



Freescale Semiconductor, Inc.

Debugger HCS08 Onchip DBG Module User Interface

Revised 10/16/2003



For More Information: www.freescale.com



Freescale Semiconductor, Inc.

Metrowerks, the Metrowerks logo, and CodeWarrior are registered trademarks of Metrowerks Corp. in the US and/or other countries. All other tradenames and trademarks are the property of their respective owners.

Copyright © Metrowerks Corporation. 2003. ALL RIGHTS RESERVED.

The reproduction and use of this document and related materials are governed by a license agreement media, it may be printed for non-commercial personal use only, in accordance with the license agreement related to the product associated with the documentation. Consult that license agreement before use or reproduction of any portion of this document. If you do not have a copy of the license agreement, contact your Metrowerks representative or call 800-377-5416 (if outside the US call +1-512-996-5300). Subject to the foregoing non-commercial personal use, no portion of this documentation may be reproduced or transmitted in any form or by any means, electronic or mechanical, without prior written permission from Metrowerks.

Metrowerks reserves the right to make changes to any product described or referred to in this document without further notice. Metrowerks makes no warranty, representation or guarantee regarding the merchantability or fitness of its products for any particular purpose, nor does Metrowerks assume any liability arising out of the application or use of any product described herein and specifically disclaims any and all liability. **Metrowerks software is not authorized for and has not been designed, tested, manufactured, or intended for use in developing applications where the failure, malfunction, or any inaccuracy of the application carries a risk of death, serious bodily injury, or damage to tangible property, including, but not limited to, use in factory control systems, medical devices or facilities, nuclear facilities, aircraft navigation or communication, emergency systems, or other applications with a similar degree of potential hazard.**

USE OF ALL SOFTWARE, DOCUMENTATION AND RELATED MATERIALS ARE SUBJECT TO THE METROWERKS END USER LICENSE AGREEMENT FOR SUCH PRODUCT.

How to Contact Metrowerks

Corporate Headquarters	Metrowerks Corporation 7700 West Parmer Lane Austin, TX 78729 U.S.A.
World Wide Web	http://www.metrowerks.com
Sales	Voice: 800-377-5416 Fax: 512-996-4910 Email: sales@metrowerks.com
Technical Support	Voice: 800-377-5416 Email: support@metrowerks.com

For More Information: www.freescale.com



Table of Contents

1 Debugger HCS08 Onchip DBG Module User Interface	3
Introduction	3
Reference document	3
Features.	3
Specific target menu entries	4
Specific context sensitive popup menu entries in Source, Data, Assembly and Memory component windows.	5
Source and Assembly windows	5
Triggers storing as markpoints.	8
Data and Memory windows	10
Expert triggers	12
Trigger Settings menu entry.	15
Trigger Module Usage menu entry	16
Specific Status Bar item	16
Trigger Module Settings dialog	17
DBG Module mode setup.	17
Automatic mode (default)	18
Expert mode	19
Expert mode tab	20
Profiling and Coverage mode	20
Disabled mode	21
“Memory access” triggers	22
“Instruction” triggers	24
“Capture” triggers	26
DBG module options	27
Trigger Address Edition	29
Dialog information	31
General Settings	32
Trace window component.	34
Limitations	40



Index	41
--------------	-----------

Debugger HCS08 Onchip DBG Module User Interface

Introduction

The HCS08 derivatives featuring an onchip DBG module require a debugger graphical user interface to setup this module and take full advantage of this enhanced debugging feature. This manual describes the debugger DBG module user interface.

Within several HCS08 debugger host target interfaces (e.g. P&E (PEDebug) Target Interface, HCS08 Serial Monitor and inDART-HCS08 via GDI Target Interface), a complete graphical user interface is provided, through a trigger setup dialog combined with context sensitive popup menus (mouse right-click) in Source, Assembly, Data and Memory component windows to set the onchip DBG module and triggers.

This DBG module support is automatically enabled or disabled, according to user selected derivative (if the device is user configurable) or automatically through device Part Id.

Reference document

The HCS08 onchip DBG module related overall this manual is described in:

-“ *HCS08RMv1/D, Rev. 1, 6/2003*” Motorola document (latest version at this manual release time).

Features

The debugger covers all features available within the onchip DBG module:

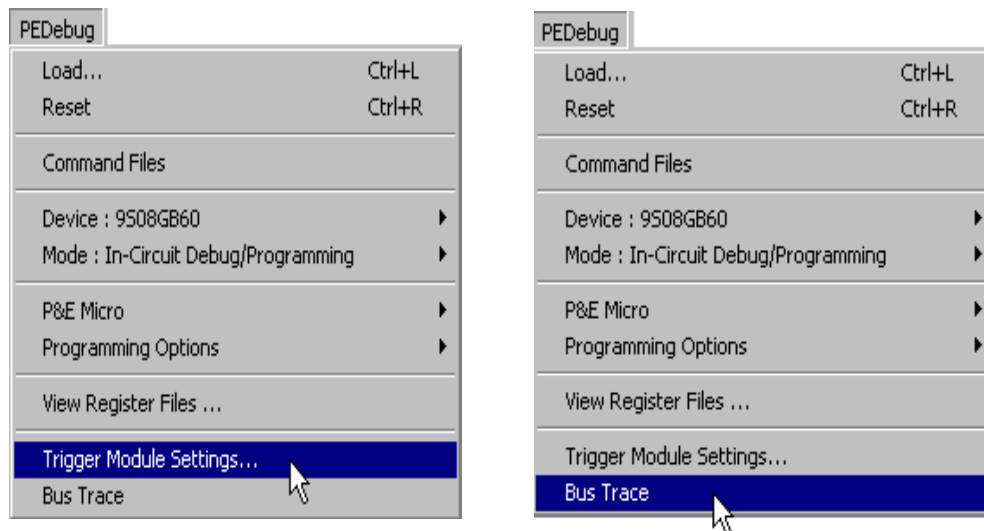
- Regular hardware breakpoints and watchpoints,

- Predefined preset [“Instruction” triggers](#), [“Memory access” triggers](#) or [“Capture” triggers](#), a wide set of complex hardware breakpoints (triggers on program code instructions) and watchpoints (triggers on device memory access) and data bus recording,
- Expert Triggers, as powerful as predefined preset triggers, “Do It Yourself” way,
- Code program flow rebuild from DBG data capture within the Trace window component (the Trace component should be opened to display the code program flow rebuild),
- Real time program code profiling and coverage within the Profiler and Coverage window components (the Profiler and/or the Coverage components should be opened to display code profiling and code coverage)

Specific target menu entries

Specific DBG support menu entries are present as soon as the target processor features the DBG module.

Two additional popup menu entries are displayed: **“Trigger Module Settings”** and **“Bus Trace”** in the target menu (here below, an example with the P&E (PEDebug) target interface).



Choose **“Trigger Module Settings”** to open the [Trigger Module Settings dialog](#).

Choose **“Bus Trace”** to open the [Trace window component](#).

Specific context sensitive popup menu entries in Source, Data, Assembly and Memory component windows

Specific DBG support context sensitive popup menu entries are present as soon as the target processor features the DBG module.

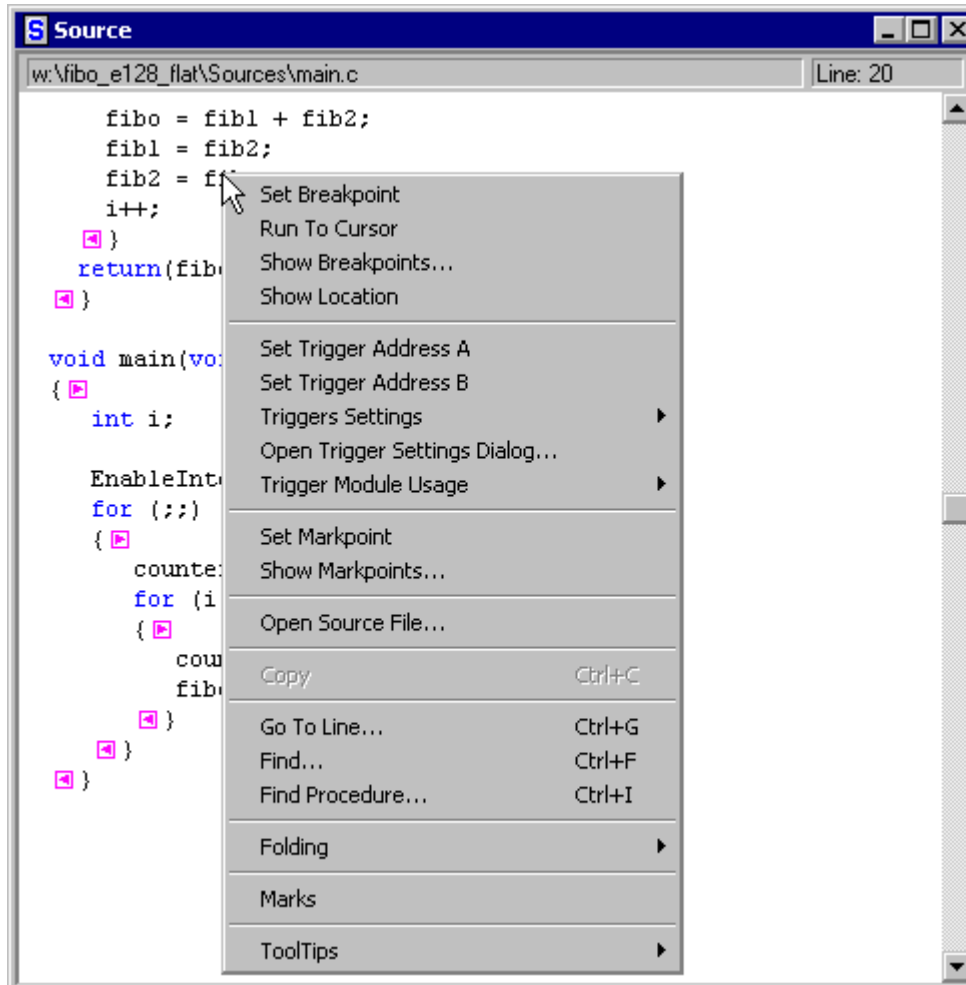
Source and Assembly windows

Source and Assembly windows have menu entries to set/delete [“Instruction” triggers](#) A and/or B, a [Trigger Settings menu entry](#) to set the DBG module Triggers Settings and the [Trigger Module Usage menu entry](#) to set globally the DBG module functionality. From the user point of view, setting a trigger - which can be assimilated as a complex breakpoint or watchpoint - is as simple as setting a breakpoint.

Freescal Semiconductor, Inc.

Debugger HCS08 Onchip DBG Module User Interface

Specific context sensitive popup menu entries in Source, Data, Assembly and Memory component



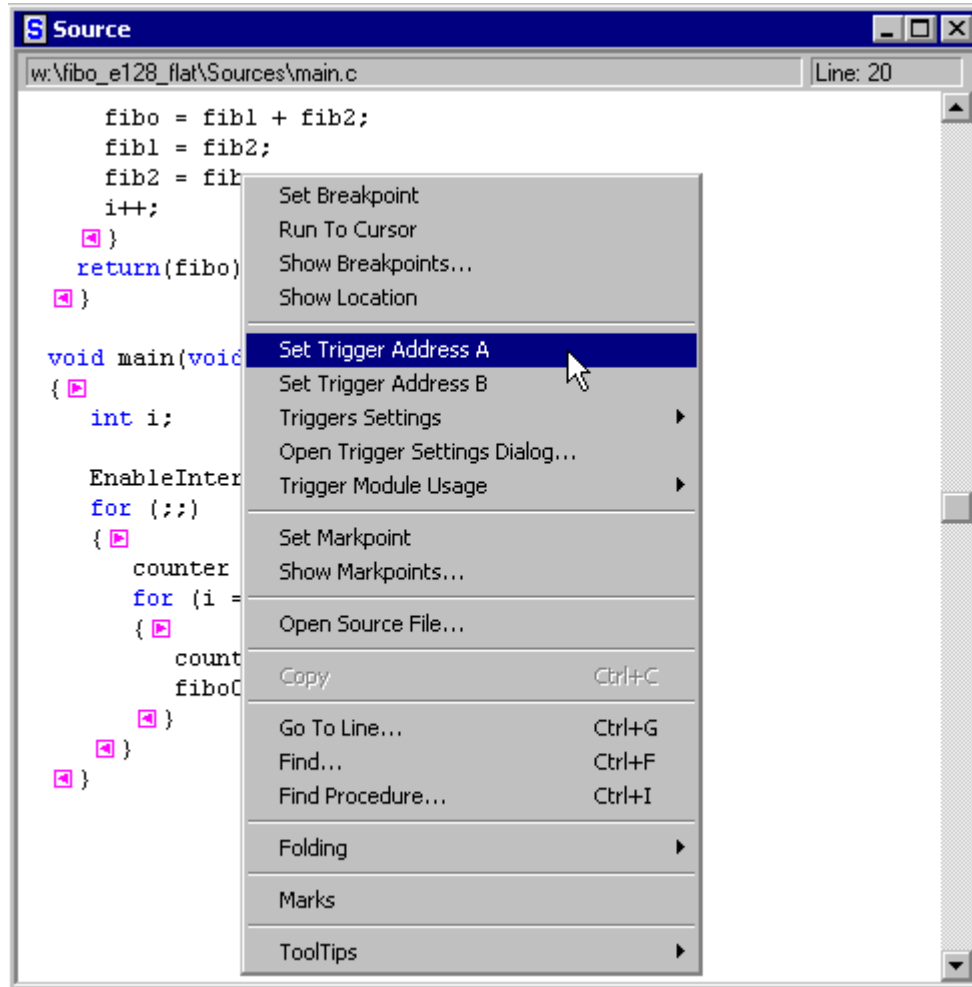
Instead of setting a breakpoint, a trigger can be set. Note that only 2 triggers can be set: **Trigger A** and **Trigger B**. In a general way, the onchip DBG module provides combinations of trigger A and trigger B conditions, and according to the amount of trigger defined (one or two), different trigger [DBG Module mode setup](#) can be chosen.

