

Getting Started with MCUXpresso Configuration Tools



Contents

Chapter 1 Introduction.....	5
1.1 Conventions.....	5
1.2 Versions.....	5
1.3 Tools localization.....	7
Chapter 2 Workflow.....	8
2.1 Desktop workflow.....	8
2.2 Web workflow.....	9
Chapter 3 Tools Common Framework User Interface.....	10
3.1 Configuration.....	10
3.1.1 Creating a new configuration.....	10
3.1.2 Cloning an SDK example.....	11
3.1.3 Saving a configuration.....	12
3.1.4 Opening an existing configuration.....	13
3.1.5 Boards and kits.....	15
3.1.6 User templates.....	16
3.2 Main menu.....	18
3.3 Preferences.....	19
3.4 Updates.....	21
3.5 Problems view.....	21
3.6 Registers view.....	22
3.7 Log view.....	23
Chapter 4 Pins Tool.....	24
4.1 Example usage.....	24
4.2 Selecting Pins Tool.....	31
4.3 Pins routing principle.....	31
4.4 User interface.....	32
4.4.1 Package.....	33
4.4.2 Routed Pins view.....	36
4.4.3 View controls.....	36
4.4.4 Filtering routed pins.....	37
4.4.5 Highlighting and color coding.....	38
4.4.6 Filtering in the Pins view.....	40
4.4.7 Functions.....	41
4.4.8 Peripherals view.....	43
4.4.9 Pins table view.....	45
4.4.9.1 Labels and identifiers.....	46
4.5 Errors and warnings.....	48
4.5.1 Incomplete routing.....	49
4.6 Code generation.....	49
4.6.1 Exporting source code.....	52
4.6.2 Importing source code.....	53
4.7 Options.....	54
4.7.1 Pins properties.....	54

Chapter 5 Clocks Tool.....	55
5.1 Features.....	55
5.2 Workflow.....	55
5.3 User interface overview.....	56
5.4 Clock configuration.....	58
5.5 Global settings.....	58
5.6 Clock sources.....	58
5.7 Setting states and markers.....	59
5.8 Frequency settings.....	60
5.8.1 Pop-up menu commands.....	60
5.8.2 Frequency precision.....	61
5.9 Dependency arrows.....	61
5.10 Details view.....	61
5.11 Clock diagram.....	62
5.11.1 Mouse actions in diagram.....	63
5.11.2 Color and line styles.....	63
5.11.3 Clock model structure.....	64
5.12 Main menu.....	65
5.13 Troubleshooting problems.....	65
5.14 Code generation.....	66
5.14.1 Working with the code.....	67
5.14.2 Restoring clock configuration from source code.....	68
5.15 Module Clocks view.....	68
 Chapter 6 Project Generator Tool.....	 70
6.1 Features.....	70
6.2 Workflow.....	70
6.3 User interface.....	71
6.3.1 Views.....	71
6.3.2 Specific menu commands.....	72
6.3.3 Errors and warnings.....	72
6.3.4 Project Generator view.....	73
6.3.5 User-specific paths.....	73
6.3.6 Project Configuration view.....	73
6.3.7 Component selection.....	74
6.4 Multicore projects.....	75
6.4.1 Core booting role.....	75
6.4.2 Project Configuration view.....	75
6.4.3 Compiler symbols.....	76
6.5 Project generation.....	76
6.5.1 Export.....	77
6.6 Structure of a generated project.....	78
6.7 Toolchain-specific information.....	79
6.7.1 MCUXpresso IDE.....	79
6.7.2 Kinetis Design Studio (KDS).....	79
6.7.3 ARM GCC embedded.....	79
6.7.4 IAR embedded workbench for ARM.....	80
6.7.5 Keil MDK.....	80
6.7.6 Somnium DRT.....	80
6.8 Project updates and backup files.....	80
6.8.1 Project updates.....	80
6.8.2 Backup files.....	80

Chapter 7 Advanced Features.....81

- 7.1 Switching processor **(for desktop version only)** 81
- 7.2 Exporting Pins table..... 82
- 7.3 Export processor data..... 84
- 7.4 Tools advanced configuration..... 85
- 7.5 Generating HTML report..... 85
- 7.6 Export registers..... 86
- 7.7 Command line execution..... 87
 - 7.7.1 Command line execution - Pins Tool..... 89
 - 7.7.2 Command line execution - Clocks Tool..... 91
 - 7.7.3 Command line execution - Project Generator Tool..... 92
- 7.8 Working offline..... 95

Chapter 8 Support..... 96

Chapter 1

Introduction

The MCUXpresso Configuration 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 1. MCUXpresso Configuration Tools

Name	Description
Pins Tool	Enables you to configure the pins of a device. Pins Tool enables you to create, inspect, change, and modify any aspect of the pin configuration and muxing of the device.
Clocks Tool	Enables you to configure initialization of the system clock (core, system, bus, and peripheral clocks) and generates the C code with clock initialization functions and configuration structures.
Project Generator	Enables you to create SDK-based projects for the MCUXpresso IDE, Kinetis Design Studio 3.x, GCC ARM Embedded (command line), IAR Embedded Workbench, Keil MDK μ Vision, and Somnium DRT toolchains.

1.1 Conventions

The following conventions are used in this document.



Table 2. Conventions used in the document

Icon	Description
	Indicates that the content is related to the desktop version of the tool.
	Indicates that the content is related to the Web version of the tool.
	Indicates useful tips.

1.2 Versions

The suite of these tools is called MCUXpresso Configuration Tools. These tools are provided as an online Web application or as a desktop application.

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.
	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.

Versions



1.3 Tools localization

The Tools support English and Chinese languages, based on your locale settings.

To manually set the locale, add the following parameter to the command line:

```
tools.exe -nl zh
```

It is possible to set the locale in the tools.ini file by adding the following line:

```
-Duser.language=zh
```

The supported languages are:

- en - English
- zh – Chinese

NOTE

Setting your system locale to Chinese will automatically launch the tool with localized Chinese menu items, tool tips, and help. You may need to delete the *[home_dir]\.nxp* folder after switching languages because some menu items may be cached.

Chapter 2 Workflow

You can use the Pins Tool and the Clocks Tool autonomously. If you do not need to create project for toolchain, it is possible to use either the Pins Tool or the Clocks Tool or both to generate initialization code for pins and/or clocks. For more information, see:

- [Pins Tool](#)
- [Clocks Tool](#)

The following sections list the workflow steps for all the tools:

- [Desktop](#)
- [Web](#)

2.1 Desktop workflow

NOTE

The first step in the following workflow is optional and is necessary only for project generation. If you need just the source files for Clocks\Pins, you can skip the first step and select the processor in the **New Configuration** wizard. Data for the selected processor will be downloaded automatically.

1. Before you start the tool:
 - Go to MCUXpresso web site (<http://mcuxpresso.nxp.com>), create new configuration for your device/board and download SDK package
2. Start the tool and create new configuration
 - Start MCUXpresso Configuration Tools and create new configuration based on the SDK package, select configuration for board or processor, or start with example projects for the board. For details, see [Creating a new configuration](#).
3. Customize configuration:
 - Using command from **Tools** menu, switch to Pins tool and/or Clock tool and review/adjust configuration; ensure there are no problems
4. Project for selected toolchain
 - Use command "**Project Generator**" from **Tools** menu to start Project Generator.
 - For details on how to configure project, see [Workflow](#).
 - If there are no errors, click the "**Create Project**" button to generate project for selected toolchain.
 - Click on the link to open the folder with the generated code.

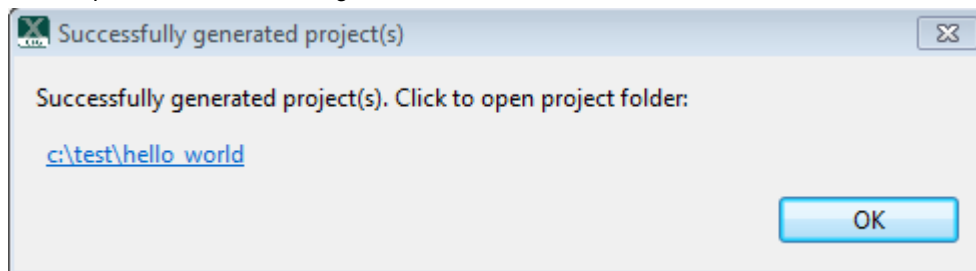


Figure 3. Click on the link to open the project folder

2.2 Web workflow

- Create configuration:
 - Go to MCUXpresso web site (<http://mcuxpresso.nxp.com>), log in, create new configuration for your device/board.
- Customize configuration:
 - Go to Pins tool and Clock tool and review/adjust configuration; ensure there are no problems.
- Project for selected toolchain:
 - Use command “**Project Generator**” from **Tools** menu to start the Project Generator.
 - For details on how to configure project, see [Workflow](#).
 - If there are no errors, click the “**Create Project**” button to generate project for selected toolchain. The generated project is stored into the SW archive.

Chapter 3

Tools Common Framework User Interface

3.1 Configuration

Configuration stands for common tools settings stored in .mex file. This file contains settings of all available tools and can be used in both web and desktop versions.

3.1.1 Creating a new configuration


To create a new configuration in desktop version:

1. Select the **File > New** command.

The **Create a new configuration** dialog appears.


2. Select SDK path you want to start with. For example, absolute path to the root SDK folder.

Figure 4. Select SDK path

	SDK package can be downloaded from the mcuxpresso.nxp.com web site.
---	---

If you do not have SDK package yet, you can start without the SDK package and specify the SDK path later in the **Project Generator** tool.


3. Click **Next**.
4. Select either the processor or the board or the SDK example project.
Alternatively, type the name of the processor in the **Search Processor/Board** text box and select the processor from the filtered results.
5. To specify a different name for your configuration, edit the text in the **Name your configuration** text box at the bottom of the dialog.
6. For multi-core processors:
 - a. Click **Next**.
 - b. Select boot role for each core.
7. Click **Finish** to create the configuration.

	If you select a board or an example, the configuration contains board specific configuration for the pins and the clocks. However, if you select processor, the configuration of the Pins/Clocks tool is empty.
---	---

3.1.2 Cloning an SDK example

The desktop tool is capable to create standalone project for the selected toolchain from any SDK example.

For example, create an example project with all sources and libraries needed for compilation. Such an example does not contain any dependency on the SDK package and can be compiled on a machine, where SDK is not installed.

	To customize the configuration of the example, use the option to create a new configuration and select an example you want to modify. Next, invoke the Project Generator tool and generate the project.
---	---

To clone the SDK example project:

1. Select **File > New...** in the main menu.
2. Select the SDK path.
3. Select the option **Clone an example project**.

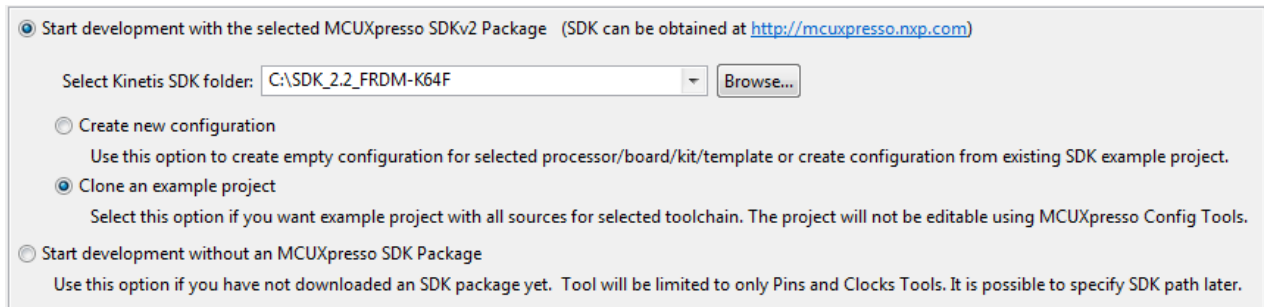


Figure 5. Clone an example project

4. Click **Next**.
5. Expand the **Boards** tree.
6. Select the required board.
7. Expand the **Examples** and select any example.
8. Click **Next**.

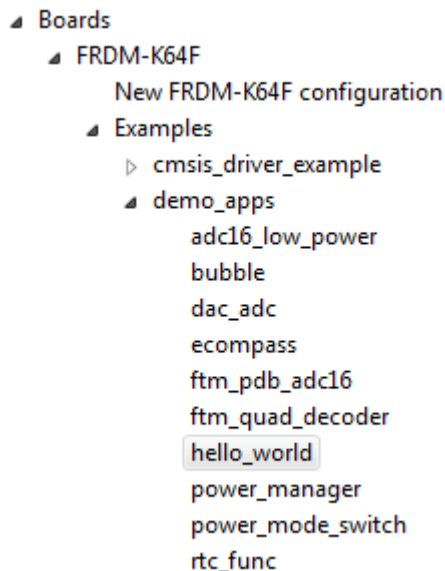


Figure 6. Select a board and an example

9. Select the target directory, the project name and the toolchain.

NOTE

Project name will be appended to target path. Before you finish the cloning ensure that the project directory does not exist on the disk.

10. Click **Finish**.

NOTE

After the cloning is finished, the result is displayed at the end of the operation. The operation does not create any new configuration. The result of the operation is a project for the selected toolchain in the specified directory. If the operation is successful, the wizard restarts and allows you to directly clone an additional project.

3.1.3 Saving a configuration

To save a configuration or a profile, select **File > Save As**.

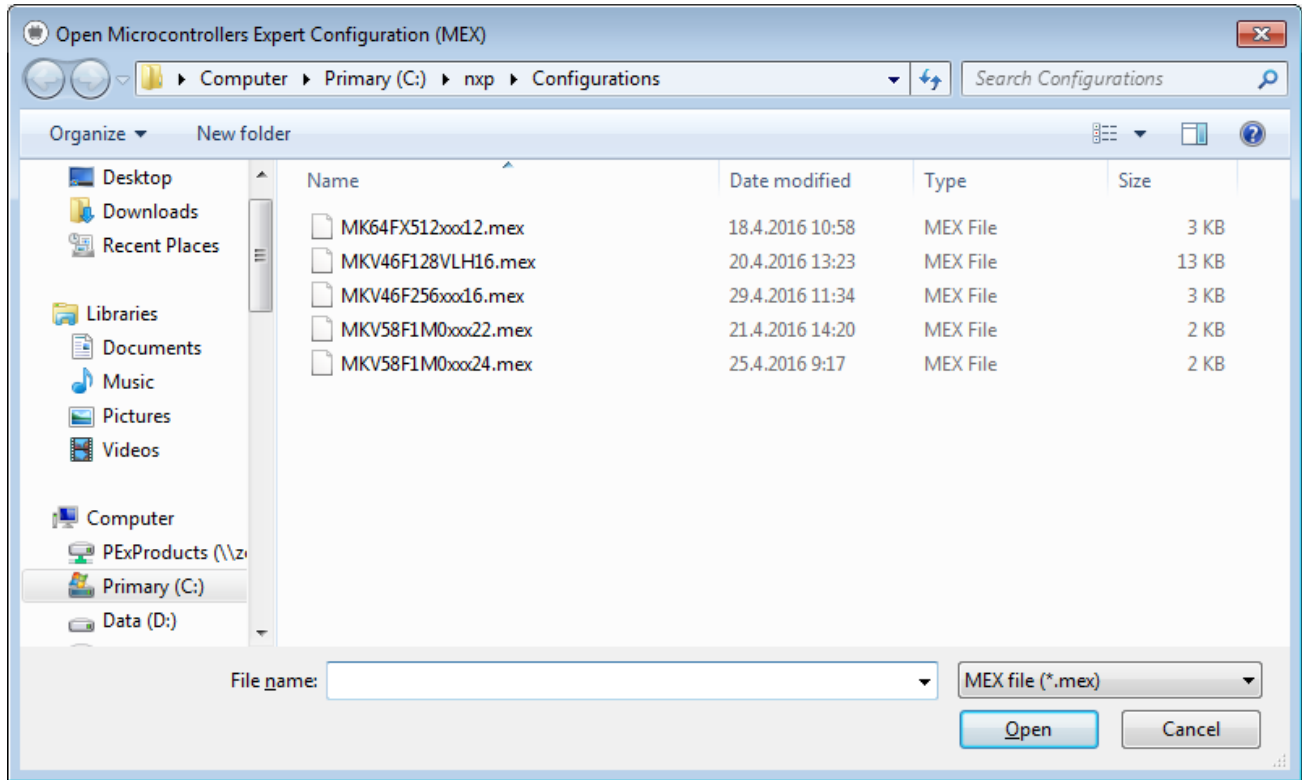


Figure 7. Save configuration

NOTE

The configuration is stored with a .mex file extension.

3.1.4 Opening an existing configuration

To open a previously saved profile:

1. Select **File > Open**.

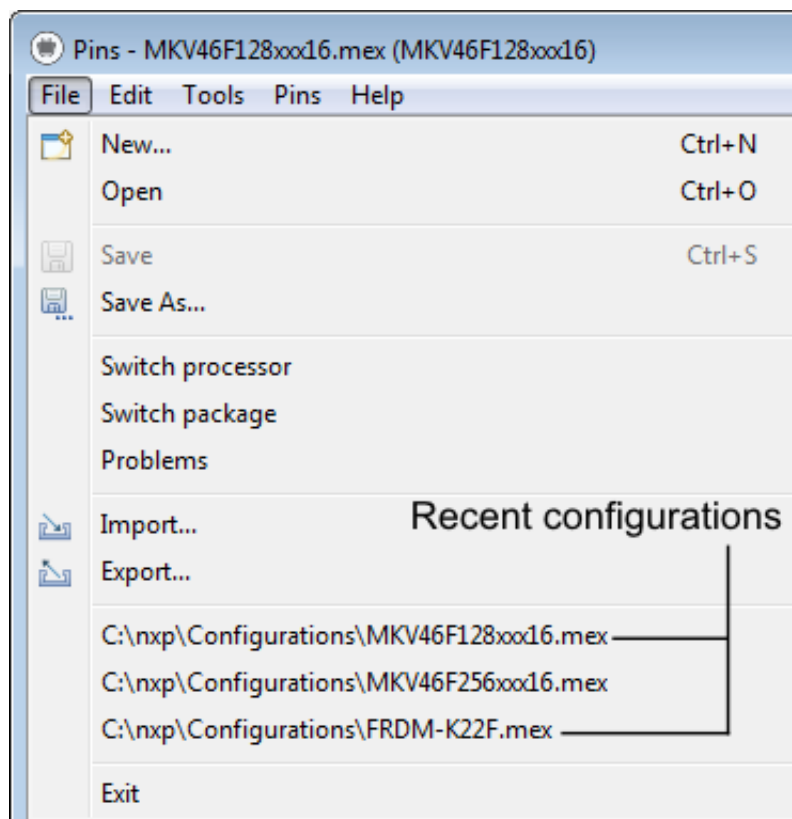


Figure 8. Open profile

The **Open profile** dialog appears.

2. Navigate to the folder where the previous profile has been saved.

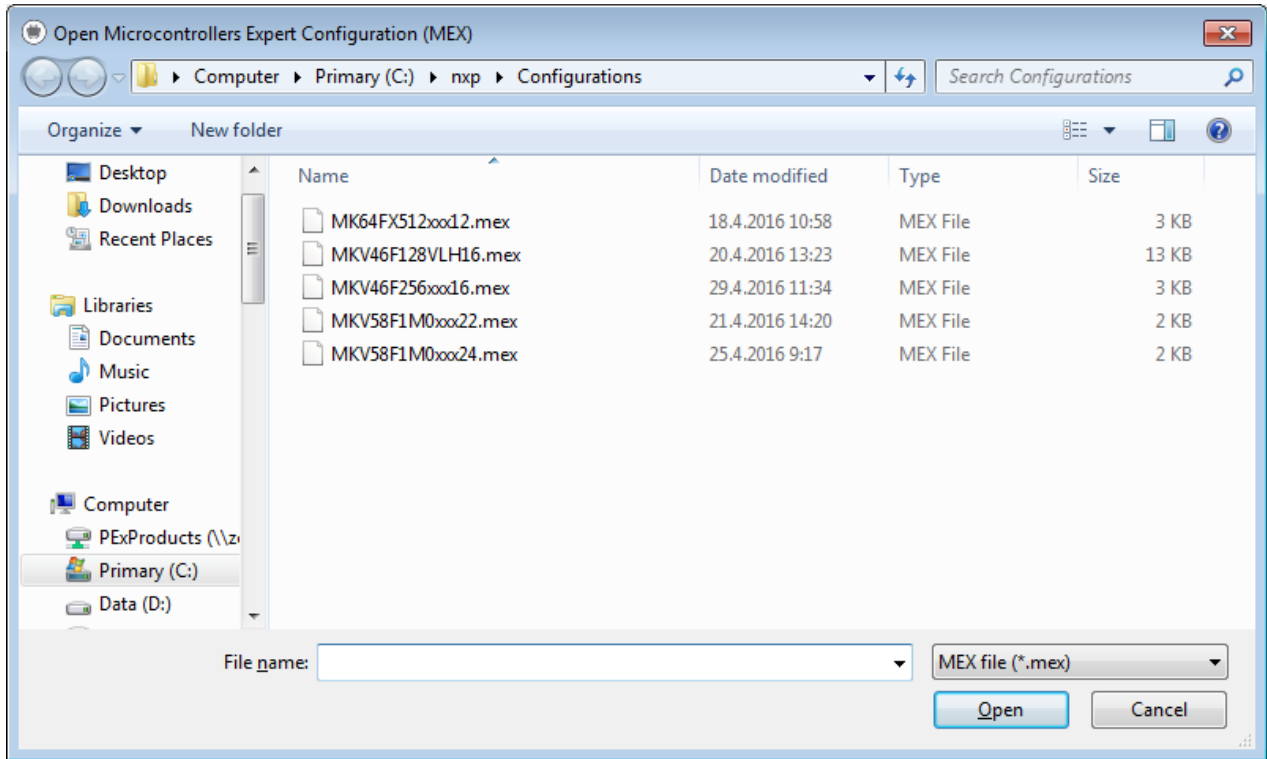


Figure 9. Open dialog

3. Select the profile and click **Open**.

The configuration is loaded by all the tools.

3.1.5 Boards and kits

The tool helps in creating a new configuration for the given board/kit.

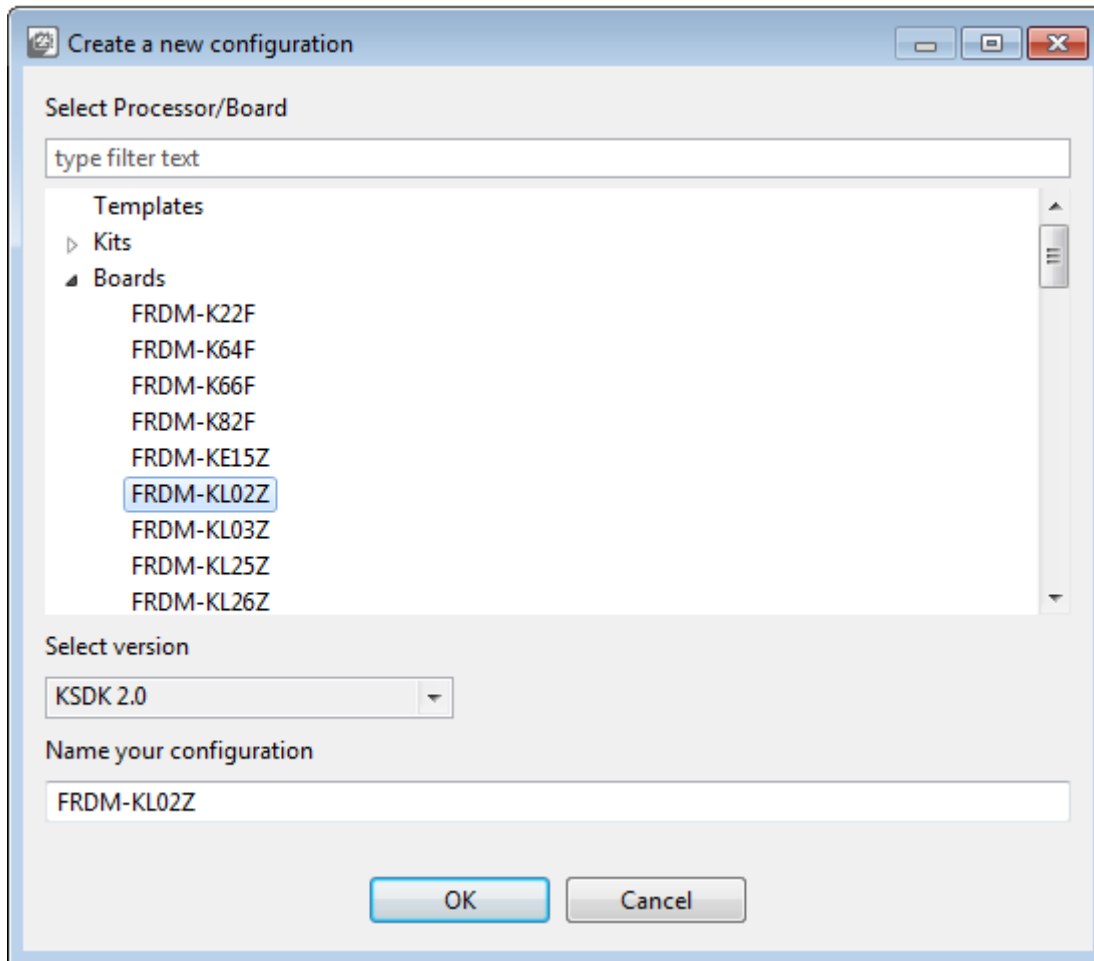


Figure 10. Create a new configuration for a given board/kit

The configuration contains pre-defined settings that corresponds to the board/kit description. For the boards/kits, additional examples are available from the SDK package if the configuration is based on the SDK package.

3.1.6 User templates

You can export and store the current configuration as a reference configuration for later use as a user template.

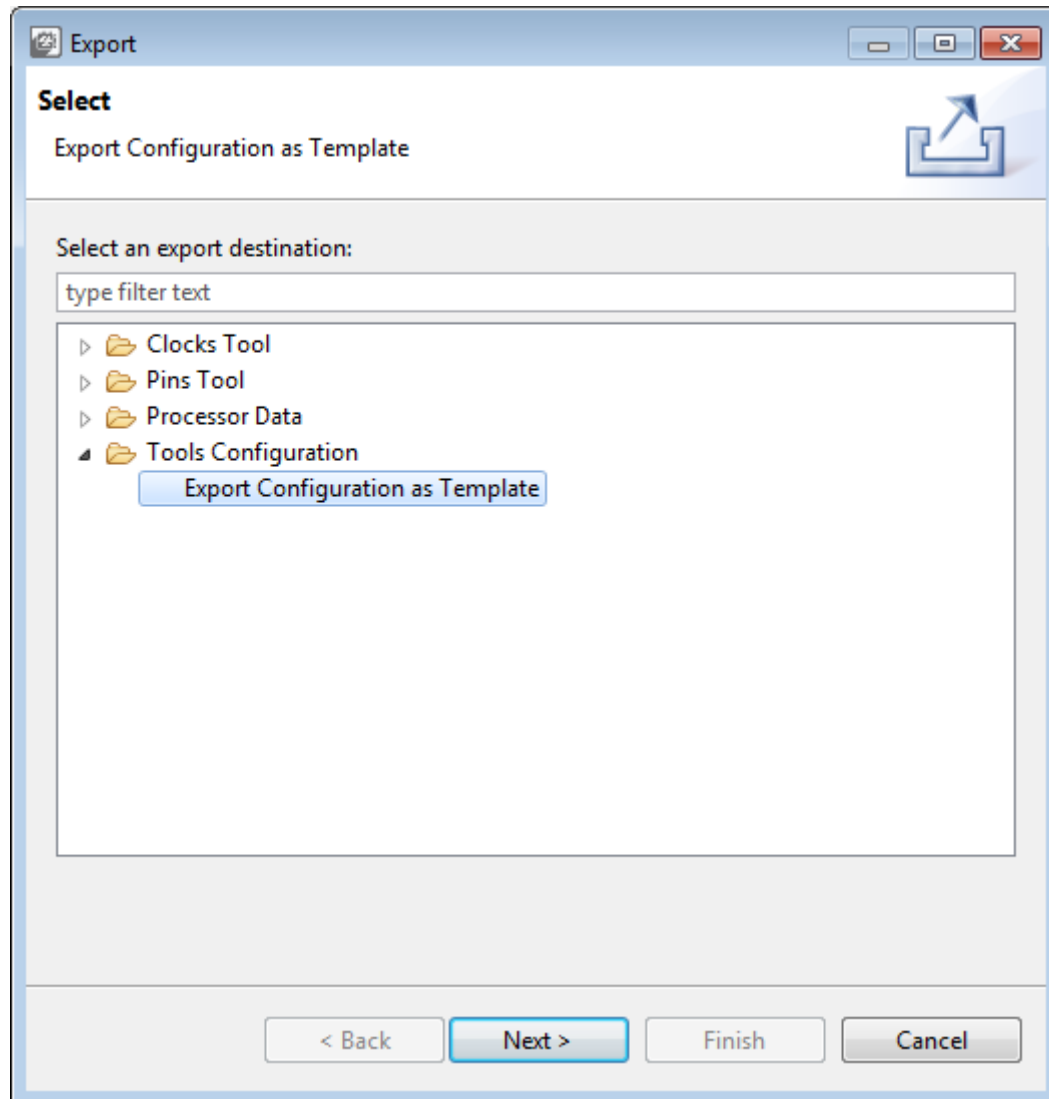


Figure 11. Export template

The exported template is available in the **New Configuration** dialog and can be used to create a new configuration. You can also define custom labels for pins or identifiers prefixes for `#define` in generated code. You can export the configuration by selecting **Tools Configuration > Export Configuration as Template** option in the **Export** dialog.

