Freescale Semiconductor Document Number: PEXDRVSGETSTARTEDUG
Rev. 2, 09/2012

Processor Expert Software – Microcontrollers Driver Suite

Getting Started Guide

This document introduces Microcontrollers Driver Suite tool. The document describes how to create a new Processor Expert project, create a simple *HelloWorld* application and use the code generated by Processor Expert by a third party C compiler.

Contents

1 Introduction

Microcontrollers Driver Suite is a rapid application design tool targeted for Freescale Kinetis and ColdFire+ Microcontrollers providing the following key features:

A Graphical User Interface which allows an application to be specified by the functionality needed.

An application created from Embedded Components encapsulating initialization and functionality of basic elements of embedded systems.

An automatic code generator which creates tested and optimized C code which is tuned to your application needs and the selected Freescale device.

A built-in knowledge base, which immediately flags resource conflicts and incorrect settings, so errors are caught early in design cycle allowing you to get to market faster with a higher quality product.

Processor Expert creates a Hardware Abstraction Layer (HAL) which provides a unified application programming interface (API) allowing easy migration between Freescale devices.

The application created in Processor Expert is built from the building blocks called the Embedded Components and the CPU components. The CPU components are special type of Embedded Components. An Embedded Component provides:

Selection of peripheral module, pins and timing used by the selected component

Initialization of selected peripheral module according to the current setting of the selected component

Methods to interface component functionality

Events to handle hardware or software events related to the component

The Processor Expert tool can generate code for IAR C Compiler, CodeWarrior C Compiler or GNU C Compiler:

The tool can generate code for IAR C Compiler, CodeWarrior C Compiler, Arm Keil or GNU C Compiler.

For the Kinetis devices, the following compilers are supported:

CodeWarrior for MCU, 10.1 and higher.

IAR ARM compiler 6.3 and higher

GNU C Compiler for ARM architecture

Arm Keil C/C++ compiler 5.4 and higher

For the ColdFire+ devices, the following compilers are supported:

CodeWarrior for MCU, 10.1 and higher.

IAR ColdFire C compiler 1.2

GNU C Compiler for ColdFire architecture

The Microcontrollers Driver Suite is implemented as Eclipse plug-in that can be installed in an existing Eclipse environment (3.6.x or 3.7.x). For installation instructions, see Installation Guide (*PEXDRVSINSTALLUG.pdf*).

2 Creating New Processor Expert Project

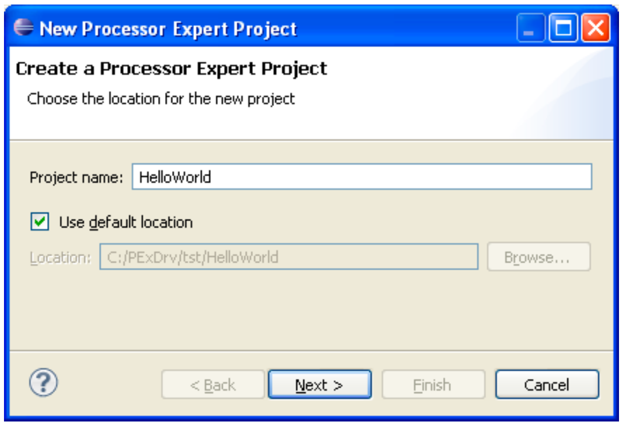
The Microcontrollers Driver Suite provides two options for creating a new Processor Expert project. You can either create an empty project directly using the **New Processor Expert Project** Wizard or you can extend an already existing Eclipse project by the Processor Expert project.

2.1 Using New Project Wizard

To create a new Processor Expert project:

1. Run the Eclipse environment.

2. Select **File > New > Processor Expert Project** from the IDE menu bar. The **Create a Processor Expert Project** screen of the **New Processor Expert Project Wizard** appears (Figure 1Figure 1).
3. Type a name for the project in the **Project name** text box. For example, *HelloWorld* (Figure 1).



Create a Processor Expert Project

Choose the location for the new project

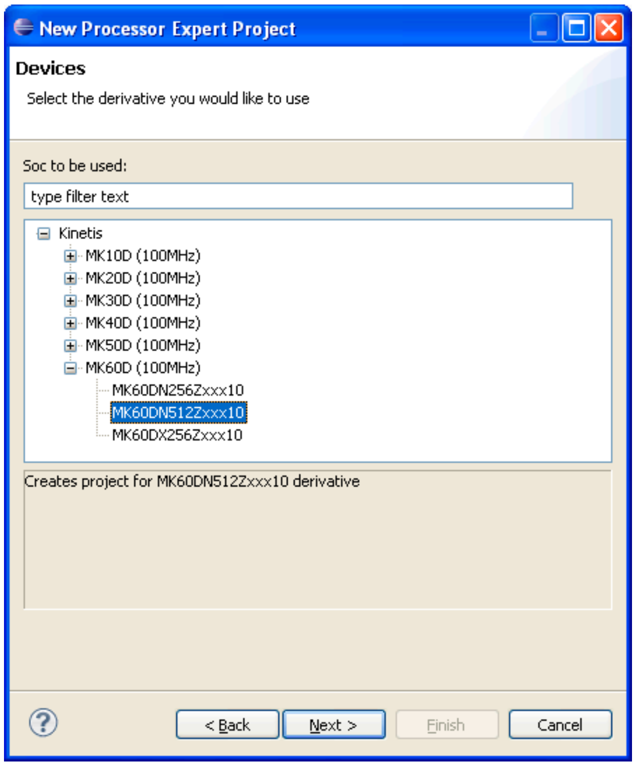
Project name: HelloWorld

Use default location: ✓

Location: C:/PExDrv/tst/HelloWorld

Figure 1. New Processor Expert Project Wizard - Create a Processor Expert Project Screen

4. To specify a different location for the new project, uncheck the **Use default location** checkbox, and click **Browse**. The default setting of the **Use default location** checkbox is checked.
5. Click **Next** to display the **Devices** screen as shown in Figure 2.



Devices

Select the derivative you would like to use

Soc to be used:

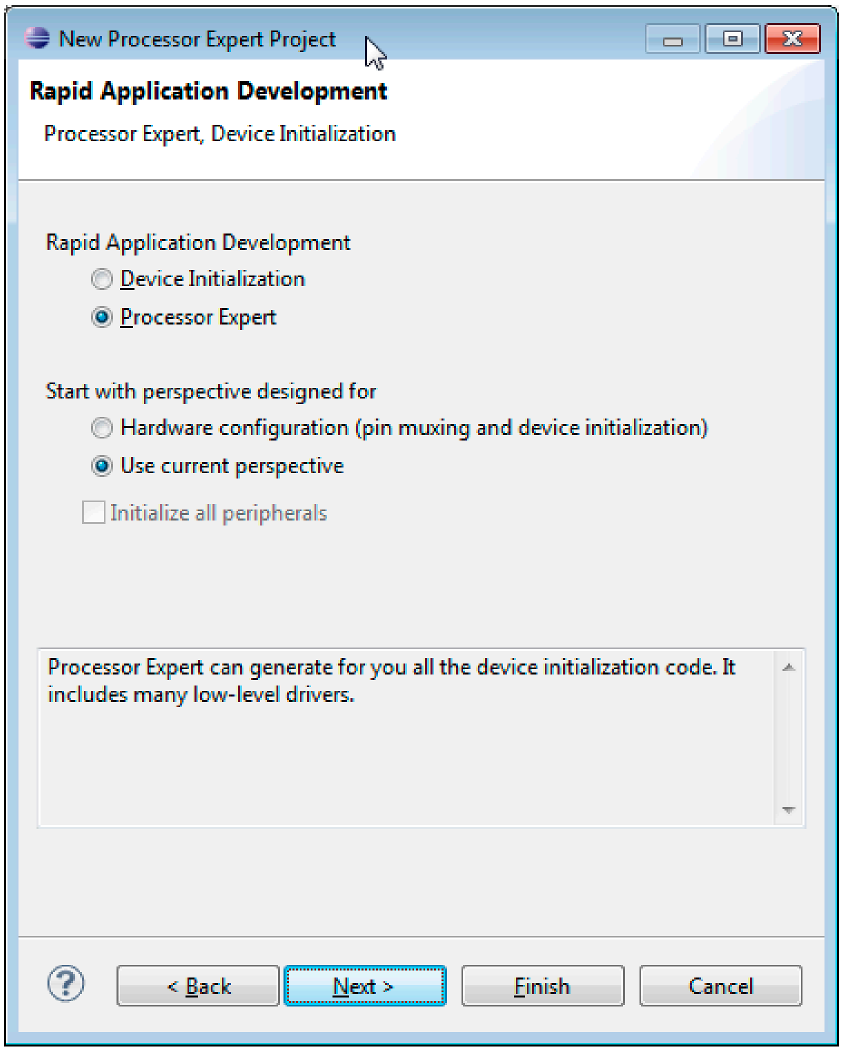
type filter text

- Kinetis
  - MK10D (100MHz)
  - MK20D (100MHz)
  - MK30D (100MHz)
  - MK40D (100MHz)
  - MK50D (100MHz)
  - MK60D (100MHz)
    - MK60DN256Zxxx10
    - MK60DN512Zxxx10
    - MK60DX256Zxxx10

Creates project for MK60DN512Zxxx10 derivative

Figure 2. New Processor Expert Project Wizard– Devices Screen

6. Select the target derivative to be used for the project and click **Next** to display the **Rapid Application Development** screen (Figure 3).


New Processor Expert Project
Rapid Application Development
Processor Expert, Device Initialization

Rapid Application Development
- ( ) Device Initialization
- (●) Processor Expert

Start with perspective designed for
- ( ) Hardware configuration (pin muxing and device initialization)
- (●) Use current perspective

☐ Initialize all peripherals

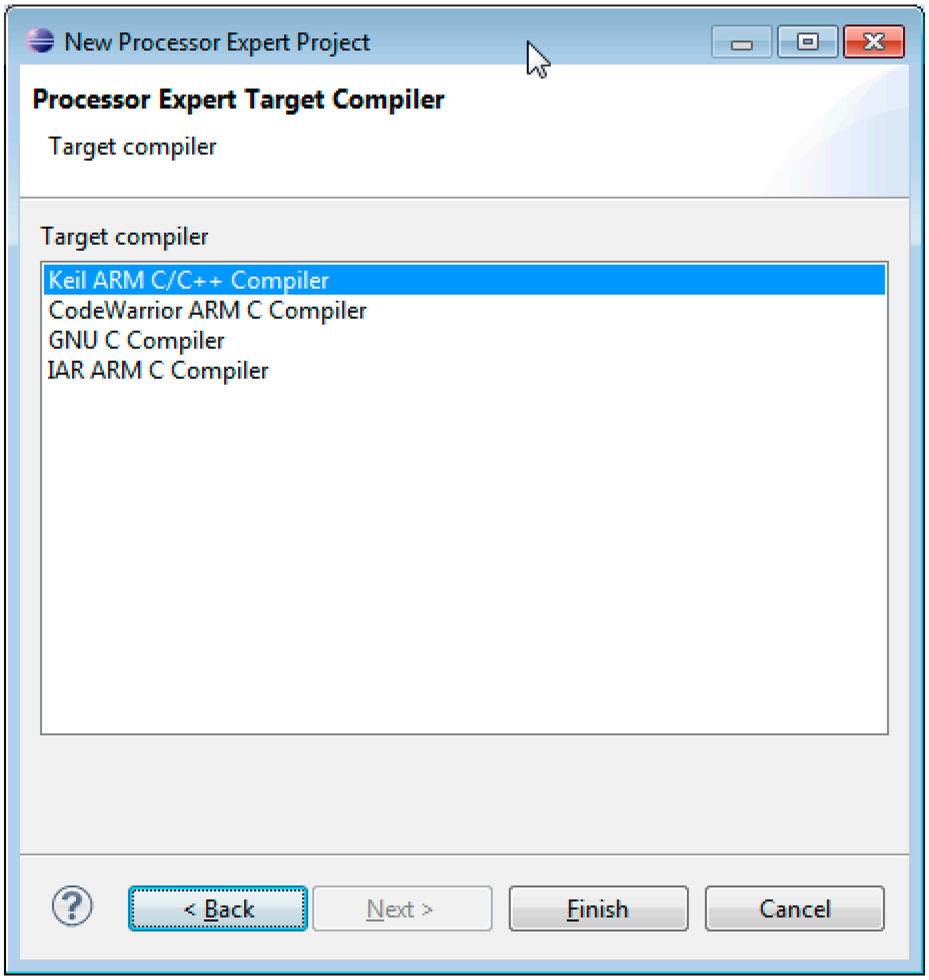
Processor Expert can generate for you all the device initialization code. It includes many low-level drivers.

< Back
Next >
Finish
Cancel

Figure 3. New Processor Expert Project Wizard - Rapid Application Development Screen

7. Select the type of project, **Processor Expert** or **Device Initialization**.

8. Click **Next** to display the **Processor Expert Target Compiler** screen shown below. This screen lets you select the target C compiler for which the project will generate code (the target compiler can be later changed in the ‘Build options’ tab of the CPU component).


