CodeWarrior Plug-in for Freescale HCS12/HCS12X

Processor Expert User Manual

Help version 4.12

CONTENTS

1. Introduction

Both hardware and software design have progressed so much with the ever-advancing new technologies emerging everyday, but their interrelationships and interdependence have been mostly neglected. On one hand, we 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 Code Warrior plug-in is designed for **rapid application development** of embedded applications for a wide range of microcontrollers and microprocessor systems.

Processor Expert Main features

- The application is created from **components** called **Embedded Components.**
- Embedded Components encapsulate functionality of basic elements of embedded systems like CPU core, CPU on-chip peripherals, FPGA, standalone peripherals, virtual devices, and pure software algorithms, and change these facilities to properties, methods, and events (like objects in OOP).
- Processor Expert suggests, connects, and generates the drivers for embedded system hardware, peripherals, or used algorithms. This allows the user to concentrate on the creative part of the whole design process.
- Processor Expert allows true **top-down** style of application design - the user starts the design directly by defining the application behavior instead of spending days just trying to make the chip work.
- Processor Expert works with an **extensible components library** of supported microprocessors, peripherals, and virtual devices.
- Processor Expert Peripheral Initialization components generate effective initialization code for all on-chip devices and support all their features.
- Processor Expert allows to easily examine the details of the architecture and the relationship between the Embedded Component setup, and CPU control registers initialization.
- The user can create his/her own components using the **Component Wizard** external tool. See *5 Component Wizard Description* for details.

1.1. Processor Expert Plug-in Overview

Processor Expert provides an efficient development environment for rapid application development of the embedded applications. You can develop embedded applications for a wide range of microcontrollers and microprocessor systems using Processor Expert.

Processor Expert is integrated as a plug-in into the CodeWarrior IDE. You can access Processor Expert from the CodeWarrior IDE using the Processor Expert menu in the CodeWarrior IDE menu bar. The Processor Expert plug-in generates code from the Embedded Components and the CodeWarrior IDE manages the project files, and compilation and debug processes.

Figure below shows the Processor Expert plug-in that appears when you select the Processor Expert menu in the CodeWarrior IDE menu bar.

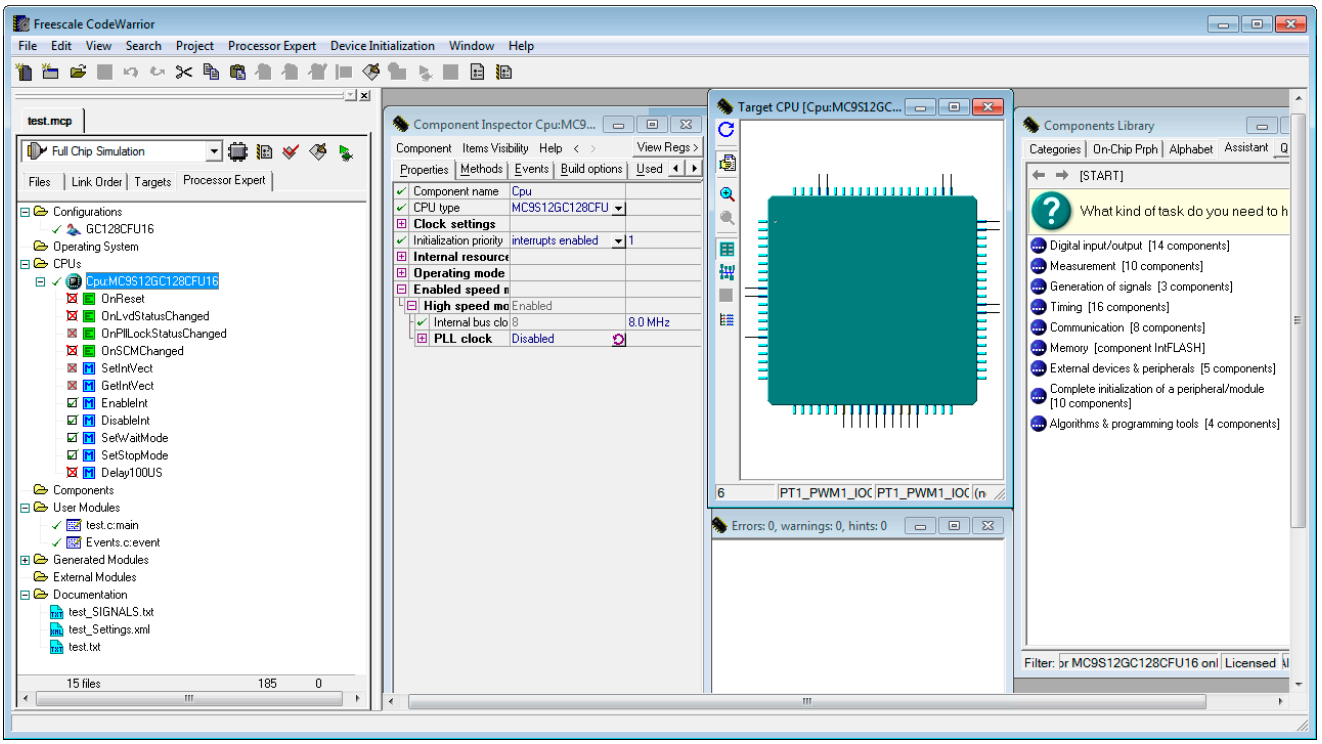


Figure 1.1 - CodeWarrior IDE with Processor Expert active

How to Create a New Project

See the chapter *4 Processor Expert Tutorials* or *3.1 Quick Start in Processor Expert* for step-by-step instructions on how to create a new Processor Expert project.

Compiler and Linker settings

To set the compiler and linker options, select the command **{TargetName} Settings** in the "Edit" menu in the Code Warrior main menu. You can find linker and compiler specific settings in the "Target" and "Linker" folders. The command **{TargetName} Settings** is not available when no project is open.

Where to find source code and user modules

Processor Expert generates all drivers during the code generation process. The generated files are automatically inserted into the active (default) target in the CodeWarrior project. Generated files corresponding to Embedded Components can be accessed in the "Generated Code" folder in the "Files" tab in the Code Warrior project window.

Other files, intended to be modified by users, are generated into the "User modules" folder in the "Files" tab in the Code Warrior Project window. A user can also add his/her own specific source code files into this folder. If the linker setting of the default target does not match the CPU in the Processor Expert project, the user is asked whether to automatically set the correct linker settings in the default target or to create a new target with correct linker settings. In the latter case the files will be generated in the new target (more information about the CodeWarrior Project panel can be found in the CodeWarrior documentation). For more information on generated files please see the chapter *3.5.1 Code Generation*.

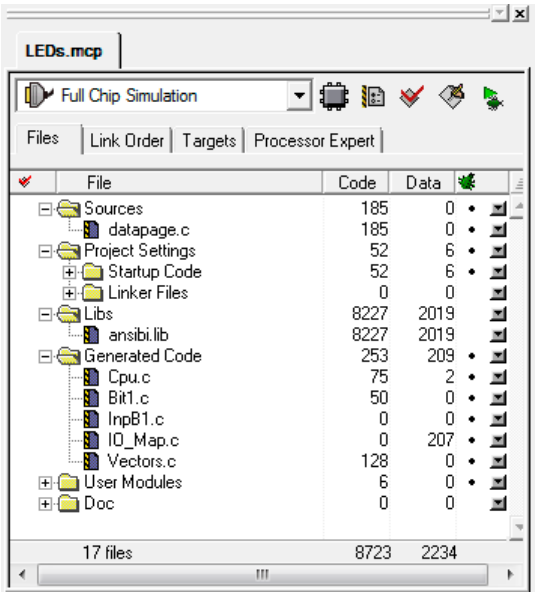


Figure 1.2 - CodeWarrior project panel

1.2. Features

Processor Expert has built-in knowledge (internal definitions) about all microcontroller units and integrated peripherals. The microcontroller units and peripherals are encapsulated into configurable components called Embedded Components, each of which provides a set of useful properties, methods and events.

An intuitive and powerful User Interface (UI) allows the user to define the system behavior in several steps. A simple system can be created just by selecting the necessary components, setting their properties to the required values and maybe also dragging and dropping some of their methods to the user part of the project source code.

Processor Expert Key Components

- Graphical IDE
- Built-in detailed design specifications of the Freescale devices
- Code generator

PE Key Features

- Design-time verifications
- CPU selection from multiple CPU derivatives available
- CPU pin detailed description and structure viewing
- Configuration of functions and settings for the selected CPU and its peripherals
- Definition of system behavior during initialization and at runtime
- Design of application from pre-built functional components
- Design of application using component methods (user callable functions) and events (templates for user written code to process events, e.g. interrupts)
- Customization of components and definition of new components
- Tested drivers

- Library of components for typical functions (including virtual SW components)
- Verified reusable components allowing inheritance
- Verification of resource and timing contentions
- Concept of project panel with ability to switch/port between CPU family derivatives
- Code generation for components included in the project
- Implementation of user written code
- Interface with Freescale CodeWarrior

PE based tool solution offers the following advantages to Freescale CPU customers:

- In all phases of development, customers will experience substantial reductions in
  - development cost
  - development time
- Additional benefits in product development process are
  - Integrated development environment increases productivity
  - Minimized time to learn Freescale CPU
  - Rapid prototyping of entire applications
  - Modular and reusable functions
  - Easy to modify and port implementations

Integrated development environment increases users' productivity

- "This tool lets me produce system prototypes faster because the basic setup of the controller is easier. This could mean that I will implement more of my ideas into a prototype application having a positive effect on the specification, analysis and design phase. PE justifies its existence even when used for this purpose alone!"
- "This system frees you up from the hardware considerations and allows you to concentrate on software issues and resolve them thoroughly."
- "Very good for CPUs with embedded peripherals. It significantly reduces project development time."

Primary Reasons Why Users Feel that Way:

- Processor Expert has built-in knowledge (internal definition) of the entire microcontroller with all its integrated peripherals.
- Processor Expert encapsulates functional capabilities of microcontroller elements into concepts of configurable components.
- Processor Expert provides an intuitive graphical UI, displays the microcontroller structure, and allows the user to take advantage of predefined and already verified components supporting all typically used functions of the microcontroller.
- Applications are designed by defining the desired behavior using the component settings, drag & drop selections, utilizing the generated methods and events subroutines, and combining the generated code with user code.
- Processor Expert verifies the design based on actual microcontroller resource and timing contentions.
- Processor Expert allows the efficient use of the microcontroller and its peripherals and building of portable solutions on a highly productive development platform.

Minimized Time to Learn Microcontroller

There are exciting possibilities in starting a new project if the user is starting from ground zero even if the user is using a new and unfamiliar processor.

- The user is able to utilize the microcontroller immediately without studying the microcontroller's documentation.
- The user is able to implement simple applications even without deep knowledge of programming.
- PE presents all necessary information to the user using built-in descriptions and hints.
- PE has built-in tutorials and example projects.

Rapid Prototyping of Entire Applications

"Processor Expert allows the users to try different approaches in real time and select the best approach for the final solution. Users are not confined to a pre-determined linear approach to a solution."

- Easy Build of application - based on system functional decomposition (top-down approach)
- Easy CPU selection
- Easy CPU initialization
- Easy initialization of each internal peripheral
- Simple development of reusable drivers
- Simple implementation of interrupt handlers
- Inherited Modularity and reuse
- Inherited ease of implementation of system hardware and software/firmware modifications

Modular and Reusable Functions

Processor Expert greatly decreases the start-up time and minimizes the problems of device idiosyncrasies.

- It uses the concept of a function encapsulating entity called Embedded Component with supporting methods and events
- Uses a library of predefined components
- Uses the concept of device drivers and interrupt handlers that are easy to reapply
- Uses the concept of well-documented programming modules to keep the code well organized and easy to understand

Note: Processor Expert Embedded Component were formerly called "Processor Expert Embedded Beans."

Easy to modify and port implementations

Processor Expert allows optimal porting to a previously unused processor.

- Supports multiple devices within a project and makes it extremely easy to switch them
- Supports desired changes in the behavior of the application with an instant rebuild
- Supports interfacing of the CodeWarrior IDE

1.3. Concepts

The main task of **Processor Expert** is to manage CPU and other hardware resources and to allow virtual prototyping and design.

Code generation from components, the ability to maintain user and generated code, and an event based structure significantly reduce the programming effort in comparison with classic tools.

Embedded Components

Component is the essential encapsulation of functionality. For instance, the *TimerInt* component encapsulates all CPU resources that provide timing and hardware interrupts on the CPU.

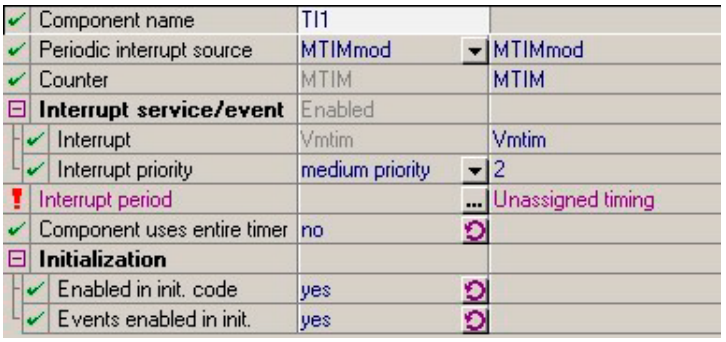


Figure 1.3 - Example of TimerInt component (periodical event timer) properties

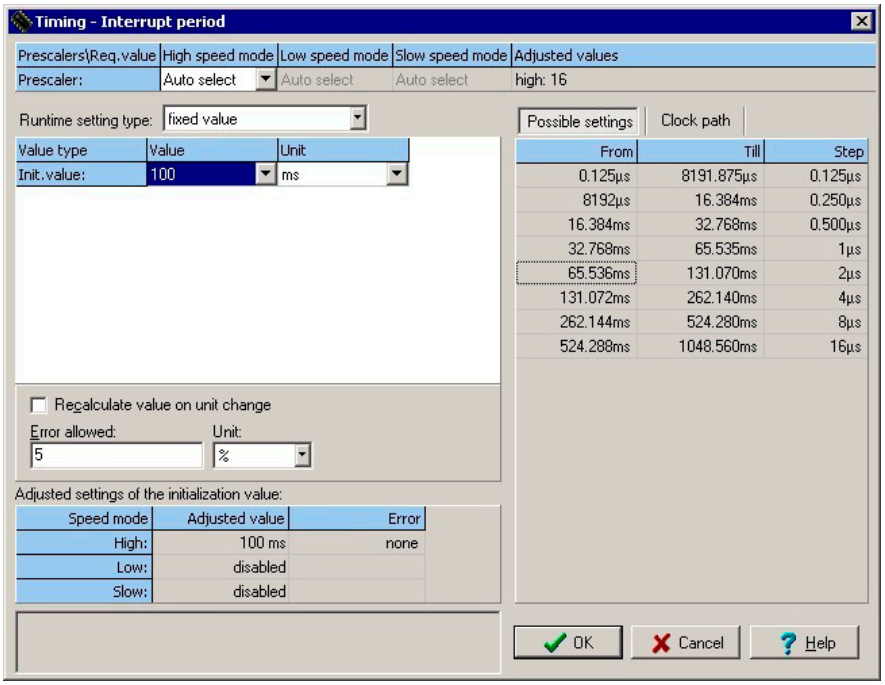


Figure 1.4 - Timing dialog box allows a user friendly setting of components' timing

You'll find many components that we call Embedded Components in the Processor Expert Components library window. These components are designed to cover the most commonly required functionality used for the microcontroller applications - from handling port bit operations, external interrupts, and timer modes up to serial asynchronous/synchronous communications, A/D converter, I2C, CAN etc.

A component provides a **clear interface**. By setting **properties**, a user defines the future behavior of the component in runtime. The user controls properties in design time by using the Component Inspector. Runtime control of the component function is done by the **Methods**. **Events** are interfacing hardware or software events invoked by the component to the user's code.
The user can enable or disable the appearance (and availability) of methods of the component in generated source code. Disabling unused methods could make the generated code shorter. See *3.6.1 General Optimizations* for details.

Events, if used, can be raised by interrupt from the hardware resource such as timer, SIO or by pure software reason, such as overflow in application runtime. You can enable or disable interrupts using component **methods** and define priority for event occurrence and for executing its Interrupt Service Routine (ISR). The hardware ISR provided by the component handles the reason for the interrupt. If the interrupt vector is shared by two (or more) resources, then this ISR provides the resource identification and the user is notified by calling the **user event handling code.**

Creating Applications

Creation of an application with Processor Expert on any microcontroller is very fast. To create an application, first choose and set up a CPU component, add other components, modify their properties, define events and select Generate Code. Processor Expert generates all code (well commented) from components according to your settings. See *3.5.1 Code Generation* for details.

This of course is only part of the application code that was created by the "virtual application engineer" - Processor Expert CPU knowledge system and solution bank. The solution bank is created from hand written and tested code optimized for efficiency. These solutions are selected and configured in the code generation process.

Enter your code for desired events, provide main code, add existing source code - and build the application using classic tools - compiler, assembler - and debug it before the final burn-in. These are typical steps when working with Processor Expert.

Other components may help you to very quickly include pictures, files, sounds, and string lists in your application .

The other components can be obtained from www.processorexpert.com or created from existing sources. Other components can incorporate already existing components. They can inherit their properties, methods, and events.

Assume that you want to share a component with other developers. For example a component that can drive an LED segment display. Because it is used often for different hardware configurations - on different CPU pins - then it must be portable and independent of CPU resources.
A lot of tasks and algorithms can be incorporated into a component. Such components are called software (SW) components. SW components can be pure SW components (FFT) or can inherit even multiple components that encapsulate hardware (HW) resources. The advantage is independence on a physical layer, portability and sharing of once written and tested code.

For this example we simply select as parents BitIO, BitsIO or ByteIO and TimerInt components from the component library. The new LED display component will provide the properties of a component reference type for this component. In design time this allows the new component access to its parents' properties and defines the physical connection pins or timer resources. Additionally, the new component will have its own properties and methods. Methods and events can be constructed using the parent component's methods.

Don't be concerned about the complexity of this process - simply select from the Processor Expert Tools menu the Components Wizard tool which makes all the arrangements for you. You only need to enter the code of methods and events, save new component and install it on the Components Palette or share it with others.

Processor Expert has built-in knowledge (internal definitions) about the entire CPU with all integrated peripherals. The CPU units and peripherals are encapsulated into configurable components called Embedded Components and the configuration is fast and easy using a graphical Component Inspector.

Peripheral Initialization Components are a subset of Embedded Components allow the user to **setup initialization of the particular on-chip device** to any possible mode of operation. The user can easily view all initialization values of the CPU produced by Processor Expert with highlighted differences between the last and current properties settings.

Processor Experts performs a **design time checking of the settings** of all components and reports errors and warnings notifying users 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 CPU** showing the details about the allocated on-chip peripherals and pins.

Processor Expert generates a **ready-to-use source code** initializing all on-chip peripherals used by the component according to the component setup.

1.4. Benefits of Embedded Components and Processor Expert Technology

The key benefit of **Embedded Components** is same as using components in software design environments such as Microsoft Visual Basic or Borland Delphi. In comparison with components used within these products, Embedded Components provide hardware encapsulation in the form of a platform-independent standard. Different players in the embedded market should benefit from such a standardized approach.

Microprocessor producers

Each year microprocessor producers introduce many new microprocessor families or derivatives. As the complexity of microprocessors increases, programmers must handle more and more registers to get the required functionality. Classical development tools usually do not support the rapid prototyping phase of design, and classical programming languages are unable to describe the on-chip peripherals structure efficiently. On the other hand, microprocessor producers need to speed up the learning, design and coding processes for their customers.
For the designers, Processor Expert and its configuration and code generation features eliminate the necessity to be otherwise preoccupied with the hardware dependencies. Processor Expert could also even suggest the right member of a microprocessor family for the specific application.

Producers of intelligent peripheral I/Os and other devices

Complex and feature-rich peripherals and controllers require immense efforts to use them efficiently, even if device drivers are supplied by the factory. But imagine the possibility of supporting customers with components providing a standard software interface that allows building applications and using new hardware device features easily.
The Processor Expert environment allows this — customers can easily download new components from the internet and install them into Processor Expert.

Producers of hardware of microprocessor systems

Microprocessor boards that are to be programmed by a customer must be well supported by software. Processor Expert can handle software configuration and generation of drivers for microprocessor devices and off-chip peripheral devices. Creating an application using Processor Expert takes usually 70% less time than with standard Integrated Development Environments (IDEs) containing only a source code editor/compiler/debugger.

Producers of OS

Processor Expert can be used to build an OS kernel or OS drivers. Also, due to its open component architecture and support of pure software components, Processor Expert can be used to build applications benefiting from underlying operating system services.

Educational institutes

Microprocessor-oriented courses can benefit from the information available in Processor Expert about microprocessor structures and hardware independence delivered by Processor Expert. Design of applications begins with definition of functionality, which can be obtained very quickly by building the application from Embedded Components. Students can get the results very fast without facing problems that are not related to the subject of the course, for example compiler bugs and errors in the documentation.

Hardware and software developers

Shortening of the design and learning phase, speeding up of the deployment of new components, full use of hardware using tested software components, reduction in time and cost of design — all these are keys to success provided by Processor Expert.

1.5. Terms and Definitions Used in Processor Expert

Component - An Embedded Component is a component that can be used in Processor Expert. Embedded Components encapsulate the functionality of basic elements of embedded systems like CPU core, CPU on-chip peripherals, standalone peripherals, virtual devices and pure software algorithms and wrap these facilities to properties, methods, and events (like objects in OOP). Components can support several languages (ASM, ANSI C, Modula and others) and the code is generated for the selected language.

Component Inspector - Window with all parameters of a selected component: properties, methods, events.

Bus clock - A main internal clock of the CPU. Most of the CPU timing is derived from this value.

CPU Component - Component that encapsulates the CPU core initialization and control. This component also holds a group of settings related to the compilation and linking, such as Stack size, Memory mapping, linker settings. Only one CPU component can be set active as the target CPU. See *3.2.2 CPU Components* for details.

Component Driver - Component drivers are the core of Processor Expert code generation process. Processor Expert uses drivers to generate the source code modules for driving an internal or external peripheral according to the component settings. A Component can use one or more drivers.

Counter - Represents the whole timer with its internal counter.

Events - Used for processing events related to the component's function (errors, interrupts, buffer overflow etc.) by user-written code. See *3.2.1 Embedded Components* for details.

External user module - External source code attached to the PE project. The external user module may consist

of two files: implementation and interface (*.C and *.H).

Free running device - Virtual device that represents a source of the overflow interrupt of the timer in the free running mode.

High level component - Component with the highest level of abstraction and usage comfort. An application built from these components can be easily ported to another microcontroller supported by the Processor Expert. They provide methods and events for runtime control. See *3.2.1.1 Component Categories* for details.

Internal peripherals - internal devices of the CPU such as ports, timers, A/D converters, etc. usually controlled by the CPU core using special registers.

ISR - Interrupt Service Routine — code which is called when an interrupt occurs.

Low level component - a component dependent on the peripheral structure to allow the user to benefit from the non-standard features of a peripheral. The level of portability is decreased because of this peripheral dependency. See *3.2.1.1 Component Categories* for details.

MCU - Microcontroller Unit — microcontroller used in our application.

Methods - user callable functions or sub-routines. The user can select which of them will be generated and which not. Selected methods will be generated during the code generation process into the component modules.

Module - Source code module — could be generated by Processor Expert (Component modules, CPU Module, events.c) or created by the user and included in the project (user module).

OOP - Object-oriented programming (OOP) was invented to solve certain problems of modularity and reusability that occur when traditional programming languages such as C are used to write applications.

PE - Abbreviation of Processor Expert that is often used within this documentation.

Peripheral Initialization component - encapsulates the whole initialization of the appropriate peripheral. Components that have the lowest levels of abstraction and usage comfort. See *3.2.1.1 Component Categories* for details. They usually do not support any methods or events except the initialization method. The rest of the device driver code needs to be written by hand using either PESL or direct control of the peripheral registers. See *3.8 Low-level Access to Peripherals* for details.

Popup menu - this menu is displayed when the right mouse button is pressed on some graphical object.

PLL - Phase Locked Loop. This circuit is often built-in inside the CPU and can be used a main source of the clock within the CPU.

Prescaler - A fixed or configurable device that allows to divide or multiply a clock signal for a peripheral CPU peripheral or its part.

Properties - Parameters of the component. Property settings define which internal peripherals will be used by the component and also initialization and behavior of the component at runtime.

RTOS - Real Time Operating System is an operating system (OS) intended for real-time applications.

Target CPU - The CPU derivative used in a given project.

Template - Component Template — component with preset parameters.

User-defined Component Template - User-defined component template is a component with preset parameters saved under a selected name. Also the name of the author and short description can be added to the template.

User module - Source code module created or modified by the user. (Main module, event module or external

user module).

Xtal - A crystal - a passive component used as a part of an oscillator circuit.

2. User Interface

Menu

Processor Expert menu is integrated in the CodeWarrior IDE. It contains a new item named **"Processor Expert"** .

See Processor Expert plug-in Main menu page for description of individual items.

The user interface of Processor Expert consists of the following windows (integrated in CodeWarrior IDE):

Project Editing Windows

- Project panel with components (including CPU(s)), external modules and documentation included in project. Project Panel supports several configurations of one project.
- Inspector - a window which allows the user to setup Components and Configurations of the project.
- Components Library - shows all supported components in the appropriate version of the Processor Expert including CPU components and component templates.
- Target CPU - a window graphically showing CPU package, structure and components connected to internal peripherals. Allows to easily add components related to a specific peripheral to the project using a pop-up menu of the peripheral.

Project Information Windows

- Error window - a window with errors, warning messages and hints from project checking, generation and from external tools
- CPU Timing Model - a window showing the target CPU's timing.
- Peripheral Initialization - shows overview of peripheral initialization settings for the current CPU.
- Peripherals Usage Inspector - a window showing which component allocates which on-chip peripheral.
- Resource Meter - a window displaying the amount of the target CPU's resources already allocated.
- Memory Map - a window showing the CPU address space and internal and external memory mapping.

Processor Expert Overview Windows

- Installed Components Overview - this window contains information about installed components in the current version of Processor Expert.
- CPU Parameters Overview - a window providing access to the CPU's database.
- PDF Search - a window allows the user to quickly browse in a PDF documentation for the CPU.

Dialogs

There are the following dialogs for setting the Processor Expert environment:

- Environment Options - Processor Expert plug-in environment options
- Tools Setup - setup dialog box for the tools and the tools menu

There are the following dialog for setting the Processor Expert project:

- Project Options - project options are options concerning the current project and options for the current CPU.

2.1. Main Menu

The Processor Expert Plug-in is integrated into the CodeWarrior IDE application. The CodeWarrior IDE main menu contains a new menu item named "Processor Expert".

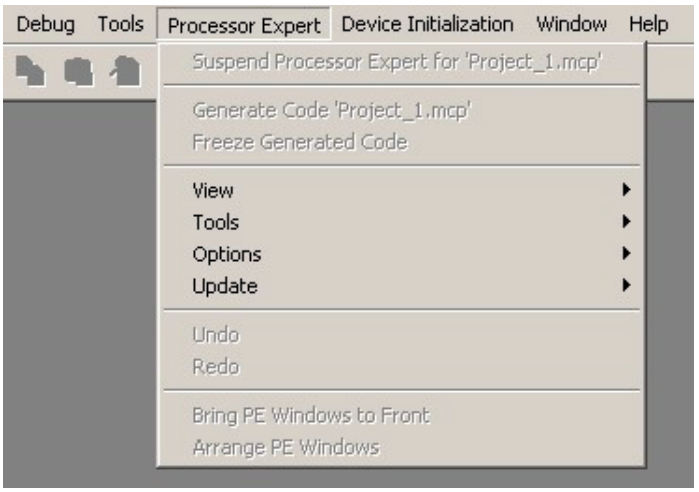


Figure 2.1 - Processor Expert's Integrated Menu

The Processor Expert's plug-in menu consists of the following items:

- **Suspend/Open Processor Expert**
- **Generate Code "ProjectName.mcp"**
- **Freeze Generated Code**
- **View**
- **Tools**
- **Options**
- **Update**
- **Undo**
- **Redo**
- **Bring PE Windows to Front**
- **Arrange PE Windows**

Note: Processor expert's help is placed in the Codewarrior's 'Help' menu.

Suspend/Open Processor Expert

- **Suspend Processor Expert for ...** will disable the usage of Processor Expert for the currently opened project. The Processor Expert tab disappears from the project panel and the generated code of the project will stay in the project in the state it is after the last code generation in.
  See also the **Freeze Generated Code** command that allows to temporarily disable generation of all generated code without completely suspending PE.
- **Open Processor Expert for ...** will enable PE for the current project. If the PE is opened for a project that had never had PE activated, PE will popup a dialog that notifies the user that PE will add the files (IOMap.c/h) containing declarations for the peripheral modules to the project. The user should know how to adapt the project himself (there might be other files (e.g. linker parameter file) that can cause conflicts as well. In case that the command is applied for the project which has already been using Processor Expert previously (even if it is disabled at the moment), PE will automatically find an existing Processor Expert project file (.pe) and will use data from that file. This command is intended for **experienced users only.** A new Processor Expert project should be created using the **File > New** command and selecting the Processor Expert in the new project wizard.

Code Generation - Invokes code generation for the current project. The generated files are automatically inserted into the active (default) target in the CodeWarrior's project. Generated files corresponding to the Embedded Components can be accessed in the "Generated Code" folder in the "Files" tab in the CodeWarrior project window. The other files, intended to be modified by the user, are generated into the "User modules" folder in the "Files" tab in the CodeWarrior project window. A user can also add specific source code files into this folder. If the linker setting of the default target does not match the CPU in the Processor Expert project, the user is asked whether to automatically correct linker settings in the current target. See *3.5.1 Code Generation* for details.

Freeze Generated Code - This option will freeze the state of the generated code and the code generation will be disabled until the user will un-check this option. All components and project settings will became read-only and it won't be possible to add or remove any components. Processor Expert won't make any changes to the source code. Processor expert, if it detects any changes since last code generation, will offer code generation before switching to the 'frozen' mode.

View

- **Project Panel** - displays the Processor Expert plug-in Project panel.
- **Inspector** - displays inspector window for the currently selected item of the project (Component, CPU component, Peripheral Initialization component, Configuration)
- **Components Library** - shows the Components Library. Components Library shows all supported components in the appropriate version of the Processor Expert plug-in including CPU components.
- **Target CPU Package** - displays the Target CPU Window in CPU Package view. This window displays the target CPU (CPU selected as destination) with its peripherals and pins.
- **Target CPU Block Diagram** - displays the Target CPU Window in Block Diagram view. This window displays the target CPU (CPU selected as destination) block diagram with its peripherals.
- **Error Window** - displays the Error window. This window displays errors, warnings, and hints.
- **Target CPUTiming Model** - displays the CPU timing hierarchy.
- **Peripheral Initialization** - opens the Peripheral Initialization window for the Target CPU. (this command is available only if a target CPU is selected)
- **Peripherals Usage** - shows the CPU Peripherals Usage window.
- **Resource Meter** - displays the Resource Meter window. The Resource Meter shows the current status of a chip resources usage (or availability).

- **Memory Map** - opens the Memory Map window. This window shows the CPU address space and internal and external memory mapping.
- **Installed Components Overview** - displays a list of installed components and CPUs with additional information about component drivers and projects with typical settings.
- **CPU Parameters Overview** - displays the CPU parameters overview table and the query dialog that provides help for the selection of the most adequate processor.

Tools

- **Tool #1**
- **Tool #2....** - optionally, any other external tool can be added. The tools can be added, modified or deleted in the "Tools Setup" dialog.

Note: Component Wizard can be added also to the Tools menu. Help for the Component Wizard can be found in the Component Wizard.

Options

- **Environment Options**, **Project Options**, **Application Options** - opens an appropriate page within the Processor Expert Options dialog window that allows to customize all settings related to the environment and project. See *2.1.1 Processor Expert Options* for details.
- **Tools Setup** - allows external tools to be included in the Processor Expert's plug-in environment. The tools may then be accessed via the "Tools" menu. The setting changes will take effect after the restart of the CodeWarrior application. See *2.1.2 Tools Setup* for details.
- **Save desktop** - saves the desktop settings (windows' position) to the .DSK file. The desktop file can also be saved automatically if the option **Environment Options | Autosave desktop** is enabled in Environment Options.

Update

- **Update Processor Expert from Package** - this command allows the user to update or add new components from compressed packages (*.PEupd) that can be downloaded from the Processor Expert web site. It is possible to select more components in the selected directory using a multi-select function. To add more components select the requested component using the mouse and holding down the CTRL (or Shift) key.

  The **information on the package content** is shown when the package is selected within an opening dialog window.

Undo [actionname]

Restores the state of the project before a last operation. **This command affects only changes in the project** (i.e. adding or removing components, disabling components etc.). It doesn't work on the source code editor actions. Functionality of this command is influenced by the option **Environment Options | Number of UNDO operations**. The '0' value of this option will disable the functionality of this command.

Redo [actionname]

Applies again the change previously discarded by a use of the Undo command. **This command affects only changes in the project** (i.e. adding or removing components, disabling components etc.). It doesn't work on the source code editor actions. Functionality of this command is influenced by the option **Environment Options | Number of UNDO operations**. The '0' value of this option will disable the functionality of this command.

Bring PE Windows to Front

Sets the main Processor Expert's windows to the front on the screen.

Arrange PE Windows

Arranges all open windows to the default placement on the screen. (Project Panel, Components Library, Cpu Panel, Error Window, Resource Meter, Component Inspector)

2.1.1. Processor Expert Options

Processor Expert > Options > Environment Options
Processor Expert > Options > Project Options
Processor Expert > Options > Application Options
Processor Expert options allows to customize all Processor Expert's settings within one dialog window.

Figure 2.2 - Environment Options Example

The options are organized within three pages

- **Application Options** - options for the Processor Expert code generation (for one target CPU). They are local, i.e they are valid only for the current application (an application is the subset of the project that concerns a given target processor).
- **Environment Options** - options related to Processor Expert's environment behavior.
- **Project Options** - options concerning the current project (all target CPUs).

The item description for an item is provided as a hint when the user places mouse cursor on the item. Press **Help** button to open the options description pages.

Basic, Advanced and **Expert** buttons allow to customize the amount of options shown along to the user's experience level.

This window uses a limited version of Processor Expert Inspector to show the options information. Thus the way of changing options is very similar to the way of configuring a component or configuration. See *2.5.1 Inspector Items* for details.

2.1.2. Tools Setup

Processor Expert > Options > Tools Setup

Tools Setup - allows to include other tools in the Processor Expert environment. The tools may then be accessed via the **Tools** menu.

