Freescale Semiconductor
Application Note

Document Number: AN2497
Rev. 1, 08/2006

HCS08/RS08 Background Debug Mode versus HC08 Monitor Mode

By Kazue Kikuchi and John Suchyta
8/16 Bit Microcontroller Applications Engineering
Austin, Texas

Update to include RS08 7/2006
By Inga Harris
Microcontroller Division Applications Engineering
East Kilbride, Scotland

1 Introduction

Freescale's 8-bit HCS08 and RS08 Family microcontrollers (MCUs) are upwardly compatible with the M68HC08 (HC08) Family. The HCS08 and RS08 Families have an enhanced CPU core that preserves the HC08 CPU registers and offers additional improvements for efficiency, compiler support, and development support.

Regarding microcontroller application development, a good development tool environment is important to reduce total development time and cost. Users want to debug their application program under conditions that imitate the actual setup of their system. Because of that, the capability to debug a user program in an actual target system is required. This is known as in-circuit debugging. Furthermore, most new MCUs have nonvolatile memory such as Flash so that programming code on the target system is also required. This is known as in-circuit programming.

Contents

freescale™ semiconductor

To support in-circuit debugging and programming requirements, the HC08 Family has the monitor mode and the HCS08 and RS08 utilize a background debug mode (BDM). The background debug hardware on the HCS08 consists of a background debug controller (BDC) and debug module (DBG). The background debug hardware on the RS08 consists of the background debug controller (BDC) only.

This application note has three purposes.

1. To describe the differences between the HC08 monitor mode and the HCS08 and RS08 BDM.
2. To introduce the new BDC feature available in the HCS08 and RS08 Families.
3. To describe the differences between the S08 and RS08 BDM interfaces.

2 Differences Between HC08 Monitor Mode and HCS08/RS08 BDM

The HC08 Family has firmware embedded in ROM to support the monitor mode, which is a privileged non-user mode. To enter the monitor mode, specific general-purpose I/O pins must satisfy certain conditions out of reset. A high voltage may also be required on the IRQ pin at the same time. Once the conditions are satisfied, the monitor ROM code is executed instead of running the user application program. A host computer can then communicate with the MCU through a single I/O pin that “bit-bangs” a standard NRZ serial protocol.

The monitor ROM supports six commands; four of them are used for reading/writing memory locations (READ, WRITE, IREAD, and IWRITE commands), the fifth is used for running the user program (RUN command), and the sixth is used for reading the stack pointer (READSP command). Just before the MCU enters the monitor mode, an SWI instruction is executed in the monitor code. To run the user program, the RUN command executes an RTI instruction. While an MCU is in the monitor mode the CPU is used for running the monitor code, so the user program cannot be run at the same time.

The HC08 Family has additional modes for testing a device other than user and monitor modes. These modes are not open to the public. In one of the test modes, data and address buses, and other signals such as a read/write signal, replace I/O signals on the MCU pins. In addition to being able to test a device, this mode enables building high-end emulator tools such as the Freescale FSICE emulator. Since the internal buses can be observed through the MCU pins, an emulator uses this mode to support a bus state analyzer function. However since many I/O pins become unavailable to support those signals, the lost MCU pin functions must be re-built outside the MCU. Therefore, the re-built pin functions may not be exactly the same as the actual MCU. Furthermore, some small pin count MCUs do not have enough pins to support the buses, so it is more difficult to build an emulator.

The HCS08 and RS08 Family has hardware that supports the background debug mode. Specifically, the background debug controller (BDC) was designed to support the background mode. Since the BDC consists of hardware logic, firmware like the HC08 monitor ROM is not embedded inside the MCU. Therefore, the BDC does not need to use the CPU or its instructions. The benefit of this is the BDC can access internal memory even while the user program is running.

In the HC08 Family, the break module, which is independent of the monitor mode, is implemented in the MCU to support the breakpoint and trace functions for the debugger. When the CPU address matches the value written in the break address register, this module generates a software interrupt instruction (SWI).

