CodeWarrior Development Studio for StarCore 3900FP DSP Architectures FAQ Guide

Document Number: CWSCFAQUG
Rev. 10.9.0, 06/2015

Contents

Section number **Title** **Page**

Chapter 1
Introduction

Chapter 2
IDE

Section number Title Page

Chapter 3 Project Management

Chapter 4 Project

Section number | Title | Page

Chapter 5
Compiler

Chapter 6
Linker

Chapter 7
Debugger

Section number Title Page

Chapter 8
Debugger Shell

Chapter 9
Profiling and Analysis Tool

Section number	Title	Page

Chapter 1
Introduction

This FAQ Guide lists most frequently asked or most anticipated questions about CodeWarrior Development Studio for StarCore 3900FP DSP v10.x.

In this document, the FAQs are divided into various categories based on the feature they are associated with, such as Installation, IDE, Editor, Project Management, Project, Compiler, Assembler, Linker, Debugger, Debugger Shell, USBTAP and Ethernet TAP.

In this chapter:

- Contents of this Manual - Describes the contents of this manual.
- New FAQs - Lists new FAQs in this manual.
- Accompanying Documentation - Describes supplementary CodeWarrior documentation, third-party documentation, and references.

1.1 Contents of this Manual

This chapter describes the contents of this manual.

The following table lists and describes each chapter in this manual. Each chapter lists a specific category of FAQs.

Table 1-1. Manual Contents

Chapter	Description
IDE	Lists IDE related FAQs
Project Management	Lists project management related FAQs
Project	Lists project related FAQs
Compiler	Lists compiler related FAQs
Linker	Lists linker related FAQs
Debugger	Lists debugger related FAQs
Debugger Shell	Lists debugger shell related FAQs
Profiling and Analysis Tool	Lists profiler related FAQs

1.2 Accompanying Documentation

The **Documentation** page describes the documentation included in this version of CodeWarrior Development Studio for StarCore 3900FP DSP Architectures.

You can access the **Documentation** by any of the following methods:

- a shortcut link on the Desktop that the installer creates by default.
- opening the `START_HERE.html` in `CWInstallDir\SC\Help` folder.

1.3 New FAQs

The manual includes these new FAQs for the current release.

- NEW How do I switch between the source files in the Editor view?
- NEW I want to edit my source files, but all the other views are using the space as well. How do I maximize the editor view to have better working area?
- NEW How do I remove blank lines in the Editor view?
- NEW How do I customize my Workspace perspectives?
- NEW How do I enable full screen mode in my CodeWarrior IDE?
- NEW How do I hide the toolbar from the IDE for better visibility of other views that I am currently working with?
- NEW How can I view all keyboard shortcuts available in the CodeWarrior IDE?
- NEW How do I edit the keyboard shortcuts in the CodeWarrior IDE?
- NEW What is a Working Set? How do I build/compile a Working Set?
- NEW How do I search within the Working Set?
- NEW How do I manage the Working Sets?
- NEW How can I can filter the projects listed in the CodeWarrior Projects view based on Working Sets?
- NEW How can I import/export a Working Set in the CodeWarrior eclipse IDE?
- NEW What are Icon decorators?
- NEW How do I enable or disable an icon decorator?
- NEW Can the icon decorators be combined?
- NEW When I open a large file in CodeWarrior IDE, an Editor Scalability dialog box appears, listing the message, "You are opening a large file. Scalability mode has been turned on for this editor to help improve performance by disabling features requiring to parse the source file language." What does it mean?
- NEW How do I configure the Scalability settings?

- NEW While launching a project, the CodeWarrior software automatically builds the current active target. How can I specify the build configuration I want to debug?
- NEW While building a project, the CodeWarrior fails to write an ELF file. How do I resolve this issue?
- NEW How do I enable the incremental build on my project?
- NEW While debugging in RAM my debugger halts and reports an "illegal breakpoint" error. What do I do?
- NEW Can I use software breakpoints on code running in flash memory?
- NEW What is a hardware breakpoint?
- NEW How do I select the type of breakpoint for my application in the Editor view?
- NEW By default, the debugger shows me the global variables of the currently debugged source file. How can I view my variables from another module?
- NEW How do I debug my application using global variables?
- NEW How do I filter variables in files?
- NEW How do I enable watch on a variable?
- NEW What is a static local variable?
- NEW When I enable watch on a static variable, it shows the error, "<error(s)_during_the_evaluation>". What do I do?

Chapter 2
IDE

In this chapter, you find Frequently Asked Questions (FAQs) related to the CodeWarrior IDE.

The FAQs listed in this chapter are divided into following categories:

- Editor
- Workbench Window
- CVS
- Search and Replace Action
- Miscellaneous

2.1 Editor

In this topic, Editor related FAQs are listed.

- How can I quickly open declaration of any variable, function name, macro, or header file from within Editor?
- If I am using a Macintosh computer, what is the alternate to using the F3 key in Editor to open declarations?
- How can I change the color that highlights the current line in a source code file?
- Is it possible to view definition of a macro or a variable in the source code file?
- Can the CodeWarrior IDE assist me in writing the source code?
- How can I change the tab width/size?
- Is it possible to display line numbers in Editor?
- Is it possible to change the default code format settings?
- How can I quickly switch to a header file from within the source code file?
- How does the CodeWarrior IDE differentiate between enabled and disabled macros?
- How can I configure predefined macros in the CodeWarrior IDE?
- Is it possible to view evaluated expansion of a macro in the CodeWarrior IDE?

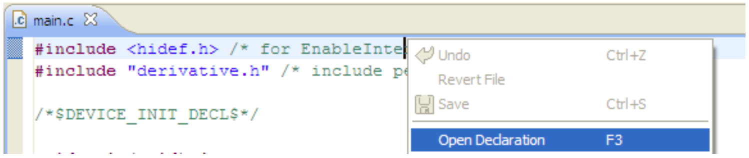
- Is it possible to roll back the changes I did to my source code?
- Is there a way to change the default color of comments that span across multiple lines?
- Is there a way to revert to the original contents of a source code file?
- How do I convert Line Delimiters to the Unix format?
- Why do I get the following error message when I try to edit a source code file?
- What does an asterisk on the Editor title bar mean?
- How can I find out where a function is declared in my source code?
- How can I open .tcl extension files in the Editor window?
- NEW How do I switch between the source files in the Editor view?
- NEW I want to edit my source files, but all the other views are using the space as well. How do I maximize the editor view to have better working area?
- NEW How do I remove blank lines in the Editor view?

2.1.1 How can I quickly open declaration of any variable, function name, macro, or header file from within Editor?

To quickly open declaration of any variable, function name, macro or header file from within Editor, press the **Ctrl** key and click the text. Alternatively, place the mouse cursor on the text and perform either of the following:

- press F3,
- right-click and select **Open Declaration** (as the figure, *Open Declaration* listed below shows), or
- select **Navigate > Open Declaration** from the IDE menu bar .

This will open the logical file associated with the selected text. For example, for a header file include, the header file will open. For variables, the file containing the definition or declaration of the variable will open.



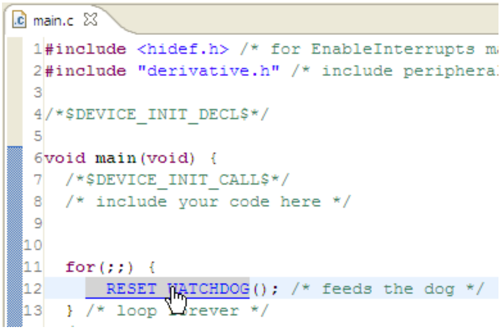
```
#include <hidef.h> /* for EnableInte
#include "derivative.h" /* include pe

/*$DEVICE_INIT_DECL$*/
```

Figure 2-1. Open Declaration

2.1.2 If I am using a Macintosh computer, what is the alternate to using the F3 key in Editor to open declarations?

If you are using a Macintosh computer, just keep the CTRL key pressed and move the mouse cursor over the header file or function whose declaration or definition you want to open. The hyperlink for that header file or function enables.

Click the hyperlink to switch to the required declaration or definition, as the following figure shows:



```
#include <hidef.h> /* for EnableInterrupts m
#include "derivative.h" /* include periphera

/*$DEVICE_INIT_DECL$*/

void main(void) {
  /*$DEVICE_INIT_CALL$*/
  /* include your code here */


  for(;;) {
    RESET_WATCHDOG(); /* feeds the dog */
  } /* loop forever */
```

Figure 2-2. Hyperlink in Source Files

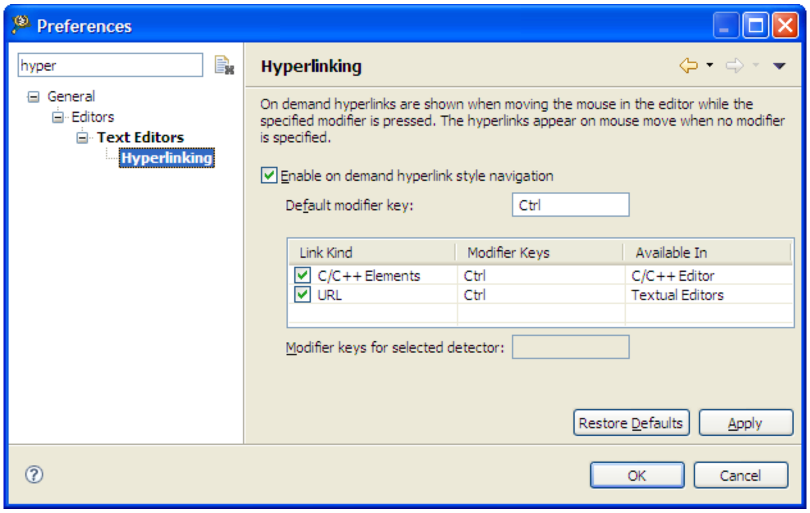
By default, the modifier key for hyperlinks in editor is Ctrl; however, you can change the modifier key as per your preference.

1. Select **Window > Preferences**.

   The **Preferences** dialog box appears.

2. Type **Hyperlink** as the filter text to narrow down the list of preferences.
3. Select **General > Editors > Text Editors > Hyperlinking**.

   The **Hyperlinking** page appears in the right panel.



On demand hyperlinks are shown when moving the mouse in the editor while the specified modifier is pressed. The hyperlinks appear on mouse move when no modifier is specified.

☑ Enable on demand hyperlink style navigation

Default modifier key: Ctrl

Link Kind	Modifier Keys	Available In
☑ C/C++ Elements	Ctrl	C/C++ Editor
☑ URL	Ctrl	Textual Editors

Modifier keys for selected detector:

Figure 2-3. Hyperlinking Page

4. Enter the required modifier key in the **Default modifier key** text box.
5. Click **OK**.

The modifier key changes, as specified.

2.1.3 How can I change the color that highlights the current line in a source code file?

To change the color for the current line highlight in a source file:

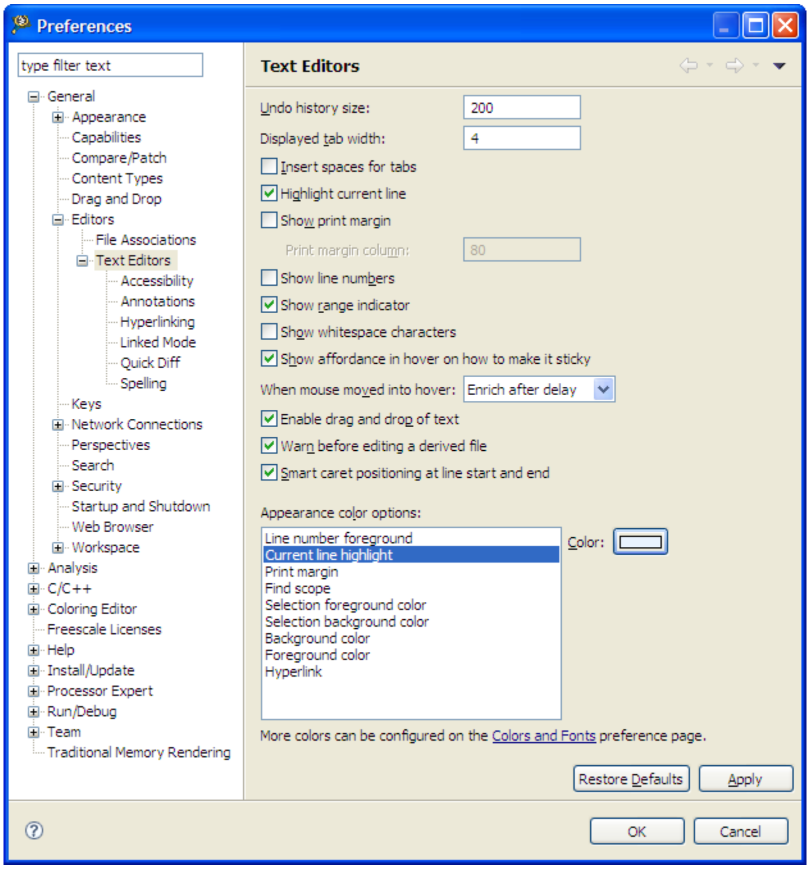
1. Select **Window > Preferences** .

   The **Preferences** dialog box appears.

2. Type **text editor** as the filter text to narrow down the list of preferences.
3. Select **General > Editors > Text Editors** .

   The **Text Editors** page appears in the right panel.

4. Ensure that **Highlight current line** checkbox is checked.
5. Select **Current line highlight** in the **Appearance color options** list.



Text Editors

Undo history size: 200

Displayed tab width: 4

☐ Insert spaces for tabs

☑ Highlight current line

☐ Show print margin

Print margin column: 80

☐ Show line numbers

☑ Show range indicator

☐ Show whitespace characters

☑ Show affordance in hover on how to make it sticky

When mouse moved into hover: Enrich after delay

☑ Enable drag and drop of text

☑ Warn before editing a derived file

☑ Smart caret positioning at line start and end

Appearance color options:

- Line number foreground
- Current line highlight
- Print margin
- Find scope
- Selection foreground color
- Selection background color
- Background color
- Foreground color
- Hyperlink

Color:

More colors can be configured on the Colors and Fonts preference page.

Figure 2-4. Text Editors Page > Current line highlight Selected

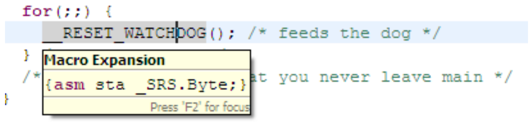
6. Click the **Color** button to open the color panel, and select the required color.
7. Click **OK** to close the color panel.
8. Click **OK** to save the settings.

The current line highlight color changes, as specified.

2.1.4 Is it possible to view definition of a macro or a variable in the source code file?

Yes, you can open definition of a macro or variable by hovering the mouse cursor over the macro or variable. A small popup window displaying the definition of the macro (refer to the image, *Macro Definition* as listed below) or variable (refer to the image, *Variable Declarations and Occurrences* as listed below) appears.

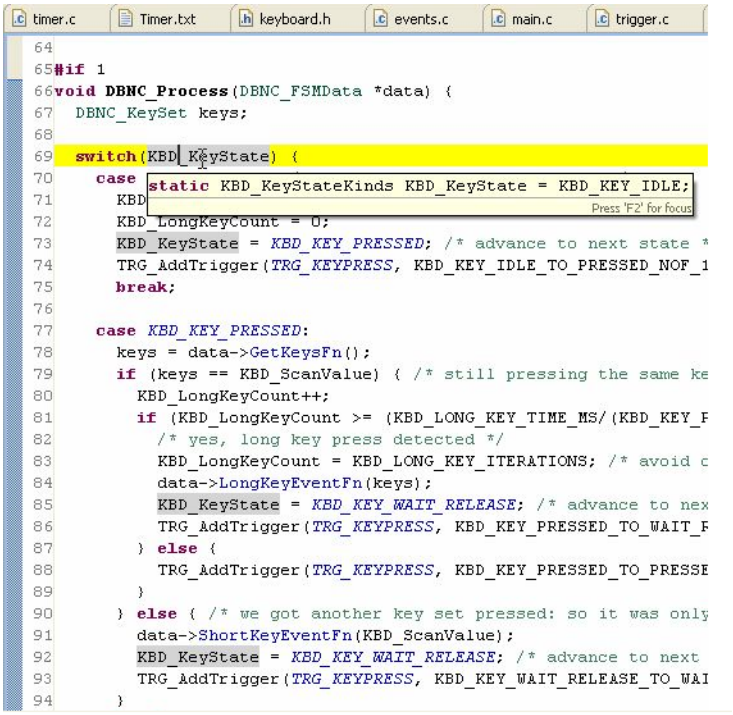
Also, the IDE highlights all the occurrences of the variable in the source file, so you do not need to search for it manually.



```
for(;;) {
  __RESET_WATCHDOG(); /* feeds the dog */
}
/*                            t you never leave main */
}
```

```
{asm sta _SRS.Byte;}
```

Figure 2-5. Macro Definition



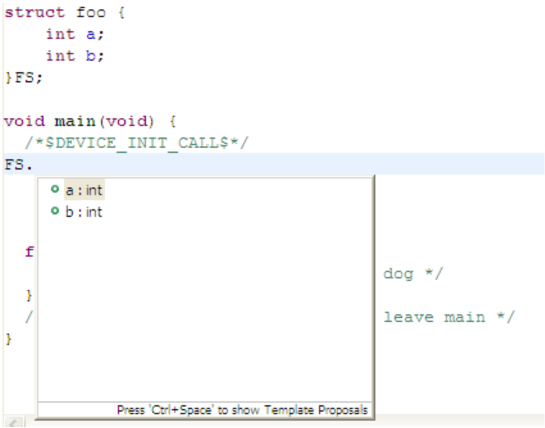
```

#if 1
void DBNC_Process(DBNC_FSMData *data) {
  DBNC_KeySet keys;

  switch(KBD_KeyState) {
    case static KBD_KeyStateKinds KBD_KeyState = KBD_KEY_IDLE;
      KBD
      KBD_LongKeyCount = 0;
      KBD_KeyState = KBD_KEY_PRESSED; /* advance to next state *
      TRG_AddTrigger(TRG_KEYPRESS, KBD_KEY_IDLE_TO_PRESSED_NOF_1
      break;

    case KBD_KEY_PRESSED:
      keys = data->GetKeysFn();
      if (keys == KBD_ScanValue) { /* still pressing the same ke
        KBD_LongKeyCount++;
        if (KBD_LongKeyCount >= (KBD_LONG_KEY_TIME_MS/(KBD_KEY_F
          /* yes, long key press detected */
          KBD_LongKeyCount = KBD_LONG_KEY_ITERATIONS; /* avoid c
          data->LongKeyEventFn(keys);
          KBD_KeyState = KBD_KEY_WAIT_RELEASE; /* advance to nex
          TRG_AddTrigger(TRG_KEYPRESS, KBD_KEY_PRESSED_TO_WAIT_F
        } else {
          TRG_AddTrigger(TRG_KEYPRESS, KBD_KEY_PRESSED_TO_PRESSE
        }
      } else { /* we got another key set pressed: so it was only
        data->ShortKeyEventFn(KBD_ScanValue);
        KBD_KeyState = KBD_KEY_WAIT_RELEASE; /* advance to next
        TRG_AddTrigger(TRG_KEYPRESS, KBD_KEY_WAIT_RELEASE_TO_WAI
      }
```

Figure 2-6. Variable Declaration and Occurrences

2.1.5 Can the CodeWarrior IDE assist me in writing the source code?

Yes, CodeWarrior IDE can parse the source files in the background. Therefore, it provides you the code completion feature. To use the code completion feature, select **Edit > Content Assist** from the IDE menu bar.

The Content Assist feature allows you to view the list of the field member of a structure, class, or union. The list of the field members appears automatically the moment you type `.' following the structure, class, or union name (refer to the image listed below). You can also press **Ctrl+Space** to view the list.



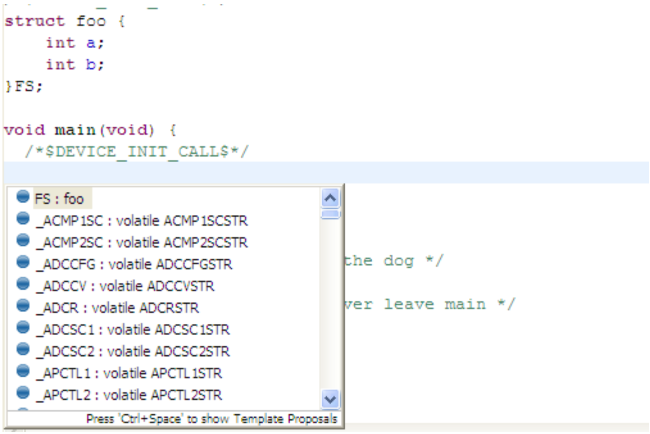
```
struct foo {
    int a;
    int b;
}FS;

void main(void) {
  /*$DEVICE_INIT_CALL$*/
FS.
```

- a : int
- b : int

Press 'Ctrl+Space' to show Template Proposals

Figure 2-7. View Field Members of Structure

If you press **Ctrl+Space** without typing anything or by placing the cursor in a blank line in the source file, the complete list appears, as shown in the figure listed below.



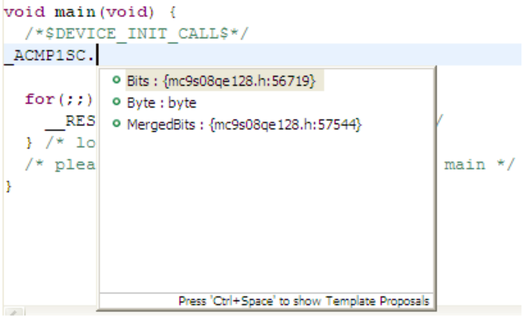
```
struct foo {
    int a;
    int b;
}FS;

void main(void) {
  /*$DEVICE_INIT_CALL$*/
```

- FS : foo
- _ACMP1SC : volatile ACMP1SCSTR
- _ACMP2SC : volatile ACMP2SCSTR
- _ADCCFG : volatile ADCCFGSTR
- _ADCCV : volatile ADCCVSTR
- _ADCR : volatile ADCRSTR
- _ADCSC1 : volatile ADCSC1STR
- _ADCSC2 : volatile ADCSC2STR
- _APCTL1 : volatile APCTL1STR
- _APCTL2 : volatile APCTL2STR

Press 'Ctrl+Space' to show Template Proposals

Figure 2-8. Content Assist - Without Typing Anything

You can also view the list of bitfields using Content Assist, as shown in the figure listed below.



```
void main(void) {
  /*$DEVICE_INIT_CALL$*/
_ACMP1SC.
```

- Bits : {mc9s08qe128.h:56719}
- Byte : byte
- MergedBits : {mc9s08qe128.h:57544}

Press 'Ctrl+Space' to show Template Proposals

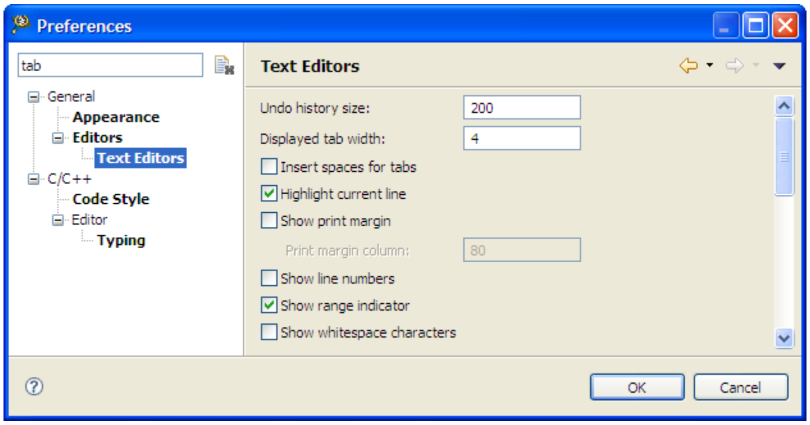
Figure 2-9. Bitfields

2.1.6 How can I change the tab width/size?

By default, the tab size in the CodeWarrior IDE is 4. To change the default tab size:

1. From the IDE menu bar, select **Window > Preferences**.
2. Type **tab** as the filter text to narrow down the list of preferences.
3. Select **Text Editors**.

   The **Text Editors** preference page appears in the right panel of the **Preferences** dialog box.



Undo history size: 200
Displayed tab width: 4
☐ Insert spaces for tabs
☑ Highlight current line
☐ Show print margin
Print margin column: 80
☐ Show line numbers
☑ Show range indicator
☐ Show whitespace characters

Figure 2-10. Text Editors Preferences Page

4. Enter the desired tab size in the **Displayed tab width** text box.
5. Click **OK**.

The default tab size changes.

2.1.7 Is it possible to display line numbers in Editor?

Yes, to display line numbers in the editor area:

1. From the IDE menu bar, select **Window > Preferences** .

   The **Preferences** dialog box appears.

2. Type **number** as the filter text to narrow down the list of preferences.
3. Select **Text Editors** . The **Text Editors** preference page appears in the right panel of the **Preferences** dialog box.
4. Check the **Show line numbers** checkbox.
5. Click **OK** .

The editor will display line numbers.

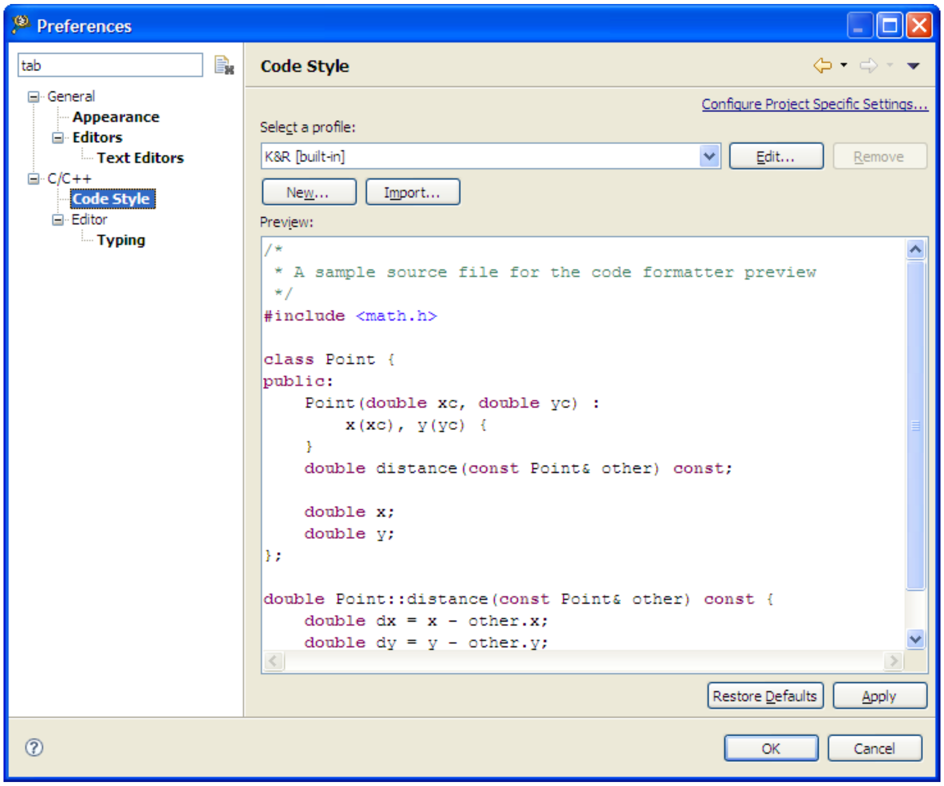
2.1.8 Is it possible to change the default code format settings?

Yes, to change the default code format settings:

1. From the IDE menu bar, select **Window > Preferences**.

   The **Preferences** dialog box appears.

2. Type **code** as the filter text to narrow down the list of preferences.
3. Select **Code Style** in the left pane to modify the code formatting.



tab

Select a profile: K&R [built-in]

```
/*
 * A sample source file for the code formatter preview
 */
#include <math.h>

class Point {
public:
    Point(double xc, double yc) :
        x(xc), y(yc) {
    }
    double distance(const Point& other) const;

    double x;
    double y;
};

double Point::distance(const Point& other) const {
    double dx = x - other.x;
    double dy = y - other.y;
```

Figure 2-11. Code Style Preference Page

4. Select a predefined profile from the **Select a profile** drop-down list. Each of the profile specify different ways of formatting the code.

   If the predefined profiles does not suit your requirements, you can create a new profile.

   a. Click the **New** button.

      The **New Code Formatter Profile** dialog box appears.

b. Enter the name of the new profile in the **Profile name** text box.
c. Select the profile based on which you want to create the new profile from the **Initialize settings with the following profile** drop-down list.
d. Click **OK**.

   The edit dialog box appears.

e. Specify the code formatting settings as required and click **OK**.

5. Click **OK** to close the **Preferences** dialog box.

The editor will now use the selected profile to format the code.

2.1.9 How can I quickly switch to a header file from within the source code file?

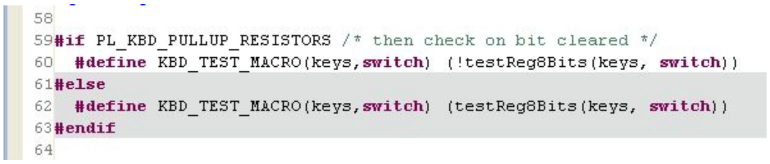
If you are working in an implementation (*.c/*.cpp) file, and you quickly want to open the corresponding header file, perform either of the following:

- press **CTRL+`** or
- right-click and select **Toggle Source/Header** .

Similarly, you can also toggle back to the source file from the header file.

2.1.10 How does the CodeWarrior IDE differentiate between enabled and disabled macros?

CodeWarrior IDE automatically marks the disabled macros in gray to help you determine which macros are active and which not.



```

#if PL_KBD_PULLUP_RESISTORS /* then check on bit cleared */
  #define KBD_TEST_MACRO(keys,switch) (!testReg8Bits(keys, switch))
#else
  #define KBD_TEST_MACRO(keys,switch) (testReg8Bits(keys, switch))
#endif

```

Figure 2-12. Disabled Macros Marked in Gray

2.1.11 How can I configure predefined macros in the CodeWarrior IDE?

To configure predefined macros in the CodeWarrior IDE:

1. Select the project for which you want to configure the predefined macros in the **CodeWarrior Projects** view.
2. Select **Project > Properties** in the IDE menu bar.

   The **Properties** dialog box appears.

3. Select **C/C++ General > Paths and Symbols** .

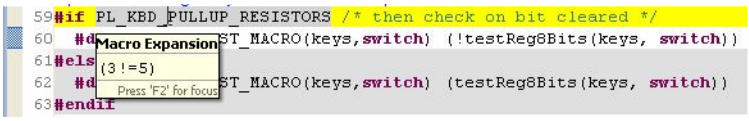
   The **Paths and Symbols** properties page opens in the right panel of the **Properties** dialog box.

4. Select the **Symbols** tab.
5. Configure the symbols listed in the **Symbols** tab page.
6. Click **OK** .

The predefined macros are configured for the selected project.

2.1.12 Is it possible to view evaluated expansion of a macro in the CodeWarrior IDE?

Yes, if you hover mouse cursor over a complex macro, the CodeWarrior IDE shows the evaluated expression (refer to the image listed below).

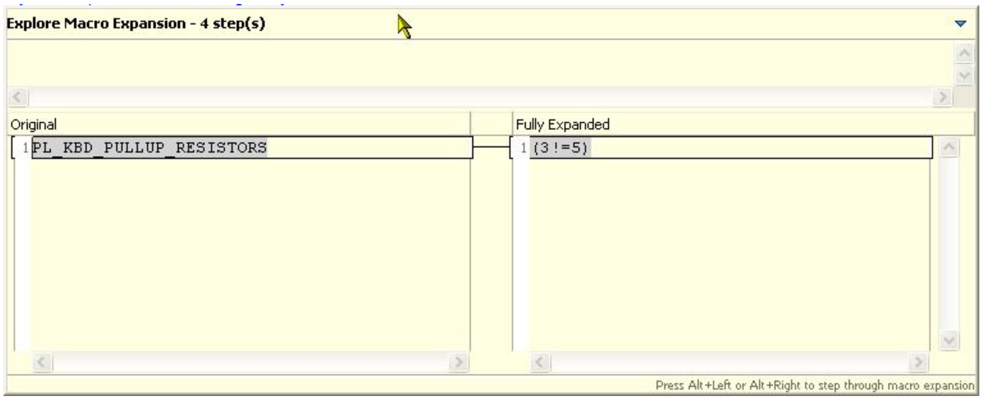


```
#if PL_KBD_PULLUP_RESISTORS /* then check on bit cleared */
  #d                  ST_MACRO(keys,switch) (!testReg8Bits(keys, switch))
#els
  #d                  ST_MACRO(keys,switch) (testReg8Bits(keys, switch))
#endif
```

(3 !=5)

Figure 2-13. Pop-up Window Displaying Macro Expansion

Press F2 to bring in focus the macro expansion (refer to the image listed below).

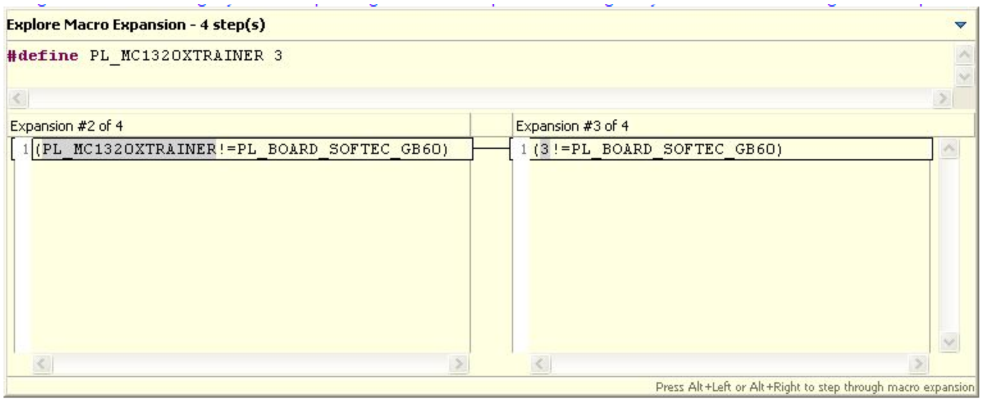

Explore Macro Expansion - 4 step(s)
Original
Fully Expanded
Press Alt+Left or Alt+Right to step through macro expansion

```
PL_KBD_PULLUP_RESISTORS
```

```
(3 !=5)
```

Figure 2-14. Focus over Macro Expansion

Using **Alt+Left** and **Alt+Right** you can step through the macro expansion. This gives you details of the steps in the macro expansion (refer to the image listed below).


Explore Macro Expansion - 4 step(s)
Expansion #2 of 4
Expansion #3 of 4
Press Alt+Left or Alt+Right to step through macro expansion

```
#define PL_MC1320XTRAINER 3
```

```
(PL_MC1320XTRAINER!=PL_BOARD_SOFTEC_GB60)
```

```
(3 !=PL_BOARD_SOFTEC_GB60)
```

Figure 2-15. Macro Expansion Steps

2.1.13 Is it possible to roll back the changes I did to my source code?

Yes, local history can help roll back the changes you make to your source code.

To roll back the changes, perform these steps.

1. Right-click the updated source code in the editor.
2. From the context menu, select the **Replace with > Local History** command.

   The **Compare** dialog box appears. The dialog box lists the date and time of the changes you saved.

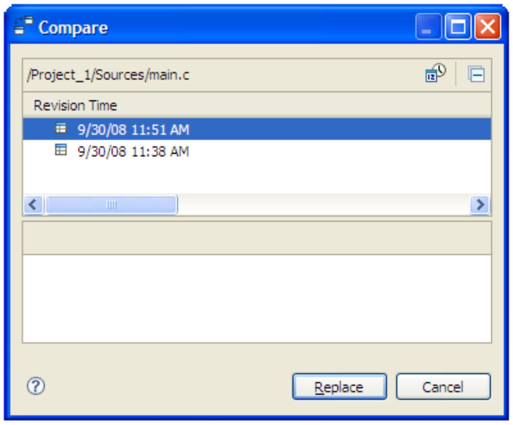


Figure 2-16. Compare Dialog Box

3. From the **Revision Time** options, select the desired date and time.
4. Click the **Replace** button.

The changes made to the file will be roll backed to the selected date and time.

NOTE

The local history is maintained for files only and not for projects or folders.

2.1.14 Is there a way to change the default color of comments that span across multiple lines?

Yes, in order to change the default color of comments that span across multiple lines in source code follow steps below:

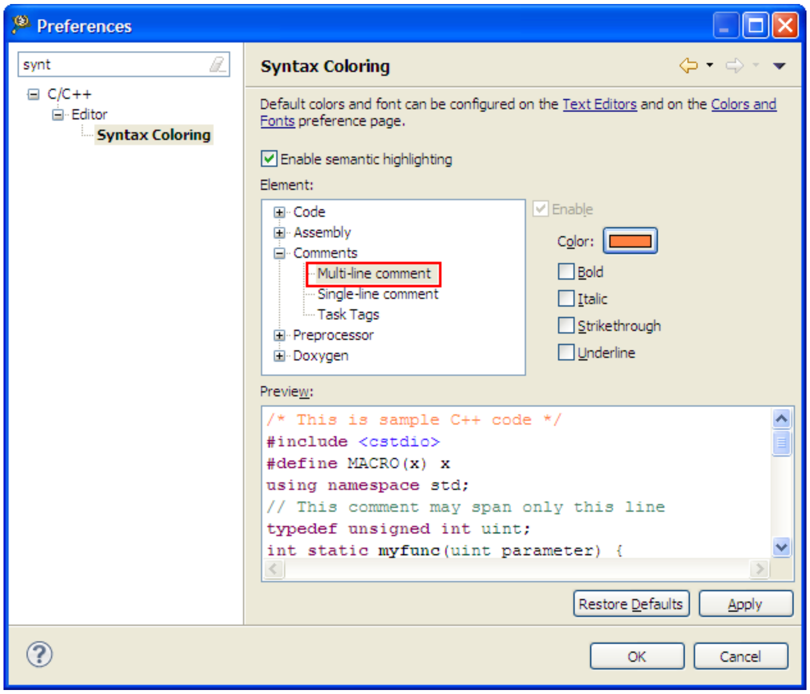
1. From C/C++ perspective toolbar, select **Preferences**.

   The **Preferences** window appears.

2. Select **C/C++ > Editor > Syntax Coloring**.

   The **Syntax Coloring** page appears in the right panel (refer to the image listed below).

3. Select **Comments > Multi-line comment**.



Syntax Coloring

Default colors and font can be configured on the Text Editors and on the Colors and Fonts preference page.

☑ Enable semantic highlighting

Element:

☑ Enable

Color:

□ Bold

□ Italic

□ Strikethrough

□ Underline

Preview:

```
/* This is sample C++ code */
#include <cstdio>
#define MACRO(x) x
using namespace std;
// This comment may span only this line
typedef unsigned int uint;
int static myfunc(uint parameter) {
```

Figure 2-17. Preferences Dialog Box - Syntax Coloring

4. Select **Color**.

   The color palette window appears.

5. From Color Palette, select a color of your choice.
6. Click **OK**.
7. Click **Apply**.
8. Click **OK**.

NOTE

If you want to change the color of a comment that is across a single line, then follow the steps given above, except selecting **Single-line comment** instead of **Multi-line comment**.

The color of the comments in the source code changes, as specified.

2.1.15 Is there a way to revert to the original contents of a source code file?

Yes, if undesired changes were made to source file, there is a way to revert to the previous file. In order to do this follow steps below:

1. Right-click the source file and select **Replace With > Local History f** rom the context menu.

   The **Compare** view appears.

2. Double-click **Revision Time** tab.

   The **Compare** window appears.

3. Select **Replace** .

The file is replaced.

NOTE

Another way to replace the file is to select **Replace With > Previous from Local History** from context menu and this command replaces the file with the last file that you saved.

2.1.16 How do I convert Line Delimiters to the Unix format?

The Default is Windows. To change to UNIX, select **File > Convert Line Delimiters To > Unix** from the IDE menu bar.

2.1.17 Why do I get the following error message when I try to edit a source code file?

```
File '/../../filename.c' is read-only. Do you wish to make it writable?
```

You get this error message because the file that you are trying to edit is read-only. If you do not want to make it writable then select **No** . If you want to find out the settings for this file, follow steps below:

1. From the CodeWarrior Projects window, right-click on the source file.

   A context menu appears.

2. From context menu, select **Properties** .

   The **Properties** window appears.

3. Select **Resource** .

On the right-hand side, the settings for the file appear.

2.1.18 What does an asterisk on the Editor title bar mean?

The asterisk on the editor's toolbar indicates there are unsaved changes.

2.1.19 How can I find out where a function is declared in my source code?

To find out where a function is declared in source file, follow steps below:

1. Double-click on source file.

   Source file appears in the **Editor** view.

2. In Source file, right-click the function name.

   A context menu appears.

3. From context menu, select **Open Declaration** .

The source file that contains the function declaration appears.

2.1.20 How can I open .tcl extension files in the Editor window?

In order to open .tcl extension files in the Editor window, a File Association needs to be created. To create a File Association, follow steps below:

1. From the CodeWarrior toolbar, select **Window > Preferences**.

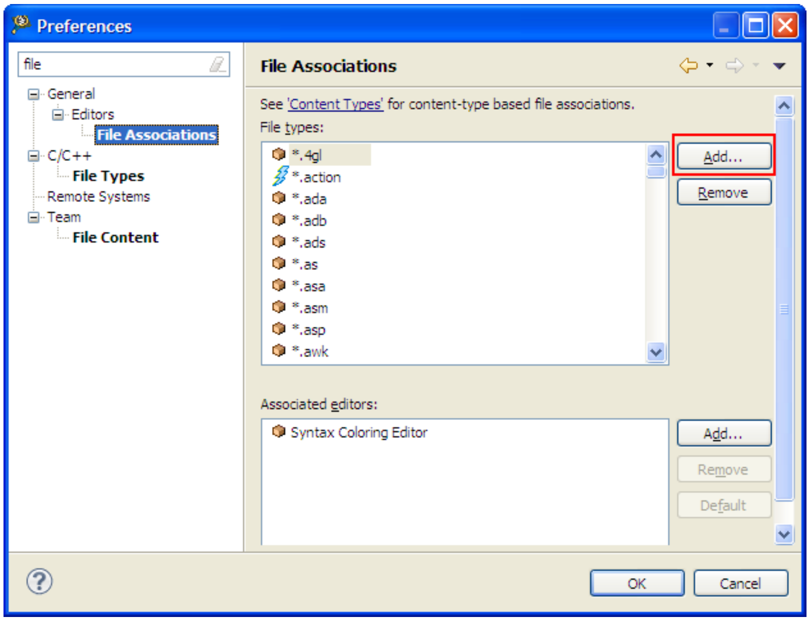
   The **Preferences** dialog box appears.

2. Select **General > Editors > File Associations**.

   The **File Associations** preferences appear on the right hand side.

3. Select **File Associations**.

   The **File Associations** pane appears.



See 'Content Types' for content-type based file associations.

