1 Creating, saving, and opening a configuration ................................................................. 6
  2.1.1 Creating a new configuration ......................................................................................... 6
  2.1.2 Saving a configuration ................................................................................................. 6
  2.1.3 Importing sources ........................................................................................................ 6
    2.1.3.1 Importing configuration ......................................................................................... 7
    2.1.3.2 Importing registers ............................................................................................... 7
  2.1.4 Restoring configuration from source code ................................................................. 9
2.2 Toolbar .......................................................................................... 10
  2.2.1 Update project .......................................................................................................... 10
  2.2.2 Eclipse project selection ......................................................................................... 12
  2.2.3 Functional groups ...................................................................................................... 13
    2.2.3.1 Functional group properties ................................................................................. 14
  2.2.4 Switching the tools .................................................................................................... 14
2.3 Status bar ........................................................................................ 14
2.4 Preferences ..................................................................................... 15
2.5 Configuration preferences ......................................................................................... 16
2.6 Problems view ..................................................................................... 16
2.7 Registers view ....................................................................................... 17
2.8 Log view ............................................................................................. 19
2.9 Config tools overview ......................................................................................... 19

### Chapter 3 Pins Tool
3.1 Pins routing principle ................................................................. 21
  3.1.1 Beginning with peripheral selection ......................................................................... 21
  3.1.2 Beginning with pin/internal signal selection ............................................................ 22
3.2 Workflow ........................................................................................ 22
3.3 Example usage .................................................................................. 22
3.4 User interface .................................................................................... 25
  3.4.1 Functions ............................................................................................................... 26
  3.4.2 Package ................................................................................................................. 27
  3.4.3 Routed Pins view ..................................................................................................... 30
    3.4.3.1 View controls ................................................................................................... 30
    3.4.3.2 Filtering routed pins ........................................................................................ 31
  3.4.4 Peripheral Signals view ............................................................................................ 32
  3.4.5 Pins table view ....................................................................................................... 34
    3.4.5.1 Labels and identifiers ....................................................................................... 35
  3.4.6 Filtering in the Pins and Peripheral Signals views .................................................... 36
    3.4.7 Highlighting and color coding .............................................................................. 37
3.5 Errors and warnings ........................................................................ 39
  3.5.1 Incomplete routing ................................................................................................. 39
3.6 Code generation .............................................................................. 39

### Chapter 4 Clocks Tool
4.1 Features ......................................................................................... 41

---

MCUXpresso Config Tools User's Guide (IDE), Rev. 4, 28 October 2018

NXP Semiconductors
Chapter 5 Peripherals Tool ............................................................................................................. 55

Chapter 6 Advanced Features ......................................................................................................... 63

Chapter 7 Support ............................................................................................................................ 73
Chapter 1
Introduction

The MCUXpresso Config Tools set is a suite of evaluation and configuration tools that helps you from first evaluation to production software development. It includes the following tools.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pins Tool</td>
<td>Enables you to configure the pins of a device. Pins Tool enables you to create, inspect, change, and modify any aspect of the pin configuration and muxing of the device.</td>
</tr>
<tr>
<td>Clocks Tool</td>
<td>Enables you to configure initialization of the system clock (core, system, bus, and peripheral clocks) and generates the C code with clock initialization functions and configuration structures.</td>
</tr>
<tr>
<td>Peripherals Tool</td>
<td>Enable you to configure the initialization for the MCUXpresso SDK drivers.</td>
</tr>
</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 2. Web version of Pins Tool
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.

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.

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.

Configuration is also automatically saved during Update Project Code action.

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 dialog.

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 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 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 *.dti 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 frequently used actions.

#### 2.2.1 Update project

To update the generated code in the related toolchain project, click the **Update Project** button. In the dialog, select the tools or files you want to update. If update is not possible, the button is filled with a black square. The reason is displayed in the tooltip.
To inspect the code difference between the versions, click the *show differences* link.

![Update Project Files dialog](image)
To update the project without opening the Update Project Files dialog, clear the Always show details before Update Project option.

To access the Update Project Files dialog from the Update Project drop-down menu, select Open Update Project Dialog.

The generated code is always overwritten.

The Update Project action is enabled under following conditions:
- Processor selected in the tool matches with processor selected in the toolchain project
- Core is selected (for multicore processors)

### 2.2.2 Eclipse project selection

You can use the Eclipse project drop-down menu to switch between projects.
2.2.3 Functional groups

Each configuration can contain several functional groups. These groups represent functions which will be generated into source code. Use the drop-down menu to switch between functional groups and configure them.

