Using the Device Initialization and Processor Expert with CodeWarrior™ Development Studio for ColdFire® Architectures v7.0

By: Alfredo Soto and Oscar Gueta

Introduction

This document demonstrates using Processor Expert embedded beans for a ColdFire microcontroller (MCU) to integrate peripheral initialization and functionality on 32-bit ColdFire MCUs and provides an overview of the Processor Expert development tool with CodeWarrior Development Studio for ColdFire Architectures. Processor Expert is a tool that helps reduce development time for embedded software creation and can reduce time to market for new products.

Freescale Semiconductor's MCUs are at the heart of countless applications, and to help designers maintain a competitive advantage, CodeWarrior Development Studio for ColdFire Architecture development tools supporting these MCUs are now integrated with Processor Expert, providing engineers with an integrated development environment (IDE) for design, implementation, verification and optimization of embedded 32-bit microcontroller-based applications. This combination allows efficient use of the MCU and its peripherals, enabling building of portable solutions and saving development time and cost. The CodeWarrior development tool helps you reuse and speed the application development through the board portfolio of the ColdFire Architectures.

Processor Expert and Device Initialization Comparison

Freescale CodeWarrior offers two plug-ins for rapid application development: Processor Expert and Device Initialization. Both tools have many advanced features that accelerate the development cycle.

Table 1 Features Comparison

Feature	Processor Expert	Device Initialization
Easy to use graphical IDE	Yes	Yes
Interactive design specifications of Freescale ColdFire MCUs	Yes	Yes
Generated Code	Peripheral Drivers	Initialization code only
Generated Code language	C	C
Peripheral Init Beans	Yes	Yes
Low-level beans	Yes	No
High-level beans	Yes	No
Project configuration	Yes	No
User-friendly linker parameter file configuration	Yes	No
Generated code changes tracking	Yes	No
User beans creation	Yes	No

Device Initialization

Device Initialization provides a fast and easy way to configure and generate initialization source code for the central processor unit (CPU). It contains only one set of beans: Peripheral Initialization Beans. You can generate the code initializing the peripheral in C.

Features and Benefits of Device Initialization

The Device Initialization tool features the following:

- Graphical user interface with CPU package, peripherals and pins
- User-friendly access to initialization setup of CPU peripherals
- Initialization code generator
- C format for generated code
- Built-in detailed design specifications of Freescale CPUs

The Device Initialization tool provides these benefits:

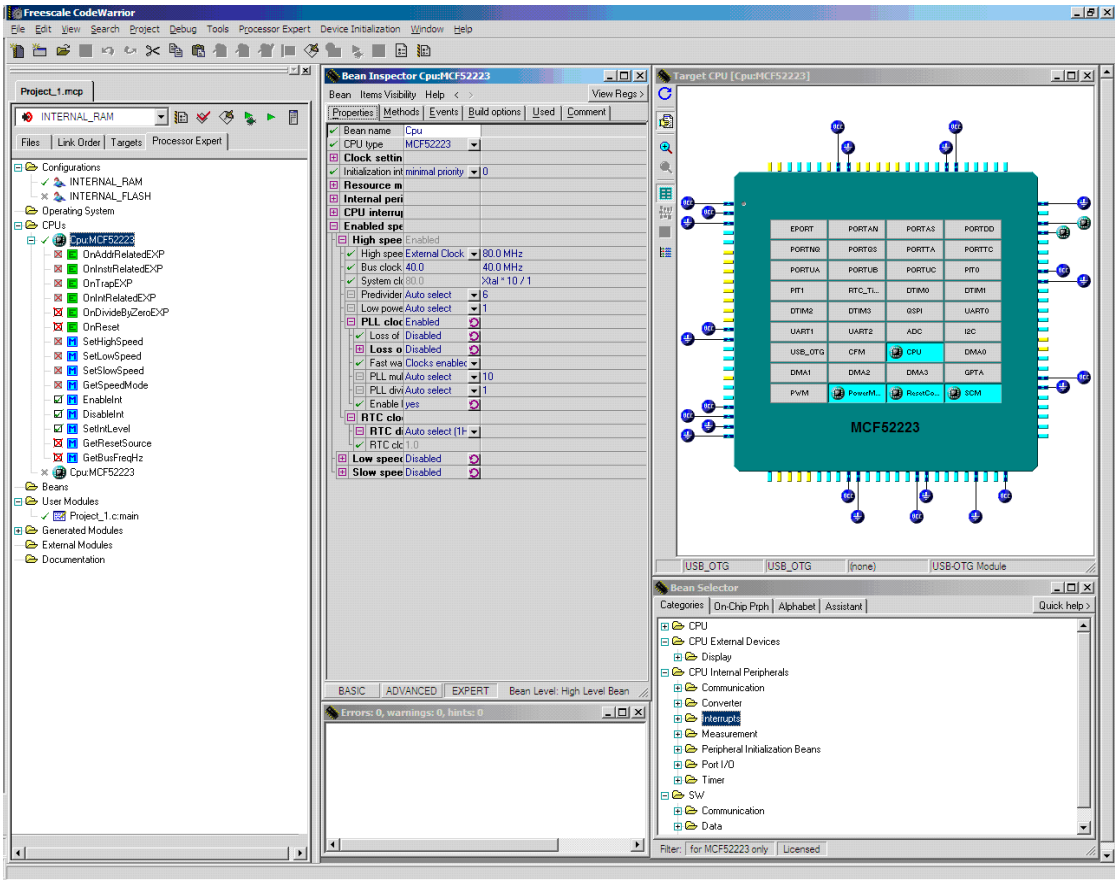
- Easy to learn design environment
- Possibility for reusing the individual peripheral setup in other designs
- No need to generate code to see the resulting peripheral control register values.

Processor Expert and Embedded Beans

Processor Expert was originally developed as a stand-alone product. Now, to provide a more efficient and comfortable development environment, it has been integrated as an optional software plug-in for Freescale CodeWarrior Development tools. Processor Expert provides graphical interface for quick application development, in an object-oriented manner, for embedded systems. The Processor Expert plug-in generates code from the Embedded Beans, and the CodeWarrior software manages the project files and the compilation and debugging processes. Use the graphical user interface (GUI) within the CodeWarrior Integrated Development Environment (IDE) to configure MCU peripherals, generating the initialization and other user support code.

The CodeWarrior IDE menu contains a new menu item named Processor Expert. Figure 1 shows CodeWarrior IDE workspace with the enabled Processor Expert function. It shows the project manager, bean selector, error window, bean inspector, and CPU.

Figure 1 CodeWarrior IDE with Processor Experts Workspace



Processor Expert Benefits

Processor Expert uses an object-oriented application that builds methods using embedded beans. The embedded beans read the MCU hardware and register details into an intuitive software Application Programming Interface (API). Embedded beans provide a software API and graphical interface to initialize the MCU.

