MCUXCTWEBUG

User Guide for MCUXpresso Config Tools (Web)

Rev. 18.0 — 17 September 2026 **User guide**

Document information

Information	Content
Keywords	MCUXpresso Config Tools
Abstract	MCUXpresso Configuration Tools set is a suite of evaluation and configuration tools that help you from initial evaluation to production software development.

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. The following tools are included:

Table 1. MCUXpresso Config Tools

Name	Description
Pins Tool	Enables you to configure the pins of a device. The Pins tool enables you to create, inspect, and modify any aspect of the pin configuration and muxing of the device.
Clocks Tool	The web version of the Clocks tool gives a preview of the system clock (core, system, bus, and peripheral clocks) and configuration structures of the clocking environment.

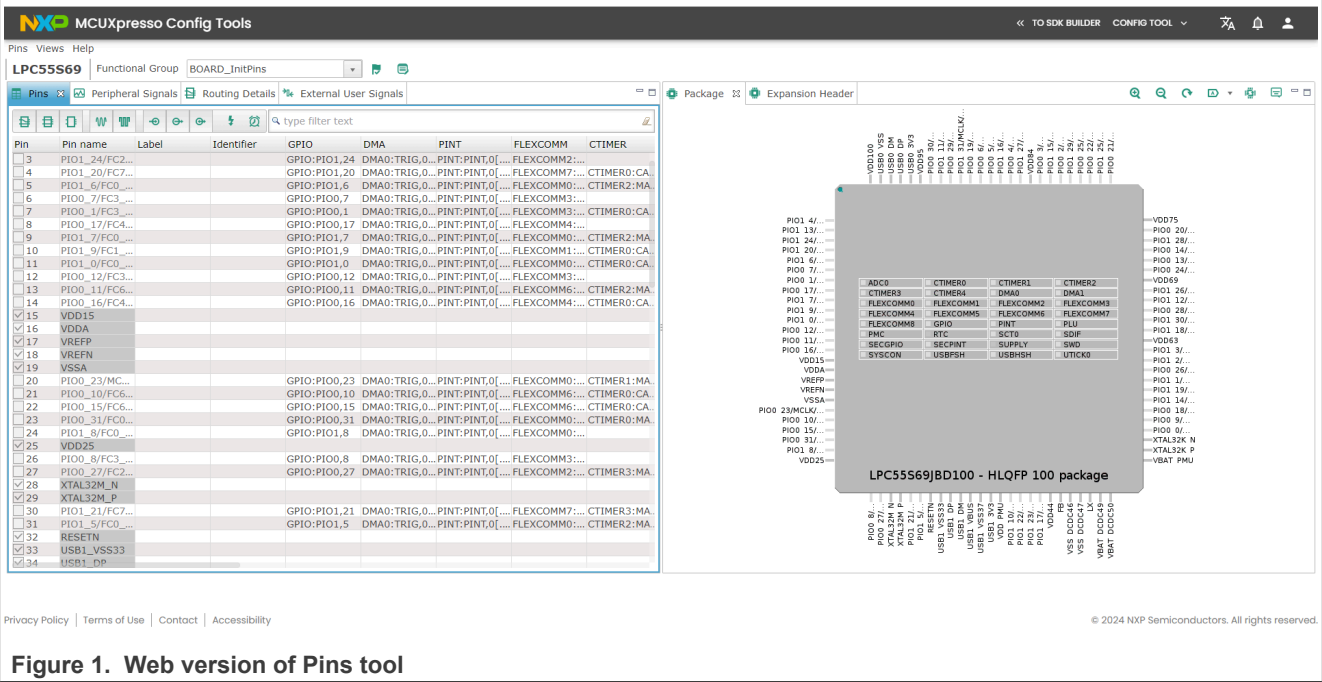


Figure 1. Web version of Pins tool

2 User interface

This section provides a detailed description of the 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.

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

Table 2. MCUXpresso Config Tools

Toolchain	Project file
IAR Embedded Workbench	EWP
Keil MDK uVision	UVPROJX
ARM GCC	CMakeLists.txt
CodeWarrior with SDK	.cproject
Open-CMSIS csolution	*.cbuild-gen-idx.yml

2.1.1 Creating a new configuration

A new configuration is created automatically on the first run of the tool. The initial processor selection is taken from the active processor/board/kit selection on the **Select Development Board** page.

If you start creating your development for any NXP board or kit, we recommend you start with an MCUXpresso SDK example to create a configuration for a board or a kit. Such a configuration contains board-specific settings. If you select a processor, the configuration will be empty.

2.1.2 Saving a configuration

The configuration is automatically saved in the background after every change.

2.1.3 Preliminary processor data

Some processors are published for public release before their configuration data reaches full RFP (Ready-For-Production) quality. Such processors are marked as **preliminary** in the processor data.

When you use a processor whose data is marked as preliminary, MCUXpresso Config Tools indicates this in the **Problems** view by displaying a warning that notifies you that the current configuration relies on preliminary processor data.

Note: Preliminary processor data is subject to change. Configurations based on preliminary data should be re-validated once the final (RFP-quality) processor data becomes available.

2.2 Main menu

This section describes the common main menu commands that are available for the Tools.

- **Pins and Clocks**
  - **Functional Groups -** Opens the Functional group properties dialog.
  - **Refresh -** Refreshes both the generated code and the whole GUI.
  - **Reset to Board Defaults** - Resets the configuration of the Board/Kit defaults.
  - **Reset to Processor Defaults** - Resets the configuration of the processor defaults.
  - **Unlock All Settings** - Clocks Tool only. Unlocks all the currently active clocks.
  - **Import** - Opens the Import dialog. The Import dialog allows you to import a saved MEX configuration file.
  - **Export** - Opens the Export dialog. The Export dialog allows you to export source files and HTML reports. For details on additional exporting of Pins and Clocks configuration from legacy tools, see the desktop version of the User Guide, section Advanced Features.
  - **Switch package** - Opens a dialog for switching packages.
  - **Automatic Routing** - Attempts to resolve routing issues. Opens the Automatic Routing dialog, which displays the routing issues that have been resolved and the ones that require manual correction.
  - **Apply Expansion Board**- Applies an expansion board to an already created expansion header.

  - **Create Default Routing** - Opens a dialog for the creation of a new functional group containing the after-reset state of pins and internal signals.
- **Views**
  - List of views available for selected tool. Click the view to open it.
- **Help**
  - **Contents -** Shows the documentation for the product.
  - **Release Notes -** Shows the release notes document for the installed version.
  - **Community -** Shows a web-browser window with web pages of the community related to the product.
  - **About -** Shows dialog with information about a product.

2.3 Toolbar

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

Table 3. Toolbar

Item	Description
Call from default initialization	Set the current functional group to be initialized by the default initialization function.
Functional group properties	Open the **Functional group properties** dialog to modify name and other properties of the function group.

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

2.3.1 Functional groups

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

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

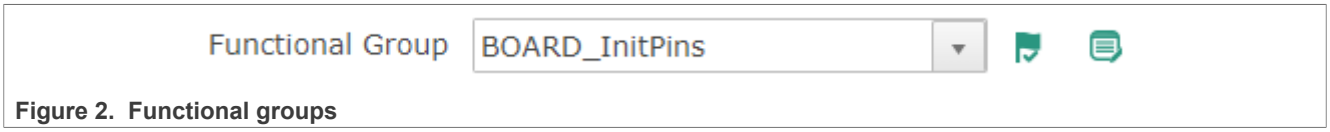


Figure 2. Functional groups

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

Table 4. Functional Groups

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

Note:

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

2.3.1.1 Functional group properties

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

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

Functional group properties are specific for individual Config Tools:

The Pins tool:

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

Clocks tool:

- **Set custom #define prefix** - If this property is set, the custom prefix is used for macros define in `clock_config.h` Otherwise, the name of the functional group is used as the prefix.
- **Prefix** - The custom prefix string. If it is empty, no prefix is used.
- **Other settings** - The processor-specific settings are specific for each processor. See the tooltips for details.
- **Prefix** - It is used for identifiers, constants, and functions related to the functional group that is used in generated code. If it is not specified, no prefix is used.
- **Set custom #define prefix** - If this property is checked, the custom prefix is used for macros define in the generated code. Otherwise, the name of the functional group is used as the prefix.

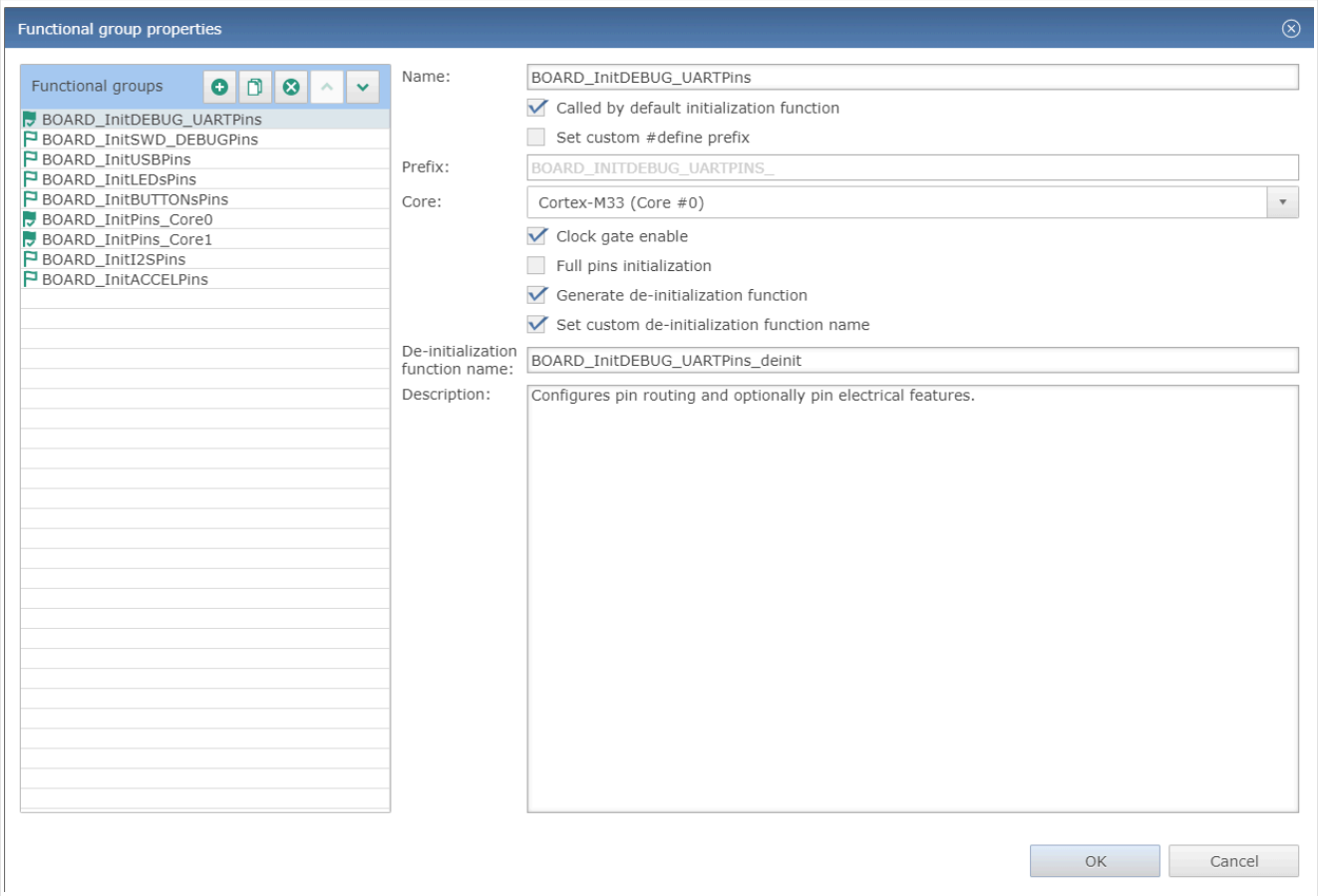


Figure 3. Functional group properties for the Pins tool

3 Pins Tool

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

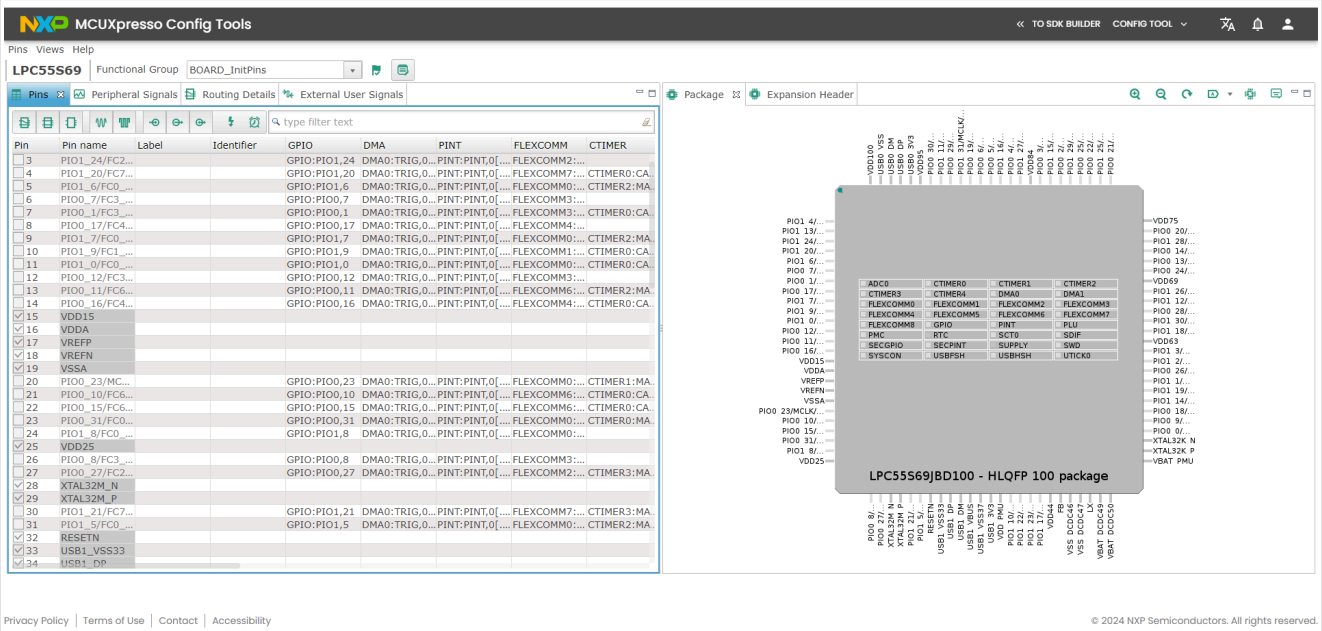


Figure 4. Pins tool

3.1 Pins routing principle

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

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

This routing configuration can be done in the following views:

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

The following two sections describe the two methods for defining the routing path.

3.1.1 Beginning with pin/internal signal selection

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

1. Select the pin/internal signal (**Routed pin/signal**).
2. Select one of the available **Peripherals**.
3. For the selected peripheral, select one of the available **Signals**.
   Items in **Peripheral** column in **Routing Details** view have the following symbols:
   - Exclamation mark and default text color indicate that such item selection can cause a register conflict or the item does not support selected signal.
   - Exclamation mark and gray text color indicate that the item cannot be routed to the selected pin/internal signal. The item is available for different pin/internal signal using the same signal.
     Note: In the **Pins** view and the **Package** view, you can configure only pins and not internal signals.

3.1.2 Routing of peripheral signals

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

Three types of peripheral signal routing are available:

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

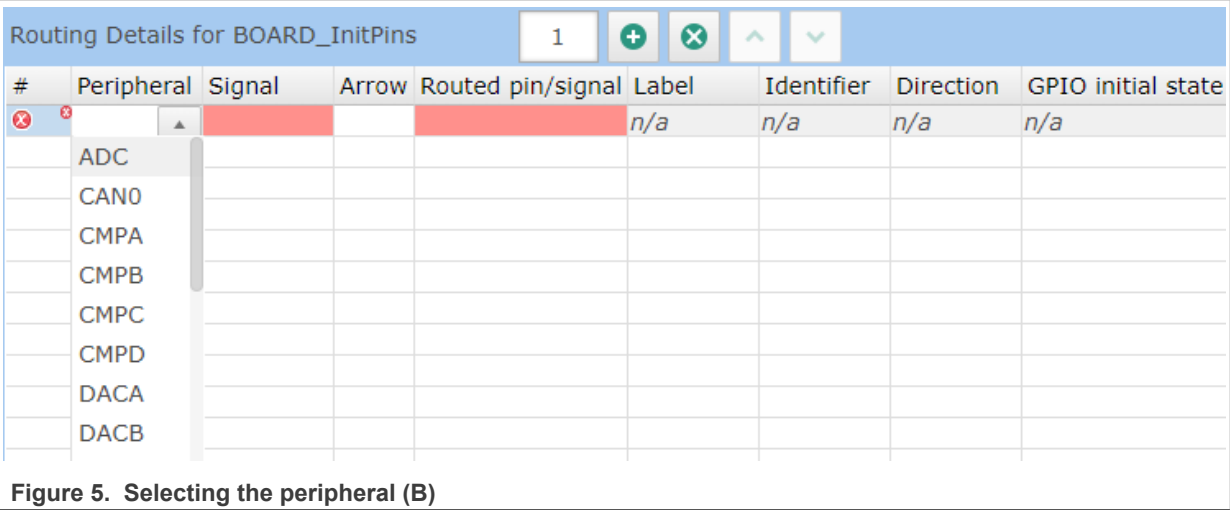


Figure 5. Selecting the peripheral (B)

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

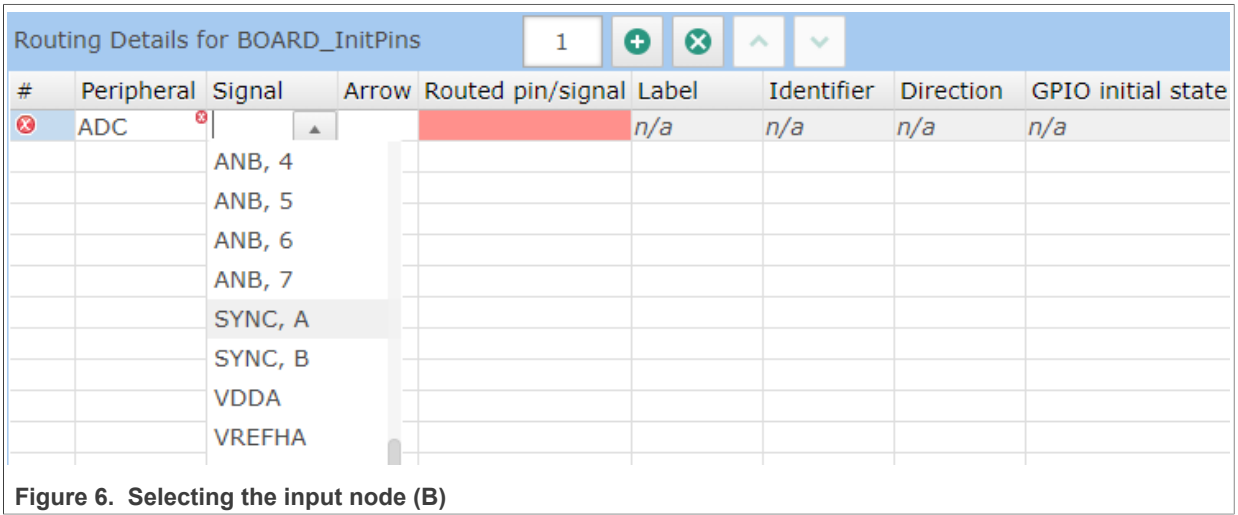


Figure 6. Selecting the input node (B)

