Stellaris® LM3S8962 RevA2 Errata

This document contains known errata at the time of publication for the Stellaris® LM3S8962 microcontroller. The table below summarizes the errata and lists the affected revisions. See the data sheet for more details.

See also the ARM® Cortex™-M3 errata, ARM publication number PR141-PRDC-007452 v9.0.

Erratum Number	Erratum Title	Revision(s) Affected
1.1	JTAG pins do not have internal pull-ups enabled at power-on reset	A1, A2
2.1	JTAG INTEST instruction does not work	A1, A2
3.1	Clock source incorrect when waking up from Deep-Sleep mode in some configurations	A1, A2
3.2	PLL may not function properly at default LDO setting	A1, A2
3.3	Hard Fault possible when waking from Sleep or Deep-Sleep modes and Cortex-M3 Debug Access Port (DAP) is enabled	A1, A2
3.4	I/O buffer 5-V tolerance issue	A1, A2
4.1	Hibernation module WAKE input pin does not work as specified	A1
4.2	Hibernation module low VBAT detection does not work as expected	A1, A2
4.3	Performing a system-wide reset also resets the Hibernation module and all of its registers	A1, A2
4.4	Hibernation module may have higher current draw than specified in data sheet under certain conditions	A1, A2
4.5	Hibernation module returns from the Hibernation state to the Wake state regardless of the status of the VDD supply to the microcontroller	A1, A2
4.6	Certain Hibernation module register writes cause RTC Counter register inaccuracy	A1, A2
4.7	Hibernation module state retention registers may corrupt after Wake sequence	A1, A2
5.1	MERASE bit of the FMC register does not erase the entire Flash array	A1, A2
6.1	GPIO input pin latches in the Low state if pad type is open drain	A1, A2
6.2	GPIO pins may glitch during power supply ramp up	A1, A2
7.1	General-purpose timer Edge Count mode count error when timer is disabled	A1, A2
7.2	General-purpose timer 16-bit Edge Count mode does not load reload value	A1, A2
8.1	Use of "Always" triggering for ADC Sample Sequencer 3 does not work	A1, A2
8.2	Incorrect behavior with timer ADC triggering when another timer is used in 32-bit mode	A2
9.1	CAN register accesses require software delays	A1, A2
10.1	Number of Packets decremented early	A1, A2
11.1	PWM pulses cannot be smaller than dead-band time	A1, A2

Erratum Number	Erratum Title	Revision(s) Affected
11.2	PWM interrupt clear misses in some instances	A1, A2
11.3	PWM generation is incorrect with extreme duty cycles	A1, A2
11.4	PWMINTEN register bit does not function correctly	A1, A2
12.1	QEI index resets position when index is disabled	A1, A2
12.2	QEI hardware position can be wrong under certain conditions	A1, A2

1 JTAG and Serial Wire Debug

1.1 JTAG pins do not have internal pull-ups enabled at power-on reset

Description:

Following a power-on reset, the JTAG pins $\overline{\mathtt{TRST}}$, `TCK`, `TMS`, `TDI`, and `TDO` (`PB7` and `PC[3:0]`) do not have internal pull-ups enabled. Consequently, if these pins are not driven from the board, two things may happen:

- The JTAG port may be held in reset and communication with a four-pin JTAG-based debugger may be intermittent or impossible.
- The receivers may draw excess current.

Workaround:

There are a number of workarounds for this problem, varying in complexity and impact:

1. Add external pull-up resistors to all of the affected pins. This workaround solves both issues of JTAG connectivity and current consumption.
2. Add an external pull-up resistor to $\overline{\mathtt{TRST}}$. Firmware should enable the internal pull-ups on the affected pins by setting the appropriate `PUE` bits of the appropriate **GPIO Pull-Up Select (GPIOPUR)** registers as early in the reset handler as possible. This workaround addresses the issue of JTAG connectivity, but does not address the current consumption other than to limit the affected period (from power-on reset to code execution).
3. Pull-ups on the JTAG pins are unnecessary for code loaded via the SWD interface or via the serial boot loader. Loaded firmware should enable the internal pull-ups on the affected pins by setting the appropriate `PUE` bits of the appropriate **GPIOPUR** registers as early in the reset handler as possible. This method does not address the current consumption other than to limit the affected period (from power-on reset to code execution).

