Stellaris® LM3S Sandstorm-, Fury-, and DustDevil-Class Microcontrollers Errata

Silicon Errata

Literature Number: SPMZ860
June 2014

Contents

Stellaris® LM3S Sandstorm-, Fury-, and DustDevil-Class Microcontrollers Errata

1 Introduction

This document describes known exceptions to the functional specifications for the Stellaris Sandstorm-class Rev C2, the Stellaris Fury-class Rev A2 and the Stellaris DustDevil-class Rev A0 microcontrollers. See also the ARM® Cortex™-M3 errata, SPMZ091 (for Sandstorm-class) or SPMZ092 (for Fury- or DustDevil-class).

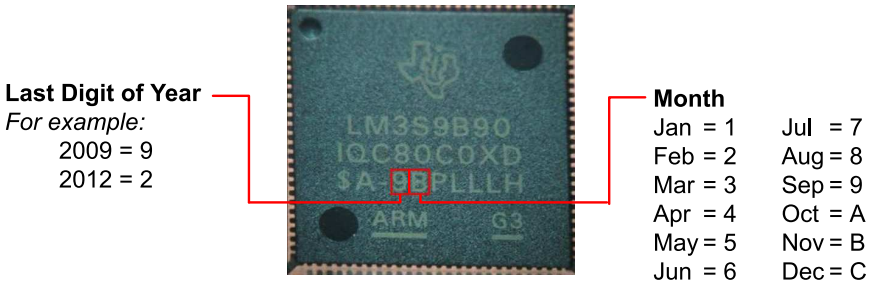
Note that each Stellaris microcontroller may not have all peripherals so not all errata may apply to your device.

For product details on the microcontrollers, see:

- Sandstorm-Class MCUs
- Fury-Class MCUs
- DustDevil-Class MCUs

2 Device Date Code

To determine the date code of your part, look at the third line in the part markings, at the fourth and fifth characters following the dash (outlined in red below). The first number after the dash indicates the last decimal digit of the year. The next character indicates the month, in hexadecimal. So, in the below example, the 9B indicates a date code of November 2009.



3 Advisory to Silicon Revision Correlation

Table 1. Advisory to Silicon Revision Matrix

Advisory Number	Advisory Title	Silicon Revision(s) Affected		
		Sandstorm Rev C2	Fury Rev A2	DustDevil Rev A0
ADC				
LM3ADC#01	Use of "Always" triggering for ADC Sample Sequencer 3 does not work	X	X	X
LM3ADC#02	Incorrect behavior with timer ADC triggering when another timer is used in 32-bit mode	X	X	X
LM3ADC#03	ADC hardware averaging produces erroneous results in differential mode	X	X	X
LM3ADC#04	Retriggering a sample sequencer before it has completed the current sequence results in continuous sampling	X	X	X
LM3ADC#05	Using the PWM unit to trigger the ADC results in only one sample taken			X
LM3ADC#06	TIMER3 cannot trigger ADC			X
LM3ADC#07	For devices with 4 or 6 ADC channels, inputs have the incorrect channel assignments			X
LM3ADC#18	Data may not be present in the FIFO at the time of the sequence interrupt or trigger	X	X	X
LM3ADC#19	The first two ADC samples may be incorrect	X	X	X
Boot Loader				
LM3BOOT#01	ROM-resident boot loader does not operate			X
CAN				
LM3CAN#01	CAN register accesses require software delays		X	
DMA				
LM3DMA#01	No interrupt generated with software-triggered DMA transfer using channel 30			X
LM3DMA#02	The µDMA controller fails to generate capture mode DMA requests from Timer A in the Timer modules			X
Electrical Characteristics				
LM3ELEC#01	Certain pins do not fully comply with the JEDEC ESD standard			X
GPIO				
LM3GPIO#01	GPIO input pin latches in the Low state if pad type is open drain	X	X	
LM3GPIO#02	GPIO pins may glitch during power supply ramp up	X	X	
LM3GPIO#03	GPIO commit register control not working as expected			X
LM3GPIO#04	PB0 and PB1 have permanent internal pull-down resistance			X
General-Purpose Timers				
LM3GPTM#01	General-purpose timer Edge Count mode count error when timer is disabled	X	X	
LM3GPTM#02	General-purpose timer 16-bit Edge Count or Edge Time mode does not load reload value	X	X	X
LM3GPTM#03	The General-Purpose Timer Match register does not function correctly in 32-bit mode	X	X	X
LM3GPTM#04	A spurious DMA request is generated when the timer rolls over in Input-Edge Time mode			X
LM3GPTM#14	Writes to some General-Purpose Timer registers cause the counter to increment and decrement in some cases	X	X	X
Hibernation Module				
LM3HIB#01	Hibernation module does not operate correctly		X	
LM3HIB#02	Hibernation module may have higher current draw than specified in data sheet under certain conditions			X
LM3HIB#03	Hibernation module HIB pin is not an open-drain output			X
LM3HIB#04	Hibernation module registers are cleared by POR if not in Hibernate mode			X