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

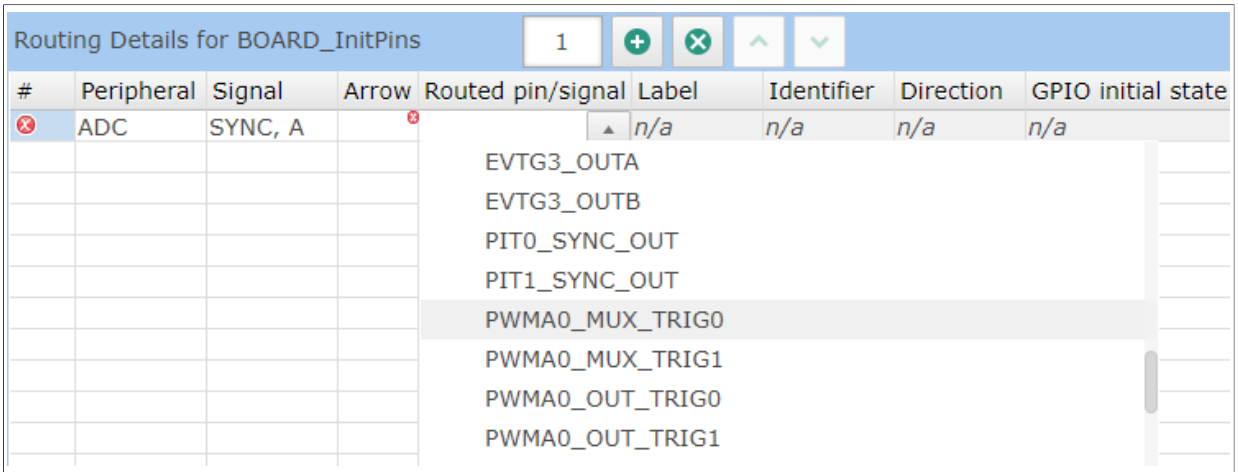


Figure 7. Selecting the output signal

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

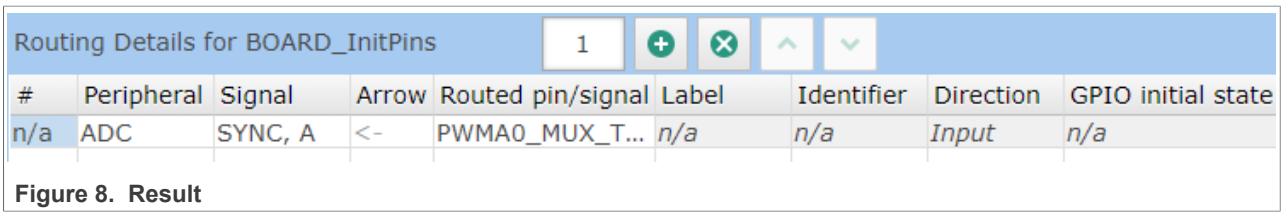


Figure 8. Result

Note: Select the ADC peripheral where the signal leads to (input in ADC). It is a limitation of the Pins tool that the signal is not listed for the PWM peripheral (output). Notice the direction of the signal in the **Arrow** column.

2. Routing the signal from a pin on the package to an internal peripheral input signal through an inter-peripheral crossbar
   Note: Only if a crossbar switch is present.
   The signal leads from a pin on the package (XB_IN) connected through an inter-peripheral crossbar, to an internal peripheral (B) input node. In other words, the signal leads from XB_IN to B (XB_IN > B). To configure a signal in this way, perform the following steps (routing pin 55 using XB_IN6 to EVTG0 input A (XB_IN6 > EVTG0) used as example):
   a. Add a row in the **Routing Details** view.
   b. Select peripheral B from the drop-down list in the **Peripheral** column.

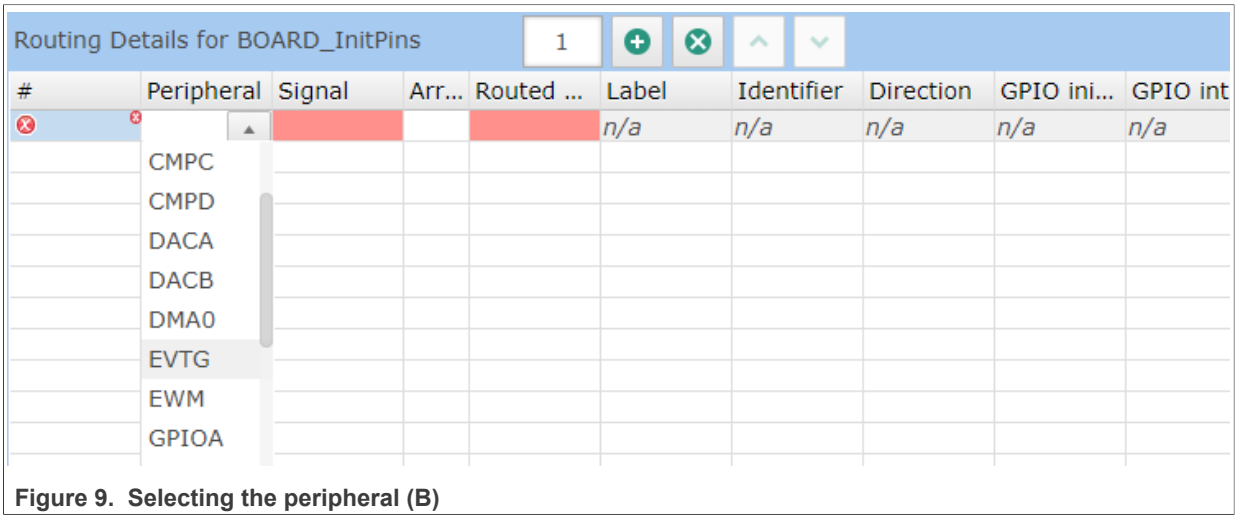


Figure 9. Selecting the peripheral (B)

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

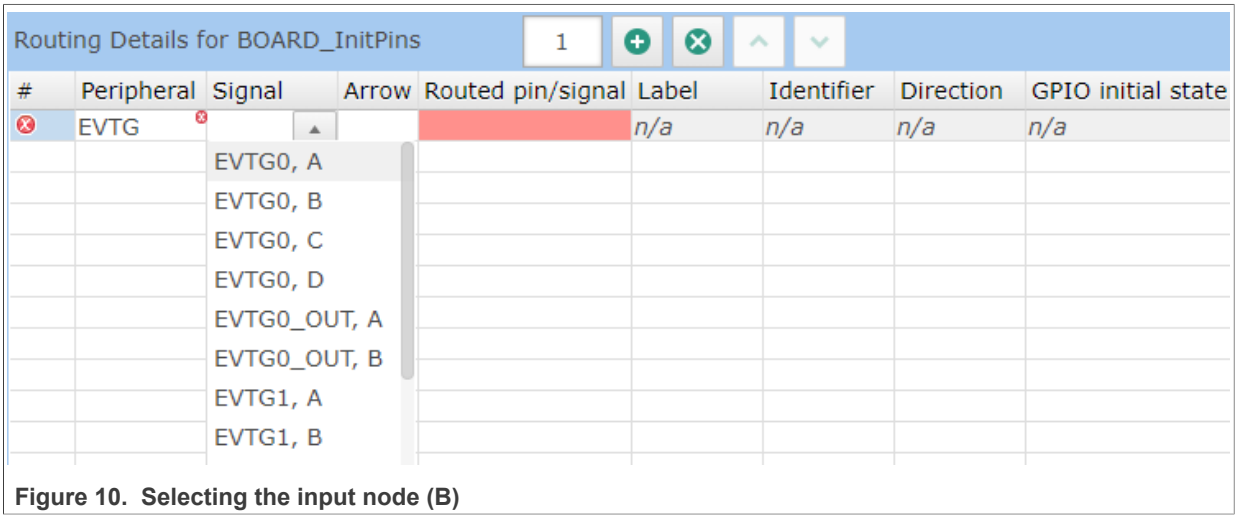


Figure 10. Selecting the input node (B)

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

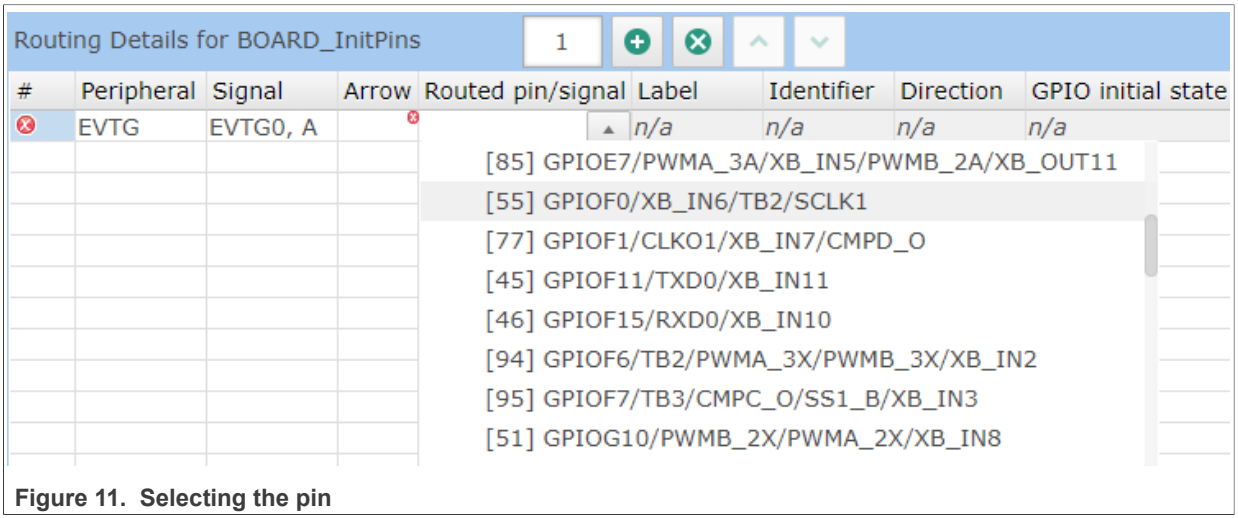


Figure 11. Selecting the pin

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

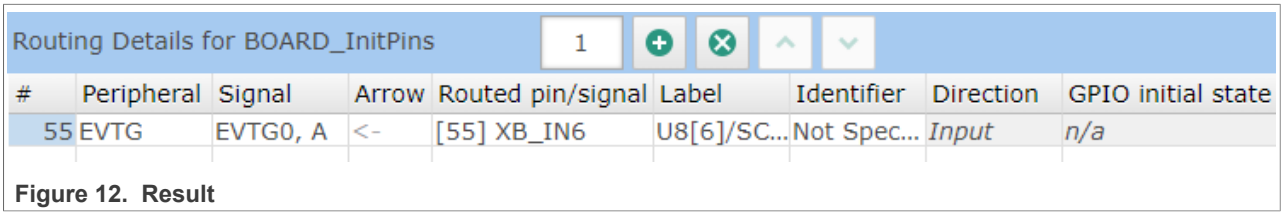


Figure 12. Result

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

3. Routing the signal from internal peripheral (A) output to a pin via inter-peripheral crossbar
Note: Only if a crossbar switch is present.
The signal leads from an internal peripheral (A) output to a pin connected through an inter-peripheral crossbar on the package (XB_OUT). In other words, the signal leads from A to XB_OUT (A > XB_OUT).

To configure a signal in this way, perform the following steps (routing EVTG0 output to a pin 87 using XB_OUT4 used as an example):

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

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

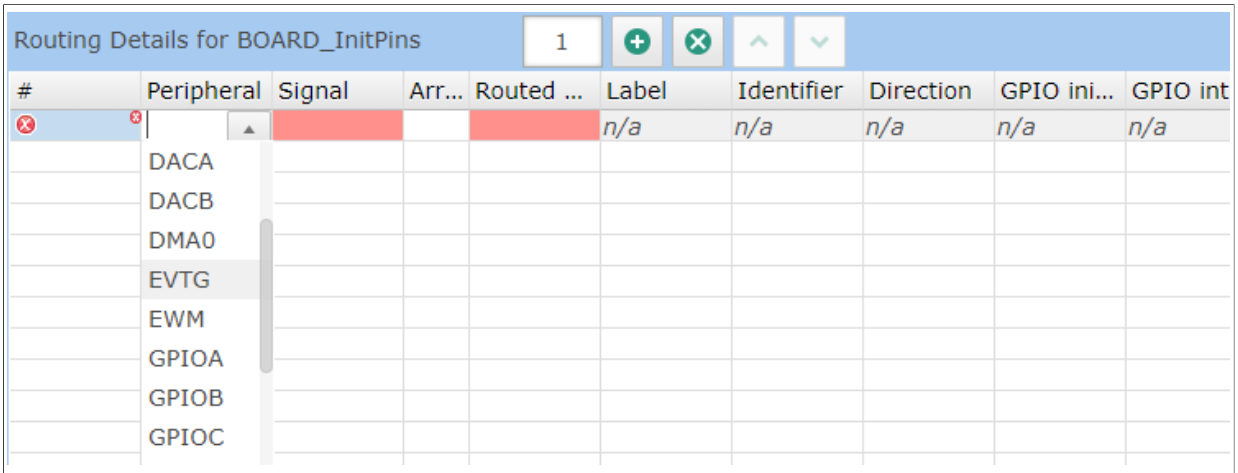


Figure 13. Selecting the peripheral (A)

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

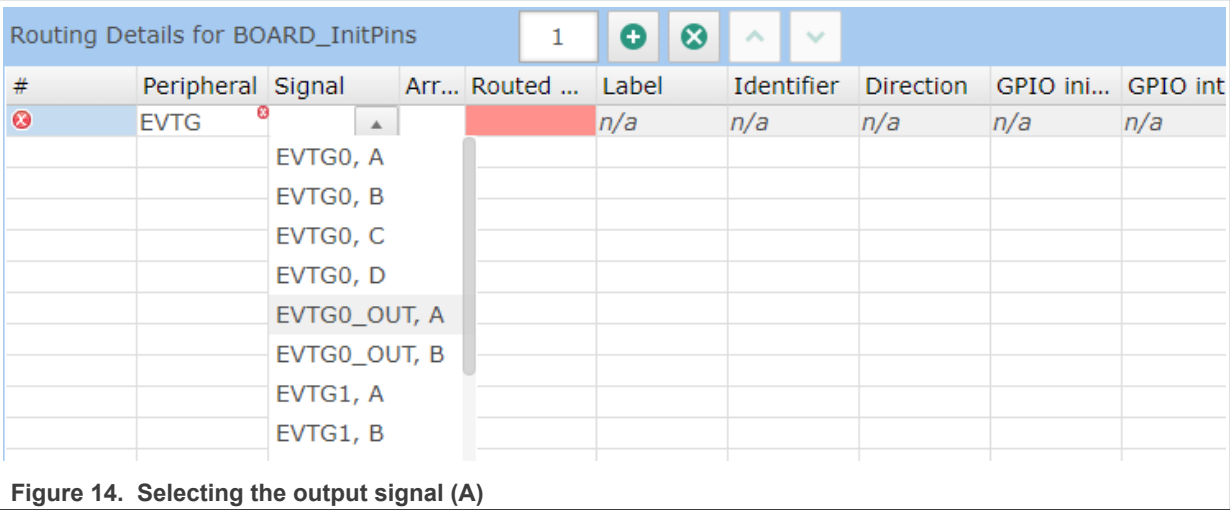


Figure 14. Selecting the output signal (A)

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

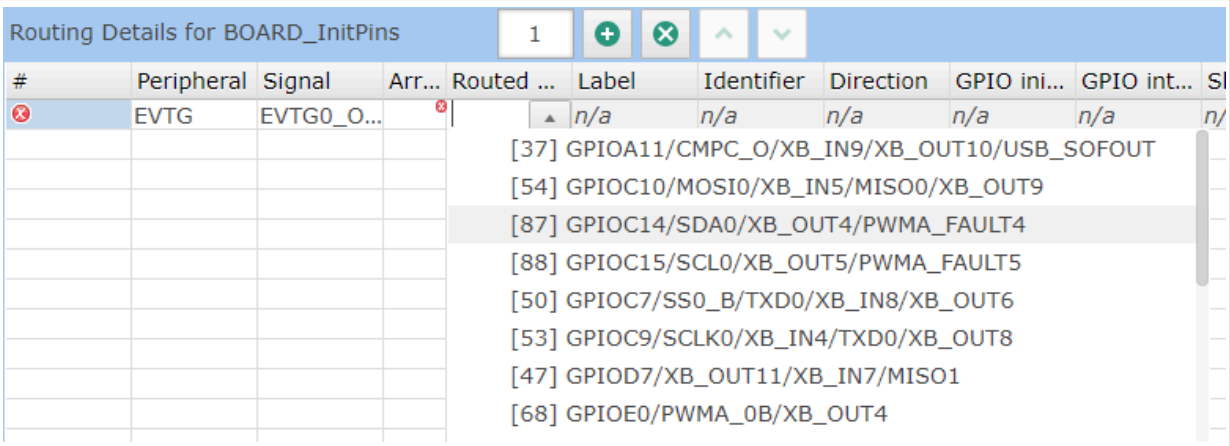


Figure 15. Selecting the pin

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

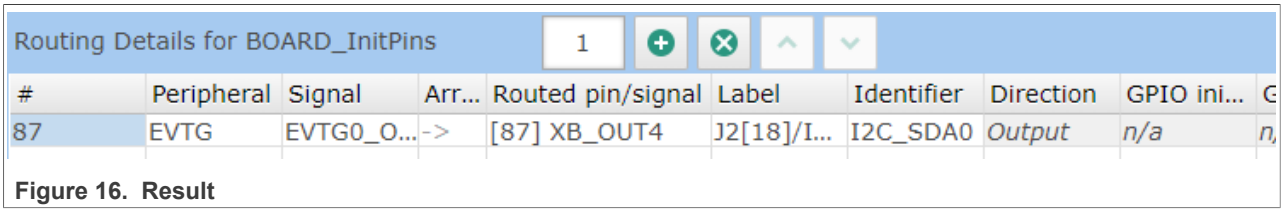


Figure 16. Result

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

Note: In some cases, multiple routings are configured by the same register change. When such routing is created, the user is offered an option to add the related routing to the functional group. Similarly, when a routing is removed, related routings are offered for removal.

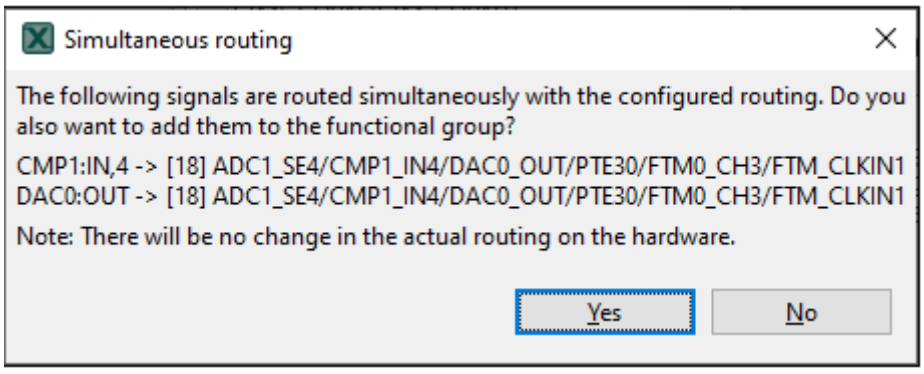


Figure 17. Simultaneous routing

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 the signal cell.
   Note: This routing configuration can be similarly done in other views **Peripheral Signals**, **Package, Routing Details**.
2. Optionally, configure the electrical properties in the **Routing Details** view by selecting the required state.

