MSE9S08AW60_1M75B
Rev. 3, 8/2006

Mask Set Errata for Mask 1M75B

Introduction

This mask set errata applies to the mask 1M75B for these products:

- MC9S08AW60
- MC9S08AW48
- MC9S08AW32
- MC9S08AW16

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 1M75B. 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.

When FLL is Engaged, ICG May Experience Locking Issues

SE107-ICG_FLL

Description

An issue exists with the internal clock generator module (ICG) where, when the frequency-locked loop (FLL) is engaged and the device is operating at certain ICGDCLK frequencies, it may be unable to obtain a lock or unexpectedly lose lock. In addition, the FLL may be unable to maintain a lock, which will cause a toggling FLL lock (a condition where the FLL continuously locks and then unlocks).

The FLL clock mode is engaged using the CLKS[1:0] bit field. When CLKS[1:0] = 01, the FLL is engaged while using the internal reference clock; this mode is referred to as FLL engaged internal (FEI). When CLKS[1:0] = 11, the FLL is engaged while using an external reference clock; this mode is referred to as FLL engaged external (FEE). This loss-of-lock issue affects both the FEI and FEE modes at certain frequencies.

The FLL, when engaged, is used to multiply the external or internal clock source to higher frequencies. An output of the FLL is the DCO clock or ICGDCLK. Valid ICGDCLK frequencies are between 8 and 40 MHz. The loss-of-lock issue has been identified to occur when the FLL is configured with an ICGDCLK frequency within one of the following ranges:

- 8.0 to 10.8 MHz
- 12.3 to 15.8 MHz
- 20.3 to 24.8 MHz
- 33.7 to 40.0 MHz

The ICGDCLK frequency ranges provided account for process variation and temperature effects between –40°C and 125°C.

At these ICGDCLK frequencies, with the FLL engaged, the ICG may exhibit high jitter during operation. The jitter is caused by nonlinearity in a current DAC (digital-to-analog converter). This nonlinearity causes the output frequency to jump more than expected for a single-bit control word change, causing the FLL to lose lock.

If the ICG experiences an unexpected loss-of-lock or the inability to maintain lock of the FLL, the configuration of the LOLRE bit (loss-of-lock reset enable) in the ICGC2 register determines how the MCU will respond:

- If LOLRE = 1, generate a reset request
- If LOLRE = 0, generate an interrupt request

In the case of toggling FLL lock, cascading interrupts or resets should be expected. Note, the ICG interrupt is not maskable.

The FLL lock status is indicated by the LOCK bit. An unexpected loss-of-lock for the FLL is indicated by the LOLS bit. Both the LOCK and LOLS bit are located in ICGS1 (ICG status register 1).

Workarounds

Two workarounds are available for this issue.

Workaround 1: Select Different Bus Frequency

Engage the FLL, but avoid MFD[2:0] configurations that yield ICGDCLK values where the loss-of-lock has been observed. Table 1 and Table 2 provide acceptable MFD[2:0] values for both FEE and FEI modes.

Table 1. FEE Mode Acceptable MFD[2:0] Values

DCO output = ICGDCLK
ICGOUT = $f_{ext} \times P \times N \div R$ = ICGDCLK ÷ R
where N = 2 × MFD[2:0] + 4 and P = 64 if $f_{ext} \leq 100$ kHz or P= 1 if f_{ext} >100 kHz
Bus frequency = ICGOUT ÷ 2

MFD[2:0]	N	Clock Input Value (MHz)				
		Low Frequency		High Frequency		
		0.032768	0.1	2	4	8
		ICGDCLK (MHz)				
000	4	8.39	25.60	8.00	16.00	32.00
001	6	12.58	38.40	12.00	24.00	out of spec
010	8	16.78	out of spec	16.00	32.00	out of spec
011	10	20.97	out of spec	20.00	40.00	out of spec
100	12	25.17	out of spec	24.00	out of spec	out of spec
101	14	29.36	out of spec	28.00	out of spec	out of spec
110	16	33.55	out of spec	32.00	out of spec	out of spec
111	18	37.75	out of spec	36.00	out of spec	out of spec

= Avoid configurations that result in shaded ICGDCLK values.