Table 1. Advisory to Silicon Revision Matrix (continued)

Advisory Number	Advisory Title	Silicon Revision(s) Affected		
		Sandstorm Rev C2	Fury Rev A2	DustDevil Rev A0
LM3HIB#05	Hibernation Raw Interrupt Status (HIBRIS) register may not properly indicate Hibernation wake cause			X
LM3HIB#06	In Hibernate mode, the microcontroller may be parasitically powered by external USB signals			X
LM3HIB#07	Writes to certain Hibernation module registers sometimes fail			X
LM3HIB#08	Writes to Hibernation module registers may change the value of the RTC			X
I2C				
LM3I2C#01	I^2C Slave Masked Interrupt Status (I2CSMIS) register bits are incorrect			X
JTAG and Serial Wire Debug				
LM3JTAG#01	JTAG INTEST instruction does not work	X	X	X
LM3JTAG#02	JTAG pins do not have internal pull-ups enabled at power-on reset		X	
LM3JTAG#03	Subsequent attempts to mass erase the Flash memory following a locked device recovery will fail			X
LM3JTAG#04	Boundary scan is not functional			X
Memory				
LM3MEM#01	MERASE bit of the FMC register does not erase the entire Flash array		X	
LM3MEM#02	FMPPE and FMPRE registers cannot be committed to non-volatile storage			X
LM3MEM#03	ROM_SSIConfigSetExpClk function is incorrect			X
PWM				
LM3PWM#01	PWM pulses cannot be smaller than dead-band time	X	X	
LM3PWM#02	PWM interrupt clear misses in some instances	X	X	
LM3PWM#03	PWM generation is incorrect with extreme duty cycles	X	X	X
LM3PWM#04	PWMINTEN register bit does not function correctly	X	X	
LM3PWM#05	Sync of PWM does not trigger "zero" action	X	X	X
LM3PWM#06	PWM "zero" action occurs when the PWM module is disabled	X	X	X
LM3PWM#07	PWM sync status is not properly cleared			X
LM3PWM#08	PWM fault latch does not operate correctly			X
LM3PWM#09	Under certain circumstances, the PWM load interrupt is triggered as soon as the PWM is enabled	X	X	X
LM3PWM#10	Setting the PWMSYNC bits may not synchronize the PWM counters if PWMDIV is used	X	X	X
QEI				
LM3QEI#01	QEI index resets position when index is disabled	X	X	X
LM3QEI#02	QEI hardware position can be wrong under certain conditions	X	X	X
LM3QEI#03	When using the index pulse to reset the counter, a specific initial conditions in the QEI module causes the direction for the first count to be misread	X	X	X
SSI				
LM3SSI#02	SSI Receive FIFO Time-out interrupt may assert sooner than expected in slave	X	X	X
System Control				
LM3SYSCTL#01	MOSC verification circuit does not detect all failures of the main oscillator	X		
LM3SYSCTL#02	Excessive input pin current when level exceeds V_{DD}	X		
LM3SYSCTL#03	LDO Current Limit interrupt does not function properly	X		
LM3SYSCTL#04	LDO Power Unregulated interrupt unpredictable after LDO voltage adjustments	X		
LM3SYSCTL#05	PLL Lock Raw Interrupt Status triggers when PLL is powered down	X		
LM3SYSCTL#06	I/O buffer 5-V tolerance issue	X	X	X

Table 1. Advisory to Silicon Revision Matrix (continued)