An expert system works in the background ensuring that the MCU settings and configurations do not conflict with each other. The Processor Expert software API and the expert system allow an application to be portable among third-party MCU processors and Freescale MCU processors. Apart from reusing code, other benefits include:

- Easy to program and set up CPU/MCU peripherals with limited knowledge
- Interface allows you to configure modules in real-world terms, such as baud rates
- Provides ready-to-use hardware drivers for peripherals
- Ability to create user-defined embedded beans
- Automatic code generator, which creates tested, optimized C code tuned to the application needs and the selected Freescale MCU
- Design-time settings verified by the expert knowledge system
- Allows the use of external code, libraries, and modules

What is an Embedded Bean?

The ready-to-use embedded beans are tested building blocks for application creation. Embedded beans abstract embedded programming by providing a unified API across platforms and hiding the implementation details. That way, if and when the hardware implementation changes, there is no need to change the API functions. This characteristic makes the application portable.

Embedded Beans encapsulate the initialization and functionality of an embedded system's basic elements, such as CPU core, CPU on-chip peripherals, FPGAs, stand-alone peripherals, virtual devices and pure software algorithms. The embedded beans link functionalities into properties, methods, and events:

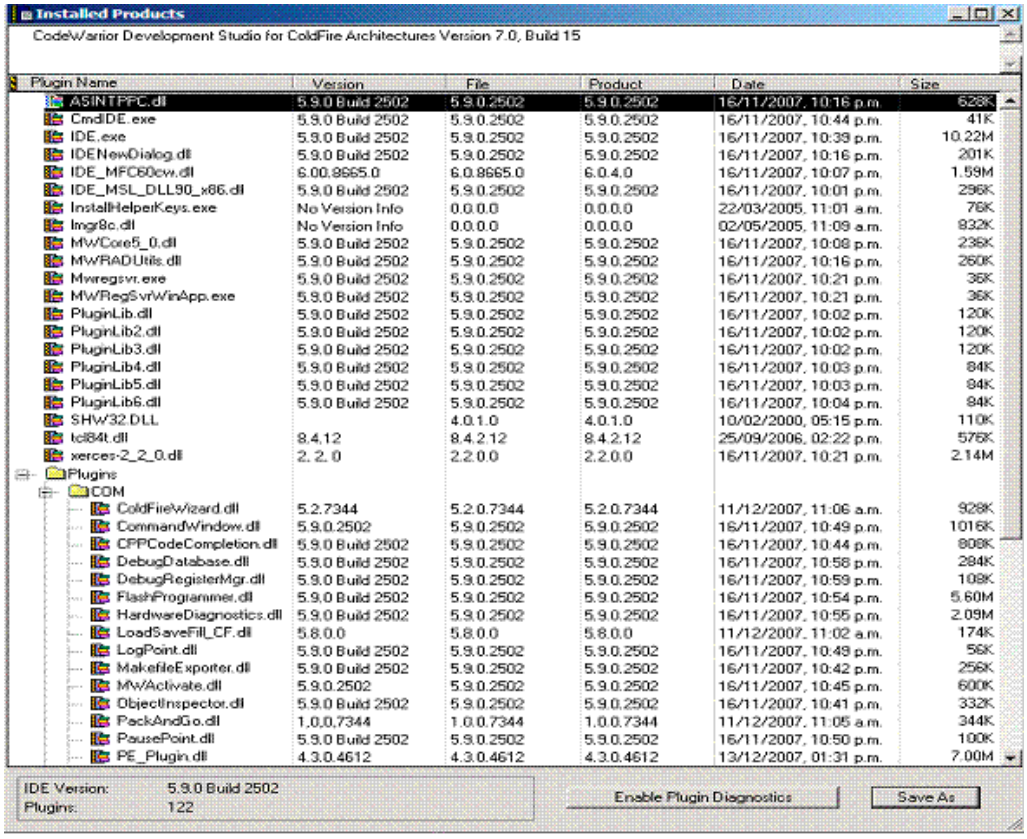
- Properties - During the application design-time, you define the behavior of the embedded beans and then compile. These behaviors include MCU initialization settings such as speed of serial line, time period of the periodical interrupt, or number of channels of A/D converter. (Memory allocations or external crystal speed cannot change during run-time.)
- Methods - These embedded bean behavior attributes can be modified during the application runtime. These behaviors include receiving serial characters, changing the SCI baud rate, or driving/reading a pin value. When modifying an attribute during run time the embedded beans provide the programmer with functions to insert in the code.
- Events - These embedded beans provide function calls when important changes happen in the bean (i.e., interrupts, received character via serial line, analog value measured, etc.)

In this document our main purpose is to provide the basic steps to use Processor Expert. We describe several uses of the embedded beans that allow users to develop real applications with this powerful tool. We also explore properties, methods and events to demonstrate how they can be used in many types of applications.

Environment Setup

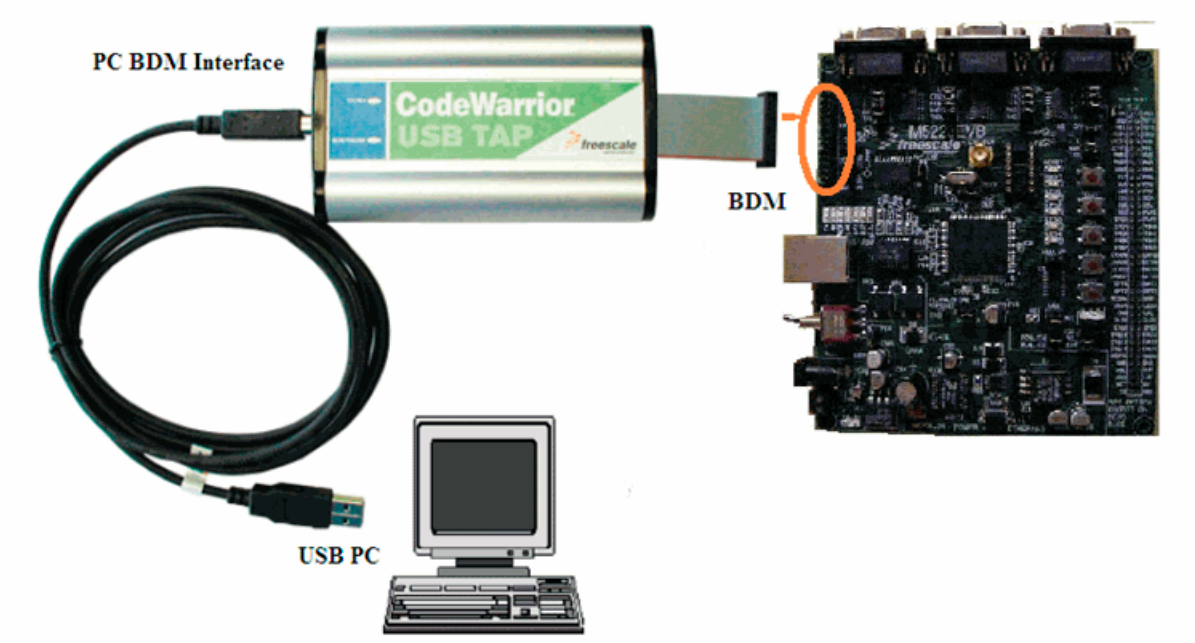
We developed and tested the application example provided in this document using the CodeWarrior Development Studio for ColdFire v7.0 and Processor Expert running on Windows® XP. Figure 2 shows the version information for these tools.