The HCS08 and RS08 Family does not have this module. Instead, the BDC includes hardware logic to support these functions. Therefore, when the break address matches the value written in the BDC breakpoint register, the mode is switched to the active background mode instead of executing the SWI instruction. The HCS08 BDC also allows access to the on-chip debug module (DBG) that includes more 16-bit comparators for more flexible debugging.

The single-wire background interface pin (BKGD) on HCS08 and RS08 devices is designated as the mode entry and serial communication pin. Since no other pins are used for active background mode entry, none of the MCU pin functions have any limitations.

The HCS08 BDC has a total of 30 background commands. The RS08 BDC has a total of 21 background commands. The CPU registers can be directly accessed by the commands. Some commands can be performed not only in the active background mode but also while the user program is running. Since the BDC uses hardware logic and has a variety of commands it provide a more flexible debugging environment and the HCS08 debugger can support real-time debugging.

In HCS08 and RS08 devices, the BDC is also used for device tests. To support the bus state analyzer function in an emulator, most HCS08 Family members have an on-chip debug system (DBG) in addition to the BDC. This system can capture address or data bus information by setting one of nine trigger modes. In effect, a small bus state analyzer function is built into the MCU, so none of the I/O pins are lost to this function. The details of the DBG function will not be discussed in this note. For more details, refer to the *HCS08 Family Reference Manual* (HCS08RMv1).

Table 1 shows the major differences between HC08 monitor, HCS08 BDM, and RS08 BDM.

Table 1. Differences Between HC08 Monitor Mode and HCS08/RS08 Background Debug Mode

	HC08 Monitor Mode	HCS08 Background Debug Mode	RS08 Background Debug Mode
Mode Entry	• At least four I/O pins required • High voltage required on IRQ pin	• Only BKGD pin required • No high voltage required	• Only BKGD pin required • No high voltage required
Firmware/Hardware	• Firmware embedded in ROM	• Hardware module • BDC registers not in user map	• Hardware module • BDC registers not in user map
Communication Pin	• General-purpose I/O pin	• Dedicated pin (BKGD pin)	• Dedicated pin (BKGD pin)
Communication Protocol	• RS232 type serial (NRZ) • Standard PC communication rates	• Custom serial protocol • Faster • Hardware handshake protocol supported	• Custom serial protocol • Faster • Hardware handshake protocol supported

Table 1. Differences Between HC08 Monitor Mode and HCS08/RS08 Background Debug Mode (continued)

	HC08 Monitor Mode	HCS08 Background Debug Mode	RS08 Background Debug Mode
Commands	Total 5 commands • User program can be executed from the monitor (RUN) • CPU registers can be accessed indirectly • Memory can be accessed in monitor mode but not while user program is running	Total 30 commands: 17 active background mode commands 13 non-intrusive commands • User program can be executed from the active background mode • Active background commands can access CPU registers directly • Non-intrusive commands can access memory not only in active background mode but also while user program is running	Total 21 commands: 10 active background mode commands 10 non-intrusive commands 1 any CPU mode command • User program can be executed from the active background mode • Active background commands can access CPU registers directly • Non-intrusive commands can access memory not only in active background mode but also while user program is running
Breakpoint Function	• One breakpoint function supported by break module	• One hardware breakpoint and one user instruction trace supported by BDC • Two additional flexible breakpoints supported by DBG module	• One hardware breakpoint and one user instruction trace supported by BDC
Timer Counter	• Active in monitor mode	• Inactive in active background mode	• Inactive in active background mode
Stop and Wait Modes	• Monitor mode not available in stop or wait mode	• Background access available in stop3 or wait mode if BDM enabled (ENBDM = 1)	• Background access available in stop3 or wait mode if BDM enabled (ENBDM = 1)
Standard Tool Connector	• 16-pin MON08 connector	• 6-pin BDM connector	• 6-pin BDM connector

The following sections introduce the background debug mode and provide more details about the background debug controller.