File types:

*.4gl
*.action
*.ada
*.adb
*.ads
*.as
*.asa
*.asm
*.asp
*.awk

Associated editors:

Syntax Coloring Editor

Figure 2-18. Preferences Dialog Box - File Association

4. Click **Add**.

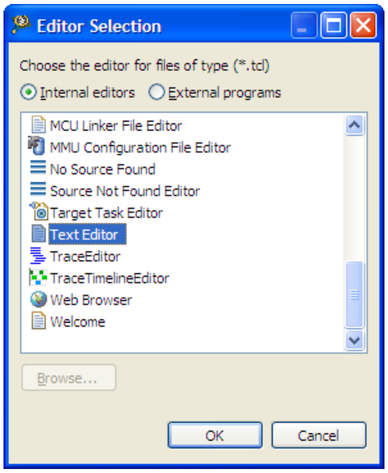
   The **New File Type** dialog box appears.

5. In the **File Type** textbox, enter `.tcl`.
6. Click **OK**.

   The `.tcl` extension appears under **File Types** .

7. Under **Associated editors** pane, Click **Add**.

   The **Editor Selection** dialog box appears.



Choose the editor for files of type (*.tcl)

● Internal editors ○ External programs

MCU Linker File Editor
MMU Configuration File Editor
No Source Found
Source Not Found Editor
Target Task Editor
Text Editor
TraceEditor
TraceTimelineEditor
Web Browser
Welcome

Figure 2-19. Editor Selection Dialog Box

8. Click **OK**.

The .tcl extension file types get associated with a Text Editor (as the following figure shows).

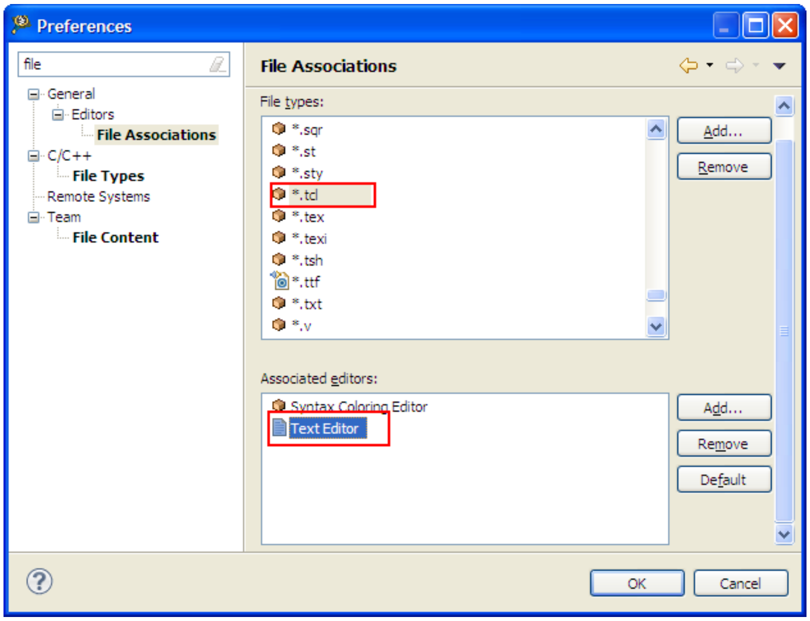


Figure 2-20. Preferences Dialog Box - File Association

9. Click **OK**.

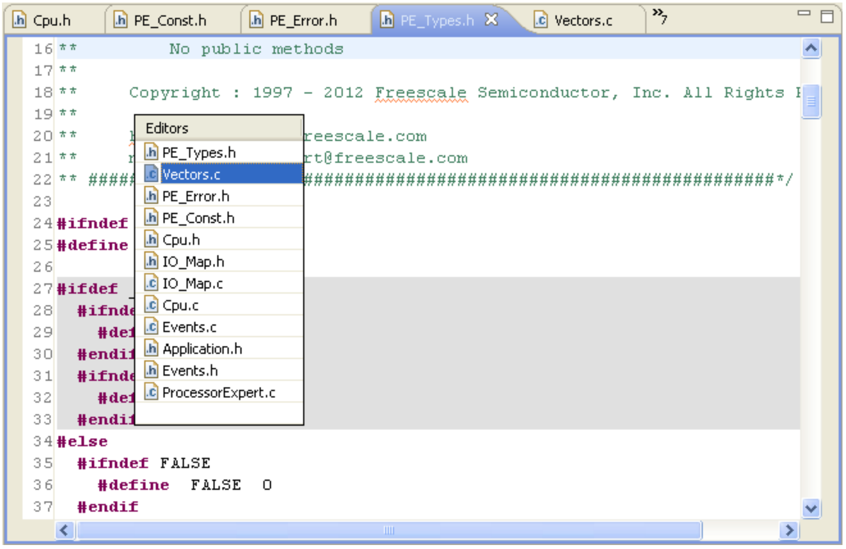
You can now open the tcl extension files in the Editor window.

2.1.21 How do I switch between the source files in the Editor view?

To switch between the source files in the Editor view while creating your project:

1. Press **Ctrl+Tab** .

   A small pop-up window appears listing all open source files in the Editor view.



```
**          No public methods
**
**     Copyright : 1997 - 2012 Freescale Semiconductor, Inc. All Rights R
**
**     h                        reescale.com
**     m                        rt@freescale.com
** ####                         ###############################################*/

#ifndef
#define

#ifdef _
  #ifnde
    #def
  #endi
  #ifnde
    #def
  #endi
#else
  #ifndef FALSE
    #define  FALSE  0
  #endif
```

Figure 2-21. Switching Between Source Files in Editor View

2. Press **Ctrl+Tab** repeatedly to iterate through the source files list.

By using this shortcut, you can quickly swap between different source files.

2.1.22 I want to edit my source files, but all the other views are using the space as well. How do I maximize the editor view to have better working area?

To maximize the views in the CodeWarrior IDE, just double-click on the view tab to toggle between maximized view and the normal view. Alternatively, right-click on the view tab and select **Maximize** from the context-menu to maximize the view. Similary, right-click on the maximized view tab select **Minimize** from the context-menu to to return to the normal view.

2.1.23 How do I remove blank lines in the Editor view?

You can remove the empty or blank lines in the Editor view by following any of these methods:

- By using the keyboard shortcut

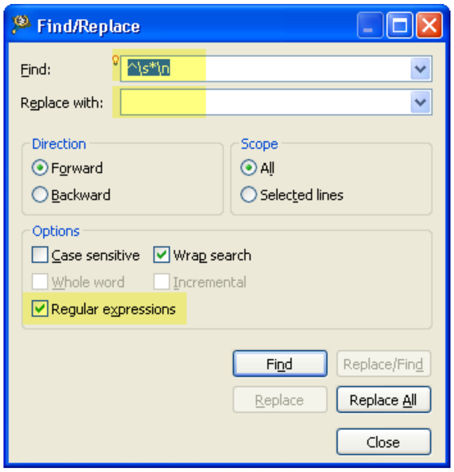
  Place the cursor on a line and press Ctrl+D (for line Delete). You can also use Ctrl+D to delete multiple selected line.

Tip

To get more commands for *deleting* the blank lines, press **Ctrl+3** , enter *delete* in the textbox, the related commands with their keyboard shortcut appears in the **Commands** list.

- By using the search and replace method
  a. Select **Edit > Find/Replace** from IDE menu bar, alternatively press **Ctrl+F** .

  The **Find/Replace** dialog box appears.



Find: ^\s*\n
Replace with:

Direction
● Forward
○ Backward

Scope
● All
○ Selected lines

Options
☐ Case sensitive ☑ Wrap search
☐ Whole word ☐ Incremental
☑ Regular expressions

Figure 2-22. Find/Replace Dialog Box

  b. Enter **^\s*\n** in the **Find** textbox and leave the **Replace with** textbox empty.
  c. Check the **Regular expressions** checkbox.
  d. Click **Replace All** .

The editor removes the blank lines.

2.2 Workbench Window

In this topic, Workbench window related FAQs are listed.

- What controls the initial layout of a view in a Workbench window?
- Is it possible to restrict the number of resources that appear on my Workbench window?
- How do I minimize a Workbench window?
- NEW How do I customize my Workspace perspectives?
- NEW How do I enable full screen mode in my CodeWarrior IDE?
- NEW How do I hide the toolbar from the IDE for better visibility of other views that I am currently working with?

- NEW How can I view all keyboard shortcuts available in the CodeWarrior IDE?
- NEW How do I edit the keyboard shortcuts in the CodeWarrior IDE?

2.2.1 What controls the initial layout of a view in a Workbench window?

A perspective defines the initial layout of the views in a workbench window. One workbench window contains many perspectives. Each of the perspectives is task oriented.

2.2.2 Is it possible to restrict the number of resources that appear on my Workbench window?

Yes, working sets help reduce the number of resources displayed. A working set is a group of elements you want to display.

To create or define a new working set, perform the following steps.

1. Click ▽ in the **CodeWarrior Projects** view or **Project Explorer** view toolbar.

   The pull-down menu appears.

2. Select the **Select Working Set** command.

   The **Select Working Set** dialog box appears.

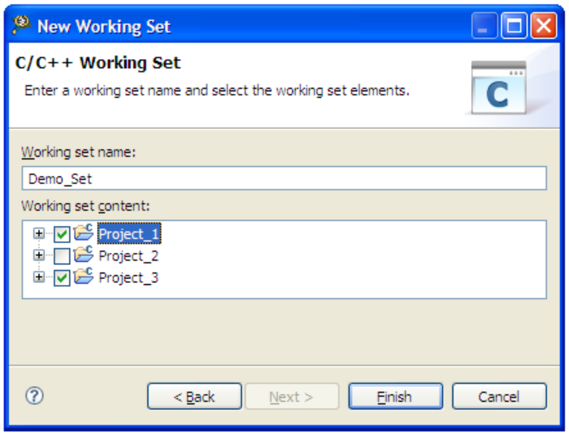
3. Click the **New** button.

   The **New Working Set** wizard appears.

4. From the **Working set type** options, select the appropriate working set. For example, select `C/C++`.
5. Click **Next**.

   The **<selected> Working Set** page appears. For example, `C/C++ Working Set`.

6. In the **Working set name** field, enter an appropriate name for the working set. For example, `Demo_Set`.
7. In the **Working set content** tree, check the items you want in your view. For example, check `Project_1` and `Project_3` (refer to the image listed below).



New Working Set

C/C++ Working Set

Enter a working set name and select the working set elements.

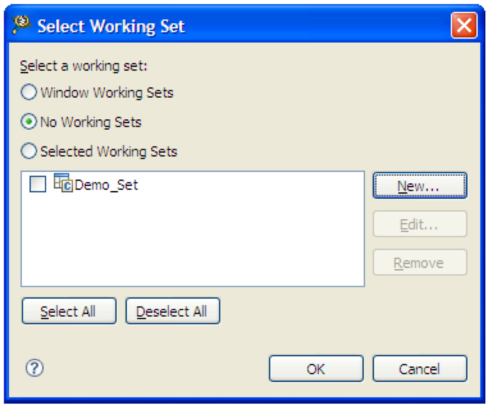
Working set name: Demo_Set

Working set content:

☑ Project_1
□ Project_2
☑ Project_3

Figure 2-23. New Working Set Wizard - C/C++ Working Set Page

8. Click **Finish** to close the **New Working Set** wizard.
9. The **Select Working Set** dialog box reappears with the newly created working set.



Select Working Set

Select a working set:

○ Window Working Sets
● No Working Sets
○ Selected Working Sets

□ Demo_Set

Figure 2-24. Select Working Set Dialog Box

10. Check the checkbox corresponding to the desired working set and click **OK**.

The **CodeWarrior Projects** view or **Project Explorer** view will now display the selected resources only.

NOTE

To deselect an active working set, select the **Deselect Working Set** command from the view pull-down menu.

NOTE

To edit an active working set, select the **Edit Active Working Set** command from the view pull-down menu.

2.2.3 How do I minimize a Workbench window?

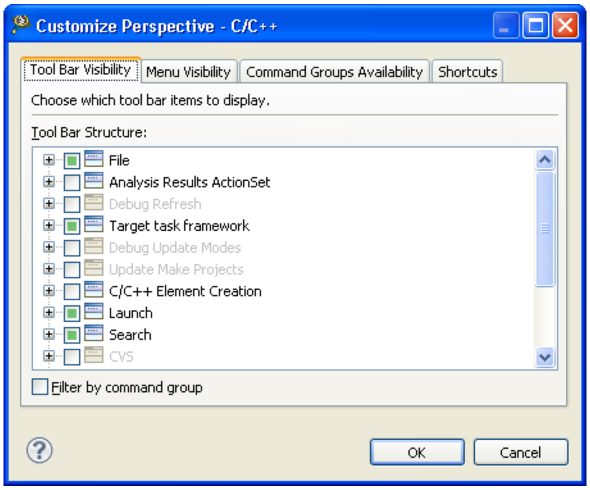
When initiated for the first time, the Workbench window tends to appear in a peculiar state. The window seems to stretch to fit the entire screen but might not be maximized. This generally happens, because the window takes the dimensions of the last application window you opened. To minimize such a window, ensure that the **Maximize** button indicates an unmaximized window. Next, drag the window edge (the cursor will appear as a left-right arrow ↔) to the desired width.

2.2.4 How do I customize my Workspace perspectives?

To customize your Workspace perspectives:

1. Create a new workspace. This way you have a clean start for customization and no changes from an existing modified workspace.
2. Close the views that you do not want to use for your project manually.
3. For customizing the buttons and menus visibility in the desired perspective:
   a. Select **Window > Customize Perspective** from the IDE menu bar.

      The **Customize Perspective - <***perspective name***>** dialog box appears.



Choose which tool bar items to display.

Tool Bar Structure:

- ☑ File
- □ Analysis Results ActionSet
- □ Debug Refresh
- ☑ Target task framework
- □ Debug Update Modes
- □ Update Make Projects
- □ C/C++ Element Creation
- ☑ Launch
- ☑ Search
- □ CVS

□ Filter by command group

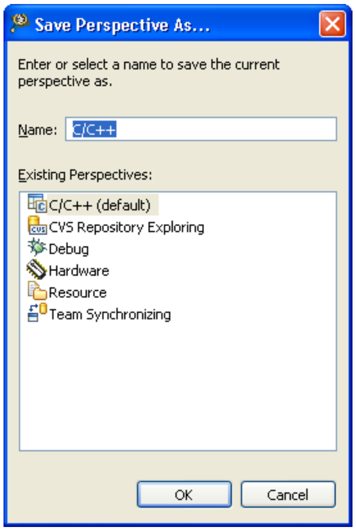
Figure 2-25. Customize Perspective - <perspective name> Dialog Box

   b. Click the desired tab to select the desired settings for tool bar, menu, command groups and shortcuts visibility.
   c. Click **OK**.

      The **Customize Perspective - <***perspective name***>** dialog box closes.

4. To save the changes you have made to the IDE's perspective, select **Window > Save Perspective As** from the IDE menu bar.

   The **Save Perspective As** dialog box appears.



Save Perspective As...

Enter or select a name to save the current perspective as.

Name: C/C++

Existing Perspectives:

- C/C++ (default)
- CVS Repository Exploring
- Debug
- Hardware
- Resource
- Team Synchronizing

Figure 2-26. Save Perspective As Dialog Box

5. Select **OK**.
6. The **Overwrite Perspective** dialog box appears.
7. Click **Yes** to overwrite the previous default setting with the changes you made.

2.2.5 How do I enable full screen mode in my CodeWarrior IDE?

To change your CodeWarrior IDE to full screen mode:

1. The Eclipse full screen plugin is available at http://code.google.com/p/eclipse-fullscreen/ web address.
2. Download the `zip` file.
3. Extract the `cn.pande.eclipsex.fullscreen_<version>.jar` file from the `zip` file you have downloaded.
4. Copy the extracted `jar` file in the <*CWInstallDir*>`\CW MCU v10.x\eclipse\plugins` folder, where <*CWInstallDir*> is the installation directory of your CodeWarrior software.

   The full screen mode option is now available in your CodeWarrior Eclipse IDE.

To use the full screen mode feature:

1. Open the CodeWarrior IDE.
2. Select **Window > Full Screen** from the IDE menu bar, alternatively, press **Ctrl+Alt +Z** to switch to the full screen mode.

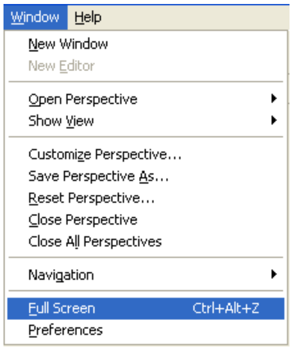
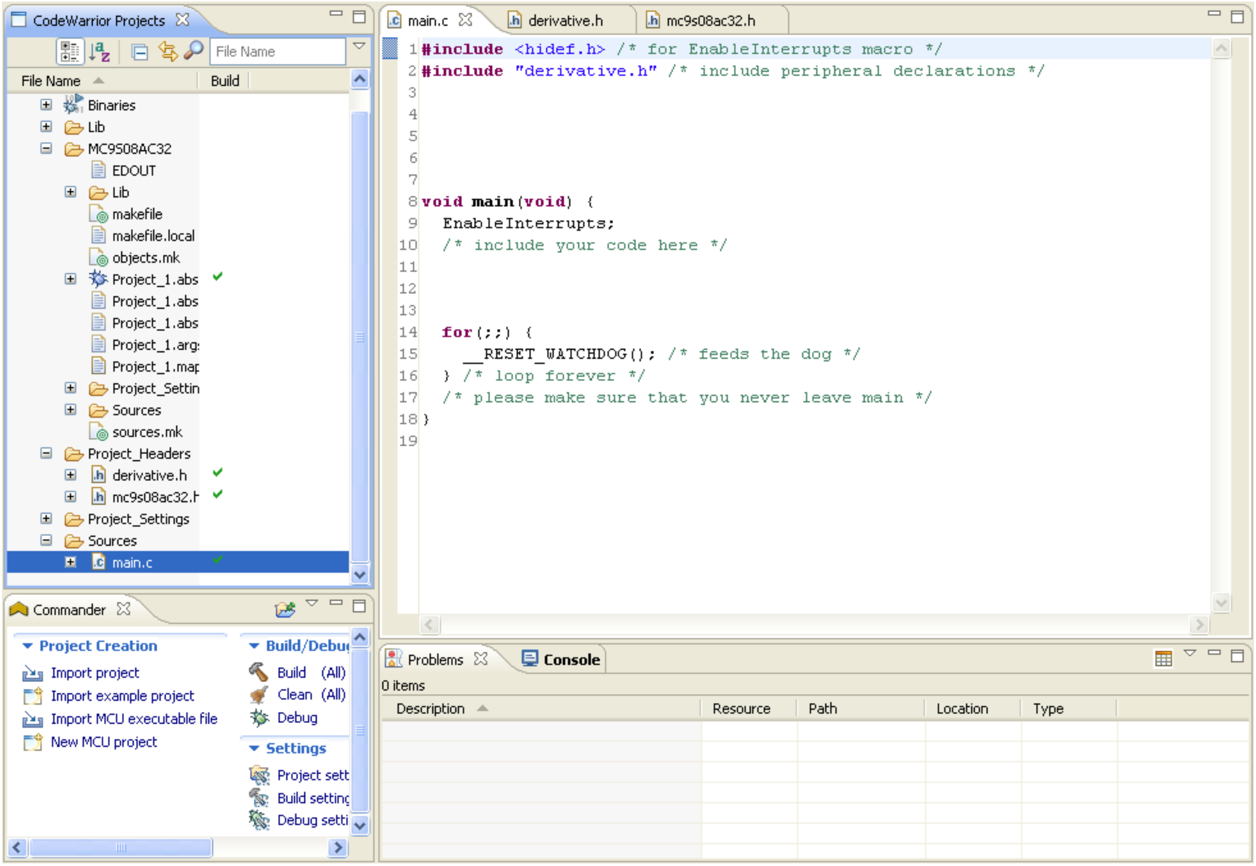


Figure 2-27. Full Screen Menu Option

The CodeWarrior Eclipse IDE switches to full screen mode, hiding the toolbar, menu bar and status bar from the screen.



```
#include <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */





void main(void) {
  EnableInterrupts;
  /* include your code here */



  for(;;) {
    __RESET_WATCHDOG(); /* feeds the dog */
  } /* loop forever */
  /* please make sure that you never leave main */
}

```

0 items

Description	Resource	Path	Location	Type

Figure 2-28. CodeWarrior Eclipse IDE - Full Screen Mode

3. Press **Esc** , or **Ctrl+Alt+Z** to exit the full screen mode.

   The CodeWarrior Eclipse IDE switches back to the normal screen mode.

NOTE

The **Ctrl+Alt+Z** is the default shortcut to toggle between *normal* and the *full screen* mode for the CodeWarrior Eclipse IDE.

2.2.6 How do I hide the toolbar from the IDE for better visibility of other views that I am currently working with?

To hide the toolbar from the IDE, right-click on the toolbar and select **Hide Toolbar** from the context-menu. The following figure shows the IDE toolbar context-menu:

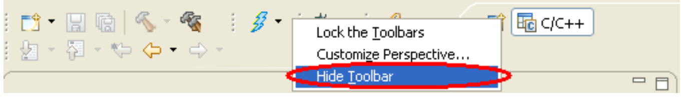


Figure 2-29. Hide Toolbar - Toolbar Context-menu Option

To display the toolbar again in the IDE, select **Window > Show Toolbar** from the IDE menu bar. The following figure shows the **Show Toolbar** menu option:

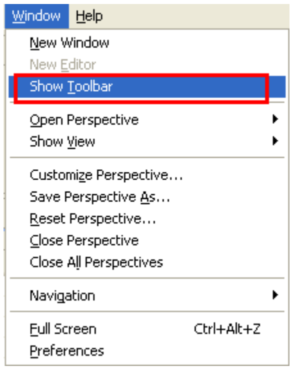


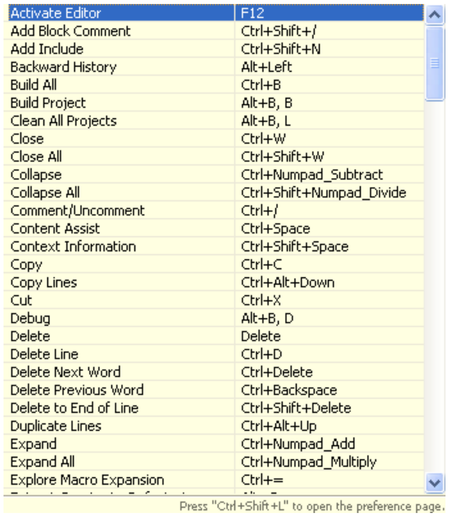
Figure 2-30. Show Toolbar - Menu Option

2.2.7 How can I view all keyboard shortcuts available in the CodeWarrior IDE?

To view all keyboard shortcuts available in the CodeWarrior IDE:

1. Open the CodeWarrior IDE.
2. Press **Ctrl+Shift+L** .

The drop-down list containing the commands and their keyboard shortcuts appears in the IDE.



Activate Editor	F12
Add Block Comment	Ctrl+Shift+/
Add Include	Ctrl+Shift+N
Backward History	Alt+Left
Build All	Ctrl+B
Build Project	Alt+B, B
Clean All Projects	Alt+B, L
Close	Ctrl+W
Close All	Ctrl+Shift+W
Collapse	Ctrl+Numpad_Subtract
Collapse All	Ctrl+Shift+Numpad_Divide
Comment/Uncomment	Ctrl+/
Content Assist	Ctrl+Space
Context Information	Ctrl+Shift+Space
Copy	Ctrl+C
Copy Lines	Ctrl+Alt+Down
Cut	Ctrl+X
Debug	Alt+B, D
Delete	Delete
Delete Line	Ctrl+D
Delete Next Word	Ctrl+Delete
Delete Previous Word	Ctrl+Backspace
Delete to End of Line	Ctrl+Shift+Delete
Duplicate Lines	Ctrl+Alt+Up
Expand	Ctrl+Numpad_Add
Expand All	Ctrl+Numpad_Multiply
Explore Macro Expansion	Ctrl+=

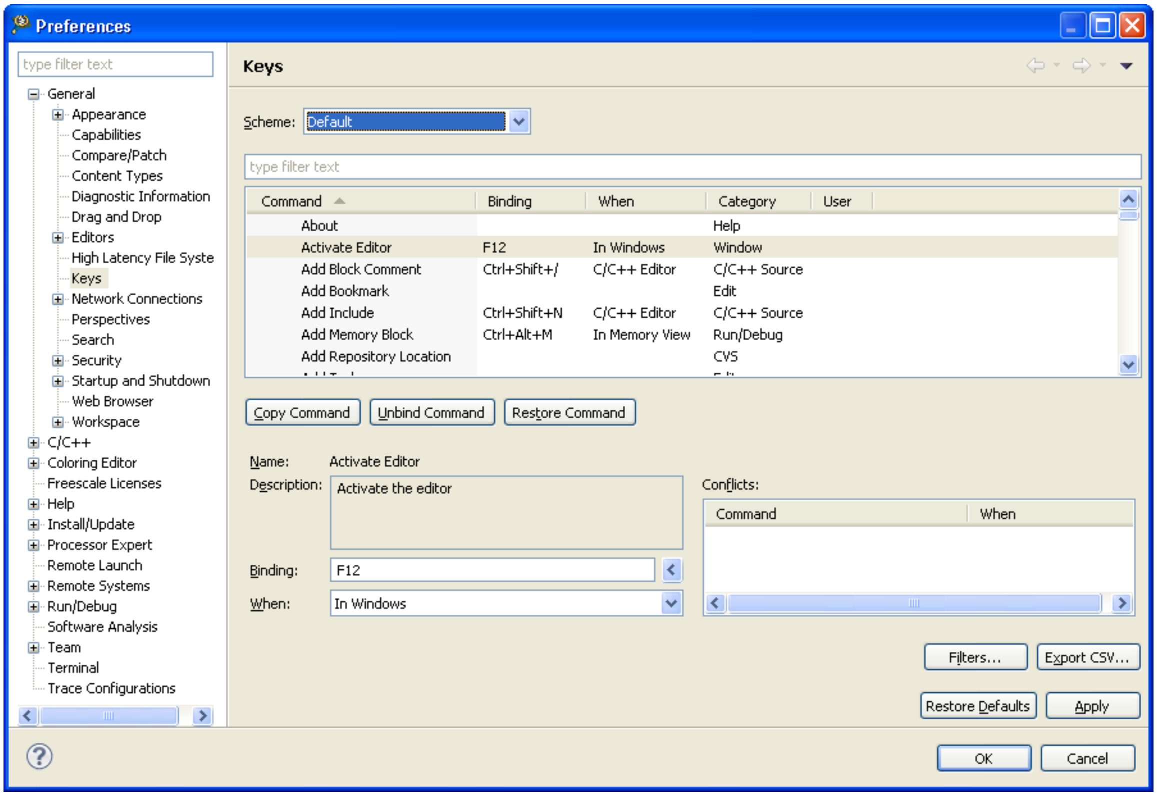
Press "Ctrl+Shift+L" to open the preference page.

Figure 2-31. Keyboard Shortcuts Drop-down List

2.2.8 How do I edit the keyboard shortcuts in the CodeWarrior IDE?

To edit the keyboard shortcuts available in the CodeWarrior IDE:

1. Press **Ctrl+Shift+L** twice.
2. The **Preferences** dialog box with the **Keys** page appears.



Scheme: Default

Command	Binding	When	Category	User
About			Help	
Activate Editor	F12	In Windows	Window	
Add Block Comment	Ctrl+Shift+/	C/C++ Editor	C/C++ Source	
Add Bookmark			Edit	
Add Include	Ctrl+Shift+N	C/C++ Editor	C/C++ Source	
Add Memory Block	Ctrl+Alt+M	In Memory View	Run/Debug	
Add Repository Location			CVS	

Name: Activate Editor

Description: Activate the editor

Binding: F12

When: In Windows

Figure 2-32. Preferences Dialog Box - Keys Page

3. Select the desired command.
4. Click the **Copy Command** , **Unbind Command** , and **Restore Command** buttons to copy, unbind or restore the selected command respectively.
5. Click **Apply** .
6. Click **Ok** .

The IDE saves and applies the changes you made.

2.3 CVS

In this topic, CVS related FAQs are listed.

- Does the CodeWarrior IDE support version control systems?
- How can I add my project to CVS from within the CodeWarrior IDE?

2.3.1 Does the CodeWarrior IDE support version control systems?

Yes, CodeWarrior IDE does provide support for the version control systems. One of the version control system supported by the CodeWarrior IDE is CVS.

To use CVS with the CodeWarrior IDE:

1. From the IDE menu bar, select **Window > Show View > Other**.

   The **Show View** dialog box appears.

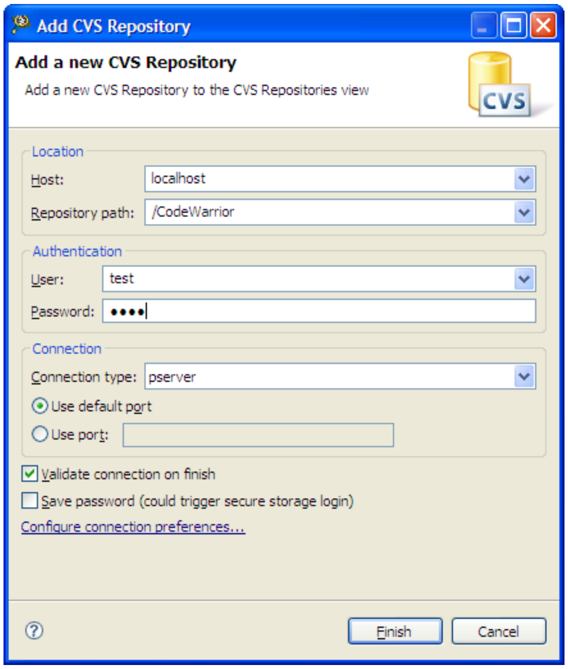
2. Expand the **CVS** tree control and select **CVS Repositories**.
3. Click **OK**.

   The **CVS Repositories** view appears.

4. Click the **Add CVS Repository** command on the **CVS Repositories** view toolbar.

   The **Add CVS Repository** dialog box appears (refer to the image, *Add CVS Repository Dialog Box* listed below).

5. Enter host name and path of the repository that you want to browse through in the **Host** and **Repository path** text boxes respectively.
6. Enter user name and password in the **User** and **Password** text boxes respectively.



Add CVS Repository

Add a new CVS Repository

Add a new CVS Repository to the CVS Repositories view

Location

Host: localhost

Repository path: /CodeWarrior

Authentication

User: test

Password: ••••

Connection

Connection type: pserver

● Use default port

○ Use port:

☑ Validate connection on finish

□ Save password (could trigger secure storage login)

Figure 2-33. Add CVS Repository Dialog Box

7. Click **Finish**.

   You can browse through the specified repository in the **CVS Repository** view.

8. Right-click the desired folder and select the desired action from the context menu to perform CVS actions, such as **Check Out** (refer to the image listed below).

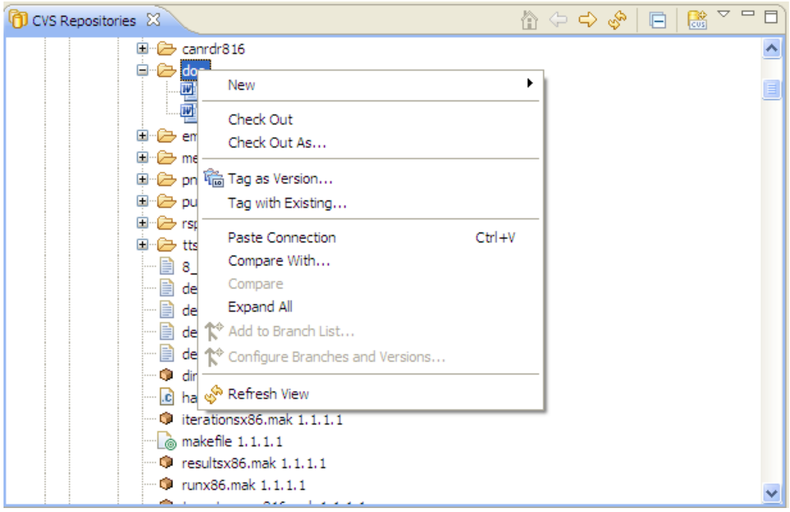


Figure 2-34. CVS Repository View

You can now use CVS with the CodeWarrior IDE.

2.3.2 How can I add my project to CVS from within the CodeWarrior IDE?

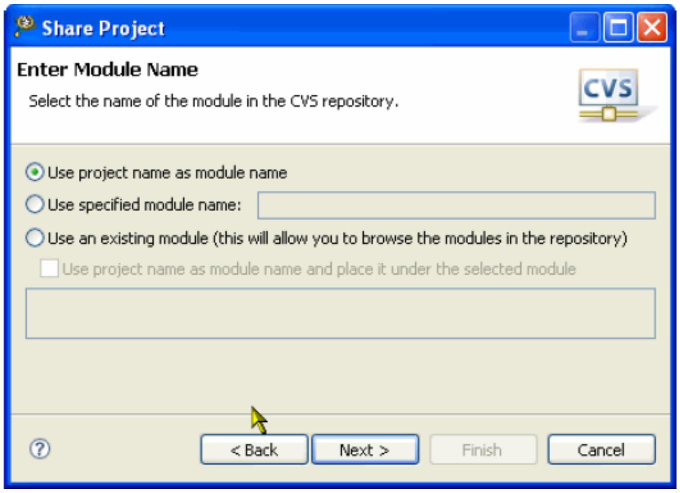
To add a project to the CVS:

1. Right-click the project in the **CodeWarrior Projects** view and select **Team > ShareProject**.

   If you have already configured a CVS repository with the CodeWarrior IDE, the **Share Projectwith CVS Repository** page of the **Share Project** wizard appears. Otherwise, the **Enter Repository Location Information** page appears.

2. Specify the repository to which you want to add the project and click **Next**.

   The **Enter Module Name** page appears.



Share Project

Enter Module Name

Select the name of the module in the CVS repository.

● Use project name as module name

○ Use specified module name:

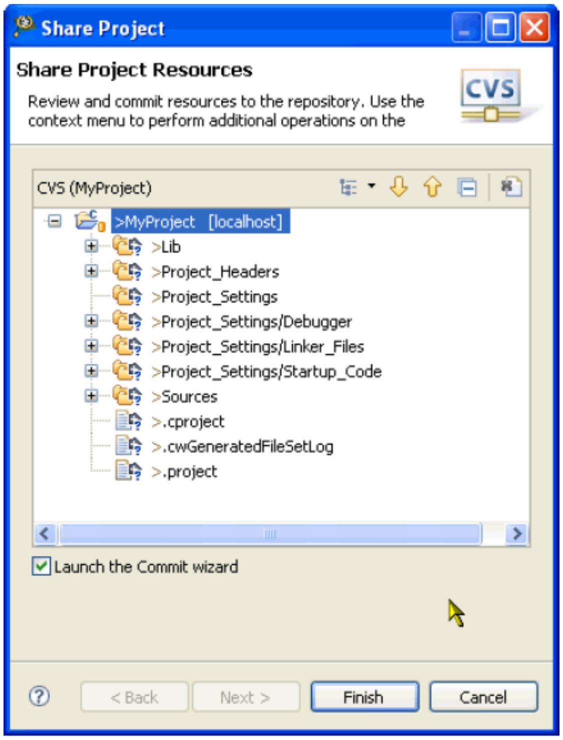
○ Use an existing module (this will allow you to browse the modules in the repository)

□ Use project name as module name and place it under the selected module

Figure 2-35. Enter Module Name Page

3. Specify name of the module in the CVS repository and click **Next**.

   The **Share Project Resources** page appears.



Share Project

Share Project Resources

Review and commit resources to the repository. Use the context menu to perform additional operations on the

CVS (MyProject)

```
>MyProject  [localhost]
    >Lib
    >Project_Headers
    >Project_Settings
    >Project_Settings/Debugger
    >Project_Settings/Linker_Files
    >Project_Settings/Startup_Code
    >Sources
    >.cproject
    >.cwGeneratedFileSetLog
    >.project
```

☑ Launch the Commit wizard

Figure 2-36. Share Project Resources Page

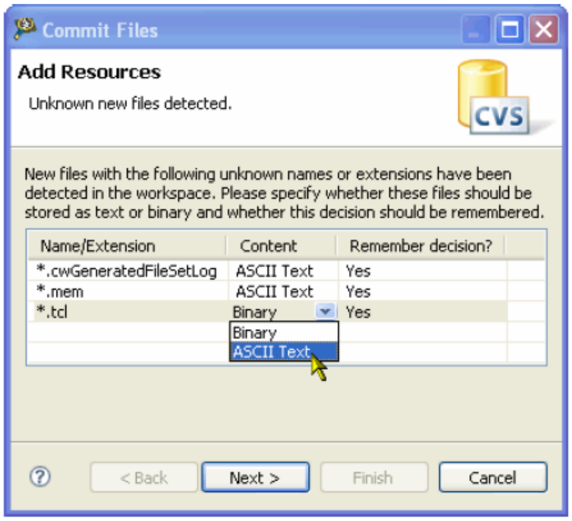
4. Select the files that you want to add to the CVS.

   If the project includes generated file, then you may not want to add them to the CVS module. You can add such files to the `.cvsignore` list. The `.cvsignore` is a special text file which specifies all the files that should be ignored/filtered out. The `.cvsignore` file can also use wildcards, such as `*.tmp` for filtering.

5. Check the **Launch the Commit wizard** checkbox and click **Finish**.

The **Commit Files** wizard starts displaying the list of unknown files.

6. Specify whether these files are binary or ASCII Text. In this example, CVS does not identify three file extensions and assumes them as binary files. However, as `*.mem`, `*.tcl`, and `*.cwGeneratedFileSetLog` are text files, the file type is changed to ASCII Text.



Add Resources

Unknown new files detected.

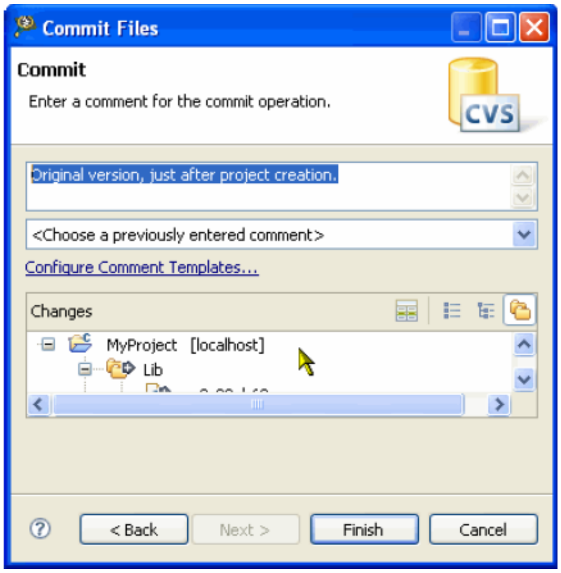
New files with the following unknown names or extensions have been detected in the workspace. Please specify whether these files should be stored as text or binary and whether this decision should be remembered.

Name/Extension	Content	Remember decision?
*.cwGeneratedFileSetLog	ASCII Text	Yes
*.mem	ASCII Text	Yes
*.tcl	Binary Binary ASCII Text	Yes

Figure 2-37. Specify How to Store Unknown Files

7. Click **Next**.

The **Commit** page appears.



Commit

Enter a comment for the commit operation.

Original version, just after project creation.

<Choose a previously entered comment>

Configure Comment Templates...

Figure 2-38. Commit Page

8. Enter appropriate comment for the commit action and click **Finish**.

The icons of the files and folders in the project change, which specify that the project is now under CVS control.

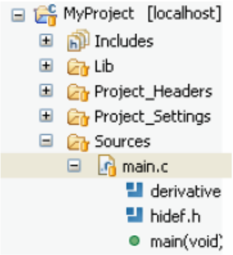
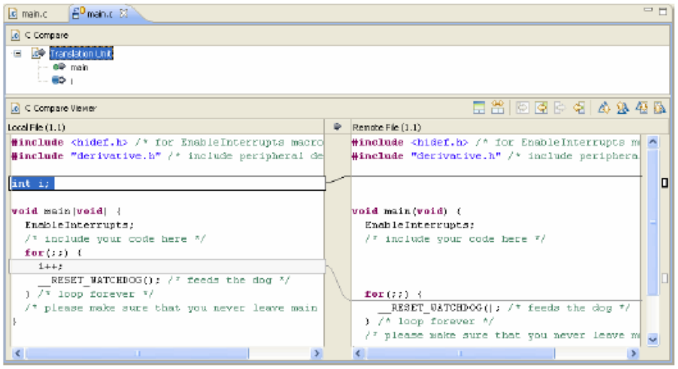


Figure 2-39. Project Under CVS Control

Now, if you have done changes to a source file, and you need to view the differences between your local copy and the repository copy, perform these steps.

1. Right-click the modified file and select **Team > Synchronize with Repository**. The **C Compare Viewer** opens displaying comparison between the files (refer to the image listed below).



```
#include <hidef.h> /* for EnableInterrupts macro
#include "derivative.h" /* include peripheral de

int i;

void main(void) {
  EnableInterrupts;
  /* include your code here */
  for(;;) {
    i++;
    __RESET_WATCHDOG(); /* feeds the dog */
  } /* loop forever */
  /* please make sure that you never leave main
}
```

```
#include <hidef.h> /* for EnableInterrupts m
#include "derivative.h" /* include periphera


void main(void) {
  EnableInterrupts;
  /* include your code here */


  for(;;) {
    __RESET_WATCHDOG(); /* feeds the dog */
  } /* loop forever */
  /* please make sure that you never leave m
```

Figure 2-40. C Compare Viewer

2. Review the changes. If you are ready to commit the changes, right-click the file and select **Team > Commit**.
3. Enter the appropriate comment and click **Finish**.

You can also perform the synchronization in the **Synchronize** view in the **Team Synchronizing** perspective (refer to the image listed below).

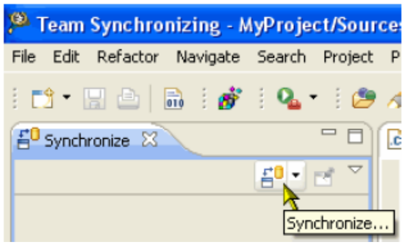


Figure 2-41. Synchronize View

The **Synchorize** view displays what has been changed either locally or remotely. For example, in the **Synchorize** view (in the following figure), the file `main.c` is on revision 1.2 with some outgoing changes, that is some changes are done locally in the file, and there is a new file `test.c` which is not commited yet.

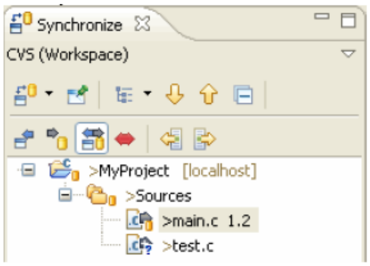


Figure 2-42. Synchronize View Displaying File Revisions

2.4 Search and Replace Action

In this topic, Search and Replace related FAQs are listed.

- How do I perform a simple search in the CodeWarrior IDE?
- Is it possible to preview the changes of the Replace action?
- Is there a way to change the variable or the function name in a CodeWarrior project?
- How can I search for specific files in a workspace?
- Is there a way to replace a variable easily?
- How can I go back to the previous location if the Editor takes me to another source or header file when, for example, I select the Open Declaration option?
- Is there a shortcut to open the search window for a specified text that is equivalent to the Ctrl + Shift + M shortcut key available in the Classic CodeWarrior IDE?
- Which set of files I need to backup or store in a version control system to completely recover project settings and related debug settings?