Note: Source code is automatically generated.

3. To open the **Code Preview** view and inspect the source code, select **Views > Code Preview**.
4. To export the source code, select **Pins > Export** from the **Menu bar**.

3.3 Example workflow

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

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

You can use the generated files with the application code.

1. In the **Pins** view, select the **UART3_RX** and **TX** signals. To do so, click the cells to make them green.

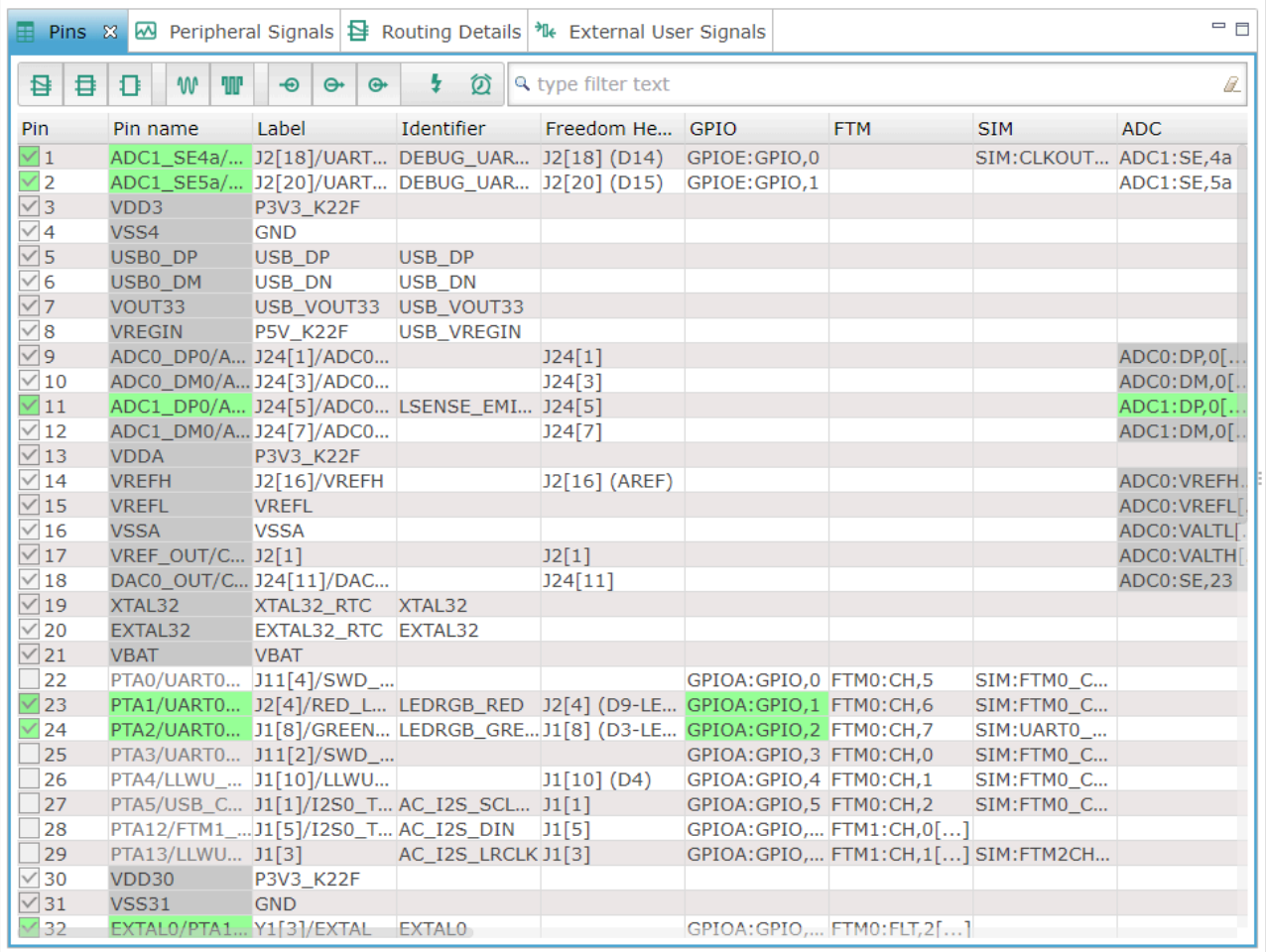


Figure 18. Configuring signals in the Pins view

2. In the **Routing Details** view, select the **Output** direction for the TX and PTB20 signals.
   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`.
4. To open the **Code Preview** view, click the **Views > Code Preview** menu.

5. You can now copy-paste the content of the source(s) to your application and IDE. Alternatively, you can export the generated files or update the code with the **Update Code** button in **Toolbar**. To export the files, select **File > Export**. In **Export**, select the **Export Source Files** option.
6. 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).
7. Click **Finish** to export the files.
8. Integrate and use the exported files in your application as source files.

3.4 User interface

The Pins tool consists of several views.

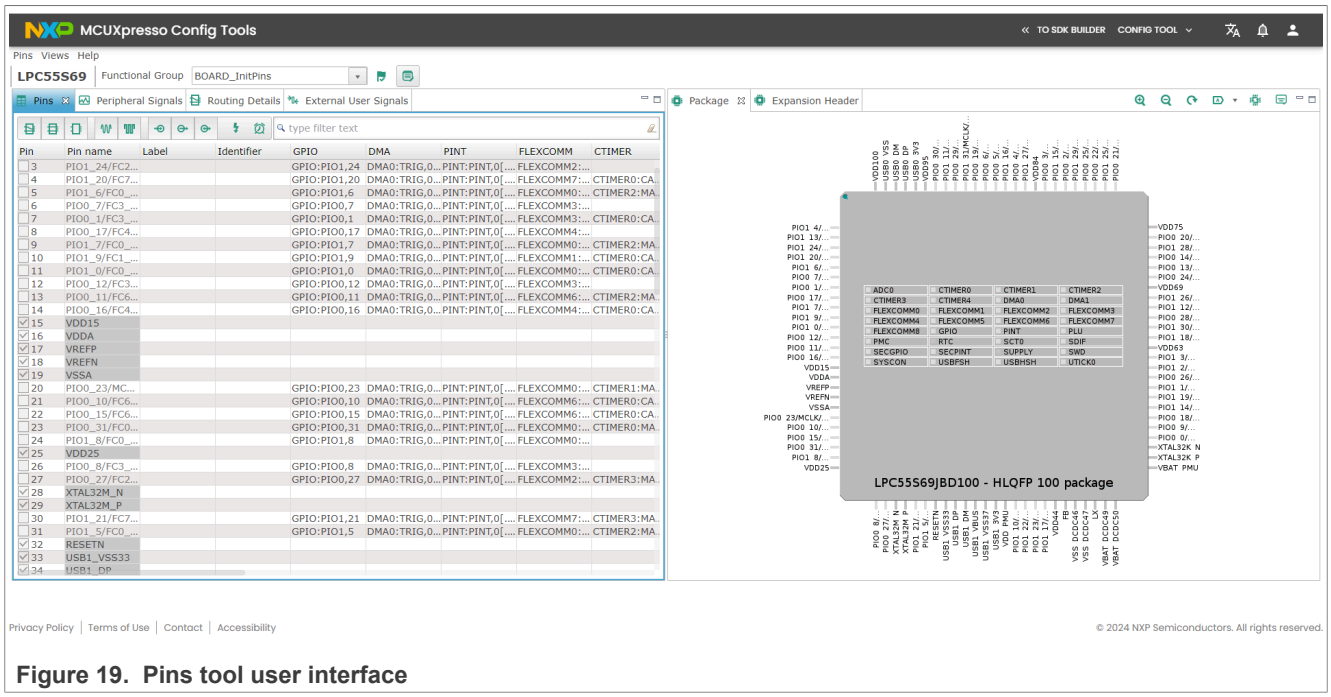


Figure 19. Pins tool user interface

3.4.1 Pins view

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

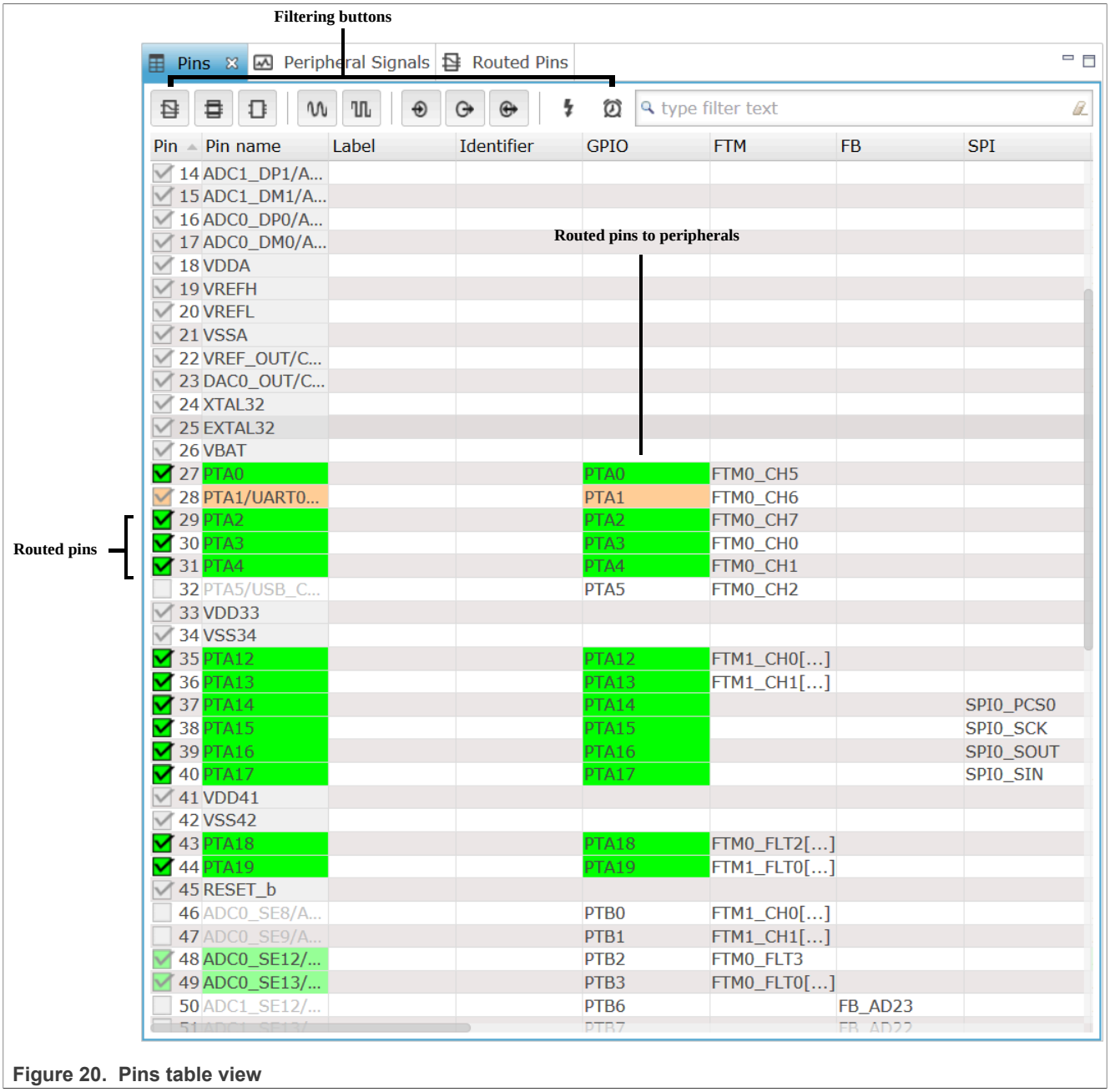


Figure 20. Pins table view

This view shows the list of all the pins available on a given device. The **Pin name** column shows the default name of the pin, or if the pin is routed. The next columns are optional. They are **Label**, **Identifier, External User Signals** and **Expansion header connections** (One column for each expansion header). The pin name is changed to show the appropriate function for the selected peripheral if routed. The next column of the table shows peripherals and signals and pin name(s) on a given peripheral. Peripherals with a few items are cumulated in the last column.

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

Gray background color in the Pin name column indicates that the pin/peripheral is dedicated. It is routed by default and has no impact on the generated code.

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

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

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

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

Tip: An ellipsis (…) indicates that more signals can be routed to a single pin. Select the cell to open a dialog and choose from multiple available signals. The dialog also displays which signals are routed by default.

3.4.2 Package

The **Package** view displays the processor package, providing an overview of resource allocation.

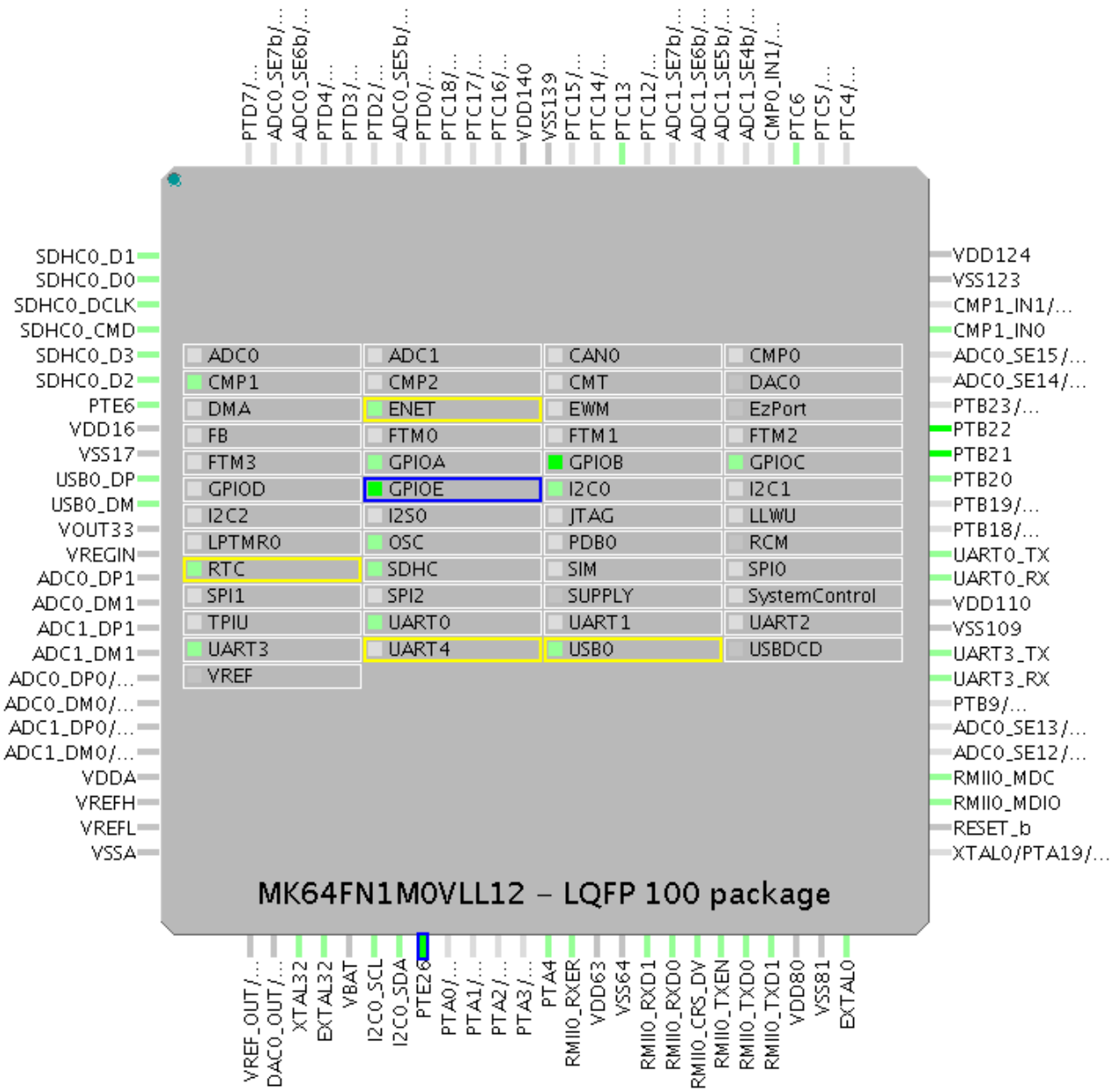


Figure 21. Processor package

This view shows the package overview with pin locations. The peripherals appear in the center.

To highlight the pin/peripheral configuration in the **Pins** and **Routing Details** views, right-click the pin or peripheral and select **Highlight**.

For BGA packages, use the **Resources** icon to see them.

- Green color indicates the routed pins/peripherals.
- Gray color indicates that the pin/peripheral is not routed.
- Dark gray color indicates that the pin/peripheral is dedicated, 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 5. Toolbar options

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

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

The **Switch package** icon opens the **Switch package for the Processor** window, which lists available processor packages with their package type and number of pins.

