NXP Semiconductors
User's Guide

Document Number: QCVS_PEX_User_Guide
Rev. 4.x, 06/2016

QCVS Processor Expert User Guide

Contents

Chapter 1 Introduction

ProcessorExpert is the development framework used by QorIQ Configuration and Validation Suite (QCVS). This document describes how to get started with the Processor Expert tool and explains the user interface of the tool.

This chapter explains:

- Overview on page 4
- Features on page 4
- Benefits on page 4
- Concepts on page 5

1.1 Overview

This section explains why the Processor Expert tool was developed.

Both hardware and software design have progressed with the ever-advancing new technologies emerging everyday, but their interrelationships and interdependence have been mostly neglected. On one hand, you often see a good new hardware architecture, but the software design is too expensive for such an architecture. On the other hand, the computerization of nearly all mechanical gadgets all over the modern world leads to the use of embedded computer systems.

In situations where expense is a consideration, embedded computer systems with efficient software can significantly reduce the overall design cost. Processor Expert is designed for rapid application development of embedded applications for a wide range of processors.

1.2 Features

This section explains the key features of Processor Expert tool.

The key features of Processor Expert are:

- Includes a library of embedded system building blocks
- Created from embedded components
- Provides built-in SOC verification system for verifying component settings
- Supports automatic configuration management and data generation
- Easy to use graphical user interface
- Integrated into Eclipse environment

1.3 Benefits

This section explains the benefits of using the Processor Expert tool.

The benefits of Processor Expert are:

- Embedded components encapsulate the functionality of basic elements of embedded systems, such as CPU core, on-chip peripherals, FPGA, standalone peripherals, virtual devices, and pure software algorithms, and change these facilities to properties, methods, and events such as objects in OOP.

- The Peripheral Initialization components of Processor Expert generate effective initialization code for all on-chip devices and support all their features.
- Processor Expert allows easy examination of the details of the architecture and the relationship between the embedded component setup and processor's control registers initialization.

1.4 Concepts

This section explains the concepts of Processor Expert tool.

Embedded device's configuration or functionality is separated into building blocks, called embedded components. The Processor Expert application is created from embedded components. These components generate a source code or data files for initialization and configuration of selected Freescale silicon. They have an interface similar to an object-oriented programming model that consists of:

- Properties: Used to modify or customize the object behavior. You can control properties in design time by using the Component Inspector.
- Methods (not used in this version): The procedures that can be executed in application code and function calls
- Events (not used in this version): The interfacing hardware or software events that are invoked by the component to the user's code

The embedded components have a subset, called peripheral initialization components, which allow you to setup initialization of the particular on-chip device to any possible mode of operation. You can easily view all initialization values of the processor produced by Processor Expert with highlighted differences between the last and current properties settings.

Processor Expert performs a design time checking of the settings of all components and reports errors and warnings notifying you about wrong property values or conflicts in the settings with other components in the project.

Processor Expert contains many useful tools for exploring a structure of the target processor showing the details about the allocated on-chip peripherals and pins. It generates a ready-to-use code or the data files initializing all on-chip peripherals used by the component according to the component setup.

Chapter 2 User Interface

The Processor Expert menu is integrated as a plugin in the Eclipse IDE, providing a set of views. The Eclipse IDE main menu has a menu item, named **Processor Expert**. The user interface of Processor Expert consists of the following views:

- Component Inspector: Allows you to set up components of the project
- Component Library: Shows all supported components including processor components and component templates
- Configuration Registers: Shows overview of the peripheral initialization settings for the current processor
- Memory Map: Shows the processor address space and internal and external memory mapping
- Components: Shows an embedded component that can be used in Processor Expert
- Processor: The processor used in a given project

This chapter explains:

- Processor Expert menu on page 6
- Components view on page 9
- Components Library view on page 13
- Component Inspector view on page 15
- Processor view on page 21
- Memory Map view on page 25
- Configuration Registers view on page 26

2.1 Processor Expert menu

The Processor Expert menu is an IDE menu that shows options specific to the Processor Expert tool.

The Processor Expert menu includes:

- **Show Views**: Shows standard Processor Expert windows in case they are hidden
- **Hide Views**: Hides Processor Expert views
- **Import Component(s)**: This command allows to select and install Processor Expert update packages (`.PEUpd`) files. These files can be created in CDE by exporting a user's component.

This section contains the following subsections:

- Project pop-up menu on page 6
- Project options on page 7
- Preferences on page 7

2.1.1 Project pop-up menu

This section explains the project pop-up menu, which can be accessed by right-clicking the `ProcessorExpert.pe` file.

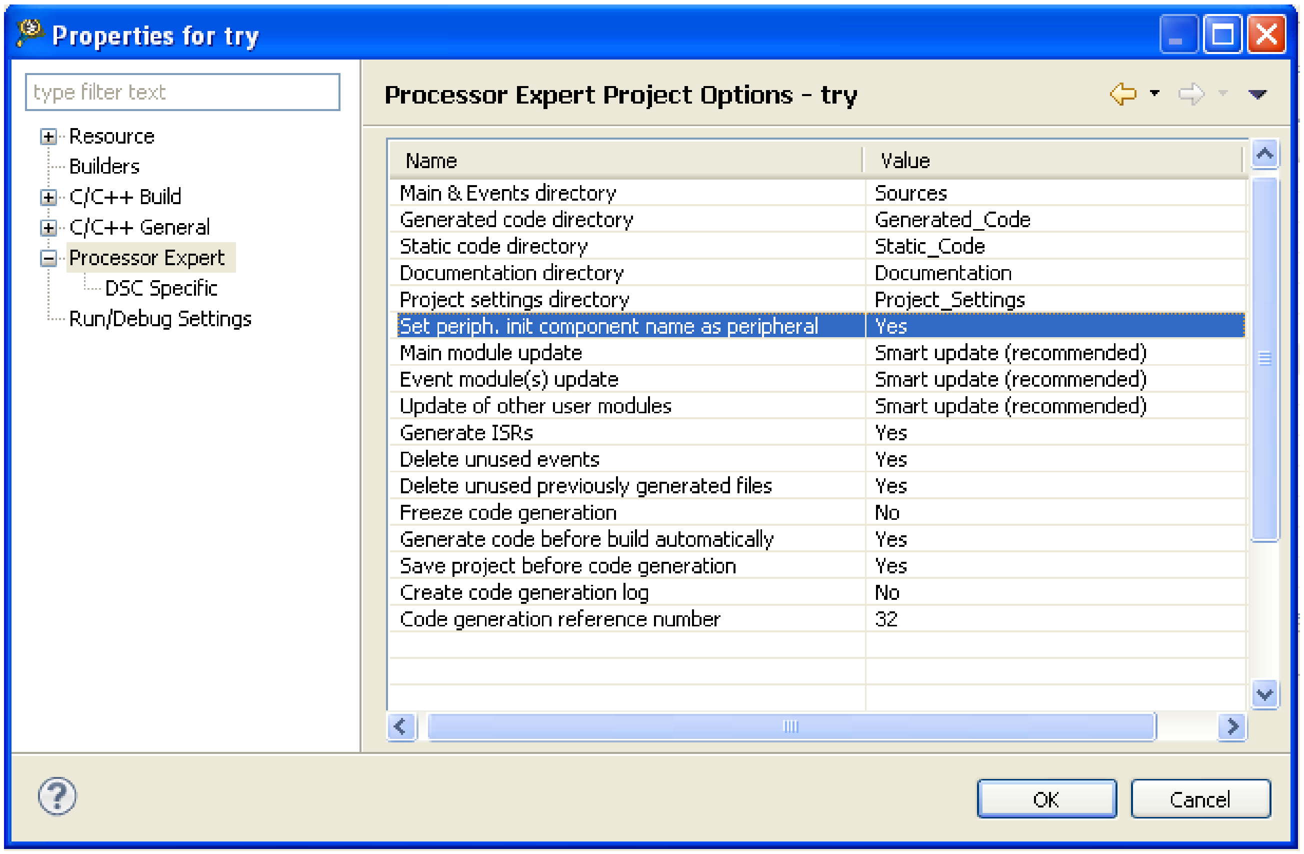
It contains the standard commands with the Processor Expert - specific command, **Generate Processor Expert Code**, which invokes code generation for the current project. The generated files are automatically inserted into the active (default) target in the QorIQ Configuration project. Generated files corresponding to the embedded components can be accessed from the **Generated_Code** folder. For more details, see Code generation and usage on page 46.

2.1.2 Project options

This section describes how to view the project options specific to Processor Expert.

Project options related to Processor Expert can be found in the **Properties** window. To access this window, click **Project > Properties**. The **Properties** window appears.

Select **Processor Expert** option in the list on the left. Description of individual options can be found in the hint window displayed when the cursor is placed on an item.



Processor Expert Project Options - try

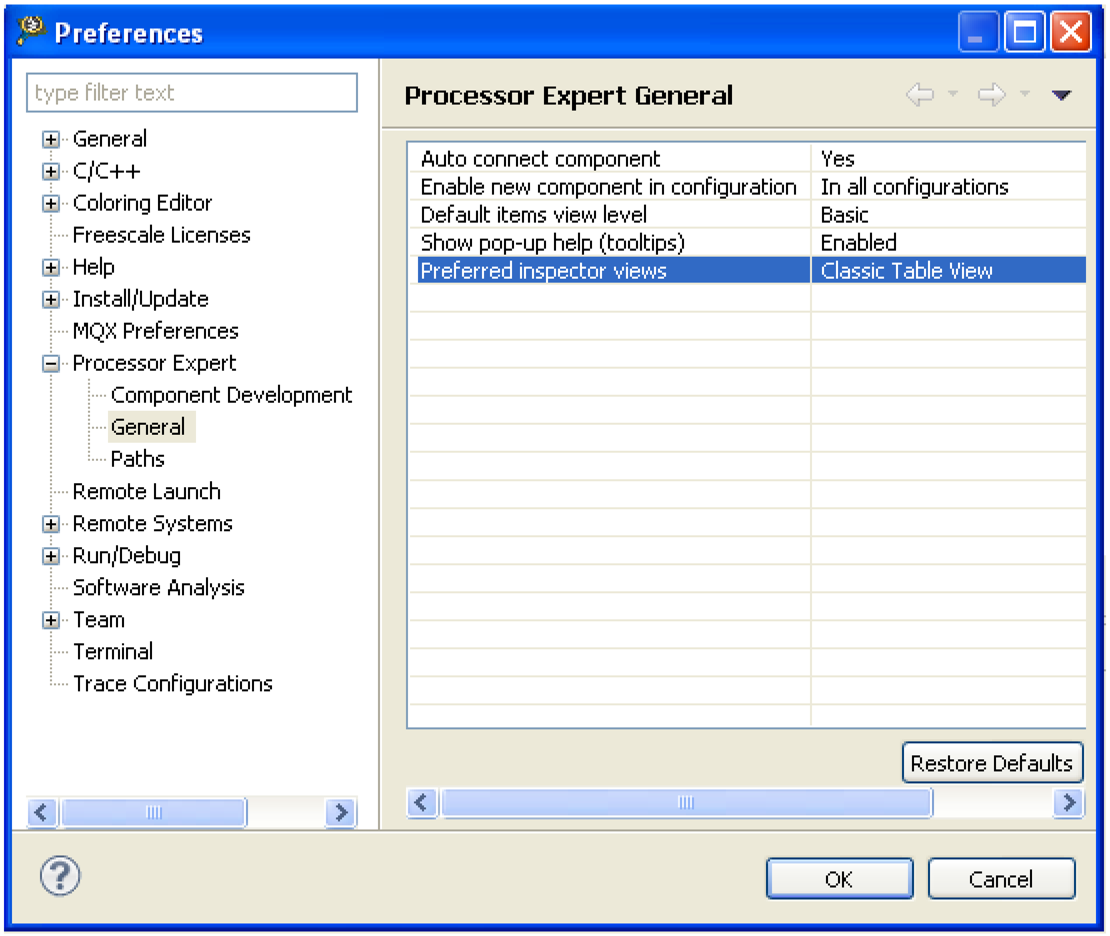
Name	Value
Main & Events directory	Sources
Generated code directory	Generated_Code
Static code directory	Static_Code
Documentation directory	Documentation
Project settings directory	Project_Settings
Set periph. init component name as peripheral	Yes
Main module update	Smart update (recommended)
Event module(s) update	Smart update (recommended)
Update of other user modules	Smart update (recommended)
Generate ISRs	Yes
Delete unused events	Yes
Delete unused previously generated files	Yes
Freeze code generation	No
Generate code before build automatically	Yes
Save project before code generation	Yes
Create code generation log	No
Code generation reference number	32

Figure 1. Properties window

2.1.3 Preferences

This section describes how to view the global settings specific to Processor Expert.

Global settings related to Processor Expert can be found in the **Preferences** window available using the command **Window > Preferences**. The Processor Expert - related items can be found under Processor Expert in the list on the left. Description of the individual options can be found in the hint window displayed when the cursor is placed on an item.

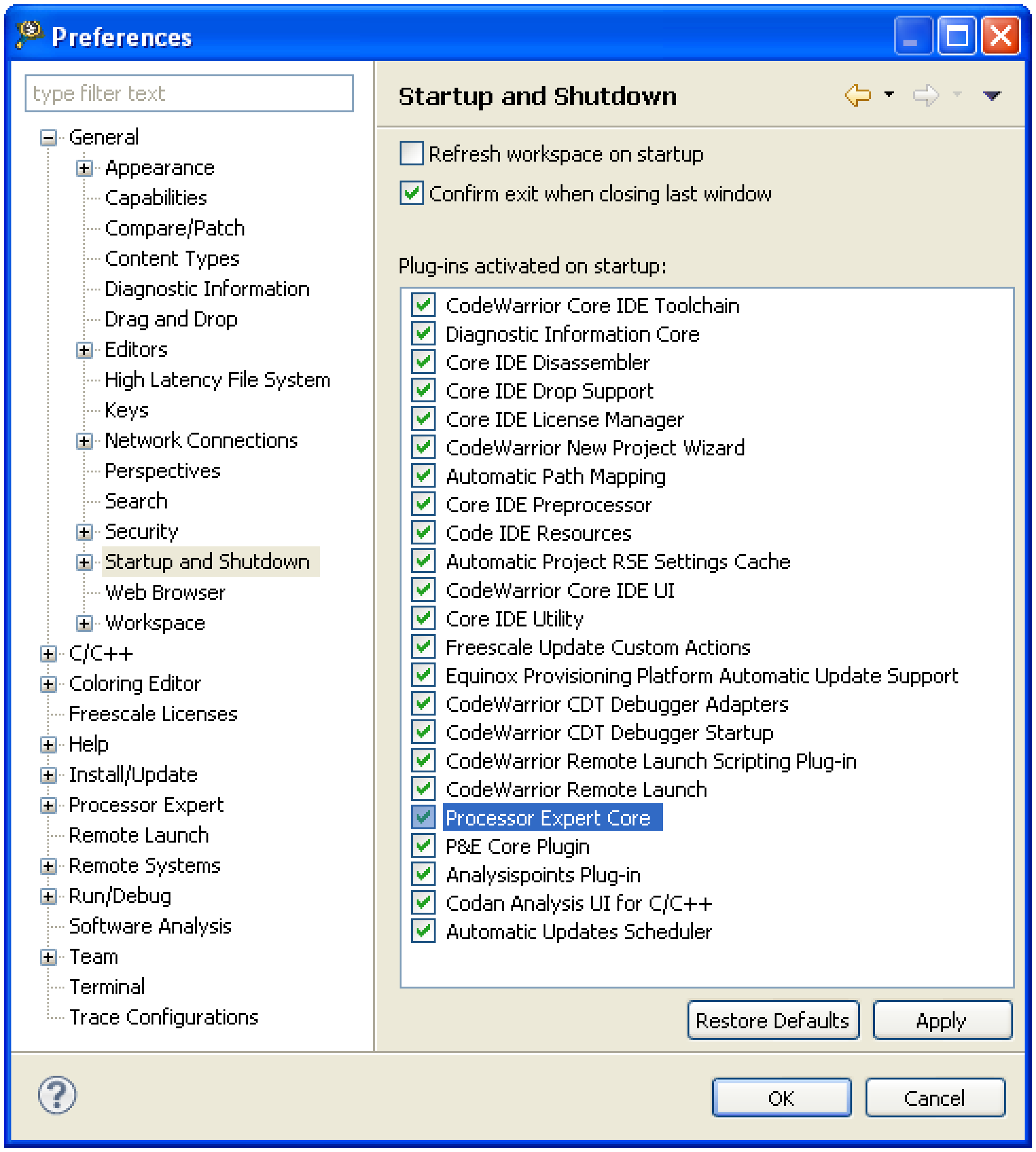


Auto connect component	Yes
Enable new component in configuration	In all configurations
Default items view level	Basic
Show pop-up help (tooltips)	Enabled
Preferred inspector views	Classic Table View

Figure 2. Preferences window

There is an option **Preferred inspector views** that allows you to decide how to display the tabs of **Component Inspector** view. There are two views Custom and Classic.