2.4.1 How do I perform a simple search in the CodeWarrior IDE?

You can either use the **Search** menu in the menu bar or select the **Edit > Find/Replace** command. However, there is a difference. The **Search** menu displays the commands **C/C++** , **Search** , **File** , and **Text** .

- **C/C++** - Opens the search dialog on the C/C++ search page
- **Search** - Opens the search dialog for your current editor

- **File** - Opens the search dialog on the File search page
- **Text** - Opens the submenu for full-text search in given scope (workspace, project, file or working set)

Selecting the **Select** menu opens the **Search** dialog box that includes two specialized tabs: **File Search** and **C/C++ Search** .

Whereas, the **Find/Replace** command displays the **Find/Replace** dialog box that lets you search for an expression in the active editor, and replace the expression with a new expression.

Thus, for a simple find and replace use **Edit > Find/Replace** .

2.4.2 Is it possible to preview the changes of the Replace action?

Yes, to preview the changes of the Replace action, follow these steps:

1. Select **Search > File** (refer to the image, *Search Menu* as listed below) from the IDE menu bar.

   The **File Search** tab page of the **Search** dialog box appears (refer to the image, *Search Dialog Box* as listed below).

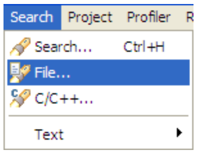
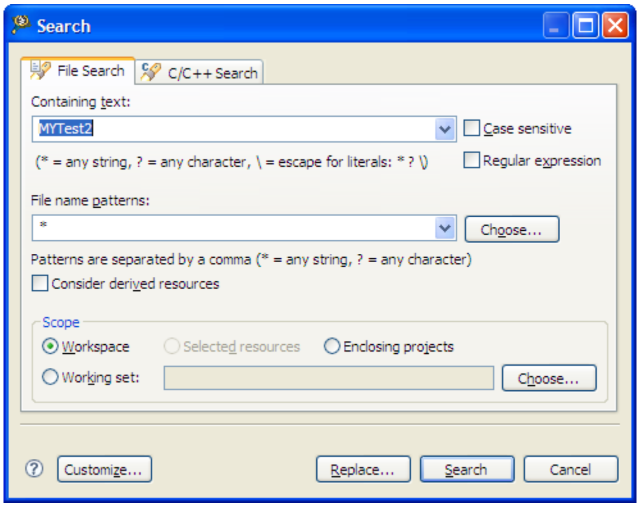


Figure 2-43. Search Menu

2. Specify the text that you want to replace in the **Containing text** text box.



Search

File Search | C/C++ Search

Containing text: MYTest2

□ Case sensitive

(* = any string, ? = any character, \ = escape for literals: * ? \)

□ Regular expression

File name patterns: *

Choose...

Patterns are separated by a comma (* = any string, ? = any character)

□ Consider derived resources

Scope

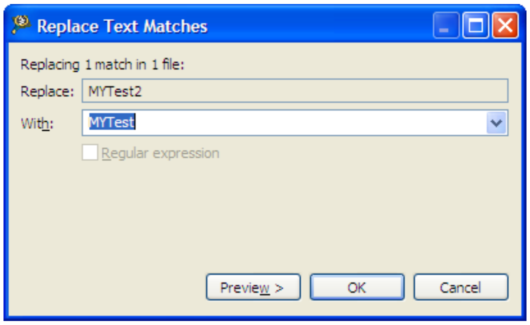
● Workspace ○ Selected resources ○ Enclosing projects

○ Working set: Choose...

Customize... | Replace... | Search | Cancel

Figure 2-44. Search Dialog Box

3. Click the **Replace** button. The **Replace Text Matches** dialog box appears.
4. In the **With** text box, enter the text with which you want to replace the text entered in the **Replace** text box.



Replace Text Matches

Replacing 1 match in 1 file:

Replace: MYTest2

With: MYTest

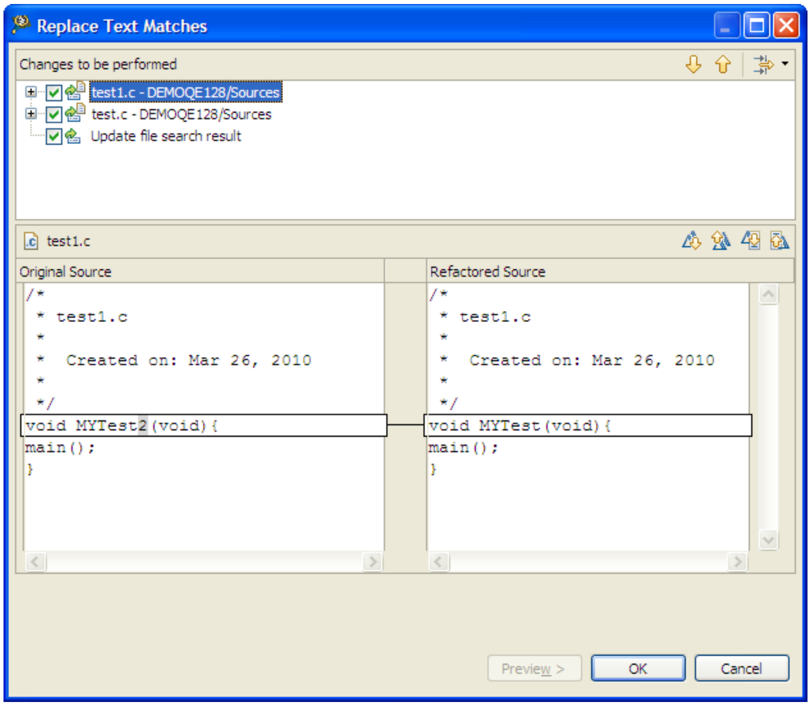
□ Regular expression

Preview > | OK | Cancel

Figure 2-45. Replace Text Matches Dialog Box

5. Click the **Preview** button.

   The dialog box lists the files to which the changes will be performed as a result of the replace action. You can preview each of the change and check or clear a file to accept or reject the change as required.



Original Source:

```
/*
 * test1.c
 *
 *  Created on: Mar 26, 2010
 *
 */
void MYTest2(void){
main();
}
```

Refactored Source:

```
/*
 * test1.c
 *
 *  Created on: Mar 26, 2010
 *
 */
void MYTest(void){
main();
}
```

Figure 2-46. Preview Replace Action

6. Click **OK**.

The **Replace Text Matches** dialog box is closed.

The changes as a result of the replace action are performed to the selected files.

2.4.3 Is there a way to change the variable or the function name in a CodeWarrior project?

Yes, you can use the Refactoring feature of the CodeWarrior IDE to replace a function or a variable name.

1. Select the variable or function and select **Refactor > Rename** from the IDE menu bar.

   The **Rename global variable**`<*variable name*>' dialog box appears.

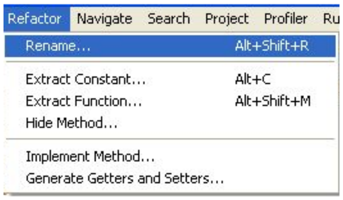
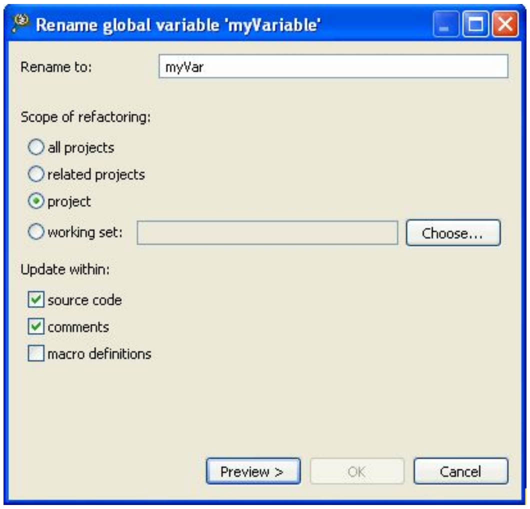


Figure 2-47. Refactor Menu



Rename global variable 'myVariable'

Rename to: myVar

Scope of refactoring:

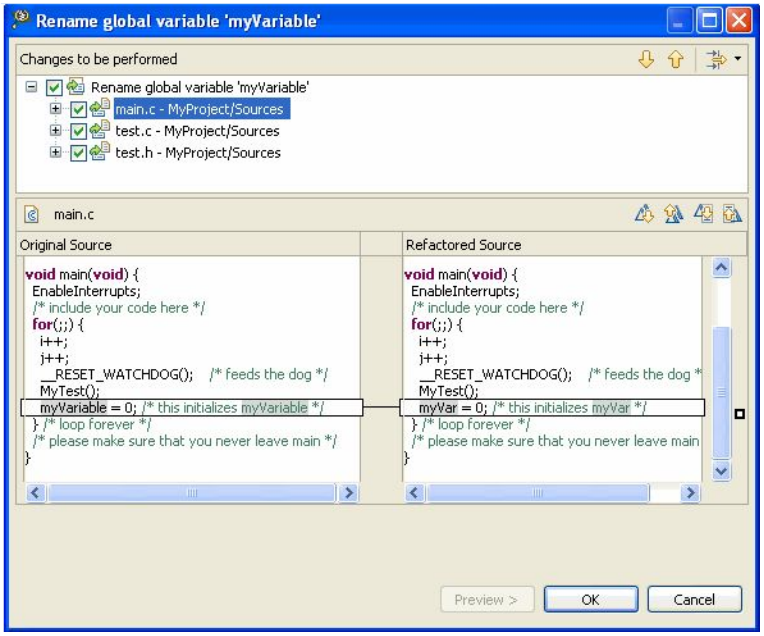
- ○ all projects
- ○ related projects
- ● project
- ○ working set: Choose...

Update within:

- ☑ source code
- ☑ comments
- ☐ macro definitions

Figure 2-48. Rename global variable Dialog Box

2. Specify the new name for the variable.
3. Specify the scope of refactoring, such as all projects, related projects, project, or working set.
4. Specify where in the source file you want to make the changes, such as source code, comments, and/or macro definitions.
5. Click the **Preview** button to preview the changes.



Rename global variable 'myVariable'

Changes to be performed

- ☑ Rename global variable 'myVariable'
  - ☑ main.c - MyProject/Sources
  - ☑ test.c - MyProject/Sources
  - ☑ test.h - MyProject/Sources

main.c

Original Source

```
void main(void) {
 EnableInterrupts;
 /* include your code here */
 for(;;) {
  i++;
  j++;
  __RESET_WATCHDOG();    /* feeds the dog */
  MyTest();
  myVariable = 0; /* this initializes myVariable */
 } /* loop forever */
 /* please make sure that you never leave main */
}
```

Refactored Source

```
void main(void) {
 EnableInterrupts;
 /* include your code here */
 for(;;) {
  i++;
  j++;
  __RESET_WATCHDOG();    /* feeds the dog *
  MyTest();
  myVar = 0; /* this initializes myVar */
 } /* loop forever */
 /* please make sure that you never leave main
}
```

Figure 2-49. Preview Replace Action

6. Check or clear a file to accept or reject a change as required.
7. Click **OK** .

The specified changes are performed.

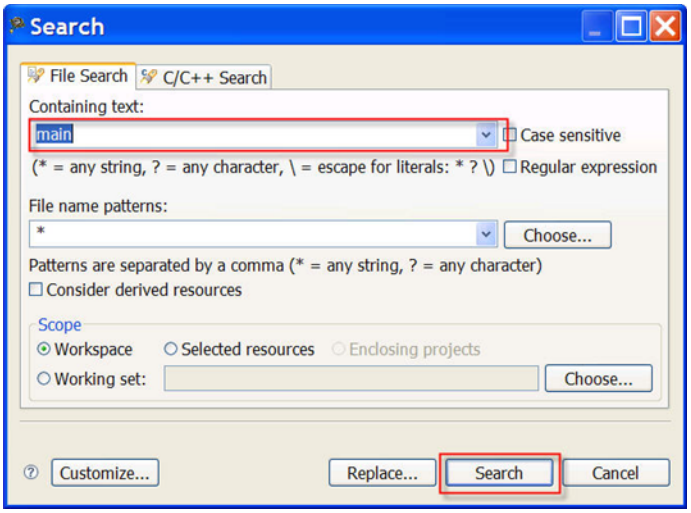
2.4.4 How can I search for specific files in a workspace?

To search for specific files in a workspace, follow these steps:

1. From the IDE menu bar, select the **Search > File...** option.

   The **Search** dialog box appears.

2. Specify the search string in the **Containing Text** field. Specify other search options according to your requirements (refer to the image listed below).



Containing text: main

☐ Case sensitive

(* = any string, ? = any character, \ = escape for literals: * ? \)

☐ Regular expression

File name patterns: *

Patterns are separated by a comma (* = any string, ? = any character)

☐ Consider derived resources

Scope

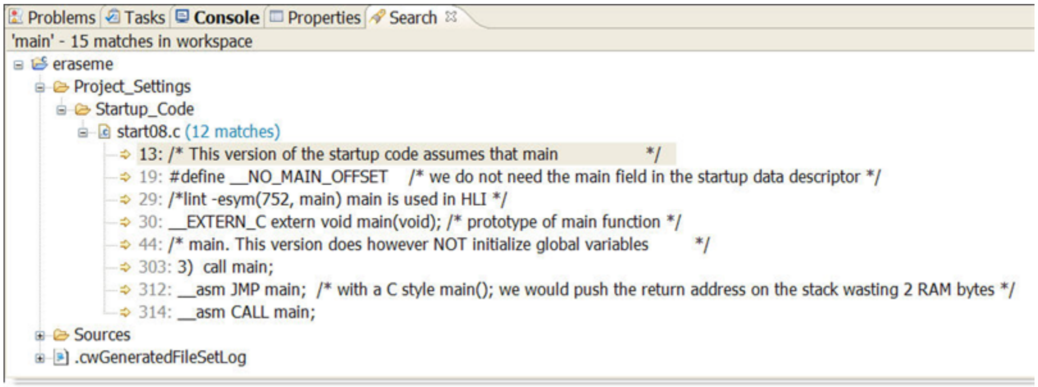
● Workspace ○ Selected resources ○ Enclosing projects

○ Working set:

Figure 2-50. Search Dialog Box

3. Click the **Search** button.

The **Search** view appears (as the figure listed below shows). The **Search** view displays the results of your search. Right-click any item in the **Search** view to open a pop-up menu that allows you to remove items from the list, copy search results to the clipboard, or rerun the search.



```
'main' - 15 matches in workspace
eraseme
  Project_Settings
    Startup_Code
      start08.c (12 matches)
        13: /* This version of the startup code assumes that main           */
        19: #define __NO_MAIN_OFFSET    /* we do not need the main field in the startup data descriptor */
        29: /*lint -esym(752, main) main is used in HLI */
        30: __EXTERN_C extern void main(void); /* prototype of main function */
        44: /* main. This version does however NOT initialize global variables      */
        303: 3)  call main;
        312: __asm JMP main;  /* with a C style main(); we would push the return address on the stack wasting 2 RAM bytes */
        314: __asm CALL main;
  Sources
  .cwGeneratedFileSetLog
```

Figure 2-51. Search View

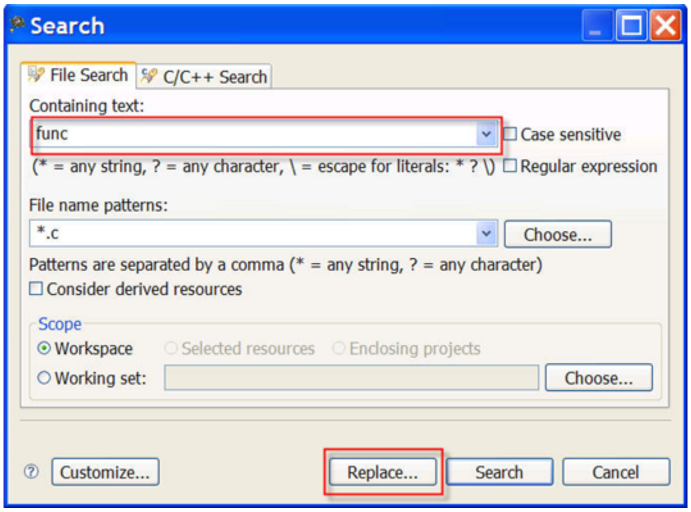
2.4.5 Is there a way to replace a variable easily?

Yes, to replace a variable, follow these steps:

1. From the IDE menu bar, select the **Search > File...** option.

   The **Search** dialog box appears.

2. Specify the search string in the **Containing Text** field. Specify other search options according to your requirements (refer to the image listed below).



Containing text: func

☐ Case sensitive

(* = any string, ? = any character, \ = escape for literals: * ? \) ☐ Regular expression

File name patterns: *.c

Patterns are separated by a comma (* = any string, ? = any character)

☐ Consider derived resources

Scope

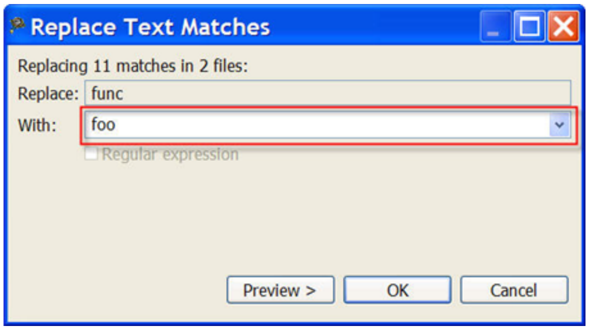
● Workspace ○ Selected resources ○ Enclosing projects

○ Working set:

Figure 2-52. Search Dialog Box

3. Click **Replace**.

   The **Replace Text Matches** dialog box appears.



Replacing 11 matches in 2 files:

Replace: func

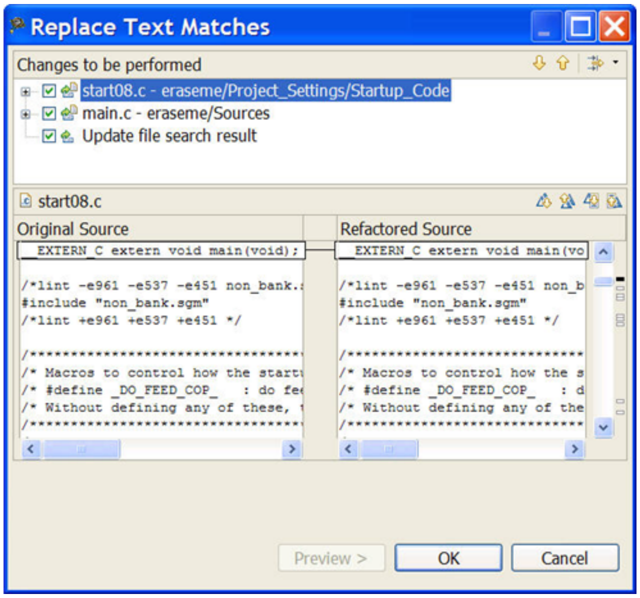
With: foo

□ Regular expression

Figure 2-53. Replace Text Matches Window

4. Click **Preview**.

The **Replace Text Matches** window appears (refer to the image listed below). This window gives lets you preview the changes that will be made. This allows you to inspect each proposed change and accept or deny it individually.



Changes to be performed

- ☑ start08.c - eraseme/Project_Settings/Startup_Code
- ☑ main.c - eraseme/Sources
- ☑ Update file search result

start08.c

Original Source

```
__EXTERN_C extern void main(void);

/*lint -e961 -e537 -e451 non_bank.:
#include "non_bank.sgm"
/*lint +e961 +e537 +e451 */

/**********************************
/* Macros to control how the start
/* #define _DO_FEED_COP_    : do fe
/* Without defining any of these, 
/**********************************
```

Refactored Source

```
__EXTERN_C extern void main(vo

/*lint -e961 -e537 -e451 non_b
#include "non_bank.sgm"
/*lint +e961 +e537 +e451 */

/*****************************
/* Macros to control how the s
/* #define _DO_FEED_COP_    : d
/* Without defining any of the
/*****************************
```

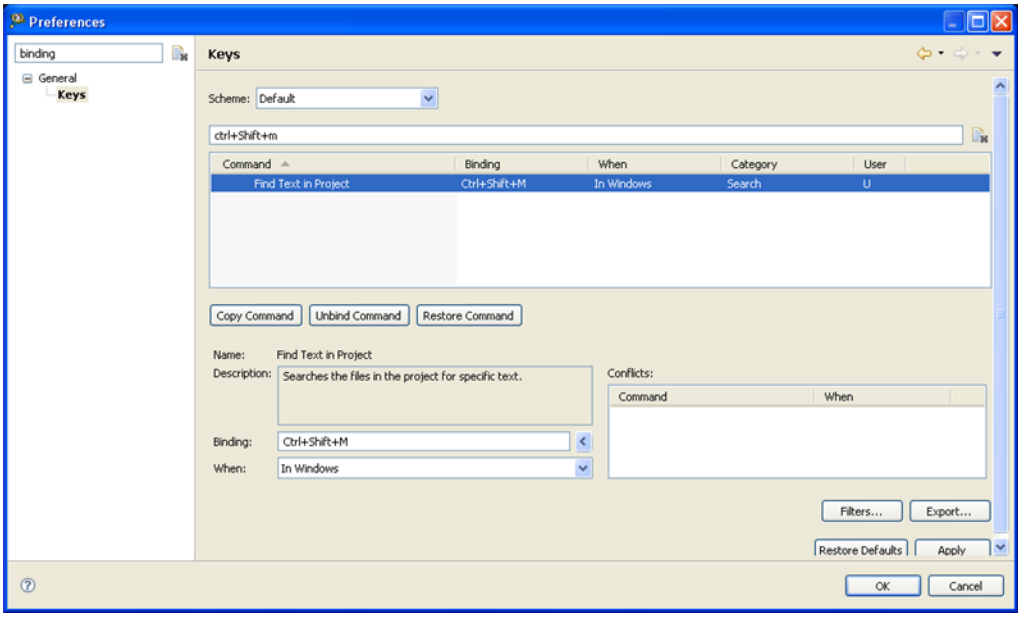
Figure 2-54. Replace Text Matches Window

2.4.6 How can I go back to the previous location if the Editor takes me to another source or header file when, for example, I select the Open Declaration option?

Use the **Alt+left** shortcut key. This option is also listed under the **Navigation** menu.

2.4.7 Is there a shortcut to open the search window for a specified text that is equivalent to the Ctrl + Shift + M shortcut key available in the Classic CodeWarrior IDE?

You can create a custom key binding for the **Ctrl+Shift+M** shortcut key in the **Preferences > General > Keys** dialog box.



Scheme: Default

ctrl+Shift+m

Command	Binding	When	Category	User
Find Text in Project	Ctrl+Shift+M	In Windows	Search	U

Name: Find Text in Project

Description: Searches the files in the project for specific text.

Binding: Ctrl+Shift+M

When: In Windows

Conflicts:

Command	When

Figure 2-55. Preferences Dialog Box - Keys Page

NOTE

For more information on key binding, refer to the topic How can I change a key binding?

2.4.8 Which set of files I need to backup or store in a version control system to completely recover project settings and related debug settings?

The files you need to store in a version control system:

- `.project` and `.cproject` - contains the list of files and the build tool settings.
- `.ttf` - targets the needing target tasks for flash programming (ColdFire V2 and Kinetis IAR JLink)
- `.launch` - required for the debugger launch configuration

- rseHostSettingsCache.xml- contains the cached RSE settings. You need this to have the RSE settings imported.
- *.c/*.h/etc source files and linker file, if not using Processor Expert
- If you are using Processor Expert - ProcessorExpert.pe (contains all the settings), events.c/h, and the ProcessorExpert.c with user modifications

Any generated folders (like PE Generated_Code, PE Documentation or the folder where the object/make files are placed (usually named by the CPU name) can be stripped off to reduce file size.

You can also export/store the workspace settings, like editor/VCS settings/etc, by selecting, **File > Export > General > Preferences** from the IDE menu bar.

2.5 Miscellaneous

In this topic, miscellaneous FAQs related to the CodeWarrior IDE are listed.

- How can I improve the performance of the CodeWarrior IDE I am working with?
- What is the first thing that I see when I start the CodeWarrior IDE?
- What is a perspective?
- How can I find the version of the CodeWarrior that I am using?
- Is it possible to retrieve a file that I deleted accidentally from my CodeWarrior project?
- How do I update my local history settings?
- Is there a way to filter the settings in the Preferences window to find a particular setting?
- How can I change a key binding?
- Is it possible to view the change history of a source code file?
- How can I determine which header files are associated with my source code file?
- How do I view call hierarchy in my source code?
- Why the project that I just created is not visible in the CodeWarrior Projects view?
- I created a new file in Windows Explorer but the file does not appear in the CodeWarrior Projects view. Why?
- Why does the Rename option appear grayed out?
- Why duplicating a configuration in the debugger perspective does not duplicate the run configuration?
- Can I still use Ctrl+Tab keys to navigate between open windows?
- How can I change the debugger key bindings to the ones that I used to have in the Classic CodeWarrior IDE?

- How can I change the name of the executable that is generated when I build my project?
- Why the console view does not display all the warning and error messages by default?
- Is there a file that contains all the launch configuration settings?
- How the CodeWarrior Project Importer handles recursive access paths?
- Can I disassemble my source code file?
- Where exactly the disassemble file gets created?
- Why the.metadata folder in my workspace stores a huge history?
- How to have projects from different workspaces or locations in a workspace?
- How can I switch to another workspace?
- What is the purpose of the Tasks view?
- What is the purpose of the Properties view?
- What is the purpose of the Outline view?
- How can I find out if certain files contain debug information?
- How can I resolve the following error message that I get when I start the CodeWarrior IDE?
- How can I open an existing project in the CodeWarrior IDE?
- What is the Manage Configurations button in the C/C++ perspective toolbar used for? Could it be replaced by the Properties button?
- How can I start the post-build steps in the CodeWarrior IDE?
- Is it necessary to have the project name identical to the name of the directory that contains the .project file?
- Where does the CodeWarrior IDE save the debug configuration as a local file by default?
- When I save my launch configurations as a local file and then delete the project, all the local configurations get deleted as well. How can I resolve this issue?
- How can I modify and save the files in GBK encoding using the CodeWarrior IDE?
- How are the PARENT-COUNT-MyVariable definitions defined in the CodeWarrior IDE?
- Is there a way to instruct the CodeWarrior IDE to use relative paths instead of absolute ones to store the project file location in a workspace?
- NEW What is a Working Set? How do I build/compile a Working Set?
- NEW How do I search within the Working Set?
- NEW How do I manage the Working Sets?
- NEW How can I can filter the projects listed in the CodeWarrior Projects view based on Working Sets?
- NEW How can I import/export a Working Set in the CodeWarrior eclipse IDE?
- NEW What are Icon decorators?
- NEW How do I enable or disable an icon decorator?
- NEW Can the icon decorators be combined?

- NEW When I open a large file in CodeWarrior IDE, an Editor Scalability dialog box appears, listing the message, "You are opening a large file. Scalability mode has been turned on for this editor to help improve performance by disabling features requiring to parse the source file language." What does it mean?
- NEW How do I configure the Scalability settings?

2.5.1 How can I improve the performance of the CodeWarrior IDE I am working with?

Yes, you can improve the performance of your CodeWarrior IDE by disabling the features that slow down the software.

1. **Disable Indexer Options**
   a. From the CodeWarrior IDE menu bar, select **Window > Preferences**.

      The **Preferences** dialog box appears.

   b. Type **index** as the filter text to narrow down the list of preferences.
   c. Select **C/C++ > Indexer**.

      The **Indexer** page appears in the right panel of the **Preferences** dialog box.

   d. Clear the **Enable indexer** checkbox.
2. **Disable Scalability Mode Settings**
   a. In the **Preferences** dialog box, type **scalability** as the filter text.
   b. Select **C/C++ > Editor > Scalability**.

      The **Scalability** page appears in the right panel of the **Preferences** dialog box.

   c. Clear the **Enable all scalability mode options** checkbox.
3. **Disable Content Assist Auto Activation**
   a. In the **Preferences** dialog box, type **content** as the filter text.
   b. Select **C/C++ > Editor > Content Assist**.

      The **Content Assist** page appears in the right panel of the **Preferences** dialog box.

   c. Clear all checkboxes in the **Auto-Activation** section.
4. Click **OK**.

The performance of the CodeWarrior IDE improves.

2.5.2 What is the first thing that I see when I start the CodeWarrior IDE?

The window that you see when you bring up the IDE is called a workbench. A workbench consists of perspective, views, and editors. The term *workbench* refers to the host development environment.

2.5.3 What is a perspective?

A perspective is an arrangement of views and editors. You can quickly switch between different perspectives.

2.5.4 How can I find the version of the CodeWarrior that I am using?

To find the version of the CodeWarrior that you are using, follow these steps:

1. From the IDE menu bar, select the **Help > About CodeWarrior Development Studio** option.

   The **About CodeWarrior Development Studio** dialog box appears.

The dialog box that appears displays the version number and the build number under **Installed Products** section.

2.5.5 Is it possible to retrieve a file that I deleted accidentally from my CodeWarrior project?

Yes, you can retrieve your deleted files. However, instead of the system's Recycle Bin you need to retrieve the file from the local history. The local history maintains a copy of files you modify or delete.

To restore a deleted file, you need to perform the following steps.

1. In the **CodeWarrior Projects** view, right-click the project branch you deleted the file from.
2. From the context menu, select the **Restore from Local History** command.

   The **Restore from Local History** dialog box appears.

   NOTE

   The dialog box contains a list of deleted files, along with a checkbox for each file in the list.

3. Check the files you want to restore from the local history.
4. Click the **Restore** button.

The selected files are restored.

2.5.6 How do I update my local history settings?

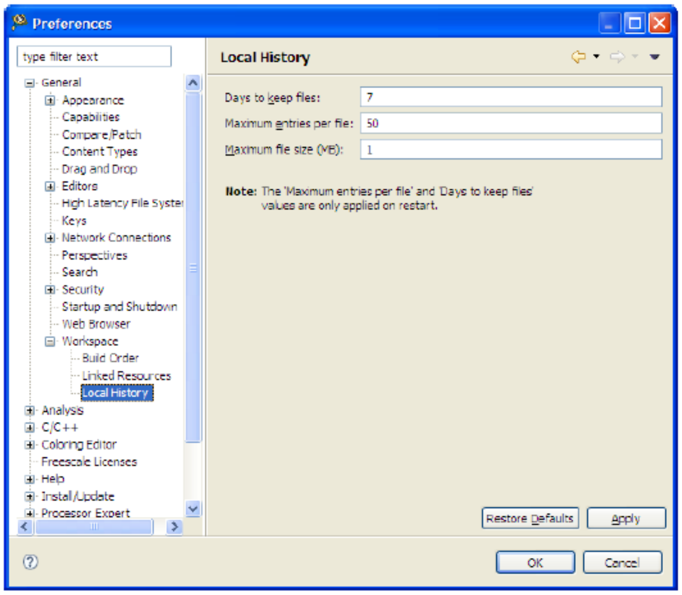
To update the local history settings or preferences, perform the following steps.

1. From the IDE menu bar, select the **Window > Preferences** option.

   The **Preferences** dialog box appears.

2. Expand the tree control to select the **General > Workspace > Local History** option.

   The **Local History** page appears.



Days to keep files: 7

Maximum entries per file: 50

Maximum file size (MB): 1

Note: The 'Maximum entries per file' and 'Days to keep files' values are only applied on restart.

Figure 2-56. Preferences Dialog Box - Local History Page

3. In the **Days to keep files** field, specify the number of days you want to maintain changes in the local history. The default value is `7`.

   NOTE

   History state older than the default value will be lost.

4. In the **Maximum entries per file** field, specify how many history states per file you want to maintain in the local history. The default value is `50`.

   NOTE

   If you exceed the default value, you will lose older history to make room for new history.

5. In the **Maximum file size (MB)** field, specify the maximum size of individual states in the history store. The default value is `1`.

   NOTE

   If you exceed the default value, the individual states will not be stored.

6. Click **Apply** to confirm and save the settings.
7. Click **OK.**

   The **Preferences** dialog box closes.

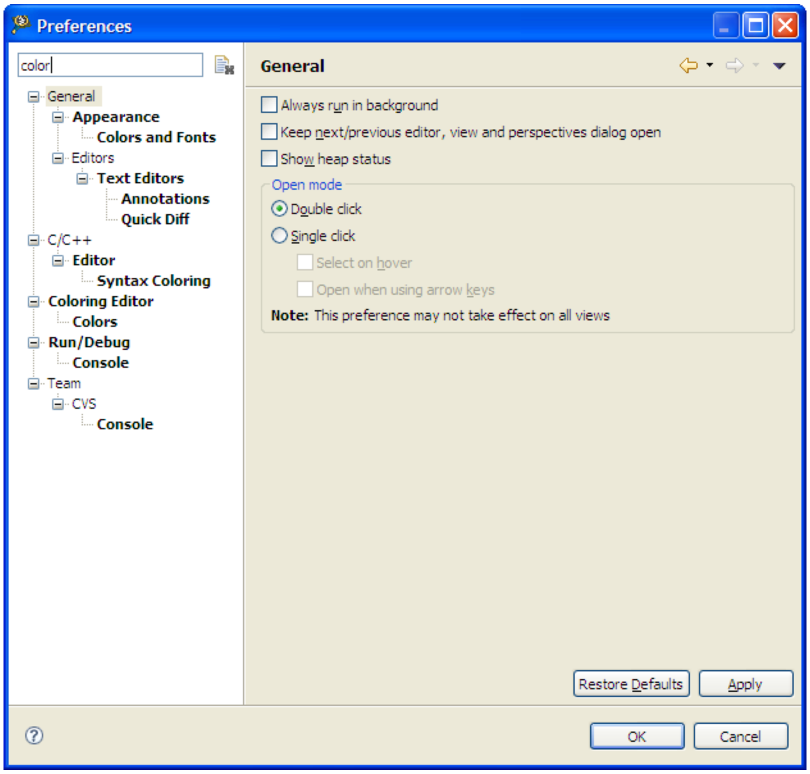
The local history settings is updated.

NOTE

To revert to the default settings, click the **Restore Defaults** button in the **Preferences** dialog box.

2.5.7 Is there a way to filter the settings in the Preferences window to find a particular setting?

Yes, you can filter settings using the filter text. For example, if you are looking for some settings to change the color in a view, type **color** as the filter text in the Preferences dialog box (refer to the image listed below). The preferences with text **color** in it are listed. This reduces the number of preferences to browse through.



□ Always run in background
□ Keep next/previous editor, view and perspectives dialog open
□ Show heap status

Open mode
● Double click
○ Single click
□ Select on hover
□ Open when using arrow keys

Note: This preference may not take effect on all views

Figure 2-57. Filtered Preferences

You can clear the filter text using either the keyboard or the 'X' icon.

You can also use wildcards in the filter text.

- * - Substitutes for zero or more characters
- ? - Substitutes for any one character

2.5.8 How can I change a key binding?

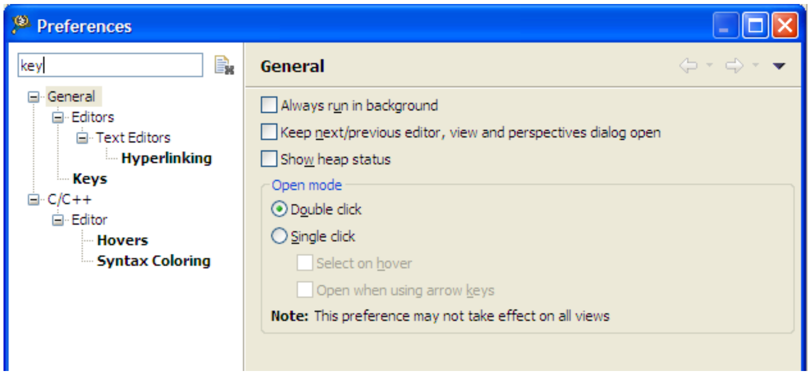
To change a keyboard shortcut or a keyboard binding:

1. From the IDE menu bar, select **Window > Preferences** .

   The **Preferences** dialog box appears.

2. Type filter text as **key** to narrow down the list of the preferences.

   The modifications need to be done in the **General > Editors > Keys** preference page.



□ Always run in background

□ Keep next/previous editor, view and perspectives dialog open

□ Show heap status

Open mode

● Double click

○ Single click

□ Select on hover

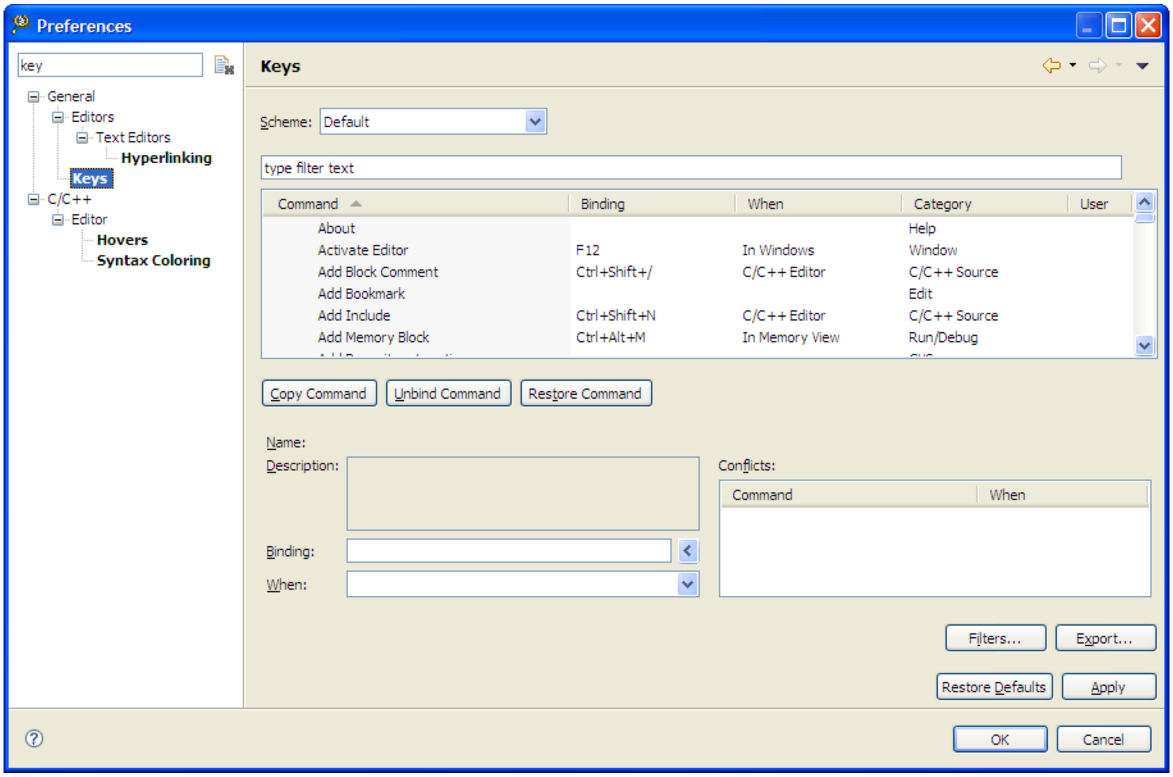
□ Open when using arrow keys

Note: This preference may not take effect on all views

Figure 2-58. Preferences Dialog Box

3. Select **General > Editors > Keys** .

   The **Keys** preferences page appears in the right panel of the **Preferences** dialog box.



Scheme: Default

type filter text

Command	Binding	When	Category	User
About			Help	
Activate Editor	F12	In Windows	Window	
Add Block Comment	Ctrl+Shift+/	C/C++ Editor	C/C++ Source	
Add Bookmark			Edit	
Add Include	Ctrl+Shift+N	C/C++ Editor	C/C++ Source	
Add Memory Block	Ctrl+Alt+M	In Memory View	Run/Debug	

Name:

Description:

Binding:

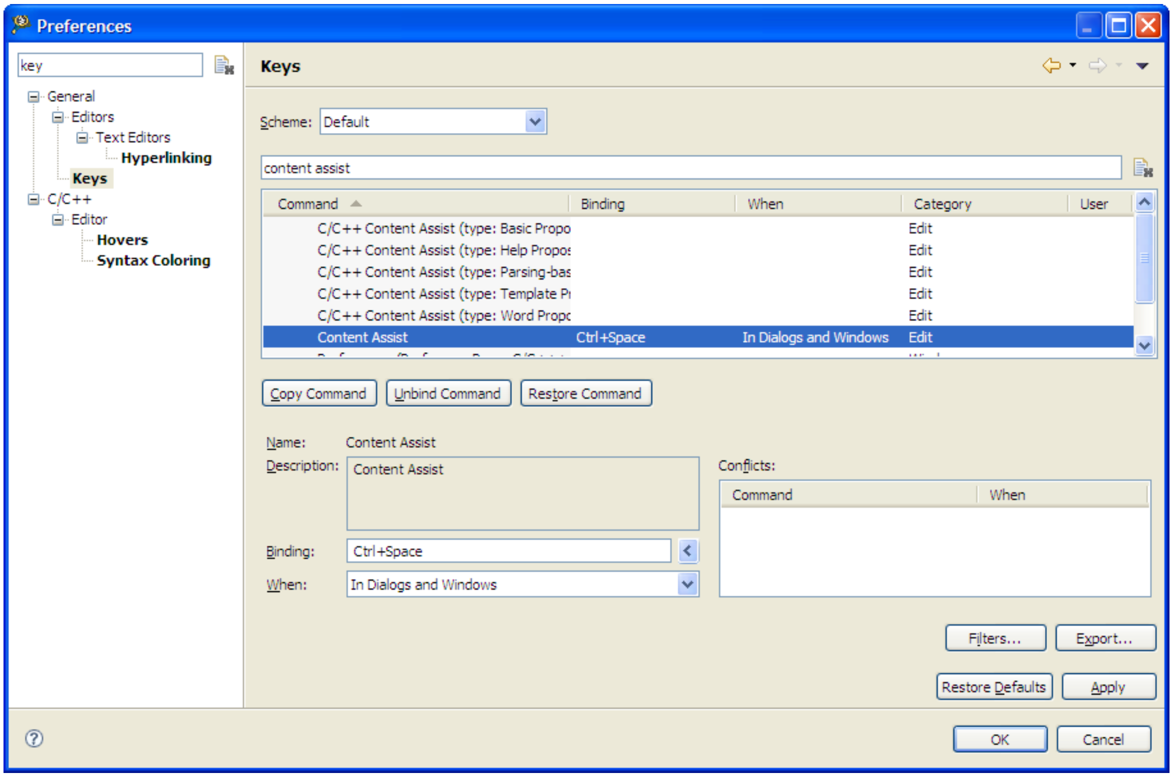
When:

Conflicts:

Command	When

Figure 2-59. Keys Preferences Page

4. From the **Scheme** drop-down list select the require scheme, such as **Microsoft Visual Studio** or **Emacs** .
5. Type **content assist** as the filter text to filter the list of keys.