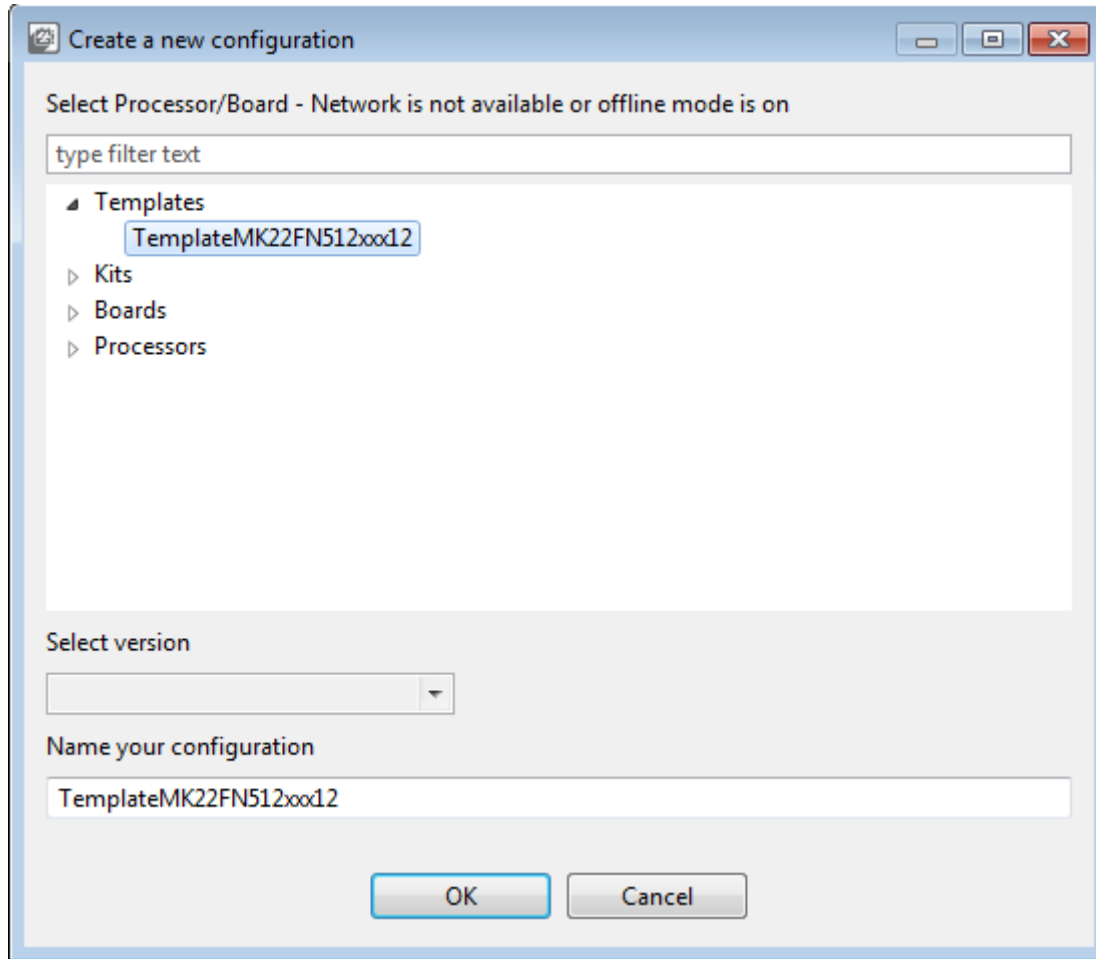


Figure 12. Create a new configuration from the template

NOTE

The templates are stored in at the following location on your local hard disk: `{user}/.nxp/{tools_folder}/{version}/templates`.

3.2 Main menu

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

NOTE

The menu may also contains a tool specific commands that are described in the chapter dedicated to the particular tool.

- **File** (Desktop version only)
 - **New...** – Creates a new configuration (used for all tools). See the [Configuration](#) section for details.
 - **Open** – Opens a configuration settings of all tools. from .mex file. Shows a file selection dialog allowing to select the file.
 - **Save** – Saves the current configuration. If invoked for the first time, it shows a file selection dialog allowing to set the name.
 - **Save As...** – Saves the current configuration, always allowing to specify the name using a file selection dialog.

- **Switch processor** – Allows switching to a different processor. See the [Switching processor](#) section for details.
- **Switch package** – Allows switching to a different processor package. See the [Switching processor](#) section for details.
- **Problems** – Opens the **Problems view**. See the [Problems view](#) section for details.
- **Import...** – Opens import dialog and allows to import settings from source files. For details on additional importing of Pins [Pins](#) and [Clocks](#) configuration from legacy tools, see [Advanced Features](#).
- **Export...** – Opens export dialog and allows to export source or other information from the tools. For details on additional exporting of [Pins](#) and [Clocks](#) configuration from legacy tools, see [Advanced Features](#).
- **Exit** – Ends the application. If there are any unsaved changes, you are prompted to save the changes.
- **Edit** (Desktop version only) – command related to text editing useful in text fields or source code view.
 - **Copy** – Copies the selected text into the clipboard.
 - **Select All** – Selects the whole text in the current field/view.
 - **Preferences** - See the [Preferences](#) section for details.
- **Tools** (Desktop version only) – Lists all the tools available in the tools framework. Use this menu to switch between the tools.
- **Tool specific menu** – See the respective tool section for details. This product supports only Pins tool.
- **Views** - which consists of:
 - List of views available for selected tool. After clicking a view from the list it will become visible (reopens if closed).
 - Reset views command which reset the actual tool perspective to default state.
- **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.
 - **Check for updates** (Desktop version only) – Checks if there is a newer version of the product available. If a new version is available, it offers you to perform the update after a confirmation.
 - **Show diagram legend** (Clocks tool only) – Shows a window with samples and descriptions of styles used within the clocks diagram.
 - **About** – Shows dialog with information about a product.

3.3 Preferences

To configure preferences, select **Edit > Preferences** from the main menu. The configuration Preferences dialog appears.

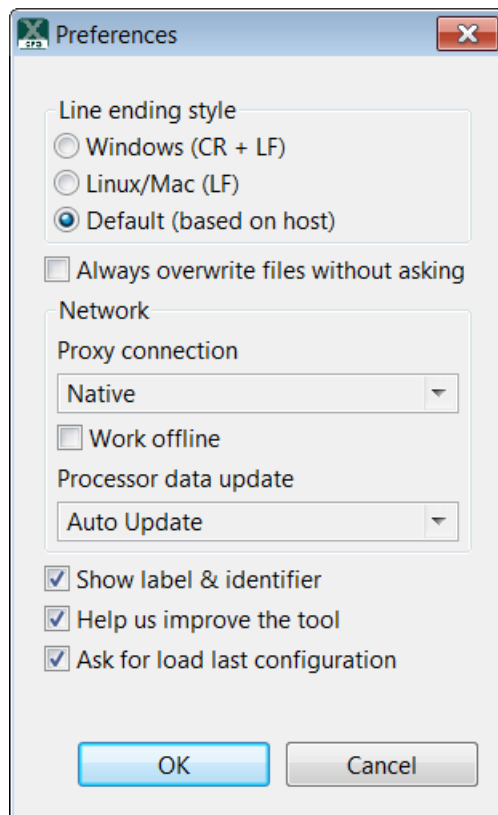


Figure 13. Preferences dialog

In this dialog it is possible to set:

- Line ending style (Windows, Linux, or default based on host)
- Auto overwrite files on save command
- Proxy connection
 - Direct – direct network connection without any proxy.
 - Native – uses system proxy configuration for network connection.
- Work offline
 - It will not download/update new data from the NXP cloud as it will not show all possible processors available for download if this feature is enabled.
- Processor data update options
 - Update will proceed either automatically or manually (user has to confirm it when requested) or get disabled.
- Show label & identifier – Check this option to display label and identifier in the routing view. For description see [Pins table](#).
- Help us improve the tool - If enabled, the tool can send info to NXP, such as device configuration and info about how you use the tool. This helps us fix problems and improve the tool. You can turn this off any time using this setting.
- Ask for load last configuration - Check this option if you want the tool to prompt on each start whether to load the last opened configuration or not. The question dialog contains the “Do not ask again” check box which is related to this option.

3.4 Updates

To perform a check for updates select the **Help > Check for updates** menu. It contacts the server and checks whether there is a new version available.

NOTE

To check updates, internet connection is required.

3.5 Problems view

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

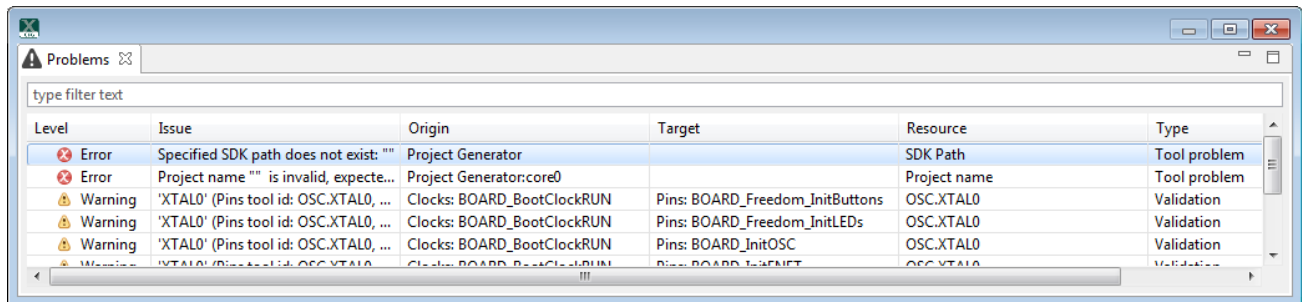


Figure 14. Problems view

To open the **Problems** view select **Views > Problems** or **File > 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

There is a context-menu for each problem that shows the problem in the tool (to see context of the problem) or the quick-fix to the problem.

NOTE

The quick-fix is not provided for all the listed problems.

3.6 Registers view

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

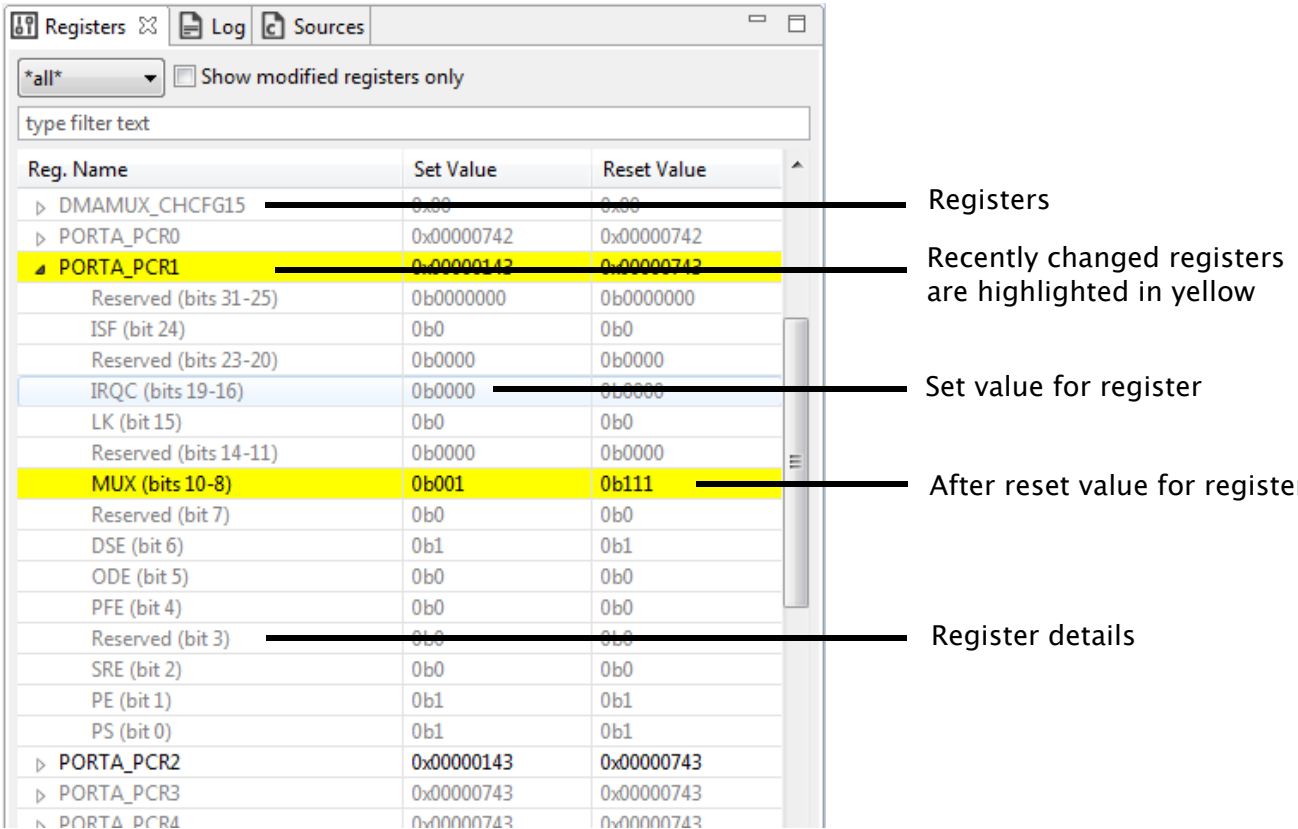


Figure 15. Registers view

The **Registers** view contains:

- **Peripheral filter** drop-down list – Use this filter 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** - Enables you to filter content by text.

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

Table 3. Color codes

Color	Description
Yellow background	Indicates that the bit-field has been affected by the last change made in the tool.
Table continues on the next page...	

Table 3. Color codes (continued)

Color	Description
Gray text color	Indicates the bit-field is not edited and the value is the after-reset value.
Black text	Indicates the bit-fields that the tool modifies.

NOTE

This view contains registers for the 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.

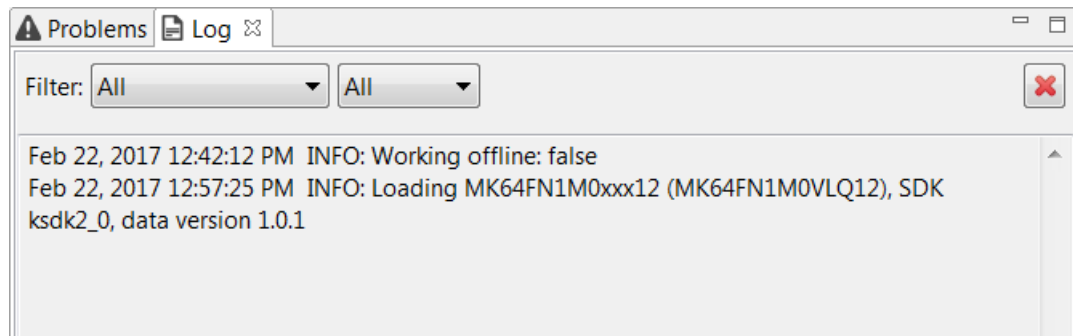
3.7 Log view

The **Log** view shows user-specific information about the progress of the tools. The **Log** view can show up to 100 records throughout the 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 record is created by the shared functionality.

The content of the **Log** view is filtered using the combo boxes and shows only the specific tool and/or severity of the record.

The buffered log records are cleared using the clear button.

**Figure 16. Log view**

Chapter 4

Pins Tool

The Pins Tool is an easy-to-use way to configure the pins of the device. The Pins Tool software enables you to create, inspect, change, and modify any aspect of the pin configuration and muxing of the device. The following sections introduce you to the Pins Tool. The sections describe the basic components of the tool and lists the steps to configure and use the tool to configure the pins.

This chapter describes the Pins Tool principle and its use to generate the routing and muxing for pins.

4.1 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.

The steps are listed both for the desktop and the Web version.

NOTE

The Web version exists only for the Kinetis and LPC devices.

You can use the generated files with the application code.

1. For the **Desktop** version, launch the MCUXpresso Configuration Tools with the shortcut present in the installation folder.
2. Skip the **Create a new configuration** dialog and select **Tools > Pins** to launch the **Pins Tool**.

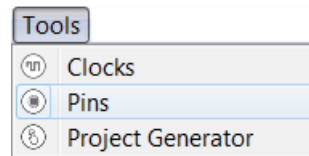


Figure 17. Select the Pins Tool

3. For the **Web** version, go to [Step 6](#).
4. In the **Desktop** version, create a new configuration with the menu **File > New**.
5. Select the SDK Package and the processor/board/kit you want to use.

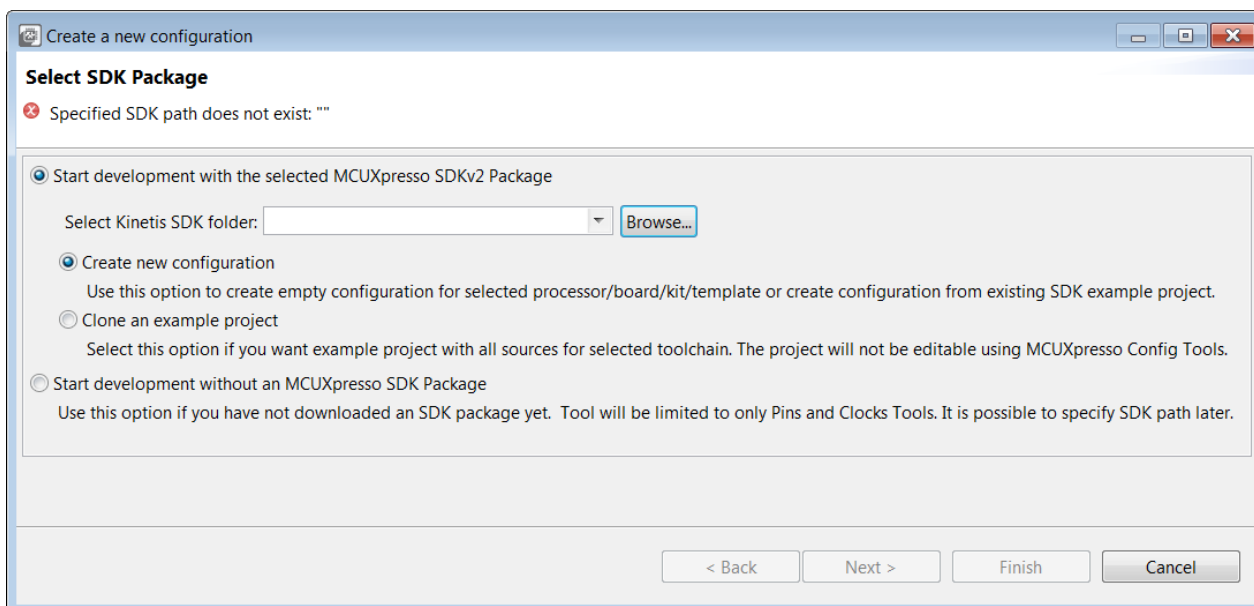


Figure 18. Select SDK Package

Optionally, name the configuration to match your project.

6. For the Web version visit <http://mcuxpresso.nxp.com>
 - a. Select an existing configuration or create a new configuration.

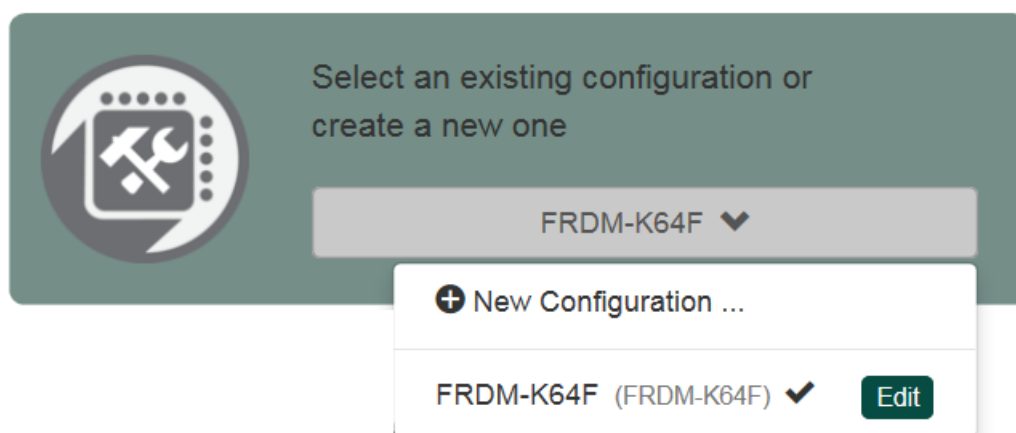


Figure 19. New Configuration

- b. For a new configuration, select the processor/board/kit and the configuration.

Search by Name

Select a Device, Board, or Kit

▼ Boards

FRDM-K64F

▼ Processors

▼ Kits

FRDM-K64F-AGM01

FRDM-K64F-AGM04

FRDM-K64F-MULT2B

Figure 20. Select a configuration to begin

- c. Optionally download the SDK package for the configuration. SDK package is necessary for the project generation.

SDK Builder

Generate a downloadable SDK archive for use with desktop MCUXpresso Tools.

Current Configuration

FRDM-K64F-demo ▼

MCUXpresso SDK Details [MCUXpresso SDK Documentation](#)

Toolchains and Host OS selections can be edited using the Tools->Configurations Settings menu

Included SDK Contents

MCU Device(s):	MK64FN1M0xxx12
Development Board(s):	FRDM-K64F
MCUXpresso SDK Version:	KSDK 2.1.0
Toolchain:	Kinetis Design Studio
Host OS:	Windows

Figure 21. Build SDK Package

- d. After creating a new configuration, navigate to **Tools > Pins Configuration**.

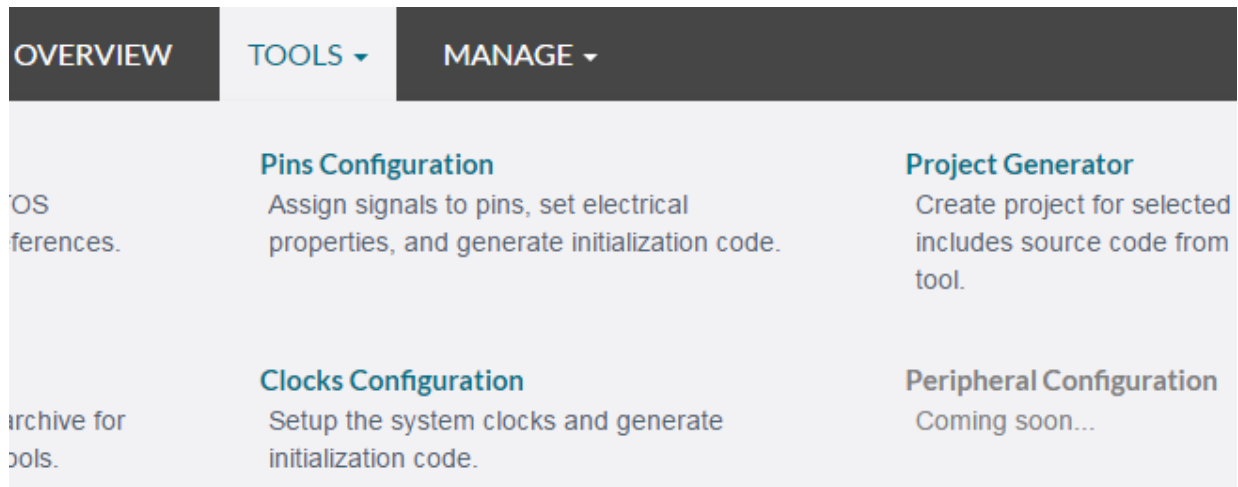


Figure 22. Switch to Pins Tool

7. In the **Pins** view on the left, select the **UART3_RX** and TX signals and the PTB20 signals. For this, you can click into the cells to make them 'green'.

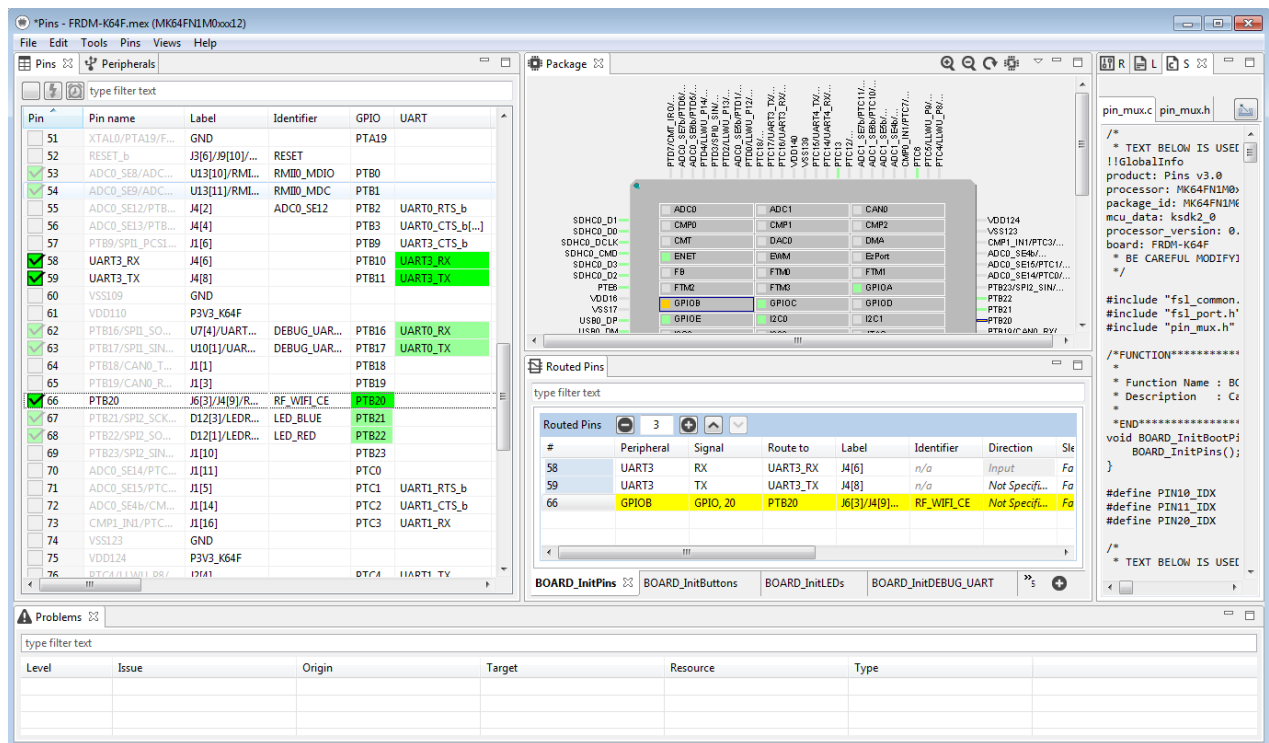


Figure 23. Configure Signals in Pins Tool

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

Example usage

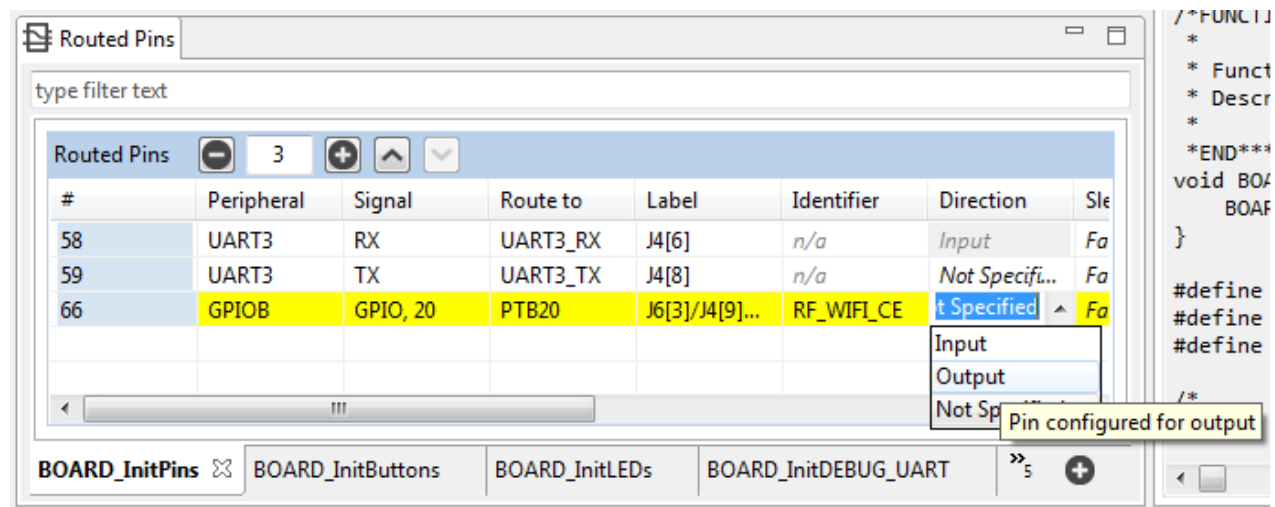
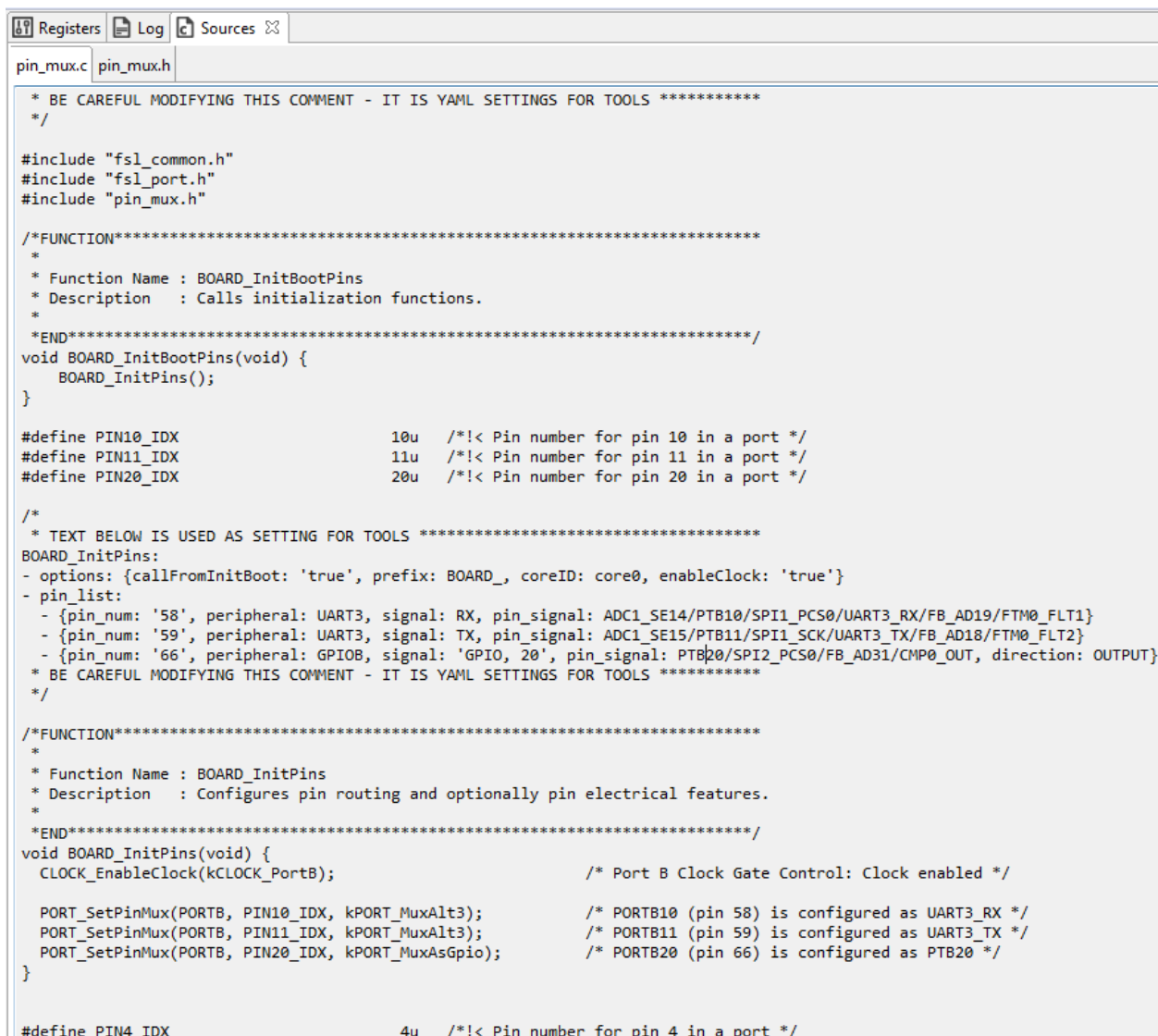