Advisory Number	Advisory Title	Silicon Revision(s) Affected		
		Sandstorm Rev C2	Fury Rev A2	DustDevil Rev A0
LM3SYSCTL#07	Standard R-C network cannot be used on $\overline{\text{RST}}$ to extend POR timing	X	X	
LM3SYSCTL#08	Clock source incorrect when waking up from Deep Sleep mode in some configurations		X	
LM3SYSCTL#09	PLL may not function properly at default LDO setting		X	
LM3SYSCTL#10	PLL runs fast when using a 3.6864-MHz crystal		X	
LM3SYSCTL#11	External reset does not reset the XTAL to PLL Translation (PLLCFG) register		X	X
LM3SYSCTL#12	Debugger cannot halt processor when in Sleep Mode			X
LM3SYSCTL#13	MOSC valid detect circuit should not be enabled			X
LM3SYSCTL#24	DSDIVORIDE value of 0x1 does not divide Deep Sleep clock by 2		X	X
LM3SYSCTL#25	Special considerations for power transitions are required to ensure correct device operation	X	X	X
UART				
LM3UART#01	The RTRIS bit in the UARTRIS register is only set when the interrupt is enabled	X	X	X
LM3UART#03	When UART SIR mode is enabled, µDMA burst transfer does not occur			X
USB				
LM3USB#01	Device Capabilities 6 (DC6) register incorrectly specifies USB OTG functionality			X
LM3USB#02	Violating USB VBUS in-rush current specifications may cause USB controller to hang			X
LM3USB#03	PB0 and PB1 pins may not be used for GPIO or peripheral functions if USB Host or Device mode functionality is used			X
LM3USB#04	Transfer may stall when size of µDMA transfer does not match USB FIFO			X
LM3USB#05	USB Host controller may not be used to communicate with a low-speed Device when connected through a hub			X
LM3USB#06	The USB PLL may fail to lock after reset			X
LM3USB#07	USB OTG signaling does not function correctly			X
LM3USB#15	USB controller sends EOP at end of device Remote Wake-Up			X
LM3USB#16	Device sends SE0 in response to a USB bus reset			X
LM3USB#17	USB Resume occasionally does not wake device from Deep Sleep			X

4 Known Design Exceptions to Functional Specifications

Table 2. Advisory List

Table 2. Advisory List (continued)

LM3ADC#01 *Use of "Always" triggering for ADC Sample Sequencer 3 does not work*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

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(s): Software must disable and re-enable the sample sequencer to capture another sample.

LM3ADC#02 *Incorrect behavior with timer ADC triggering when another timer is used in 32-bit mode*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

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(s): 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.

LM3ADC#03 *ADC hardware averaging produces erroneous results in differential mode*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

Description: The implementation of the ADC averaging circuit does not work correctly when the ADC is sampling in differential mode and the difference between the voltages is approximately 0.0 V.

Workaround(s): Do not use hardware averaging in differential mode. Instead, use the FIFO to store results and average them in software.

LM3ADC#04 *Retriggering a sample sequencer before it has completed the current sequence results in continuous sampling*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

Description: Re-triggering a sample sequencer before it has completed its programmed conversion sequence causes the sample sequencer to continuously sample. If interrupts have been enabled, interrupts are generated at the appropriate place in the sample sequence. This problem only occurs when the new trigger is the same type as the current trigger.

Workaround(s): Ensure that a sample sequence has completed before triggering a new sequence using the same type of trigger.

LM3ADC#05 *Using the PWM unit to trigger the ADC results in only one sample taken*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: If the ADC is configured to trigger from a PWM generator source, the ADC unit is triggered on the first matched event in the **PWMn Interrupt and Trigger Enable (PWMnINTEN)** register; however, it will receive no further triggers at the subsequent event matches.

Workaround(s): If the raw interrupt status is cleared before the next event match, the ADC correctly receives the next event signal. Therefore, the trigger condition must be cleared using the same interrupt condition in the **PWMn Interrupt Status and Clear (PWMnISC)** register.

Trigger Condition	Interrupt Status and Clear
TrCntZero	IntCntZero
TrCntLoad	IntCntLoad
TrCmpAU	IntCmpAU
TrCmpAD	IntCmpAD
TrCmpBU	IntCmpBU
TrCmpBD	IntCmpBD

For example, consider the scenario where the ADC performs sequence 0 and generates a processor interrupt when triggered by PWM generator 2 CmpAD (compararator A value count down match). The DriverLib call to enable the PWM to trigger the ADC is:

```
PWMGenIntTrigEnable(PWM_BASE, PWM_GEN_2, PWM_TR_CNT_AD);
```

In the ADC sequence 0 interrupt handler, both the ADC interrupt status bit and the PWM generator 2 status bits must be cleared, as follows:

```
void
ADC0IntHandler(void)
{
    //
    // Clear the PWM trigger source
    // This step is for an errata workaround
    //
    PWMGenIntClear(PWM_BASE, PWM_GEN_2, PWM_INT_CNT_AD);
    //
    // Clear the ADC interrupt status
    //
    ADCIntClear(ADC_BASE, 0);
    ...remaining ADC interrupt handler code...
```

LM3ADC#06 *TIMER3 cannot trigger ADC*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: The **ADC Event Multiplexer Select (ADCEMUX)** register can be used to select the event (trigger) that initiates sampling for each sample sequencer. However, Timer 3 cannot be used to trigger analog-to-digital conversions.

Workaround(s): Use one of the other timers to trigger ADC.

LM3ADC#07 *For devices with 4 or 6 ADC channels, inputs have the incorrect channel assignments*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description:

For devices with 4 channels

Instead of channels 0, 1, 2, and 3 available on pins PE3, PE2, PE1, and PE0, respectively, the available channels are 4, 5, 6 and 7 on pins PD3, PD2, PD1 and PD0, respectively.