Scheme: Default

content assist

Command	Binding	When	Category	User
C/C++ Content Assist (type: Basic Propo			Edit	
C/C++ Content Assist (type: Help Propos			Edit	
C/C++ Content Assist (type: Parsing-bas			Edit	
C/C++ Content Assist (type: Template Pı			Edit	
C/C++ Content Assist (type: Word Propc			Edit	
Content Assist	Ctrl+Space	In Dialogs and Windows	Edit	

Name: Content Assist

Description: Content Assist

Binding: Ctrl+Space

When: In Dialogs and Windows

Conflicts:

Command	When

Figure 2-60. Filtered List of Commands in Keys Preference Page

6. Select the entry you want to modify and type the new key binding, such as **Ctrl+Alt +C** in the **Binding** text box. You can also assign multiple key bindings.
7. From the **When** drop-down list, select the context in which the key binding apply.
8. Click **OK** .

   The **Preferences** dialog box closes.

The keyboard shortcut or the keyboard binding are modified, as specified.

2.5.9 Is it possible to view the change history of a source code file?

Yes, you can view the history of the changes done in a source file by following the steps given below:

1. Right-click the source file and select **Compare With > History f** rom the context menu.

   The **History** view opens displaying the date and time when the changes were made in the source file.

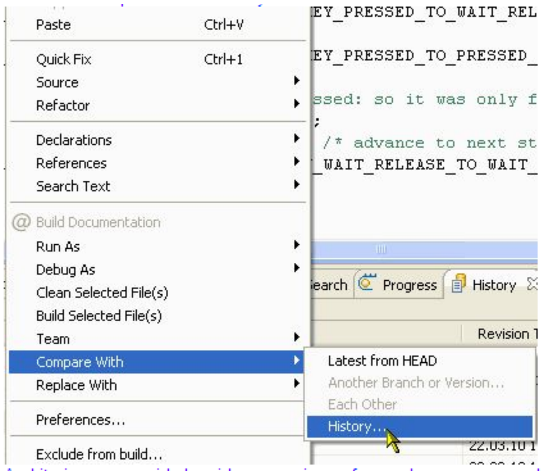
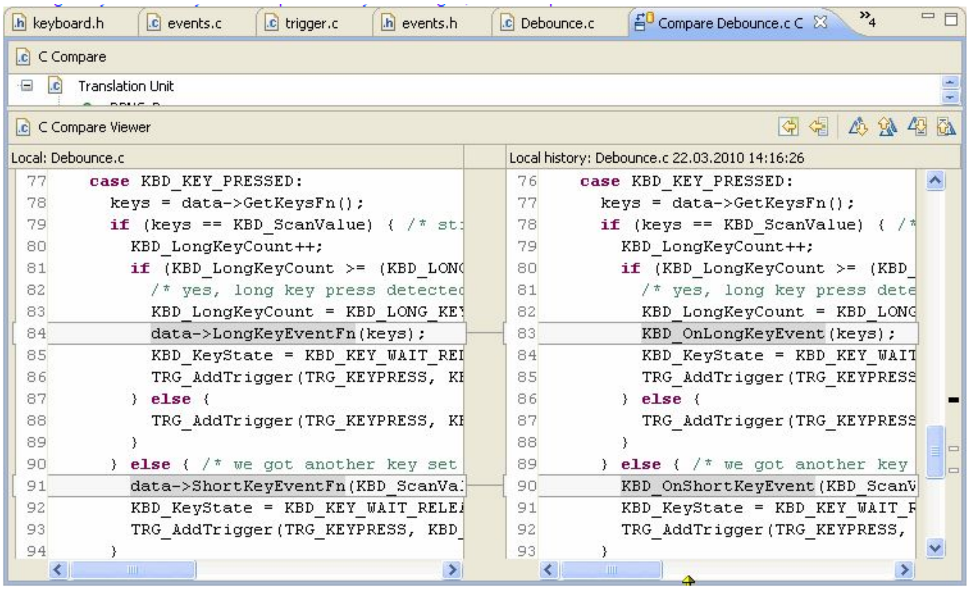


Figure 2-61. Compare With > History

2. Double-click the required revision time in the **History** view.

The editor area displays the comparison of different file revisions, and also highlights the changes. You can determine the changes done in the source file by browsing through the compared files.



```
      case KBD_KEY_PRESSED:
        keys = data->GetKeysFn();
        if (keys == KBD_ScanValue) { /* st
          KBD_LongKeyCount++;
          if (KBD_LongKeyCount >= (KBD_LON
            /* yes, long key press detecte
            KBD_LongKeyCount = KBD_LONG_KEY
            data->LongKeyEventFn(keys);
            KBD_KeyState = KBD_KEY_WAIT_REL
            TRG_AddTrigger(TRG_KEYPRESS, KE
          } else {
            TRG_AddTrigger(TRG_KEYPRESS, KE
          }
        } else { /* we got another key set
          data->ShortKeyEventFn(KBD_ScanVa
          KBD_KeyState = KBD_KEY_WAIT_RELE
          TRG_AddTrigger(TRG_KEYPRESS, KBD_
        }
```

```
      case KBD_KEY_PRESSED:
        keys = data->GetKeysFn();
        if (keys == KBD_ScanValue) { /*
          KBD_LongKeyCount++;
          if (KBD_LongKeyCount >= (KBD_
            /* yes, long key press dete
            KBD_LongKeyCount = KBD_LONG
            KBD_OnLongKeyEvent(keys);
            KBD_KeyState = KBD_KEY_WAIT
            TRG_AddTrigger(TRG_KEYPRESS
          } else {
            TRG_AddTrigger(TRG_KEYPRESS
          }
        } else { /* we got another key
          KBD_OnShortKeyEvent(KBD_ScanV
          KBD_KeyState = KBD_KEY_WAIT_R
          TRG_AddTrigger(TRG_KEYPRESS,
        }
```

Figure 2-62. Comparison of File Revisions

2.5.10 How can I determine which header files are associated with my source code file?

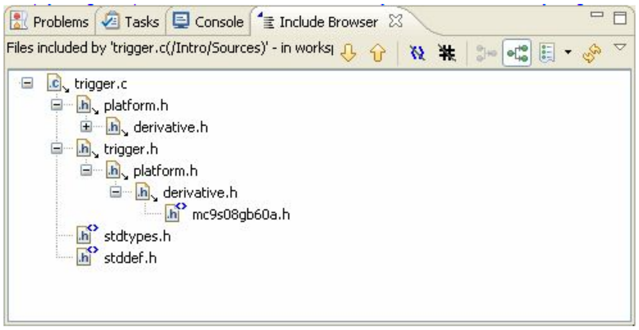
To determine which header files are associated with your source code file, you can view the graphical representation of the include files in the **Include Browser** view.

1. Select **Window > Show View > Other**.
2. Select **C/C++ > Include Browser** in the **Show View** dialog box and click **OK**.

   The **Include Browser** view appears.

3. Drag and drop the required source file from the **CodeWarrior Projects** view in the **Include Browser** view.

The **Include Browser** view displays all the include file in the selected source file (refer to the image listed below).



```
Files included by 'trigger.c(/Intro/Sources)' - in works
trigger.c
    platform.h
        derivative.h
    trigger.h
        platform.h
            derivative.h
                mc9s08gb60a.h
    stdtypes.h
    stddef.h
```

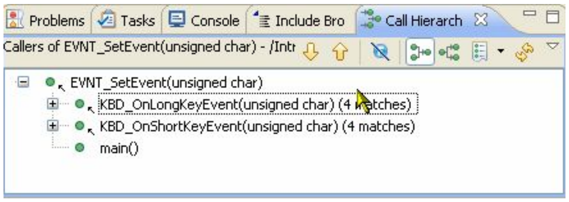
Figure 2-63. Include Browser View

2.5.11 How do I view call hierarchy in my source code?

To view or browse through the call hierarchy:

1. Right-click on a symbol in the source file and select **Open Call Hierarchy** .

   The call graph appears in the **Call Hierarchy** view.

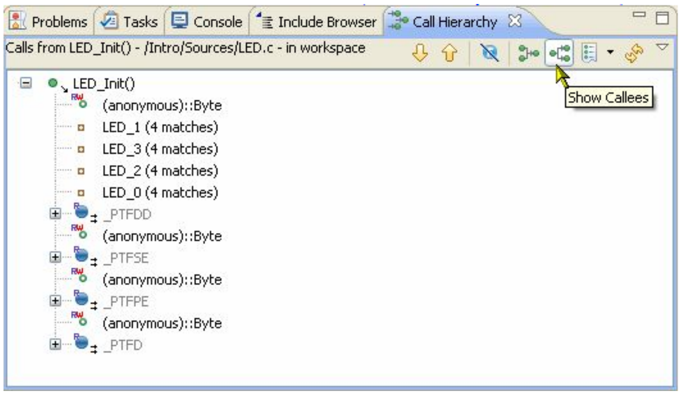


```
Callers of EVNT_SetEvent(unsigned char) - /Intr
EVNT_SetEvent(unsigned char)
    KBD_OnLongKeyEvent(unsigned char) (4 matches)
    KBD_OnShortKeyEvent(unsigned char) (4 matches)
    main()
```

Figure 2-64. Call Hierarchy View

2. Click the **Show Callees** icon in the **Call Hierarchy** view toolbar.

The callees appear in the view.



Calls from LED_Init() - /Intro/Sources/LED.c - in workspace

```
LED_Init()
    (anonymous)::Byte
    LED_1 (4 matches)
    LED_3 (4 matches)
    LED_2 (4 matches)
    LED_0 (4 matches)
    _PTFDD
    (anonymous)::Byte
    _PTFSE
    (anonymous)::Byte
    _PTFPE
    (anonymous)::Byte
    _PTFD
```

Figure 2-65. Show Callees

2.5.12 Why the project that I just created is not visible in the CodeWarrior Projects view?

A newly created project might not be visible in the CodeWarrior Projects view if you choose to use working sets. This is because a new project is not added to the active working set automatically. Thus, to see the new project in your active working set, select **Edit Active Working Set** from the view pull-down menu. From the **Working set content** options, select the newly created project and click the **Finish** button. The new project will now be visible in the active working set.

2.5.13 I created a new file in Windows Explorer but the file does not appear in the CodeWarrior Projects view. Why?

The **CodeWarrior Projects** view cannot watch for changes in your computer's file system. When you create a new file away from the Eclipse platform the CodeWarrior application is not aware of the changes. Thus to view the new file, you need to force the **CodeWarrior Projects** view to update its content. To forcefully update contents, select the project whose directory contains the updated file and from the **File** menu bar select the **Refresh** command. Alternatively, you can press the F5 key. The new file will now appear in the **CodeWarrior Projects** view.

2.5.14 Why does the Rename option appear grayed out?

The **Rename** command will be enabled only if a file or folder is selected in the **CodeWarrior Projects** view. In case you select a file or folder and click elsewhere on the Workbench, the **Rename** command appears grayed out. So, ensure that you select the desired file or folder (highlighted in blue) before trying to rename it.

2.5.15 Why duplicating a configuration in the debugger perspective does not duplicate the run configuration?

In CodeWarrior, the debug and the launch configuration are not associated with a build configuration. Instead, they are associated with an executable and a project. For example, by duplicating the build configuration, the executable remains the same. Therefore, even for the new build configuration, the old launch/debug configurations still apply. This behavior is different than the previous CodeWarrior IDE tools in that respect.

2.5.16 Can I still use Ctrl+Tab keys to navigate between open windows?

Yes, you can. When you select Ctrl+Tab, a new floating window appears that lets you navigate between the open CodeWarrior windows.

2.5.17 How can I change the debugger key bindings to the ones that I used to have in the Classic CodeWarrior IDE?

The corresponding default binding for this functionality is Ctrl+F6. You can reassign this feature to Ctrl+Tab by doing the following steps:

1. From the IDE menu bar, select **Window > Preferences**.

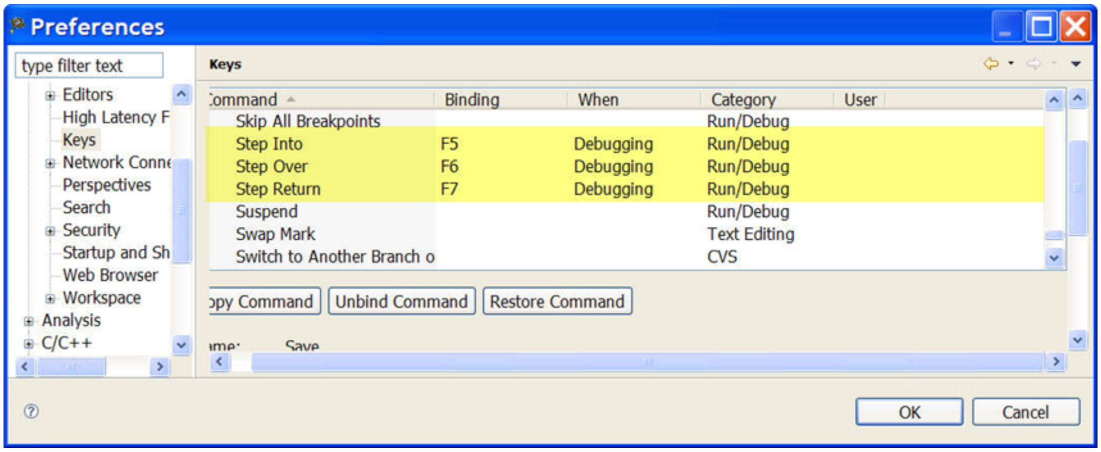
   The **Preferences** dialog box appears.

2. Select **General > Keys**.

   Under the **Command** column is a list of available commands. The **Binding** column contains the keys that can be used to invoke the command. The **When** column indicates when this command will be executed.

3. Click on the **Command** column to get the key sequence in alphabetical order.

For the debugger, the keys to change are **Step Into**, **Step Over** and **Step Return**.



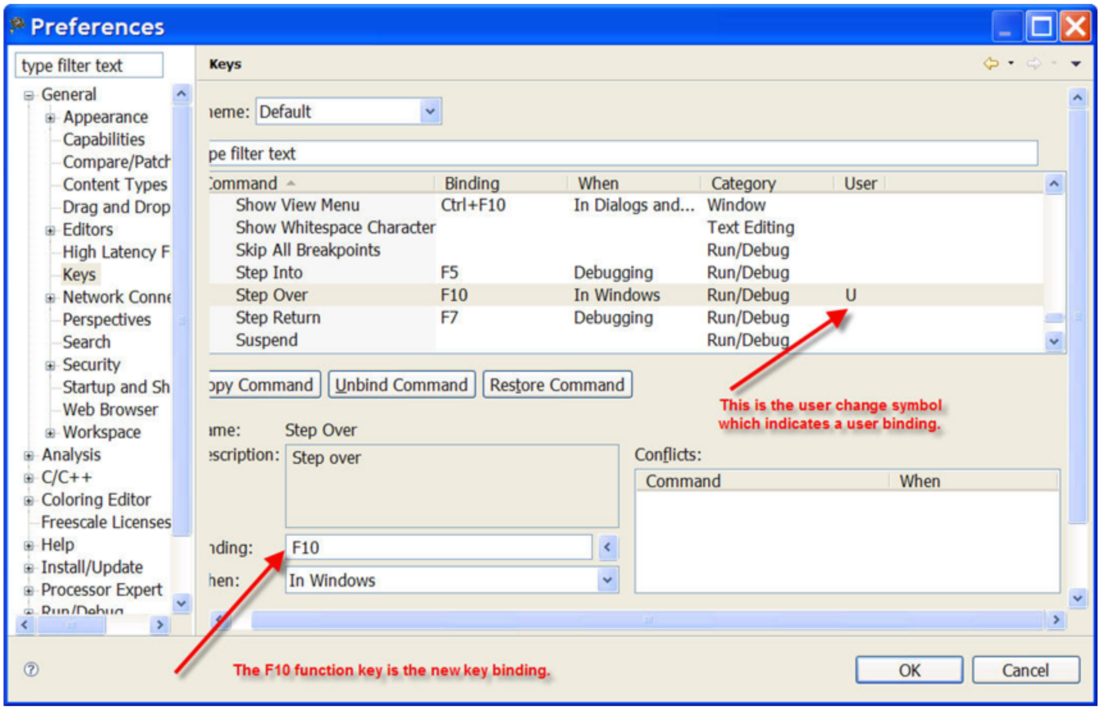
:ommand	Binding	When	Category	User
Skip All Breakpoints			Run/Debug	
Step Into	F5	Debugging	Run/Debug	
Step Over	F6	Debugging	Run/Debug	
Step Return	F7	Debugging	Run/Debug	
Suspend			Run/Debug	
Swap Mark			Text Editing	
Switch to Another Branch o			CVS	

Figure 2-66. Preferences Dialog Box-Debugger Keys

By default, these key bindings are set to F5, F6 and F7 respectively. In order to change the key sequence for Step Over from F6 to F10, follow these steps:

a. Select **Step Over** in the **Command** column.
b. From the **Binding** textbox, remove F6 by deleting it from the binding text box.
c. Press F10 on computer keyboard so that F10 will be inserted into Binding textbox.

The user change symbol, which is a blue triangle, will appear in the right-most column to indicate a user binding.



ieme: Default

:ommand	Binding	When	Category	User
Show View Menu	Ctrl+F10	In Dialogs and...	Window	
Show Whitespace Character			Text Editing	
Skip All Breakpoints			Run/Debug	
Step Into	F5	Debugging	Run/Debug	
Step Over	F10	In Windows	Run/Debug	U
Step Return	F7	Debugging	Run/Debug	
Suspend			Run/Debug	

ime: Step Over

escription: Step over

nding: F10

hen: In Windows

This is the user change symbol which indicates a user binding.

The F10 function key is the new key binding.

Figure 2-67. Preferences Dialog Box-New Key Binding

d. Click **Apply**.

The Binding for **Step Over** appear as **F10**. When you use debugger, press the **F10**key for a **Step Over**. In the context menu that appears on the Debugger window, you see **Step Over F10**.

4. Click **OK**.

   The **Preferences** dialog box closes.

The default key bindings changes, as specified.

2.5.18 How can I change the name of the executable that is generated when I build my project?

To change the name of the executable that is generated when you build your project, follow the steps given below:

1. Select the project in the **CodeWarrior Projects** view and select **Project > Properties** from the IDE menu bar.

   The **Properties** dialog box appears.

2. Select **C/C++ Build> Settings** from the left panel.

   The **Settings** page appears in the right panel of the **Properties** dialog box.

3. Select **Build Artifact** tab.

   The **Artifact name** textbox contains the name of the executable that will be created when the **Build Project** option is selected. The **Artifact extension** textbox contains the name extension of the binary that will be created. If you want to change the name of the executable, replace the one in the **Artifact name** textbox.

4. Click **Apply** .
5. Click **OK** .

Now when you build the project using the **Build Project** option, the new binary file name appears in the project folder.

2.5.19 Why the console view does not display all the warning and error messages by default?

By default, only 500 lines are displayed in the console window. In order to change the number of lines displayed in the console window, follow these steps:

1. From the IDE menu bar, select **Window > Preferences** .

   The **Preferences** dialog box appears.
2. Select **C/C++ > Build Console** from the left panel.

   The **Build Console Settings** page appears.
3. Change the **Limit console output (#lines)** to the number of lines that you would like to appear in the console window. By default, it is set to 500.
4. Click **Apply** .
5. Click **OK** .

The number of lines displayed in the console window changes, as specified.

NOTE

Since this is a global setting, the console window will display the specified number of lines for every project that is compiled within the workspace.

2.5.20 Is there a file that contains all the launch configuration settings?

Yes there is a file that describes the launch configuration settings. This file is located in the project folder with a **.launch** extension.

2.5.21 How the CodeWarrior Project Importer handles recursive access paths?

The CodeWarrior Project Importer allows the user to automatically have the recursive access paths flattened. For example, if the original project has a recursive access path `"{\Project}\"`, the imported access paths could be something like this:

```
"{Project}\"
```

```
"{Project}\src"
```

```
"{Project}\src\subdir"
```

2.5.22 Can I disassemble my source code file?

Yes you can disassemble source code files. To create a disassemble file follow steps given below:

1. Right-click the source file

   A context menu appears.

2. From the context menu, select **Disassemble** option.

   The **Disassemble Job** window appears.

The disassembling file provides a way to show the results of object code produced from a C/C++ source file in the Editor. Once the Disassemble command is executed, it will proceed to compile, disassemble the file and show the resulting disassembled file in a new editor window, titled **sourcefilename** , where represent random numbers.

2.5.23 Where exactly the disassemble file gets created?

This file gets created in the `temp` directory located under:

```
<Profiles>\<username>\<LocalSettings>\<Temp>
```

Under this path you find **sourcefilename** file.

2.5.24 Why the.metadata folder in my workspace stores a huge history?

This is a standard Eclipse feature. Eclipse keeps a local history of edited files by default. It lets you restore a file content from a previously saved local history by right-clicking a source file and selecting the **Restore from local history...** option. You can configure how the local history is stored. See How do I update my local history settings?

2.5.25 How to have projects from different workspaces or locations in a workspace?

You can have projects which are in different workspaces or locations in your workspace. For this, you can import projects from other workspaces, without copying them into your workspace.

However, keep in mind that you can only have one workspace open in one eclipse instance. If you want to view multiple workspaces side by side, then run another instance of the CodeWarrior software with a different workspace.

2.5.26 How can I switch to another workspace?

To switch to another workspace:

1. From the CodeWarrior IDE menu bar, select **File > Switch Workspace > Other** .

   The **Workspace Launcher** dialog box appears.
2. Click **Browse** .

   The **Select Workspace Directory** dialog box appears.
3. Select the workspace you want to switch to.
4. Click **OK.**
5. Click **OK.**

The selected workspace appears in the **CodeWarrior Projects** view.

2.5.27 What is the purpose of the Tasks view?

The **Tasks** view lets you view tasks that have been created.

NOTE

> The task does not refer to operating system related process or task, but with a (engineering) work task.

A task could be created to record reminders or follow-up on something later. In order to create a **Task** , select **Add Task** in the **Task** view. Tasks can also be linked to a resource that would allow you to write a note to yourself so that it can be executed or looked at a later time. In order to create a new task that is associated with a resource follow steps given below:

1. Double-click the source file.

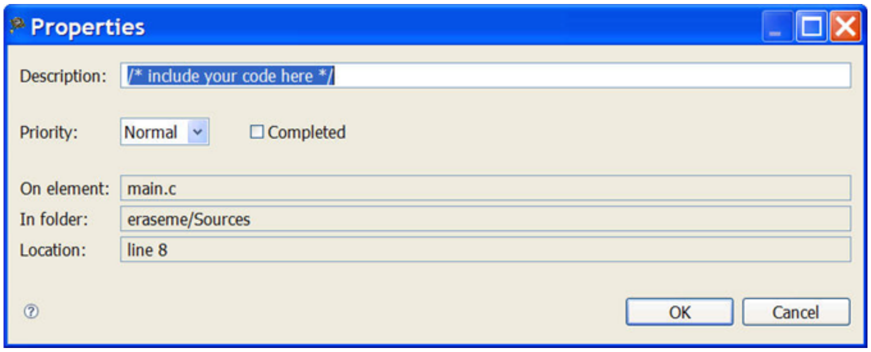
   Source file appears in the Editor view.

2. Right-click the left-hand side of source line where a task should be added.

   A context menu appears.

3. From the context menu, select **Add Task**.

   The **Properties** dialog box appears.



Properties

Description: /* include your code here */

Priority: Normal

□ Completed

On element: main.c

In folder: eraseme/Sources

Location: line 8

Figure 2-68. Properties Dialog Box

4. Type a description of what should be done with this task in the **Description** textbox.
5. Click **OK**.

The new task appears in the **Task** view.

Alternatively, you can also add a comment block with a `\todo` tag in the source code file. For example, following comment block in the source code file creates a new task in the **Tasks** view.

```
/*! \todo I need to fix this bug here... */
```

2.5.28 What is the purpose of the Properties view?

The purpose of the **Properties** view is to display property names and the basic properties of a selected resource. The kind of properties that are displayed depends on the specific resource selected as well as the features and plug-ins that are installed on the Workbench.

2.5.29 What is the purpose of the Outline view?

The Outline view displays an outline of a structured file that is currently open in the editor area, and lists the structural elements.

2.5.30 How can I find out if certain files contain debug information?

2.5.31 How can I resolve the following error message that I get when I start the CodeWarrior IDE?

```
A debug or protocol-plugin license for the product expired
or was not found
```

To resolve this error message, add the path to the V10.x license file to the LM_LICENSE_PATH environment variable. In addition, it is also possible that the launch config file is corrupted.

2.5.32 How can I open an existing project in the CodeWarrior IDE?

You can open only those projects that are already in your workspace and have been closed. To bring an external project (one that is created on another machine or in another workspace) into your workspace you can follow the steps given below:

1. From the C/C++ Perspective toolbar, select **File > Import** .

   The **Import** dialog box appears.

2. Select **General Folder > Existing Project into Workspace**
3. Click **Next** .

   The **Import Project** s dialog box appears.

4. Click **Browse** .

   The **Browse For Folder** dialog box appears.

5. Select **Project to import**
6. Click **OK** .

   The imported project appears under Projects in the Import Projects window.

7. Click **Finish** .

The selected project is imported into the workspace.

2.5.33 What is the Manage Configurations button in the C/C++ perspective toolbar used for? Could it be replaced by the Properties button?

It can not be replaced by the Properties button. The **Manage Configuration** button displays the list of defined configurations for a given project. You can select a button to add, rename, or remove a configuration. You can also set an active configuration.

2.5.34 How can I start the post-build steps in the CodeWarrior IDE?

In order to start a batch file after build, you just need to specify the name of the .bat file in the Command for post-build step.

Note that the current directory is the **Build location** specified in the project properties **C/C++ Build** panel. Therefore, you need to adjust the path to retrieve the .bat file relative to this location.

For example, consider that:

- the project is in `D:\Freescale\Work\Eclipse\testbss`
- the build location is `D:\Freescale\Work\Eclipse\testbss\Debug_SC3x50_PACC`
- the batch file is located in `D:\Freescale\Work\Eclipse\testcbe\testbss` and is called `postb.bat`.

Then, the post-build command is `..\postb.bat`.

2.5.35 Is it necessary to have the project name identical to the name of the directory that contains the .project file?

Yes, it has to be located in the folder with the same name as the project.

2.5.36 Where does the CodeWarrior IDE save the debug configuration as a local file by default?

It goes inside the .metadata subfolders and then it becomes user dependent. Perform a search on the launch and you will still find it. If you create a new launch configuration, it is saved as a local file by default.

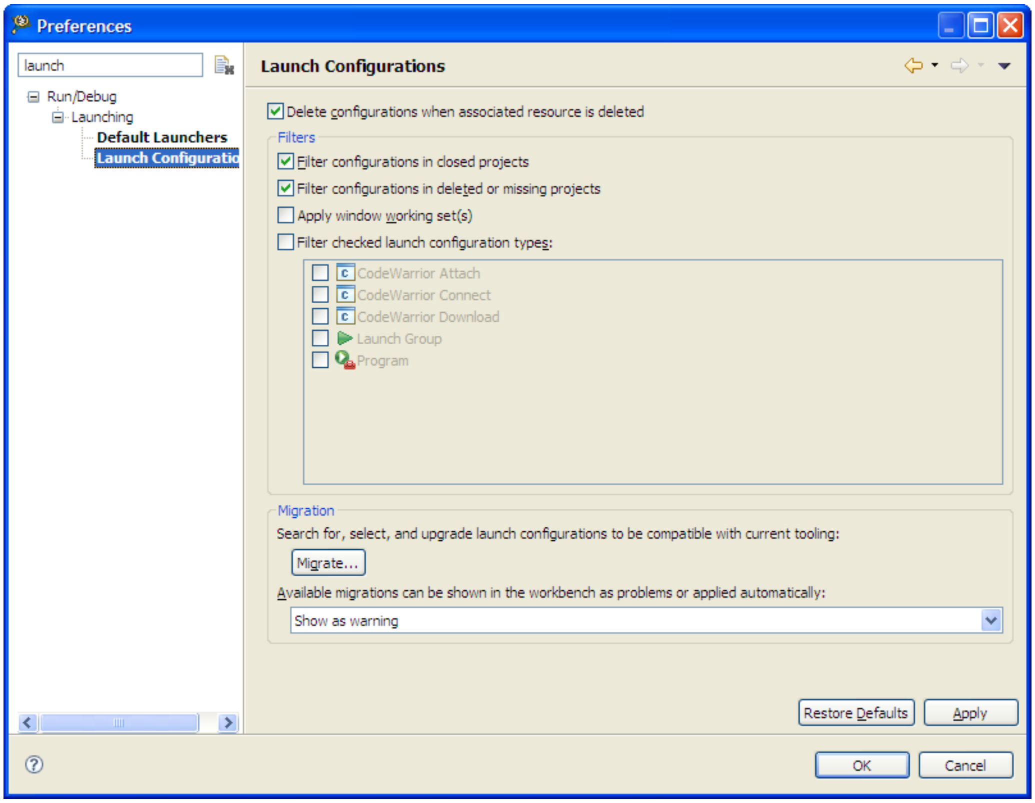
2.5.37 When I save my launch configurations as a local file and then delete the project, all the local configurations get deleted as well. How can I resolve this issue?

Follow the steps given below:

1. From the IDE menu bar, select **Window > Preferences** .
2. Select **Run/Debug> Launching > Launch Configurations** .

   The **Launch Configuration** settings appear.

3. Make sure that the **Delete configurations when associated resource is deleted** checkbox is cleared.



Launch Configurations

☑ Delete configurations when associated resource is deleted

Filters

☑ Filter configurations in closed projects
☑ Filter configurations in deleted or missing projects
☐ Apply window working set(s)
☐ Filter checked launch configuration types:

☐ CodeWarrior Attach
☐ CodeWarrior Connect
☐ CodeWarrior Download
☐ Launch Group
☐ Program

Migration

Search for, select, and upgrade launch configurations to be compatible with current tooling:

Migrate...

Available migrations can be shown in the workbench as problems or applied automatically:

Show as warning

Figure 2-69. Preferences Dialog Box - Launch Configuration

4. Click **OK** .

Now, the configurations will not be deleted when the associated resource is project.

2.5.38 How can I modify and save the files in GBK encoding using the CodeWarrior IDE?

In order to be able to save these files, follow the steps given below:

1. From C/C++ perspective toolbar, select **Window > Preferences** .

   The **Preferences** window appears.

2. Select **General > Workspace** .
3. Under T **ext file encoding** , type **GB18030** in the **Other** textbox.
4. Click **Apply** .
5. Click **OK** .

Now, you can save the files in GBK encoding.

2.5.39 How are the PARENT-COUNT-MyVariable definitions defined in the CodeWarrior IDE?

The Linked resource variable PARENT can be used to define path variables relative to a parent directory of another one. For example, a path variable FOO can be set to ${PARENT-2-PROJECT_LOC}/foo, which is equivalent of setting it to ${PROJECT_LOC}../../foo. The reason of this strange syntax is for the Eclipse compatibility constraints. You can not have the "..." characters inside a path variable value (at the resource level).

2.5.40 Is there a way to instruct the CodeWarrior IDE to use relative paths instead of absolute ones to store the project file location in a workspace?

Yes. When you create the project in the default location which is the workspace directory, no .location file gets created and you just have to zip the project and the workspace. Before you create the zip file make sure that you remove the following:

- The project build directory since there will be absolute paths in the .mk and .args files.
- The workspace .metadata\.plugins\.org.eclipse.ltk.core.refactoring directory.
- The workspace .metadata\.plugins\org.eclipse.ltk.core.resources\.history directory.
- The .PDOM files from .metadata\.plugins\org.eclipse.cdt.core directory.

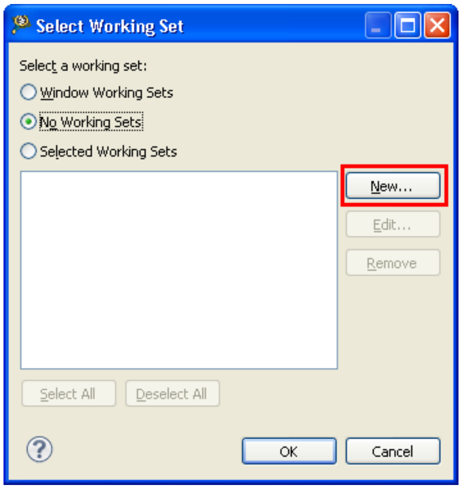
2.5.41 What is a Working Set? How do I build/compile a Working Set?

The Working Set is a CodeWarrior eclipse IDE feature, that allows you to group elements for display in views. With that, you can do operations on a set of elements in that working set. Especially when you are using many projects the same time, working sets are a big help to focus on the right set of things at a time.

To build/compile a Working Set:

1. Select **Project > Build Working Set > Select Working Set** from the IDE menu bar.

   The **Select Working Set** dialog box appears.



Select a working set:

○ Window Working Sets

● No Working Sets

○ Selected Working Sets

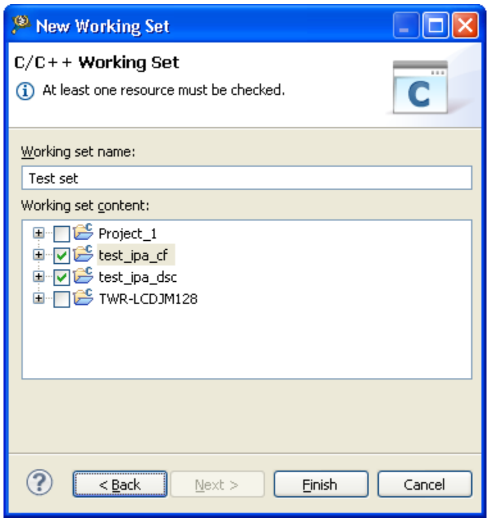
Figure 2-70. Select Working Set Dialog Box

2. If you have no Working Set defined, then click **New**.

   The **New Working Set** dialog box appears.

3. Select **C/C++** and click **Next**.

   The **C/C++ Working Set** page appears.



C/C++ Working Set

At least one resource must be checked.

Working set name: Test set

Working set content:

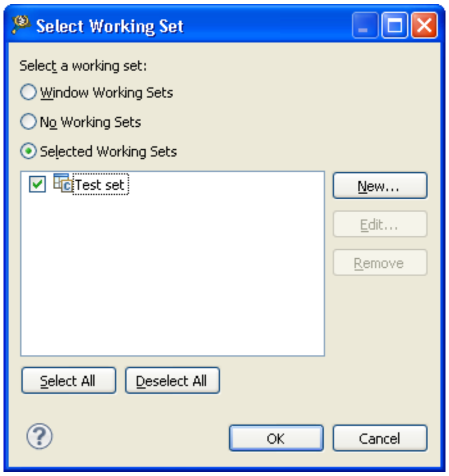
- □ Project_1
- ☑ test_ipa_cf
- ☑ test_ipa_dsc
- □ TWR-LCDJM128

Figure 2-71. New Working Set - C/C++ Working Set Page

4. Specify a name in the **Working set name** text box, and select the desired projects that you want to include from the **Working set content** list.
5. Click **Finish**.

   The **New Working Set** dialog box closes and the newly created working set appears in the **Select Working Set** dialog box.

6. Select the newly created Working Set.



Select Working Set

Select a working set:

○ Window Working Sets

○ No Working Sets

● Selected Working Sets

☑ Test set

Figure 2-72. Select Working Set Dialog Box with Newly Created Working Set

7. Click **OK**.

The IDE builds/compiles the projects included in the Working Set.

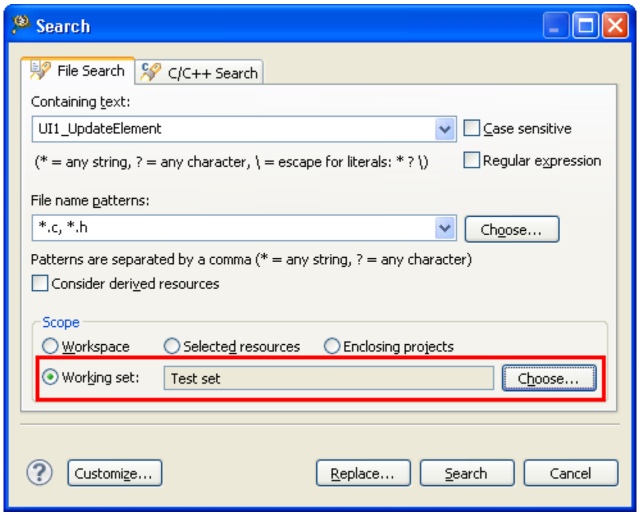
2.5.42 How do I search within the Working Set?

To search within a Working Set:

1. Select **Search > Search** from the IDE menu bar. Alternatively, press **Ctrl+H**.

   The **Search** dialog box appears.

2. Specify the desired settings in the **Containing text** and the **File name patterns** text boxes.



Containing text: UI1_UpdateElement

□ Case sensitive

(* = any string, ? = any character, \ = escape for literals: * ? \)

□ Regular expression

File name patterns: *.c, *.h

Patterns are separated by a comma (* = any string, ? = any character)

□ Consider derived resources

○ Workspace ○ Selected resources ○ Enclosing projects

● Working set: Test set

Figure 2-73. Search Dialog Box

3. Select **Working set** from the **Scope** group and click **Choose** to select the desired working set where you want the IDE to conduct the search.
4. Click **Search**.

The IDE conducts the search and lists the result in the **Search** view.

2.5.43 How do I manage the Working Sets?

To manage the Working Sets:

1. Select **Project > Build Configurations > Manage Working Sets** from the IDE menu bar.

   The **Manage Working Set Configurations** dialog box appears listing the available working sets.

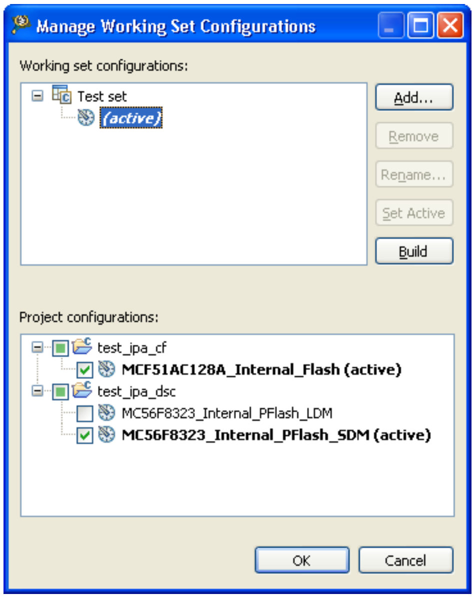

Manage Working Set Configurations
Working set configurations:
Test set
(active)
Add...
Remove
Rename...
Set Active
Build
Project configurations:
test_ipa_cf
MCF51AC128A_Internal_Flash (active)
test_ipa_dsc
MC56F8323_Internal_PFlash_LDM
MC56F8323_Internal_PFlash_SDM (active)
OK
Cancel

Figure 2-74. Manage Working Set Configurations Dialog Box

Using this dialog box, you can add, remove, rename, set active, and build the desired Working Sets.

2.5.44 How can I can filter the projects listed in the CodeWarrior Projects view based on Working Sets?

To filter the projects listed in the **CodeWarrior Projects** view based on working sets, you have to click the *inverted triangle* from the toolbar and select the **Select Working Set** or select one from the most recently used listed working sets.

The following figure shows the inverted triangle menu to filter the projects listed in the **CodeWarrior Projects** view based on working sets:

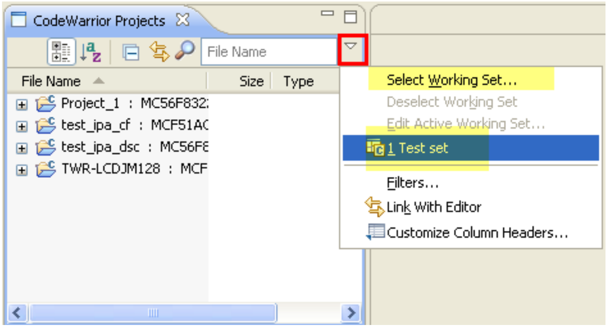


Figure 2-75. CodeWarrior Projects View - Select Working Set

2.5.45 How can I import/export a Working Set in the CodeWarrior eclipse IDE?

To import/export a Working Set in the CodeWarrior eclipse IDE, you have to install AnyEdit Tools plug-in provided by Andrey Loskutov. This plug-in allows you to:

- Import and Export of Eclipse Working Sets
- Text Conversion and Sorting actions
- Extended Compare and Replace
- Save Console View
- Open File under Cursor

To install AnyEdit Tools plug-in:

1. Select **Help > Install New Software** from the IDE menu bar.

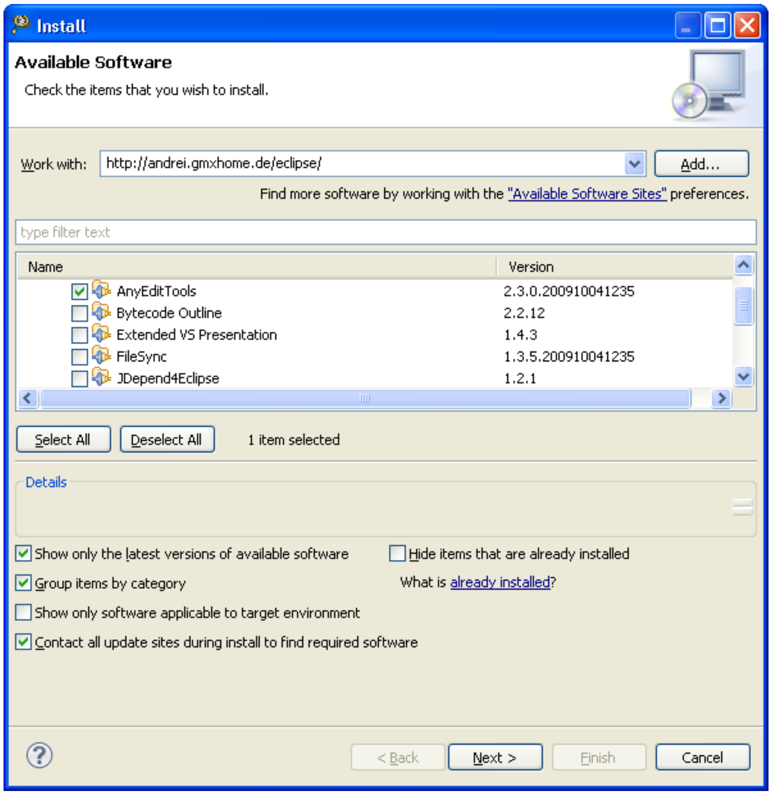
   The **Install** dialog box appears.

2. Enter the following web address in the **Work with** textbox and press *Enter*:

   ```
   http://andrei.gmxhome.de/eclipse/
   ```

   The available plug-in for various eclipse versions appears in the list.

3. Expand the tree-control of the desired eclipse version and select **AnyEditTools**.



Available Software

Check the items that you wish to install.

Work with: http://andrei.gmxhome.de/eclipse/

Find more software by working with the "Available Software Sites" preferences.

Name	Version
☑ AnyEditTools	2.3.0.200910041235
□ Bytecode Outline	2.2.12
□ Extended VS Presentation	1.4.3
□ FileSync	1.3.5.200910041235
□ JDepend4Eclipse	1.2.1

1 item selected

☑ Show only the latest versions of available software

□ Hide items that are already installed

☑ Group items by category

What is already installed?