Figure 24. Select output pull-up/down

- The Pins Tool automatically generates the source code for `pin_mux.c` and `pin_mux.h` on the right panel of the **Sources** tab.



```

Registers Log Sources
pin_mux.c pin_mux.h

/* BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS *****
*/

#include "fsl_common.h"
#include "fsl_port.h"
#include "pin_mux.h"

/*FUNCTION*****
*
* Function Name : BOARD_InitBootPins
* Description   : Calls initialization functions.
*
*END*****/
void BOARD_InitBootPins(void) {
    BOARD_InitPins();
}

#define PIN10_IDX          10u  /*!< Pin number for pin 10 in a port */
#define PIN11_IDX          11u  /*!< Pin number for pin 11 in a port */
#define PIN20_IDX          20u  /*!< Pin number for pin 20 in a port */

/*
* TEXT BELOW IS USED AS SETTING FOR TOOLS *****
BOARD_InitPins:
- options: {callFromInitBoot: 'true', prefix: BOARD_, coreID: core0, enableClock: 'true'}
- pin_list:
  - {pin_num: '58', peripheral: UART3, signal: RX, pin_signal: ADC1_SE14/PTB10/SPI1_PCS0/UART3_RX/FB_AD19/FTM0_FLT1}
  - {pin_num: '59', peripheral: UART3, signal: TX, pin_signal: ADC1_SE15/PTB11/SPI1_SCK/UART3_TX/FB_AD18/FTM0_FLT2}
  - {pin_num: '66', peripheral: GPIOB, signal: 'GPIO, 20', pin_signal: PTB20/SPI2_PCS0/FB_AD31/CMP0_OUT, direction: OUTPUT}
* BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS *****
*/

/*FUNCTION*****
*
* Function Name : BOARD_InitPins
* Description   : Configures pin routing and optionally pin electrical features.
*
*END*****/
void BOARD_InitPins(void) {
    CLOCK_EnableClock(kCLOCK_PortB); /* Port B Clock Gate Control: Clock enabled */

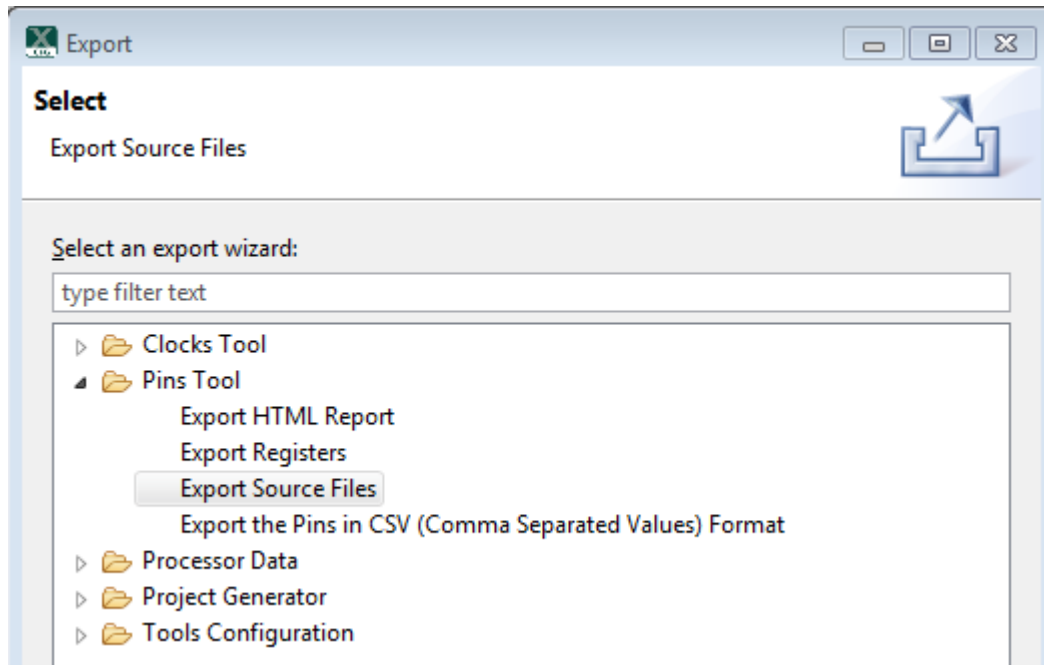
    PORT_SetPinMux(PORTB, PIN10_IDX, kPORT_MuxAlt3); /* PORTB10 (pin 58) is configured as UART3_RX */
    PORT_SetPinMux(PORTB, PIN11_IDX, kPORT_MuxAlt3); /* PORTB11 (pin 59) is configured as UART3_TX */
    PORT_SetPinMux(PORTB, PIN20_IDX, kPORT_MuxAsGpio); /* PORTB20 (pin 66) is configured as PTB20 */
}

#define PIN4_IDX          4u  /*!< Pin number for pin 4 in a port */

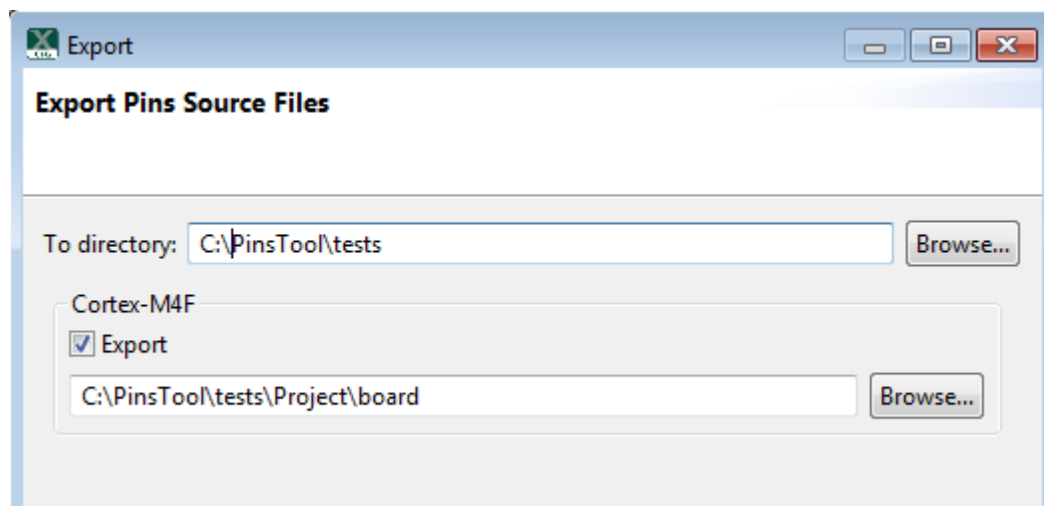
```

Figure 25. Generated code

10. 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.

Example usage**Figure 26. Export Source Files**

11. 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).

**Figure 27. Exported Pins Source Files Directory**

12. Click **Finish** to export the files.
 13. Integrate and use the exported files in your application as source files.
- You have created the configuration for your pins.
- The following sections in this User's Guide describe the features of the tool in detail.

4.2 Selecting Pins Tool

Select the Pins Tool either through the menu or by using the shortcut installed on the **Desktop** version.

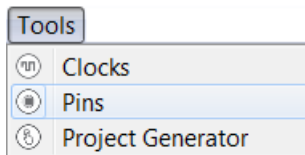


Figure 28. Selecting Pins tool - Desktop

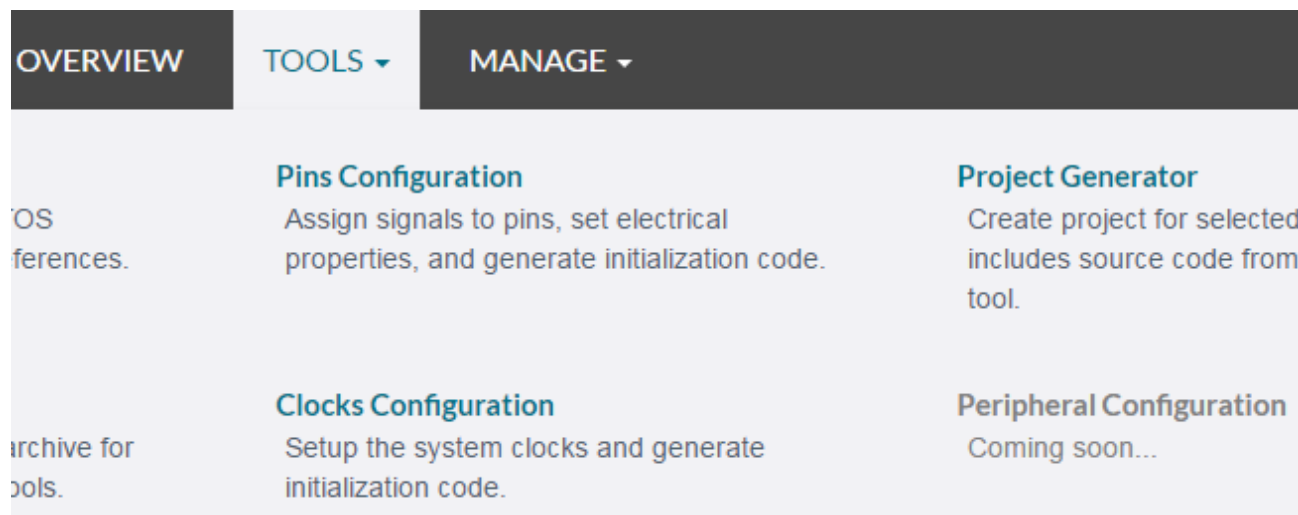


Figure 29. Selecting Pins Tool - Web

4.3 Pins routing principle

The Pins Tool is designed to configure routing of signals from peripherals either to pins or to internal signals.

To define routing path, first select a peripheral, then select the signal, and finally select the pin.

1. For the selected **Peripheral**, select one of the available signals.
2. Route the selected **Signal** to the desired pin.
3. Select one of non-conflicting/available pins. Once you have selected **Peripheral**, **Signal**, and **Route to**, the pin configuration is done. Later, it is also possible to configure the pin electrical features.

Pins Tool
User interface

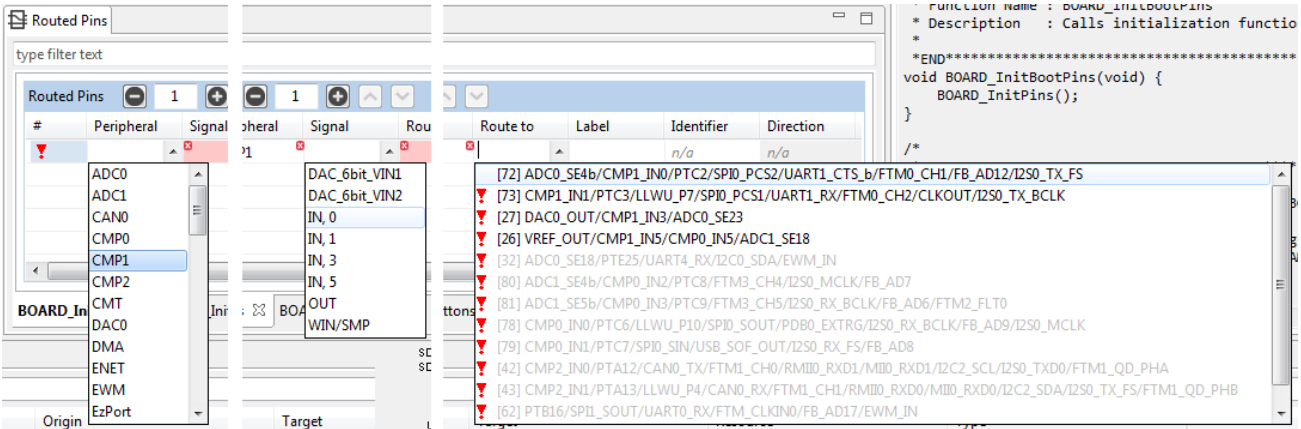


Figure 30. Defining routing path

4.4 User interface

The Pins Tool consists of several views.

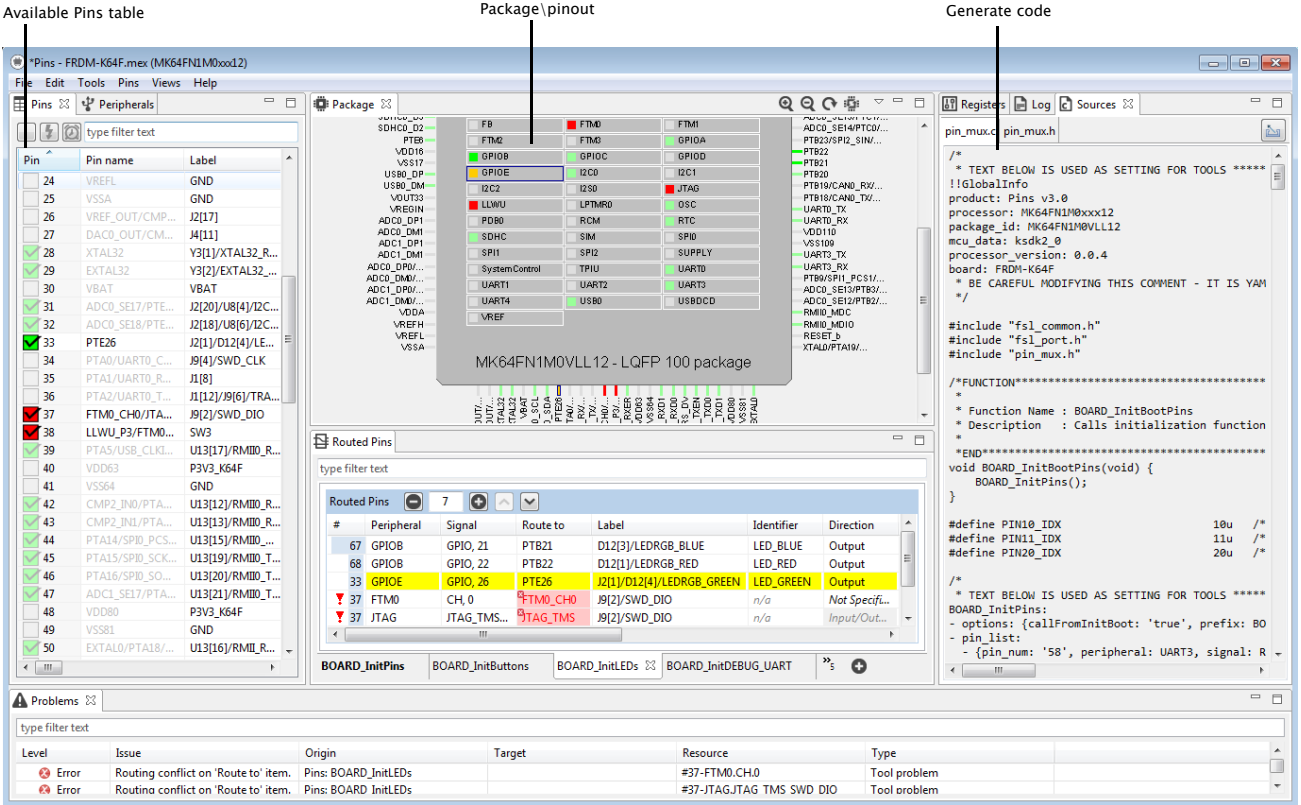


Figure 31. Pins Tool user interface

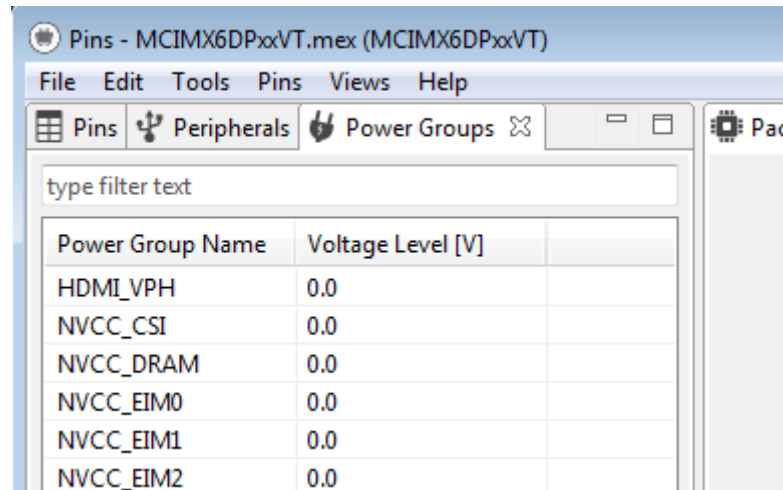


Figure 32. Selecting power group

NOTE

Power Groups are not supported for all processors.

4.4.1 Package

The processor package appears in the middle of the Pins Tool window. The processor package shows an overall overview of the package including the resources allocation.

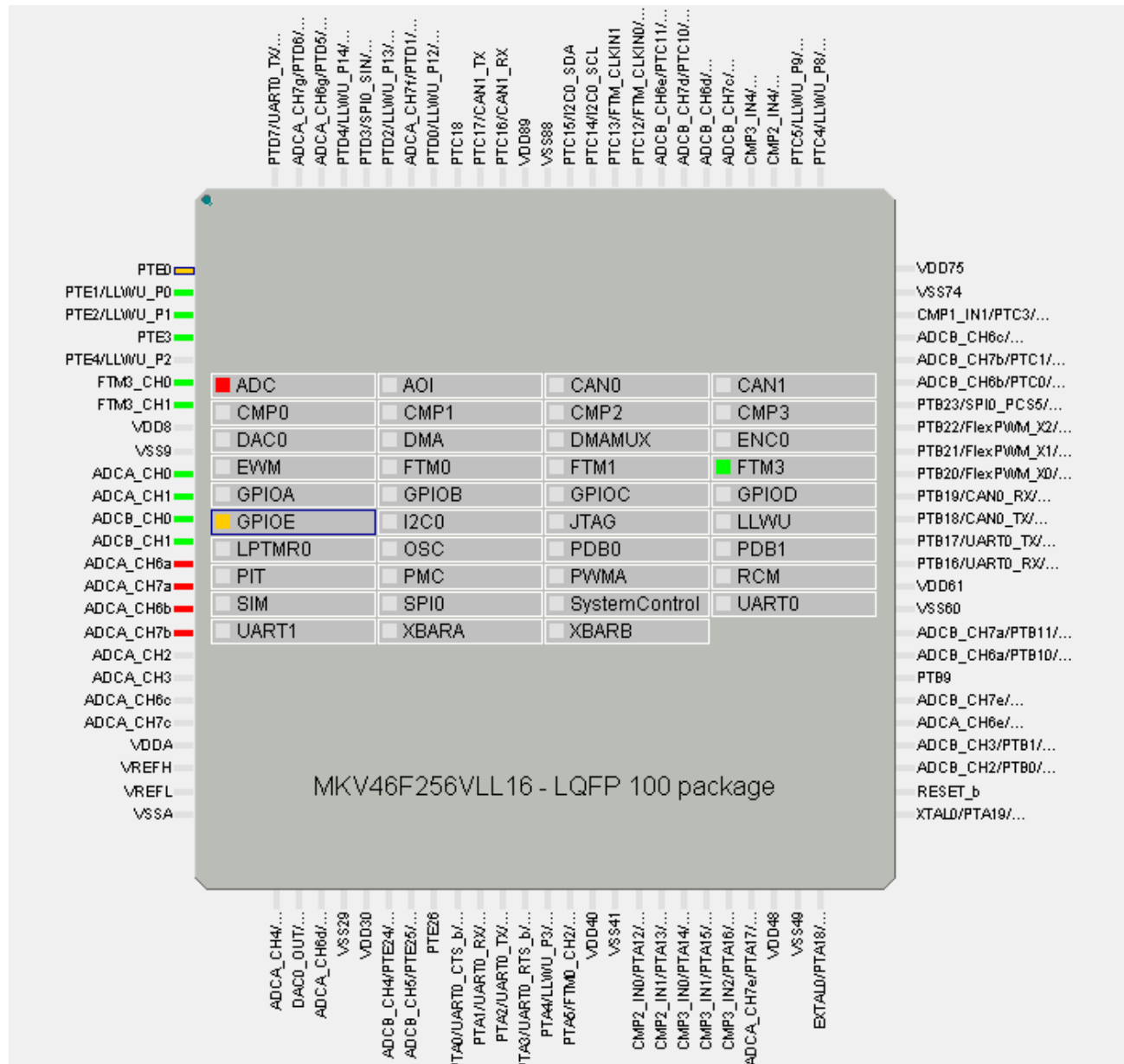


Figure 33. Processor package

This view shows Package overview with pins location. In the center are the peripherals.








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

- Green color indicates the routed pins/peripherals.
- Gray color indicates that the pin/peripheral is not routed.

The view also shows the package variant and the description (type and number of pins).

The following icons are available in the toolbar:

Table 4. Toolbar options

Icon	Description
	Zoom in package image.
	Zoom out package image.
	Rotate package image.
	Show pins as you can see it from the bottom. This option is available on BGA packages only.
	Show pins as you can see it from the top. This option is available on BGA packages only.
	Show resources. This option is available on BGA packages only.
	Switch package.

NOTE

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

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

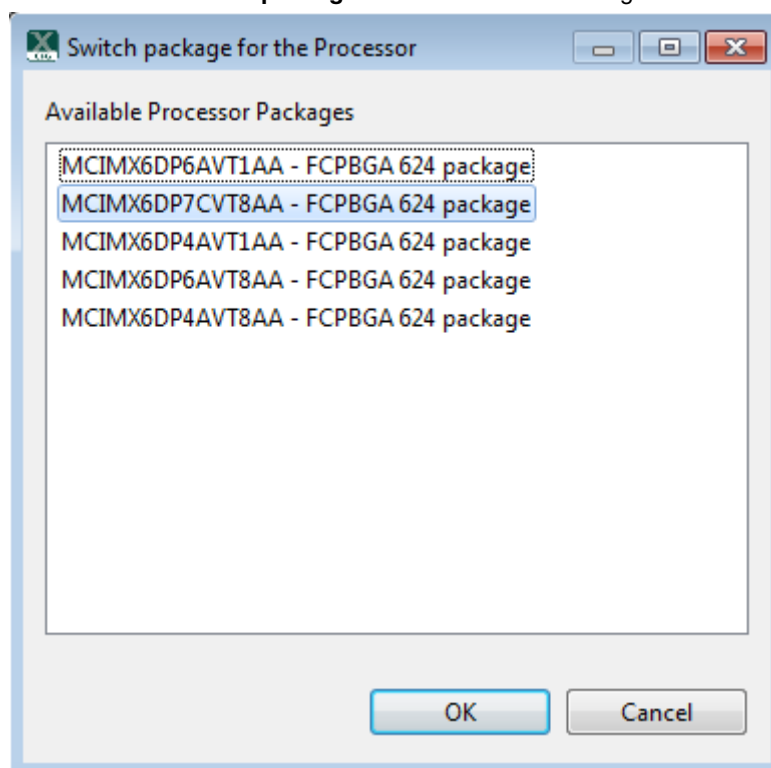


Figure 34. Switch package

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

4.4.2 Routed Pins view

The **Routed Pins** view shows a list of routed pins and allows 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 configured. Each row represents configuration of one 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 select required signal, and finally select the routed pin.

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




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

#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Open drain	Drive strength	Pull select
1	GPIOE	GPIO, 0	PTE0		<i>n/a</i>	Output	Fast	Disabled	Low	Pulldown
2	GPIOE	GPIO, 1	PTE1		<i>n/a</i>	Not Specified	Fast	Disabled	Low	Pulldown
5	UART3	TX	UART3_TX		<i>n/a</i>	Not Specified	Fast	Disabled	Low	Pulldown
6	UART3	RX	UART3_RX		<i>n/a</i>	Input	Fast	Disabled	Low	Pulldown
39	I2S0	TX_BCLK	I2S0_TX_B...		<i>n/a</i>	Not Specified	Fast	Disabled	High	Pullup
31	ADC0	SE, 17	ADC0_SE17		<i>n/a</i>	Input	Fast	Disabled	Low	Pulldown
83	FB	RW	FB_RW_b		<i>n/a</i>	Output	Fast	Disabled	Low	Pulldown

Figure 35. 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.

	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 menu to reset the value to its after-reset state.	
	If an item does not support reset to after reset value, or current value already represents after reset state, the Reset menu is not available.

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

4.4.3 View controls

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

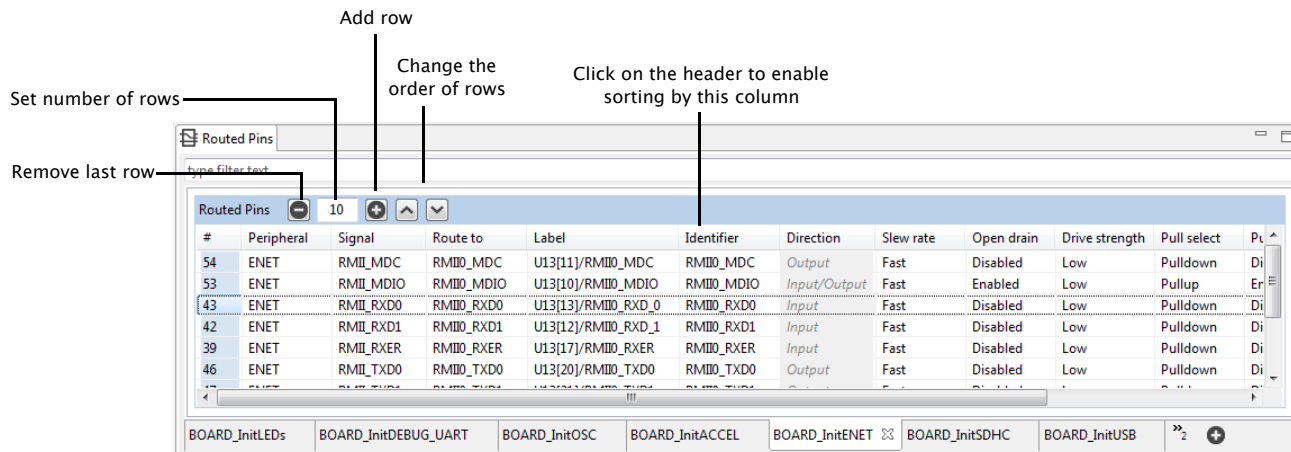


Figure 36. View controls

Add / remove rows:

- To add a new row to the end of table, click on the [+] button.
- To remove the last row, click on the [-] 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.