Figure 2 CodeWarrior IDE version



Other development components include a terminal program to display UART data via the serial port and a USB BDM pod to program the MCU. Figure 3 shows the connections required to program the MCU using the CodeWarrior IDE and the BDM programmer.

Figure 3 Development Environment Debugger/ Programmer Connections



Creating a Project with Device Initialization

This section explains the creation of a new project with the Device Initialization tool. It shows how to configure the required peripheral using graphical user interface, generate the initialization code and test the functionality on a simple application. The demonstration is based on a ColdFire M52223EVB board.

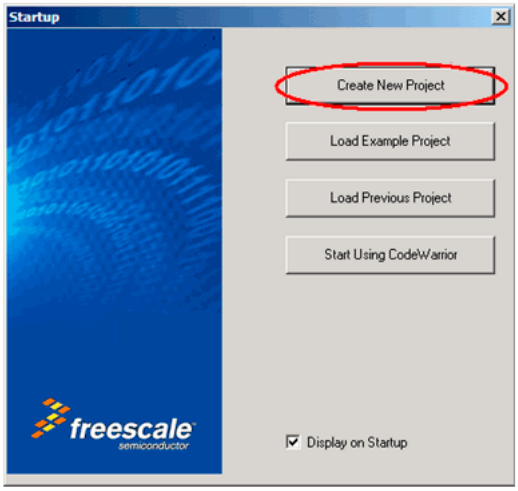
Creating an Empty Project

You can create a CodeWarrior project using Device Initialization and Processor Expert with just a few clicks:

1. Begin the project by opening CodeWarrior™ Development Studio for ColdFire v7.0 or later. The Startup dialog box appears (see Figure 4).

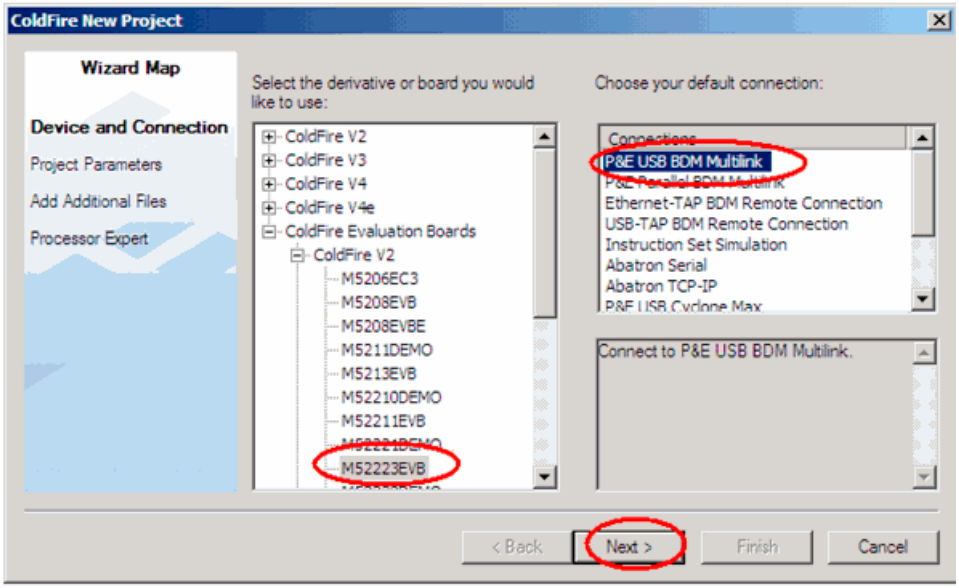
NOTE If the Startup dialog does not appear, go to the **File** menu and select **Startup Dialog**.

Figure 4 Startup Dialog Box



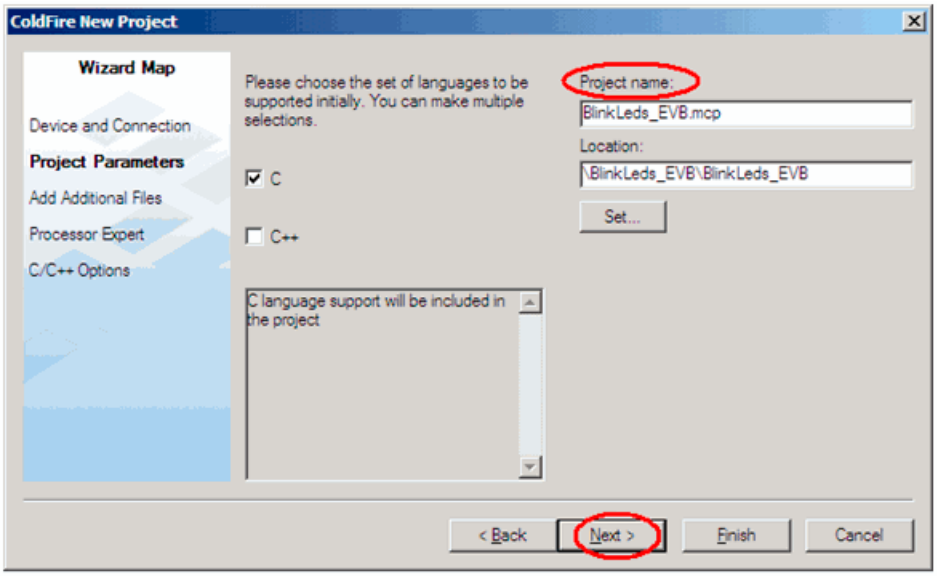
2. Click the Create New Project button.
3. For the ColdFire family MCUs, expand ColdFire Evaluation Boards
4. Choose one MCU. (This example uses the M52223EVB.)
5. Choose the default connection. (For this example we chose P&E USB BDM Multilink.)

Figure 5 MCU Device and Connection Dialog Box



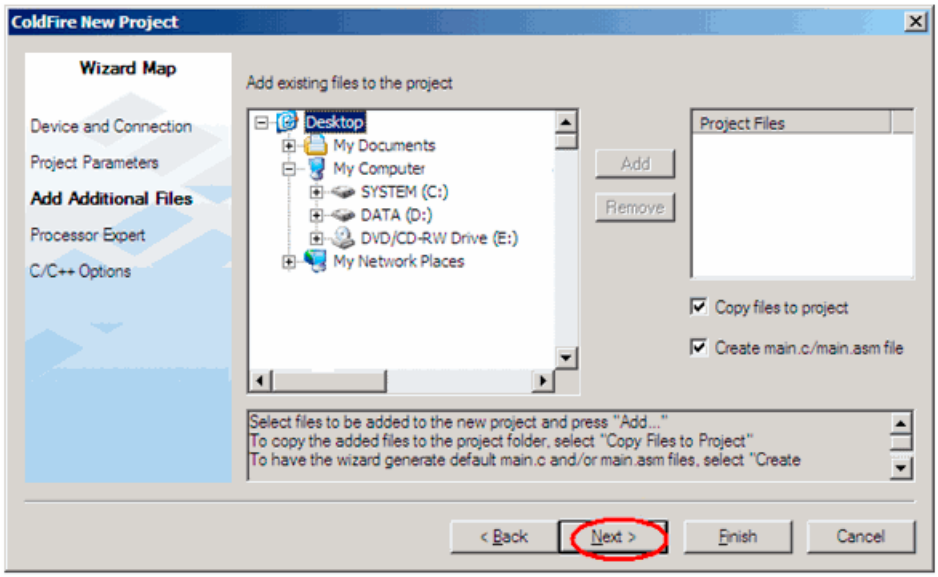
6. Click **Next** — The Project Parameters dialog box appears
7. In Project Parameters dialog box enter a project name: `BlinkLeds_EVB.mcp`