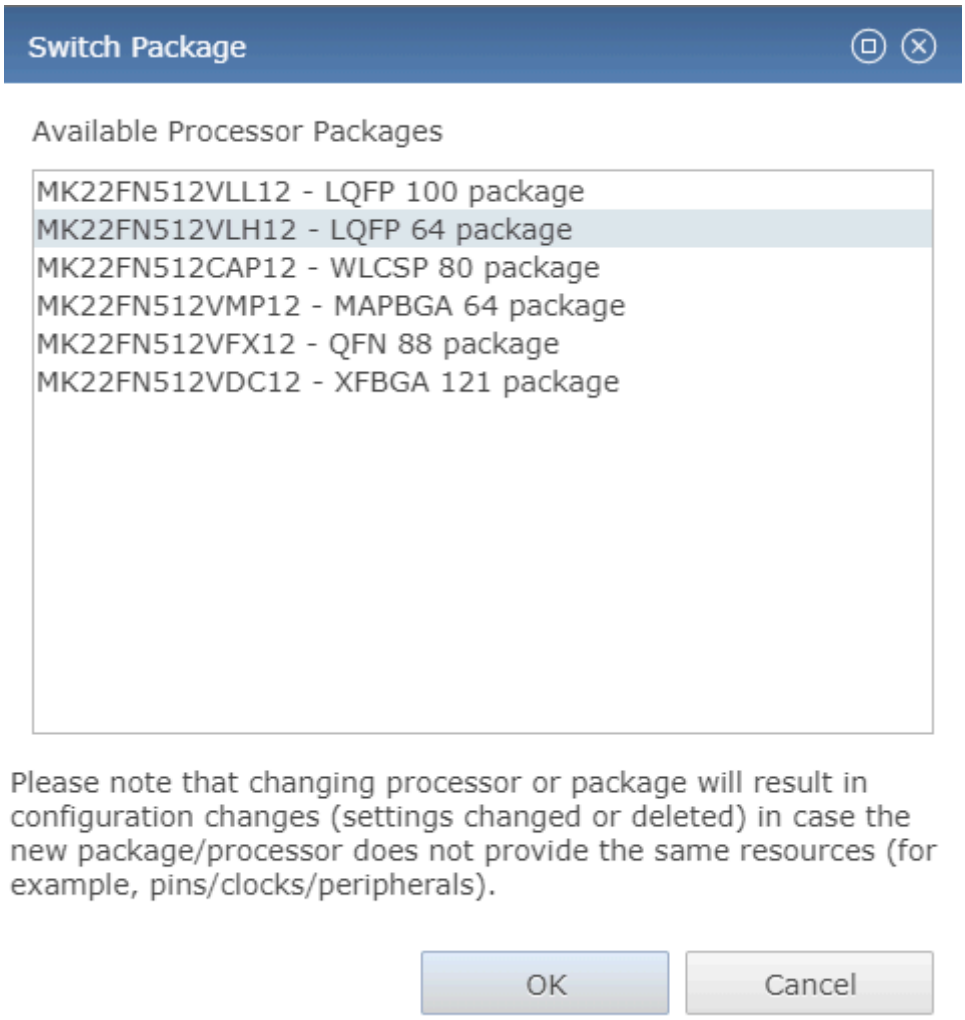


Figure 22. Switch package

3.4.3 Peripheral Signals view

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

Table 6. Status codes

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

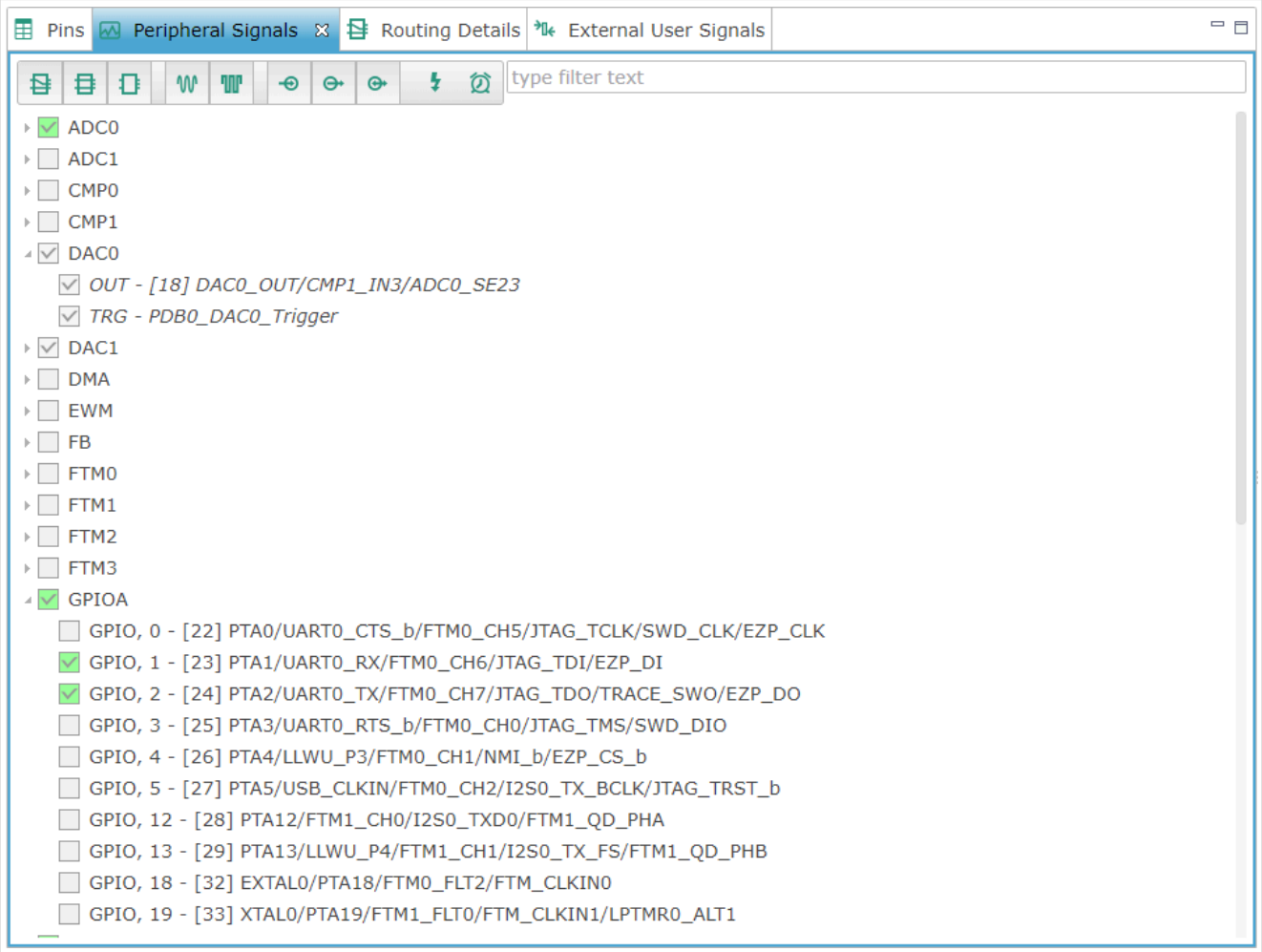


Figure 23. Peripheral Signals view

Use the checkbox to route/unroute the pins.

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

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

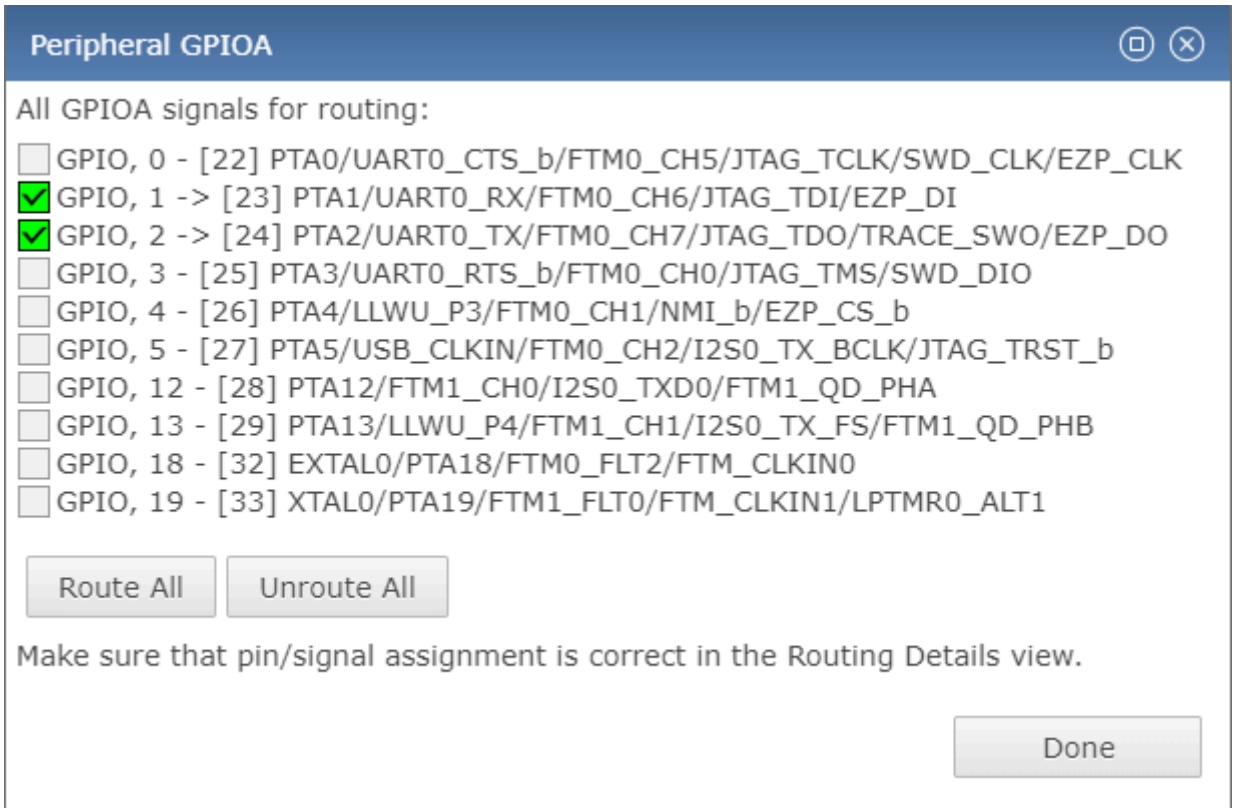
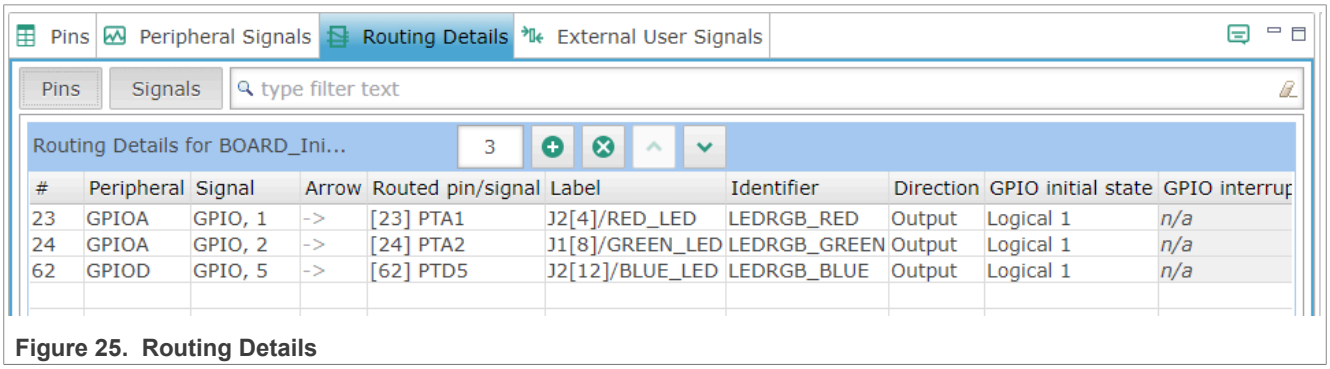


Figure 24. Select signals dialog

3.4.4 Routing Details view

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

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



#	Peripheral	Signal	Arrow	Routed pin/signal	Label	Identifier	Direction	GPIO initial state	GPIO interrup
23	GPIOA	GPIO, 1	->	[23] PTA1	J2[4]/RED_LED	LEDRGB_RED	Output	Logical 1	*n/a*
24	GPIOA	GPIO, 2	->	[24] PTA2	J1[8]/GREEN_LED	LEDRGB_GREEN	Output	Logical 1	*n/a*
62	GPIOD	GPIO, 5	->	[62] PTD5	J2[12]/BLUE_LED	LEDRGB_BLUE	Output	Logical 1	*n/a*

Figure 25. Routing Details

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

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

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

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

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

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

Note: When you enter the search text, it also searches the text in the full pin names display rows that contain the search text.

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

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

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

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

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

To clear the table, type 0.

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

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

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

The gray background indicates read-only items.

3.4.4.1 Properties configured in Routing Details view

The properties of pins and internal signals that can be configured are shown in dedicated columns. The properties include:

- Common properties available for all pins and internal signals
  - **#** - Package pin number/coordinate
  - **Peripheral** - Name of the selected peripheral module
  - **Signal** - Name of the selected peripheral signal/signal function
  - **Arrow** - Arrow indicating input, output, input/output, or not specified direction of the signal
  - **Routed pin/signal** - Name of the pin or internal signal
  - **Label** - Pin label with a maximum length of 128 characters; submitting an empty label also deletes the identifier
  - **Identifier** - Pin identifier used for #define code generation
  - **Direction** - Pin direction
- General-Purpose Input Output (GPIO) properties available only for GPIO pins
  - **GPIO initial state** - GPIO output initial state. It is available only if pin direction is set to output.
  - **GPIO interrupt** - Configuration of interrupt request for the pin. It is available only if the pin direction is set to input.
- Processor-specific properties
  - Properties that may differ for different processors, for example configuration of pull ups, open drain, drive strength, slew rate, power groups

Tip:

- Click the **Routing Details Legend** button in the top-right corner of the view to display a dialog explaining all the properties and their values.

Generation of initialization code

The Pins tool generates initialization code only for pins and internal signals configured in the Routing Details view. Generally, initialization code is generated if it is necessary for proper routing of the pin or internal signal to the selected peripheral signal. On the other hand, the code related to the direction, GPIO functionality, or processor-specific properties is automatically optimized out in the following cases:

- The user does not explicitly select a value of a property. This is indicated by the italic font of the value. It is the default state after adding a pin or internal signal to the Routing Details view. The displayed value shows either the value set by another configuration of the same pin in the same function or in a function called from the default initialization function. When there is no such configuration, the displayed value matches the after-reset state. To generate the code even if the value is the same as the default value, select the value from the drop-down menu: the value is shown with normal font and code is generated. To return to the default value, select “No init” from the drop-down menu: the value is again shown with italic font and no code is generated.
- A “Not specified” value is selected in case of the direction property.

Additionally, it is possible to force generation of full initialization code of all properties using “Full pins initialization option” in Functional groups. If this option is enabled, it is not possible to use the “No init” value from the drop-down menu and the initialization code is generated unless the “Not specified” value is selected (for direction only). For more information about the “Full pins initialization option” option, refer to the Functional group properties section in this documentation.

Validation of routing

The Pins tool automatically performs validation of the selected properties. An error is shown for a property in the Routing Details view in the following cases:

- The value of the property is in conflict with the value of another property. Typically, conflicts arise when two pins or internal signals configure the same register and bitfield with different initialization values. To resolve the conflict, the configuration of one of the pins or internal signals must be changed.
- There is no after-reset value specified for the property. In this case, the error indicates that the user must select the value of the property or “Not specified” to indicate that the tool can ignore the property.
- The property must be set by the user. In this case, “No init” and “Not specified” are not available and the user must decide what value to use. A typical use case is when routing of the internal signal would not be complete without such selection or when the user must confirm the intended usage of the pin or signal.

3.4.4.2 Labels and identifiers

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

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

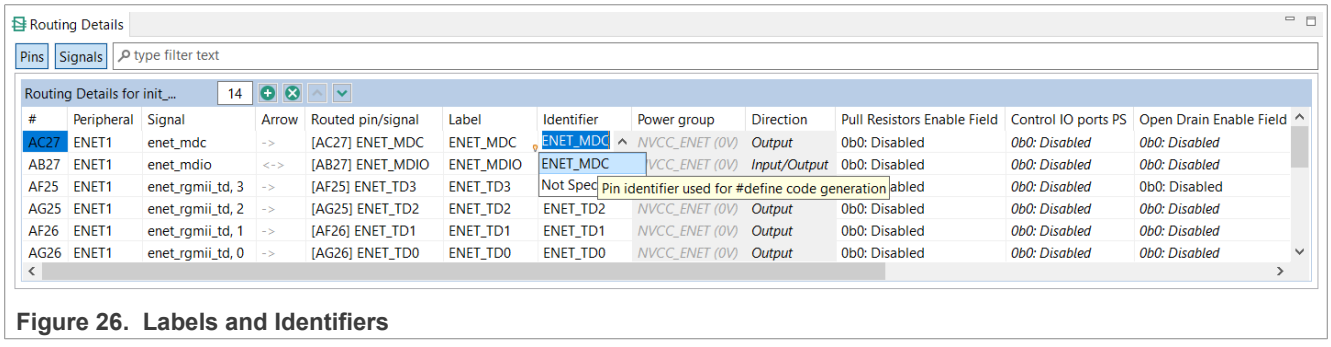


Figure 26. Labels and Identifiers

The pin identifier is used to generate the `#define` in the pin_mux.h file. However, it is an optional parameter. If the parameter is not defined, the code for `#define` is not generated. Also, you can define multiple identifiers,

using the ";" character as a separator. You can also set the identifier by typing it directly into the cell in the **Identifier** column in the **Routing Details** views.

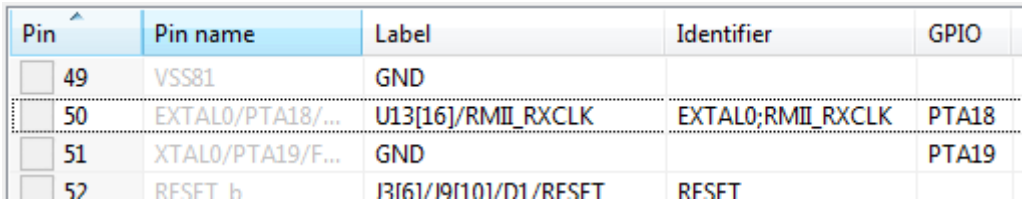


Figure 27. Pin identifier

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

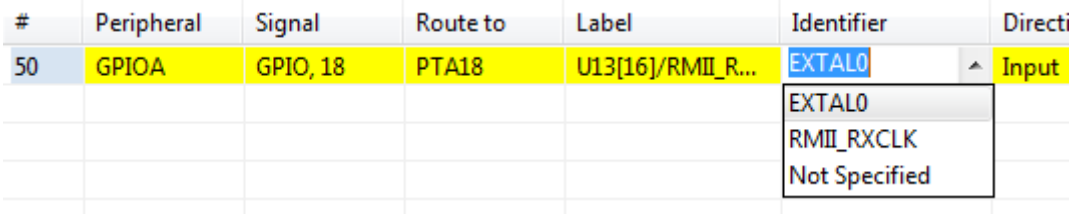


Figure 28. Identifier in Routing Details table

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

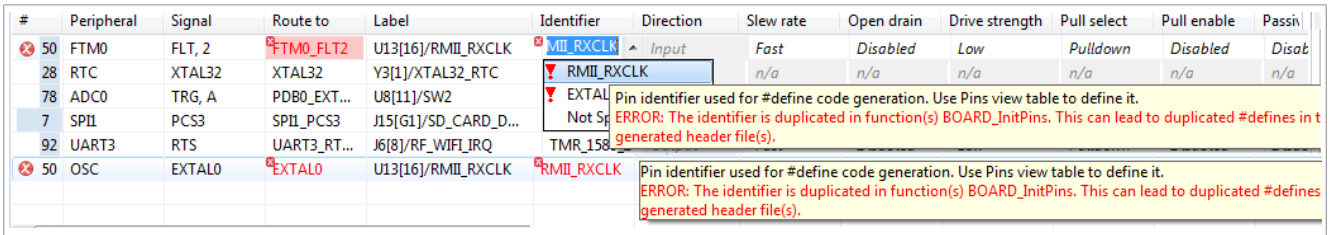


Figure 29. Identifier errors

By typing a text into the Identifier column, you can define a new identifier for the pin. It is automatically used only in the selected routing. Available identifiers can be revised in a dialog invoked from a context menu or in the Pins view.