3 Background Debug Mode Interface

Freescale has defined a standard 6-pin connector that allows an interface pod to be connected to any target HCS08 or RS08 Family MCU. This connector definition is also used in the HC12 and HCS12 Family MCUs and was derived from the 10-pin interface found on Freescale's high performance 16-bit and 32-bit MCUs. The 6-pin interface has connections for the BKGD pin, $\overline{\text{RESET}}$, V_{DD}, and ground (GND). The BKGD pin does not require an external pullup resistor since it has an on-chip pullup. The connector includes an optional reset signal so that a development system can remotely force a target system reset. A V_{DD} connection is optional and allows the BDM interface pod to get power from or to the target system.

Figure 1 shows the standard BDM connector. This is typically a 2 by 3 pin header with 0.025" square posts on 0.1" centers. Figure 2 shows the BDM implementation in a typical HCS08 target system.

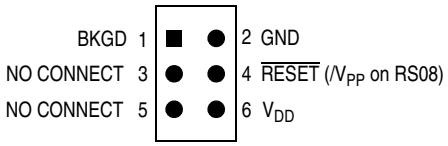

Figure 1. BDM Connector

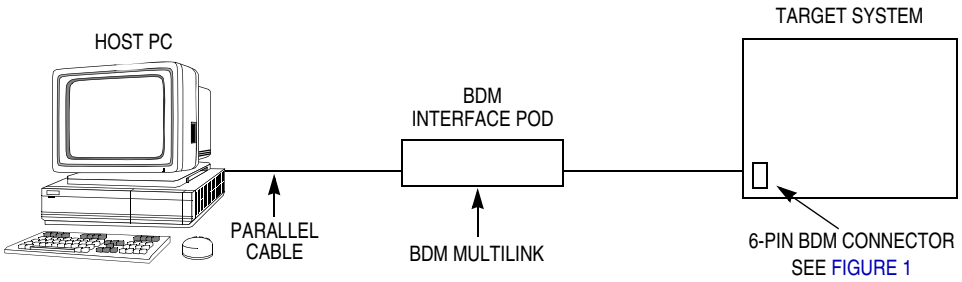

Figure 2. Typical HCS08/RS08 System with BDM Access

The BKGD pin is used for bidirectional communication between an external host, such as a PC or development tool, and the target MCU. To keep this to a single pin, a custom serial protocol was devised. (This protocol is also used as the HC12 and HCS12 Family background communication protocol.) The HCS08 BDC communication clock has two options, the CPU bus clock or a special BDM clock that is defined for each HCS08 derivative. This option allows a host to choose a faster FLL-based bus speed after the FLL has stabilized. For more details, refer to the *HCS08 Family Reference Manual* and the data book for the specific MCU.

There are two main methods to communicate with the target MCU through the BDC. In the reset method, the $\overline{\text{RESET}}$ pin is released after the BKGD and $\overline{\text{RESET}}$ pins are pulled low, and then the BKGD pin is released. In this method, the MCU enters the active background mode instead of the normal user mode. The hot sync method does not require a reset of the MCU. By sending non-intrusive commands to the MCU, the user can communicate with the MCU without disturbing the running application program. Background entry methods will be discussed later.

Brief details of BDC commands are discussed in a later section.

4 HCS08 Background Debug Controller (BDC) and Registers

The major benefit of the BDC is that it does not interfere with normal application resources. It does not share any on-chip peripherals. The single BKGD interface pin is a separate dedicated pin that is not typically accessible to user programs. Background mode does not require high voltage for entry.

The BDC can access internal memory while the user program is running because its hardware is independent of the CPU. The BDC steals a cycle as soon as it can to access the memory. This has little impact on real-time operation of the user's program because a single bus cycle is stolen for each memory access command which requires more than 500 BDC clock cycles. In the HCS08, stealing a cycle means the CPU is suspended for a cycle so the BDC can use the address and data buses to access the requested memory location. However, the CPU suspension does not affect peripheral clocks, such as the timer and serial clocks.