For devices with 6 channels

Instead of channels 0, 1, 2, 3, 4 and 5 available on pins PE3, PE2, PE1, PE0, PD3 and PD2, respectively, the available channels are 0, 1, 4, 5, 6, 7 on pins PE3, PE2, PD3, PD2, PD1 and PD0, respectively.

Workaround(s): Software must utilize the alternate channels when programming the ADC (for example, use ADC4 instead of ADC0). When configuring the **GPIOAMSEL** register, software must enable analog inputs for the alternate pins (for example, PD3 instead of PE3).

Fixed: Fixed devices have 8 channels available so that board changes are not required and have date codes of:

0x07 or later for LM3S1625, LM3S1626, LM3S1627, LM3S1776, LM3S2276, LM3S2616, LM3S2671, LM3S2776, LM3S5632, LM3S5732

0x19 or later for LM3S3651, LM3S5652, LM3S5752, LM3S5662, LM3S5762

See Section 2, Device Date Code, for more information on date codes.

LM3ADC#18 *Data may not be present in the FIFO at the time of the sequence interrupt or trigger*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2 and DustDevil-class Rev A0

Description: The interrupt or trigger for a sample sequence may occur before data is placed in the ADC sample sequence FIFO.

Workaround(s): Insert a delay after receiving the interrupt or trigger and before reading the data in the FIFO. The minimum length of the delay is given by the following equation, where H is the number of samples to be averaged if hardware averaging is enabled (H=1 if hardware averaging is not used), and S is the sample rate:

Delay = H / S

For example, if sampling at a rate of 1 MSPS and 4 samples are to be hardware averaged, delay at least 4 µs before reading the data in the FIFO. The StellarisWare API SysCtlDelay() can be used to add a delay based on your system clock frequency.

LM3ADC#19 *The first two ADC samples may be incorrect*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2 and DustDevil-class Rev A0

Description The first two ADC samples taken after the ADC clock is enabled in the **xCGC0** register may be incorrect.

Workaround(s) Reset the ADC peripheral using the **SRCR0** register after the ADC peripheral clock is enabled and before initializing the ADC and enabling the sample sequencer.

LM3BOOT#01 *ROM-resident boot loader does not operate*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: The ROM-resident boot loader cannot be used to program the on-chip Flash. The boot loader enters the hard-fault state as a result of trying to access a peripheral that does not exist.

Workaround(s): Use the Stellaris Flash-resident boot loader to update the on-chip Flash.

LM3CAN#01 *CAN register accesses require software delays*

Device(s) Affected: Stellaris Fury-class Rev A2

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.

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(s): 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 erratum 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.

LM3DMA#01 *No interrupt generated with software-triggered DMA transfer using channel 30*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: When performing a software-triggered data transfer using µDMA channel 30, no interrupt is triggered when the transfer completes.

Workaround(s): Use any of the other channels (except for channels 30 and 31) that are not already assigned to a peripheral. See the "DMA Channel Assignments" table in the Micro Direct Memory Access (µDMA) chapter in the data sheet.

This issue is fixed on parts with a date code of 06 (June 2010) or later. See Section 2, Device Date Code, for more information.

LM3DMA#02 *The µDMA controller fails to generate capture mode DMA requests from Timer A in the Timer modules*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: The µDMA controller fails to generate DMA requests from Timer A in the General-Purpose Timer modules when in the Event Count and Event Time modes.

Workaround(s): Use Timer B.

LM3ELEC#01 *Certain pins do not fully comply with the JEDEC ESD standard*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: The pins listed below do not fully meet the industry ESD standard of 2.0 KV Human Body Model (HBM) under all conditions. All Stellaris devices are tested using JEDEC Standard JESD22-A114. These pins fail only in the pin-to-power condition using a positive voltage transient. The pins pass at 500 V. All other HBM conditions pass on these pins. All other device pins fully comply with this HBM JEDEC standard.

- PB0/USB0ID
- PB1/USB0VBUS
- USB0DM
- USB0DP
- USB0RBIAS
- PF0
- OSC0
- OSC1

Workaround(s): Extra caution should be taken to ensure proper ESD handling of these devices. Use appropriate caution when implementing ESD protection circuits on signals routed to these pins.

LM3GPIO#01 *GPIO input pin latches in the Low state if pad type is open drain*

Device(s) Affected: Stellaris Sandstorm-class Rev C2 and Fury-class Rev A2

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(s): 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.