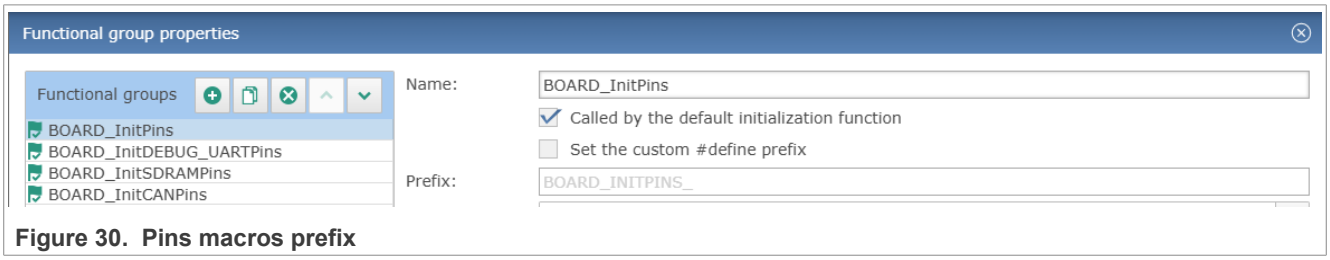


Figure 30. Pins macros prefix

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

3.4.5 Expansion Header

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

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

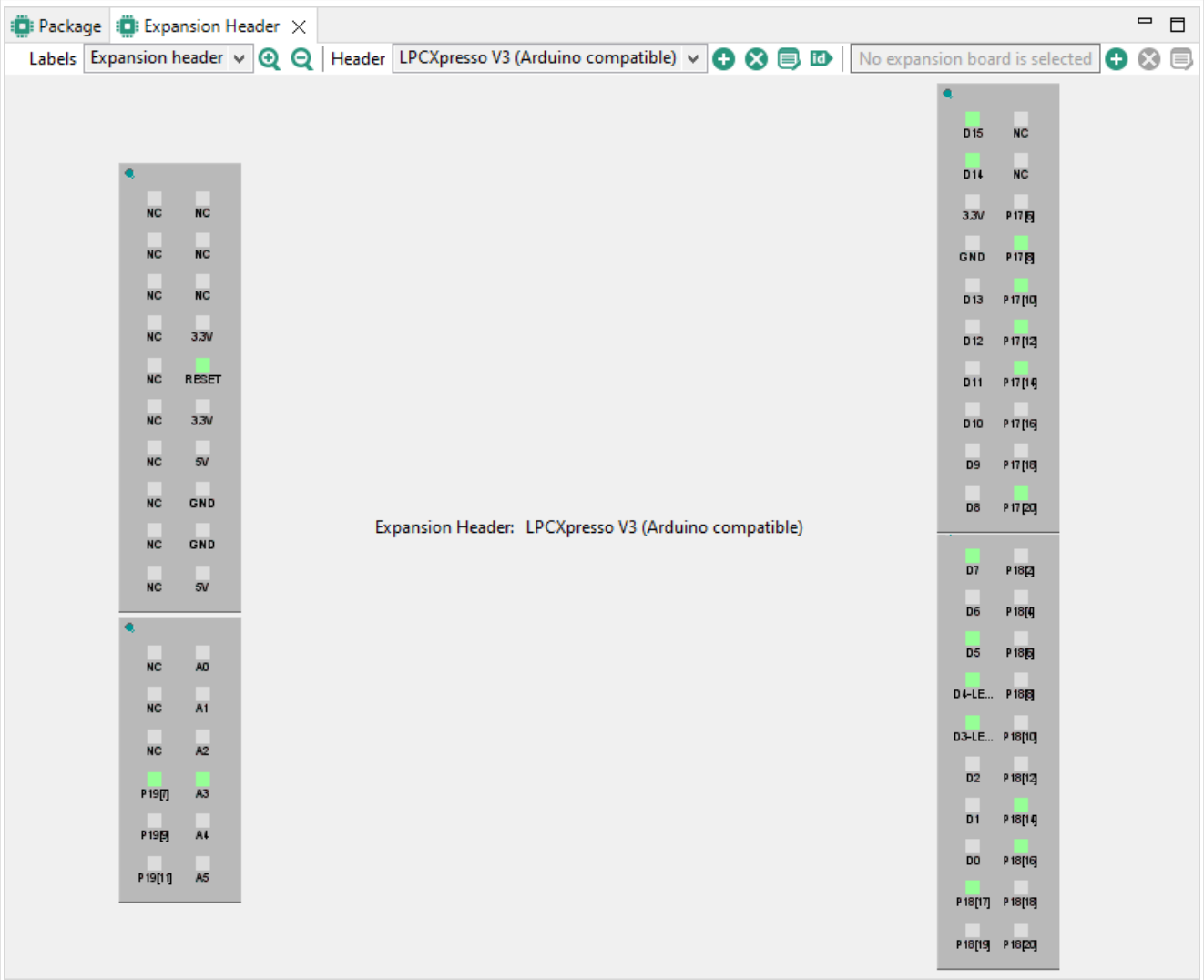


Figure 31. Expansion header

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

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

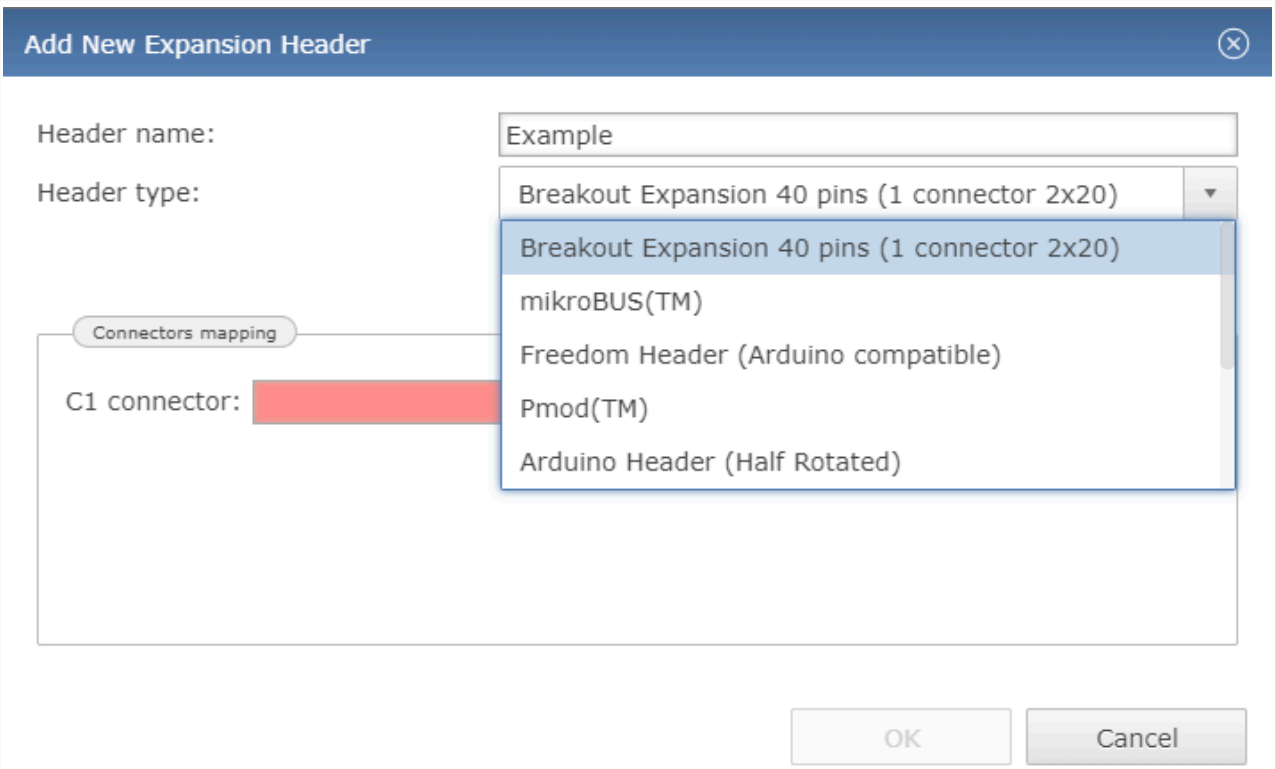


Figure 32. Adding new expansion header

5. Name the header and map the connectors.

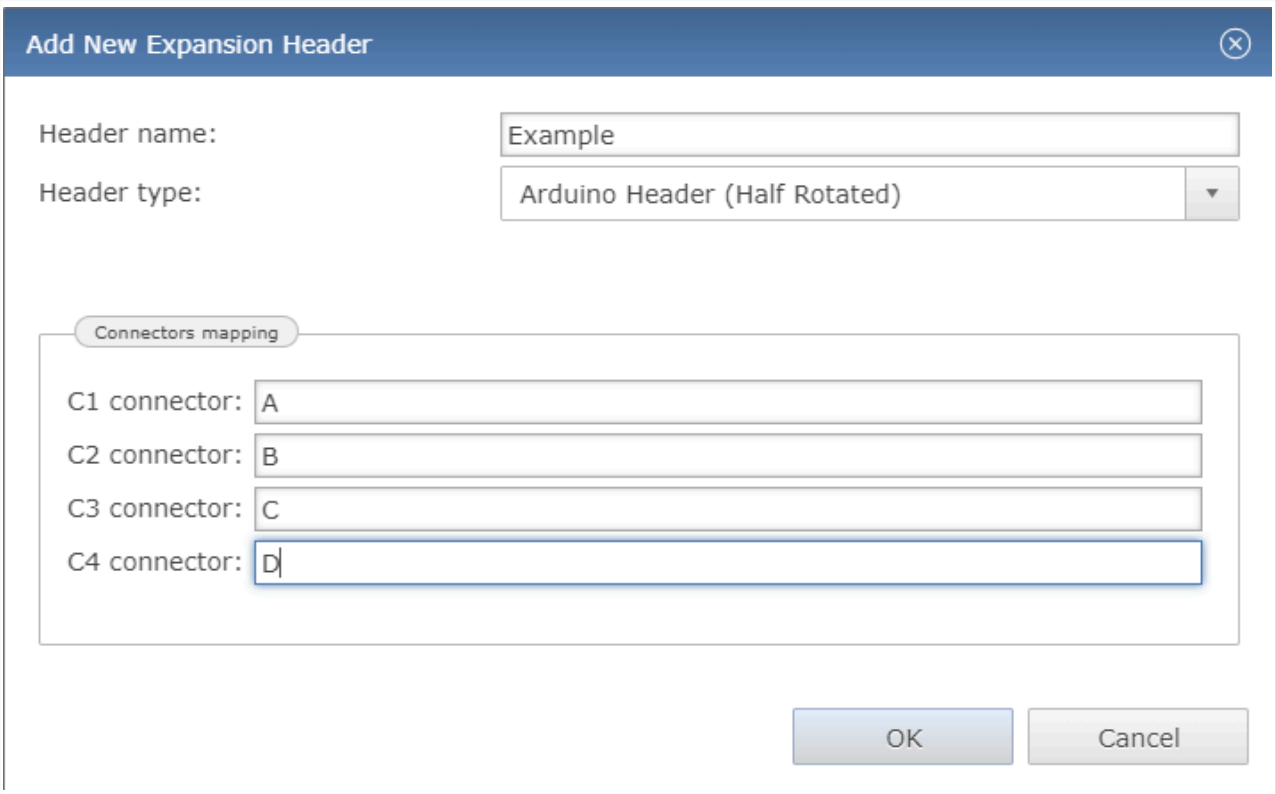


Figure 33. Adding new expansion header

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

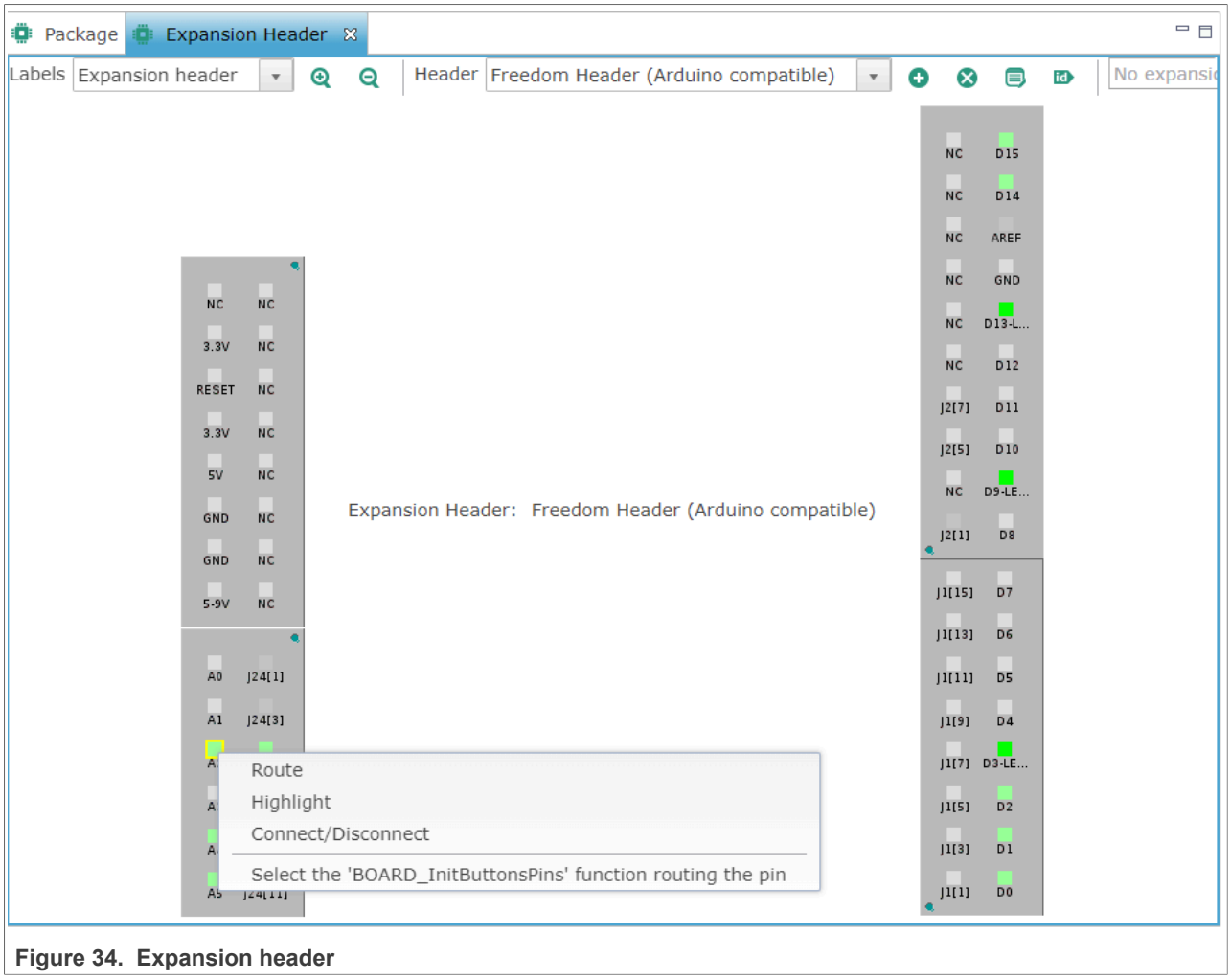


Figure 34. Expansion header

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

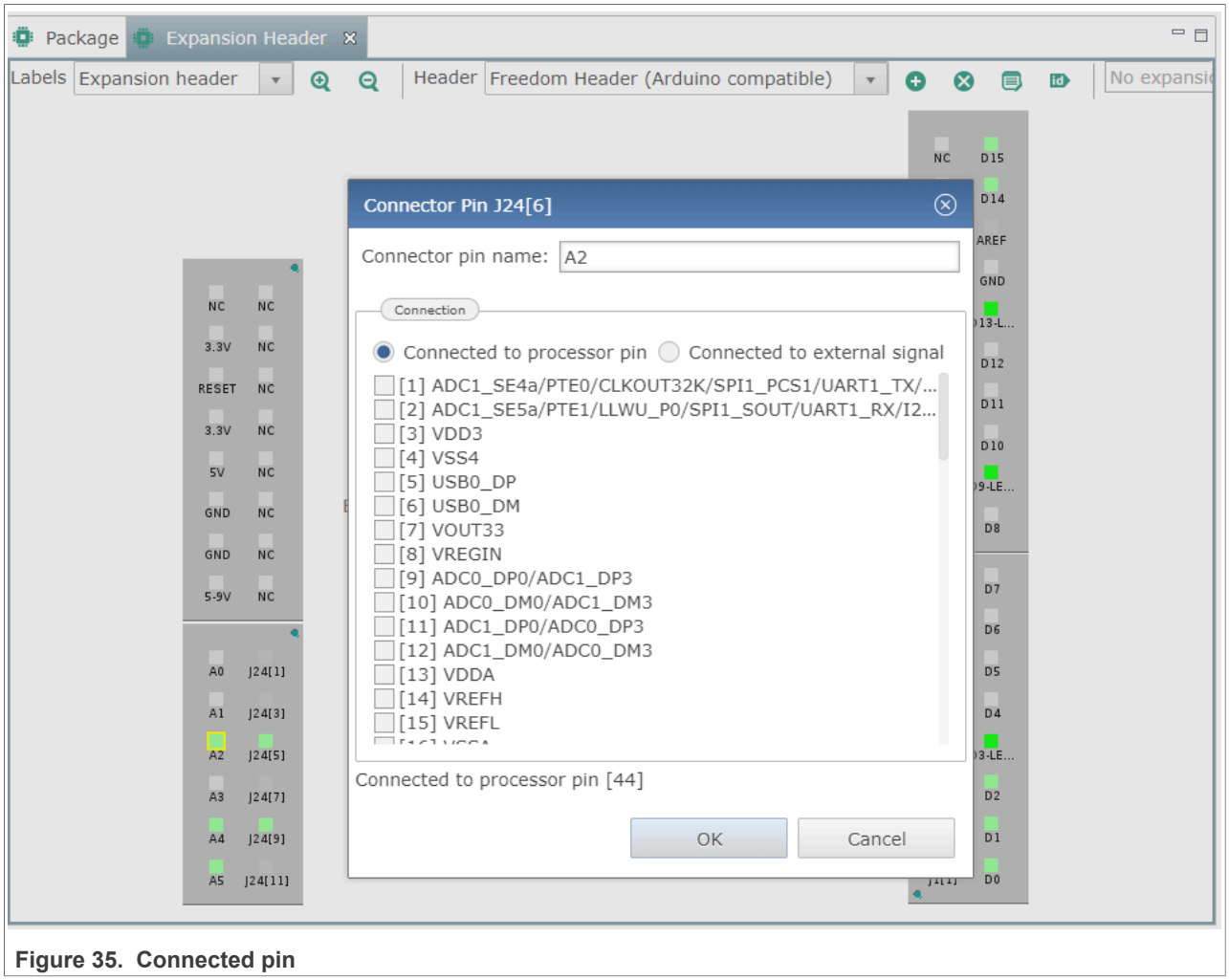


Figure 35. Connected pin

9. To route the pin, right-click the header pin and select **Route**.
10. In the **Pin** dialog, select the signal from the list and click **OK**.
    The connector pin is now routed.

