Document Number: AN3335
Rev. 1, 11/2006

Introduction to HCS08 Background Debug Mode

by: Steven McLaughlin
Applications Engineer
East Kilbride, Scotland

Contents

1 Introduction 1
2 Overview 2
2.1 Features and Operation. 2
2.2 Background Debug Controller 4
2.3 Communication Details. 8
3 Hardware and Software Set-up 9
4 Application Tutorial 10
4.1 Debug 10
4.2 Setting Breakpoints 12
5 Setting Triggers 15
6 Conclusion. 19
7 References 19
8 Glossary 19

1 Introduction

This application note outlines background debug mode (BDM) on Freescale Semiconductor's 8-bit microcontrollers, primarily the S08 and RS08 cores. The beginning of this application note focuses on defining BDM, its functions, and operation. Also included in this document are a step-by-step guide and examples of in-circuit debugging using a P&E microcomputer systems BDM pod with CodeWarrior™.

By using the background debug hardware on the S08/RS08, traditional in-circuit emulators (ICE) that require external components are unnecessary; this allows a reduction in hardware costs.

The S08 and RS08 family differs from many of its peers by providing a background debug mode, which consists of the background debug controller (BDC), and for the S08 only, the debug module (DBG).

When the BDC is programmed with non-intrusive and/or active-background commands, the background communications are controlled between the host and target MCU, allowing access to the target MCU address and data buses. Through built-in debug hardware, this is a valuable and cost-effective tool that accelerates time-to-market.

2 Overview

To debug the MCU using the background debug method, the following equipment is required:

- Host PC, complete with appropriate CodeWarrior software
- Target MCU and board
- BDM pod (does not require separate power supply) and appropriate USB connecting cable.

Figure 1 illustrates the BDM pod with its USB cable. This connects the two pieces of hardware.

NOTE

The pod contains two indicator LEDs to confirm connection to the PC and target MCU.

Figure 1. BDM Pod

2.1 Features and Operation

The target board is connected to a host PC via a BDM interface pod (Figure 2). The connection on the target board is a custom 6-pin connector (only 4 pins are used) and communication is carried out via the BKGD pin. The RESET signal allows for a direct connection for the host to initiate a target system reset, if required.

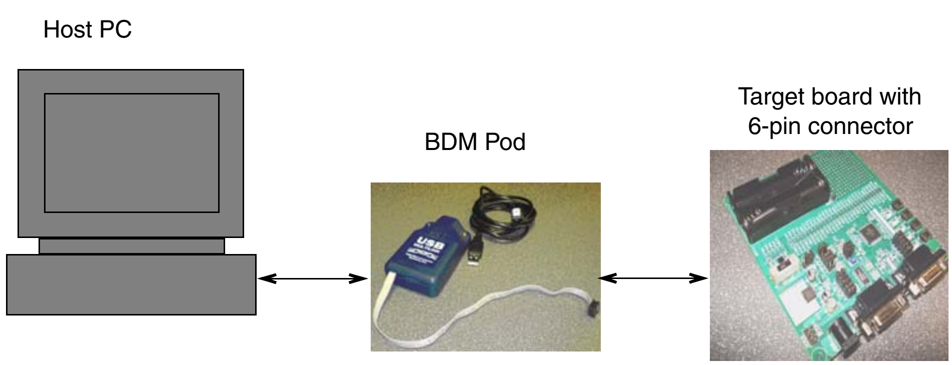

Figure 2. Connection Between Host PC and MCU Using BDM

The pins on the custom 6-pin connector are described as follows:

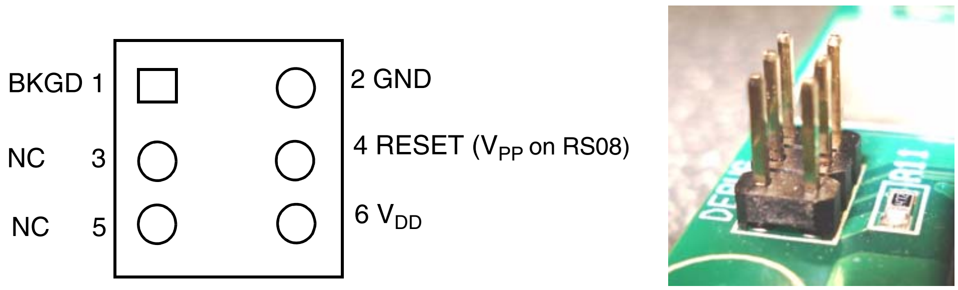

Figure 3. BDM Connector Pins

The BKGD pin is a pseudo open-drain pin with an on-chip pullup, that is, no external pullup resistor is required for noise immunity. This pin provides a single-wire debug interface to the target MCU; however, this PIN can also access the RAM, on-chip flash, and other non-volatile memories such as EEPROM—this depends what BDM commands are made: active background or non-intrusive commands (see Section 2.2, “Background Debug Controller”).

As the debug facility consists of hardware logic, the BDC does not need to use the CPU or its instructions, allowing the BDC to access internal memory even while the program runs.

BDC features:

- Single dedicated pin for mode selection and background communications – BKGD pin
- BDC registers not located in memory map
- SYNC command to determine target communicates rate
- Non-intrusive commands for memory access
- Active background mode commands for CPU register access
- GO and TRACE1 commands
- BACKGROUND command can wake CPU from stop or wait modes
- One hardware address breakpoint built into BDC
- Oscillator runs in stop mode if BDM is enabled.

DBG features:

- Two trigger comparators:
  - — Two address and read/write (R/W) or
  - — One full address + data + R/W
- Flexible 8-word by 16-bit first-in, first-out (FIFO) for capture information:
  - — Change of flow addresses or
  - — Event data only
- Two types of breakpoints
  - — Tag breakpoints for instruction opcodes
  - — Force breakpoints for any address access

- Nine trigger modes
  - — A only
  - — A or B
  - — A then B
  - — A and B data (full mode)
  - — A but not B data (full mode)
  - — Event-only B (store data)
  - — A then event-only B (store data)
  - — Inside range (A address B)
  - — Outside range (address < A or address > B)

2.2 Background Debug Controller

The BDC is responsible for communication via the single BKGD pin. The main advantage—compared to earlier 8-bit devices—is that this method does not interfere with normal application processes because it does not use memory or locations in the memory map, and it does not share on-chip peripherals.