To start or shutdown the processor expert, click **Windows > Preferences** and expand **General** and select **Startup and Shutdown**. Processor Expert starts after the Eclipse workbench user interface is displayed if the **Processor Expert Core** checkbox is selected as shown below.



Startup and Shutdown

- [ ] Refresh workspace on startup
- [✓] Confirm exit when closing last window

Plug-ins activated on startup:

- [✓] CodeWarrior Core IDE Toolchain
- [✓] Diagnostic Information Core
- [✓] Core IDE Disassembler
- [✓] Core IDE Drop Support
- [✓] Core IDE License Manager
- [✓] CodeWarrior New Project Wizard
- [✓] Automatic Path Mapping
- [✓] Core IDE Preprocessor
- [✓] Code IDE Resources
- [✓] Automatic Project RSE Settings Cache
- [✓] CodeWarrior Core IDE UI
- [✓] Core IDE Utility
- [✓] Freescale Update Custom Actions
- [✓] Equinox Provisioning Platform Automatic Update Support
- [✓] CodeWarrior CDT Debugger Adapters
- [✓] CodeWarrior CDT Debugger Startup
- [✓] CodeWarrior Remote Launch Scripting Plug-in
- [✓] CodeWarrior Remote Launch
- [✓] Processor Expert Core
- [✓] P&E Core Plugin
- [✓] Analysispoints Plug-in
- [✓] Codan Analysis UI for C/C++
- [✓] Automatic Updates Scheduler

Figure 3. Startup and Shutdown page of Preferences window

2.2 Components view

The Components view displays the components of various embedded components.

To open this view, click **Window > Show View > Other** and select **Processor Expert > Components**.

Components view shows the tree with the following items:

- **Generator_Configurations**: Configurations of the project.
- **Operating System**: Contains special components that provide operating system interface and configuration if there are any used
- **Processors**: Contains processor components included in the project

- **Components**: It is included in the project. Every component inserted in the project is displayed in the **Component Inspector** view and may have a sub tree showing items available for the component (note that components can offer only some or even none of these items):
  - **Methods**: Methods allow run-time control of the component's functionality
  - **Events routines**: Events allow handling of the hardware or software events related to the component. If the event is disabled, the name of the event is shown. For enabled events, the name of the handling function is shown.
  - **ISRs**: Represent component-related interrupt routines that is created by you for low-level interrupt processing. For items, whose ISR names have been specified within component settings, a user-specified name of an ISR and name of the interrupt vector is shown. If an ISR name is not specified (interrupt has to be disabled in this case), only the interrupt vector name is present.

All component's items have status icons that signify the enabled or disabled state. If this state cannot be changed directly, the background of the icon is gray. For more details, see Embedded components on page 29.

This table explains the various states of a component.

Table 1. Description of Component States

Component Status Icon	Description
✓	Signifies that component is enabled. It can be configured and code can be generated from this component.
✗	Signifies that component is disabled. It can be configured, but the configuration validation/ generation is disabled. No code is generated from this component.
❎	Signifies error in the component. For example, Components folder contains component with error.
🔹	Signifies that component is frozen and will not be re-generated. When the user generates the code again, files of this component are not modified and the generated code is frozen.

Shared components are automatically placed into a dedicated subfolder **Referenced_Components**. You can move the component from this folder to anywhere.

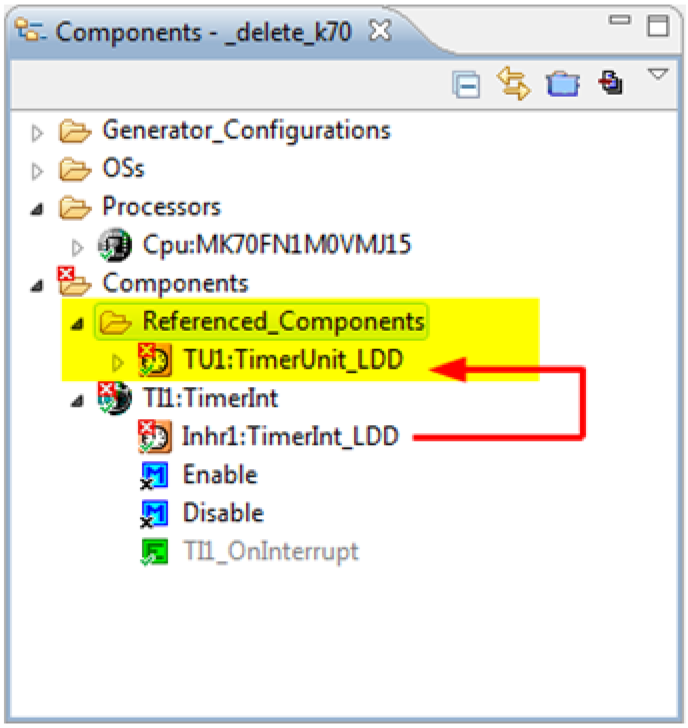


Figure 4. Referenced components

When you have more than one Processor Expert project in your workspace and you are working with those projects, the last project shown in **Components** view is recorded in the workspace history. When you restart the Eclipse IDE, the last session project is opened automatically.

This section contains the following subsections:

- View menu on page 11
- Pop-up menu on page 11

2.2.1 View menu

This section explains the options of the View menu of the Components view.

The options are:

- **Generate Processor Expert Code**: Invokes code generation for the current project
- **Close/Open Project**: Closes the project if it is open or opens the project if it is closed
- **Properties**: Displays the Processor Expert properties for a specific project
- **Export**: Allows to export component settings or configuration of selected Processor Expert components
- **Import**: Allows to import component settings or configuration of selected Processor Expert components

2.2.2 Pop-up menu

This section explains the options of the pop-up menu of the Components view.

The options are:

- **Inspector**: Opens **Component Inspector** view for the component. For more details, see Component Inspector view on page 15.

- **Inspector - Pinned**: Opens **Component Inspector** view for the component in "pinned" mode. This command allows to have several inspector views for different components opened at once. For more details, see Component Inspector view on page 15.
- **Code Generation**: Allows to disable/enable the generated module for the component and CPU component
- **Configuration Registers**: Displays the **Configuration Registers** view for the peripheral initialized by the selected component. For more details, see Configuration Registers view on page 26.
- Target Processor Package: Displays the **Processor** view for the processor derivative used in a given project
- Processor Memory Map: Displays the **Memory Map** view for the processor address space and internal and external memory mapping
- Rename Component: Changes the name of the component
- **Select Distinct/ Shared mode**: Switches between shared and distinct mode of the component. This setting is available for LDD components only.
- **Open File**: Opens the generated code from the selected component for the source code editor. Note that the code is available only after successful code generation.
- **Component Enabled**: Enables/disables component in the project
- **Copy/Paste**: Components can be copied and pasted inside the same project, between two projects in the same workspace, and between two projects having each project in separate workspace.
- **Remove component from project**: Deletes the component from the project
- **Help on component**: Shows a help page for the component
- **Save Component Settings As Template**: Creates a template of the selected component. For more details, see Creating User Component Templates.

NOTE
Hot key **Delete** is used for deleting components from the project.

- **View Code**: Opens code editor at the code of the selected method or event
- **Toggle Enable/Disable**: Enables/Disables the Method or Event

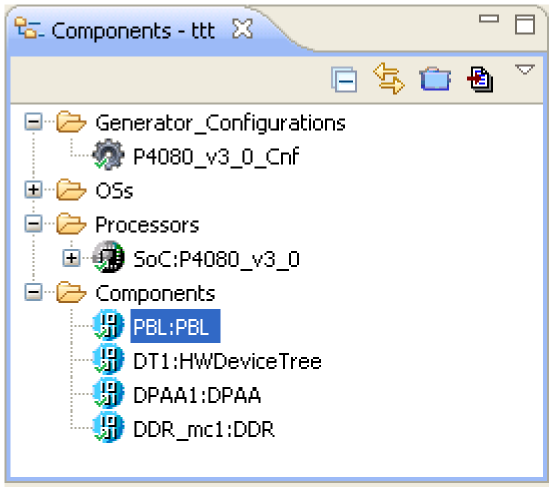


Figure 5. Components view

2.3 Components Library view

The Components Library view allows you to select a component and add it to the project.

This view allows you to select a component and add it to the project. To open this view, click **Window > Show View > Other** and select **Processor Expert > Components Library**.

The **Components Library** view shows supported embedded components including processor components and component templates. It lets you select a desired component or template and add it to the project.

This section contains the following subsections:

- Modes on page 13
- Filtering on page 13
- Pop-up menu on page 13
- Component Assistant on page 14

2.3.1 Modes

This section explains the various modes of Component Library.

The Components Library has the following four tabs allowing you to select components in different modes:

- **Categories**: Contains all available components. The components are sorted in a tree based on the categories defined in the components. For more details, see Component categories on page 30.
- **Alphabetical**: Shows alphabetical list of the available components
- **Assistant**: Guides you during the component selection process. The user answers a series of questions that finally lead to a selection of a component that suits best for a required function. For more details, see Component Assistant on page 14.
- **Processors**: Contains list of the available processors

Component names are colored black and the component template names are colored blue. The components that are not supported for the currently selected target processor are gray. By double-clicking on the component, it is possible to insert the component into the current project. The description of the component is shown in a hint.

2.3.2 Filtering

This section explains how to filter the components displayed in the Components Library view.

Filtering can be activated or deactivated by selecting or deselecting the filtering icon. If the filtering icon is selected, only the components that could be used with the currently selected target processor are shown.

If the filtering icon is deselected, then Processor Expert also shows components that are not available for the current processor.

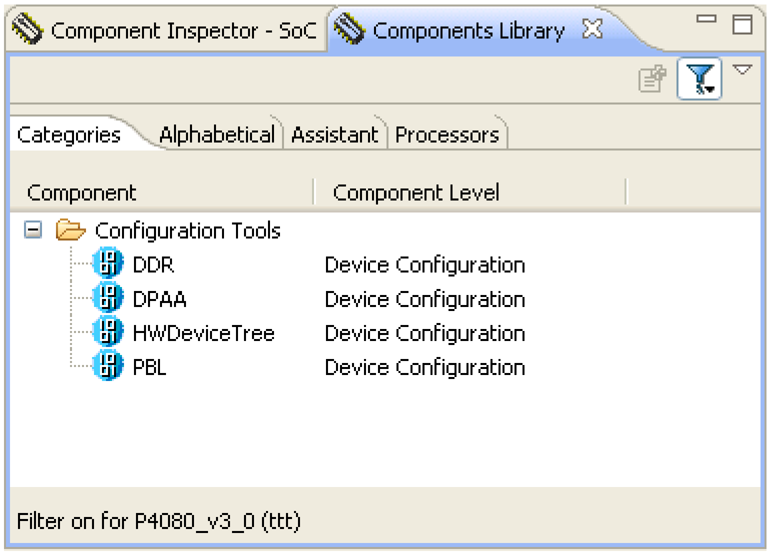
2.3.3 Pop-up menu

This section explains the options of the pop-up menu of the Components Library view.

A pop-up menu opens by right-clicking a component or folder. It contains the following commands:

- **Add to project**: Adds the component to the current project
- **Add to project with wizard**: Adds the component to the current project and opens a configuration wizard
- **Expand all**: Expands the folder and all its subfolders
- **Collapse all**: Collapses the folder and all its subfolders

- **Refresh**: Refreshes the view area
- **Delete**: Only user templates and components are deleted. User component is deleted from the User Working Directory, from the Beans sub-directory. See Processor Expert Eclipse console for the location of the User Working Directory. Other files like `*.inc`, `*.drv`, `*.src` remain intact.
- **Help on component**: Opens help information for the selected component



Component	Component Level
Configuration Tools	
DDR	Device Configuration
DPAA	Device Configuration
HWDeviceTree	Device Configuration
PBL	Device Configuration

Filter on for P4080_v3_0 (ttt)

Figure 6. Component Library view

2.3.4 Component Assistant

The section explains the Component Assistant, which helps you select components while creating basic application building blocks.

The Component Assistant is a mode of **Components Library** view. You will have to answer a series of questions that finally lead to a selection of a component that suits best for a required function. In this mode, the **Components Library** view consists of the following parts:

- History navigation buttons and the history line showing answers for already answered questions. You can walk through the history using the arrow buttons or by clicking the individual items.
- A box with a current question
- A list of available answers for the current question.

  If the answer already corresponds to a single component (it has an icon of the component and there is a [component name] at the end of the list line) and user double-clicks it, it is added into the project. Also, you can right-click on the line to open the pop-up menu of the component, allowing to add it into the project or view its documentation (for details, see Components Library view on page 13).

  If more questions are necessary for the component selection, the line with the answer contains a group icon and in brackets a number of components that still can possibly be selected. After clicking on such line a next question is displayed.

This mode of Components Library does not offer addition of processor components. If you would like to add another processor component, switch to the Processors tab.

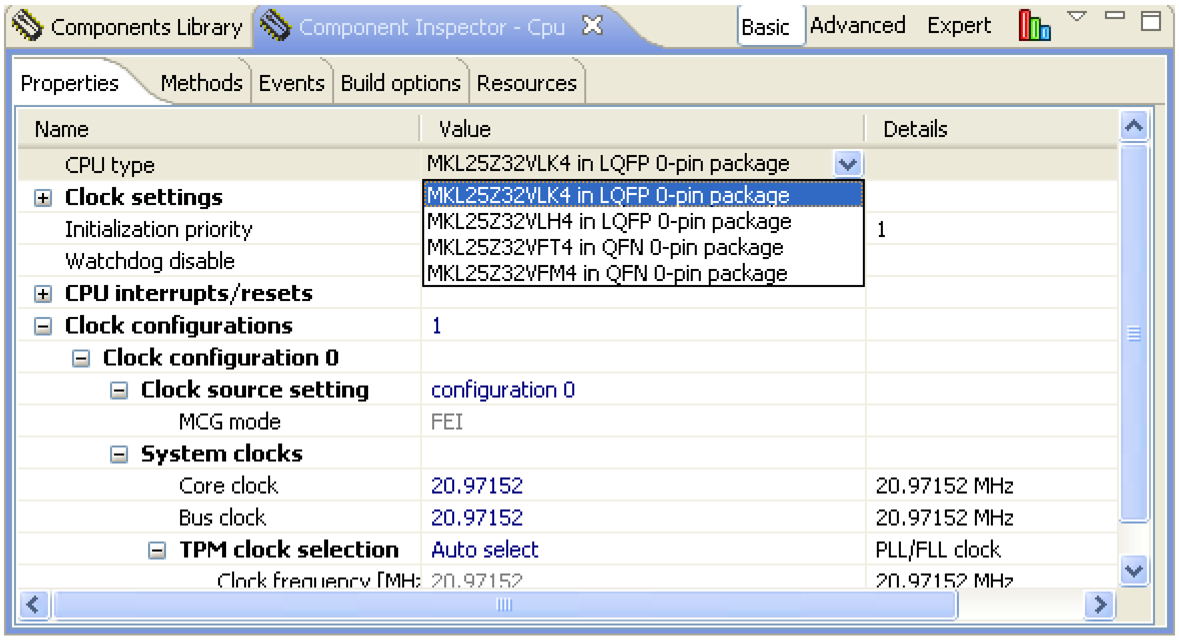
2.4 Component Inspector view

The Component Inspector view allows you to view and edit attributes of the item selected in the Project Explorer.

To open this view, choose **Window > Show View > Other** from the CodeWarrior menu bar and then select **Processor Expert > Components Inspector**.

