MSE9S08RE16_0L78P
Rev. 7, 8/2006

Mask Set Errata for Mask 0L78P

Introduction

This mask set errata applies to the mask 0L78P for these products:

- MC9S08RE16
- MC9S08RE8
- MC9S08RD16
- MC9S08RD8
- MC9S08RC16
- MC9S08RC8

MCU Device Mask Set Identification

The mask set is identified by a 5-character code consisting of a version number, a letter, two numerical digits, and a letter, for example 0L78P. All standard devices are marked with a mask set number and a date code.

PWM Boundary Case Issues in HCS08 Timer PWM Module (TPM) SE110-TPM

This errata describes boundary case issues that primarily affect the center-aligned PWM mode of operation. While investigating these issues, additional, less significant, issues were discovered. These will be explained, although they should not cause any significant problems in normal applications.

In center-aligned PWM mode, the timer counter counts up until it reaches the modulo value in TPMMODH:TPMMODL, reverses direction, and then counts down until it reaches zero, where it reverses and counts up again. A period of the PWM output is centered around the leading edge of the zero count and the period is considered to start when the count changes from TPMMODH:TPMMODL–1 to TPMMODH:TPMMODL (the same point where the counter changes from up-counting to down-counting). The zero value and the maximum modulo value occur for only one timer count cycle each, while all other values occur twice (once during the down-counting phase and again during the up-counting phase). Therefore, the total period of the PWM signal is two times the value in TPMMODH:TPMMODL.

The value on each TPM timer output pin is controlled by an internal flip-flop that is cleared at reset but is not readable by software. These internal flip-flops change state when timer output compare events or PWM duty cycle compare events occur (when the channel value registers match the timer count registers). This leads to these outputs remaining in a previous state until a compare event occurs after changing the configuration of the timer system. When the timer is initialized the first time after a reset, the state of these output flip-flops is known to be reset (logic low). If the configuration is changed after the channel has been running in another configuration for some period of time, you sometimes do not know the state of these internal flip-flops (and therefore the state of the timer output pins) until a new channel value register compare event occurs. There is nothing improper about these periods before the first event occurs, however some users might be surprised the first time they notice this behavior.

When the MCU is reset, the count (TPMCNTH:TPMCNTL) is reset to 0x0000. If the timer is configured for center-aligned pulse-width modulation (PWM) and then the clock is started, this corresponds to the middle of a PWM period. If the internal flip-flop corresponding to the output was at the inactive level when the PWM started, this would appear as if there was an extra half period of delay before the first full PWM cycle started. If the internal flip-flop corresponding to the output happened to be at the active level when this PWM was started, a pulse equivalent to half of a normal duty cycle pulse could be produced at the PWM output pin.

There are eight cases discussed in this errata:

- Cases 1 and 2 — These are two error cases near the 100% duty cycle boundary. The first is when the channel value registers are set equal to the modulo value. The second is when the channel value registers are set to one less than the modulo value.
- Cases 3, 4, and 5 — These cases are related to changing the channel value to or from 0x0000. The errors depend upon whether this is done during the first or second half of the center-aligned PWM period. In all of these cases, the workaround strategy is to produce 0% duty cycle with a negative channel value instead of using the 0x0000 value. This can be done by checking any value that is about to be written to the channel value registers, and then decrement the 16-bit value or the high-order byte of the value before writing it to the channel value registers. This produces the desired 0% duty cycle and avoids the problems related to a zero in the channel value registers.

- Case 6 — Although this behavior wasn't discussed in the data sheet, the operation is different than some users might expect. In edge-aligned PWM mode, when the channel value is changed from zero to a non-zero value, the new PWM settings can take an extra half PWM period to take effect. It is unlikely that this would cause any problems in any practical application system.
- Case 7 — This case is more of a clarification of an unusual situation rather than a design problem. This case happens when the prescale factor is changed during operation and only affects center-aligned PWM. It would be very unusual to change the prescale setting after it is set during reset initialization. The prescale flip-flops are not reset when the prescale setting is changed, so the first prescaled clock period after a change may be shorter or longer than expected.
- Case 8 — This case would only arise when a series of unlikely events happened to occur. It affects only center-aligned PWM mode if the timer counter is stopped, reset, and restarted when the count value happened to be equal to the TPMxMODH:TPMxMODL value. Because the timer counter would not normally be stopped during operation in center-aligned PWM mode, this case should never arise in a practical application.