Silicon Revision Affected:

A1, A2

2 JTAG

2.1 JTAG INTEST instruction does not work

Description:

The JTAG INTEST (Boundary Scan) instruction does not properly capture data.

Workaround:

None.

Silicon Revision Affected:

A1, A2

3 System Control

3.1 Clock source incorrect when waking up from Deep-Sleep mode in some configurations

Description:

In some clocking configurations, the core prematurely starts executing code before the main oscillator (MOSC) has stabilized after waking up from Deep-Sleep mode. This situation can cause undesirable behavior for operations that are frequency dependent, such as UART communication.

This issue occurs if the system is configured to run off the main oscillator, with the PLL bypassed and the DSOSCSRC field of the **Deep-Sleep Clock Configuration (DSLPCLKCFG)** register set to use the internal 12-MHz oscillator, 30-KHz internal oscillator, or 32-KHz external oscillator. When the system is triggered to wake up, the core should wait for the main oscillator to stabilize before starting to execute code. Instead, the core starts executing code while being clocked from the deep-sleep clock source set in the **DSLPCLKCFG** register. When the main oscillator stabilizes, the clock to the core is properly switched to run from the main oscillator.

Workaround:

Run the system off of the main oscillator (MOSC) with the PLL enabled. In this mode, the clocks are switched at the proper time.

If the main oscillator must be used to clock the system without the PLL, a simple wait loop at the beginning of the interrupt handler for the wake-up event should be used to stall the frequency-dependent operation until the main oscillator has stabilized.

Silicon Revision Affected:

A1, A2

3.2 PLL may not function properly at default LDO setting

Description:

In designs that enable and use the PLL module, unstable device behavior may occur with the LDO set at its default of 2.5 volts or below (minimum of 2.25 volts). Designs that do not use the PLL module are not affected.

Workaround:

Prior to enabling the PLL module, it is recommended that the default LDO voltage setting of 2.5 V be adjusted to 2.75 V using the **LDO Power Control (LDOPCTL)** register.

Silicon Revision Affected:

A1, A2

3.3 Hard Fault possible when waking from Sleep or Deep-Sleep modes and Cortex-M3 Debug Access Port (DAP) is enabled

Description:

If the Cortex-M3 Debug Access Port (DAP) has been enabled, and the device wakes from a low power sleep or deep-sleep mode, the core may start executing code before all clocks to peripherals have been restored to their run mode configuration. The DAP is usually enabled by software tools accessing the JTAG or SWD interface when debugging or flash programming. If this condition occurs, a Hard Fault is triggered when software accesses a peripheral with an invalid clock.

Workaround:

A software delay loop can be used at the beginning of the interrupt routine that is used to wake up a system from a WFI (Wait For Interrupt) instruction. This stalls the execution of any code that accesses a peripheral register that might cause a fault. This loop can be removed for production software since the DAP is most likely not enabled during normal execution.

Since the DAP is disabled by default (power on reset), the user can also power cycle the device. The DAP will not be enabled unless it is enabled through the JTAG or SWD interface.

Silicon Revision Affected:

A1, A2

3.4 I/O buffer 5-V tolerance issue

Description:

GPIO buffers are not 5-V tolerant when used in open-drain mode. Pulling up the open-drain pin above 4 V results in high current draw.

Workaround:

When configuring a pin as open drain, limit any pull-up resistor connections to the 3.3-V power rail.

Silicon Revision Affected:

A1, A2

4 Hibernation Module

4.1 Hibernation module WAKE input pin does not work as specified

Description:

The Hibernation module $\overline{\text{WAKE}}$ input pin is used by an external device to trigger a wake-up event and bring the device out of Hibernate mode. This input signal does not behave properly when the device goes into Hibernate mode, and therefore, does not properly wake up the device from hibernation.

Workaround:

The device must be configured to wake up from hibernate mode using the Real-Time-Clock Wake-up by setting the `RTCWEN` bit in the **Hibernation Control (HIBCTL)** register. Disable the External Event Wake-up Enable by clearing the `EXTWEN` bit in the **HIBCTL** register.

Silicon Revision Affected:

A1

4.2 Hibernation module low VBAT detection does not work as expected

Description:

The Hibernation module is designed to detect a low V_{BAT} condition. This feature is enabled by setting the `LOWBATEN` bit in the **Hibernation Control (HIBCTL)** register. When enabled, the low V_{BAT} detection incorrectly detects a low V_{BAT} condition at 3.15 V. This is supposed to trigger a low V_{BAT} condition at 2.35 V.