Inspector window contains three columns:

- **Name**: Name of the item to be selected. Groups of items may be collapsed or expanded by double clicking on the first line of the group with its name, it has '+' or '-' sign on the left.
- **Value**: The settings of the items are made in this column. For list of item types, see Inspector items on page 18 for details.
- **Details**: The current setting or an error status may be reflected on the same line, in the rightmost column of the inspector



Name	Value	Details
CPU type	MKL25Z32VLK4 in LQFP 0-pin package	
⊞ **Clock settings**	MKL25Z32VLK4 in LQFP 0-pin package	
Initialization priority	MKL25Z32VLH4 in LQFP 0-pin package	1
Watchdog disable	MKL25Z32VFT4 in QFN 0-pin package	
⊞ **CPU interrupts/resets**	MKL25Z32VFM4 in QFN 0-pin package	
⊟ **Clock configurations**	1	
⊟ **Clock configuration 0**		
⊟ **Clock source setting**	configuration 0	
MCG mode	FEI	
⊟ **System clocks**		
Core clock	20.97152	20.97152 MHz
Bus clock	20.97152	20.97152 MHz
⊟ **TPM clock selection**	Auto select	PLL/FLL clock
Clock frequency [MH:	20.97152	20.97152 MHz

Figure 7. Component Inspector view - Displaying pin variant and package

This section contains the following subsections:

- Read-only items on page 16
- View mode buttons on page 16
- View menu on page 16
- Graphical mode on page 16
- Pop-up menu on page 17
- Inspector items on page 18
- Items visibility on page 20
- Configuration Inspector on page 20

2.4.1 Read-only items

The values that are read-only appear in gray.

Some items are read-only so you can not change the content. Such values are gray.

2.4.2 View mode buttons

The buttons allow you to switch to the complex view of the component's items.

They are placed at the top of the window (Basic, Advanced, Expert). See Items visibility on page 20 for details.

2.4.3 View menu

The View menu of Component Inspector is invoked by clicking on the down-arrow icon.

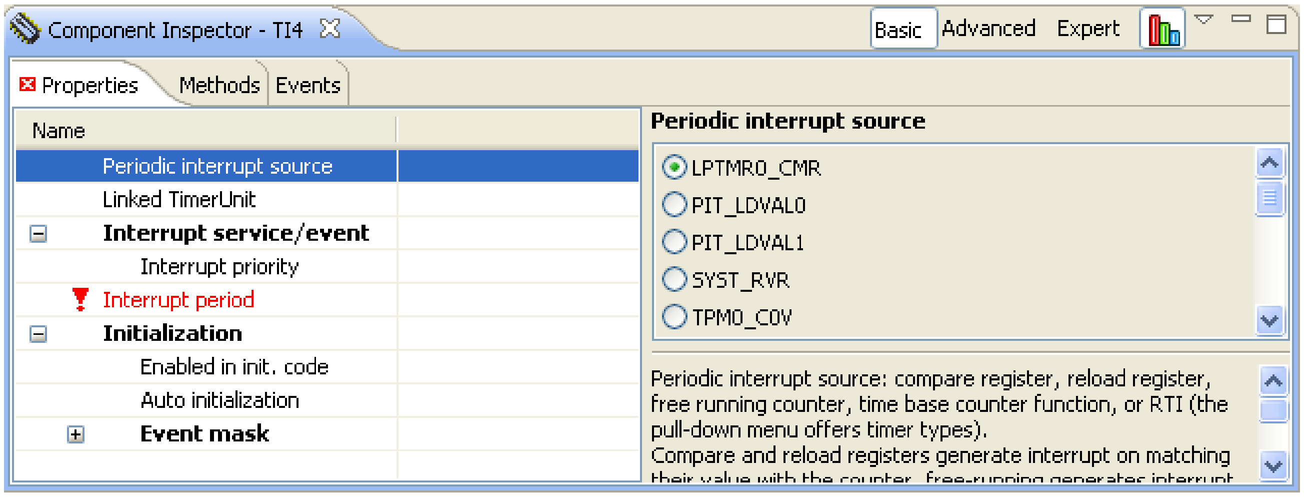
The menu contains the following options:

- **Basic, Advanced, Expert**: View mode switching. These options have the same meaning as the view mode buttons.
- **Ignore Constraints and non-Critical Errors**: This option enables a special mode of Processor Expert. In this mode, Processor Expert allows you to generate output files, even though some settings may break some constraints or limits and errors are reported. The mode also allows you to set some enumeration of numeric properties to a custom value out of allowed range. Use the Graphical Mode of Component Inspector to enter the custom value.
- **Expand All**: If a group is selected, expands all items within the selected group. Otherwise, all groups in the Inspector are expanded. If the expanded group contains any groups that are disabled (gray), the user is asked if the disabled groups should all be expanded.
- **Collapse All**: If a group is selected, collapses all items within the selected group. Otherwise, all groups in the Inspector are collapsed.
- **Help on Component**: Shows a help page for the component
- **Save component settings as template**: Creates a template for the current component settings. See Creating user component templates on page 32 for details.
- **Open pinned view**: Opens a copy of the inspector for currently selected component. This command allows to have several inspector views for different components opened at once.
- **Search**: Searches Inspector item by name. It also accepts wild cards like * or ? (* =any string and ? = any character).

2.4.4 Graphical mode

The graphical mode can be switched on/off using the Graphical Mode toolbar button.

When you click on the **Graphical Mode** button, the inspector view is split into two panels. The left contains a list of items (properties) and the right one allows an alternative way of editing the property selected on the left.



Periodic interrupt source: compare register, reload register, free running counter, time base counter function, or RTI (the pull-down menu offers timer types).
Compare and reload registers generate interrupt on matching

Figure 8. Component Inspector with Graphical mode on

2.4.5 Pop-up menu

The pop-up menu is invoked by right-clicking a specific Component Inspector item.

The menu contains the following options:

- **Expand All**: If a group is selected, expands all items within the selected group. Otherwise, all groups in the inspector are expanded. If the expanded group contains any groups that are disabled (gray), the user is asked if the disabled groups should all be expanded.
- **Collapse All**: If a group is selected, collapses all items within the selected group. Otherwise, all groups in the inspector are collapsed.
- **Help on Component**: Shows a help page for the component
- **Delete Item**: Does not delete the component, but can delete the property item from the list of property. The list of items can have some constraints on minimal or maximum number of items. Add ADC component into the project and add at least one extra channel then you will be able to see this option enabled as shown below.

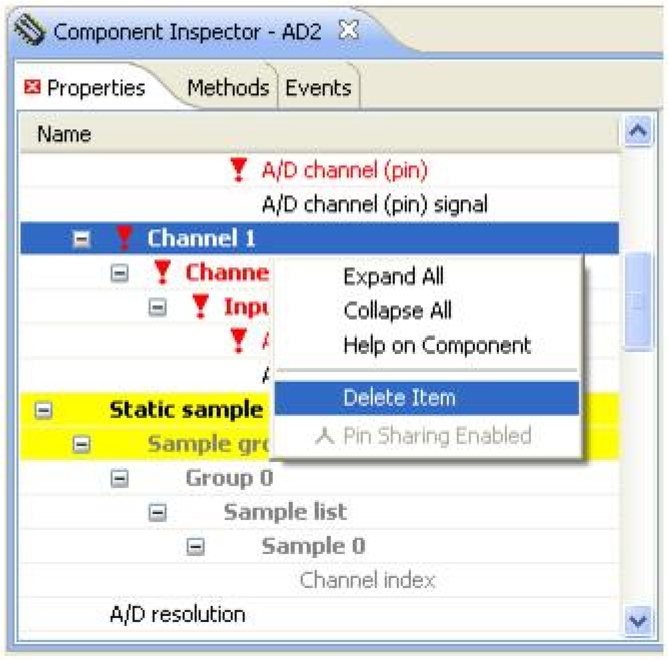


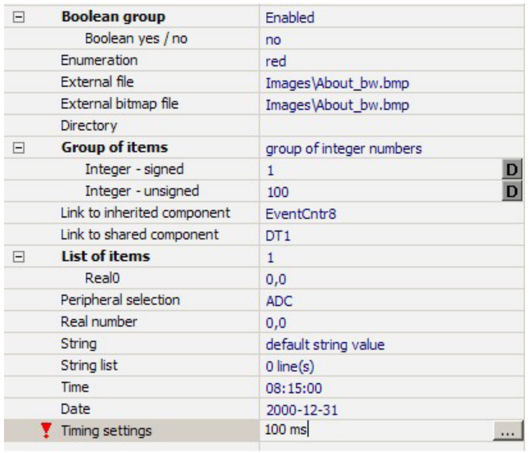
Figure 9. Delete item

- **Pin Sharing Enabled**: Enables the pin sharing. This command is available only for pin properties. For more information, see Sharing pins among peripherals on page 36.

2.4.6 Inspector items

This section describes the various types of Inspector Items.

The following types of the items are there in the **Component Inspector** view.



Boolean group	Enabled
Boolean yes / no	no
Enumeration	red
External file	Images\About_bw.bmp
External bitmap file	Images\About_bw.bmp
Directory	
Group of items	group of integer numbers
Integer - signed	1
Integer - unsigned	100
Link to inherited component	EventCntr8
Link to shared component	DT1
List of items	1
Real0	0,0
Peripheral selection	ADC
Real number	0,0
String	default string value
String list	0 line(s)
Time	08:15:00
Date	2000-12-31
Timing settings	100 ms

Figure 10. Example component with various Inspector item types

This table explains the various types of items.

Table 2. Inspector item types

Field	Description
Boolean Group	A group of settings controlled by this boolean property. If the group is enabled, all the items under the group are valid; if it is disabled, the list of items is not valid. Clicking + sign will show/hide the items in the group but doesn't affect value or validity of the items.
Boolean yes/no	You can switch between two states of the property using a drop-down menu. The Generate code/Don't generate code settings of methods and events works the same way and determines whether the implementation code for the corresponding method or event will be generated or not (you may thus generate only the methods and events used by your application).
Enumeration	Selection from a list of values. If you click the arrow icon ( ), a list of the possible values for the property is offered. For some derivatives, pin and package details are displayed for processor variant.
Enumeration Group	A list of items. Number of visible (and valid) items in the group depends on chosen value. Clicking the arrow icon ( ) will show a list of the possible values of the property. Clicking the + sign shows/hides the items in the group but does not influence value or validity of the items.
File/Directory Selection	Allows to specify a file or directory. Clicking the icon opens a system dialog allowing to select a file/directory.
Group	A list of items that can be expanded/collapsed by clicking on the plus/minus icon or by double-clicking at the row. Values of the items in the group are untouched.
Integer Number	You can insert a number of a selected radix. Radix of the number could be switched using the icons (D = Decimal, H = Hexadecimal, B = Binary). Only reasonable radixes are offered for the property. If the radix switching icon is not present, Processor Expert expects the decimal radix.
Link to Inherited component	The down-arrow button allows to change the ancestor from the list of possible ancestor. See Component inheritance and component sharing on page 34 for details.
Link to shared component	The down-arrow button allows to change the component from the list of the available components or add a new component to the project. See Component inheritance and component sharing on page 34 for details.
List of items	A list of items may be expanded/collapsed by clicking on the plus/minus button in the left side of the row or by double clicking on the row. You may add/remove items by clicking on the plus/minus button. The items in the list can be arranged using the Pop-up menu on page 17 related commands.
Peripheral selection	You can select a peripheral from the list of the available peripherals. The peripheral that are already allocated have the component icon in the list. The properties that conflicts with the component settings have the red exclamation mark.
Real Number	You can insert any real (floating point) number.

Table continues on the next page...

Table 2. Inspector item types (continued)

Field	Description
String	Allows to enter any text or value. If there are no recommended values specified, there is no change in the user interface. If the values are specified, these values helps user to select typical or recommended value for the property. This feature is supported for String and Real Number properties. The recommended values used for the property are ONE, TWO, and THREE. The string property can offer predefined values accessible hitting key stroke `ctrl+space`. You can see that this is available when the string value is edited and there are predefined values available, small yellow bulb is displayed before top-left corner of edit field.
String list	Clicking the browse button (...) opens the simple text editor that allows to enter an array of text lines.
Time, Date	Allows to setup the Time/Date in a format according to the operating system settings.
Timing settings	Allows a comfortable setting of the component's timing. The timing dialog opens on clicking on browse button (...).

2.4.7 Items visibility

The Processor Expert supports three visibility levels of the component items.

Each item is assigned a predefined level of visibility. **Higher visibility level** means that items with this level are more special and rarely used than the others with the lower visibility level. Component Inspector displays only items on and below the selected level. It could help especially beginners to set only basic properties at first and do optimization and improvements using advanced and expert properties or events later. There are three visibility levels:

- **Basic view**: The key and most often used items that configure the basic functionality of the components.
- **Advanced view**: All items from **Basic** view and the settings that configure some of more advanced and complex features of the component.
- **Expert view**: Maximum visibility level - All possible settings, including all settings of basic and advanced view.

The visibility can be switched in the Component Inspector View using **Basic**, **Advanced**, and **Expert** buttons or within its view menu.

NOTE

If an error occurs in a property with a higher visibility level than the level currently selected, then also this error is displayed.

2.4.8 Configuration Inspector

Configuration Inspector is a variant of Component Inspector.

It displays the settings of a selected component. It could be invoked from configurations pop-up menu in the **Components** view (right-click a configuration and choose **Configuration Inspector**). For details on configurations, see Configurations on page 31.

This section contains the following subsection:

- Properties on page 21

2.4.8.1 Properties

The Properties tab of Component Inspector contains optimization settings related to the configuration.

These settings should be used when the code is already debugged. It could increase speed of the code, but the generated code is less protected for the unexpected situations and finding errors could be more difficult.

Note that some of the options may not be present for all Processor Expert versions.

- **Ignore range checking**: This option can disable generation of the code that provides testing for parameter range. If the option is set to `yes`, methods do not return error code `ERR_VALUE` neither `ERR_RANGE`. If the method is called with incorrect parameter, it may not work correctly.
- **Ignore enable test**: This option can disable generation of the code that provides testing if the component/peripheral is internally enabled or not. If the option is set to yes, methods do not return error code `ERR_DISABLED` neither `ERR_ENABLED`. If the method is called in unsupported mode, it may not work correctly
- **Ignore speed mode test**: This option can disable generation of the code, that provides a testing, if the component is internally supported in the selected speed mode. If the option is set to yes, methods do not return error code `ERR_SPEED`. If the method is called in the speed mode when the component is not supported, it may not work correctly.
- **Use after reset values**: This option allows Processor Expert to use the values of peripheral registers which are declared by a chip manufacturer as default after reset values. If the option is set to no, all registers are initialized by a generated code, even if the value after reset is the same as the required initialization value. If the option is set to yes, the register values same as the after reset values are not initialized.
- **Complete initialization in Peripheral Init. Component**: This option can disable shared initialization peripheral in Init methods of Peripheral Initialization Components. If this option is set to yes, the complete peripheral initialization is provided in Init method, even for parts that are already initialized in processor or elsewhere. It could mean longer code, but the initialization can be repeated in application using the Init method.

2.5 Processor view

The Processor view displays the selected target processor with its peripherals and pins.

It allows you to generate code from processor and also to switch the processor's package. To open this view, choose **Window > Show View > Other** from the CodeWarrior IDE menu bar and then select **Processor Expert > Processor**.