Figure 6 MCU Project Parameter Dialog Box



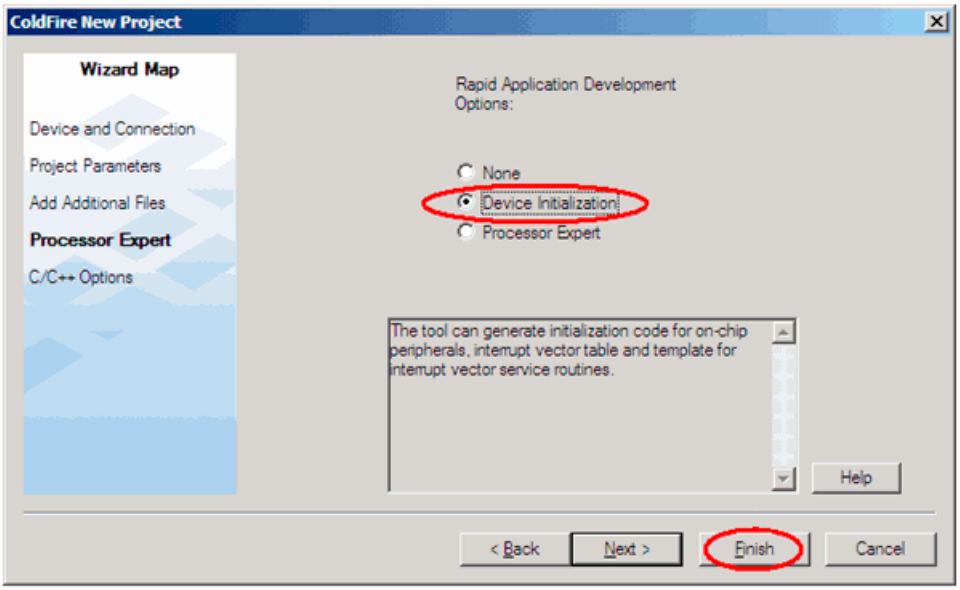
8. Click **Next** — The Add Additional Files dialog box appears

Figure 7 Add Additional Files Dialog Box



9. No additional files are required, so click **Next** — The Processor Expert dialog box appears.

Figure 8 Processor Expert Dialog Box



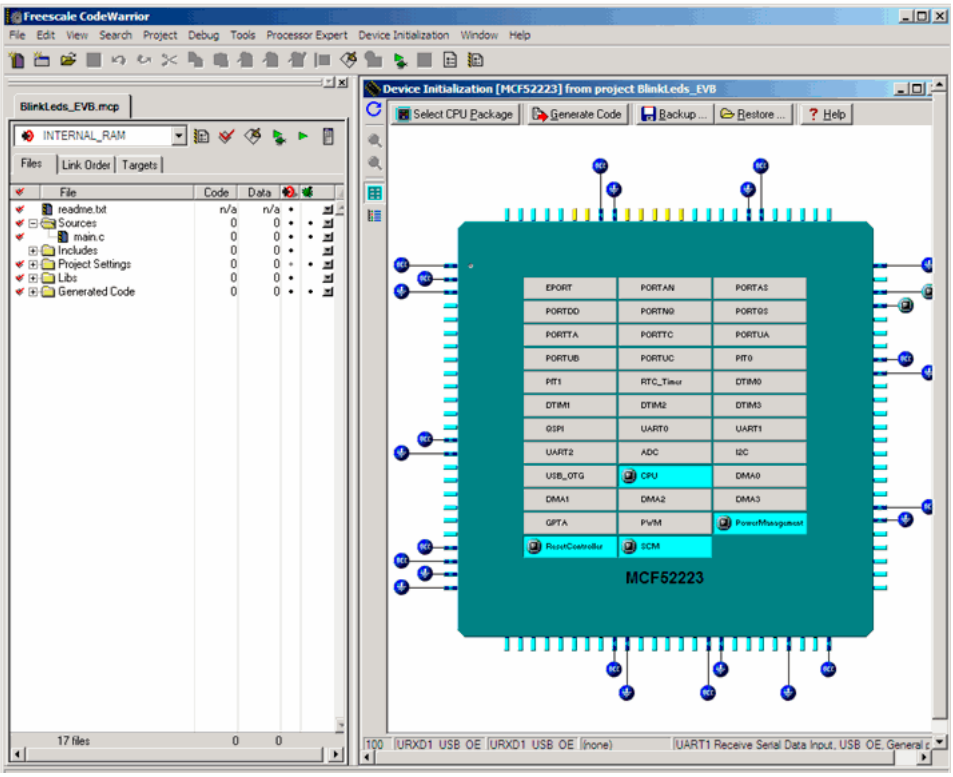
10. Select Device Initialization option.
11. Click **Finish** — The software creates the new project.

Example 1: Device Initialization Project

This example uses the project created in the previous section to show how to configure the required peripheral using the GUI, generate the initialization code, and test the functionality on a simple application.

Once you create a new project, the Device Initialization window displays the CPU view with the peripherals available on the selected package.

Figure 9 Project with Device initialization Window

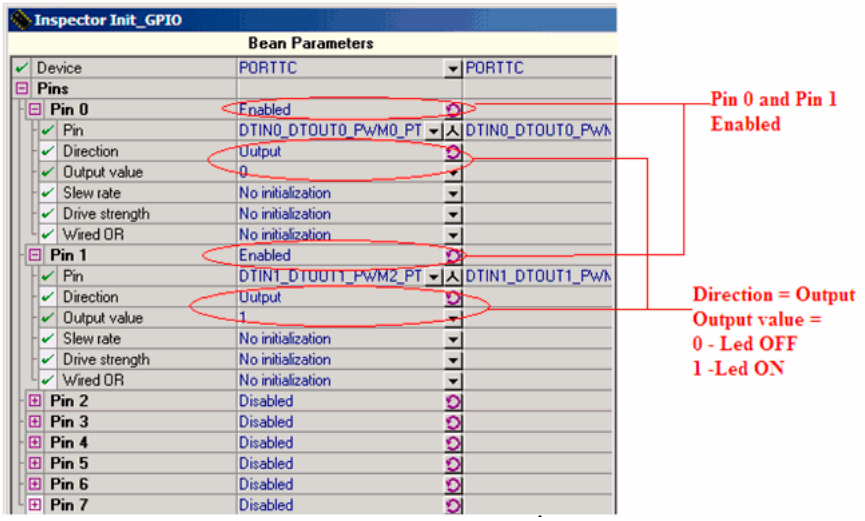


This example toggles the output level of the GPIO pin DTIN0_DTOUT0_PWM0_PTC0 and DTIN1_DTOUT1_PWM1_PTC1 connected to LED1 and LED2 on the board with a period of 1 second.

