



CEVA-Toolbox[™]



CEVA-XM4[™] Profiler Reference Guide

Rev 15.1.0

August 2015

Documentation Control

History Table:

Version	Date	Description	Remarks
10	1/12/2012	Initial version	
10.1	7/11/2013	Add Linux Support	
CEVA-XC 11.0 Beta	10/02/2015	Updated the invocation chapter to match the eclipse IDE. Remove CLI references	
CEVA-XM4 15.1.0	09/08/2015	Update for XM4	Added chapter for XM4 profiler counters

Disclaimer and Proprietary Information Notice

The information contained in this document is subject to change without notice and does not represent a commitment on any part of CEVA®, Inc. CEVA®, Inc. and its subsidiaries make no warranty of any kind with regard to this material, including, but not limited to implied warranties of merchantability and fitness for a particular purpose whether arising out of law, custom, conduct or otherwise.

While the information contained herein is assumed to be accurate, CEVA®, Inc. assumes no responsibility for any errors or omissions contained herein, and assumes no liability for special, direct, indirect or consequential damage, losses, costs, charges, claims, demands, fees or expenses, of any nature or kind, which are incurred in connection with the furnishing, performance or use of this material.

This document contains proprietary information, which is protected by U.S. and international copyright laws. All rights reserved. No part of this document may be reproduced, photocopied, or translated into another language without the prior written consent of CEVA®, Inc.

CEVA®, CEVA-XC™, CEVA-XC321™, CEVA-XC323™, CEVA-Xtend™, CEVA-XC4000™, CEVA-XC4100™, CEVA-XC4200™, CEVA-XC4210™, CEVA-XC4400™, CEVA-XC4410™, CEVA-XC4500™, CEVA-TeakLite™, CEVA-TeakLite-II™, CEVA-TeakLite-III™, CEVA-TL3210™, CEVA-TL3211™, CEVA-TeakLite-4™, CEVA-TL410™, CEVA-TL411™, CEVA-TL420™, CEVA-TL421™, CEVA-Quark™, CEVA-Teak™, CEVA-X™, CEVA-X1620™, CEVA-X1622™, CEVA-X1641™, CEVA-X1643™, Xpert-TeakLite-II™, Xpert-Teak™, CEVA-XS1100A™, CEVA-XS1200™, CEVA-XS1200A™, CEVA-TLS100™, Mobile-Media™, CEVA-MM1000™, CEVA-MM2000™, CEVA-SP™, CEVA-VP™, CEVA-MM3000™, CEVA-MM3100™, CEVA-MM3101™, CEVA-XM™, CEVA-XM4™, CEVA-X2™ CEVA-Audio™, CEVA-HD-Audio™, CEVA-VoP™, CEVA-Bluetooth™, CEVA-SATA™, CEVA-SAS™, CEVA-Toolbox™, SmartNcode™ are trademarks of CEVA, Inc.

All other product names are trademarks or registered trademarks of their respective owners

Support

CEVA® makes great efforts to provide a user-friendly software and hardware development environment. Along with this, CEVA® provides comprehensive documentation, enabling users to learn and develop applications on their own. Due to the complexities involved in the development of DSP applications that may be beyond the scope of the documentation, an on-line Technical Support Service (support@ceva-dsp.com) has been established. This service includes useful tips and provides fast and efficient help, assisting users to quickly resolve development problems.

How to Get Technical Support:

FAQs: Visit our web site <http://www.ceva-dsp.com> or your company's protected page on the CEVA® website for the latest answers to frequently asked questions.

Application Notes: Visit our website <http://www.ceva-dsp.com> or your company's protected page on the CEVA® website for the latest application notes.

Email: Use CEVA's central support email address support@ceva-dsp.com. Your email will be forwarded automatically to the relevant support engineers and tools developers who will provide you with the most professional support in order to help you resolve any problem.

License Keys: Please refer any license key requests or problems to sdtkeys@ceva-dsp.com. Refer to the *SDT Installation & Licensing Scheme Guide* for SDT license keys installation information.

Email: support@ceva-dsp.com

Visit us at: www.ceva-dsp.com

List of Sales and Support Centers

Israel	USA	Ireland	Sweden
<p>2 Maskit Street P.O.Box 2068 Herzeliya 46120 Israel</p> <p>Tel: +972 9 961 3700 Fax: +972 9 961 3800</p>	<p>1943 Landings Drive Mountain View, CA 94043 USA</p> <p>Tel: +1-650-417-7923 Fax: +1-650-417-7924</p>	<p>Segrave House 19/20 Earlsfort Terrace 3rd Floor Dublin 2 Ireland</p> <p>Tel: +353 1 237 3900 Fax: +353 1 237 3923</p>	<p>Klarabergsviadukten 70 Box 70396 107 24 Stockholm, Sweden</p> <p>Tel: +46(0)8 506 362 24 Fax: +46(0)8 506 362 20</p>
China (Shanghai)	China (Beijing)	China Shenzhen	Hong Kong
<p>Room 517, No. 1440 Yan An Road (C) Shanghai 200040 China</p> <p>Tel: +86-21-22236789 Fax: +86 21 22236800</p>	<p>Rm 503, Tower C, Raycom InfoTech Park No.2, Kexueyuan South Road, Haidian District Beijing 100190, China</p> <p>Tel: +86-10 5982 2285 Fax: +86-10 5982 2284</p>	<p>2702-09 Block C Tiley Central Plaza II Wenxin 4th Road, Nanshan District Shenzhen 518054</p> <p>Tel: +86-755-86595012</p>	<p>Level 43, AIA Tower, 183 Electric Road, North Point Hong Kong</p> <p>Tel: +852-39751264 :</p>
South Korea	Taiwan	Japan	France
<p>#478, Hyundai Arion, 147, Gumgok-Dong, Bundang-Gu, Sungnam-Si, Kyunggi-Do, 463-853, Korea</p> <p>Tel: +82-31-704-4471 Fax: +82-31-704-4479</p>	<p>Room 621 No.1, Industry E, 2nd Rd Hsinchu, Science Park Hsinchu 300 Taiwan R.O.C</p> <p>Tel: +886 3 5798750 Fax: +886 3 5798750</p>	<p>3014 Shinoharacho Kasho Bldg. 4/F Kohoku-ku Yokohama, Kanagawa 222-0026 Japan</p> <p>Tel: +81 045-430-3901 Fax: +81 045-430-3904</p>	<p>RivieraWaves S.A.S 400, avenue Roumanille Les Bureaux Green Side 5, Bât 6 06410 Biot - Sophia Antipolis, France</p> <p>Tel: +33 4 83 76 06 00 Fax: +33 4 83 76 06 01</p>

Table of Contents

1	PROFILER OPERATION.....	1-1
1.1	Invocation (Windows).....	1-1
1.2	Menu	1-2
1.3	Invocation (Linux)	1-2
1.3.1	Create and Using Profiler Setup File	1-2
1.3.2	View Profiler Results	1-3
1.4	Setup.....	1-3
1.4.1	Program Flow Profiling	1-4
1.4.2	Event Profiling.....	1-4
1.4.3	Range Selection.....	1-5
1.4.3.1	Adding Profiling Ranges.....	1-6
1.4.3.2	Profiling Ranges Effect on Data Collection	1-8
2	PROFILER RESULTS	2-1
2.1	Function Viewer Window.....	2-1
2.1.1	Function Viewer Window Right-Click Menus.....	2-2
2.1.1.1	Column Menu	2-2
2.1.1.2	Function Menu.....	2-3
2.2	Function Details Window.....	2-4
2.2.1	General Information Section	2-5
2.2.2	Parents Section.....	2-5
2.2.3	Children Section.....	2-6
2.3	Call Graph Window	2-7
2.3.1	Call Tips	2-8
2.3.2	Call Graph Window Right-Click Menus.....	2-8
2.3.2.1	Function Menu.....	2-8
2.3.2.2	Line Menu	2-9
2.3.2.3	View Menu	2-9
2.4	Branch Prediction Window	2-10
2.4.1	Branch Predication Detailed Tab.....	2-10
2.4.2	Branch Prediction per Function.....	2-11
2.4.3	Branch Prediction Window Right-Click Menu	2-12
2.5	Source Code Profiling Information	2-13
2.5.1	Source Window Right-Click Menu	2-13
2.6	Event Viewer Window	2-14
2.6.1	Graphical Display Tabs – Overview.....	2-15
2.6.1.1	Zoom Operations	2-15
2.6.1.2	Info Box.....	2-15
2.6.1.3	Graphical Event Selection.....	2-15
2.6.1.4	Peaks of Interest.....	2-16
2.6.2	Text Tab	2-16
2.6.2.1	Event Window – Text tab Right-Click Menu	2-17
2.6.3	Function Tab	2-17
2.6.4	Cache Performance Tab.....	2-17
2.6.5	Cache Conflicts Tab	2-19
2.6.6	Events Tab.....	2-19
2.6.7	Data Memory Tab.....	2-20
2.7	Configuring the Profiler Settings.....	2-21
2.7.1	Call Graph Settings	2-21
2.7.2	Table View Settings.....	2-22
3	APPLICATION PROFILING FLOW.....	3-1
3.1	Enable the Profiler	3-1
3.2	Run the Application and Collect Results	3-2
3.3	Advanced Users	3-3
4	PROFILER EXPORT COMMAND LINE	4-1
5	CEVA-XM4 EMULATION PROFILER COUNTERS	5-2

List of Figures

Figure 1-1: Profiler Enable Option	1-1
Figure 1-2: Profiler Setup Dialog Box – Program Flow Tab	1-4
Figure 1-3: Profiler Setup Dialog Box – Events Tab	1-5
Figure 1-4: Profiler Setup Dialog Box – Ranges Tab	1-6
Figure 1-5: Functions Selection Dialog Box	1-7
Figure 1-6: Address Range Selection Dialog Box	1-7
Figure 1-7: Section Range Selection Dialog Box	1-8
Figure 2-1: Function Viewer Window	2-2
Figure 2-2: Columns Choose Dialog	2-3
Figure 2-3: Function Details Window	2-5
Figure 2-4: Branch Prediction Window	2-10
Figure 2-5: Branch Prediction Window	2-11
Figure 2-6: Source Code Window with Profiling Information	2-13
Figure 2-7: Event Viewer - the Text Tab	2-16
Figure 2-8: Event Viewer - the Function Tab	2-17
Figure 2-9: Event Viewer - Cache Performance Tab – Events for Each Address	2-18
Figure 2-10: Event Viewer - the Cache Performance Tab – Events for Each Set	2-18
Figure 2-11: Event Viewer - the Cache Conflicts Tab	2-19
Figure 2-12: Event Viewer – the Events Tab	2-20
Figure 2-13: Event Viewer – the Data Memory Tab	2-20
Figure 2-14: Call Graph Window Settings	2-21
Figure 3-1: How to Activate the Profiler	3-1
Figure 3-2: How Checkboxes should be set for Manual Setting	3-2

1 Profiler Operation

1.1 Invocation (Windows)

The Profiler is disabled by default. It can be enabled by checking “Enable profiler” option in the projects Debug Configuration window. See Figure 1-1:

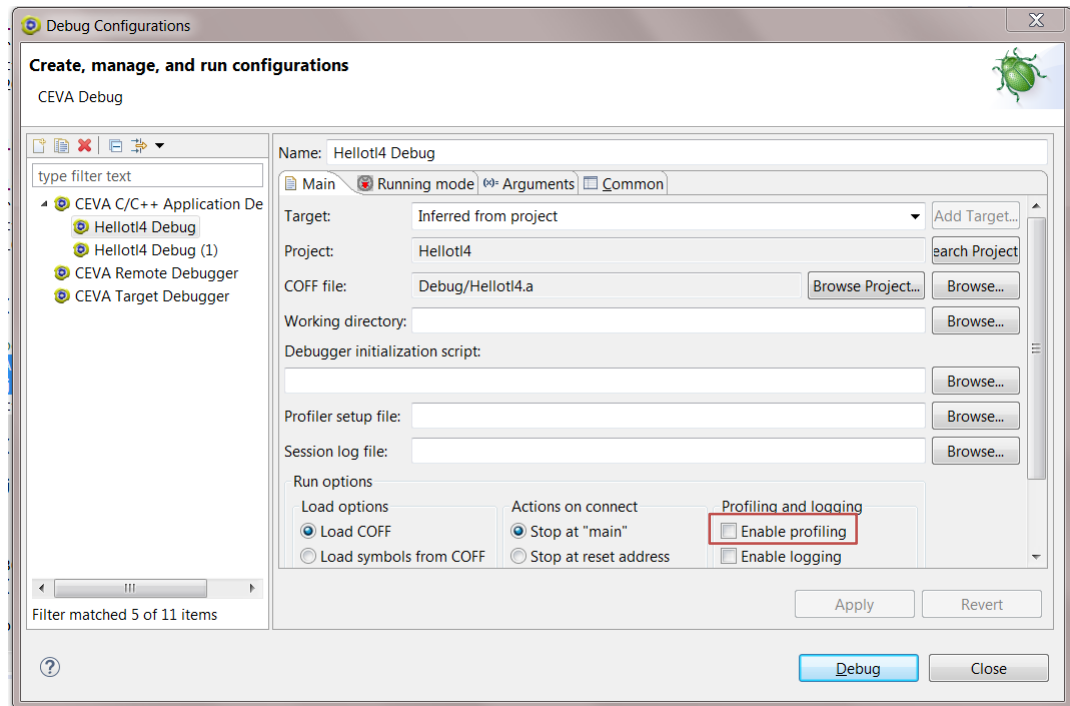
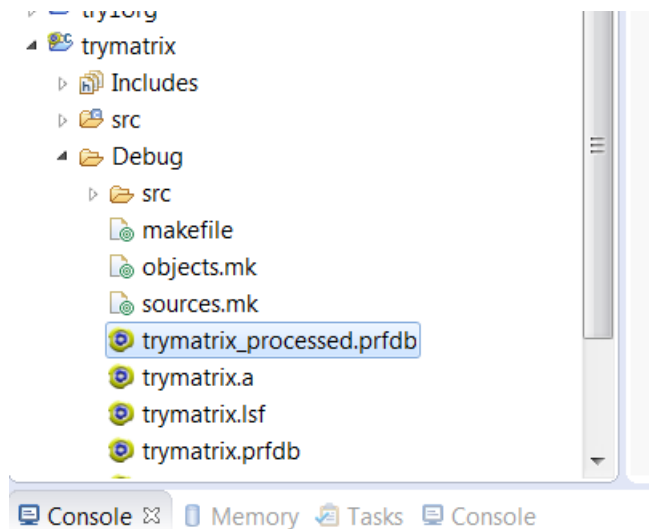


Figure 1-1: Profiler Enable Option

Once enabled, profile data is collected when running the application. At the end of the execution, a profiler database file (.prfdb extension) is created next to the executable file:



Note: You may need to refresh the view (F5) to see the newly generated file.

Double click at the '.prfdb' file automatically will open the Profiler window.

1.2 Menu

A Profiler menu is displayed in the IDE toolbar. The menu includes the following items:

- **Function view** – opens the Function view window
- **Function detail view** – opens the Function view window
- **Call graph view** – opens the Call Graph window
- **Branch view** – opens the Branch Prediction window
- **Source view** – opens the Source view window
- **Event view** – opens the Event Viewer window
- **Create new profiler setup** – opens the Profiler setup window
- **Settings** – opens the Profiler appearance settings

1.3 Invocation (Linux)

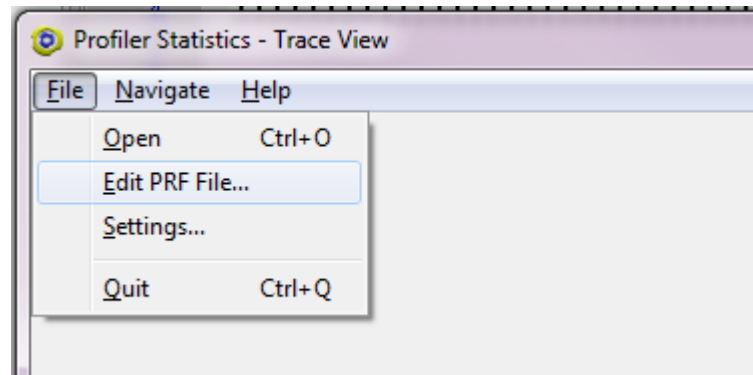
XC V11.0.0 Beta version of the tools does not contains a Linux version of the tools.

1.3.1 Create and Using Profiler Setup File

The user can affect the way the profiler collects information. In order to create a new profiler setup, the user should open the standalone profiler and choose Edit PRF File... from main menu.

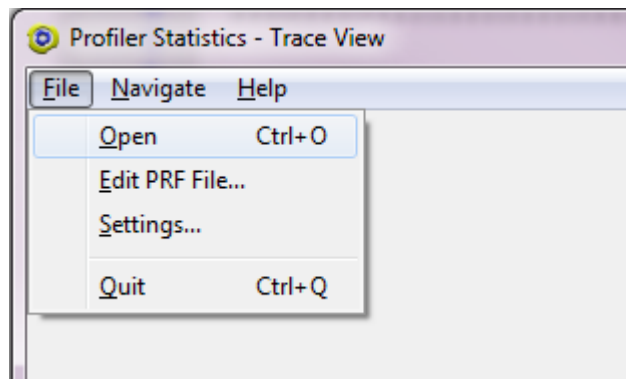
At the end of this process a debugger script file is generated with the configuration of the profiler.

For more information please refer to Section 1.4 Setup.



1.3.2 View Profiler Results

When the user completes the debug session, he can view the result inside the profiler standalone application. To open the profiler results the user should use the open command from the file main menu.



1.4 Setup

The Profiler setup enables the user to configure the Profiler. This window includes the following tabs:

- **Program Flow** – enables the user to activate program flow profiling, and to select the data collection method.
- **Events** – enables the user to activate event profiling, and to select the events that should be profiled.
- **Ranges** – enables the user to select the profiling ranges. The supported range types are: functions, address ranges and sections.

Note that program flow profiling data and event profiling data can be collected either separately or simultaneously.

At the bottom of the Profiler setup window the user can find warnings for the current configuration that indicates whether there is a danger that the database size is too large, or that the run time will be very long.

The following sections detail the configuration tabs.

1.4.1 Program Flow Profiling

The Program Flow tab includes a radio button, which enables selecting data collection method: trace or sampling. When sampling is selected, the user can also specify the sampling interval in machine cycles.

Figure 1-2 shows the Program Flow tab of the Profiler Setup window.

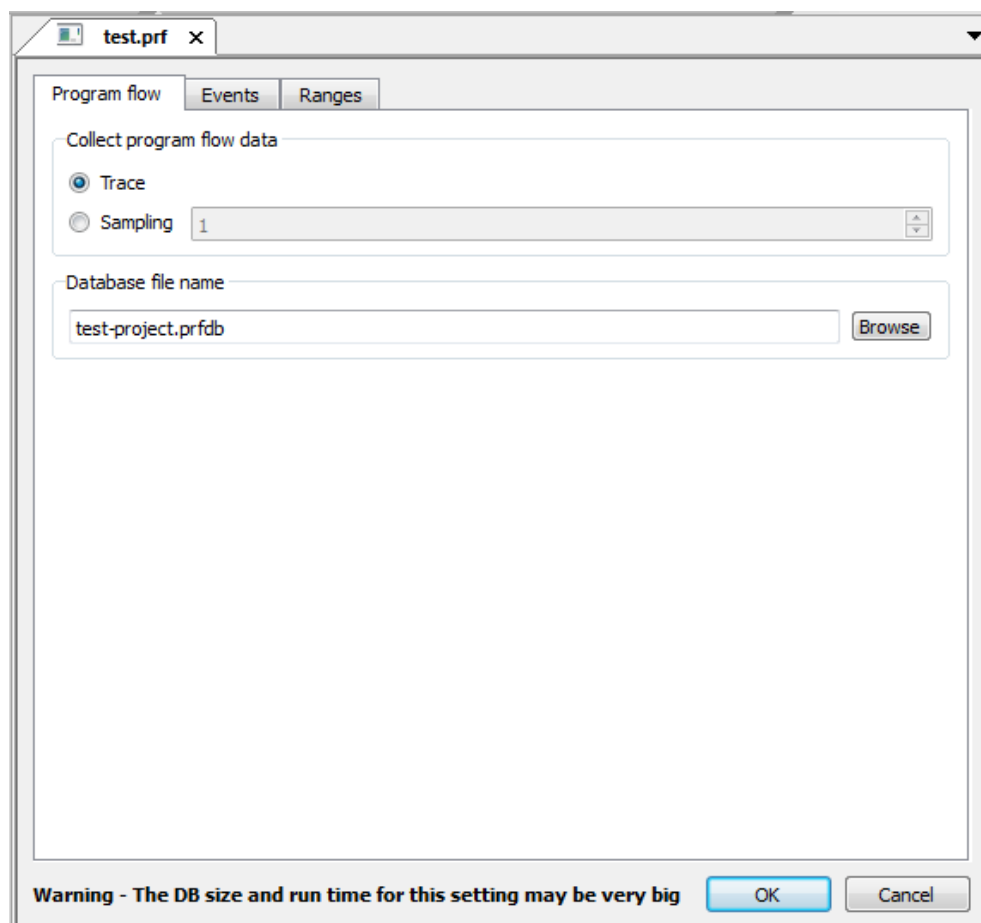


Figure 1-2: Profiler Setup Dialog Box – Program Flow Tab

In this window the user can also set the name of the database that is created.

Note: The trace method is not supported when working in emulation mode.

1.4.2 Event Profiling

Event profiling enables collecting and displaying event-driven profiling information. Event profiling is supported only in simulation mode.

The Events tab includes a tree control for specifying the events that should be monitored and profiled. When an event is selected, a checkbox appears next to it.

When an event is detected, the following information is recorded to the profile database:

- The cycle during which this event occurred
- The program counter of the instruction that was executed when this event occurred
- Information on the event itself, as applicable

Figure 1-3 shows the Events tab of the Profiler Setup dialog box:

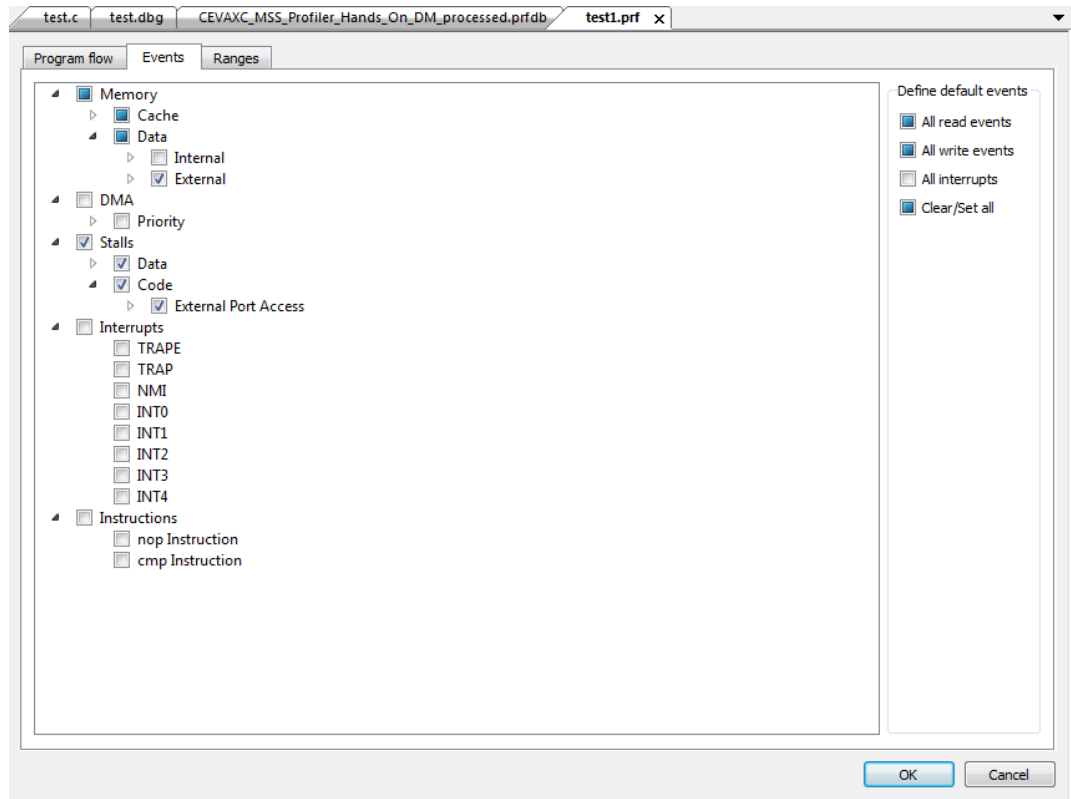


Figure 1-3: Profiler Setup Dialog Box – Events Tab

1.4.3 Range Selection

The Ranges tab enables the user to indicate the ranges that should be profiled. By default, the whole application is profiled. The Ranges tab includes the following radio buttons:

- Profile whole application (selected by default)
- Profile only selected ranges

When the “Profile only selected ranges” radio button is selected, the Ranges tab displays a tree that includes the profiling range types. The tree is initially empty. The following range types are supported:

- Functions

- Address ranges – Code/Data. The Profiler treats each range as if there was a function in that range.
- Sections (code/data)

Figure 1-4 shows the Ranges tab of the Profiler Setup dialog box:

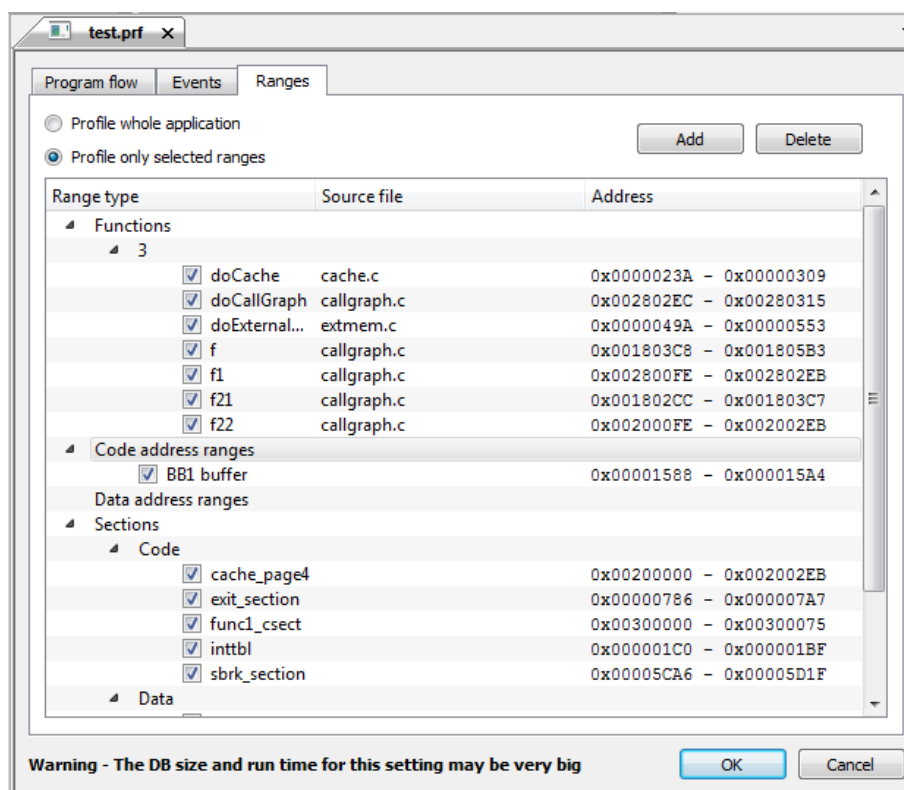


Figure 1-4: Profiler Setup Dialog Box – Ranges Tab

1.4.3.1 Adding Profiling Ranges

The profiling ranges can be added to the range selection categories by double-clicking the category in the Profiler Setup dialog box. A dialog box that opens enables the user to configure the profiling ranges of the selected category. See section 1.4.3.1.1 Range Selection Dialog Boxes for a description of the dialog boxes.

A range entry can be enabled/disabled by pressing on the checkbox next to the range entry.

1.4.3.1.1 Range Selection Dialog Boxes

This section describes the dialog boxes that enable the user to add/remove profiling ranges to/from the Range Selection tree. These dialog boxes open when double-clicking a category in the Range Selection Tree or when selecting a category and pressing the add button.

The Functions Selection dialog box lists the group of enabled/disabled functions and the group of all unmentioned functions, and enables the user to move functions between these two groups. The functions are sorted according to their function level configuration. Figure 1-5 shows the Functions Selection dialog box:

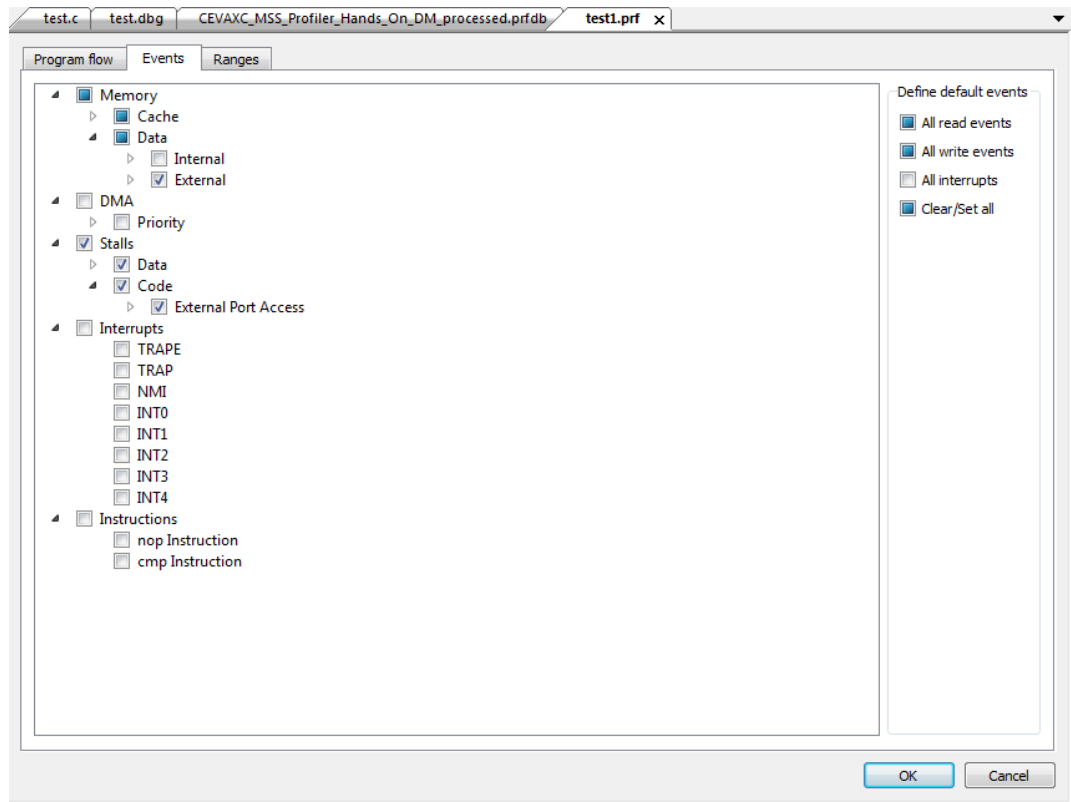


Figure 1-5: Functions Selection Dialog Box

Similar dialog boxes enable to configure which sections are profiled.

Error! Reference source not found. illustrates the Code Address Range Selection dialog box:

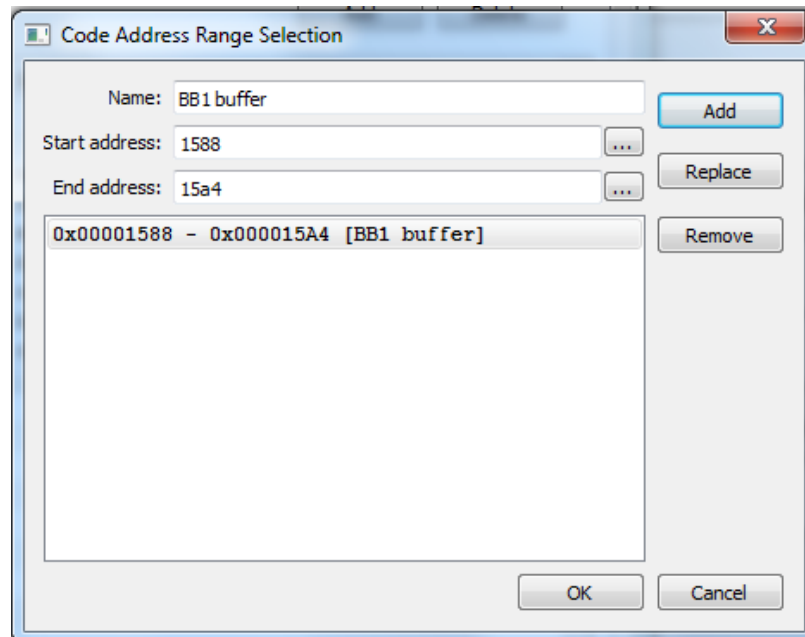


Figure 1-6: Address Range Selection Dialog Box

Similar dialog box enables to configure the code/data section ranges.

Figure 1-7 illustrates the Data section Range Selection dialog box:

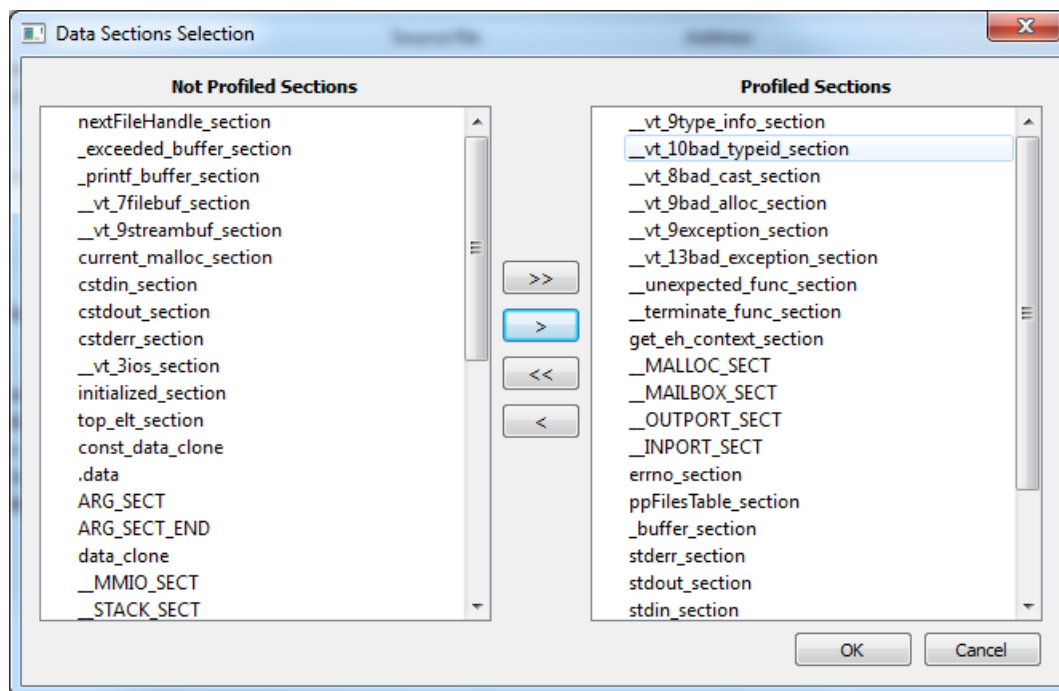


Figure 1-7: Section Range Selection Dialog Box

1.4.3.2 Profiling Ranges Effect on Data Collection

The following guidelines describe how the configured profiling ranges affect the data collection:

- Complete data is collected only for the selected ranges, and only for the enabled ranges.
- Partial data (inclusive cycles and call count) is collected for child functions that are called from parent functions that are selected and enabled, even if the child functions are not selected (or disabled).
- When selecting a range which is part of a function, and the range is disabled, the cycle count of the function still includes the cycle count of the range.