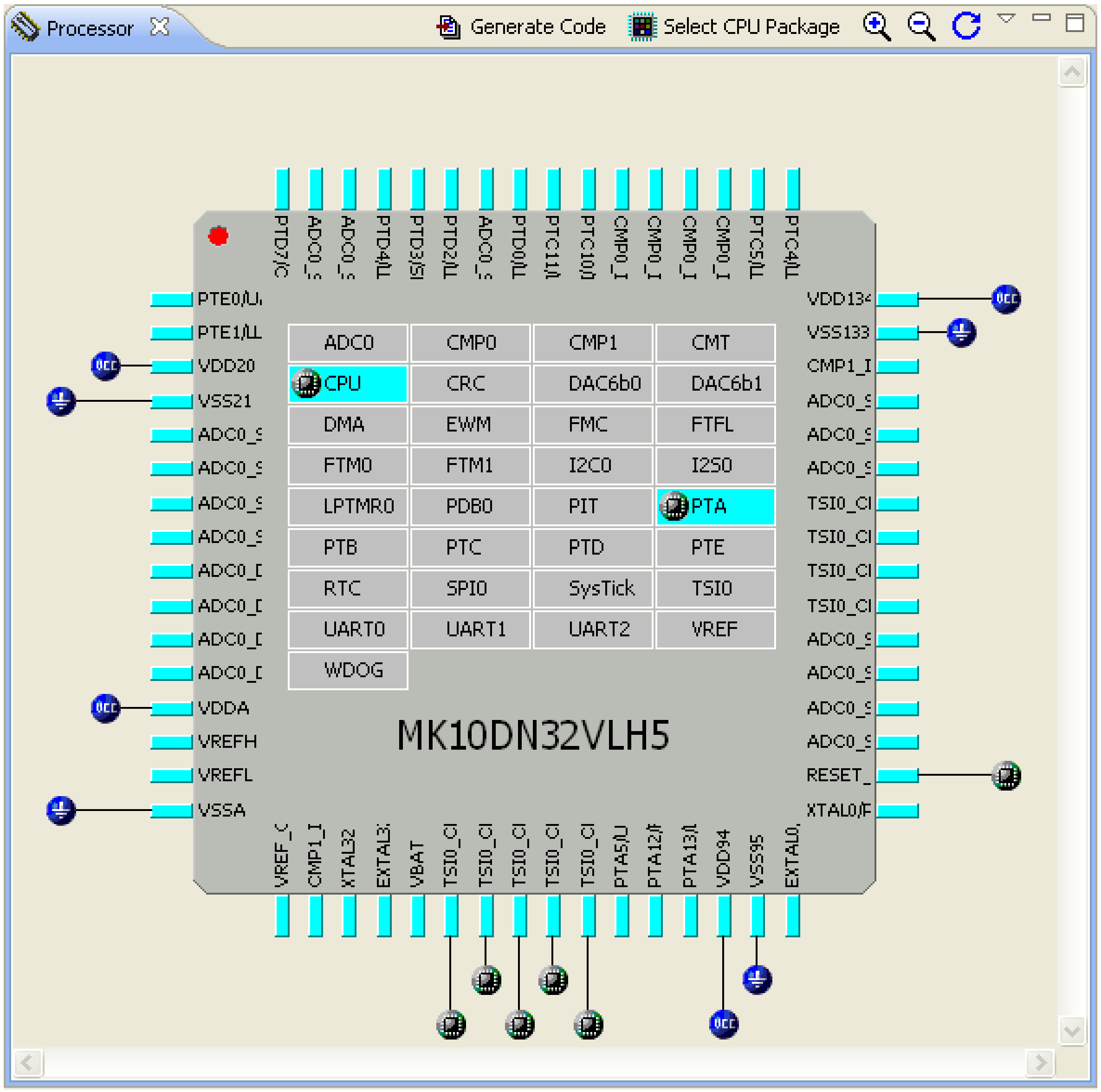


Figure 11. Processor view

This section contains the following subsection:

- Control buttons on page 22

2.5.1 Control buttons

This section explains the functionality of control buttons that appear in the Processor view.

The following table lists and describes the control buttons.

Table 3. Control buttons

Buttons	Description
🔍	Zoom in: Increases the detail level of the view. The whole picture might not fit the viewing area.

Table continues on the next page...

Table 3. Control buttons (continued)

Buttons	Description
🔍	Zoom out: Decreases the detail level of the view. Processor Expert tries to fit the whole picture to the viewing area.
↻	Rotate: Rotates the package clockwise
▦	Resources (*available for BGA type packages only*): Selects Resources view mode that shows a top side of the package without pins but including list of peripherals and showing their allocation by components.
▦	Pins Bottom (*available for BGA type packages only*): Selects Pins view mode that shows a bottom side of the package with pins. The peripherals are not shown in this mode because the surface is covered with pins.
▦	Pins Top: Select Pins view mode that shows a top side of the package with pins

This section contains the following subsections:

- Pins on page 23
- Hints on page 23
- Shared pins on page 24
- On-chip peripherals on page 24

2.5.1.1 Pins

This section provides information about each pin displayed on the processor.

The following information about each pin: (in case of BGA type package the pins are displayed only in the Pins view mode)

- Pin name (default or user-defined)
- Icon of a component that uses (allocates) the pin
- Direction of the pin (input, output, or input/output) symbolized by blue arrows if a component is connected
- With new pin model (supported for few derivatives only), the background color of the pin reflects routing of the pin to the peripheral.

Pin names are shortened and written either from left to right or from top to bottom and are visible only if there is enough space in the diagram.

Some signals and peripherals cannot be used by the user because they are allocated by special devices such as power signals, external, or data bus. The special devices are indicated by a special blue icons. The allocation of peripherals by special devices can be influenced by processor component settings.

In case of BGA package, the pins that are used by some component are colored yellow. Move the cursor on the pin to get detailed information.

2.5.1.2 Hints

This section describes the hints displayed for pins and components.

A hint displays the basic information about a pin or component.

Pin hint contains:

- Number of the pin (on package)
- Both names (default and user-defined)

- Owner of the pin (component that allocates it)
- Short pin description from processor database

Component icon hint contains:

- Component name
- Component type
- Component description

2.5.1.3 Shared pins

This section explains how a pin is shared among peripherals.

If a pin is shared by multiple components, the line connecting the pin to the component has a red color. See Sharing pins among peripherals on page 36 for details.

2.5.1.4 On-chip peripherals

This section describes the peripherals added to a chip.

The basic information about each on-chip peripheral is displayed on the processor package. The following information is displayed on the processor package:

- Peripheral device name (default or user-defined)
- Icon of the component that uses (allocates) the peripheral device

Peripheral device hint contains:

- Peripheral device name
- Owner of the pin (component that allocates it)
- Short peripheral device description

Hint on icon contains:

- Component name
- Component type
- Component description

If a peripheral is shared by several components (for example, several components may use single pins of the same port), the icon is displayed.

NOTE

Some peripherals work in several modes and these peripherals can be represented by a several devices in the processor databases. For example, the device `TimerX_PPG` and `TimerX_PWM` represents `TimerX` in the PPG and PWM mode. These devices can be displayed on the processor package, but they are also represented as a single block in the microcontroller block diagram.

This section contains the following subsection:

- Peripheral or pin pop-up menu on page 24

2.5.1.4.1 Peripheral or pin pop-up menu

The peripheral or pin pop-up menu displays peripheral or pin specific pop-up options.

The following options are available in the pop-up menu:

- **Show Peripheral Initialization**: Shows initialization values of all control, status, and data registers. This option is supported for all devices displayed on a processor package. See Configuration Registers view on page 26 for details.
- **Zoom in**: Increases the detail level of the view. The whole picture might not fit the viewing area.

- **Zoom out**: Decreases the size of the picture and detail level of the view
- **Rotate**: Rotates the package by 90 degrees
- **Add Component/Template**: Adds a component or template for the appropriate peripheral; all available components and templates suitable for the selected peripheral are listed. The components and templates in the list are divided by a horizontal line. It is possible to add only components or templates which are applicable for the peripheral. It means that is possible to add the component or template only if the peripheral is not already allocated to another component or components. The components/templates that cannot be added to the peripheral are grayed in the pop-up menu as unavailable. This option is supported for all devices displayed on processor package.
- **Remove Component**: Allows to remove all components allocating peripheral in **Processor** view. Processor component cannot be removed.

2.6 Memory Map view

The Memory Map view provides detailed information about the memory area.

To open this view, choose **Window > Show View > Other** from CodeWarrior IDE menu bar and then select **Processor Expert > Memory Map**.

The figure below shows the processor address space and internal and external memory mapping. Detailed information for an individual memory area is provided as a hint when you move the cursor over it.

Table 4. Legends

Legend	Description
⬜	White: Non-usable space
🟦	Dark blue: I/O space
🟦	Blue: RAM
🟦	Light blue: ROM, OTP, or Firmware
🟩	Cyan: Flash memory or EEPROM. This area can also contain a flash configuration registers area.
⬛	Black: External memory

The address in the diagram is increasing upwards. To improve the readability of the information, the size of the individual memory blocks drawn in the window are different from the ratio of their real size (small blocks are larger and large blocks are smaller).

The black line-crossed area shows the memory allocated by a component or compiler. The address axis within one memory block goes from the left side to the right (it means that the left side means start of the block, the right side means the end).

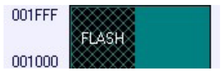


Figure 12. Sample used part of memory area

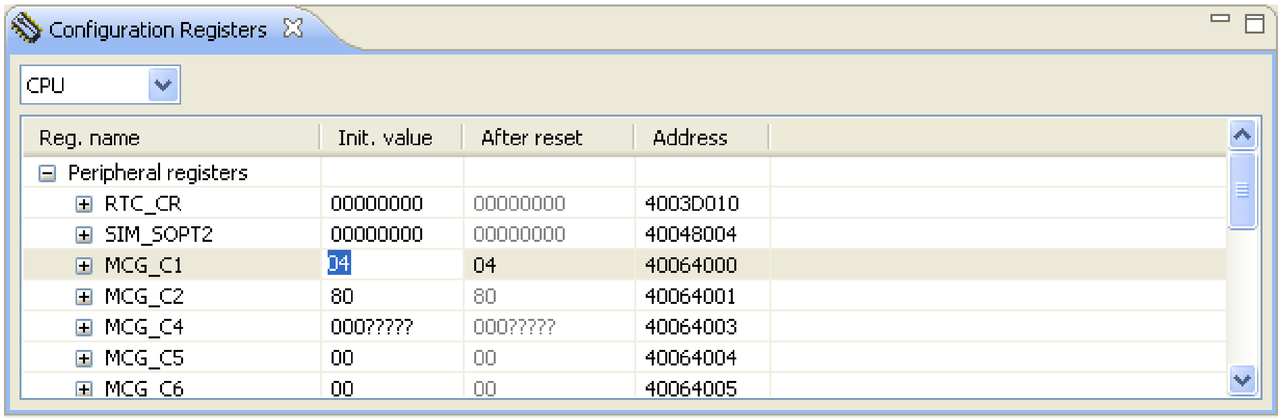
2.7 Configuration Registers view

The Configuration Registers view shows an overview of the peripheral initialization settings for the selected microcontroller target.

It displays initialization values of all control, status, and data registers of selected peripheral/device including single bits. The values are grouped into two parts: **Peripheral registers** containing registers directly related to the selected peripheral/device and **Additional registers** containing the registers that are influenced by the component but are not listed for the peripheral currently selected in this view. To open this view, choose **Window > Show View > Other** from the CodeWarrior IDE menu bar and then select **Processor Expert > Configuration Registers**.

The initialization information reflects:

- Processor default settings: When the peripheral is not utilized by any embedded component
- Embedded component settings: When the peripheral is utilized by an embedded component and the component settings are correct. Peripheral Initialization Inspector shows initialization as required by the component settings.



Reg. name	Init. value	After reset	Address
⊟ Peripheral registers			
⊞ RTC_CR	00000000	00000000	4003D010
⊞ SIM_SOPT2	00000000	00000000	40048004
⊞ MCG_C1	04	04	40064000
⊞ MCG_C2	80	80	40064001
⊞ MCG_C4	000?????	000?????	40064003
⊞ MCG_C5	00	00	40064004
⊞ MCG_C6	00	00	40064005

Figure 13. Configuration Registers view

The table shows the registers and their initialization value displayed in the column **Init. value**. You can modify the register value. Registers value can be changed if:

- They are not read-only and when the project is without errors
- Editing of configuration registers is supported by given component

This value written into the register or bit by the generated code during the initialization process of the application. It is the last value that is written by the initialization function to the register. The **After reset** column contains the value that is in the register by default after the reset.

The values of the registers are displayed in the hexadecimal and binary form. In case the value of the register (or bit) is not defined, an interrogation mark "?" is displayed instead of the value. The **Address** column displays the address of registers.

NOTE

For some registers, the value read from the register after sometime can be different than the last written value. For example, some interrupt flags are cleared by writing 1. For details, see the processor manual on registers.

In case the peripheral is allocated by a component and the setting of the component is incorrect, the initialization values are not displayed in the **Configuration Registers** view.

Chapter 3 Using Processor Expert

This chapter describes how to start using the Processor Expert tool and embedded components.

This chapter explains:

3.1 Create a project

This section explains how to create a Processor Expert QorIQ configuration project.

To create a new project, select **File > New > QorIQ Configuration Project** from the Eclipse IDE menu bar. Follow the steps in the **New QorIQ Configuration Project** wizard that appears, and select a processor you want to use. For details on creating a new Processor Expert project, see *QCVS Getting Started Guide*.

The Processor Expert views can be opened any time using the menu option, **Processor Expert > Show Views**.

This section contains the following subsections:

3.1.1 Select components

The components to be used are selected in the **Components Library** view.

You can add the selected component into the project by right-clicking it and selecting the **Add to project** option from the pop-up menu. The component gets added to the **Components** folder of the **Components** view.

For details on individual components, see tooltip that is displayed when cursor is placed on the component or use the pop-up menu command, **Help on Component**.

3.1.2 Configure components

The component to be configured is selected in the **Components** view.

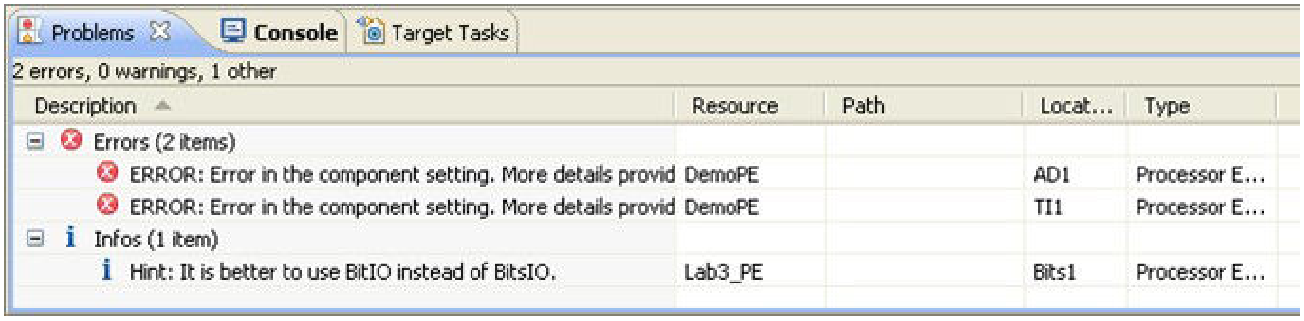
The component gets opened in the Component Inspector, see Component Inspector view on page 15. For details on how to configure components, see Configuring components on page 31 and also the user guide of each component.

3.1.3 Verify settings

You can verify your project settings using a standard Eclipse **Problems** view.

The **Problems** view displays Processor Expert errors, warnings, and status on project, code generation, and embedded components settings. If it is not open, select **Window > Show View > Other > General > Problems**.

The content of the view is continuously updated with the recent status. All error messages disappear when a project contains no settings with error. Double-clicking on an item takes you to the place where the particular problem occurred.



2 errors, 0 warnings, 1 other

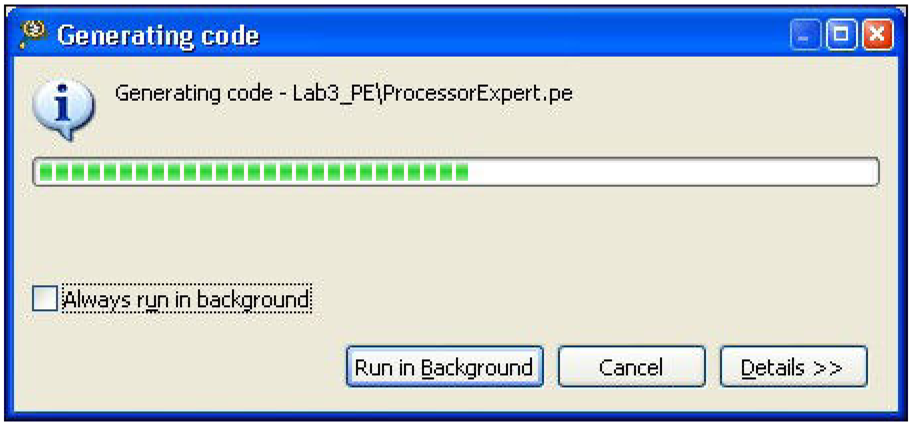
Description	Resource	Path	Locat...	Type
Errors (2 items)				
ERROR: Error in the component setting. More details provid	DemoPE		AD1	Processor E...
ERROR: Error in the component setting. More details provid	DemoPE		TI1	Processor E...
Infos (1 item)				
Hint: It is better to use BitIO instead of BitsIO.	Lab3_PE		Bits1	Processor E...