To set a trigger, choosing a **Set TriggerAddress** entry sets a trigger at the select source location/address.

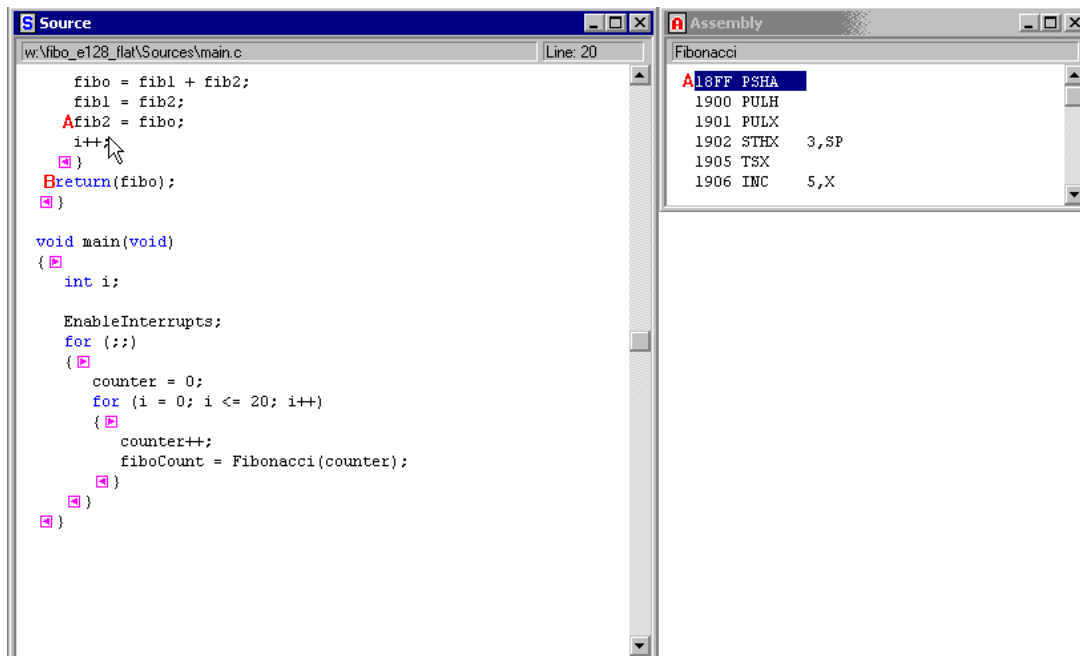
Freescale Semiconductor, Inc.

Debugger HCS08 Onchip DBG Module User Interface

Specific context sensitive popup menu entries in Source, Data, Assembly and Memory component



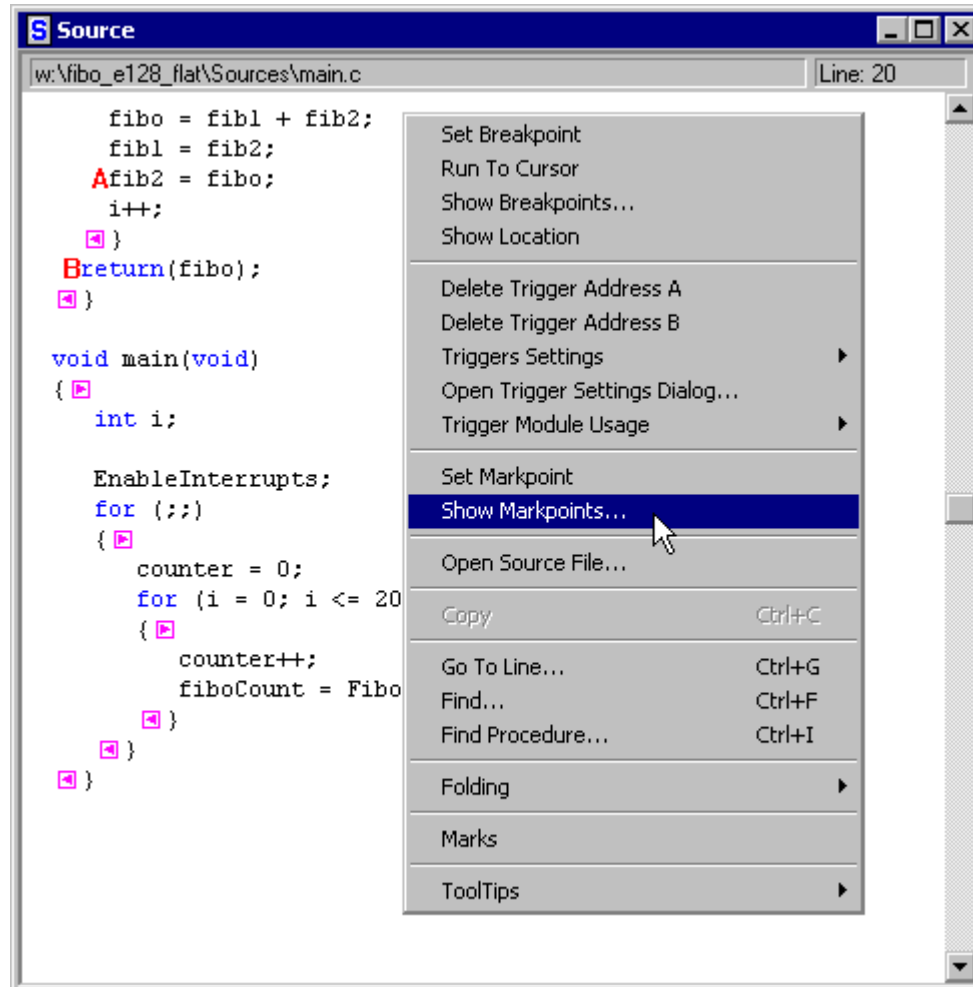
The trigger is displayed in the Source window and the corresponding address in the Assembly window like a breakpoint icon. To be differentiated with breakpoints, the trigger A is marked with a red “A” icon and trigger B with a red “B” icon.



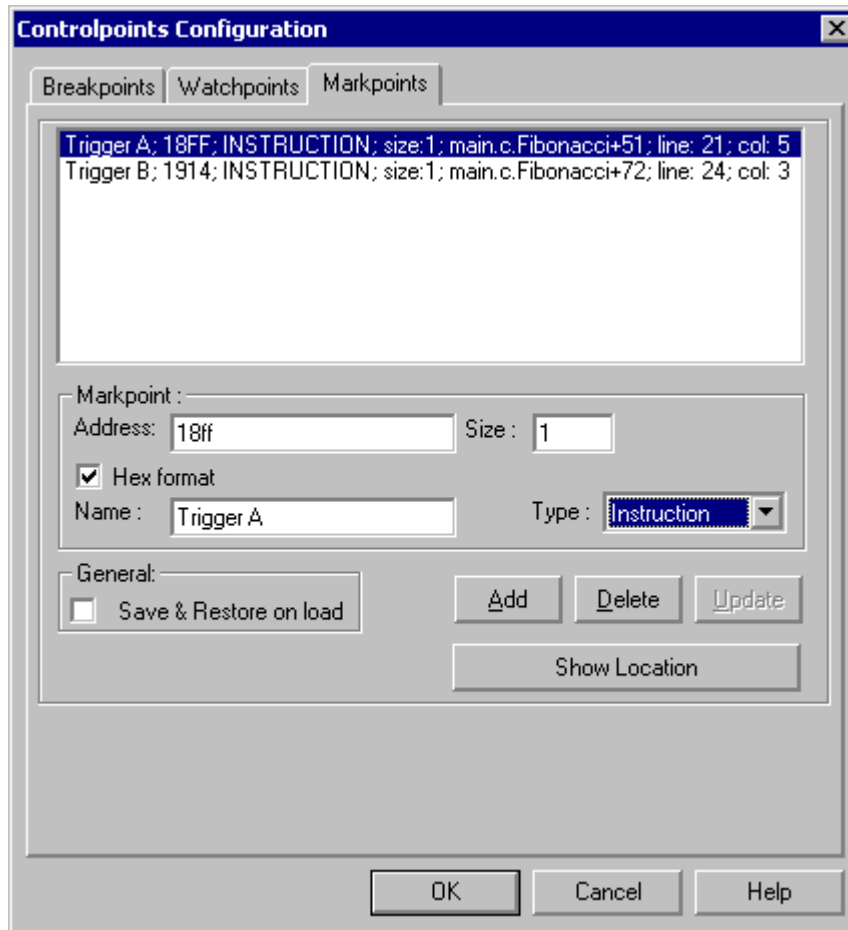
Triggers storing as markpoints

Once a trigger is set, it can easily be deleted when opening from any place the context sensitive popup menu with the *Delete Trigger Address* entries.

Also triggers are stored in the debugger as special markpoints. Like breakpoints, markpoints can be viewed on choosing *Show Markpoints...* in the menu.



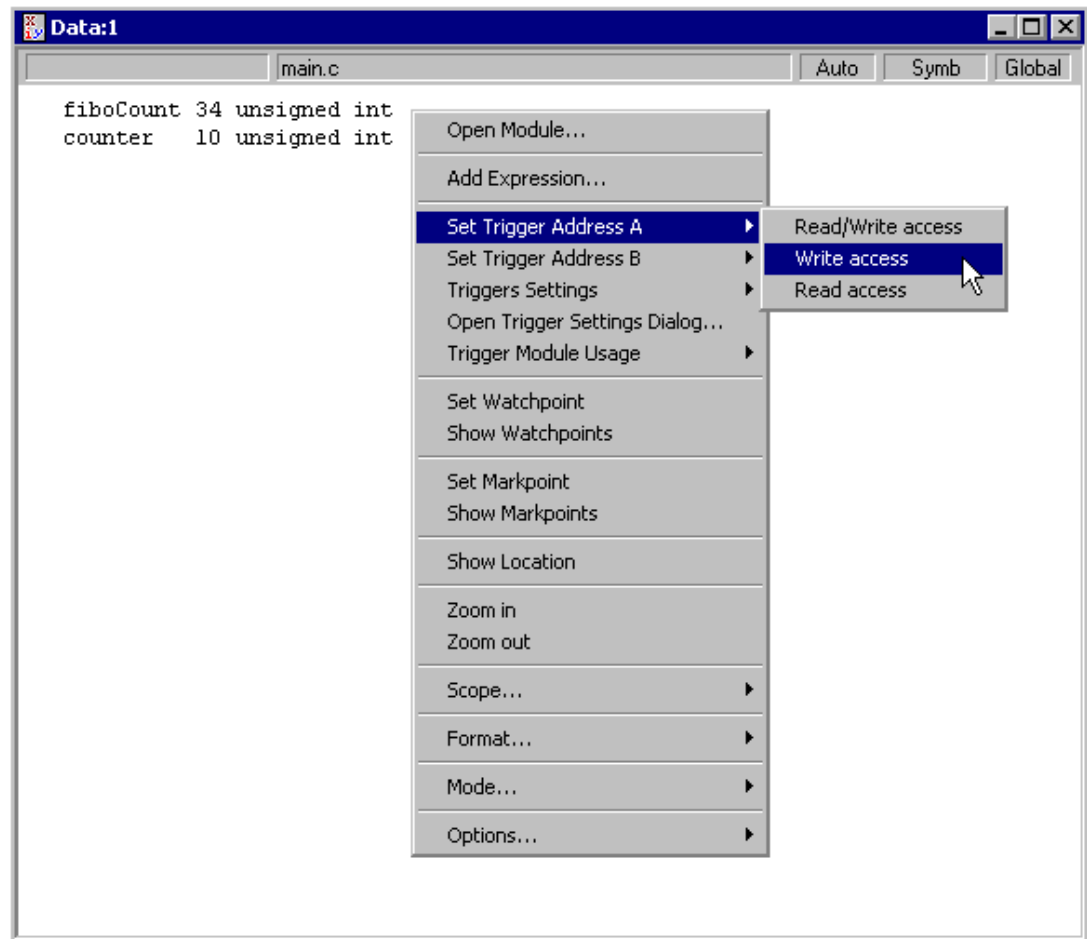
Triggers are stored as “*Trigger A*” and “*Trigger B*” markpoints. These markpoint names are therefore reserved by the debugger. The markpoint type “**INSTRUCTION**” is automatically selected when the trigger was set from the Source or the Assembly window. Editing triggers through the *Markpoints* tab in the *Controlpoints Configuration* dialog below is not user friendly. However, the “*Save and Restore on load*” option (also available with breakpoints and watchpoints) can be very useful to automatically save with the application the current DBG module setup and trigger positions in the current application.