The HCS08 has two registers related to the BDC. One is called the BDC status and control register (BDCSCR). (See Figure 3.) This register contains the enable BDM (ENBDM) bit which permits the active background mode, the communication clock option bit (CLKSW), and several BDC status bits (BDMACT, WS, WSF, and DVF bits). The WS bit indicates whether the target CPU is in wait or stop mode. The WSF bit indicates whether a memory access command failed due to the target CPU executing WAIT or STOP instruction. The DVF bit indicates whether accessed data is valid. The BDCSCR register is not in the user memory map. Since this register can only be accessed by the debugger and not by the user program, it avoids the possibility of enabling the BDM unintentionally while the user program is running. To enable BDM (ENBDM = 1), a BDC WRITE_CONTROL command has to be used.

	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 BDM:	1	1	0	0	1	0	0	0

= Unimplemented or Reserved

Figure 3. BDC Status and Control Register

The other register is called the BDC breakpoint register (BDCBKPT). This 16-bit register contains the address for the BDC hardware breakpoint. Although it can be read and written by the debugger while a user program is running, it is normally only written while in active background mode.

In addition to the above registers, there are two device identification registers that contain a part identification code and mask set version number. These registers are called the system device identification registers (SDIDH:SDIDL) and consist of 2 bytes. This identification code allows the debugger or programmer to select a proper setup associated to a particular target MCU, such as memory map, size, registers, etc. These two registers are accessible to the user program and the debugger.

A special control register called the system background debug force reset register (SBDFR) is also available to the debugger. This register contains a single control bit (BDFR) the debugger can use to force an MCU reset without having to access the $\overline{\text{RESET}}$ pin. This register is accessible only from the active background mode so the bit is protected from unintentional writing in the user program.

5 RS08 Background Debug Controller (BDC) and Registers

The benefits and operation of the RS08 BDC are the same as the HCS08.

The RS08 has two registers related to the BDC. One is called the BDC status and control register (BDCSCR). (See Figure 4.) This register contains the enable BDM (ENBDM) bit which permits the active background mode and several BDC status bits (BDMACT, WS, and WSF bits). The WS bit indicates whether the target CPU is in wait or stop mode. The WSF bit indicates whether a memory access command failed due to the target CPU executing WAIT or STOP instruction. The DVF bit indicates whether accessed data is valid. The BDCSCR register is not in the user memory map. Since this register can only be accessed by the debugger and not by the user program, it avoids the possibility of enabling the BDM unintentionally while the user program is running. To enable BDM (ENBDM = 1), a BDC WRITE_CONTROL command has to be used.

	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

= Unimplemented or Reserved

Figure 4. BDC Status and Control Register

The system background debug force register (SBDRF) is currently not available on RS08 devices.

6 BDC Commands — Active Background Mode and Non-Intrusive

In HCS08 and RS08 MCUs, many flexible operations are included in the background commands. The BDC commands are divided into two distinct groups called “active background mode” and “non-intrusive.” The active background mode commands can only be used when the MCU is not running a user program. On the other hand, the non-intrusive commands can be used when the MCU is in normal user mode or active background mode. These commands do not affect the real-time operation of the user program even when the user program is running.

The RS08 BDC is enhanced with a BDC_RESET command which works in any CPU mode. This command is described in a later section.

As mentioned in the previous section, the BDCSCR register contains the BDC status bits. Some of BDC commands report the contents of the BDCSCR register when they are executed. The status report is very helpful to understand not only the current condition of the CPU (the CPU is in stop or wait mode) but also the validity of the read/write data.

6.1 HCS08 Active Background Mode Commands

6.1.1 READ_CCR, WRITE_CCR, READ_A, WRITE_A, READ_HX, WRITE_HX, READ_SP, WRITE_SP, READ_PC, and WRITE_PC

CPU registers such as accumulator (A), stack pointer (SP), H and X register pair (H:X), program counter (PC), and condition code register (CCR) can be directly read or written by the active background commands. The non-intrusive commands cannot access these registers.

6.1.2 TRACE1

This command is used for tracing one user instruction. (For the HC08, this function is performed by using the break module or SWI instruction.)

6.1.3 GO and TAGGO

These commands are used for executing the user application program and are the same as the HC08 monitor RUN command.