4.4.4 Filtering routed pins

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

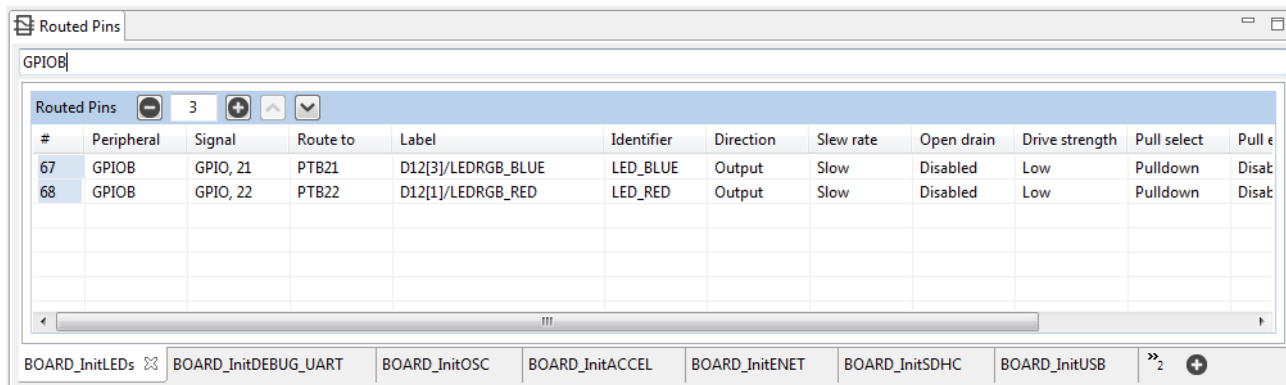


Figure 37. Filter area

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

NOTE

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

4.4.5 Highlighting and color coding

It is 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 selected pin/peripheral is highlighted by yellow color in the Package view. If the selected pin/peripheral is routed 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.

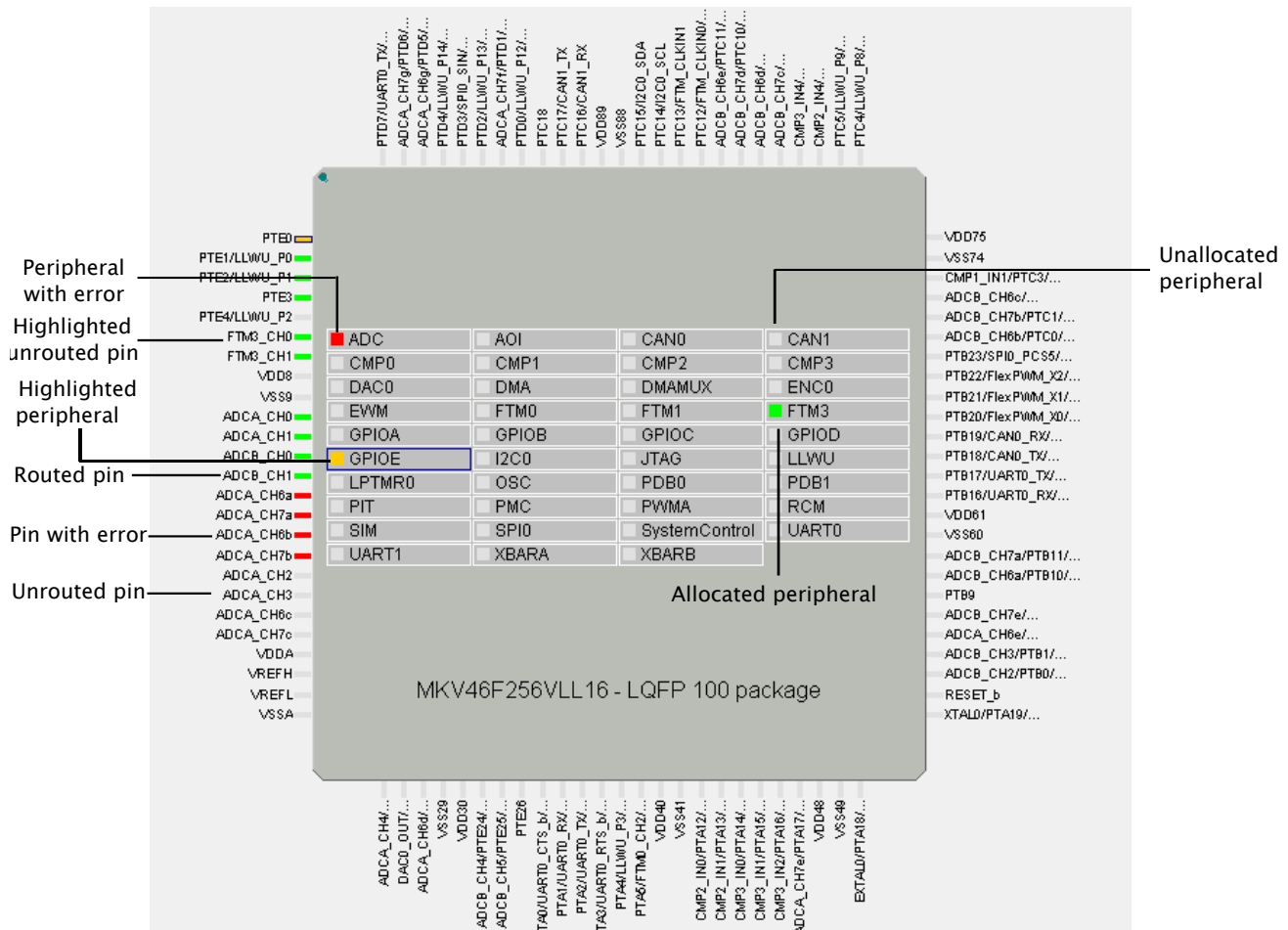


Figure 38. Highlighting and color coding

type filter text

Routed Pins 6

#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Open drain	Drive strength	Pull sel
32	GPIOE	GPIO, 25	PTE25	J2[18]/U8[6]/I2C0_SDA	Not Specif...	Not Specified	Fast	Disabled	Low	Pulldown
32	I2C0	SDA	I2C0_SDA	J2[18]/U8[6]/I2C0_SDA	ACCEL_SDA	Input/Output	Fast	Enabled	Low	Pulldown
	I2C0	SCL			n/a	n/a	Fast	Enabled	Low	Pulldown
	GPIOC				n/a	n/a	Fast	Enabled	Low	Pullup
85	GPIOC	GPIO, 13	PTC13	U8[9]	ACCEL_INT2	Input	Fast	Enabled	Low	Pullup
					n/a	n/a	n/a	n/a	n/a	n/a

BOARD_InitLEDs BOARD_InitDEBUG_UART BOARD_InitOSC BOARD_InitACCEL BOARD_InitENET BOARD_InitSDHC

Figure 39. Pins conflicts

Routed Pins 4

#	Peripheral	Signal	Route to	Label	Identifier	Power group	Direction	Software Input On Fi
U.	ECSP11	miso	KEY_COL1	CSPI1_MISO	SPI_IN	NVCC_GPIO (3.3V)	Input/Output	0b0: Disabled
P..	ECSP11	mosi	CSIO_DAT5	AUD3_TXD	AUD_TXD	NVCC_CSI (1.8V)	Input/Output	0b0: Disabled
W	ECSP11	sclk	KEY_COL0	CSPI1_CLK	SPI_CLK	NVCC_GPIO (3.3V)	Input/Output	0b0: Disabled
U.	ECSP11	ss0	KEY_ROW1	CSPI1_CS0	SPI_CS0	NVCC_GPIO (3.3V)	Output	0b0: Disabled

Figure 40. Warnings

- Package view
 - Click on the peripheral or use the pop-up menu to highlight peripherals:
 - and all allocated pins (to selected peripheral).
 - or all available pins if nothing is allocated yet.
 - Click on the pin or use the pop-up menu to highlight the pin and the peripherals.
 - Click outside the package to cancel the highlight.
- Peripherals / Pins view
 - The peripheral and pin behaves as described above image. See [Highlighting and color coding](#) and [Pins conflicts](#).
 - The keyboard can be used for selection. This works only for Desktop version.

Pins Tool
User interface

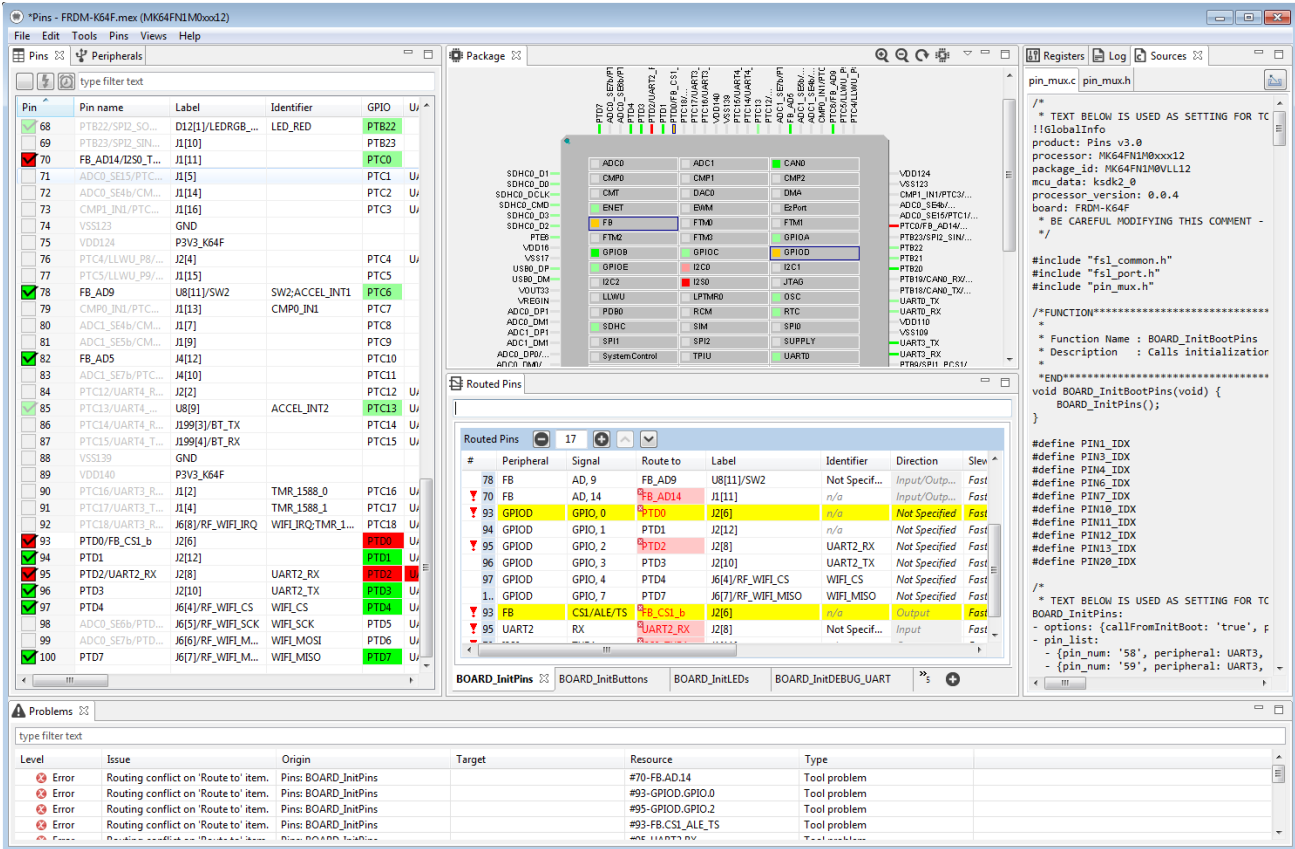


Figure 41. Pins Tool interface

Desktop version of the tool highlights pins during the drop-down menu traversal on pins

4.4.6 Filtering in the Pins view

The following image illustrates the filtering controls in the Pins view.

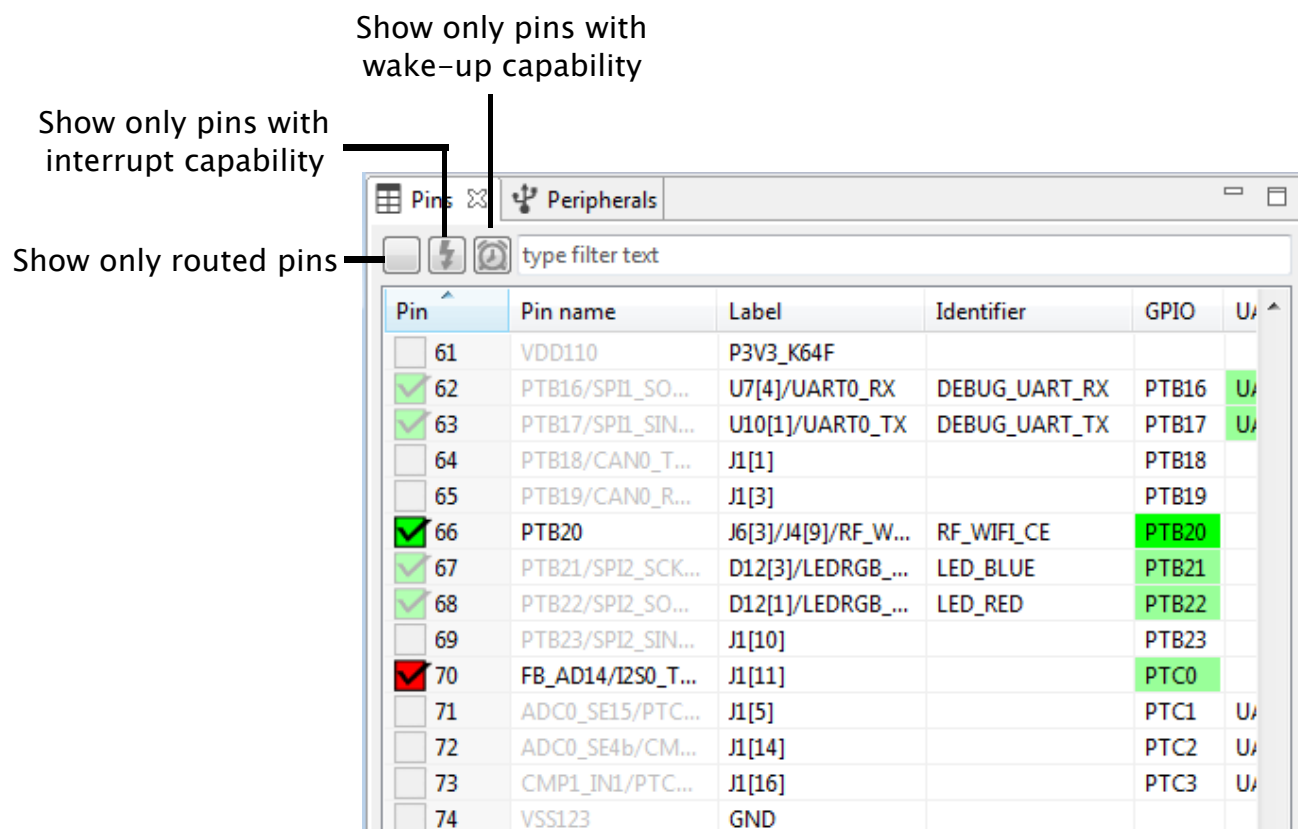


Figure 42. Filtering Controls

Type any text to search across the table. It will search for the pins features, like wake-up feature, or the low-power feature.

To see only pins on the selected peripheral, click on the header to sort pins for given peripheral by name.

Click on the first checkbox icon to see all or the routed pins only.

4.4.7 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 creates multiple functions that can be used to configure pin muxing.

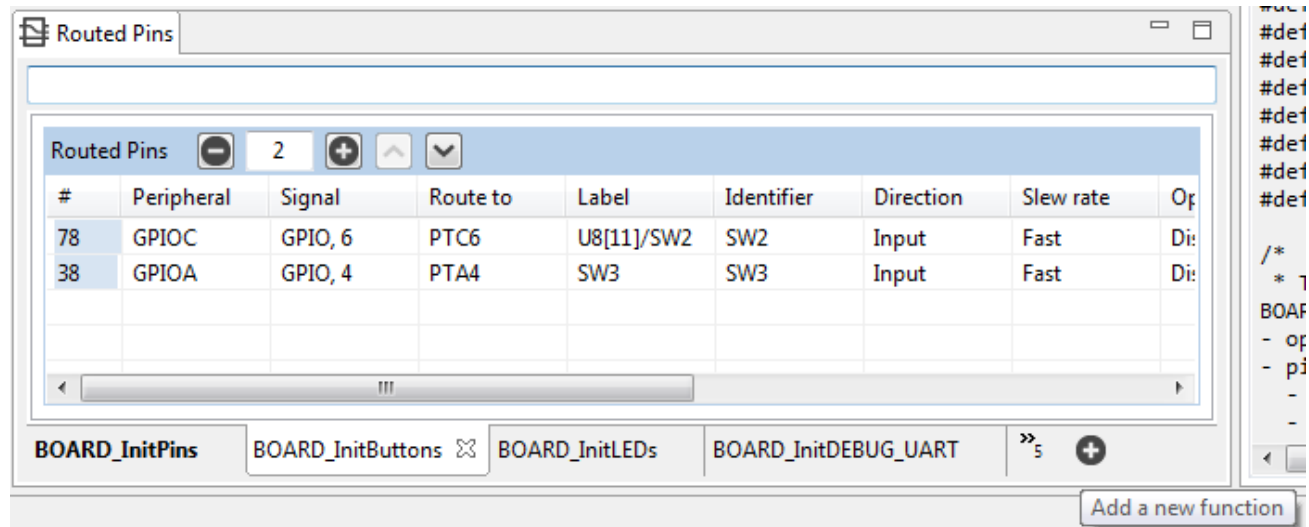


Figure 43. Routed Pins view - Add new function

The usage of pins is indicated by 50% opacity in **Pins**, **Peripherals**, 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.

Right-click on the function tab to show a context menu with the following commands:

- **Call function from BOARD_InitBootPins** - Sets the function which is called from the default initialization function BOARD_InitBootPins.
- **Delete** - Removes the function. It is available only if more than one function is present.
- **Properties** - Invokes a dialog, and allows you to change the function properties. For details, see [Pin properties](#).

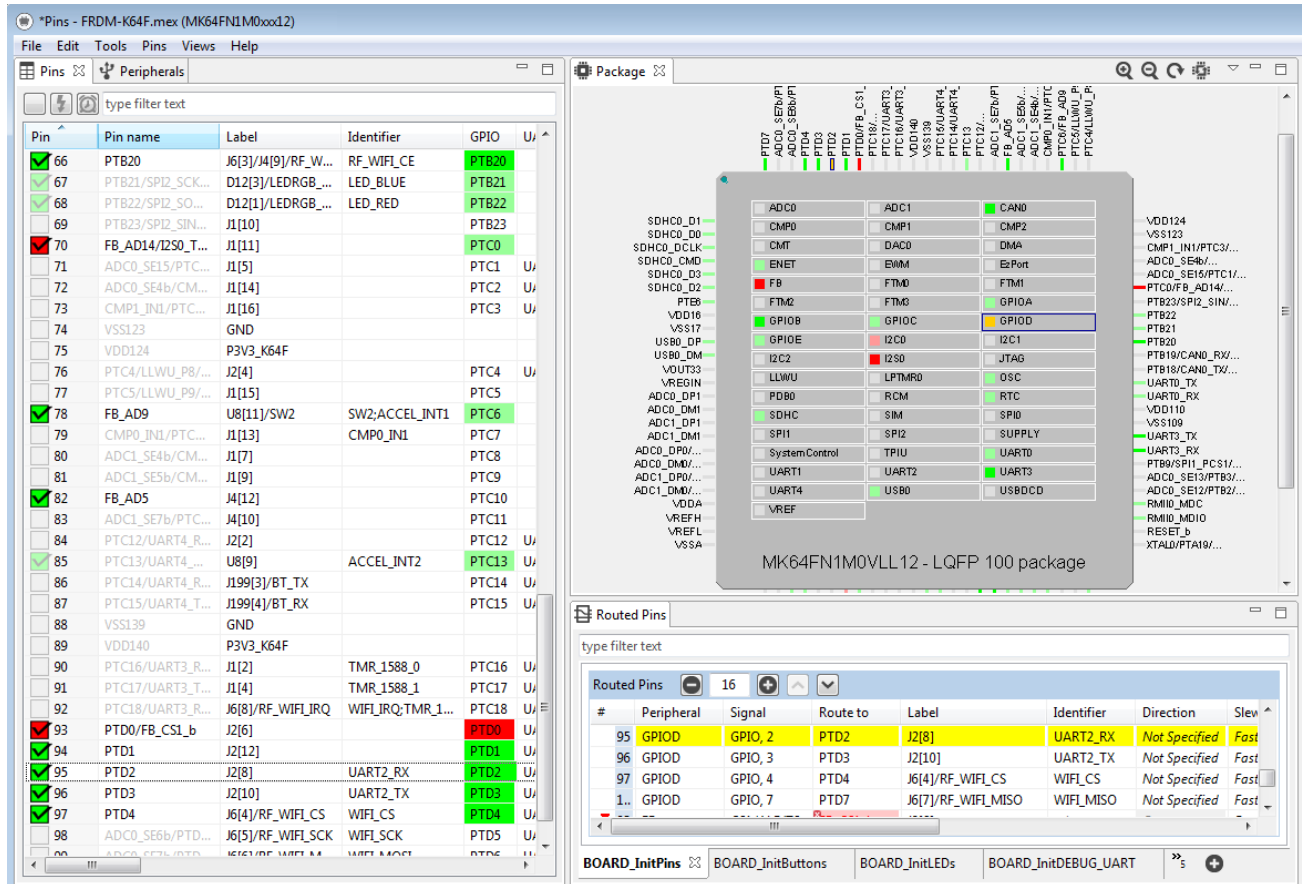






Figure 44. Pins and peripherals

4.4.8 Peripherals view

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

Table 5. Status codes

Color code	Status
 port, 1	Error
 port, 2	Allocated
 port, 3	Available
 port, 16	Warning

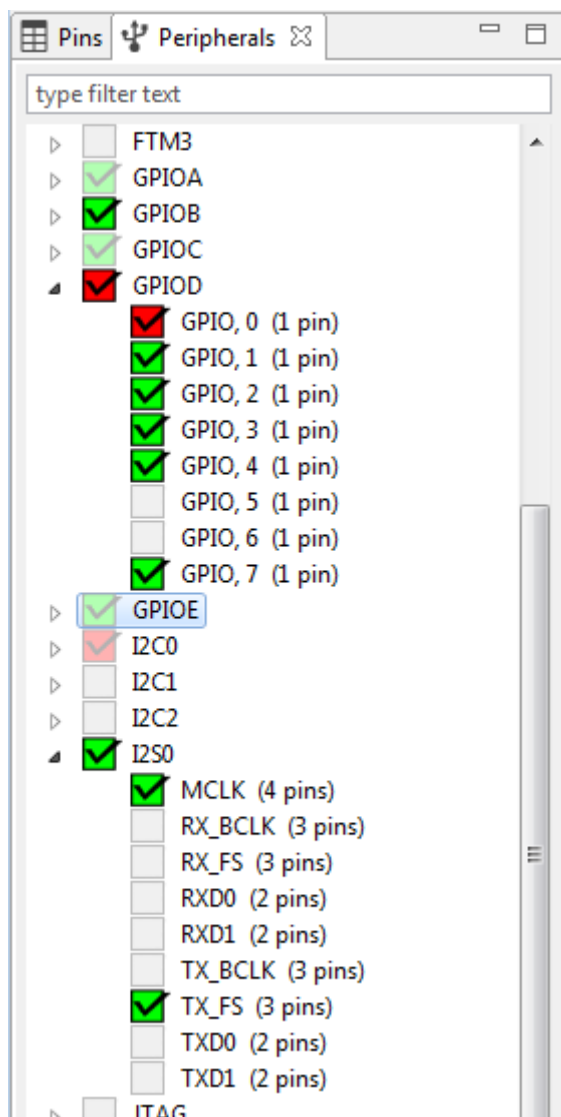


Figure 45. Peripherals view

Use the checkbox to route/unroute the selected pins.

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

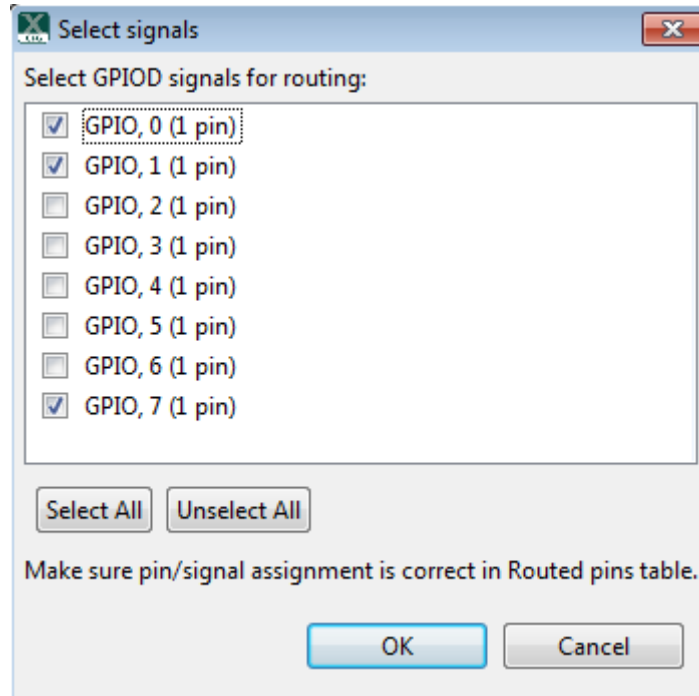


Figure 46. Select signals dialog

4.4.9 Pins table view

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

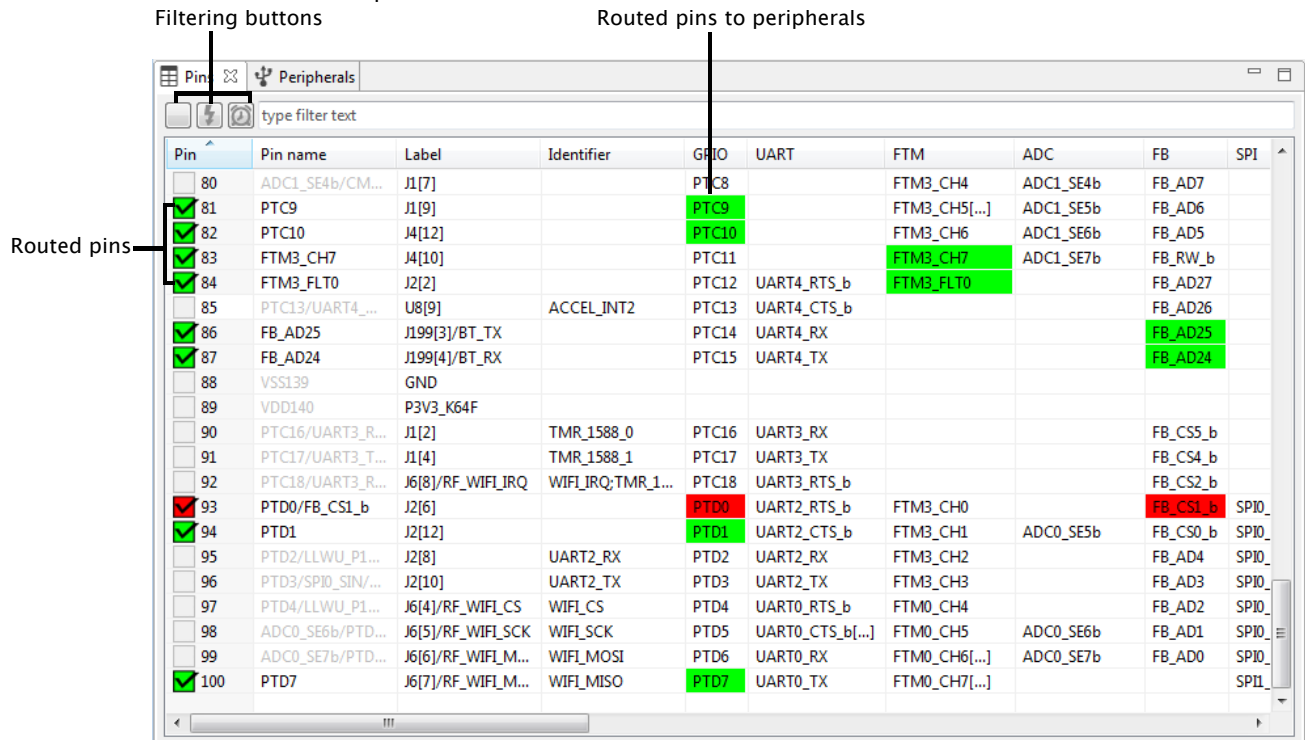


Figure 47. Pins table view

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

To route/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.

