Introduction to HCS08 Background Debug Mode

by: Steven McLaughlin
Applications Engineer
East Kilbride, Scotland

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.

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

---

![Figure 3. BDM Connector Pins](image)

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
Overview

- 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>
<thead>
<tr>
<th>RS08 Commands</th>
<th>S08 Commands</th>
<th>mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNC</td>
<td>SYNC</td>
<td>non-intrusive</td>
<td>Request a timed reference pulse to determine target BDC communication speed</td>
</tr>
<tr>
<td>ACK_ENABLE</td>
<td>ACK_ENABLE</td>
<td>non-intrusive</td>
<td>Enable handshake; issues an ACK pulse after the command is executed</td>
</tr>
<tr>
<td>ACK_DISABLE</td>
<td>ACK_DISABLE</td>
<td>non-intrusive</td>
<td>Disable handshake; command does not issue an ACK</td>
</tr>
<tr>
<td>BACKGROUND</td>
<td>BACKGROUND</td>
<td>non-intrusive</td>
<td>Enters active background mode if enabled</td>
</tr>
<tr>
<td>READ_STATUS</td>
<td>READ_STATUS</td>
<td>non-intrusive</td>
<td>Read BDC status from BDCSCR</td>
</tr>
<tr>
<td>RS08 Commands</td>
<td>S08 Commands</td>
<td>mode</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>---------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>WRITE_CONTROL</td>
<td>WRITE_CONTROL</td>
<td>non-intrusive</td>
<td>Write BDC status from BDCSCR</td>
</tr>
<tr>
<td>READ_BYTE</td>
<td>READ_BYTE</td>
<td>non-intrusive</td>
<td>Read a byte from target memory</td>
</tr>
<tr>
<td>READ_BYTE_WS</td>
<td>READ_BYTE_WS</td>
<td>non-intrusive</td>
<td>Read a byte and report status</td>
</tr>
<tr>
<td>__</td>
<td>READ_LAST</td>
<td>non-intrusive</td>
<td>Re-read byte from address just read and report status</td>
</tr>
<tr>
<td>WRITE_BYTE</td>
<td>WRITE_BYTE</td>
<td>non-intrusive</td>
<td>Write a byte to target memory</td>
</tr>
<tr>
<td>WRITE_BYTE_WS</td>
<td>WRITE_BYTE_WS</td>
<td>non-intrusive</td>
<td>Write a byte and report status</td>
</tr>
<tr>
<td>READ_BKPT</td>
<td>READ_BKPT</td>
<td>non-intrusive</td>
<td>Read BDCBKPT breakpoint register</td>
</tr>
<tr>
<td>WRITE_BKPT</td>
<td>WRITE_BKPT</td>
<td>non-intrusive</td>
<td>Write BDCBKPT breakpoint register</td>
</tr>
<tr>
<td>GO</td>
<td>GO</td>
<td>active background mode</td>
<td>Go to execute the user application program starting at the address currently in the PC</td>
</tr>
<tr>
<td>TRACE1</td>
<td>TRACE1</td>
<td>active background mode</td>
<td>Trace 1 user instruction at the address in the PC, then return to active background mode</td>
</tr>
<tr>
<td>__</td>
<td>TAGGO</td>
<td>active background mode</td>
<td>Same as GO but enable external tagging</td>
</tr>
<tr>
<td>READ_A</td>
<td>READ_A</td>
<td>active background mode</td>
<td>Read accumulator (A)</td>
</tr>
<tr>
<td>__</td>
<td>READ_CCR</td>
<td>active background mode</td>
<td>Read condition code register (CCR)</td>
</tr>
<tr>
<td>READ_CCR_PC</td>
<td>__</td>
<td>active background mode</td>
<td>Read CCR PC</td>
</tr>
<tr>
<td>__</td>
<td>READ_PC</td>
<td>active background mode</td>
<td>Read program counter (PC)</td>
</tr>
<tr>
<td>__</td>
<td>READ_HX</td>
<td>active background mode</td>
<td>Read H and X register pair (H:X)</td>
</tr>
</tbody>
</table>