Additional buttons can be used on functional groups:

- Toggle "Called from default initialization function" feature (in source code)

- Open the Functional group properties dialog

Red/orange background indicates errors/warnings in the configuration.
2.2.3.1 Functional group properties

In this dialog, 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).

**Set custom #define prefix:** If enabled, it uses the specified prefix for the identifiers in the source code. You can also modify functions order (on the left), the order is applied in the generated code.\(^1\)

Configure the **Called from default initialization function** option to set it for the function. If the option is set, the function is called from the default initialization function.

2.2.4 Switching the tools

Buttons on the right side of the toolbar represent available tools. Click them to quickly navigate between Clocks, Pins and Peripherals tools.

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.

\(^1\) *if supported by processor*
2.4 Preferences

To configure preferences, select Window>Preferences>MCUXpresso Config Tools from the Main Menu. The Preferences dialog appears.

In this dialog 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.
- **Always overwrite files without asking** – Select to update existing files automatically, without prompting.
- **Always show details before Update Project** – 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**
2.5 Configuration preferences

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

To configure the preferences related to the configuration, use the 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.

**WARNING**

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

2.6 Problems view

This view displays problems in the tools and the inter-dependencies between the tools.

To open the Problems view, select Views > Problems.

The table contains the following information:
• **Level** – Lists the severity of the problem: Information, Warning, or Error.

• **Issue** – Description of the problem.

• **Origin** – Information on the dependency source.

• **Target** – Lists the tool that handled the dependency and where it should be fulfilled.

• **Resource** – Lists the resource which is related to the problem. For example, the signal name, the clock signal, and so on.

• **Type** – The type of the problem. It is either the validation that is checking dependencies between the tools, or the tool problem that describes problem related just to one tool.

**Context-menu**

Every problem comes with a context menu accessible by right-clicking the problem row. Use this menu to access information about the problem or to apply a quick fix where applicable.

---

**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.

- 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** drop-down 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 2. 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)

Figure 16. Log view

2.9 Config tools overview

By default, the Config Tools Overview icon is located on the left side of the toolbar, and opens a dialog with 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 the toolchain project information.

• **Pins/Clocks/Peripherals** – Displays basic information about the **Pins**, **Clocks**, and **Peripherals** 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.

![Config Tools Overview dialog](image)
Chapter 3
Pins Tool

The 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.

3.1 Pins routing principle

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

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

This routing configuration can be done in 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.

3.1.1 Beginning with peripheral selection

You can select peripheral in the **Routed Pins** view and the **Peripheral Signals** view.

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 Peripherals. 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 Pins views Peripheral Signals, Package, Routed Pins.

2. Optionally, configure the electrical properties in the Routed pins view in the middle by selecting required state.

   **NOTE**
   Source code is automatically generated.

3. Open the Code Preview view to inspect the source code.

4. Click the Update Project 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’.

![Figure 19. Configure Signals in Pins View](image)

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

![Figure 20. Select Direction](image)

**NOTE**

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

3. The Pins Tool automatically generates the source code for pin_mux.c and pin_mux.h on the right panel of the Code Preview view.
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.
NOTE

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 create 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 processor package appears in the middle of the Pins Tool window. The processor package shows an overall overview of the package including resource allocation.
Figure 26. Processor package

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><img src="" alt="Icon" /></td>
<td>Zoom in package image.</td>
</tr>
</tbody>
</table>

*Table continues on the next page...*
### Table 3. Toolbar options (continued)

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![image icon]</td>
<td>Zoom out package image.</td>
</tr>
<tr>
<td>![rotate icon]</td>
<td>Rotate package image.</td>
</tr>
<tr>
<td>![low angle icon]</td>
<td>Show pins as you can see it from the bottom. This option is available on BGA packages only.</td>
</tr>
<tr>
<td>![top angle icon]</td>
<td>Show pins as you can see it from the top. This option is available on BGA packages only.</td>
</tr>
<tr>
<td>![resources icon]</td>
<td>Show resources. This option is available on BGA packages only.</td>
</tr>
<tr>
<td>![switch icon]</td>
<td>Switch package.</td>
</tr>
<tr>
<td>![legend icon]</td>
<td>Package legend</td>
</tr>
</tbody>
</table>

#### NOTE

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

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

![Switch package](image)

**Figure 27. Switch package**

The **Switch package for the Processor** dialog 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.