Data and Memory windows

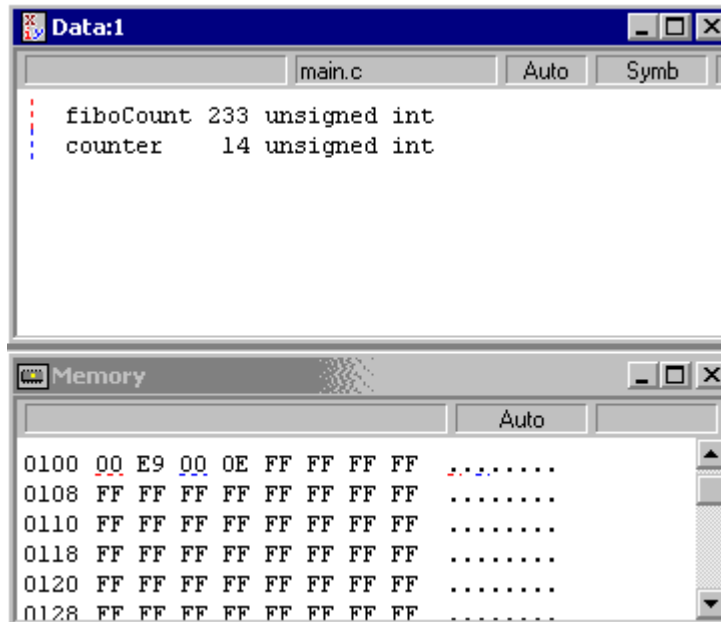
Data and Memory windows have menu entries to set/delete [“Memory access” triggers](#) A and/or B, a [Trigger Settings menu entry](#) to set the DBG module triggers settings and the [Trigger Module Usage menu entry](#) to set globally the DBG module functionality.

From the user point of view, setting a trigger - which can be assimilated as a complex breakpoint or watchpoint - is as simple as setting a watchpoint.



Instead of setting a watchpoint, a trigger can be set. Note that only 2 triggers can be set: Trigger A and trigger B. In a general way, the onchip DBG module provides combinations of trigger A and trigger B conditions, and according to the amount of trigger defined (one or two), a different trigger [DBG Module mode setup](#) can be chosen.

To set a trigger, choosing a *Set Trigger Address* entry and immediately the kind of access - **Read, Write, Read/Write** - sets a trigger at the select place.



The trigger is displayed in the Data window and the corresponding address in the Memory window like a watchpoint icon. To be differentiated with watchpoints, the trigger A is marked with a red dotted vertical line and trigger B with a blue dotted vertical line.

Expert triggers

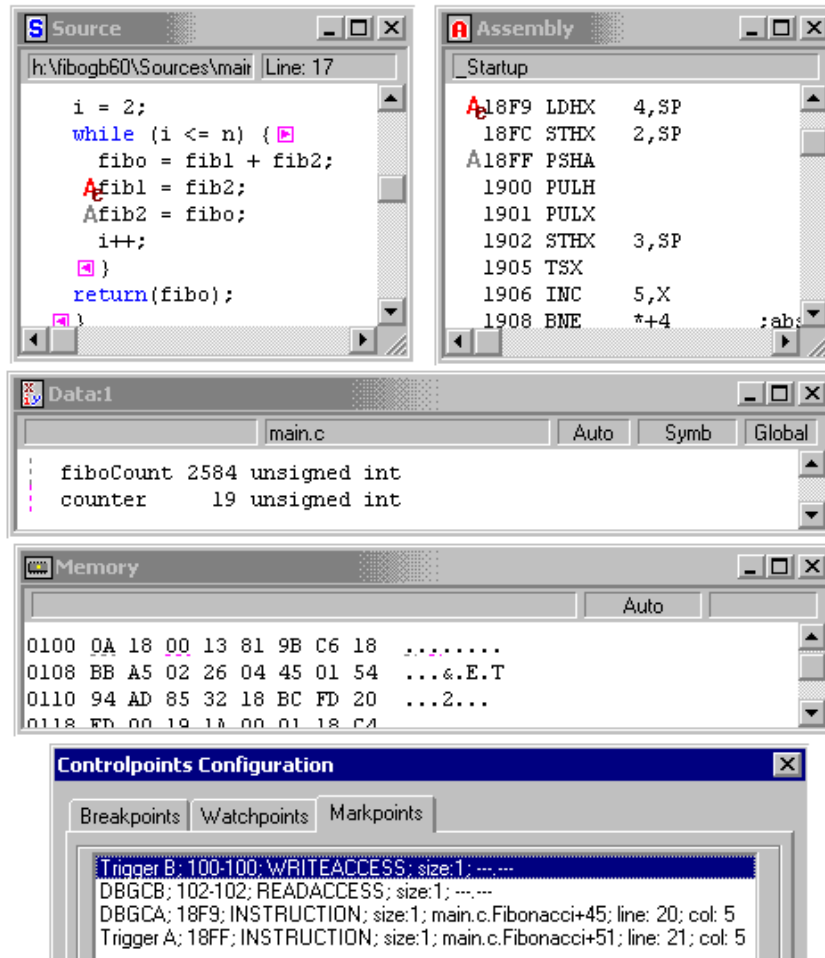
The [Expert mode](#) has a different set of triggers and trigger designs. Indeed, to completely separate the Expert mode from the Automatic mode, the debugger provides a second set of triggers for the Expert mode. [Expert triggers](#) are independent from regular triggers described previously.

Context sensitive popup menu entries are slightly different, basically replacing the “*Set Trigger Address A*” entry by a “*Set DBGCA*” entry and the “*Set Trigger Address B*” entry by a “*Set DBGCB*” entry. The renaming is due to a more physical DBG registers approach in Expert mode and in the [Expert mode tab](#).



As shown on the next picture, Expert triggers are displayed in Source and Assembly windows with a small additional “e” character and different colors in the Memory component.

NOTE When the Expert mode is set, preset [“Instruction” triggers](#), [“Memory access” triggers](#) or [“Capture” triggers](#) designs are grayed. When the automatic mode is set or a predefined preset trigger is set, the Expert mode trigger designs are grayed.



Expert triggers are stored as “**DBGCA**” and “**DBGCB**” markpoints. These markpoint names are therefore reserved by the debugger.

The markpoint type “**INSTRUCTION**” is automatically selected when the trigger was set from the Source or the Assembly window.

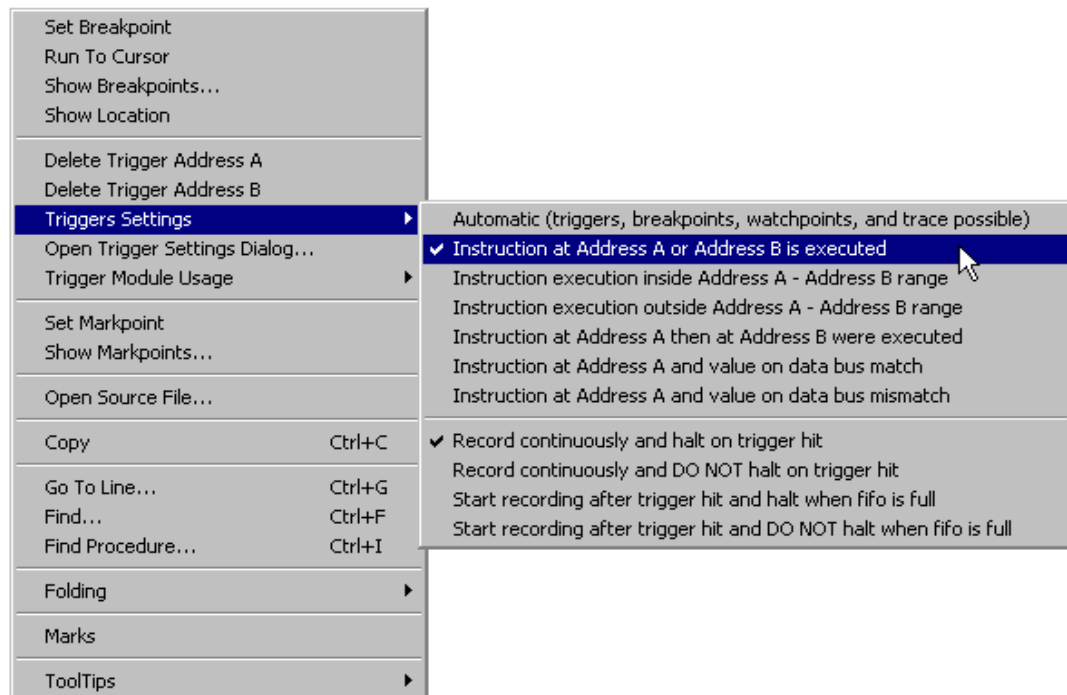
As for regular triggers, the markpoint types “**READACCESS**”, “**WRITEACCESS**” or “**READWRITEACCESS**” are automatically selected when the trigger was set from the Data or the Memory window.

Editing triggers through the *Markpoints* tab in the *Controlpoints Configuration* dialog below is not user friendly. However, the “*Save and Restore on load*” option (also available with breakpoints and watchpoints) can be very useful to automatically save with the application the current DBG module setup and trigger positions in the application.

Trigger Settings menu entry

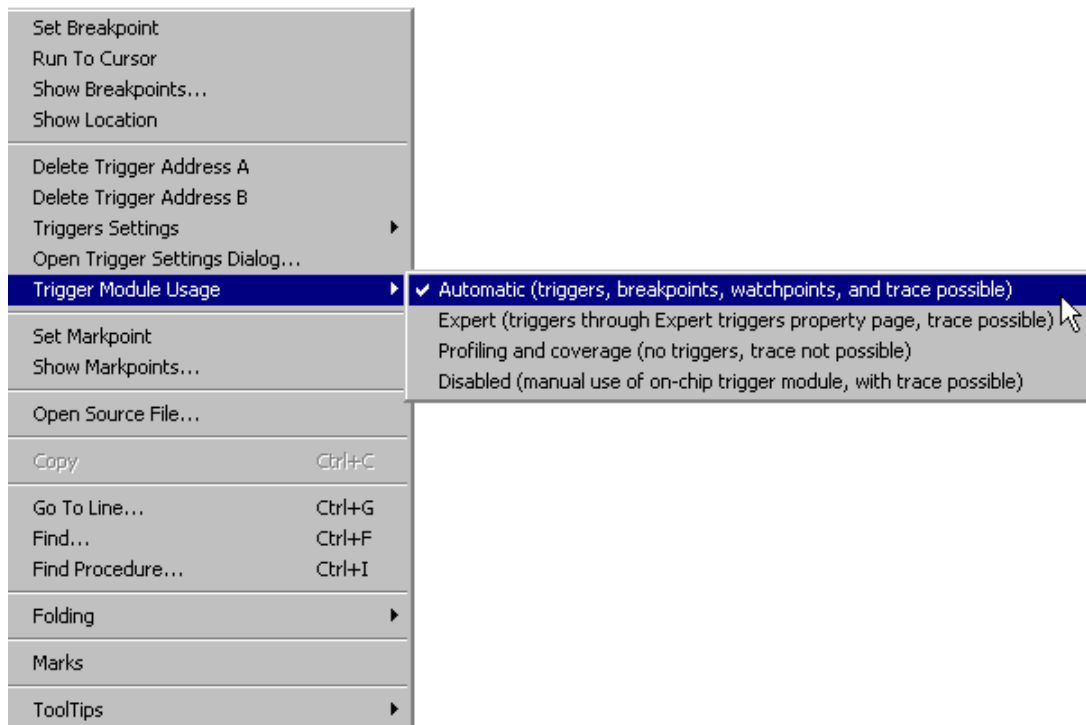
This menu entry can be chosen to set all kind of triggers without opening the [Trigger Module Settings dialog](#). However, the amount of trigger types is **dynamic**, dependending if no triggers are defined, if only Trigger A is defined, if only trigger B is defined if both triggers are defined, and also depending on the trigger type (Instruction, Read Access, Read/Write Access, Write Access. **Only possible combinations are displayed.**

Also [DBG module options](#) can be directly edited.



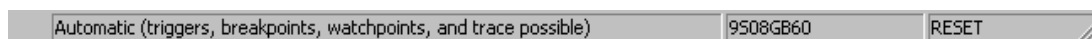
Trigger Module Usage menu entry

This menu entry can be used to set globally the DBG module functionality without opening the [Trigger Module Settings dialog](#) to do the [DBG Module mode setup](#).