Pin	Pin name	Label	Identifier	GPIO	UART	FTM	ADC	FB	SPI
80	ADC1_SE4b/CM...	J1[7]		PTC8		FTM3_CH4	ADC1_SE4b	FB_AD7	
81	PTC9	J1[9]		PTC9		FTM3_CH5[...]	ADC1_SE5b	FB_AD6	
82	PTC10	J4[12]		PTC10		FTM3_CH6	ADC1_SE6b	FB_AD5	
83	FTM3_CH7	J4[10]		PTC11		FTM3_CH7	ADC1_SE7b	FB_RW_b	
84	FTM3_FLT0	J2[2]		PTC12	UART4_RTS_b	FTM3_FLT0		FB_AD27	
85	PTC13/UART4_...	U8[9]	ACCEL_INT2	PTC13	UART4_CTS_b			FB_AD26	
86	FB_AD25	J199[3]/BT_TX		PTC14	UART4_RX			FB_AD25	
87	FB_AD24	J199[4]/BT_RX		PTC15	UART4_TX			FB_AD24	
88	VSS139	GND							
89	VDD140	P3V3_K64F							
90	PTC16/UART3_R...	J1[2]	TMR_1588_0	PTC16	UART3_RX			FB_CS5_b	
91	PTC17/UART3_T...	J1[4]	TMR_1588_1	PTC17	UART3_TX			FB_CS4_b	
92	PTC18/UART3_R...	J6[8]/RF_WIFI_IRQ	WIFI_IRQ;TMR_1...	PTC18	UART3_RTS_b			FB_CS2_b	
93	PTD0/FB_CS1_b	J2[6]		PTD0	UART2_RTS_b	FTM3_CH0		FB_CS1_b	SPI0_
94	PTD1	J2[12]		PTD1	UART2_CTS_b	FTM3_CH1	ADC0_SE5b	FB_CS0_b	SPI0_
95	PTD2/LLWU_P1...	J2[8]	UART2_RX	PTD2	UART2_RX	FTM3_CH2		FB_AD4	SPI0_
96	PTD3/SPI0_SIN/...	J2[10]	UART2_TX	PTD3	UART2_TX	FTM3_CH3		FB_AD3	SPI0_
97	PTD4/LLWU_P1...	J6[4]/RF_WIFI_CS	WIFI_CS	PTD4	UART0_RTS_b	FTM0_CH4		FB_AD2	SPI0_
98	ADC0_SE6b/PTD...	J6[5]/RF_WIFI_SCK	WIFI_SCK	PTD5	UART0_CTS_b[...]	FTM0_CH5	ADC0_SE6b	FB_AD1	SPI0_
99	ADC0_SE7b/PTD...	J6[6]/RF_WIFI_M...	WIFI_MOSI	PTD6	UART0_RX	FTM0_CH6[...]	ADC0_SE7b	FB_AD0	SPI0_
100	PTD7	J6[7]/RF_WIFI_M...	WIFI_MISO	PTD7	UART0_TX	FTM0_CH7[...]			SPI1_

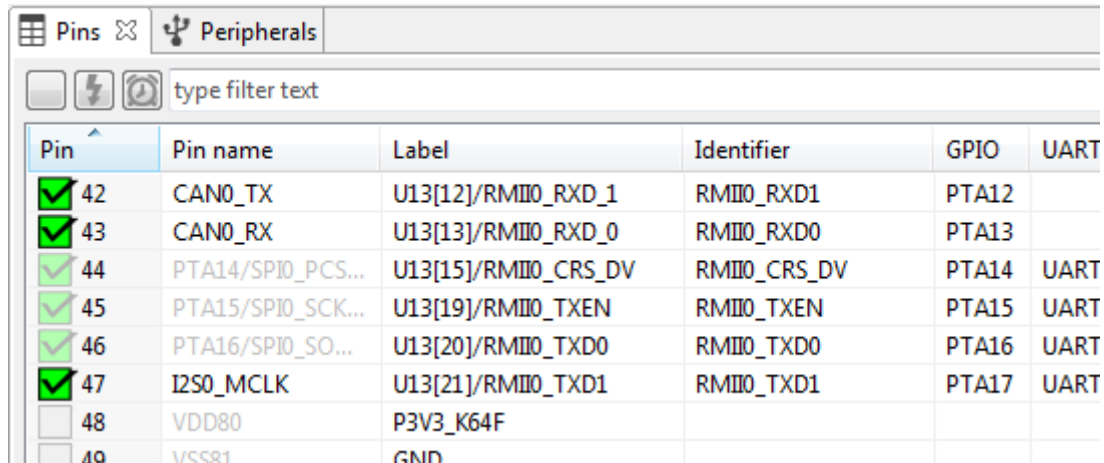
Figure 48. Example of routing multiple functions

If more signals can be routed to one pin, it is indicated by [...]. The **Multiple Signals Configuration** dialog appears, if clicked.

4.4.9.1 Labels and identifiers

It is 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**.



Pin	Pin name	Label	Identifier	GPIO	UART
<input checked="" type="checkbox"/> 42	CAN0_TX	U13[12]/RMII0_RXD_1	RMII0_RXD1	PTA12	
<input checked="" type="checkbox"/> 43	CAN0_RX	U13[13]/RMII0_RXD_0	RMII0_RXD0	PTA13	
<input checked="" type="checkbox"/> 44	PTA14/SPI0_PCS...	U13[15]/RMII0_CRS_DV	RMII0_CRS_DV	PTA14	UART
<input checked="" type="checkbox"/> 45	PTA15/SPI0_SCK...	U13[19]/RMII0_TXEN	RMII0_TXEN	PTA15	UART
<input checked="" type="checkbox"/> 46	PTA16/SPI0_SO...	U13[20]/RMII0_TXD0	RMII0_TXD0	PTA16	UART
<input checked="" type="checkbox"/> 47	I2S0_MCLK	U13[21]/RMII0_TXD1	RMII0_TXD1	PTA17	UART
<input type="checkbox"/> 48	VDD80	P3V3_K64F			
<input type="checkbox"/> 49	VSS81	GND			

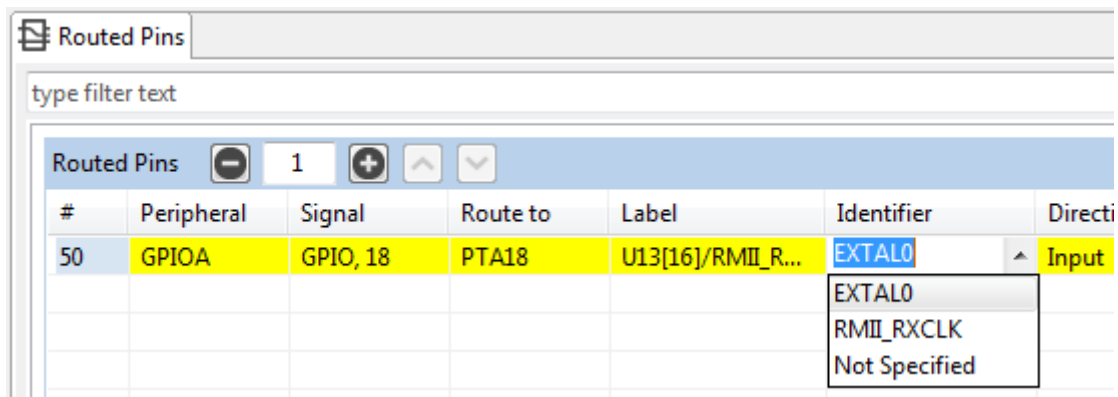
Figure 49. Labels and Identifiers

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. Additionally, you can define multiple identifiers, using the “;” character as a separator.

Pin	Pin name	Label	Identifier	GPIO
<input type="checkbox"/> 49	VSS81	GND		
<input type="checkbox"/> 50	EXTAL0/PTA18/...	U13[16]/RMII_RXCLK	EXTAL0;RMII_RXCLK	PTA18
<input type="checkbox"/> 51	XTAL0/PTA19/F...	GND		PTA19
<input type="checkbox"/> 52	RFSFT h	I3[61]/I9[101]/D1/RFSFT	RFSFT	

Figure 50. Pin Identifier

In this case it is possible to select from values if the pin is routed. See [Routed pins table](#).



#	Peripheral	Signal	Route to	Label	Identifier	Direction
50	GPIOA	GPIO, 18	PTA18	U13[16]/RMII_R...	EXTAL0 RMII_RXCLK Not Specified	Input

Figure 51. Identifier in Routed Pins table

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

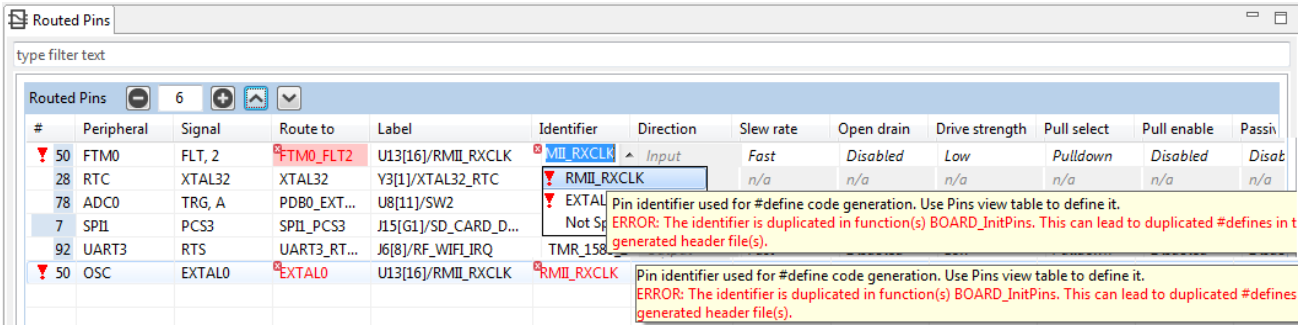


Figure 52. Identifier errors

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.

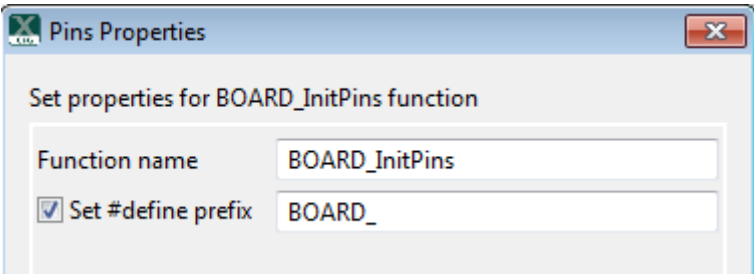


Figure 53. Pins macros prefix

If multiple functions are used, each individual function can include a special prefix. Check the **Set #define prefix** input box 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.

4.5 Errors and warnings

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

#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Ope
1	ADC1	SE, 4a	ADC1_SE4a	J15[P8]/SDHC0_D1	Not Specif...	Input	Fast	Disa
1	I2C1	SDA	I2C1_SDA	J15[P8]/SDHC0_D1	Not Specif...	Input/Outp...	Fast	Disa
10	USBDCD	DP	USB0_DP	J22[3]/K64_MICRO_...	Not Specif...	Input/Outp...	n/a	n/a
4	GPIOE	GPIO, 3	PTE3	J15[P3]/SDHC0_CMD	Not Specif...	Output	Fast	Disa

Figure 54. Error and warnings

If an error or warning is encountered, the conflict 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.

4.5.1 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.

#	Peripheral	Signal	Route to	Label	Identifier	Direction	Slew rate	Ope
1	I2C1	SDA	I2C1_SDA	J15[P8]/SDHC0_D1	Not Specif...	Input/Outp...	Fast	Disa
10	USBDCD	DP	USB0_DP	J22[3]/K64_MICRO_...	Not Specif...	Input/Outp...	n/a	n/a
19	ADC0				n/a	n/a	n/a	n/a

Figure 55. Incomplete routing

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

4.6 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 the main menu **Pins > Refresh**.

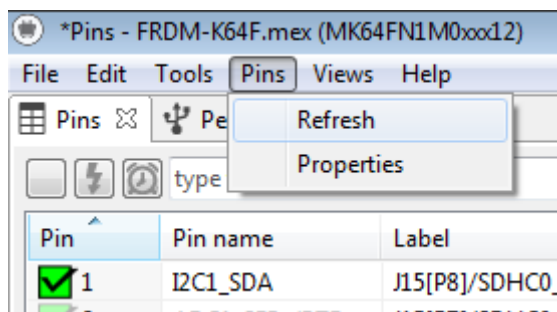


Figure 56. Pins > Refresh

The generated code is shown in the **Sources** tab on the right window. It shows all generated files and each file has its own tab.

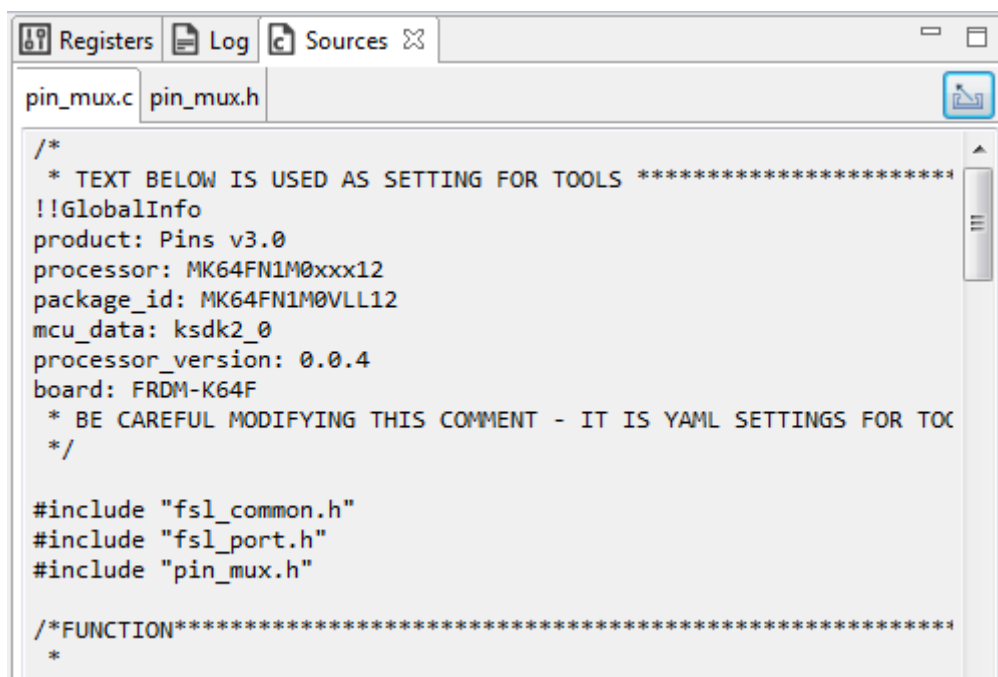


Figure 57. Sources view

For multicores, the sources are generated for each core. Appropriate files are shown with @Core #1 tag.

NOTE

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

It is also possible to 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 YAML format. This comment is not intended for direct editing and can be used later to re-store the pins configuration.



```

/*
 * TEXT BELOW IS USED AS SETTING FOR TOOLS *****
!!GlobalInfo
product: Pins v3.0
processor: MK64FN1M0xxx12
package_id: MK64FN1M0VLL12
mcu_data: ksdk2_0
processor_version: 0.0.4
board: FRDM-K64F
 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS *****
 */

#include "fsl_common.h"
#include "fsl_port.h"
#include "pin_mux.h"

/*FUNCTION*****
 *
 * Function Name : BOARD_InitBootPins
 * Description   : Calls initialization functions.
 *
 *END*****/
void BOARD_InitBootPins(void) {
    BOARD_InitPins();
}

#define PIN0_IDX          0u /*!< Pin number for pin 0 in a port */

/*
 * TEXT BELOW IS USED AS SETTING FOR TOOLS *****
BOARD_InitPins:
- options: {callFromInitBoot: 'true', prefix: BOARD_, coreID: core0, enableClock: 'true'}
- pin_list:
  - {pin_num: '1', peripheral: I2C1, signal: SDA, pin_signal: ADC1_SE4a/PTE0/SPI1_PCS1/UART1_TX/SDHC0_D1/TRACE_CLK
  - {pin_num: '10', peripheral: USBDCD, signal: DP, pin_signal: USB0_DP, identifier: ''}
  - {pin_num: '19', peripheral: ADC0, signal: '', pin_signal: ADC0_DM0/ADC1_DM3}
 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS *****
 */

/*FUNCTION*****
 *
 * Function Name : BOARD_InitPins
 * Description   : Configures pin routing and optionally pin electrical features.
 *
 *END*****/
void BOARD_InitPins(void) {
    /* There are conflicts or other incorrect settings in the configuration, the code below is generated only for
       those registers which are set correctly and without a conflict. Open this file in Pins Tool for more details.


    CLOCK_EnableClock(kCLOCK_Porte);                /* Port E Clock Gate Control: Clock enabled */

    PORT_SetPinMux(PORTE, PIN0_IDX, kPORT_MuxAlt6);    /* PORTE0 (pin 1) is configured as I2C1_SDA */
}

```

Figure 58. Generated code

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

	<p>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.</p>
---	--

4.6.1 Exporting source code

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

To launch the Export wizard:

1. Select **File > Export** from the main menu.
2. Select the **Export Source Files** option.

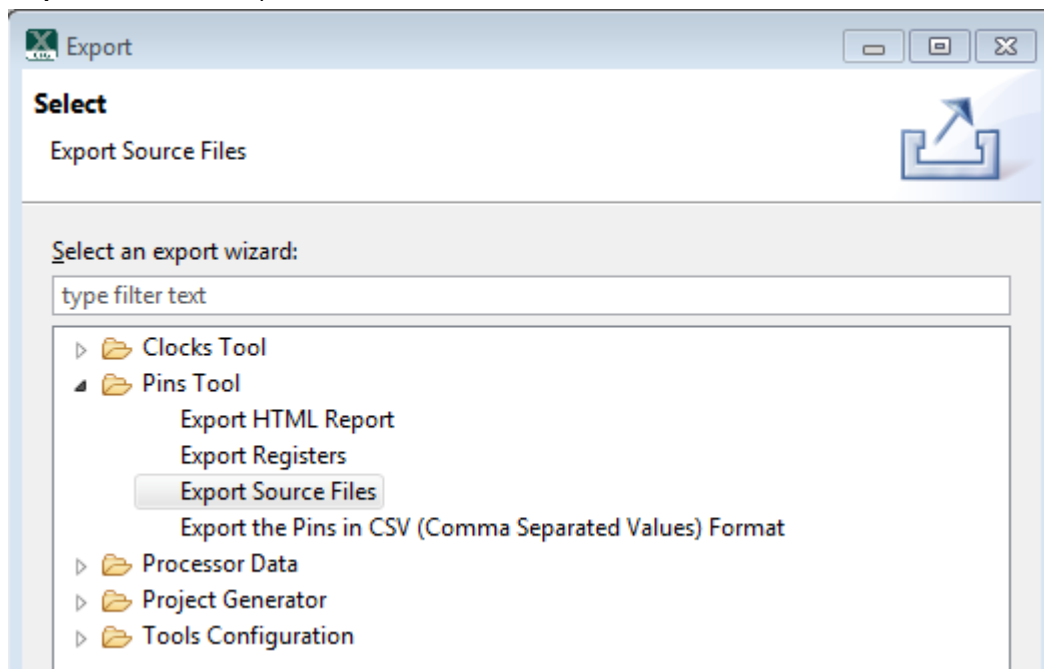


Figure 59. Export wizard

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

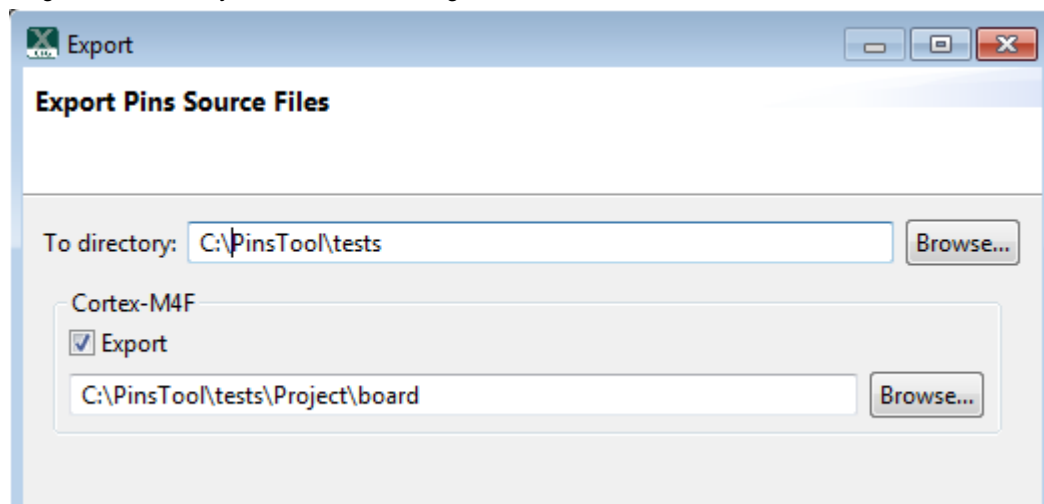


Figure 60. Select target folder

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

4.6.2 Importing source code

To import source code files:

1. Select **File > Import** from the main menu.

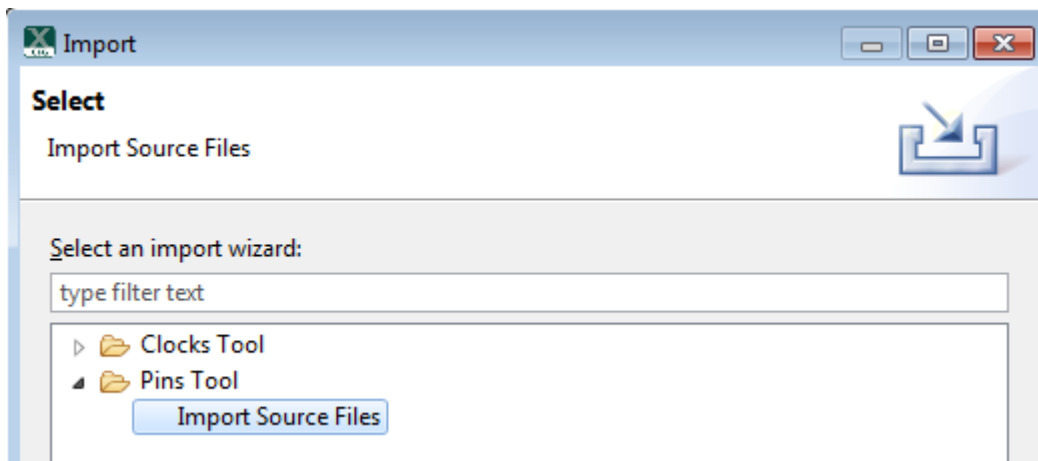


Figure 61. Import sources wizard

2. Select the **Import Source Files** option.
3. Click **Next**.
4. It is possible to select one or more C files to import using the **Browse** button in the **Import Pins Source Files** dialog.

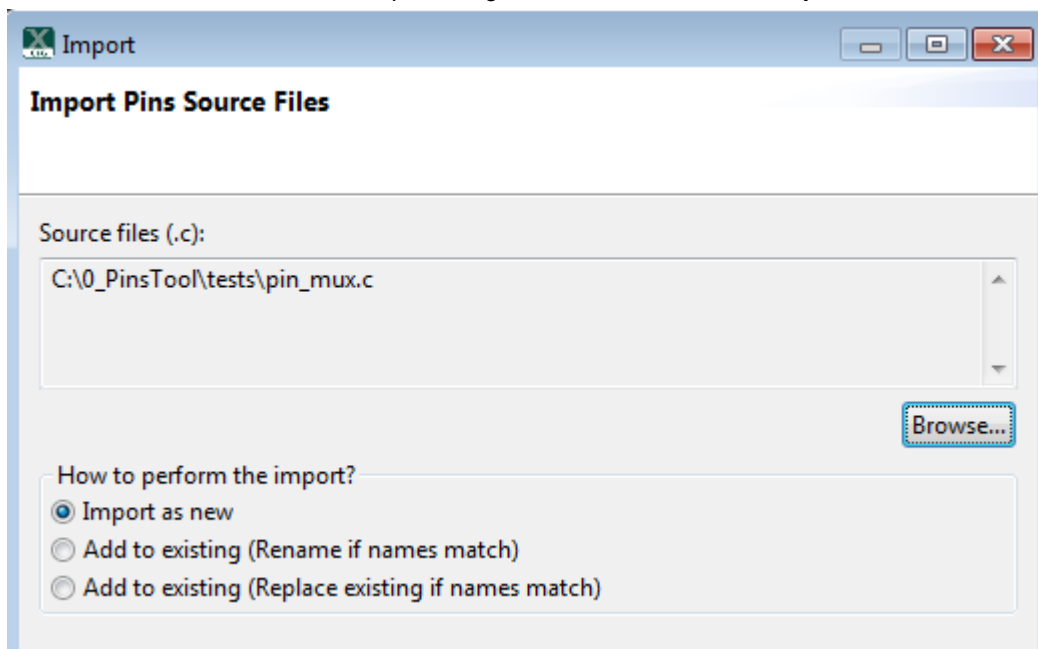



Figure 62. Import Pins Source Files

5. Select how to import the files:
 - **Import as new** - The current configuration is dropped and the files are imported.
 - **Add to existing (Rename)** - All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, it is automatically renamed to the indexed one. For example, if BOARD_InitPins already exists in the configuration then the imported function is renamed to BOARD_InitPins1.

- **Add to existing (Replace)** - All files are merged into the current configuration. It imports all the functions only. If the imported function has the same name as an existing one, then the existing one is replaced with the imported one.

6. Click **Finish**.

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

4.7 Options

4.7.1 Pins properties

To set pins properties, select **Pins > Properties** from the main menu. The **Pins Properties** dialog appears.

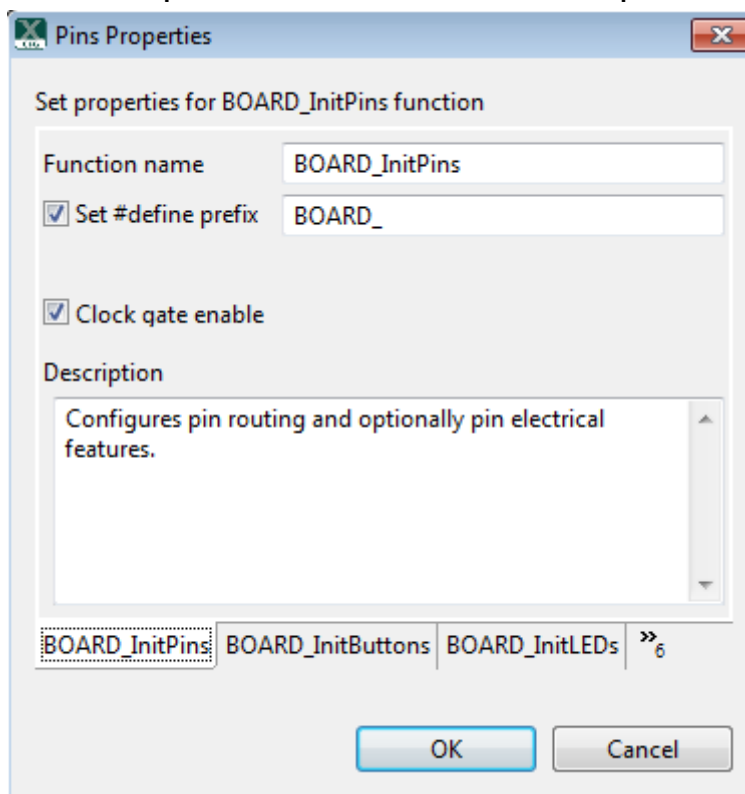


Figure 63. Pins Properties

In this option it is possible to configure several options for functions and code generation. Each settings is applicable for selected function, which can be specified by selecting appropriate tab. It is possible to 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 #define prefix: If enabled, it uses the specified prefix for the identifiers in the source code. [\[1\]](#)

[1] *if supported by processor

Chapter 5

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.

5.1 Features

The following are the Clock tool features:

- 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 selected 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.

5.2 Workflow

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

1. Select **Tools > Clocks** from the tools framework main menu to open the Clocks tool.

The Clocks tool is available in both the Desktop and Web versions.

2. Set the global settings in the bottom-left part. For example: power modes and MCG mode.
3. Enable the external sources, if they are available on your board using checkboxes and setting frequencies.
4. Check and adjust the clock selectors (multiplexers).

NOTE

The tool does not change the clock selectors automatically.

5. Specify the frequencies of the required clock outputs.

- or -

Change the individual pre-scalers/PLL/FLL parameters.

6. Resolve the reported issues, if any.
7. Open the **Sources** view and see the output source code.

NOTE

The source code is automatically generated if no errors are reported.

8. Export the source code.

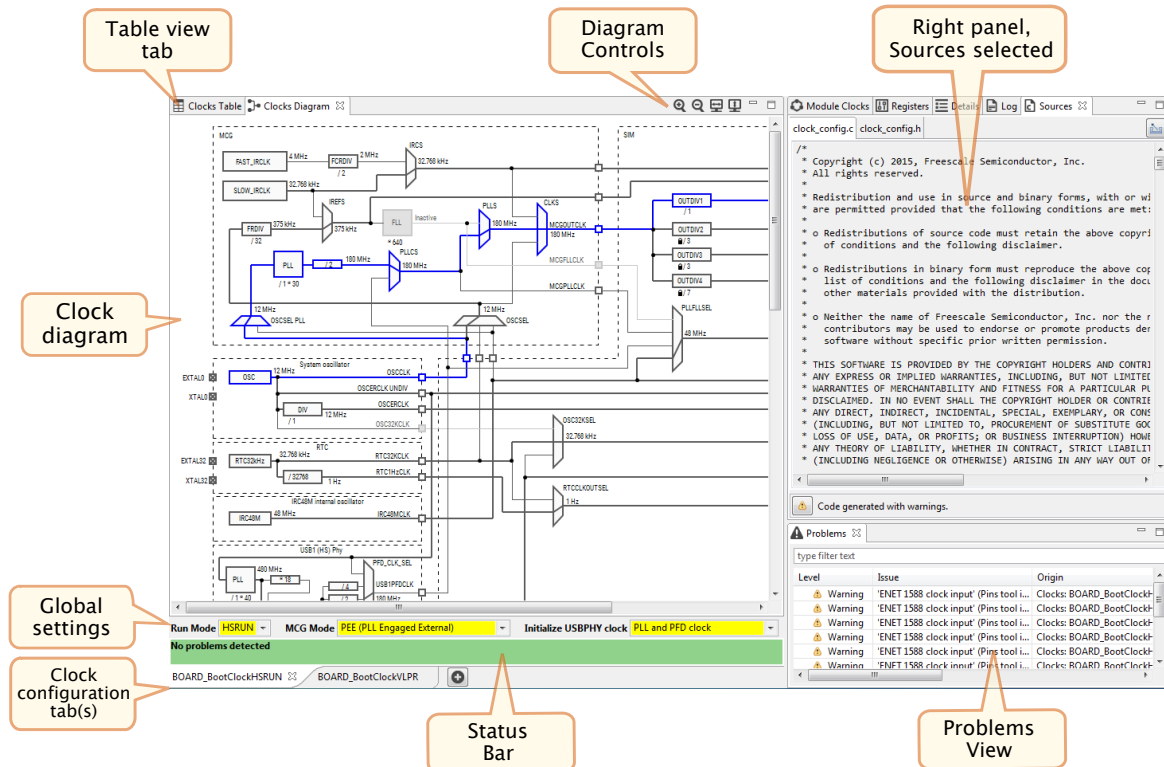
- For the *Desktop* version: Select **File > Export** from the main menu.
- For the *Web* version: Select **Clocks > Export** from the main menu.