Table 2. FEI Mode Acceptable MFD[2:0] Values
(for an internal reference clock source trimmed to 243 kHz)

DCO output = ICGDCLK
ICGOUT = ($f_{icg} \div 7$) × 64 × N ÷ R = ICGDCLK ÷ R, where N = 2 × MFD[2:0] + 4
Bus frequency = ICGOUT ÷ 2

MFD[2:0]	N	ICGDCLK (MHz)
000	4	8.89
001	6	13.33
010	8	17.77
011	10	22.22
100	12	26.66
101	14	31.10
110	16	35.55
111	18	39.99
	f_{icg} (MHz)	0.243

= Avoid configurations that result in shaded ICGDCLK values.

In Table 1 and Table 2, R is related to the bit field RFD[2:0] (reduced frequency divider), i.e., $R = 2^{RFD[2:0]}$. RFD[2:0] is found in the ICGC2 register (ICG control register 2). The value of R has no effect on the FLL loss-of-clock issue.

Workaround 2: Use Alternative ICG Mode

Do not engage the FLL. The ICG can be used in two modes where the FLL is bypassed. These two modes are self-clocked mode (SCM) and FLL bypassed external clock mode (FBE).

- SCM
  - 3 MHz < f_{Bus} < 5 MHz (default) or 3 MHz < f_{Bus} < 20 MHz (via filter bits)
  - Medium power
- FBE
  - f_{Bus} range ≤ 8 MHz when a crystal or resonator is used
  - Lowest power mode
  - Highest clock accuracy
  - Crystal, resonator, or external clock source required

Negative Voltage on IRQ Pin Results in High I_{DD} Currents

SE108-IDD

Description

During latch up testing, a negative voltage (referenced to the V_{SS} pin) applied to the IRQ pin has been observed to damage the V_{SS} circuitry. This results in run I_{DD} and/or stop I_{DD} currents exceeding the maximum specification limits. In all cases, the MCU remained fully functional and no latch up condition was observed.

Latch up testing involves applying voltages that are outside of the absolute maximum ratings of the device to the MCU pins. This is to determine the capability of the MCU to withstand unintentional excessive stresses. The applied voltage that causes the IRQ pin damage is outside of the specified input voltage limits of V_{DD} + 0.3 V to V_{SS} – 0.3 V and results in a current pull of 100 mA from the IRQ pin. Higher temperatures contribute to the issue. No damage was observed on devices tested at 25°C. Increasing the temperature, particularly above 85°C, increases probability of pin damage. Freescale's HCS08 Family of microcontrollers normally can withstand such stress without damaging the device.

Workaround

Pin damage can be avoided by adhering to the specifications in the published data sheet. Limit the voltage applied to the IRQ pin to the absolute maximum limits from V_{DD} + 0.3 V to V_{SS} – 0.3 V. Also, instantaneous current on this pin must be limited to the absolute maximum limits of ± 25 mA.

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.

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

SE94A-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. The ICG will revert to self-clock mode (SCM) at the beginning of the reset cycle. Therefore, the first sample window is about 13 µs to 26 µs after the initial reset signal, due to the frequency range and characteristics of the internal oscillator.