Specific Status Bar item

A specific DBG support debugger status bar item is present as soon as the target processor features the DBG module. Clicking on this item opens immediately the [Trigger Module Settings dialog](#) (future debugger revision only).



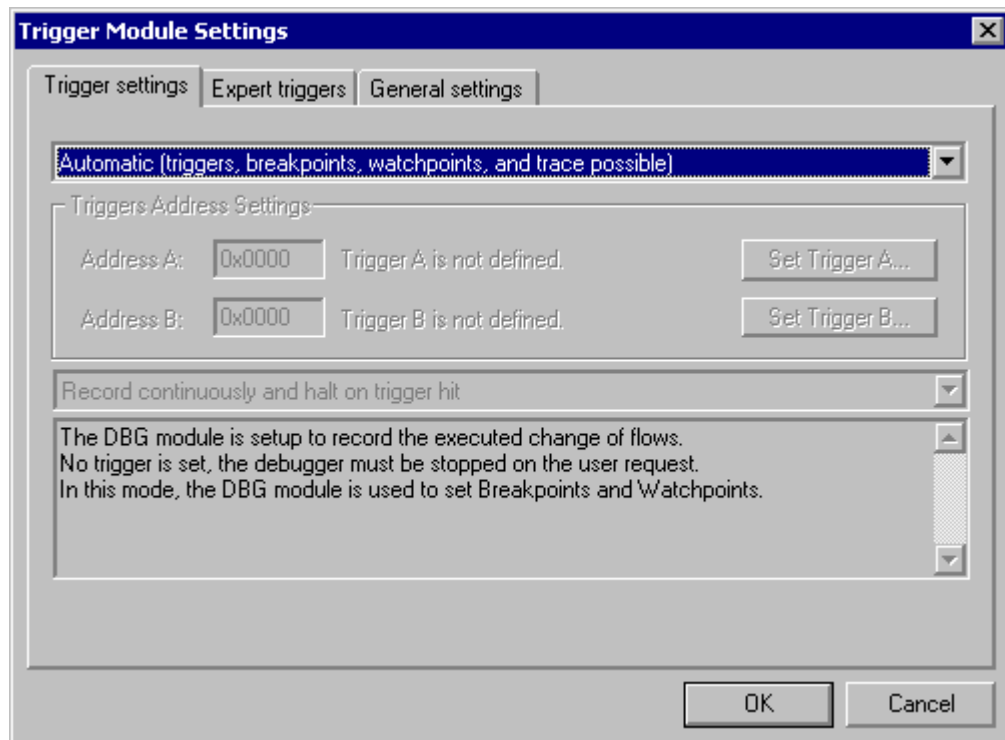
It displays the current [DBG Module mode setup](#) (here above) or the current preset [“Instruction” triggers](#), [“Memory access” triggers](#) or [“Capture” triggers](#) used or the current [DBG Module mode setup](#) (here below).



Trigger Module Settings dialog

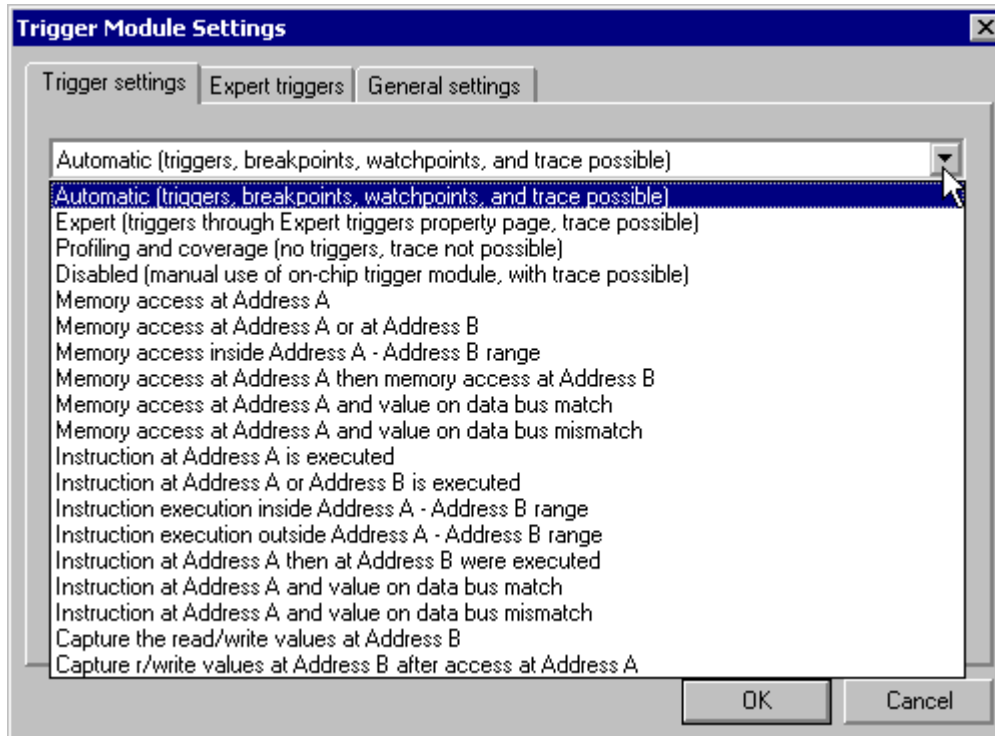
This dialog can be opened from context sensitive popup menus in Source, Data, Memory and Assembly windows, from the target interface menu and also when clicking on the [Specific Status Bar item](#) (future debugger revision only).

The onchip DBG module can be fully controlled within this dialog.



DBG Module mode setup

First of all, the onchip DBG module provides some exclusive debugging features. Open the top drop down list to display all modes and complex breakpoints/wachtpoints i.e. kind of triggers available.



Automatic mode (default)

The DBG Module is used to set up **three** hardware breakpoints or one watchpoint or to set up triggers selected by the user from the list or from a context sensitive popup menu. This mode is simply the default selection when no triggers have been set yet.

The trigger condition and trigger addresses can be set from the debugger Source, Assembly, Memory and Data component using Set Trigger A or Set Trigger B context sensitive pop up menu entry, or within this dialog.

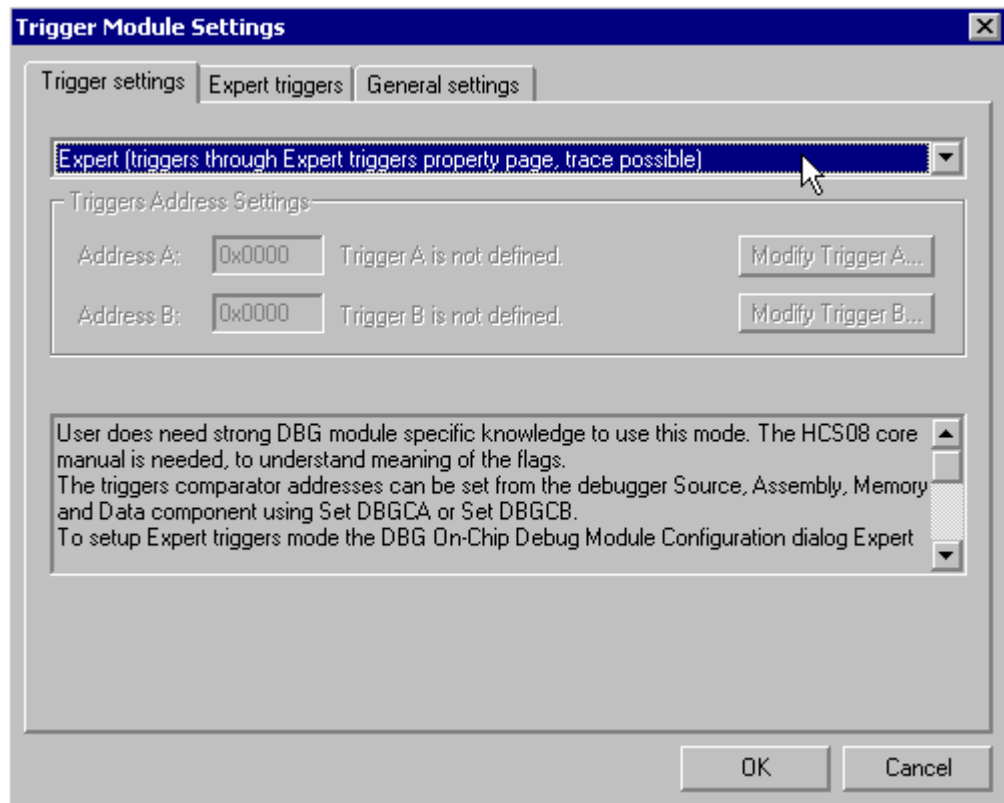
The DBG module is setup to record the executed change of flows. As no triggers are set, the debugger is stopped on the user request or typical breakpoints/watchpoint. To summarize, in this mode, the DBG module is used to set regular hardware breakpoints and watchpoints.

Expert mode

The User needs to know the onchip DBG module to use this mode. It can be seen as a “Do It Yourself” way to set the DBG module. The HCS08 core manual is needed, to understand the meaning of the registers and flags.

The triggers comparator addresses can be set from the debugger Source, Assembly, Memory and Data windows using Set DBGCA or Set DBGCB. The DBG module is set by the debugger. DBG module enabling and arming depend on the selected flags set within the DBG register control registers (through the Expert triggers tab property page). The settings are written to the hardware right before the application is run. The DBG module is reset when the application stops.

To set Expert triggers , the Trigger Module Settings dialog “*Expert triggers*” property page must be used. Select the *Expert* mode in the drop down list to enable the [Expert mode tab](#).



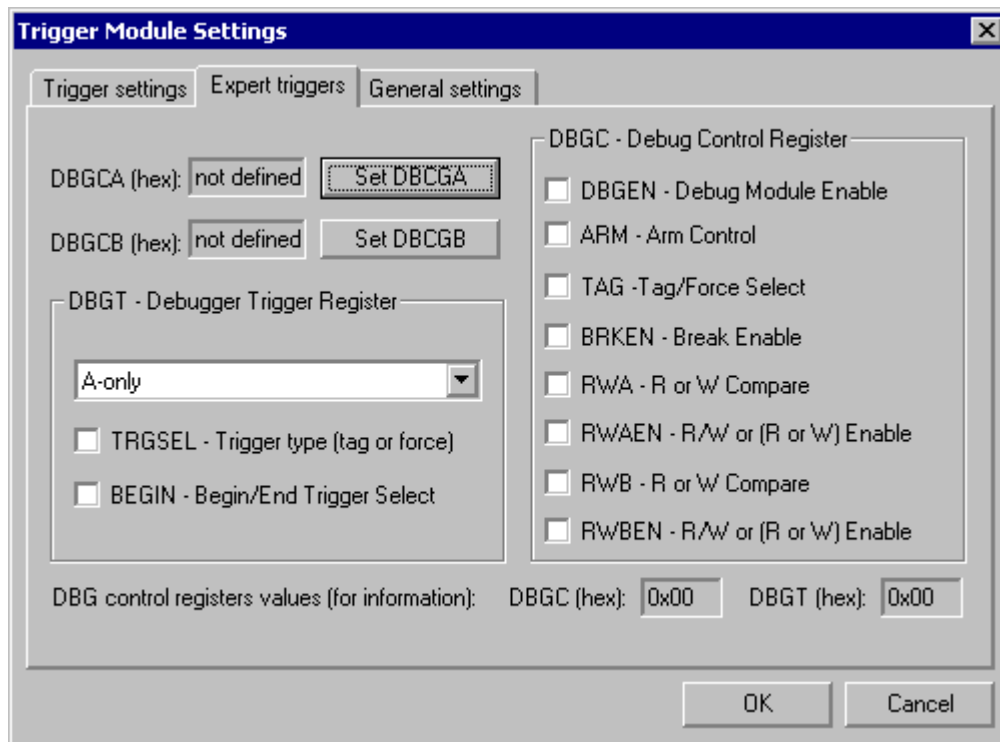
Expert mode tab

The expert mode tab gives an access to most of the onchip DBG module registers.

Please see [Reference document](#) section and documents to get all information about the HCS08 onchip DBG module.

Trigger types can be directly set from the “*DBGT - Debugger Trigger Register*“ drop down list.

Code program flow rebuild and data recording are also synchronized with the Expert mode and results are displayed in the [Trace window component](#).