LM3GPIO#02 *GPIO pins may glitch during power supply ramp up*

Device(s) Affected: Stellaris Sandstorm-class Rev C2 and Fury-class Rev A2

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(s): Ensuring that the V_{DD} power supply ramp up is as 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.

LM3GPIO#03 *GPIO commit register control not working as expected*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: The commit control registers provide a layer of protection against accidental programming of critical hardware peripherals, especially the JTAG/SWD pin functionality. Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL), GPIO Digital Enable (GPIODEN)**, and **GPIO Pull-up Select (GPIOPUR)** registers are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register has been unlocked and the appropriate bits of the **GPIO Interrupt Clear (GPIOCR)** register have been set. Registers **GPIODEN** and **GPIOPUR** should be protected by this mechanism but are not.

Workaround(s): Extra caution should be taken by software when modifying the **GPIODEN** and **GPIOPUR** registers as they are not protected by the **GPIOCR** commit register mechanism. Writes to the register bits that affect the JTAG/SWD pins (PC3:PC0) should be done with great care as this interface may become permanently disabled if done incorrectly. The NMI pin (PB7) is not protected either.

LM3GPIO#04 *PB0 and PB1 have permanent internal pull-down resistance*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: Regardless of their configuration, PB0 and PB1 have an internal pull-down resistance. The internal structure of these pins is shown in Figure 1.

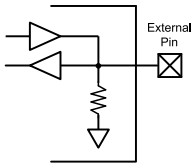


Figure 1. Internal Structure of PB0 and PB1

The characteristic of the pull-down for PB0 is shown in Figure 2. The data labels for each data point show the external pull-up resistance in Ohms.

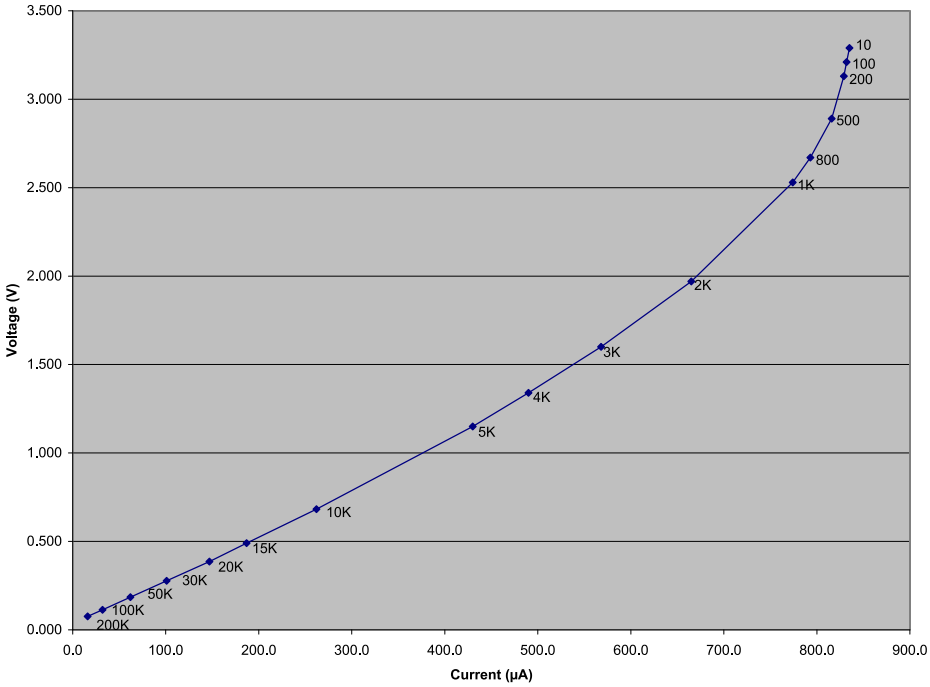


Figure 2. Voltage versus Current for Various External Pull-up Resistors on PB0

The characteristic of the pull-down for PB1 is shown in Figure 3. The data labels for each data point show the equivalent resistance in Ohms.