New Processor Expert Project

Processor Expert Target Compiler

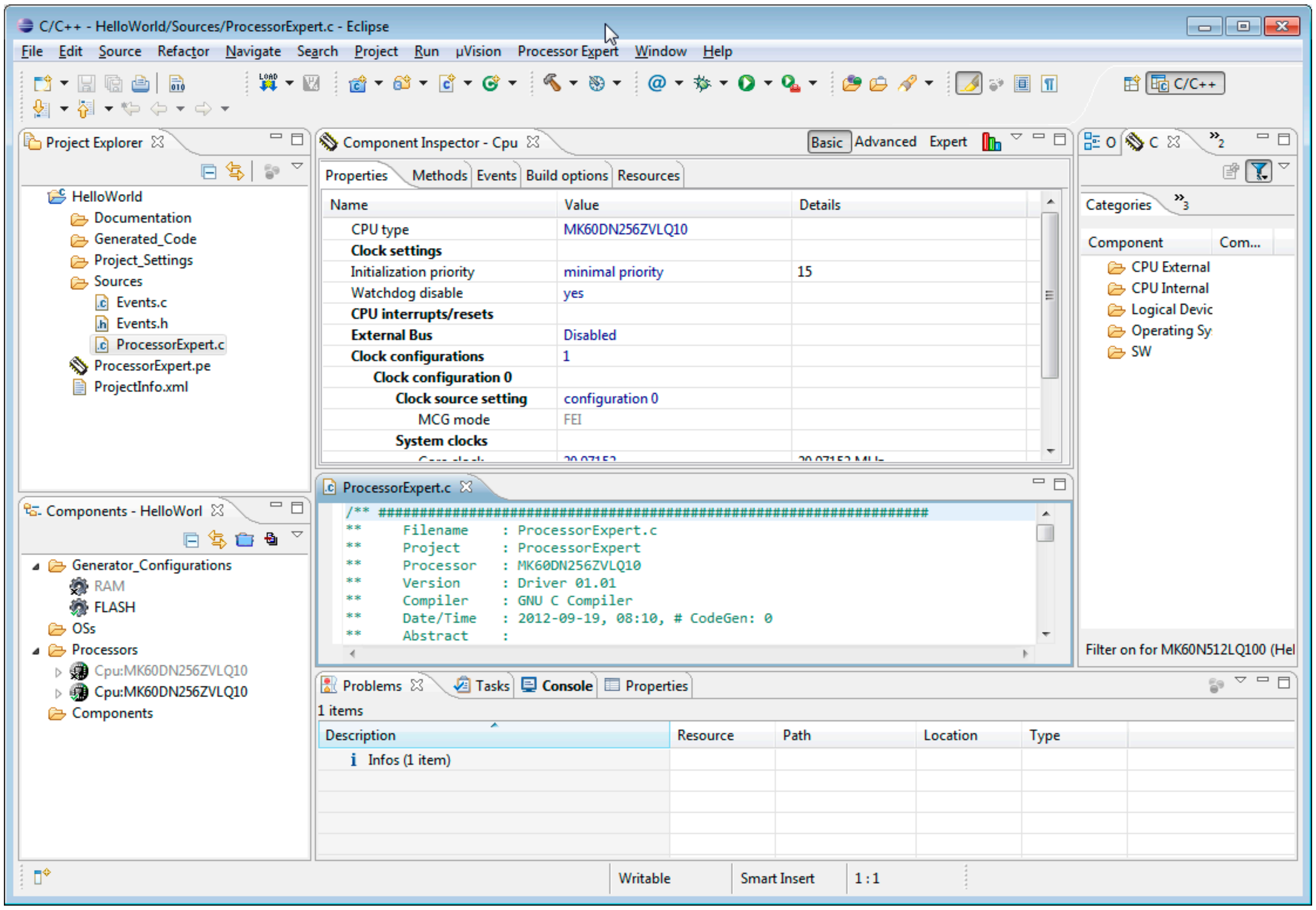
Target compiler

Target compiler

- Keil ARM C/C++ Compiler
- CodeWarrior ARM C Compiler
- GNU C Compiler
- IAR ARM C Compiler

Figure 4. New Processor Expert Project Wizard - Processor Expert Target Compiler Screen

9. Select the required target C compiler and click **Finish** to create the Processor Expert project. The Processor Expert perspective is displayed as shown in figure below. The project *HelloWorld* is created and appears in the **Project Explorer** view. The other views of the Processor Expert perspective are also shown below.



Name	Value	Details
CPU type	MK60DN256ZVLQ10	
Clock settings		
Initialization priority	minimal priority	15
Watchdog disable	yes	
CPU interrupts/resets		
External Bus	Disabled	
Clock configurations	1	
Clock configuration 0		
Clock source setting	configuration 0	
MCG mode	FEI	
System clocks		

```
/** ###################################################################
**     Filename   : ProcessorExpert.c
**     Project    : ProcessorExpert
**     Processor  : MK60DN256ZVLQ10
**     Version    : Driver 01.01
**     Compiler   : GNU C Compiler
**     Date/Time  : 2012-09-19, 08:10, # CodeGen: 0
**     Abstract   :
```

1 items

Description	Resource	Path	Location	Type
Infos (1 item)				

Figure 5. Processor Expert Perspective

The main views in the Processor Expert perspective are:

- Project Explorer
- Components
- Component Inspector
- Components Library
- Problems

For more information on the Processor Expert perspective and its views in detail, refer the *Microcontrollers Driver Suite Processor Expert User Manual*.

2.2 Adding Processor Expert Project to Existing Project

If an Eclipse project is already created in the Eclipse Environment (for example, a project for a third party Eclipse plugin), it is possible to add the Processor Expert support to use generated code in this project.

1. Run Eclipse environment.
2. Open the Eclipse project in which you want to add the Processor Expert support.

3. Select **File > New > Other** from the IDE menu bar. The **Select a Wizard** screen appears.

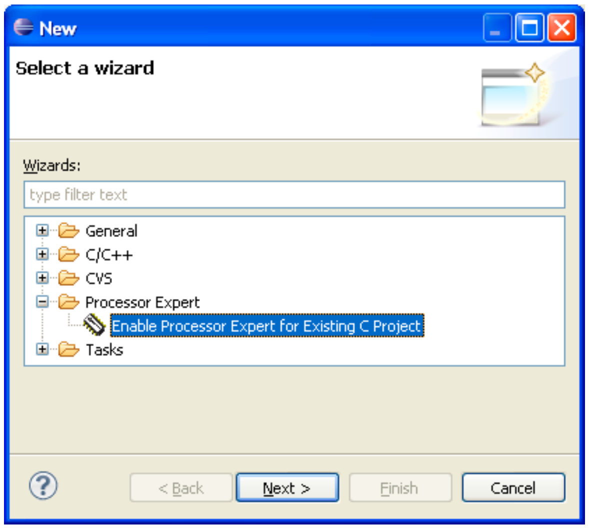
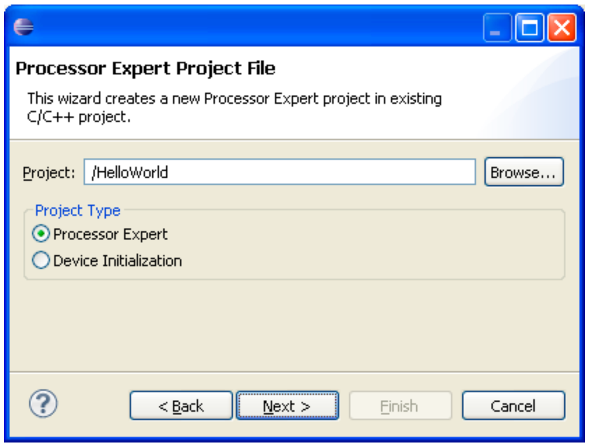


Figure 6. Select a Wizard Screen

4. Click **Next** to open the **Processor Expert Project File** screen as shown in Figure 7.
5. Click **Browse** to select the project to which the Processor Expert project will be added.
6. Select **Project Type** as Processor Expert or Device Initialization.



Processor Expert Project File

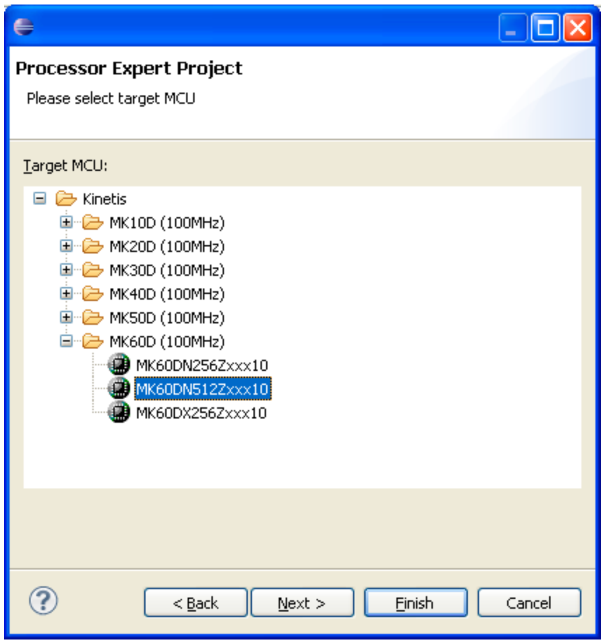
This wizard creates a new Processor Expert project in existing C/C++ project.