Profiling and Coverage mode

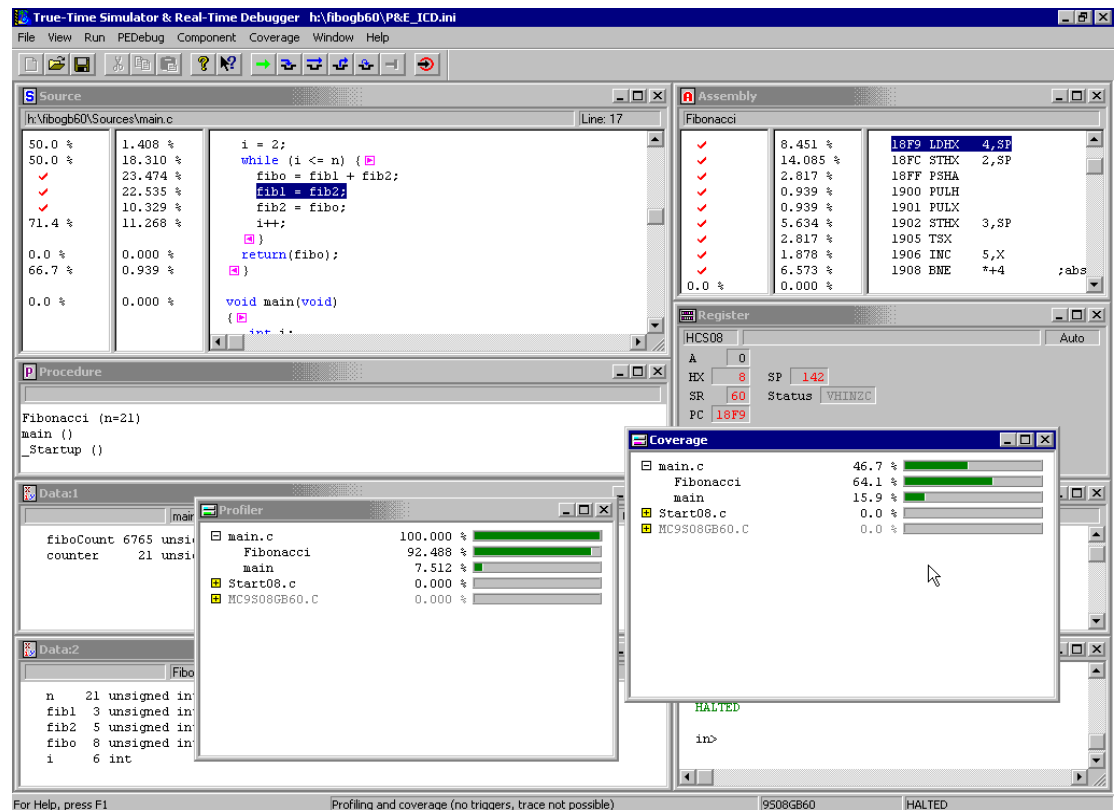
Choosing this mode, the DBG module is setup to source code execution profiling and source code execution coverage . The Profiler and/or Coverage components should be opened to display results.

Neither triggers nor DBG based controlpoints can be set in this mode, and the debugger must be stopped on the user request (software breakpoints can still be used).

Profiling and Coverage features are based on a periodical debugger program counter real time fetch from the debugger to the onchip DBG module. Also this fetch is statistical and cannot cover all program counters and longer is the program running and tests period, more precise would get resulted statistics.

Please see also [Limitations](#) section for this mode.

Please refer to the debugger engine manual for Coverage and Profiler component features.



Disabled mode

The User needs to know the onchip DBG module to use this mode. It can be seen as a “Do It Yourself” way to set hardware breakpoints , watchpoints and triggers.

Please see [Reference document](#) section and documents to get all information about the HCS08 onchip DBG module.

There is no dedicated graphical user interface to access DBG module register. The triggers comparator addresses and DBG control registers are handled by the user

through the debugger Memory component or using command line commands. The DBG module is NOT set by the debugger. DBG module enabling and arming depend on the selected flags set within the DBG register control registers. The DBG module is NOT reset when the application stops. By default, the FIFO content is protected from unexpected reads, the DBG module is automatically disarmed and the FIFO is analyzed when the debugger stops. This can be optionally disabled by the user.

“Memory access” triggers

Memory access at Address A

This mode is used to trigger on a program instruction read and/or write at Address A memory location.

The code program flow rebuild is displayed in the [Trace window component](#) automatically switched to [Instructions display](#) mode.

Memory access at Address A or at Address B

This mode is used to trigger on a program instruction read and/or write at Address A or at Address B memory location.

The code program flow rebuild is displayed in the [Trace window component](#) automatically switched to [Instructions display](#) mode.

Memory access inside Address A - Address B range

This mode is used to trigger on a program instruction read and/or write inside the Address A - Address B memory range locations.

The code program flow rebuild is displayed in the [Trace window component](#) automatically switched to [Instructions display](#) mode.

Memory access at Address A then memory access at Address B

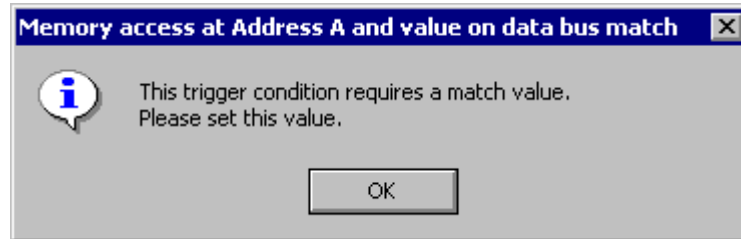
This mode is used to trigger on a program instruction sequence first reading and/or writing at Address A memory location then reading and/or writing at Address B memory location.

The code program flow rebuild is displayed in the [Trace window component](#) automatically switched to [Instructions display](#) mode.

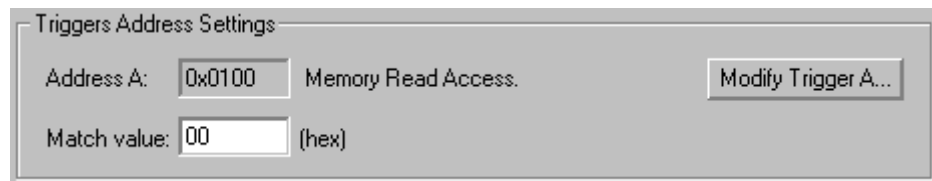
Memory access at Address A and value on data bus match

This mode is used to trigger on a program instruction read and/or write of a specific matching byte value at Address A memory location.

When choosing this trigger type, the **trigger B** address is used as a **match value** rather than an address. Also when setting this trigger via a context sensitive popup menu, the following message is displayed if the match value was never set.



The [Trigger Address Edition](#) dialog is not available for the trigger B. Special “*Match value*” edit boxes are displayed instead of *Address B* edit box.

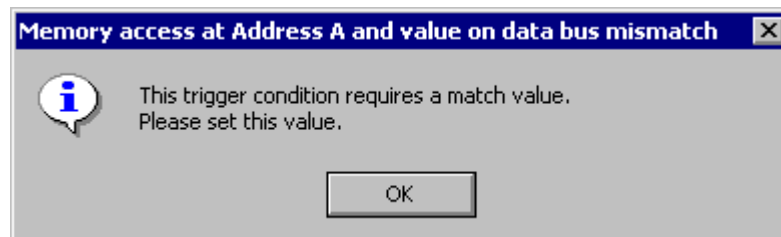


The code program flow rebuild is displayed in the [Trace window component](#) automatically switched to [Instructions display](#) mode.

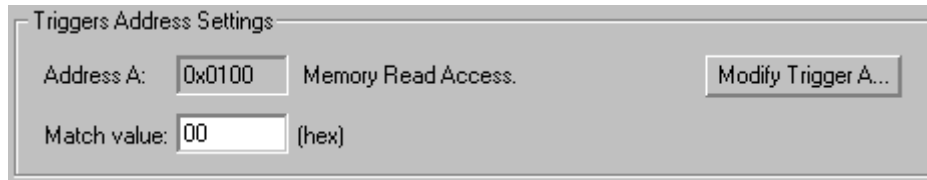
Memory access at Address A and value on data bus mismatch

This mode is used to trigger on a program instruction read and/or write of a NOT matching byte value at Address A memory location.

When choosing this trigger type, the **trigger B** address is used as a **mismatch value** rather than an address. Also when setting this trigger via a context sensitive popup menu, the following message is displayed if the match value was never set.



The [Trigger Address Edition](#) dialog is not available for the trigger B. Special “*Match value*“ edit boxes are displayed instead of *Address B* edit box.



The code program flow rebuild is displayed in the [Trace window component](#) automatically switched to [Instructions display](#) mode.

“Instruction” triggers

Instruction at Address A is executed

This mode is used to trigger on a program instruction execution (program counter) at Address A.

The code program flow rebuild is displayed in the [Trace window component](#) automatically switched to [Instructions display](#) mode.

Instruction at Address A or Address B is executed

This mode is used to trigger on a program instruction execution (program counter) at Address A or at Address B.

The code program flow rebuild is displayed in the [Trace window component](#) automatically switched to [Instructions display](#) mode.

Instruction execution inside Address A - Address B range"

This mode is used to trigger on a program instruction execution (program counter) inside the Address A - Address B range.

The code program flow rebuild is displayed in the [Trace window component](#) automatically switched to [Instructions display](#) mode.

Instruction execution outside Address A - Address B range"

This mode is used to trigger on a program instruction execution (program counter) outside the Address A - Address B range.

NOTE **IMPORTANT:** With the **HCS08 Serial Monitor via GDI target interface**, this trigger type might be be interfered by the monitor code intself and therefore the debugger break for executed code not belonging to the user application.

The code program flow rebuild is displayed in the [Trace window component](#) automatically switched to [Instructions display](#) mode.

Instruction at Address A then at Address B were executed

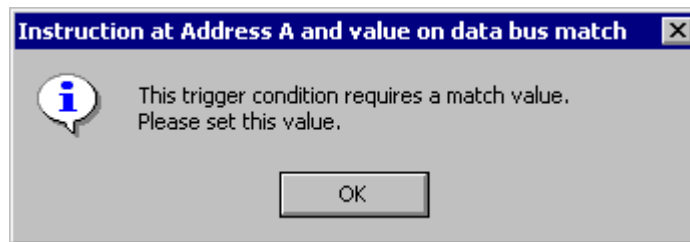
This mode is used to trigger on a program instruction execution (program counter) sequence first at Address A then at Address B.

The code program flow rebuild is displayed in the [Trace window component](#) automatically switched to [Instructions display](#) mode.

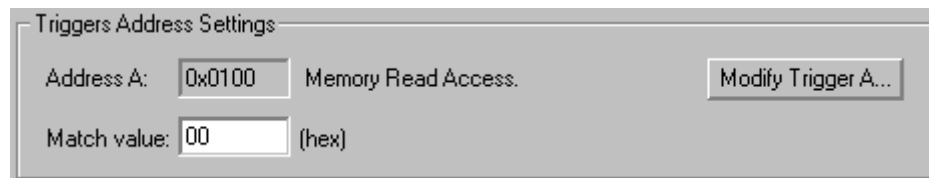
Instruction at Address A and value on data bus match

This mode is used to trigger on a program instruction execution (program counter) at Address A, this instruction opcode matching a specific byte value.

When choosing this trigger type, the **trigger B** address is used as a **match value** rather than an address. Also when setting this trigger via a context sensitive popup menu, the following message is displayed if the match value was never set.



The [Trigger Address Edition](#) dialog is not available for the trigger B. Special “**Match value**“ edit boxes are displayed instead of **Address B** edit box.

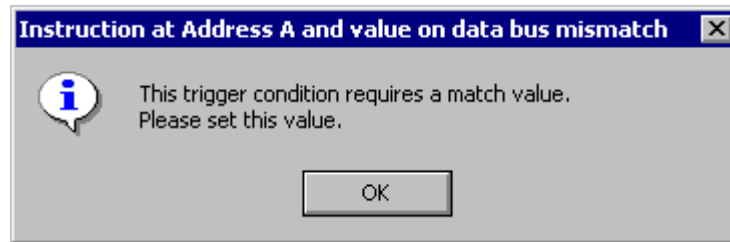


The code program flow rebuild is displayed in the [Trace window component](#) automatically switched to [Instructions display](#) mode.

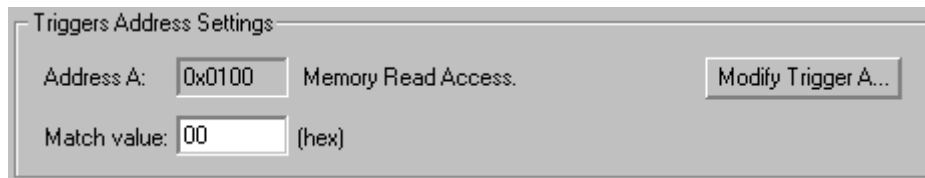
Instruction at Address A and value on data bus mismatch

This mode is used to trigger on a program instruction execution (program counter) at Address A, this instruction opcode NOT matching a specific byte value.

When choosing this trigger type, the **trigger B** address is used as a **mismatch value** rather than an address. Also when setting this trigger via a context sensitive popup menu, the following message is displayed if the match value was never set.



The [Trigger Address Edition](#) dialog is not available for the trigger B. Special “*Match value*” edit boxes are displayed instead of *Address B* edit box.



The code program flow rebuild is displayed in the [Trace window component](#) automatically switched to [Instructions display](#) mode.

“Capture” triggers

Capture the read/write values at Address B

This mode is used to capture the data involved in a read and/or write access to the address specified by the **trigger B**, such as the address of a particular control register or program variable.

Captured byte data are displayed in the [Trace window component](#) automatically switched to [Recorded data display](#) mode.