2 Profiler Results

2.1 Function Viewer Window

The Function Viewer window displays profiling information about the functions of the application. The information is displayed in a table that includes a line for each function. Code address ranges are also displayed in this table, as if they were functions. Most of the columns of this table can be included or excluded from the display.

The following information is displayed for the functions:

- The location of the function
- Access count – how many times the function was called
- Code Size
- Cycle count of the function, exclusive (min/max/avg)
- Cycle count of the function, inclusive (min/max/avg)
- Total cycle count (exclusive/inclusive), also in a histogram view
- Function level

The columns that display cycle counts display also the percentage of the cycle count relative to the total cycle count of the application.

Each column can be removed from the display by right-clicking it and selecting **Hide** from the right-click menu. The **Columns** right-click menu item enables selecting which columns are displayed.

The order of the displayed columns can be changed by dragging a column and dropping it in the desired location.

The list can be sorted by each one of the columns, by double-clicking the column header.

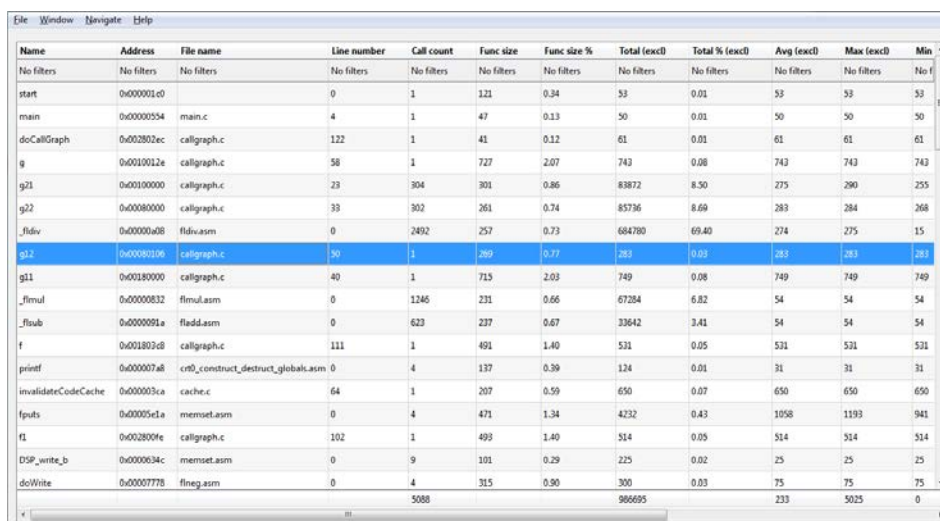
Each column can be filtered by the information inside the column.

Double-clicking a function entry (anywhere in the line) opens the Function Details window. Refer to section 2.2 for details.

Selecting several functions in the Function Viewer window displays the sum of the cycle counts of these functions in the status bar.

The title and total row is placed statically in the windows and always been shown, the function should be scrolled between them.

Figure 2-1 shows the structure of the Function Viewer window:



Name	Address	File name	Line number	Call count	Func size	Func size %	Total (exc)	Total % (exc)	Avg (exc)	Max (exc)	Min
No filters	No filters	No filters	No filters	No filters	No filters	No filters	No filters	No filters	No filters	No filters	No f
start	0x00001c0		0	1	121	0.34	53	0.01	53	53	53
main	0x0000554	main.c	4	1	47	0.13	50	0.01	50	50	50
doCallGraph	0x002802ec	callgraph.c	122	1	41	0.12	61	0.01	61	61	61
g	0x0010012e	callgraph.c	58	1	727	2.07	743	0.08	743	743	743
g21	0x00100000	callgraph.c	23	304	301	0.86	83872	8.50	275	290	255
g22	0x00080000	callgraph.c	33	302	261	0.74	85736	8.69	283	284	268
_fdiv	0x0000a08	fdiv.asm	0	2492	257	0.73	684780	69.40	274	275	15
g12	0x00001096	callgraph.c	30	1	269	0.77	283	0.03	283	283	283
g11	0x00180000	callgraph.c	40	1	715	2.03	749	0.08	749	749	749
_fmul	0x00000332	fmul.asm	0	1246	231	0.66	67284	6.82	54	54	54
_fsub	0x0000091a	fadd.asm	0	623	237	0.67	33642	3.41	54	54	54
f	0x001802d8	callgraph.c	111	1	491	1.40	531	0.05	531	531	531
printf	0x000007a8	cn0_construct_destruct_globals.asm	0	4	137	0.39	134	0.01	31	31	31
invalidateCodeCache	0x000003ca	cache.c	64	1	207	0.59	650	0.07	650	650	650
fputs	0x00005c1a	memset.asm	0	4	471	1.34	4232	0.43	1058	1193	941
f1	0x0028009e	callgraph.c	102	1	493	1.40	514	0.05	514	514	514
DSP_write_b	0x0000034c	memset.asm	0	9	101	0.29	225	0.02	25	25	25
doWrite	0x00007778	fineg.asm	0	4	315	0.90	300	0.03	75	75	75
				5088			986695		233	5025	0

Figure 2-1: Function Viewer Window

The appearance of the Function Viewer window, and the data that it displays can be configured using the Settings dialog box of the IDE (see section 2.7), or by using the right-click menus of the Function Viewer window (see section 2.1.1).

2.1.1 Function Viewer Window Right-Click Menus

The Function Viewer window supports the following right-click menus:

- **Column** – available when right-clicking a column header.
- **Function** – available when right-clicking a function entry (anywhere in the line).

The following sections describe the right-click menus of the Function Viewer window.

2.1.1.1 Column Menu

The Column right-click menu includes the following items:

- **Hide** – hides the selected column(s). This menu item is enabled only when clicking a column header.
- **Columns** – opens a dialog box that includes all optional columns, all columns that already displayed are appear in the enable column, all other symbols appears in the disable column. In the enable column the user can define the order of the columns to display by selecting the desired column and pushing the up and down arrows.

- Figure 2-2 shows the columns choose dialog:

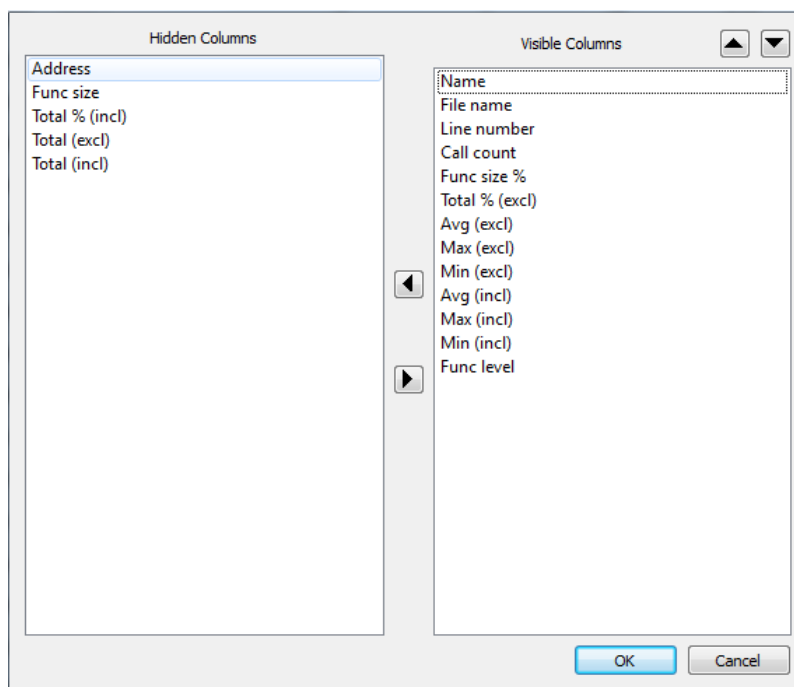


Figure 2-2: Columns Choose Dialog

- **Export** – export the table information to tab delimited file

2.1.1.2 Function Menu

The Function right-click menu includes the following items:

- **Show Details** – opens the Function Details window for the selected function (see section 2.2).
- **Show Source** – displays the function in the source code window (see section 2.5).
- **Show References** – shows all the references of this function in the output window of the IDE.
- **Export** – exports table's information to tab delimited file

2.2 Function Details Window

The Function Details window provides detailed information about a single function. This window is opened in the following scenarios:

- When double-clicking a function rectangle in the Call Graph window.
- When selecting the **Show Details** item in the function right-click menu of the Call Graph window.
- When double-clicking a function entry in the Function Viewer window (anywhere in the line).
- When selecting the **Show Details** item in the function right-click menu of the Function Viewer window.
- From the source code window – by clicking the **Show Profile Details** item in the right-click menu of the source code window.

This window is split into three sections:

- General information – shows textual information about the function.
- Parents – shows details on the parent functions of this function
- Children – shows details on the child functions of this function

Forward and Back buttons in the Function Details window enable to switch between details of functions that were already displayed in the window.

A functions dropdown list enables selecting which function is displayed in the window.

Note: The Function Details window is disabled if the program flow data was collected using the sampling method.

Figure 2-3 shows the structure of the Function Details window:

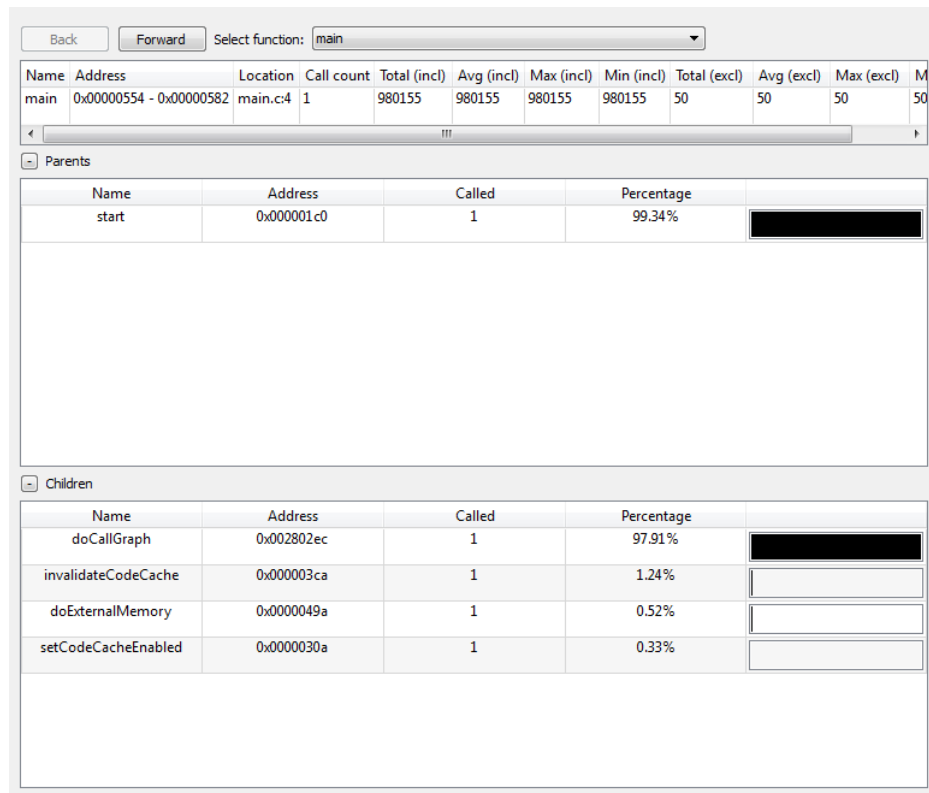


Figure 2-3: Function Details Window

2.2.1 General Information Section

The general information section displays some of the information that is displayed in the Function Viewer window.