These commands are sent serially from the PC to the BKGD pin, MSB first, using a custom BDC communications protocol. Commands do not affect the real-time operation of the program even as the program runs. Table 1 summarizes the BDC commands.

In BDC there are 30 (21 for RS08) commands divided into two groups:

- Active background mode commands — User program is not running. Background command causes MCU to enter active background mode. The CPU registers can be written/read and allow you to trace one instruction at a time.
- Non-Intrusive Commands — User program running. Allow the MCU to be read/written to and allow access to the status and control registers.

Table 1. BDC Command Summary (BDCSCR)

RS08 Commands	S08 Commands	mode	Description
SYNC	SYNC	non-intrusive	Request a timed reference pulse to determine target BDC communication speed
ACK_ENABLE	ACK_ENABLE	non-intrusive	Enable handshake; issues an ACK pulse after the command is executed
ACK_DISABLE	ACK_DISABLE	non-intrusive	Disable handshake; command does not issue an ACK
BACKGROUND	BACKGROUND	non-intrusive	Enters active background mode if enabled
READ_STATUS	READ_STATUS	non-intrusive	Read BDC status from BDCSCR

Table 1. BDC Command Summary (BDCSCR) (continued)

RS08 Commands	S08 Commands	mode	Description
WRITE_CONTROL	WRITE_CONTROL	non-intrusive	Write BDC status from BDCSCR
READ_BYTE	READ_BYTE	non-intrusive	Read a byte from target memory
READ_BYTE_WS	READ_BYTE_WS	non-intrusive	Read a byte and report status
—	READ_LAST	non-intrusive	Re-read byte from address just read and report status
WRITE_BYTE	WRITE_BYTE	non-intrusive	Write a byte to target memory
WRITE_BYTE_WS	WRITE_BYTE_WS	non-intrusive	Write a byte and report status
READ_BKPT	READ_BKPT	non-intrusive	Read BDCBKPT breakpoint register
WRITE_BKPT	WRITE_BKPT	non-intrusive	Write BDCBKPT breakpoint register
GO	GO	active background mode	Go to execute the user application program starting at the address currently in the PC
TRACE1	TRACE1	active background mode	Trace 1 user instruction at the address in the PC, then return to active background mode
—	TAGGO	active background mode	Same as GO but enable external tagging
READ_A	READ_A	active background mode	Read accumulator (A)
—	READ_CCR	active background mode	Read condition code register (CCR)
READ_CCR_PC	—	active background mode	Read CCR PC
—	READ_PC	active background mode	Read program counter (PC)
—	READ_HX	active background mode	Read H and X register pair (H:X)

Table 1. BDC Command Summary (BDCSCR) (continued)

RS08 Commands	S08 Commands	mode	Description
—	READ_SP	active background mode	Read stack pointer (SP)
READ_SPC	—	active background mode	Read shadow program counter (SPC)
—	READ_NEXT	active background mode	Increment H:X by one, then read memory byte located at H:X.
—	READ_NEXT_WS	active background mode	Increment H:X by one, then read memory byte located at H:X. Report status and data
READ_BLOCK	—	active background mode	Blocks of data read from target memory
WRITE_A	WRITE_A	active background mode	Write accumulator (A)
—	WRITE_CCR	active background mode	Write CCR
WRITE_CCR_PC	--	active background mode	Write CCR PC
—	WRITE_PC	active background mode	Write PC
—	WRITE_HX	active background mode	Write H:X
—	WRITE_SP	active background mode	Write SP
WRITE_SPC	—	active background mode	Write SPC
—	WRITE_NEXT	active background mode	Increment H:X by one, then write memory byte located at H:X.
—	WRITE-NEXT_WS	active background mode	Increment H:X by one, then write memory byte located at H:X. Report status
WRITE_BLOCK	—	active background mode	Blocks of data written from target memory

Related to the BDC are the status and control registers (BDCSCR) and the breakpoint register (BDCBKPT). BDCSCR is an 8-bit register which contains the ENBDM bit, bit-7, which is required to initiate active background mode. The 16-bit BDCBKPT register holds the address for the hardware breakpoint in the BDC.

Figure 4 and Figure 5 illustrate each bit of the status and control register. The BDCSR register is not held in the user memory map. Because the register can only be accessed by the debugger, it avoids enabling the BDM unintentionally while the user program runs.

The benefits and operation of the RS08 BDC and HCS08 are the same. The only main difference is that the DVF and CLKSW bits are not present in the RS08 BDCSCR register.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	ENBDM	BDMACT	BKPTEN	FTS	CLKSW	WS	WSF	DVF
Write:								
Normal Reset:	0	0	0	0	0	0	0	0
Reset in active background mode:	1	0	0	0	1	0	0	0

Figure 4. HCS08 BDC Status and Control Register (BDCSCR)

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	ENBDM	BDMACT	BKPTEN	FTS	0	WS	WSF	0
Write:								
Normal Reset:	0	0	0	0	0	0	0	0
Reset in Active BDM:	1	1	0	0	0	0	0	0

Figure 5. RS08 BDC Status and Control Register (BDCSCR)

Table 2. BDC Status and Control Register (BDCSCR) Summary

Bit	Name	Function	State	
			0	1
7	ENBDM – Enable BDM	Written to 1 by debug host when debugging	BDM cannot be made active (non-intrusive commands only)	Active BDM allows active background mode commands
6	BDMACT – background active status	Read only status bit	BDM inactive	BDM active
5	BKPTEN – BDC breakpoint enable	If 0 BDC breakpoint disabled and FTS and BDCBKPT are ignored	BDC breakpoint disabled	BDC breakpoint enabled
4	FTS – Force/Tag select	If 1 breakpoint requested whenever CPU address bus matched BDCBKPT register. If 0 match between CPU address bus and BDCBKPT register causes fetched opcode to be tagged.	Tag opcode at breakpoint address and enter active background mode if CPU attempts to execute that instruction	Breakpoint match forces active background mode at next instruction boundary.