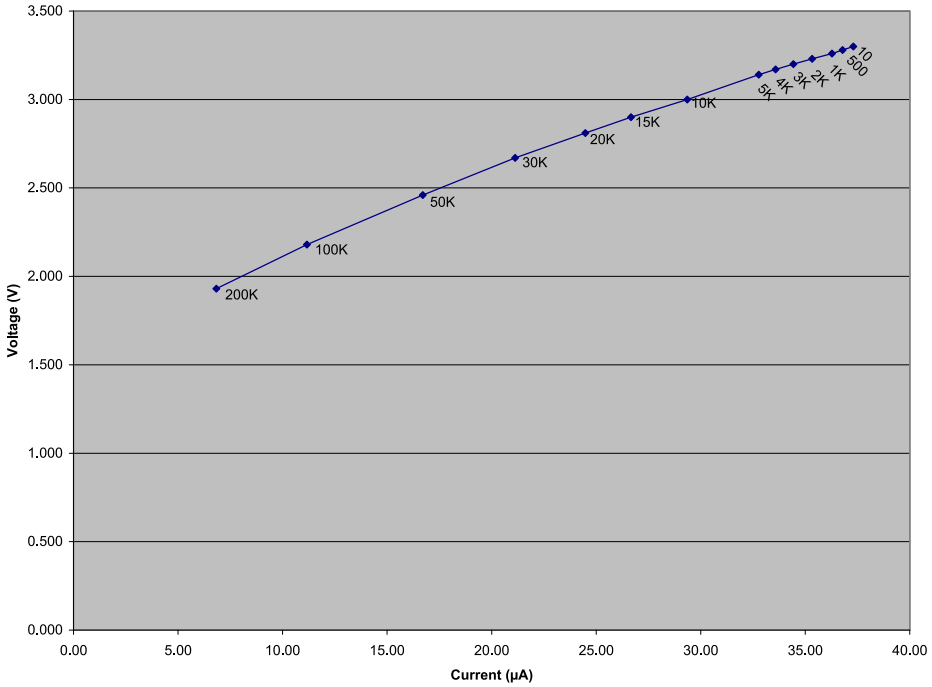


Figure 3. Voltage versus Current for Various External Pull-up Resistors on PB1

Workaround(s): When either of these pins is configured as an input, the external circuit must provide enough drive strength to over-drive the internal pull-down and achieve the necessary V_{IH} voltage level. If an external pull-up resistor is used, it must have a low value such as ~1 KΩ or less for PB0 and ~50 KΩ or less for PB1.

When either of these pins is configured as an output, the drive current needed to over-drive the internal pull-down resistance must be subtracted from the drive capabilities of the pin. In some applications, it may be necessary to select a higher drive strength (such as 4 mA instead of 2 mA) to achieve an acceptable output voltage on PB0.

LM3GPTM#01 *General-purpose timer Edge Count mode count error when timer is disabled*

Device(s) Affected: Stellaris Sandstorm-class Rev C2 and Fury-class Rev A2

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(s) 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.

LM3GPTM#02 *General-purpose timer 16-bit Edge Count or Edge Time mode does not load reload value*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2 and DustDevil-class Rev A0

Description In Edge Count or Edge Time 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(s) Rewrite the **GPTM Timern Interval Load (GPTMTnILR)** register before restarting.

LM3GPTM#03 *The General-Purpose Timer Match register does not function correctly in 32-bit mode*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2 and DustDevil-class Rev A0

Description: The **GPTM Timer A Match (GPTMTAMATCHR)** register triggers a match interrupt and a DMA request, if enabled, when the lower 16 bits match, regardless of the value of the upper 16 bits.

Workaround(s): None.

LM3GPTM#04 *A spurious DMA request is generated when the timer rolls over in Input-Edge Time mode*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: When the timer is in Input-Edge Time mode and rolls over after the terminal count, a spurious DMA request is generated.

Workaround(s): Either ignore the spurious interrupt, or disable DMA requests shortly before the terminal count and enable them again shortly after the terminal count.

LM3GPTM#14 *Writes to some General-Purpose Timer registers cause the counter to increment and decrement in some cases*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2 and DustDevil-class Rev A0

Description: Writes to the following registers when the timer is enabled cause the counter to increment in up count mode and decrement in down count mode when incrementing or decrementing the counter inside the General-Purpose timers:

- **GPTM Timer n Match (GPTMTnMATCHR)**
- **GPTM Timer n Prescale (GPTMTnPR)**

Situations in which the counter is incremented or decremented include:

- RTC Mode
- Input edge count mode

Workaround(s): None.

LM3HIB#01 *Hibernation module does not operate correctly*

Device(s) Affected: Stellaris Fury-class Rev A2

Description: The Hibernation module on this microcontroller does not operate correctly.

Workaround(s): None.

LM3HIB#02 *Hibernation module may have higher current draw than specified in data sheet under certain conditions*

Device(s) Affected: Stellaris DustDevil-class Rev A0

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(s): 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.

LM3HIB#03 *Hibernation module HIB pin is not an open-drain output*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: Because the HIB pin is not an open-drain output, the pin is capable of sourcing current to the power supply circuit that it controls. Under certain conditions, this configuration may result in the Hibernation module drawing more current than expected.

If the voltage on the circuit connected to HIB drops below V_{BAT} before HIB is asserted, current may be sourced out of the HIB pin, increasing Hibernation module current to approximately 300 µA instead of 16 µA. This condition also occurs if the Hibernation module attempts to wake but the power supply circuit remains below V_{BAT}.