2.2.2 Parents Section

The Parents section displays a function list. The list includes an entry for each parent, and displays the following information:

- Parent function name
- Parent function address
- The number of times the current function was called by this parent
- A bar chart that indicates the percentage of time spent in the current function relative to the inclusive time spent in the parent function. Note that these percentages don't add up to 100%.

Double-clicking a function entry in the list causes the Function Details window to display information about the selected parent.

2.2.3 Children Section

The Children section displays a function list. The list includes an entry for each child, and displays the following information:

- Child function name
- Child function address
- The number of times the current function called this child function
- A bar char that indicates the percentage of time spent in the child function relative to the inclusive time spent in the current function

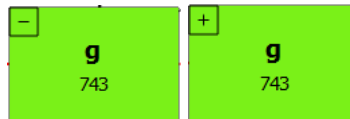
Double-clicking a function entry in the list causes the Function Details window to display information about the selected child.

2.3 Call Graph Window

The Call Graph window displays the call graph of the application. It provides information about the program flow, and information about each function:

- The number of times the function was called
- The number of cycles the function costs

Functions are drawn from left to right, according to the program flow. Each function is denoted by a rectangle that displays its name. The function rectangles are color-coded, meaning that the functions that cost the most cycles are colored red, and the colors vary between red (most cycles) to green (least cycles). By default, there are four color levels, but the colors can be changed in the **Options** dialog box of the Profiler (see section 2.7.1). The colored calculation is based on the exclusive cycle count of the function. a +/- indication is shown at the top left corner to notify the user if a function has children to be printed – an indication meaning that the user can close all children under this function, + indication means that the user can show all children of the function. By default the call graph window is open with the root element.



Parent functions are connected to their child functions by lines. The width of the line indicates how many times the child function was called by the parent function. The edges of the lines that are drawn from a parent function to child functions originate at a “plus/minus” box, which enables collapsing/expanding the sub-graph of each function.

The critical path of the application can be displayed in the call graph. When enabled (by the View menu, see section 2.3.2.3), the lines that connect the functions of the critical path are highlighted in red (the functions colors are unaffected by the critical path display).

Resizing the Call Graph window causes the call graph to be resized accordingly.

Selecting a function (by a mouse-click) highlights them, and enables dedicated keyboard operations (that can be performed also by the right-click menus). These operations are mentioned in section 2.3.2.

Call graph window is constructed as a tree. The root is printed in the most left part of the window. The root Children are printed to the right. Recursive function is printed as a rectangle with a line to itself. If a function is called from more than one lower levels.

Each function element can be used as the root of the graph according to a user request (see section 2.3.2).

Double-clicking a function rectangle or selecting **Show Details** from the functions right-click menu opens the Function Details window. Refer to section 2.2 for details.

The user can set the threshold of functions that are shown in the call graph window. The percentage determines the minimum level of a function cycle count in relation to the entire application. If the function does not cross this lower boundary it is not be displayed. The coverage shows the total percentage of the remaining functions in relation to the entire application. The find function allows the user to search for a function in the call graph. Once a function is found and selected, the view is zoomed in on the function. The appearance of the Call Graph window, and the data that it displays can be configured using the Options dialog box of the IDE (see section 2.7), or by using the right-

click menus of the Call Graph window (see section 2.3.2). Note: the Call Graph window is disabled if the program flow data was collected using the sampling method.

2.3.1 Call Tips

When the mouse pointer is placed over a function, a call tip is displayed, with the following information:

- The function name
- The number of times the function was called
- The number of cycles the function costs (excluding child functions), and the percentage of this count relative to the whole application
- The number of cycles the function costs (including child functions), and the percentage of this count relative to the whole application

When the mouse pointer is placed over a line, a call tip is displayed, showing the access count and the functions that correspond to this line.

2.3.2 Call Graph Window Right-Click Menus

The Call Graph window supports the following right-click menus:

- **Function** – available when right-clicking a function rectangle.
- **Line** – available when right-clicking a line.
- **View** – available when right-clicking the call graph background.

The following sections describe the right-click menus of the Call Graph window.

2.3.2.1 Function Menu

The right-click menu that opens when right-clicking a function rectangle includes the following items:

- **Remove** – removes the function from the call graph (a function can also be removed by selecting it and pressing the Delete key).
- **Focus** – changes the call graph so that it originates at this function (can also be performed by selecting a function and pressing the “+” key).
- **Display Cycles** – adds the function cycle count inside each function rectangle (checked item, applies to all functions in the call graph).
- **Show Details** – opens the Function Details window for the selected function (see section 2.2).
- **Show Source** – displays the function in the source code window.
- **Show References** – shows all the references of this function in the output window of the IDE.
- **Use As Root** – refreshes the window and locates the function as the root of the graph

2.3.2.2 Line Menu

The right-click menu that opens when right-clicking a line includes the following item:

- **Show Access Counts** – adds the access count above each line (checked item, applies to all lines in the call graph).

2.3.2.3 View Menu

The right-click menu that opens when right-clicking the call graph background includes the following items:

- **Display cycles** – shows the cycle count in the functions.
- **Show access count** – show the call count of all the functions.
- **Show all** – displays all the functions that were removed from the graph.
- **Critical Path** – displays the critical path by highlighting the lines on this path in red.
- **Fit to window** – resizes the graph so it is fit entirely in the window.
- **Re-layout** – activates show all and fit to window.
- **Zoom In**
- **Zoom Out**

Note: the zoom operations can also be performed by pressing the Ctrl + mouse scroll button.

2.4 Branch Prediction Window

The Branch Prediction window displays the branch prediction statistics. The information is displayed in two tabs:

- **Detailed** – the information is per branch instruction
- **Summary** – the information is per function

2.4.1 Branch Predication Detailed Tab

The information is displayed in a table that lists the number of times each branch instruction was taken or not taken. This table includes the following columns:

- **Source File** – the file that includes the branch
- **Function Name** – if the code is inside c files then it shows the relevant function name
- **Line** – the line number of the branch instruction in the source file
- **Code address** – the assembly code address the branch instruction is at
- **Instruction** – the branch instruction
- **Static Prediction** – the static prediction of the branch instruction
- **Hit Ratio** – the percentage of correct prediction relative to the total number of times the branch was executed
- **Taken** – the number of times this branch was taken
- **Not Taken** – the number of times this branch was not taken
- **Total Hit** – the total number of time this instruction was executed

Figure 2-4 illustrates the Branch Prediction window:

Details Summary									
Source File	Function Name	Line	Code Address	Instruction	Static Prediction	Hit Ratio (%)	Taken Count	Not Taken Count	Total Hit
..\main.c	main	8	0x00000556	Call	Taken	100	1	0	1
..\main.c	main	11	0x00000560	Call	Taken	100	1	0	1
..\main.c	main	14	0x00000568	Call	Taken	100	1	0	1
..\main.c	main	17	0x00000570	Call	Taken	100	1	0	1
..\main.c	main	22	0x00000582	Ret	Taken	100	1	0	1
..\extmem.c	doExternalMem...	10	0x000004ba	Call	Taken	100	1	0	1
..\extmem.c	doExternalMem...	11	0x000004d8	Br	Taken	0	0	1	1
..\extmem.c	doExternalMem...	15	0x0000053c	Br	Not Taken	0	127	1	128
..\extmem.c	doExternalMem...	16	0x00000552	Ret	Taken	100	1	0	1
..\callgraph.c	g22	34	0x00080010	Call	Taken	100	302	0	302
..\callgraph.c	g22	35	0x00080028	Call	Taken	100	302	0	302
..\callgraph.c	g22	35	0x00080042	Call	Taken	100	302	0	302
..\callgraph.c	g22	35	0x00080066	Call	Taken	100	302	0	302
..\callgraph.c	g22	35	0x0008007e	Call	Taken	100	302	0	302
..\callgraph.c	g22	35	0x000800a8	Call	Taken	100	302	0	302
..\callgraph.c	g22	35	0x000800ba	Call	Taken	100	302	0	302
..\callgraph.c	g22	35	0x000800c8	Call	Taken	100	302	0	302
Total:						93	7050	304	7354

Figure 2-4: Branch Prediction Window

Double-clicking a branch prediction entry (anywhere in the line) opens the source view window.

Total row, show information regard the entire project and it show the total hit ratio of the application, the total taken count and the total not taken count. Selecting several branch prediction entries in the window displays the same fields as the total but with the calculation of the selected branches.

The title and total row is placed statically in the windows and always been shown, the function should be scrolled between them. The table is sorted according to source file name by default but the user can sort according to other columns by double-clicking on the column header.

2.4.2 Branch Prediction per Function

The information is displayed in a table that lists the number of times each function had a branch instruction in it. This table includes the following columns:

- **Source File** – the file that includes the branch
- **Function Name** – if the code is inside c files then it shows the relevant function name
- **Total branches** – the total number of branch instructions in the function
- **Taken branches** – the total number of times branch instructions were taken
- **Not Taken branches** – the total number of times branch instructions were not taken
- **Prediction hit** ratio – the function average hit ratio on the branch instruction

Figure 2-5 illustrates the Branch Prediction window summary tab:

Summary						
Function Name	Function Address	File Name	Total Branches	Taken Branches	Not Taken Branches	Prediction Hit Ratio (%)
start	0x000001c0		8	8	0	100
setCodeCacheEnabled	0x0000030a	..\cache.c	4	4	0	100
setCodeCacheDisabled	0x0000036a	..\cache.c	4	4	0	100
main	0x00000554	..\main.c	5	5	0	100
initConfigRegs	0x000005ac		1	1	0	100
initFuncVect	0x000006a6		4	4	0	100
crt0HookPreMain	0x000006dc		1	1	0	100
exit	0x00000786		1	1	0	100
_flmul	0x00000832		1246	1246	0	100
_flsub	0x0000091a		1869	1869	0	100
memset	0x00000b0a		4	2	2	100
init_malloc	0x00000be0		2	1	1	100
sbrk	0x00000628		6	6	0	100
DSP_write_b	0x00000634c		36	18	18	100
genMailboxTrap	0x0000063b2		5	5	0	100
writelCpm	0x00000793e		3	3	0	100
readloCpm	0x000007952		33	33	0	100
g22	0x00080000	..\callgraph.c	3322	3322	0	100
g12	0x00080106	..\callgraph.c	12	12	0	100

Figure 2-5: Branch Prediction Window

The table is sorted by any of the column headers.

2.4.3 Branch Prediction Window Right-Click Menu

The branch prediction window supports the following right-click menus:

- **Export** – export the table information to tab delimited file.
- **Show Source** – open the source file on the relevant line.
- **Show external code** – display all the code that does not have source debug information.
- **Show zero hit branches** – by default all the branches that have zero hit count are hidden. In order to display them this menu should be selected.

2.5 Source Code Profiling Information

When program flow data is collected for an application using the trace method, profiling information can be displayed in the Source Code window.

The Source Code window can be opened by right-clicking a function in the Call Graph window or the Function Viewer window, and selecting the Show Source menu item.

The following information can be displayed for each source code line:

- The number of times this line was executed
- The total time spent on executing this line (inclusive/exclusive)
- The execution time percentage, relative to the application (inclusive/exclusive)

Figure 2-6 shows the structure of the Source Code window, when profiling information is displayed:

	Execute Count	Total(excl)	Rel(excl)	Total(incl)	Avg(incl)	Min(incl)	Max(incl)	Rel(incl)	Source Code
1									char tmp_in[128] __attribute__((aligned(16)))
2									char tmp_out[128] __attribute__((aligned(16)))
3									
4									int doExternalMemory (void)
5	1	6	0.00						{
6									int i;
7									char a;
8									
9									
10	1	8	0.00	65	65	65	65	0.0066	memset(tmp_in, 0x55, sizeof(tmp_in))
11	1	12	0.00						for (i=0; i<128; i++)
12									{
13	128	1152	0.12						a = tmp_in[i];
14	128	1536	0.16						tmp_out[i] = a;
15	128	2300	0.23						}

Figure 2-6: Source Code Window with Profiling Information

The Source Code window of the Profiler does not display syntax highlighting for the code. Instead, the most expensive source lines are highlighted in red.