<table>
<thead>
<tr>
<th>#</th>
<th>Peripheral</th>
<th>Signal</th>
<th>Route to</th>
<th>Label</th>
<th>Identifier</th>
<th>Direction</th>
<th>GPIO initial state</th>
<th>GPIO interrupt</th>
<th>Slew rate</th>
<th>Open drain</th>
<th>Drive strength</th>
<th>Pull select</th>
<th>Full enable</th>
<th>Passive filter</th>
<th>Digital filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FTM3</td>
<td>Ch. 6</td>
<td>FTM3-Ch6</td>
<td>FTM3-Ch6</td>
<td>SW3 Pin</td>
<td>Input</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>2</td>
<td>GPIO1</td>
<td>GPIO1_1</td>
<td>GPIO_1</td>
<td>GPIO_1</td>
<td>GPIO_1 Pin</td>
<td>Output</td>
<td>Logical 1</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

**Figure 28. Routed Pins view**

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.
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 area and choosing Paste.

3.4.3.2 Filtering routed pins
The following image illustrates the filter area of the Routed Pins view.

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

<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>
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** dialog.
Figure 32. Select signals dialog

3.4.5 Pins table view

The Pins table view shows all the pins in a table format.
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/un-route pin to the given peripheral, click in the cell of the table. Routed pins are marked with checkbox and green color. Colored cells indicate that a pin is routed to given peripherals. If there is conflict in routing, red color is used.

Unroute is possible by clicking on a given cell, or by checkbox in the first column.

Every routed pin appears in the Routed pins table.

When multiple functions are specified in the configuration, the Pins Table 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.

**TIP**

The option to route more signals to a single pin is indicated by an ellipsis (...). The **Multiple Signals Configuration** dialog appears, if clicked.

### 3.4.5.1 Labels and identifiers

It’s possible to define label of any pin that can be shown in UI for easy pin identification.

The boards and kits have pre-defined labels. However, it is also possible to define a pin label listed in the **Routed Pins** view. 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 **Routed Pins** view.
In this case it's possible to select from values if the pin is routed. See **Routed pins table**.

![Figure 34. 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 35. 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 36. Identifier errors](image)

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

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

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

![Figure 37. Pins macros prefix](image)
Type any text to search across the table/tree. It will search for the pins/peripheral signals containing the specified text. You can also use wildcards "*" and "?" to help you filter results you want. Use "space" to search for multiple strings at the same time.

### 3.4.7 Highlighting and color coding

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

- The pin/peripheral is highlighted by yellow border around it in the Package view. If the highlighted pin/peripheral is selected then it has a blue border around it.
- Red indicates that the pin has an error.
- Green indicates that the pin is muxed or used.
- Light grey indicates that the pin is available for mux, but is not muxed or used.
- Dark gray indicates that the pin/peripheral is dedicated. It is routed by default and has no impact on generated code.
Figure 39. Highlighting and color coding

Figure 40. Pins conflicts

Figure 41. Warnings

- 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 reconfiguration.

If an error or warning is encountered, the conflict in the Routed Pins view is represented in the first column of the row and the error/warning is indicated in the cell, where the conflict was created. The first two rows in the figure above show the peripheral/signal where the erroneous configuration occurs. The fourth row shows the warning on the unconfigured identifier while specifying a direction. The detailed error/warning message appears as a tooltip.

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

3.5.1 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.

3.6 Code generation

The tool generates source code that can be incorporated into an application to initialize pins routing. The source code is generated automatically on change or can be generated manually by selecting Pins > Refresh from the main menu. The generated code is shown in the Code Preview view. It shows all generated files and each file has its own tab.
For multicores, the sources are generated for each core. Appropriate files are shown with @Core #{number} tag.

---

**NOTE**

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

---

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

**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.

---

Figure 44. Generated code

YAML configuration contains configuration of each pin. It stores only non-default values.
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 45. User interface**

**Figure 46. Default clock configuration**
The code generation engine of the tool generates function with the name derived from the Clock configuration name.

4.4 Global settings

Global settings are settings that influence the entire clock system. It's recommended to set them first. Global settings can be modified in the Clock Table view.

Global settings can be changed at any time.

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.

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 5. Setting states and markers

<table>
<thead>
<tr>
<th>State/Style/ Marker</th>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error marker</td>
<td>☓</td>
<td>Indicates that there is an error in the settings or something related to it. See the tooltip of the setting for details.</td>
</tr>
<tr>
<td>Warning marker</td>
<td>☣</td>
<td>Indicates that there is a warning in the settings or something related to it. See the tooltip of the setting for details.</td>
</tr>
<tr>
<td>Lock icon</td>
<td>🛠</td>
<td>Indicates that the settings (that may be automatically adjusted by the tool) are locked to prevent any automatic adjustment. If the setting can be locked, they are automatically locked when you change the value. To add/remove the lock manually, use the pop-up menu command Lock/Unlock.</td>
</tr>
</tbody>
</table>

**NOTE**
The clock element settings that cannot be automatically adjusted by the tool keep their value as is and do not allow locking. These are: clock sources, clock selectors and configuration elements.