Workaround(s): Add a diode between the HIB pin and the power supply to prevent current being sourced by the Hibernation module. The cathode of the diode should be connected to the HIB signal. A diode with a low reverse current, such as the Panasonic MA2J728, is recommended.

LM3HIB#04 *Hibernation module registers are cleared by POR if not in Hibernate mode*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: The Hibernation module registers are reset by a POR event if the module is not in Hibernate mode. This condition disables the module and stops the 32-kHz clock.

Workaround(s): To preserve Hibernation module settings, always enter Hibernate mode before removing power. Note that the battery-backed memory is not cleared and is valid as long as V_{BAT} voltage remains within specification.

LM3HIB#05 *Hibernation Raw Interrupt Status (HIBRIS) register may not properly indicate Hibernation wake cause*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: The **Hibernation Raw Interrupt Status (HIBRIS)** register may not properly indicate the event that caused the wake-up from hibernation. After the first wake-up event, the external wake (EXTW) status bit will get set incorrectly each time a wake-up event is triggered, regardless of the wake-up source.

Workaround(s): If the EXTW bit is the only bit set, the cause is an external wake-up. If the EXTW bit and another bit is set, the wake-up source is indicated by the other bit.

LM3HIB#06 *In Hibernate mode, the microcontroller may be parasitically powered by external USB signals*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: If a USB cable is still connected when the device is in Hibernation mode, the part may be parasitically powered by external voltages on USB signals. These signals include the USB bidirectional differential data pins USB0DP and USB0DM, the USB OTG signals USB0VBUS and USB0ID, the USB VBUS supply if connected to PB1, and the pull-up on PB0 if the part is used as a USB device.

Workaround(s): If Hibernate mode is required, it may be necessary to insert a switching circuit in the USB signals to provide isolation during power down. The switching circuit should be controlled by the HIB pin.

LM3HIB#07 *Writes to certain Hibernation module registers sometimes fail*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: Due to a synchronization issue with the independent clock domain of the Hibernation module, writes to certain registers may sometimes fail, even though the WRC bit in the **HIBCTL** register is set after the write occurs. Registers affected include **HIBRTCC, HIBRTCM0, HIBRTCM1, HIBRTCLD, HIBRTCT**, and **HIBDATA**.

Workaround(s): After performing a write to any Hibernation module register or battery-backed memory, read the contents back and verify that they are correct. If they are incorrect, perform the write operation again.

LM3HIB#08 *Writes to Hibernation module registers may change the value of the RTC*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: If the Hibernation module's RTC counter is active, any write to certain Hibernation module registers that occurs while the RTC counter is changing from the current value to the next can cause corruption of the RTC counter stored in the **HIBRTCC** register. Registers affected are: **HIBRTCC, HIBRTCM0, HIBRTCM1, HIBRTCLD, HIBRTCT**, and **HIBDATA**.

Workaround(s): The user application must guarantee that writes to the affected Hibernation module registers cannot occur on the RTC counter boundary. Any initial configuration of the affected Hibernation module registers must be done before enabling the RTC counter.

There are two ways to update affected Hibernation Module registers after initial configuration:

1. Use the Hibernation RTC match interrupt to perform writes to the affected Hibernation module registers. Assuming the interrupt is guaranteed to be serviced within 1 second, this technique provides a mechanism for the application to know that the RTC update event has occurred and that it is safe to write data to the affected Hibernation module registers. This method is useful for applications that don't require many writes to Hibernation module registers.
2. Set up a secondary time-keeping resource to indicate when it is safe to perform writes to the affected Hibernation module registers. For example, use a general-purpose timer in combination with the Hibernation RTC match interrupt. In this scenario, the RTC match interrupt is used to both update the match register value and enable the general-purpose timer in one-shot mode. The timer must be configured to have a maximum time-out period of less than 1 second. In this configuration, a global variable is used to indicate that it is safe to perform writes to the affected Hibernation module registers. When the one-shot timer times out, the timer interrupt updates the global variable to indicate that writes are no longer safe. This procedure is repeated on every RTC match interrupt.

LM3I2C#01 *I^2C Slave Masked Interrupt Status (I2CSMIS) register bits are incorrect*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: The bits within the **I^2C Slave Masked Interrupt Status (I2CSMIS)** register do not operate properly. This register should not be used by software.

Workaround(s): Software should use the **I^2C Slave Raw Interrupt Status (I2CSRIS)** register instead.

LM3JTAG#01 *JTAG INTEST instruction does not work*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

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

Workaround(s): None.

LM3JTAG#02 *JTAG pins do not have internal pull-ups enabled at power-on reset*

Device(s) Affected: Stellaris Fury-class Rev A2

Description: Following a power-on reset, the JTAG pins $\overline{\text{TRST}}$, 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(s): 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 $\overline{\text{TRST}}$. 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).