Project: /HelloWorld

Project Type
[✓ Processor Expert]
Device Initialization

Figure 7. Processor Expert Project File Screen

7. Click **Next** to open the **Target MCU** selection screen as shown below. This screen lets you select the target derivative to be used for the project.


Processor Expert Project

Please select target MCU

Target MCU:

- Kinetis
  - MK10D (100MHz)
  - MK20D (100MHz)
  - MK30D (100MHz)
  - MK40D (100MHz)
  - MK50D (100MHz)
  - MK60D (100MHz)
    - MK60DN256Zxxx10
    - MK60DN512Zxxx10
    - MK60DX256Zxxx10

Figure 8. Target MCU Select Screen

8. Click **Next** to display the **Processor Expert Target Compiler** screen shown in Figure 4. This screen lets you select the target C compiler for which the project will generate code (the target compiler can be later changed in the 'Build options' tab of the CPU component).
9. Click **Finish** to create Processor Expert project. The Processor Expert perspective is displayed as shown in Figure 5.

NOTE

The *ProcessorExpert.c* file placed in the **Sources** folder contains definition of the `main()` function. If you are adding Processor Expert to a project which already contains user code, it may be necessary to modify the code. Some parts of the existing user code may also conflict with the code generated by Processor Expert, for example, definition of the interrupt vector table. It is recommended to back up the whole project before the conversion.

3 Creating Simple Application

This section shows creation of a simple application, which blinks a LED and sends *Hello World* text to the serial output.

NOTE

The project is designed to work with MK60DN512ZVMD10 CPU and Tower System Kit TWR–K60N512-KIT. However, it is not necessary to have this hardware; the project can be created without it.

3.1 Adding Components to Project

The *HelloWorld* application consists of four components:

CPU component MK60DN512ZVMD10

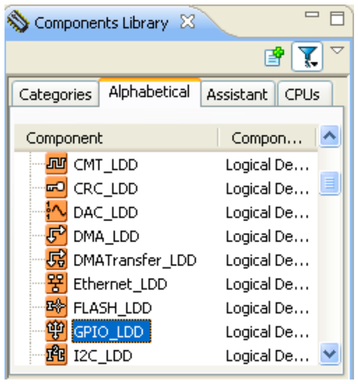
GPIO_LDD which will control LED output, connected to PTA10 pin

TimerUnit_LDD which will provide periodical timing.

Serial_LDD which will send *Hello World* message to the serial output

To create the *HelloWorld* application:

1. Create a new project for Kinetis MK60DN512ZVMD10 derivative. The new project will have the CPU component automatically added when the project is created. See Creating New Processor Expert Project.
2. To add components, select the **Alphabetical** tab in the **Components Library** window as shown below.



Component	Compon...
CMT_LDD	Logical De...
CRC_LDD	Logical De...
DAC_LDD	Logical De...
DMA_LDD	Logical De...
DMATransfer_LDD	Logical De...
Ethernet_LDD	Logical De...
FLASH_LDD	Logical De...
GPIO_LDD	Logical De...
I2C_LDD	Logical De...

Figure 9. Components Library - Alphabetical Tab

3. Find the GPIO_LDD component and double-click on the component to add it to the project.
4. Find the TimerUnit_LDD component and double-click on the component to add it to the project.
5. Find the Serial_LDD component and double-click on the component to add it to the project. All the components will be visible now in the Component view.

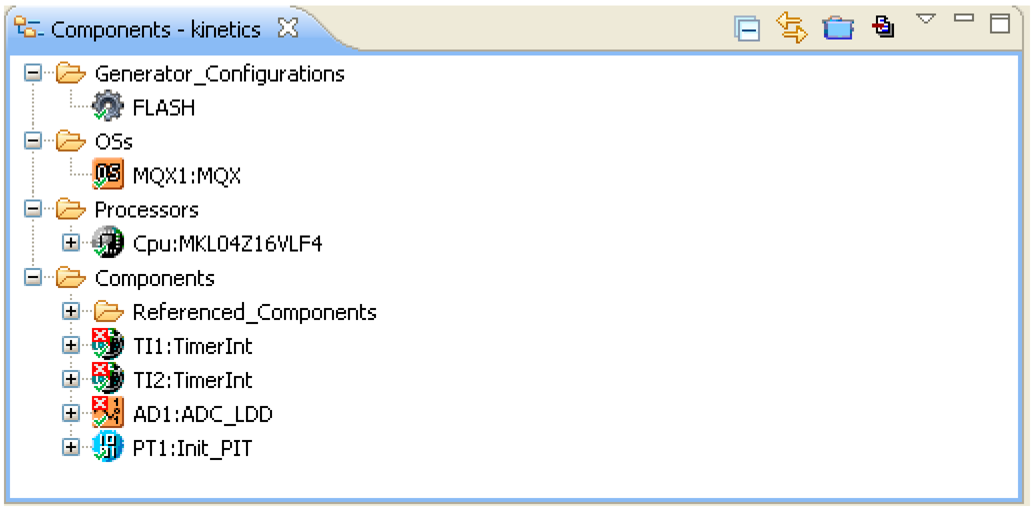


Figure 10. Component View – List of Components

3.2 Configuring Components

The components are configured by defining/changing the values of the properties and enabling/disabling methods and events. The **Component Inspector** view is used for accessing the component properties:

1. To configure **CPU** component, click on this component in the **Component** view to display **Component Inspector**.
2. To match the CPU component settings to the parameters of the TWR-K60N512 board, set the following properties:

    System oscillator to *Enabled*

    System oscillator > Clock source to *External reference clock*

    System oscillator > Clock source > Clock frequency to *50.0*

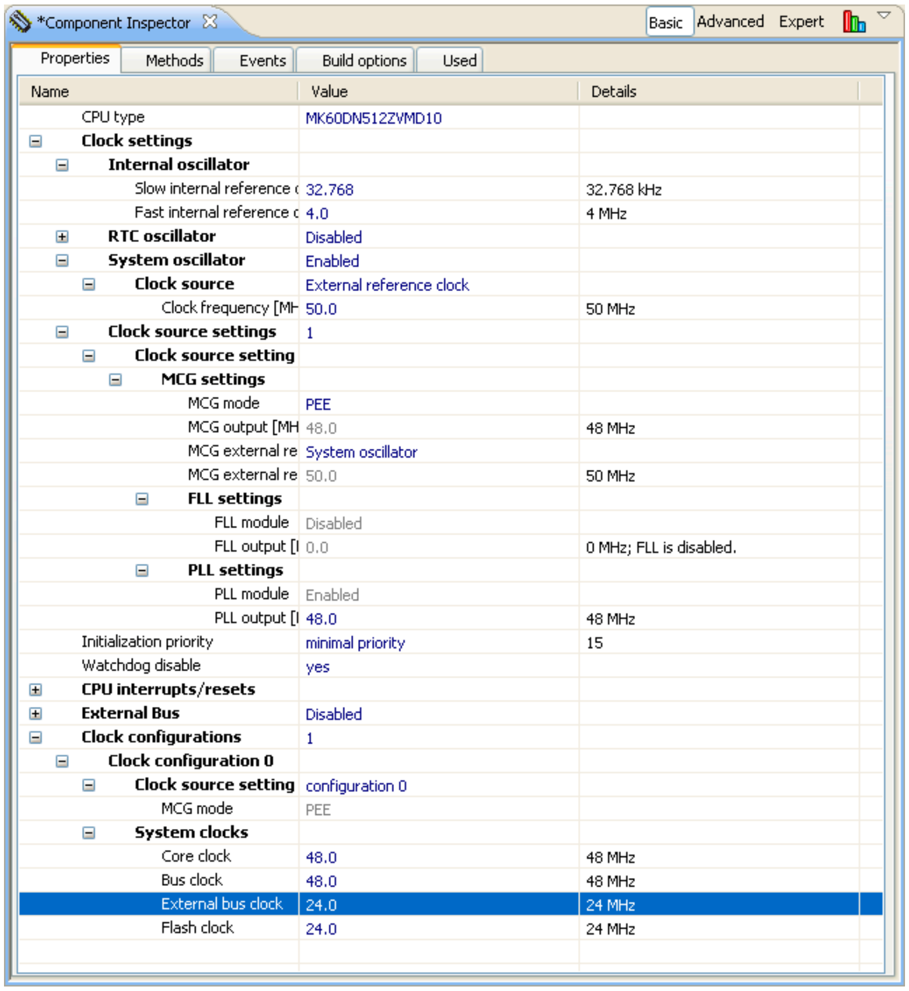
    Clock source settings > Clock source setting 0 > MCG settings > MCG mode to *PEE*