Workaround:

This feature should not be used and must be disabled by clearing the `LOWBATEN` bit in the **HIBCTL** register. The battery voltage can be sensed using a hardware workaround by wiring an analog comparator input pin to the battery and setting the comparator reference pin voltage to the desired detection trip level.

Silicon Revision Affected:

A1, A2

4.3 Performing a system-wide reset also resets the Hibernation module and all of its registers

Description:

Performing a system-wide reset by asserting the $\overline{\text{RST}}$ pin, encountering a Brown-Out Reset, or setting the SYSRESETREQ bit in the ARM Cortex-M3 Application Interrupt and Reset Control register also incorrectly causes the Hibernation module to be reset. All of the Hibernation module registers are reset, including the non-volatile **Hibernation Data (HIBDATA)** and the **Hibernation RTC Counter (HIBRTCC)** registers.

Workaround:

None.

Silicon Revision Affected:

A1, A2

4.4 Hibernation module may have higher current draw than specified in data sheet under certain conditions

Description:

If a battery voltage is applied to the VBAT power pin prior to power being applied to the VDD power pins of the device, the current draw from the VBAT pin is greater than expected. The current may be as high as 1.6 mA instead of the data sheet specified 17 µA. The condition exists until power is applied to the VDD pin. Once the VDD pin has been powered, the VBAT current draw functions as expected. The VDD pin can then be powered up and down as required and the VBAT pin current specification is maintained.

Workaround:

The VBAT pin higher-than-specified current draw condition can be avoided if the microcontroller's VDD power pins are powered on prior to the time a battery voltage is initially applied to the VBAT pin.

Silicon Revision Affected:

A1, A2

4.5 Hibernation module returns from the Hibernation state to the Wake state regardless of the status of the VDD supply to the microcontroller

Description:

When the Hibernation module is in Hibernation mode and receives an event to initiate a wake-up (assertion of the $\overline{\text{WAKE}}$ pin, RTC match, or low-battery detect), the $\overline{\text{HIB}}$ signal is de-asserted, enabling the external regulator to provide V_{DD} to the device. The device then performs a Power-On Reset (POR) and the software can query the **Hibernation Raw Interrupt Status (HIBRIS)** register to determine if a hibernation wake occurred.

If the regulator does not have a power source, or for some reason V_{DD} is not immediately applied to the device when the $\overline{\text{HIB}}$ signal is de-asserted, the Hibernation module still exits from its Hibernation state to its Wake state, but the device is not able to perform its POR sequence. If power is then restored to the regulator (providing V_{DD} to the device), the device executes a POR, however, the state of the Hibernation module is incorrectly reset, and all of the registers within the Hibernation module, including the **Hibernation RTC Counter (HIBRTCC)** and **Hibernation Data (HIBDATA)** registers are set to their reset state.

Workaround:

Always ensure that power is available to the regulator controlled by the $\overline{\text{HIB}}$ signal when the $\overline{\text{HIB}}$ signal is de-asserted during a wake event.

Silicon Revision Affected:

A1, A2

4.6 Certain Hibernation module register writes cause RTC Counter register inaccuracy

Description:

The Hibernation module contains a **Hibernation RTC Counter (HIBRTCC)** register that keeps an accumulated running one-second count. This register is updated from an internal, 32-KHz counter which is not visible to the user. As this counter value rolls over, it provides a tick count once per second to the **HIBRTCC** register. Register writes to the **Hibernation RTC Match (HIBRTCM0, HIBRTCM1)** registers, the **Hibernation RTC Trim (HIBRTCT)** register, or the **Hibernation Data (HIBDATA)** register cause the 32-KHz counter to reset to its start value. The result is that the **HIBRTCC** register value loses accuracy each time one of these Hibernation registers is written because the 32-KHz counter is reset in the midst of its one-second count window, effectively delaying the **HIBRTCC** register counter value from its ideal value.

If the regulator does not have a power source, or if V_{DD} is not immediately applied to the device when the $\overline{\text{HIB}}$ signal is de-asserted, the Hibernation module exits from its Hibernation state to its Wake state, but the device cannot perform its POR sequence. If power is then restored to the regulator (providing V_{DD} to the device), the device performs a POR, however, the state of the Hibernation module is then incorrectly reset. This means that all of the registers within the Hibernation module, including the **Hibernation RTC Counter (HIBRTCC)** and **Hibernation Data (HIBDATA)** registers are set to their Reset state.