Case 1: Center-Aligned PWM
Channel Value (TPMxCnVH:TPMxCnVL) = Modulo Value (TPMxMODH:TPMxMODL)

Description

This should produce 100% duty cycle where the TPM output pin remains at the active level continuously. Instead, the output remains at the inactive level, which corresponds to 0% duty cycle.

Workarounds

Check any value that is about to be written to the channel value registers. If the value is the same as the modulo value, increment the value before writing it to the channel value register. This workaround will work for any modulo value that is greater than zero and less than 0x7FFF. Setting the channel value to any 2's complement negative value (0x8000 through 0xFFFF) results in 0% duty cycle as expected and described in the original TPM documentation.

Another workaround would be to choose not to use 100% duty cycle in the application. Not all applications require the range to include the 100% duty cycle case.

Case 2: Center-Aligned PWM
Channel Value (TPMxCnVH:TPMxCnVL) = Modulo Value Minus 1 (TPMxMODH:TPMxMODL – 1)

Description

This should produce almost 100% duty cycle where the TPM output pin remains at the active level for [(TPMxCnVH:TPMxCnVL $\times$ 100)/(TPMxMODH:TPMxMODL)]% of the period. Instead, the output remains at the inactive level which corresponds to 0% duty cycle.

Workarounds

Reduce the prescale factor by a factor of two and then multiply the modulo and channel value settings by a factor of two. In this way, the frequency and resolution of the PWM output remain the same but channel values are always even numbers and are never equal to the modulo setting minus one.

Consider the case of a 20-MHz bus frequency, 25-kHz PWM frequency, and 0.25% resolution on the duty cycle. Before making the adjustments suggested in this workaround, you could have the following setup: Set the modulo to 400 and the prescale factor in PS2:PS1:PS0 to divide by 2 (0:0:1). Each step of the channel value from 0–1–2…398–399–400 would increase the duty cycle by 0.25%.

Increasing the modulo value to 800 and reducing the prescale factor to divide-by one, would still produce the same period or PWM frequency. If the original channel values were multiplied by two (shift left one bit position) before writing them to the channel value register, the resolution would still be 0.25% per step of the channel value, but the channel values would step by 2 each time as in 0-2-4-6…796-798-800. With this workaround, the channel value would never be equal to the modulo value minus one, and the error condition would not arise.

With common HCS08 bus frequencies, practical PWM frequencies, and reasonable resolution requirements, there is enough speed and flexibility in the TPM system so this workaround should work well with all except the most unusual application systems.

Another workaround would be to limit the range of allowed values in the channel value register so it does not include the TPMxMODH:TPMxMODL or (TPMxMODH:TPMxMODL – 1) values. Not all applications require the range to include these values.

Case 3: Center-Aligned PWM TPMxCnVH:TPMxCnVL Changed from 0x0000 to a Non-Zero Value

Description

This case occurs only while the counter is counting down (first half of the center-aligned PWM period). The PWM output changes to the active level at the middle of the current PWM period as the count reaches 0x0000 instead of waiting for the start of a new PWM period to begin using the new duty cycle setting.

Workaround

Use a negative channel value instead of 0x0000 to produce 0% duty cycle. This can be done by checking any value that is about to be written to the channel value registers, and then decrementing the 16-bit value or the high-order byte of this value before writing it to the channel value registers. This produces the desired 0% duty cycle and it avoids the problems related to a zero in the channel value registers.

Case 4: Center-Aligned PWM TPMxCnVH:TPMxCnVL Changed from a Non-Zero Value to 0x0000

Description

This case occurs only while the counter is counting up (second half of the center-aligned PWM period) but before the count reaches the channel value setting in TPMxCnVH:TPMxCnVL. The PWM output remains at the active level until the end of the current PWM period instead of finishing the current PWM period using the old channel value setting.