<table>
<thead>
<tr>
<th>Yellow background</th>
<th>100 MHz</th>
<th>Indicates that the field is directly or indirectly changed by the previous user action.</th>
</tr>
</thead>
<tbody>
<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 49. 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 50. 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:
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.

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.

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

4.10.2 Color and line styles

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

The color and line styles can indicate:

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

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

4.10.3 Clock model structure

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

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

![Clock source](image)

**Figure 60. Clock source**

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

![Clocks selector](image)

**Figure 61. Clocks selector**

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

![Prescaler](image)

**Figure 62. Prescaler**

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

![Frequency Locked Loop](image)

**Figure 63. Frequency Locked Loop**

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

![Phase Locked Loop](image)

**Figure 64. 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](image)

**Figure 65. Clock output**
• **Clock component** – Group of clock elements surrounded with a border. The clock component can have one or more outputs. The clock component usually corresponds to the processor modules or peripherals. The component output may behave like clock gates, allowing or preventing the signal flow out of the component.

![Figure 66. Clock component](image)

• **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 Main menu

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

- **Functional groups** – Invokes 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 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.

### 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.

*Figure 67. Code Preview view*
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:

- 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 Project Code in the main toolbar (works only for toolchain project).

4.14 Clock Consumers view

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

NOTE

Information about which peripherals are consuming which output clock is available in the clock output tooltip.
Figure 68. 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.

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

5. Open the **Code Preview** 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. In case you are using a toolchain project, you can use **Update project** 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

![Figure 69. User interface](image)

### 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:
• **Update project code** – this button opens update dialog allowing to update generated peripheral initialization code directly within specified toolchain project. This command is available only when the toolchain project has been specified.

• **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** – section containing icons of individual tools. Click these icons to switch the currently visible tool.

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

### 5.6 Documentation view

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

To open the Documentation view, do the following:

• In the Peripherals view, right-click the row 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.

De-selecting a previously-selected instance disables it. You can enable it by selecting the checkbox, or by clicking the switch in the settings editor of the component instance.

Second column contains a name of component instance handling the peripheral. This name is freely customizable in the settings editor and it is used in generated code.

Double-click on the second column opens the editor for the component instance.

### 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 dark gray.

• **Enabled/with warning** - Highlights the configuration component in dark gray with the alert symbol.

• **Enabled/with error** - Highlights the configuration component in red with the error symbol.
• **Disabled** - Highlights the configuration component in light gray.

![Components view](image)

**Figure 70. Components view**

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.

- **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.
- Double-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.

![Quick selection example](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).

  ![Boolean setting example](image)

  **Figure 72. Boolean setting example**

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

  ![Integer/Float setting example](image)

  **Figure 73. Integer/Float setting example**

- **String** – Textual input.

  ![String setting example](image)

  **Figure 74. String setting example**

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

![Set setting example](image)

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

![Structure setting example](image)

- **Array** – Array of multiple settings of same type – user can add/remove items. The array of simple structures may also be represented as a table grid.

The ‘+’ button adds a new item at the end of array. To rearrange the position or delete an item, click on the menu icon and select one of the following options: Move up, Move down, Move to top, Move to bottom, or Remove.

![Array setting example](image)

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

**Figure 80. Settings Editor header**

- **Component instance information**
- **Component title**
- **Documentation**

**Settings Editor** header contains the following:

- **Title** - Displays the configuration component title.
- **Name** - Displays the component instance name. This name is used in the generated code in constants and function identifiers. You can change it at any time.
- **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 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 81. Error decorators**
5.11 Code generation

The code generation is performed automatically after every change in the configuration.

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.

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.
Chapter 6
Advanced Features

6.1 Switching the processor

You can switch the processor or the package of the current configuration to a different one. However, switching to a completely different processor may lead to 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 window](image)

Figure 82. Switch processor

Select **File > Switch package** to change the package of the current processor.
6.2 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.

Figure 84. Exported file content
The exported content can be used in other tools for further processing. For example, see it after aligning to blocks in the image below.