Workaround:

To minimize the amount of introduced error, the application can write to the affected registers immediately after the RTC rolls over. The RTC rollover can be detected by polling the RTC value in the **HIBRTCC** register and waiting for it to change. If the application writes to the affected registers with a predictable and repeatable pattern, then the **HIBRTCT** register can be used to compensate for introduced error.

Silicon Revision Affected:

A1, A2

4.7 Hibernation module state retention registers may corrupt after Wake sequence

Description:

When transitioning from a Hibernate condition to a Wake-up state, the **Hibernation Control (HIBCTL)** register and **Hibernation Interrupt Mask (HIBIM)** register values may occasionally become corrupted. This situation occurs after a wake-up event caused by an RTC match condition or an external signal asserting the $\overline{\text{WAKE}}$ pin of the device.

Workaround:

Software should mirror the data in the **HIBCTL** and **HIBIM** registers to the **Hibernation Data (HIBDATA)** register before initiating a Hibernation sequence. Immediately upon returning from the Hibernation state to the Wake-up state, software should write the data mirrored from the HIBDATA registers back into the **HIBCTL** and **HIBIM** registers.

Silicon Revision Affected:

A1, A2

5 Flash Controller

5.1 MERASE bit of the FMC register does not erase the entire Flash array

Description:

The `MERASE` bit of the **Flash Memory Control (FMC)** register does not erase the entire Flash array. If the contents of the **Flash Memory Address (FMA)** register contain a value less than 0x20000, only the first 128 KB of the Flash array are erased. If bit 17 (value of 0x20000) is set, then only the upper address range of Flash (greater than 128 KB) is erased.

Workaround:

If the entire array must be erased, the following sequence is recommended:

1. Write a value of 0x00000000 to the **FMA** register.
2. Write a value of 0xA4420004 to the **FMC** register, and poll bit 2 until it is cleared.
3. Write a value of 0x00020000 to the **FMA** register.
4. Write a value of 0xA4420004 to the **FMC** register, and poll bit 2 until it is cleared.

The entire array can also be erased by individually erasing all of the pages in the array.

Silicon Revision Affected:

A1, A2

6 GPIO

6.1 GPIO input pin latches in the Low state if pad type is open drain

Description:

GPIO pins function normally if configured as inputs and the open-drain configuration is disabled. If open drain is enabled while the pin is configured as an input using the **GPIO Alternate Function Select (GPIOAFSEL)**, **GPIO Open Drain Select (GPIOODR)**, and **GPIO Direction (GPIODIR)** registers, then the pin latches Low and excessive current (into pin) results if an attempt is made to drive the pin High. The open-drain device is not controllable.

A GPIO pin is not normally configured as open drain and as an input at the same time. A user may want to do this when driving a signal out of a GPIO open-drain pad while configuring the pad as an input to read data on the same pin being driven by an external device. Bit-banging a bidirectional, open-drain bus (for example, I^2C) is an example.

Workaround:

If a user wants to read the state of a GPIO pin on a bidirectional bus that is configured as an open-drain output, the user must first disable the open-drain configuration and then change the direction of the pin to an input. This precaution ensures that the pin is never configured as an input and open drain at the same time.

A second workaround is to use two GPIO pins connected to the same bus signal. The first GPIO pin is configured as an open-drain output, and the second is configured as a standard input. This way the open-drain output can control the state of the signal and the input pin allows the user to read the state of the signal without causing the latch-up condition.

Silicon Revision Affected:

A1, A2

6.2 GPIO pins may glitch during power supply ramp up

Description:

Upon completing a POR (power on reset) sequence, the GPIO pins default to a tri-stated input condition. However, during the initial ramp up of the external V_{DD} supply from 0.0 V to 3.3 V, the GPIO pins are momentarily configured as output drivers during the time the internal LDO circuit is also ramping up. As a result, a signal glitch may occur on GPIO pins before both the external V_{DD} supply and internal LDO voltages reach their normal operating conditions. This situation can occur when the V_{DD} and LDO voltages ramp up at significantly different rates. The LDO voltage ramp-up time is affected by the load capacitance on the `LDO` pin, therefore, it is important to keep this load at a nominal 1 µF value as recommended in the data sheet. Adding significant more capacitance loading beyond the specification causes the time delay between the two supply ramp-up times to grow, which possibly increases the severity of the glitching behavior.