    Clock configurations > Clock configuration 0 > System clocks > Core clock to *48*

    Clock configurations > Clock configuration 0 > System clocks > Bus clock to *48*

    Clock configurations > Clock configuration 0 > System clocks > External bus clock to *24*

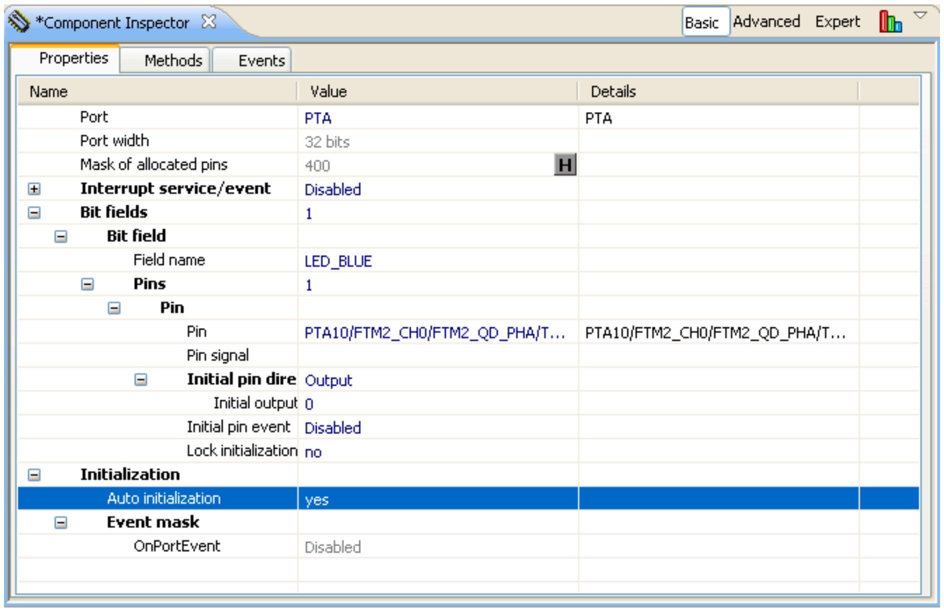
    Clock configurations > Clock configuration 0 > System clocks > Flash clock to *24*



Name	Value	Details
CPU type	MK60DN512ZVMD10	
⊟ Clock settings		
⊟ Internal oscillator		
Slow internal reference c	32.768	32.768 kHz
Fast internal reference c	4.0	4 MHz
⊞ RTC oscillator	Disabled	
⊟ System oscillator	Enabled	
⊟ Clock source	External reference clock	
Clock frequency [MH	50.0	50 MHz
⊟ Clock source settings	1	
⊟ Clock source setting		
⊟ MCG settings		
MCG mode	PEE	
MCG output [MH	48.0	48 MHz
MCG external re	System oscillator	
MCG external re	50.0	50 MHz
⊟ FLL settings		
FLL module	Disabled	
FLL output [I	0.0	0 MHz; FLL is disabled.
⊟ PLL settings		
PLL module	Enabled	
PLL output [I	48.0	48 MHz
Initialization priority	minimal priority	15
Watchdog disable	yes	
⊞ CPU interrupts/resets		
⊞ External Bus	Disabled	
⊟ Clock configurations	1	
⊟ Clock configuration 0		
⊟ Clock source setting	configuration 0	
MCG mode	PEE	
⊟ System clocks		
Core clock	48.0	48 MHz
Bus clock	48.0	48 MHz
External bus clock	24.0	24 MHz
Flash clock	24.0	24 MHz

Figure 11. Component Inspector - CPU

3. To configure **GPIO1** component, click on this component in the **Component** view to display **Component Inspector**.



Name	Value	Details
Port	PTA	PTA
Port width	32 bits	
Mask of allocated pins	400	
Interrupt service/event	Disabled	
Bit fields	1	
Bit field		
Field name	LED_BLUE	
Pins	1	
Pin		
Pin	PTA10/FTM2_CH0/FTM2_QD_PHA/T...	PTA10/FTM2_CH0/FTM2_QD_PHA/T...
Pin signal		
Initial pin dire	Output	
Initial output	0	
Initial pin event	Disabled	
Lock initialization	no	
Initialization		
Auto initialization	yes	
Event mask		
OnPortEvent	Disabled	

Figure 12. Component Inspector – GPIO1

4. Set the following properties:

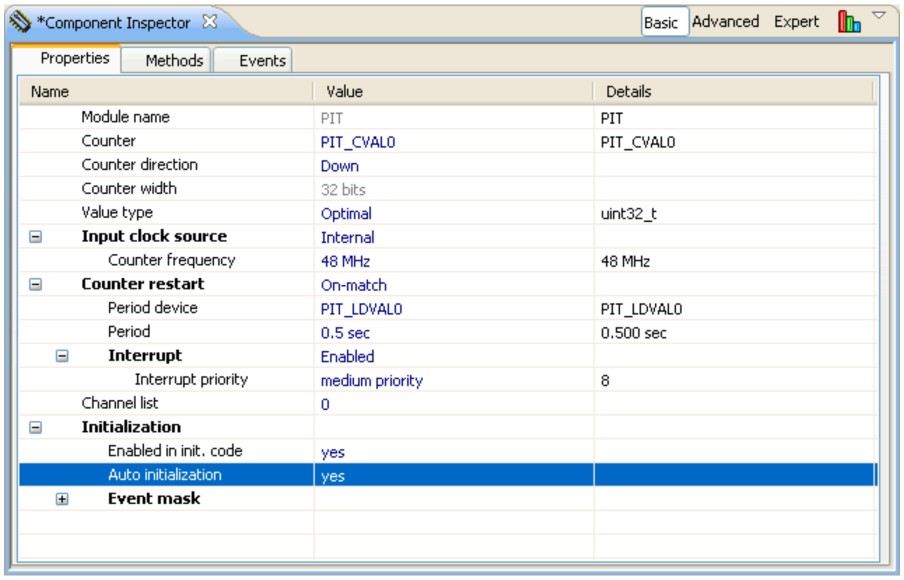
   Field name in the first **Bit field** group to *LED_BLUE*.

   Pin to *PTA10*. This pin corresponds to blue LED on the TWR-K60N512 board.

   Initial pin direction to *Output*.

   Auto initialization to *yes* to automatically initialize the component during startup.