NOTE For this example, we used the M52223EVB Rev. B.

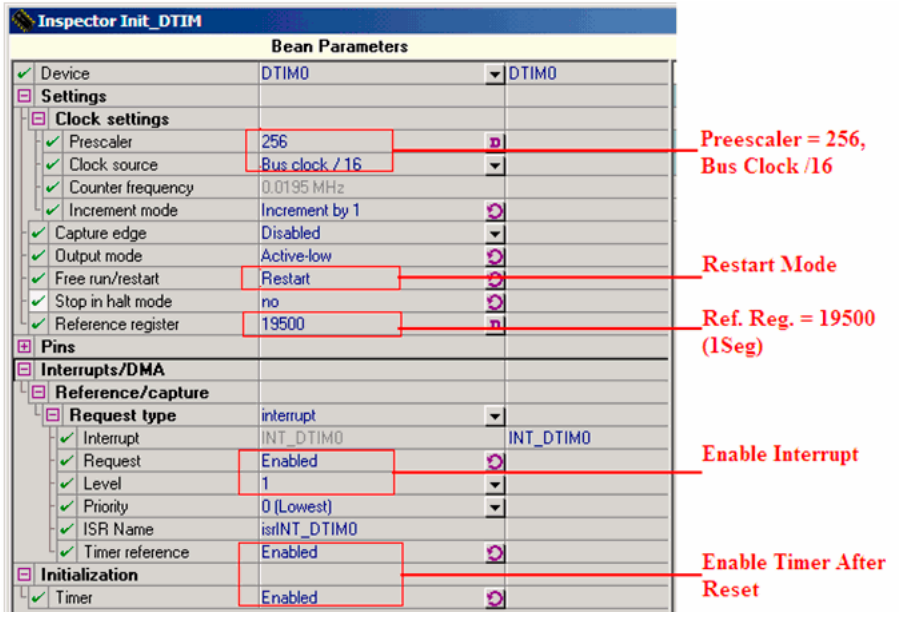
1. Click on the PORTTC peripheral to enable peripheral initialization and view the Peripheral Inspector.
2. Use the Inspector to set up and modify the settings.
3. Click **OK**.

Figure 10 Configuration of PORTTC



4. Click on the DTIM0 peripheral to enable peripheral initialization and view the Peripheral Inspector.
5. Use the Inspector to modify the settings.
6. Click **OK**.

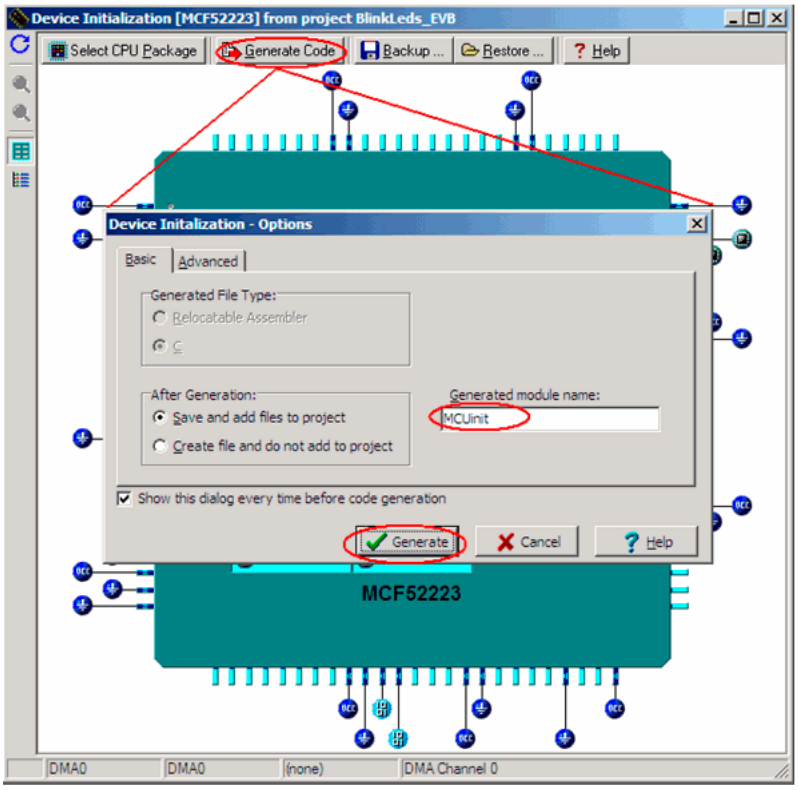
Figure 11 Configuration of DTIM0



7. Invoke code generation by clicking on the Generate Code button.

   The dialog box offers several options that can be applied while generating the code. The Device Initialization plug-in creates an initialization (`MCUinit.c`) module that is easily accessible using the Generated Code folder in the Project Panel (the file is optionally added into the project).

Figure 12 Generating the Source Code



8. At this point, the project contains a folder called Generated Code. Open the `MCUinit.c` file and write the following code to the pre-generated DTIM0 interrupt service routine:

```
__declspec(interrupt) void isrINT_DTIM0(void)
{
/* Clear interrupt flags of DTIM0 device - DTER0: REF=1,CAP=1 */
DTER0 = 3;
/* Invert the output value of the pin DTIN0_DTOUT0_PWM0_PTC0 and
DTIN1_DTOUT1_PWM1_PTC1 */
PORTTC ^= 3;
}
```

9. Build and run the project:
   a) From main menu bar, select **Project > Make** — The IDE updates the files and links the code into the application.
   b) From main menu bar, select **Project > Debug** — The IDE builds (assembles, compiles, and links) the project, and the Debug Window appears
   c) From main menu bar, select **Project > Run** — The debugger downloads and runs the program
   d) From main menu bar, select **Debug > Kill** — The debug session ends; you may close all open windows

   At this point, this project makes LED1 and LED2 on the M52223EVB switch between ON and OFF approximately once per second.