The trigger address is typically not a program code address (program counter), but rather a data/memory address.

Capture r/write values at Address B after access at Address A

This mode is used to capture the data involved in a read and/or write access to the addresses specified by the **trigger A** and the **trigger B**, such as the address of a particular control register or program variable. Triggering/capture will start only after the **trigger A** address was accessed.

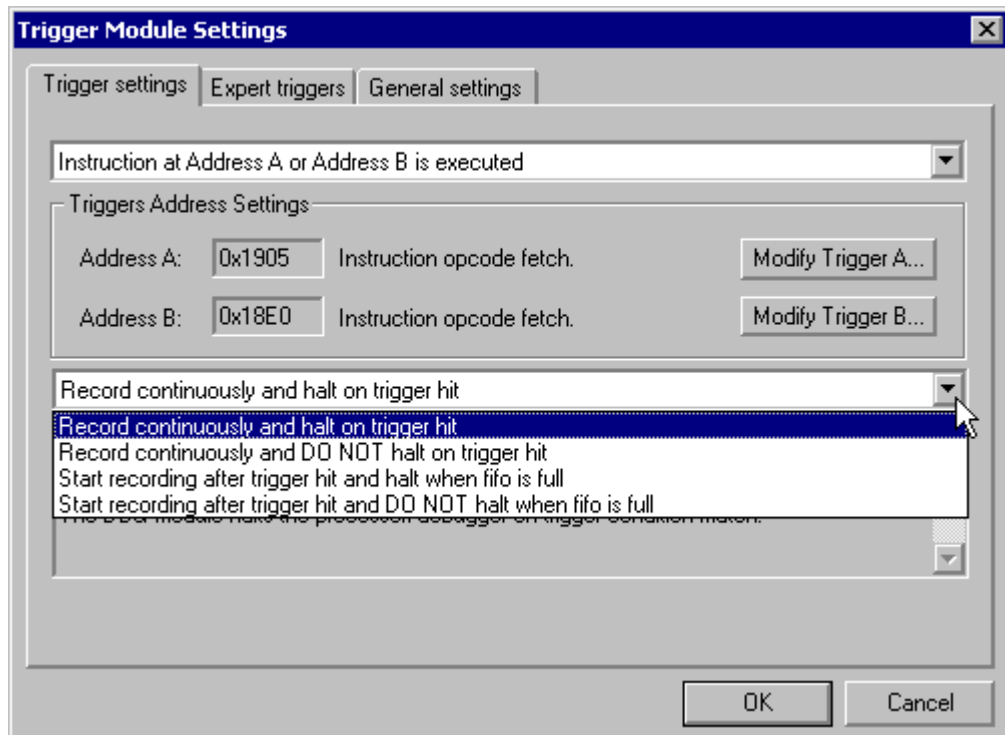
The trigger addresses is typically not a program code address (program counter), but rather data/memory addresses.

Captured byte data are displayed in the [Trace window component](#) automatically switched to [Recorded data display](#) mode.

DBG module options

When program code change of flow recording

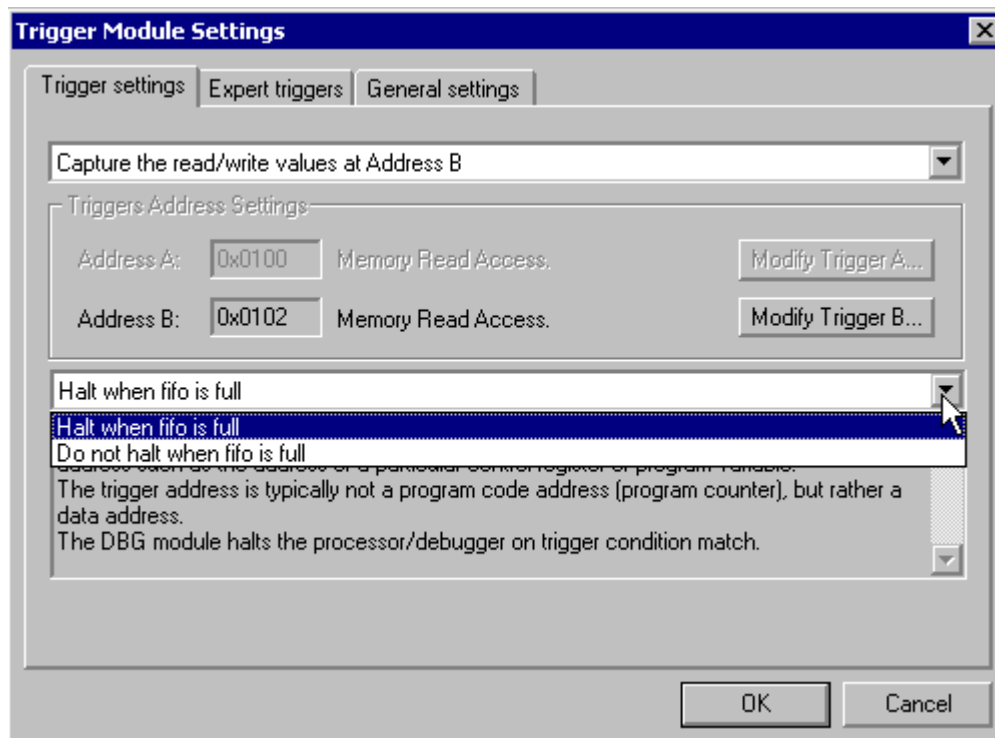
These options are available for [“Instruction” triggers](#) and [“Memory access” triggers](#):



- **Record continuously and halt on trigger hit:** The DBG module starts recording program flow information immediately after run. The DBG module halts the processor/debugger on trigger condition match.
- **Record continuously and DO NOT halt on trigger hit:** The DBG module starts recording program flow information immediately after run. The DBG module does not halt the processor/debugger on trigger condition match.
- **Start recording after trigger hit and halt when the fifo is full:** The DBG module starts recording program flow information on trigger condition match and halts the processor/debugger when the capture buffer is full.
- **Start recording after trigger hit and halt when the fifo is full:** The DBG module starts recording program flow information on trigger condition match. The DBG module does not halt the processor/debugger on trigger condition match.

When data recording

These options are available for [“Capture” triggers](#).



- **Halt when the fifo is full:** The DBG module records continuously data accesses and halts the processor/debugger when the capture buffer is full.

- **Do not halt when the fifo is full:** The DBG module records continuously data accesses and but does not halt the processor/debugger when the capture buffer is full.

Trigger Address Edition

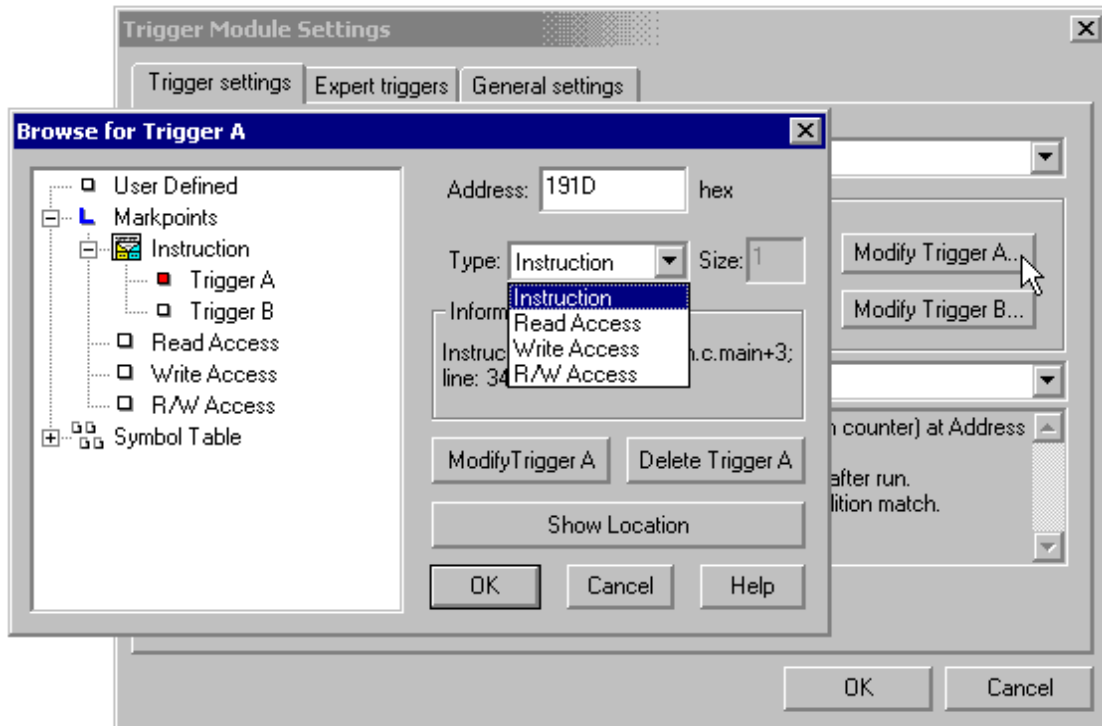
Typically trigger addresses can be set using context sensitive popup menus. It is also possible to modify trigger addresses within the [Trigger Module Settings dialog](#). Pressing “**Modify Trigger**” buttons opens a trigger editor dialog.

The “**Address**” edit box contains the initial and final trigger address value. This value can be directly set while typing in the edit box. The “**Type**” drop down list should be used to select/change the type of trigger. “Instruction” type should be used for [“Instruction” triggers](#) and “Read”, “Write” and “R/W Access” should be used for [“Memory access” triggers](#) and [“Capture” triggers](#).

Pressing “**Modify Trigger**” in this sub dialog will modify and record the trigger in the trigger database ([Triggers storing as markpoints](#)).

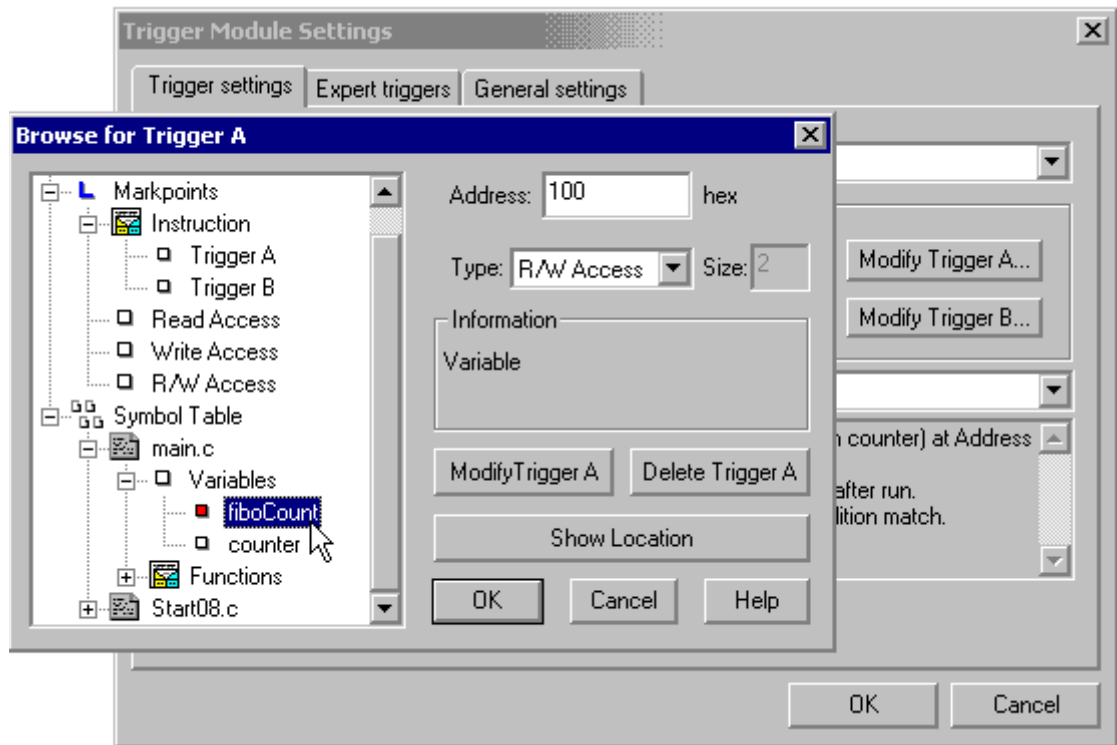
NOTE	The OK will NOT update the trigger database. “ Modify Trigger ” must be explicitly pressed before closing the dialog with OK.
-------------	--

Pressing “**Delete Trigger**” in this sub dialog will remove the trigger in the trigger database ([Triggers storing as markpoints](#)). This trigger address is then considered as “**undefined**”.



The “**Show Location**” button will show in the Source, Data, Assembly and Memory windows the location of the trigger (as program code location or program data).

The left hand side tree is a user friendly way to find a trigger address in the debugger symbol database by selecting a variable (the address of the variable will be taken and copied in the **Address** edit box) or a function (the entry point of the function will be taken and copied in the **Address** edit box) , and also regular markpoints (the address of the markpoint will be taken and copied in the **Address** edit box) from the markpoint list.