Workaround:

Ensuring that the V_{DD} power supply ramp up is a fast as possible helps minimize the potential for GPIO glitches. Follow guidelines for LDO pin capacitive loading documented in the electrical section of the data sheet. System designers must ensure that, during the V_{DD} supply ramp-up time, possible GPIO pin glitches can cause no adverse effects to their systems..

Silicon Revision Affected:

A1, A2

7 Timers

7.1 General-purpose timer Edge Count mode count error when timer is disabled

Description:

When a general-purpose timer is configured for 16-Bit Input Edge Count Mode, the timer (A or B) erroneously decrements by one when the `Timer Enable (TnEN)` bit in the **GPTM Control (GPTMCTL)** register is cleared (the timer is disabled).

Workaround:

When the general-purpose timer is configured for Edge Count mode and software needs to “stop” the timer, the timer should be reloaded with the current count + 1 and restarted.

Silicon Revision Affected:

A1, A2

7.2 General-purpose timer 16-bit Edge Count mode does not load reload value

Description:

In Edge Count mode, the input events on the CCP pin decrement the counter until the count matches what is in the **GPTM Timern Match (GPTMTnMATCHR)** register. At that point, an interrupt is asserted and then the counter should be reloaded with the original value and counting begins again. However, the reload value is not reloaded into the timer.

Workaround:

Rewrite the **GPTM Timern Interval Load (GPTMTnILR)** register before restarting.

Silicon Revision Affected:

A1, A2

8 ADC

8.1 Use of "Always" triggering for ADC Sample Sequencer 3 does not work

Description:

When using ADC Sample Sequencer 3 (SS3) and configuring the trigger source to "Always" to enable continuous sampling by programming the SS3 Trigger Select field (EM3) in the **ADC Event Multiplexer Select (ADCEMUX)** register to 0xF, the first sample will be captured, but no further samples will be updated to the sequencer FIFO. Interrupts are continuously generated after the first sample and the FIFO status remains empty.

Workaround:

Software must disable and re-enable the sample sequencer to capture another sample.

Silicon Revision Affected:

A1, A2

8.2 Incorrect behavior with timer ADC triggering when another timer is used in 32-bit mode

Description:

When a timer is configured to trigger the ADC and another timer is configured to be a 32-bit periodic or one-shot timer, the ADC is triggered continuously instead of the specified interval.

Workaround:

Do not use a 32-bit periodic or one-shot timer when triggering ADC. If the timer is in 16-bit mode, the ADC trigger works as expected.

Silicon Revision Affected:

A2

9 CAN

9.1 CAN register accesses require software delays

Description:

Because of a synchronization issue between the processor clock and the 8-MHz CAN clock, both read and write accesses to CAN registers require a software delay in order to ensure proper operation. If this delay is not observed between reads or writes, then register data corruption will occur, causing problems that are difficult to debug. Due to the nature of the synchronization issue, write accesses and read accesses have slightly different issues.

When performing CAN register write accesses, a delay is required between successive writes to any CAN register. The amount of delay required is related to the ratio of the processor clock to the CAN clock. For example, if the processor clock is 4 times greater than the CAN clock, then there must be a 4-processor-cycle gap between successive writes to the CAN controller. However, in the case that the processor clock is less than or equal to the CAN clock, then there are no write access limitations.

When performing CAN register read accesses, a delay is required between the reads of the CAN registers. The difference with read accesses is that all read accesses to CAN registers must perform a double read to receive the correct data. The first read initiates the read request to the CAN controller and the second read access retrieves the data. This sequence cannot be interrupted by another read to the same CAN controller or the data read by the second read access will have invalid data. This means that code that reads the CAN registers must protect this read/delay/read sequence from other asynchronous code, such as interrupt handlers, that access the same CAN controller. Like the case for writing CAN registers, the delay between successive reads to CAN registers is related to the ratio of the processor to the CAN clock. For example, if the processor clock is 4 times greater than the CAN clock, then there must be a 4-cycle gap between reads. However, unlike the write case, when the processor clock is less than or equal to the CAN clock, there still must be a 2-processor cycle delay between read accesses in order to retrieve the correct data. Because this erratum will be fixed in future revisions, software should not take advantage of "pipelining" read operations to help improve access time to the CAN registers. This scheme will not work in future versions of the microcontroller and should be avoided.