NOTE

To export the source code, you can also click the **Export** button located in **Source** view. The **Export** button is available in both the Desktop and Web versions.

5.3 User interface overview

The tool is integrated and runs with the MCUXpresso Configuration tools framework. For documentation on the common framework and menu items, see the common framework section.



5.4 Clock configuration

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

There can be one or more clock configurations handled by the Clocks tool. The default clock configuration is created with the name "BOARD_BootClockRUN". Multiple configurations 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.

Clock configuration are presented as tabs at the bottom of the view. You can switch between these clock configurations and add more configurations using the '+' button.

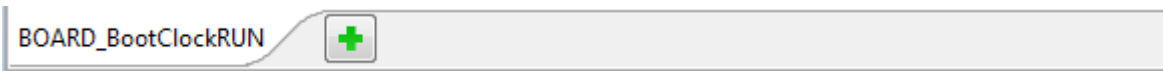


Figure 65. Default clock configuration

NOTE

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

Right-click on the **Clock configuration** tab to show a context menu with the following commands:

- **Properties** - Invokes a dialog, and allows you to change the name and the description of the clock configuration.
- **Copy** - Creates a copy of the current/active clock configuration.
- **Remove** - Removes the clock configuration. It is available only if more than one clock configuration present.
- **Call function from BOARD_InitBootClocks** - Sets the function, which is called from the default initialization function BOARD_InitBootClocks.

5.5 Global settings

The global settings are the settings that influence the entire clock system. It is recommended to start with these settings, but they can be changed later.

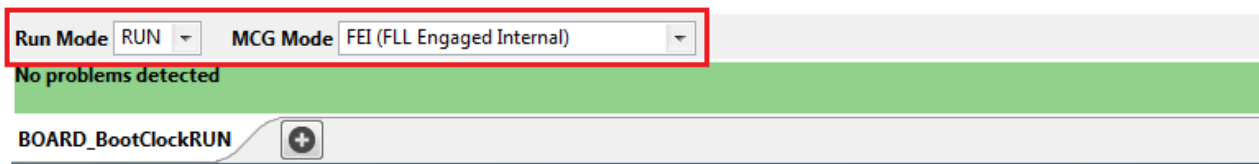


Figure 66. Global settings

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

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

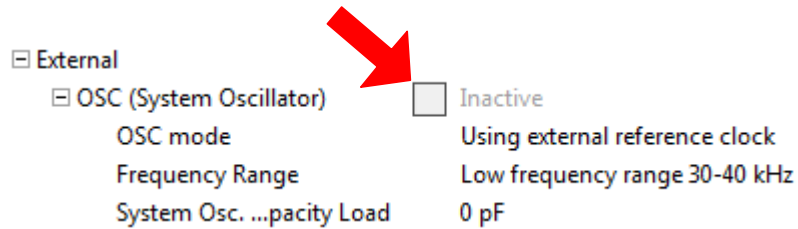


Figure 67. External clock source configuration

NOTE

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

5.7 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 6. 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 tool-tip of the setting for details.
Lock icon		Indicates that the settings (that may be automatically adjusted by the tool) are locked to prevent any automatic adjustment. If the setting can be locked, they are automatically locked when you change the value. To add/remove the lock manually, use the pop-up menu command Lock/Unlock . <div> <p>NOTE</p> <p>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.</p> </div>
Yellow background		Indicates that the field is directly or indirectly changed by the previous user action.

Table continues on the next page...

Table 6. Setting states and markers (continued)

State/Style/ Marker	Icon	Description
Gray text	FCTRIM	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.

5.8 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 value shown 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 with ellipsis ‘...’ character 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 the **Enter** key.



Figure 68. Setting the core clock frequency

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



Figure 69. Tool attains the required frequency

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



Figure 70. Tool encounters problem

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

NOTE

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

5.8.1 Pop-up menu commands

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



Figure 71. Pop-up menu commands

5.8.2 Frequency precision

For the locked frequency settings (user indicated a requested value) the frequency precision value is also shown. By default, the value is 0.1% but can be individually adjusted by clicking on the value.

Name	Lock	Value	Accuracy
Core clock		21 MHz [20.97... MHz]	±5%

Figure 72. Frequency precision

5.9 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.

Clock Sources					
Name	A...	Value			
FAST_IRCLK		4 MHz			
SLOW_IRCLK		32.768 kHz			
IRC48M		Inactive			

	Core clock	20.97... MHz
	System clock	20.97... MHz
	Bus clock	20.97... MHz
	FlexBus clock	20.97... MHz
	Flash clock	10.48... MHz

Figure 73. Dependency arrows

5.10 Details view

The **Details** view contains a list of settings on the selected element, clock path, component, or on the entire processor.

The content of the **Details** view depends on the selected element and can be one of the following.

- **Clock element** - Lists the settings of the selected clock source, prescaler, and so on.
- **Clock path** - Lists the settings of the element on the path from the selected output to used clock source.
- **Component** - Lists the settings for all elements located in the selected component.
- **Processor** - Lists all the settings related to the selected processor.

Component Details: OSC			
Name	A...	L...	Value
<input checked="" type="checkbox"/> OSC (System Oscillator)	<input type="checkbox"/>		Inactive
OSC mode			Using external reference clock
Frequency Range			Low frequency range 30-40 kHz
System Osc. Capacity Load			0 pF
<input checked="" type="checkbox"/> OSCERCLK Frequency			Inactive
OSCERCLK output			Disabled
OSCERCLK out...n Stop mode			Disabled
OSCCLK Frequency			Inactive
OSC32KCLK Frequency			Inactive

Figure 74. Details view

5.11 Clock diagram

The clock diagram shows the complete structure of the 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 reflects the current state of the clock model.

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

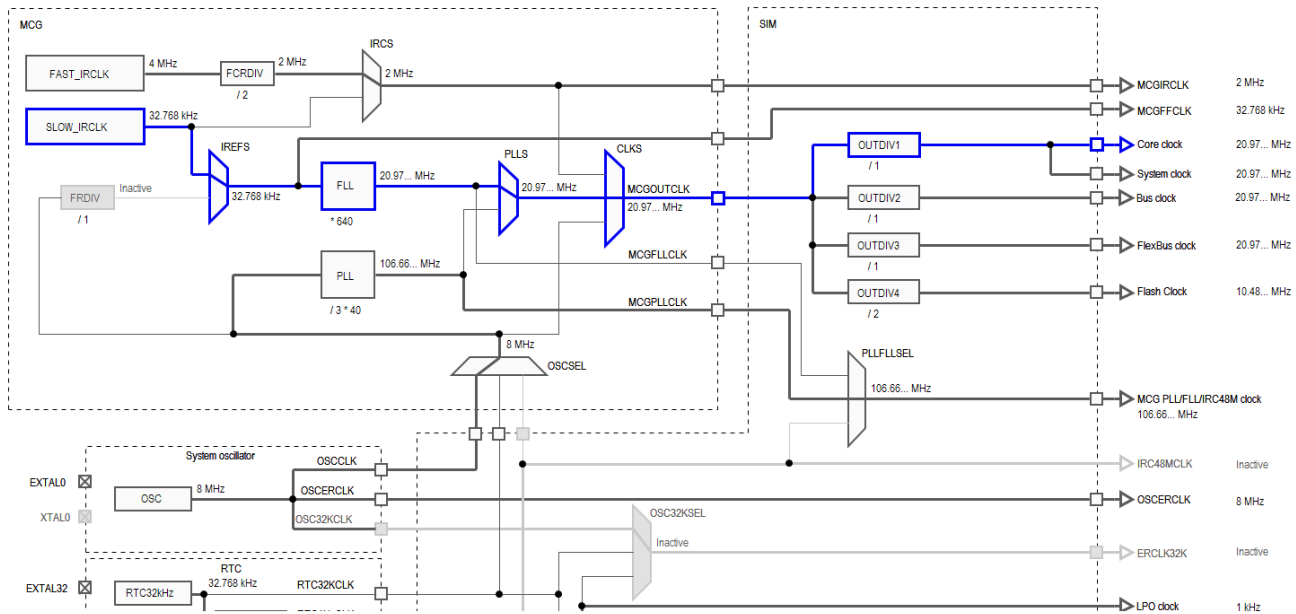


Figure 75. Clock diagram

5.11.1 Mouse actions in diagram

The following interactions are available in Clock diagram view.

- **Move 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.
- **Double-click on the element** to open the floating window with its settings.
- **Single-click on the element** to open the drop-down menu with the main setting of the element. The main setting is the most obvious setting for the clicked element. For instance, scale the setting for the prescaler or the branch selection for the multiplexer.

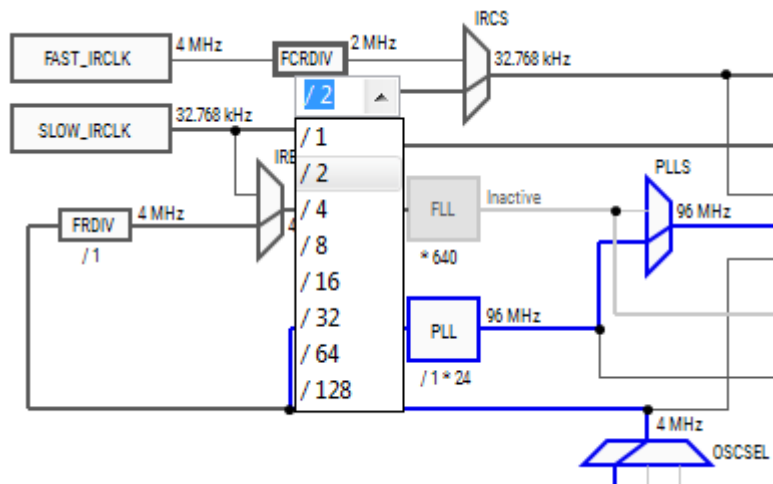


Figure 76. Clocks mouse actions in diagram

- **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.

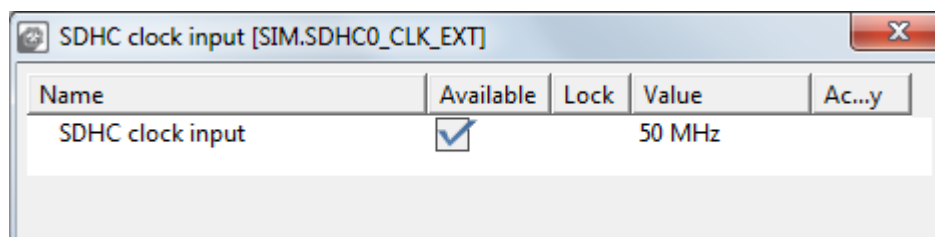


Figure 77. Floating view

5.11.2 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 get the exact colors and style appearance, select **Help > Show diagram legend** from the main menu.

5.11.3 Clock model structure

The clock model consists of the clock elements that are interconnected. The clock signal flows from the clock sources through the various clock elements to the clock outputs. The clock element can have specific enable conditions that can stop the signal from passing it to the successor. The clock element can also have specific constraints and limits that are watched by the clocks tool. To get these details, put the cursor on the element in the clock diagram and see its tooltip.

The following are the clock model elements.

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



Figure 78. Clock source

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

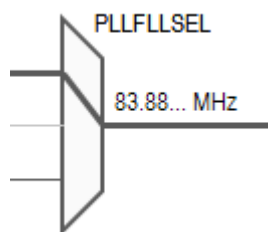


Figure 79. Clocks selector

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

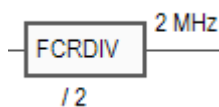


Figure 80. Prescaler

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

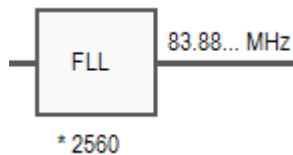


Figure 81. Frequency Locked Loop

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

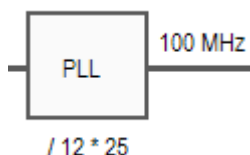


Figure 82. Phase Locaked Loop

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

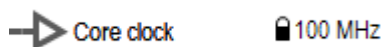


Figure 83. Clock output

- **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.

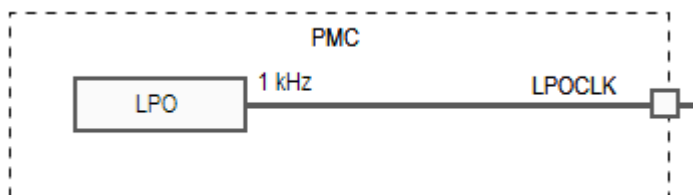


Figure 84. Clock component

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

5.12 Main menu

The commands related to Clocks are present in the **Clocks** menu and include the following commands:

- **Copy Configuration** – Creates a copy of the current/active clock configuration.
- **Configuration Properties** – Invokes a dialog, and allows you to change the name and the description of the clock configuration.
- **Call function from BOARD_InitBootClocks** - set the function to be called from the default initialization function BOARD_InitBootClocks.
- **Remove Configuration** – Enables you to remove the current shown clock configuration. However, it is available only if there is more than one clock configuration present.
- **Unlock All Settings** – Removes all locks in all settings.
- **Reset To Defaults** – Resets the clock model into the state corresponding to the initial after-reset state of the clocks.
- **Refresh** – Refreshes each clocks configuration with explicit invocation of code generation.

For description of other menus, see the common [Tools framework menu documentation](#).

5.13 Troubleshooting problems

It is possible that while working with the tool some problems or mismatches occur. Such problems and the overall status is indicated in red on the central status bar of the tool. The status bar displays the 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 some settings is not valid. For example:
The current state of settings is beyond the acceptable range.

The following are some tips to troubleshoot the encountered problems.

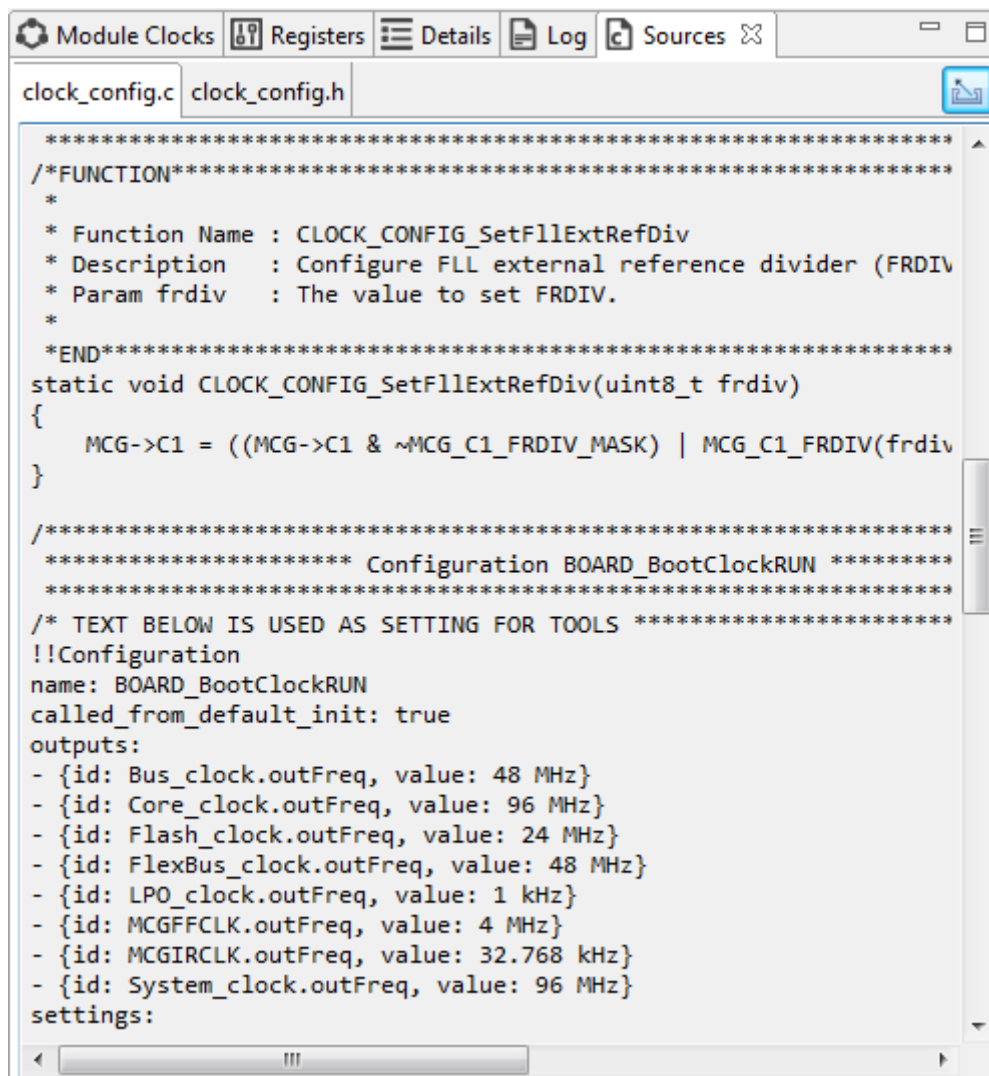
1. Find the elements and settings with marked errors in the diagram or tables and see the details in the tooltip.
2. Start with only one locked frequency and let the tool find and calculate other ones. After you are successful you can add more.
3. Go through the locked outputs, if there are any, and 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 is 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 defaults**.

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

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

5.14 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 **Sources** view.



```

Module Clocks Registers Details Log Sources
clock_config.c clock_config.h

/******
/*FUNCTION*****
*
* Function Name : CLOCK_CONFIG_SetFllExtRefDiv
* Description   : Configure FLL external reference divider (FRDIV
* Param frdiv   : The value to set FRDIV.
*
*END*****
static void CLOCK_CONFIG_SetFllExtRefDiv(uint8_t frdiv)
{
    MCG->C1 = ((MCG->C1 & ~MCG_C1_FRDIV_MASK) | MCG_C1_FRDIV(frdiv)
}

/*****
***** Configuration BOARD_BootClockRUN *****
*****
/* TEXT BELOW IS USED AS SETTING FOR TOOLS *****
!!Configuration
name: BOARD_BootClockRUN
called_from_default_init: true
outputs:
- {id: Bus_clock.outFreq, value: 48 MHz}
- {id: Core_clock.outFreq, value: 96 MHz}
- {id: Flash_clock.outFreq, value: 24 MHz}
- {id: FlexBus_clock.outFreq, value: 48 MHz}
- {id: LPO_clock.outFreq, value: 1 kHz}
- {id: MCGFFCLK.outFreq, value: 4 MHz}
- {id: MCGIRCLK.outFreq, value: 32.768 kHz}
- {id: System_clock.outFreq, value: 96 MHz}
settings:

```

Figure 85. Sources tab

5.14.1 Working with the code

The generated code is aligned with the selected SDK version; selected when the global configuration is created. 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:

1. 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.
2. User export command:
 - a. Select **File > Export**.
 - b. Select **Clocks Tool > Export Source Files**.
3. Click the **User export** button in **Sources** view.

5.14.2 Restoring clock configuration from source code

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

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

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

Figure 86. Setting Information in the source code

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

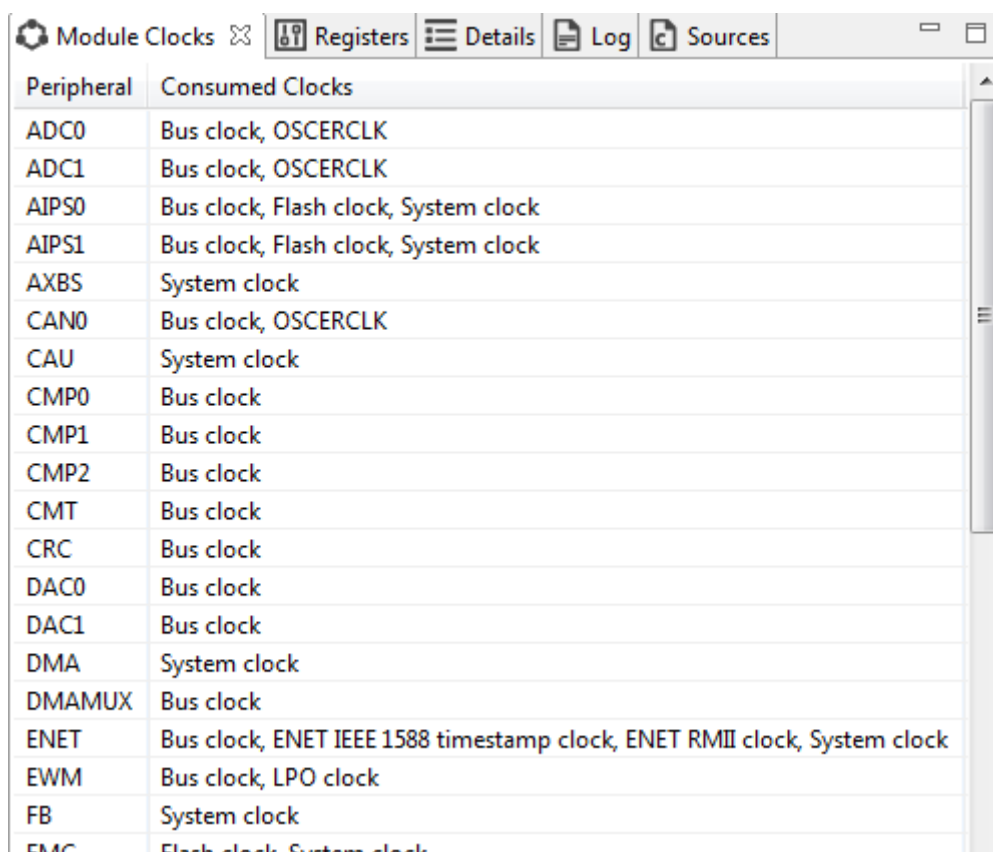
1. Select the command: **File > Import....**
2. Select **Clocks Tool / Import Source Files**.
3. Click **Next**.
4. Click **Browse**.
5. Navigate and select the *clock_config.c* file previously produced by the Clocks tool.
6. If the settings parse successfully, the clock configurations are added into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

5.15 Module Clocks view

The **Module Clocks** view provides an overview of the peripheral instances. It also provides the information on which clock can be consumed by the particular clock instance. This view is not editable and is for information only.

NOTE

The information on which peripherals are consuming a particular output clock is available in the clock output tooltip.



Peripheral	Consumed Clocks
ADC0	Bus clock, OSCERCLK
ADC1	Bus clock, OSCERCLK
AIPS0	Bus clock, Flash clock, System clock
AIPS1	Bus clock, Flash clock, System clock
AXBS	System clock
CAN0	Bus clock, OSCERCLK
CAU	System clock
CMP0	Bus clock
CMP1	Bus clock
CMP2	Bus clock
CMT	Bus clock
CRC	Bus clock
DAC0	Bus clock
DAC1	Bus clock
DMA	System clock
DMAMUX	Bus clock
ENET	Bus clock, ENET IEEE 1588 timestamp clock, ENET RMII clock, System clock
EWM	Bus clock, LPO clock
FB	System clock
FMC	Flash clock, System clock

Figure 87. Module Clocks view

Chapter 6

Project Generator Tool

This tool enables you to create SDK-based projects for the MCUXpresso IDE, Kinetis Design Studio 3.x, GCC ARM Embedded (command line), IAR Embedded Workbench, Keil MDK μ Vision, and Somnium DRT toolchains.

6.1 Features

- Allows you to create SDK-based projects for the following toolchains.
 - MCUXpresso IDE
 - Kinetis Design Studio 3.x
 - GCC ARM Embedded (command line)
 - IAR Embedded Workbench
 - Keil MDK μ Vision
 - Somnium DRT
- Allows you to clone SDK example projects.
- Generates stand-alone projects. For example, all sources and libraries needed for project compilation are copied into project folder.

NOTE

The project files hierarchy in the toolchain IDE is same as on the disk

- Supports all SDK drivers, utilities, CMSIS drivers, and RTOSes.
- Supports C and C++ project types.
- Creates template of the main file, so the project is ready to compile and run.
- Allows you to update project that was already generated.
- Supports MCUXpresso SDK 2.2 and higher.
- Supports multi-core devices (Cortex-M cores).
- Provides possibility of automatically fixing the component selection problems. For example, auto-resolve component dependencies or components required from other tools.

6.2 Workflow

To use the Project Generator tool, ensure that you adhere to the following steps.

1. Create a new or open an existing configuration. For details, see chapter [Configuration](#).
2. Select the **Tools > Project Generator** option from the tools menu to invoke the Project Generator tool.
3. For **Desktop**: Select the project-base directory. This is the directory where the project(s) will be generated.
4. Select the toolchain and the language.
5. In the project configuration, select the name of the project.
6. Select **Real-Time Operating System**.

7. Select the components required for the application. Optionally, you can click the button to select the minimal driver set or the button to auto resolve all component problems.
8. Click **Create Project** to generate the project for the selected toolchain.
9. **Desktop**: Once the project is successfully generated, click the link to open the generated project folder.
10. Open the project in the toolchain.
11. Build the project.
12. **Web**: The generated project is available for download as a software archive.
13. **Desktop**: If you need additional components in the project, select the required components and click the **Update Project** button. The project is updated and your changes are preserved.

6.3 User interface

The tool is integrated and runs with the MCUXpresso Config Tools framework. The main window and the menu are provided by this framework (both for the Web and Desktop versions).

For documentation on the common framework and menu items, see the [Tools Common Framework User Interface](#).

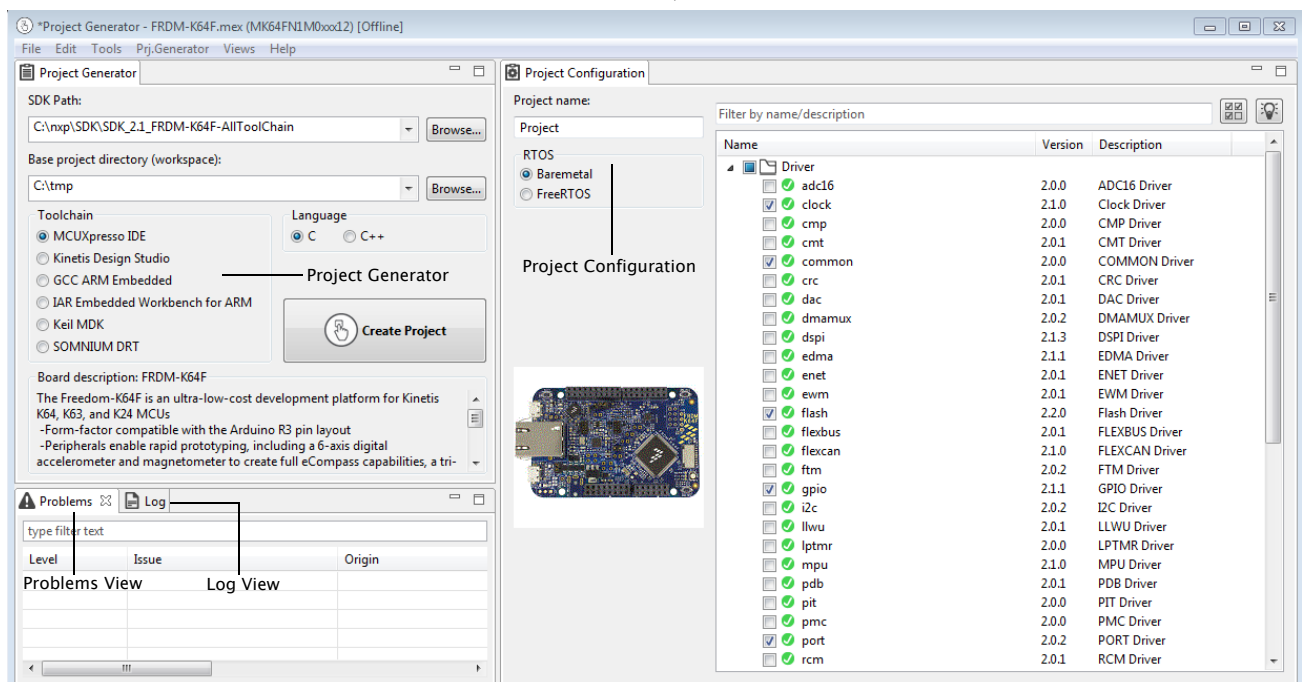


Figure 88. Project Generator interface

6.3.1 Views

The following views are supported in the Project Generator tool.

- **Project Generator** view – Contains main configuration, selection of SDK, toolchain, and target path.
- **Project Configuration** view – Contains the project-specific configuration.
- **Problems** view – Displays the configuration problems.
- **Log** view – Displays the result of the **Project Generator** tool commands.

All the views are accessible through view menu.

6.3.2 Specific menu commands

The Project Generator tool contains the shortcut command for the [project generation](#) of the configured properties.

To invoke the Project Generator tool, select **Prj.Generator > Create/Update Project** from the main menu.

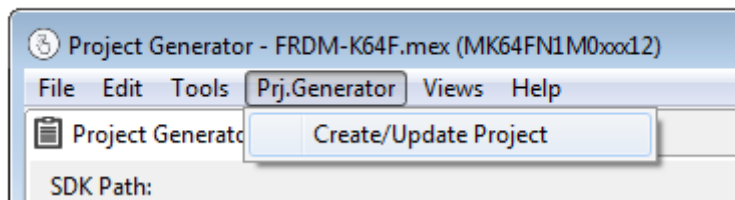


Figure 89. Create/Update project command

Alternatively, you can use the **Project Generator > Export** wizard.