6.1.4 READ_NEXT and WRITE_NEXT

These commands are used in the active background mode for reading and writing data to an address specified in the H:X registers.

6.1.5 READ_NEXT_WS and WRITE_NEXT_WS

These commands are used in the active background mode for reading and writing data to an address specified in the H:X registers. These commands report the contents of the BDCSCR register when they are executed.

6.2 HCS08 Non-Intrusive Commands

6.2.1 WRITE_CONTROL

This command is used for writing the BDCSCR register. This is the only command that allows the user to write the ENBDM bit in the BDCSCR register since BDCSCR is not in the user memory map.

6.2.2 READ_STATUS

This command is used for reading the contents of the BDCSCR register.

6.2.3 BACKGROUND

This command is used for switching from the normal user mode to the active background mode. However if the ENBDM bit is cleared, this command is ignored. When the target CPU is in wait or stop mode, most BDC commands cannot function. However, the BACKGROUND command can be used to force the target CPU out of wait or stop mode and into the active background mode if the BDM is enabled (ENBDM = 1).

6.2.4 SYNC

This command is performed by driving the BKGD pin low for at least 128 cycles of the slowest possible BDC clock. This command is used for detecting a BDC communication speed by receiving a 128-BDM-cycle low pulse on the BKGD pin from the target MCU.

6.2.5 READ_LAST, READ_BYTE, and WRITE_BYTE

Memory access commands can read or write memory not only in the active background mode but also while the user application code is running.

6.2.6 READ_BYTE_WS and WRITE_BYTE_WS

These commands are used for reading and writing data to an address specified by the user. These commands report the contents of the BDCSCR register when they are executed.

6.2.7 READ_BKPT and WRITE_BKPT

These commands are used for reading and writing the BDC breakpoint register. This register is not in the user memory map. One hardware breakpoint is placed in the user program when the BDC breakpoint enable bit (BKPTEN) in BDCSCR is enabled. When a break occurs, the mode is switched from the user mode to the active background mode. Note the MCU does not need to be in the active background mode when the breakpoint is set and enabled. The breakpoint function is available even while the user program is running.

6.2.8 ACK_ENABLE and ACK_DISABLE

These commands can provide an option for a handshake protocol in the BDC communication. When the handshake protocol is enabled, an acknowledge (ACK) pulse is issued when a command is executed.

6.3 RS08 Active Background Mode Commands

6.3.1 READ_CCR_PC, WRITE_CCR_PC, READ_A, WRITE_A, READ_BLOCK, WRITE_BLOCK, READ_SPC, WRITE_SPC

CPU registers such as accumulator (A), shadow program counter (SPC), program counter (PC) and condition code register (CCR) or blocks of data from target memory can be directly read or written by the active background commands. The non-intrusive commands cannot access these registers.

6.3.2 TRACE1

This command is used for tracing one user instruction. (For the HC08, this function is performed by using the break module or SWI instruction.)

6.3.3 GO

This command is used for executing the user application program and are the same as the HC08 monitor RUN command.

6.4 RS08 Non-Intrusive Commands

6.4.1 WRITE_CONTROL

This command is used for writing the BDCSCR register. This is the only command that allows the user to write the ENBDM bit in the BDCSCR register since BDCSCR is not in the user memory map.

6.4.2 READ_STATUS

This command is used for reading the contents of the BDCSCR register.

6.4.3 BACKGROUND

This command is used for switching from the normal user mode to the active background mode. However if the ENBDM bit is cleared, this command is ignored. When the target CPU is in wait or stop mode, most BDC commands cannot function. However, the BACKGROUND command can be used to force the target CPU out of wait or stop mode and into the active background mode if the BDM is enabled (ENBDM = 1).

6.4.4 SYNC

This command is performed by driving the BKGD pin low for at least 128 cycles of the slowest possible BDC clock. This command is used for detecting a BDC communication speed by receiving a 128-BDM-cycle low pulse on the BKGD pin from the target MCU.