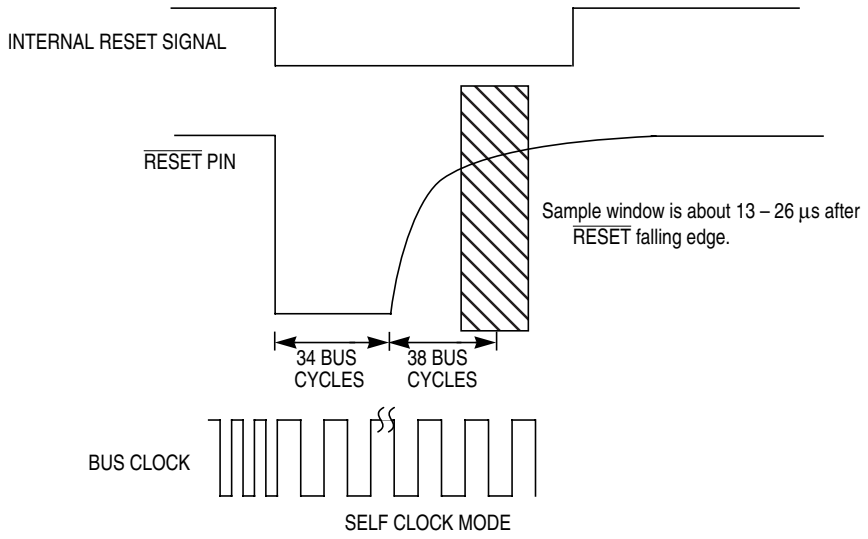
Case 2 — After remaining in reset for an extended period of time due to an externally applied low level on $\overline{\text{RESET}}$, the internal reset circuit will sample the pin level about 9 to 10 bus cycles after the pin raises above its V_{IH} level. The ICG will revert to self-clock mode (SCM) during reset. Therefore, the second sample window is about 1.7 µs to 3.6 µs after the reset signal rises above V_{IH}, due to the frequency range and characteristics of the internal oscillator.

Workaround — Case 1

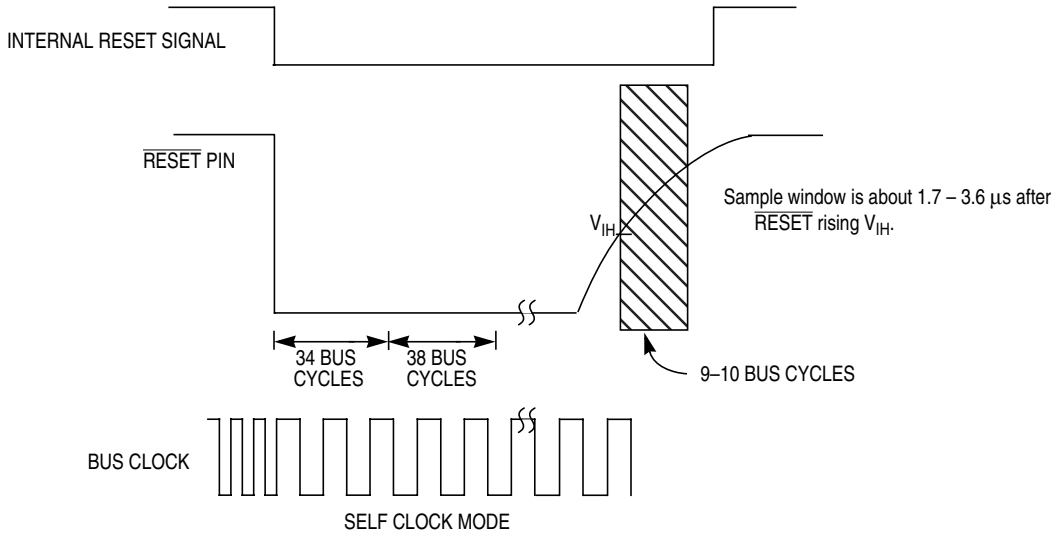
To avoid multiple edges on the $\overline{\text{RESET}}$ pin due to switch contact ringing or EMC noise, eliminate any voltage bounce through RC filtering. A 0.1 µF capacitor to ground and an external 4.7 kΩ to 10 kΩ resistor to V_{DD} works well to ensure noise suppression.

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 $\overline{\text{RESET}}$.



Case 1



Case 2

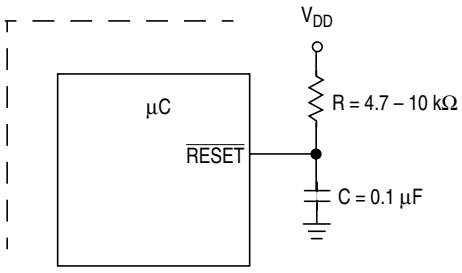


Figure 1. HCS08 SIM Reset Lockup