Options

Note: *Most of the options allow to use macros allowing an access to various Processor Expert and system values. See 2.1.2.1 Tools Setup Macros for details.*
The following options are available for every tool :

- **Tool name** - name of the tool, as it appears in the Tools menu.
  Processor Expert defines the following reserved names that are linked to the Processor Expert operations:
  - *After PE Code Generation* - If a tool with this name is defined, it's automatically invoked after each succesful code generation. This tool allows to perform additional user-defined processing between code generation and compilation (for example based on the XML file with PE settings generated into the DOCs subdirectory).
- **Visible in Tools menu** - whether the tool is available in the Tool menu (it may not be necessary to let it appear in the menu if the settings are meant only for internal make)
- **Application** - the full name (name and path name) of the application (executable file, EXE or COM extension).
- **Working dir** - working directory of the application
- **Application type**
  - *MS-DOS real* - MS-DOS real mode application.
  - *MS-DOS protected* - MS-DOS protected mode application.
  - *Windows 16-bit* - 16-bit MS Windows application.
  - *Windows 32-bit* - 32-bit MS Windows application.
  - *Autodetect* - auto detection (enabled under the Windows NT, Windows 2000 and newer).
- **Hot Key** - Hot Key for launching the tool.
- **Parameters** - parameters of the application.
- **Input file(s)** - only for backward compatibility, value of the $OUTPTH macro (see macros).
- **Output file(s)** - only for backward compatibility, value of the $IN?PTH macro (See *2.1.2.1 Tools Setup Macros* for details.).
  There is no warranty that these items will be supported in the next version of Processor Expert.
- **Comment** - any text describing the tool.
- **Wait for application termination** - Processor Expert waits for application termination before executing any

other operation (it has the advantage of reserving the error window for the application).

- **Redirection of application output** - Processor Expert captures the standard output of the application and displays errors in the Message Window.
  - **Input file** - name of the input file for the application. If you don't specify any, a temporary file will be created.
  - **Output file** - name of the file for the redirection of the application output. If you don't specify any, a temporary file will be created.
  - **Error output** - name of the file for the redirection of the application error output. If you don't specify any, a temporary file will be created.
  - **Hint format** - format of the hint messages. See *2.1.2.1 Tools Setup Macros* for details.
  - **Warning format** - format of the warning messages. See *2.1.2.1 Tools Setup Macros* for details.
  - **Error format** - format of the error messages. See *2.1.2.1 Tools Setup Macros* for details.
  - **Fatal error format** - format of the fatal error messages. See *2.1.2.1 Tools Setup Macros* for details.
- **Exitcode <> 0** - defines the action that will be done when the exit code of the application will not be zero. The following actions are possible:
  - **Ignore** - no action
  - **Display error** - an error message will be displayed
  - **Show output file** - opens the file defined as a tool output file in the editor.

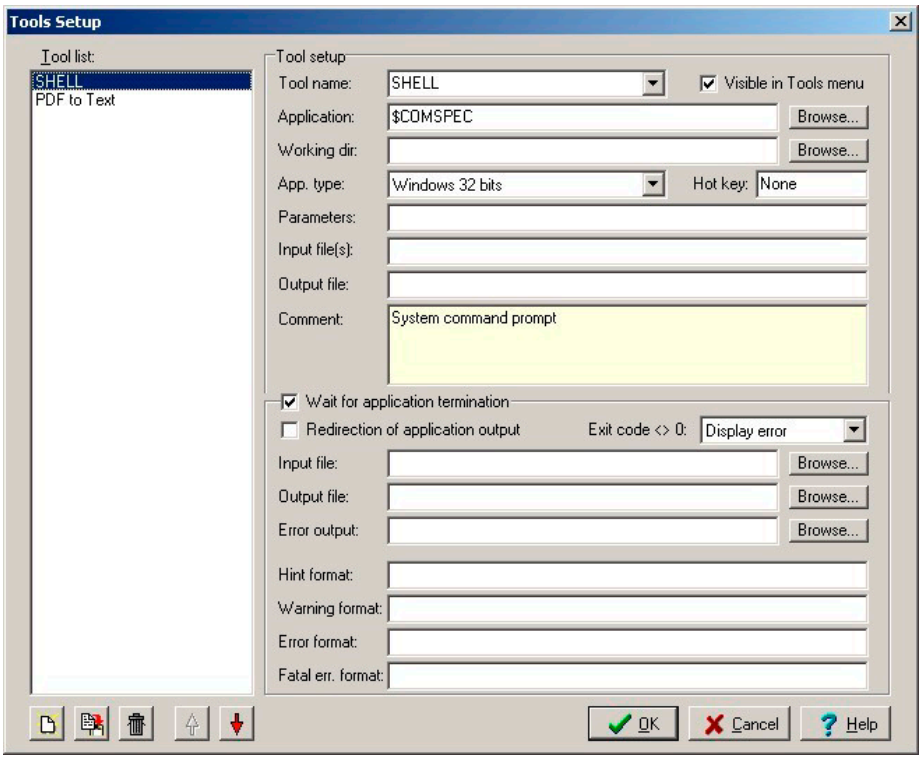


Figure 2.3 - Tools Setup Dialog

2.1.2.1. Tools Setup Macros

Global macros

Global macros can be used in the following tools setup options: *Application, Working directory, Parameters, Input file(s), Output file, Input files for redirection, Output file for redirection, Error output file for redirection, Format of all messages. See 2.1.2 Tools Setup for details.*

- **$PRJNAME** - current name of the project without extension
- **$DIRPRJ** - current directory of the project, absolute path, terminated with a backslash
- **$DIRDRV** - current destination directory of drivers, absolute path, terminated with a backslash
- **$DIRRELDRV** - current destination directory of drivers, relative or absolute path from Project Options setting, terminated with a backslash ($DIRRELEVENT+$EVENTTODRV)
- **$DIREVENT** - current destination directory of main and event modules, absolute path terminated with a backslash
- **$DIRRELEVENT** - current destination directory of main and event modules, relative or absolute path from Project Options setting
- **$DIRBIN** - current destination directory of binary files (maker, linker and object), absolute path terminated with a backslash
- **$DIRRELBIN** - current destination directory of binary files, absolute or relative path starting from Project Options setting, terminated with a backslash ($DIRRELEVENT+$EVENTTOBIN)
- **$DRVTOEVENT** - relative path from drivers directory to main and event modules directory (drivers - Driver subdir., event - Main and event dir. from Project Options)
- **$EVENTTODRV** - relative path from main and event modules directory to drivers directory
- **$EVENTTOBIN** - relative path from main and event modules directory to binary files directory
- **$DIR_PE** - system directory of Processor Expert, absolute path, terminated with a backslash
- **$FILEDIR** - directory of the file that is currently edited or directory of the file that will be opened in External Text Editor, terminated with a backslash.
- **$FILENAME** - name of the file that is currently edited or name of the file that will be opened in External Text Editor (including extension).
- **$FILENAM** - name of the file that is currently edited or name of the file that will be opened in External Text Editor (without extension).
- **$GOTOLINE** - can be used only with the External Text Editor. It is replaced by the line number.
- **$?FILE(Question)** - name of the file set manually, Question is displayed to the title of window
- **$?PARAM(Question, Default)** - parameters set manually, Question is displayed to the title of window, Default is a default value
- **$COMSPEC** - setting of the COMSPEC variable in the Windows environment
- **$MAKEFILE** - name of the currently used makefile for the current project.

Global macros after code generation

The following macros are supported only after successful code generation. *Can be used in the same items as the Global Macros.*

- **$GENDIRPRJ** - directory of the project during last successful code generation, absolute path, terminated with a backslash
- **$GENDIRDRV** - destination directory of drivers during last successful code generation, absolute path, terminated with a backslash
- **$GENDIRRELDRV** - same as $GENDIRDRV during last successful code generation, only relative path ($GENDIRRELEVENT+$GENEVENTTODRV)
- **$GENDIREVENT** - destination directory of main and event modules during last successful code generation, absolute path terminated with a backslash
- **$GENDIRRELEVENT** - same as $DIRRELEVENT during last successful code generation
- **$GENDIRBIN** - destination directory of binary files during last successful code generation
- **$GENDIRRELBIN** - same as $GENDIRBIN during last successful code generation, only relative path ($GENDIRRELEVENT+$EVENTTOBIN)
- **$GENDRVTOEVENT** - relative path from drivers directory to main and event modules directory during last code generation
- **$GENEVENTTODRV** - relative path from main and event modules directory to drivers directory during last code generation
- **$GENEVENTTOBIN** - relative path from main and event modules directory to binary files directory during last code generation
- **$GENPRJNAME** - name of the project during last successful code generation

Macros in format

List of macros that can be used for definition of tool output messages format:

- **$ERRPTH** - name of file where errors were found
- **$ERRMSG** - full / partial error message
- **$FIRROW** - first row position
- **$FIRCOL** - first column position
- **$LASROW** - last row position
- **$LASCOL** - last column position
- **$MSGSKP** - skip next string
- **$MSGEND** - end/continuation of error message
- **$MSGSTR**"string1"string2 - string1 is written to error message and then string2 is skipped as in command $MSGSKP

Backward compatibility

The following macros are supported only for backward compatibility. There is no warranty that they will be supported in the next version of Processor Expert.

- **$REDDIR** - full path name of project directory for redirection
- **$REDINP** - full path name of input file for redirection
- **$REDOUT** - full path name of output file for redirection
- **$REDERR** - full path name of error file for redirection
- **$IN?DIR** - directory of input file, "?" is a number of input file from interval 0..9
- **$IN?NAM** - name of input file, "?" is a number of input file from interval 0..9
- **$IN?EXT** - extension of input file, "?" is a number of input file from interval 0..9
- **$IN?PTH** - full path name of input file, "?" is a number of input file from interval 0..9
- **$OUTDIR** - directory of output file
- **$OUTNAM** - name of output file
- **$OUTEXT** - extension of output file
- **$OUTPTH** - full path name of output file
- **$DRIVERS(FORMAT)** - list of the all generated drivers
- **$EVENTS(FORMAT)** - list of all generated event modules
- **$SHARED(FORMAT)** - list of all generated shared modules

2.2. Help and Manuals

Help > Processor Expert >

The following items are available within this menu:

- **Processor Expert Help** - the start page of the Processor Expert plug-in help.
- **Concepts** - introduction to Processor Expert concepts.
- **Benefits** - who Processor Expert may benefit.
- **User Interface** - description of Processor Expert plug-in environment.
- **Tutorial** - tutorial course.
- **Quick Start** - how to start with Processor Expert plug-in.
- **Embedded Components** - index page of the Embedded Components documentation.
- **Component Categories** - index page of the Embedded Components Categories.
- **Supported CPUs, Compilers and Debuggers** - list of CPUs/Compilers/ Debuggers supported in the current version of Processor Expert plug-in.
- **View Readme and Revision History** - Displays information about used Processor Expert plugin version, basic installation instructions, content of installation, FAQ, history of the signifiacnt changes from previous versions, known problems and limitations and other related information.
- **User Guide** - Opens a brief user's guide delivered with Processor Expert.
- **Search in PDF Documentation of the Target CPU** - displays PDF documentation of the current CPU in

the PDF Search window. It is possible to search any keyword in the CPU documentation based on the original manufacturer's CPU manual. See *2.16 PDF Search* for details.

- **Go to Processor Expert Home page** - display Processor Expert home page in default Internet browser.
- **Processor Expert On-line Support** - opens the web pages related to the customer support for the currently run Processor Expert version.
- **About Processor Expert & Tip Of The Day** - displays the **About dialog** containing information about the Processor Expert product version for the target CPU family and current version of Processor Expert IDE. The **Tip of the day** is displayed along with this dialog. Next tips can be viewed with using the button 'Next tip'.

  The **Installed updates** button opens the dialog containing all already installed update packages. After each update Processor Expert automatically copies installed update package to the folder shown by this dialog. Selecting an update package will show a window with detailed update description. The dialog is for information only and no action is done with the selected file.

2.3. Project Panel

Processor Expert > View > Project Panel

Processor Expert Project Panel is a tab in CodeWarrior's project window (panel). When the 'Project Panel' is mentioned in Processor Expert documentation the 'Processor Expert Project Panel' is understood.

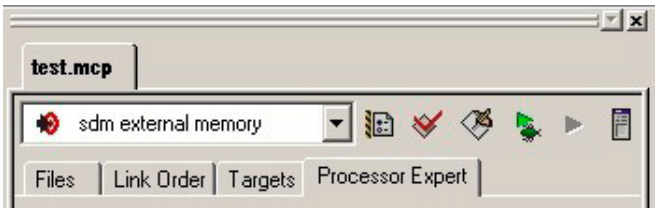


Figure 2.4 - Processor Expert tab

The **Project panel** shows the application components:

- **Configurations** of the project.
- Operating System - Components related to operating system used by the application (if there is some operating system present).
- CPUs (**CPU components**) included in the project
- **Embedded Components** included in the project. Every component inserted in the project is displayed in the project panel and has a subtree showing
  - **Methods** - Methods allow runtime control of the component's functionality.
  - **Event 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 can be created by the user for low level interrupt processing. For items, whose ISR names have been specified within a component settings, a user-specified name of an ISR and name of the interrupt vector is shown. If an ISR name was not specified (interrupt has to be disabled in this case), only the interrupt vector name is present.

All component's items has its status icon that signalizes the enabled (☑) or disabled (☒) state. If this state cannot be directly changed, the background of the icon is gray. For more details please see chapter *3.2.1 Embedded Components*.

- **User modules** included in the project (main module, event module, external user modules ...).
- **Generated Modules** - This folder contains the modules generated by Processor Expert. There is a special subfolder for the generated Component Modules. For the component module description please see chapter *3.5.1 Code Generation*.
- **External Modules** - This folder contains the modules that are not generated by Processor Expert but are required for the application such as libraries or system modules. These files are not influenced by Processor Expert but they are linked to the final application.

Note about user, generated and external modules: *The header file (.h) of a module is not shown if its name (without extension) corresponds to an already displayed .c file. To access such header file, there is possible to use the* ***View Header File*** *command of the module (.c file) pop-up. See 2.3.5 User and Generated Modules Pop-up Menus for details.*

- **Documentations** - list of files attached into project as documentation, with relative or absolute path. No actions are made with these files. Please refer to chapter *3.5.1 Code Generation* for details on generated documentation files.

All Project Panel items are organized in folders in a tree. You can expand and collapse a tree's branches by clicking on the plus "+" or minus "-" signs, respectively. You can create your own folders in the ***Components*** folder and **move components between them** using mouse drag and drop function.

The following icons indicate the **status** of each project panel item:

✓	**Components** - Processor Expert didn't found any problems in the component's settings. **Configurations** - configuration is selected as active. **CPUs** - CPU is currently selected as a target CPU. **User Module, Generated Module** - The module is all right and included in the project.
✖	**Components** - Gray cross means that component is disabled and code won't be generated for it. **Configurations** - Configuration is not active. Double click the ✖ icon to select it as active configuration. **CPUs** - CPU is not selected as target CPU. Double click the ✖ icon to select it as the Target CPU.
!	**Components, CPUs** - components (CPU component's) settings are wrong or conflict with another component. See *3.3.3 Design Time Checking: Consequences and Benefits* for details.
?	Possibly incorrect setup was found in project or in component's settings. The warnings are displayed in Error window including simple description.

Icons  near the component's icon mean the component is individually setup for preserving user changes in generated code. See *3.5.4 User Changes in Generated Code* for details.

The Project panel window allows **quick access** to supported **methods and events** using mouse. See paragraph Other mouse actions.

Pop-up menu

Project panel pop-up menu is accessible by right click on the empty (white) area of the Project Panel. Contains basic operations related to the project and components.

- **Open Project** - allows the user to open Processor Expert project from disk.
- **Save Project** - saves the current state of the project (e.g. all the component and processor expert settings). Project is also automatically saved when CodeWarrior or project is closed.
- **Copy Project to...** - copies the Processor Expert project file (.PE containing settings of all components) and the user modules into another directory. It is useful for backing up the state of the project. The stored file could be opened again using **Open project** command from this menu.
- **Reload Project** - Reloads project from the last saved state on the disk.
- **Add Component(s)** - allows to add components from the project. Shows components library dialog.
- **Import...** - imports the content of the file containing exported objects (e.g. components, configurations...) or whole project.
  All items from the imported file will be added into the current project.
- **Export...** - exports the selected objects in the project panel (e.g. components, configurations...)to the specified file. It's possible to insert them into another project using the **Import...** command.
- **Cut** - cuts selected component with its settings to the clipboard.
- **Copy** - copies selected component with its settings to the clipboard.
- **Paste** - inserts component from the clipboard to the current project.
- **Help** - displays related information for currently selected component or method. If there is nothing selected this help page for Project Panel is displayed.

Pop-up Menus of Objects

Pop-up menus of individual objects in project panel are accessible with **a right mouse button click on the object's icon or label**.

- **Configurations pop-up menu**
- **CPUs pop-up menu**
- **Components and its methods/events/init code pop-up menus**
- **User, generated and external modules menus**
- **Documentations pop-up menu**

Other mouse actions

Drag'n'drop

- **Dragging method with the left mouse button** to the source editor will place a method call to the source code. If the shift key is hold while the users drag and drops the method, the call is placed exactly to the mouse cursor position on the line. Otherwise the call is placed on the new line. A behavior of this function is controlled by the option **Environment Options | Drag'n'drop method declaration** . See *2.1.1 Processor Expert Options* for details.
- **Dragging user module** into the source code will create an #include command (#include "user_module_name.h") at the place of the cursor.
- The user can drag'n'drop **components within the Project Panel** to reorganize component trees (CPUs,

Components, Documentation)

- The user can drag'n'drop **component folders within the Project Panel** to reorganize them. By default, the folder is placed after the folder where it is dropped onto on the same level as that folder. If the **SHIFT** key is pressed when the mouse button is released, the dragged folder is placed inside the folder it is dropped into.

Multiselect

- Using the **Ctrl and Shift key** together with cursor key or left mouse button allows to select multiple items.

Double click

- Double-clicking the **component icon** in the Project panel opens the **Component Inspector**
- **Clicking on the selected component name** in the Project Panel allows you to edit the name of the selected component
- Double clicking on any event/method/initialization **enable/disable icon** changes its **enable/disable state** (you can do it also via the component inspector)
- Double clicking on any **event/method name after code generation** opens the **file editor/viewer** at the position of the event/method's code
- Double clicking on any **ISR** opens the source code editor at the interrupt routine (if its name has been specified within the component properties).

Automatic Hints

- Placing the cursor on any **event/method icon/name** displays the event/method's and parameter's description and syntax
- Placing the mouse over any **component icon/name** displays the component description

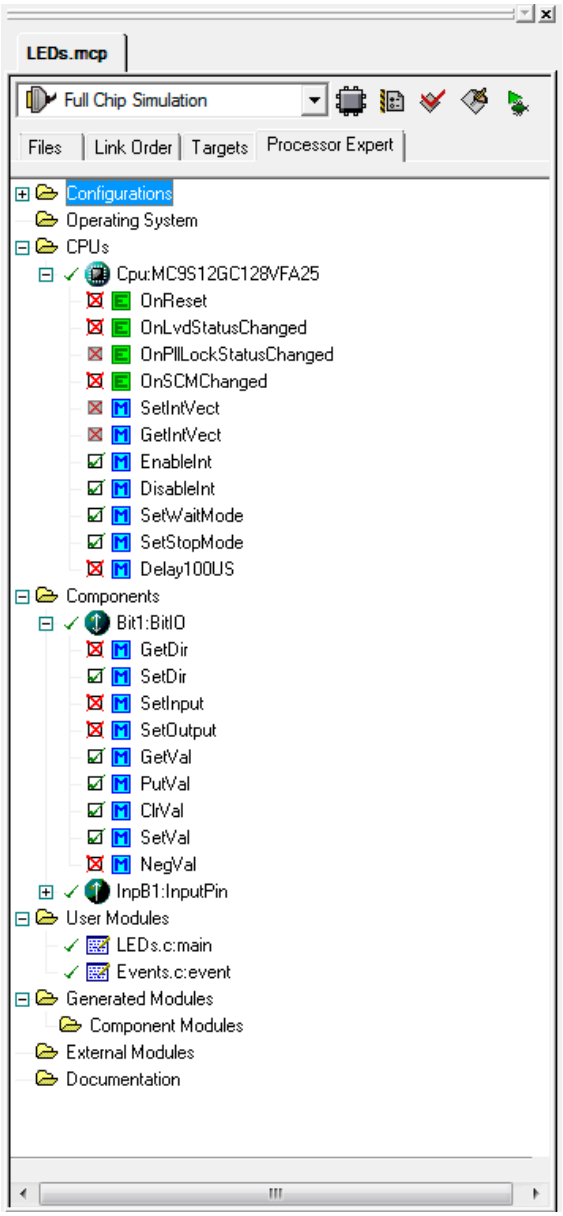


Figure 2.5 - The content of the Project Panel

2.3.1. Configurations Pop-up Menus

Configurations folder pop-up menu

This menu is opened by right-clicking on the *Configurations* folder icon in the Project panel . Following commands are available:

- **Add new configuration** - add a new configuration into the project. All configuration settings (i.e. target CPU selection and state of all components) are copied from the currently active configuration to the new one.
- **Configurations Editor** - opens the Configuration Editor.
- **Expand/Collapse** - expands/collapses one level of the folder's tree.
- **Expand all** - completely expands the folder's tree.
- **Collapse all** - completely collapses the folder's tree.

- **Delete all configurations** - deletes all the content of the folder.
- **Help** - displays documentation.

Configuration pop-up menu

This menu is opened by clicking on the icon of one of the configurations in the configurations folder of the Project panel. Following commands are available:

- **Configuration Inspector** - invokes Configuration Inspector (default on double-click) .
- **Select Configuration as Active** - selects this configuration as active.
- **Delete Configuration** - removes this configuration from the project.
- **Add New Configuration** - adds a new configuration to the project.
- **Rename Configuration** - renames this configuration.
- **Help** - displays documentation.

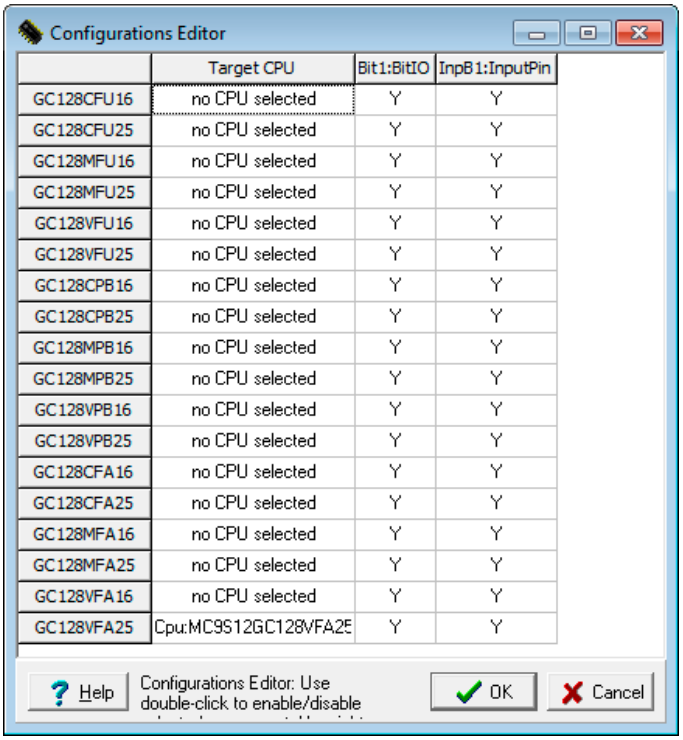
For more information about configurations see chapter *3.3.2 Configurations*.

2.3.2. Configurations Editor

Project panel > Configurations pop-up menu > Configurations Editor
Configurations editor shows a table of all configurations and currently selected target CPUs and Embedded Components state in the configurations. The leftmost gray column contains a name of one configuration in each row and the header of the table contains the names of the components. For details on configurations please see the chapter *3.3.2 Configurations*.

The first column of the table data contains the name of the currently selected target CPU for each configuration. To choose a different target CPU use the pop-up menu available after the right mouse button click on the appropriate cell.

Next columns represent the components in the project. Each field in the components columns contains a 'Y' if the component is enabled or 'N' if the component is disabled in the specific configuration. To change the value of the field from 'Y' to 'N' or vice versa double click the field with the left button.



2.3.3. CPUs Pop-up Menus

CPUs Folder Pop-up Menu

This menu is opened by right-clicking on the *CPUs* folder icon in the Project panel. It proposes a set of commands to manage CPUs folder. Following commands are available:

- **Expand/Collapse** - expands/collapses one level of the folder's tree.
- **Expand all** - completely expands the folder's tree.
- **Collapse all** - completely collapses the folder's tree.
- **Change folder name** - edit the subfolder name.
- **Delete folder** - deletes the subfolder and all its contents.
- **Delete all components** - deletes all the contents of the folder.
- **Add subfolder** - creates a new subfolder.
- **Help** - Displays related information for currently selected component or method. If there is nothing selected this help page for Project Panel is displayed.

CPU Pop-up Menu

This menu is opened by right-clicking on a CPU icon in the CPUs folder in the Project Panel.
Following commands are available:

- **CPU inspector** - opens the CPU's Component inspector. Detailed help concerning the Component Inspector items can be found in the CPU's page of Processor Expert help (see**Processor Expert > Help > Supported CPUs and Compilers** page).
- **Select CPU as Target** - if several CPUs are in the current project, it sets the CPU as target - the CPU will appear on the Target CPU window and the code will be generated with the Project options of active target.
- **Rename CPU** - allows you to give a project-specific name to the selected CPU

- **CPU Peripherals Names** - opens the CPU peripherals' names editor.
- **View Target CPU Package** - opens the Target CPU window in Package view.
- **View Target CPU Block Diagram** - opens the Target CPU window in Block Diagram view.
- **View CPU Timing Model** - displays the CPU Timing Model window.
- **View Memory Map** - displays the Memory map window. This window shows the CPU address.

Space and internal and external memory mapping.

- **Search in PDF Documentation** - displays the PDF Search window. The window allows a full-text search in the original CPU manufacturer's documentation. See *2.16 PDF Search* for details.
- **View Source** - displays the generated CPU module in the File editor.
- **View/Edit Event Module** - displays the generated CPU events module in the File editor.
- **View/Edit Main Module** - displays the generated main module in the File editor.
- **View Linker File** - displays the generated linker file (if it exists) in the File editor.
- **View Makefile** - displays the generated maker file (if it exists) in the File editor.
- **View MAP File** - displays the MPA file in the File editor.
- **View List of Methods** - displays the list of methods.
- **Restore Default Template Settings** - restores default setting of the template.
- **Customize this component template** - saves the highlighted (chosen) CPU and its settings as a template.
- **Remove CPU From Project** - removes the highlighted (chosen) CPU from the project.
- **Help** - displays documentation.

2.3.4. Component Pop-up Menus

Components Folder Pop-up Menu

This menu is opened by right-clicking on the *Components* folder icon in Project Panel. Following commands are available:

- **Expand/Collapse** - expands/collapses one level of the folder's tree.
- **Expand all** - completely expands the folder's tree.
- **Collapse all** - completely collapses the folder's tree.
- **Change folder name** - edits the subfolder name.
- **Delete folder** - deletes the subfolder and all its contents.
- **Delete all components** - deletes all components from the project.
- **Add component(s)** - invokes a dialog which allows the user to choose and add new components to project.
- **Add subfolder** - creates a new subfolder.
- **Import** - Imports all items from a .pe file containing exported objects or whole project.
- **Export** - Exports selected objects in the project panel to the file.
- **Help** - displays documentation.

Component Pop-up Menu

This menu is opened by clicking right mouse button on the icon of the component from the components folder of the Project panel. Following commands are available:

- **Component inspector** - opens the Component inspector of the component. Detailed help concerning the Component Inspector items can be found in the component's page of Processor Expert help.
- **Component enabled** - if checked, the selected component is enabled (included in project).
- **Code generation** - allows to individually specify how the and user changes and code generation for the component are handled by Processor Expert. See *3.5.4 User Changes in Generated Code* for details.
  - **Always Write Generated Component Modules** (default) - generated component modules are always written to disk and any existing previous module is overwritten
  - **Preserve User Changed in Generated Component Modules** - smart detection of user changes.

    Note: Smart user changes preservation is not available in this version.

  - **Don't Write Generated Component Modules** - the code from component is not generated. Any initialization code of the component, which resides in the CPU component, interrupt vector table and shared modules are updated.
  - **Compare with Previously Generated Module, Compare with Previously Generated Header Module** - compares a file generated by the component with a previously generated one. The user can use this function to easily track his/her changes made in the generated code. The both commands are available only when the 'Preserve User Changes' option is switched. See *2.1.1 Processor Expert Options* for details. The component has to be setup to 'Preserve User Changed in Generated Component Modules' or 'Don't Write Generated Component Modules' mode (in this pop-up menu). An internal file-editor in read-only comparison mode is used to show the files differences. See *2.15 File Editor* for details.
- **Rename Component** - allows you to give a project-specific name to the selected component.
- **View Source** - displays the generated module of the component in File editor.
- **View/Edit Event Module** - displays the generated events module of the component in the File editor.
- **Restore Default Template Settings** - restores default template settings. All old settings will be lost.
- **Customize this component template** - saves the selected component as a template
- **Disconnect Component From CPU** - removes the link(s) between the component and the associated CPU peripheral(s) (it clears the corresponding properties of the component).
- **Remove Component From Project** - removes the selected component from the current project.
- **Copy to Clipboard** - component with its settings is copied to the clipboard.
- **Help** - displays documentation.

Methods and Events

This menu is opened by right-clicking on a method or event icon in the Project Panel. It proposes a set of commands concerning the selected method/event.

- **Enable/Disable** - enables/disables the selected method/event in current project.

- **View source***(method only)* - shows generated method source. Available only after successful code design.
- **Edit code***(event only)* - opens a selected event in editor. Available only after successful code generation.
- **Help** - displays documentation.

ISRs

This menu is opened by righ-clicking on ISR item in the Project Panel.

- Rename ISR - opens the Component Inspector and selects the property that specifies the name of the interrupt routine.
- Edit Code - opens the source code editor at the interrupt routine (if its name has been specified within the component properties).
- Help - Shows the related help for the component.

2.3.5. User and Generated Modules Pop-up Menus

User Modules Folder Pop-up Menu

- **Add User Module(s)** - This menu allows to add one or more user source code modules of the specified type to the project. The user can choose the file extension from the sub-menu and select the file using a standard windows file-selection dialog. Please note that when a .c module is added, Processor Expert automatically finds a header file (.h) with the same name for it (if it exists). The header file is not visible as a separate item in the list of user files but it's accessible using the **View header file** command of the .c module pop-up menu. Such header file is automatically treated as a part of the project so it cannot be explicitly added into the user files list.
- **New User Module** - The user can choose a new file type and specify a file name and path. The new file is added to the project. Please note that when a .c module is created, Processor Expert creates automatically a header file (.h) with the same name for it. Such header file is not visible as a separate item in the list of user files but it's accessible using the **View Header File** command of the .c module pop-up menu. It also cannot be explicitly added into the user files list.
- **Expand/Collapse** - expands/collapses one level of the folder's tree
- **Expand All** - completely expands the folder's tree.
- **Collapse All** - completely collapses the folder's tree.
- **Delete All User Modules** - removes all previously added user modules from the project. The user is asked for a permission on removing the modules. The main user module *{projectname}.c* and events module *events.c* cannot be removed from the project.
- **Help** - displays an appropriate help page.

User Module Pop-up Menu

- **Edit Module** - opens the source code in the editor.
- **Edit Header File** - opens the header file with the same name as the module in the editor.
- **User Module Enabled** - The user module is enabled for the compilation.
- **User Module Inspector** - The *User Module Inspector* window is shown. It allows the user to customize the name and directory of the module.
- **Remove User Module** - removes the user module from the project.
- **Help** - displays an appropriate help page.

Generated Modules Pop-up Menu

- **View Module** - opens the module source code in the editor.
- **View Header File** - opens the header file with the same name the module.
- **Code Generation** - enables/disables overwriting of the module by Processor Expert. This option is available only for common modules (like vectors.c) or modules not related to a specific component. For component modules generation control use the pop-up menu of the component instead. See *3.5.4 User Changes in Generated Code* for details.
  - **Always Write** - allows a module to be overwritten.
  - **Don't Write** - disables any modification by Processor Expert.
- **Help** - displays an appropriate help page.

External Modules Pop-up Menu

- **View External Module** - opens the module source code in the editor.
- **View Header File** - opens the header file with the same name the module.
- **Help** - displays an appropriate help page.

2.3.6. Documentations Pop-up Menu

Documentations Folder Pop-up Menu

This menu is opened by right-clicking on the *Documentation* folder icon in the Project Panel. Following commands are available:

- **Add documentation file** - add a new documentation file into project
- **New documentation**
  - **Text file** - creates a new text documentation file.
  - **HTML file** - creates a new HTML documentation file.
- **Expand/Collapse** - expands/collapses one level of the folder's tree.
- **Expand all** - completely expands the folder's tree.
- **Collapse all** - completely collapses the folder's tree.
- **Delete all documentation** - deletes all the contents of the folder.
- **Help** - displays documentation.

Documentation Pop-up Menu

This menu is opened by clicking right mouse button on the documentation file from the Documentations folder of the Project panel. Following commands are available:

- **View documentation** - shows the document in the File Editor.
- **Edit documentation** - edits the document in the File Editor.
- **Open in external viewer** - uses the default shell editor to open the document.
- **Remove from project** - removes the document from the project.
- **Help** - displays documentation.

2.4. Components Library

Processor Expert > View > Components Library

Components Library shows supported embedded components including CPU components and component templates. It lets the user select a desired component or template and add it to the project.

Modes

The Components Library contains the following four tabs allowing the user to select components in the following modes:

- **Component Categories** - contains all available components. The components are sorted in a tree based on the categories defined in the components. See *3.2.1.1 Component Categories* for details. Please see below for menu and control details.
- **On Chip Peripherals** - shows all components available for the specific peripherals. All chip peripherals, sorted by name, are listed in the appropriate CPU folder, depending on which peripheral can be used. Current target CPU component is displayed at the top (only if a target CPU component is selected).
  There are three different icons of peripheral folders which depends on the usage of the peripheral.
  - If the peripheral is **fully available**, the folder is displayed by yellow icon.
  - If the peripheral is **partially used**, the folder is displayed by light blue icon.
  - The **fully used** peripheral is displayed by blue icon.

  the Components Library provides the "**On Chip Peripherals**" view for the users, that are not familiar with the components functionality yet (but they know the chip peripherals). This page contains all on-chip peripherals of the selected CPU and for each peripheral list of supported components. So it's very easy to find component, that supports functionality of the selected peripheral. Please see below for menu and control details.
- **Alphabet** - shows alphabetical list of available components. The user can speed-up searching the right component **typing the start of the component name on the keyboard**. Filters can be used here like on the other Components Library tabs. Please see below for menu and control details.
- **Assistant** - guides the user 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. See *2.4.1 Component Assistant* for details.

Component names are colored black and the component template names are colored blue. The components that are not supported for the currently selected target CPU 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.

Quick Help

The button **Quick Help** shows short information about function of the component. The Quick Help is displayed as a part of the Components Library window and is updated when the user selects another component in the tree.