6.4.5 READ_BYTE, and WRITE_BYTE

Memory access commands can read or write memory not only in the active background mode but also while the user application code is running.

6.4.6 READ_BYTE_WS and WRITE_BYTE_WS

These commands are used for reading and writing data to an address specified by the user. These commands report the contents of the BDCSCR register when they are executed.

6.4.7 READ_BKPT and WRITE_BKPT

These commands are used for reading and writing the BDC breakpoint register. This register is not in the user memory map. One hardware breakpoint is placed in the user program when the BDC breakpoint enable bit (BKPTEN) in BDCSCR is enabled. When a break occurs, the mode is switched from the user mode to the active background mode. Note the MCU does not need to be in the active background mode when the breakpoint is set and enabled. The breakpoint function is available even while the user program is running.

6.5 RS08 Any CPU Mode Commands

6.5.1 BDC_RESET

This command can be used in any CPU mode to request an MCU reset without using a reset pin.

7 Background Mode Entry

As mentioned in an earlier section, the background debug mode can be entered in several ways. In all cases, a BDM pod of some sort is connected to the BKGD pin. The reset entry method would generally be used when the MCU's memory is not programmed, such that it cannot execute a user program. External hardware or the BDM pod releases the $\overline{\text{RESET}}$ pin after the BKGD and $\overline{\text{RESET}}$ pins are pulled low, and then pulls the BKGD pin high. In this method, the MCU enters the active background mode instead of the normal user mode. A Flash memory programming or erase function can be accomplished at this time. This sequence is typically used to in-circuit program a blank MCU.

The hot sync entry methods are more typically used in a debug session. In these cases the MCU is operating in the normal user mode and an MCU reset is not required. The BDM pod simply sends non-intrusive commands to the MCU through the BKGD pin. The user can communicate with the MCU without disturbing the running application program. Debug operations such as reading or writing a RAM variable can be performed while the program is running. Erasing and programming Flash memory is not performed with these non-intrusive commands.

For more in-depth debugging sessions, the host can command the BDC to enable background mode (ENBDM = 1) and then switch from the running user program to active background mode (program operation stops). This is generally the case when the user wants to debug code by tracing instructions and setting breakpoints. The following three methods are used to enter active background mode.

With background mode enabled (ENBDM = 1), the host can send the BACKGROUND command to the BDC and change the operating mode from user program to active background mode. At this point, all of the active background mode and non-intrusive BDC commands can be used by the debugger.

Alternatively, the MCU can switch from the user program to active background mode by executing the BGND instruction after the host enables background mode. If the background mode is not enabled (ENBDM = 0) when the CPU comes to a BGND instruction, it will be treated as a NOP and program execution will continue.

Finally, the MCU can switch from the user program to active background mode after the host enables both background mode and breakpoints (BKPTEN = 1) and then the CPU encounters a breakpoint set in the BDC breakpoint register (BDCBKPT). As above, all active background mode and non-intrusive BDC commands can then be used by the debugger.

8 Development Tools

HCS08 and RS08 development tools using the BDC are currently available. P&E Microcomputer Systems, Inc. (P&E) developed a BDM interface pod and cable called USB MULTILINK, which supports multiple voltages and frequencies. This pod communicates to an MCU through a standard 6-pin BDM

connector. A standard USB port is used for communication between the host and interface pod. The USB MULTILINK supports the HCS08, RS08 and the HCS12 Families. For more information, please refer to P&E's web site http://www.pemicro.com.

An upgraded version of P&E's popular MON08 CYCLONE tool is available to support the HCS08 (and soon available for RS08). The CYCLONE PRO has the capability to perform standalone programming in addition to the debugging and programming functions of the MULTILINK. The CYCLONE PRO connects to a PC or host through a serial port, a USB port, or an Ethernet port.

Softec Microsystems have an In-System Programmer and Debugger tool which supports HC08, HCS08, RS08, S12, and S12X Families. Softec Microsystems web site is http://www.softecmicro.com

HCS08 and RS08 integrated development software is also available. Freescale's CodeWarrior (Figure 5) and P&E's WinIDE provide complete development environments that support debugging and Flash programming in the combination with the USB MULTILINK. The Codewarrior web site is http://www.codewarrior.com.