5. To configure **TU1** component, click on this component in the **Component** view to display the **Component Inspector**.



Name	Value	Details
Module name	PIT	PIT
Counter	PIT_CVAL0	PIT_CVAL0
Counter direction	Down	
Counter width	32 bits	
Value type	Optimal	uint32_t
Input clock source	Internal	
Counter frequency	48 MHz	48 MHz
Counter restart	On-match	
Period device	PIT_LDVAL0	PIT_LDVAL0
Period	0.5 sec	0.500 sec
Interrupt	Enabled	
Interrupt priority	medium priority	8
Channel list	0	
Initialization		
Enabled in init. code	yes	
Auto initialization	yes	
Event mask		

Figure 13. Component Inspector - TU1

6. Set the following properties:

Counter to *PIT_CVAL0*

Counter direction to *Down*

Counter restart to *On-match*. This will allow setting desired period of interrupt, otherwise interrupt invocation period is fixed to counter overflow.

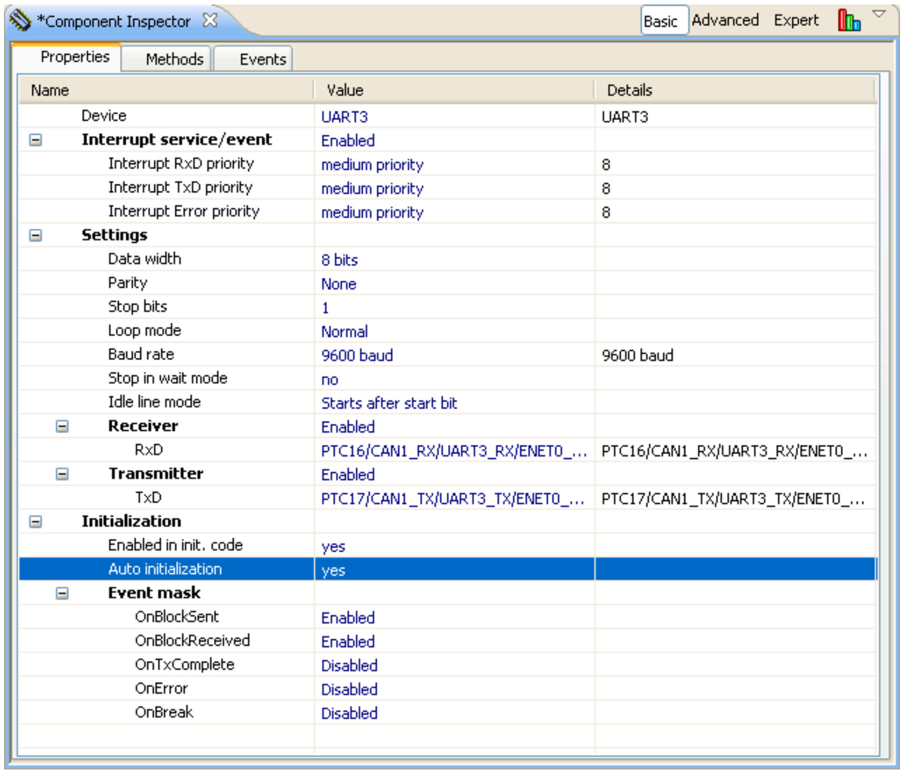
Counter frequency - Use the available button and select the only value offered in the right pane: *48 MHz*.

Counter restart > Period - Click on the available button and set Init. value to *0.5 sec*

Interrupt to *Enabled*

Auto initialization to *yes*

7. To configure **AS1** component, click on this component in the **Component** view to display **Component Inspector**.



Name	Value	Details
Device	UART3	UART3
Interrupt service/event	Enabled	
Interrupt RxD priority	medium priority	8
Interrupt TxD priority	medium priority	8
Interrupt Error priority	medium priority	8
Settings		
Data width	8 bits	
Parity	None	
Stop bits	1	
Loop mode	Normal	
Baud rate	9600 baud	9600 baud
Stop in wait mode	no	
Idle line mode	Starts after start bit	
Receiver	Enabled	
RxD	PTC16/CAN1_RX/UART3_RX/ENET0_...	PTC16/CAN1_RX/UART3_RX/ENET0_...
Transmitter	Enabled	
TxD	PTC17/CAN1_TX/UART3_TX/ENET0_...	PTC17/CAN1_TX/UART3_TX/ENET0_...
Initialization		
Enabled in init. code	yes	
Auto initialization	yes	
Event mask		
OnBlockSent	Enabled	
OnBlockReceived	Enabled	
OnTxComplete	Disabled	
OnError	Disabled	
OnBreak	Disabled	

Figure 14. Component Inspector - AS1

8. Set the following properties:

Device to *UART3* (the Tower serial board is connected to the UART3 port on the TWR-K60N512 board, using pins PTC16 and PTC17)

Baud rate - Click on the available button and set **Init. value** to *9600* baud

RxD to *PTC16*

TxD to *PTC17*

Auto initialization to *yes*

3.3 Generating Code

Select **Project > Generate Processor Expert Code** in the IDE menu to generate code. This process generates source files for components to the **Generated_Code** folder in the **Project Panel** view. Other modules (*ProcessorExpert.c* and *Events.c*) can be found in the **Sources** folder.

3.4 Writing Code

The Processor Expert components provide API, which is used by user code. Initially, you can put the code into the *ProcessorExpert.c* file, which is automatically created by Processor Expert:

1. In the **Project Explorer** view, expand the **Sources** folder and double-click on the **ProcessorExpert.c** module to open the file in the editor.

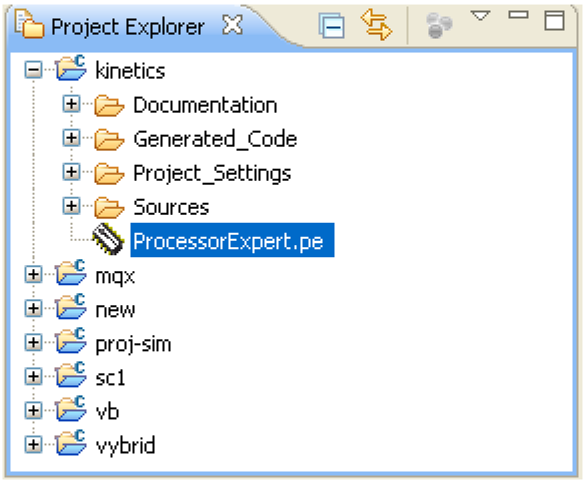


Figure 15. Project Explorer - Main Module

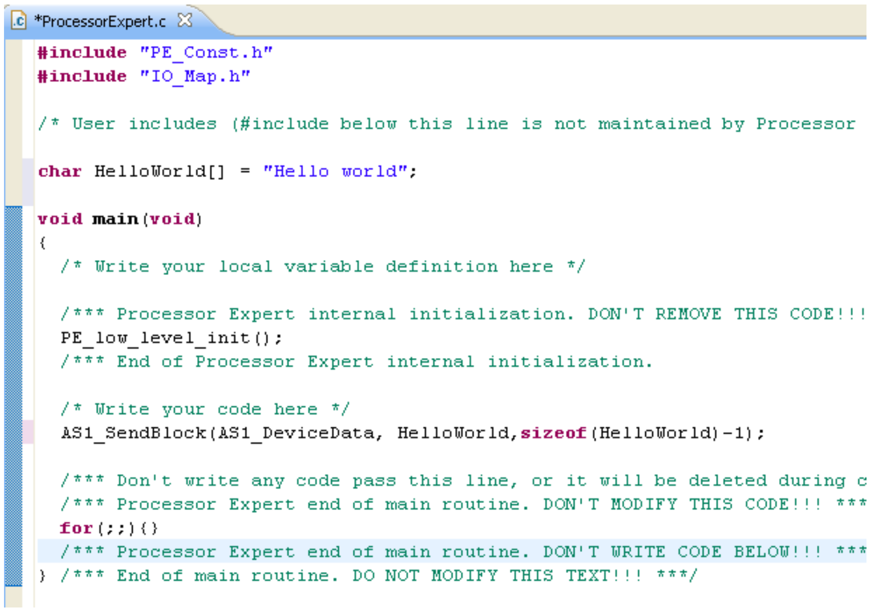
2. In the *ProcessorExpert.c* file, do the following modifications:

   Declare global variable with the message:

   ```
   char HelloWorld[] = "Hello world";
   ```

   In the `main()` function after `PE_low_level_init();` call the `SendBlock` function to send the message:

   ```
   AS1_SendBlock(AS1_DeviceData, HelloWorld,sizeof(HelloWorld)-1);
   ```



```
#include "PE_Const.h"
#include "IO_Map.h"

/* User includes (#include below this line is not maintained by Processor

char HelloWorld[] = "Hello world";

void main(void)
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!!
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.

  /* Write your code here */
  AS1_SendBlock(AS1_DeviceData, HelloWorld,sizeof(HelloWorld)-1);

  /*** Don't write any code pass this line, or it will be deleted during c
  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***
  for(;;){}
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/
```

Figure 16. ProcessorExpert.c File

3. In the **Component** view, expand the list of events and methods of **TU1** component.
4. Right-click on **TU1_OnCounterRestart** and select the **View Code** command from the pop-up menu to open the code of **TU1_OnCounterRestart** in the editor.

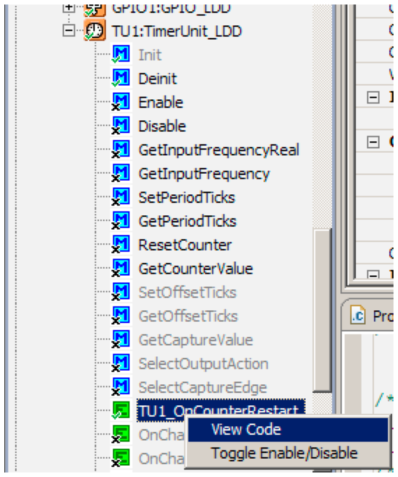


Figure 17. View Code

5. Insert the following lines (toggling LED output) into the `TU1_OnCounterRestart` event handler function body:

```
GPIO1_ToggleFieldBits(GPIO1_DeviceData,LED_BLUE,1);
```

The application is complete and can be compiled and run using a third party toolchain.

4 Using Generated Code

Processor Expert uses the following sub-directory structure within the project directory:

Generated_Code – the directory containing all generated source code modules for components.

Documentation – the directory with the project documentation files generated by Processor Expert.

Sources – the directory for main module, event module and other user modules.

Project_Setings\Linker_Files – the directory containing linker file generated by Processor Expert

Project_Setings\Startup_Code – the directory containing the Processor Expert startup code (compiler dependent)

4.1 Project Include Path

Set the following paths in the settings of the target IDE to correctly use the code generated by Processor Expert with a third party C compiler:

```
{ProjectDirectory}\Sources
{ProjectDirectory}\Generated_Code
```

For Kinetis family, set the following paths:

```
{ProcessorExpertDirectory}\lib\Kinetis\iofiles
{ProcessorExpertDirectory}\lib\Kinetis\pdd\inc
```

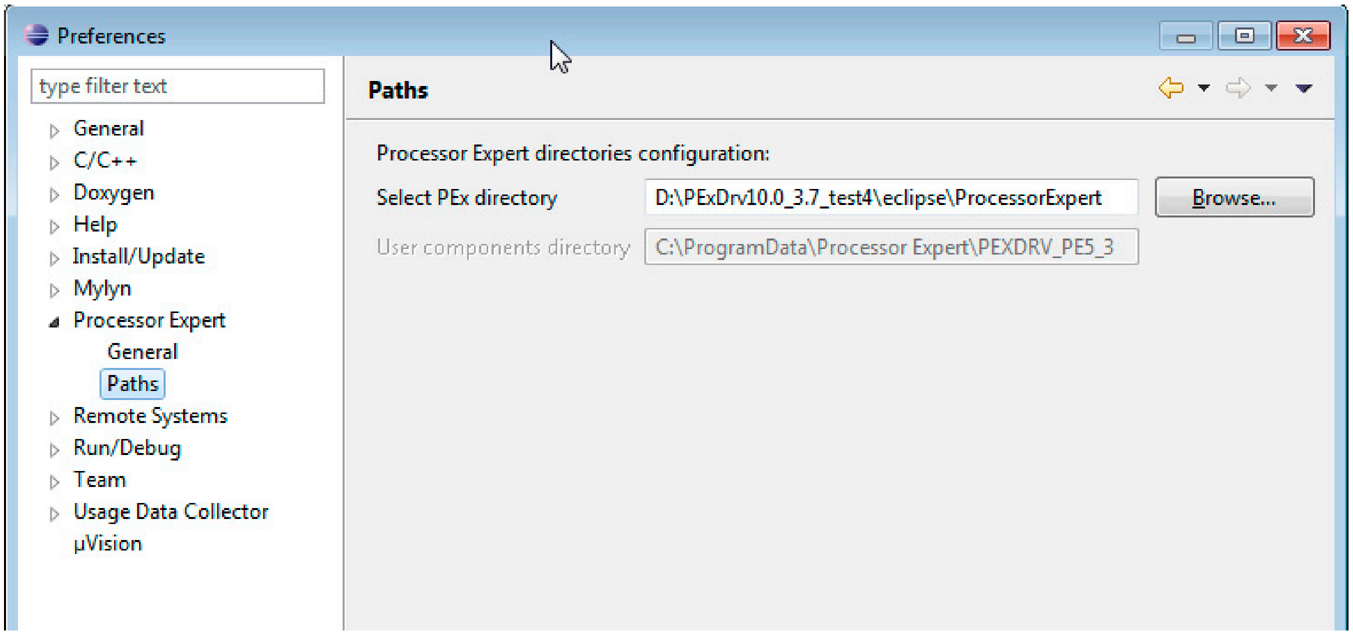
For ColdFire+ family, set the following paths:

```
{ProcessorExpertDirectory}\lib\ColdFirePlus\iofiles
{ProcessorExpertDirectory}\lib\ColdFirePlus\pdd\inc
```

The default location of the Processor Expert directory depends on the operating system and user rights. By default, the Processor Expert directory is created as a subdirectory of the Eclipse installation:

```
{eclipse}\ProcessorExpert
```

The actual location can be found in the preferences of Processor Expert. Select **Window > Preferences > Processor Expert > Processor Expert Service** to view the actual location.