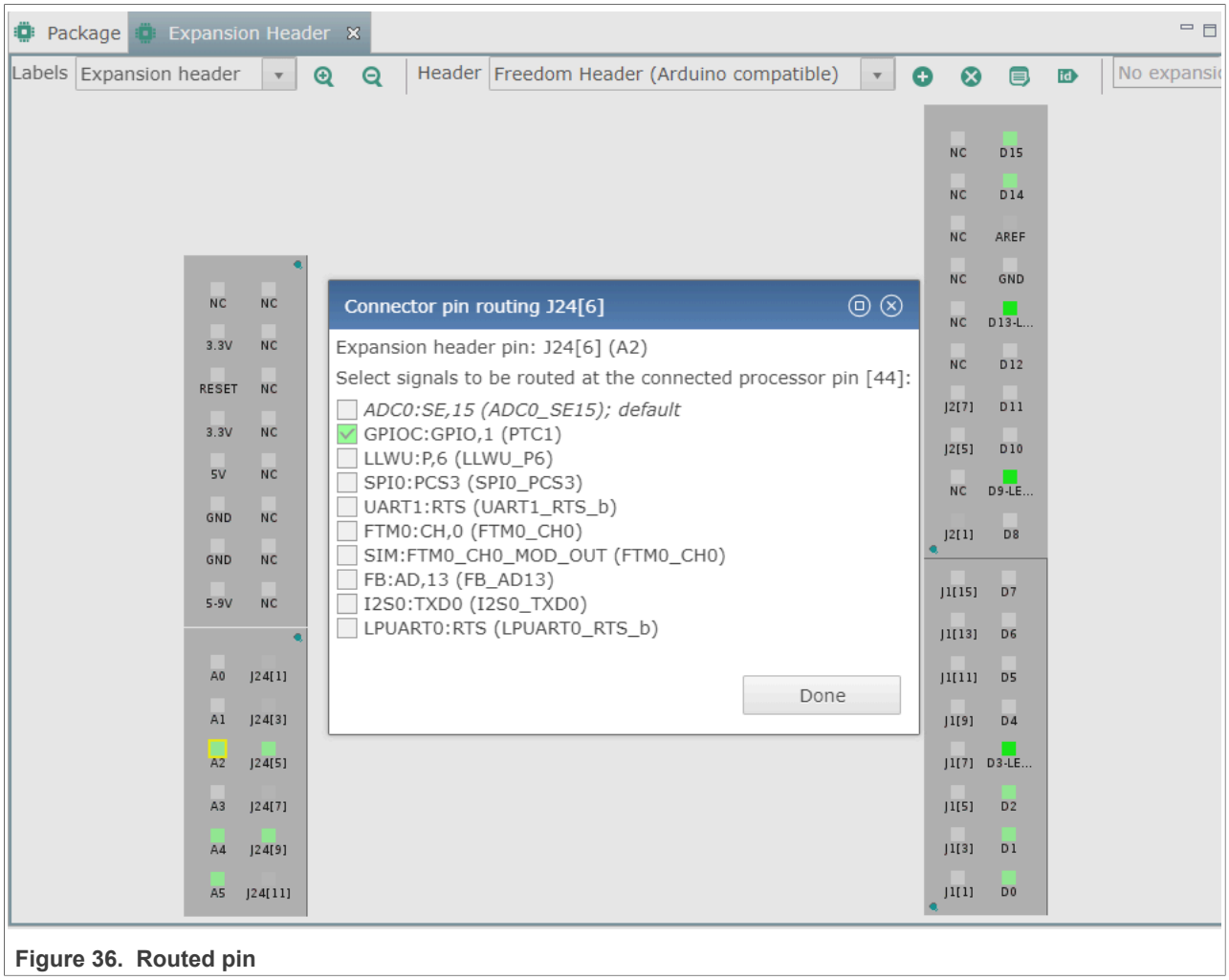


Figure 36. Routed pin

You can create more than one expansion header configuration. Switch between the configurations in the view's drop-down list.
To highlight the pin/routing configuration in the **Pins** and **Routing Details** views, right-click the connector pin and select **Highlight**.
Modify the configuration parameters at any time by selecting the **Edit** button. Information in the **Pins** view is updated automatically. Pin connections between the header and the processor and their labels can be locked to prevent any modifications.
Remove a configuration by selecting the **Remove** button.
Use the **Label** drop-down list to switch between display information for header, board, and routing.
The **ID** button allows you to set expansion header pin labels as processor pin identifiers. Upon user selection, the new identifiers can also be explicitly selected.

3.4.5.1 Expansion Board

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

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

1. In the **Expansion Header** view, click the **Apply expansion board to the selected header**. Alternatively, select **Pins>Apply expansion board** from the **Menu bar**.

2. In the **Apply expansion board** dialog, click **Browse** to locate the XML file with expansion board information and click **OK**.

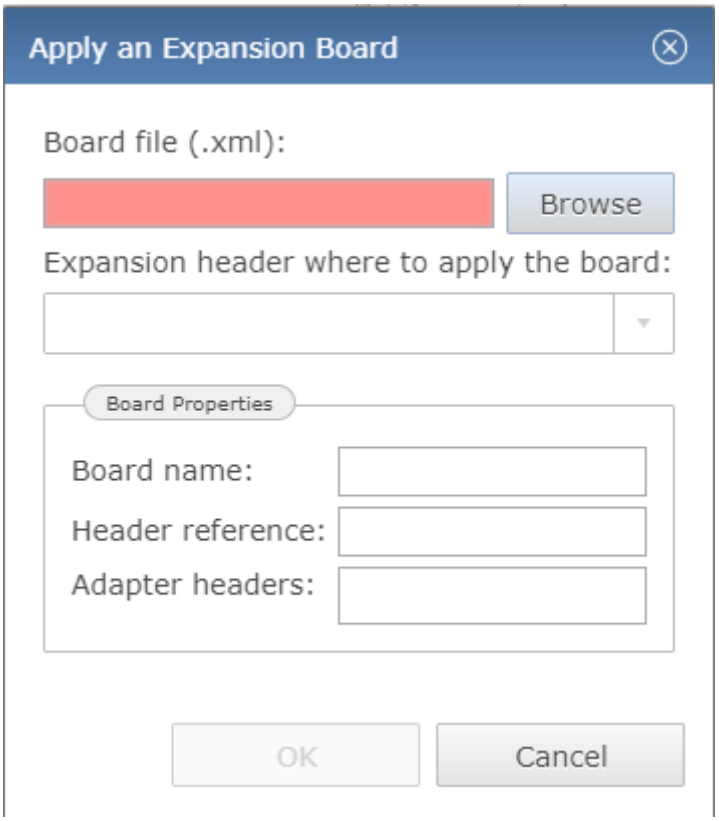


Figure 37. Apply expansion board

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

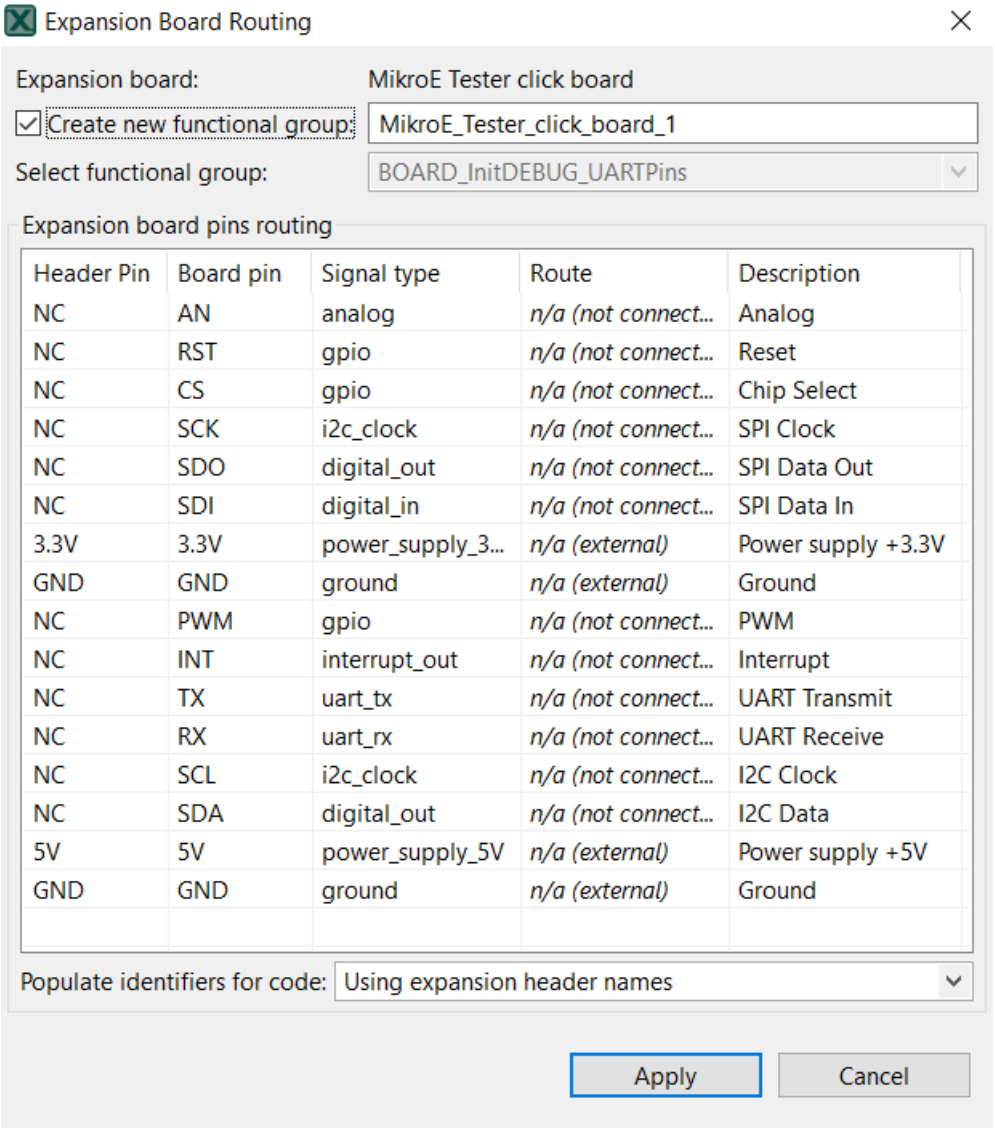


Figure 38. Expansion board routing

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

3.4.6 External User Signals view

This view allows the user to define a custom description of the signals. An External User Signal has a defined unique ID within the table, pins to which it is connected, and any amount of additional text information. All of these properties can be customized.

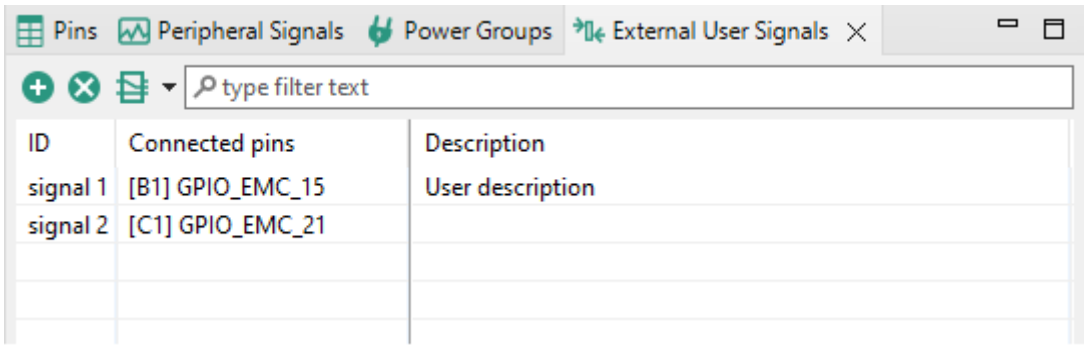


Figure 39. External User Signals view

Connecting to a pin(s) can be done from a context menu of the selected signal. Multiple pins can be connected to the signal, and multiple signals can be connected to the pin. When some signals are defined, the External User Signals column is added to the **Pins** view. The connection between pins and signals can also be done from there.

Additional columns can be specified using the table header context menu.

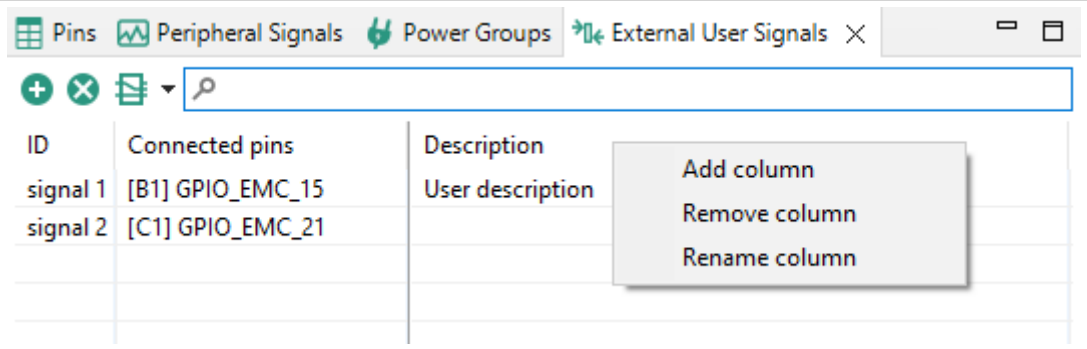


Figure 40. External User Signals header context menu

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

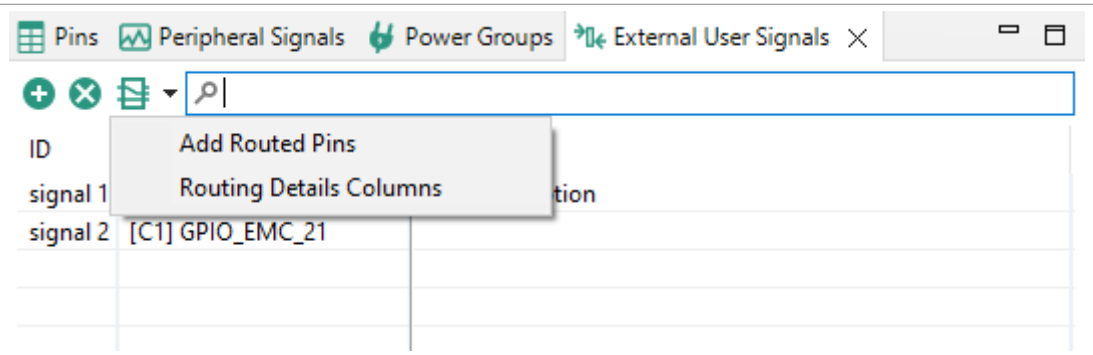


Figure 41. External User Signals view

The **Add Routed Pins** option collects routed pins from all functional groups and adds them to the table when they are not already present. A new signal is created for each added pin.

Select Routing Details Columns to open a dialog with Routing Details columns. The selected columns appear in the table. Those columns are read-only and always reflect actual values in the **Routing Details** view for all functional groups.

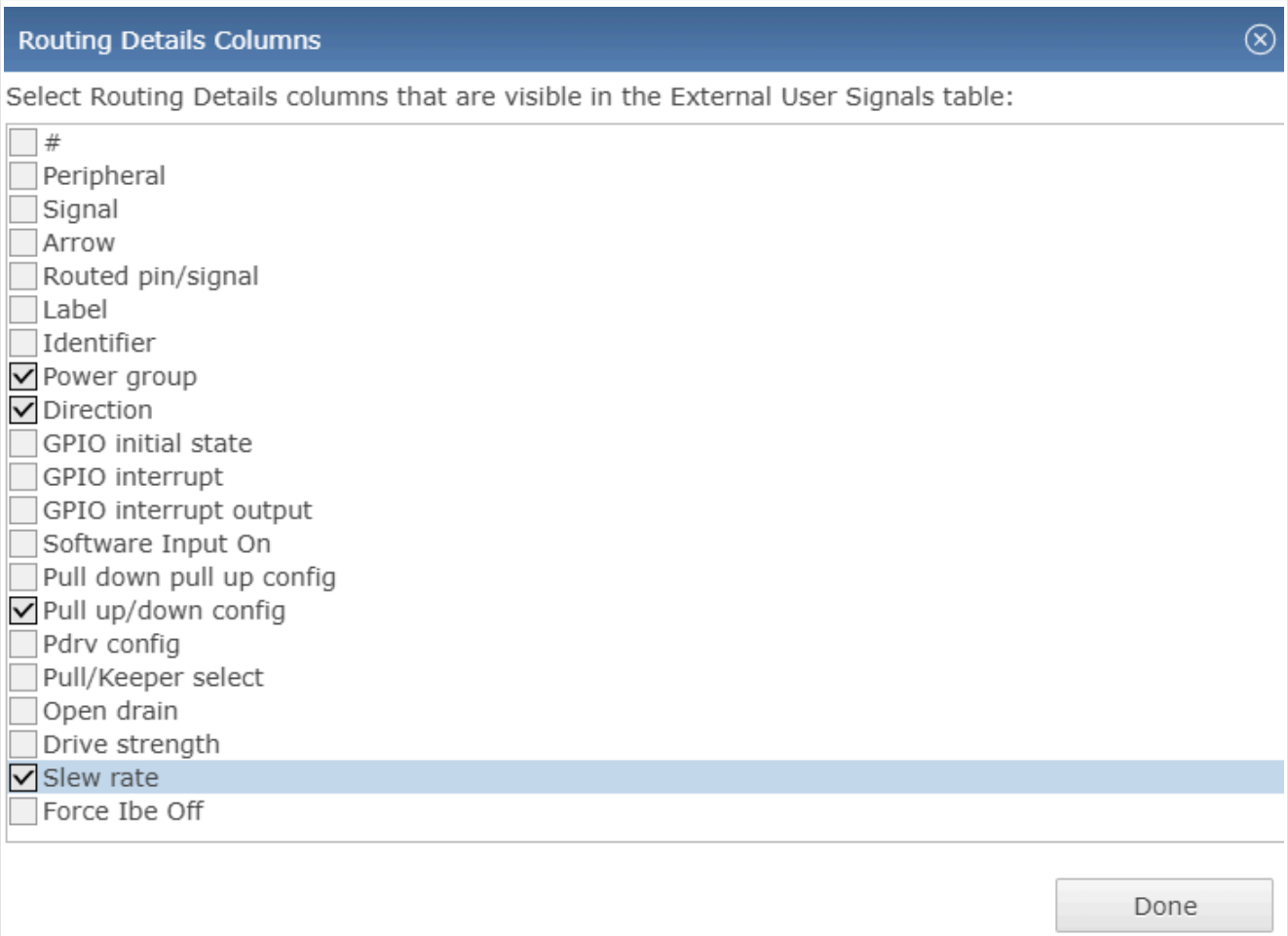


Figure 42. Routing Details columns dialog

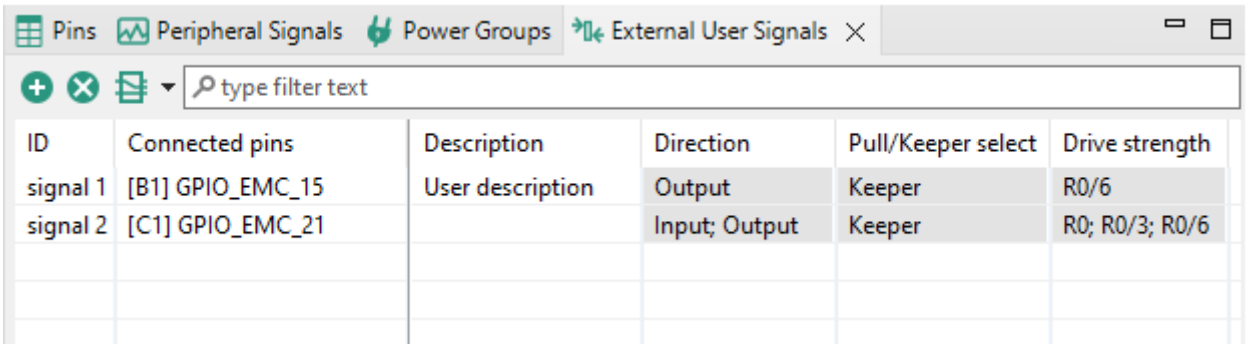


Figure 43. Routing Details table

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

3.4.7 Miscellaneous view

This view contains Generate extended information into the header file (previously located in Configuration preferences) and possible processor specific settings.

When the **Generate extended information into the header file** option is selected, the tool generates extended information into the header file. For projects created in earlier MCUXpresso versions, this option is selected by default.

When the **Automatically select property value for a new routing** option is set, the tool explicitly sets the after-reset state for every electrical property of a new routing.