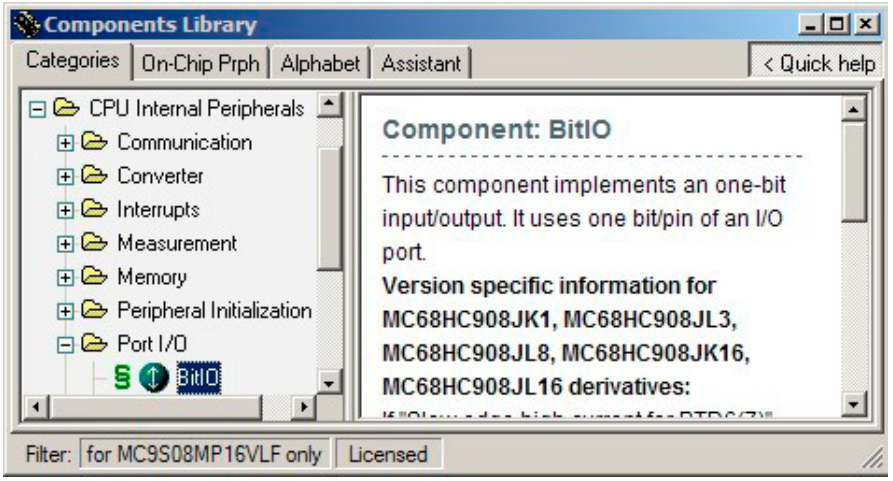


Figure 2.7 - Components Library with Quck Help panel

Filtering

The following filters can be applied on the component list. They could be switched on/off by clicking on two buttons on the bottom bar.

- **for XXXXX only** - If this filter is active, only the components that could be used with the current target CPU derivative are shown (the XXX is the currently selected target CPU).
  If the filter is inactive, Processor Expert also shows components that are not available for the current CPU but in the gray color.
- **Licensed** - If active, only the components with valid license are shown.
  The icon § in *Component categories*, *On-Chip peripherals* or *Alphabet* tab means that there is an available license for the component. If the icon is displayed as a "greyed" § icon, it means that for the selected component a valid license is not available.
- **All questions** - This button appears only in Assistant mode of the Components Library. If it's active, all questions are shown. If it's not, the questions with only one available answer are automatically skipped.

Component Categories, On-chip periherals and Alphabet Modes

In these modes the workspace contains the following controls:

A folder pop-up menu is available by clicking the right mouse button on a folder. It contains the commands:

- **Expand/Collapse** - expands or collapses the folder
- **Expand all** - expands the folder and all its subfolders
- **Collapse all** - collapses the folder and all its subfolders

- **Help on Components Library**- displays documentation for the Components Library.

A Component pop-up menu is available by clicking the right mouse button on a component. It contains the commands:

- **Add the component to the current project** - adds the component to the current project.
- **Delete selected template**- removes the selected template from the Components Library.
- **Help on Component**- displays component documentation.
- **Help on Components Library**- displays documentation for the Components Library.

Components Library pop-up menu is available by clicking right mouse button on the area inside the Components Library window. It contains the commands:

- **Update** - updates new components and templates to the tree according to the appropriate category in the Components Library window.
- **Help on Components Library** - displays documentation for the Components Library.

The target CPU folder pop-up menu is available by clicking the right mouse button on the Target CPU folder in the On Chip Peripheral mode. This menu is the same as the pop-up menu for the target CPU in the project panel. See *2.3.3 CPUs Pop-up Menus* for details. for details.

Peripheral folder pop-up menu is available by clicking right mouse button on the peripheral in the On Chip Peripheral mode. It contains the commands:

- **Expand/Collapse** - expands or collapses the folder
- **Expand All** - expands the folder and all of its subfolders
- **Collapse All** - collapses the folder and all of its subfolders
- **Show Peripheral Structure** - opens the peripheral's structure view - (it is supported for I/O ports, timer's counters, serial ports. It is also supported for devices working in several modes in the CPU block diagram. A list of represented devices for these modes is displayed.
- **Rename Peripheral** - allows the user to rename the selected peripheral. It is supported for I/O ports and pins, watchdog and timers (counters, compare and capture registers, free running devices), A/D converters and A/D channels, CAN, serial ports. See details for renaming peripherals.
- **Show Peripheral Usage** - shows which part of the peripheral is used by the application (visible after code generation). It is supported for I/O ports and pins, timers, A/D converters and A/D channels, CAN, serial ports, watchdog, internal memories (EEPROM and FLASH). See *2.14 Peripherals Usage* for details.
- **Show Peripheral Initialization** - shows initialization values of all "control, status and data" registers. It is supported for all devices displayed on CPU package. See *2.13 Peripheral Initialization* for details.
- **Search Related Info in CPU PDF Documentation** displays the PDF Search window and finds information about the peripheral in the appropriate CPU documentation. It is for possible to search for any keyword in the CPU documentation based on the original manufacturer's CPU manual. (This item is available on the package and on the CPU block only.) See *2.16 PDF Search* for details.
- **View CPU Block Diagram** - displays the CPU block diagram in the Target CPU window.
- **Help on Components Library** - displays documentation on the Components Library

2.4.1. Component Assistant

The Component Assistant is a mode of Components Library window. It guides the user during the selection of components - basic application building blocks. The user answers 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 window has the following parts:

- Control bar with the history navigation buttons and the history line showing answers for already answered questions. The user can walk through the history using the arrow buttons and 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, its added into the project. Also a **pop-up menu** of the component, allowing to add it into the project or show it's documentation, is available on right mouse button click on the line (for details see *2.4 Components Library* help page).
  If more questions are necessary for the component selection, the line with the answer contains a group icon and a in a brackets a number of components that still can possibly be selected. After clicking on such line a next question is displayed.

Please note that a component filtering, controlled by the button(s) on the bottom bar of the Components library window, works in this mode too. The questions and answers are filtered so the components that do not suit to the enabled filter(s) are not visible. Notice that enabling of the filtering might remove some components from the list. If the 'All questions' filter button is not pressed and there is only one answer available, the question is automatically skipped like the user would select an answer. For details please see the Components library help page.

This mode of Components Library doesn't offer addition of CPU components. If you would like to add another CPU component, please switch to another Components Library tab.

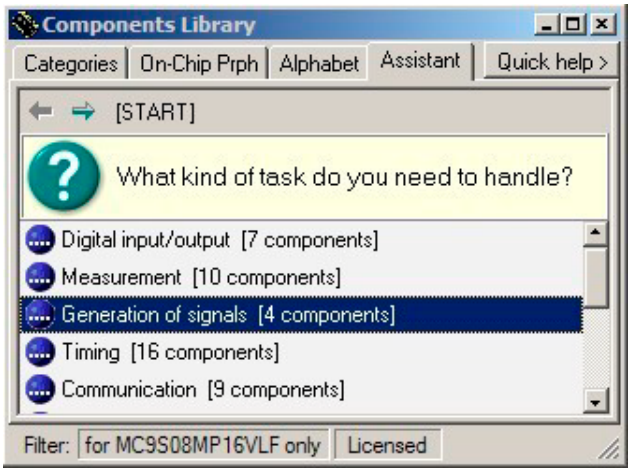


Figure 2.8 - Components Library in Assistant Mode

2.5. Inspector

Processor Expert > View > Inspector

Inspector is universal window, which allows to view and edit attributes of the object selected in the Project Panel. It could be a Component, Configuration, User module or Peripheral Initialization Component. Inspector can work in these modes depending on the type of inspected object.

- **Component Inspector** - provides access to Properties, Methods, Events, and Comments for the Components, arranged in switchable pages. See *2.5.3 Component Inspector* for details. Component Inspector for CPU component offers additional Build options (if a target compiler is selected) and Used peripherals pages.
- **Configuration Inspector** - Provides access to settings of a configuration. See details in chapter Configuration Inspector.
- **User module inspector** - Provides access to settings of a user module.

Figure 2.9 - Example of the Inspector Window content

Window Columns

Inspector window contains the four columns:

- **Item status**
  - green checkmark - item setting is correct
  - red exclamation - item setting is not correct. Items that cause errors or warnings are written in magenta color. See description in the last column or the Error Window.
  - plus or minus - item is a group of settings that can be expanded/collapsed.
  - light background - item is version specific. See *2.5.3.4 Version Specific Items* for details.
- **Item names** - items that are to be set are listed in the second column of the inspector. Groups of items describing certain features may be collapsed/expanded by double clicking on the first line of the group. By double clicking on a method or event item, you may open the File Editor at the position of the corresponding method or event.
- **Selected settings** - the settings of the items are made in the third column. See chapter *2.5.1 Inspector Items* for list of item types.
- **Setting status** - the current setting or an error status may be reflected on the same line, in the rightmost column of the inspector.

Read only items

Any item can be presented as read-only so the user could not change its content. Read only values are gray.

Menu

The following items are available:

- **Component** *(enabled in Component Inspector Only)*
  - **Template**
    - **Restore default template settings** - restores settings of the template.
    - **Save component settings as template** - invokes template editor. See details on Component Templates here.
    - **Active template** - Shows list of currently available templates for the component with currently active template selected.
  - **Change component icon** - allows the user change the component icon.
  - **Autoconnect** - auto connects the component to the CPU.
  - **Disconnect** - disconnects the component from the CPU.
- **Items Visibility** - see the chapter Items Visibility for more information about view modes.
- **Help**
  - **Help on Selected tab** - displays documentation for the current tab.
  - **Help on Inspector** - displays Component Inspector documentation.
  - **Help on Component***(enabled in Component Inspector Only)* - displays documentation for the selected component.
  - **Embedded Components Page** - displays Embedded Components documentation.
- Navigation buttons < > allow the user to browse over the previously inspected components.
- **View regs. button** *(present in Component Inspector Only)* - Pressing this button opens the Peripheral initialization window.

View mode buttons

They are placed at the bottom of the window (Basic, Advanced, Expert). They allow users to switch complexity of the view of the component's items. See *2.5.2 Items Visibility* for details.

Pop-up Menu

This menu is invoked by a click of the right mouse button on the specific inspector item. The menu contains the following commands:

- **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.

- **New Item Into List** - adds a new item before the currently selected one. This item is available only for the list-type properties.
- **Delete Item From List** - deletes a selected item from the list. This item is available only for the list-type properties.
- **Move List Item Up** - Moves the selected row towards the start of the list. This item is available only for the list-type properties.
- **Move List Item Down** - Moves the selected row towards the end of the list. This item is available only for the list-type properties.
- **Help on the Item** - shows the appropriate help page for the selected item.
- **Copy Error Message to Clipboard** - copies the text of an error message for the selected inspector line to the clipboard.

2.5.1. Inspector Items

The following types of the items could be found in the Inspector

Alphabetical list

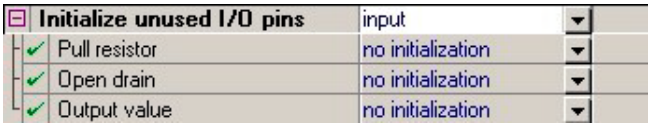
Descriptions

- **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 the + sign will show/hide the items in the group but doesn't influence value nor validity of the items.

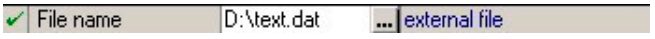
- **Boolean yes / no** - The user can switch between two states of the property using a round icon  .
  The **Generate code / Don't generate code** settings of methods and events works the same way and determines whether the implementation code for the corresponding method or event will be generated or not (you may thus generate only the methods and events used by your application).

- **Enumeration** - Selection from a list of values. If the user clicks the arrow icon (), a list of the possible values for the property is offered.

- **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 will show/hide the items in the group but doesn't influence value nor validity of the items.



- **File/Directory Selection** - allows to specify a file or directory. Clicking the  icon will open a system dialog window allowing to choose a file/directory.



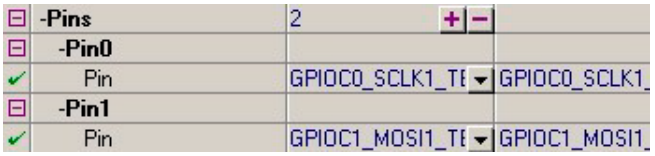
- **Group** - A list of items which can be expanded/collapsed by clicking on the plus/minus icon or by double clicking at the row. Values of the items in the group are untouched.

- **Integer Number** - The user can insert a number of a selected radix. Radix of the number could be switched using the icons D H B (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 arrow icon switches the inspector to the ancestor component that is inherited by the current component. The down-arrow button allows to change the ancestor from the list of possible ancestor. See *3.3.7 Component Inheritance and Component Sharing* for details.

- **Link to shared component** - The dialog button switches the inspector to the shared component that is used by the current component. The down-arrow button allows to change the component from the list of the available components. See *3.3.7 Component Inheritance and Component Sharing* 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. The user may add/remove items by clicking on the plus/minus button. The items in the list can be arranged using a related pop-up menu commands.



- **Peripheral selection** - The user 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** - the user can insert any real (floating point) number.

- **String** - Allows to enter any text or value

- **String list** - Clicking the dialog button will open 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 box gets opened when clicking on ... See *2.5.3.1 Dialog Box for Timing Settings* for details.

2.5.2. Items Visibility

Processor Expert supports **selectable visibility** of 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. To view these items select command **Inspector > Items Visibility > Basic view**.
- **Advanced view** - all items from Basic view and the settings that configure some of more advanced and complex features of the component. To view these items select command **Inspector > Items Visibility > Advanced view**.
- **Expert view** - maximum visibility level - all possible settings and information, including all items from basic and advanced view. To view these items select command **Inspector > Items Visibility > Expert view**.

See also the main page of the Inspector chapter for more information on inspector and view modes switching.

Note: If an error occurred in a property with a higher visibility level than the level currently selected, this error nevertheless will be displayed.

2.5.3. Component Inspector

Component inspector is one of the Inspector window variants. It allows to setup **Properties**, **Methods**, and **Events** of a component. Use command **Help > Help on Component** from Component Inspector menu to see documentation for currently opened component.

Note: Property settings influencing the hardware can often be better presented by the CPU package view using Target CPU window. See 2.7 Target CPU Window for details.

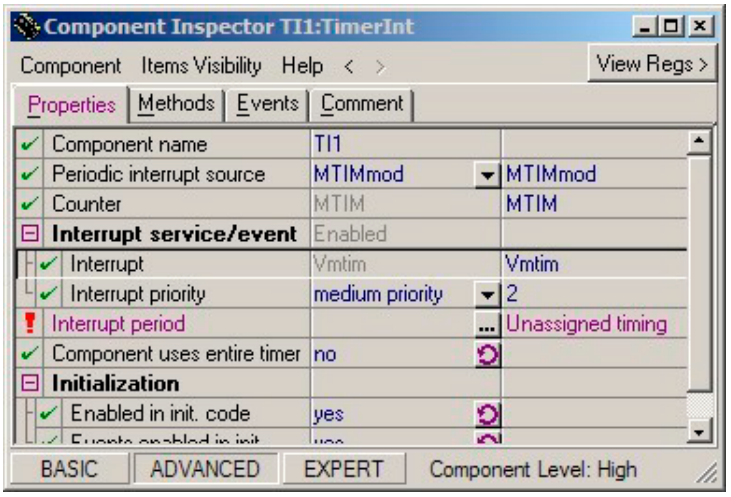


Figure 2.27 - Component Inspector Window

The **Comments** page allows the user to write any comment about the component or setting used. This comment will be displayed in the hint when the mouse cursor will be placed on the component.

The **Build options** page is present **only in the CPU component** and it provides access to the settings of the compiler (or debugger) selected in Project Options. These settings are different for each compiler (or debugger), and are reset every time the compiler (or debugger) is changed.

Peripheral usage

The **Used** page shows list of the CPU component resources. The user can also manually block individual resources for using them in Processor Expert.
The page consists of the three columns:

- First shows the name of the resource. Resources are in groups according to which device they belong to.
- Second column allows the user to reserve resource (for example pin) for external module. Click on icon to reserve/free a resource. **Reserved resource could not be use in Processor expert any more.**
- Third column shows the current status of the resource and the name of the component which uses it (if the resource is already used).

For **menu and view mode description** and other common Inspector window features see chapter *2.5 Inspector* and *2.5.1 Inspector Items*.

Pin sharing

Note: This feature is not available in this version.

Some components allow sharing of the pins. This ability is indicated by a presence of the pin sharing button in the pin selection property line.

Component level

The Component Level is displayed at the bottom of the window besides the view mode buttons. It describes the amount of the peripheral abstraction and a cross platform portability.

- **High Level Components** - highest level of peripheral abstraction. An application built from these components can be easily ported to another microcontroller supported by the Processor Expert.
- **Low Level Components** - The components that are dependent on the peripheral structure to allow the user to benefit from a non-standard features of a peripheral.
- **Peripheral Initialization Components** - These components 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 were designed for initialization of these peripherals (contain only one method "Init" and no events).

Please see chapter *3.2.1.1 Component Categories* for more information.

For more details on component inspector items, please see also the following sub-chapters

- Dialog Box for Timing Settings
- Syntax for the Timing Setup in the Component Inspector
- Defaut Values for Properties
- Version Specific Items

2.5.3.1. Dialog Box for Timing Settings

The **Timing dialog** box provides a user-friendly interface for the settings of component timing features. When clicking on the  button of a timing item in the Component Inspector, the timing dialog box is displayed.

Before you start to edit component timing you should set:

- **Target CPU** in the Project Panel
- **Used peripherals** in the component's properties
- **Supported speed modes** in the component's properties

The settings are instantly validated according to the Processor Expert timing model, for details on the timing settings principles please see the chapter *3.3.4 Timing Settings*.

Timing Dialog Controls

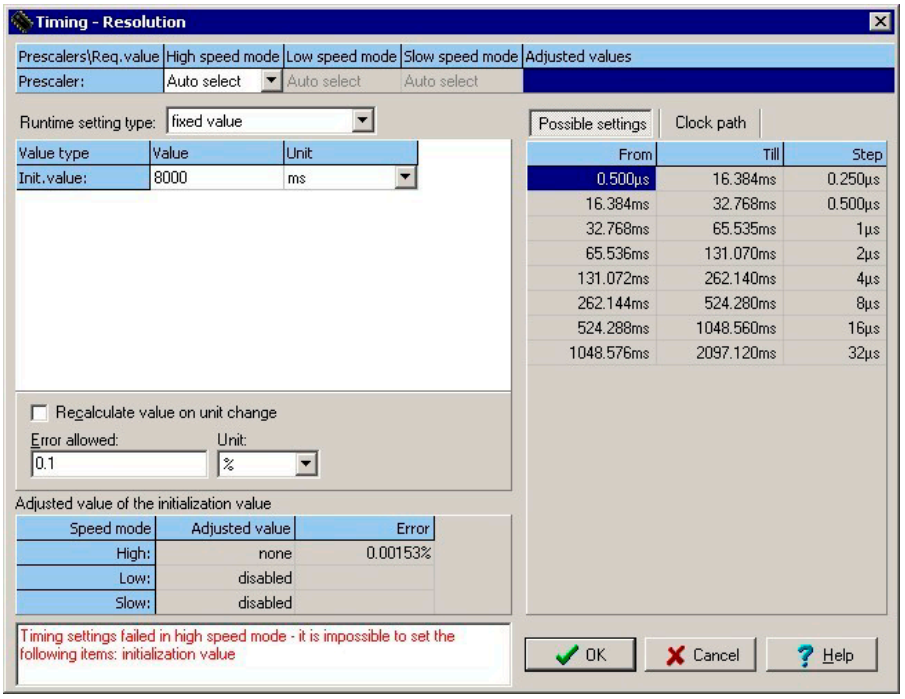


Figure 2.28 - Timing Settings Dialog

Prescalers table

Note: This table is visible only in Advanced or Expert view mode and if it is possible to select a prescaler for the selected peripheral. 2.5.2 Items Visibility

This table occupies the topmost part of the window. It shows a clock source and prescaler(s) values related to the currently edited timing value (see the section Timing values table for details). A presence and number of rows in the table are optional and depend on the hardware structure and a type of the configured value.

The table allows the user to configure his/her requirements on the prescalers. *Autoselect* means that the value is configured automatically by Processor Expert to achieve the requested timing. Otherwise, if there are some values set manually, only these values will be used and possible timing values are limited to the values that can be achieved with the manually set clock source and/or prescaler values.

The table contains a row for each prescaler which is used by the device selected for the component. Each column contains a value corresponding to one speed mode except the last column which shows values of the prescalers that are really set. It's possible to adjust the values only in the columns of the speed modes that are enabled.

The table can contain the following rows:

- Clock source - the clock source is fixed for each speed mode. Even if the runtime setting is used, this value has to be the same for all values from the list (in case of "from a list of values") or for the whole interval range (in case of "from interval"). See below for details on Runtime setting.
- Prescaler, Pre-prescaler - the prescaler influencing the clock of the peripheral selected for the component. If the runtime setting "from a list of values" is used, this value can be different for every item of the list. Otherwise it has to be fixed. In some cases it's necessary to select the clock source before adjusting this value.

Runtime setting configuration

Note: Runtime setting cannot be selected in the BASIC view mode.

Runtime setting type selection determines how the timing setting can be modified at runtime. The following options are available:

- **fixed value**: the timing cannot be changed at runtime.
- **from a list of values**: allows to change the timing by selecting one of predefined values (from the list) using component method "SetXXXMode". This method sets the values(s) corresponding to the selected timing into appropriate prescaler and other peripheral register(s). The values (modes) in the list can be added/removed by editing the timing values table.
- **from interval**: allows to change a timing freely within a selected interval, while all values of the interval are selected with specified precision. Prescaler value is fixed in this mode, timing is set only using compare/reload registers value. **It means that there must be possible to set all values within the interval by using the same prescaler.**
  Please note that this kind of runtime setting requires runtime computations that can be time and space consuming and may not be supported on all microcontrollers.

Note: Some of the methods used for runtime setting of timing will be enabled only if the appropriate runtime setting type is selected.

Timing values table

This table allows to set or modify a requested value(s) for the configured timing. Each row represents one time value and the number of rows depends on the selected type of runtime setting.

- For the option "*fixed value*" there is only one row (Init.Value) containing the fixed initialization value.
- For the option "*from a list of values*" there is one row for each of the possible timing modes. Please see the section There is possible to enter 16 possible values (modes). The empty fields are ignored. The user can drag and drop rows within the table to change their order. *Runtime setting configuration* within this chapter for more information.
- For the option "*from interval*" the table has three rows that contain the Initial value, low limit and high limit of the interval. Please see the section *Runtime setting configuration* within this chapter for details on this type of runtime setting.

There are two editable columns:

- **Value** - Fill in a requested time value (without units). The drop-down arrow button allows to display a list of neighboring values and the user can select one of them. There is also possible to set the value by double-clicking on a value from the possible settings table (see below).
- **Units** - units for the value. When the **Recalculate on unit change** check-box below the table is checked, the value (in the previous column) is automatically re-calculated to be the same in the newly selected units.

Timing precision configuration

It is possible to specify desired precision of the timer settings by using one of the following settings (which one is used depends on the type of the timing) :

- The field **Error allowed** allows to specify a tolerated difference between the real timing and the requested value. The **Unit** field allows to specify the units for the error allowed field (time units or a percentage of the requested value).
- The **Min. resolution** field is used for setting interval or capture component timing. Allows the user to specify maximal acceptable length of one tick of the timer.

In the case of interval settings type, the **% of low limit** (percentage of the low limit value) can be used as the unit for this value.

Minimal timer ticks

Note: This item is available only for setting of period in components where it's meaningful (e.g. PWM, PPG).

This item allows to set a minimal number of timer-ticks in period. It means that it will be possible to set a duty of the output signal to at least the specified number of distinct values for any period of the output signal at runtime. Value 0 means no requirements for the timer settings.

Adjusted values

This table displays a real values for each speed mode the currently selected row in the Timing values table. These values are computed from the chosen on-chip peripheral settings, selected prescaler(s) value and the difference between a value selected by the user and the real value.

Status box

The status box displays a status of the timing setting(s). If the timing requirements are impossible to meet, a red error message is displayed, otherwise it's blank and gray.

Possible settings table

This table is displayed on the right part of the timing dialog if the the button **Possible settings** on the top is pressed. The table shows values supported by the target CPU for the selected peripheral.

If there are only individual values available to set, the table contains a list of values - each row represents one value. If there are intervals with a constant step available, each row contains one of the intervals with three values: **From**, **Till** - minimum and maximum value, **Step** - a step between values within the interval.

The way the values are displayed may be dependent on

- Runtime setting type - if it's "fixed value" or "from list of values" - the values present in more rows (overlapping intervals) are shown only once. If the "from time interval" runtime setting type is used, all intervals possible to set by various prescalers combinations are shown, even if they overlap. It's because intervals can differ in resolution (i.e. number of individual timing steps that can be achieved within them).
- Timing unit - if a frequency unit is used (e.g. Hz, kHz), the *step* column is not visible

By clicking on the table header, there is possible to order the rows by the selected column. By clicking the same column again, the user can switch between ascending or descending order.
Double clicking on a value will place the value into the currently edited row within the Timing values table.

The values listed in the possible settings table depend on the following timing settings:

- prescalers
- minimal timer ticks

and it also depend on

- selected CPU
- selected peripheral
- speed-modes enabled for the component

The table contains a **speed mode tabs** (speed modes and related settings are supported only in EXPERT view

mode) that allow to filter the displayed intervals for a specific speed mode or show intersection of all. Please note that the intersection contains only a values that can be set in all speed modes with absolute precision (without any error), so some values that are still valid due to non-zero *Error allowed* value are not shown.

Clock path

Note: This table is available in Advanced or Expert view modes only.

This table is displayed on the right part of the timing dialog if the the button **Clock path** on the top is pressed. Each row of the table represents a device influencing a clock on the path from the clock source on the first row (e.g. the crystal, internal oscillator or PLL) to the peripheral or a part of it. The last row contains total divider (a total ratio between the source clock and the resulting peripheral clock) in the form of ***{mult}/{div}** where {mult} is multiplier and {div} is divider or **/{div}** if no multiplier is present.
When multiple speed modes are used, it's possible to see the clock path in the individual speed modes by switching the tabs at the bottom.

Please note that the clock path may not be displayed correctly in case that there is not possible to set the required timing (i.e. when error message is displayed).

The table contains the following columns:

- Clock device icon - a symbol representing a type of the clock device. See *2.8 CPU Timing Model* for details.
- Name - name of the clock source, prescaler or other timing device
- Presc.value - If the row represents division or multiplication, this field contains a value of of divisor/multiplier for the clock signal. Divisor and multiplier are distinguished by icon and hint text.
- Frequency - a clock frequency on the output from the clock device

Note: Some devices shown in the clock path are not physically present on the CPU but are only a necessary part of a virtual model that allows to represent a real timing behavior. The prescaler values in rows may not match the values exactly written into the control registers (e.g. compare/reload) but they are clock divisor values used by the timing model. Also if the component uses virtual clock device as a clock source (RTIShared), there is a device named SW extension that represent a divisor provided by software counter.

For more information on the CPU timing and list of clock device icons with descriptions please see the chapter *2.8 CPU Timing Model*.

2.5.3.2. Syntax for the Timing Setup in the Component Inspector

The properties that contain timing settings can be configure using the timing dialog (For details please see the chapter *2.5.3.1 Dialog Box for Timing Settings*) or directly by entering the timing value. If the timing values are specified directly, it's necessary to type not only a **value** (integer or real number) but also the **unit** of that value. The following units are supported:

- **microseconds** - a value must be followed by `us`.
- **milliseconds** - a value must be followed by `ms`.
- **seconds** - a value must be followed by `s`.
- **CPU ticks** - a unit derived from the frequency of external clock source. If there is no external clock enabled or available, it is derived from the value of internal clock source. A value must be followed by `ticks`.
- **Hertz** - a value must be followed by `Hz`.
- **kilohertz** - a value must be followed by `kHz`.

- **bit/second** - a value must be followed by `bits`.
- **kbit/second** - a value must be followed by `kbits`.

Example

If you want to specify 100 milliseconds, enter `100 ms`

For more details on timing configuration please see the chapter *3.3.4 Timing Settings*.

2.5.3.3. Default Values for Properties

Some properties can have a global default value. Once you enter the value for the property and you confirm that it will be the default value for the property then it will be used automatically in the future.

If you open a project with different settings or change the value of the property in the Component Inspector, a Processor Expert automatically offers the following options:

- **use new value as default value** - remember the newly entered (or just loaded) value as the new default value (change default value),
- **use my default value** - cancel changes (discard loaded value) and use previous default value,
- **use new value and do not change default value** - use the newly entered (or just loaded) value and do not change the default value,
- **do not use default value for this item** - *permanent settings*, never use default value for this property,
- **use always default value for this item during project loading** - *permanent settings*, do not display this dialog and always use the default value during project loading and template creation.
  Note: This item is accessible only during project loading.

Permanent settings or default values can be removed in Environment Options.

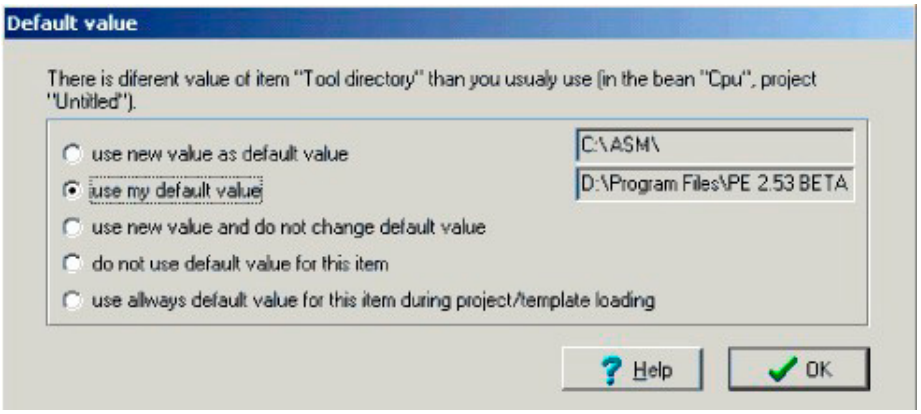


Figure 2.29 - Changing the Default Value

2.5.3.4. Version Specific Items

The **Version specific items** (properties, methods and events) are displayed only for CPU derivatives that support it. These items cover the special capabilities of the CPU and they are not present for all CPUs.

The version specific item is displayed as a highlighted field in the first column of the Component Inspector. See the following picture. There are two items with this feature:

Some items of the component are displayed as mirrored items from the CPU component or global settings. The global setting means that it is possible to set the item in any component where the item is available and the setting is used for all appropriate components. The items are visible if they have any relation to the component settings. The information about the mirroring is visible in the hint of the item.

Figure 2.31 - Hint With Version Specific Info

2.5.4. Configuration Inspector

Configuration Inspector is a variant of an Inspector Window. It shows the settings that belong to one configuration. It could be invoked from configurations pop-up menu in the Project Panel (Click on a configuration with the right button and choose the *Configuration Inspector*). For details on configurations please see the chapter *3.3.2 Configurations*.

Properties

The *Properties* tab contains optimization settings related to the configuration. These setting should be used when the code is already well debugged. They could increase speed of the code, but the generated code is less protected for the unexpected situations and finding errors could be more difficult.

- **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 the 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.

Comment

The *Comment* tab allows to enter a text that will be shown in the hint when the mouse cursor will be placed on the configuration. It could be any text related to the configuration, for example an explanation of the configuration purpose or some necessary hardware settings.

2.6. Error Window

Processor Expert > View > Error window

This window displays errors, warnings, and hints that are found during:

- project checking
- code generation,
- running of an external tool.

Some errors are found right after inconsistent or incorrect data have been entered, others during the code generation of a project. The single messages mention the component where the error was found. If an error concerns two components (where conflict results for example from using the same on-chip peripheral), the error will be attributed to both components.

If the user clicks the right mouse button a pop-up menu is shown allowing user to delete either tools or code generation errors, warnings and hints in order to improve the readability of the Error window.

Figure 2.32 - Processor Expert Error window

Pop-up Menu

The pop-up menu invoked by a right mouse button click contains the following items:

- **Delete All Tool Errors, Warnings and Hits** - removes all tool errors, warnings and hints listed in the error window
- **Delete All code generation errors, warnings and Hints** - removes all code generation errors, warnings and hits listed in the Error window
- **Copy to Clipboard** - copies the whole content of the window as a text to the clipboard.
  Note: This command can be very useful in the case of contacting our support personnel with a component

setup issue.

- **Help** - display documentation

2.7. Target CPU Window

Processor Expert > View > Target CPU Package
Processor Expert > View > Target CPU Block Diagram
Processor Expert > View > Target CPU Structure

This window displays selected target CPU with its peripherals and pins (possible data directions of single pins are indicated by blue arrows on the CPU package when a component uses these pins). Several **display modes** are supported. It is possible to switch the display mode by pushing buttons in the left side menu of the window.

Control Buttons

The meanings of the buttons on the left side are:

-  **Rotates CPU** - rotate the CPU 90 degrees to the right.
-  **Show user names on CPU package** - switches the pins' and peripherals' default names (from catalog) for user-defined names.
-  **Zoom in** - increases the detail level of the view. The whole picture might not fit the viewing area.
-  **Zoom out** - decreases the detail level of the view. Processor Expert tries to fit the whole picture to the viewing area.
-  **Show CPU package and peripheral** - switches to the CPU package view mode.
-  **Show BGA CPU package** - switches to the CPU BGA package view mode.
-  **Show CPU block diagram** - switches to the CPU block diagram view mode.
-  **Show CPU peripherals in a list** - switches to the CPU peripherals list view mode.

View Modes

- **CPU package mode** - a realistic view of the CPU package with pins and peripherals. Each allocated peripheral contain an icon of the component that allocates it. For allocated pins also the component icon with the connection is shown.

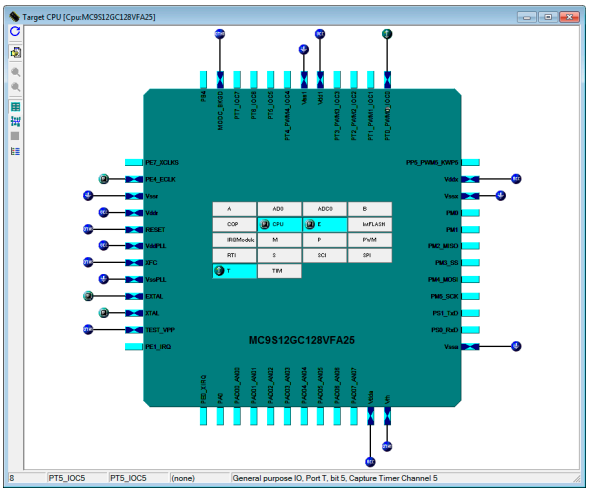


Figure 2.33 - Target CPU - CPU package view mode

- **CPU BGA package mode** - This mode is available only for CPUs with grid-array pins layout. It is similar to the package mode, but the pins hidden by package are shown and the peripherals are hidden.