Figure 14. Problems view

3.1.4 Generate code

You can generate Processor Expert code using the option provided in the Project menu or the icon provided in the Components view toolbar.

Select **Project > Generate Processor Expert Code** from the Eclipse IDE menu bar or the icon at the top-right corner of the Components view to generate the code.



Generating code - Lab3_PE\ProcessorExpert.pe

Figure 15. Generate code

The generated code is placed in the **Generated_Code** folder of the **Components** view. You can start using it in your application.

3.1.5 Build and debug

When you are done with the application creation, build the application, and run it on the target.

3.2 Basic principles

This section explains the basic principles of how to create an application using the Processor Expert tool.

The application created in Processor Expert is built from the building blocks, called *embedded components*. The following sections describe the features of the embedded components and SoC components:

- Embedded components on page 29
- SoC components on page 31

3.2.1 Embedded components

This section describes embedded components, such as processor core, on-chip peripherals, FPGAs, standalone peripherals, virtual devices, and pure software algorithms.

Embedded components encapsulate the initialization and functionality of embedded systems basic elements, such as processor core, on-chip peripherals, FPGAs, standalone peripherals, virtual devices, and pure software algorithms. These facilities are displayed through properties, methods, and events. It is very similar to objects in the object-oriented programming (OOP) concept.

This section contains the following subsections:

- #unique_57
- #unique_58
- #unique_59
- #unique_60

3.2.1.1 Easy initialization

You can initialize components by setting their initialization properties in Component Inspector.

Processor Expert generates the initialization code for the peripherals according to the properties of the appropriate components. You can decide whether the component will be initialized automatically at startup or manually by calling the component's `Init` method.

3.2.1.2 Easy on-chip peripherals management

Processor Expert knows exactly the relation between the allocated peripherals and selected components.

When you select a peripheral in the component properties, Processor Expert proposes all the possible candidates but signals the peripherals, which are allocated already, using the icon of the component allocating the peripheral. Processor Expert also signalizes the peripherals that are not compatible with the current component settings with a red exclamation mark. In case of an unrealizable allocation, an error is generated. Unlike common libraries, embedded components are implemented for all possible peripherals with optimal code. The most important advantages of the generated modules for driving peripherals are that you can:

- Select any peripheral that supports component function and change it whenever you want during design time.
- Be sure that the component setting conforms to peripheral parameters.
- Choose the initialization state of the component.
- Choose which methods you want to use in your code and which event you want to handle.
- Use several components of the same type with optimal code for each component.

The concept of the peripheral allocation generally does not enable sharing of peripherals because it would make the application design too complicated. The only way to share resources is through the components and their methods and events. For example, it is possible to use the RTI shared component for sharing periodic interrupt from timers.

3.2.1.3 Component categories

The complete list of the component categories and corresponding components can be found in the Categories tab of the Components Library view.

See Components Library view on page 13 for more information.

3.2.1.4 Levels of abstraction

Processor Expert provides components with several levels of abstraction and configuration comfort.

The Processor Expert components can be divided into the following categories:

- High-level components: Components that are the basic set of components designed carefully to provide functionality to most microcontrollers in market. An application built from these components can be easily ported to another microcontroller supported by Processor Expert. This basic set contains, for example, components for
  - Simple I/O operations, such as BitIO, BitsIO, ByteIO, and so on
  - Timers, such as EventCounter, TimerInt, FreeCntr, TimerOut, PWM, PPG, Capture, WatchDog, and so on
  - Communication, such as AsynchroSerial, SynchroMaster, SynchroSlave, AsynchroMaster, AsynchroSlave, IIC, ADC, and so on
  - Internal memories

  This group of components allows comfortable settings of a desired functionality, such as time in ms or frequency in Hz without letting you know about the details of the hardware registers. The processor-specific features are supported only as processor-specific settings or methods and are not portable.
- Low-level components: Components that are dependent on the peripheral structure to allow you to benefit from the non-standard features of a peripheral. The level of portability is decreased due to a different component interface and the component is usually implemented only for a processor family offering the appropriate peripheral. However, you can easily set device features and use effective set of methods and events.
- Peripheral-initialization components: Components that are on the lowest level of abstraction. An interface of such components is based on the set of peripheral control registers. These components cover all features of the peripherals and are designed for initialization of these peripherals.
- Device configuration components: Components that can produce various data, for example, configuration files, for configuring functional blocks of processors or operating system.

NOTE
This version provides only Device Configuration components.

The features of components at different level of abstraction are described in table below.

Table 5. Features of components at different levels of abstraction

Feature	High level	Low level	Peripheral initialization	Device configuration
High-level settings portable between different processor families	Yes	Partially	No	Partially
Portable method interface for all processor families	Yes	Partially (usually direct access to control registers)	*Init* method only	No
Processor specific peripheral features support	Partially	Mostly yes	Full	Yes

Table continues on the next page...

Table 5. Features of components at different levels of abstraction (continued)

Feature	High level	Low level	Peripheral initialization	Device configuration
Low-level peripheral initialization settings	No	Partially	Yes	Yes
Speed mode independent timing	Yes	Mostly yes	No	No
Events support	Yes	Yes	No (direct interrupt handling)	No
Software emulation of a component function (if the specific hardware is not present)	Yes	No	No	No
Support for RTOS drivers creation	No	No	No	No

3.2.2 SoC components

This section describes the SoC component that encapsulates a processor.

An SoC component is an Embedded component encapsulating one processor type. A Processor Expert project may contain one or more SoC components. The processors included in a project are displayed in the Processors folder of the Components view. It is possible to switch among the SoC components, but only one of the SoC can be active at a time.

3.3 Configuring components

This section explains how to configure embedded components.

Configuring the components in the project is one of the main activities in Processor Expert. It affects the initialization, run-time behavior and range of functionality available to the user of the generated code.

The following sections describe how to configure the embedded components used in the project correctly and effectively:

- Configurations on page 31
- Design time checking on page 32
- Creating user component templates on page 32
- Signal names on page 32

3.3.1 Configurations

You can have several configurations of a project in one project file.

Every configuration keeps the enable/disable state of all components in the project; it does not keep any component settings. If you enable/disable a component in the project, the component state is updated in the currently selected configuration. If you create a new configuration, the current project state is memorized.

The configurations of the current project are listed in the Configurations folder of the **Components** view. The configurations can also hold additional settings that may influence code generation and can be changed in the Configuration Inspector (see Configuration Inspector on page 20).

NOTE

It is possible to have two components with the same name in the project. Each of the components could be enabled in different configuration. This way you can have different setup of a component (a component with the same name) in multiple configurations.

3.3.2 Design time checking

Processor Expert performs instant checking of the project during design time.

During project design time, Processor Expert performs instant checking of the project. As a result of this checking, the error messages may appear in the **Problems** view or directly in the third column of the Component Inspector. Sometimes it may happen that only one small change in the project causes error messages.

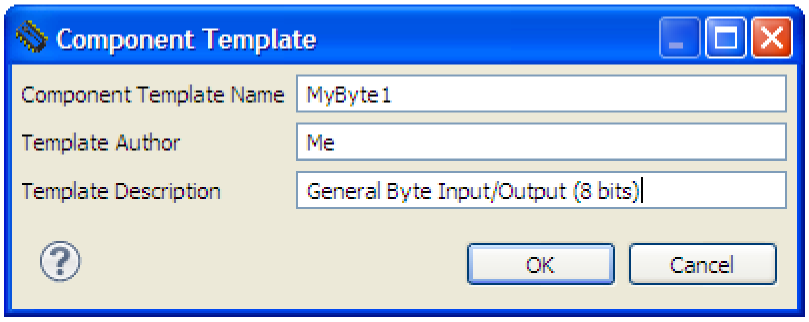
3.3.3 Creating user component templates

If you frequently use a component with some specific settings, you can save the component with its settings as a template.

This template is displayed in the **Components Library** view, behaves as a normal component and could be added to any project. The template has the same properties as the original component. The values of the properties are preset in the template and could be marked as read only.

To create a component template and save it:

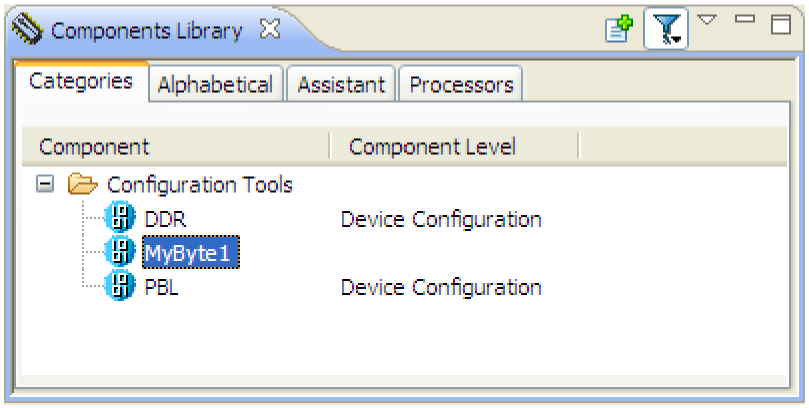
1. Open the pop-up menu of the required component in the **Components** view and select the **Save Component Settings as Template** option. Alternatively, you can use the **Component Inspector** view; open the view menu using the icon at the top right corner, and select **Save Component Settings as Template**.
2. In the Component Template dialog, fill in the appropriate details of the template.



Component Template Name	MyByte1
Template Author	Me
Template Description	General Byte Input/Output (8 bits)

Figure 16. Save template

3. Click **OK**. The template appears in the **Components Library** view and can be inserted into projects. Make sure that you refresh the **Components Library** view by selecting the **Refresh** option from pop-up menu.



Component	Component Level
Configuration Tools	
DDR	Device Configuration
MyByte1	
PBL	Device Configuration

Figure 17. Components Library with template created

3.3.4 Signal names

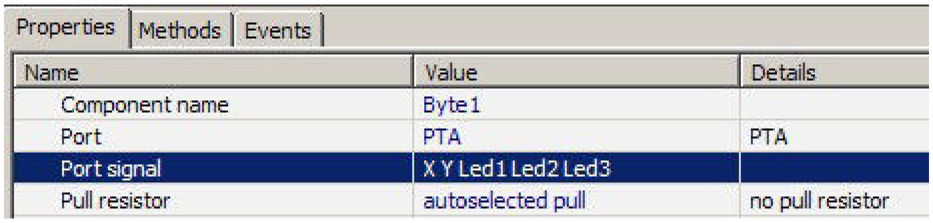
Signals allow you to name the pins used by components.

The main purpose of the signals is to allow you to name the pins used by components with names corresponding to the application.

A signal name can be assigned to an allocated pin by specifying the signal name into the appropriate property (for example, Pin_signal) in the component properties (available in Advanced view mode). A signal name is an identifier that must start with a letter and rest of the name must contain only letters, numbers, and underscore characters.

For the components that allocate a whole port, such as ByteIO, there are two options:

- Assign a same signal name to all pins of port by writing the name into the Port signal property. Processor Expert automatically assigns this name extended with a bit number suffix to each of the individual pins.
- Assign a different signal name to individual pins by writing pin signal names (from the lowest bit to the highest one) separated by commas or spaces into the Port signal property.



Name	Value	Details
Component name	Byte1	
Port	PTA	PTA
Port signal	X Y Led1 Led2 Led3	
Pull resistor	autoselected pull	no pull resistor

Figure 18. Signal name list for a port

This section contains the following subsection:

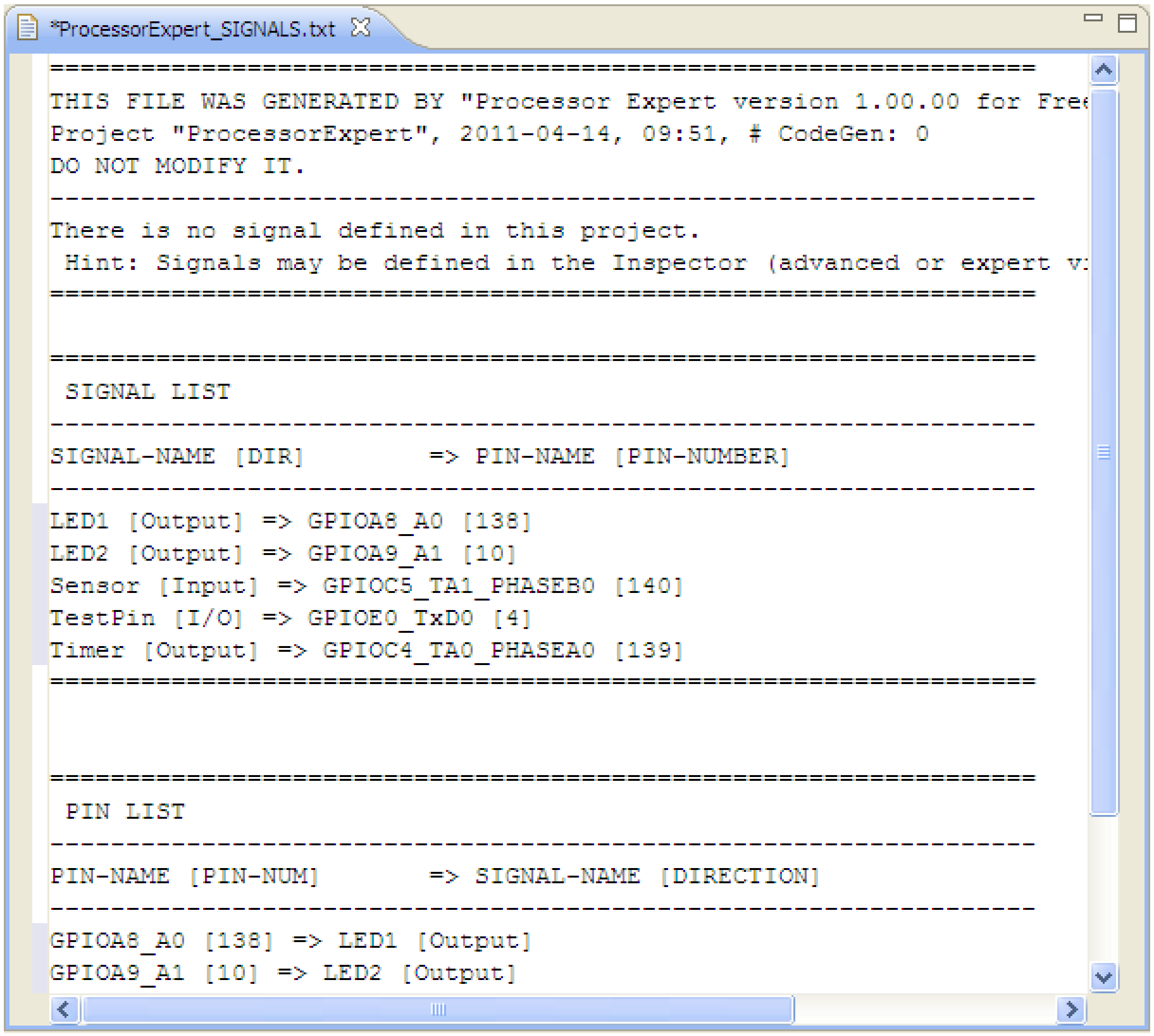
- Documentation on page 33

3.3.4.1 Documentation

Processor Expert automatically generates a document, `ProcessorExpert_SIGNALS.txt`, which contains a list of relationships defined between the signals and corresponding pins.

There is an additional signal direction information added next to each signal name and pin number information next to each pin name.

This document is available in the Documentation folder of the **Components** view. A sample of generated signals documentation is shown below.