3.4.8 Functions

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

The tool allows users to create multiple functions that 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.

3.4.9 Highlighting and color coding

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

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

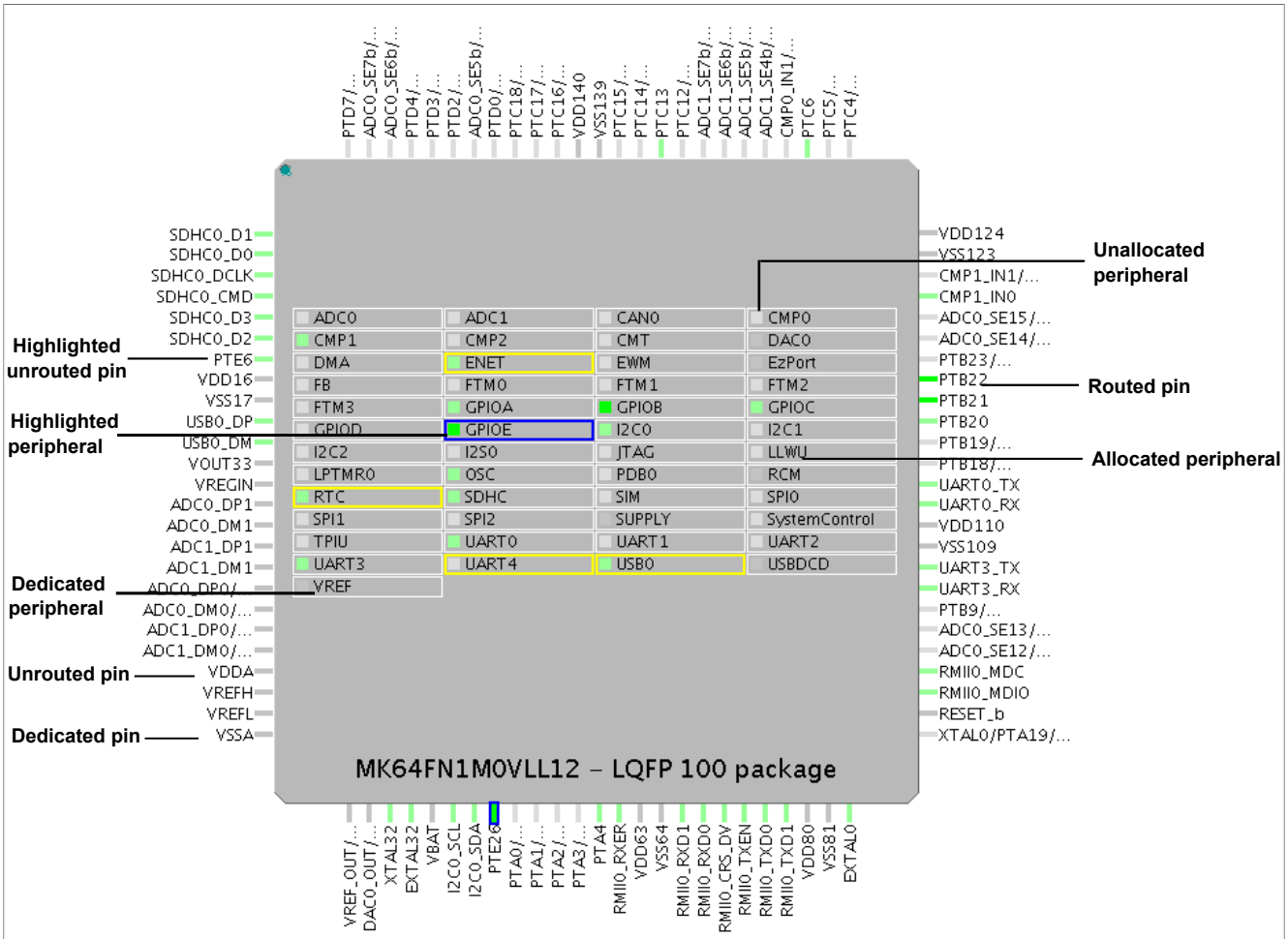


Figure 44. Highlighting and color coding

#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Open drain	Drive strength	Pull select
31	CAN0	TX	ADC0_S...	J2[20]/U8[4]/I2C0_SCL	Not Specified	n/a	Fast	Disabled	Low	Pulldown
81	CMP0	IN, 1	ADC1_S...	J1[9]	n/a	n/a	Fast	Disabled	Low	Pulldown
36	GPIOA	GPIO, 2	PTA2	J1[12]/J9[6]/TRACE_SWO	n/a	Not Specified	Fast	Disabled	High	Pullup
	GPIOE	GPIO, 24			n/a	n/a	n/a	n/a	n/a	n/a
32	GPIOE	GPIO, 25	PTE25	J2[18]/U8[6]/I2C0_SDA	ACCEL_SDA	Not Specified	Fast	Enabled	Low	Pulldown
32	GPIOE	CH, 2	ADC0_S...	J2[18]/U8[6]/I2C0_SDA	Not Specified	n/a	Fast	Disabled	Low	Pulldown
					n/a	n/a	n/a	n/a	n/a	n/a

Figure 45. Pins conflicts

#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Open dr...	Drive str...	Pull select	Pull ena...
33	GPIOE	GPIO, 26	PTE26	J2[1]/D...	Not Specified	Output	Fast	Disabled	Low	Pulldown	Disabled
71	FTM0	CH, 0	FTM0_C...	J1[5]	n/a	Output	Fast	Disabled	Low	Pulldown	Disabled

Figure 46. Warnings

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

– Click the pin or use the pop-up menu to highlight the pin and the peripherals.
– Click outside the package to cancel the highlight.

- **Peripherals / Pins** view
  – The peripheral and pin behave as described above.

3.4.10 Filtering in the Pins and Peripheral Signals views

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

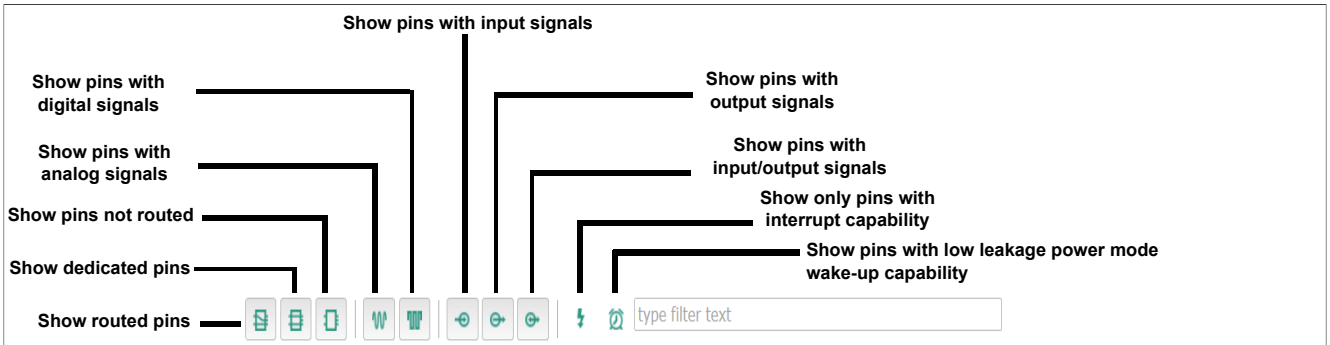


Figure 47. Filtering controls

Type any text to search across the table/tree. The tool searches for pins and peripheral signals containing the specified text. You can also use wildcards “*” and “?” to filter results. Use “space” to search for multiple strings at the same time.

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 across all **INIT** functions (default initialization functions). It is possible to configure different routing of one pin in different functions (not INIT functions) to allow dynamic pins routing reconfiguration. If an error or warning is encountered, the conflict in the **Routing Details** view is represented in the first column of the row and the error/warning is indicated in the cell, where the conflict was created. The detailed error/warning message appears as a tooltip.

For more information on error and warning colors, 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 the drop-down arrow and select the suitable value. The tooltip of the cell shows more details about the conflict or the error, typically it lists the lines where conflict occurs.

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

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

3.5.2 Power groups voltage level conflicts

The Pins tool provides information about possible voltage level conflicts when the peripheral signals routed pins are configured from a different power groups and the power groups have different voltage level value set in **Power groups** view.

When a potential voltage level conflict is detected, the tool displays a warning — a useful feature for hardware board designers.

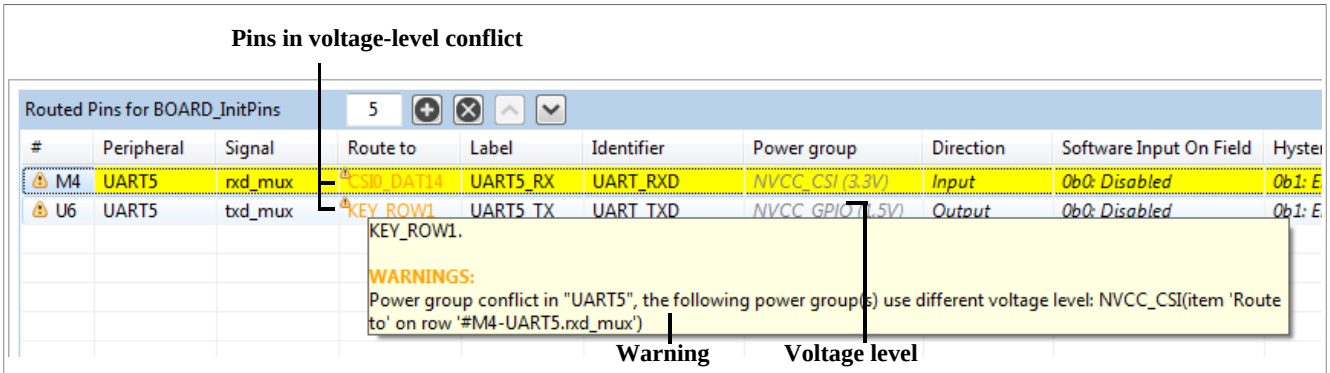


Figure 48. Power group view

3.6 Code generation

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

Note: Code Preview view is not opened by default. To open this view, click **Views > Code Preview**.

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

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

3.6.1 Exporting sources

It is possible to export the generated source using the Export wizard.

To launch the Export wizard:

1. Select **File > Export** from the **Menu bar**.
2. Select **Export Source Files**.
3. Click **Next**.
4. Select the target folder where you want to store the generated files.
5. In case of multicore processors, select the cores you want to export.
6. Click **Finish**.

3.7 Using pins definitions in code

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

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

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

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

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

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

3.8 Full initialization of pins

In some cases, the default values are not reliable, as there may be code running before the application that modifies the pin configuration (for example, a bootloader). The option **Full initialization of pins** ensures that the initialization is fully done even for items that use after-reset state. This option is specific for each **Functional group** allowing to force full initialization of routing. **Full initialization of pins** is not enabled by default. When enabled, the electrical properties of existing routing are changed. The values in italics are changed to explicit values corresponding with them. When the option is disabled, the pins tool removes initialization of values that match the “No init” state.

3.9 Create default routing

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

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

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

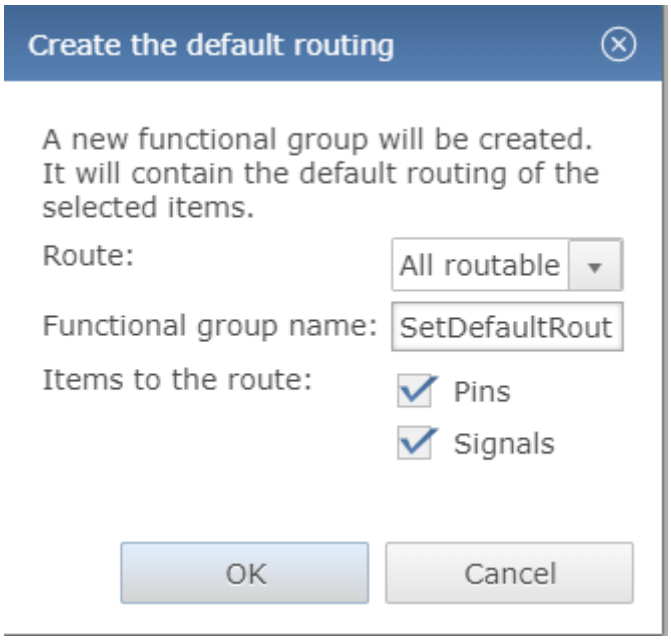


Figure 49. Create the Default Routing

4 Clocks Tool

The web version of Clocks Tool gives a preview of the system clock (core, system, bus, and peripheral clocks) and configuration structures of the clocking environment.

4.1 Features

The Clocks tool supports the following actions related to clock initialization:

- Inspect and modify element configurations on the clock path from the clock source up to the core/peripherals.
- Validate clock elements settings and calculate the resulting output clock frequencies.
- Modify the settings and provide 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.
- Find clock elements settings that fulfill given requirements for outputs.
- Register values define generation of C.

4.2 User interface overview

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

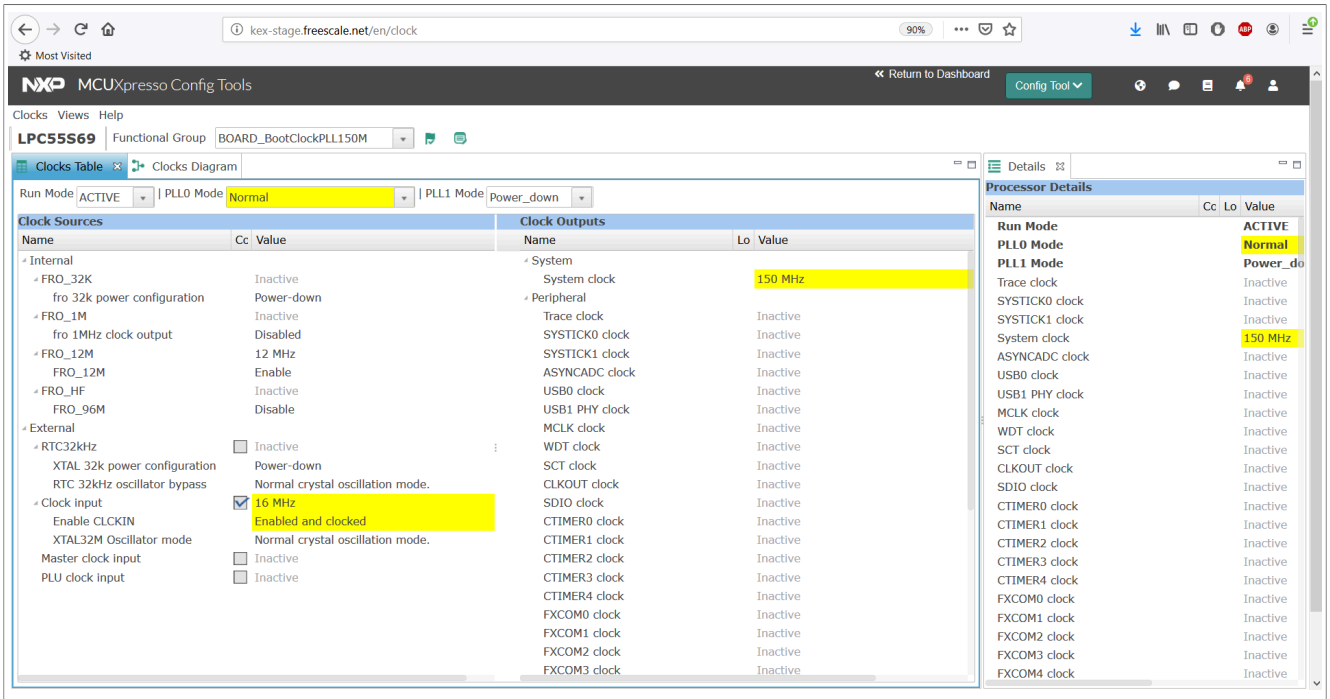


Figure 50. User interface

4.3 Clock configuration

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

The Clocks tool handles one or more clock configurations. The default clock configuration is named "*BOARD_BootClockRUN*". Multiple configurations provide multiple options for 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 51. Default clock configuration

4.4 Global settings

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

Note: Global settings can be changed at any time.

4.5 Clock sources

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

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

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

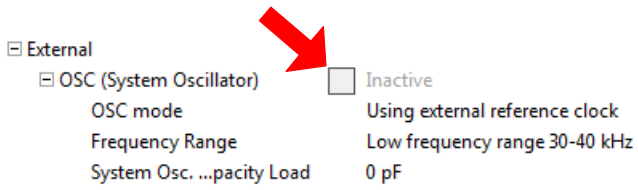


Figure 52. External clock source configuration

Note: Some clock sources remain inactive even though the checkbox is checked. This is because clock source functionality depends on other settings, such as power mode or additional enable/disable options. Hover over 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 setting rows in the settings tables (clock sources, output, details, or individual).

Table 7. Setting states and markers

State/Style/Marker	Icon	Description
Error marker		Indicates that there is an error in the settings or something related to it. See the tooltip of the setting for details.
Warning marker		Indicates that there is a warning in the settings or something related to it. See the tooltip of the setting for details.
Lock icon		Indicates that the settings (that may be automatically adjusted by the tool) are locked to prevent any automatic adjustment. If a setting can be locked, it is automatically locked when you change the value. To add or remove the lock manually, use the pop-up menu command **Lock/Unlock**. **Note:** Clock element settings that cannot be automatically adjusted by the tool keep their value as-is and do not allow locking. They are: clock sources, clock selectors, and configuration elements.
Yellow background	100 MHz	Indicates that the field is directly or indirectly changed by the previous user action.
Gray text	FCTRIM	Indicates that the value of a setting does not actively influence the clock. It is disabled or relates to an inactive clock element — for example, on the clock path following an unavailable clock source or disabled element. The frequency signal also shows “inactive” instead of a frequency value. The value

Table 7. Setting states and markers...*continued*

State/Style/Marker	Icon	Description
		is also gray when it is read-only. In such a state, the value cannot be modified.

4.7 Frequency settings

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

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

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

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

Figure 53. Setting the core clock frequency

When the tool reaches the required frequency, it appears locked and is displayed as follows:

Figure 54. Tool attains the required frequency

If the tool cannot reach the required frequency or another problem occurs, it is displayed as follows:

Figure 55. Tool encounters problem

The frequency value in square brackets [ ] indicates the value the tool uses in calculations, not the requested value.

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

4.7.1 Pop-up menu commands

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

- **Lock/Unlock** - Removes a lock on the frequency which enables the tool to change any valid value that satisfies all other requirements, limits, and constraints.
- **Find Near Valid Value** - Finds a valid frequency near the specified value when the tool cannot reach the requested frequency.
- **Advanced resolver for {Clock output}** - Invokes a more advanced search for valid settings that fulfill the requirements. It may take significant time, so a progress dialog is shown. If the resolver is unsuccessful, you are informed. This command can alter clock selectors and modify other clock settings on the clock path. If the result is unsatisfactory, use the `UNDO` command to return to the original state.
- **Edit settings of: {element}** - Invokes the floating view with the settings for a single element.