Workaround

Use a negative channel value instead of 0x0000 to produce 0% duty cycle. This can be done by checking any value that is about to be written to the channel value registers, and then decrement the 16-bit value or the high-order byte of this value before writing it to the channel value registers. This produces the desired 0% duty cycle and it avoids the problems related to a zero in the channel value registers.

Case 5: Center-Aligned PWM TPMxCnVH:TPMxCnVL Changed from 0x0000 to a Non-Zero Value

Description

This case occurs only while the counter is counting down (first half of the center-aligned PWM period) and then TPMxCnVH:TPMxCnVL is changed back to 0x0000 during the first half of the next PWM period (while the counter is counting down). This is a very unlikely case in any practical application. The PWM output changes to the active level at the middle of the first PWM period as the count reaches 0x0000 instead of waiting for the start of a new PWM period to begin using the new duty cycle setting, and then the output remains active until the end of the second PWM period. In this very unusual case, the PWM output remains active for one and one-half PWM periods rather than remaining inactive for the first PWM period and then active for 2 $\times$TPMxCnVH:TPMxCnVL during the next PWM period.

Workaround

Use a negative channel value instead of 0x0000 to produce 0% duty cycle. This can be done by checking any value that is about to be written to the channel value registers, and then decrementing the 16-bit value or the high-order byte of this value before writing it to the channel value registers. This produces the desired 0% duty cycle and it avoids the problems related to a zero in the channel value registers.

Case 6: Edge-Aligned PWM TPMxCnVH:TPMxCnVL Changed from 0x0000 to a Non-Zero Value

Description

This is a minor issue related to edge-aligned PWM when duty cycle is changed from 0x0000 to a non-zero value. This issue is a specification clarification rather than a design error.

In this case, the channel value update occurs at the same time as the new PWM period begins, but due to circuit delays, the update occurs slightly too late for the new duty cycle to take effect for that PWM period and an extra period of 0% duty cycle is produced. This causes the new PWM duty cycle to take effect one PWM period later than expected. This should not cause any application problems so the data book functional description will be changed to clarify this situation.

Case 7: Changing the Counter Prescaler while the TPM Counter Is Disabled

Description

This case would not arise in most applications because it would be unusual to change the prescaler at any time other than initial timer setup after reset.

1. TPM counter was previously running
2. Counting is stopped by writing 0:0 to CLKS[1:0]
3. Change prescale value PS[2:1:0] to a different value while keeping clocks off (CLKS[1:0] = 0:0)
4. Clear the counter by writing any value to TPMxCNTH:TPMxCNTL
5. Turn clocks back on by writing to CLKS[1:0]

Unexpected Operation: The prescaler divider flip-flops begin counting from the prior value rather than starting from zero. This can result in the counter detecting the first clock edge after restarting, either earlier or later than expected.

Case 8: Center-Aligned PWM, Counter is Stopped, Reset, and Restarted when Counting Up and Count Equals the Modulo Value

Description

This case is extremely unlikely to occur in any practical application because it would be very unusual to stop or reset the TPM counter while using center-aligned PWM mode.

1. TPM counter is counting up in center-aligned PWM mode (second half of a PWM period)
2. Counter is stopped (write CLKS[1:0] = 0:0) when count equals modulo value (the direction would normally change from up counting to down counting at the next clock edge)
3. Counter is reset to 0x0000 by writing any value to TPMxCNTH:TPMxCNTL
4. Counter is turned on again by writing to CLKS[1:0]

Unexpected Operation: Because the internal up/down indicator was not cleared when the counter was reset, the counter begins counting down from 0x0000 to 0xFFFF–0xFFFE… This causes the timing of the first PWM period after the counter reset to be longer than expected.

Possible High Current in Stop Mode If KBI/IRQ Enabled

SE96B-IRQ_KBI

Description

In stop mode, with the IRQ or KBI pin functions enabled but the IRQ/KBI interrupt disabled, a toggle on the IRQ/KBI pin will turn on the voltage regulator and oscillator, causing the clocks to run. This results in higher stop I_{DD}. In this condition, the CPU is halted (as if in wait mode) and the chip level interrupt is not generated. This higher current condition can also happen if the flag is set at time of stop entry.