Table 1. BDC Command Summary (BDCSCR) (continued)
<table>
<thead>
<tr>
<th>RS08 Commands</th>
<th>S08 Commands</th>
<th>mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>READ_SP</td>
<td>active background mode</td>
<td>Read stack pointer (SP)</td>
</tr>
<tr>
<td>READ_SPC</td>
<td>_</td>
<td>active background mode</td>
<td>Read shadow program counter (SPC)</td>
</tr>
<tr>
<td>_</td>
<td>READ_NEXT</td>
<td>active background mode</td>
<td>Increment H:X by one, then read memory byte located at H:X.</td>
</tr>
<tr>
<td>_</td>
<td>READ_NEXT_WS</td>
<td>active background mode</td>
<td>Increment H:X by one, then read memory byte located at H:X. Report status and data</td>
</tr>
<tr>
<td>READ_BLOCK</td>
<td>_</td>
<td>active background mode</td>
<td>Blocks of data read from target memory</td>
</tr>
<tr>
<td>WRITE_A</td>
<td>WRITE_A</td>
<td>active background mode</td>
<td>Write accumulator (A)</td>
</tr>
<tr>
<td>_</td>
<td>WRITE_CCR</td>
<td>active background mode</td>
<td>Write CCR</td>
</tr>
<tr>
<td>WRITE_CCR_PC</td>
<td>_</td>
<td>active background mode</td>
<td>Write CCR PC</td>
</tr>
<tr>
<td>_</td>
<td>WRITE_PC</td>
<td>active background mode</td>
<td>Write PC</td>
</tr>
<tr>
<td>_</td>
<td>WRITE_HX</td>
<td>active background mode</td>
<td>Write H:X</td>
</tr>
<tr>
<td>_</td>
<td>WRITE_SP</td>
<td>active background mode</td>
<td>Write SP</td>
</tr>
<tr>
<td>WRITE_SPC</td>
<td>_</td>
<td>active background mode</td>
<td>Write SPC</td>
</tr>
<tr>
<td>_</td>
<td>WRITE_NEXT</td>
<td>active background mode</td>
<td>Increment H:X by one, then write memory byte located at H:X.</td>
</tr>
<tr>
<td>_</td>
<td>WRITE_NEXT_WS</td>
<td>active background mode</td>
<td>Increment H:X by one, then write memory byte located at H:X. Report status</td>
</tr>
<tr>
<td>WRITE_BLOCK</td>
<td>_</td>
<td>active background mode</td>
<td>Blocks of data written from target memory</td>
</tr>
</tbody>
</table>

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.

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Function</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>ENBDM – Enable BDM</td>
<td>Written to 1 by debug host when debugging</td>
<td>BDM cannot be made active (non-intrusive commands only)</td>
</tr>
<tr>
<td>6</td>
<td>BDMACT – background active status</td>
<td>Read only status bit</td>
<td>BDM inactive</td>
</tr>
<tr>
<td>5</td>
<td>BKPTEN – BDC breakpoint enable</td>
<td>If 0 BDC breakpoint disabled and FTS and BDCBKPT are ignored</td>
<td>BDC breakpoint disabled</td>
</tr>
<tr>
<td>4</td>
<td>FTS – Force/Tag select</td>
<td>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.</td>
<td>Tag opcode at breakpoint address and enter active background mode if CPU attempts to execute that instruction</td>
</tr>
</tbody>
</table>
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.
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.
Figure 8. Main Compiler Window

1. Debug button is circled

Introduction to HCS08 Background Debug Mode, Rev. 1

Freescale Semiconductor 11
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).
Introduction to HCS08 Background Debug Mode, Rev. 1

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.
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).
Setting Triggers

Introduction to HCS08 Background Debug Mode, Rev. 1

Freescale Semiconductor

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)

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 14. Modify Trigger Settings

Figure 15. Trigger A and B Settings
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).

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
Glossary

MSB — Most significant bit
NC — Not connected
PC — Program counter
R/W — Read/Write
SP — Stack pointer
SPC — Shadow program counter
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

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2006. All rights reserved.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to http://www.freescale.com/epp.