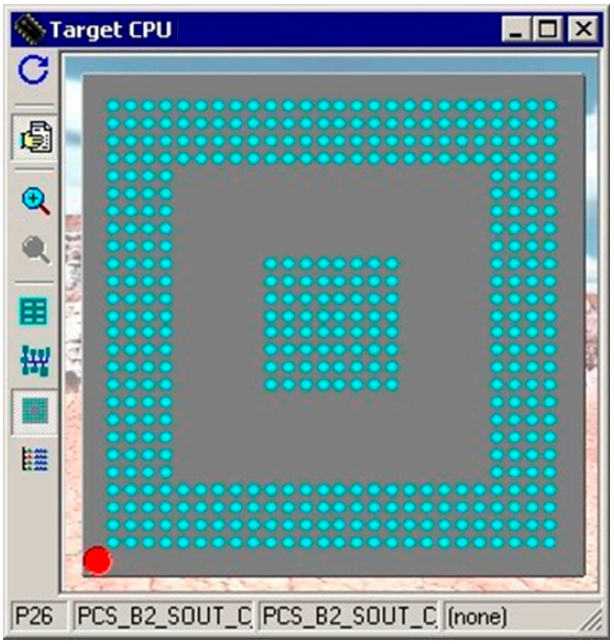


Figure 2.34 - Target CPU - BGA CPU package view mode

- **CPU block diagram mode** - a view of the CPU block diagram based on the documentation of the CPU manufacturer. Every part of the CPU is represented by a block. Every block that contains a resources that can be allocated by Processor Expert contains the slots for every resource (e.g. pin or channel). If the resource is allocated, the slot contains the icon of the allocating component.

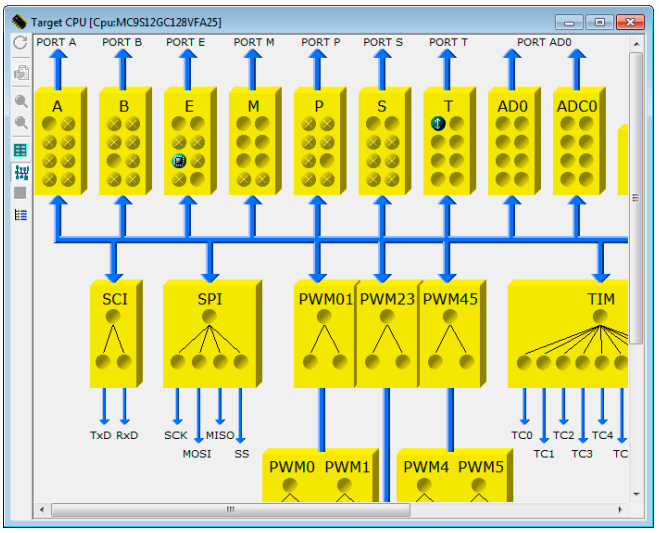


Figure 2.35 - Target CPU - MCU block diagram view mode

- **CPU peripherals list** - a list of all peripherals of the CPU is displayed. If a peripheral is unallocated by Processor Expert, it is displayed as a gray icon. Otherwise, the icon of the component that allocates the peripheral is displayed. The same mouse commands are available as in the other view-modes, except the operations with pins (pins are not visible in this mode).

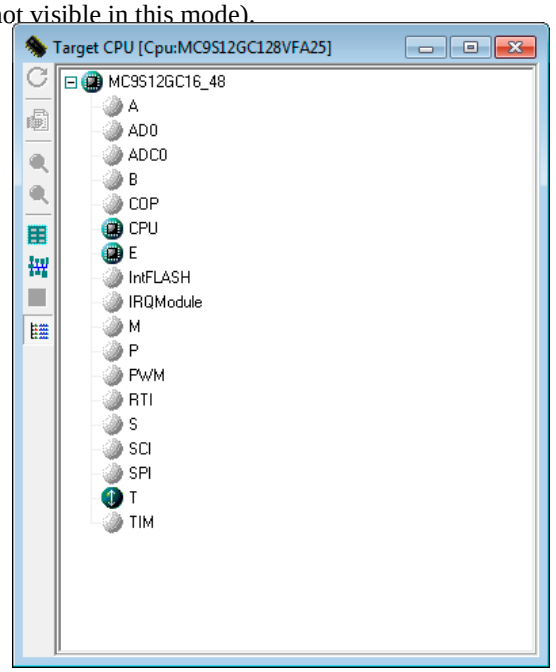


Figure 2.36 - Target CPU - Peripherals List Mode

Pins

The following information about each pin is displayed on the CPU picture:
(all pins are displayed only in the CPU package 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

Pin names are shortened and written either from left to right or from up to down 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, for example . The allocation of peripherals by special devices can be influenced by CPU properties.

Hints

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 CPU database

Component icon hint contains:

- component name
- component type
- component description

Shared Pins

If a pin is shared by multiple components, the line connecting the pin to the component has a red color.

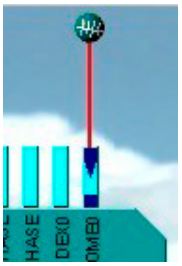


Figure 2.37 - Shared pin connection

On-chip peripherals

The following information about each on-chip peripheral is displayed on the CPU 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 for peripherals working in several modes:
Some peripherals work in several modes and these peripherals can be represented by a several devices in the CPU databases. For example, the device "TimerX_PPG" and "TimerX_PWM" represents TimerX in PPG and in PWM mode. These devices can be displayed on the CPU package, but they are also represented as a single block in the MCU block diagram.

Mouse Operations For Individual Items

- Single click on a component icon selects the component in the Project panel.
- Double click on a **component icon** opens its Component Inspector and selects the property specifying the peripheral used by the component.
- Double click on a **peripheral** opens the simple item structure view.
- Double click on an  icon opens a selection menu with all the components that use single parts of the peripheral. Selecting one component opens it in the Component Inspector.
- Right button click on a **component icon** opens the Component pop-up menu. If the Component Inspector is invoked from this pop-up menu, an appropriate property allocating the used peripheral is selected.
- Right button click on an  icon opens selection menu with all the components that use single parts of the peripheral. Selecting one component opens the Component pop-up menu.
- Right click on the peripheral opens the Peripheral Pop-up menu (see below).

Peripheral/Pin Pop-up Menu

The following commands 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 CPU package. See *2.13 Peripheral Initialization* for details.
- **Show Peripheral Structure** - opens the peripheral's structure view - (it is supported for I/O ports, timer's counters, serial ports. This option is also supported for devices working in several modes in the CPU block diagram. A list of represented devices for these modes is displayed.
- **Show Peripheral Usage** - shows which part of the peripheral is used by the application (visible after code generation). This option is supported for I/O ports and pins, timers, A/D converters and A/D channels, CAN, serial ports, watchdog, internal memories (EEPROM and FLASH). See *2.14 Peripherals Usage* for details.
- **Rename Peripheral** - allows you to rename the selected peripheral. It is supported for I/O ports and pins, watchdog and timers (counters, compare and capture registers, free running devices), A/D converters and A/D channels, CAN, serial ports.
- **Search Related Info In CPU PDF Documentation** displays PDF Search window and finds the information about the peripheral in the appropriate CPU documentation. It is also possible to search for any keyword in the CPU documentation based on the original manufacturer's CPU manual. (This item is available on the package and on the CPU block only.) See *2.16 PDF Search* for details.
- **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 CPU package.
- **Help on Target CPU Window** - displays help for the current window

2.8. CPU Timing Model

Project Panel > CPU pop-up menu > View CPU Timing Model

This window presents the schematic structure of the target CPU timing stored in the database of Processor Expert. The information can be displayed in two levels of complexity:

- **Simple view** - In this mode, only the active prescalers are shown. Unimportant tree nodes like inactive branches, divider by one etc... are hidden. Every clock source is shown as a node of the tree at the topmost level. *This is the default mode.*
- **Normal view** - All nodes of the timing model are shown.

The complexity can be switched by checking/un-checking the *Simple View* item of the pop-up menu of the window.

Click plus "+" or minus "-" signs to expand or collapse the branches of the tree.

Icons Description

- ✓ - active endpoint - a device using the clock.
- ✖ - inactive device or unused branch
-  - clock source
-  - adjustable prescaler
-  - constant prescaler
-  - adjustable multiplier
-  - constant multiplier
-  - clock distribution to multiple branches
-  - element selecting one of the connected branches. The inactive branches have the gray cross ( ) icons (for an inactive branch example please see the **RTIJoin** prescaler on the figure below).
-  - join of multiple clock sources

- The values of all prescalers are for high speed mode. For details on speed modes please see the chapter *3.2.2.2 Speed Modes Support*.

More details on individual items are available as hints after the mouse cursor is placed on the item's name. The displayed prescaler values are automatically set after the reset for enabled devices. The values are influenced by the components configuration. For details on the component timing principles please see the chapter *3.3.4 Timing Settings*.

Some nodes contain a frequency information. It is a clock frequency on the node output.

Clock Path Direction

It is possible to change the direction of the timing path. This can be set by switching the CLOCK field value:

- **From CPU clock source to individual peripherals**
  The prescalers and clock-driven CPU devices are ordered in a tree structure, starting with the main branch which represents the main clock source (PLL, X-tal etc...).

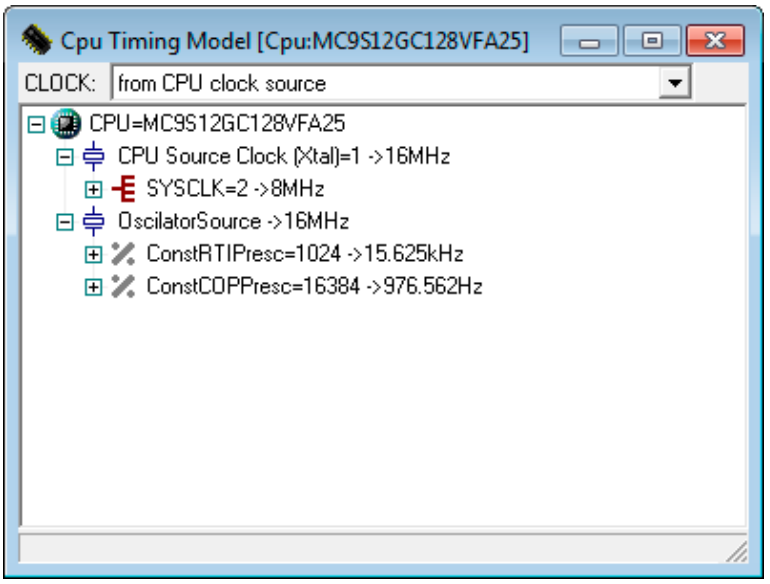


Figure 2.38 - CPU Timing Model Window

- **From selected peripheral to CPU clock source**
  The selected peripheral is the root of a tree showing the current sequence of the prescalers form the device to the main clock source. Each sub-node represents a source of the clock used for its parent.

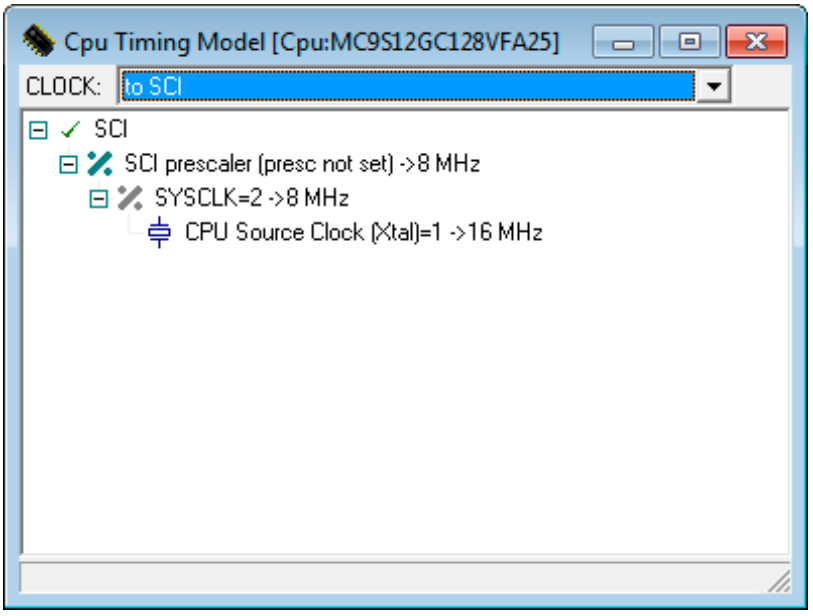


Figure 2.39 - CPU Timing Of Selected Peripheral

2.9. Resource Meter

Processor Expert > View > Resource Meter

The Resource Meter shows the current status of a chip resource usage (or availability).
Note that if a peripheral is allocated, all its parts are reserved. For example if you use the 8-bit I/O port, all the I/O pins of the port are allocated and it is not possible to use them in other components.

- **Pins usage** meter shows pins' usage. In general, there are always some pins used, such as the power supply pin.
- **Port usage** meter shows ports' usage. A port is considered allocated if a part of it is used, or if its pins are allocated to another device.
- **Compare/Reload** meter shows the allocation of the timer compare registers (depending on CPU type). If it is possible to combine several smaller registers into one large, then the allocation of one of the smaller ones means allocation of the large one and the allocation of the large one means allocation of the two smaller ones.
- **Capture regs** shows timer capture registers usage.
- **Communication** shows the allocation of the serial communication channels (including also CAN).
- **A/D channels** shows the allocation of the A/D converter channels.

By placing the mouse over a resource meter field, you may get a hint that provides details about which resources are used concerning this field.

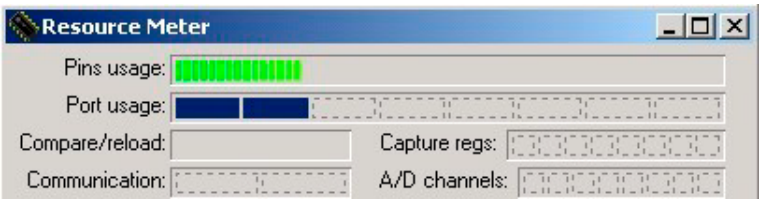


Figure 2.40 - Resource Meter Window

2.10. Memory Map Window

Processor Expert > View > Memory Map

This window shows the CPU address space and **internal** and **external memory** mapping. Detailed information for an individual memory area is provided as a hint when the user moves cursor over it.

Legend:

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. The sizes of individual memory areas blocks drawn in the window are different from the ratio of their real sizes to improve readability of the information (Small blocks are larger and large blocks are smaller).

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

Figure 2.41 - Sample Of Used Part Of The Memory Area

Mouse Actions

If you move the mouse pointer to any part of the CPU address space, a detailed description of the chosen part will be displayed in the hint.

Double click on the used (line-crossed) memory area will open the CPU component inspector window with the selected definition for this area (same as the Edit Usage pop-up menu command).

Pop-up Menu

- **Edit Usage** - opens a Component Inspector window that allows to customize the memory setup for the selected area. This command is available only for used areas (line-crossed).
- **Display Mode**
  - **Full** - displays all items.
  - **Only Memory** - displays only Flash, RAM and ROM.
- **Summary** - displays window with summary of memory usage (percentage and absolute view).
- **Help** - shows this page.

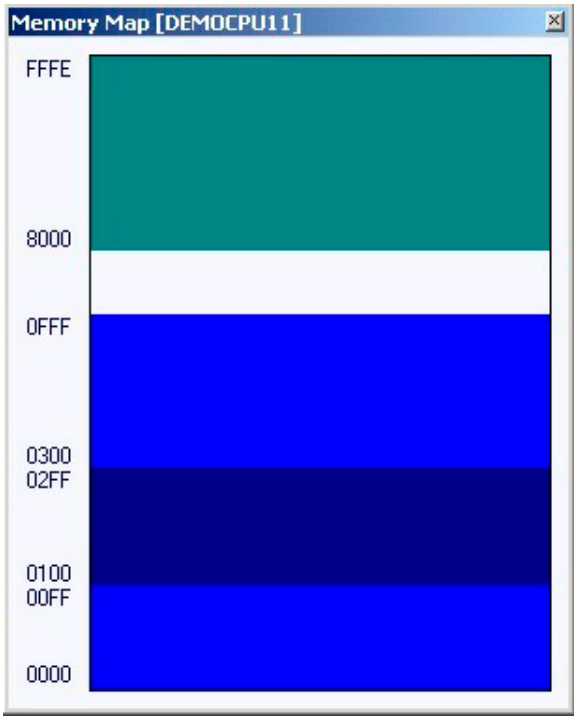


Figure 2.42 - Sample Memory Map Window

2.11. CPU Parameters Overview

Processor Expert > View > CPU Parameters Overview

You may get the technical features of a CPU by selecting the **CPU Parameters Overview** command of the View menu. The complete database then appears together with a query window. By specifying requirements on technical features, you may filter the database in order to display only relevant CPUs. If you press **OK**, you will get the list of CPUs that meet your requirements. If you press **All**, all the CPUs supported by your version of Processor Expert will be listed.

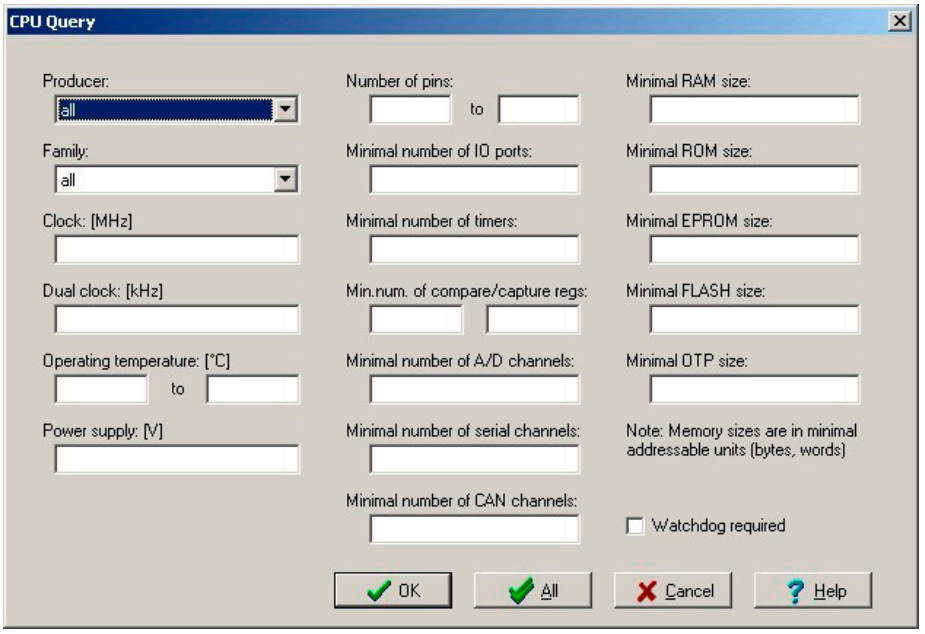


Figure 2.43 - CPU Query Dialog

Description

The **CPU Parameters Overview** windows displays the list of CPUs, including their technical features:

- *CPU type* - CPU type
- *Producer* - CPU producer
- *Family* - CPU family
- *Clock* - CPU xtal clock
- *Dual clock* - if CPU does have a dual clock
- *Operating Temperature* - CPU operating temperature
- *#pins* - number of pins on CPU package
- *#IO ports* - number of I/O ports and I/O pins
- *#timers* - number of timers, compares and captures
- *#A/D* - number of A/D channels and converters. For example 12/1 means 12 A/D channels and 1 A/D converter.
- *#serial* - number of asynchro/synchro serial channels
- *#CAN* - on-chip CAN channels
- *Watchdog* - on-chip watchdog
- *#Special* - Special features and devices
- *RAM* - on-chip RAM size
- *ROM* - on-chip ROM size
- *EPROM* - on-chip EPROM size
- *FLASH* - on-chip FLASH size
- *OTP* - on-chip OTP size
- *Power supply* - power supply voltage
- *Storage* - CPU storage temperature

Note: Memory sizes are in minimal addressable units (bytes, words).
If you right-click on the window, a menu appears allowing you to add the selected CPU to the current project or to refine your previous query.

2.12. List of Installed Components with Additional Information

Processor Expert > View > Installed Components Overview

This window contains information about **installed components** in the current version of Processor Expert:

- **COMPONENTS**
  All installed components including CPUs and user-created components are listed in the report. Placing the mouse cursor on the component's name will display the hint containing the detailed information about the latest component version.
  See **menu > View > Component Type** for component type selection.
- **Component Info**
  The amount of the information shown in this column can be controlled via **View | Component info** menu.

  Note: This information is also displayed as a hint on the component.

  - Component description
  - File format and access *(encrypted, full source, compressed)*
  - Author of the component
  - version
  - List of revisions of the component. Each revision contains a version number, date and author of the revision.
- **Drivers**
  Installed component's drivers.
- **Driver Info**
  The amount of the information shown in this column can be controlled via **View | Driver info** menu.

  Placing the mouse cursor on the file name will display the detailed information about the latest component driver version.

  - Status - file format and access *(encrypted, full source, compressed)*
  - Author of the component's driver
  - Current version of the component's driver
  - Init. date - date of creation
  - Last modification - date of last modification
  - List of revisions of the driver. Each revision contains a version number, date and author of the revision.

The contents of the window can be configured using commands in menu **View**. It is possible to view this window also in HTML format using command **View > As HTML**.

Menu

- **View**
  - **Component Type**
    - **all** - displays all components.
    - **Components only** - hides CPU components.
    - **CPUs only** - displays CPU components only.
  - **Component Info**
    - **all** - displays information about the component.
    - **basic** - displays basic information about the component only.
    - **none** - hides information about the component.
  - **Driver Info**
    - **all** - displays the most information about component's drivers.
    - **basic** - displays only a basic information about the component's driver.
    - **none** - hides information about component's drivers.
  - **Implementation**
    - **display all** - shows all components.
    - **Hide components without any supported driver** - shows only components that contain supported driver.
    - **Hide components with any supported driver** - shows only components that do not contain any supported driver.
    - **Show only full source components** - shows only components that do not have the source code encrypted.
  - **As HTML** - display this table in default HTML browser.
    Temporary HTML file will be removed from the disk after closing Processor Expert.
- **Help**
  - **Help on Component**- displays help for currently selected component (except CPU components).
  - **Help on this Window** - displays this page.

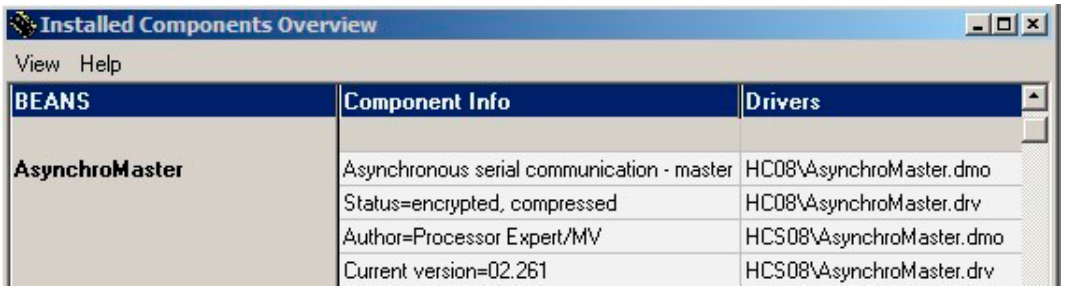


Figure 2.44 - List of Installed Components

2.13. Peripheral Initialization

Processor Expert > View > Peripheral Initialization

The Peripheral Initialization window shows overview of peripheral initialization settings for the current CPU. It displays initialization values of all *control, status and data* registers of selected peripheral including single bits. The user can also see peripheral schematic diagram. Peripheral Initialization can be invoked from the View menu or from Target CPU window or using a Peripheral Initialization button in the Component Inspector.

If the Peripheral Initialization window is docked with the Component Inspector window, the peripheral is automatically changed according to the peripheral selection in the Components library . Automatic changes of the peripheral can be disabled using the lock icon  .

Note: The Peripheral Initialization and Peripheral Usage are both only one window in two different modes. These windows could not be displayed both at once.
The initialization information reflects:

- CPU default settings - when the peripheral is not utilized by Embedded Component
- Embedded Component settings - when the peripheral is utilized by the Embedded Component and the component settings are correct. Peripheral Initialization Inspector shows initialization as required by Components settings.

Registers

There is a value displayed in the middle column which will be written into the register or bit by the generated code during the initialization process of the application. It is the last value that will be written by the initialization function to the register.

Note: For some registers, the value read from the register afterwards can be different than the last written value. For example, some interrupt flags are cleared by writing 1. Please see the CPU manual for details on registers behavior.

In case the peripheral is allocated by a component and the setting of the component is incorrect, then the initialization values are not displayed in the Peripheral Initialization window. Instead, a value of the register (or bit) after reset is displayed in the right column. The *after-reset values* can contain also a characters with special meaning. The list and description of these characters is displayed as a hint when the mouse cursor is placed on the header of the registers table.
The values of the registers can be displayed in hexadecimal, decimal or in binary form. In case the value of the register (or bit) is not defined, an interrogation mark "?" is displayed instead of the value. In this case it is possible to display the value of the register in binary form only.

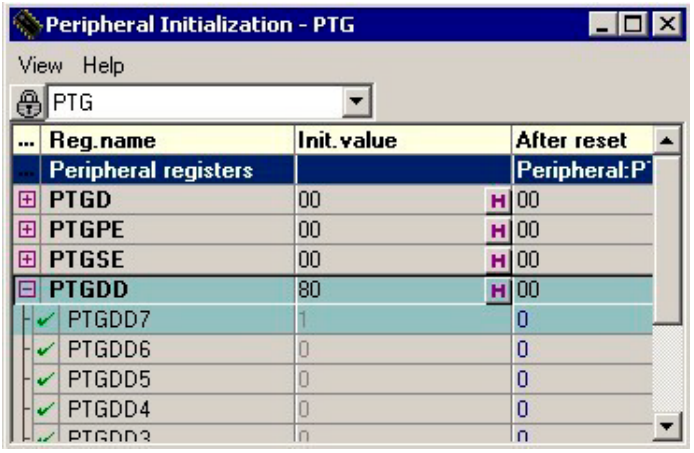
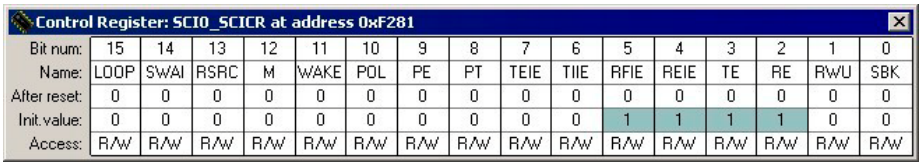


Figure 2.45 - Register List in the Peripheral Initialization window

Register details command from pop-up menu of a register (invoked by the right button click) opens a window containing a detailed information for the register.

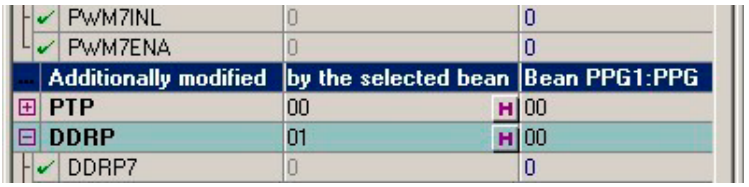


Bit num:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name:	LOOP	SWAI	RSRC	M	WAKE	POL	PE	PT	TEIE	TIIE	RFIE	REIE	TE	RE	RWU	SBK
After reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Init.value:	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
Access:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Figure 2.46 - Register Content Details

Additionally modified registers

A list of additionally modified registers is displayed at the bottom of the registers list. It contains registers that are influenced by the component but are not listed for the peripheral currently selected in this window. The list is displayed only if such registers exist.



Changes Highlighting

The user can watch reflections of user settings to Embedded Component Properties directly in CPU peripheral registers and bits. The registers influenced by a last component settings change are highlighted with a green color (see the previous two pictures for example). The highlighting works only in the case that the component was set-up correctly before the change was made and the new setup is correct as well and there is no error reported in a component settings. If there is an error in a component settings and no other component is influencing the register, the after-reset values are shown.

Menu

- **View**
  - **Sort Registers by Address** - If this item is checked, registers are sorted in list by their address. Otherwise they are sorted by name.
  - **Group Registers** - If this item is checked, groups of numbered registers with the same name are shown as expandable folders. The name of each folder is the same as the name of the registers with numbers replaced by 'xx'.
  - **Show Unused Bits** - This option enables/disables displaying the registers' bits unused by the manufacturer. When it is enabled, the unused bits are listed with the name 'unused'.
  - **Expand all** - expands the folder and all its subfolders.
  - **Collapse all** - collapses the folder and all its subfolders.
- **Help**
  - **Help** - displays documentation.

Pop-up menu for a register

- **Register Details** - opens a window containing a detailed information for the register.
  - **Search Register in PDF Documentation**

  - searches the register in the CPU documentation using PDF search. See *2.16 PDF Search* for details. This option is available only if PDF search is available and installed.

Peripheral schematic

Note: Peripheral schematics view is not available in this version.

It is possible to switch peripheral initialization window style between register list, peripheral schematic and both of them using **registers**, **schematic** and **both** buttons. *Schematic and both buttons are present only when the schematic for the current peripheral is provided.*

Schematic diagram of the peripheral contains names of the control registers. Highlighting of changes of a related component settings works also in the schematic mode.

- If the user places mouse cursor on the register name, the hint containing register description and address is shown.
- When the user clicks the register name with the left mouse button, new dialog window containing **detailed information** (including initialization value) for the register is shown. The register bits affected by a changes are highlighted.

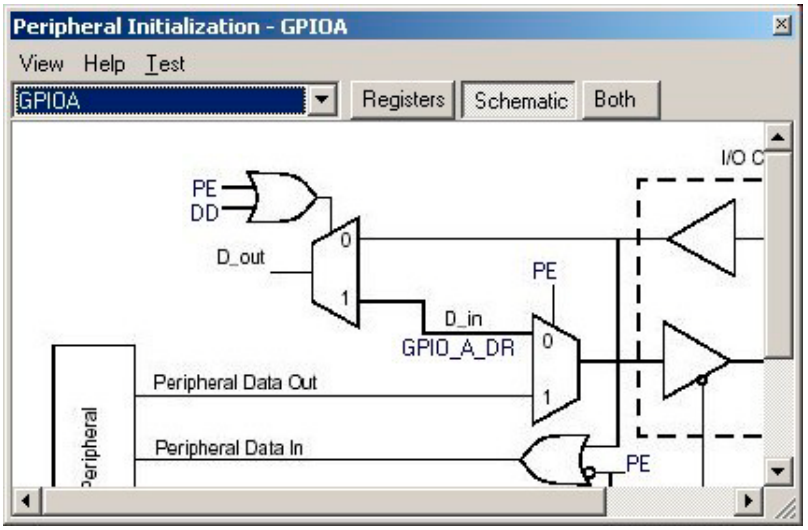


Figure 2.48 - Schematic Diagram of the Peripheral

2.14. Peripherals Usage

Processor Expert > View > Peripherals Usage

The Peripheral Usage window shows the current status of on-chip peripherals usage in detail.
The names of the components using peripherals appear in the third column, next to the corresponding peripheral names. When the mouse is placed over a peripheral name, a short description of the peripheral appears as a hint.
Note that if a peripheral is allocated, all its parts are reserved. For example if you use the 8-bit I/O port, all the I/O pins of the port are allocated and it is not possible to use them in other components.
The list can be filtered for only the used peripherals/interrupts/channels to be shown by using the menu command **View > Show Used Peripherals Only**.

Note: The Peripheral Initialization and Peripheral Usage are both only one window in two different modes. These windows could not be displayed both at once.
The following items are available:

- **I/O page** shows I/O pins' and ports' usage. If a pin/port is allocated to a component, the I/O properties of the component are displayed below the pin/port's name (unless all the pins of a port are used by one component, in which case, the properties appear under the name of the allocated pins).
- **Interrupts page** shows interrupt vectors usage. Each item of the list contains a number of the interrupt vector and its name. If an interrupt vector is allocated to a component, the component's name and the interrupt priorities are displayed as a sub-items in the vector's group.
- **Timers page** shows the allocation of the timers.
- **Channels page** shows A/D, CAN and serial channels usage.

Menu

- **View**
  - **Sort registers by address**, **Group registers**, **Show Unused Bits** - In this mode of the window these commands have no use and are not available. **Show Used Peripherals Only** - After this option is enabled, only the peripherals allocated by Processor Expert are shown.
  - **Expand all** - expands the folder and all its subfolders.
  - **Collapse all** - collapses the folder and all its subfolders.
- **Help**
  - **Help** - displays documentation.

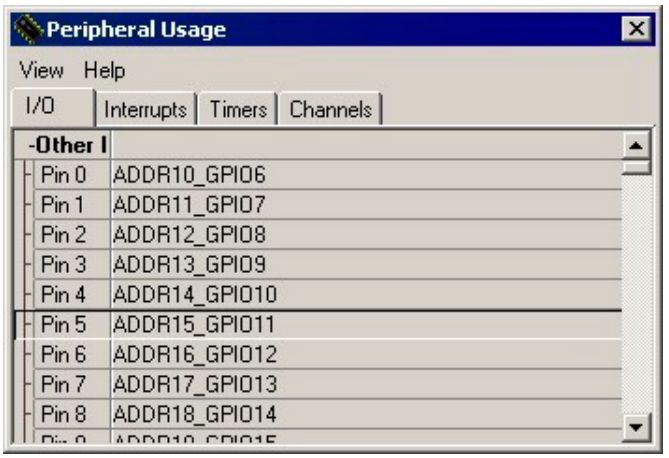


Figure 2.49 - Peripheral Usage Window

2.15. File Editor

About File Editor

File editor is the Processor Expert internal editor allowing to

- Edit files - All common text editor functions are available for comfortable work with the source code.
- View files - Editor is opened in a read only mode (it's shown in the title of the window).
- Compare files - A visual file-comparison mode with two panels showing a differences between two files.

Note: The Edit and View modes of the internal editor are not used in the CodeWarrior plugin. Internal editor is for these operations fully replaced by the native CodeWarrior source code editor.

Meaning of Buttons:

-  **Switches between module Extensions** - switch between assembler implementation of the driver and header file of the module or implementation (body file) of the module.
  Note: it is available only if it is allowed in the Editor Options.
-  **Open file** - opens a file.
-  **Save file** - saves the currently displayed file.
-  **Save all** - saves all opened files (if they were edited).
-  **Close file** - closes the currently displayed module (file).
-  **Print file** - prints the currently displayed file on a printer.
-  **Editor options** - editor settings.
-  **Help** - opens Help.
-  **Change font** - changes the font of File Editor.
-  **Undo** - restores the state of a file before the last change.
-  **Redo** - restores the last change.
-  **Find** - finds a string in the currently displayed file.
-  **Replace** - replaces a string with another string.

Mouse Actions

- To move editor cursor to a specific place, click the left button on the desired place in a text.
- To select a text, move mouse and hold the left button.
- To move the selected text to a different location, drag a selected text with left button pressed.
- To invoke editor pop-up menu, click the right button.
- To scroll editor view, move mouse with middle button pressed.
- To close an opened file, click the middle button on the tab with a file name (when multiple files are open).
- To select a square text block inside the window press and hold the ALT key and left mouse button while moving the mouse.
- To jump on a specified line number, click the first status-bar field containing a line number information.
- To place a method invocation to the source code, drag the method from the Project Panel. See *2.3 Project Panel* for details.

File Editor Pop-up Menu

To open the File Editor pop-up menu, click the right mouse button on the text area of the File Editor window.