Double-clicking a source line opens the file in a regular editor window.

2.5.1 Source Window Right-Click Menu

The source window supports the following right-click menus:

- **Export** – export the table information to tab delimited file.
- **Function detail** – open the function detail window on the relevant function

2.6 Event Viewer Window

The Event Viewer window displays the event-driven profiling information that was collected when running the application.

The Event Viewer can display only the data that corresponds to the events that were monitored when the application was simulated (no support for emulation mode). Refer to section 1.4.2 for details on selecting the events to be monitored.

The Event Viewer includes two textual display tab, and four graphical display tabs:

- **Text** – in this tab, textual data is displayed in a list. See section 2.6.2 for details.
- **Functions** – in this tab, data is displayed in a graph, and the horizontal axis of this graph represents the functions. See section 2.6.1 for details.
- **Cache Performance** – in this tab, cache performance data is displayed in a graph. See section 2.6.40 for details.
- **Cache Conflicts** – in this tab, cache conflicts are displayed in a table. See section 2.6.5 for details.
- **Events** - in this tab, non-cache event data is displayed in a graph. See section 2.6.6 for details.
- **Data Memory** - in this tab, data memory access counts are displayed in a graph, where the horizontal axis represents the data memory address range. See section 2.6.7 for details.

Text	Function	Cache Performance	Cache Conflicts	Events	Data Memory				
	Function	Address	Exclusive count	Code Cache Rd Hit	Code Cache Rd Miss	Data Rd Internal	Data Wr Internal	Code Wait Ext Rd	Bank Conflicts
	start	0x000001c0	53	0 (0.00 %)	0 (0.00 %)	2 (3.77 %)	1 (1.89 %)	0 (0.00 %)	0 (0.00 %)
	doCache	0x0000023a	0	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)
	setCodeCacheEnabled	0x0000030a	45	0 (0.00 %)	0 (0.00 %)	4 (8.89 %)	4 (11.11 %)	0 (0.00 %)	0 (0.00 %)
	setCodeCacheDisabled	0x0000036a	45	0 (0.00 %)	0 (0.00 %)	4 (8.89 %)	4 (11.11 %)	0 (0.00 %)	0 (0.00 %)
	invalidateCodeCache	0x000003ca	650	0 (0.00 %)	0 (0.00 %)	35 (5.38 %)	5 (0.92 %)	0 (0.00 %)	0 (0.00 %)
	doExternalMemory	0x0000049a	5025	0 (0.00 %)	0 (0.00 %)	771 (15.34 %)	387 (12.82 %)	0 (0.00 %)	0 (0.00 %)
	main	0x00000554	50	0 (0.00 %)	0 (0.00 %)	1 (2.00 %)	1 (2.00 %)	0 (0.00 %)	0 (0.00 %)
	initConfigRegs	0x000005ac	20	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)
	initDataSections	0x000005ec	4029	0 (0.00 %)	0 (0.00 %)	284 (7.05 %)	284 (7.05 %)	0 (0.00 %)	0 (0.00 %)
	initFuncVect	0x000006a6	29	0 (0.00 %)	0 (0.00 %)	1 (3.45 %)	1 (3.45 %)	0 (0.00 %)	0 (0.00 %)
	crt0HookPreMain	0x000006dc	5	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)
	construct_globals	0x000006de	24	0 (0.00 %)	0 (0.00 %)	3 (12.50 %)	3 (12.50 %)	0 (0.00 %)	0 (0.00 %)
	destruct_globals	0x00000732	24	0 (0.00 %)	0 (0.00 %)	3 (12.50 %)	3 (12.50 %)	0 (0.00 %)	0 (0.00 %)
	exit	0x00000786	5	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	2 (40.00 %)	0 (0.00 %)	0 (0.00 %)
	printf	0x000007a8	124	0 (0.00 %)	0 (0.00 %)	24 (19.35 %)	16 (12.90 %)	0 (0.00 %)	0 (0.00 %)
	_fmul	0x00000832	67284	0 (0.00 %)	0 (0.00 %)	4984 (7.41 %)	4984 (7.41 %)	0 (0.00 %)	0 (0.00 %)
	_fsub	0x0000091a	33642	0 (0.00 %)	0 (0.00 %)	7476 (22.22 %)	7476 (22.22 %)	0 (0.00 %)	0 (0.00 %)
			988199	23336 (4.72 %)	8630 (4.58 %)	36423 (3.73 %)	30173 (3.35 %)	8630 (2.84 %)	333 (0.03 %)

Note: When code address ranges are profiled, they are displayed in the Event Viewer window as if there were functions in this code address ranges.

2.6.1 Graphical Display Tabs – Overview

This section describes the common features of the graphical display tabs.

These tabs display the event-driven profiling information in a graph. The window is divided into two sections:

- The graph section (on the right side)
- The details section (on the left side)

2.6.1.1 Zoom Operations

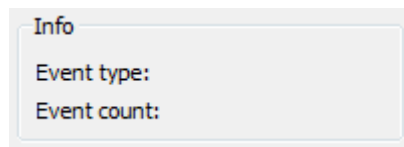
The Event Viewer window supports zooming operations on the graphs. These operations are supported by a dedicated toolbar which is included in the graphical display tabs of the Event Viewer window. The zooming options are:

- **Zoom to Selection** – enables selecting the zoom area by the mouse
- **Zoom Out Full (w)** – zoom out full, in both dimensions
- **Horizontal Zoom In (x+)** – zoom in, only in the horizontal dimension
- **Horizontal Zoom Out (x-)** – zoom out, only in the horizontal dimension
- **Vertical Zoom In (y+)** – zoom in, only in the vertical dimension
- **Vertical Zoom Out (y-)** – zoom out, only in the vertical dimension

When zooming into a graph, scroll bars enable scrolling in both directions.

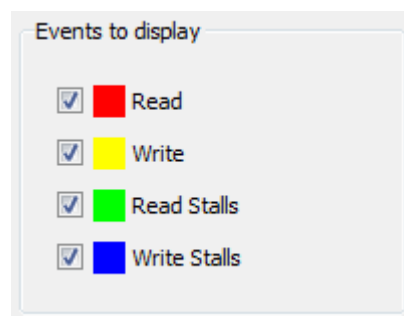
2.6.1.2 Info Box

The info box displays information about the point in the graph that the mouse points at, and enables selecting which events are displayed.



2.6.1.3 Graphical Event Selection

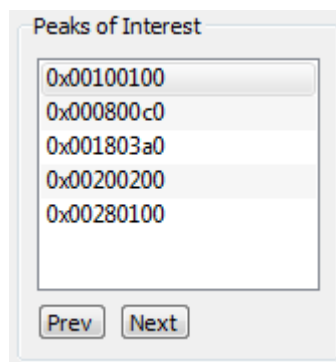
Each graph has the relevant event selection box that is in the left side of the graph.



By selecting the relevant event the graphical display changes according to the selection.

2.6.1.4 Peaks of Interest

Some of the graphs have a Peaks of Interest box. This box has a list of accumulated areas in the graph that the user may be interested in, sorted from the highest hit to the lowest.



The user can jump from area to area using the Next and Prev buttons, or by selecting the relevant address.

2.6.2 Text Tab

The Text tab lists the number of counted events per function. The results are displayed in a table.

Figure 2-7 shows the Text tab of the Event Viewer window:

Text	Function	Cache Performance	Cache Conflicts	Events	Data Memory				
	Function	Address	Exclusive count	Code Cache Rd Hit	Code Cache Rd Miss	Data Rd Internal	Data Wr Internal	Code Wait Ext Rd	Bank Conflicts
	start	0x000001c0	53	0 (0.00 %)	0 (0.00 %)	2 (3.77 %)	1 (1.89 %)	0 (0.00 %)	0 (0.00 %)
	doCache	0x0000023a	0	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)
	setCodeCacheEnabled	0x0000030a	45	0 (0.00 %)	0 (0.00 %)	4 (8.89 %)	4 (11.11 %)	0 (0.00 %)	0 (0.00 %)
	setCodeCacheDisabled	0x0000036a	45	0 (0.00 %)	0 (0.00 %)	4 (8.89 %)	4 (11.11 %)	0 (0.00 %)	0 (0.00 %)
	invalidateCodeCache	0x000003ca	650	0 (0.00 %)	0 (0.00 %)	35 (5.38 %)	5 (0.92 %)	0 (0.00 %)	0 (0.00 %)
	doExternalMemory	0x0000049a	5025	0 (0.00 %)	0 (0.00 %)	771 (15.34 %)	387 (12.82 %)	0 (0.00 %)	0 (0.00 %)
	main	0x00000554	50	0 (0.00 %)	0 (0.00 %)	1 (2.00 %)	1 (2.00 %)	0 (0.00 %)	0 (0.00 %)
	initConfigRegs	0x000005ac	20	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)
	initDataSections	0x000005ec	4029	0 (0.00 %)	0 (0.00 %)	284 (7.05 %)	284 (7.05 %)	0 (0.00 %)	0 (0.00 %)
	initFuncVect	0x000006a6	29	0 (0.00 %)	0 (0.00 %)	1 (3.45 %)	1 (3.45 %)	0 (0.00 %)	0 (0.00 %)
	crt0HookPreMain	0x000006dc	5	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)
	construct_globals	0x000006de	24	0 (0.00 %)	0 (0.00 %)	3 (12.50 %)	3 (12.50 %)	0 (0.00 %)	0 (0.00 %)
	destruct_globals	0x00000732	24	0 (0.00 %)	0 (0.00 %)	3 (12.50 %)	3 (12.50 %)	0 (0.00 %)	0 (0.00 %)
	exit	0x00000786	5	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	2 (40.00 %)	0 (0.00 %)	0 (0.00 %)
	printf	0x000007a8	124	0 (0.00 %)	0 (0.00 %)	24 (19.35 %)	16 (12.90 %)	0 (0.00 %)	0 (0.00 %)
	_flimul	0x00000832	67284	0 (0.00 %)	0 (0.00 %)	4984 (7.41 %)	4984 (7.41 %)	0 (0.00 %)	0 (0.00 %)
	_flsub	0x0000091a	33642	0 (0.00 %)	0 (0.00 %)	7476 (22.22 %)	7476 (22.22 %)	0 (0.00 %)	0 (0.00 %)
		988199		23336 (4.72 %)	8630 (4.58 %)	36423 (3.73 %)	30173 (3.35 %)	8630 (2.84 %)	333 (0.03 %)

Figure 2-7: Event Viewer - the Text Tab

The user can select which columns are displayed by selecting the **columns** item of the right-click menu (see section 2.6.2.1). The data can be sorted by each one of the columns, by double-clicking the column header.

2.6.2.1 Event Window – Text tab Right-Click Menu

The Text Tab supports the following right-click menus:

- **Export** – export the table information to tab delimited file.
- **Columns** – open the columns select dialog box.

2.6.3 Function Tab

The function tab is a graphical display of the Text Tab. The user can select which event to display and the graph is automatically sorted according to the highest event hit per function.