Creating a Project with Processor Expert

This section explains how to create a new CodeWarrior project using Processor Expert and gives a brief explanation of configuring the selected microcontroller CPU. The discussion focuses on the major steps of the application development including:

- Starting a project with Processor Expert
- Adding embedded beans to the project
- Resolving bean errors identified by the Processor Expert knowledge system and configuring the embedded bean properties, methods, and events.
- Programming the CPU and demonstrating the applications

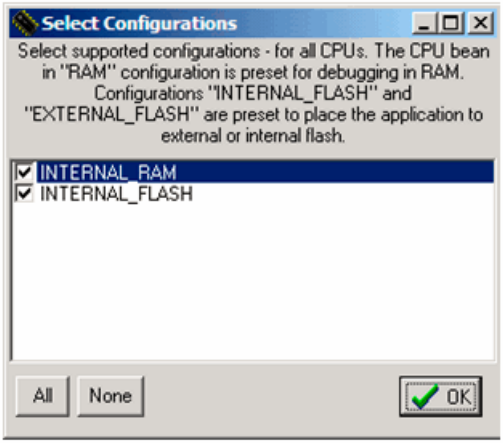
Example 2: Timer and I/O with Processor Expert

This particular example shows the creation of a project with the Processor Expert tool, configuring the required peripherals using high-level beans, generating the code and testing the functionality on a simple application. The example runs on ColdFire M52223EVB board.

This example uses a timer and an I/O port to make an LED blink. The timer generates a periodic interrupt that controls the LED state with an I/O port. Every time an interrupt occurs, the LED state changes.

1. Create a new project by following the directions in Creating an Empty Project. For this example name the project `BlinkLed_PE1.mcp`.
2. In the Processor Expert window, select the Processor Expert option.
3. Click **Finish** — The Select Configurations window appears (see Figure 13).
4. Specify whether you are creating this project for Internal RAM, Internal Flash, or both.

Figure 13 Select Configurations Window

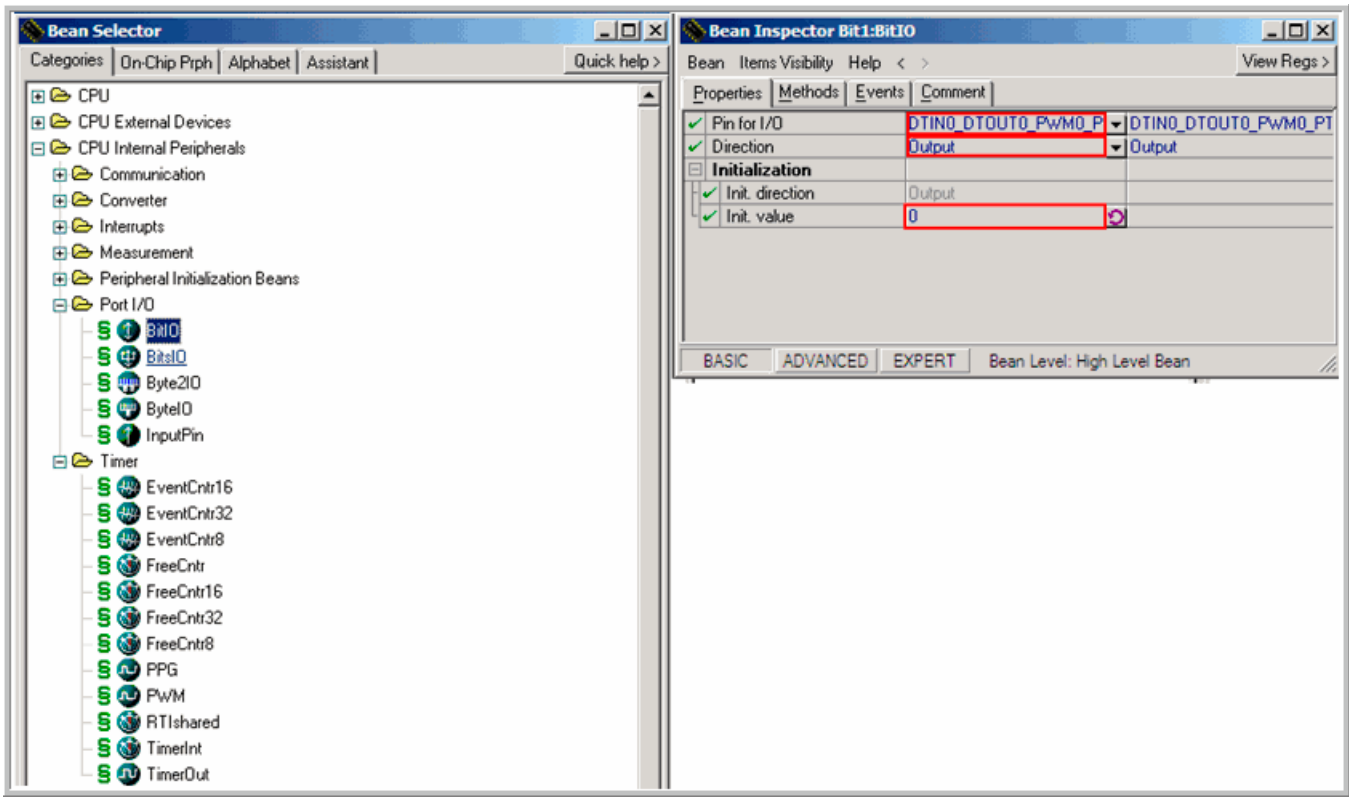


5. Click **OK**. The Bean Selector window appears and displays the list of available beans on the selected CPU.
6. Add the BitIO bean from the Bean Selector window to the project
7. Enable it in all configurations (use the **Yes** button in the confirmation dialog).
8. Use the Inspector to set up the project (Figure 14).

   To configure the bean, select the Properties tab and modify the properties values. Below is a list and description of the most important properties for the BitIO bean:

   - Pin for I/O — MCU pin used by this bean.
   - Direction — Direction of the bean: input, output, input/output. You can switch the direction of the pin in run time.

Figure 14 Bean Selector and Configuration of BitIO in the Bean Inspector



9. On the **Methods** tab in the Bean Inspector, set **NegVal** to **Generate Code**.

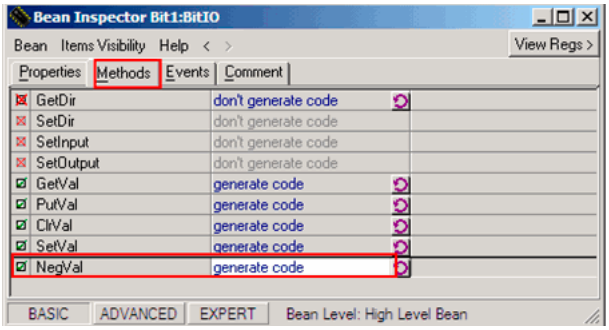
   Setting NegVal to Generate Code inverts the output level of the configured pin. This project uses the NegVal method from the BitIO bean to invert the LED state, it is not generated by default. Figure 15 shows the Methods tab with the changes proposed for the LED Blink Project.