Meanings of items:

- **File**

- **Open** - opens a file.
- **Save** - saves the currently displayed file.
- **Save As** - saves the currently displayed file under a new name.
- **Save All** - saves all opened files (if files were edited.
- **Close**- closes the currently displayed file.
- **Close all** - closes all files.
- **Delete** - closes and deletes the currently displayed file.
- **Reopen** - opens a list of used files.
- **Print** - prints the currently displayed file on a printer.
- **Print Preview ...** - opens a print preview dialog. The dialog allows the user to enter printer options, set zoom size for preview, display document margins and print the page.

- **Edit**
  - **Undo** - Restores the state of a file before the last change.
  - **Redo** - Resumes the last change.
  - **Cut** - Cuts the selected text to the clipboard.
  - **Copy** - Copies the selected text to the clipboard.
  - **Paste** - Pastes the text from the clipboard.
  - **Clear** - Clears the selected text.
  - **Select All** - Selects all text.
  - **Delete Line** - Removes the current line.
  - **Delete word** - Removes characters from the current cursor position to the end of the word (including space).

- **Search**
  - **Find** - Displays SearchReplace dialog
  - **Find Next** - Finds next position of a string in the currently displayed file.
  - **Replace** - Displays SearchReplace dialog
  - **Go to Line** - Moves the cursor to a line determined by its number.

- **Debug**

Note: Internal debugger is not available in this version.

- **Toggle break** - Set breakpoints on actual cursor position.
- **Add watch ...** - Add watch to watches list in Watches window.
- **Evaluate/Modify ...** - Open watch editor and enable to edit value of any variable.
- **Run** - Runs application in target system.

- **Go to cursor**- Runs application and it stops on actually selected row and breakpoint.
- **Stop** - Stops running application.
- **Reset** - Resets application in target system.
- **Step into** - Steps program including subprogram call.
- **Step over** - Steps program without subprogram call.
- **Step into in assembler** - Steps program including subprogram call. After one assembler instruction execution actual program position is displayed in window.
- **Step over in assembler** - Steps program without subprogram call. After one assembler instruction execution actual program position is displayed in window.
- **Go to Source Line** - Opens window and goes to actual program position.
- **Modules** - Shows modules list into ABS file.
- **Disassembler** - Opens Disassembler for the current module.
- **Disassembler Whole Code** - Opens Disassembler for the whole program memory.
- **Options**
  - **Editor Options...** - setting of the File Editor

  (see Editor Options below).
  - **Change Font...** - change font of File Editor.
  - **Disassembler Op-codes**

    These commands in this submenu are not supported in this version of Processor Expert.
- **Editor Toolbar Visible** - show or hide toolbar of the File Editor. The toolbar is visible as a part of the window or as a single panel. Clicking on the toolbar corner it is possible to place the toolbar to other side of the window or to any other place on the screen as a floating panel.
- **Help** - opens documentation.

File Editor Options

- **Preserve cursor position during paste** - keeps the current position of the cursor during a paste operation.
- **Use tab character** - The editor will write tabulator character after TAB key is pressed and these characters will not be replaced by spaces.
- **Show modules in separate tabs** - Displays tabs for each opened module (extension). When this option is not checked, it is possible to switch between module extensions by clicking the right mouse button on the tab of the file and choosing the appropriate extension or by clicking on the "Switch between module extensions" button in the toolbar menu.
- **Show line numbers** - Displays line numbers in the editor window.
- **Syntax highlighting** - Displays file content in specific colors with respect to file extension and compiler.
- **No horizontal scroll-bar** - Disables horizontal scroll-bar when line is not longer than window.
- **Outline current line** - Editor shows a frame around the current line.
- **Tab size** - number of spaces for tabulator.
- **Number of backup copies** - how many backup copies will be maintained for each saved document. The backup files are created within the same directory as the saved file. The name of file is the same, but there is '~' character added before the extension and a number of the backup copy is added at the end. The latest backup file has the number 0. The bigger the number is, the older is the backup.
- **Hint delay** - how long will an editor hint stay on the screen.

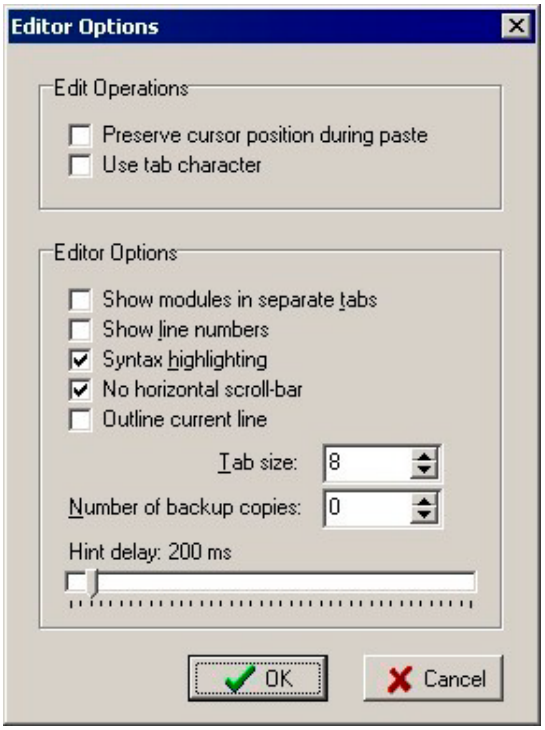


Figure 2.50 - Editor Options

Search And Replace Dialog

This dialog window is invoked by the pop-up menu commands **Search > Find...** and **Search > Replace...**. It allows to specify the subject of search (or replace) and mode of operation.

Figure 2.51 - Search And Replace Dialog

Input Fields

- **Find what** - searched text
- **Replace with** - a new text that will replace the searched text in replace function.

Options

- **Case sensitive** - When checked, the case of letter of the searched text has to match.
- **Whole words only** - The searched text is found only as a whole word.
- **Backward** - direction of the search.

- **Prompt on replace** - specifies if the user will be asked about each item replacement during the replace process. (This option is available only if any replacement text is entered).

Scope

- **Entire text** - Entire text will be scanned.
- **From cursor** - Text from cursor to the end of file will be scanned only.
- **Selected text** - The current selection will be scanned only.

Buttons

- **Find Next** - Invokes the search process. The cursor will be placed on the next occurrence of the searched text. If the find/replace operations have not been done yet, the first occurrence of the text is found.
- **Find All Files** - Invokes the search process within all opened files. Places cursor in each file on the last occurrence of the searched text within the file.
- **Replace** - A next occurrence of the searched text is searched and if found, it is replaced by the replacement text.
- **Replace All** - All occurrences of the text within the current file are replaced by the searched text.

Comparison Mode

File editor in this mode has two panels showing a differences between the compared files. The **different lines are highlighted** with the light-yellow color and the different characters are red. The lines added to the file have a green background. To speed-up navigation between changes, the editor offer the **arrow buttons on the toolbar**. Pressing the right/left arrow button will move the cursor to the next/previous difference in the file.
This mode can be invoked automatically by Processor Expert in a case of comparing a component modules with previously generated ones (See *2.3.4 Component Pop-up Menus* for details.) or by a 'DIFF' button when a changes tracking is enabled (See *3.5.1.1 Tracking Changes in Generated Code* for details.)

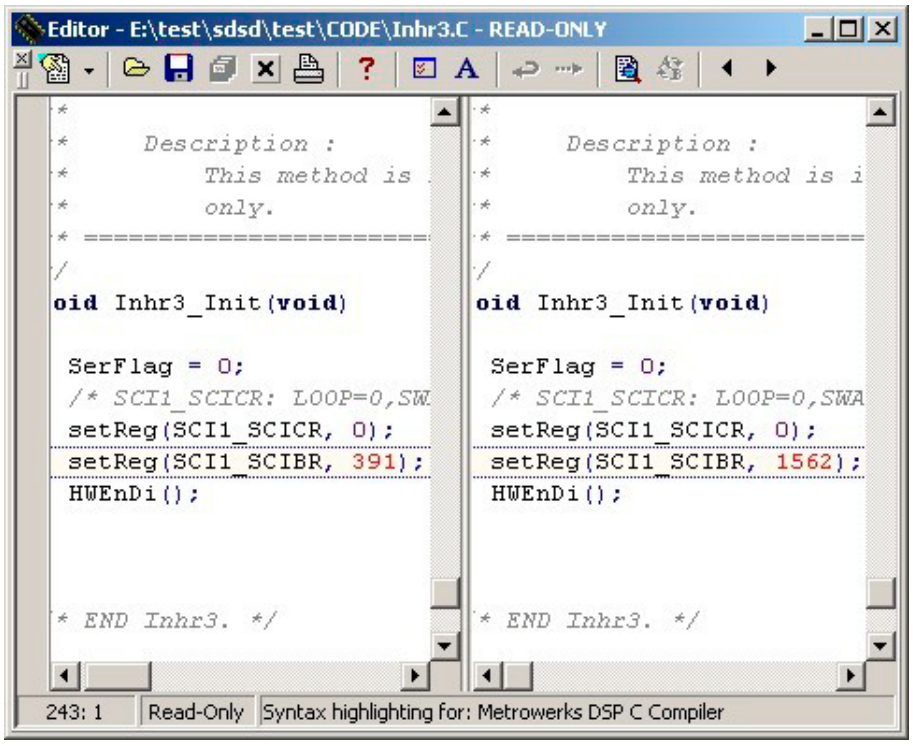


Figure 2.52 - The comparison mode of editor

2.16. PDF Search

Help > Processor Expert > Search in PDF Documentation of the Target CPU

The **PDF Search** window allows the user to quickly browse PDF documentation for an CPU. It can also be invoked directly from the pop-up menu of the CPU component.

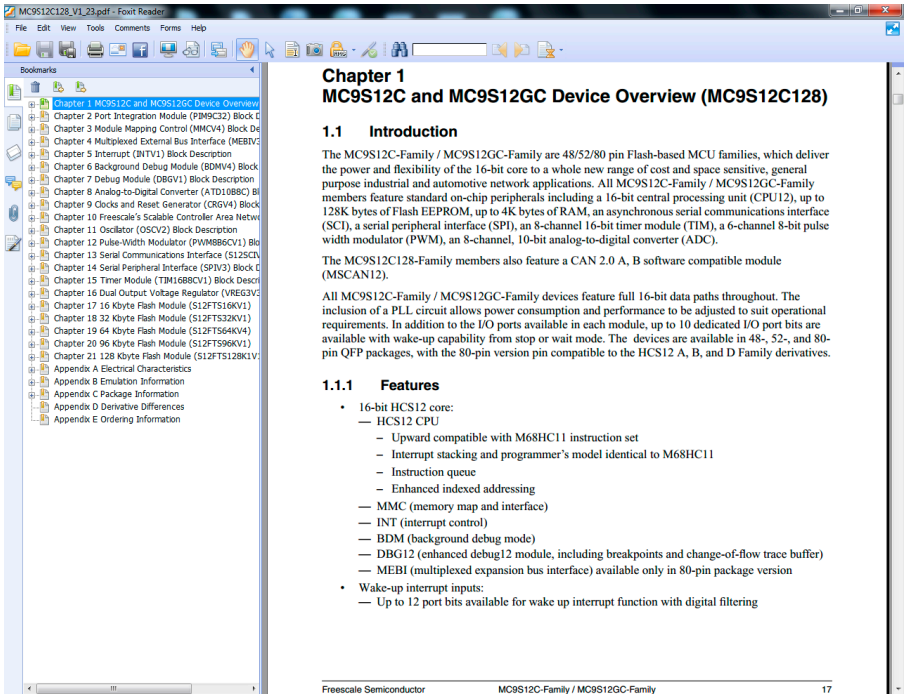
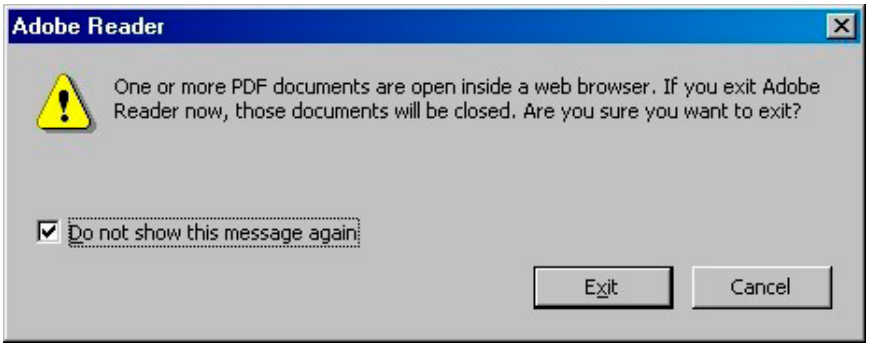


Figure 2.53 - PDF Search Window

Note for Acrobat Reader 6.0 users: *The Acrobat Reader can show the following dialog box while the PDF files are switched within the PDF search window:*



For a proper function of the PDF search the user should press the ***Cancel*** *button.*

Three Panels Of The Window

- The **top panel** allows the user to specify the searched text/expression and enable/disable additional switches influencing the search process.
- On the **left side** of the widow is a narrow panel containing as a result of search a list of the pages containing the required information. Clicking on numbers in this listing invokes the appropriate page in the Adobe Acrobat Reader panel.
  Unfortunately Acrobat Reader does not permit highlighting of any text results found on the page. You may find on the page using the Acrobat Reader "Find" function (see Acrobat Reader toolbar).
- The **Acrobat Reader (or Adobe Reader) panel** shows PDF file content. This panel has it's own toolbar which allows you to browse within the PDF file. This panel has also an internal Find function which is useful for locating searched phrase on the page. Using the toolbar you can print, move, and zoom in or out on the document. More documentation about controlling the Acrobat Reader plug-in can be found in Adobe Acrobat Reader documentation.

Additional Switches

- **Case sensitive** - If this switch is enabled, all letters of searched text must exactly match the text in the document including its case. If the switch is disabled, the case of the letters is ignored.
- **Regular expressions** - If this switch is enabled, the text entered as a searched phrase is treated as a regular expression. Users can use a power of regular expressions (For example it allows using a logical or set operators etc...). The syntax of regular expressions is a subset of commonly used Perl regular expressions. You can find more information on regular expressions and their syntax in chapter *2.16.1 Regular Expressions* .

Our search technology uses the **PdfToText** program included in XPDF package. This package must be installed in a subdirectory named XPDF. The PDF documents are being converted to the text form in the background and stored in your TEMP folder. This conversion is executed only once for each PDF file. The conversion may take a moment: please wait until it is finished.

It is possible to find more information about XPDF on www.foolabs.com.
The configuration for the execution of the program PDFTOTEXT.EXE must be included in the Processor Expert Tools.

The Acrobat Reader must be installed on your computer as well.
The PDF Search feature was tested with Acrobat Reader 6.0 and 7.0

Opening CPU Documentation in Default PDF Viewer

To use a default PDF viewer instead of PDF search window for viewing the CPU documentation, set the option **Environment Options | Use default PDF viewer**

2.16.1. Regular Expressions

Regular Expressions are a widely-used method of specifying patterns of text to search for. Special metacharacters allow the user to specify, for instance, that a particular string the user is looking for occurs at the beginning or end of a line or contains *n* recurrences of a certain character.

Simple Matches

Any single character matches itself, unless it is a metacharacter with a special meaning as described below.

A series of characters matches that series of characters in the target string, so the pattern "bluh" would match "bluh" in the target string.

You can cause characters that normally function as metacharacters or escape sequences to be interpreted literally by 'escaping' them by preceding them with a backslash "\", for instance: metacharacter "^" matches the beginning of a string, but "\^" match character "^", "\\" match "\" and so on.

```
Examples:
  foobar          matches string 'foobar
'
  \^FooBarPtr     matches '^FooBarPtr'
```

Escape Sequences

Characters may be specified using an escape sequence syntax much like that used in C and Perl: "\n" matches a newline, "\t" a tab, etc. More generally, \xnn, where nn is a string of hexadecimal digits, matches the character whose ASCII value is nn. If You need wide (Unicode) character code, You can use '\x{nnnn}', where 'nnnn' - one or more hexadecimal digits.

```
  \xnn     char with hex code nn
  \x{nnnn} char with hex code nnnn
  (one byte for plain text and two bytes for Unicode
)
  \t       tab (HT/TAB), same as \x09
  \n       newline (NL), same as \x0a
  \r       car.return (CR), same as \x0d
  \f       form feed (FF), same as \x0c
  \a       alarm (bell) (BEL), same as \x07
```

```
  Examples:

  foo\x20bar   matches 'foo bar'
  (note space in the middle)
  \tfoobar     matches 'foobar' predefined by ta
```

Character Classes

You can specify a character class, by enclosing a list of characters in straight brackets [], which will match any one character from the list.

If the first character after the "[" is "^", the class matches any character not in the list.

```
Examples:
  foob[aeiou]r   finds strings 'foobar', 'foober' etc
.

  foob[^aeiou]r  find strings 'foobbr', 'foobcr' etc
.
  but not 'foobar', 'foober' etc.
```

Within a list, the "-" character is used to specify a range, so that a-z represents all characters between "a" and "z", inclusive.

If you want "-" itself to be a member of a class, put it at the start or end of the list, or escape it with a backslash. If you want ']' you may place it at the start of list or escape it with a backslash.

```
Examples:
  [-az]      matches 'a', 'z' and '-

  [az-]      matches 'a', 'z' and '-'
  [a\-z]     matches 'a', 'z' and '-'
  [a-
z]      matches all twenty six small characters from 'a' to 'z'
  [\n-\x0D]  matches any of #10,#11,#12,#13.
  [\d-t]     matches any digit, '-' or 't'.
  []-a]      matches any char from ']'..'a'.
```

Metacharacters

Metacharacters are special characters which are the essence of Regular Expressions. There are different types of metacharacters, as described below.

Metacharacters - line separators

^ start of line
$ end of line
. any character in line

Examples:

^foobar matches string 'foobar' only if it's
at the beginning of the line

foobar$ matches string 'foobar' only if it's
at the end of the line

^foobar$ matches string 'foobar' only if it's the only string in the line

foob.r matches strings like 'foobar', 'foobbr', 'foob1r' and so on

Metacharacters - predefined classes

\w an alphanumeric character (including "_")
\W a nonalphanumeric
\d a numeric character
\D a non-numeric
\s any space (same as [\t\n\r\f])
\S a non space

You may use \w, \d and \s within custom character classes.

Examples:

foob\dr matches strings like 'foob1r', "foob6r' and so on but not 'foobar', 'foobbr' and so on

foob[\w\s]r matches strings like 'foobar', 'foob r', 'foobbr' and so on but not 'foob1r', 'foob=r' and so on

Metacharacters - word boundaries

\b Match a word boundary
\B Match a non-(word boundary)

A word boundary (\b) is a spot between two characters that has a \w on one side of it and a \W on the other side of it (in either order), counting the imaginary characters off the beginning and end of the string as matching a \W.

Metacharacters - Iterators

Any item of a regular expression may be followed by another type of metacharacters - iterators. Using this metacharacters you can specify a number of occurrences of a previous character, metacharacter or subexpression.

```
  *      zero or more, similar to {0,
}
  +      one or more, similar to {1,}

  {n}    exactly n times
  {n,}   at least n times
  {n,m}  at least n but not more than m time
```

So, digits in curly brackets of the form {n,m}, specify the minimum number of times to match the item n and the maximum m. The form {n} is equivalent to {n,n} and matches exactly n times. The form {n,} matches n or more times. There is no limit to the size of n or m, but large numbers will chew up more memory and slow down regular expressions execution.

If a curly bracket occurs in any other context, it is treated as a regular character.

```
Examples:
  foob.*r     matches strings like 'foobar'
,

  foob.+r     matches strings like 'foobar'
,

  foob.?r     matches strings like 'foobar', 'foobbr
'

  fooba{2}r   matches the string 'foobaar

  fooba{2,}r  matches strings like 'foobaar', 'foobaaar'
,

  fooba{2,3}r matches strings like 'foobaar', or 'foobaaar
'
```

Metacharacters - Alternatives

You can specify a series of alternatives for a pattern using "|" to separate them , so that fee|fie|foe will match any of "fee", "fie", or "foe" in the target string (as would f(e|i|o)e). The first alternative includes everything from the last pattern delimiter ("(", "[", or the beginning of the pattern) up to the first "|", and the last alternative contains everything from the last "|" to the next pattern delimiter. For this reason, it's a common practice to include alternatives in parentheses to minimize confusion about where they start and end.

Alternatives are tried from left to right, so the first alternative found for which the entire expression matches, is the one that is chosen. This means that alternatives are not necessarily greedy. For example: when matching foo|foot against "barefoot", only the "foo" part will match, as that is the first alternative tried, and it successfully matches the target string. (This might not seem important, but it is important when you are capturing matched text using parentheses.)

Also remember that "|" is interpreted as a literal within square brackets, so if you write [fee|fie|foe] You're really only matching [feio|].

```
Examples:
  foo(bar|foo)  matches strings 'foobar' or 'foofoo'
.
```

Metacharacters - Subexpressions

The bracketing construct (...) may also be used for defining regular expression subexpressions. Subexpressions are numbered based on the left to right order of their opening parenthesis. The first subexpression has the number '1'.

```
Examples:
(foobar){8,10}  matches strings which contai
n

foob([0-
9]|a+)r matches 'foob0r', 'foob1r' ,
'foobar', 'foobaar', 'foobaar' etc.
```

Metacharacters - Backreferences

```
Metacharacters \1 through \9 are interpreted as backreferences
.

Examples:

  (.)\1+         matches 'aaaa' and 'cc'.
  (.+)\1+        also match 'abab' and '123123'
  (['"]?)(\d+)\1 matches '"13" (in double quotes), or '4
'
```

3. Application Design

This chapter will hep you to design application using Processor Expert and Embedded Components. You will find here recommendations and solutions to write and optimize a code effectively. If you are a beginner, please see the section Quick start that shows how to generate the code of your first project.

The following subchapters explain:

3.1. Quick Start in Processor Expert

Step 1 - Open an example

You can start learning Processor Expert by opening one of the available examples. All Processor Expert examples are accessible from the CodeWarrior IDE.

To open an example select the command **CodeWarrior main menu > File > Open...** and find the directory ***{CodeWarrior}*\ProcessorExpert\projects\HCS12\Demo.Tutorial** (where the {CodeWarrior} is the path where the CodeWarrior is installed into) and select the file **LED.mcp**.

Step 2 - Code generation

After opening an example, you need to invoke the code generation of the project to obtain all sources. Select command **CodeWarrior Main Menu > Processor Expert > Generate Code 'Project Name'**. After the code generation, the component source modules will be inserted into the ***Generated Code*** folder in the CodeWarrior project window and the event and main source modules will be inserted into the ***User Modules*** folder in the CodeWarrior project window. Generated source code can be displayed in editor by the left mouse button double-click on selected module in the project window.

Step 3 - More Complicated Example

Once you have learned the basic skills, you can open a more complicated example in order to get to more advanced level of code generation.

Creating New Projects

See the chapter *4 Processor Expert Tutorials* for step-by-step tutorials on creating Processor Expert projects from the beginning.

3.2. Basic Principles

The application created in Processor Expert is built from the building blocks called Embedded Components. The following sub-chapters describe the features of the Embedded Components and the CPU components that are special type of Embedded Components and what they offer to the user.

- Embedded Components
- CPU Components

3.2.1. Embedded Components

Embedded components encapsulate the initialization and functionality of embedded systems basic elements, such as CPU core, CPU on-chip peripherals (for details on categories of components delivered with Processor Expert see chapter *3.2.1.1 Component Categories*), FPGAs, standalone peripherals, virtual devices, and pure software algorithms.
These facilities are interfaced to the user through properties, methods and events. It is very similar to objects in the Object Oriented Programming (OOP) concept.

Easy Initialization

A user can initialize components by setting their initialization properties in the Component Inspector. Processor Expert generates the initialization code for the peripherals according to the properties of the appropriate components. User can decide whether the component will be initialized automatically at startup or manually by calling the component's Init method.

Easy On-chip Peripherals Management

Processor Expert knows exactly the relation between the allocated peripherals and the selected components. When the user selects a peripheral in the component properties, Processor Expert proposes all the possible candidates but signals which peripherals are allocated already (with the icon of the component allocating the peripheral). PE also signalizes peripherals that are not compatible with the current component settings (with a red exclamation mark). In the 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 RTIshared component for sharing periodic interrupt

from timers.

Methods

Methods are interfacing component functionality to user's code. All enabled methods are generated into appropriate component modules during the code generation process. All Methods of each component inserted into the project are visible as a subtree of the components in the Project panel.
You can use in your code all enabled methods. The easiest way to call any method from your code is to drag and drop the method from project panel to the editor. The complexity and number of methods depend on the component's level of abstraction.

Events

Some components allow handling the hardware or software events related to the component. The user can specify the name on function invoked in the case of event occurrence. They are usually invoked from the internal interrupt service routines generated by Processor Expert. If the enabled event handling routine is not already present in the event module then the header and implementation files are updated and an "empty" function (without any code) is inserted. The user can write event handling code into this procedure and this code will not be changed during the next code generation.
All Methods and Events of each component inserted into the project are visible as a subtree of components in the Project panel.

Interrupt Subroutines

Some components, especially the Low-level components and the Peripheral Initialization components (please see more details in chapter *3.2.1.1 Component Categories*) allow to assign an interrupt service routine (ISR) name to a specific interrupt vector setup.
The name of the Interrupt service is generated directly to the interrupt vector table and the user has to do all necessary control registers handling within the user code. See *3.5.3.1 Typical Usage of Peripheral Initialization Components* for details.
ISRs items are listed in the subtree of a component in the Project panel.

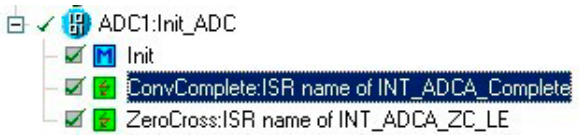


Figure 3.1 - Example Of a Component With Two ISRs

Highly Configurable and Extensible Library

Embedded Components can be created and edited manually or with the help of Component Wizard. CPU Components are a special category of components.

More information about Embedded components can be found in Processor Expert:

Help > Processor Expert > Embedded Components
Help > Processor Expert > Supported CPUs

3.2.1.1. Component Categories

Components Library Categories

Complete list of the component categories and corresponding components can be found in the Component Categories page of the Components Library

The components are categorized based on their functionality, so you can find an appropriate component for a desired function in the appropriate category. There are the following four main categories, which further contain various sub-categories.

- **CPU** - All available CPU components. The CPU folder in Components Library contains subfolders for the CPU families.
- **CPU External Devices** - Components for devices externally controlled to the CPU. For example sensors, memories, displays or EVM equipment.
- **CPU Internal Peripherals** - Components using any of on-chip peripherals offered by the CPU. The Components Library folder with the same name contains sub-folders for the specific groups of functionality. (i.e. Converters, Timers, PortIO etc.)

  Note: It seems that components (especially in this category) correspond to on-chip peripherals. Even this declaration is close to the true, the main purpose of the component is providing the same interface and functionality for all supported CPU derivatives. This portability is the reason why the component interface often doesn't copy all features of the specific peripheral.

- **SW** - Components encapsulating a pure software algorithms or inheriting a hardware-dependent components for accessing peripherals. These components (along with components created by the user) can be found in a components library in the folder 'SW'.

Specific functionality of the CPU derivative may be supported as a **version-specific** settings of the component. For more information about this feature please refer to *Version specific* parts in the component documentation or Components Implementation help chapter.

Levels of Abstraction

Note: LDD beans are not available in this version.

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

- **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 the Processor Expert. This basic set contains for example components for simple I/O operations (BitIO, BitsIO, ByteIO, ...), timers (EventCounter, TimerInt, FreeCntr, TimerOut, PWM, PPG, Capture, WatchDog,...), communication (AsynchroSerial, SynchroMaster, SynchroSlave, AsynchroMaster, AsynchroSlave, IIC), ADC, internal memories.
  This group of components allows comfortable settings of a desired functionality such as time in ms or frequency in Hz without user knowing about the details of the hardware registers. CPU specific features are supported only as CPU specific settings or methods and are not portable. See *2.5.3.4 Version Specific Items* for details.
  The components inheriting or sharing a high-level component(s) to access hardware are also high-level components.
- **Low Level Components** - Components that are dependent on the peripheral structure to allow the user 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 CPU 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. Usually contain only "Init" method, see *3.5.3.1 Typical Usage of Peripheral Initialization Components* for further details). The rest of the function has to be implemented using a low level access to the peripheral. This kind of components could be found in the " **CPU Internal Peripherals / Peripheral Initialization Components** " folder of the Components library and they are available only for some CPU families. The interface of these components might be different for a different CPU. The name of these components starts with the prefix 'Init_'.

Features of Components at Different Level of Abstraction

Feature	High level	Low level	Peripheral Init
High-level settings portable between different CPU families	yes	partially	no
Portable method interface for all CPU families	yes	partially (usually direct access to control registers)	*Init* method only
CPU specific peripheral features support	partially	mostly yes	full
Low-level peripheral initialization settings	no	partially	yes
Speed mode independent timing	yes	mostly yes	no
Events support	yes	yes	no (direct interrupt handling)
Software emulation of a component function (if the specific hardware is not present)	yes	no	no

3.2.2. CPU Components

Processor Expert > Help > Supported CPUs, Compilers and Debuggers

A CPU component is an Embedded Component encapsulating one CPU type. Like all other components, CPU Components also have properties, methods and events. A Processor Expert project may contain several CPU components. The project generated for one CPU is called an Application. CPUs included in a project are displayed in the upper part of the Project panel (depending on the Project panel settings). Before starting the code generation, one of the CPU components must be active (selected as the **Target CPU**).

The **Build options** accessible in the Component Inspector of the CPU component allow the user to set properties of the **Compiler** and **Debugger** (if it is supported).

In the CPU Component Inspector (page **Used**), it is also possible to select the peripherals that should not be used by Processor Expert. These peripherals are then not available to the components in the project and can be freely used by any external module.

Portability

- It is possible to change the target CPU during the development of an application and even to switch between multiple CPUs. This can be done simply by adding another CPU to the project and selecting it as the target CPU.
- To connect the new CPU peripherals to the application components correctly, it is possible to specify the CPU on-chip peripheral names. See *3.2.2.3 Changing Names of Peripheral Devices* for details.*This way the same peripheral could be used on different CPU derivatives even if the original name is different.*
- Specific application options for single targets are set in Project Options

Note: CPU peripherals names and Application Options are in the same pop-up menu as CPU inspector. To open the pop-up menu, right-click the CPU icon in the Project panel.

Adding a CPU to a Project.

1. In the Components Library window, select the **CPU** category and find the desired CPU component.
2. Double-click the desired CPU icon to add it to the project. When the CPU component is added, it appears in the upper part of the Project panel. If selected as the target CPU, the processor will be displayed in the Target CPU window.

Selecting a CPU as Target CPU

The first CPU added to the project is automatically selected as the **target CPU**. It means that code will be generated for this CPU. When there are more than one CPU in the project, the target CPU can be changed by following these steps:

1. Right-click the CPU icon in the Project panel to display a pop-up menu.
2. Select **Select CPU as target** - the CPU is selected as target.

This setting doesn't affect the setting of the target in the **Targets** tab in the CodeWarrior project panel. If the user changes the target CPU in the Processor Expert project panel and the CPU doesn't match with the current CodeWarrior target settings, the **Linker dialog box** is invoked during the code generation allowing the user to update the linker setup.

Changing CPU Settings

The only way to modify CPU settings (its properties, methods, events, chip selects, timing, user-reserved peripherals, compiler and debugger settings etc.) is to invoke the Component Inspector for the selected CPU component.

If you have added CPU to your project, you can invoke Component Inspector by performing either of the following:

- Right click the CPU icon in the Project panel to display pop-up menu and select the **CPU inspector**.
- Double-click the CPU icon in the Project panel.

For a detailed description of the current CPU properties, methods and events, select **Help > Help on Component** menu item in the Component Inspector window.

3.2.2.1. CPU Properties Overview

CPU Properties can be set in CPU Component Inspector. The complete list of CPU properties and their description is available in the help page for the CPU. To open the CPU help page, select **Help > Help on Component** from the menu bar in the Component Inspector window.

Following properties define the basic settings of the CPU:

- CPU type
- External Xtal frequency (and sub-clock xtal frequency)
- PLL settings
- Initialization interrupt priority
- External bus and signals
- Speed modes (See the following chapter Speed Modes).
- All other functions that are not directly encapsulated by components

3.2.2.2. Speed Modes Support

The CPU component supports up to three different **speed modes**. The three **speed modes** is a Processor Expert specific concept which (among all the other PE features and concepts) ensures the portability of the PE projects between different CPU models.

In fact, the three speed modes are a generalization of all the possible CPU clock speed modes used for power-saving that can be found in most of the modern microcontrollers. In the area of embedded systems, power saving and power management functions are so important that we could not neglect the proper HW- independent software implementation of these functions.
Therefore, for keeping the portability (HW independence) of PE projects, we recommend not to program the CPU speed functions manually, but use these three CPU Component speed modes instead:

- **High speed mode** - this mode is selected after reset and must be enabled in the project. This speed mode must be the fastest mode of the main CPU clock.
- **Low speed mode** - this mode is usually used for another PLL or main prescaler settings of the main CPU clock.
- **Slow speed mode** - this mode is usually used for the slowest possible mode of the main CPU clock.

Switching Speed Modes at Runtime

The modes can be switched in the runtime by the following **CPU component methods**:

- SetHighSpeed
- SetLowSpeed
- SetSlowSpeed

If a speed mode is enabled in the CPU Component properties, the corresponding method is enabled automatically.

It is highly recommended to disable interrupts before switching to another speed mode and enable them afterwards.

Speed Modes Support in Components

Using the component property *CPU clock/speed selection*, it is possible to define the speed modes supported by the component.
Some components allow to set two values of the *CPU clock/speed selection* property:

- *Supported* - The *CPU clock/speed selection* group contains properties defining which speed modes are supported for the component.
- *Ignored* - The speed mode settings are ignored and there are no action is performed when a speed mode is changed, that is the peripheral continues to run with the same configuration for all speed modes. No speed mode handling code is generated for this component. The component timing values are valid only for high-speed mode.

The following features are available for high-level components, if the *CPU clock/speed selection* is not set to *ignored*:

- During the design, all the timing-related settings for such a component are checked to be correct in all the speed modes that the component supports and the component is enabled in these modes.
- If the speed mode is changed, the current timing components are preserved (recalculated to be the same in the new speed mode), except the timing that is set at runtime from interval.
  If the CPU speed mode is changed to a mode that the component does not support for any reason, the component is disabled right after the CPU speed mode is changed. Otherwise the component is enabled.
- Before or after the speed mode is changed, the *BeforeNewSpeed and AfterNewSpeed* event functions are called.

3.2.2.3. Changing Names of Peripheral Devices

Purpose of name changes

Many components are using on-chip peripherals. The link between a component and a peripheral is done by the peripheral name that is selected by the user in the appropriate component's property. The user can change the names of peripherals for each target CPU within the project.