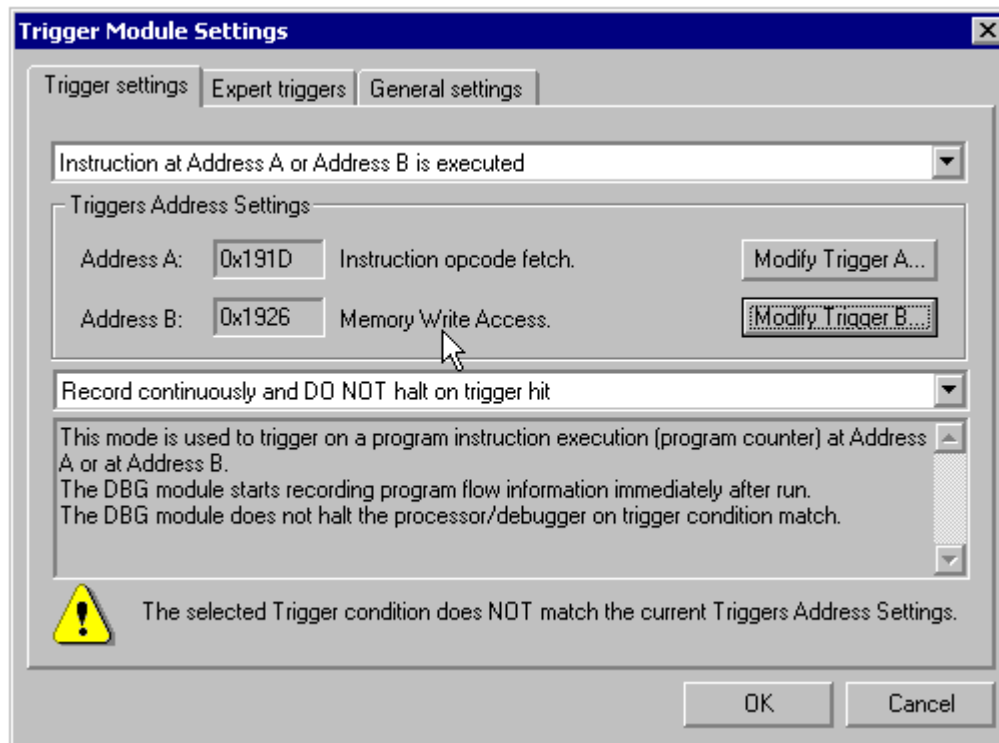
Dialog information

A large grayed edit box provides dynamically information about the current triggers and selected options.

As context sensitive popup menus will only display triggers matching the amount and the kind of triggers which are currently set, the [Trigger Module Settings dialog](#) checks dynamically the validity are current triggers set vs. the trigger mode.

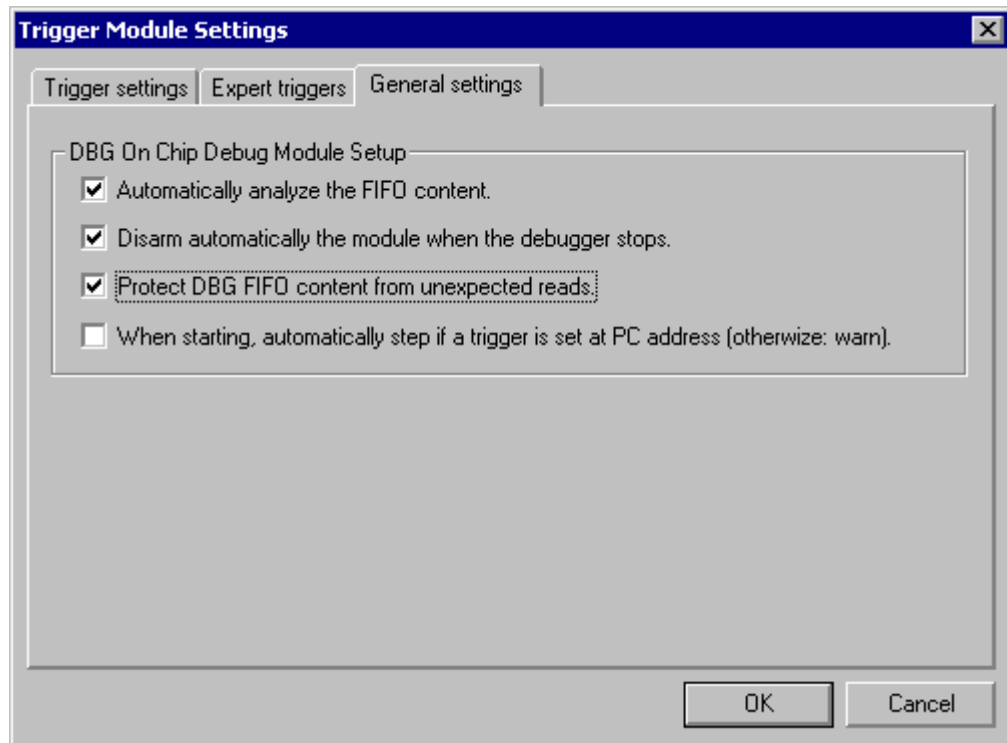
As shown below, if one or more triggers do not match the trigger mode selection, a warning icon and message is displayed on the bottom of the dialog.

Here below, the **Memory Write Access** type of trigger pointed by the mouse cursor does not match with the [“Instruction” triggers](#) type selected in the drop down list.



General Settings

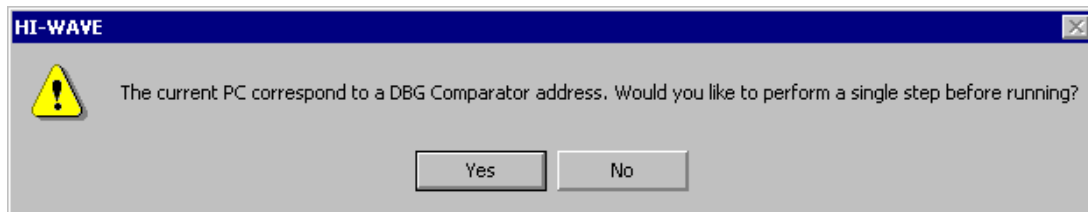
Most of the time, there would be no reason to change any of these settings, which are rather default settings of the DBG user interface. However, in some debug special cases, it is possible to disable some automated debugger background processes.



- **Automatically analyze the FIFO content:** When the [Trace window component](#) is open, after the debugger is halted by the user or a breakpoint, watchpoint or a trigger, DBG module results are automatically analysed then displayed in the Trace window. If the Trace window is closed, the DBG user interface does not perform any result analyse except trigger flags reported in the status bar. Unchecking this check box would do the same, with the Trace window open.
- **Disarm automatically the module when the debugger stops:** By default, once the target processor is halted due to user break (not any trigger), the onchip DGB module is still armed. If this option is selected (by default) the debugger will disarm it to retrieve data from the DBG Fifo. If not selected, the DBG Fifo/buffer information cannot be retrieved until the module is disarmed.
- **Protect DBG FIFO content from unexpected reads:** The DBG Fifo data are retrieved from *DBGFH-DBGFL* registers (address 0x1814-0x1815 in register block at reset location). Several reads are performed to retrieve the entire shifting buffer. However, when the debugger is halted, while refreshing Data and Assembly windows, it might read also the target processor memory at the same location, reading the first DBG Fifo data, shifting the buffer, and therefore corrupt the DBG user interface DBG Fifo data retrieving. This option hides to the

debugger and also user (see blue "-- --" designs in the Memory window at address 0x1814-0x1815) the DBG Fifo buffer location.

- ***When starting, automatically step if a trigger is set at PC address (otherwise: warn):*** To run again the application, the debugger usually needs to exit the trigger current match condition and avoid being stuck/halted/locked by the trigger. A single step is usually required to "escape" from ["Instruction" triggers](#). When this option is disabled, the debugger prompts the following dialog to validate this choice.



Trace window component

The Trace component is a debugger generic component used to display in a Trace window a debugger internal database. The context sensitive popup menu is set up by the target interface (or the GDI DLL) making usage of the component.

Any debugger host target interfaces including the DBG user interface are synchronized with the Trace component.

It is not necessary to open the Trace window/component to make usage of the DBG user interface triggers. However, several triggers are used to collect code program flow information or access data information. The Trace window can be opened from [Specific target menu entries](#), from [Specific context sensitive popup menu entries in Source, Data, Assembly and Memory component windows](#), and from the [Specific Status Bar item](#). The window can be saved in the debugger layout when pressing the debugger Save icon.

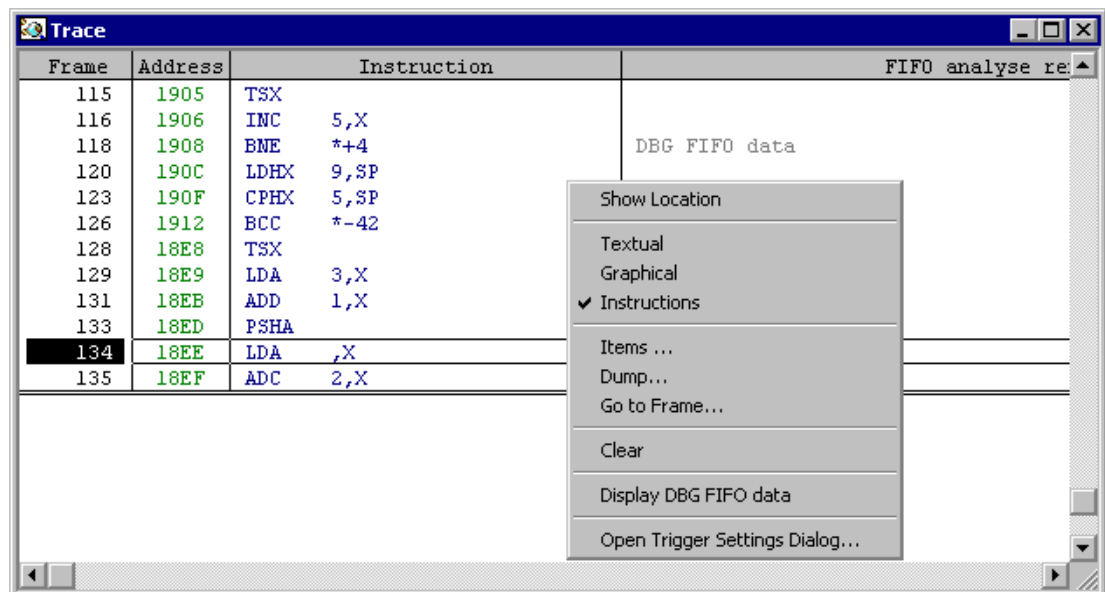
NOTE When the Trace component/window is closed, the debugger might be faster, as code program flow rebuild is discarded, this last disassembling back the assembly data from the target processor memory.

Instructions display

This display mode is automatically set when [“Instruction” triggers](#) and [“Memory access” triggers](#) are used. It is also the default display in [Automatic mode \(default\)](#).

Displayed columns:

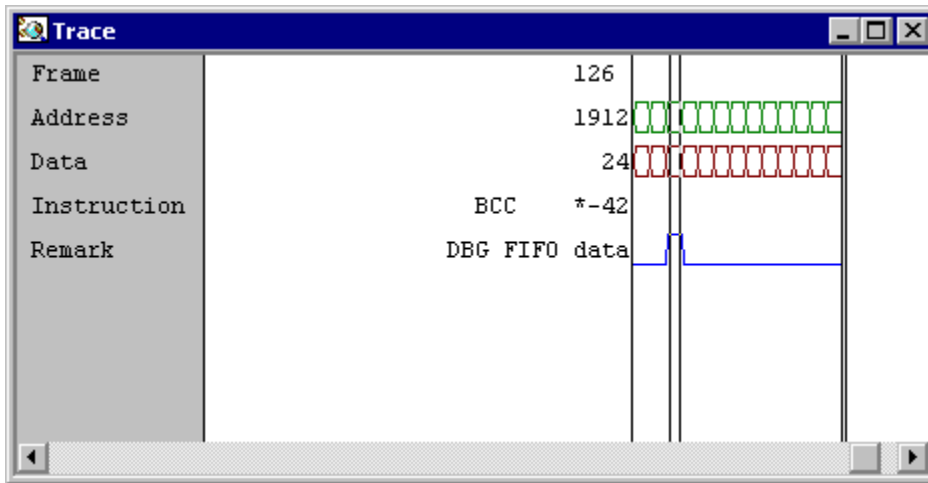
- **Frame:** A number representing an information item stored in the Trace component database.
- **Address:** instruction program counter.
- **Instruction:** code program flow instruction disassembly.
- **FIFO Analyse remark:** a debugger information: **“DBG FIFO data”** means that this data was recorded by the onchip DBG module. **“traced”** means an item/instruction obtained by debugger/user singlestep or assembly step. **“Program flow rebuild gap”** means that the debugger could not track completely the code program flow between two frames.



Selecting **“Show Location”** in the Trace window context sensitive popup menu will simply display in Source and Assembly window the frame matching source and assembly code.