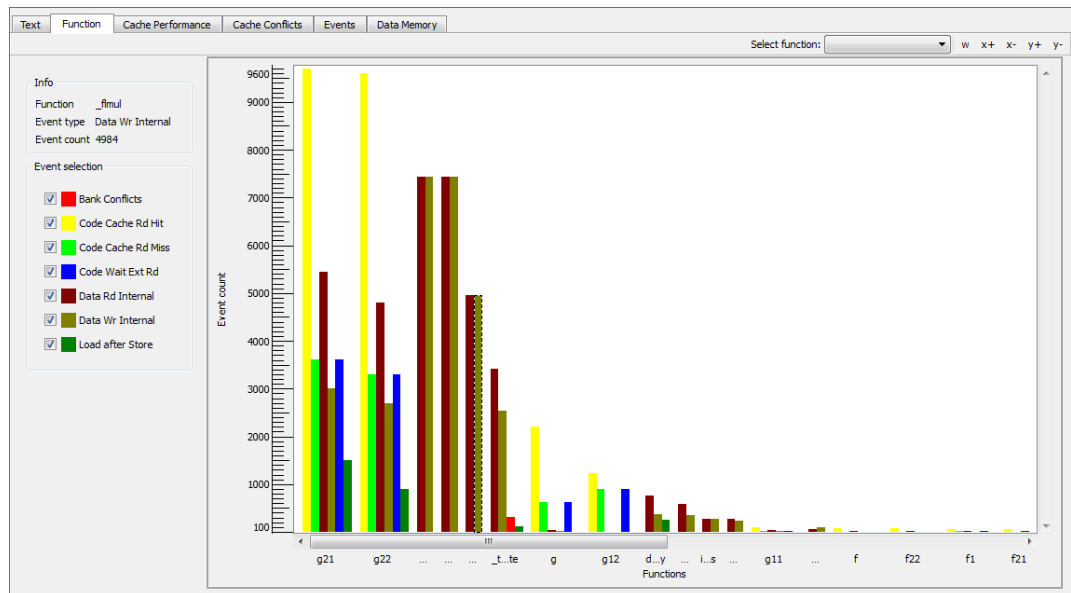


Figure 2-8: Event Viewer - the Function Tab

2.6.4 Cache Performance Tab

The Cache performance tab displays cache events. The events can be displayed per address or for each cache set when the group by cache set is selected.

Cache selection: radio buttons in the details section enable the user to select one of the following options:

- **Program** – when selected, the graph shows events for the program cache. The horizontal axis represents the program address range.
- **Data** – when selected, the graph shows events for the data cache. The horizontal axis represents the data address range.

In the details section, when the Group by cache set checkbox is disabled the hit/miss graph is displayed.

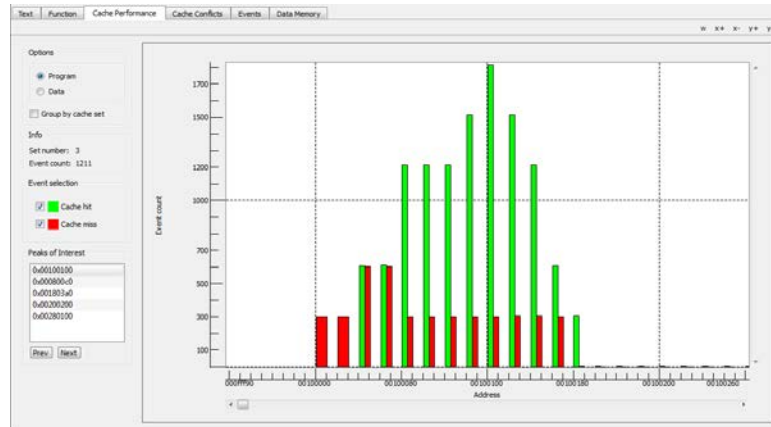


Figure 2-9: Event Viewer - Cache Performance Tab – Events for Each Address

This graph shows for each address, the amount of hits/misses of the profiled application.

In the details section, when the Group by cache set checkbox is set, the graph is displayed a histogram of the number of events in each one of the cache sets (the horizontal axis represents the cache sets).

Figure 2-10 is an example of a cache performance graph, which shows the events per set:

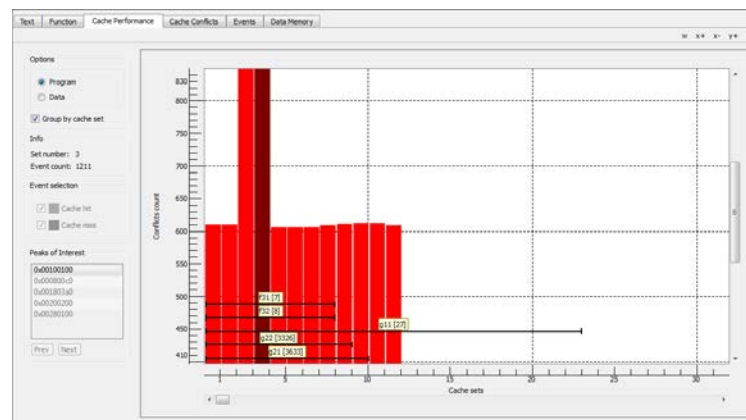


Figure 2-10: Event Viewer - the Cache Performance Tab – Events for Each Set

When pressing on the column of a cache set the functions that occupy this cache set are displayed and each function marks the other cache set that it occupies. When pressing on any of the functions, the user receives an indication of which other functions conflict with the function that has been selected.

When double-clicking on any of the functions, the Cache conflict tab is opened on the selected function.

2.6.5 Cache Conflicts Tab

The Cache Conflicts tab of the Event Viewer displays a table that lists the functions that cause conflicts with the current function. The current function can be selected by using the function browser of this tab. The table displays the following information on each conflicting function:

- The function name.
- The function address.
- The number of conflicts.
- Exclusive count of the function.
- The ratio of the conflicting function with the conflicted function.
- The source file of the function, and the line of the beginning of this function in it.

Figure 2-11 shows the Cache Conflicts tab of the Event Viewer window:

Function Name	Address	Number of Conflicts	Exclusive count	Ratio(%)	Line	Source File
g22	0x00000000 1812	85736	1	33		..localgraphic.c
g21	0x00100000 606	83872	0	23		..localgraphic.c
g11	0x00180000 2	749	0	40		..localgraphic.c
g12	0x00000106 1	283	2	50		..localgraphic.c

Figure 2-11: Event Viewer - the Cache Conflicts Tab

2.6.6 Events Tab

The Events tab displays non-cache events in a graph which is similar to the graph of the Cache Performance tab (Events/cycle). The horizontal axis represents the application execution cycles. The vertical axis represents the program counter.

This graph indicates when events were detected, and which function was being performed when each event occurred. See Figure 2-12:

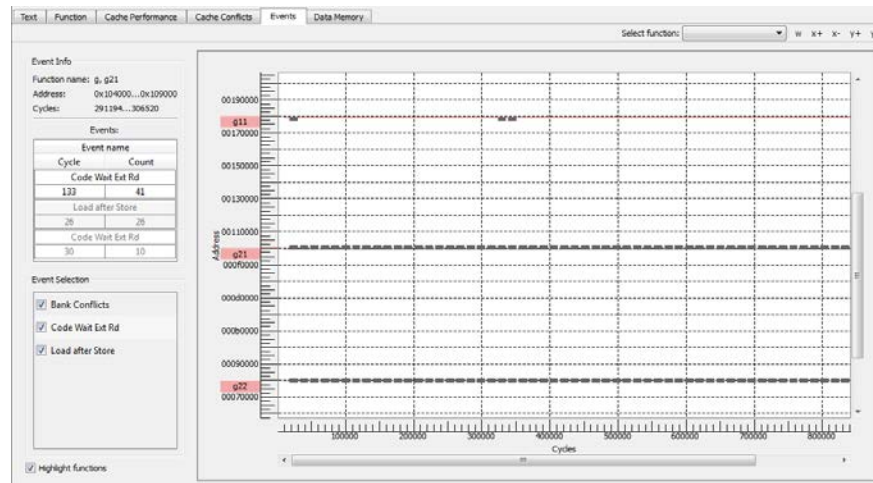


Figure 2-12: Event Viewer – the Events Tab

Function names and ranges can be displayed along the vertical axis – if the **Highlighted functions** checkbox is checked. The Events tab also includes an event selection that enables filtering the events so that only events that correspond to the selected function(s) are displayed. The function browser includes a checkbox next to each function that enables selecting whether the events of this function should be displayed. Clicking the **Filter** button changes the display accordingly. The events of all functions can be displayed by clicking the **Check all** button.

2.6.7 Data Memory Tab

The Data Memory tab displays data memory access counts in a graph. The horizontal axis represents the data memory address range. The vertical axis represents the access counts of the data memory locations. This graph indicates how many times each data memory location was accessed (read access, write access or both). See Figure 2-12:

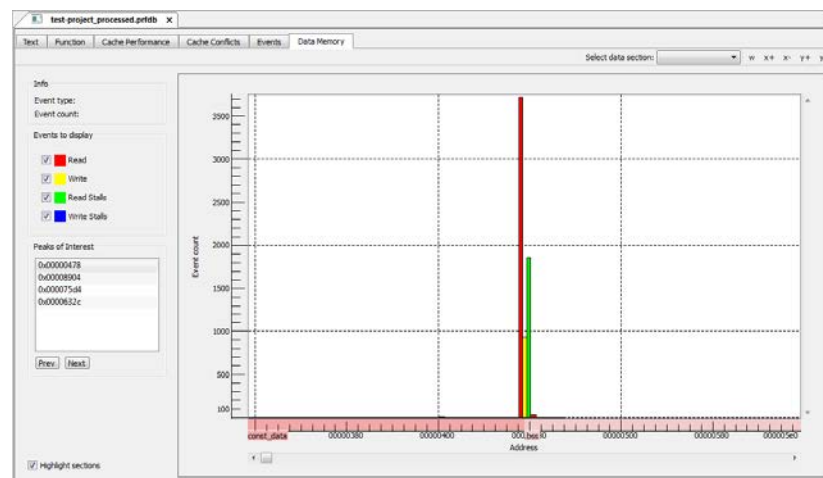


Figure 2-13: Event Viewer – the Data Memory Tab

Section names and ranges can be displayed along the horizontal axis – if the **Highlighted sections** checkbox is checked. The data memory tab also includes a section browser that allows the user to jump quickly to a desired section.

2.7 Configuring the Profiler Settings

The appearance of the Profiler windows and the data that they display can be configured using the **Profiler->Settings** dialog box of the IDE. This dialog box includes a Profiler category. This category includes the following sub categories:

- **Call Graph** – this category enables configuring the Call Graph window
- **Table View** – this category enables configuring all the tables in the Profiler

Figure 2-14 shows the Settings dialog box of the IDE, displaying the Call Graph window settings:

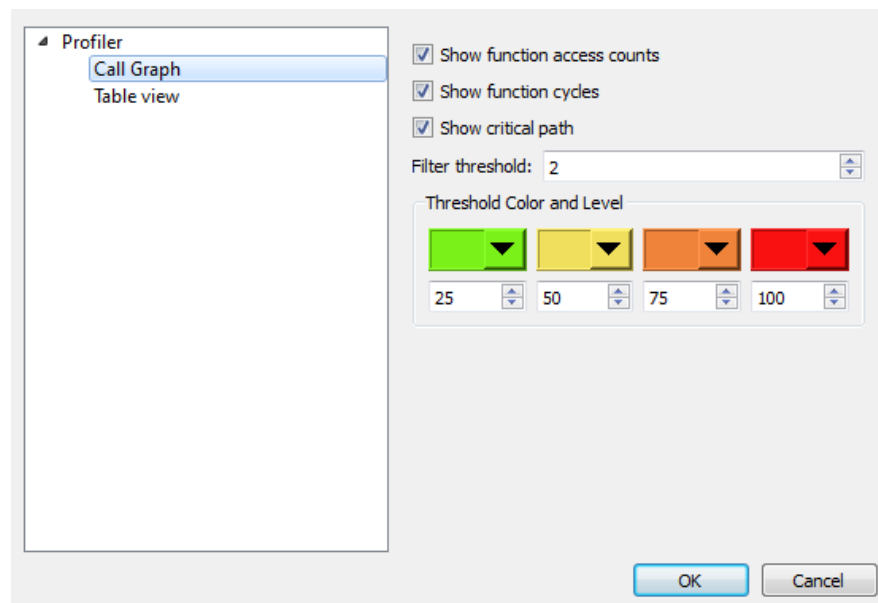


Figure 2-14: Call Graph Window Settings

The following sections describe the Profiler settings categories.