Possible reasons for changing names:

- Customized peripheral names may better express the reality of the CPU environment or company conventions may be different from Processor Expert naming.
- When working with several MCUs (having different peripheral names), one may assign a universal name in order to simplify the CPU changes in the project.
  Example: **Timer0** of CPU1 and **TmrA** of CPU2 have the same function in the project. They are both associated with **Component1:TimerOut**. Every time you change the Target CPU from CPU1 to CPU2 and vice versa, you need to modify in the Component Inspector the name of the peripheral linked to the **TimerOut** component. To avoid this, you can rename Timer0 and TmrA to one common name - **MyTimer** - and link the component (Component1) to the **MyTimer** peripheral. Now, when CPU1 will be set active (selected as target CPU), Component1 will be linked to Timer0, and when CPU2 will be set active, Component1 will be linked to TmrA.

Automated Peripherals' Names Change

*Note: To activate this feature, enable the PE environment option **Project Options | Ask to rename peripheral**. Please note that this feature is intended for experienced users and inappropriate usage may lead to the confusion in peripheral names.*

When the target CPU is switched to another one, the peripheral names often do not match. PE offers to change the peripheral names that are not found on the newly selected CPU to the name that has been selected when the previous CPU was active.
If there is only one peripheral of some type, so it's clear what should be selected, PE offers to rename it automatically when the target CPU is switched. If it's not clear which peripheral (or pin) should be selected, the names from the previous CPU stay filled within the properties, but PE reports error for that properties and the user has to select the peripherals manually. After selecting the peripheral, PE offers to rename the selected peripheral to the name from previous CPU.

CPU Peripherals' Names Editor

The editor window lists all **peripherals' names** . Initially, on-chip peripheral devices (pins, ports, timers, converters, etc.) have default names based on the reference manual.

In order to open the CPU Peripherals Names Editor, follow these steps:

- Move the mouse pointer to the selected CPU icon on the Project panel.
- Press the right mouse button - a popup menu will appear.
- Click on "**CPU peripherals names**".

Alternatively, the peripheral names editor can be invoked using the pop-up menu of the peripheral inside the Target CPU window. See *2.7 Target CPU Window* for details.

To change the peripherals' names follow these steps:

- Select a peripheral type in order to display all the CPU peripherals of this type.
- In the Name edit box, insert a new name for the selected CPU peripheral.
  Note: If you want to get back the default Processor Expert name, press the left-arrow button.
- Press the OK button.
- Answer the confirmation request of the dialog box.

The **Restore default** button allows to restore names for all peripherals.

The peripherals can also be renamed using the target CPU pop-up menu command. See *2.7 Target CPU Window* for details.

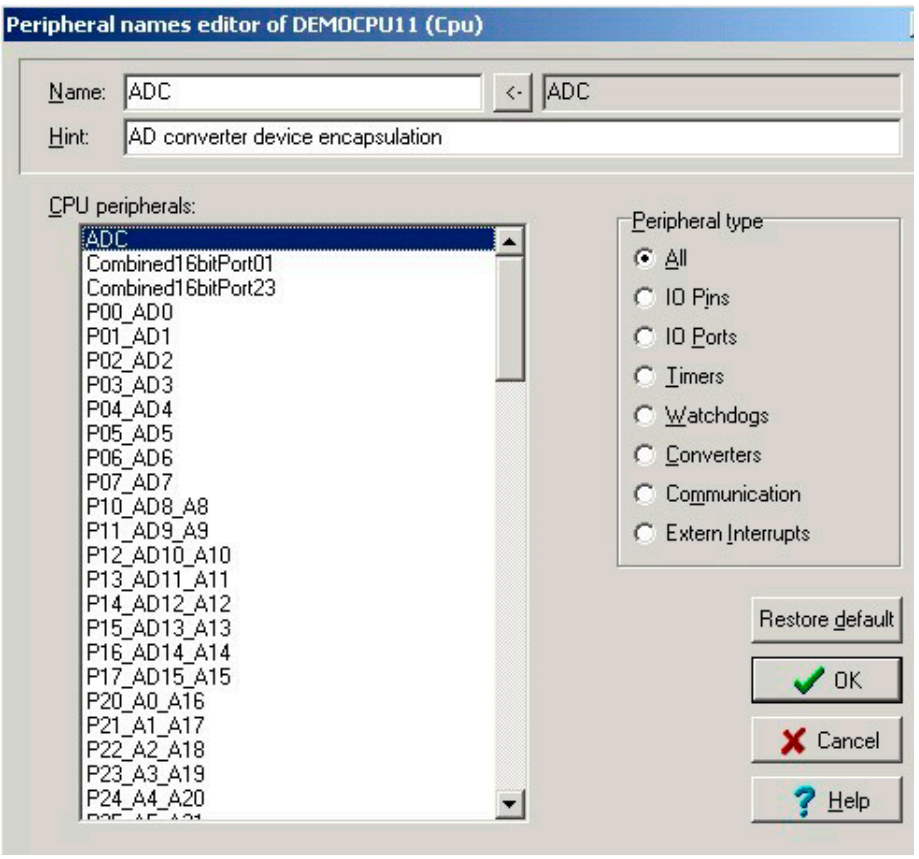


Figure 3.2 - CPU Peripheral Names Editor

3.3. Configuring 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. For the description of the user interface of the components settings please see the chapters *2.3 Project Panel* and *2.5.3 Component Inspector*.

The following sub-chapters provide hints and information about how to configure the Embedded Components used in the project correctly and effectively.

- Interrupts and Events
- Configurations
- Design Time Checking: Consequences and Benefits
- Timing Settings
- Creating User Component Templates
- Signal Names
- Component Inheritance and Component Sharing
- Pin Sharing

3.3.1. Interrupts and Events

This chapter describes the details of interrupt and events processing in the code generated by Processor Expert.

An **Interrupt** is a signal that causes the CPU stop the execution of the code and execute the Interrupt service routine instead. When the execution of the code is suspended, the current state of the CPU core is saved on the stack. After the execution finishes, the previous state of the CPU core is restored from the stack and the suspended program continues from the point where it was interrupted. The signals causing interrupts can be hardware events or software commands. For more details, please see an appropriate CPU manual.
Each interrupt can have an assigned **Interrupt Service Routine** (ISR) that is called when the interrupt occurs. The table assigning the subroutines to interrupts is called **Interrupt Vector Table** and it is completely generated by Processor Expert. See *3.3.1.1 Interrupt Vector Table* for details. Most of the interrupts have corresponding Processor Expert Events that allow handling of these interrupts. Processor Expert also allows to configure interrupt priorities, if they are supported by the CPU. See *3.3.1.2 Processor Expert Priority System* for details.

Processor Expert **Events** are part of the Embedded component interface and encapsulate the hardware or software events within the system. Events are offered by the High and Low Level components to help the user to service the events without any knowledge of the platform specific code required for such service.

Processor Expert **Events** can be enabled and disabled and have a user-written program subroutines that are invoked when the event occurs. Events often correspond to interrupts and are for that case invoked from the generated ISR. Moreover, the event can also be a pure software event caused by a buffer overflow or improper method parameter.

Interrupts Usage in Component's Generated Code

Some high-level components use interrupt service routines to provide their functionality. Usage of interrupts can usually be enabled/disabled through property "Interrupt service/event". If the interrupt service is used, complete interrupt service routine is generated into component's driver and the generated code contains configuration of the corresponding peripheral to invoke the interrupts.

The user should be careful while disabling an interrupt. If a component should operate properly, it is necessary to allow the interrupt invocation by having interrupts enabled and (if the CPU contains priority system) set the corresponding interrupt priority level. This can be done using the appropriate method of the CPU component.

Note: It is a common "bug" in user code, if the application is a waiting for a result of the component action while the interrupts are disabled. In this situation, the result of the component method does not change until the interrupt service routine is handled. See description of the property "Interrupt service/event" for detailed information about the particular component.

Enabling Event

Functionality of each event can be enabled or disabled. The user can easily enable the event and define its name within the Component Inspector of the appropriate component. Another possibility is to double-click an event icon in the component's subtree or use a pop-up menu in the Project Panel window.

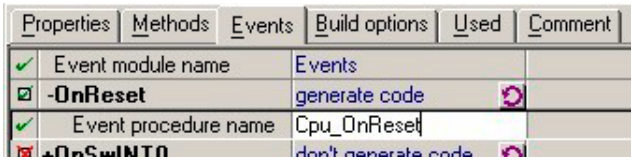


Figure 3.3 - Event Example in the Component Inspector Events Tab

Writing an Event Handler

Event handler is a subroutine that is assigned to a specific event. After the event is enabled Processor Expert generates the empty function of the specified name to the Event module. See *3.5.1 Code Generation* for details. The user can directly open the Event handler code (if it already exists) using a **Component pop-up menu > View/Edit event module** or a double-click on the event within the Project Panel. The event handler is an ordinary function and the user needs not to provide the interrupt handling specific code in his/her event code.

Interrupt Service Routines

When **High or Low-level components** are used, the interrupts functionality is covered by the events of the components. The interrupt subroutines calling the user's event handlers are generated to the component modules and PE provides parts of the background code necessary to correctly handle the interrupt requests.

The **Peripheral Initialization components** can only provide the initialization of the interrupt and generate a record to the Interrupt Vector Table. The user has to provide a full implementation of the interrupt subroutine. See *3.5.3.1 Typical Usage of Peripheral Initialization Components* for details.

3.3.1.1. Interrupt Vector Table

An **interrupt vector** is a pointer to an interrupt handling subroutine.
An **Interrupt Vector Table** (IVT) is a table (list) which contains single interrupt vectors. Each interrupt corresponds to one interrupt vector. IVT may be:

- placed in **ROM** - usually first-level IVTs: the jump to the right vector is done by hardware, no interrupt vectors can be changed at runtime.
- placed in **RAM** - usually second-level IVTs: the jump to the right vector is done by software first-level interrupt handling subroutine. This implementation can be slower due to this software redirection to users routine.
  The interrupt vectors placed in RAM and **not allocated by any component** can be changed at runtime using CPU component methods (GetIntVect and SetIntVect).

The type of the IVT (ROM/RAM) can be setup using the option **Application Options | Interrupt Vector Table** .

Each processor that can handle and process interrupts has a first-level IVT (in ROM). The second-level is used only when it is necessary (i.e. IVT is placed in RAM).

Processor Expert generates content of the whole IVT into the file **vectors.c**. The content of the file depends on the compiler syntax of the interrupt declarations.

3.3.1.2. Processor Expert Priority System

Some CPUs support selectable **interrupts priorities**. The user may select a priority for each interrupt vector. The interrupt with a higher priority number can interrupt a service routine with the lower one.

Processor Expert supports the following settings in design-time: interrupt priority and priority of the event code. Priority can also be changed in the user code. The user may use a CPU component method to adjust the priority to a requested value.

Interrupt Priority

The user may select interrupt priority in the component properties, just below the interrupt vector name.
Processor Expert offers the following values, which are supported for all microcontrollers:

- **minimum priority**
- **low priority**
- **medium priority**
- **high priority**
- **maximum priority**

The selected value is automatically mapped to the priority supported by the target microcontroller. It is indicated in the third column of the Component Inspector.
The user may also select a target-specific numeric value (such as priority 255), if portability of the application to another architecture is not required.
Peripheral Initialization components on some platforms also allow to set the value **default** that means that the user doesn't have any requirement on it so the priority value will be the default after-reset value.

Priority of Event Code

The user can also select a priority for the processing of the event code. This setting is available for the events that are invoked from the Interrupt Service Routines. This priority may be different from the interrupt priority. However, the meaning of the number is the same - the event may be interrupted only by the interrupts with the higher priority. Processor Expert offers the following architecture independent values:

- **same as interrupt** - default value which means that Processor Expert doesn't generate any code affecting the priority of the event - the priority is in the state determined by the default hardware behavior.
- **minimum priority**
- **low priority**
- **medium priority**
- **high priority**
- **maximum priority**
- **interrupts disabled** - *For example the highest priority supported by the microcontroller, which may be interrupted only by non-maskable interrupts.*

The selected value is automatically mapped to the priority supported by the target microcontroller and the selected value is displayed in the third column of the Component Inspector.
Please see version specific information below. The user may also select a target-specific value, if portability of the application to another architecture is not required.

Note: Some events do not support priorities, because their invocation is not caused by the interrupt processing.

Warning: *Please note that Processor Expert does not allow the user decrease an event code priority (with the exception of 'Interrupts enabled' value on some platforms, please see below). This is because Processor Expert event routines are not generally reentrant so there is a risk that the interrupt would be able to interrupt itself during the processing. If there is such functionality requested, the user has to do it manually (e.g. by calling a appropriate CPU component method setting a priority) and carefully check possible problems.*

Version specific information for HCS12X derivatives
Processor Expert offers the following event priority options:

- **interrupts enabled** - Interrupts are enabled and the interrupts with the higher priority than the current interrupt priority can interrupt the event code. (The state of the register CCRH is not changed.)
- **interrupts disabled** - All maskable interrupts are disabled. (The state of the register CCRH is not changed.)
- **0** - The same as 'interrupts disabled'
- **1..7** - Priorities from lowest (1) to highest (7). The code generated by Processor Expert before the event invocation sets the event code priority to the specified value (by writing to the CCRH register) and enables interrupts.
- **same as interrupt** - Default behavior of the architecture - no interrupts can interrupt the event. The same as Interrupts Disabled.
- Other values are mapped to the priorities 1..7.

Version specific information for HCS12 derivatives
Processor Expert offers the following event priority options:

- **interrupts disabled** - All maskable interrupts are disabled.

- **interrupts enabled** - All maskable interrupts are enabled. Please note that this settings might lead to possible problems, see the warning within this chapter.
- **same as interrupt** - Default behavior of the architecture - no interrupts can interrupt the event. The same as Interrupts Disabled.

3.3.2. Configurations

The user can have several configurations of the project in one project file. The configuration system is very simple. 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.

Configurations of the current project are listed in the Project Panel configurations folder. They can be managed using a pop-up menu of the specific configuration or the whole Configurations folder.

Configuration properties and a user comment can be changed in the configuration inspector. See *2.5.4 Configuration Inspector* for details.

The symbol for conditional compilation is defined if it is supported by the selected language/compiler. The symbol **PEcfg_*[ConfigurationName]*** is defined in the CPU interface.
The user can switch using this symbol between variants of code according to the active configuration (see example in this chapter).

Configuration also stores which CPU is selected as the target CPU.
If the name of the configuration matches the name of one of the CodeWarrior's targets, the target is automatically selected as an active target when the user runs code generation.

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

Example

Suppose, there is a configuration named 'Testing case'. We use a component and part of our code using the component only in the 'Testing case' configuration. Then we make the testing case configuration active. After the successful code generation the CPU.H file contains the following definition:

```
/* Active configuration define symbol *
/
```

The parts of the code intended to be used only in this configuration need to be surrounded with the following lines:

```
...
#ifdef PEcfg_TestingCase
   Component_MethodCall(...);
#endif
```

3.3.3. Design Time Checking: Consequences and Benefits

During the design time, Processor Expert performs instant checking of the project. As a result of this checking, error messages may appear in the Error Window or directly in the third column of the Component Inspector (on the faulty items line). Sometimes, it may happen that only one small change in the project causes several (general) error messages. The most common reasons for this behavior are stated below.

On-Chip Peripherals

Some components use on-chip peripherals. In the Component Inspector, you can choose from all possible peripherals that can be used for implementation of the function of the current component. Processor Expert provides checking for required peripheral features such as word width and stop bit for serial channel, pull resistor for I/O pin and others.
Processor Expert also protects against the use of one peripheral in two components. If the peripheral is allocated for one component then the settings of this peripheral cannot be changed by any other component. **The state of an allocated peripheral should never be changed directly in the user code. (Using special registers, I/O ports etc.) It is recommended to always use methods generated by Processor Expert.**

Note that if a peripheral is allocated to any component, all its parts are reserved. For example if you use the 8-bit I/O port, all the I/O pins of the port are allocated and it is not possible to use them in other components.
In some timer components you can choose if you want to use only a part of the timer (compare register) or an entire timer. If you select the entire timer, the driver can be optimized to work best with the timer. It can, for example, invoke reset of the timer whenever is it needed by the component function.

Interrupt Priority

If the target CPU shares interrupt priority between several interrupt vectors or shares interrupt vectors, Processor Expert provides checking of interrupt priority settings. For detailed information about Interrupt Priority see the Priorities page.

Memory

Processor Expert always checks the usage of the internal and external memories accessible throught CPU address and data bus. Position and size of internal memory is defined by the CPU type and can be configured in the CPU Properties (if supported). External memories must be defined in CPU Properties.
Any component can allocate a specified type of memory. See component descriptions for detailed information about requirements for types of memory. Processor Expert provides checking of memory and protects you from making a wrong choice. (*For example: if a component requires external Flash, it is not possible to enter an address in internal RAM*).

The bits can also allocate memory. Therefore, you can be sure that only one component uses an allocated bit of a register in external address space.

Timing

The settings of all timed high-level components are checked using the internal timing model. See *3.3.4 Timing Settings* for details. If there is no error reported, it means that Processor Expert was successful in calculating the initialization and runtime control values for all components and hence the settings should work according to the configuration.

3.3.4. Timing Settings

Many high-level components contain a timing configuration (e.g. speed of the serial communication, period of the interrupt, conversion time of the ADC etc.). Processor Expert allows to configure such timing using user-friendly units and it also contains a model of the complete MCU timing. This model allows calculation of the required values of control registers and continuous validation of the timing settings.

Timing Model

A component timing can be viewed like a chain of elements, such as dividers and multipliers between the main clock source and the device configured by the component. The user sets the desired timing value using the Timing Dialog box (see chapter *2.5.3.1 Dialog Box for Timing Settings*) or directly by specifying the value in Component Inspector (see chapter *2.5.3.2 Syntax for the Timing Setup in the Component Inspector*). Processor Expert tries to configure individual elements in the timing chain to achieve the result and the user is informed if it was successful. After leaving the Timing Dialog box, the real value of the prescaler and timing are shown in the **third column of the component inspector**.

Figure 3.4 - Example of the timing and prescaler setup

The complete MCU timing model with the current state of the individual elements can be viewed using the CPU Timing Model Window. See *2.8 CPU Timing Model* for details.

Timing Setup Problems

The errors are reported in red in the Timing Dialog box or in the timing property line in the Component Inspector. The error summary is available in the Error window. See *2.6 Error Window* for details. Please follow the error message text to find the source of the problem. If no error is reported, it means that Processor Expert can achieve the desired timing for all components in the project.

Common problems that make impossible to set a timing value:

- *It is impossible to set some item(s).*
  This problem is reported in the component or the Timing dialog box and the user is informed which value has incorrect value. The reason is usually the hardware structure and limitations of the CPU. The Timing dialog box shows the list of values (ranges) that are allowed to be set. It might be necessary to increase the allowed error (using the 'Error' field in the Timing Dialog) that specifies the allowed difference between the required value and possible value that can be produced by the hardware.
- *Settings for the device are mutually incompatible (or can't be used with another device).*
  In this case, the problem is reported by all components that share some timing hardware. Due to dependencies between used parts of the timer, there might be necessary to adjust the values of the shared elements (such as prescalers) to the same value.

For example, if two TimerInt components are using two channels of one timer and all timer channels are connected to one common prescaler, it is not possible to set the values that would require a different prescaler values. In this case it is useful to manually adjust the prescaler values of all components to the same value (switch to Expert view mode and adjust the property in the Component Inspector window). Structure of the current target CPU timing model can be viewed by using the CPU Timing Model window.

- *The Runtime setting from interval is selected and it is not possible to set the values.*
  The required run-time settings are outside the range of one prescaler. This is a limitation of this mode of runtime setting. Please see the text below in this chapter.

Run-time Timing Settings Limitation

Some components allow to change the timing at run-time by switching among several predefined values or by setting a value from given interval.
For the runtime setting *from interval* the prescaler value is fixed and the Processor Expert allows to adjust the time using a compare/reload registers. **It means that Processor Expert allows to configure the limits of an interval only within a range of one prescaler and it's possible to set values from this interval only.** See *2.5.3.1 Dialog Box for Timing Settings* for details.

Speed Modes

Processor Expert provides three speed modes that are generalization of all the possible CPU clock speed modes used for power-saving supported by most of the modern microcontrollers. See *3.2.2.2 Speed Modes Support* for details.

3.3.5. Creating User Component Templates

If you frequently use a component with the same specific settings, you may need to save the component with its settings as a template. This template is displayed in the Components library under given name, **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.
This section describes how to create a component template and save it.

Creating and Saving Templates

Right-click the required component icon in the Project panel in order to display the Component pop-up menu.

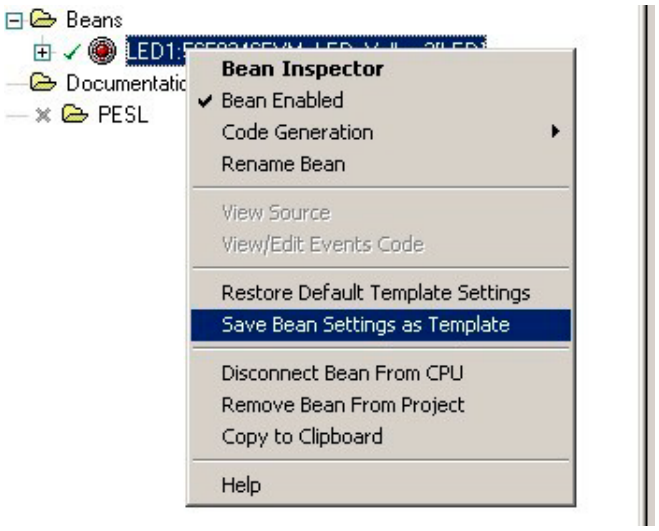


Figure 3.5 - Component Pop-up Menu

Select the **Save component settings as template**. The Component Template dialog box appears that can be used to create and save the template.

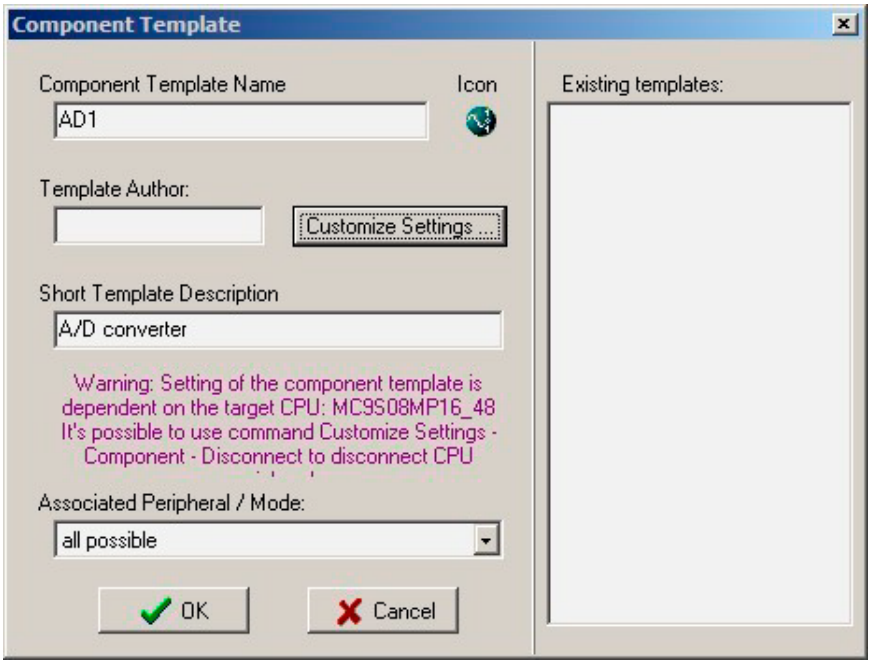


Figure 3.6 - Component Template Dialog

The Component Template dialog box consists of the following elements:

- **Component template name** - Name of the new template
- **Icon** Icon representing the template. Click the icon to select a different icon.
- **Template author** - Your name, company name, etc.
- **Customize settings...** - Button invokes the Template Editor which allows to change the settings of the component. These settings are saved as a template. (See the following section 'Customizing Component Template Settings' for details).
- **Short template description:** Type a short description.
- User gets one of the messages: "Setting of the component template is not dependent on the target CPU." or "Warning: Setting of the component template is dependent on the target CPU". The second one means that the template will be shown in components library only for the current target CPU.
- **Associated peripheral / Mode** - *This setting is present only if it is meaningful.* If the user chooses a peripheral, the template will be shown in *OnChipPeripheral* mode only for this peripheral.
- **Existing templates** - Shows list of the existing templates for the component. Clicking on the item in the list loads the templates settings into the input fields.

After clicking the **OK** button, the component template is saved and automatically added to the Components Selector tree.

Customizing Component Template Settings

If you click the **Customize settings...** button in the Component Template dialog box, the Template Editor dialog box is shown and the user can make the following changes in the template:

- Set default values of properties,
- Set default values of methods or events (whether it has be generated or not generated),
- Rename methods (by double-clicking on method name),
- Set feature of properties, methods or events as:
  - *Read Only* - The user cannot edit default values of properties or change features such as whether or not to generate or not to generate methods or events
  - *Changeable* - The user can edit default values of properties or change features, such as whether or not to generate methods or events, by double clicking in the first column in Component Inspector.

  The icon shows the status:
  -  - Feature is ReadOnly.
  -  - Feature is Changeable.
- Set level of visibility of properties, methods, or events by repeatedly double-clicking in last column in Component Inspector. When the user uses the template, it's possible to change the level of visibility in the View menu in the Component Inspector (See *2.5.3 Component Inspector* for details.).
  Possible values:
  - **Type: BASIC** - Property/method/event will always be visible.
  - **Type: ADVANCED** - Property/method/event will be visible if it is selected in Advanced view.
  - **Type: EXPERT** - Property/method/event will be visible if it is selected in Expert view.
  - **Type: @ HIDDEN @** - Property/method/event will never be visible.

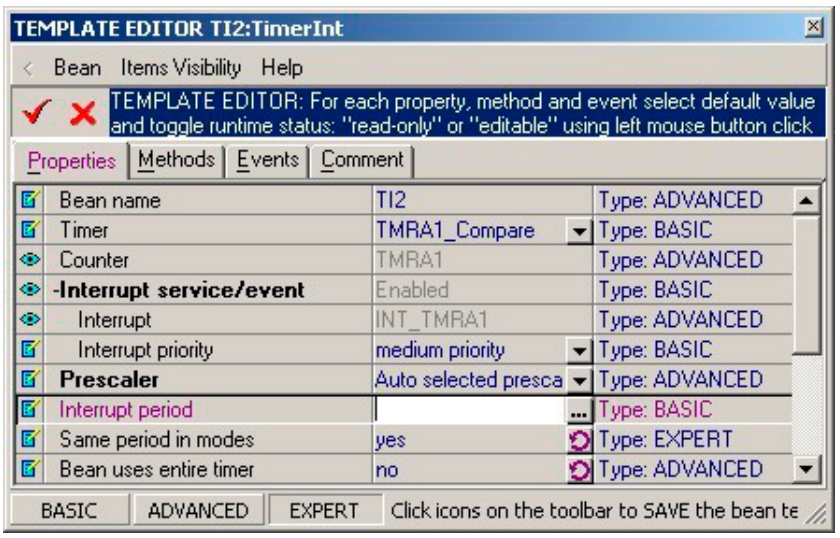


Figure 3.7 - Template Editor Window

Finally, click the  icon on the component inspector's toolbar to accept the changes.

To discard the changes and return without changing a template settings, close the window or click the  icon.

3.3.6. Signal Names

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

Assigning Signals to Pins

A signal name can be assigned to an allocated pin by specifying the signal name into the appropriate property (e.g. Pin_signal) in the Component Properties (available in ADVANCED view mode). **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 names 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.

Figure 3.8 - Signal names list for a port

Generated Documenation

Processor Expert automatically generates a document *{projectname}_SIGNALS.txt* or *{projectname}_SIGNALS.doc* containing a list of relationship between defined signals and corresponding pins and vice versa. There is an additional signal direction information added next to each signal name and pin number information next to each pin name. This document can be found in the *Documentation* folder of the Project Panel.

Sample of generated signals documentation:

```
================================================================
=
 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]


================================================================
=
```

```
------------------------------------------------------------------
- PIN-NAME [PIN-NUM]       => SIGNAL-NAME [DIRECTION]
------------------------------------------------------------------
-
GPIOA8_A0 [138]          => LED1 [Output]
GPIOA9_A1 [10]           => LED2 [Output]
GPIOC4_TA0_PHASEA0 [139] => Timer [Output
]
GPIOC5_TA1_PHASEB0 [140]  > Sensor [Input
```

3.3.7. Component Inheritance and Component Sharing

Basic Terms

- 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.

Inheritance

Inheritance means that an ancestor component is used only by the descendant component. Inheritance is supported in order 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 (for example some MPC5500 Peripheral Initialization 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.

Figure 3.9 - Inherited component item in inspector.

Clicking on the button unfolds the properties of the inherited component directly under the intherited component property line.
Clicking on the button fully opens the ancestor component with properties, methods, and events within the Component inspector.
Processor Expert allows the user to select from several ancestors that implement a required interface and are registered by the descendant component.
The ancestor component is displayed under its descendant in the project structure tree in the project panel.

Figure 3.10 - Example of ancestor and descendant components in the project panel tree.

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.

Component Sharing

Component sharing allows the user to cause several components to use capability of one component with the way similar to inheritance. This feature allows sharing of its resources and its drivers with other Components.
For example, components may share an I2C component for communication with peripherals connected to the I2C bus.

Settings in Processor Expert

A shared ancestor component contains a property that allows the user 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.
Click the button to select an existing shared ancestor component from the current project. Click the button to select an existing ancestor component or create a new ancestor component using the wizard (see below).

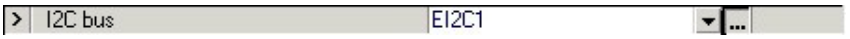


Figure 3.11 - Example of popup menu for creating a new shared ancestor component.

Selection/Creation Wizard

When a component with a link to a shared ancestor component is added to the project, the following wizard dialog box appears. This wizard helps you to select or create the shared ancestor component quickly.

Figure 3.12 - Example of popup menu for creating new shared ancestor component.

Run-time Resources Allocation

Processor Expert (generated code) does not check the usage of shared resources/code. It's up to the user 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.

3.3.8. Pin Sharing

Reading a Pin Value without its Initialization

The **InputPin** component supports reading of input pin signal without any previous pin initialization. The feature can be used only if the pin (port) contains RAW DATA register, which allows reading the input pin signal in any settings of the related peripherals. See the component documentation for more details.

Sharing Pins among Peripherals

Note: This feature is not available in this version.

Some CPUs allows some pins to be used by multiple peripherals. This may lead to the need of sharing pin(s) by multiple components. Normally, if the user selects 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 by clicking the sharing icon on the right side (see the picture below).

Pin sharing can be set in the component inspector. The Component Inspector must be switched into the **EXPERT view mode**, and then the pin sharing button must be pressed down. The figure below displays the Pin Sharing button at the right side of the second column.

Figure 3.13 - Pin Sharing Button

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

ConnectPin Method

Note: This feature is not available in this version.

It's necessary to invoke the component method *ConnectPin* to connect a component to the shared pin. It's also necessary to invoke the main component method to connect 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's not necessary to use the *ConnectPin* method.

Shared pins are presented in the Target CPU view as well. The component to pin connection line is red.

3.4. Implementation Details

This chapter contains implementation details for Embedded Components and Processor Expert generated code. The following subchapters describe:

- Reset Scenario with PE
- Version Specific Information for HCS12 and HCS12X

Additional implementation specific information can be found on individual component documentation pages.

3.4.1. Reset Scenario with PE

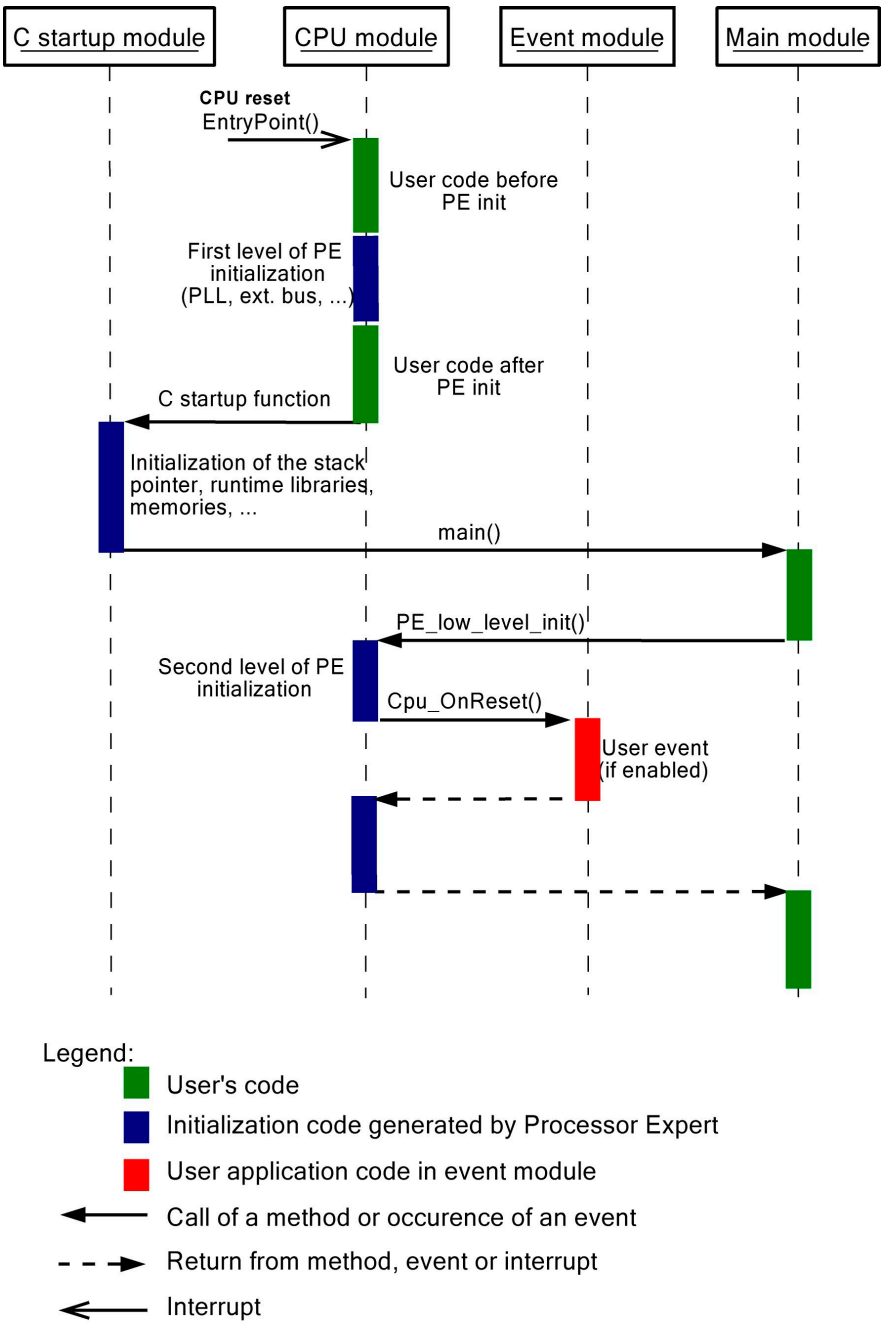


Figure 3.14 - Reset sequence diagram with Processor Expert

_EntryPoint function

The *_EntryPoint()* function is called as the first function after the reset. This function is defined in the cpu module, usually *Cpu.c*, and provides necessary system initialization such as PLL and external bus.
Sometimes it is necessary to do some special user initialization immediately after the cpu reset. Processor Expert provides a possibility to **insert user code into the *_EntryPoint()* function**. There is a *User Initialization property* in the build options tab of a CPU component inspector defined for this purpose. See *2.5.3 Component Inspector* for details.