□ Show only software applicable to target environment

☑ Contact all update sites during install to find required software

Figure 2-76. Install Dialog Box - AnyEditTools Plug-in

4. Click **Next**.

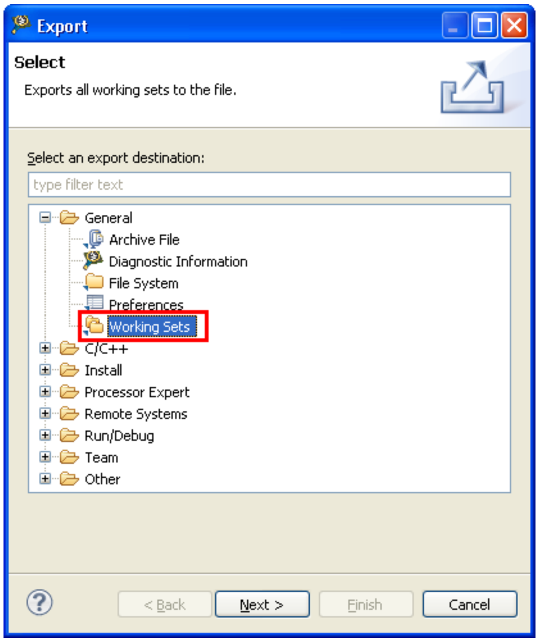
   The **Install details** page appears.

5. Select **AnyEditTools** from the list.
6. Click **Next**.
7. Select the option to accept the license agreement.
8. Click **Finish**.

The IDE installs the AnyEdit Tools for your CodeWarrior Software.

The AnyEdit Tools includes an importer and exporter File action for Working Sets. To export and import the Working Sets:

- Select **File > Export** from the IDE menu bar and follow the onscreen instructions to export a working set. The following figure shows the **Export** dialog box with **Working Sets** option:



Export

Select

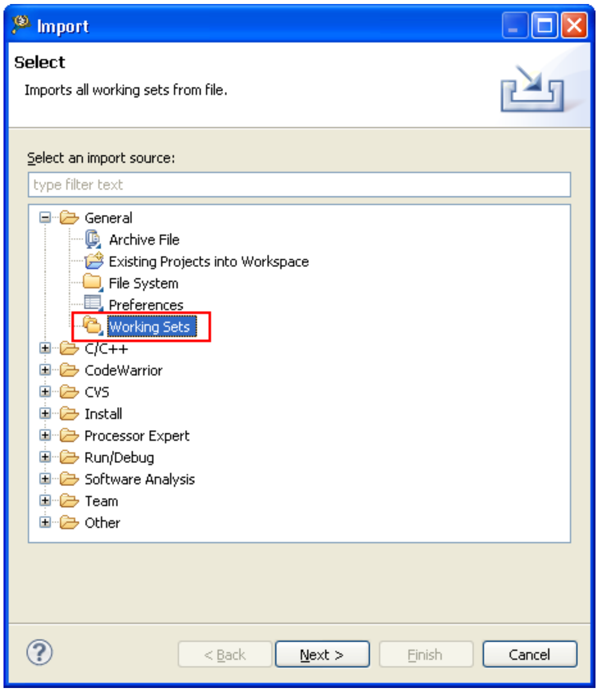
Exports all working sets to the file.

Select an export destination:

type filter text

Figure 2-77. Export Dialog Box with Working Sets Option

- Select **File > Import** from the IDE menu bar and follow the onscreen instructions to import a working set. The following figure shows the **Import** dialog box with **Working Sets** option:



Import

Select

Imports all working sets from file.

Select an import source:

type filter text

Figure 2-78. Import Dialog Box with Working Sets Option

2.5.46 What are Icon decorators?

The icon and label decorators are the symbols used for annotating the icon or label with the useful information. The following figures shows the example of source file icon decorators:

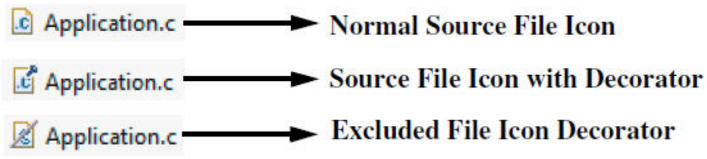


Figure 2-79. Example - Decorator for Application.c File

2.5.47 How do I enable or disable an icon decorator?

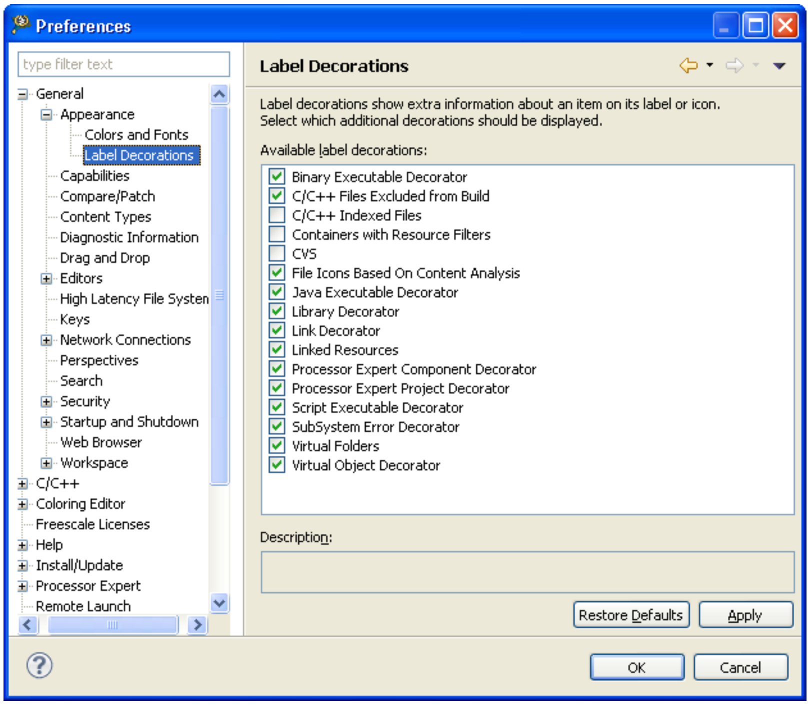
To enable/disable an icon decorator:

1. Select **Window > Preferences** from the IDE menu bar.

   The **Preferences** dialog box appears.

2. Expand the **General** tree control and select **Label Decorations** from the left panel.

   The **Label Decorations** page appears in the right panel. The following figure shows the **Preferences** dialog box with **Label Decorations** page:



Label decorations show extra information about an item on its label or icon.
Select which additional decorations should be displayed.

Available label decorations:

- ☑ Binary Executable Decorator
- ☑ C/C++ Files Excluded from Build
- ☐ C/C++ Indexed Files
- ☐ Containers with Resource Filters
- ☐ CVS
- ☑ File Icons Based On Content Analysis
- ☑ Java Executable Decorator
- ☑ Library Decorator
- ☑ Link Decorator
- ☑ Linked Resources
- ☑ Processor Expert Component Decorator
- ☑ Processor Expert Project Decorator
- ☑ Script Executable Decorator
- ☑ SubSystem Error Decorator
- ☑ Virtual Folders
- ☑ Virtual Object Decorator

Figure 2-80. Preferences Dialog Box - Label Decorations Page

3. Check or clear the checkboxes for the desired label decorators to enable or disable the decorators respectively.
4. Click **Apply** to save the changes.
5. Click **OK** to close the dialog box.

The IDE applies and saves the changes you selected for the icon decorations.

2.5.48 Can the icon decorators be combined?

Yes, the icon decorators can be combined. The following figure shows an example:

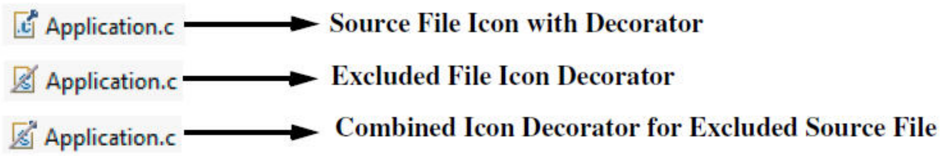

Application.c
Source File Icon with Decorator
Application.c
Excluded File Icon Decorator
Application.c
Combined Icon Decorator for Excluded Source File

Figure 2-81. Example - Combined Icon Decorators

2.5.49 When I open a large file in CodeWarrior IDE, an Editor Scalability dialog box appears, listing the message, "You are opening a large file. Scalability mode has been turned

on for this editor to help improve performance by disabling features requiring to parse the source file language." What does it mean?

While processing a file in the editor, there are various other processes that runs in the background, this includes *Eclipse Indexer*, *Outline View* or *Syntax Coloring*.

When you open a large file, the IDE tends to get slower to use more memory to process that file. To avoid this, the IDE turns on the *Scalability Mode*, and close the functions like, *Eclipse Indexer*, *Outline View* or *Syntax Coloring* to increase the performance.

2.5.50 How do I configure the Scalability settings?

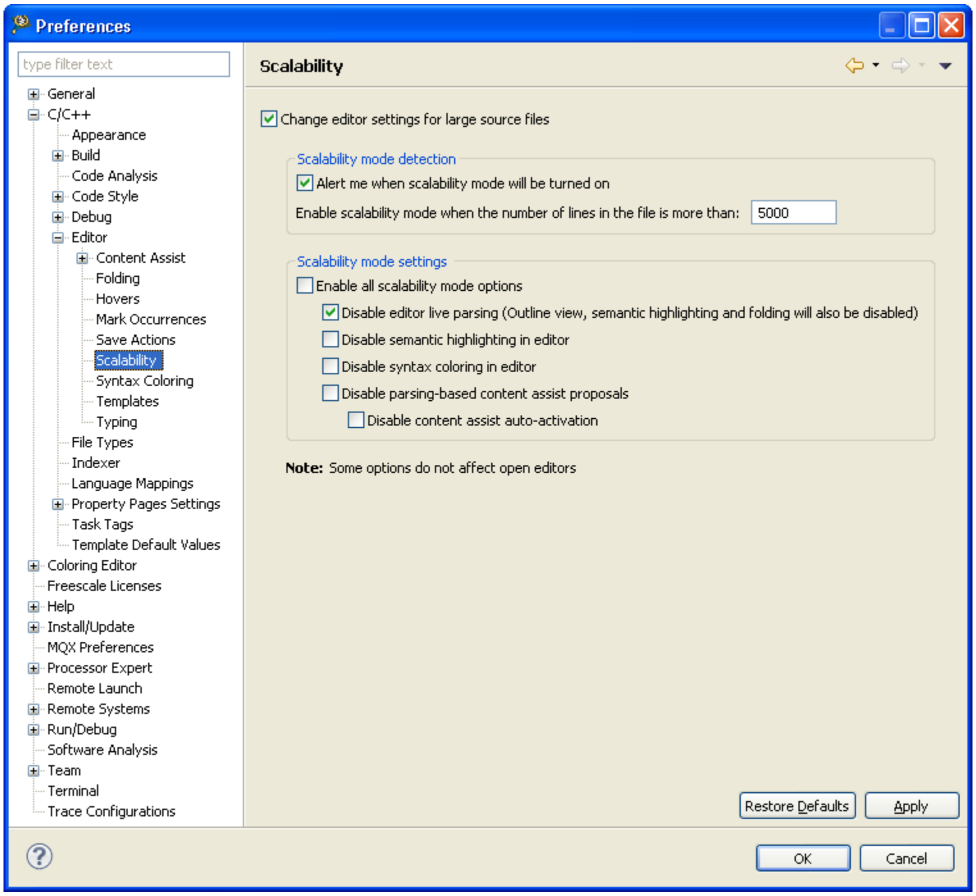
You can configure the Scalability Mode to be enabled only for very large files. To configure the **Scalability** settings:

1. Select **Window > Preferences** from the IDE menu bar.

   The **Preferences** dialog box appears.

2. From the left panel, expand the **C/C++ > Editor** tree controls and select **Scalability** .

   The **Scalability** page appears in the right panel.



Scalability

☑ Change editor settings for large source files

Scalability mode detection

☑ Alert me when scalability mode will be turned on

Enable scalability mode when the number of lines in the file is more than: 5000

Scalability mode settings

□ Enable all scalability mode options

☑ Disable editor live parsing (Outline view, semantic highlighting and folding will also be disabled)

□ Disable semantic highlighting in editor

□ Disable syntax coloring in editor

□ Disable parsing-based content assist proposals

□ Disable content assist auto-activation

Note: Some options do not affect open editors

Figure 2-82. Preferences Dialog Box - Scalability Page

3. Specify the desired settings in the **Scalability mode detection** and the **Scalability mode settings** groups.
4. Click **Apply** .
5. Click **OK** .

The **Preferences** dialog box closes. The IDE saves and applies the changes you specified.

Chapter 3
Project Management

In this chapter, you find Frequently Asked Questions (FAQs) related to project management in CodeWarrior IDE.

- FAQs - Project Management

3.1 FAQs - Project Management

This topic lists the project management related FAQs.

In this topic:

- How can I find out where the source files related to a CodeWarrior project are saved?
- How can I add another source or header file to my project?
- What is a workspace?
- What is a project?
- How can I view the contents of various types of source code files that appear in my project folder?
- Is there a way to compare the Property Settings of two different projects?
- How do I resize the various views in a Perspective?
- How can I add a project from some other workspace to my current workspace?
- How do I ensure that the CodeWarrior IDE saves all modifications that I make to the project properties?

3.1.1 How can I find out where the source files related to a CodeWarrior project are saved?

To know where your source file is located, follow these steps:

1. In the C/C++ project window, right-click on source file.

   A context menu appears.
2. In the context menu, select **Show in Windows Explorer** .

   The Window Explorer window appears.

Alternatively, follow these steps:

1. In the C/C++ project window, right-click on source file.

   A context menu appears.
2. In the context menu, select **Properties** .

   The **Properties** window appears.
3. Select **Resource** .

The **Location** field displays the full path of the folder.

NOTE

CodeWarrior always warns the user before performing a delete operation and describes whether it is about to delete just the link, or to delete the file or folder permanently.

3.1.2 How can I add another source or header file to my project?

To add a source file or header to project follow steps below:

1. In the CodeWarrior Project window, right-click on the folder where you want the new source file or header to be added.

   A context menu appears.
2. From the context menu, select **New > Source File** .

   The **New Source File** window appears.
3. In the **Source File** textbox, enter the name of the source file.
4. Click **Finish** .

The source file gets added to your project.

NOTE

Another way to accomplish is to drag and drop the file from Windows Explorer. A copy of the file is added to the project's workspace. If you make changes to the source file, the original source file is not modified. Only source file in the project is modified.

3.1.3 What is a workspace?

A workspace is a directory for your work. This is where your project is located. The name and location of the workspace is specified when you start the CodeWarrior IDE.

3.1.4 What is a project?

A project is a container for organizing files and folders related to a specific area.

3.1.5 How can I view the contents of various types of source code files that appear in my project folder?

To view the contents of source files, follow steps below:

1. From the CodeWarrior Projects window, right-click on the source file.

   A context menu appears.
2. From the context menu, select **Open With > Text Editor** .

The text editor window appears showing the contents of the file.

Tip

Double-click on the source file to view its contents in the **C/C++ Editor** .

3.1.6 Is there a way to compare the Property Settings of two different projects?

Currently the way to do this is to do a text compare of the .project files. To do this, follow steps below:

1. In the CodeWarrior Projects folder, click on .cproject.

   The .cproject is highlighted.

2. Press the **Ctrl** key.
3. In the CodeWarrior Projects folder, click on second .cproject.

   The second .cproject is highlighted.

4. Right-click on .cproject.

   A context menu appears.

5. From context menu, select **Compare With > Each Other** .

   The **Compare** window appears.

You can compare property settings of the projects using the **Compare** window.

3.1.7 How do I resize the various views in a Perspective?

To resize the various view in a Perspective, click on the dividers (gray bands) between the different views and drag them with the mouse.

3.1.8 How can I add a project from some other workspace to my current workspace?

In order to add a project from another workspace follow steps below:

1. Open the workspace in Windows Explorer.
2. Select the project folder and drag it over to the CodeWarrior IDE.

The IDE effectively handles the files and folders dropped to the workbench. A link to the existing project is created in the **CodeWarrior Project** view.

3.1.9 How do I ensure that the CodeWarrior IDE saves all modifications that I make to the project properties?

The settings specified in the Properties dialog box are saved when you click **Apply** and then **OK** to close the dialog box. The only reason this does not work could be:

- You only have read access to the `.project` file in the project directory.

-or-

- You closed the **Properties** dialog box by clicking **Cancel** instead of **OK** .

Chapter 4
Project

In this chapter you find Frequently Asked Questions (FAQs) related to projects in the CodeWarrior IDE.

- FAQs - Project

4.1 FAQs - Project

This topic lists the project related FAQs.

- How do I add a new Source Folder to my project?
- How can I add to my project a folder that exists outside the current workspace?
- How do I add to my project a new file that exists outside the current workspace?
- How can I export a project that I could send to someone else?
- How do I instruct the CodeWarrior IDE to save changes to my source code files automatically before building the project?
- How do I compare two source code files?
- How do I compare two directories?
- How can I specify the number of days to keep local history for source code files?
- How can I see or change the preprocessor macros defined for my project?
- Is there a shortcut to remove object code that is equivalent to the Ctrl + - shortcut key available in the Classic CodeWarrior IDE?
- NEW While launching a project, the CodeWarrior software automatically builds the current active target. How can I specify the build configuration I want to debug?
- NEW While building a project, the CodeWarrior fails to write an ELF file. How do I resolve this issue?
- NEW How do I enable the incremental build on my project?
- NEW How do I filter the items listed in the Problems view based on the project selected?

4.1.1 How do I add a new Source Folder to my project?

To add a new source folder to a project follow steps below:

1. In the CodeWarrior Project window, right-click on project name.

   A context menu appears.

2. From context menu, select **New > SourceFolder** .

   The **New Source Folder** window appears.

3. In the folder name textbox, enter the name for new folder.
4. Click **Finish** .

The new source folder appears in window.

4.1.2 How can I add to my project a folder that exists outside the current workspace?

To add files to an existing project, follow steps below:

1. In the CodeWarrior Project window, right-click on project name.

   A context menu appears.

2. From context menu, select **New > Folder** .

   The **New Folder** window appears.

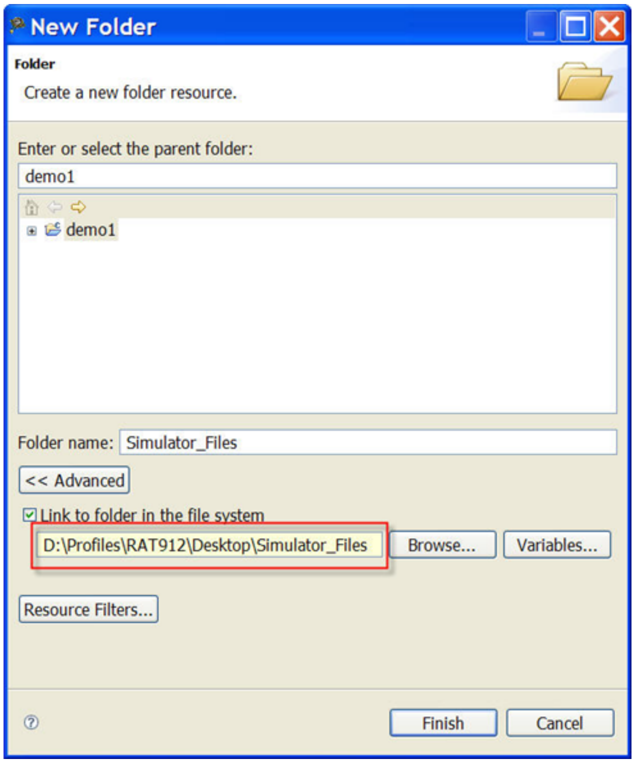
3. Click **Advanced** .

   The Advanced Settings appear.

4. Select **Link to folder in the file system** checkbox.
5. Click **Browse** .

   The **Browse for Folder** window appears.

6. Select the folder that you want to add to the project.
7. Click **OK** .



New Folder

Folder

Create a new folder resource.

Enter or select the parent folder: demo1

Folder name: Simulator_Files

☑ Link to folder in the file system

D:\Profiles\RAT912\Desktop\Simulator_Files

Figure 4-1. New Folder

8. Click **Finish** .

This will put the new folder into the project under CodeWarrior projects. When the new folder is expanded, it will point to the location where the folder is linked from. If you delete the whole folder from the project, it will be deleted from the project only. If you try to delete a file from this folder, it will be deleted from that location leaving no way to recover it.

4.1.3 How do I add to my project a new file that exists outside the current workspace?

To add a specific file to an existing project, drag the file that you want to add from the Windows Explorer into the CodeWarrior Projects.

4.1.4 How can I export a project that I could send to someone else?

To export a project follow steps below:

1. In CodeWarrior Project window, right-click on project name.

   A context menu appears.

2. From context menu, select **Export** .

   The **Export** window appears.

3. Expand **General Folder** .
4. Select **Archive File** .
5. Click **Next** .

   The **Archive File** window appears.

   NOTE

   By default, the project that you select will be exported along with their children. Optionally, use the checkboxes in the left and right panes to select the set of resources to export. Also, use radio buttons, such as **Select Types** to filter the types of files that you want to export.

6. In the **To Archive File** textbox, enter the archive file which you want to export the selected resources to.
7. Click **Finish** .

A zip file is created in the **Eclipse** folder within the **Tools Installation** directory.

4.1.5 How do I instruct the CodeWarrior IDE to save changes to my source code files automatically before building the project?

To instruct the CodeWarrior IDE to save changes to your source code files automatically before building the project, follow steps below:

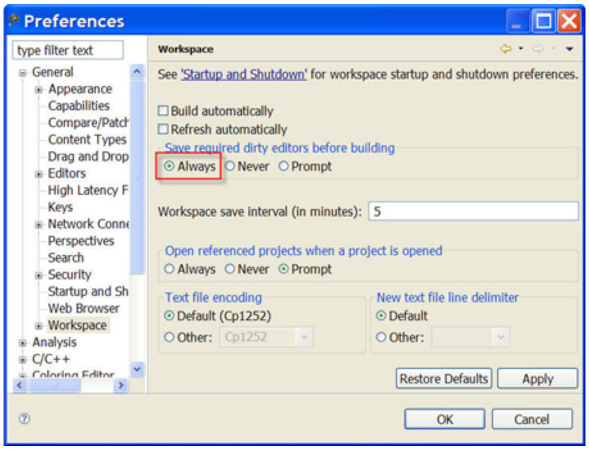
1. From the C/C++ perspective toolbar, select **Window > Preferences** .

   The **Preferences** window appears.

2. Expand **General** .
3. Select **Workspace** .

   The **Workspace** settings appear.

4. Under **Save required dirty editors before building** , select the **Always** radio button.



See 'Startup and Shutdown' for workspace startup and shutdown preferences.

□ Build automatically

□ Refresh automatically

Save required dirty editors before building

● Always ○ Never ○ Prompt

Workspace save interval (in minutes): 5

Open referenced projects when a project is opened

○ Always ○ Never ● Prompt

Text file encoding

● Default (Cp1252)

○ Other: Cp1252

New text file line delimiter

● Default

○ Other:

Figure 4-2. Preferences-Workspace Page

5. Click **Apply** .
6. Click **OK** .

Now, the CodeWarrior IDE will save changes to the source code files automatically before building the project.

4.1.6 How do I compare two source code files?

In order to compare two source files, follow steps below:

1. In the CodeWarrior Projects folder, click on source file.

   The source file is highlighted.

2. Press the **Ctrl** key.
3. In the CodeWarrior Projects folder, click on second source file.

   The source file is highlighted.

4. Right-click on source file.

   A context menu appears.

5. From context menu, select **Compare With > Each Other** .

   The **Compare** window appears.

You can compare the source files using the **Compare** window.

4.1.7 How do I compare two directories?

In order to compare two directories follow steps below:

1. In the CodeWarrior Projects folder, click on directory.

   The directory folder is highlighted.

2. Press the **Ctrl** key.
3. In the CodeWarrior Projects folder, click on second directory folder.

   The directory folder is highlighted.

4. Right-click on directory folder.

   A context menu appears.

5. From context menu, select **Compare With > Each Other** .

   The **Compare** window appears.

You can compare the directories files using the **Compare** window.

4.1.8 How can I specify the number of days to keep local history for source code files?

To change the history length for the C source editor, follow steps below:

1. From the C/C++ perspective toolbar, select **Window > Preferences** .

   The **Preferences** window appears.

2. Select **General > Workspace > Local History** .
3. The **Local History** settings appear.
4. In the **Days to Keep Files** textbox, change the days to keep files value.
5. Click **Apply** .
6. Click **OK** .

The history length for the C source editor is updated.

4.1.9 How can I see or change the preprocessor macros defined for my project?

To see or change the preprocessor macros defined for your project, follow steps below:

1. In the CodeWarrior Project window, right-click on project name.

   A context menu appears.

2. From the context menu, select **Properties** .

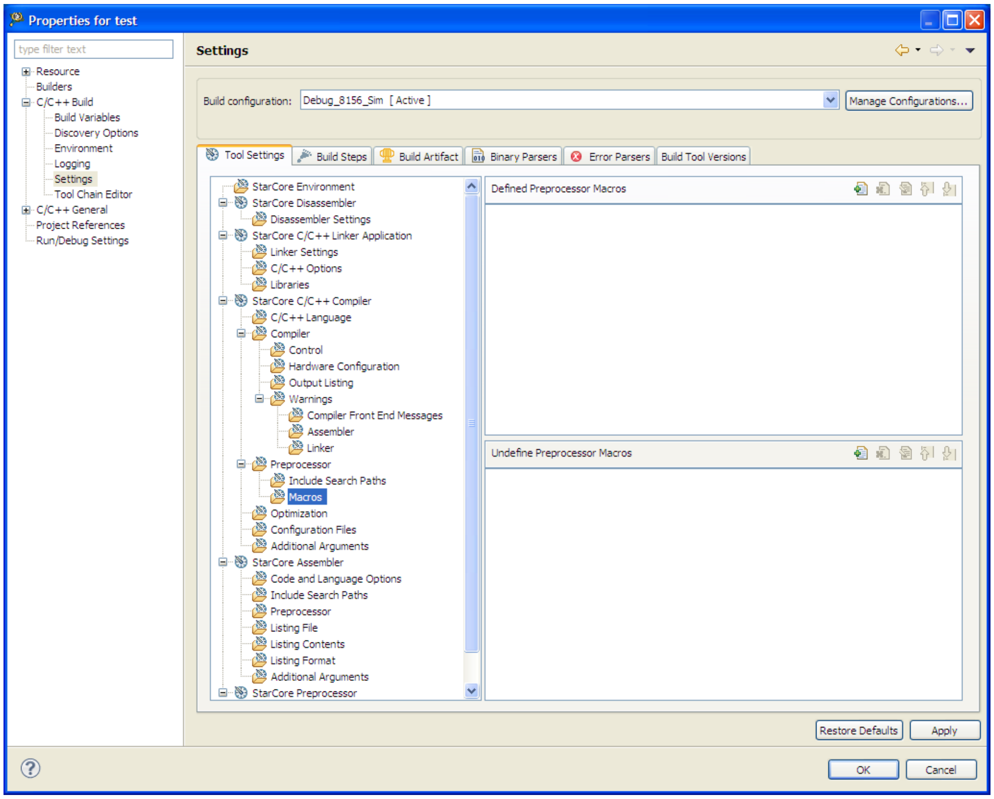
   The **Properties** window appears.

3. Select **C/C++ Build > Settings** .

   The **Settings** panel appears on the right-hand side.

4. Select **StarCore C/C++ Compiler > Preprocessor > Macros**.

   The **Macros** panel appears on the right-hand side displaying `#define` preprocessor macros.



Build configuration: Debug_8156_Sim [Active]

Figure 4-3. Properties Dialog Box-Settings Page

5. Make the required changes and click **Apply**.
6. Click **OK**.

The preprocessor macros defined for your project changes as specified.

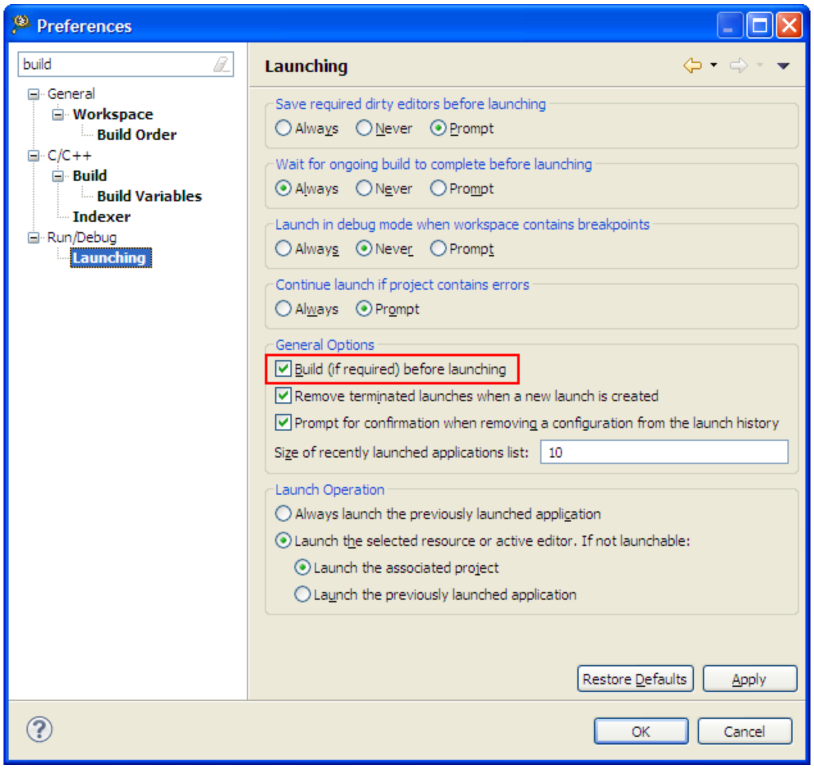
4.1.10 Is there a shortcut to remove object code that is equivalent to the Ctrl + - shortcut key available in the Classic CodeWarrior IDE?

No. However, if you want to assign the **Ctrl+-** shortcut key for removing the object code, then you can manually assign it by changing the key binding. For more information on how to assign or change a key binding, see How can I change a key binding?

When you change the key binding, a dialog box appears asking you which project you want to clean, and if you want to automatically start the build thereafter. If you do not want this dialog box to appear, then you can select the **Build Automatically** checkbox under **Window > Preferences > General > Workspace Settings** .

4.1.11 While launching a project, the CodeWarrior software automatically builds the current active target. How can I specify the build configuration I want to debug?

The CodeWarrior software has the following default settings.



Launching

Save required dirty editors before launching
○ Always ○ Never ● Prompt

Wait for ongoing build to complete before launching
● Always ○ Never ○ Prompt

Launch in debug mode when workspace contains breakpoints
○ Always ● Never ○ Prompt

Continue launch if project contains errors
○ Always ● Prompt

General Options
☑ Build (if required) before launching
☑ Remove terminated launches when a new launch is created
☑ Prompt for confirmation when removing a configuration from the launch history
Size of recently launched applications list: 10

Launch Operation
○ Always launch the previously launched application
● Launch the selected resource or active editor. If not launchable:
● Launch the associated project
○ Launch the previously launched application

Figure 4-4. Build Before Launching

And, the launch configuration refers the project and not the build target of the project.

This is why the CodeWarrior software automatically build the current active build target.

To ensure that the CodeWarrior software builds the right target in CodeWarrior for StarCore:

- Make sure you have the right build set as active, or
- Or, clear the **Build (if required) before launching** checkbox, and build the project yourself before launching.

4.1.12 While building a project, the CodeWarrior fails to write an ELF file. How do I resolve this issue?

The `DE.EXE` might locks the file. The `DE.EXE` (Debugger Engine Executable) locks the file during debugging, but might have failed to release it. To release the lock, simply close (kill) the `DE.EXE` application from the task manager.

4.1.13 How do I enable the incremental build on my project?

Follow the steps given below to perform incremental build on your project:

1. In the CodeWarrior Project window, right-click on project name.

   A context menu appears.

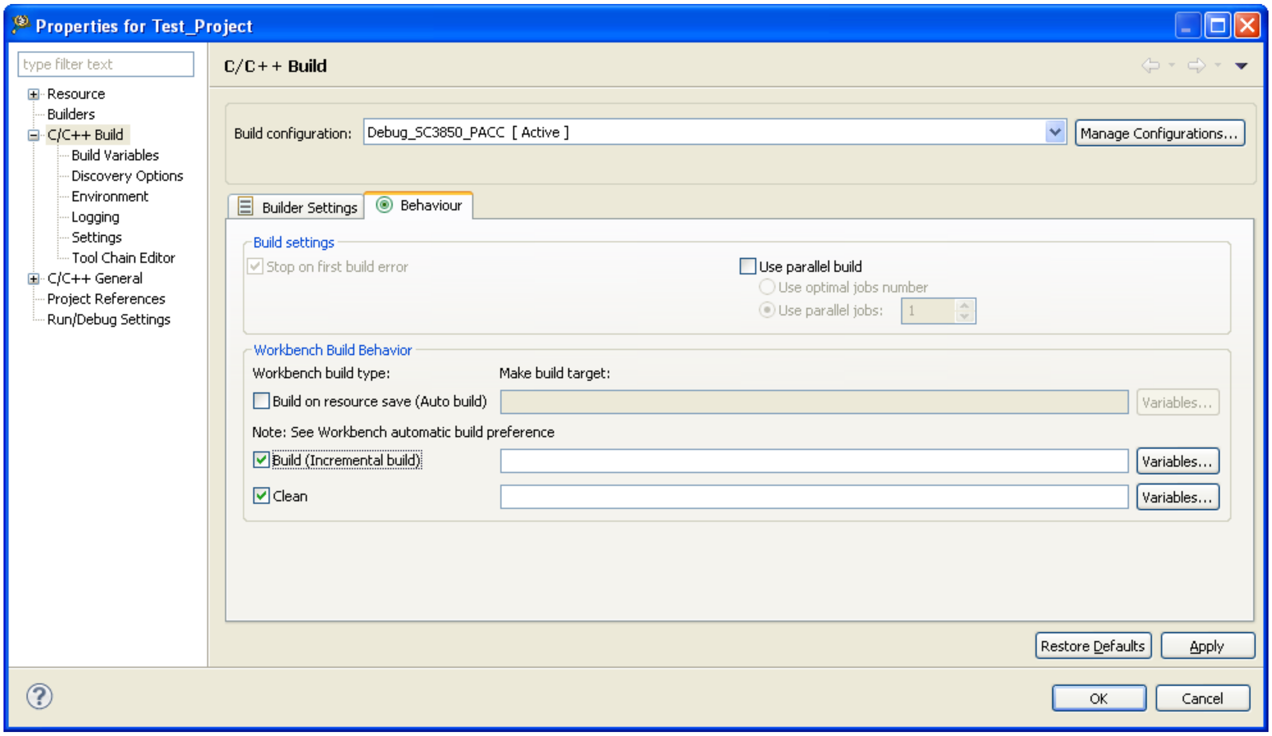
2. From the context menu, select Properties.

   The Properties window appears.

3. Select C/C++ Build.

   The C/C++ Build page appears in the right panel of the Properties dialog box.

4. Select the **Behaviour** tab.
5. Check the **Build (Incremental build)** checkbox to build your project incrementally.



Build configuration: Debug_SC3850_PACC [Active]

Build settings

☑ Stop on first build error

□ Use parallel build

○ Use optimal jobs number

● Use parallel jobs: 1

Workbench Build Behavior

Workbench build type: Make build target:

□ Build on resource save (Auto build)

Note: See Workbench automatic build preference

☑ Build (Incremental build)

☑ Clean

Figure 4-5. Properties Dialog Box-C/C++ Build Page

6. Click OK.

Also, you need to check if there are some pre-build steps associated with your build configuration. If yes, then the CodeWarrior will always perform a full rebuild. Follow the steps given below to check if there are any pre-build steps associated with your project.

1. In the CodeWarrior Project window, right-click on project name.

A context menu appears.

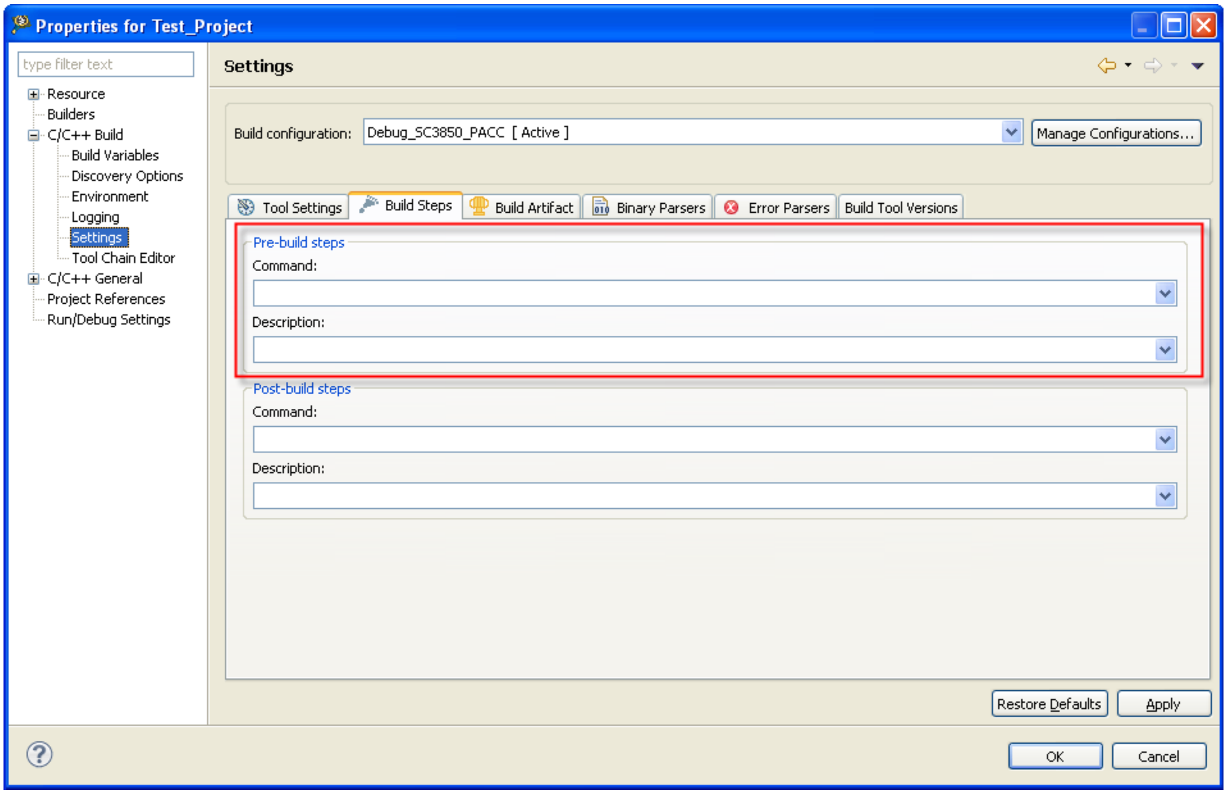
2. From the context menu, select Properties.

   The Properties window appears.

3. Select C/C++ Build > Settings.

   The Settings options appear on right hand side of properties window.

4. Click the **Build Steps** tab.
5. Check if pre-build steps are defined in the **Command** field.



Build configuration: Debug_SC3850_PACC [Active]

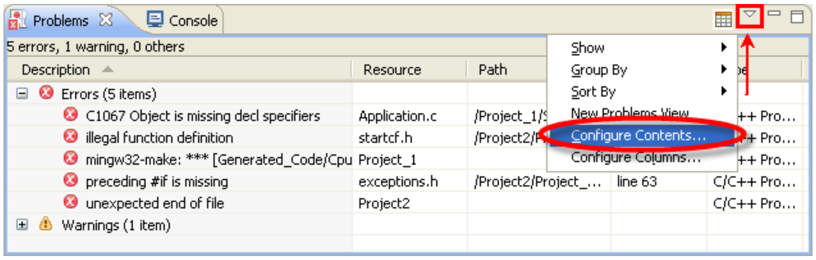
Figure 4-6. Properties Dialog Box-Settings Page

6. Click Ok.

4.1.14 How do I filter the items listed in the Problems view based on the project selected?

To filter the items listed in the **Problems** view based on the project selected:

1. Select the inverted triangle from the the top left corner of the **Problems** view.



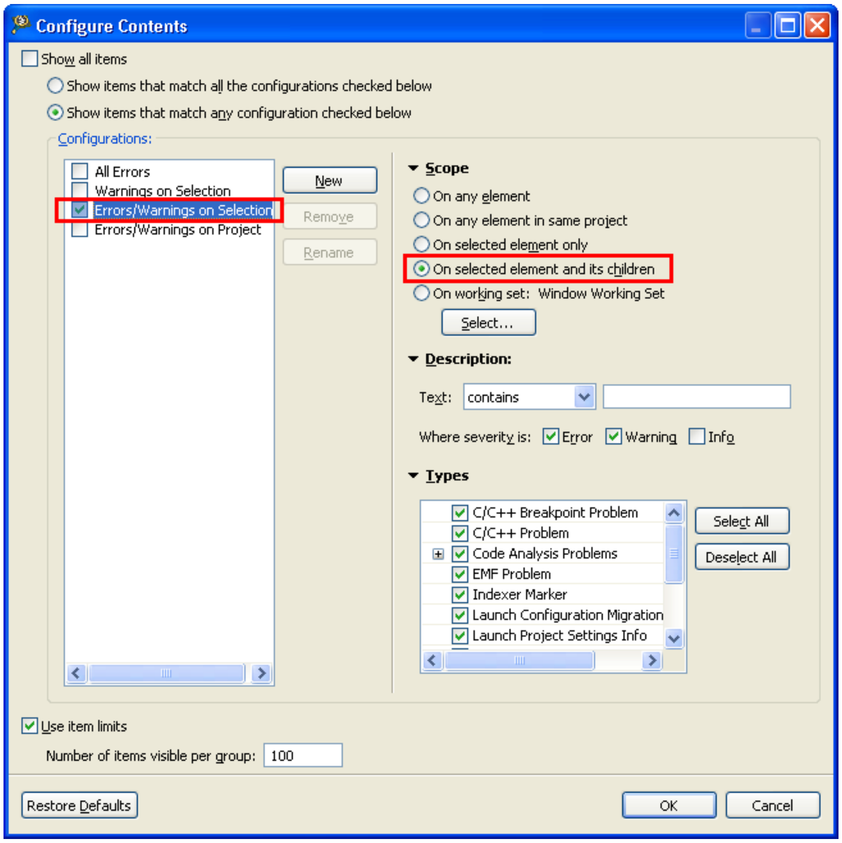
5 errors, 1 warning, 0 others

Description	Resource	Path		
Errors (5 items)				
C1067 Object is missing decl specifiers	Application.c	/Project_1/S		++ Pro...
illegal function definition	startcf.h	/Project2/P		++ Pro...
mingw32-make: *** [Generated_Code/Cpu	Project_1			++ Pro...
preceding #if is missing	exceptions.h	/Project2/Project_...	line 63	C/C++ Pro...
unexpected end of file	Project2			C/C++ Pro...
Warnings (1 item)				

Figure 4-7. Problems View

2. Select **Configure Contents** from the menu.

   The **Configure Contents** dialog box appears.