2.7.1 Call Graph Settings

The following settings are supported for the Call Graph window:

Checkboxes:

- **Show function access counts** – adds the access count above each line
- **Show function cycles** – adds the function cycle count above each function rectangle
- **Show critical path** – mark on the call graph the critical path of the application
- **Filter threshold** – set the default threshold of the call graph

Color definitions:

- **Define highlight color** – sets the color for each threshold
- **The threshold for each color** – sets the threshold of maximum exclusive cycle count for each function and the color assigned to it

2.7.2 Table View Settings

The Table View setting allows the user to customize the display of the tables in the Profiler. If the **System settings** check box is selected, all the tables are set to the default settings.

3 Application Profiling Flow

By default, the Profiler is disabled. The following steps should be taken when using the Profiler:

- Enable the Profiler.
- Run the application and collect results.
- Advance users

3.1 Enable the Profiler

If the user wishes to run the application with the Profiler he needs to activate it.

There are two ways to run with profiling:

1. **Automatic configuration** – The profiler can be turned on through the profiler menu by selecting Enable Profiler. This can also be done through the project settings under debug, by checking the enable profiler checkbox.

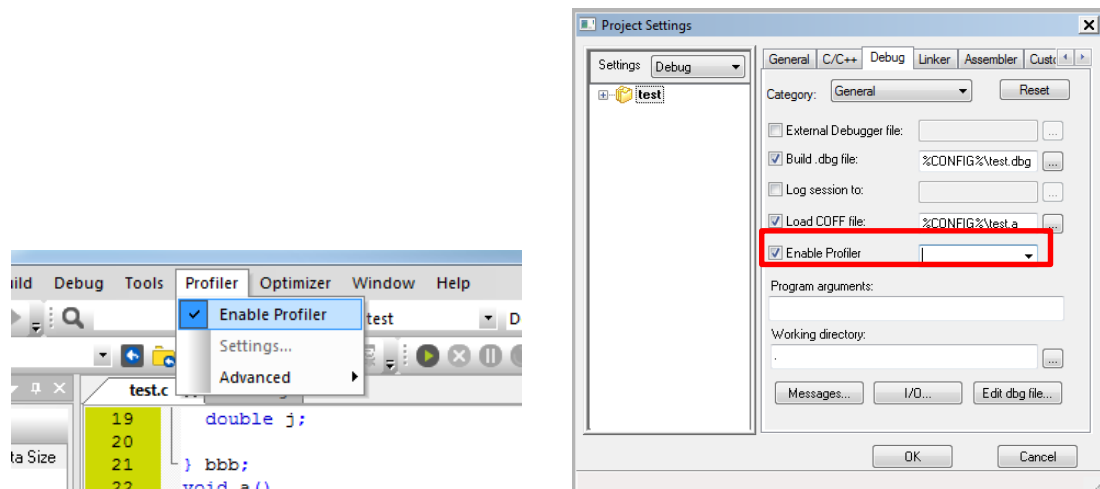


Figure 3-1: How to Activate the Profiler

2. **Manual configuration** – The user can select to use an external debugger file instead of the automatic Build '.dbg' option. In this case the '.dbg' file must contain the following CLIs in order for the profiler to run:

```
load coff <Name Of the COFF>.a
start memory profiler
```

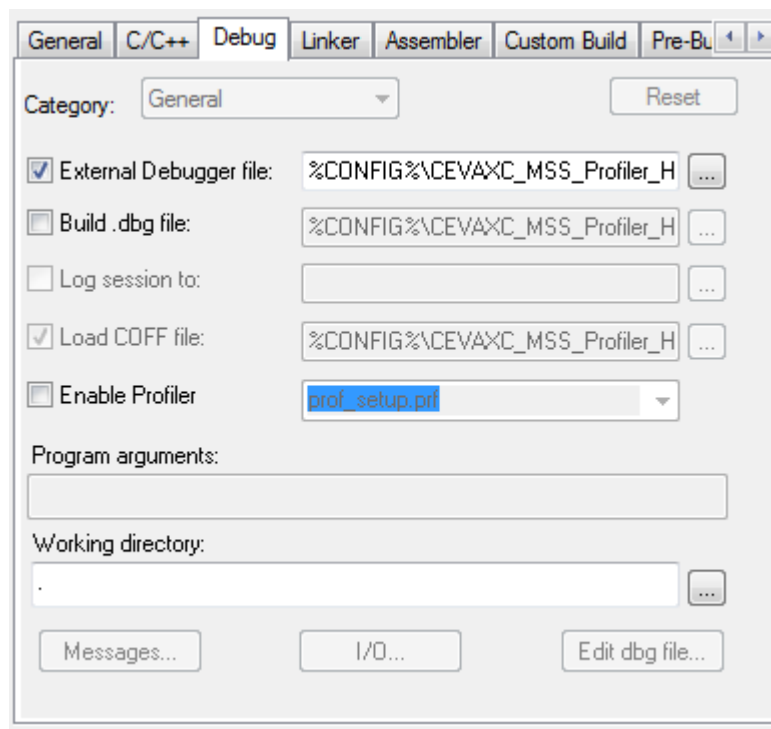


Figure 3-2: How Checkboxes should be set for Manual Setting

3.2 Run the Application and Collect Results

When the profiler is enabled, the user can run the application.

After the execution is complete and the debugger stops, the results of the profiler can be viewed. The way to see the results is affected by the way the user runs the profiling:

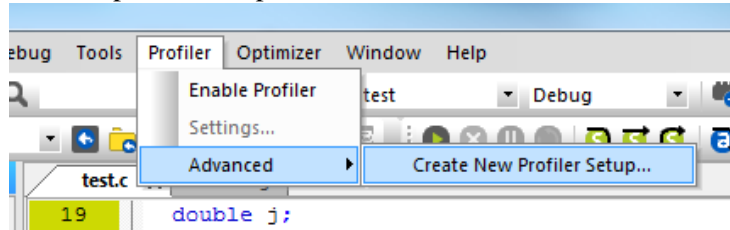
1. **Automatic configuration** – Once the abort button is pressed and the user exits debug mode, the profiler results will automatically open.
2. **Manual configuration** – When the user exits debug mode, the profiler results will not open automatically. The user must open the 'prfdb' file in the configuration directory manually. This can be done by:
 - a. Double-clicking the file. A new IDE will open with the profiler results.
 - b. Dragging the 'prfdb' into an open IDE and the profiler results will open there.

3.3 Advanced Users

Advanced users can profiler the application using a setup file, for more help on setup files please refer to section [1.3 Setup File](#).

In order to run with a setup file, follow these steps:

1. Create a profiler setup file



2. In the profiler setup file, set the parameters of the profiler run.
3. Run the application.
4. Exit debug.
5. The profiler Result will open.

The user can set several setup files and switch between them by selecting them in the Profiler->Advance menu.

4 Profiler Export Command Line

The user can use command lines in order to export tables after a profiling session is done.

In order to do this the profiler standalone must be used.

Usage:

```
profilerstandalone [<database>] [{--export|-e} <export key>] [{--outfile|-o} <out filename>]
```

Keys:

--help -h Show help screen

--export -e Exports CSV formatted data to stdout. Valid export keys (long and short form):

- function_viewer f
- branch_viewer b
- source_viewer s
- events_text e
- cache_conflicts c

--outfile -o Out to specified file instead of stdout

5 CEVA-XM4 Emulation Profiler Counters

The following table is a comparison between the Emulation profiler counters and their matching profiler events in the simulation profiler:

OCEM Counter Name	SDT Profiler Event	Description	Remarks
FRCC	Clock	Free Running Clock Counter. Counts the number of core cycles	Cycle count per function/application
WAIT_CNT		Wait counter. Counts the number of wait cycles asserted by any source to the Core.	Missing from the profiler as an event but can be calculated from all other wait events.
VPU_STL_CNT	"Load after store"	VPU stall counter Counts the number of stall VPU signals due to read after write.	
DBLK_CONF_CNT	"Block conflicts"	DMSS block conflict counter. Counts the number of conflicts between the core, write-buffer and the EDAP performed on DMSS.	Incremented in case of a wait event.
OSFC	"Write buffer full"	Output stage full counter. Counts the number of wait caused by write buffer full.	
TDRC	"Data Read Internal"	TCM DMSS read counter. Counts the number of TCM accesses due to any kind of a read transaction (Is0/1 EDAP and DMA). Aborted fetches are not counted. TCM writes are not counted.	

OCEM Counter Name	SDT Profiler Event	Description	Remarks
TDWC	"Data Write Internal"	TCM DMSS write counter. Counts the number of TCM accesses due to any kind of a write transaction (Is0 EDAP and DMA). Aborted fetches are not counted. TCM writes are not counted.	
ERWC	"Data Wait External Read"	EDP read wait counter. Counts the number of cycles for which the core is held in wait due to an external data memory read access.	
VPU0_NOP_INST	"nop instruction"	VPU0 nop instruction. Counts the number of nop instructions in VPU0. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.	Shown as total number of nops in the application (not divided to units)
VPU1_NOP_INST	"nop instruction"	VPU1 nop instruction. Counts the number of nop instructions in VPU1. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.	Shown as total number of nops in the application (not divided to units)

OCEM Counter Name	SDT Profiler Event	Description	Remarks
PCU_NOP_INST	"nop instruction"	PCU nop instruction. Counts the number of nop instructions in PCU. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.	Shown as total number of nops in the application (not divided to units)
SPU0_NOP_INST	"nop instruction"	SPU0 nop instruction. Counts the number of nop instructions in SPU0. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.	Shown as total number of nops in the application (not divided to units)
SPU1_NOP_INST	"nop instruction"	SPU1 nop instruction. Counts the number of nop instructions in SPU1. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.	Shown as total number of nops in the application (not divided to units)
SPU2_NOP_INST	"nop instruction"	SPU2 nop instruction. Counts the number of nop instructions in SPU2. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.	Shown as total number of nops in the application (not divided to units)

OCEM Counter Name	SDT Profiler Event	Description	Remarks
SPU3_NOP_INST	"nop instruction"	SPU3 nop instruction. Counts the number of nop instructions in SPU3. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.	Shown as total number of nops in the application (not divided to units)
LSU0_NOP_INST	"nop instruction"	LSU0 nop instruction. Counts the number of nop instructions in LSU0. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.	Shown as total number of nops in the application (not divided to units)
LSU1_NOP_INST	"nop instruction"	LSU1 nop instruction. Counts the number of nop instructions in LSU1. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.	Shown as total number of nops in the application (not divided to units)
PMSS_HIT_CNT	"Code Cache Read Hit" + "Code Cache Write Hit"	PMSS hit counter. Counts the number of PMSS cache hits.	2 counters in the SDT profiler
PMSS_MIS_CNT	"Code Cache Read Miss" + "Code Cache Write Miss"	PMSS miss counter. Counts the number of PMSS cache misses.	2 counters in the SDT profiler
PWCWC	"Block conflicts"	Parallel write contention wait counter. Counts the number of wait cycles caused by parallel write contention.	Counted on the same "block conflict" profiler event.

OCEM Counter Name	SDT Profiler Event	Description	Remarks
P_ADD_COUNTX		PC address range counter X Counts the number of times the pc was between P_LOW_ADD and P_HI_ADD * X is 0 to 7	This data is shown in the profiler by other means, function call count for example. Also can be extracted from trace.
Counters in SDT profiler that are not in the OCEM			
	"Data Read External"	Read counter from external memory	Counter of events, not waits as above
	"Data Write External"	Write counter to external memory	
	"Code Wait External Read"	Wait cycles counter due to code external read	
	Interrupts events	Counts the number of interrupts according to int type	Split counters, 1 per interrupt type