C startup function

The C startup function in the C startup module is called at the end of the *_EntryPoint()* function. It provides a necessary initialization of the stack pointer, runtime libraries, and so on. At the end of the C startup function the *main()* function is called.

PE_low_level_init()

There is a second level of Processor Expert initialization *PE_low_level_init()* called at the beginning of the *main()* function. *PE_low_level_init()* function provides initialization of all components in project and it is necessary for proper functionality of the Processor Expert project.

OnReset event

The user can write the code that will be invoked from the PE_low_level_init() function after the Processor Expert internal initialization, but before the initialization of individual components. Thus, the user should expect that peripherals are not completely initialized yet. This event can be enabled/disabled in the CPU component inspector's events page.

3.4.2. Version Specific Information for HCS12 and HCS12X

All components were tested with the CodeWarrior with the following compiler settings:

- Other parameters = -Onf

The ROM and RAM ranges depend on the target CPU. It is recommended to increase the stack size if some standard libraries are used.

Components' implementation details :

- **All the components**:
  - *Interrupt priority* and *Event priority* - Please see the version specific information details in the chapter *3.3.1.2 Processor Expert Priority System*.
  - *MISRA compliance* - All component have been developed to be MISCRA C 2004 standard compliant as much as possible. The exceptions to this compliance are documented in the HTML page accessible in the file `{CodeWarrior}\ProcessorExpert\DOCs\Misra2004Compliance.html`.
- **CPU**:
  - *Speed Mode* selection (CPU methods *SetHighSpeed, SetLowSpeed, SetSlowSpeed*): if CPU clock-dependent components are used then signals generated from such an internal peripheral may be corrupted at the moment of the speed mode selection (if function of clocked devices is enabled). Handling of such a situation may be done via events **BeforeNewSpeed** and **AfterNewSpeed**.

- *Interrupt vectors table (IVT)* is by default generated on the default addresses for of the current target CPU. However, Processor Expert offers additional configuration of the IVT:

  - **On HCS12 derivatives:**
    The placement of the IVT can be configured in the Build Options tab of the CPU Component Inspector by changing the address of the memory area with the name **INT_VECTORS**.
    Please note that if the IVT placement is changed, the user has to provide a full IVT on the the address defined by the CPU datasheet and the vectors allocated by Processor Expert have to be redirected into the IVT generated by PE.
    If the *interrupt vector table in RAM* application option is selected then it generates the table in RAM and special redirection code to ROM. This code transfers program control to the selected address according the table in RAM. You can use the CPU method *SetIntVect* to set the address of interrupt service routine. It is recommended to select the event OnSWI together with this option to minimize the size of generated code. *Please note that the redirection is available only for interrupt vectors not used by Embedded Components in the current project.*

  - **On HCS12X derivatives:**
    These derivatives allow to change the placement of the interrupt vectors beginning. So the Processor Expert allows to adjust both - the physical placement of vectors or the placement of the generated IVT. Please see the content of the property group *Interrupt/Reset vector table* in the group *Interrupt resource mapping* and its documentation.

- **PPG**: HW doesn't support an interrupt. Aligned Center Mode Counter counts from 0 up to the value period register and then back down to 0. If the align mode is switched to Center align mode then real lengths of Period and Starting pulse width signals will be twice as much as is being displayed in the Component Inspector. *Note: See the Internal peripheral property group of the CPU component for special settings.*
- **PWM**: HW doesn't support an interrupt. Aligned Center mode Counter counts from 0 up to the value period register and then back down to 0. If align mode is switched to Center align mode then the real lengths of Period and Starting pulse width signals will be twice as much as is being displayed in the Component Inspector. *Note: See the Internal peripheral property group of the CPU component for special settings.*
- **EventCntr8/16/32**: Functionality of this component is a subset of the pulse accumulator. For work with hold registers, gated time mode use the PulseAccumulator component instead of the EventCounter component.
- **PulseAccumulator**:

  - **Method Latch**
    This method causes capture of the counter in the hold registers of all capture and pulse accumulator components in PE project because this method is invoked for all ECT modules.

  Note: See Internal peripheral property group of the CPU component for special settings.

- **Capture**:

  - **Method Reset** -If the counter can't be reset (is not allowed by HW or the counter is shared by more components) this method stores the current value of the counter into a variable instead of a reset.
  - **Method GetValue** -If the counter can't be reset (is not allowed by HW or the counter is shared by more components) this method doesn't return the value of register directly, but returns the value as a difference between the register value and the previously stored register value. This causes values that are proportional to time elapsed from the last invocation of the method Reset.
  - **Method Latch** -This method causes capture of the counter in the hold registers of all capture and pulse accumulator components in PE project because this method is invoked for all ECT modules.
  - **Method GetHoldValue** -This method transfers the contents of the associated pulse accumulator to its hold register.

Note: See the Internal peripheral property group of the CPU component for special settings.

- **BitIO, BitsIO, ByteIO, Byte2IO, Byte3IO, Byte4IO**:
  The *GetVal* and *GetDir* methods are always implemented as macros.
- **LongIO:**
  This component could not be implemented on Freescale HCS12 - this CPU has no instructions for 32-bit access into the I/O space.
- **IntEEPROM:**
  The EEPROM array is organized as rows of word (2 bytes), the EEPROM block's erase sector size is 2 rows (2 words). Therefore it is preferable to use word aligned data for writing - methods SetWord and SetLong - with word aligned address or to use virtual page - property 'Page'. The size has to be a multiple of 4 bytes.
- **SynchroMaster:**
  The mode fault causes disability of the component (and SPI device) automatically (inside interrupt service) if interrupt service is enabled. If the interrupt service isdisabled and a mode fault occurs, the component will be disabled at the beginning of RecvChar method.
- **IntFlash:**
  The Virtual page - *Allocated by the user* feature and corresponding methods and events are not implemented.
- **ExtInt:**
  If XIRQ is selected, the method 'Disable' can't be generated, because it isn't supported by hardware. For pins of H, J, and P ports it is not possible to switch pull resistor (pull up/pull down) and sensitive edge (rising edge/falling edge) arbitrarily. Because of hardware limitations, pull down with falling edge and pull up with rising edge settings aren't allowed.

3.5. Code Generation and Usage

This chapter informs the user 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.

Please refer to the following subchapters for more information:

3.5.1. Code Generation

Processor Expert > Generate Code '{ProjectName.mcp}'

Generate Code command initiates the code generation process. During this process source code modules containing functionality 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.

Files Produced by Processor Expert

The existence of the files can be conditional to project or Processor Expert environment settings and their usage by the components.

- **Component module**
  This module with its header file is generated for every component in the project with exception of some components that generate only an initialization code or special source code modules. Name of this file is the same as the name of the component.
  Header file (.h) contains definitions of all public symbols, which are implemented in the component module and can be used in the user modules.
  The module contains implementation of all enabled methods and may also contain some subroutines for internal usage only. Processor Expert allows to track and review changes in the generated modules. See *3.5.1.1 Tracking Changes in Generated Code* for details.

- **CPU module**
  The CPU module is generated according to the currently active target CPU component. The CPU module additionally contains:
  - CPU initialization code
  - interrupt processing

- **Main module**
  The main module is generated only if it does not already exist (if it exists it is not changed). Name of this module is the same as the name of the project.
  The main module contains the **main** function, which is called after initialization of the CPU (from the CPU module). By default this function is generated empty (without any reasonable code). It is designed so that the user can write code here.

- **Event module**
  The event module is generated only if it does not exist. If it exists, only new events are added into the module; user written code is not changed.
  The event module contains all events selected in the components. By default these event handler routines are generated empty (without any meaningful code). It is considered that user will write code here.
  Event module can also contain the generated ISRs for the components that require a direct interrupt handling (Peripheral Initialization Components). Generation of the ISRs is controlled by the project option **Project Options | Generate ISR**.
  It is possible to configure the name of event module individually for each component in the ADVANCED view mode of the Component Inspector. However, please note that the event module is not generated by Processor Expert if there is no event enabled in the component, except the CPU component, for which the event module is always generated.

- **Method list file** with description of all components, methods and events generated from your project. The name of the file is *{projectname}.txt* or *{projectname}.doc*. This documentation can be found in the *Documentation* folder of the Project Panel.

- **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 Project Panel. See *3.3.6 Signal Names* for details.

- **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 Project Panel. It is updated after each successful code generation.

- **Shared modules** with shared code (the code which is called from several components). Complete list of generated shared modules depends on selected CPU, language, compiler and on the current configuration of your project. Typical shared modules are:
  - **IO_Map.h**
    Control registers and bit structures names and types definitions in C language.
  - **IO_Map.c**
    Control registers variable declarations in C language. This file is generated only for the HC(S)08/HC(S)12 versions.
  - **Vectors.c**
    A source code of the interrupt vector table content.
  - **PE_Const.h**
    Definition of the constants, such as speed modes, reset reasons. This file is included in every driver of the component.
  - **PE_Types.h**
    Definition of the C types, such as bool, byte, word. This file is included in every driver of the component.
  - **PE_Error.h**
    Common error codes. This file contains definition of return error codes of component's methods. See the generated module for detailed description of the error codes. This file is included in every driver of the component.
  - **PE_Timer**
    This file contains shared procedures for runtime support of calculations of timing constants.
  - **{startupfile}.c**
    This external module, visible in the External Modules folder of the Project Panel, contains a platform specific startup code and is linked to the application. The name of the file is different for the Processor Expert versions. For datils on the use of the startupfile during the reset see chapter *3.4.1 Reset Scenario with PE*

See also chapter *3.5.2 Predefined Types, Macros and Constants*.

3.5.1.1. Tracking Changes in Generated Code

Processor Expert allows to compare generated modules with the previously generated ones after each code generation which may prevent from unwanted changes in the component modules. This function has to be enabled by the **Project Options | Track changes**.

The 'Modified Files' dialog appears after the successful code generation and shows the list of the files that have been modified by Processor Expert. The user can specify which files will be saved and will replace the old ones using a check marks and a buttons 'All' and 'None'.
The user can also visually compare the changes in the currently selected file by pressing a button **DIFF**. Pressing the 'OK' button will save all selected modules and replace the current ones.

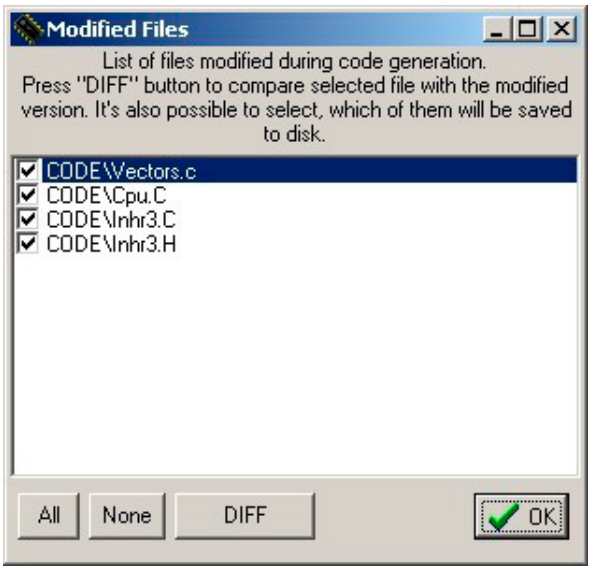


Figure 3.15 - List of modyfied files

If the user presses the *DIFF* button, a file editor in a comparison mode is shown. It contains highlighted parts that had been changed during the code generation. See *2.15 File Editor* for details.

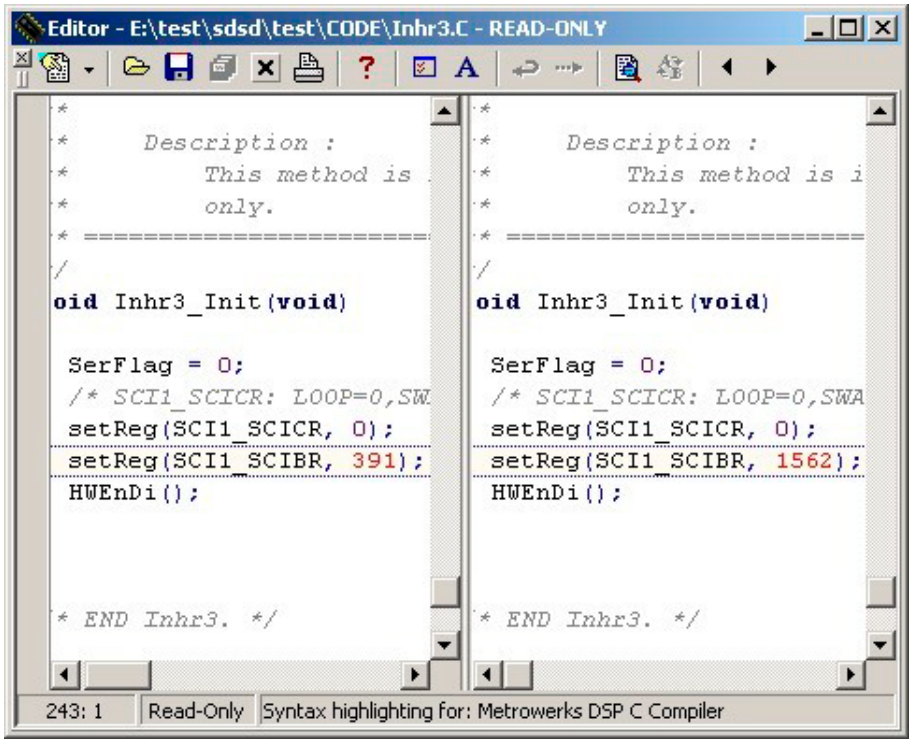


Figure 3.16 - A file before and after the generation

After the 'Modified Files' dialog, user can also review a list of automatically deleted unused files (The **Environment Options | Delete unused files** environment option has to be enabled).

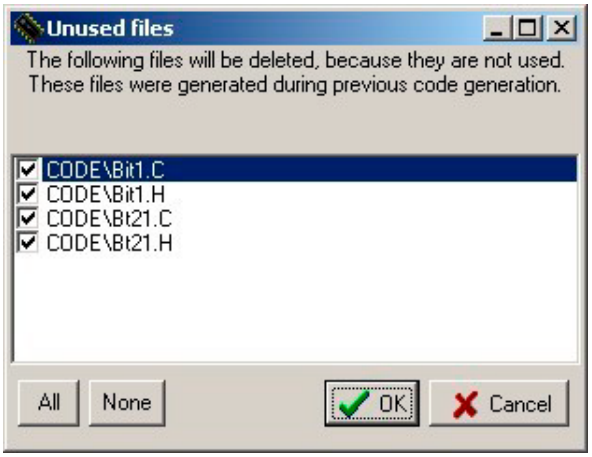


Figure 3.17 - List of unused files

3.5.2. Predefined Types, Macros and Constants

Processor Expert generates definitions of all hardware register structures to the file **IO_Map.h.** The Processor Expert type definitions are generated to the file **PE_Types.h** which also containins definitions of macros used for a peripheral register access. See *3.8.1 Direct Access to Peripheral Registers* for details.

Types

Type	Description	Supported for
byte	8-bit unsigned integer (unsigned char)	all
bool	Boolean value (unsigned char) (TRUE = any non-zero value / FALSE = 0)	all
word	16-bit unsigned integer (unsigned int)	all
dword	32-bit unsigned integer (unsigned long)	all
dlong	array of two 32-bit unsigned integers (unsigned long)	all
TPE_ErrCode	Error code (uint8_t)	all except MPC55xx

Structure for images

```
typedef struct {          /* Image */
  word width;             /* Image width in pixels */
  word height;            /* Image height in pixels *
/
  byte *pixmap;           /* Image pixel bitmap */
  word size;              /* Image size in bytes */
  char *name;             /* Image name */
} TIMAGE;
```

Structure for 16-bit register:

```
/* 16-
bit register (big endian format) */
  typedef union {
     word w;
     struct {
       byte high,low;
     } b;
  } TWREG;
```

Macros

```
  __DI()              -
 Disable global interrupts
  __EI()              - Enable global interrupts

  EnterCritical()     -
 It saves CCR register and disable
                       global interrupts
  ExitCritical()      - It restores CCR register saved
```

For the list of macros available for Peripheral registers access, please see the chapter *3.8.1 Direct Access to Peripheral Registers*

Constants

Methods Error Codes

The error codes are defined in the *PE_Error* module. Error code value is 8-bit unsigned byte. Range 0 - 127 is reserved for PE, and 128 - 255 for user.

ERR_OK	0	OK
ERR_SPEED	1	This device does not work in the active speed mode
ERR_RANGE	2	Parameter out of range
ERR_VALUE	3	Parameter of incorrect value
ERR_OVERFLOW	4	Timer overflow
ERR_MATH	5	Overflow during evaluation
ERR_ENABLED	6	Device is enabled
ERR_DISABLED	7	Device is disabled
ERR_BUSY	8	Device is busy
ERR_NOTAVAIL	9	Requested value not available
ERR_RXEMPTY	10	No data in receiver
ERR_TXFULL	11	Transmitter is full
ERR_BUSOFF	12	Bus not available
ERR_OVERRUN	13	Overrun is present
ERR_FRAMING	14	Framing error is detected
ERR_PARITY	15	Parity error is detected
ERR_NOISE	16	Noise error is detected
ERR_IDLE	17	Idle error is detected
ERR_FAULT	18	Fault error is detected
ERR_BREAK	19	Break char is received during communication
ERR_CRC	20	CRC error is detected
ERR_ARBITR	21	A node loses arbitration. This error occurs if two nodes start transmission at the same time
ERR_PROTECT	22	Protection error is detected.
ERR_UNDERFLOW	23	Underflow error is detected.
ERR_UNDERRUN	24	Underrun error is detected.
ERR_COMMON	25	General unspecified error of a device. The user can get a specific error code using the method GetError.
ERR_LINSYNC	26	LIN synchronization error is detected
ERR_FAILED	27	Requested functionality or process failed.
ERR_QFULL	28	Queue is full.

3.5.3. Typical Usage of Component in User Code

This chapter describes usage of methods and events that are defined in most hardware components. Usage of other component specific methods is described in the component documentation, in the section "Typical Usage" *(if supported)*.

In the following examples assume there is a component named "B1".

Peripheral Initialization Components

Peripheral Initialization Components are the components at the lowest level of peripheral abstraction. These components contain only one method Init providing the initialization of the used peripheral. See *3.5.3.1 Typical Usage of Peripheral Initialization Components* for details.

Methods Enable, Disable

Most of the hardware components support the methods Enable and Disable. These methods enable or disable peripheral functionality, which causes disabling of functionality of the component as well.
Hint: Disabling of the peripheral functionality may save CPU resources.

Overview of the method behavior according to the component type:

- Timer components: timer counter is stopped if it is not shared with another component. If the timer is shared, the interrupt may be disabled (if it is not also shared).
- Communication components, such as serial or CAN communication: peripheral is disabled.
- Conversion components, such as A/D and D/A: converter is disabled. The conversion is restarted by Enable.

If the component is disabled, some methods may not be used. Please refer to components documentation for details.

MAIN.C

```
  void main(void)
  {
    ...
    B1_Enable();   /* enable the component functionality *
/
                /* handle the component data or settings *
/
    B1_Disable(); /* disable the component functionality *
/
```

Methods EnableEvent, DisableEvent

These methods enable or disable invocation of all component events. These methods are usually supported only if the component services any interrupt vector.

The method DisableEvent may cause disabling of the interrupt, if it is not required by the component functionality or shared with another component. The method usually does not disable either peripheral or the component functionality.

MAIN.C

```
  void main(void)
  {
    ...
    B1_EnableEvent();  /* enable the component events */
                       /* component events may be invoked *
/
    B1_DisableEvent(); /* disable the component events */
                       /* component events are disabled */
    ...
  }
```

Events BeforeNewSpeed, AfterNewSpeed

Timed components that depend on the CPU clock such as timers, communication and conversion components, may support speed modes defined in the CPU component (in EXPERT view level). The event BeforeNewSpeed is invoked before the speed mode changes and AfterNewSpeed is invoked after the speed mode changes. Speed mode may be changed using the CPU component methods SetHigh, SetLow, or SetSlow.

EVENT.C

```
  int changing_speed_mode = 0;

  void B1_BeforeNewSpeed(void)
  {
    ++changing_speed_mode;
  }

  void B1_AfterNewSpeed(void)
  {
    --changing_speed_mode;
  }
```

Note: If the speed mode is not supported by the component, the component functionality is disabled, as if the method Disable is used. If the supported speed mode is selected again, the component status is restored.

TRUE and FALSE Values of bool Type

Processor Expert defines the TRUE symbol as 1, however true and false logical values in C language are defined according to ANSI-C:

- False is defined as 0 (zero)
- True is any non-zero value

It follows from this definition, that the bool value cannot be tested using the expressions, such as `if (value == TRUE) ...`
Processor Expert methods returning bool value often benefit from this definition and they return any non-zero value as TRUE value instead of 1. The correct C expression for such test is: **`if (value) ...`**.

In our documentation, the "true" or "false" are considered as logical states, not any particular numeric values. The capitalized "TRUE" and "FALSE" are constants defined as FALSE=0 and TRUE=1.

3.5.3.1. Typical Usage of Peripheral Initialization Components

Init method

Init method is defined in all Peripheral Initialization Components. Init method contains a complete initialization of the peripheral according to the component's settings.

In the following examples, let's assume a component named "Init1" has been added to the project.

The Init method of the Peripheral Initialization component can be used in two ways.

- The Init method is called by Processor Expert
- The Init method is called by the user in his/her module

Automatic calling of Init
The user can let Processor Expert call the Init method automatically by selecting "yes" for the **Call Init method** in the Initialization group of the Component's properties.

When this option is set, Processor Expert places the call of the Init method into the *PE_low_level_init* function of the CPU.c module.

Manual calling of Init
Add the call of the Init method into the user's code, for example in main module.

Enter the following line into the main module file:

```
Init1_Init();
```

Put the Init method right below the PE_low_level_init call.

```
void main(void)
{
  /*** Processor Expert internal initialization. ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization. ***
/
  Init1_Init();
  for(;;) {}
```

Interrupt Handling

Some Peripheral Initialization components allow the initialization of an interrupt service routine. Interrupt(s) can be enabled in the initialization code using appropriate properties that can be usually found within the group *Interrupts*.
After enabling, the specification of an Interrupt Service Routine (ISR) name using the *ISR name* property is required. This name is generated to Interrupt Vector table during the code generation process. See *3.3.1.1 Interrupt Vector Table* for details.
Please note that if the ISR name is filled, it is generated into the Interrupt Vector Table even if the interrupt property is disabled.

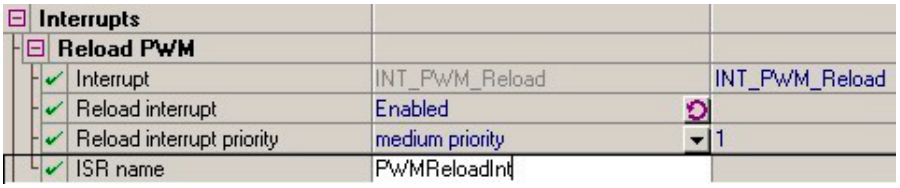


Figure 3.18 - Example Of The Interrupt Setup

Enabling/disabling peripheral interrupts during runtime has to be done by user's code, for example by utilizing PESL or direct register access macros, because the Peripheral Initialization Components do not offer any methods for interrupt handling.

The ISR with the specified name has to be declared according to the compiler conventions and fully implemented by the user. If the project option **Project Options | Generate ISR** is enabled, declarations of the ISRs that already do not exist can be generated automatically by PE during the code generation process into the Event module of the CPU component. The project option is visible in the Advanced level of view. The name of the CPU event module could be changed in the Advanced view mode of the Component Inspector.

3.5.4. User Changes in Generated Code

It's necessary to say at the beginning of the chapter, that modification of the generated code may be done only at user's own risk. Generated code was thoroughly tested by the skilled developers and the functionality of the modified code cannot be guaranteed. We strongly don't recommend modification of the generated code to the beginners. See more information for generated modules in chapter Code Generation.

To support user changes in the component modules, Processor Expert supports the following features:

1. CH file - manifest constants

Processor Expert automatically produce list of manifest constants for all components, which encapsulate any CPU peripheral and modifies any of the CPU control registers. The name of each manifest constant is in the following format:

C_*[component+method]*_reg_*[register name][additional-info]*

where

- *[component+method]* is name of component (and optionally also component's method),
- *[register name]* is name of the control register,
- *[additional-info]* is additional information about usage of the value (modification of one bit, bits mask or whole value).

These constants may be used to write user code, which reflects the component settings. Once the constant is generated into the CH file, it is preserved there even it is already not used in the component module. The most important advantage of these constants is, that small changes in the component settings (for example timing) does not cause change of the component module, but the only CH file is changed.

The name of the CH file is derived from the CPU component name ([CPUcomponent].CH). To generate CH file it is necessary to set the following option: **Project Options | Generate manifest constants**. CH file is generated also while smart generation of component modules code, see paragraph 3 for details. CH file is always overwritten during the code generation.

2. Mode of code generation for component modules

It's possible to select mode of the code generation for each component, the following options can be found in the components's pop-up menu in the **project panel**:

Code Generation

- **Always Write Generated Component Modules** (default) - generated component modules are always written to disk and any existing previous module is overwritten
- **Preserve User Changed in Generated Component Modules** - smart detection of user changes.

  Note: Smart user changes preservation is not available in this version.

- **Don't Write Generated Component Modules** - the code from component is not generated. Any initialization code of the component, which resides in the CPU component, interrupt vector table and shared modules are updated.

The mode of code generation is indicated as a component's status in the project panel. This mode influences only the generation of component modules (component.c, component.h).

3. Mode of code generation for non-component modules

Processor Expert also allows to enable/disable generation of the modules that not related to a specific component or that are common for several components. This option can be configured in the pop-up menu of the module in the Project Panel. Current enable/disable state of this option is signalled by the icon near the component module name in the Project Panel (- enabled, - disabled).

Code Generation

- **Always Write** (default) - generated component modules are always written to disk and any existing previous module is overwritten
- **Don't Write** - the content of the module is not overwritten. *Please note that this can lead to malfunction of components dependent on the module when the module update is required due to component's settings change.*

Viewing User Changes in a Component Module

The user changes done in a component module or component header module can be viewed using a component pop-up menu commands **Compare With Previously Generated Module** and **Compare With Previously Generated Header Module** . See *2.3.4 Component Pop-up Menus* for details. The user can also enable reviewing all changes done into the generated code after each code generation. See *3.5.1.1 Tracking Changes in Generated Code* for details.

3.6. Embedded Component Optimizations

This chapter describes how the size and speed of the code could be optimized by choosing right component for the specific task. It also describes how to setup components to produce optimized code. The optimizations that are described are only for the High or Low level components, not for the Peripheral Initialization components.

Please refer to sub-chapters for more details:

- General Optimizations
- General Port I/O Optimizations
- Timer Components Optimizations
- Code Size Optimization of Communication Components

3.6.1. General Optimizations

This chapter describes how to setup Processor Expert and components to generate optimized code. The following optimization are only for the High or Low-level components, and not for the Peripheral Initialization components.

Disabling Unused Methods

Note: These optimization are not usable for the Peripheral Initialization Components.
When Processor Expert generates the code certain methods and events are enabled by default setting, even when the methods or events are not needed in the application, and thus while they are unused, the code may still take memory. Basically, the unused methods code is dead stripped by the linker but when the dependency among methods is complex some code should not be dead stripped. When useless methods or events are enabled the generated code can contain spare source code because of these unused methods or events. Moreover some methods can be replaced by more efficient methods that are for special purposes and therefore these methods are not enabled by default.

Disabling Unused Components

Disable unused and test purpose components or remove them from the project. Disabling of these components is sufficient because the useless code is removed but the component setting remains in the project. If these components are required for later testing then add a new configuration to the project and disable these useless component only in the new configuration. The previous configuration will be used when the application is tested again. Moreover if it is required to use the same component with different setting in several configurations, its possible to add one component for each configuration with same name and different setting.

Speed modes

Note: These optimizations are not usable for the Peripheral Initialization Components.
Timed components which depend on the CPU clock (such as timer, communication and conversion components), may support speed modes defined in the CPU component (in EXPERT view level). The Processor Expert allows the user to set closest values for the component timing in all speed modes (if possible) . If the requested timing is not supported by the component, for example if the CPU clock is too low for the correct function of the component, the component can be disabled for the appropriate speed mode. The mode can be switched in the runtime by a CPU method. The component timing is then automatically configured for the appropriate speed mode or the component is disabled (according to the setting). Note, however, that use of speed modes adds extra code to the application. This code must be included to support different clock rates. See speed mode details here.

See Configuration Inspector for more optimization settings.

See chapter Embedded Components Optimizations for details on choosing and setting the components to achieve optimized code.

3.6.2. General Port I/O Optimizations

Note: These optimizations are not usable for the Peripheral Initialization Components.

ByteIO Component Versus BitsIO Component

ByteIO component instead of BitsIO component should be used when whole port is accessed. The BitsIO component is intended for accessing only part of the port (e.g. 4 bits of 8- bit port)

Using the BitsIO component results more complex code because this component provides more general code for the methods, which allows access to only some of the bits of the port. On the other side, the ByteIO component provides access only to the whole port and thus the resulted code is optimized for such type of access.

BitsIO component versus BitIO components

In case of using only a part of the port the multiple BitIO components could be used. A better solution is to use the BitsIO component replacing multiple calls of the BitIO component's methods. The application code consist only of one method call and is smaller and faster.

3.6.3. Timer Components Optimizations

Note: These optimizations are not usable for the Peripheral Initialization Components.

For better **code size performance,** it's recommended to not to use a bigger counter/reload/compare register for timer than is necessary. Otherwise the code size generated by a component may be increased (e.g. For 8-bit timer choose 8bit timer register).

In some cases, several timing periods are required when using timers (For example, the TimerInt component). The Processor Expert allows changing the timer period during run-time using several ways (note that this is an advanced option and the Component Inspector Items visibility must be set to at least 'ADVANCED').

These ways of changing the run-time period of timer requires various amount of code and thus the total application code size is influenced by the method chosen. **When the period must be changed during run-time**, use fixed values for period instead of an interval if possible to save code. There are two possibilities (See *2.5.3.1 Dialog Box for Timing Settings* for details.):

- **From list of values** - this allow to specify several (but fixed in run-time) number for given periods. This allows only exact values - modes, listed in the listbox. The resulted code for changing the period is less complex than using an interval.
- **From time interval** - this is an alternative to using 'list of values', which requires more code. Using an interval allows setting whatever value specified by the component during run-time. This code re-calculates the time period to the CPU ticks and this value is used when changing the timer period.

If the application requires only a few different timing periods, even if the functionality is the same for both cases, the correct usage of list of periods produces smaller code compared to code using an interval.

3.6.4. Code Size Optimization of Communication Components

Note: These optimizations are not usable for the Peripheral Initialization Components.
Communication components should be used with the smallest possible buffer. Thus the user should compute or check the maximum size of the buffer during execution of the application. For this purpose the method GetCharsInTxBuffer/GetCharsInTxBuffer (AsynchroSerial component), which gets current size of a used buffer, can be used after each time the SendBlock/RecvBlock method is called.

Use **interrupts** if you require faster application response. The interrupt routine is performed only at the event time, that is the code does not check if a character is sent or received. Thus the saved CPU time can be used by another process and application is faster.

Use **polling mode** instead of interrupts if you require less code because usually overhead of interrupts is bigger than overhead of methods in polling mode. But the polling mode is not suitable for all cases. For example when you use the SCI communication for sending only the data, and a character is sent once in a while, then it is better to use the polling mode instead of using interrupt because it saves the code size, that is when the interrupt is used an interrupt subroutine is needed and code size is increased.

Examples:

A module of an application sends once in a while one character to another device through the SCI channel. If the delay between two characters is sufficient to sent one character at a time then the polling mode of the SCI (the AsynchroSerial component) should be used in this case.

A module of an application communicates with another device, that is it sends several characters at one time and receives characters from the device. Thus the interrupt mode of the SCI (the AsynchroSerial component) should be used in this case because when a character is received the interrupt is invoked and the underlying process of the application need not check if a character is received. When a buffer for sending is used, the characters are saved into the buffer and AsynchroSerial's service routine of the interrupt sends these characters without additional code of the application.

Note: The polling mode of the component is switched on by disabling of the Interrupt service of the component (AsynchroSerial, AsynchroMaster, and AsynchroSlave).

3.7. Converting Project to Use Processor Expert

The C project that doesn't use Processor Expert can be can be converted to Processor Expert. This is useful when the user finds out that he/she would like to use additional features of Processor Expert.

Warning: Please note that in most cases this conversion involves necessary manual changes in the application code, because for example the register interrupt vectors table definitions created by the user often conflicts with Processor Expert definitions. Don't forget to backup the whole project before the conversion. Some files will have to be removed from the project. The conversion to Processor Expert is recommended to experienced users only.

Conversion Steps

1. Select the menu command **Open Processor Expert for {Project Name}** from Processor Expert menu.
2. Start code generation using **Generate Code {Project name}** menu command from Processor Expert menu, new "Information" window with information message about different prm files reported by Processor Expert may be opened. At this point please ignore this message and go to the "Files" tab on "Project panel" window.
3. Move code from original **main.c** located in "Sources" folder into new **{ProjectName}.c** generated by Processor Expert in previous step, consequently remove origin main.c module from the project.
4. Remove "Includes" folder (including all file within this folder).
5. This step applies only in case that you are using some interrupts within original project. Processor Expert generates complete interrupt vector table (for details please see *3.3.1.1 Interrupt Vector Table*). It´s necessary to transform their original definiton into Processor Expert. For each interrupt vector (for example defined in linker file), add one InterruptVector component configured to ISR of the vector. Remove the original definition.
6. Unfold "Project Settings" and "Linker files" folders and follow one of the followin steps to avoid information message mentioned in first step about different prm files in project:
   a. Option 1: Modify Build options tab on CPU component according to the origin **Project.prm** and remove it from project.
   b. Option 2: Disable generation of prm file by Processor Expert and just copy contain of original **Project.prm** file into **{Project Name}.prm** file.
   c. Option 3: Just remove original prm file and use new prm file generated by Processor Expert. This step is recommended only for converting clean CW projects.
7. Unfold Libs folder and remove I/O map **{Derivative}.c** file, as it will be replaced by PE generated I/O map file.
8. Press "CTRL" + "-" and make project.
9. We recommend to check compiler settings. The following list of compiler options that are safe for Processor Expert projects. Otherwise, we couldn't guarantee the generated code operating properly.
   - -CpuHCS12X
   - -CpuHCS12
   - -Mb
   - -Ms
   - -Ml
   - -D _NO_FLOAT_
   - -Cf
   - -CpPPAGE=0x..