10. If needed, select which Methods to create by selecting the Methods tab in the Bean Inspector window (Figure 15).

11. Click the round arrow button. For this bean, these methods are most important:

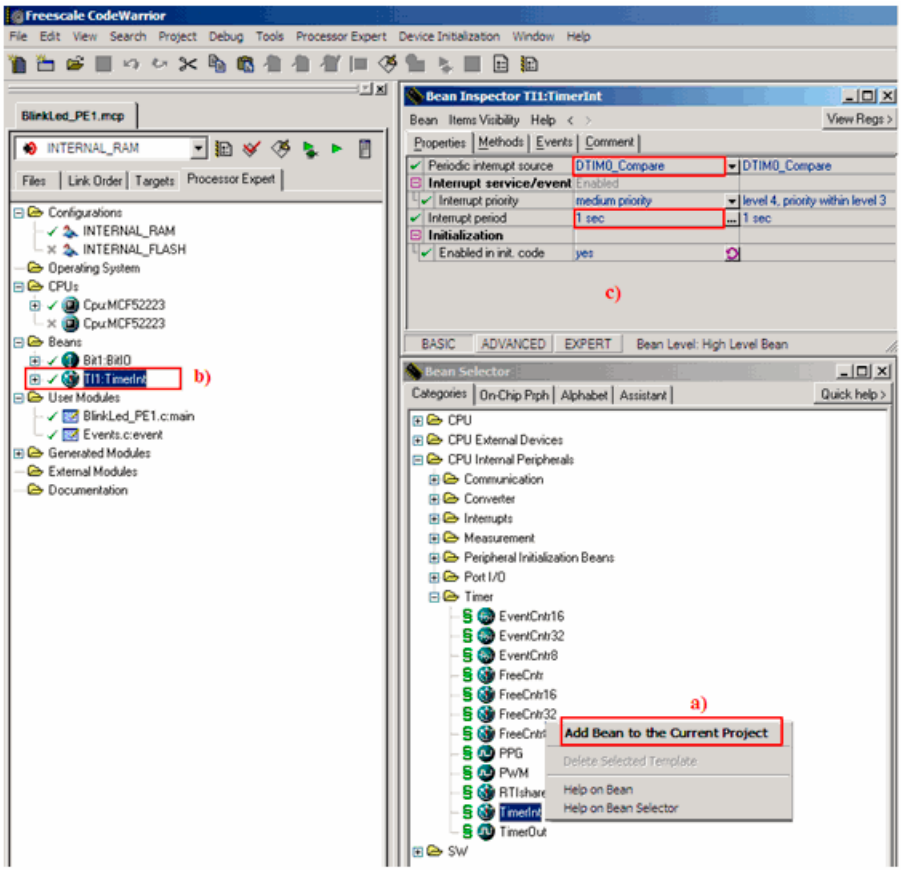
    - GetVal — Returns the input/output value. If the direction is input then the input value of the pin is read and returned. If the direction is output then the last written value is returned.
    - PutVal — Sets the specified output value. If the direction is input, the bean saves the value to a memory or a register. If the direction is output, it writes the value to the pin.
    - ClrVal — Clears the output value when set to zero. It is equivalent to the PutVal(FALSE). This method is available only if the direction = output or input/output.
    - SetVal — Sets the output value to one. It is equivalent to the PutVal(TRUE). This method is available only if the direction = output or input/output.
    - NegVal — Negates the output value by inverting it. This method is available only if the direction = output or input/output.

Figure 15 Methods for the Bean BitIO



12. Add the TimerInt bean from Bean Selector to the project to configure periodic interrupt. Use the Inspector to set up the embedded bean:
    a) Mouse right click on Bean Selector TimerInt
    b) Select the TI1:TimerInt bean in the Bean Folder
    c) Configure the bean properties with the Bean Inspector

Figure 16 Configuration of the Bean TimerInt



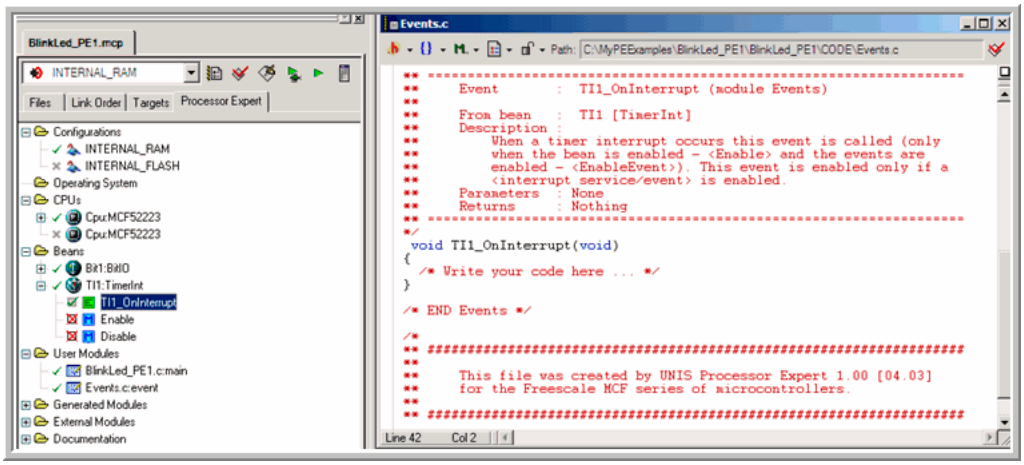
13. Generate the code by selecting **Processor Expert > Generate Code** **`'BlinkLed_PE1.mcp'`**.

    This creates the TimerInt and BitIO beans. Every time TimerInt occurs use the method and events of these beans to make the LED change.

14. Double-click on TI1_OnInterrupt event in the Project Panel.

15. Open the `Events.c` module at the position of an empty function body of TI1_OnInterrupt.

Figure 17 Events.c file TI1_On Interrupt Function



16. Write the following code to the pre-generated timer event TI1_OnInterrupt. If needed, drag and drop the NegVal method inside the TI1_OnInterrupt function. (The NegVal method is inside the BitIO bean.)

```
void TI1_OnInterrupt(void)
{
Bit1_NegVal();
}
```

17. Press F7 to make the application.

18. Test the project:

    a) Connect the board to the PC

    b) Press the debug button

    c) Run it by pressing F5.

> **NOTE** For this example, we used M52223EVB Rev. B.

Example 3: Pulse Width Modulation (PWM)

This embedded bean example implements a PWM that generates a signal with a variable duty and fixed frequency. The PWM bean generates a PWM wave which is its duty-cycle. The PWM output controls an LED. The LED brightness changes according to the PWM duty-cycle applied. The example uses the ColdFire M52235EVB board.

Follow these steps to configure the PWM - LED Blink project:

1. Create a new project for the M52235EVB according to Creating an Empty Project. For this example use the name `PWMTest_PE2.mcp`.
2. Select the **Processor Expert** option.
3. Click **Finish**.
4. Select all configurations (see Figure 13).
5. Click **OK**.