6.3.3 Errors and warnings

There are three types of problems reported in the tool.

- Error
- Warning
- Information or hint

A decorator appears on the top-left corner of the control to highlight the status of the editing value.

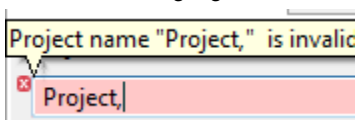


Figure 90. Error on the edit box

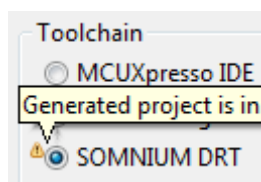


Figure 91. Warning on item the radio group

The status of the component is shown between the checkbox and the name in the first column. The description of the highlighted status appears as a tooltip for component when hovered.

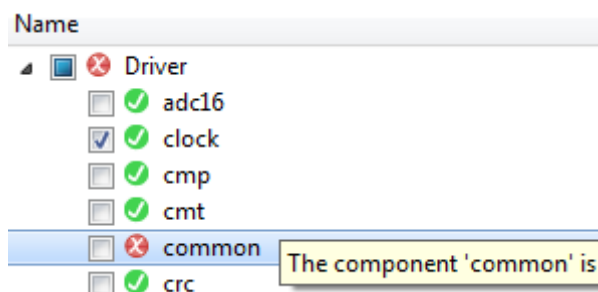


Figure 92. Component error

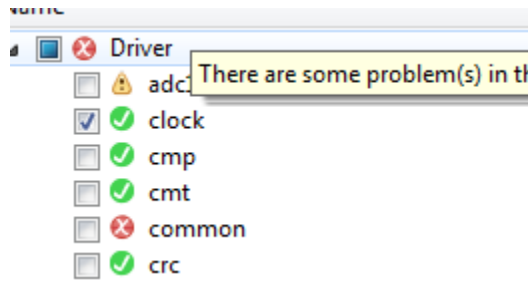


Figure 93. Highest severity status of the component is propagated to the parent node

NOTE

All problems are listed in the Problems view, except in case of unsupported components.

6.3.4 Project Generator view

This view contains controls to configure general core nonspecific properties. On desktop, it is possible to select path SDK, in drop down list is possible reselect previously selected existing path. Second selectable path will be used as a root folder to generate project(s), drop down list also contains previously selected existing path. Root folder must exist. Path are possible to select via folder dialog. On the left site of the view is group of **Toolchain** supported in selected SDK. Generated project will be possible to open by selected toolchain. Button activating [] is located [on the right-hand side]/[to the right of] the Toolchain [box]. There is a **Language** group above the **Generate** button, where it is possible to select project type.

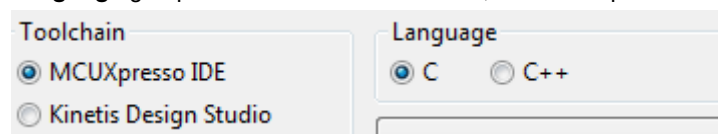


Figure 94. Language group

Examples are creates in specific language, that is why **Language** group will be replaced by Example group, when example is selected instead of the new configuration. The **Example** group helps identify selected example by showing name (relative path in SDK).

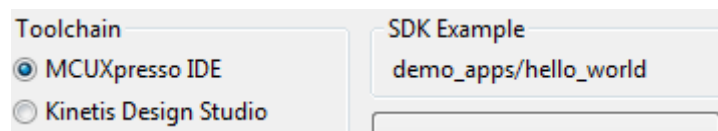


Figure 95. Example group

Description of selected kit, board or processor from SDK filled remaining space on bottom of the view.

6.3.5 User-specific paths

The "SDK path" and the "Project base path" parameters are user-specific and these settings are stored in the used preferences and not in the configuration file.

- *SDK path*: is set based on the SDK version and the configuration.
- *Project base path*: is same for all the projects. You can select the path from the drop-down list, where the paths are listed. If the path is not used, you can select the path using the **Browse** dialog.

6.3.6 Project Configuration view

The Project configuration view allows you to configure specific properties. For details on layout differences of the multicore processors, see [Multicore projects](#).

You can configure the Project name in the view. The value must be identifier because it is used the name of the generated project folder.

You can select the real-time operating system (RTOS) from the list of the supported ones in the selected SDK.

The **Components** tree occupies most of the space in the **Project Configuration** view, where the [components selection](#) is performed. The tree contains all the supported components divided by type in the selected SDK. You can perform the following tasks in the components tree:

- **Filter by name or description:** Displays only those components, which satisfy a given condition. Wildcard * is supported, representing any substring.
- **Select minimal set of components:** Required only by other tools or the selected example. The result of the operation is logged.
- **Auto resolve problems in components:** Fixes problems by selecting or de-selecting components. It fixes even problem in RTOS. The number of fixed problems is found in the **Log** view.

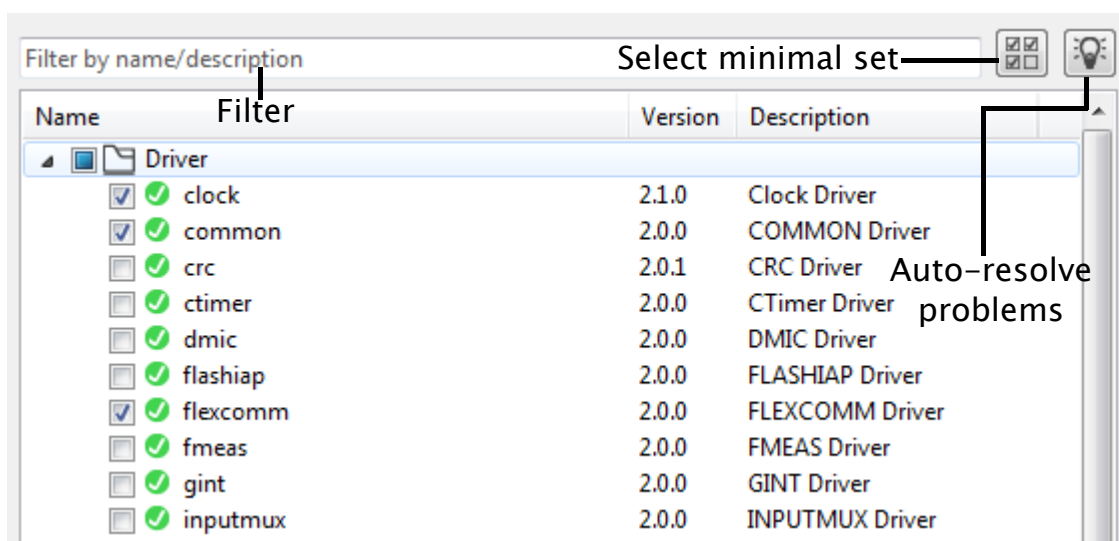


Figure 96. Components tree

NOTE

The image of the device is shown only when it exists in the SDK.

6.3.7 Component selection

The components tree is a summary of components within the SDK. The checked or unchecked checkbox help recognize the included or excluded components in the generated project. The components are divided in to several subtrees with the root node representing their common type.

You can also select or deselect components in the second selection mode by checking or unchecking the checkbox based on the component type. The selected value is propagated to the individual components, when the subtree of the selected type node is filtered.

There are three types of configurations for the component type.

- **All** - Components of a given type are included, regardless of the component name. When SDK supports new component of this type, the component is included. Renaming component in SDK does not cause any problem.

You can set it by checking the type node checkbox and the whole type subtree is untouched by the filter.

- **None** – Opposite of the aforementioned type “All”. Excludes components by this type.

- **Custom** – Included components are collected in a list. Other components are excluded. Modifying the component identification causes a problem.

You can set it when components are included or excluded individually.

6.4 Multicore projects

This section describes the differences between selecting the supported multicore and single-core processors.

6.4.1 Core booting role

The core booting role defines the used initialization of the core:

- **Primary** - If the core initializes the processor. You can set only one core as the primary.
- **Secondary** - If the core is started after the processor, the primary core initializes.
- **Unused** - If the core is not used, the project is not generated for the core.

6.4.2 Project Configuration view

The **Project Configuration** view is enhanced to configure the multicore project. The new control appears in the **Window** view. The control helps configure the [Core booting role](#).

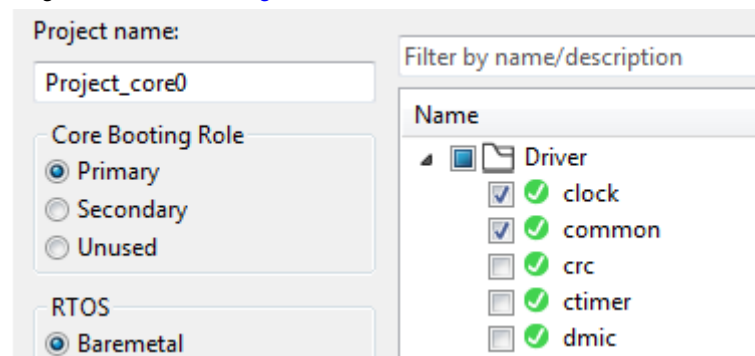


Figure 97. Core booting role controller

To prevent unwanted modification of the project for the unused core, the widgets are disabled.

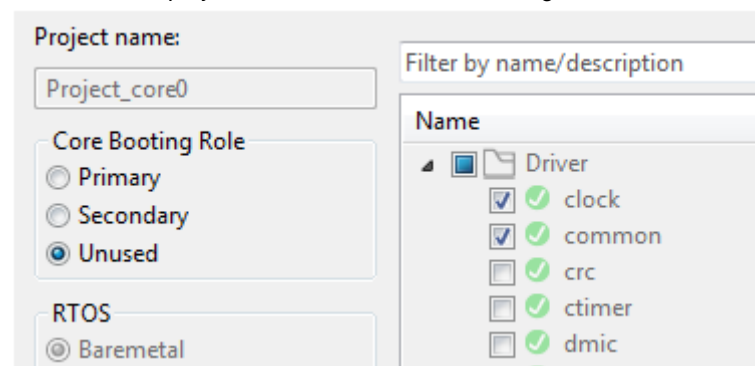


Figure 98. Core booting role set to Unused

The extended single core view is duplicated into the **Window** tabs, where each tab represents individual project for the core. The project configuration does not influence different projects.

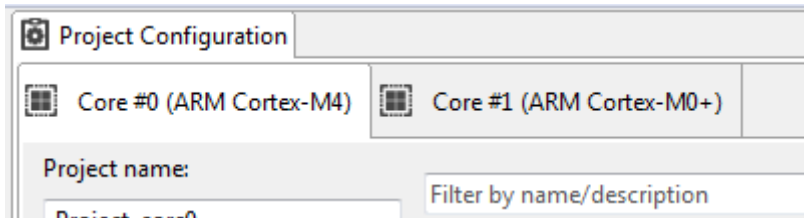


Figure 99. Tabs for multicore configuration

6.4.3 Compiler symbols

The following symbols are defined in the compiler for multicore applications:

Table 7. Compiler symbols

Symbol name	Description
SDK_PRIMARY_CORE	Defined for project running on the primary core; only for multicore applications.
SDK_SECONDARY_CORE	Defined for project running on the secondary core.

6.5 Project generation

Before code generation ensure there are no issues in the Pins and Clocks tool, because code generation contains code from all tools. Click the **Create Project** button to generate project. Alternatively, you can use the menu command.

The description of all the buttons is available as a tooltip.

For Desktop:

- **Create Project** – Click the **Create Project** to generate the project for the first time.

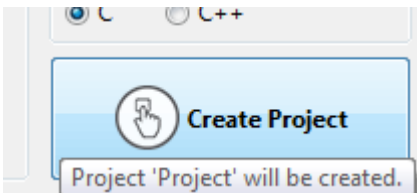


Figure 100. Create project

- **Update Project** – Click the **Update Project** button to add, remove, or update components to an existing project. For details, see chapter [Project update and backup files](#).

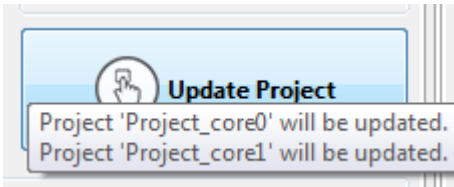


Figure 101. Update multicore project

- **Recreate Project** – Click the **Recreate Project** button to recreate an existing project when the existing project do not allow project updates. The reason for regeneration is described in the tooltip above the button.

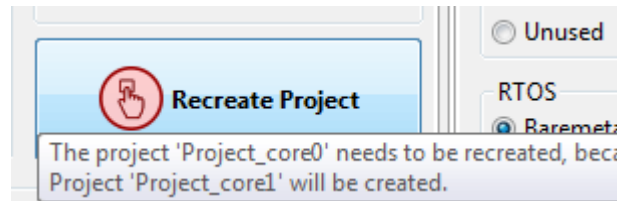


Figure 102. Re-Create project

- **Invalid Configuration** – Click to correct the configuration problems that did not allow project generation.

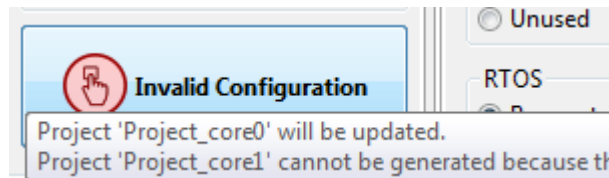


Figure 103. Invalid configuration

A dialog appears on the successful generation of the project. However, if project generation fails, the reason appears in the dialog.

	A dialog appears on the successful generation of the project. The dialog contains the link to the generated project folder. To open the project in the file browser, click the respective link.
--	---

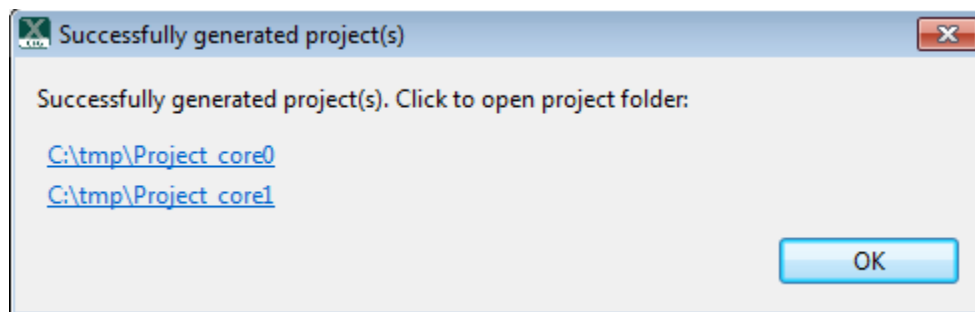


Figure 104. Invalid configuration

6.5.1 Export

The **Export** wizard helps you to export a generated project to a different location.

1. Select **File > Export** or **Prj.Generator > Export** from the main menu.

The **Select** dialog appears.

2. Expand the **Project Generator** tree.
3. Select the **Create/Update project** option.

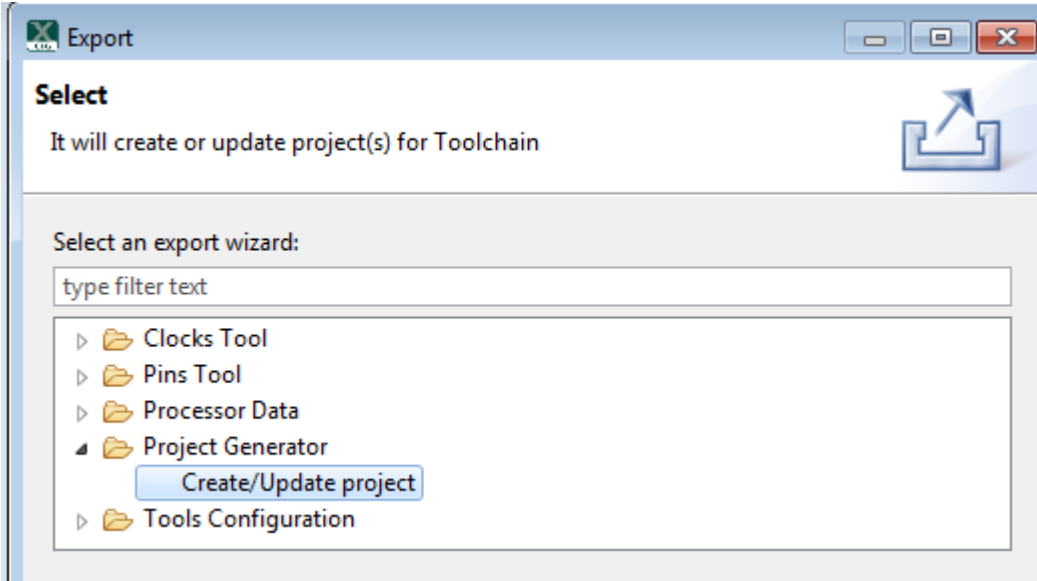


Figure 105. Export project

- 4. Click **Next**
- 5. Select target folder. This folder is temporarily set as a root folder to generate projects.
- 6. Click **Finish**.

6.6 Structure of a generated project

The generated project contains the following subfolders:

Table 8. Structure of a generated project

Subdirectory	Content
<i>root</i>	Toolchain-specific project files and linker file; MEX configuration file; <i>ProjectGenerator.gen</i> with information about the generated last project.
<i>board</i>	Code generated from Pins, Clocks, and other board-specific sources
<i>CMSIS</i>	Device-specific CMSIS header files.
<i>doc</i>	Example-Specific documentation.
<i>drivers</i>	Source files of the SDK drivers.
<i>middleware</i>	Source files and libraries for the middleware. For example, FreeRTOS and others.
<i>settings</i> or <i>.settings</i>	Toolchain-specific configuration files
<i>source</i>	Main file or other source templates.
<i>startup</i>	Device startup files.
<i>utilities</i>	Source files of the SDK utilities.

NOTE

For details, see [Toolchain specific information](#).

6.7 Toolchain-specific information

6.7.1 MCUXpresso IDE


MCUXpresso IDE project directory structure is simplified when compared to other toolchains. The MCUXpresso IDE toolchain is based on SDK and can add SDK sources during the project import. However, the project updates are not fully supported due to the same reason. The project structure is as follows.

Table 9. MCUXpresso IDE toolchain structure

Subdirectory	Content
<i>root</i>	Project.xml with information about project creation in MCUXpresso IDE. MEX configuration file. ProjectGenerator.gen with information about the last generated project.
<i>board</i>	Code generated from Pins, Clocks, and other board-specific sources.
<i>source</i>	Main file or other source templates.

To open a generated project in the MCUXpresso IDE:

1. Start the MCUXpresso IDE.
2. Open Installed SDKs view and ensure that the SDK is installed.
3. Navigate to the **Quickstart Panel** view and select **Import SDK example(s)...**
4. Select *SDK* and click **Next**.
5. Click the **Import from XML...** button on the toolbar.
6. Select the generated XML file
7. In the list of examples, expand the **Project Generator** category and select the generated example.
8. Confirm to import.

	It is highly recommended to use the same SDK package in Project Generator and in the MCUXpresso IDE.
---	--

6.7.2 Kinetis Design Studio (KDS)

Project Generator generates native Eclipse project format. Generate the project directory into the Eclipse workspace. To open the project in KDS, select **File > Import > General > Existing Projects into Workspace**.

The launch configurations are provided in the settings subfolder.

6.7.3 ARM GCC embedded

Batch files are provided to build the application in the project root. Use build_all script to build both the configurations.

6.7.4 IAR embedded workbench for ARM

To open the generated project in IAR, double-click the *Project.eww* file. Alternatively, select **File > Open > Workspace...** from the main menu.

6.7.5 Keil MDK

To open generated project for MDK, double-click the *Project.uvmpw* file. Alternatively, select **Project > Open Project...** from the main menu.

6.7.6 Somnium DRT

The file format is same as for the Kinetis Design Studio. For details, see [Kinetis Design Studio \(KDS\)](#).

6.8 Project updates and backup files

The project update and backup files feature applies to the desktop version only.

For Web version, the complete project is always generated.

For details on toolchain-specific information, see [Toolchain-specific information](#).

6.8.1 Project updates

You can regenerate a project anytime and make the following updates:

- Any changes in the Clocks or Pins configuration
- Selection of RTOS or selection of any component
- Core role for multi-core devices

You can apply the following changes in the project file and preserve any user changes:

- List of project source files
- List of include paths is updated
- List of compiler define symbols is updated
- List of libraries is updated

The content of the main file is never updated. The content is rewritten only if RTOS is changed. The main file extension is either *.c* or *.cpp* depending on the selected language.

NOTE

The ProjectGenerator.gen file from the project root folder detects the project update and you should never remove it.

6.8.2 Backup files

During the project update, no source file or project file is removed or rewritten. Instead of removal or overwriting, the file is renamed with “backup” extension. Only one backup copy of the file is maintained. The previous backup file is always overwritten. If there is no change in the source file, the backup is kept untouched.

Chapter 7 Advanced Features

7.1 Switching processor (for desktop version only)

It is possible to switch the processor or the package of the current configuration to a different one. If switched to completely different processor, it may lead to a conflict or problems. For example, inaccessible pin routing or unsatisfiable clock output frequency. It is necessary to fix the problems manually. For example, go to the Pins Routing table and re-configure all pins which reports an error or conflicts. 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.

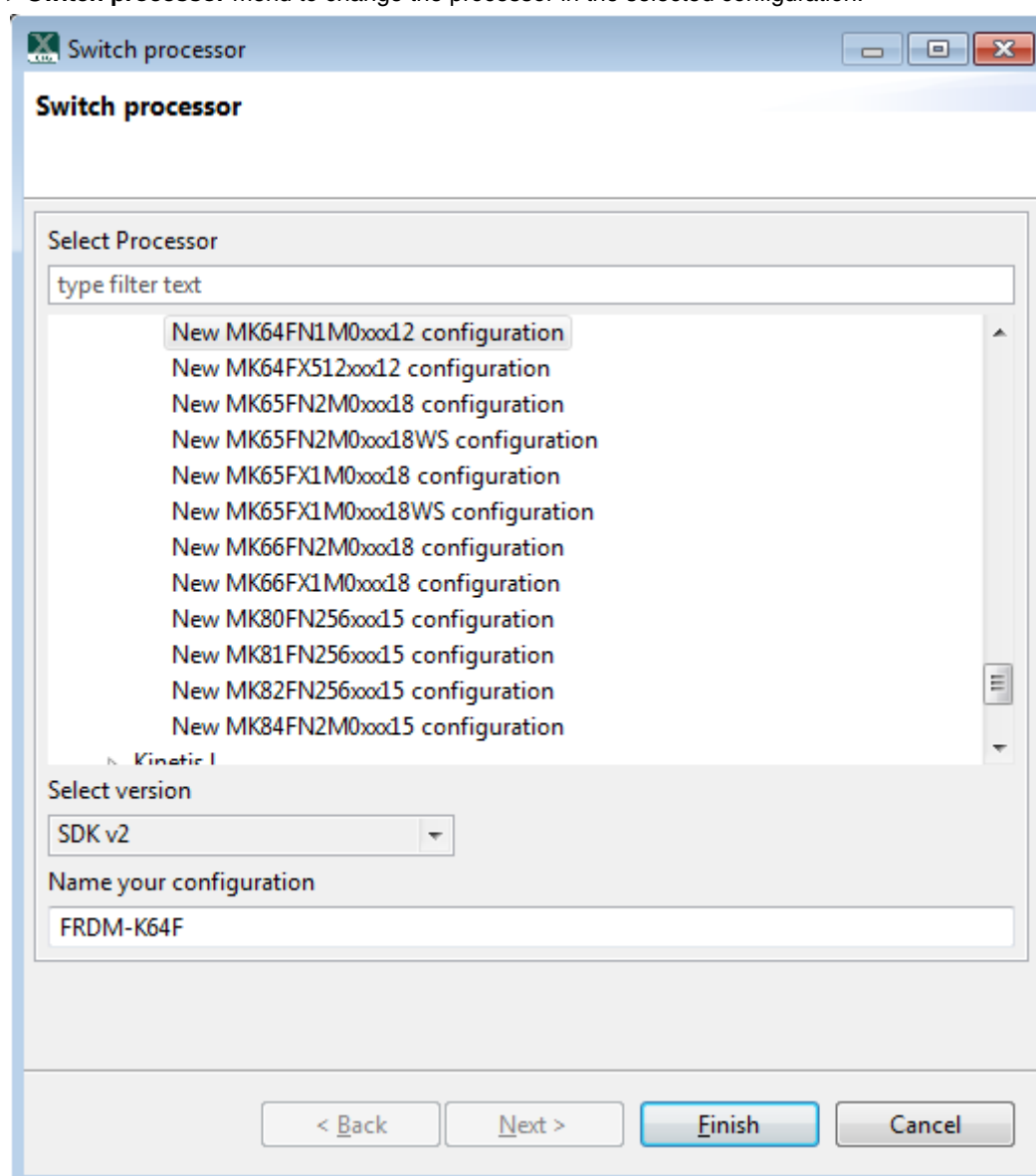


Figure 106. Switch processor

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

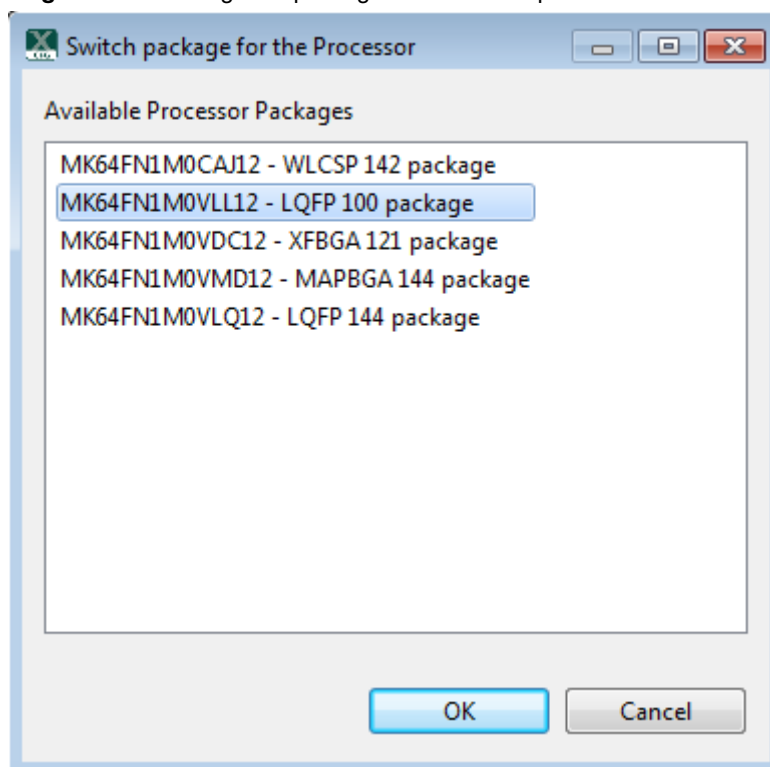


Figure 107. Switch package

7.2 Exporting Pins table

To export Pins table:

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.