Preferences

type filter text

- General
- C/C++
- Doxygen
- Help
- Install/Update
- Mylyn
- Processor Expert
  - General
  - Paths
- Remote Systems
- Run/Debug
- Team
- Usage Data Collector
- µVision

Paths

Processor Expert directories configuration:

Select PEx directory D:\PExDrv10.0_3.7_test4\eclipse\ProcessorExpert

User components directory C:\ProgramData\Processor Expert\PEXDRV_PE5_3

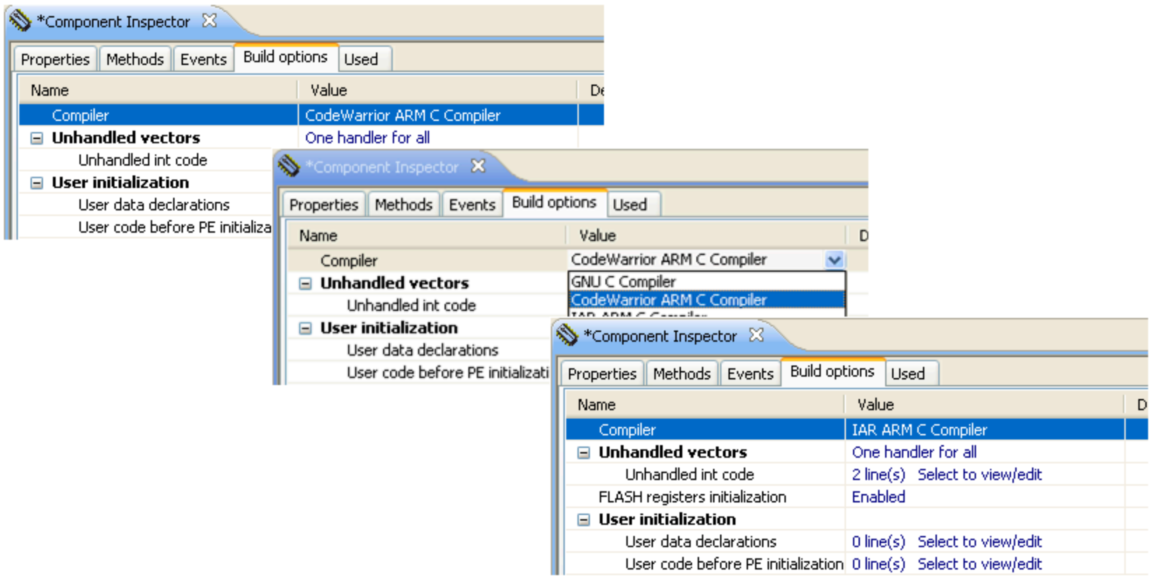
Figure 18. Processor Expert Directory Location

4.2 Project Linker File

Linker file specific for the selected target CPU and compiler is generated by Processor Expert. The file is stored at the following location: `{ProjectDirectory}\Project_Settings\Linker_Files`

The name of the linker file is ProcessorExpert, extension of the linker file is compiler specific.

The code generated by Processor Expert components is tailored to the selected target compiler. Make sure that the selected compiler matches used build tools. The target compiler is selected during project creation. It can be later changed in the **Build options** tab of the CPU component.


*Component Inspector

Properties | Methods | Events | Build options | Used

Name	Value
Compiler	CodeWarrior ARM C Compiler
Unhandled vectors	One handler for all
Unhandled int code	
User initialization	
User data declarations	
User code before PE initializa	

*Component Inspector

Properties | Methods | Events | Build options | Used

Name	Value
Compiler	CodeWarrior ARM C Compiler
Unhandled vectors	GNU C Compiler / CodeWarrior ARM C Compiler
Unhandled int code	
User initialization	
User data declarations	
User code before PE initializati	

*Component Inspector

Properties | Methods | Events | Build options | Used

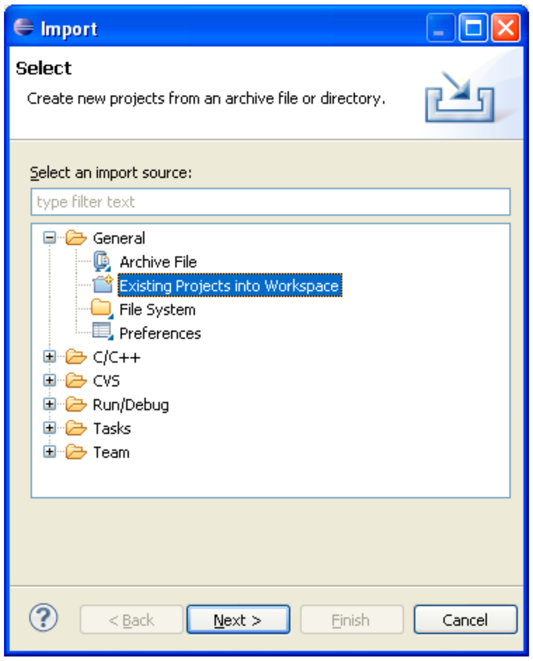
Name	Value
Compiler	IAR ARM C Compiler
Unhandled vectors	One handler for all
Unhandled int code	2 line(s) Select to view/edit
FLASH registers initialization	Enabled
User initialization	
User data declarations	0 line(s) Select to view/edit
User code before PE initialization	0 line(s) Select to view/edit

Figure 19. Changing Target Compiler

5 Importing Example Projects

The Microcontrollers Driver Suite comes with a set of example projects. The example projects are stored in the **ProcessorExpert\Projects** folder. To import an example project:

1. Select **File > Import** to open the Import wizard shown below.
2. Select **General > Existing Projects into Workspace**.


Import

Select

Create new projects from an archive file or directory.

Select an import source:

type filter text

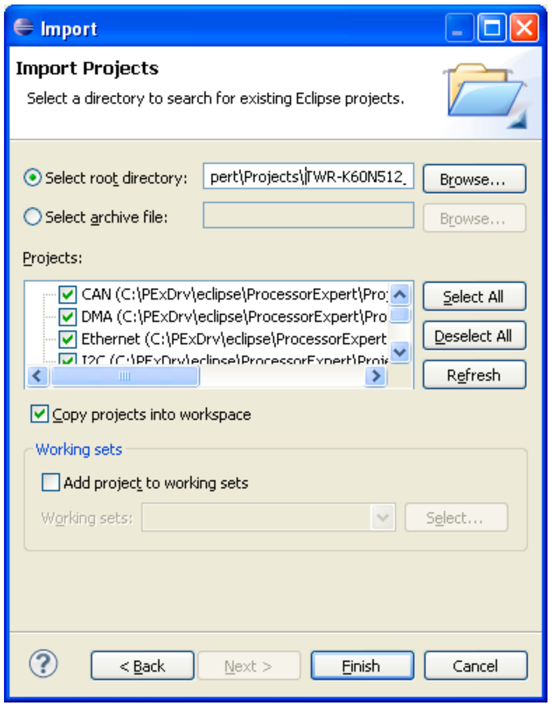
- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
- C/C++
- CVS
- Run/Debug
- Tasks
- Team

< Back | Next > | Finish | Cancel

Figure 20. Import Screen

3. Click **Next** to display the **Import Projects** screen.

4. Click **Browse** to select the **ProcessorExpert\Projects** folder. The **Projects** list box lists all the projects available for import.



Import
Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory:
pert\Projects\TWR-K60N512
Browse...
Select archive file:
Browse...
Projects:
CAN (C:\PExDrv\eclipse\ProcessorExpert\Pro
DMA (C:\PExDrv\eclipse\ProcessorExpert\Pro
Ethernet (C:\PExDrv\eclipse\ProcessorExpert
I2C (C:\PExDrv\eclipse\ProcessorExpert\Proj
Select All
Deselect All
Refresh
Copy projects into workspace
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Finish
Cancel

Figure 21. Import Projects Screen

5. Check all projects that you want to import into your workspace. To copy the example project into your workspace, check the **Copy projects into workspace** checkbox.
6. Click **Finish** to import selected projects.

Document Number: PEXDRVGETSTARTEDUG
Rev: 2
Date: 06/2011