*ProcessorExpert_SIGNALS.txt

```
=================================================================
THIS FILE WAS GENERATED BY "Processor Expert version 1.00.00 for Free
Project "ProcessorExpert", 2011-04-14, 09:51, # CodeGen: 0
DO NOT MODIFY IT.
-----------------------------------------------------------------
There is no signal defined in this project.
 Hint: Signals may be defined in the Inspector (advanced or expert v:
=================================================================


=================================================================
 SIGNAL LIST
-----------------------------------------------------------------
SIGNAL-NAME [DIR]        => PIN-NAME [PIN-NUMBER]
-----------------------------------------------------------------
LED1 [Output] => GPIOA8_A0 [138]
LED2 [Output] => GPIOA9_A1 [10]
Sensor [Input] => GPIOC5_TA1_PHASEB0 [140]
TestPin [I/O] => GPIOE0_TxD0 [4]
Timer [Output] => GPIOC4_TA0_PHASEA0 [139]
=================================================================



=================================================================
 PIN LIST
-----------------------------------------------------------------
PIN-NAME [PIN-NUM]       => SIGNAL-NAME [DIRECTION]
-----------------------------------------------------------------
GPIOA8_A0 [138] => LED1 [Output]
GPIOA9_A1 [10] => LED2 [Output]
```

Figure 19. Document containing signals details

3.4 Component inheritance and component sharing

This section explains the terminology used in Processor Expert.

- **Ancestor** is a component that is inherited (used) by another component
- **Descendant** is a new component that inherits (uses) another component(s)
- **Shared Ancestor** is a component that can be used and shared by multiple components

This section contains the following subsections:

- Inheritance on page 35
- Component sharing on page 35

3.4.1 Inheritance

Inheritance in Processor Expert means that an ancestor component is used only by the descendant component.

Inheritance is supported to allow components to access peripherals by hardware-independent interface of the ancestor components. For example, a component that emulates a simple I2C transmitter may inherit two BitIO components for generation of an output signal.

On several complex components, inheritance is used to separate component settings into several logical parts, for example, settings of channel is inherited in the component with settings of the main peripheral module.

Settings in Processor Expert

The descendant component contains a property that allows selecting an ancestor component from a predefined list of templates. The component is created after selection of an appropriate template name (or component name) from the list of the templates fitting the specified interface. Any previously used ancestor component is discarded.

DMATransfer	DMATransfer

Figure 20. Inherited component item in Component Inspector

Processor Expert allows you to select from ancestors that implement the required interface and are registered by the descendant component.

The ancestor component is displayed under its descendant in the project structure node in the **Components** view.

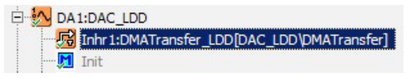


Figure 21. Example ancestor and descendant components in Components view

An ancestor component requires a list of methods and events (interface), which must be implemented by an ancestor component. The error is shown if the ancestor component does not implement any of them. For example, if the settings of the descendant component do not allow it to generate this method.

3.4.2 Component sharing

Component sharing helps you make several components use the capability of one component, similar to inheritance.

This feature allows one component to share its resources and drivers with other components. For example, components may share an I2C component for communication with peripherals connected to the I2C bus or some component may do DMA transfers using DMA component.

Settings in Processor Expert

A shared ancestor component contains a property that allows you to select existing shared ancestor component or create a new one. In this case, the ancestor component is included in the project tree as the other components. The ancestor component may be used with the descendant component only if it is created from a template registered in the descendant component or if the component type is registered in the descendant component. It is recommended that you always create a shared ancestor component through a descendant component.

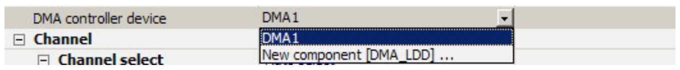


Figure 22. Pop-up menu for selecting/creating a shared ancestor component

Run-time resource allocation

Processor Expert (generated code) does not check the usage of shared resources/code. It is up to you to ensure the correct run-time resources allocation of a shared ancestor component. Often, it is not possible for a shared ancestor component to be used simultaneously by several components.

3.5 Sharing pins among peripherals

This section explains how to share pins among processor peripherals.

Some processors allow few pins to be used or shared by multiple peripherals. This may lead to the need of sharing pin(s) by multiple components. Normally, if you select one pin in more than one component, a conflict is reported. However, it is possible to setup a sharing for such pin in the component inspector.

One of the components sharing a pin has to be chosen as a main component. This component will initialize the pin. In the properties of other components that use the pin, the pin has to be marked as shared (see figure below).

Pin sharing can be set in the **Component Inspector**. The **Component Inspector** must be in **EXPERT** view mode. Use the pop-up menu of the property and select the command **Pin Sharing Enabled**.

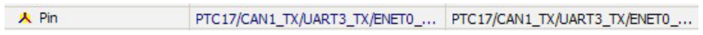

Pin
PTC17/CAN1_TX/UART3_TX/ENET0_...
PTC17/CAN1_TX/UART3_TX/ENET0_...

Figure 23. Pin property with sharing enabled

Pin sharing is advanced usage of the processor peripherals and should be done only by skilled users. Pin sharing allows advanced usage of the pins even on small processor packages and allows application-specific usage of the pins.

This section contains the following subsection:

- ConnectPin method on page 36

3.5.1 ConnectPin method

The ConnectPin method is a component method that allows you to connect a component to the shared pin.

You need to invoke this method to connect a component to the shared pin. You also need to invoke the main component method to connect the pin back to the main component. In fact, the peripherals can usually operate simultaneously, but they have no connection to the shared pins unless the `ConnectPin` method is executed. In case that all components control the shared pin using one peripheral, it is not necessary to use the `ConnectPin` method.

Shared pins are presented in the Processor view on page 21 as well. The component to pin connection line is red.

3.6 Export and import

This section explains how to import or export component settings or configuration of the selected Processor Expert components.

This section explains:

- Export Component Settings
- Export Board Configuration
- Import Component Settings
- Apply Board Configuration
- Import Component(s) from Package

3.6.1 Export component settings

This section explains how to export component settings of a selected component.

It is possible to export one or more component settings, such as:

- Configurations
- Operating system

- Processors
- Components

To export component settings:

1. In the IDE, select **File > Export**. The **Export** wizard appears.

   Expand **Processor Expert** node. Select **Export Component Settings** option, as shown in the figure below.

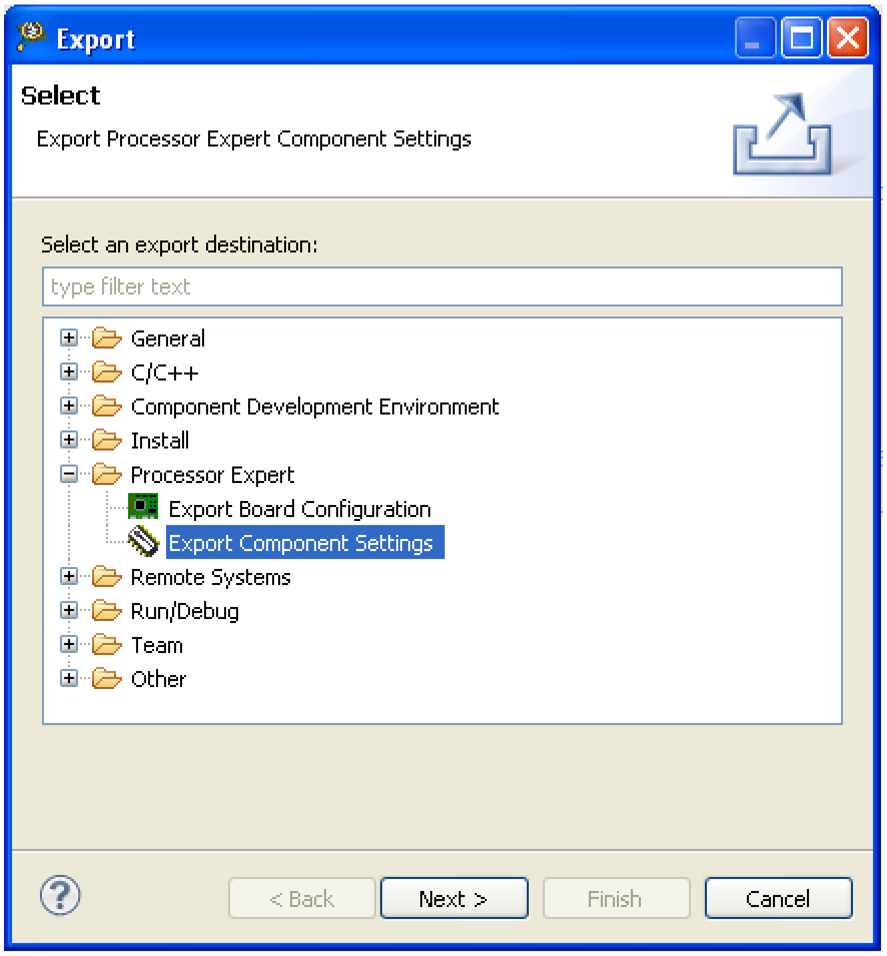

Export
Select
Export Processor Expert Component Settings
Select an export destination:
type filter text
General
C/C++
Component Development Environment
Install
Processor Expert
Export Board Configuration
Export Component Settings
Remote Systems
Run/Debug
Team
Other
< Back
Next >
Finish
Cancel

Figure 24. Export wizard

2. Click **Next**. The **Export Processor Expert Component Settings** page appears.

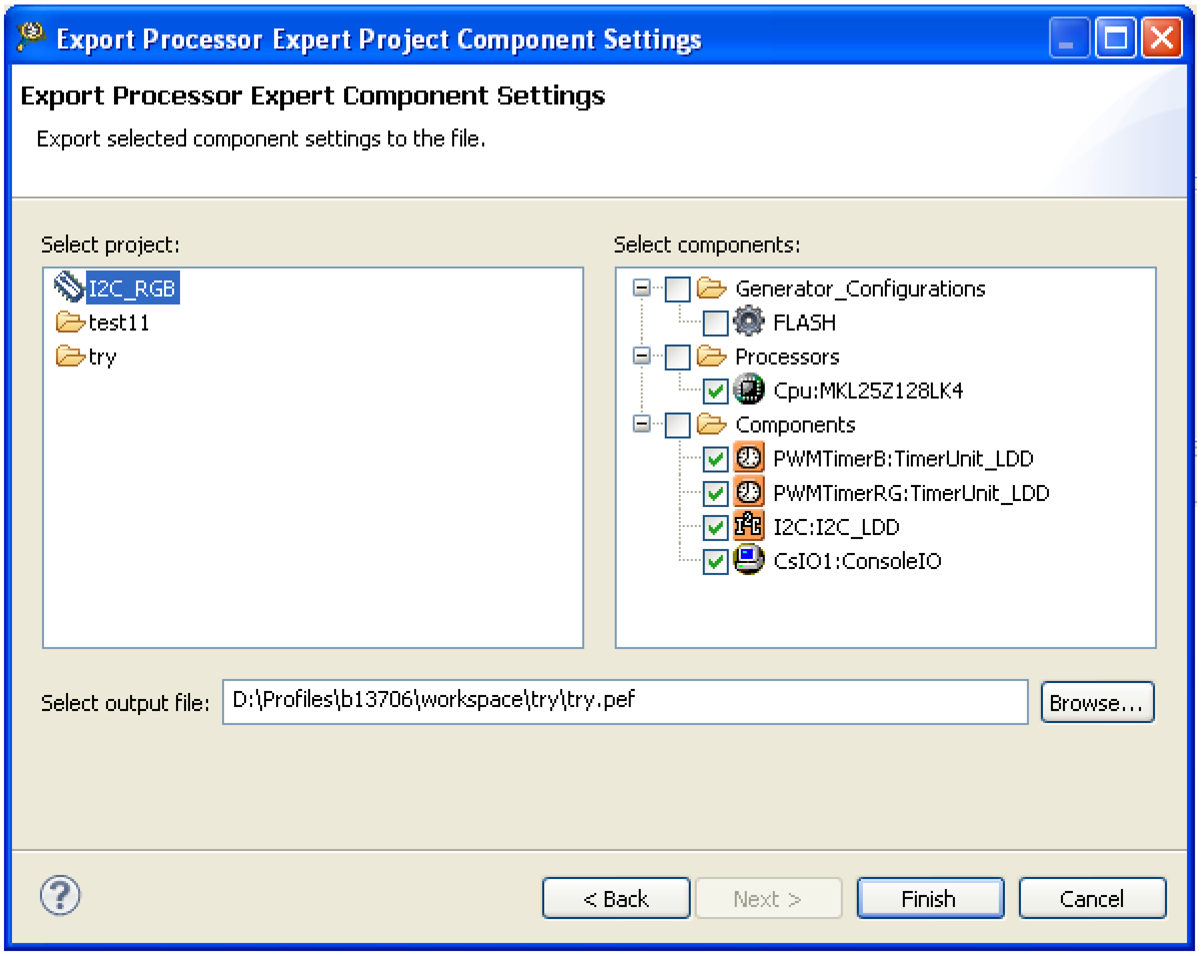


Figure 25. Export Processor Expert Component Settings page

In the left panel, select the project for which you want to export the component settings. In the right panel, select the components to export. Click the **Browse** button to select the output file in which the export settings are saved. The default location for saving the output file is recently selected folder, your home directory. The extension of the file is `.pef`.

3. Click **Finish** to complete the exporting of component settings.

Now you know how to export component settings.

3.6.2 Export board configuration

This section explains how to export board configuration of a selected component.

It is possible to export one processor and one or more components (the components automatically selected are CPU, Pin settings, LDD, and Init components).

You can save the current state of components related to board configuration in to the external file. The extension of the file is `.peb`. The default location for saving the output file is recently selected folder, your home directory.

To export board configuration:

1. In the IDE, select **File** > **Export**. The **Export** wizard appears.

   Expand **Processor Expert** node. Select **Export Board Configuration** option, as shown in the figure below.

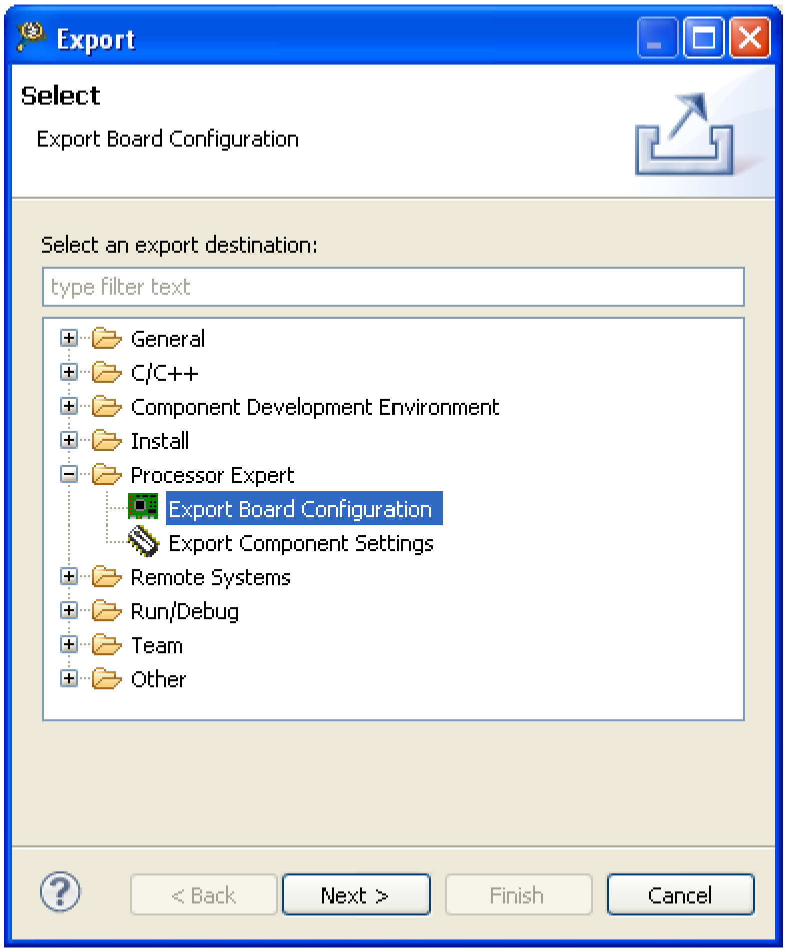
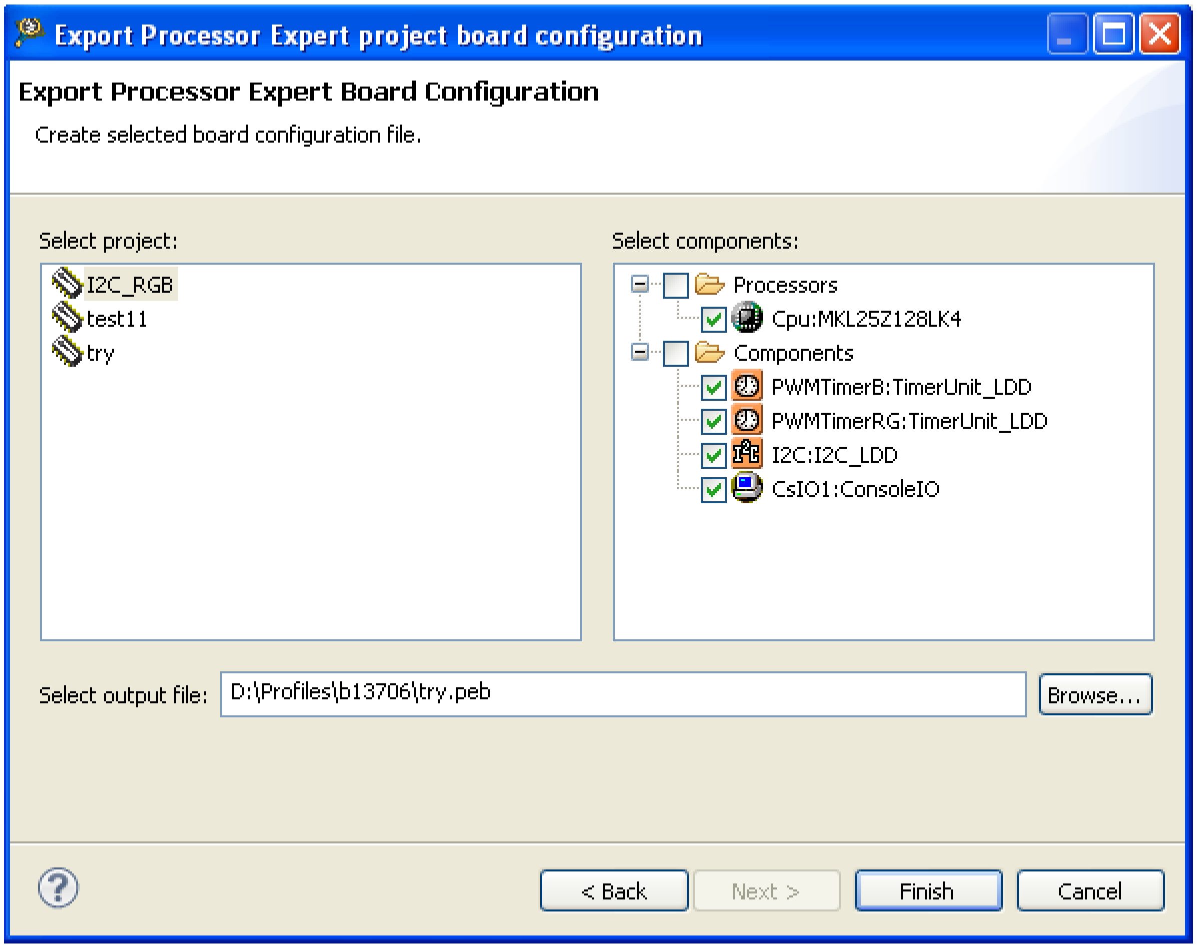