Graphical display

This display mode can be select when selecting “*Graphical*” in the Trace window context sensitive popup menu. It provides a graphical representation of the same information.



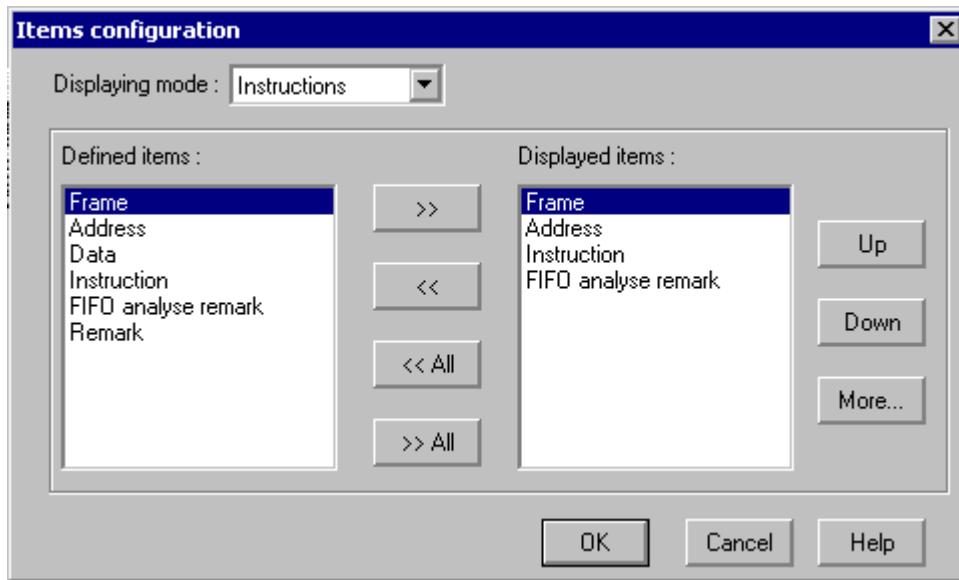
Textual display

This display mode can be select when selecting “*Textual*” in the Trace window context sensitive popup menu , when using “[Instruction triggers](#)” and “[Memory access triggers](#)” are used. This display mode is rather useless for the DBG user interface, as no read/write accesses are recorded at the same time than program change of flow information by the onchip DBG module. By consequence, the Textual display mode simply expands instruction assembly code in the Trace window.

Frame	Address	Data	Instruction	FIFO ar
119	1909	02		
120	190C	9E	LDHX 9,SP	
121	190D	FE		
122	190E	09		
123	190F	9E	CPHX 5,SP	
124	1910	F3		
125	1911	05		
126	1912	24	BCC *-42	DBG FIFO data
127	1913	D4		
128	18E8	95	TSX	
129	18E9	E6	LDA 3,X	
130	18EA	03		
131	18EB	EB	ADD 1,X	
132	18EC	01		
133	18ED	87	PSHA	
134	18EE	F6	LDA ,X	
135	18EF	E9	ADC 2,X	
136	18F0	02		

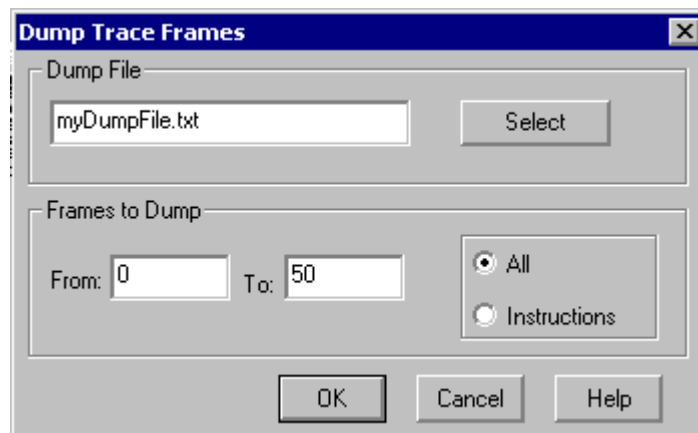
Columns display and moving

Selecting “*Items...*” in the Trace window context sensitive popup menu opens a small dialog to setup the columns to hide/display in each display mode. The “Displaying mode” drop down list can be opened to make column display modification in *Textual*, *Instructions* or *Graphical* mode.



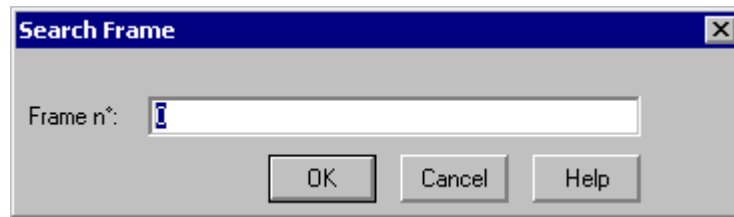
Frames dumping to file

Selecting “*Dump...*” in the Trace window context sensitive popup menu opens a small dialog to dump/save Trace component frames to a text file.



Goto frame

Selecting “*Go to Frame...*” in the Trace window context sensitive popup menu opens a small dialog to go to a frame in the Trace window.



Frames clearing

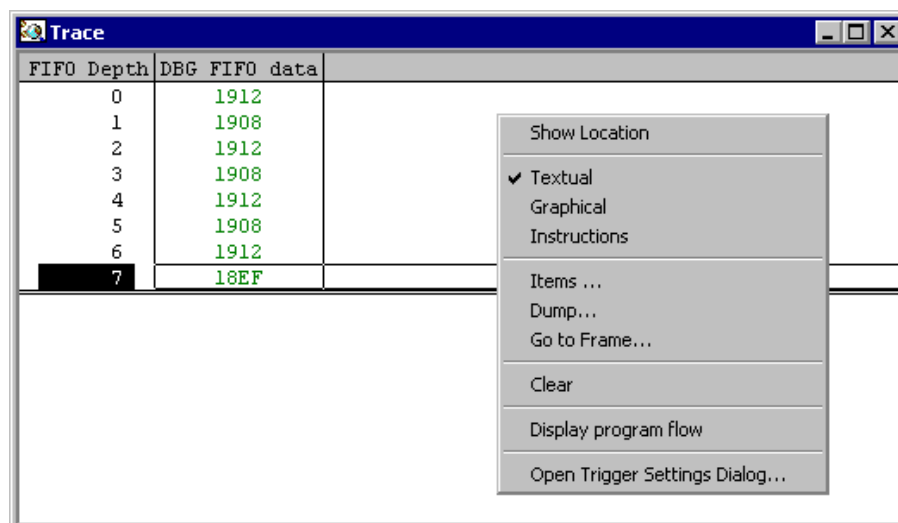
Selecting “*Clear*” in the Trace window context sensitive popup menu will simply flush the frames in the Trace window (flushing in background the database).

DBG module Fifo/buffer display

Selecting “*Display DBG FIFO data*” in the Trace window context sensitive popup menu will simply display data information retrieved from the onchip DGB module Fifo/buffer. Selecting “*Display program flow*” in the Trace window context sensitive popup menu will simply turn back to code program flow display.

Displayed columns:

- **FIFO Depth:** A number representing the depth in the DBG/Fifo of the word data value. The first frame (Depth 1) is the oldest value in the time.
- **DBG FIFO Data:** the word value retrieved from the DBG Fifo/buffer from DBGFH and DBGFL DBG onchip module registers.

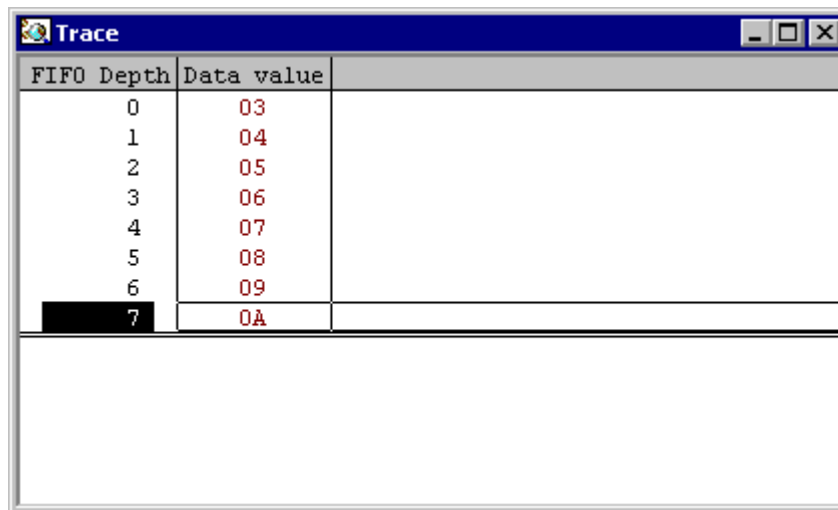


Recorded data display

This display mode is automatically set when [“Capture” triggers](#) are used.

Displayed columns:

- **FIFO Depth:** A number representing the depth in the DBG/Fifo of the byte data value. The first frame (Depth 1) is the oldest value in the time.
- **Data value:** the byte value retrieved from the DBG Fifo/buffer from the DBGFL DBG onchip module register.



FIFO Depth	Data value
0	03
1	04
2	05
3	06
4	07
5	08
6	09
7	0A

Limitations

Demo/unregistered debugger mode

- In demo/unregistered debugger mode, code program reconstruction has a limited number of frames displayed in the Trace window.
- Also Real time code Profiling and code Coverage are disabled.
- No preset/predefined [“Instruction” triggers](#), [“Memory access” triggers](#) or [“Capture” triggers](#) are provided. Only [Expert triggers](#) can be set.

Index

A

A 7
 Address 30
 Address B 23, 24
 Automatically analyze the FIFO content 33

B

B 7
 breakpoint 5
 Bus Trace 4

C

Capture 26
 Capture r/write values at Address B after access at Address A 27
 Capture the read/write values at Address B 26
 Clear 39
 Code coverage 20
 Code profiling 20
 complex breakpoint 5
 Controlpoints Configuration 9, 14
 Coverage 20, 40

D

Data 10
 Data value 40
 DBG 35
 DBG FIFO Data 39
 DBGCA 12, 14, 19
 DBGCB 14, 19
 DBGFH 33, 39
 DBGFL 33, 39, 40
 DBGT 20
 Debugger Trigger Register 20
 Delete Trigger 29
 Delete Trigger Address 8
 Disabled mode 21
 Disarm automatically the module when the debugger stops 33
 Do not halt when the fifo is full 29
 Dump 38
 dynamic 15

E

e 13
 Expert 19
 Expert mode 12
 Expert triggers 19

F

FIFO 33, 35, 39
 FIFO Analyse remark 35
 FIFO Depth 39, 40
 Frame 35

G

GDI 25, 34
 Go to Frame 38
 Graphical 36
 Graphical display 36

H

Halt when the fifo is full 28
 hardware breakpoints 18
 HCS08 3
 HCS08 Serial Monitor 3, 25

I

inDART-HCS08 3
 INSTRUCTION 14
 Instruction 15
 Instruction at Address A and value on data bus match 25
 Instruction at Address A and value on data bus mismatch 26
 Instruction at Address A is executed 24
 Instruction at Address A or Address B is executed 24
 Instruction at Address A then at Address B were executed 25
 Instruction execution inside Address A - Address B range" 24
 Instruction execution outside Address A - Address B range" 24
 Instructions display 35
 Items 37

M

Markpoints 9, 14
 markpoints 8
 match value 23, 25
 Memory 10
 Memory access at Address A 22
 Memory access at Address A and value on data bus match 23
 Memory access at Address A and value on data bus mismatch 23
 Memory access at Address A or at Address B 22
 Memory access at Address A then memory access at Address B 22
 Memory access inside Address A - Address B range 22
 Memory Write Access 31
 mismatch value 23, 26
 Modify Trigger 29

P

P&E 3
 PEDebug 3
 Profiling 20, 40
 Program flow rebuild gap 35
 Protect DBG FIFO content from unexpected reads 33

R

R/W Access 29
 Read 11, 29
 Read Access 15
 Read/Write 11
 Read/Write Access 15
 READACCESS 14
 READWRITEACCESS 14
 Record continuously and DO NOT halt on trigger hit 28
 Record continuously and halt on trigger hit 28

S

Save and Restore on load 9, 14
 Serial Monitor 25
 Set DBGCA 12
 Set DBGCB 12
 Set Trigger A 18
 Set Trigger Address 11
 Set Trigger Address B 12
 Set Trigger B 18

Set TriggerAddress 6
 Show Location 30, 35
 Show Markpoints... 8
 Start recording after trigger hit and halt when the fifo is full 28
 Status bar 16

T

Textual 36
 Textual display 36
 Trace component 34
 Trace window 34
 traced 35
 Trigger A 6, 9
 Trigger address edition 29
 Trigger B 6, 9
 trigger B 23
 Trigger Module Settings 4
 Type 29

U

undefined 29

W

watchpoint 5, 11
 Write 11, 29
 Write Access 15
 WRITEACCESS 14