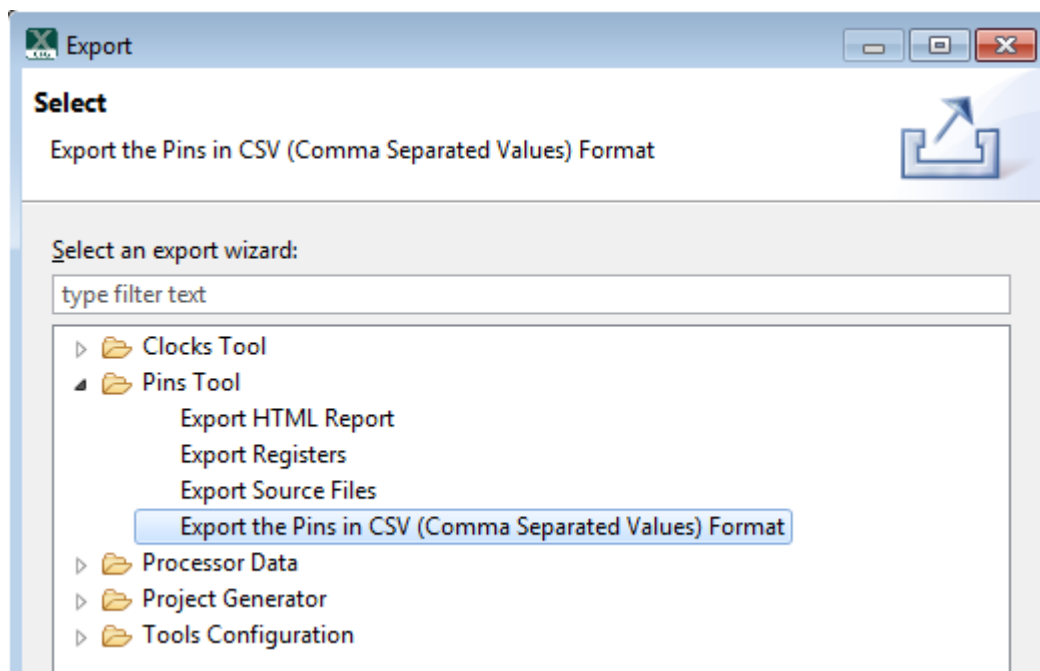


Figure 108. Export dialog

3. Click **Next**.
4. Select the folder and specify the file name to which you want to export.

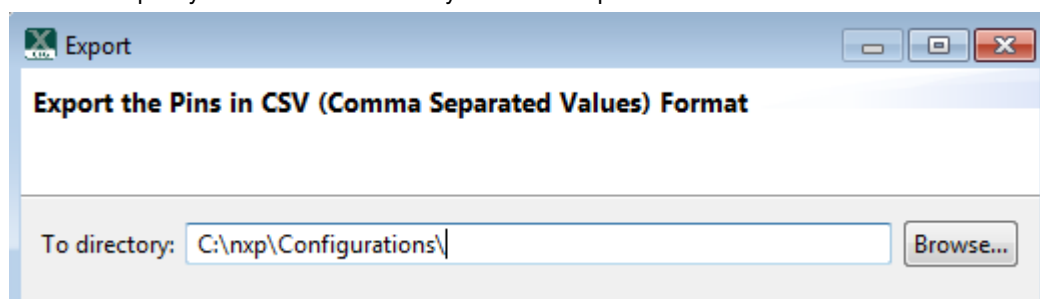


Figure 109. Export the Pins in CVS

5. The exported file contains content of the current Pins view table, plus lists the functions and the selected routed pins.

```
sep=;
Pin;Pin name;GPIO;FTM;ADC;UART;SPI;I2S;LLWU;I2C;CMP;SUPPLY;LPUART;USB;SIM;JTAG;RTC;EWM;Other;Routing for BOARD_InitPins
A1;PTE0/CLKOUT32K;PTE0/CLKOUT32K(GPIOE,GPIO,0);;ADC1_SE4a(ADC1,SEa,4);UART1_TX(UART1,TX);SPI1_PCS1(SPI1,PCS1);;I2C1_SDA(I2C1,SDA);;PTE0
B1;PTE1/LLWU_P0;PTE1/LLWU_P0(GPIOE,GPIO,1);;ADC1_SE5a(ADC1,SEa,5);UART1_RX(UART1,RX);SPI1_SOUT(SPI1,SOUT)/SPI1_SIN(SPI1,SIN);;PTE1/LLWU_P0(
C1;PTD5;PTD5(GPIOD,GPIO,5);FTM0_CH5(FTM0,CH,5);ADC0_SE6b(ADC0,SEb,6);UART0_CTS_b(UART0,CTS);SPI0_PCS2(SPI0,PCS2)/SPI1_SCK(SPI1,SCK);;
D1;USB0_DM;USB0_DM(USB0,DM);;
E1;USB0_DP;USB0_DP(USB0,DP);;
F1;ADC0_DM0/ADC1_DM3;ADC0_DM0/ADC1_DM3(ADC0,DM,0)/ADC0_DM0/ADC1_DM3(ADC0,SE,19)/ADC0_DM0/ADC1_DM3(ADC1,DM,3);;ADC0_DM0/ADC1_
G1;ADC0_DP0/ADC1_DP3;ADC0_DP0/ADC1_DP3(ADC0,DP,0)/ADC0_DP0/ADC1_DP3(ADC0,SE,0)/ADC0_DP0/ADC1_DP3(ADC1,DP,3)/ADC0_DP0/ADC1_DP3(ADC1,SE,3);
H1;VREF_OUT/CMP1_IN5/CMP0_IN5/ADC1_SE18;VREF_OUT/CMP1_IN5/CMP0_IN5/ADC1_SE18(ADC1,SE,18);;VREF_OUT/CMP1_IN5/CMP0_IN5/ADC1_SE18(CMP1,I
A2;PTD7/UART0_TX/FTM0_CH7/FTM0_FLT1/SPI1_SIN;PTD7(GPIOD,GPIO,7);FTM0_CH7(FTM0,CH,7)/FTM0_FLT1(FTM0,FLT,1);UART0_TX(UART0,TX);SPI1_SIN(SPI1
B2;ADC0_SE7b/PTD6/LLWU_P15/SPI0_PCS3/UART0_RX/FTM0_CH6/FTM0_FLT0/SPI1_SOUT;PTD6/LLWU_P15(GPIOD,GPIO,6);FTM0_CH6(FTM0,CH,6)/FTM0_FLT0(FTM0,F
C2;PTD2/LLWU_P13/SPI0_SOUT/UART2_RX/LPUART0_RX/I2C0_SCL;PTD2/LLWU_P13(GPIOD,GPIO,2);;UART2_RX(UART2,RX);SPI0_SOUT(SPI0,SOUT);;PTD2/LLWU_P1
D2;VREGIN;VREGIN(USB0,VREGIN);;
E2;VOUT33;VOUT33(USB0,VOUT33);;
F2;ADC1_DM0/ADC0_DM3;ADC1_DM0/ADC0_DM3(ADC1,DM,0)/ADC1_DM0/ADC0_DM3(ADC1,SE,19)/ADC1_DM0/ADC0_DM3(ADC0,DM,3);;
G2;ADC1_DP0/ADC0_DP3;ADC1_DP0/ADC0_DP3(ADC1,DP,0)/ADC1_DP0/ADC0_DP3(ADC1,SE,0)/ADC1_DP0/ADC0_DP3(ADC0,DP,3)/ADC1_DP0/ADC0_DP3(ADC0,SE,3);
H2;DAC0_OUT/CMP1_IN3/ADC0_SE23;DAC0_OUT/CMP1_IN3/ADC0_SE23(ADC0,SE,23);;DAC0_OUT/CMP1_IN3/ADC0_SE23(CMP1,IN,3);;DAC0_OUT/CMP1_I
A3;PTD4/LLWU_P14/SPI0_PCS1/UART0_RTS_b/FTM0_CH4/EWM_IN/SPI1_PCS0;PTD4/LLWU_P14(GPIOD,GPIO,4);FTM0_CH4(FTM0,CH,4);UART0_RTS_b(UART0,RTS);SP
B3;PTD3/SPI0_SIN/UART2_TX/LPUART0_TX/I2C0_SDA;PTD3(GPIOD,GPIO,3);;UART2_TX(UART2,TX);SPI0_SIN(SPI0,SIN);;I2C0_SDA(I2C0,SDA);;LPUART0_TX(
C3;PTD0/LLWU_P12;PTD0/LLWU_P12(GPIOD,GPIO,0);;UART2_RTS_b(UART2,RTS);SPI0_PCS0(SPI0,PCS0/SS);PTD0/LLWU_P12(LLWU,WAKEUP,P12);;LPUART0_RT
D3;PTA0/UART0_CTS_b/FTM0_CH5/JTAG_TCLK/SWD_CLK/E2P_CLK;PTA0(GPIOA,GPIO,0);FTM0_CH5(FTM0,CH,5);;UART0_CTS_b(UART0,CTS);;JTAG_TCLK(JT
```

Figure 110. 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.

;;pin			
P1n ;Pin name	;GPIO	;FTM	;ADC
A1 ;PTE0/CLKOUT32K	;PTE0/CLKOUT32K(GPIOE,GPIO,0)		;ADC1_SE4a(ADC1,SEa,4)
B1 ;PTE1/LLWU_F0	;PTE1/LLWU_F0(GPIOE,GPIO,1)		;ADC1_SE5a(ADC1,SEa,5)
C1 ;PTD5	;PTD5(GPIOD,GPIO,5)	;FTM0_CH5(FTM0,CH,5)	;ADC0_SE6b(ADC0,SEb,6)
D1 ;USBD_DM			
E1 ;USBD_DP			
F1 ;ADCO_DM0/ADC1_DM3			;ADCO_DM0/ADC1_DM3(ADCO,DM,0)/ADCO_DM0/ADC
G1 ;ADCO_DPO/ADC1_DPS			;ADCO_DPO/ADC1_DPS(ADCO,DP,0)/ADCO_DPO/ADC
H1 ;VREF_OUT/CMP1_INS/CMP0_INS/ADC1_SE18			;VREF_OUT/CMP1_INS/CMP0_INS/ADC1_SE18(ADC1
A2 ;PTD7/UART0_TX/FTM0_CH7/FTM0_FLT1/SPI1_SIN	;PTD7(GPIOD,GPIO,7)	;FTM0_CH7(FTM0,CH,7)/FTM0_FLT1(FTM0,FLT,1)	
B2 ;ADCO_SE7b/PTD6/LLWU_P15/SPI0_PCS3/UART0_RX/FTM0_CH6/FTM0_FLT0/SPI1_SOUT	;PTD6/LLWU_P15(GPIOD,GPIO,6)	;FTM0_CH6(FTM0,CH,6)/FTM0_FLT0(FTM0,FLT,0)/FTM0_FLT0(FTM0,TRG,2)	;ADCO_SE7b(ADC0,SEb,7)
C2 ;PTD2/LLWU_P13/SPI0_SOUT/UART2_RX/LPUART0_RX/I2C0_SCL	;PTD2/LLWU_P13(GPIOD,GPIO,2)		
D2 ;VREGIN			
E2 ;VOUT33			
F2 ;ADC1_DM0/ADCO_DM3			;ADC1_DM0/ADCO_DM3(ADC1,DM,0)/ADC1_DM0/ADC
G2 ;ADC1_DPO/ADCO_DPS			;ADC1_DPO/ADCO_DPS(ADC1,DP,0)/ADC1_DPO/ADC
H2 ;DAC0_OUT/CMP1_INS/ADCO_SE23			;DAC0_OUT/CMP1_INS/ADCO_SE23(ADCO,SE,23)
A3 ;PTD4/LLWU_P14/SPI0_PCS1/UART0_RTS_b/FTM0_CH4/EMM_IN/SPI1_PCS0	;PTD4/LLWU_P14(GPIOD,GPIO,4)	;FTM0_CH4(FTM0,CH,4)	
B3 ;PTD3/SPI0_SIM/UART2_TX/LPUART0_TX/I2C0_SDA	;PTD3(GPIOD,GPIO,3)		
C3 ;PTD0/LLWU_P12	;PTD0/LLWU_P12(GPIOD,GPIO,0)		
D3 ;PTA0/UART0_CTS_b/FTM0_CH5/JTAG_TCLK/SWD_CLK/EZP_CLK	;PTA0(GPIOA,GPIO,0)	;FTM0_CH5(FTM0,CH,5)	
E3 ;VSS0			
F3 ;VSSA			;VSSA(ADCO,SE,30)/VSSA(ADC1,SE,30)/VSSA(AI
G3 ;VREFL			;VREFL(ADCO,SE,30)/VREFL(ADC1,SE,30)/VREFL
H3 ;XTAL32			
A4 ;ADCO_SE5b/PTD1/SPI0_SCK/UART2_CTS_b/LPUART0_CTS_b	;PTD1(GPIOD,GPIO,1)		;ADCO_SE5b(ADC0,SEb,5)
B4 ;ADC1_SE6b/PTC10/I2C1_SCL/I2S0_RX_FS	;PTC10(GPIOC,GPIO,10)		;ADC1_SE6b(ADC1,SEb,6)
C4 ;VSS9			
D4 ;PTA1/UART0_RX/FTM0_CH6/JTAG_TDI/EZP_DI	;PTA1(GPIOA,GPIO,1)	;FTM0_CH6(FTM0,CH,6)	
E4 ;VDD01			
F4 ;VDDA			;VDDA(ADCO,SE,29)/VDDA(ADC1,SE,29)/VDDA(AI
G4 ;VREFH			;VREFH(ADCO,SE,29)/VREFH(ADC1,SE,29)/VREFH

Figure 111. Aligning to block

7.3 Export processor data

The tool automatically downloads the processor data only, if it is needed. However, it is also possible to explicitly download all available data into the local machine by selecting **Export > Processor Data > Download Processor Data**.

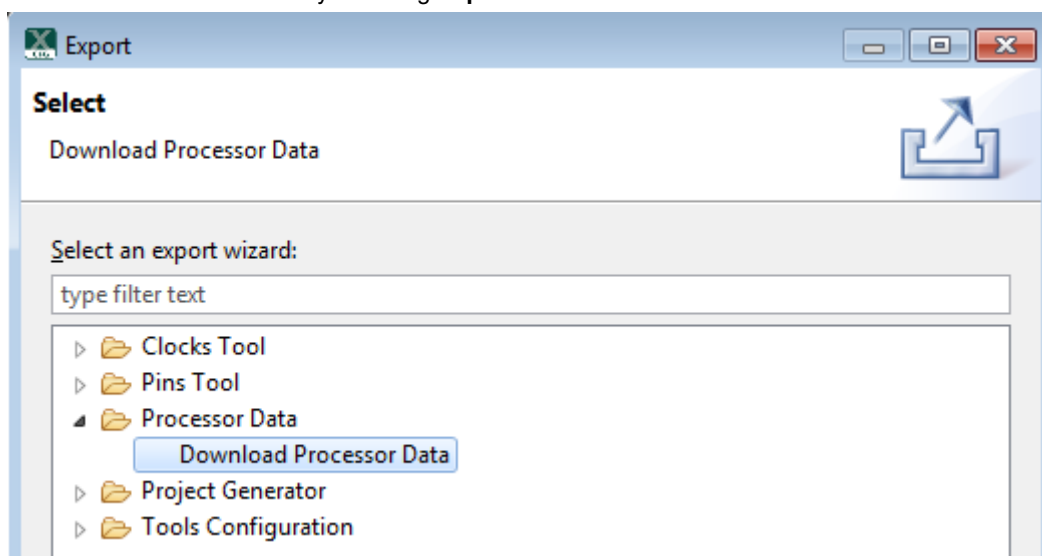


Figure 112. Download processor data options

The tool allows you to download data into a default data location or to a custom folder. The downloaded processor data includes data (processors/boards/kits) for selected series.

NOTE

The data size can be more than 1 GB and the download time depends on your network speed.

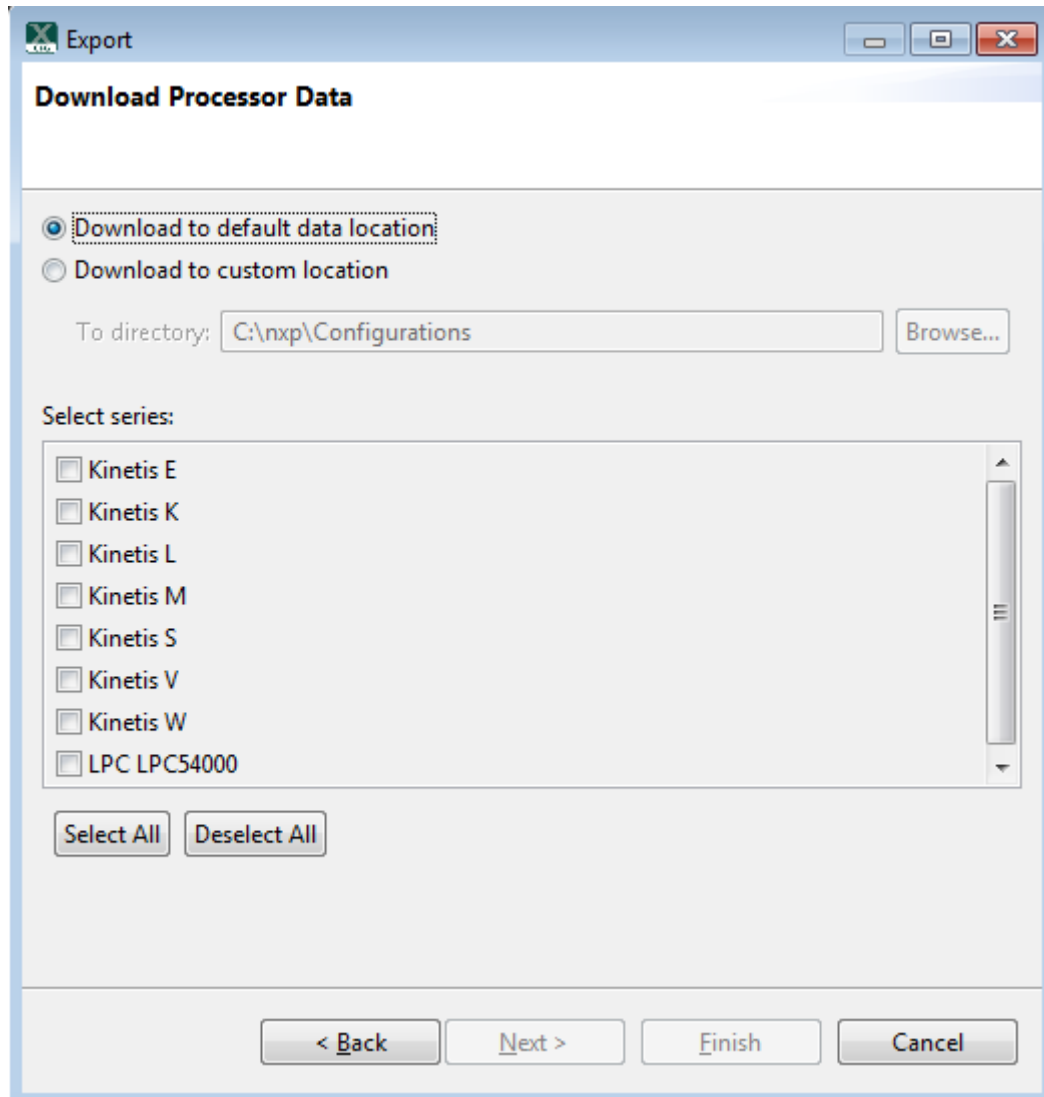


Figure 113. Export processor data

7.4 Tools advanced configuration

Use the tools.ini file to configure the processor data directory location. It is possible to define the "com.nxp.mcudata.dir" property to set the data directory location.

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

7.5 Generating HTML report

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

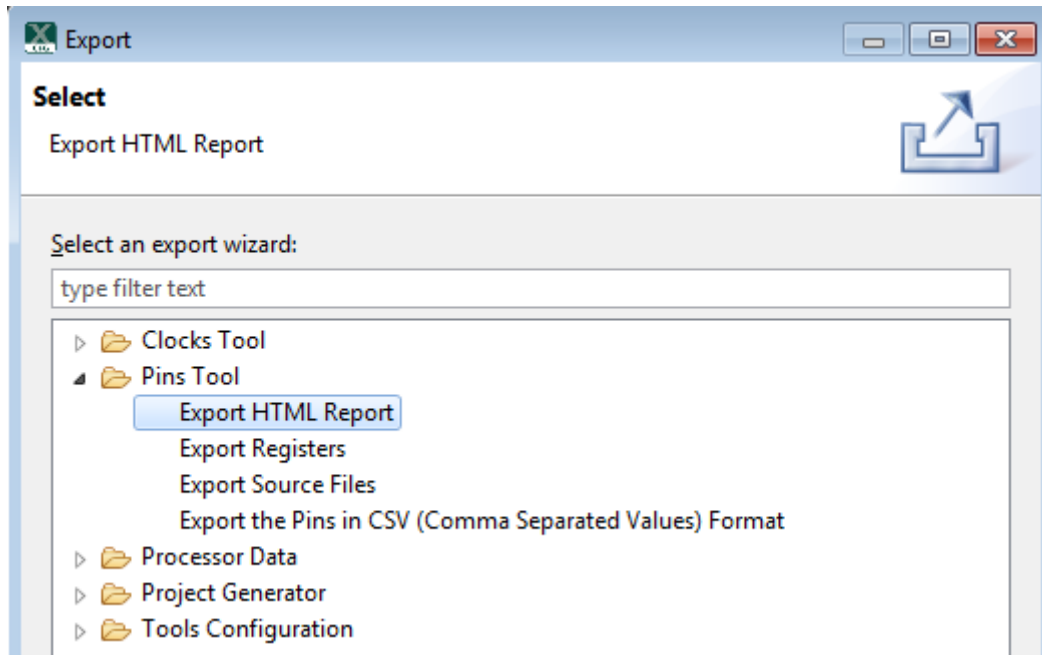


Figure 114. Export HTML report

7.6 Export registers

It is possible to export the tool modified registers data content using the Export wizard.

To launch the **Export** registers wizard:

1. Select **File > Export** from the main menu.
2. Select the **Pins Tool > Export Registers** option.

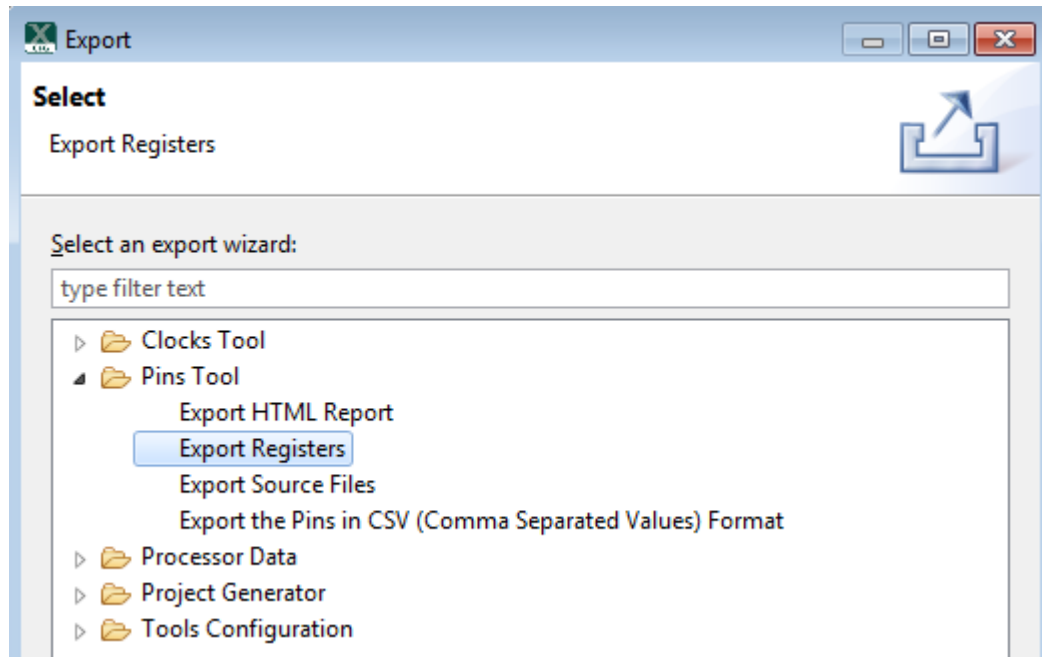


Figure 115. Export registers

3. Click **Next**.
4. Select the target file path where you want to export modified registers content.

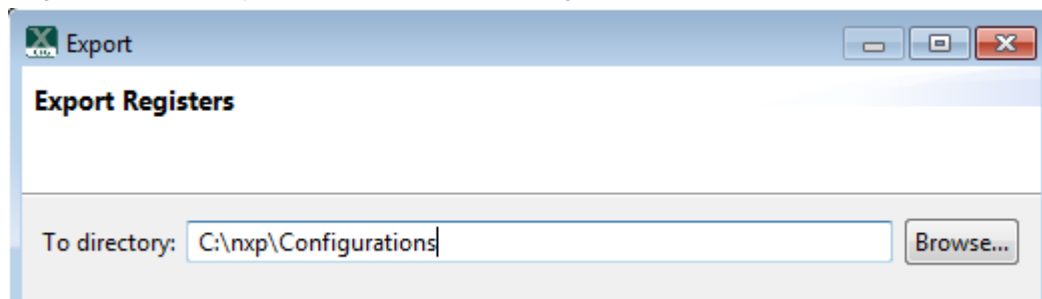


Figure 116. Export registers directory

5. Click **Finish**.

7.7 Command line execution

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

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

Table 10. Commands supported in the framework

Command name	Definition and parameters	Description	Restriction	Example
Table continues on the next page...				

Table 10. Commands supported in the framework (continued)

Force language	-nl {lang}	Force set language {lang} is in ISO-639-1 standard	Removal of the '.npx' folder from home directory is recommended, as some text might be cached Only 'zh' and 'en' are supported	-nl zh
Show console	-consoleLog	Log output is also sent to Java's System.out (typically back to the command shell if any)	None	
Select MCU	-MCU	MCU to be selected by framework	Requires – SDKversion command	-MCU MK64FX512xxx12
Select SDK version	-SDKversion	Version of the MCU to be selected by framework	Requires -MCU command	-SDKversion test_ksdk2_0
Select part number	-PartNum	Select specific package of the MCU	Requires -MCU and - SDKversion commands	-PartNum MK64FX512VLL12
Configuration name	-ConfigName	Name of newly created configuration - used in export	Name is used when new configuration is created by -MCU and -SDKversion commands	-ConfigName "MyConfig"
Select tool	-HeadlessTool	Select a tool that should be run in headless mode	None	-HeadlessTool Clocks
Load configuration	-Load	Load existing configuration from (*.mex) file	None	-Load C:/conf/ conf.mex
Export Mex	-ExportMEX	Export .mex configuration file after tools run Argument is expected as a folder name	None	-ExportMEX C:/ exports/ my_config_folder

Table continues on the next page...

Table 10. Commands supported in the framework (continued)

Export all generated files	-ExportAll	Export generated files (with source code and so on. Code is regenerated before export Includes -ExportSrc and in framework -ExportMEX Argument is expected as a folder name. Argument is expected as a folder name	Requires - HeadlessTool command	-ExportAll C:/exports/generated
Specify SDK path	-SDKpath {path}	Specify absolute path to the root directory of the SDK package.	@since v3.0	-SDKpath c:\nxp\SDK_2.0_MKL43Z256xxx4

7.7.1 Command line execution - Pins Tool

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

Table 11. Commands supported in Pins

Command name	Definition and parameters	Description	Restriction	Example
Import C files	-ImportC	Import .c files into configuration Importing is done after loading mex and before generating outputs	Requires - HeadlessTool Pins	-ImportC C:/imports/file1.c C:/imports/file2.c
Import DTSI files	-ImportDTSI	Import .dtsi files into configuration Importing is done after loading mex and before generating outputs	Requires - HeadlessTool Pins	-ImportDTSI C:/imports/file1.dtsi C:/imports/file2.dtsi

Table continues on the next page...

Table 11. Commands supported in Pins (continued)

Export all generated files (to simplify all exports commands to one command)	-ExportAll	Export generated files (with source code etc.) Code will be regenerated before export Includes -ExportSrc,-ExportCSV, -ExportHTML and in framework -ExportMEX Argument is expected as a folder name	Requires -HeadlessTool Pins	-ExportAll C:/exports/generated
Export Source files	-ExportSrc	Export generated source files. Code will be regenerated before export Argument is expected as a folder name	Requires -HeadlessTool Pins	-ExportSrc C:/exports/src
Export CSV file	-ExportCSV	Export generated csv file. Code will be regenerated before export Argument is expected as a folder name	Requires -HeadlessTool Pins	-ExportCSV C:/exports/csv
Export HTML report file	-ExportHTML	Export generated html report file. Code will be regenerated before export Argument is expected as a folder name	Requires -HeadlessTool Pins	-ExportHTML C:/exports/html
Export registers	-ExportRegisters	Export registers tab into folder. Code will be regenerated before export Argument is expected as a folder name	Requires -HeadlessTool Pins	-ExportRegisters C:/exports/regs

7.7.2 Command line execution - Clocks Tool

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

Table 12. Commands supported in Clocks

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

Table continues on the next page...

Table 12. Commands supported in Clocks (continued)

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

7.7.3 Command line execution - Project Generator Tool

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

Table 13. Commands supported in Project Generator

Command name	Definition and parameters	Description	Restriction	Example
<i>Table continues on the next page...</i>				

Table 13. Commands supported in Project Generator (continued)

Create new project	-PG_newPrj {processor-partNum-board} {core} {rtos} {toolchain} {prj-type} {wrkspc} {prjName} {compTypes}	Creates new project for selected processor or board. 1. {processor-partNum-board} - sub-directory of the board in SDK package OR part number of the device OR full name of the device from the SDK MANIFEST (see devices - device tag) 2. {core} - device core type from the SDK MANIFEST (see devices - device - core - name) optionally with suffix _master or _slave to create project for master or slave 3. {rtos} - name of the RTOS component used; none if no RTOS shall be used 4. {toolchain} - id of the toolchain to create project (see toolchains - toolchain - id) 5. {prj-type} - either C for C project type; or CPP for C++ project type 6. {wrkspc} - absolute path where new project shall be created (e.g. project workspace); if the path starts with #erase:, then the directory	Requires - HeadlessTool PG and - SDKpath {path} @since v3.0	-HeadlessTool PG - SDKpath c:\nxp \SDK_2.0_MKL43Z25 6xxx4 -PG_newPrj frdmkl43z cm0 none kds C c:\tmp prj SDK_ALL -HeadlessTool PG - SDKpath c:\nxp \SDK_2.0_MKL43Z25 6xxx4 -PG_newPrj MK64FN1M0xxx12 cm0 freertos mdk CPP c:\tmp prj SDK_ALL
--------------------	--	--	--	---

Table continues on the next page...

Table 13. Commands supported in Project Generator (continued)

		<p>content will be deleted before projects being generated</p> <p>7. {prjName} - name of the new project</p> <p>8. {compTypes} - SDK components to add into project: ALL_SDK for all drivers, CMSIS drivers and utilities; MIDDLEWARE for all middleware components; ALL for SDK_ALL plus middleware plus CMSIS drivers; NONE for none; INDIVIDUAL to create project for each component</p>		
Table continues on the next page...				

Table 13. Commands supported in Project Generator (continued)

Clone SDK example project	-PG_clone {board} {example} {toolchain} {wrkspc} {prjName}	Clones specified SDK example projects under new name 1. {board} - subdirectory of the board in SDK package 2. {example} - relative path from board sub-dir and name of the example, for example demo_apps/hello_world; use '/' as a path separator 3. {toolchain} - id of the toolchain to create project (see toolchains - toolchain - id) 4. {wrkspc} - absolute path where new project shall be created, e.g. projects workspace 5. {prjName} - name of the new project	Requires - HeadlessTool PG and - SDKpath {path} @since v3.0	-HeadlessTool PG - SDKpath c:\nxp\SDK_2.0_MKL43Z256xxx4 -PG_clone twrk64f120m demo_apps/hello kds c:\tmp exmpl
---------------------------	--	---	--	--

7.8 Working offline

To work offline, you need to first download the processor-specific data. Once the configuration is created for the processor, select **Preferences > Work offline**.

Chapter 8

Support

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

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

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

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

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and μ Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2016-2017 NXP B.V.