Workaround

This problem applies to the non-interrupt functions that are shared with IRQ and KBI functions. Because interrupts from IRQ or KBI are disabled, these pins are not used to wake up the MCU from stop. To prevent signals on IRQ or KBI from causing an unexpected partial wakeup from stop in these applications, disable the IRQ and KBI modules just before entering stop mode. Also verify that any previous interrupt flags associated with IRQ and KBI have been cleared.

Active Edge on IRQ or KBI May Not Be Detected

SE97-IRQ_KBI

Description

When the IRQ or KBI modules are enabled, an active (falling or rising, depending on configuration settings) edge on the IRQ or KBI pin may not be detected if it transitions exactly two bus clock edges before the bus clock ceases upon stop mode entry. The IRQ/KBI interrupt will not occur and the MCU will remain in stop mode. Subsequent active edges on the IRQ or KBI pin will generate interrupts and wake the MCU from stop mode.

Propagation Delay on PTA0

SE99B-PTA0

Description

When the PTA0 pin is enabled as a general-purpose input, rising edges will take longer to propagate through to the PTA data register compared to all other port pins. This delay is worst case at V_{DD} less than 2 V and cold temperatures. This pin should not be used to capture timing sensitive input edges on this version of silicon.

This is a known issue and will be fixed on the next revision of silicon.

Workaround

Use any port input pin other than PTA0 for timing sensitive inputs.

HCS08 with External Oscillator — Falling Edge on $\overline{\text{RESET}}$ During Termination of Reset Events

SE94B-Reset

Description

If a falling edge is detected on the external reset pin at either of two sample points, the MCU will cease operation and will recover only after a power-on reset.

Case 1 — After detecting a reset signal, either internal or external, the internal reset circuit latches the cause of the reset, forces a low level on the external $\overline{\text{RESET}}$ pin for about 34 bus cycles, releases the pin, and samples the pin level about 38 bus cycles later.

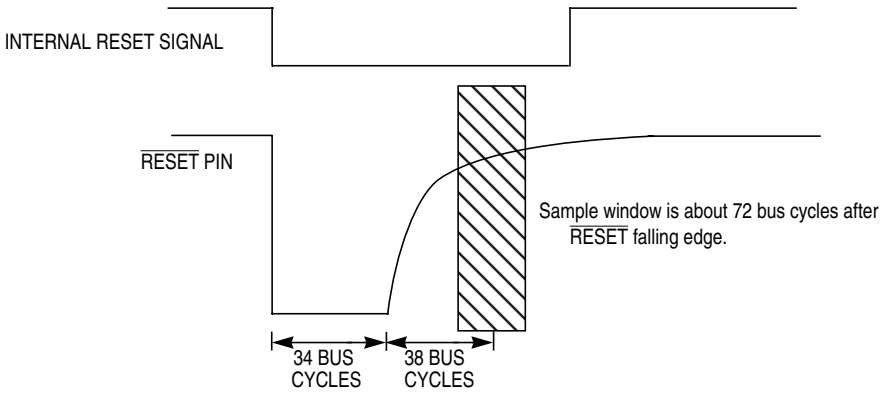
Case 2 — After remaining in reset for an extended period of time due to an externally applied low level on $\overline{\text{RESET}}$, the internal reset circuit will sample the pin level about 9 to 10 bus cycles after the pin raises above its V_{IH} level.