□ Show all items

○ Show items that match all the configurations checked below

● Show items that match any configuration checked below

Configurations:

□ All Errors
□ Warnings on Selection
☑ Errors/Warnings on Selection
□ Errors/Warnings on Project

Scope

○ On any element
○ On any element in same project
○ On selected element only
● On selected element and its children
○ On working set: Window Working Set

Description:

Text: contains

Where severity is: ☑ Error ☑ Warning □ Info

Types

☑ C/C++ Breakpoint Problem
☑ C/C++ Problem
☑ Code Analysis Problems
☑ EMF Problem
☑ Indexer Marker
☑ Launch Configuration Migration
☑ Launch Project Settings Info

☑ Use item limits

Number of items visible per group: 100

Figure 4-8. Configure Contents Dialog Box

3. Check the **Errors/Warnings on Selection** checkbox from the **Configurations** list in the left panel.
4. Make sure that the **On selected element and its children** option is selected from the **Scope** list in the right panel.
5. Click **OK** .

The **Problems** view lists the problems related to the selected project only.

Chapter 5
Compiler

In this chapter, you find Frequently Asked Questions (FAQs) related to compiler settings in the CodeWarrior IDE.

- FAQs - Compiler

5.1 FAQs - Compiler

This topic lists the compiler related FAQs.

- How do I compile my project?
- Where do I see the warning and error messages?
- When I compile my project, where can I see the commands that are being executed?

5.1.1 How do I compile my project?

In order to compile the project, follow steps below:

1. From CodeWarrior Projects, right-click on project name.

   A context menu appears.

2. From context menu, select **Clean Project** .

   If **Build Automatically** is checked, then when **Clean Project** executes, it creates a **Binaries** folder that contains the *.abs file that are used for debugging. If **Build Automatically** is unchecked, then **Clean Project** removes any previously-built binaries, including the *.abs file.

3. Select **Project > Build Project** to compile.

The project compiles.

5.1.2 Where do I see the warning and error messages?

The warning and errors messages appear in the **Problems** view. To display the **Problems** views select **Window > Show View > Problems** . The **Problems** view appears docked to the lower part of the screen. Double click on error message in **Problems** view to navigate to the relevant source code. The **Problems** view shows problems for all open projects. If there is more than one open project, the window may show problems not associated with the project you are building. Compiler and linker error message are also shown in the **Problems** view.

5.1.3 When I compile my project, where can I see the commands that are being executed?

The commands that are being executed from the compiler or linker can be viewed in the console view. From the IDE menu bar, select **Window > Show View > Console** . The **Console** view appears docked to the lower part of the screen.

Chapter 6
Linker

In this chapter, you find Frequently Asked Questions (FAQs) related to the linker settings in the CodeWarrior IDE.

- FAQs - Linker

6.1 FAQs - Linker

This topic lists the linker related FAQs.

- Can I use a different linker command file than the one being used in my project?

6.1.1 Can I use a different linker command file than the one being used in my project?

Yes, the linker command file can be changed by following the steps given below:

1. In the CodeWarrior Projects window, right-click on project name.

   A context menu appears.

2. From context menu, select **Properties**.

   The **Properties** window appears.

3. Select **C/C++ Build > Settings**.

   The **Settings** options appear on right hand side of properties window.

4. Click **Apply**.
5. Click **OK**.

The linker command file changes, as specified.

Chapter 7
Debugger

In this chapter, you find Frequently Asked Questions (FAQs) related to the debugger settings in the CodeWarrior IDE.

The FAQs listed in this chapter are divided into the following categories:

- Breakpoints
- NEW Variables
- Build Configuration
- Miscellaneous

7.1 Breakpoints

In this topic, FAQs related to breakpoints are listed.

- How do I set breakpoints?
- How do I view the breakpoints that have been set?
- How do I disable existing breakpoints?
- What does the checkmark next to the breakpoint represent?
- What is the difference between the Toggle Breakpoint and Enable/Disable breakpoint options?
- Which breakpoint option do I select if I want to add a breakpoint?
- Which breakpoint option do I select if I want to skip a breakpoint while still keeping it in my breakpoint list?
- What if I want to delete a breakpoint?
- How do I place a breakpoint on a given physical address?
- How do I add hardware breakpoints? Can I add them through the GUI?
- How can I view the hardware breakpoints that have been set?
- What is the difference between the hardware breakpoint and the software breakpoint? Why would I use one over the other?

- What is the purpose of the Limit New Breakpoint to Active Debug Context button in the Breakpoint window?
- Is there a way to add breakpoints to C and Assembly files that exist in a library?
- How do I set a breakpoint at a particular function?
- What is the difference between a breakpoint and a line breakpoint?
- The warnings symbols in Editor distort visibility of breakpoints. How can I stop these warning symbols from appearing?
- NEW While debugging in RAM my debugger halts and reports an "illegal breakpoint" error. What do I do?
- NEW Can I use software breakpoints on code running in flash memory?
- NEW What is a hardware breakpoint?
- NEW How do I select the type of breakpoint for my application in the Editor view?

7.1.1 How do I set breakpoints?

To set breakpoints in a source or assembly file, follow one of the four methods given below.

1. Right-click on the left-hand side of source file (the Gray bar)

   A context menu appears.

2. Select **Toggle Breakpoint** at the line where you want the breakpoint.

-or-

1. Select **Ctrl+Shift+B** at the line where you want the breakpoint.

-or-

1. Double-click on the line where you want the breakpoint

-or-

1. Click on the line where you want the breakpoint.
2. From the IDE menu bar, select **Run > Toggle Breakpoint** .

A breakpoint appears on the left hand side of source code file.

7.1.2 How do I view the breakpoints that have been set?

To view the breakpoints that have been set, select **Window > Show View > Breakpoints** from the IDE menu bar. The **Breakpoints** window appears docked to the upper right-hand side of debugger perspective. The breakpoints window displays the breakpoints that you have enabled.

7.1.3 How do I disable existing breakpoints?

To disable existing breakpoints:

1. Click on the line that has the breakpoint enabled.
2. From the IDE menu bar, select **Run > Toggle Breakpoint**.

-or-

- Press **Ctrl+Shift+B** at the source line where you want the breakpoint disabled.

-or-

- Double-click the line where you want the breakpoint disabled.

-or-

1. Right-click on the left-hand side of source file (the Gray area)

   A context menu appears.

2. Select **Toggle Breakpoint**.

-or-

1. Select the breakpoint that you want disabled in the **Breakpoints** window.
2. Right-click the breakpoint.

   A context menu appears.

3. Select **Disable**.

The Breakpoint gets disabled.

7.1.4 What does the checkmark next to the breakpoint represent?

The checkmark next to a breakpoint indicates that the debugger has resolved the breakpoint and will be hit by the debugger. If the breakpoint does not have a checkmark, the debugger will not hit it.

7.1.5 What is the difference between the Toggle Breakpoint and Enable/Disable breakpoint options?

The **Toggle Breakpoint** option adds or removes breakpoints to the source and breakpoint list. The **Disable/Enable** option makes breakpoints either active or inactive. New breakpoints cannot be added using the **Disable/Enable** breakpoint option.

7.1.6 Which breakpoint option do I select if I want to add a breakpoint?

To add a breakpoint, select **Toggle Breakpoint** .

7.1.7 Which breakpoint option do I select if I want to skip a breakpoint while still keeping it in my breakpoint list?

If you want to skip a breakpoint, but still keeping it in your breakpoint list, then select the **Disable Breakpoint** option.

7.1.8 What if I want to delete a breakpoint?

If you want to delete a breakpoint from the source code, select the **Toggle Breakpoint** option.

7.1.9 How do I place a breakpoint on a given physical address?

To add a breakpoint on a given physical address, follow steps below.

1. In the CodeWarrior projects window, right-click on project name.

   A context menu appears.

2. From context menu, select **Debug As > Debug Configurations** .

   The **Debug** window appears.

3. Underneath **CodeWarrior Download** , select project name.
4. Select **Debug** .

   The **Debug** perspective appears.

5. In **Debug** perspective, select **Window > Show View > Debugger Shell** .

   The **Debugger Shell** window appears.

6. In **Debugger Shell** , type `bp <address>`.

This will set breakpoint at the `<address>` memory location.

7.1.10 How do I add hardware breakpoints? Can I add them through the GUI?

You can add hardware breakpoints through the GUI. To add them through the GUI, follow steps below:

1. Right-click on the left hand side of source file (the Gray bar).

   A context menu appears.

2. From context menu, select **Breakpoint Type > C/C++ Hardware Breakpoints**.
3. Select **Toggle Breakpoint** at the line where you want the breakpoint.

The Hardware breakpoint appears next to source line.

7.1.11 How can I view the hardware breakpoints that have been set?

To view the hardware breakpoints, select **Window > Show View > Breakpoints** from the IDE menu bar. The **Breakpoints** window appears.

7.1.12 What is the difference between the hardware breakpoint and the software breakpoint? Why would I use one over the other?

A software breakpoint inserts a debug instruction into the code, so it only works in the writable memory. If you are working in ROM, then you have to use a hardware breakpoint. The hardware breakpoints make use of the hardware debug support using OCE.

7.1.13 What is the purpose of the Limit New Breakpoint to Active Debug Context button in the Breakpoint window?

This feature is basically a toggle button in the Breakpoint window that causes a new breakpoint (or a watchpoint) to get set only on the active debug context. This feature lets you set breakpoints at certain locations in different source file instead of same breakpoints being set in the same files. The behavior that happens right now is if you set a breakpoint in one file that is being shared by multiple cores it will set that breakpoint in the same file for all cores. With this feature you will be able to set different breakpoints in different files.

If no debug context exists, then the breakpoint is installed in all contexts as normal. Note that once set, the behavior of the breakpoint is the same as the existing filtering behavior, such that filtering is maintained for the individual context during a restart but is lost after a Terminate. After a terminate, the breakpoint is installed in all debug contexts.

7.1.14 Is there a way to add breakpoints to C and Assembly files that exist in a library?

Yes. You need to make sure that you compile the library with debug information. You also need to go to Project Properties and make sure that the project has been imported. Make sure that you put a checkmark on the library project.

7.1.15 How do I set a breakpoint at a particular function?

To set a breakpoint at a particular function, you can either do it through the debugger shell or the debugger GUI. If you are going to set it from the debugger shell, you would type in `bp <function_name>`. If you are going to set it from the GUI, then you right-click the function name and select **Toggle Breakpoint** .

7.1.16 What is the difference between a breakpoint and a line breakpoint?

They are practically the same thing. Eclipse introduced Method Breakpoints which we don't have in the classic CodeWarrior tools, hence the need for a distinction. The online documentation states that regular breakpoints suspend the execution of a thread before a line of code or method is executed. Regular breakpoints include:

- Line Breakpoint; that suspends thread execution when the line of code it is applied to is executed.
- Method Breakpoint; that suspends thread execution when the method that it was applied to is entered or exited (or both).
- Address Breakpoint; that breaks when the core executes code at the given address.

7.1.17 The warnings symbols in Editor distort visibility of breakpoints. How can I stop these warning symbols from appearing?

In order to remove these warning symbols, follow the steps below in the C/C++ perspective:

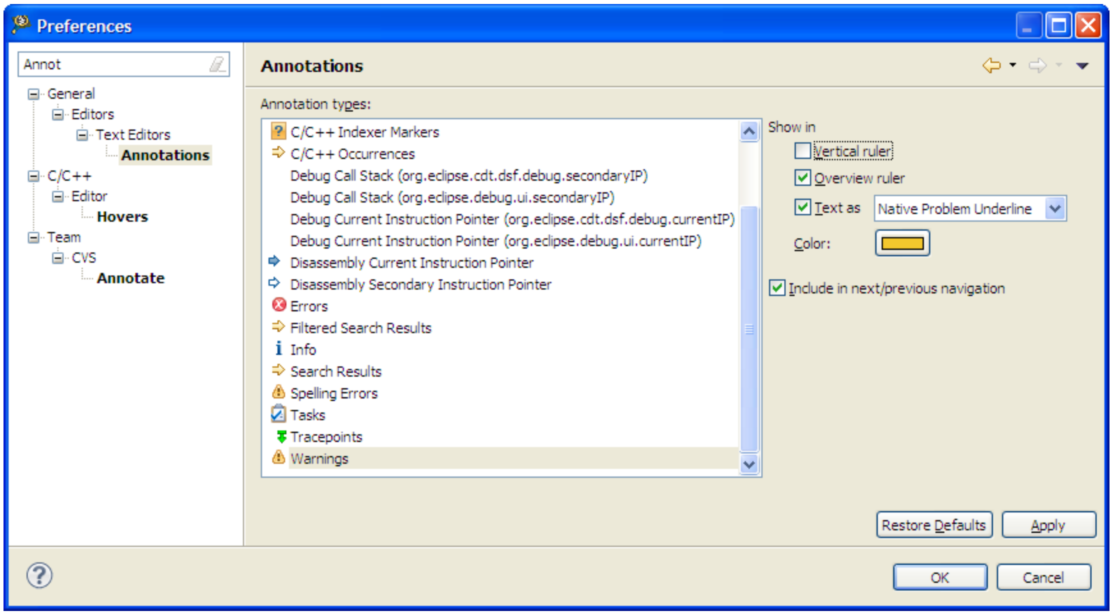
1. From the IDE menu bar, select **Window > Preferences** .

   The **Preferences** window appears.

2. Select **General > Editors > Text Editors > Annotations** .

   The **Annotations** settings pane appears.

3. Select **Warnings** .
4. In the **Show in** area, clear the **Vertical ruler** checkbox, as shown in the following figure.


Preferences
Annot
General
Editors
Text Editors
Annotations
C/C++
Editor
Hovers
Team
CVS
Annotate
Annotations
Annotation types:
C/C++ Indexer Markers
C/C++ Occurrences
Debug Call Stack (org.eclipse.cdt.dsf.debug.secondaryIP)
Debug Call Stack (org.eclipse.debug.ui.secondaryIP)
Debug Current Instruction Pointer (org.eclipse.cdt.dsf.debug.currentIP)
Debug Current Instruction Pointer (org.eclipse.debug.ui.currentIP)
Disassembly Current Instruction Pointer
Disassembly Secondary Instruction Pointer
Errors
Filtered Search Results
Info
Search Results
Spelling Errors
Tasks
Tracepoints
Warnings
Show in
Color:
Restore Defaults
Apply
OK
Cancel

□ Vertical ruler

☑ Overview ruler

☑ Text as Native Problem Underline

☑ Include in next/previous navigation

Figure 7-1. Preferences Window - Annotations Page

5. Click **Apply** .
6. Click **OK** .

The warning symbols are removed.

7.1.18 While debugging in RAM my debugger halts and reports an "illegal breakpoint" error. What do I do?

If the debugger fails to restore the original instruction, then the software breakpoints remains and causes a target halt when it reaches that instruction. The problem might be caused by a board or some hardware problem. In such case, the debugger breakpoint list might gets corrupted too. To solve this problem it is best to reload the application to the target.

7.1.19 Can I use software breakpoints on code running in flash memory?

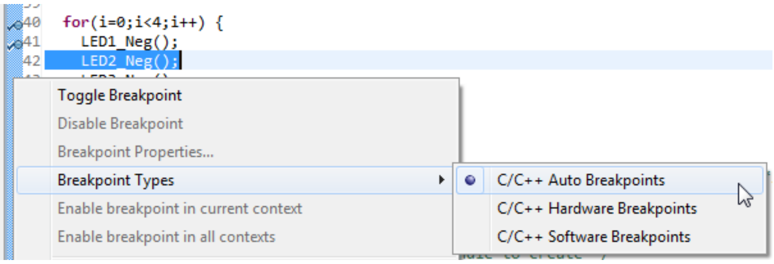
Some debuggers allows software breakpoints on code running in the flash memory. But this usually makes setting and removing breakpoints a time-consuming practice. The debugger has to erase and re-program the flash to set and remove the breakpoints. Other architectures implement a *patch list*, similar to the debugger breakpoint list in RAM. With this it is possible to set software breakpoints with hardware.

7.1.20 What is a hardware breakpoint?

A hardware breakpoint watch an internal bus or the program counter, and if it matches a certain condition, it stops the processor, or do whatever the hardware implements for that condition.

7.1.21 How do I select the type of breakpoint for my application in the Editor view?

To select the type of breakpoint in the **Editor** view, right-click on the marker bar and select the type of breakpoint from the **Breakpoint Types** option's menu list. The following figure shows the context-menu with **Breakpoint Type** option's menu list:



```
for(i=0;i<4;i++) {
  LED1_Neg();
  LED2_Neg();
```

Toggle Breakpoint
Disable Breakpoint
Breakpoint Properties...
Breakpoint Types
Enable breakpoint in current context
Enable breakpoint in all contexts
● C/C++ Auto Breakpoints
C/C++ Hardware Breakpoints
C/C++ Software Breakpoints

Figure 7-2. Breakpoint Type Menu

This is especially useful if you are running an application with parts in RAM and parts in FLASH, or if you want to use hardware breakpoints for code in RAM.

7.2 Variables

In this topic, FAQs related to variables are listed.

- NEW By default, the debugger shows me the global variables of the currently debugged source file. How can I view my variables from another module?
- NEW How do I debug my application using global variables?
- NEW How do I filter variables in files?
- NEW How do I enable watch on a variable?
- NEW What is a static local variable?
- NEW When I enable watch on a static variable, it shows the error, "<error(s)_during_the_evaluation>". What do I do?

7.2.1 By default, the debugger shows me the global variables of the currently debugged source file. How can I view my variables from another module?

While debugging, the debugger lists the local and global variables of the *currentthread* and stack frame only in the **Variables** view. To view the variables from other modules, you have to debug your application using global variables.

7.2.2 How do I debug my application using global variables?

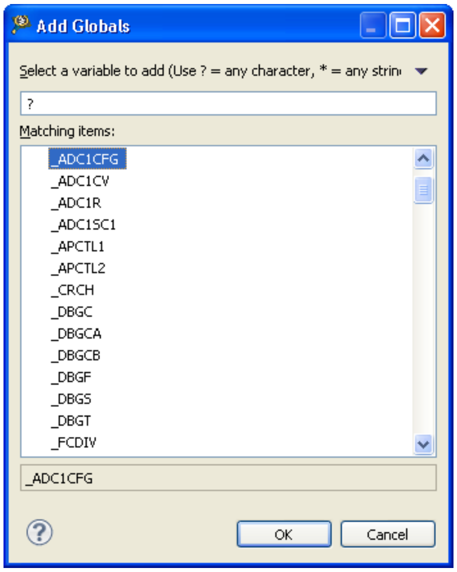
You can debug your application using the variables outside of the current thread or stackframe by adding global variables from the **Variables** view.

To add a global variable:

1. Start a debugging session.
2. The IDE switches to the **Debug** perspective.
3. Click the **Add Global Variables** button from the **Variables** view toolbar. Alternatively, right-click in the **Variables** view and select **Add Global Variables** option from the context menu.

   The **Add Globals** dialog box appears listing all global variables present in your application.

   Tip
   Select **Window > Show View > Variables** from the IDE menu bar to open the **Variables** view.



Add Globals

Select a variable to add (Use ? = any character, * = any strin

?

Matching items:

_ADC1CFG
_ADC1CV
_ADC1R
_ADC1SC1
_APCTL1
_APCTL2
_CRCH
_DBGC
_DBGCA
_DBGCB
_DBGF
_DBGS
_DBGT
_FCDIV

_ADC1CFG

Figure 7-3. Add Globals Dialog Box

4. Enter the variable name in the text box, or you can enter a filter string and select the desired variable from the **Matching items** list.
5. Click **OK** .

The selected variable appears in the **Variables** view.

7.2.3 How do I filter variables in files?

To filter the variables in files:

1. Click the **Add Global Variables** button from the **Variables** view toolbar.

   The **Add Globals** dialog box appears.

2. Use the `<variableName>#<fileName>` format in the text box to filter the file.

The desired variable appears in the **Matching items** list.

7.2.4 How do I enable watch on a variable?

To enable **Watch** on a variable:

1. Select the variable for which you want to enable watch in the **Variables** view.
2. Right-click and select **Watch** from the context menu.

The **Expressions** view appears listing the selected variable.

7.2.5 What is a static local variable?

The static local variables are like global variables, but with the function scope.

7.2.6 When I enable watch on a static variable, it shows the error, "<error(s)_during_the_evaluation>". What do I do?

To enable **Watch** on the *static* variables:

1. Click on the static variable name with error in the **Expressions** view's **Name** column.

   The editing mode is enabled for the selected variable.
2. Edit the variable name to this format, `""FileName"::variableName"`.

   For example:

   - For `static_global` variable in the `Application.c` file, enter the expression, `""static_global"::Application.c"`
   - For `static_local` variable in the `Application.c` file, enter the expression, `"&"static_global"::Application.c"`

The watch is now enabled for the edited static variable.

7.3 Build Configuration

In this topic, FAQs related to build configuration are listed.

- How do I create a new build configuration for my project?
- Is it possible to exclude a single or multiple files from the project while building it?
- Is it possible to configure build settings for a particular source code file?
- Is it possible to store output of the build configuration of my project in a separate directory?
- How can I disable the option of having the CodeWarrior IDE build my project before launching it?

- How can I export the launch configurations settings and then import them back into the project?
- How do I ensure that the launch configurations are a part of the project when it is exported?

7.3.1 How do I create a new build configuration for my project?

To create a new build configuration:

1. Select the project for which you want to create the build configuration in the **CodeWarrior Projects** view.
2. From the IDE menu bar, select **Project > Build Configurations > Manage** .

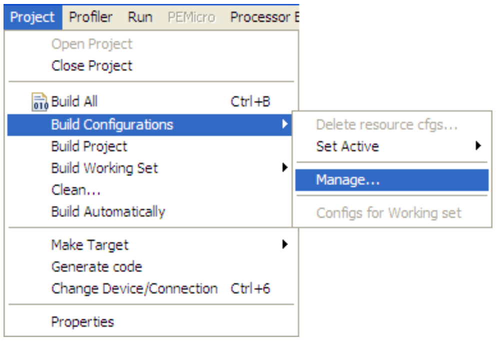


Figure 7-4. Project > Build > Configuration

3. The **Manage Configurations** dialog box appears.
4. Click the **New** button to create a new build configuration.

   The **Create New Configuration** dialog box appears.

5. Enter the name and description for the build configuration.
6. Select the appropriate option from the **Copy settings from** section and select the required option. For example, you may choose to copy settings from an existing configuration and select the required configuration.
7. Click **OK** to close the **Create New Configuration** dialog box.
8. Click **OK**.

To configure build settings of the newly created build configuration:

1. Select the project with which the build configuration is associated in the **CodeWarrior Projects** view.
2. From the IDE menu bar, select **Project > Properties**.

   The **Properties** dialog box appears.

3. Select **C/C++ Build > Settings**.
4. Select the newly created configuration from the **Build Configuration** drop-down list.
5. Specify the settings for the selected build configuration, as required.
6. Click **OK**.

The build settings are configured, as specified.

7.3.2 Is it possible to exclude a single or multiple files from the project while building it?

Yes, you can exclude single or multiple files from build:

1. Select the file in the **CodeWarrior Projects** view. To select multiple files, press and hold the **Ctrl** key.
2. Right-click and select **Resource Configurations** > **Exclude from build**

   The **Exclude from build** dialog box appears.

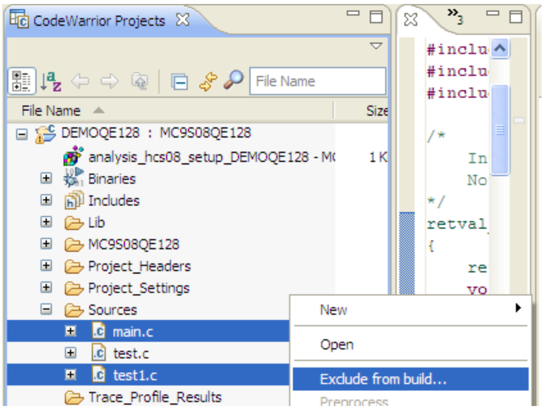


Figure 7-5. Exclude from build

3. Select the build configuration from which you want to exclude the selected file.
4. Click **OK**.

The selected files are excluded from the build configuration.

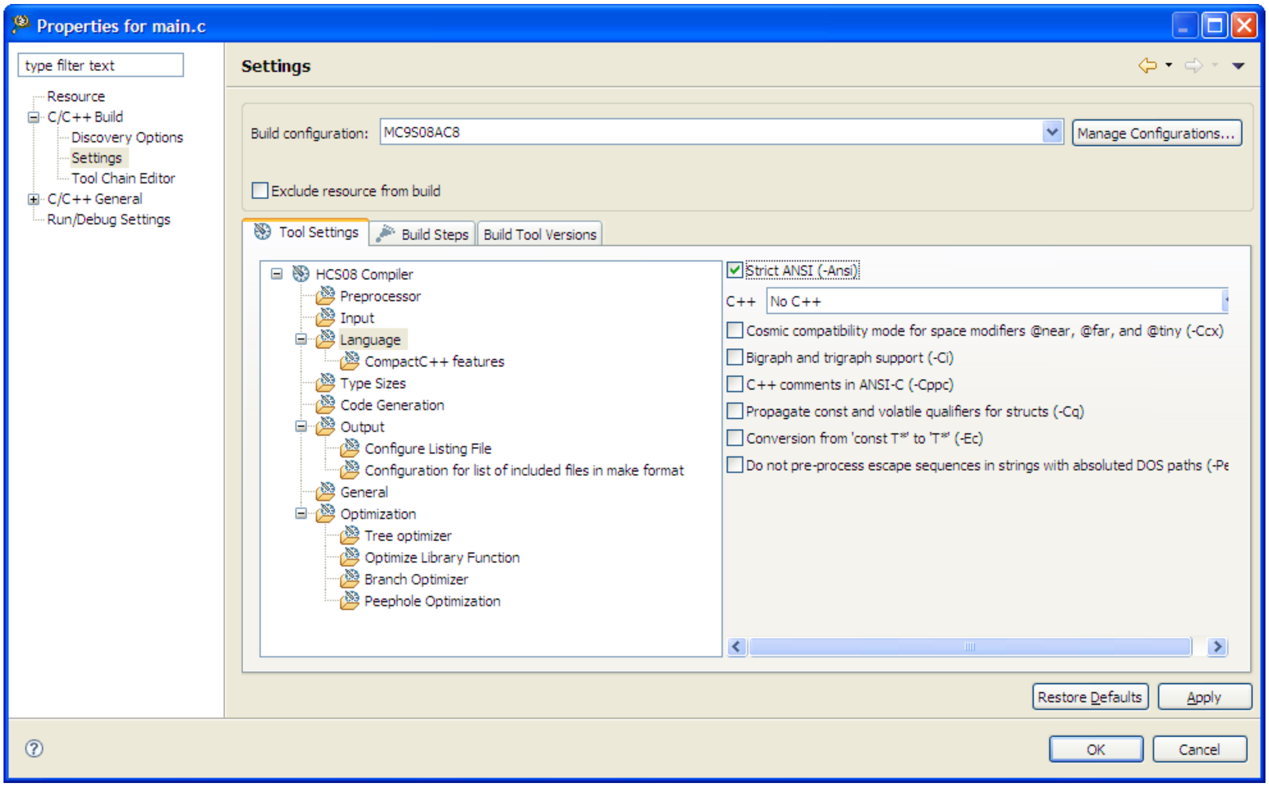
7.3.3 Is it possible to configure build settings for a particular source code file?

Yes, to set file specific build options:

1. Right-click the file in the **CodeWarrior Projects** view and select **Properties** .

   The **Properties** dialog box for the selected file appears.

2. Specify the required settings. For example, you can select a specific compiler option for compiling the file.



Properties for main.c

Settings

Build configuration: MC9S08AC8

☐ Exclude resource from build

☑ Strict ANSI (-Ansi)

C++ No C++

☐ Cosmic compatibility mode for space modifiers @near, @far, and @tiny (-Ccx)

☐ Bigraph and trigraph support (-Ci)

☐ C++ comments in ANSI-C (-Cppc)

☐ Propagate const and volatile qualifiers for structs (-Cq)

☐ Conversion from 'const T*' to 'T*' (-Ec)

☐ Do not pre-process escape sequences in strings with absoluted DOS paths (-Pe

Figure 7-6. Properties Dialog Box for Selected File

3. Click **OK** .

The file specific settings are configured, as specified.

7.3.4 Is it possible to store output of the build configuration of my project in a separate directory?

Yes, you can specify output directory for the build configuration.

1. Select the project with which the build configuration is associated in the **CodeWarrior Projects** view.
2. From the IDE menu bar, select **Project > Properties**.

   The **Properties** dialog box appears.

3. Select **C/C++ Build**.

   The **C/C++ Build** page appears in the right panel of the **Properties** dialog box.

4. Select the build configuration from the **Configuration** drop-down list.
5. Specify the required build directory in the **Build directory** text box.
6. Click **Apply**.
7. Click **OK**.
8. Right-click the project and select **Build Configurations > Build Selected**.

   The **Clean and Rebuild Configurations** dialog box appears.

9. Select the build configuration to build and click **OK**.

The project build begins and the output is stored in the specified directory.

7.3.5 How can I disable the option of having the CodeWarrior IDE build my project before launching it?

To disable build before launch:

1. From the IDE menu bar, select **Window > Preferences** .

   The **Preferences** dialog box appears.

2. Expand the tree control to select **Run/Debug > Launching** .

   The **Launching** page appears in the right pane.

3. Clear the **Build (if required) before launching** checkbox.
4. Click **Apply** .
5. Click **OK** .

Now, CodeWarrior IDE will not build the project before launching it.

7.3.6 How can I export the launch configurations settings and then import them back into the project?

You can export and import the launch configuration settings using the Export and Import wizards.

To export a launch configuration:

1. From the CodeWarrior menu bar, select **File > Export** .

   The **Export** dialog box appears.

2. Expand **Run/Debug** tree control and select **Launch Configurations** .

   The **Export Launch Configurations** wizard appears.

3. Select the launch configuration you want to export.
4. In the **Location** field, specify the location where you want to export the launch configuration.
5. Click **Finish** .

   The selected launch cofiguration exports as `.launch` file.

Similarly, you can import the launch configuration (.launch file) using **File > Import** in the CodeWarrior IDE menu bar.

Alternatively, you can copy the `*.launch` files present in the `Project_Settings\Debugger` folder of a project and place it in the project where you want to export it.

7.3.7 How do I ensure that the launch configurations are a part of the project when it is exported?

To make a launch configurations be part of the project, follow steps below:

1. From the IDE menu bar, select **Run > Debug Configurations** .

   The **Debug** window appears.

2. Under **CodeWarrior Download**, select **the launch configuration of the project that needs to be exported**.
3. Select the **Common** tab.

   The **Common** settings pane appears.

4. Under **Save as** , select the shared file radio button. Leave the default folder name as is.
5. Click **Apply** .

Now the launch configuration is physically located in the project folder, so it will be a part of the project when zipped.

7.3.8 Is it possible to debug the target multiple times with the same binary application, without flashing the device?

Yes, you can debug the target multiple times with the same application.

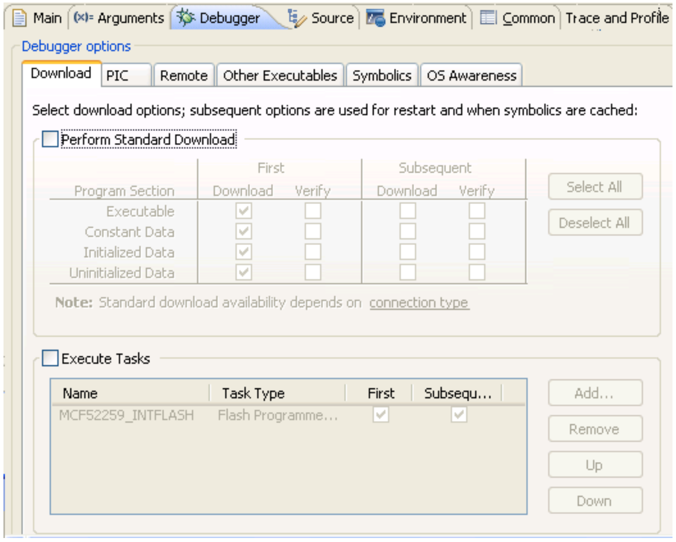
1. Flash the target once with either using a target task (standalone) or using **Flash Programmer > Flash File to Target**.
2. Select **Run > Debug Configurations**.

   The **Debug Configurations** dialog box appears.

3. Select **Debugger > Download**.

   The **Download** page appears in the **Debug Configuration** dialog box.

4. Clear the **Perform Standard Download** and **Execute Tasks** checkboxes.



Select download options; subsequent options are used for restart and when symbolics are cached:

□ Perform Standard Download

	First		Subsequent	
Program Section	Download	Verify	Download	Verify
Executable	☑	□	□	□
Constant Data	☑	□	□	□
Initialized Data	☑	□	□	□
Uninitialized Data	☑	□	□	□

Note: Standard download availability depends on connection type

□ Execute Tasks

Name	Task Type	First	Subsequ...
MCF52259_INTFLASH	Flash Programme...	☑	☑

Figure 7-7. Download Page - Debug Configurations Dialog Box

   Now as the debugger does not need to flash a potentially large application, you can start debugging within a few seconds, and you can do this multiple times.

5. To flash the target, click the **Flash Programmer** button followed by the **Debug** button on the CodeWarrior IDE toolbar.

Now, you can debug the target multiple times with the same binary application, without flashing the device

7.3.9 I need to enable logging for flash programming. How do I do it?

Various issues may occur while flash programming, such as the target is not stable, you might have problems with the target power, or the part might be secured.

To gather more information about such issues, you can enable logging.

1. From the CodeWarrior IDE menu bar, select **Run > Debug Configurations** .

   The **Debug Configurations** dialog box appears.
2. Click the **Edit** button in the **Remote system** section.

   The **Properties for <project_name>** dialog box appears.
3. Check the **Enable Logging** checkbox in the **Connection** page.
4. Click **OK** .
5. Click **Debug** .

The log results appear in the **Console** view.

7.4 Miscellaneous

In this topic, miscellaneous FAQs related to the debugger are listed.

- How can I view only certain registers?
- How can I view addresses in memory?
- How do I open up multiple memory locations in the memory view?
- How can I view variables that I have set in my project?
- How can I change the values of variables?
- How do I view global variables that have been accessed in my project?
- Is the command-line debugger available in the Eclipse-based CodeWarrior IDE as well?
- Is it possible to specify a different name for the binary file (.eld)?
- Can I debug the code from the generated assembly file step-by-step?
- If I have a debugger session running on my application, how can I halt the application?
- Why do I get the following error message when I attempt to debug my project?
- Can I change the format of the Memory view?
- Why does the DE.exe appear in the Windows Task Manager even after terminating and closing the debugger perspective?

- Is there a way to display the variables always in the Hexadecimal format?
- Can I change the endianness that is displayed in the Memory view?
- In the Memory view, is it possible to go back to the address that the rendering was created for?
- Is there a way to view disassembly at any location?
- Is it possible to display the expressions always in the Hexadecimal format instead of the default decimal format?
- What does the Reset on Connect option do?
- Can I change the Program Counter (pc) value while in the debugger without having to restart the debugger?
- How do I specify the program entry point?
- How can I run a debug session repeatedly?
- Can I direct console output to a file instead of the console window?
- How do I resolve the following error message that I get when I use Ctrl-Click to open variable declaration?

7.4.1 How can I view only certain registers?

To view only certain registers, follow these steps:

1. Select **Window > Show View > Registers** from the IDE menu bar while in the debugger perspective.

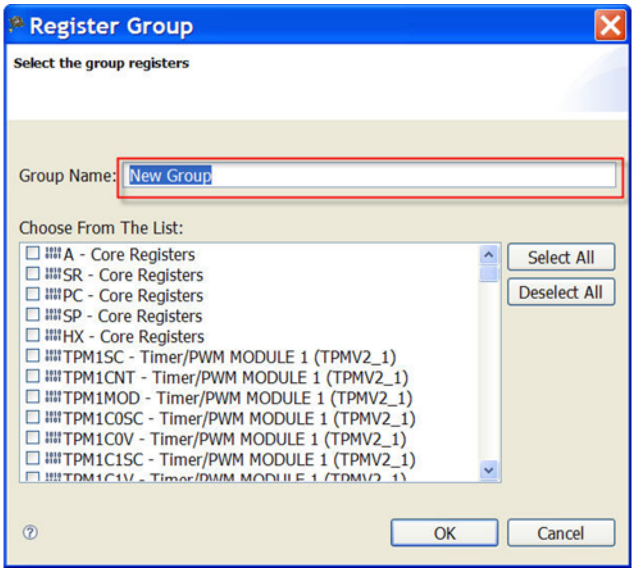
   The **Registers** view appears docked to the upper right hand side of debugger perspective.

2. Right-click anywhere in the **Registers** tab.

   A context menu appears.

3. Select Add Register Group in the context menu.

   The **Register Group** window appears.



Select the group registers

Group Name: New Group

Choose From The List:

- □ A - Core Registers
- □ SR - Core Registers
- □ PC - Core Registers
- □ SP - Core Registers
- □ HX - Core Registers
- □ TPM1SC - Timer/PWM MODULE 1 (TPMV2_1)
- □ TPM1CNT - Timer/PWM MODULE 1 (TPMV2_1)
- □ TPM1MOD - Timer/PWM MODULE 1 (TPMV2_1)
- □ TPM1C0SC - Timer/PWM MODULE 1 (TPMV2_1)
- □ TPM1C0V - Timer/PWM MODULE 1 (TPMV2_1)
- □ TPM1C1SC - Timer/PWM MODULE 1 (TPMV2_1)
- □ TPM1C1V - Timer/PWM MODULE 1 (TPMV2_1)

Figure 7-8. Register Group Window

4. Type the name of new register group in the **Group Name** textbox.
5. Select the checkbox next to registers that you want to be a part of this group.
6. Click **OK** .

   The **Register Group** window closes.

Now you can view only the selected registers.

7.4.2 How can I view addresses in memory?

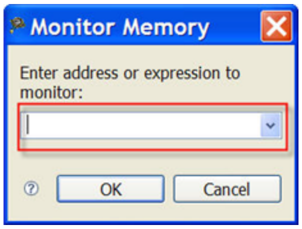
To view addresses in memory, follow these steps:

1. Select **Window > Show View > Memory** in the debug perspective.

   The **Memory** window appears docked to the lower part of debugger perspective.

2. Click + sign icon.

   The **Monitor Memory** window appears.



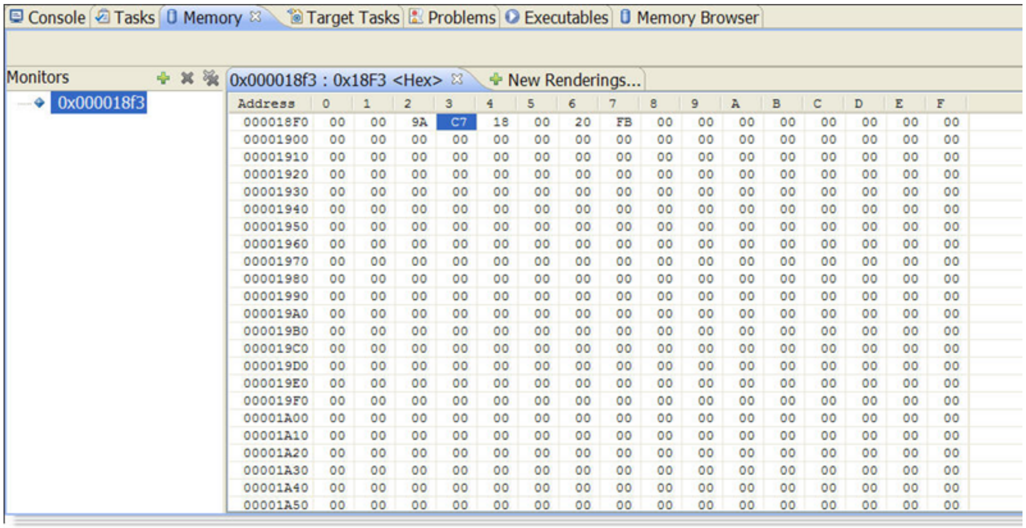
Enter address or expression to monitor:

Figure 7-9. Monitor Memory Window

3. Type memory address in the **Enter address or expression to monitor** textbox.
4. Click **OK** .

   The **Monitor Memory** window closes.

The memory location appears in the memory window. The memory contents can be viewed by selecting memory on the left hand side, as shown in the following figure.



```
Address   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000018F0  00 00 9A C7 18 00 20 FB 00 00 00 00 00 00 00 00
00001900  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001910  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001920  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001930  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001940  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001950  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001960  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001970  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001980  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001990  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 7-10. Memory View

7.4.3 How do I open up multiple memory locations in the memory view?

To open multiple memory locations in the memory view, follow these steps:

1. Select **Window > Show View > Memory** in the debug perspective.

   The **Memory** window appears docked to the lower part of debugger perspective.

2. Click + sign icon.

   The **Monitor Memory** window appears.

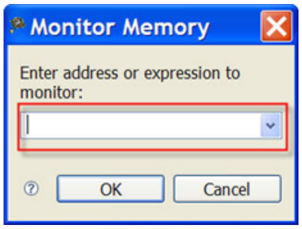


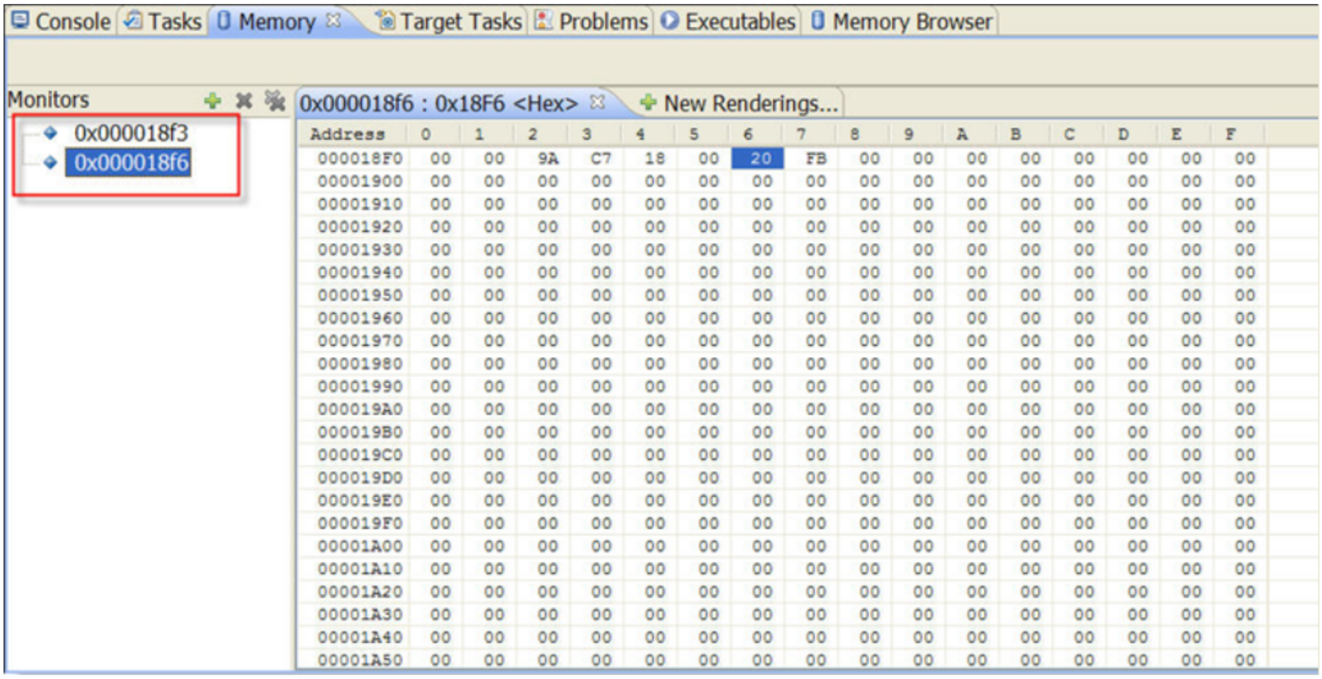
Figure 7-11. Monitor Memory Window

3. Type memory address in the **Enter address or expression to monitor** textbox.
4. Click **OK** .

   The new address appears in the memory window.

5. Click ✚ sign icon.
6. The **Monitor Memory** window appears.
7. Type another memory address in the **Enter address or expression to monitor** textbox.
8. Click **OK** .

The new address appears in the memory window. The contents can be viewed by selecting memory on the left hand side, as shown in the following figure.



