Development and Debug Limitation on MC9S08GB60(A)

Affects all MC9S08GB60 and MC9S08GB60A MCUs

by: Jim Sibigtroth
Microcontroller Division
Freescale Semiconductor

There are four known issues in the MC9S08GB60(A) that affect the on-chip ICE system (DBG module) and the interaction of the background debug controller (BDC) with the timer/PWM system. None of these issues affects normal operation of the MCU when no BDM debug pod is connected. Logic changes have been made to newer HCS08 devices so that these limitations are eliminated.

1 On-Chip DBG Module Fails to Terminate Properly When Stopped by SWI

In end-trace runs where the BEGIN control bit is 0, the act of disarming the DBG module (using a normal CPU access as opposed to a BDM access) causes an extra unexpected capture of one word into the debug FIFO. Development software, such as CodeWarrior® and the Freescale-provided resident serial monitor described in AN2140, is patched to correct for this extra FIFO word (so debugging displays show only valid information). This limitation does not affect debugging with a BDM pod, which is much more common than using a serial monitor.

1.1 Details

This issue occurs only when DBGEN, ARM, TAG, BRKEN, and TRGSEL = 1 and BEGIN = 0. When the trigger point is reached, the CPU responds as expected by generating an SWI rather than entering active background mode (because there is no BDM pod connected). However, the debug module remains armed (ARM = 1) and the AF and/or BF status flag does not get set as expected.

The ARM bit stays set and the flag does not get set because the instruction pipeline rebuild logic in the DBG module interprets the SWI as an application interrupt; so it operates as if the instruction stream was interrupted just before the instruction that caused the trigger could be executed. In other trigger modes and trace runs where BEGIN = 1, the FIFO is disarmed as a result of the flags getting set; therefore, the flags are set and ARM is already cleared before the SWI occurs. When a BDM pod is used, trace runs are terminated by entering active background mode, so there is no interrupt to be misinterpreted. This limitation has been eliminated in newer HCS08 silicon.

1.2 Workaround

The Freescale serial monitor program writes a 0 to ARM whenever the monitor program gains control of the target MCU due to an SWI or serial I/O interrupt. Writing 0 to manually disarm the DBG module causes the FIFO to effectively capture one extra word of (non-useful) information into the FIFO. If the FIFO was already full, this extra shift causes the oldest word of useful information to be lost. In addition, CodeWarrior debugger software recognizes this unusual termination of a trace run and discards the extra word of capture data so the user sees only valid information in debugger displays.

Ideally, the user should disable the periodic update feature of the CodeWarrior debugger when the serial monitor is used (rather than a BDM pod). This can be done by right-clicking in the visualization or memory window, select options, and change the refresh mode to automatic. This will prevent the debugger from performing hidden background accesses to the target MCU to update the debugger display screen. The serial monitor has been revised to save the ARM state and restore it upon return to the application program so that if a breakpoint is set, it will still operate correctly although the information from the on-chip ICE FIFO buffer may be less useful than if a BDM pod were being used for debugging.

2 Unexpected Double Capture of Some Read Accesses in Event-Only Trace Runs

In “event-only” debug trace runs where the on-chip debug FIFO is capturing read data accesses rather than change-of-flow addresses, some CPU free cycles are unexpectedly detected as data read accesses. This causes what appears to be double-captures of some read accesses.

2.1 Details

CPU free cycles are bus cycles during which the CPU is performing some internal operation and does not need the data bus. During these cycles, the address and data buses stay the same as they were for the previous bus cycle. If this address is an address that the on-chip DBG module is trying to capture, the on-chip ICE system will capture an extra copy of the data into the FIFO. This happens most commonly during read-modify-write instructions such as INC, DEC, BSET, and BCLR where there is a free cycle immediately after a data read cycle. This limitation has been eliminated in newer HCS08 silicon.

2.2 Workaround

Although there is no workaround to prevent these extra data capture events, it is relatively easy to recognize and ignore these capture events during debugging. The user typically knows the instruction that is expected to cause a data capture. By looking up this instruction in the instruction set details pages of the HCS08 Family Reference Manual and looking at the access detail, you can tell whether there is a free cycle (f-type cycle) immediately after the expected data read cycle (r-type cycle).

For example, the access detail for the direct addressing version of the DEC instruction is “rfwpp” and there is an f cycle immediately after the r cycle where the memory location is read. If the conditions for this error are met during a DEC instruction, the user will observe that the f cycle will have the same address and data of the previous read access. The R/W signal will still indicate a read cycle, so the on-chip ICE will erroneously capture this as another data read cycle, even though the CPU is not really reading the data.

Most instructions do not have a free cycle immediately after a ready cycle. For example, the access detail for the direct addressing version of the LDA instruction is “rpp” so the data read cycle (r) is immediately followed by a program fetch cycle (p), so the on-chip ICE would capture only the single data read event.

3 BDM Commands Cannot Be Used to Set Up a New PWM Waveform

If a user attempts to use BDM commands to set up the timer/PWM module to produce a PWM signal, the PWM signal will not appear at the MCU pin as expected.

3.1 Details

Changes to PWM waveforms are “buffered,” meaning changes are delayed until all associated parameters have been written. This ensures that the PWM will transition gracefully from the old settings to the new settings (and the timer will not attempt to produce a PWM cycle that is controlled by an old period and a new duty cycle, for example).

In addition, there is coherency logic in the 16-bit registers so that both 8-bit halves of these registers must be accessed before the new value takes effect. To avoid the possibility that a background debug memory access to 16-bit timer registers could erroneously interfere with user application code sequences that treat these registers as coherent 16-bit values, the coherency logic is inhibited during BDM memory accesses. For most debugging situations, this is helpful. However, if a user tries to set up a PWM waveform on a timer channel using only BDM accesses to set up the timer registers, the PWM waveform will not appear at the MCU pin as expected because the buffering logic does not detect the BDM accesses that set up the PWM period and duty cycle.

3.2 Workaround

Trace application program instructions to make changes to PWM settings. If you must change PWM settings from the debugger rather than the application code, you can use the single-line assembler in the debugger to insert PWM modification instructions in an unused RAM area (such as below the current stack). Then trace these instructions with the debugger. This limitation has been corrected in newer HCS08 silicon.

4 BDM Commands Can Interfere with Input Capture Events

When a debugger is set up to access timer value registers periodically in the background and a BDM access to an input capture register occurs at the same time as an actual input capture event, then the input capture flag gets set as expected, but the BDM access prevents the input capture register from getting updated with a new timer count value.

4.1 Details

The logic equation for loading the timer value register with the timer count value when an input capture event occurs, includes a term that blocks this update when the BDC is accessing the register. If these two events happen at the same time, the BDC access takes priority over the input capture event; therefore, the register is not updated with the new time value as expected. The input capture flag is still set as expected.

These two events rarely occur at the same time. However, the CodeWarrior debugger can be configured to perform background accesses to the target MCU to update debug screens while an application program is running. This increases the chance that a BDC access will coincide with an actual input capture event.

4.2 Workaround

If this is seen as a problem during application debug, you can configure the debugger to not perform background accesses while the application program is running. To do this, right-click in the memory window of the CodeWarrior debugger, select options, and change the refresh mode to automatic. In this refresh mode, the debugger will perform BDC accesses to target memory only when the application code stops (for example at a breakpoint) or when the user explicitly requests the contents of a target system memory location. This limitation has been corrected in newer HCS08 silicon.