Workaround — Case 1

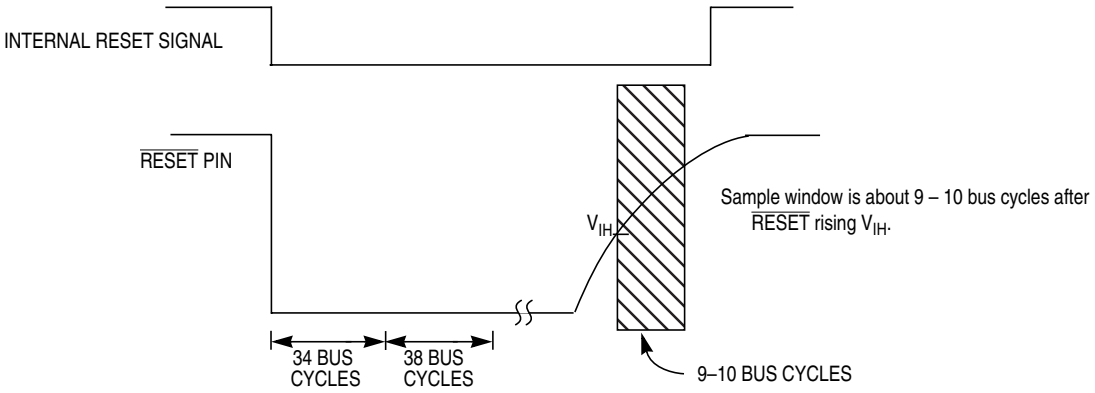
To avoid multiple edges on the $\overline{\text{RESET}}$ pin due to switch contact ringing or EMC noise, eliminate any voltage bounce through RC filtering. A 0.1 µF capacitor to ground and an external 4.7 kΩ to 10 kΩ resistor to V_{DD} works well to ensure noise suppression.

Workaround — Case 2

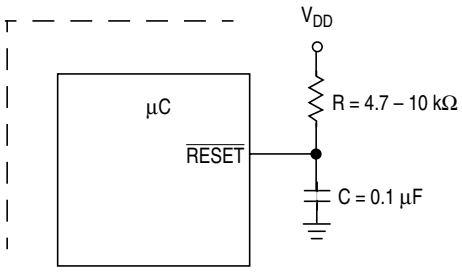
In addition to RC filtering, avoid spurious edges on the $\overline{\text{RESET}}$ pin caused by external active circuitry such as low voltage detectors and watchdog components. Do this by ensuring that external circuitry cannot drive $\overline{\text{RESET}}$ low within 10 bus cycles of the release of $\overline{\text{RESET}}$.


INTERNAL RESET SIGNAL
RESET PIN
Sample window is about 72 bus cycles after RESET falling edge.
34 BUS CYCLES
38 BUS CYCLES

Case 1


INTERNAL RESET SIGNAL
RESET PIN
VIH
Sample window is about 9 – 10 bus cycles after RESET rising VIH.
34 BUS CYCLES
38 BUS CYCLES
9–10 BUS CYCLES

Case 2


VDD
µC
RESET
R = 4.7 – 10 kΩ
C = 0.1 µF

Out of specification conditions in stop2 and stop3 modes due to error in PTD0/BKGD/MS and PTD1/$\overline{\text{RESET}}$ pins

SE82-SIDD

Detailed Description:

It has been observed that PTD0/BKGD/MS and/or PTD1/$\overline{\text{RESET}}$ pins, in certain configurations, will cause increased stop3 current ($S3I_{DD}$) and stop2 current ($S2I_{DD}$) to be consumed by the device, exceeding the specifications. It has also been observed that the device will not exit from stop2 mode with a KBI1, IRQ, or RTI interrupt in certain configurations.

The configurations that cause these behaviors are as follows:

- If PTD0/BKGD/MS is left in its default condition (BKGD) and pulled low, $S3I_{DD}$ and $S2I_{DD}$ will exceed the specifications.
- If PTD1/$\overline{\text{RESET}}$ is left in its default condition ($\overline{\text{RESET}}$) and pulled low, $S3I_{DD}$ and $S2I_{DD}$ will exceed the specifications.
- If PTD0/BKGD/MS is set as an output and driven low, $S3I_{DD}$ and $S2I_{DD}$ will exceed the specifications.
- If PTD1/$\overline{\text{RESET}}$ is set as an output and driven low, $S3I_{DD}$ and $S2I_{DD}$ will exceed the specifications.
- If PTD1/$\overline{\text{RESET}}$ is set as an output and is driven low, the device may not exit stop2 mode with a KBI1, IRQ, or RTI interrupt. The device will exit stop3 mode with a KBI1, KBI2, IRQ, or RTI interrupt.
- If PTD0/BKGD/MS is set as an output, is driven low, and PTD1/$\overline{\text{RESET}}$ is in default condition ($\overline{\text{RESET}}$), the device may not exit stop2 mode with a KBI1, IRQ, or RTI interrupt. It may, however, enter active BDM mode after stop2 recovery because in the failure case, the PTD0/BKGD/MS pin may float internally to “zero.” (Stop2 recovery forces a reset condition). The device will exit stop3 mode with a KBI1, KBI2, IRQ, or RTI interrupt.

When PTD0/BKGD/MS and PTD1/$\overline{\text{RESET}}$ are left in the default reset configuration and allowed to be pulled high by their internal pull-up device and/or pulled high by an external pull-up resistor, they function per the specification in any run or stop mode. All stop mode I_{DD}s are then within specification.

Workaround:

There is no known workaround for this issue if either of these pins is configured as output driven low while stop3 or stop2 mode is used.

If one or both of these pins are used as outputs, software must drive them high before entering any stop mode.

A possible workaround is to use other pins as outputs instead of these pins. This may require using a higher pin-count device to provide the extra output pins.

These out-of-specification conditions will vary depending on voltage, temperature, and normal fabrication process variances.

This issue and its fix have been identified and will be applied to future mask sets.

Analog Comparator and Internal Bandgap Reference Setting LVD Flag

SE57-ACMP

Detailed Description

When the analog comparator (ACMP) is enabled and the internal bandgap reference is selected, and the non-inverting input (PTD5) is either connected to a logic 1 or driven to a logic 1, the low voltage detect (LVD) flag will be set. An LVD interrupt will occur, if enabled. If LVD reset is enabled, it will occur as well.

Workaround

When using the internal bandgap reference, follow this procedure to avoid setting the LVD flag and LVD interrupt and reset, if enabled.

1. Configure port D, bit 5, to be an output low. To do this:
   a. Set its corresponding port D data register bit to a 0.
   b. Change the data direction register bit for port D, bit 5, to a 1 (output).
2. Enable the comparator and bandgap reference by setting bits 7 and 6 in ACMPSC.

Now, any user action may be applied to PTD5 and no low voltage warning (and interrupt or reset) will be generated.

Data Written to an I/O Port Register Immediately After Reset or Stop Recovery May Be Corrupted

SE52-OSC

Detailed Description:

It has been observed if an I/O port on the MC9S08RC Family is written very quickly after RESET and/or STOP recovery, the data in its data and/or data direction register could be corrupted.

Oscillator start-up times can vary based on the system using a crystal or resonator. Crystals and resonators from different manufacturers have different start-up timing characteristics. Resonators generally start more quickly than crystals.

The oscillator used on this MCU employs a start-up delay of 1024 cycles of the oscillator frequency after minimum threshold has been detected and before the internal bus clock is enabled. This delay occurs any time the oscillator is started, which is after RESET and after STOP recovery. It has been concluded the 1024 oscillator cycle delay is not sufficient to allow the oscillator to reach its final amplitude before enabling the clock to the bus. Before the oscillator has approached its final amplitude, I/O switching noise can cause a glitch in the oscillator output. Increasing the delay from 1024 oscillator cycles to 4096 oscillator cycles, before accessing any I/O, corrects the issue.

Workaround:

Execute a 1536 minimum bus cycle delay immediately after RESET and immediately after STOP recovery.

The following is an example of source code that will create the delay:

```
        clrx            ; 1 cycle                       \
        clrh            ; 1 cycle                        \
?loop?  aix     #1      ; 2 cycles                        | 3084 osc cycles
        cphx    #220    ; 2 cycles (220 decimal)   /  192.75 µs @ 16 MHz xtal
        bne     ?loop?  ; 3 cycles                /
;  1542 total bus cycles or 3084 osc cycles
;       delay is complete
```