```
Address   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000018F0  00 00 9A C7 18 00 20 FB 00 00 00 00 00 00 00 00
00001900  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001910  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001920  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001930  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001940  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001950  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001960  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001970  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001980  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001990  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 7-12. Memory View with Multiple Addresses

7.4.4 How can I view variables that I have set in my project?

To view variables from the debug perspective toolbar, select **Window > Show View > Variables** . The Variable window appears docked on the upper right hand side of debug perspective.

7.4.5 How can I change the values of variables?

To change the value of variables, follow the steps below in the debug perspective:

1. From the toolbar, select **Window > Show View > Variables** . The **Variables** window appears docked on the upper right hand side of debugger perspective.
2. Right-click on the variable.

   A context menu appears.

3. From context menu, select **Change Value** .

   The **Set Value** window appears.

4. In the **Enter a new value for variables** textbox, type a new variable value.
5. Click **OK** .

   The **Set Value** window closes.

The variable value changes in the **Variables** window. The variable being changed is highlighted in yellow.

7.4.6 How do I view global variables that have been accessed in my project?

To view global variables that have been accessed in your project, follow these steps:

1. Select **Window > Show View > Variables** from the IDE menu bar while in the debugger perspective.

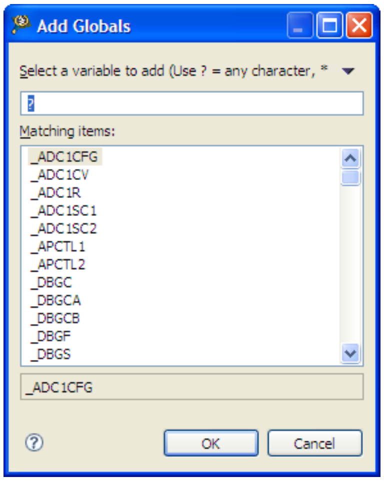
   The **Variables** view appears docked to the upper right hand side of debugger perspective.

2. Right-click a variable.

   A context menu appears.

3. Select **Add Global Variables** in the context menu.

The **Add Globals** dialog box appears.



Select a variable to add (Use ? = any character, *

?

Matching items:

_ADC1CFG
_ADC1CV
_ADC1R
_ADC1SC1
_ADC1SC2
_APCTL1
_APCTL2
_DBGC
_DBGCA
_DBGCB
_DBGF
_DBGS

_ADC1CFG

Figure 7-13. Add Globals Dialog Box

4. Select the global variables that you want to view.
5. Click **OK** .

   The **Add Globals** dialog box closes.

The selected variables appear in the **Variables** view with the symbol that identifies them as global variable.

NOTE

The selected global variables will persist throughout the session and subsequent sessions, until they are removed.

7.4.7 Is the command-line debugger available in the Eclipse-based CodeWarrior IDE as well?

Yes. To use the Command Line Debugger follow steps given below in debug perspective:

1. From the IDE menu bar, select **Window > Show View > Debugger Shell** .
2. Click **OK** .

The Debugger Shell window appears docked on the lower right-hand corner of debug perspective.

7.4.8 Is it possible to specify a different name for the binary file (.eld)?

Yes this can be done by following steps below:

1. In the CodeWarrior Projects window, right-click the project name.

   A context menu appears.
2. From the context menu, select **Properties**.

   The **Properties** window appears.
3. Select **C/C++ Build > Settings**.

   The **Settings** options appear on the right side of properties window.
4. Select **Build Artifact** tab.
5. Type the name of the executable that you want generated in the **Artifact name** textbox.
6. The **Artifact extension** textbox contains the extension of the executable that will be generated. The default extension for binary files is .eld. If another extension is desired, change the extension in the **Artifact extension** textbox.
7. Click **Apply**.
8. Click **OK**.

   The **Properties** window disappears.

When the project is rebuilt a new binary with the new extension is generated.

7.4.9 Can I debug the code from the generated assembly file step-by-step?

Yes this can be done. From the debugger perspective toolbar, select **Window > Show View > Disassembly** . The **Disassembly** window appears displaying the assembly code generated from the C code. Breakpoints can be set by double-clicking on assembly line.

7.4.10 If I have a debugger session running on my application, how can I halt the application?

In the **Debug** perspective, click on the .eld being executed. From toolbar, select **Suspend** from the toolbar which halts the application temporarily. To run the application, select **Resume** from toolbar.

7.4.11 Why do I get the following error message when I attempt to debug my project?

```
Program does not exist
```

This error occurs because the project you are trying to debug has not been built yet. To build the project, right-click the project name and select the **Build Project** option.

7.4.12 Can I change the format of the Memory view?

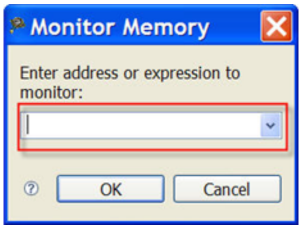
Yes. In order to change the format of the memory window do the following steps:

1. Select **Window > Show View > Memory** in the debug perspective.

   The **Memory** window appears docked to the lower part of debugger perspective.

2. Click ✚ sign icon.

   The **Monitor Memory** window appears.



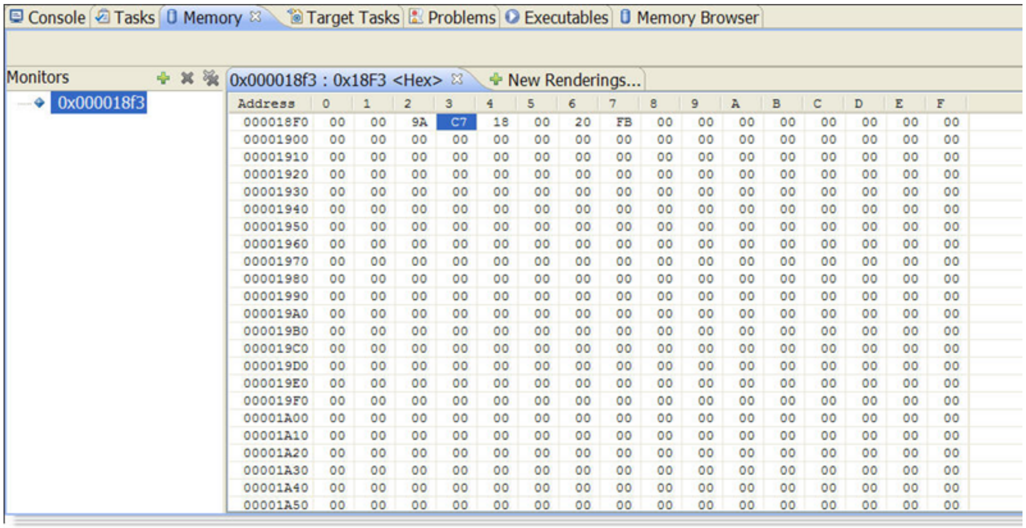
Monitor Memory

Enter address or expression to monitor:

Figure 7-14. Monitor Memory Window

3. Type memory address in the **Enter address or expression to monitor** textbox.
4. Click **OK**.

   The **Monitor Memory** window closes. The memory location appears in the memory window. The memory contents can be viewed by selecting memory on the left hand side, as shown in the following figure.



```
Address   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000018F0  00 00 9A C7 18 00 20 FB 00 00 00 00 00 00 00 00
00001900  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001910  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001920  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001930  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001940  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001950  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001960  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001970  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001980  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001990  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

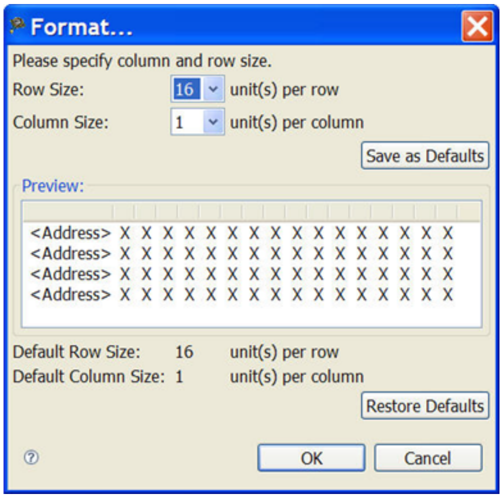
Figure 7-15. Memory View

5. Right-click anywhere in the **Memory** view

   A context menu appears.

6. From context menu, select **Format**.

   The **Format** window appears.



Please specify column and row size.

Row Size: 16 unit(s) per row

Column Size: 1 unit(s) per column

```
<Address> X X X X X X X X X X X X X X X X
<Address> X X X X X X X X X X X X X X X X
<Address> X X X X X X X X X X X X X X X X
<Address> X X X X X X X X X X X X X X X X
```

Default Row Size: 16 unit(s) per row

Default Column Size: 1 unit(s) per column

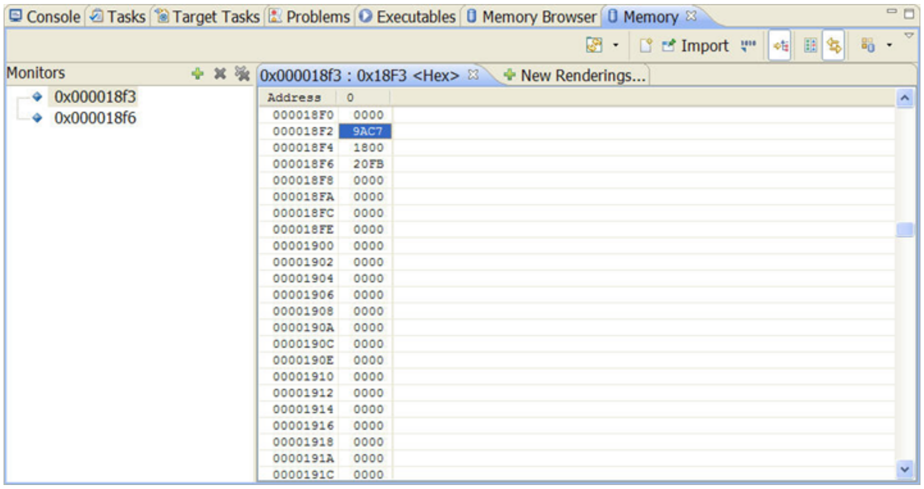
Figure 7-16. Format Dialog Box

Row Size indicates how the addresses will increase. **Column Size** displays the number of bytes in that address space. For this example, the address will increase by 16 bytes and each row will display 4 bytes in each column. The **Preview** window changes and displays address every 2 bytes by 2 bytes per column.

7. Click **OK**.

   The **Format** window closes.

The memory view changes and displays the addresses using new format, as the following image shows.



Monitors
- 0x000018f3
- 0x000018f6

Address	0
000018F0	0000
000018F2	9AC7
000018F4	1800
000018F6	20FB
000018F8	0000
000018FA	0000
000018FC	0000
000018FE	0000
00001900	0000
00001902	0000
00001904	0000
00001906	0000
00001908	0000
0000190A	0000
0000190C	0000
0000190E	0000
00001910	0000
00001912	0000
00001914	0000
00001916	0000
00001918	0000
0000191A	0000
0000191C	0000

Figure 7-17. Updated Memory View

7.4.13 Why does the DE.exe appear in the Windows Task Manager even after terminating and closing the debugger perspective?

The Debugger Engine (DE) is loaded when it is first needed in one of the following situations:

1. Debugger is launched
2. Debugger Shell is opened

The DE.exe is terminated only when CodeWarrior is terminated.

7.4.14 Is there a way to display the variables always in the Hexadecimal format?

Yes. There is a way to display the variables always in Hexadecimal instead of the default decimal format. In order to accomplish this, follow steps below.

1. Select **Window > Preferences** from C/C++ perspective toolbar.

   The **Preferences** window appears.

2. Select **C/C++ > Debug**.

   The **Debug** settings appear.

3. Change **Default variable format** to **Hexadecimal**. By default, it is set to **Natural** (for more information, refer to the image listed below).
4. Click **Apply**.
5. Click **OK**.
6. Restart CodeWarrior Development Studio.

When **Variables** view appears in the debug perspective, the variables appear in the Hexadecimal format.

NOTE

This setting is a global setting.

7.4.15 Can I change the endianness that is displayed in the Memory view?

Yes. To change the endianness that is displayed in the memory view, follow these steps:

1. Select **Window > Show View > Memory** in the debug perspective.

   The **Memory** window appears docked to the lower part of debugger perspective.

2. Click + sign icon.

   The **Monitor Memory** window appears.

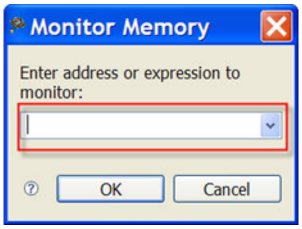
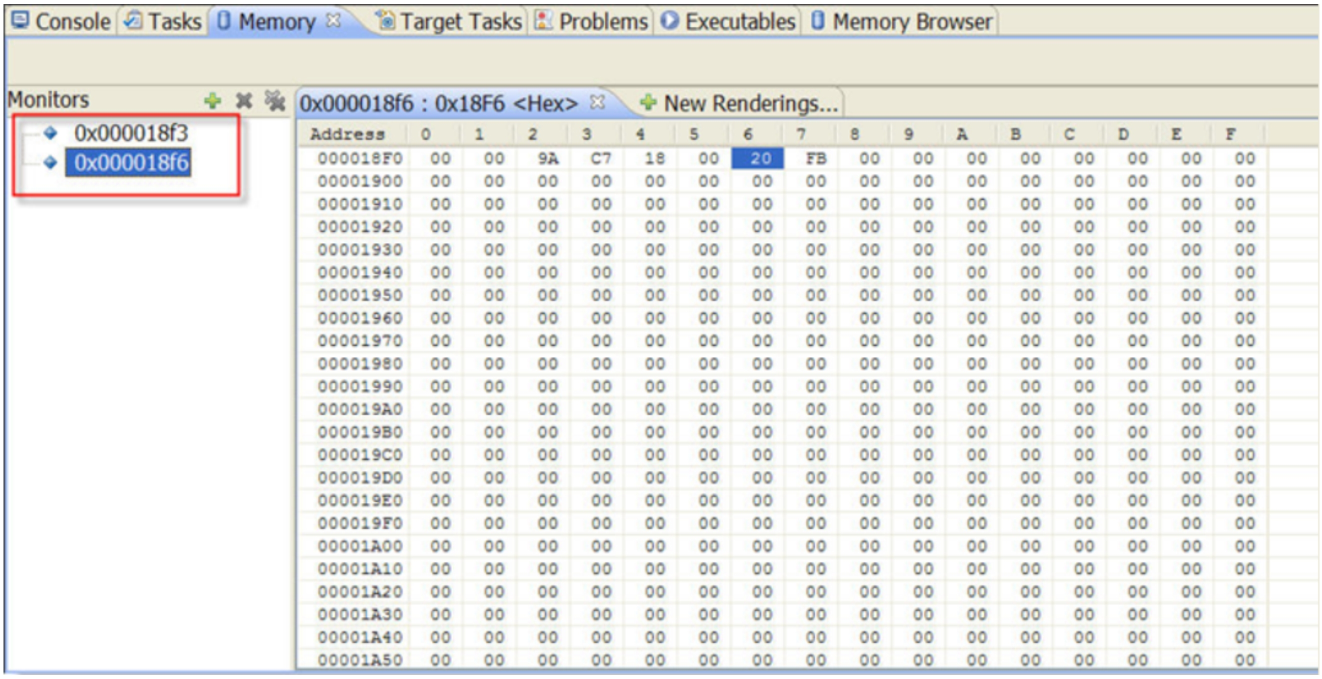


Figure 7-18. Monitor Memory Window

3. Type memory address in the **Enter address or expression to monitor** textbox.
4. Click **OK**.

   The new address appears in the memory window. The contents can be viewed by selecting memory on the left hand side, as the following figure shows.



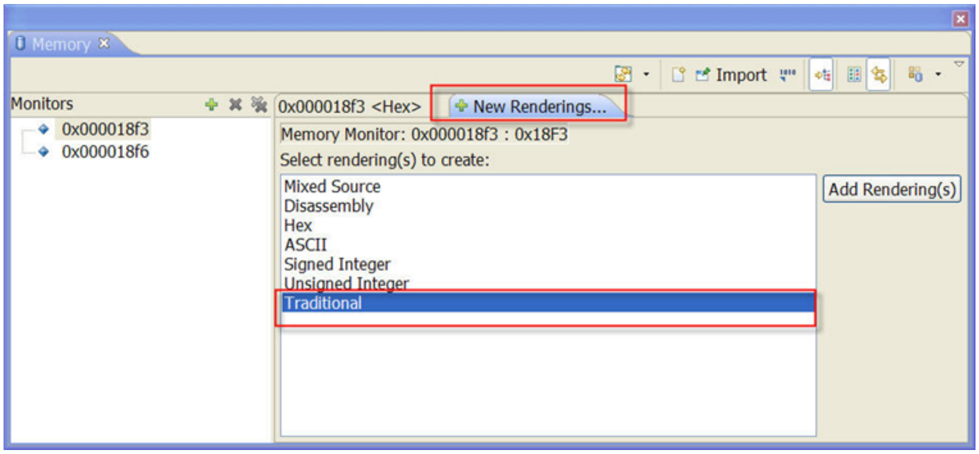
```
Address   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000018F0  00 00 9A C7 18 00 20 FB 00 00 00 00 00 00 00 00
00001900  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001910  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001920  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001930  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001940  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001950  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001960  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001970  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001980  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001990  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001A50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 7-19. Memory View with Multiple Addresses

5. From the **Memory** Toolbar, select New Renderings... .

   The **New Renderings** window appears.

6. Select **Traditional**.



Memory Monitor: 0x000018f3 : 0x18F3

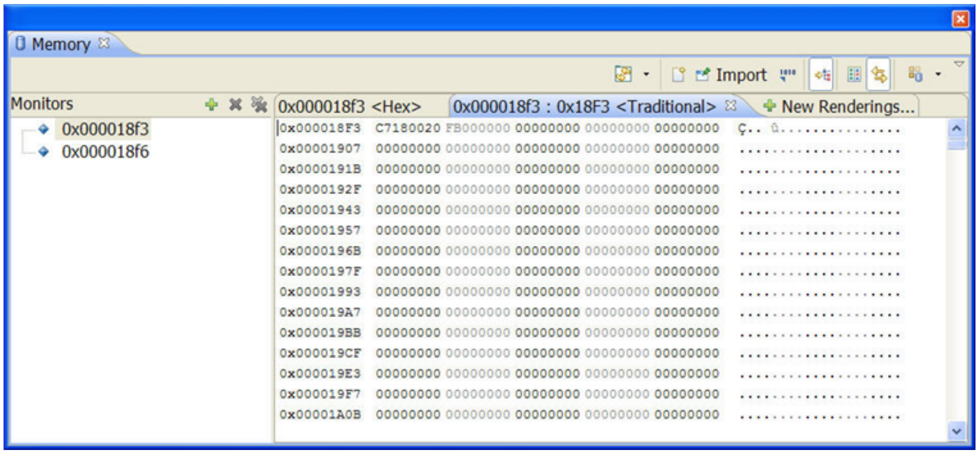
Select rendering(s) to create:

Mixed Source
Disassembly
Hex
ASCII
Signed Integer
Unsigned Integer
Traditional

Figure 7-20. Memory View-New Renderings

7. Click **Add Renderings**.
8. Click **OK**.

   The **Traditional** view appears in Memory view.



```
0x000018F3  C7180020 FB000000 00000000 00000000 00000000  Ç.. û...............
0x00001907  00000000 00000000 00000000 00000000 00000000  ....................
0x0000191B  00000000 00000000 00000000 00000000 00000000  ....................
0x0000192F  00000000 00000000 00000000 00000000 00000000  ....................
0x00001943  00000000 00000000 00000000 00000000 00000000  ....................
0x00001957  00000000 00000000 00000000 00000000 00000000  ....................
0x0000196B  00000000 00000000 00000000 00000000 00000000  ....................
0x0000197F  00000000 00000000 00000000 00000000 00000000  ....................
0x00001993  00000000 00000000 00000000 00000000 00000000  ....................
0x000019A7  00000000 00000000 00000000 00000000 00000000  ....................
0x000019BB  00000000 00000000 00000000 00000000 00000000  ....................
0x000019CF  00000000 00000000 00000000 00000000 00000000  ....................
0x000019E3  00000000 00000000 00000000 00000000 00000000  ....................
0x000019F7  00000000 00000000 00000000 00000000 00000000  ....................
0x00001A0B  00000000 00000000 00000000 00000000 00000000  ....................
```

Figure 7-21. Memory View-Traditional Tab

9. Right-click anywhere in the **Traditional** tab.

   A context menu appears.

10. From the context menu, select **Endian** > **Little**. By default it is **Big**.

The memory view displays little endianness.

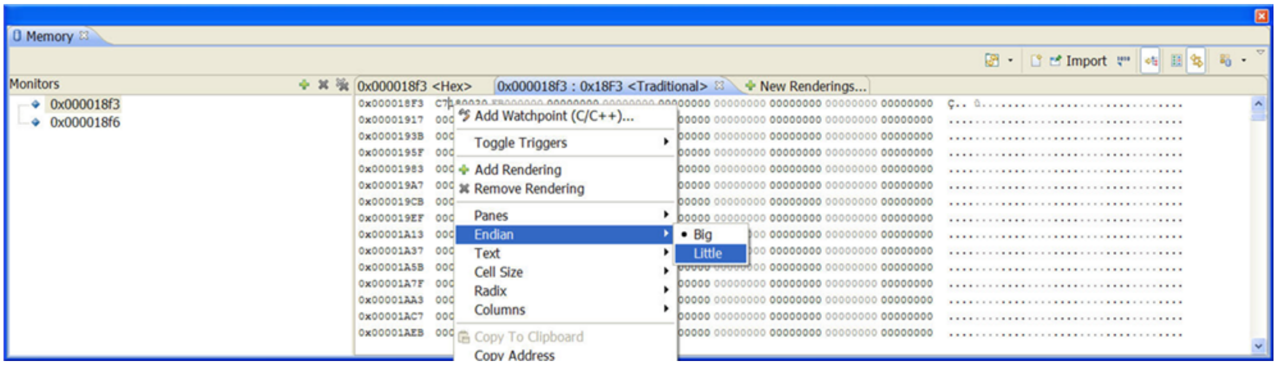


Figure 7-22. Memory View-Little Endian Addresses

NOTE

To change the endianness back to **Big** , right-click anywhere in the **Traditional** tab, and select **Endian > Big** from the context menu.

7.4.16 In the Memory view, is it possible to go back to the address that the rendering was created for?

Yes. In order to get back to the address that the rendering was created for, follow steps below:

1. Right-click anywhere in the **Memory** view.

   A context window appears.
2. From context menu, select **Reset to Base Address** .

The view changes to display the base address that the rendering was initially created for.

7.4.17 Is there a way to view disassembly at any location?

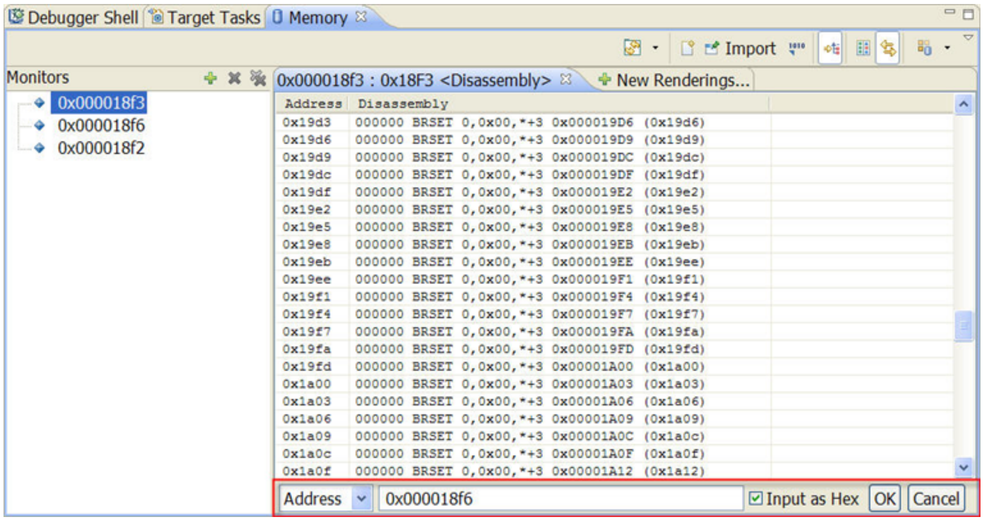
Yes it is possible to view disassembly at any location. To do so, follow steps below:

1. Right-click anywhere in the **Memory** view.

   A context window appears.
2. From context menu, select **Go To Address**.

   The **Go To Address** textbox appears.
3. Enter the address in the Address textbox. If the **Input as Hex** checkbox is selected, then the address can be entered as either a hexadecimal or a decimal address.
4. Click **OK**.

The assembly instructions located at the specified address appears in the **Memory** view (for more information, refer to the image listed below).



```
Address   Disassembly
0x19d3    000000 BRSET 0,0x00,*+3 0x000019D6 (0x19d6)
0x19d6    000000 BRSET 0,0x00,*+3 0x000019D9 (0x19d9)
0x19d9    000000 BRSET 0,0x00,*+3 0x000019DC (0x19dc)
0x19dc    000000 BRSET 0,0x00,*+3 0x000019DF (0x19df)
0x19df    000000 BRSET 0,0x00,*+3 0x000019E2 (0x19e2)
0x19e2    000000 BRSET 0,0x00,*+3 0x000019E5 (0x19e5)
0x19e5    000000 BRSET 0,0x00,*+3 0x000019E8 (0x19e8)
0x19e8    000000 BRSET 0,0x00,*+3 0x000019EB (0x19eb)
0x19eb    000000 BRSET 0,0x00,*+3 0x000019EE (0x19ee)
0x19ee    000000 BRSET 0,0x00,*+3 0x000019F1 (0x19f1)
0x19f1    000000 BRSET 0,0x00,*+3 0x000019F4 (0x19f4)
0x19f4    000000 BRSET 0,0x00,*+3 0x000019F7 (0x19f7)
0x19f7    000000 BRSET 0,0x00,*+3 0x000019FA (0x19fa)
0x19fa    000000 BRSET 0,0x00,*+3 0x000019FD (0x19fd)
0x19fd    000000 BRSET 0,0x00,*+3 0x00001A00 (0x1a00)
0x1a00    000000 BRSET 0,0x00,*+3 0x00001A03 (0x1a03)
0x1a03    000000 BRSET 0,0x00,*+3 0x00001A06 (0x1a06)
0x1a06    000000 BRSET 0,0x00,*+3 0x00001A09 (0x1a09)
0x1a09    000000 BRSET 0,0x00,*+3 0x00001A0C (0x1a0c)
0x1a0c    000000 BRSET 0,0x00,*+3 0x00001A0F (0x1a0f)
0x1a0f    000000 BRSET 0,0x00,*+3 0x00001A12 (0x1a12)
```

Address: 0x000018f6 ☑ Input as Hex

Figure 7-23. Memory View-Assembly Instructions

7.4.18 Is it possible to display the expressions always in the Hexadecimal format instead of the default decimal format?

Yes. It is possible to display the expressions always in Hexadecimal format instead of the default decimal format. In order to accomplish this follow steps below:

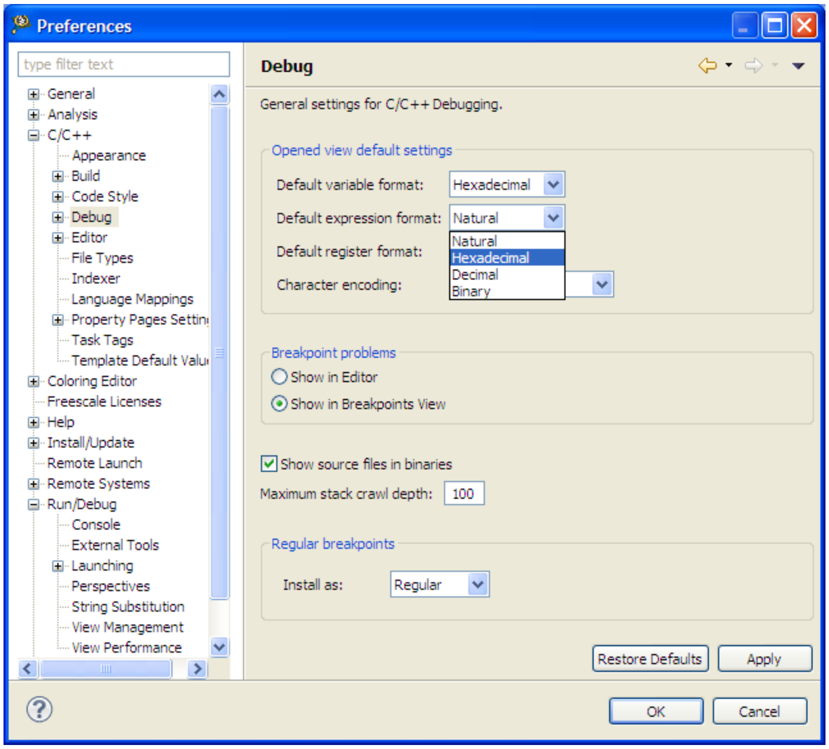
1. Select **Window > Preferences** from C/C++ perspective toolbar.

   The **Preferences** window appears.

2. Select **C/C++ > Debug**.

   The **Debug** settings appear.

3. Change **Default expression format** to **Hexadecimal**. By default, it is set to **Natural** (for more information, refer to the image listed below).



Debug

General settings for C/C++ Debugging.

Opened view default settings

Default variable format: Hexadecimal

Default expression format: Natural

Natural
Hexadecimal
Decimal
Binary

Default register format:

Character encoding:

Breakpoint problems

○ Show in Editor

● Show in Breakpoints View

☑ Show source files in binaries

Maximum stack crawl depth: 100

Regular breakpoints

Install as: Regular

Figure 7-24. Preferences Window-Debug Page

4. Click **Apply**.
5. Click **OK**.
6. Restart CodeWarrior Development Studio.

   When **Variables** view appears in the debug perspective, the variables appear in the Hexadecimal format.

NOTE

This setting is a global setting.

7.4.19 What does the Reset on Connect option do?

The **Reset on Connect** option resets all the cores.

7.4.20 Can I change the Program Counter (pc) value while in the debugger without having to restart the debugger?

Yes. In order to change the Program Counter (pc) without restarting the debugger, follow steps below in the debug perspective:

1. Start a debug session.

   In the editor view, right-click on the source line where you want the pc to move to.

2. A context menu appears.
3. From context menu, select **Move To Line** .

The pc moves to line. The debugger moves the program counter to the location you specified. The editor view shows the new location.

7.4.21 How do I specify the program entry point?

To specify the program entry point, follow steps below in C/C++ perspective:

1. In the CodeWarrior projects window, right-click on project name.

   A context menu appears.

2. From context menu, right-click on project name, select **Debug As > Debug Configurations**.

   The **Debug** window appears.

3. Under **CodeWarrior Download**, select project name.
4. Select **Debugger** tab.

The **Debugger options** page appears.

5. Under Program Execution, select the Program Entry option.
6. Click **Apply**.
7. Click **Debug**.

7.4.22 How can I run a debug session repeatedly?

There are several of ways to run a debug session repeatedly. Below is a list of possible ways to run a debug session repeatedly:

- Select the **Debug** button to invoke the last debug session.

-or-

- Right-click the stack window and select **Relaunch** from context menu.

-or-

- If still in debug mode, click **Terminate and Relaunch** .

7.4.23 Can I direct console output to a file instead of the console window?

Yes, you can direct the console output to a file instead of the console window. To accomplish this, follow steps given below:

1. In the **CodeWarrior Projects** view, right-click on project name.

   A context menu appears.

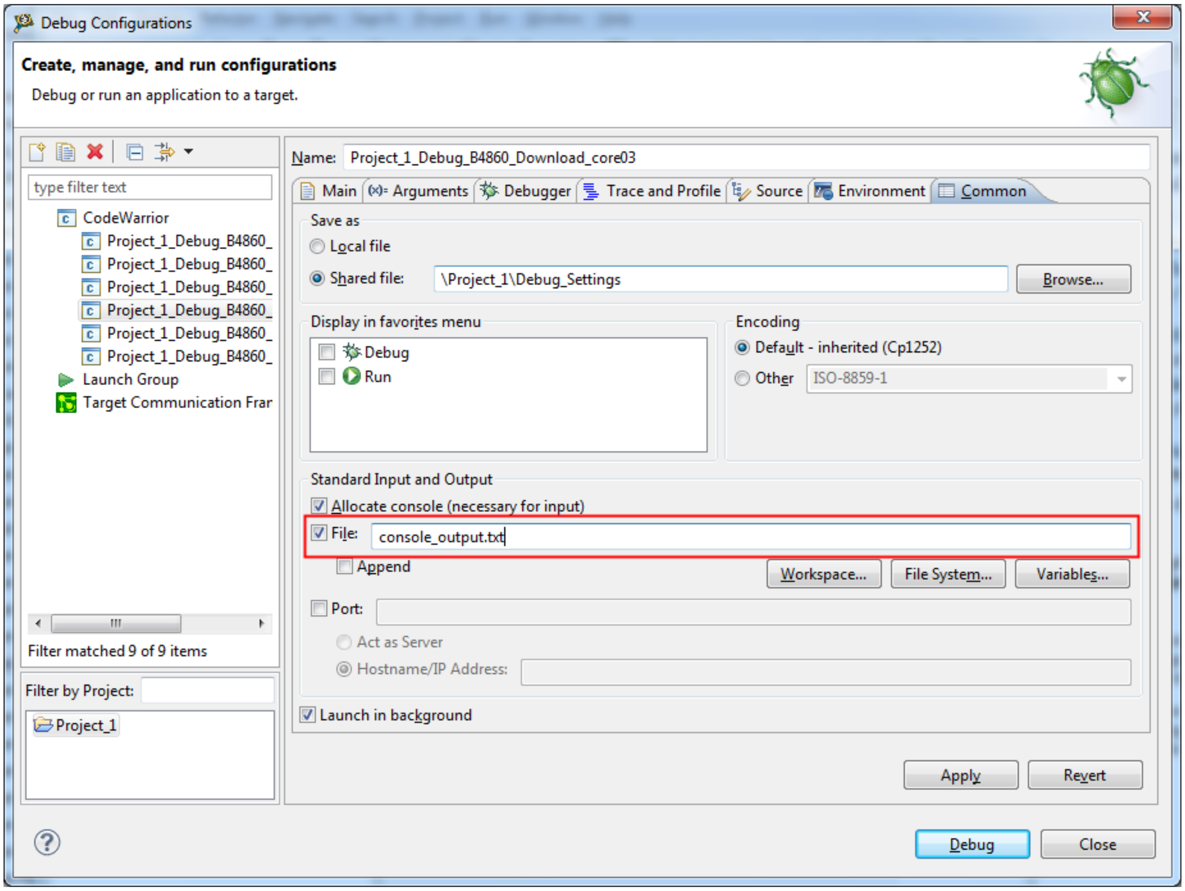
2. From context menu, right-click on project name, select **Debug As > Debug Configurations**.

   The **Debug** window appears.

3. Under **CodeWarrior**, select project name.
4. Select **Common** tab.

   The **Common** page appears (for more information, refer to the image *Debug Configurations-Common Page* listed below).

5. Select **File** and specify the file to which the console output is directed.



Name: Project_1_Debug_B4860_Download_core03

Save as

○ Local file

● Shared file: \Project_1\Debug_Settings

Display in favorites menu

□ Debug

□ Run

Encoding

● Default - inherited (Cp1252)

○ Other ISO-8859-1

Standard Input and Output

☑ Allocate console (necessary for input)

☑ File: console_output.txt

□ Append

□ Port:

○ Act as Server

● Hostname/IP Address:

☑ Launch in background

Figure 7-25. Debug Configurations-Common Page

6. Click **Apply**.
7. Click **Debug**.

The console output will be directed to the specified file.

7.4.24 How do I resolve the following error message that I get when I use Ctrl-Click to open variable declaration?

```
Could not find symbol xx in index.
```

To resolve this error message, follow the steps below:

1. In the CodeWarrior project window, right-click on project name.

   A context menu appears.

2. From context menu, select **Properties** .

The **Properties** window appears.

3. Select **C/C++ General > Indexer** .

   The **Indexer Settings** appear.

4. Check the **Enable project specific settings** checkbox.
5. From the **Select Indexer** drop-down list, select **Full C/C++ Indexer (complete parse)** .
6. Click **Apply** .
7. Click **OK** .

The issue will be resolved. Now, you can use Ctrl+click to open the variable declaration.

Chapter 8
Debugger Shell

In this chapter, you find Frequently Asked Questions (FAQs) related to the Debugger Shell.

- FAQs - Debugger Shell

8.1 FAQs - Debugger Shell

This topic lists the FAQs related to Debugger Shell.

- How can I find the default value of memory width? How can I change the value if I want to?
- Can I change the Program Counter (pc) value?
- Does the config page off option still available in the Eclipse-based CodeWarrior IDE?
- Is there a way to accelerate the execution speed and turn off the printing when I use a lot of mem commands in the debugger shell?

8.1.1 How can I find the default value of memory width? How can I change the value if I want to?

To find the default value for memory width, follow steps below in debug perspective:

1. From Debug Perspective toolbar, select **Window > Show View > Debugger Shell** .

   The **Debugger Shell** window appears docked to the lower right hand side of debug perspective.

2. In debugger shell, type `config`.

The current configurations settings appear. If you want to change the memory width, type `config MemWidth <value>`. To make sure settings were set, type `config`. The current configuration settings appear.

8.1.2 Can I change the Program Counter (pc) value?

Yes. The pc value can be changed through the debugger shell. In order to change the pc value through the debugger shell follow steps below in debug perspective:

1. From the debug perspective, select **Window > Show View > Debugger Shell** .

   The **Debugger Shell** window appears docked to the lower right hand side of debug perspective.

2. In debugger shell, type `reg pc`.

   The present value of pc is displayed.

3. In debugger shell, type `reg pc=0x10000000`.

   This command changes pc value to `0x10000000`.

4. In debugger shell, type `reg pc`.

The current value of pc is displayed; which in this case should be `PC=0x10000000`.

8.1.3 Does the config page off option still available in the Eclipse-based CodeWarrior IDE?

No. This option is not implemented. If a tcl script needs to be run and you want to advance the debugger shell display, select **Enabling Page** icon from the toolbar. By selecting this option, the debugger shell automatically advances the display without having to press the space bar.

8.1.4 Is there a way to accelerate the execution speed and turn off the printing when I use a lot of mem commands in the debugger shell?

Yes. In order to turn of the printing add the `-np` to the `mem` command in your script file. It should look like the following:

```
mem -np
```

Chapter 9
Profiling and Analysis Tool

In this chapter, you find Frequently Asked Questions (FAQs) related to the CodeWarrior Development for StarCore v10.x.

- FAQs - Profiling

9.1 FAQs - Profiling

This topic lists tracing and analysis related FAQs.

- How do I enable trace and profiling?
- How do I view trace and other data?
- How can I save my trace results for later use?
- How can I open the DataFile Editor?
- What are tracepoints?
- What are software and hardware tracepoints?
- What are start and stop tracepoints?
- Which option do I select if I want to add a tracepoint?
- How do I view the tracepoints that I have set?
- What is Performance Analysis perspective used for?
- What are scenarios?
- Is the Performance Analysis tool available in two different versions?
- What are counterpoints?
- How can I configure DPU counters?
- How can I change the value of the OCE registers based on the OCE configurator settings?
- What all registers will be affected by the settings in the Profiler Configuration panel?
- What is the difference between counted events and count cycles using the OCE counters?

- When tracing with VTB (DPU/OCE) is done all writes go directly out on the bus without passing the MMU but does not go directly to the memory. Is privilege signal required while writing on the bus?
- How can you check the profiling results during a debug session?
- What are Register Analyzers?
- What are different types of Register Analyzer?
- What does the number of VLES executed represents in the result?
- What is the use of remote launch in CodeWarrior scripts?
- When using default 8144ADS stationary project all DPU registers display hex number "FF". Also, when I try to change the value of register it does not allow to change. Why I can not change the value of DPU registers?
- Does Trace and Profile work with SmartDSP OS? I tried to use the Trace and Profile tools to see the PC Trace Subroutine and interrupt calls and returns, but I was not able to. Is there a limitation or setting that I need to set?
- What is the speed of SmartDSP HEAT? Can the SmartDSP HEAT provide higher speed than HSST when doing trace offload?
- How to get VLES and Cycle Count Numbers using hardware DPU Counters in CodeWarrior?
- NEW What is Linker Command File and how to enable trace in it?
- NEW How the type of memory and the size of VTB is determined?
- NEW When you should use VTB write modes: Overwrite, One Address, or Trace Event Request?
- NEW How the trace is extracted using JTAG, HSST, HEAT, Ethernet method and how they work ?
- NEW How many hardware and software tracepoints are available?
- NEW How can a tracepoint affect the application?
- NEW Does trace collection affect my application performance and how?
- NEW What does it mean to collect events with counters?
- NEW What is the relationship between DPU and trace?
- NEW Why it is important to import offline trace data?
- NEW What trace configuration method do I use automatic or manual?
- NEW How can you view the latest trace without terminating the debug session?

9.1.1 How do I enable trace and profiling?

To enable trace and profiling:

1. Open the **Debug Configurations** dialog box.
2. Select your project in the tree structure on the left-hand side.

3. Select the **Trace and Profile** tab.
4. Check the **Enable Trace and Profile** checkbox.

This enables the trace and profiling of your project.