![Figure 85. Aligning to block](image)

### 6.3 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.

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

### 6.4 Generating HTML report

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

### 6.5 Exporting sources

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

To launch the Export wizard:

1. Select File > Export from the 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**.

### 6.6 Exporting registers

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

To export registers, follow these steps:

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

### 6.7 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 {lang} is in ISO-639-1 standard</td>
<td>Removal of the '.nxp' folder from home directory is recommended, as some text might be cached</td>
<td>-nl zh</td>
</tr>
</tbody>
</table>

*Table continues on the next page...*
# Table 6. Commands supported in the framework (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Option</th>
<th>Description</th>
<th>Requires</th>
<th>Argument</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>
<td>None</td>
</tr>
<tr>
<td>Select MCU</td>
<td>-MCU</td>
<td>MCU to be selected by framework</td>
<td>Requires –SDKversion command</td>
<td>-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</td>
<td>-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</td>
<td>-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</td>
<td>-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>
<td>-HeadlessTool Clocks</td>
</tr>
<tr>
<td>Load configuration</td>
<td>-Load</td>
<td>Load existing configuration from (*.mex) file</td>
<td>None</td>
<td>-Load C:/conf/conf.mex</td>
</tr>
<tr>
<td>Export Mex</td>
<td>-ExportMEX</td>
<td>Export .mex configuration file after tools run</td>
<td>None</td>
<td>-MCU xxx -SDKversion xxx -ExportMEX C:/exports/my_config_folder</td>
</tr>
<tr>
<td>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</td>
<td>-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</td>
<td>Requires -HeadlessTool command</td>
<td>-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.</td>
<td>@since v3.0</td>
<td>-SDKpath c:\nxp \SDK_2.0_MKL43Z256 xxx4</td>
</tr>
</tbody>
</table>
### 6.7.1 Command line execution - Pins Tool

This section describes the Command Line Interface (CLI) commands supported in the Pins 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>Import C files</td>
<td>-ImportC</td>
<td>Import .c files into configuration</td>
<td>Requires -HeadlessTool Pins</td>
<td>-HeadlessTool Pins - ImportC C:/imports/file1.c C:/imports/file2.c</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Importing is done after loading mex and before generating outputs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Import DTSI files</td>
<td>-ImportDTSI</td>
<td>Import .dtsi files into configuration</td>
<td>Requires -HeadlessTool Pins</td>
<td>-HeadlessTool Pins - ImportDTSI C:/imports/file1.dtsi C:/imports/file2.dtsi</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Importing is done after loading mex and before generating outputs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Export all generated</td>
<td>-ExportAll</td>
<td>Export generated files (with source code etc.)</td>
<td>Requires -HeadlessTool Pins</td>
<td>-HeadlessTool Pins - ExportAll C:/exports/generated</td>
</tr>
<tr>
<td>files</td>
<td></td>
<td>Code will be regenerated before export</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Includes -ExportSrc,-ExportCSV,-ExportHTML and in framework -ExportMEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Argument is expected as a folder name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></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 as a folder name</td>
<td></td>
<td></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 as a folder name</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table continues on the next page...*
Table 7. 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>
</tbody>
</table>

6.7.2 Command line execution - Clocks Tool

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

Table 8. 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 Importing is done after loading mex and</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>
</tbody>
</table>

Table continues on the next page...
Table 8. Commands supported in Clocks (continued)

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

6.7.3 Command line execution - Peripherals Tool

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

Table 9. 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>
</table>
| Export all generated files     | -ExportAll                | Export generated files (with source code etc.)  
Code will be regenerated before export  
Includes -ExportSrc, -ExportHTML and in framework -ExportMEX  
Argument is expected to be a folder | Requires -HeadlessTool Peripherals | -HeadlessTool Peripherals -ExportAll C:/exports/generated |
| Export Source files            | -ExportSrc                | Export generated source files.  
Code will be regenerated before export  
Argument is expected to be a folder | Requires -HeadlessTool Peripherals | -HeadlessTool Peripherals -ExportSrc C:/exports/src |

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

6.7.4 Command line execution - Project Cloner

This section describes the Command Line Interface (CLI) commands supported by the Project Cloner.
Table 10. 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_MKL43Z256 \xxx4</td>
</tr>
<tr>
<td>Clone SDK example project</td>
<td>-PG_clone {board} (example) (toolchain) (wrkspc) (prjName)</td>
<td>Clones specified SDK example projecte under new name</td>
<td>Requires -HeadlessTool PrjCloner and -SDKpath (path) @since v3.0</td>
<td>-HeadlessTool PrjCloner -SDKpath c:\nxp \SDK_2.0_MKL43Z256 xxx4 -PG_clone twrk64f120m demo_apps/hello kds c:\tmp exmpl</td>
</tr>
</tbody>
</table>

6.8 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.
Chapter 7
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.