Table 2. BDC Status and Control Register (BDCSCR) Summary (continued)

3	CLKSW – Select source for BDC Comms clock	If reset in normal mode, CLKSW forced 0 and selects alternative frequency source as BDC clock. When reset in background mode, CLKSW forced 1 selecting BDC clock.	Derivative specific fixed alternative frequency source	CPU bus clock
2	WS – Wait or stop status	When CPU is in wait stop mode, most BDC commands cannot function. The BACKGROUND command can force CPU into active background mode – BDC commands work. Host would therefore issue READ_STATUS command to check BDMACT=1	Target CPU is running user application code or is in active background mode (was not in wait or stop mode when background became active)	Target CPU is in wait or stop mode, or a BACKGROUND command was used to change from wait or stop mode to active background mode.
1	WSF – Wait or stop failure status	Status bit set if memory access command failed due to target CPU executing WAIT or STOP instruction	Memory access did not conflict with a WAIT or STOP instruction	Memory access command failed because CPU entered WAIT or STOP mode
0	DVF – Data valid failure status	Status bit set if memory access command failed due CPU executing slow memory access	Memory access did not conflict with slow memory access	Memory access command failed as CPU was not finished with sow memory access.

2.3 Communication Details

Communication with the target MCU through the BDC is carried out by one of two methods:

- Hardware method—The RESET pin is released after the BKGD pin, reset pins are pulled low, and BKGD pin is released. The MCU enters active background mode.
- Software method—There is no initial reset of the MCU. Non-intrusive command can be sent to the MCU, where it is possible to enter active background mode this way (Table 2).

To indicate the beginning of each bit time, a falling edge from the BDC serial interface is generated on the BKGD pin. Data transfers MSB first at 16 BDC clock cycles per bit. Note the BDC clock speed; the BDM pod requires knowing the target MCU clock speed.

The clock frequency derives from the bus-rate clock or a fixed-frequency alternative clock source (the state of bit-3, CLKSW of the BDCSCR register defines this). The preferred clock source for general debugging occurs when the MCU is reset in normal mode (CLKSW = = 0 – selects alternative clock source). On the other hand, assigning a 1 to CLKSW selects the bus clock as the BDC clock source. This setting is commonly used when programming the flash memory because the bus clock can run at the highest bus frequency, ensuring the fastest flash programming times. Usually, this setting should not be used for general debugging: there is no way to know whether the application program will not change the clock generator settings.

3 Hardware and Software Set-up

The software used to configure the device is CodeWarrior Development Studio. The following step-by-step guide demonstrates how to connect the debugger pod to an evaluation board (EVB). The procedure of connecting the debugger to the EVB and using the software will be the same for the majority of devices. The BDM pod used is a P&E Microcomputer USB multilink (Figure 1). The evaluation board is connected to a host PC through a USB port. To implement the hardware, follow these steps only if CodeWarrior is installed:

1. Power the EVB with the required power.
2. Make sure the on-off switch on the EVB is in the ON position.
3. Insert one end of USB cable into a free USB port on the PC.
4. Insert the other end of the USB cable into the BDM pod. Installation Window (Figure 6) should appear if this is the first time the pod is used with the PC. Follow the automatic installation.
5. Attach the female socket of the BDM connection to the 6-pin connector on the EVB.

NOTE

The colored wire corresponds to pin 1 (BKGD) and, therefore, should connect to the pin marked □ or 1.

Figure 6. Installation Window

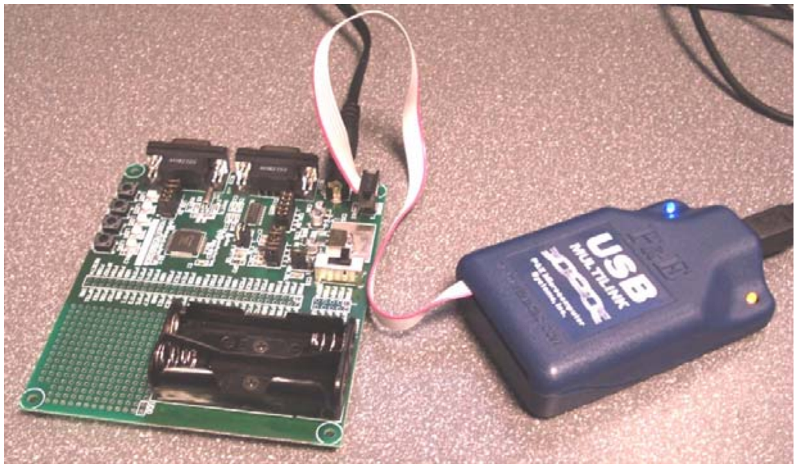

Figure 7. BDM 6-Pin Connections[1]

4 Application Tutorial

This tutorial is based on inserting breakpoints and trigger points using the HC(S)08 CodeWarrior IDE and a MC9S08GB60 DEMO board.

4.1 Debug

The first stage is to debug the program. Click the highlighted button (Figure 8).

After the debugger runs (and no errors or warnings have appeared), the ICD connection assistant window shows that the BDM debugger module was detected. Generally, the interface and port pull-down menus do not need changed.