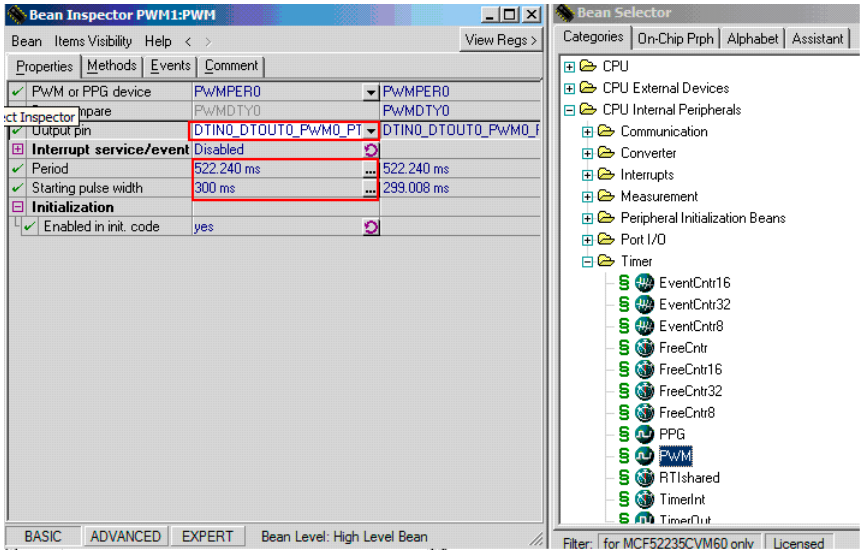
   The Bean Selector window displays the list of available beans on the selected CPU.
6. Add the PWM bean from the Bean Selector window by selecting **CPU Internal Peripherals > Timer > PWM** to configure pins `DTIN0_DTOUT0_PWM0_PTC0`. Use the Inspector to set the settings up as shown in the Figure 18.
7. In the Bean Inspector window configure the following:

   a) Set the Period value to 522.240 ms.

   b) Set the Starting Pulse Width to 300 ms.

   Figure 18 shows the Bean Inspector window for the PWM Bean with the property values of the `PWM_Example` project.

Figure 18 Bean Selector and Configurations



8. Build and run project:
   a) From main menu bar, select **Project > Make** — The IDE updates the files and links code into application.
   b) From main menu bar, select **Project > Debug** — The IDE builds (assembles, compiles, and links) the project; the debug window appears
   c) From main menu bar, select **Project > Run** — The debugger downloads and runs the program
   d) From main menu bar, select **Debug > Kill** — The debug session ends; you may close all open windows

At this point, this project makes LED1 on the M52223EVB switch between ON and OFF according to the PWM timer.

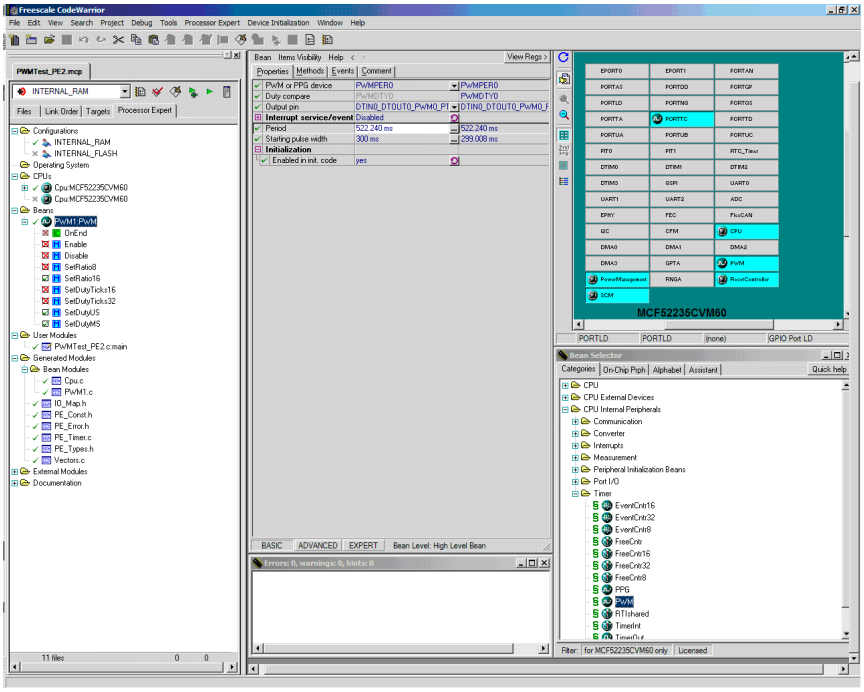
NOTE For this example, we used the M52235EVB.

Embedded Bean Help

To resolve errors and configure the embedded beans, use Processor Expert's Bean Inspector. The Bean Inspector is a GUI provided by Processor Expert within CodeWarrior IDE to configure the embedded bean properties, methods, and events. With the configurations set into the Bean Inspector, Processor Expert automatically generates the initialization and other user support code.

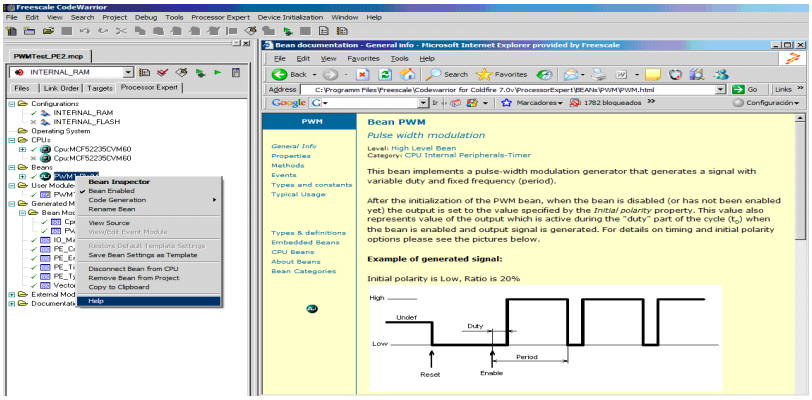
The bean inspector also provides an interface to configure the embedded bean methods and events. Figure 19 shows the Processor Expert view of the project manager window, which lists both method and event functions. An "M" icon designates the method; an "E" icon designates the events. Those method and event functions with a "V " mark generate user code, while those with an "X" mark do not. Access the methods and events tab view of the Bean Inspector to select which method and event functions are enabled for code generation.

Figure 19 CodeWarrior with Processor Expert Project Manager Window



Every embedded bean property, method, and event is documented. You can open an html help page from the Processor Expert view of the CodeWarrior project manager window. To open help for a particular embedded bean, right click the embedded bean and select Help in the menu (see Figure 20). The embedded bean help window also shows example code for each embedded bean.

Figure 20 Embedded Bean Help HTML Page



Conclusion

The cost of development using MCUs is increasingly affected by the Software content of the development projects and complexity of modern MCUs. Part of these costs are reflected in the learning curve associated with new development tools. Software engineers need a high-performance development environment designed to use all of the capabilities of the MCU architecture, while simultaneously minimizing the underlying complexity. It is critical that MCU software development tools provide the same level of ease of use and extensibility as mainstream microprocessor tools, in order to minimize the new product introduction cycle. CodeWarrior development tools, with Processor Expert, provide an integrated suite of development tools. Using Processor Expert and embedded beans can facilitate faster application development and minimize effort. Using Processor Expert to generate code requires less time for developing initialization and peripheral driver code.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

***For Literature Requests Only*:**
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: TN265

7 May 2008

freescale semiconductor