Export
Select
Export Board Configuration
Select an export destination:
type filter text
General
C/C++
Component Development Environment
Install
Processor Expert
Export Board Configuration
Export Component Settings
Remote Systems
Run/Debug
Team
Other
< Back
Next >
Finish
Cancel

Figure 26. Export wizard

2. Click **Next**. The **Export Processor Expert Board Configuration** page appears.


Export Processor Expert project board configuration

Export Processor Expert Board Configuration

Create selected board configuration file.

Select project:
- I2C_RGB
- test11
- try

Select components:
- [ ] Processors
  - [✓] Cpu:MKL25Z128LK4
- [ ] Components
  - [✓] PWMTimerB:TimerUnit_LDD
  - [✓] PWMTimerRG:TimerUnit_LDD
  - [✓] I2C:I2C_LDD
  - [✓] CsIO1:ConsoleIO

Select output file: D:\Profiles\b13706\try.peb

Figure 27. Export Processor Expert Board Configuration page

In the left panel, select the project for which you want to export the board settings. In the right panel, the processor and components are already selected. Click the **Browse** button to select the output file in which the export settings are saved. The default location for saving the output file is either recently selected folder or your home directory. The extension of the file is `.peb`.

3. Click **Finish** to complete the exporting of board configurations.

Now you know how to export board configuration.

3.6.3 Import component settings

This section explains how to import component settings of a selected component.

You can import one or more component settings from a file. Although, components with existing name results in conflict, but it is still possible to import. The imported components are added into the selected project.

To import component settings:

1. In the IDE, select **File > Import**. The **Import** wizard appears.
2. Expand **Processor Expert** node and select **Import Component Settings** option, as shown in the figure below.

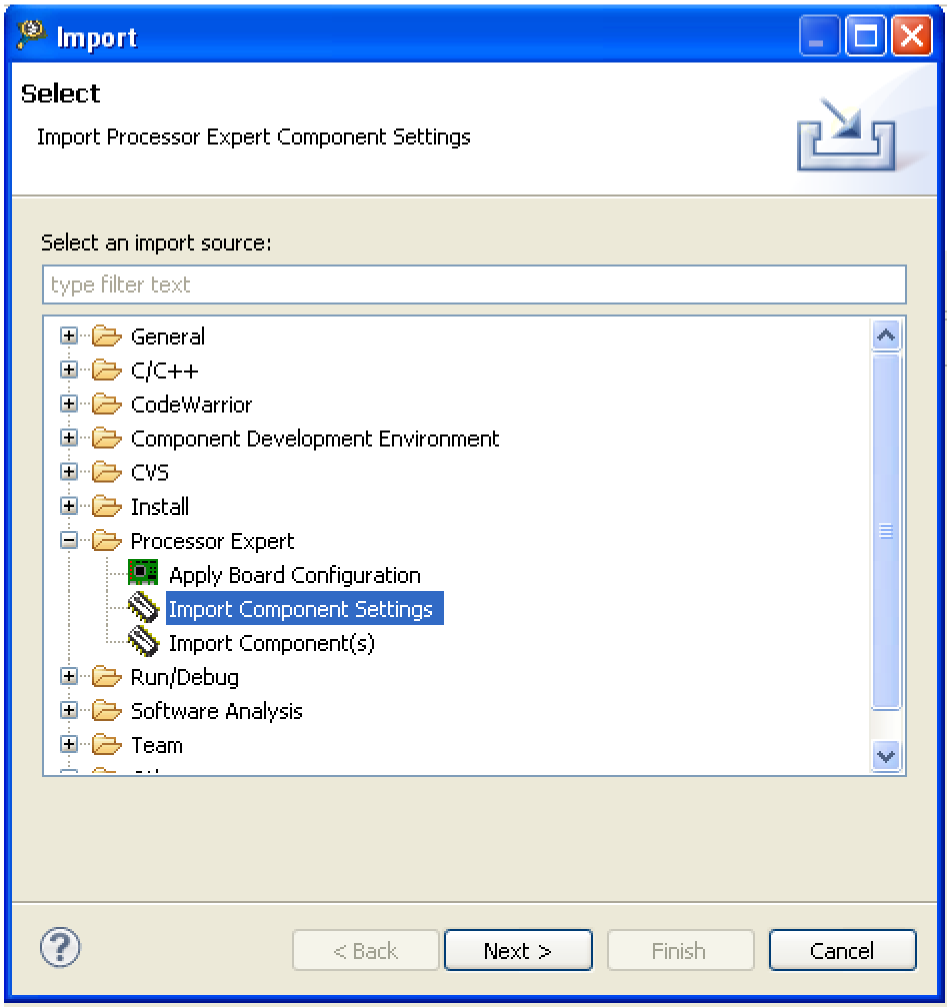
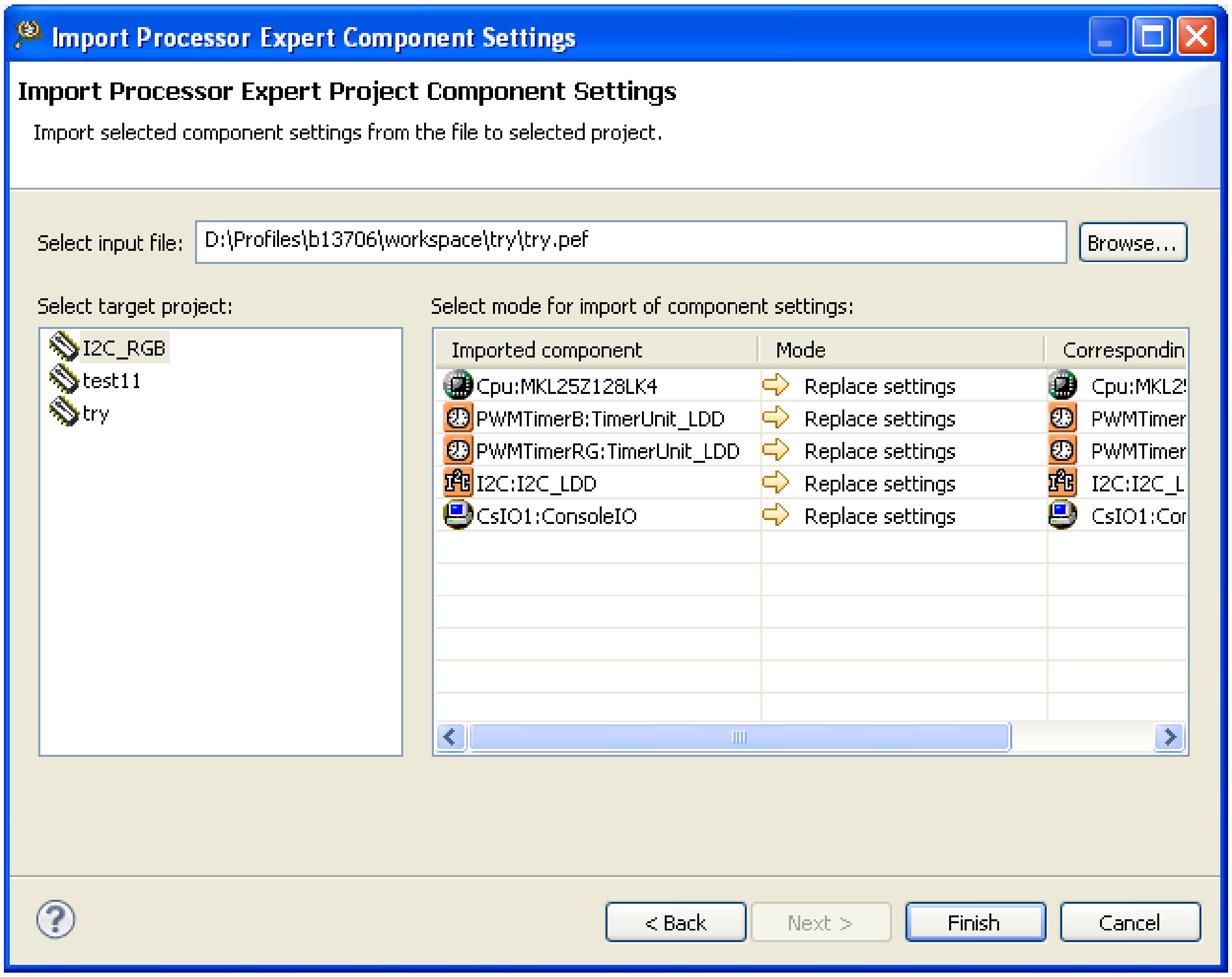


Figure 28. Import wizard

3. Click **Next**. The **Import Processor Expert Project Component Settings** page appears.



Import Processor Expert Project Component Settings

Import selected component settings from the file to selected project.

Select input file: D:\Profiles\b13706\workspace\try\try.pef

Select target project:

- I2C_RGB
- test11
- try

Select mode for import of component settings:

Imported component	Mode	Correspondin
Cpu:MKL25Z128LK4	Replace settings	Cpu:MKL2!
PWMTimerB:TimerUnit_LDD	Replace settings	PWMTimer
PWMTimerRG:TimerUnit_LDD	Replace settings	PWMTimer
I2C:I2C_LDD	Replace settings	I2C:I2C_L
CsIO1:ConsoleIO	Replace settings	CsIO1:Cor

Figure 29. Import Processor Expert Project Component Settings page

4. To import component settings from the file to the selected project, click the **Browse** button. Select the input file with the `pef or .peb or .pe`extension. The default option **Replace settings** is selected if imported component settings are having same name (or ID) and type.
5. Select the mode for importing components settings. The options are:
   - **Ignore**: Do not import this component settings
   - **Add new**: Add new component with imported settings
   - **Add new, keep existing**: If component with same name or type exists, it will add a new component with imported settings and keep the existing one (may cause conflicts)
   - **Add new, disable existing**: If component with same name or type exists, it will add a new component with imported settings and disable the existing one
   - **Replace settings**: Replace existing component with new settings from imported file
6. Click **Finish**. The settings from the `.pef` file is imported to the selected project.

You are ready to import component settings.

3.6.4 Apply board configuration

This section explains how to import board configuration.

You can import board configuration from a file. The imported configurations are added into the selected project.

To import board configuration:

1. In the IDE, select **File > Import**. The **Import** wizard appears.
2. Expand **Processor Expert** node. Select **Apply Board Configuration** option as shown in the figure below.