3.8. Low-level Access to Peripherals

In some cases, a non-standard use of the peripheral is required and it is more efficient to write a custom peripheral driver from scratch than to use the component. In addition, there are special features present only on a particular chip derivative (not supported by the component) that could make the user routines more effective; however, the portability of such code is reduced.

Peripheral Initialization

It is possible to use Processor Expert to generate only the initialization code (function) for a peripheral using the Peripheral initialization components. The user can choose a suitable Peripheral initialization component for the given peripheral using the Peripherals tab of the Components Library. See *2.4 Components Library* for details. Initial values that will be set to the peripheral control registers can be viewed in the Peripheral Initialization window. See *2.13 Peripheral Initialization* for details.

Peripheral Driver Implementation

The rest of the peripheral driver can be implemented by the user using one of the following approaches:

- Direct Access to Peripheral Registers

Warning: Incorrect change in registers of the peripheral, which is controlled by any Component driver can cause the incorrect Component driver function.

3.8.1. Direct Access to Peripheral Registers

The direct control of the Peripheral's registers is a low-level way of creating peripheral driver which requires a good knowledge of the target platform and the code is typically not portable to different platform. However, in some cases is this method more effective or even necessary to use (in the case of special chip features not encapsulated within the Embedded component implementation). See *3.8 Low-level Access to Peripherals* for details.

Register Access Macros

Processor Expert defines a set of C macros providing an effective access to a specified register or its part. The definitions of all these macros are in the file **PE_Types.h**. The declaration of the registers which could be read/written by the macros is present in the file **IO_Map.h**.

Whole Register Access Macros

- **getReg{w}** (*RegName*) - Reads the register content
- **setReg{w}** (*RegName, RegValue*) - Sets the register content

Register Part Access Macros

- **testReg{w}Bits** (*RegName, GetMask*) - Tests the masked bits for non-zero value
- **clrReg{w}Bits** (*RegName, ClrMask*) - Sets a specified bits to 0.
- **setReg{w}Bits** (*RegName, SetMask*) - Sets a specified bits to 1.
- **invertReg{w}Bits** (*RegName, InvMask*) - Inverts a specified bits.
- **clrSetReg{w}Bits** (*RegName, ClrMask, SetMask*) - Clears bits specified by ClrMask and sets bits specified by SetMask

Access To Named Bits

- **testReg{w}Bit** (*RegName, BitName*) - Tests whether the bit is set.
- **setReg{w}Bit** (*RegName, BitName*) - Sets the bit to 1.
- **clrReg{w}Bit** (*RegName, BitName*) - Sets the bit to 0.
- **invertReg{w}Bit** (*RegName, BitName*) - Inverts the bit.

Access To Named Groups of Bits

- **testReg{w}BitGroup** (*RegName, GroupName*) - Tests a group of the bit for non-zero value
- **getReg{w}BitGroupVal**(*RegName, GroupName*) - Reads a value of the bits in group
- **setReg{w}BitGroupVal** (*RegName, GroupName, GroupVal*) - Sets the group of the bits to the specified value.

RegName - Register name
BitName - Name of the bit
GroupName - Name of the group
BitMask - Mask of the bit
BitsMask - Mask specifying one or more bits
BitsVal - Value of the bits masked by BitsMask
GroupMask - Mask of the group of bits
GetMask - Mask for reading bit(s)
ClrMask - Mask for clearing bit(s)
SetMask - Mask for setting bit(s)
InvMask - Mask for inverting bit(s)
RegValue - Value of the whole register
BitValue - Value of the bit (0 for 0, anything else = 1)
{w} - Width of the register (8, 16, 32). The available width of the registers depends on used platform.

Example

Assume that we have a CPU which has a PWMA channel and it is required to set three bits (0,1,5) in the PWMA_PMCTL to 1. We use the following line:

```
setRegBits(PWMA_PMCTL,35);           /* Run counter *
```

3.9. Processor Expert Files and Directories

PE Project File

All components in the project with their state and settings and all configurations are stored in one file with extension *.PE*. A name of this file is the same as the project name. The directory where this file is located is called **project directory**. The files that are generated and used in the project are stored relatively to this 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 sub-directory structure within the project directory:

- **\CODE** - the directory containing all source code modules.
- **\DOC** - the directory with the project documentation files generated by Processor Expert.

Please see details on files generated by Processor Expert in *3.5.1 Code Generation*. The individual names of the code and documentation directories can be customized using the options **Project Options | Directory for code** and **Project Options | Documentation**.

User specific configuration

Configuration files with user settings are stored in the directory:
`%USERPROFILE%\ApplicationData\Processor Expert\{version}\`
For example C:\Documents and Settings\john\ApplicationData\Processor Expert\CW08 PE3 02\

User templates and components

User-created templates (see chapter *3.3.5 Creating User Component Templates*) and components (created in Component Wizard, see chapter *5 Component Wizard Description*) are shared by all users and they are stored in the directory:
`%ALLUSERSPROFILE%\ApplicationData\Processor Expert\{version}\`
For example C:\Documents and Settings\All Users\ApplicationData\Processor Expert\CW08 PE3 02\

Desktop File

PE desktop state (opened window with positions, their sizes etc...) and state of the environment configuration options are stored in the file with extension *.dsk*. This file can be stored:

- For each project
- One for all projects in a directory
- One globally - see *User specific configuration* paragraph in this chapter.

These desktop file storage options can be set using **Environment Options | Encode desktop file**.

4. Processor Expert Tutorials

This tutorial is provided for embedded system designers who wish to learn quickly how to use the exclusive features of Processor Expert. Reading this tutorial may be all you need to start using Processor Expert for your own application.

The following tutorials are available:

4.1. Tutorial Project 1 for Freescale HCS12 Microcontrollers

A simple tutorial is available within the CodeWarrior tutorials. This animated tutorial describes a periodically blinking LED project. The LED is connected to one pin of the CPU and it is controlled by a periodical timer interrupt.
Please follow these instructions to start the tutorial:

- Invoke the CodeWarrior startup dialog by starting the CodeWarrior or select the **File | Startup dialog** main menu command.
- Click on the **Run Getting Started Tutorial** button.
- Select **Processor Expert Tutorial** in the tutorials contents.

4.2. Tutorial Project 2 for Freescale HCS12 Microcontrollers

This tutorial describes a demo project of a simple LED controller. The LED controller has two color LEDs - a red and a green one - and one command button.

How it works

The button sends commands (external interrupts) to CPU through one pin and the CPU switches the red and green LEDs lights on or off. If you press the key you can see that lights of two LEDs have been changed. One of them is switched off and the other one is switched on. At the beginning the green LED is on and the red one is off.

Minimal required hardware design

In the demo application the following components are used:

1. CPU MC9S12GC128 Freescale processor (HCS12 family)
2. Red LED - connected to CPU output pin PB0_ADDR0_DATA0
3. Green LED - connected to CPU output pin PB1_ADDR1_DATA1
4. Button - connected to CPU input pin PH0_KWH0_MISO1

Components

This simple demo-project uses the following components:

1. MC9S12GC128 - CPU component (Freescale HCS12 processor family)
2. BitIO - General 1-bit input/output component - outputs to LEDs. The LEDs receive 1 bit data which specifies whether the light should switch on or off (value 0 = switch off, value 1 = switch on).

3. ExtInterrupt - External Interrupt component - interrupt from button. Pressing the button calls an external interrupt which switches the state of the LEDs (on/off).

steps

There are step-by-step instructions how to create this tutorial project. This tutorial goes through the following steps:

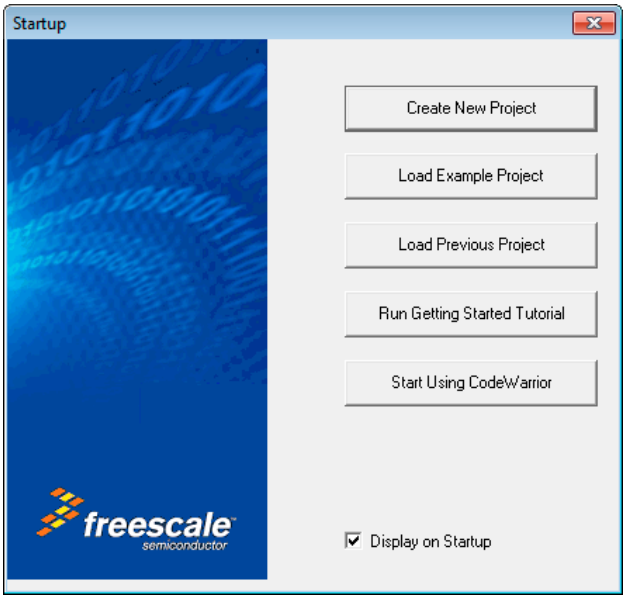
1. Creating a new project
2. Adding components
3. Generating Code
4. Adding the On-Event Code

Note: This demo project does not care about non-defined states on the output of the key during the process of key pressing. This may result in the fact that state of two LEDs stays apparently unchanged.

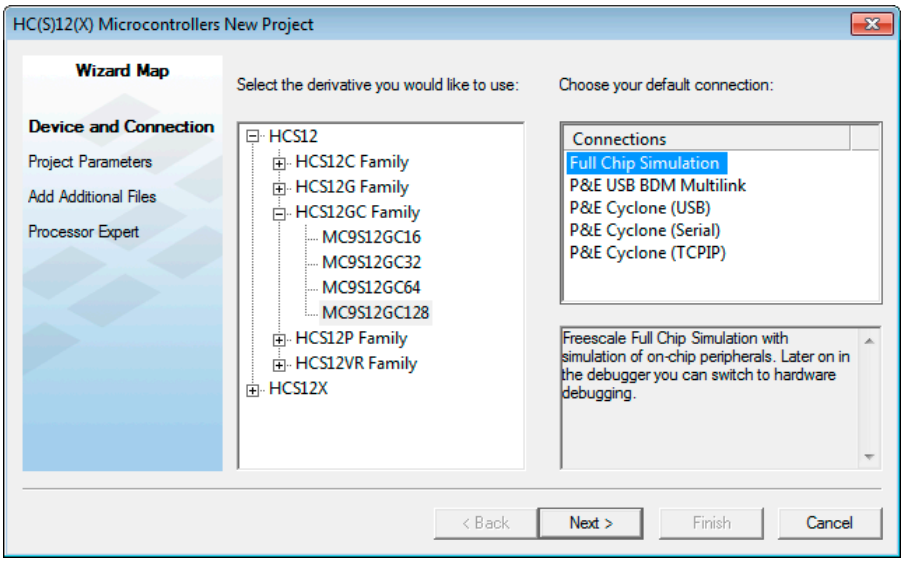
4.2.1. Tutorial for Freescale HCS12 Project 2 Step 1

Creating a New Project

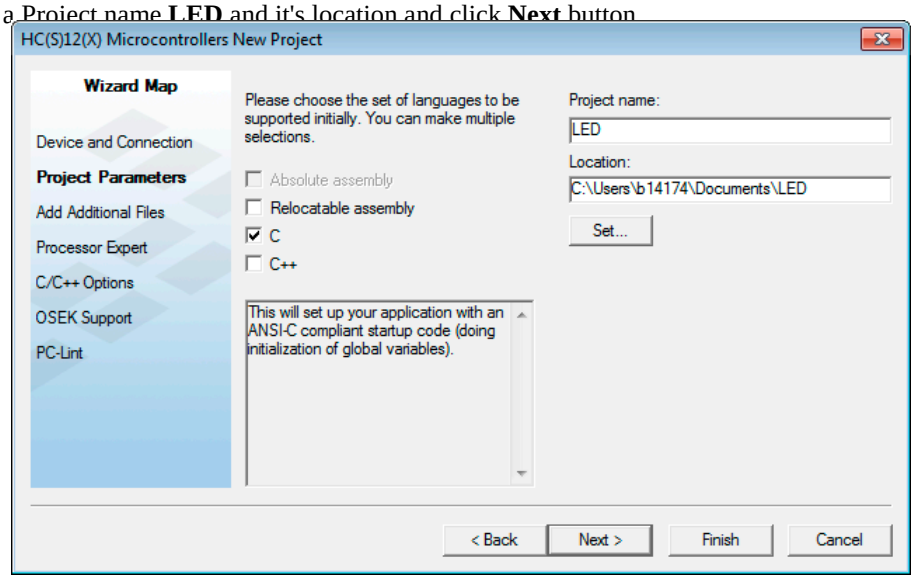
1. Click on the **Create New Project** button in the start-up screen. If the CodeWarrior is already running, invoke the project wizard using the **File - New Project...** menu.



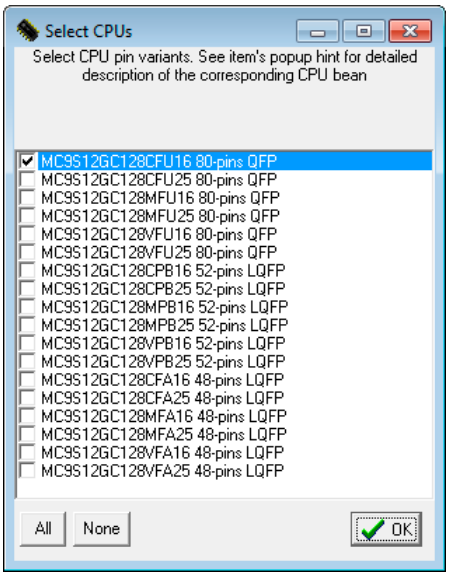
2. In the New Project Wizard window choose CPU, select a hardware connection if you have some and click on the **Next** button.



3. Specify a Project name **LED** and it's location and click **Next** button



4. Choose **Next** to skip adding external files.
5. Select **Processor Expert** in the Rapid Application Development options and click on the **Finish** button.
6. Confirm *Select CPUs* dialog by clicking on **OK**.



Now the new empty project is created and ready for adding new components.

Next step

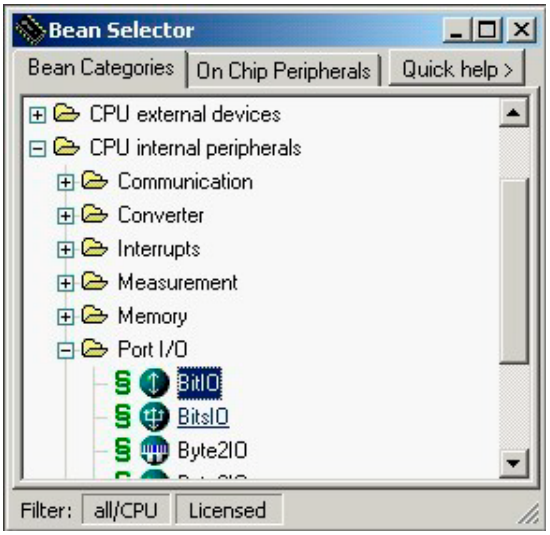
Go to Step 2 - Adding components to the project.

4.2.2. Tutorial for Freescale HCS12 Project 2 Step 2

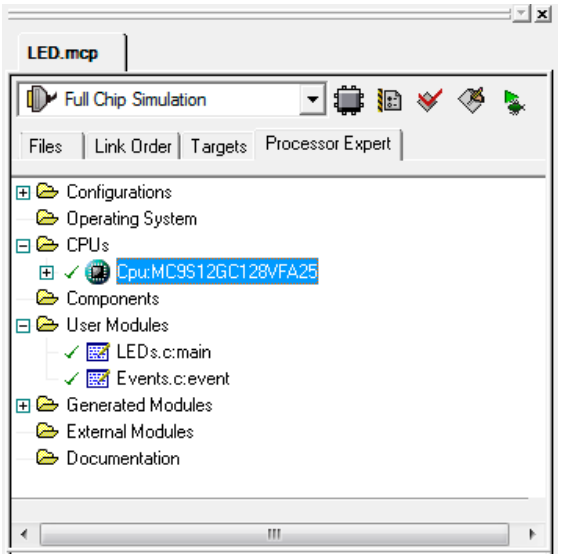
Adding Components to the Project

In the context of the LED controller, you will add two **BitIO** components for the Red LED and the Green LED, and an **ExtInt** component for the Button.

1. If the Components Library window is not already opened, open it using menu **Processor Expert > View > Components Library**
2. Double click the **BitIO** component in the folder PortI/O (subfolder of CPU Internal Peripherals folder) in the Components library window.

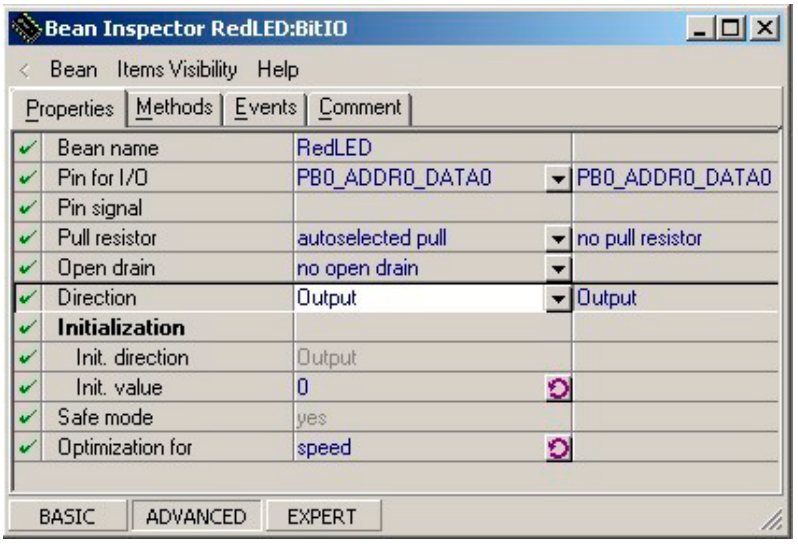


3. If you will be asked whether to enable the component in all configurations. Choose **Yes**. *(This dialog box could be enabled or disabled in environment options.)*
4. New component is added to the project. Switch to Processor Expert project tab in codewarrior's project panel. (See picture below). *Don't worry about red exclamation mark beside a new component. It means that error is present - component has not been set-up properly yet.*



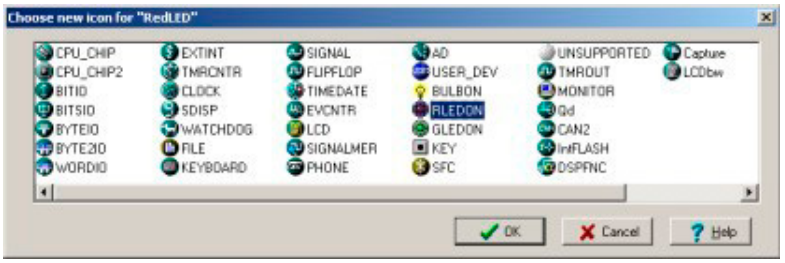
5. Open the **Component Inspector** window by double click on the new component (Bit1:BitIO) in Processor Expert project panel (if the Component Inspector hasn't been opened automatically). *Automatical opening of the component inspector is influenced by environment settings (See 2.1.1 Processor Expert Options for details.)*
6. Select **Items visibility** > **Advanced view** in the pop-up menu of the Component Inspector window in order to display detailed settings of the component. It is necessary for the following steps. See picture below.

7. Using Component inspector set the component properties as follows:
   - **Component name:** type **RedLED** into the edit box.
   - **Pin for IO:** select **PB0_ADDR0_DATA0**
   - **Direction:** select **Output**



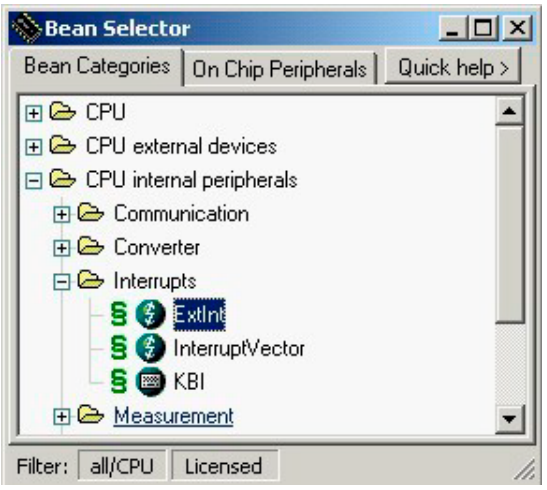
8. To setup generation of **methods** click on the Methods TAB and set all methods to "don't generate" and NegVal method to 'generate code'. See the following picture:

9. click the **Change component icon** item of the *Component* menu in order to choose a new icon in the list. Select the **RLEDON** icon and click the **OK** button. See the following pictures:

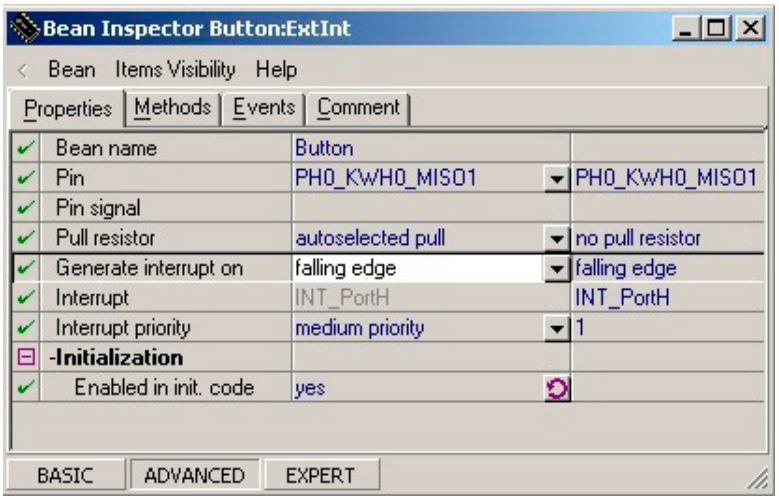


10. Using the procedure previously described for the red LED (steps a,b,c,d) add the **green LED** component to the project. The difference from the redLED component is in the component's properties (pin, name and initial value).
    Set the component properties as follows:
    - **Component name:** type **GreenLED** into the edit box.
    - **Pin for IO:** select **PB1_ADDR1_DATA1**
    - **Direction:** select **Output**
    - **Init. value:** select **1**

11. Click on the Methods TAB and switch all methods to "don't generate" and NegVal method to "generate code" using button on the right.
12. Click the **Change component icon** item of the **Component** menu in order to choose a new icon. Select the **GLEDON** icon and click the **OK** button.
13. Now is time to add component handling **the button**. Open the **Components Library window** again and double click the **ExtInt** icon on the folder Interrupts in the in order to add the component to the project. See picture below:

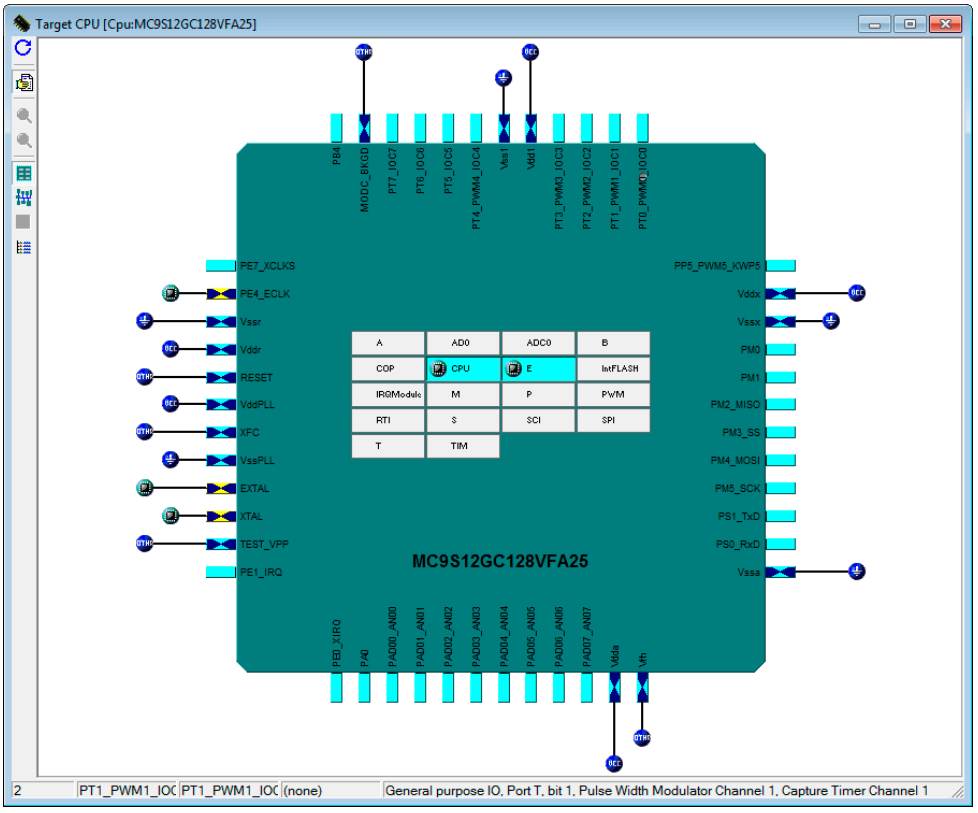


14. Open the **component inspector** for the new component (double click it in the project panel) and set the component properties as follows:
    - **Component name:** type **Button** into the edit box.
    - **Pin:** select **PH0_KWH0_MISO1**
    - **Generate interrupt on:** click the option in order to display the options. Select the **falling edge**.

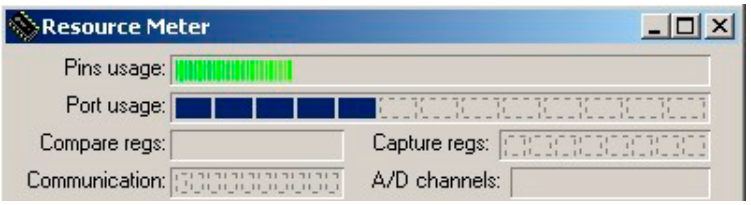


15. Click on the Methods TAB and set **all methods to "don't generate"**.
16. Click the *Change component icon* item of the *Component* menu in order to choose a new icon in the list. Select the **KEY** icon and click the **OK** button.

You can see in the Target CPU window which pins of the chip are handled by the components. You can easily identify LED components by their specific icons. If the Target CPU window is not opened, use menu command ***Processor Expert > View > Target CPU package.***



After adding all components, click ***Processor Expert | View | Resource Meter*** *in order to open the Resource Meter window and see remaining available resources of the chip.*



Next step

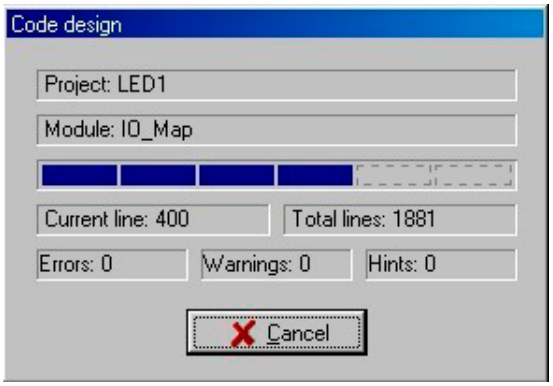
Go to Step 3 - Code Generation.

4.2.3. Tutorial for Freescale HCS12 Project 2 Step 3

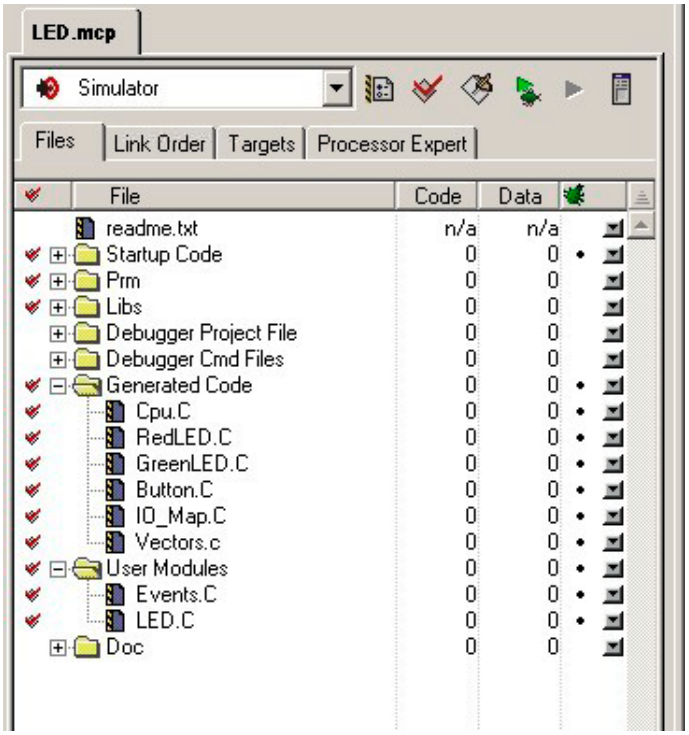
Code Generation

1. Click on the command **Processor expert > Generate Code 'Led.mcp'** in the CodeWarrior main menu in order to run the code generation process
   The code generation window shows the current state of code generation.

   Note: *There shouldn't be any errors in the Error window before code generation.*



2. The code generation process generated all source files from components to the "Generated Code" folder in the CodeWarrior project window. The other modules can be found in the "User modules" folder in the CodeWarrior project window. *The generated code is inserted only into the selected target in the CodeWarrior project window.* See the picture below.



Next Step

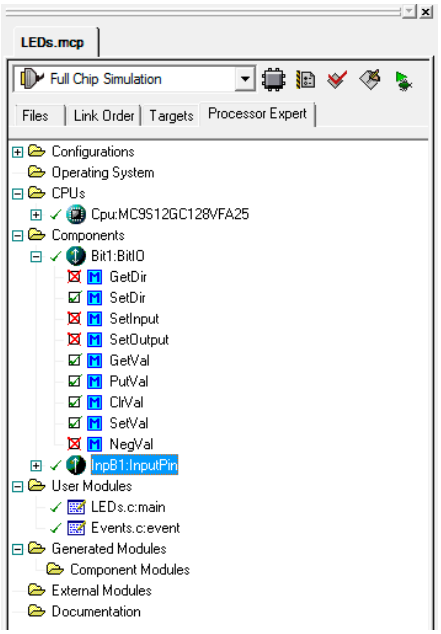
Go to Step 4 - Adding On-Event Code

4.2.4. Tutorial for Freescale HCS12 Project 2 Step 4

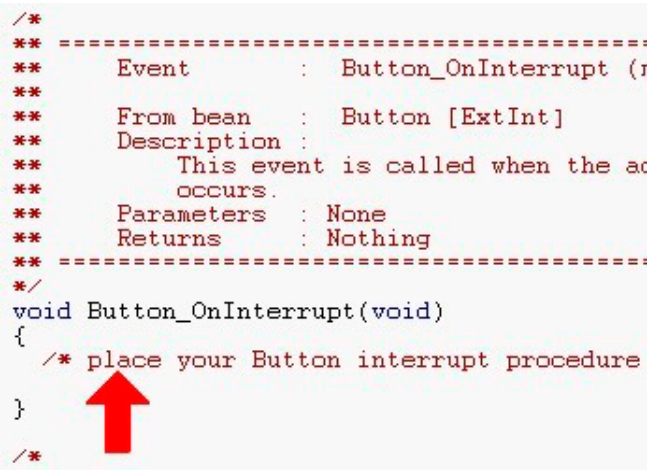
Adding the On-Event Code

1. Switch to the Project panel (Processor expert tab in the CodeWarrior project panel). All the components (including CPU components) in the project panel are organized in a tree. You may expand and collapse them by clicking on the plus "+" or minus "-" sign. Component's events and methods are present as a subnodes of the component node.
   Note: By double-clicking on any event/method icon, you change its ***enable/disable*** *state (you can do it also in the component inspector). You need to invoke code generation again to generate code according to the new settings.*

2. Click the "+" sign to expand the component and display its events and methods.



3. Double-click the **OnInterrupt** event from the *Button* component to open and find out the position of this event in code. See the picture below.



4. Enter the following lines to the body of the Button_OnInterrupt function:
   RedLED_NegVal ();
   GreenLED_NegVal ();

   The first line is a call of the method NegVal of the component RedLED. The second line is a call of the method NegVal of the component GreenLED. Calling syntax of all component's methods is 'ComponentName'_'MethodName'();

5. Finally, to create a binary executable file click on "Make" icon in the CodeWarrior Project Window.

5. Component Wizard Description

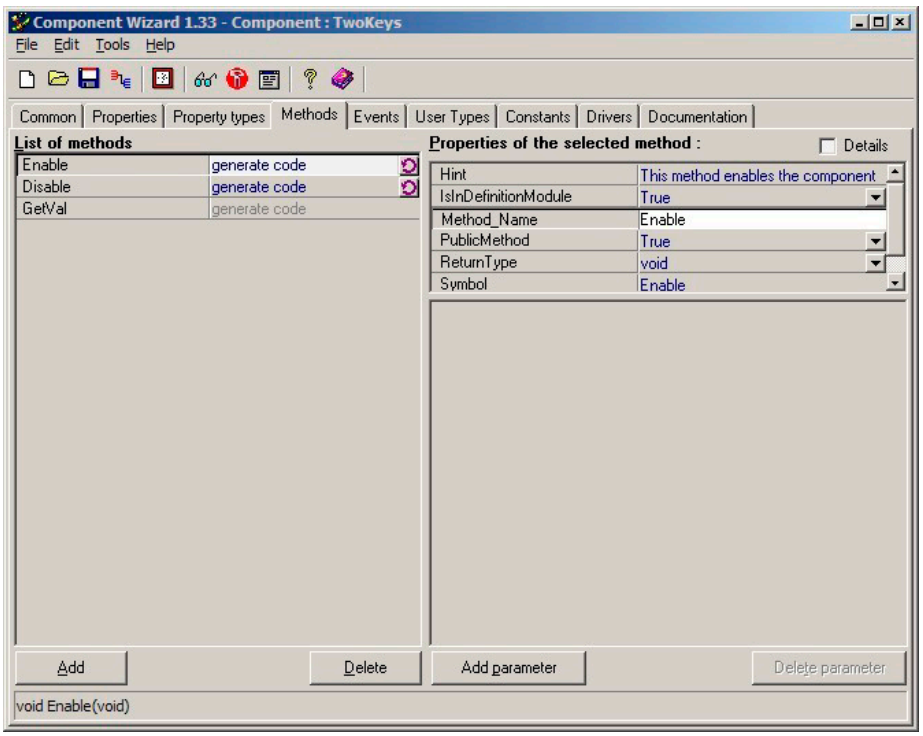
Component Wizard is a tool dedicated to the creation and edition of Embedded Components. See *3.2.1 Embedded Components* for details. It provides a powerful interface for the composition of new components, and generates the component files.

Using Component Wizard, the user can create new components very quickly and easily without errors in the generated files. The user needs only to determine the Properties, Methods and Events and write the necessary implementations of the methods and events.

Component Wizard facilitates the reusability of existing Components and helps edit the source code (quick location, editor, ...).

Component Wizard is an **external tool**, not a part of Processor Expert.

For further information, see Component Wizard Help.



INDEX