1. Colored wire corresponds to Pin 1

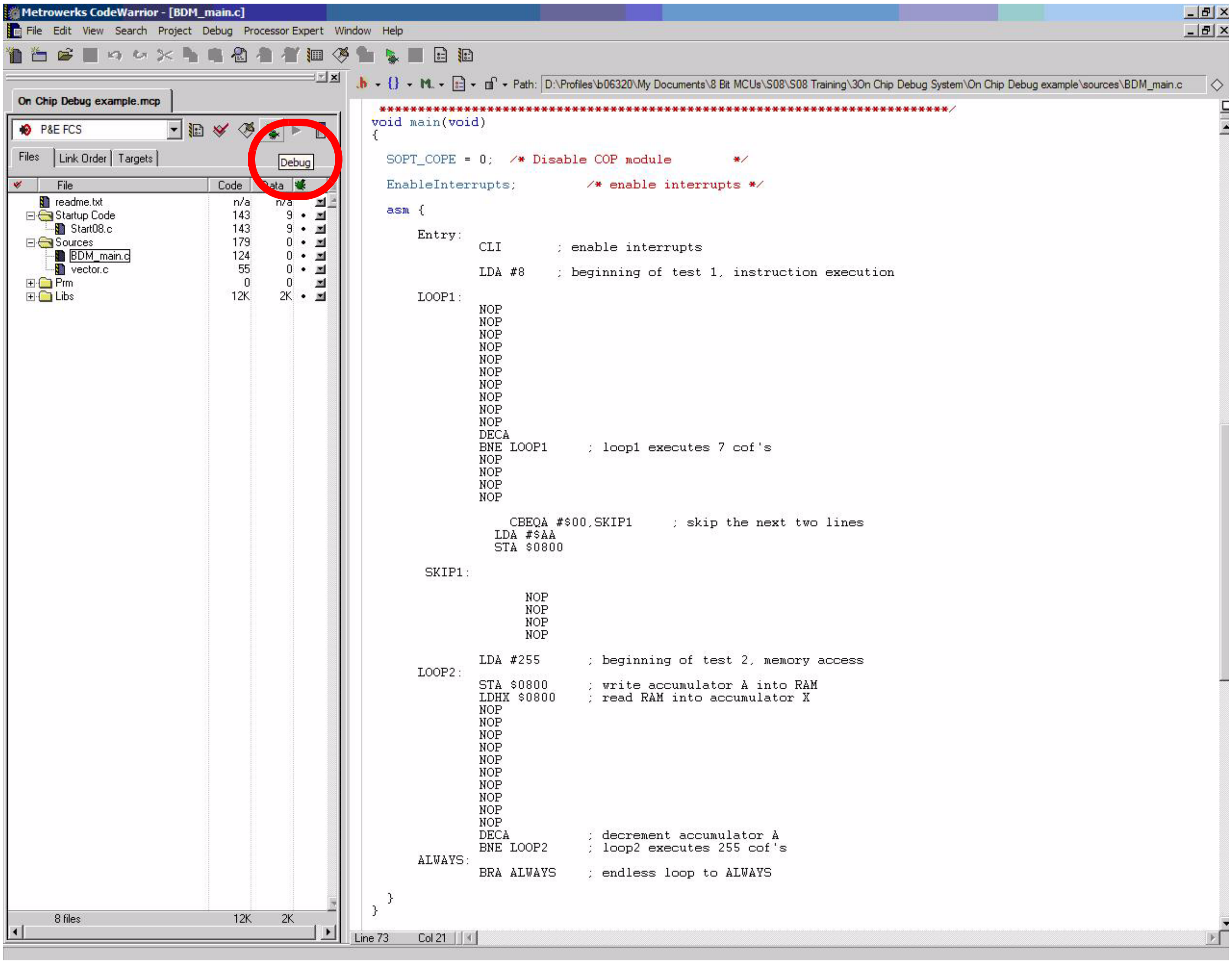

Figure 8. Main Compiler Window[1]