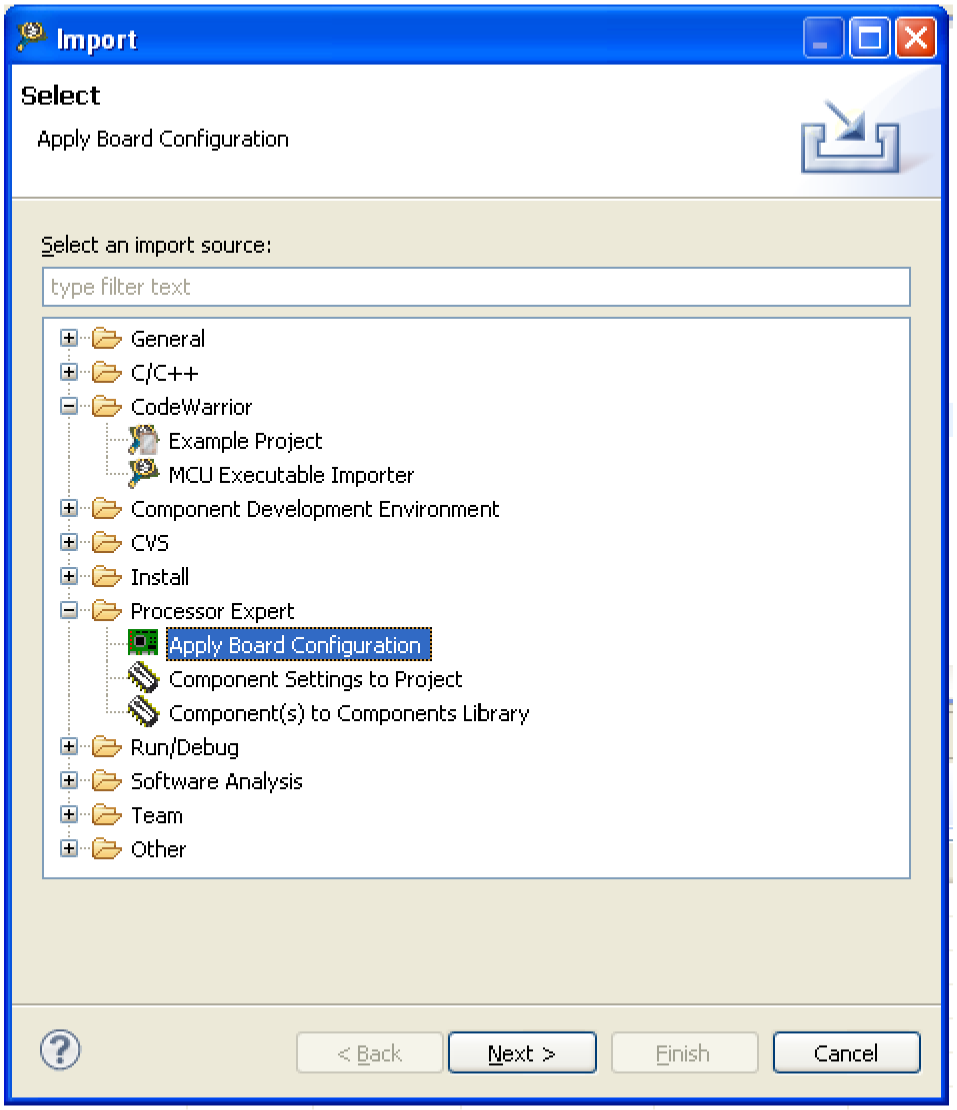
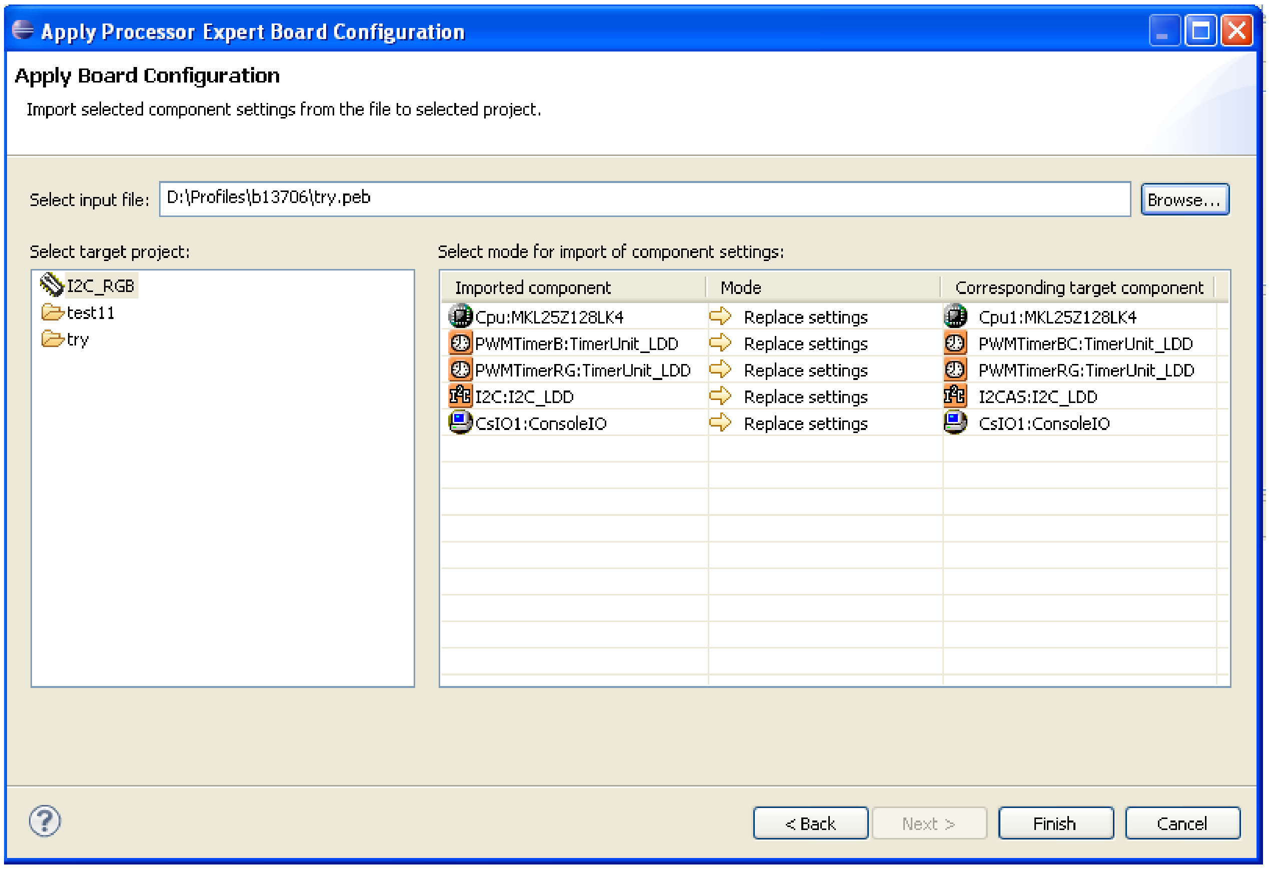


Figure 30. Import wizard

3. Click **Next**. The **Apply Board Configuration** page appears.



Apply Board Configuration

Import selected component settings from the file to selected project.

Select input file: D:\Profiles\b13706\try.peb

Select target project:

- I2C_RGB
- test11
- try

Select mode for import of component settings:

Imported component	Mode	Corresponding target component
Cpu:MKL25Z128LK4	Replace settings	Cpu1:MKL25Z128LK4
PWMTimerB:TimerUnit_LDD	Replace settings	PWMTimerBC:TimerUnit_LDD
PWMTimerRG:TimerUnit_LDD	Replace settings	PWMTimerRG:TimerUnit_LDD
I2C:I2C_LDD	Replace settings	I2CAS:I2C_LDD
CsIO1:ConsoleIO	Replace settings	CsIO1:ConsoleIO

Figure 31. Apply Board Configuration page

Before importing, you can rename some of the components (as shown in figure above), so it will show the mapping of components with different names, but same device allocation.

4. To import component settings from the file to the selected project, click the **Browse** button. Select the input file with the `.peb` extension. The default import mode, **Replace settings**, is selected if imported component settings are having same peripheral device allocation (or ID) and type. For more information on the different types of import modes, see Import component settings on page 40.
5. Click **Finish**. The settings from the `.peb` file is imported to the selected project.

You are ready to import board configuration.

3.6.5 Import component(s) from package

This section explains how to import Processor Expert components from a package.

To import a component:

1. In the IDE, select **File** > **Import**. The **Import** wizard appears.
2. Expand **Processor Expert** node. Select **Import Component(s) from Package** option as shown in the figure below.

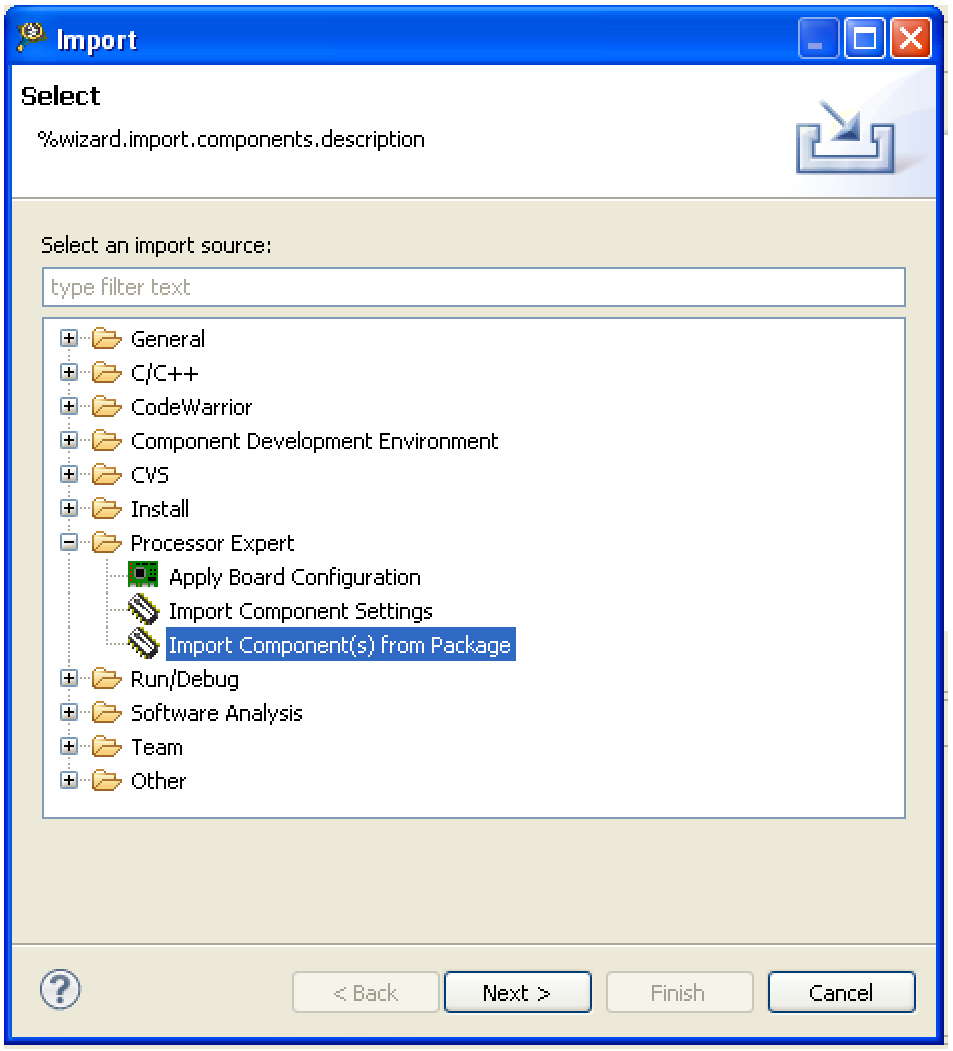


Figure 32. Import wizard

3. Click **Next**. The **Import Processor Expert Components** page appears.

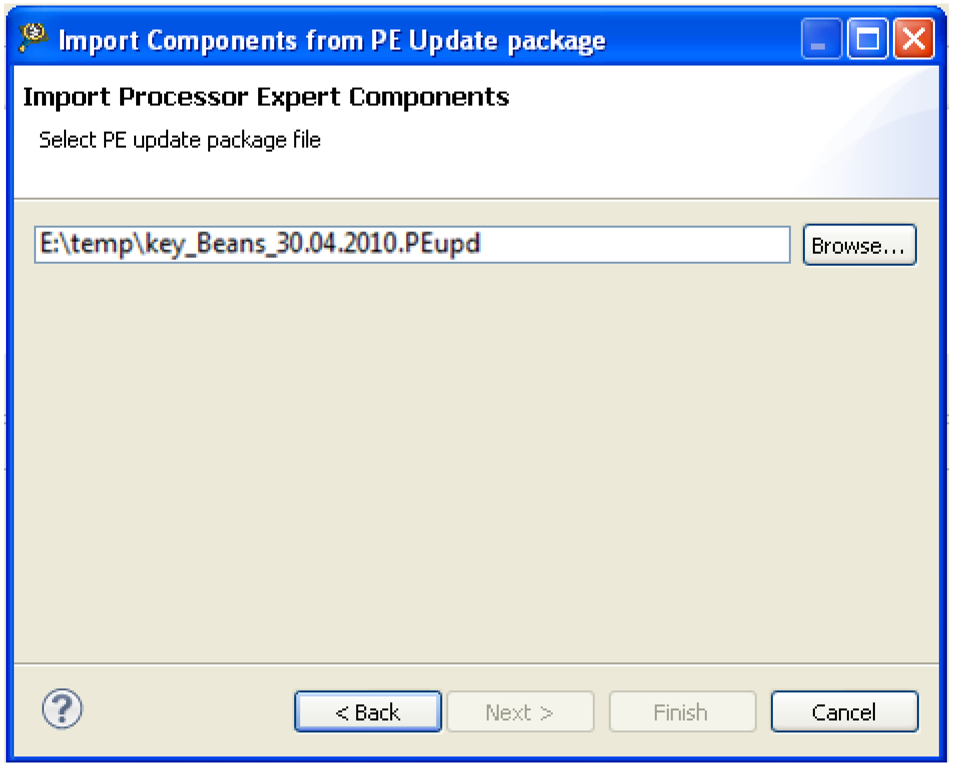

Import Components from PE Update package
Import Processor Expert Components
Select PE update package file
E:\temp\key_Beans_30.04.2010.PEupd
Browse...
< Back
Next >
Finish
Cancel

Figure 33. Import Processor Expert Components page

4. Click **Finish** to select and install Processor Expert update packages (.PEUpd) files.

3.7 Code generation and usage

This section explains how to generate the Processor Expert code.

It explains you about the principles and results of the Processor Expert code generation process and the correct ways and possibilities of using this code in the user application.

In the **Project Explorer** view, right-click the **ProcessorExpert.pe** file and select the **Generate Processor Expert Code Generate Code** option to initiate the code generation process. During this process output files of the components contained in the project are generated. The project must be set-up correctly for successful code generation. If the generation is error free all generated source code files are saved to the destination directory.

This section contains the following subsections:

- Files produced by Processor Expert on page 46
- Tracking changes in generated code on page 47

3.7.1 Files produced by Processor Expert

The existence of the files depends on the project or Processor Expert environment settings and their usage by the components.

Processor Expert may produce some or all of the following files for your project:

- **Component files**: Component output files. See user guides of individual components for details on generated output.
- **Signal names**: This is a simple text file `{projectname}_SIGNALS.txt` or `{projectname}_SIGNALS.doc` with a list of all used signal names. The signal name can be assigned to an allocated pin in the component properties (available in

Advanced view mode). This documentation can be found in the Documentation folder of the **Components** view. See Signal names on page 32 for details.

- **Code generation log** that contains information on changes since last code generation
- **XML documentation** containing the project information and settings of all components in XML format. The generated file `{projectname}_Settings.xml` can be found in the Documentation folder of the **Components** view. It is updated after each successful code generation.

3.7.2 Tracking changes in generated code

Processor Expert allows you to track changes in the generated modules.

It is necessary to enable the **Create code generation log** option on the **Processor Expert Project Options** page of the **Properties** window (Figure 1. Properties window on page 7). If this option is enabled, a file, `ProcessorExpert_CodeGeneration.txt`, is generated in the **Documentation** folder.

The file contains a list of changes with details on purpose of each change (see the example below).

```
###############################################################
Code generation 2010/10/22, 15:57; CodeGen: 1 by user:

by Processor Expert 5.00 for Freescale Microcontrollers; PE core 04.46

Configuration: Debug_S08GW64CLH

Target CPU: MC9S08GW64_64; CPUDB ver 3.00.000

# The following code generation options were changed:

> option Create code generation log: value changed from false to true

###############################################################

Code generation 2010/10/22, 16:01; CodeGen: 2 by user: hradsky

by Processor Expert 5.00 Beta for Freescale Microcontrollers; PE core
04.46

Configuration: Debug_S08GW64CLH

Target CPU: MC9S08GW64_64; CPUDB ver 3.00.000

# Component Cpu:MC9S08GW64_64, the following files modified due to
internal interdependency:

- Generated_Code\Vectors.c - changed

- Generated_Code\Cpu.h - changed

- Generated_Code\Cpu.c - changed

# New component PWM1:PWM (ver: 02.231, driver ver. 01.28) added to the
project, the following - 78 -

Processor Expert User Manual Application Design

- Generated_Code\PWM1.h - added

- Generated_Code\PWM1.c - added
```

```
# Documentation

- Documentation\ProcessorExpert.txt - regenerated

- Documentation\ProcessorExpert_Settings.xml - regenerated

# Other files have been modified due to internal interdependency:

- Generated_Code\PE_Timer.h - added

- Generated_Code\PE_Timer.c - added

# User modules

- Sources\ProcessorExpert.c - changed

> updated list of included header files

- Sources\Events.h - changed

> updated list of included header files

Totally 11 file(s) changed during code generation.
```

Figure 34. Example - Tracking changes in generated code

To view changes within the individual files, you can use a file pop-up menu command **Compare with > Local history** available in Components view. It allows to compare files with the version before the code generation.

3.8 PE project file

This section describes the Processor Expert PE file that is created during project creation.

All components in the project with their state and settings and all configurations are stored in one file `ProcessorExpert.pe` in the root of project directory. If the whole content of the project including subdirectories is copied or moved to another directory, it is still possible to open and use it in the new location.

Project directory structure

Processor Expert uses the following subdirectory structure within the project directory:

- `\Generated_Code`: The directory containing all generated source code modules for components.
- `\Documentation`: The directory with the project documentation files generated by Processor Expert.
- `\Sources`: The directory for main module, event module other user modules.

User-created templates are shared by all users and are stored in the following folder:

- On Windows: `%ALLUSERSPROFILE%\ApplicationData\Processor Expert\{version}\`

  For example, `C:\Documents and Settings\All Users\ApplicationData\ProcessorExpert\CW08_PE3_02\`
- On Linux: `{workspace}/.metadata/.plugins/com.freescale.processorexpert.core/ProcessorExpert/{version}`

The path is relative to the current Eclipse workspace directory.

Index

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 Freescale 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. Freescale reserves the right to make changes without further notice to any products herein.

Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale 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 Freescale 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. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Freescale, the Freescale logo, CodeWarrior, QorIQ, and Processor Expert are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners.

© 2016 Freescale Semiconductor, Inc.

QCVS_PEX_User_Guide
Rev. 4.x
06/2016