Debugger accesses to the CAN registers will also show these issues, usually when debuggers perform read accesses to display the register data in a memory window, or in some cases, a register display window. The data displayed in the memory window will not show the correct data for the CAN registers. In most cases, the read accesses are slow and in sequence so they will show the CAN registers in the memory window offset by one word. However, this cannot be guaranteed as the debugger could possibly read the registers too quickly or not in address order and display invalid data.

Workaround:

In order to safely read or write the CAN registers, delays must be inserted for the correct number of cycles. Writes can delay before or after the CAN register write depending on the system needs, while reads must always perform a double-read to get data back from the CAN register. The Stellaris® Peripheral Driver Library (DriverLib) provides the following two functions to perform the delays necessary for reading or writing the CAN registers: CANReadReg and CANWriteReg. The default behavior is tuned for a 50-MHz processor clock via the define (CAN_RW_DELAY) in the can.c file of DriverLib. If the processor clock is lower, this value can be changed and DriverLib can be rebuilt for more optimal performance. Care should be taken when adjusting this value as different compilers may generate the looping code differently. When this errata is fixed, future releases of DriverLib will replace these functions with direct hardware accesses to the registers.

As an example, the amount of delay necessary if the processor clock is 25 MHz and the CAN clock is 8 MHz is 3.125 processor clocks or at least 4 processor clocks. When reading CAN registers, no other CAN accesses can occur. This requires protecting the non-interrupt code from interrupt handlers corrupting the read operations. This precaution is not required for writes, as the default interrupt latency is higher than the delay necessary at 50 MHz.

To write a CAN register, use the following simple sequence:

1. Write the CAN register.
2. Delay for (processor clock/CAN clock) processor cycles.

To read a CAN register, use the following simple sequence:

1. Acquire CAN mutex (mutual exclusion).
2. Read the CAN register and discard the data.
3. Delay for (processor clock/CAN clock) processor cycles.
4. Read the CAN register again to get the correct data.
5. Release CAN mutex.

The mutex used to protect CAN access can be done more than one way. One method is to simply disable interrupts for the CAN controller that is being accessed during read accesses. Whatever method is used, it must be sure to protect against any asynchronous code that accesses the same CAN controller as the code that it interrupts.

Silicon Revision Affected:

A1, A2

10 Ethernet Controller

10.1 Number of Packets decremented early

Description:

The Number of Packets Register (NPR) decrements in the cycle before the Frame Check Sequence (FCS) is read from the Receive FIFO. As a result, software may incorrectly believe the entire packet has been received when it has not.

Workaround:

Use either the DriverLib routine or compare the number of bytes received to the Length field from the Frame to determine when the FIFO is empty.

Silicon Revision Affected:

A1, A2

11 PWM

11.1 PWM pulses cannot be smaller than dead-band time

Description:

The dead-band generator in the PWM module has undesirable effects when receiving input pulses from the PWM generator that are shorter than the dead-band time. For example, providing a 4-clock-wide pulse into the dead-band generator with dead-band times of 20 clocks (for both rising and falling edges) produces a signal on the primary (non-inverted) output that is High except for 40 clocks (the combined rising and falling dead-band times), and the secondary (inverted) output is always Low.

Workaround:

User software must ensure that the input pulse width to the dead-band generator is greater than the dead-band delays.

Silicon Revision Affected:

A1, A2

11.2 PWM interrupt clear misses in some instances

Description:

It is not possible to clear a PWM generator interrupt in the same cycle when another interrupt from the same PWM generator is being asserted. PWM generator interrupts are cleared by writing a 1 to the corresponding bit in the **PWM Interrupt Status and Clear (PWMnISC)** register. If a write to clear the interrupt is missed because another interrupt in that PWM generator is being asserted, the interrupt condition still exists, and the PWM interrupt routine is called again. System problems could result if an interrupt condition was already properly handled the first time, and the software tries to handle it again. Note that even if an interrupt event has not been enabled in the **PWM Interrupt and Trigger Enable (PWMnINTEN)** register, the interrupt is still asserted in the **PWM Raw Interrupt Status (PWMnRIS)** register.