1. Debug button is circled

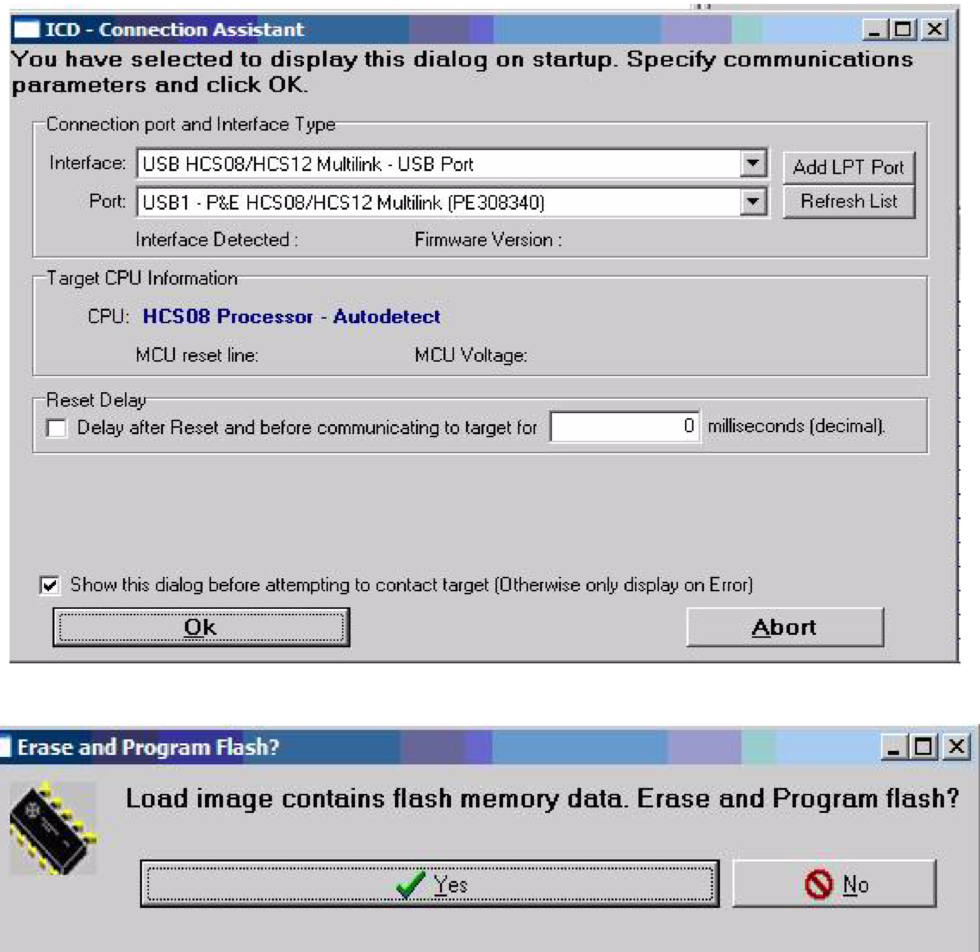

Figure 9. ICD Connection Assistant Window Followed by Program Load Confirmation

4.2 Setting Breakpoints

Breakpoints are added into the software by right-clicking next to the instruction and selecting, set breakpoint. A red arrow appears, indicating the breakpoint position and where the compiler will halt. Because there are three hardware comparators, only three breakpoints (only applicable on MCUs with the DBG module) with no triggers can be used at any one time. Figure 10 illustrates using a breakpoint.

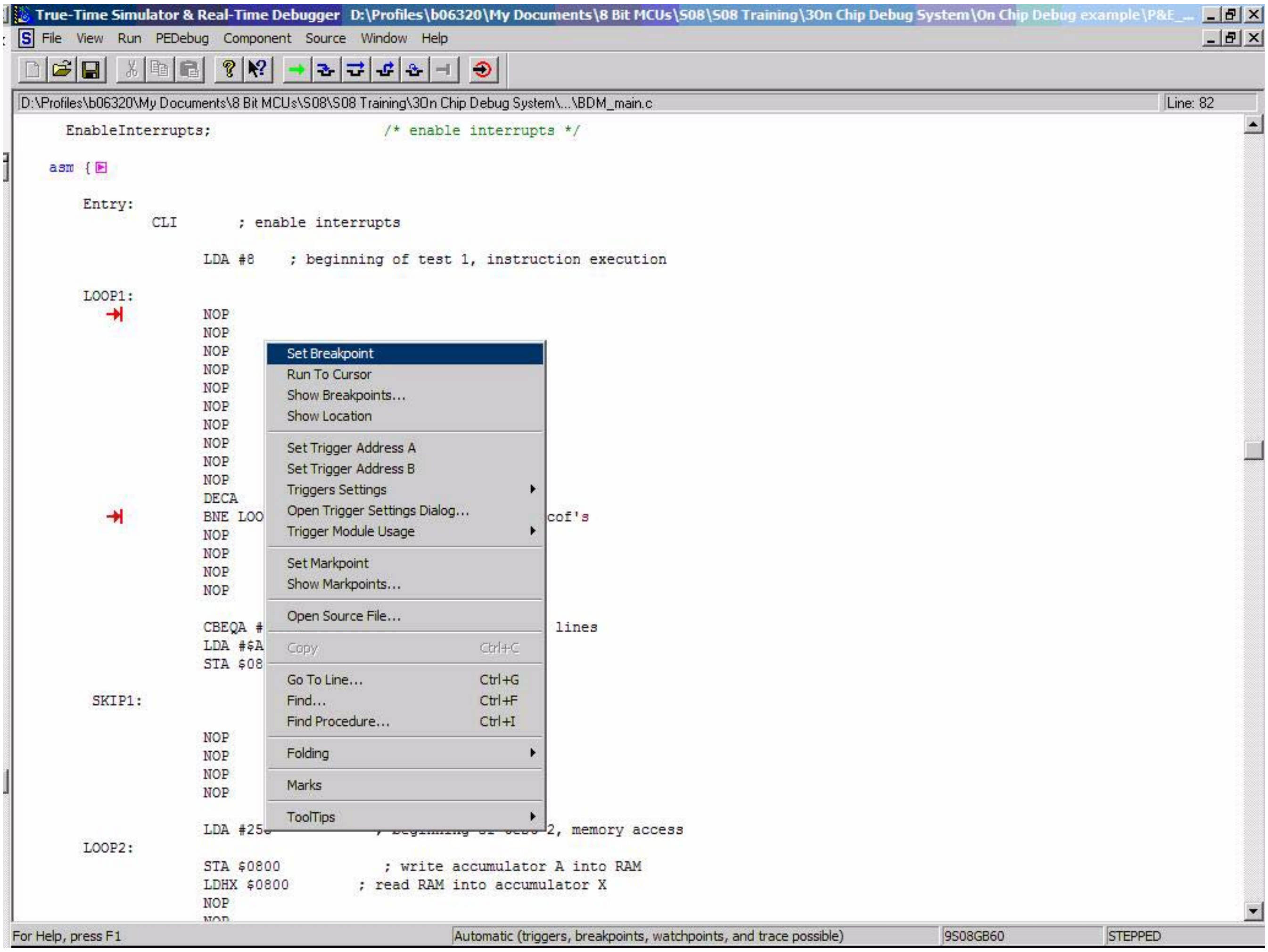

Figure 10. Red Arrows Indicate Set Breakpoints

Figure 11 illustrates the run and step function (C-step and assembly step) within the program.

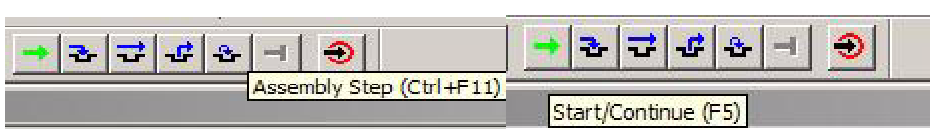

Figure 11. Run- and Step-Through Functions

Delete breakpoints by right clicking on the line they are on and select, delete breakpoint (Figure 12).