Please visit http://www.freescale.com regularly for the latest tools news.

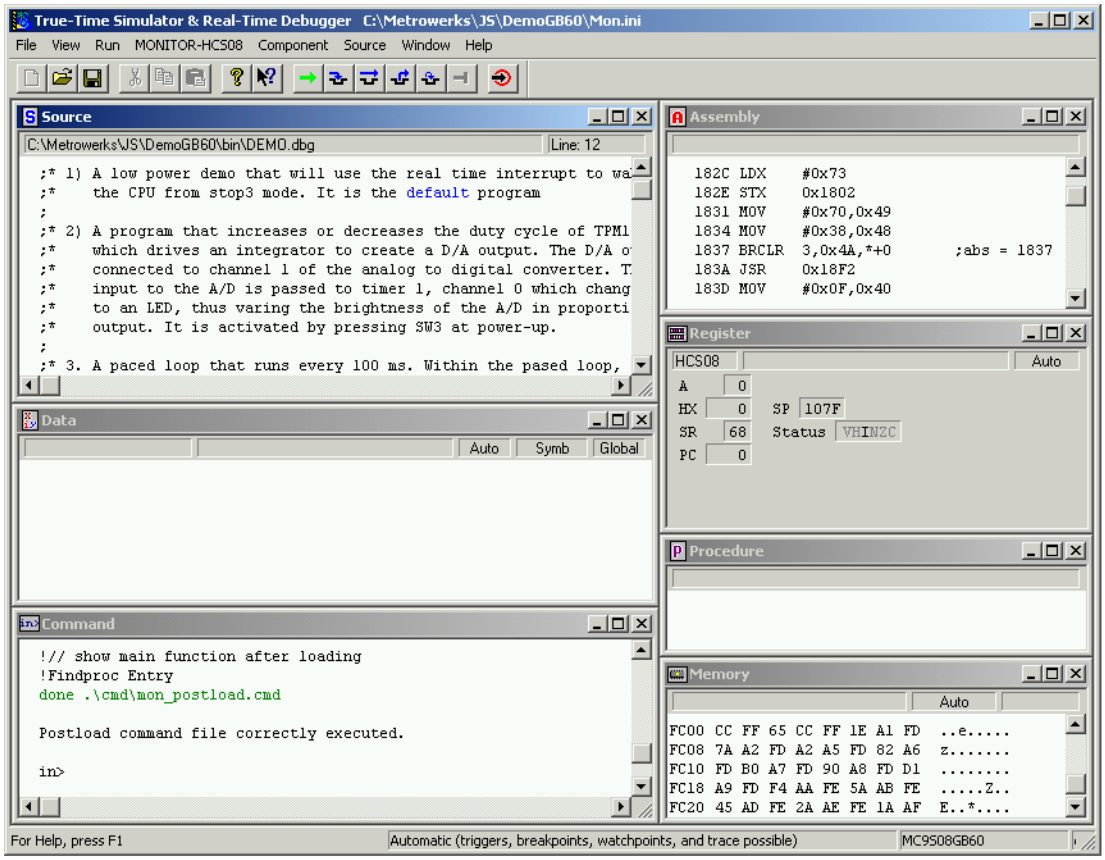

Figure 5. CodeWarrior Debugger

9 Summary

The HC08 monitor mode and the HCS08/RS08 background debug mode support in-circuit debugging and programming. The HCS08 and RS08 background debug controller (BDC) provides more advanced level of the development environment because:

1. Non-intrusive debug access.
2. No MCU pin function limitations.
3. Breakpoint and trace functions are built into the BDC.
4. Custom serial protocol is faster and more speed tolerant.
5. BDC can be active in stop or wait mode.
6. Background mode interface is small and simple.

The above benefits reduce the total development complexity, time, and cost. Furthermore, MCU application developers can build their code in the actual application setup.

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: AN2497
Rev. 1
08/2006

freescale™
semiconductor