Workaround:

In most instances, performing a double-write to clear the interrupt greatly decreases the chance that the write to clear the interrupt occurs on the same cycle as another interrupt. Because each generator has six possible interrupt events, writing the **PWMnISC** register six times in a row guarantees that the interrupt is cleared. If the period of the PWM is small enough, however, this method may not be practical for the application.

Silicon Revision Affected:

A1, A2

11.3 PWM generation is incorrect with extreme duty cycles

Description:

If a PWM generator is configured for Count-Up/Down mode, and the **PWM Load (PWMnLOAD)** register is set to a value N, setting the compare to a value of 1 or N-1 results in steady state signals instead of a PWM signal. For example, if the user configures PWM0 as follows:

- PWMENABLE = 0x00000001

  - PWM0 Enabled
- PWM0CTL = 0x00000007
  - Debug mode enabled
  - Count-Up/Down mode
  - Generator enabled
- PWM0LOAD = 0x00000063
  - Load is 99 (decimal), so in Count-Up/Down mode the counter counts from zero to 99 and back down to zero (200 clocks per period)
- PWM0GENA = 0x000000b0
  - Output High when the counter matches comparator A while counting up
  - Output Low when the counter matches comparator A while counting down
- PWM0DBCTL = 0x00000000
  - Dead-band generator is disabled

If the **PWM0 Compare A (PWM0CMPA)** value is set to 0x00000062 (N-1), PWM0 should output a 2-clock-cycle long High pulse. Instead, the PWM0 output is a constant High value.

If the **PWM0CMPA** value is set to 0x00000001, PWM0 should output a 2-clock-cycle long negative (Low) pulse. Instead, the PWM0 output is a constant Low value.

Workaround:

User software must ensure that when using the PWM Count-Up/Down mode, the compare values must never be 1 or the **PWMnLOAD** value minus one (N-1).

Silicon Revision Affected:

A1, A2

11.4 PWMINTEN register bit does not function correctly

Description:

In the **PWM Interrupt Enable (PWMINTEN)** register, the `IntPWM0` (bit 0) bit does not function correctly and has no effect on the interrupt status to the ARM Cortex-M3 processor. This bit should not be used.

Workaround:

PWM interrupts to the processor should be controlled with the use of the **PWM0-PWM2 Interrupt and Trigger Enable (PWMnINTEN)** registers.

Silicon Revision Affected:

A1, A2

12 QEI

12.1 QEI index resets position when index is disabled

Description:

When the QEI module is configured to not reset the position on detection of the index signal (that is, the `ResMode` bit in the **QEI Control (QEICTL)** register is 0), the module resets the position when the index pulse occurs. The position counter should only be reset when it reaches the maximum value set in the **QEI Maximum Position (QEIMAXPOS)** register.

Workaround:

Do not rely on software to disable the index pulse. Do not connect the index pulse if it is not needed.

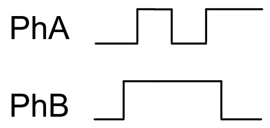
Silicon Revision Affected:

A1, A2

12.2 QEI hardware position can be wrong under certain conditions

Description:

The **QEI Position (QEIPOS)** register can be incorrect if the QEI is configured for quadrature phase mode (`SigMode` bit in **QEICTL** register = 0) and to update the position counter of every edge of both `PhA` and `PhB` (`CapMode` bit in **QEICTL** register = 1). This error can occur if the encoder is stepped in the reverse direction, stepped forward once, and then continues in the reverse direction. The following sequence of transitions on the `PhA` and `PhB` pins causes the error:



Assuming the starting position prior to the above `PhA` and `PhB` sequence is 0, the position after the falling edge on `PhB` should be -3, however the **QEIPOS** register will show the position to be -1.

Workaround:

Configure the QEI to update the position counter on every edge on `PhA` only (`CapMode` bit in **QEICTL** register = 0). The effective resolution is reduced by 50%. If full resolution position detection is required by updating the position counter on every edge of both `PhA` and `PhB`, no workaround is available. Hardware and software must take this into account.

Silicon Revision Affected:

A1, A2

Luminary Micro, Inc.
108 Wild Basin, Suite 350
Austin, TX 78746
Main: +1-512-279-8800
Fax: +1-512-279-8879
http://www.luminarymicro.com