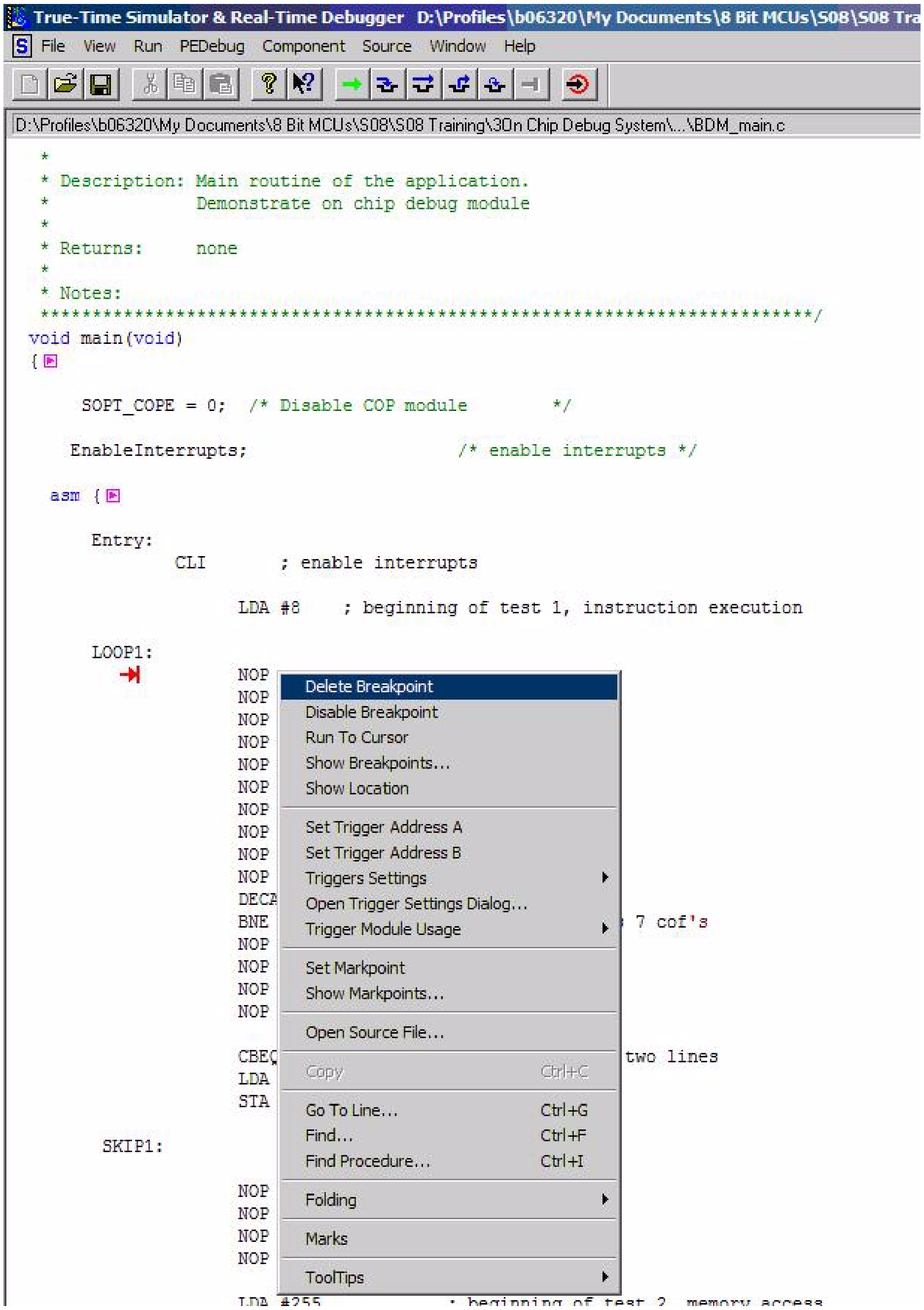

Figure 12. Delete Breakpoint

5 Setting Triggers

NOTE

This section applies to only MCUs with the DBG module. Setting a trigger (Figure 13) is similar to setting a breakpoint: right click next to the instruction and select, set trigger address A (or B). The trigger is represented by a red uppercase letter A or B. This action takes the 16-bit address associated with the instruction A or B and sets it in the corresponding hardware comparator.