9.1.2 How do I view trace and other data?

To view trace and other data:

1. From the IDE menu bar, select **Profiler > All Results** to open the **Trace and Profile Results** view.
2. Expand the data source under the project name.
3. Expand the **Current Results** data file to view the Trace and other data sets for the collected data.
4. Double-click the Trace data set node to view the results in the Trace Data viewer.

This is how you can view trace and other data.

9.1.3 How can I save my trace results for later use?

To save the trace results, right-click on the **Current Results** node, in the Trace and Profile Results viewer and select **Save Current Results** .

9.1.4 How can I open the DataFile Editor?

To open the DataFile Editor, double-click on the data file, for example, **Current Results** .

9.1.5 What are tracepoints?

Tracepoint is a point in the target program where start or stop tracepoints are set at a line of the source code or the assembly code. The start and stop tracepoints are triggers for enabling and disabling the trace output. The advantage of setting start and stop tracepoints is to capture the trace data from the specific part of the program.

9.1.6 What are software and hardware tracepoints?

There are two types of tracepoints, Hardware and Software, that can be set on the source code of the target hardware. To determine whether a tracepoint is software or hardware, it is based on OCE resources availability, trace configuration and target program code. You can set maximum six hardware tracepoints. Software tracepoints use interrupts to enable or disable the trace and you can have any number of software tracepoints.

9.1.7 What are start and stop tracepoints?

A start tracepoint begins the trace collection from the instruction where it is set. A stop tracepoint ends the trace collection at the instruction where it is set.

9.1.8 Which option do I select if I want to add a tracepoint?

If you want to skip a tracepoint, but still want to have it in your list, right-click on the Marker bar where Start or Stop tracepoint is already set and enabled, select the **Disable Tracepoint** option from the tracepoint context menu.

9.1.9 How do I view the tracepoints that I have set?

To view tracepoints:

1. From the IDE menu bar, choose **Window > Show View > Other** .
2. In the Show View dialog, expand the **Analysis** node and choose **Analysispoints**.
3. Click **OK** to close the **Show View** dialog.

The **Analysispoints** view appears displaying the tracepoints that are set on the program code.

9.1.10 What is Performance Analysis perspective used for?

The Performance Analysis perspective provides a user-friendly interface to perform the scenario measurement activities, such as adding a configuration, configuring connection, defining scenarios, and collecting data for B4 and T4 devices.

9.1.11 What are scenarios?

A scenario contains the following:

- Events - Hardware occurrences that can be counted to any event in the processor, including the output of other counters in a feedback loop.
- Counters - Counters that can be connected to any event in the processor, including the output of other counters in a feedback loop.
- Metrics - A mathematical combination of counters (which count events) to provide an additional answer.

9.1.12 Is the Performance Analysis tool available in two different versions?

Yes, the performance analysis tool is available in two different versions: one is used for configuring, collecting, and analyzing events in the QorIQ processors: P4080, P4040, P2040/41, P3041, P5010, P5040, and P5020/21.

And the other one is used for configuring, collecting, and analyzing scenarios in the B4 and T4 devices.

9.1.13 What are counterpoints?

Counterpoints allow you to count events between two points of the executed code. The results are computed based on the collected trace data, the counterpoints are searched in trace by their addresses.

9.1.14 How can I configure DPU counters?

The DPU counters are configured in the Debug Configuration window. To configure counters:

1. Right-click on your project name, context menu appears.

2. From the context menu, select **Debug As > Debug Configurations** . The Debug

Configurations window appears.

3. Select the project name executable from the CodeWarrior Download.

4. Select the **Trace and Profile** tab.

5. Click the **Advanced Settings** button. In the Preferences screen, select **DPU settings** and you should be able to configure DPU counters.

This is how you can configure DPU counters.

9.1.15 How can I change the value of the OCE registers based on the OCE configurator settings?

To set the OCE30 registers to the value specified in the OCE Configuration, you need to click on the **Change** button after you have initialized the EDCAn registers. It may happen that the **Change** button is not visible when you open the OCE Configurator (depending on the layout and screen resolution you are using). To activate the changes:

1. Set up EDCAn triggers in the OCE configurator
2. Maximize the OCE Configurator view
3. Restore OCE configurator to its original size

You should see the updated register values. Note that you can alternatively resize the OCE Configurator window and make it a little bit higher.

9.1.16 What all registers will be affected by the settings in the Profiler Configuration panel?

The settings in the profiler configuration panel will change the register settings of OCE, DPU, and MMU (only the M_CR Register).

9.1.17 What is the difference between counted events and count cycles using the OCE counters?

Counted events means profiling the cycle information either from the OCE counters or from the DPU counters. When the profiler configuration is set to Count Cycles using the OCE, the cycle count information will come only from the OCE counters.

9.1.18 When tracing with VTB (DPU/OCE) is done all writes go directly out on the bus without passing the MMU but does not go directly to the memory. Is privilege signal required while writing on the bus?

The privileged signal is propagated on to the bus as there are parts of the SOC which can only be written to in privileged mode (primarily CCSR). The DPU bypassed the MMU. Although there is no use for it as the main memories used for trace are not protected in general.

9.1.19 How can you check the profiling results during a debug session?

Yes, the trace data is collected only after the debug session has terminated. This means that you may have to wait for a long period of time depending upon the size and complexity of the application.

CodeWarrior for StarCore offers the ability to check the DPU (Debugging and Profiling Unit) results at run-time in the **DPU Analyzer** view. To open this view, select **Window > Show view > Other > Analysis > DPU Analyzer** . The DPU Analyzer shows the data when the program is halted. This can be used to trace the performance of the source code.

9.1.20 What are Register Analyzers?

Register Analyzer analyzes the state of the registers on the target. It helps you to see if there are errors in the register configuration, if there are disabled hardware units or to view informative messages. This feature is supported on both 8144ADS and 8156ADS targets. The two working modes of register analyzer are online and offline.

9.1.21 What are different types of Register Analyzer?

The two types of Register Analyzer are online and offline. The registers on the target can be analyzed online in an active debug session. The debug session is identified by the name of the launch configuration. The registers can be analyzed offline by loading a file that contains the register values. The registers can be analyzed based on a file that contains the information of all the registers.

9.1.22 What does the number of VLES executed represents in the result?

The number of VLES executed depends on various profiler configurations:

1. In case of PACC simulator, the value represents the number of VLES that are executed.
2. In case of profiling on hardware using OCE and tracing all instruction set, the value represents the number of VLES that are executed.
3. In case of profiling on hardware using OCE and tracing COF, the value represent the number of COF and not the number of VLES that are executed.
4. In case of profiling on hardware using DPU counters and tracing call/return, the value represent the number of call/return and not the number of VLES that are executed.

This is how you can view VLES in the result.

9.1.23 What is the use of remote launch in CodeWarrior scripts?

You can execute your own scripts in CodeWarrior IDE. The scripts are launched in remote launch and debugger shell. The Python scripts are executed in remote launch and TCL scripts are executed in debugger shell. The remote launch feature of CodeWarrior allows launch configurations to be executed remotely. You can use the debugger shell to execute commands in a command-line environment.

9.1.24 When using default 8144ADS stationary project all DPU registers display hex number "FF". Also, when I try to change the value of register it does not allow to change. Why I can not change the value of DPU registers?

If you are using a stationary project, you can not write to the DPU registers. The VTB and the OCE registers must be set up appropriately to modify the DPU registers.

9.1.25 Does Trace and Profile work with SmartDSP OS? I tried to use the Trace and Profile tools to see the PC Trace Subroutine and interrupt calls and returns, but I was not able to. Is there a limitation or setting that I need to set?

The SmartDSP OS Linker Command File does not include a definition for **_VTB_start** and **_VTB_end** . When these settings are not set, you cannot use the **Compute VTB Location Automatically** option. You need to define the **_VTB_start** and **_VTB_End** addresses manually.

9.1.26 What is the speed of SmartDSP HEAT? Can the SmartDSP HEAT provide higher speed than HSST when doing trace offload?

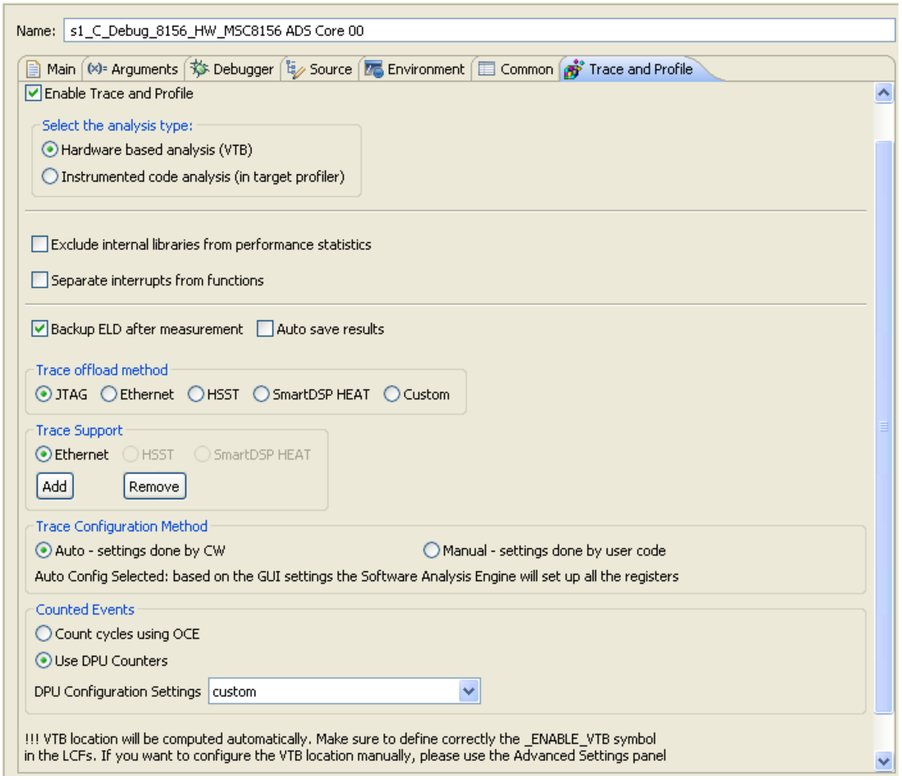
HSST uses the JTAG connection which is significantly slower than HEAT that uses Ethernet connection. The SmartDSP HEAT blocks the program while preparing the packet. During the I/O, it gives packet to the QUICC Engine for transmission with core intervention. The speed of transmission is 11.6Mbps/core on the MSC8156ADS. This is the limitation of the server capabilities and not the DSP offloading capabilities.

9.1.27 How to get VLES and Cycle Count Numbers using hardware DPU Counters in CodeWarrior?

To view VLES and Cycle count numbers in the result:

1. In the debug launch configuration, check the **Enable Trace and Profile** checkbox to enable the tracing and profiling.
2. In the Counted Events panel, select **Use DPU Counters** radio button and set the value of **DPU Configuration Settings** to **Custom** and then click **Advanced Settings** .

   The Preferences screen appears.



Name: s1_C_Debug_8156_HW_MSC8156 ADS Core 00

☑ Enable Trace and Profile

Select the analysis type:
● Hardware based analysis (VTB)
○ Instrumented code analysis (in target profiler)

□ Exclude internal libraries from performance statistics

□ Separate interrupts from functions

☑ Backup ELD after measurement □ Auto save results

Trace offload method
● JTAG ○ Ethernet ○ HSST ○ SmartDSP HEAT ○ Custom

Trace Support
● Ethernet ○ HSST ○ SmartDSP HEAT

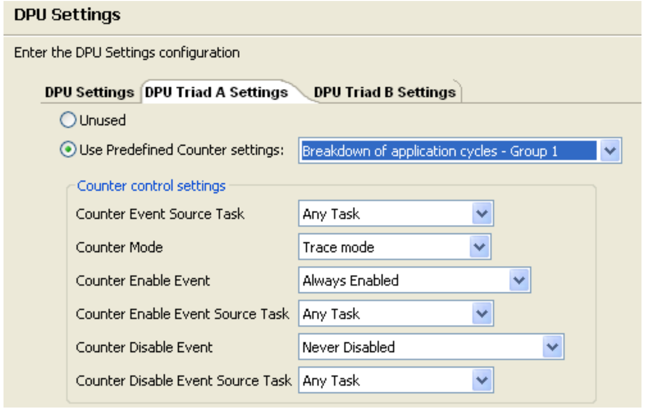
Trace Configuration Method
● Auto - settings done by CW ○ Manual - settings done by user code
Auto Config Selected: based on the GUI settings the Software Analysis Engine will set up all the registers

Counted Events
○ Count cycles using OCE
● Use DPU Counters
DPU Configuration Settings custom

!!! VTB location will be computed automatically. Make sure to define correctly the _ENABLE_VTB symbol
in the LCFs. If you want to configure the VTB location manually, please use the Advanced Settings panel

3. In the Preferences screen, select DPU settings from the list and then select **DPU Triad A Settings** tab. Select **Use Predefined Counter settings** radio button and set

the value **Breakdown of application cycles - Group 1** from the pull-down menu.



DPU Settings

Enter the DPU Settings configuration

○ Unused

● Use Predefined Counter settings: Breakdown of application cycles - Group 1

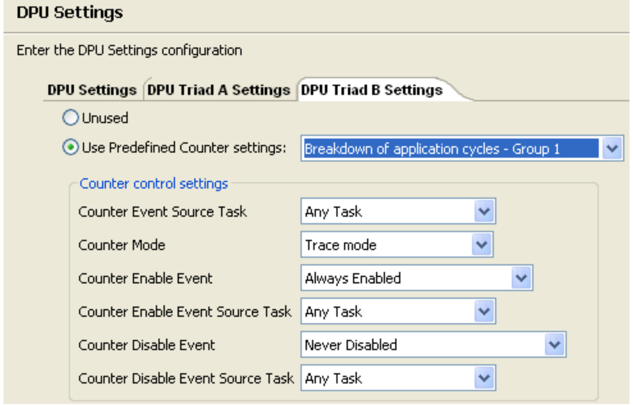
Counter control settings

Counter Event Source Task	Any Task
Counter Mode	Trace mode
Counter Enable Event	Always Enabled
Counter Enable Event Source Task	Any Task
Counter Disable Event	Never Disabled
Counter Disable Event Source Task	Any Task

Counter A0 of this group will count the total number of cycles spent in executing holds and bubbles. Counter A1 of this group counts the time spent executing no holds and no bubbles. This number is very important during the source code optimization.

4. In the DPU Triad B Settings tab, select **Use Predefined Counter settings** radio button and set the value **BTB Characterization Group 1** from the pull-down menu.

   Counter B0 of this group counts the total number of execution set.



DPU Settings

Enter the DPU Settings configuration

○ Unused

● Use Predefined Counter settings: Breakdown of application cycles - Group 1

Counter control settings

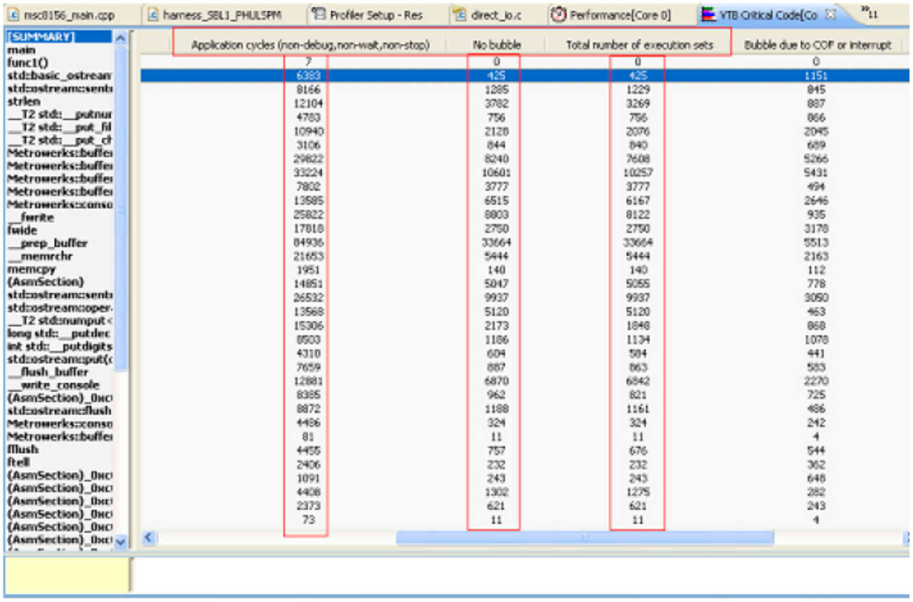
Counter Event Source Task	Any Task
Counter Mode	Trace mode
Counter Enable Event	Always Enabled
Counter Enable Event Source Task	Any Task
Counter Disable Event	Never Disabled
Counter Disable Event Source Task	Any Task

5. Click **OK** to close the Preferences window.
6. Click **Apply** and **Debug** to debug an application.
7. Click **Resume** to start the measurement. When you terminate or suspend the debug session, the tool generates the profiler reports.

If you are interested in the exclusive numbers then you need to look at the critical code data. In the critical code data, you should be able to view:

- Application cycle column shows the exclusive total number of cycles count per function.

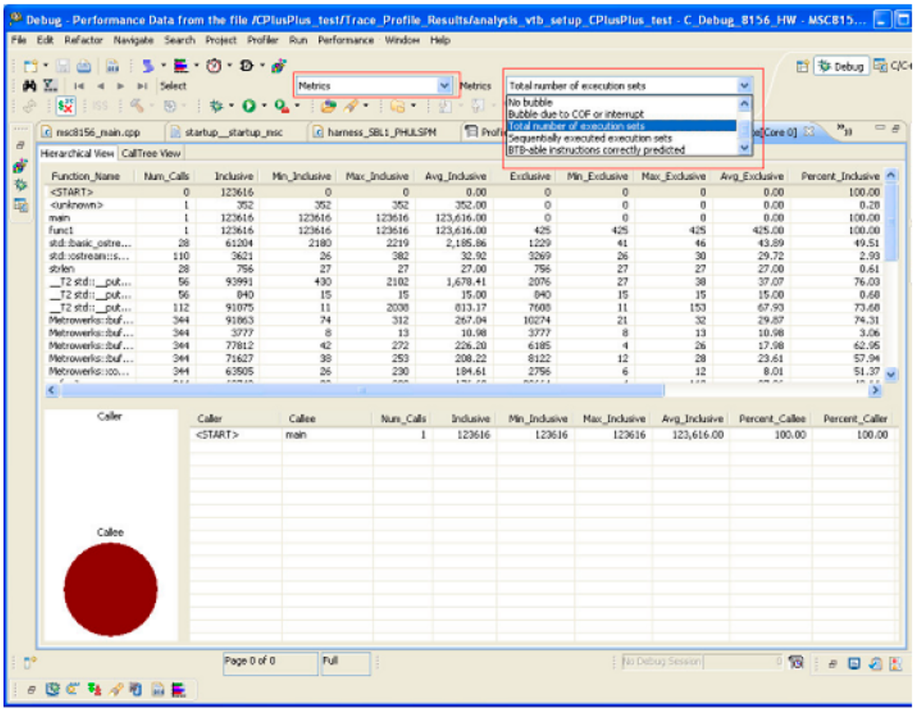
- No Bubble column shows exclusive time spent executing function instructions. This number must be equal or more than "Total Number of execution set" and equal or less than "Application cycles".
- Total number of execution set shows the total number of instruction set (VLES) per function. You can export the data to CSV format.



[SUMMARY]
main
func1()
std::basic_ostrean
std::ostream::sent
strlen
__T2 std::__putnur
__T2 std::__put_fil
__T2 std::__put_ch
Metrowerks::buffer
Metrowerks::buffer
Metrowerks::buffer
Metrowerks::buffer
Metrowerks::conso
__fwrite
fwide
__prep_buffer
__memrchr
memcpy
(AsmSection)
std::ostream::sent
std::ostream::oper.
__T2 std::numput<
long std::__putdec
int std::__putdigits
std::ostream::put(c
__flush_buffer
__write_console
(AsmSection)_0xc
std::ostream::flush
Metrowerks::conso
Metrowerks::buffer
fflush
ftell
(AsmSection)_0xc
(AsmSection)_0xc
(AsmSection)_0xc
(AsmSection)_0xc
(AsmSection)_0xc
(AsmSection)_0xc

Application cycles (non-debug,non-wait,non-stop)	No bubble	Total number of execution sets	Bubble due to COF or interrupt
7	0	0	0
6383	425	425	1151
8166	1285	1229	845
12104	3782	3269	887
4783	756	756	866
10940	2128	2076	2045
3106	844	840	689
29822	8240	7608	5266
33224	10601	10257	5431
7802	3777	3777	494
13585	6515	6167	2646
25822	8803	8122	935
17818	2750	2750	3178
84936	33664	33664	5513
21653	5444	5444	2163
1951	140	140	112
14851	5047	5055	778
26532	9937	9937	3050
13568	5120	5120	463
15306	2173	1848	868
8503	1186	1134	1078
4310	604	584	441
7659	887	863	583
12881	6870	6842	2270
8385	962	821	725
8872	1188	1161	486
4486	324	324	242
81	11	11	4
4455	757	676	544
2406	232	232	362
1091	243	243	648
4408	1302	1275	282
2373	621	621	243
73	11	11	4

If you need more detailed information such as inclusive, minimum, and maximum then you need to look at the performance data. Based on the selected metric, the tool outputs the data. For example, if you want to know the minimum and maximum number of execution set for a given function, you need to set **Metrics** to **Total number of execution set** and then browse the data to locate the desired function. You can also export the data to CSV format.



Debug - Performance Data from the file /CPlusPlus_test/Trace_Profile_Results/analysis_vtb_setup_CPlusPlus_test - C_Debug_8156_HW - MSC815...
File Edit Refactor Navigate Search Project Profiler Run Performance Window Help
Select
Metrics
Metrics
Debug
C/C
msc8156_main.cpp
startup__startup_msc
harness_SBL1_PHULSPM
Profi
e[Core 0]
Hierarchical View
CallTree View
Caller
Callee
Page 0 of 0
Full
No Debug Session

Total number of execution sets

No bubble
Bubble due to COF or interrupt
Total number of execution sets
Sequentially executed execution sets
BTB-able instructions correctly predicted

Function_Name	Num_Calls	Inclusive	Min_Inclusive	Max_Inclusive	Avg_Inclusive	Exclusive	Min_Exclusive	Max_Exclusive	Avg_Exclusive	Percent_Inclusive
<START>	0	123616	0	0	0.00	0	0	0	0.00	100.00
<unknown>	1	352	352	352	352.00	0	0	0	0.00	0.28
main	1	123616	123616	123616	123,616.00	0	0	0	0.00	100.00
funct1	1	123616	123616	123616	123,616.00	425	425	425	425.00	100.00
std::basic_ostre...	28	61204	2180	2219	2,185.86	1229	41	46	43.89	49.51
std::ostream::s...	110	3621	26	382	32.92	3269	26	30	29.72	2.93
strlen	28	756	27	27	27.00	756	27	27	27.00	0.61
__T2 std::__put...	56	93991	430	2102	1,678.41	2076	27	38	37.07	76.03
__T2 std::__put...	56	840	15	15	15.00	840	15	15	15.00	0.68
__T2 std::__put...	112	91075	11	2038	813.17	7608	11	153	67.93	73.68
Metrowerks::buf...	344	91863	74	312	267.04	10274	21	32	29.87	74.31
Metrowerks::buf...	344	3777	8	13	10.98	3777	8	13	10.98	3.06
Metrowerks::buf...	344	77812	42	272	226.20	6185	4	26	17.98	62.95
Metrowerks::buf...	344	71627	38	253	208.22	8122	12	28	23.61	57.94
Metrowerks::co...	344	63505	26	230	184.61	2756	6	12	8.01	51.37

Caller	Callee	Num_Calls	Inclusive	Min_Inclusive	Max_Inclusive	Avg_Inclusive	Percent_Callee	Percent_Caller
<START>	main	1	123616	123616	123616	123,616.00	100.00	100.00

9.1.28 What is Linker Command File and how to enable trace in it?

A Linker Command File (LCF) contains all the memory configurations and settings on which the linker relies to assemble the executable file (*.eld or *.elb). Using a linker command file (LCF), you can instruct the linker to store portions of your executable in different areas of memory. The linker relocates and binds symbols to addresses according to the directives in your LCF.

NOTE

The following scenario assumes that Enable Trace and Profile option is not selected in the Software Analysis Trace and Profile page while creating the project. If this option is already selected, you can still change the default settings from LCF.

To enable trace in LCF:

1. Expand the LCF folder of your project in the **CodeWarrior Projects** view.
2. Open the `mmu_attr.l3k` file.
3. Scroll down to the end of file, and set the value of `_ENABLE_VTB` as follows: `_ENABLE_VTB = 3;`

   This will enable trace and profile in LCF and will reserve VTB in DDR memory.

NOTE

You can set the value of `_ENABLE_VTB` as 1, 2, or 3 depending on whether you want to reserve VTB in M1, M2, or DDR memory respectively. Setting any other value will disable the trace and profile.

4. Save the file.

This is how you can enable trace in the Linker Command File.

NOTE

The size of the reserved VTB can be increased or decreased in the `common.l3k` file. _M2_VTB_size= 0x1000; //4K for each core _M3_VTB_size= 0x8000; //32K for each core _DDR_VTB_size=0x1000; //4k for each core

9.1.29 How the type of memory and the size of VTB is determined?

You can use MemoryMapViewer to identify free memory zones. To view a contiguous memory area that is not reserved for another purpose, click **Window > Show View > Other > MemoryMapViewer** . You can set the size of the memory depending on the size of the M1, M2, or DDR memory and VTB start address in the `common.l3k` file.

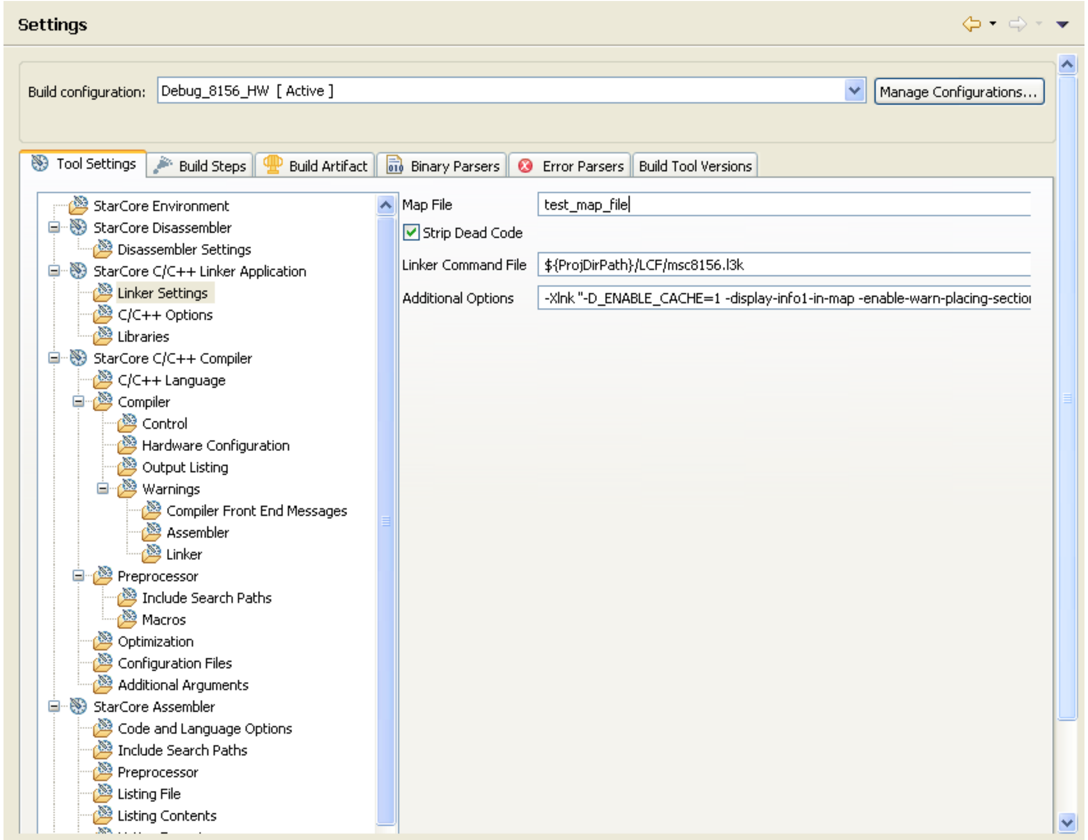
NOTE

The `common.l3k` file available in the LCF folder of your project.

To check the available free memory block:

1. Open the Properties page of your project.
2. Expand **C/C++ Build > Settings** .
3. In the Tools Settings tab, expand **StarCore C/C++ Linker Application > Linker Settings** .

4. In the **Map File** text box on the right, type a name for the map file.



Build configuration: Debug_8156_HW [Active]

Map File: test_map_file

☑ Strip Dead Code

Linker Command File: ${ProjDirPath}/LCF/msc8156.l3k

Additional Options: -Xlnk "-D_ENABLE_CACHE=1 -display-info1-in-map -enable-warn-placing-sectio

5. Click **Apply** .
6. Close the Properties page.
7. Build your project.
8. The map file appears in the C_Debug_8156_HW folder of your project.
9. Open the map file.
10. Scroll down or search until you find the Memory Map. This is the sample:

Memory Map:

```
|0x30000000..0x30007fff|  memory reserved (physical memory) (32768 bytes)

rw- |0x30008000..0x30019fff|  memory m2_private_data_c_wb (73728 bytes)

|0x3001a000..0x30037fff|  memory free space (122880 bytes)

rw- |0x30038000..0x3003ffff|  memory data_boot_c (32768 bytes)

|0x40000000..0x5fffffff|  memory free space (536870912 bytes)

|0x80000000..0x9fffffff|  memory free space (536870912 bytes)

r-x |0xc0000000..0xc00093ff|  memory m3_shared_text_c (37888 bytes)
```

```
rw-  |0xc0009400..0xc000cfff|  memory m3_shared_data_c_wb (15360 bytes)

r-x |0xc000d000..0xc000f3ff|  memory m3_shared_textboot_c (9216 bytes)

|0xc000f400..0xc00fcfff|  memory free space (973824 bytes)

rw-  |0xc00fd000..0xc00fd6ff|  memory m3_private_data_c_wb (1792 bytes)

|0xc00fd700..0xc00fffff|  memory free space (10496 bytes)

|0xc0100000..0xc0107fff|  memory free space (32768 bytes)
```

Core Architecture Memory Map:

```
|0xc0000000..0xc00fffff|  memory M3  (1048576 bytes)

|0xc0100000..0xc0107fff|  memory M3_extension  (32768 bytes)

|0x40000000..0x5fffffff|  memory DDR  (536870912 bytes)

|0x80000000..0x9fffffff|  memory DDR2  (536870912 bytes)

|0x30000000..0x3003ffff|  memory M2  (262144 bytes)
```

9.1.30 When you should use VTB write modes: Overwrite, One Address, or Trace Event Request?

The order in which VTB write modes are arranged below is according to their usefulness.

Trace Event Request mode - In this mode, when the trace buffer is full, a debug request or an interrupt is generated, which allows the tools to collect the trace and prevent data from getting overwritten. This mode specifies the address within the VTB range where an interrupt or debug request will be generated when Write Mode is set to Trace Event Request. The address at which the interrupt will occur, the data collected till that address will be transferred to the computer.

After the trace is collected, the application is automatically resumed making this process transparent to the user.

Overwrite mode - This mode is useful for investigating crashes, as it captures the information about the code that was executed just before the crash. In this mode, the last state of application is retrieved, it means the last VTB (Virtual Trace Buffer) filled by the application. Because of small size of VTB, data is overwritten every time when it gets full. You can collect and view the trace when the debug session is suspended by the user or due to a breakpoint or where the bug appears.

One Address mode - This mode is used when you select Trace Configuration Method as Manual settings done by user code. To control which data is downloaded to the computer, this option is used. It can be used if the trace information is written to a peripheral device.

9.1.31 How the trace is extracted using JTAG, HSST, HEAT, Ethernet method and how they work ?

All these methods use an interrupt instead of a debug instruction. When the trace buffer is full, which means core will not stop when the trace is being collected.

It means the core will not stop while the trace is being collected. These download methods are differentiated based on how the trace is downloaded from the target.

High Speed Simultaneous Transfer (HSST) is a method to facilitate data transfer between an application that runs on a target (for example, MSC8156 target) and a host side CodeWarrior application. HSST download method uses the same physical communication layer as JTAG download method, but the speed of the HSST method is 12.82 times faster than the JTAG download method. While the JTAG download method transfers data by halting the target, exchanging the data followed by resuming the target, the HSST download method works faster because the transfer between target and host computer occurs without stopping the target. JTAG is very slow and does not require additional hardware support.

HEAT requires an ethernet physical connection, an operating system that can manage file I/O (Freescale SDOS) and SmartDSP OS server on the host, but it provides the highest speed. It is used for streaming large data to and from PC. The transfer of data that takes hours on JTAG takes seconds on HEAT.

Ethernet uses the physical support and the NetIO software libraries. It is as fast as HEAT and unlike HEAT, it does not require any specific operating system.

9.1.32 How many hardware and software tracepoints are available?

Hardware tracepoints use OCE channels, which are non-intrusive but are in limited number (only 6 channels) and are also included in the resources shared with the debugger. Software tracepoints use interrupts, which means they might affect execution speed and you may have to modify the application in order to use them. The advantage is that you can use any number of software tracepoints.

Both hardware and software tracepoints can be set on the source code of the target hardware. Hardware tracepoints are non-intrusive and does not affect the execution speed of an application. There can be maximum of six hardware tracepoints because there are only six OCE channels present on the chip. If you have set the start and stop hardware tracepoint, debug the application and open the OCE Configurator view. In the OCE Configurator view, there are six EDCA0-5 channels showing you whether the EDCA channel is enabled or disabled and also the addresses. This address is passed through the data bus to the processor. EDCA0 channel will check this address in data bus and if start tracepoint is set on that address, tracing will start. EDCA1 channel will check for the second address in data bus and encounters stop tracepoint at that address, tracing will stop.

Software tracepoints use interrupts to enable or disable the trace. You can have any number of start and stop tracepoints.

A demo on how software tracepoints works can be found at: <InstallDir>\StarCore_Support\SoftwareAnalysis\swandemo_81xxads_tracepoints

9.1.33 How can a tracepoint affect the application?

Hardware tracepoints does not affect the application and they are non-intrusive. Software tracepoints can affect the execution speed of an application and they are intrusive. CPU allocates resources to generate and execute the software interrupts.

9.1.34 Does trace collection affect my application performance and how?

To minimize the effects of tracing on application performance, consider the following options:

- Count only the events of interest. Multiple debug sessions can be carried out automatically to obtain trace data for other events. Tracing the events individually

means that Trace Write Buffer (TWB) fills slowly and less writes to VTB are required.
- Use VTB of the appropriate size for the download method.
- Recall that the core stalls while the trace is downloaded. A larger-sized VTB generates less debug interrupts when transferring its content to the host PC. Therefore, the goal is to reduce the number of times the VTB must be emptied onto the host PC. In an ideal scenario, the VTB should have enough capacity to save all the events from a debug session, so that a VTB download occurs only when the debug session terminates. The goal is to minimize the number of downloads required to transfer the trace data.
- Use M2 or M3 memory when available.
- Use start or stop tracepoints to take the measurements only of the critical code data. This results in less data that must be saved in the VTB.

9.1.35 What does it mean to collect events with counters?

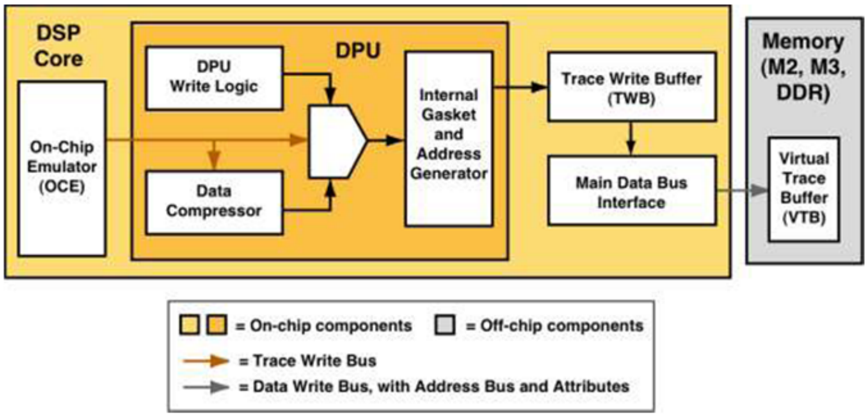
The specialized unit called DPU has 6 counters that are used to measure different events, such as execution cycles, stalls, cache events, holds due to VTB writes, bus load, BTB characterization and others. All the events that can be measured are available in the Software Analysis GUI configurator.

When such event occurs, an interrupt occurs inside the DPU and the actual value of the counter associated with that event is saved. This snapshot of the counter is then saved into the VTB along with the address that has generated the event (e.g. calling a function cause a jump to a particular address. Returning from the function is causing another jump to another address. The two snapshots of the counter associated with the jumps helps to compute the time needed for the execution of a function).

This mechanism that consists of associating an address with a counter value is the basis of collecting the event.

9.1.36 What is the relationship between DPU and trace?

At hardware level, the unit responsible for data or code events collection is the DPU. The DPU writes the trace information that is generated either by itself or the On Chip Emulator (OCE), to a Virtual Trace Buffer (VTB) as shown in figure below. The VTB can reside in system memory or, in internal or external memory.



The DPU write accesses are buffered in the Trace Write Buffer (TWB), and then written through the main data bus interface in bursts. The lowest priority bus requests are used when writing into the first unwritten address of the VTB. When TWB becomes full, it raises the priority of its bus request. At the first cycle after this priority change, the DPU generates a core stall request. When the writes into the VTB reaches the VTB Trace Event Request Address, it triggers a debug interrupt and the data is downloaded into the host PC.

9.1.37 Why it is important to import offline trace data?

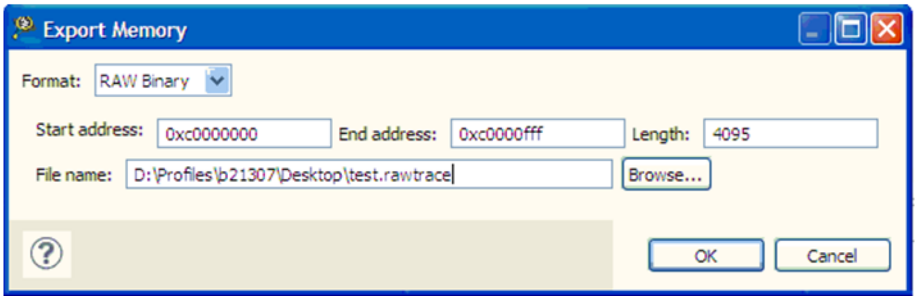
The offline trace data is used to diagnose the issue or reason for crash in the application. You do not require large memory for collecting the trace data since VTB acts as a circular buffer that is overwritten until the application crashes. The buffer will contain the last data when the application was crashed. This data will help you to diagnose the reason for crashing of an application. When an error occurs, the target is halted and the data from the VTB is saved from the target into a file. If CodeWarrior debugger connection is not available, an alternate way to save the data should be used. For example, the Ethernet connection can also be used to dump the data to a PC host.

After the file has been saved onto the host, the data can be examined to identify the cause for failure.

To save the VTB data in .rawtrace file:

1. Select **Windows > Show View > Other > Debug > Memory** to open the **Memory** view.
2. Click the *Export* icon present on the toolbar of this view.

3. **Export Memory** screen appears.



Export Memory

Format: RAW Binary

Start address: 0xc0000000 End address: 0xc0000fff Length: 4095

File name: D:\Profiles\b21307\Desktop\test.rawtrace

4. Enter the VTB Start and End addresses and also the format in which you want the data to be saved.
5. Browse the location you want the data to be saved and click **OK**.

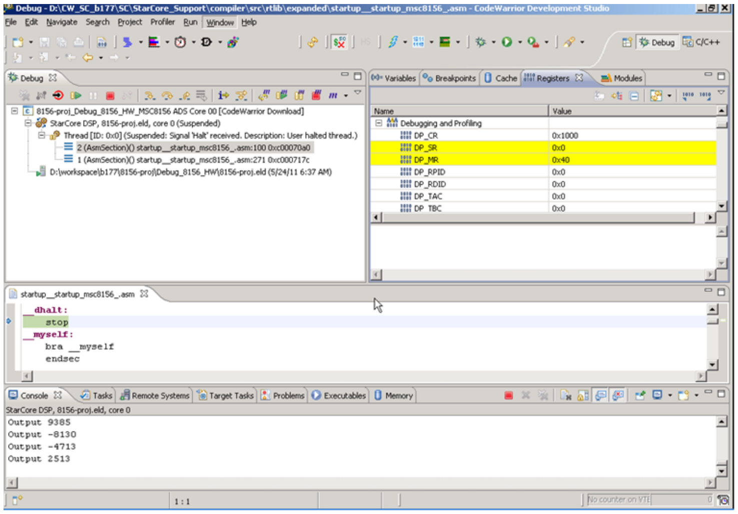
This is how you can import offline trace data.

9.1.38 What trace configuration method do I use automatic or manual?

It depends on your needs that how do you want to configure the tracing using CodeWarrior GUI or from the code of your application.

This setting affects only the way the configuration is set and applied to the target. For both options, you will have tracing and profiling results available.

If you select Auto - settings done by CW, all the settings made in the GUI configurator are applied to the trace hardware modules by CodeWarrior when the execution of the application is started. All DPU registers settings are done automatically by CodeWarrior as shown in figure below.



Debug - D:\CW_SC_b177\SC\StarCore_Support\compiler\src\rtlib\expanded\startup__startup_msc8156_.asm - CodeWarrior Development Studio

8156-proj_Debug_8156_HW_MSC8156 ADS Core 00 [CodeWarrior Download]
StarCore DSP, 8156-proj.eld, core 0 (Suspended)
Thread [ID: 0x0] (Suspended: Signal 'Halt' received. Description: User halted thread.)
2 (AsmSection)() startup__startup_msc8156_.asm:100 0xc00070a0
1 (AsmSection)() startup__startup_msc8156_.asm:271 0xc000717c
D:\workspace\b177\8156-proj\Debug_8156_HW\8156-proj.eld (5/24/11 6:37 AM)

Name	Value
Debugging and Profiling	
DP_CR	0x1000
DP_SR	0x0
DP_MR	0x40
DP_RPID	0x0
DP_RDID	0x0
DP_TAC	0x0
DP_TBC	0x0

startup__startup_msc8156_.asm

```
__dhalt:
    stop
__myself:
    bra __myself
    endsec
```

StarCore DSP, 8156-proj.eld, core 0

```
Output 9385
Output -8130
Output -4713
Output 2513
```

In case of Manual - settings done by user code, you need to ensure that the application sets the DPU registers correctly from the code before capturing any trace. The register values must be set manually from source code before debugging your application.

9.1.39 How can you view the latest trace without terminating the debug session?

In case of large applications, you can see the results for a particular function without terminating the debug session. This helps you to view the results without waiting for long time.

Index

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

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

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

Document Number CWSCFAQUG
Revision 10.9.0, 06/2015