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

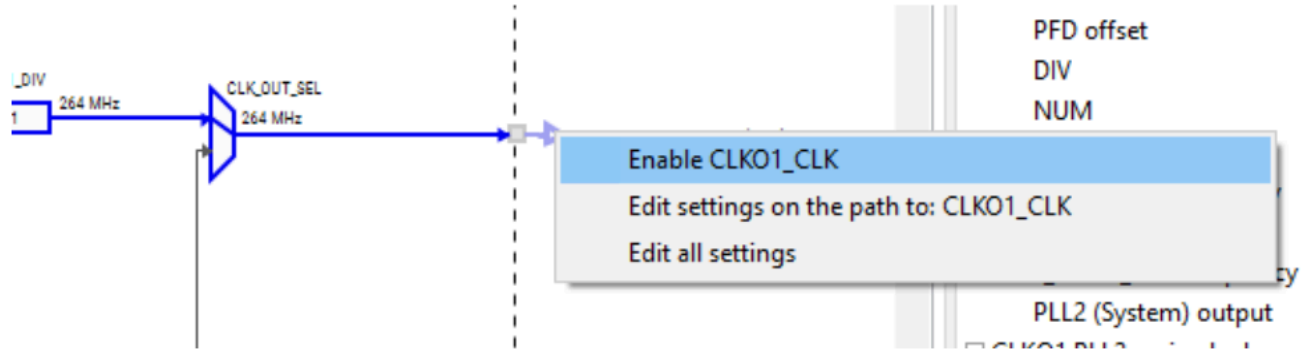


Figure 56. Pop-up commands for outputs in the clocks table view

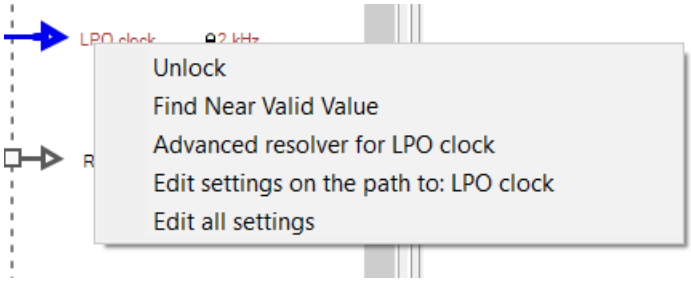


Figure 57. Pop-up commands for outputs in the clocks diagram

4.7.2 Frequency precision

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

Figure 58. Frequency precision

4.8 Dependency arrows

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

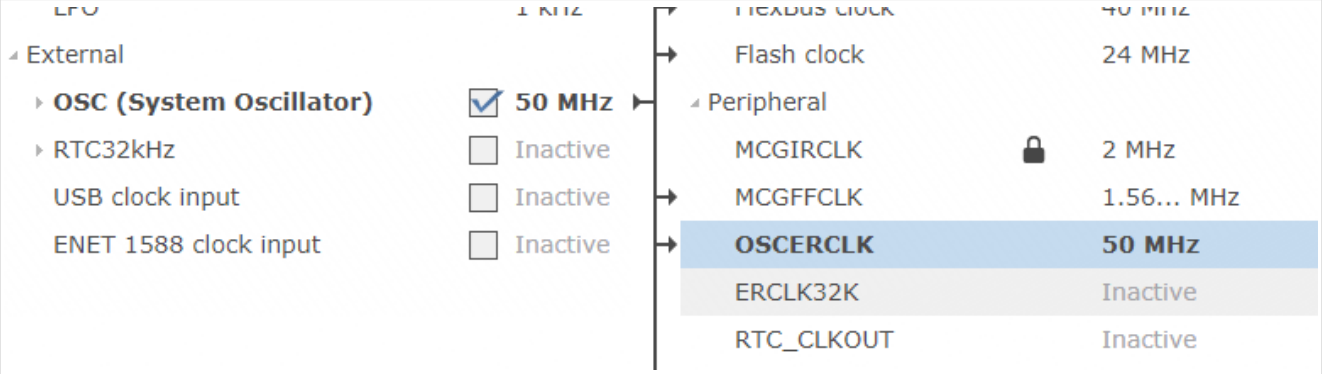


Figure 59. Dependency arrows

4.9 Details view

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

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

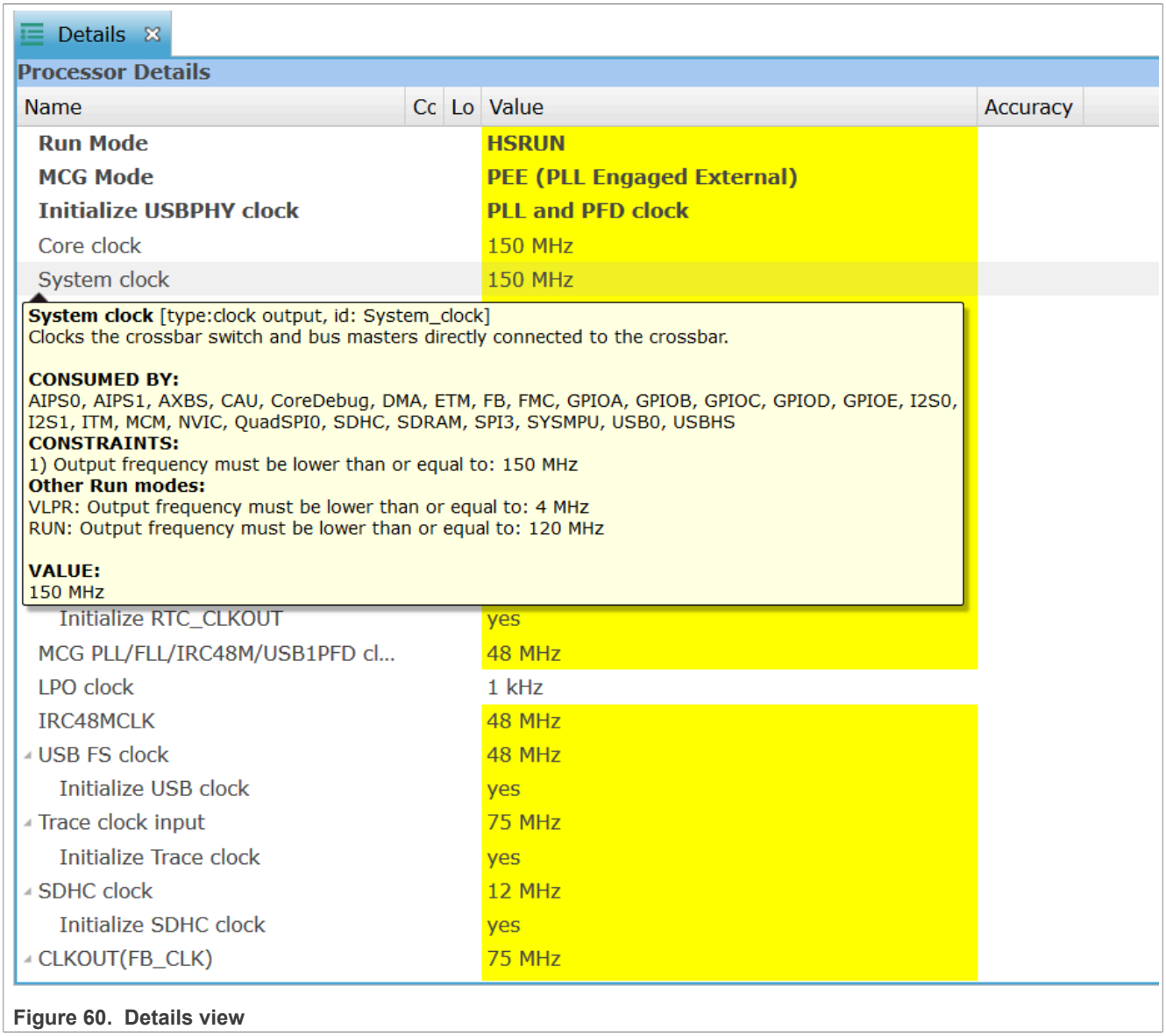


Figure 60. Details view

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

- **Display clock-element information** - Point the mouse cursor at the clock element to display general clock-element information.
- **View the clock-element in Clocks Diagram or Clocks Table** - Left-click on a clock element to highlight it in the **Clocks Diagram** or **Clocks Table** views, depending on which is currently active.
- **View detailed clock-element information** - Double-click 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.

4.10 Clocks diagram

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

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

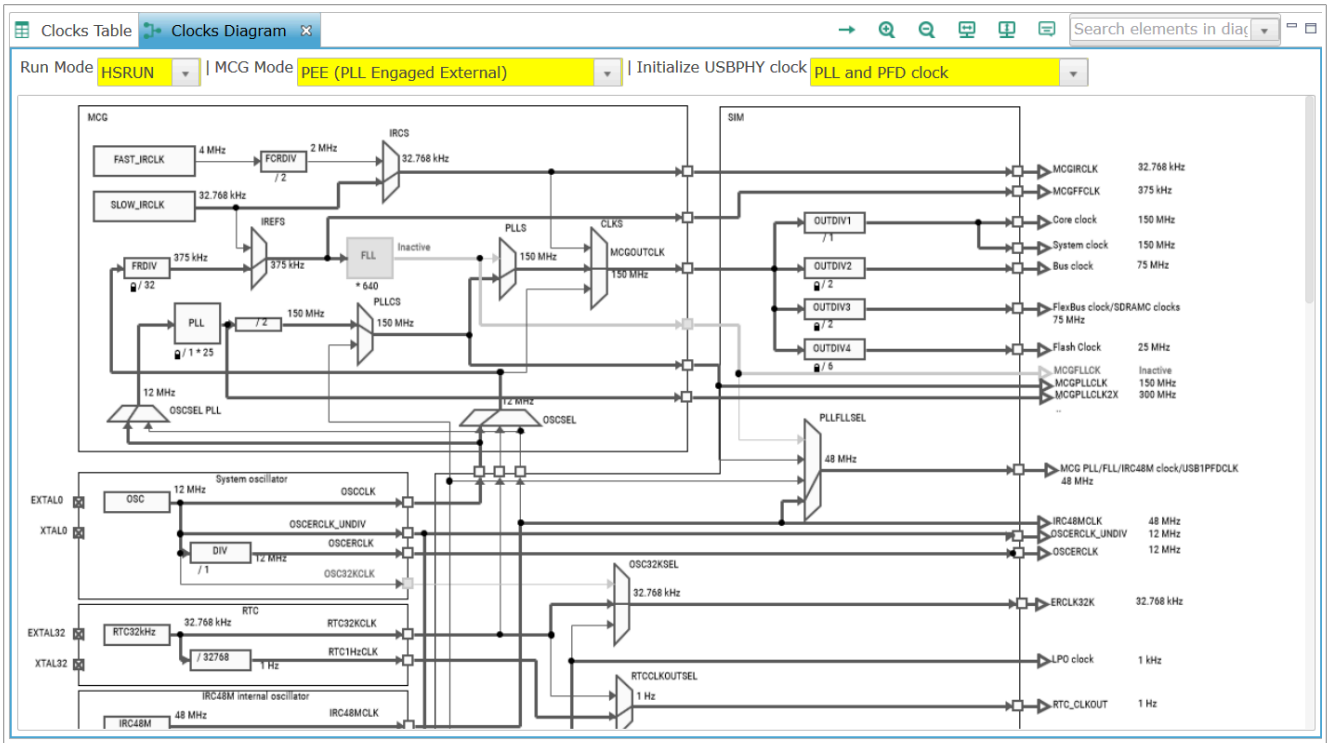


Figure 61. Clock diagram

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

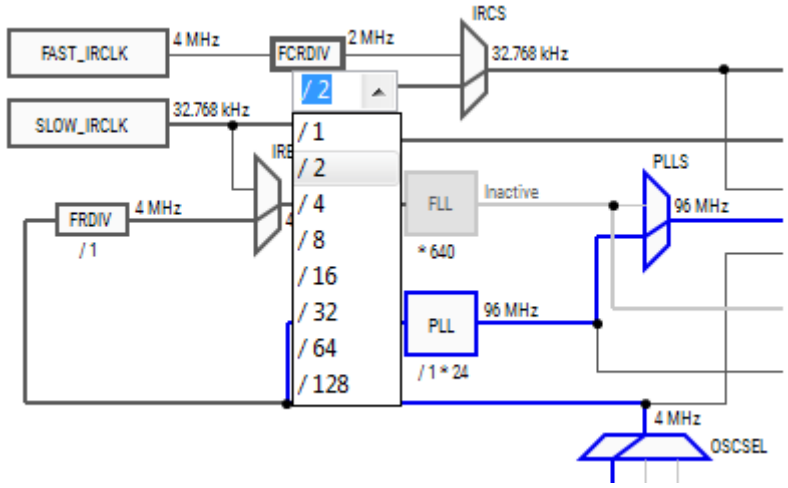


Figure 62. Clocks mouse actions in diagram

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

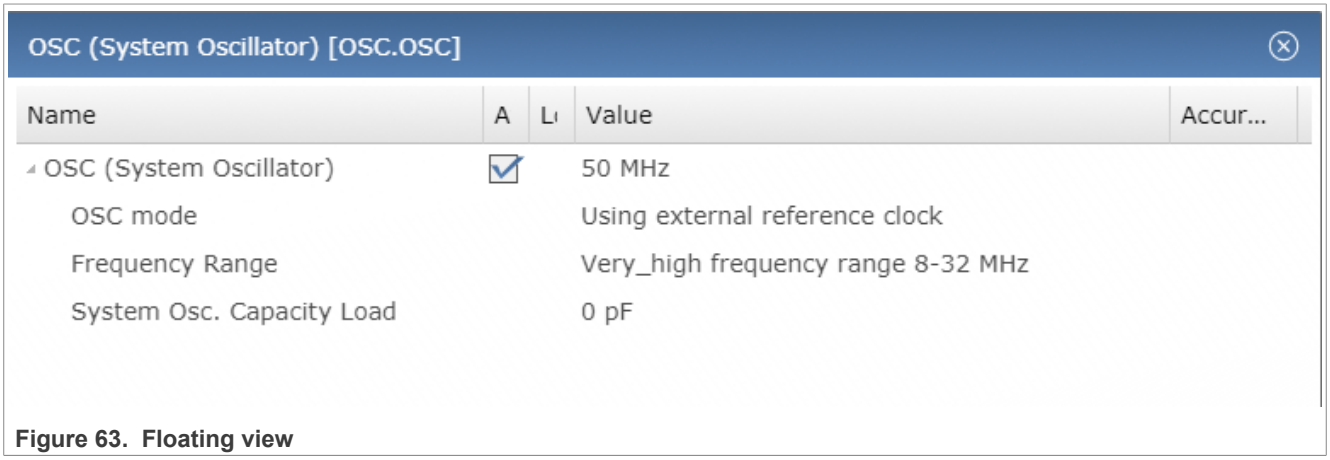


Figure 63. Floating view

4.10.2 Color and line styles

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

The color and line styles can indicate:

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

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

4.10.3 Clock model structure

The clock model consists of interconnected clock elements. The clock signal flows from the clock sources through various clock elements to the clock outputs. The clock element can have specific enable conditions that

can stop the signal from being passed to the successor. The clock element can also have specific constraints and limits that are watched by the Clocks Tool. To inspect these details, position the cursor on the element in the clock diagram to display the tooltip.

The following are the clock model elements.

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

Figure 64. Clock source

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

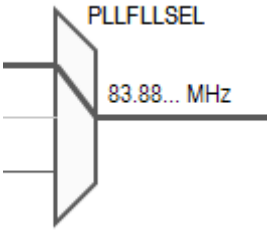


Figure 65. Clocks selector

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

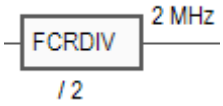


Figure 66. Prescaler

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

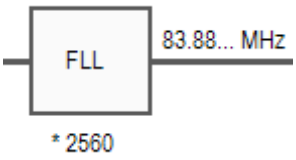


Figure 67. Frequency Locked Loop

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

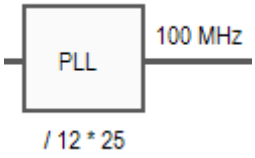


Figure 68. Phase Locked Loop

- **Clock gate -** Stops the propagation of incoming signal.

Figure 69. Clock gate

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

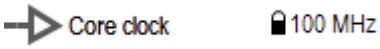


Figure 70. Clock output

- **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.

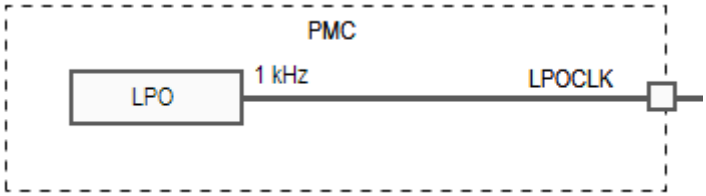


Figure 71. Clock component

- **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 Troubleshooting problems

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

You may encounter any of the following problems:

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

The following are some tips to troubleshoot encountered problems:

1. Start with only one locked clock output frequency and let the tool find and calculate other ones. After you are successful, you can add more.
2. Go through the locked outputs (if there are any) and verify the requirements (there can possibly be typos in the required frequency, wrong units, and so on).
3. If you seek only to enable some clock output, try to use pop-up the menu command **Enable** that tries to automatically find settings providing any valid frequency on clock output.
4. If the required clock output value cannot be satisfied try to use the pop-up menu command “Advanced resolver for {clock output}”.

5. If you are OK to have a near frequency value around of the requested value but would like to keep the clock selectors and clock sources unchanged, right-click and from the pop-up menu select **Clock output > Find near value**.
6. If the problems still persist, find the elements and settings with marked errors in the diagram or tables and see the details in the tooltip.
7. If you cannot reach the values you need, use the diagram view to see the elements on clock path leading to the clock output you want to have set. Try to check and adjust the settings of these elements manually in the Details view.
8. Try to remove locks by selecting **Clocks > Unlock All Settings.** In case too many changes are required and conflicting, you can simply reset the model to the default values and start from the beginning. To reset, select **Clocks > Reset to processor defaults**.

4.12 Code generation

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

4.12.1 Working with the code

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

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

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

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

5 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.

6 Note about the source code in the document

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

Copyright 2026 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

7 Revision history

Table 8. Revision history

Document ID	Release date	Description
MCUXCTWEBUG v.18.0	17 September 2026	Updated for v.26.09
MCUXCTWEBUG v.17.0	24 June 2026	Updated for v.26.06
MCUXCTWEBUG v.16.0	25 March 2026	Updated for v.26.03
MCUXCTWEBUG v.15.0	16 December 2025	Updated for v.25.12
MCUXCTWEBUG v.14.0	18 September 2025	Updated for v.25.09
MCUXCTWEBUG v.13.0	20 June 2025	Updated for v.25.06
MCUXCTWEBUG v.12.0	17 March 2025	Updated for v.25.03
MCUXCTWEBUG v.11.0	15 January 2025	Updated for v.24.12
MCUXCTWEBUG v.10.0	24 September 2024	Updated for v.16.1
MCUXCTWEBUG v.9.0	1 July 2024	Updated for v.16
MCUXCTWEBUG v.8.0	19 April 2024	Updated for v.15.1
MCUXCTWEBUG v.7.0	10 January 2024	Updated for v.15
MCUXCTWEBUG v.6.0	31 July 2023	Updated for v.14
MCUXCTWEBUG v.5.0	2 January 2023	Updated for v.13
MCUXCTWEBUG v.4.0	20 September 2022	Example workflow, Main menu, Pins routing principle are updated.
MCUXCTWEBUG v.3.0	30 June 2022	Updated for v.12
MCUXCTWEBUG v.2.0	22 December 2021	New features are added, screenshots are updated.
MCUXCTWEBUG v.1.0	1 July 2021	Minor changes
MCUXCTWEBUG v.0	27 April 2020	Initial version

Legal information

Definitions

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

Disclaimers

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

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

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

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

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

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

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

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

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

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

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

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

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

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

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

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

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

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

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

Contents