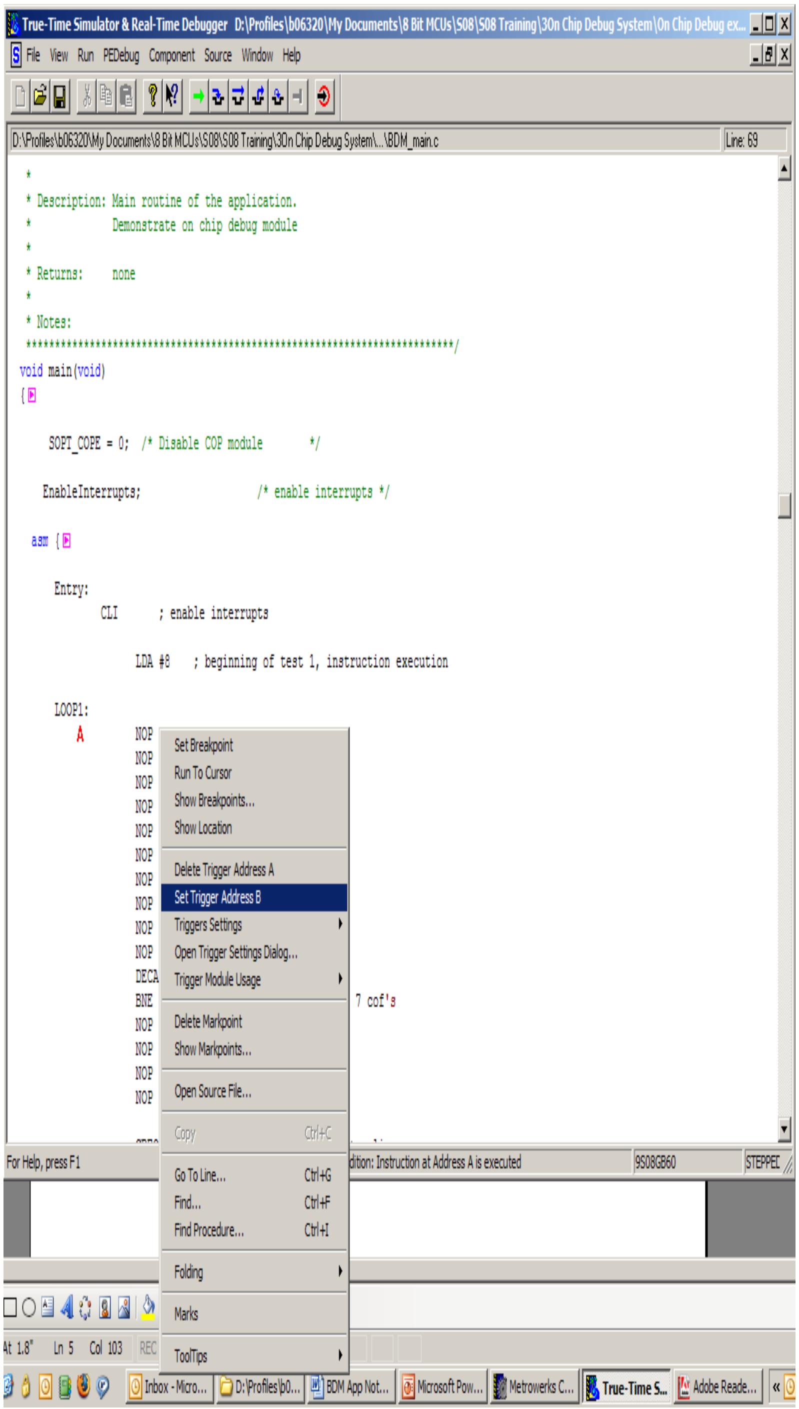

Figure 13. Setting Trigger

1. The trigger can be modified from its default settings. Click on the triggered instruction to select trigger settings. The on-chip debug system allows you to select whether to begin or end recording data on the trigger point (Figure 14).

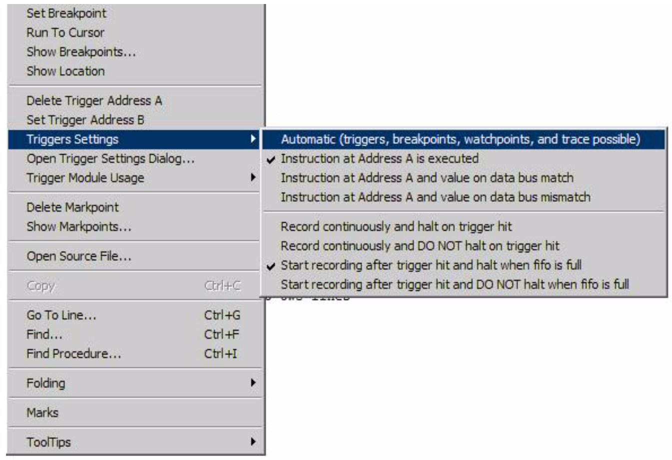

Figure 14. Modify Trigger Settings

2. In this case, one comparator is set to execute a single, 16-bit address trigger with the A-only condition. You can set a dual 16-bit address trigger, where two comparators are set at separate 16-bit addresses with one of the following conditions requiring to be satisfied: A or B, inside A or B, outside A or B, A then B. Trigger B would be set in exactly the same way as trigger A. When both triggers are set, the conditions between the two addresses can be adjusted by scrolling through trigger settings (Figure 15)

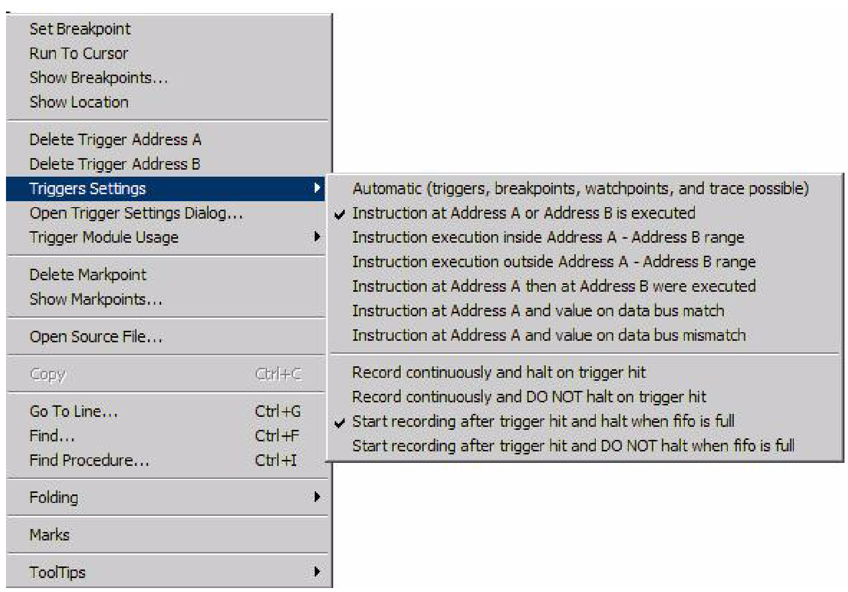

Figure 15. Trigger A and B Settings

3. In BDM mode, the trigger settings allow the user to request that software-execution halt returns the CPU to a background state. In normal mode, the user code can initiate the on-chip debug system, run the software, and perform a software interrupt following a successful trigger that can be used for ROM patching.

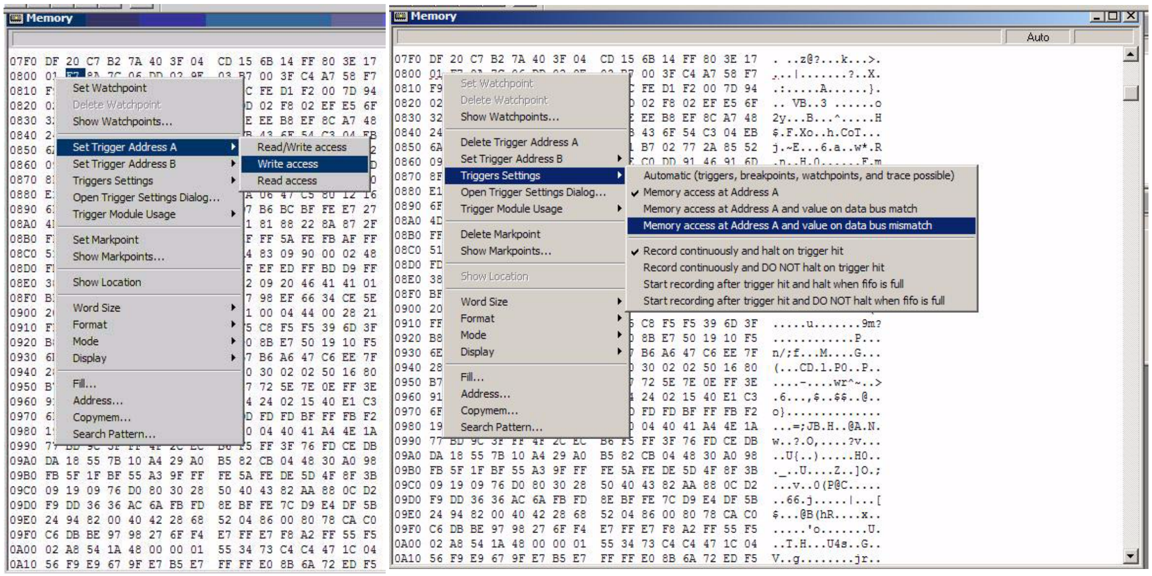

Figure 16. Setting 16-bit Address

4. When setting an 8-bit data trigger, you must select the type of memory access (read/write). Figure 16 illustrates how to select the 16-bit address: right click on the desired memory location in the memory map window by selecting, set trigger address A then write access. The trigger is marked by a dashed underline.
5. To set the 8-bit data value completing the trigger, right click on the memory window and change the trigger settings to memory access at address A and value on data bus match. This prompts for a value. The value should be altered in the match value box (Figure 17).

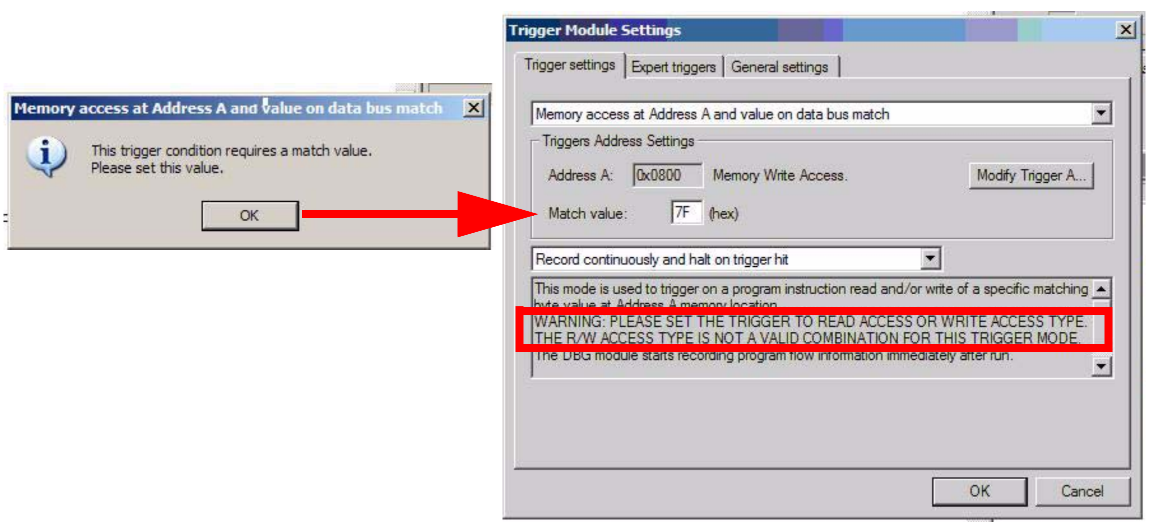

Figure 17. Setting 8-bit Data Value

6. To delete the triggers, right click on the location marked by a trigger. Select delete trigger (in the same manner as deleting a breakpoint [Figure 12]).

6 Conclusion

The BDM is a unique on-chip debug system that improves existing application debugging techniques. It eliminates the need for expensive external emulators and allows for real-time emulation of MCU functions with no limitations on operating voltage or frequency.

A few of the advantages provided by the BDC:

- Non-intrusive debugging through one single pin
- System does not interfere with normal application resources
- Does not use your memory or memory locations in the memory map
- BDC in active or stop mode

Overall, the BDC leads to reduced development time, cost, and debugging techniques.

7 References

AN2104 — Using Background Debug Mode for the M68HC12 Family, by Timothy J. Airaudi

AN2497 — HCS08/RS08 Background Debug Mode versus HC08 Monitor Mode, by Kazue Kikuchi and John Suchyta, updated by Inga Harris.

AN2596 — Using the HCS08 Family On-Chip In-Circuit Emulator (ICE), by Eduardo Montañez

AC127 — Creating your own BDM Pod, by Donnie Garcia (Orlando FTF 2006 Presentation)

HCS08RMv1 — HCS08 Family Reference Manual Volume 1

Freescale S08 Training Workshop Presentation

8 Glossary

BDC — Background debug controller

BDCBKPT — BDC breakpoint register

BDCSCR — BDC status and control register

BDM — Background debug mode

CCR — Condition code register

DBG — Debug Module

EEPROM — Electrically Erasable Programmable Read-Only Memory

EVB — Evaluation board

FIFO — First in first out

H:X — H and X register pair

ICE — In-circuit emulator

MSB — Most significant bit

NC — Not connected

PC — Program counter

R/W — Read/Write

SP — Stack pointer

SPC — Shadow program counter

THIS PAGE IS INTENTIONALLY BLANK

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

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

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

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

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

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3335
Rev. 1
11/2006