LM3JTAG#03 *Subsequent attempts to mass erase the Flash memory following a locked device recovery will fail*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: If a user performs a locked device recovery as described in the data sheet, subsequent attempts to mass erase the Flash memory will fail. The failure status in the **Flash Controller Raw Interrupt Status (FCRIS)** register following the attempted mass erase operation has the Access Raw Interrupt Status (ARIS) bit set, indicating an access error. The access error is generated because the recovery sequence incorrectly restores **Flash Memory Protection Enable 2 (FMPPE2)** and **FMPPE3** to the value 0xFFFF.FFFF. During subsequent mass erase attempts, the logic (that ensures write-protected Flash memory is not erased) incorrectly evaluates the protected condition and inhibits the erase operation.

NOTE: This erratum may affect the ROM-resident boot loader because this module utilizes Flash memory mass erase. The boot loader will not function following a recovery sequence because the boot loader attempts a mass erase, detects the error code, and terminates.

Workaround(s): There is a dynamic workaround in which the **FMPPEn** registers are written to properly set the **FMPPEn** state before a mass erase is attempted. The **FMPPEn** registers must be written to the values indicated in the following table based on the size of Flash memory available on the microcontroller. The **FMPPEn** register must be written before each mass erase operation that is attempted.

Table 3. FMPPEn Default Values Based on Flash Memory Size

Flash Size (KB)	FMPPE3	FMPPE2	FMPPE1	FMPPE0
128	0x0000.0000	0x0000.0000	0xFFFF.FFFF	0xFFFF.FFFF
96	0x0000.0000	0x0000.0000	0x0000.FFFF	0xFFFF.FFFF
64	0x0000.0000	0x0000.0000	0x0000.0000	0xFFFF.FFFF
32	0x0000.0000	0x0000.0000	0x0000.0000	0x0000.FFFF

LM3JTAG#04 *Boundary scan is not functional*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: The boundary scan is not functional on this device.

Workaround(s): None.

Fixed on devices with date codes of 0x1A (October 2011) or later. See Section 2, Device Date Code, for more information.

LM3MEM#01 *MERASE bit of the FMC register does not erase the entire Flash array*

Device(s) Affected: Stellaris Fury-class Rev A2

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(s): 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.

LM3MEM#02 *FMPPE and FMPRE registers cannot be committed to non-volatile storage*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: The Flash memory protection provided by the **Flash Memory Protection Read Enable n (FMPREn)** and **Flash Memory Protection Program Enable n (FMPPEn)** registers does not function correctly. Do not commit any of these registers or mass erase will not function.

This does not affect the Flash memory protection provided by the disabling of the JTAG/SWD port.

Workaround(s): None.

LM3MEM#03 *ROM_SSIConfigSetExpClk function is incorrect*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: If a non-Motorola format was specified in a call to the ROM_SSIConfigSetExpClk function, two lower bits of a clock divisor register could be corrupted. This corruption results in a small error in the actual clock rate.

Workaround(s): Use the StellarisWare SSIConfigSetExpClk function in Flash memory.

LM3PWM#01 *PWM pulses cannot be smaller than dead-band time*

Device(s) Affected: Stellaris Sandstorm-class Rev C2 and Fury-class Rev A2

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

LM3PWM#02 *PWM interrupt clear misses in some instances*

Device(s) Affected: Stellaris Sandstorm-class Rev C2 and Fury-class Rev A2

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 P**WM Raw Interrupt Status (PWMnRIS)** register.

Workaround(s): 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.

LM3PWM#03 *PWM generation is incorrect with extreme duty cycles*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

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(s): 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).

LM3PWM#04 *PWMINTEN register bit does not function correctly*

Device(s) Affected: Stellaris Sandstorm-class Rev C2 and Fury-class Rev A2

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

LM3PWM#05 *Sync of PWM does not trigger "zero" action*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

Description: If the **PWM Generator Control (PWM0GENA)** register has the ActZero field set to 0x2, then the output is set to 0 when the counter reaches 0, as expected. However, if the counter is cleared by setting the appropriate bit in the **PWM Time Base Sync (PWMSYNC)** register, then the "zero" action is not triggered, and the output is not set to 0.

Workaround(s): None.

LM3PWM#06 *PWM "zero" action occurs when the PWM module is disabled*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

Description: The zero pulse may be asserted when the PWM module is disabled.

Workaround(s): None.

LM3PWM#07 *PWM sync status is not properly cleared*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: When writing to the **PWM Time Base Sync (PWMSYNC)** register to sync the PWM generators, the sync bits are not automatically cleared by hardware. This condition results in the PWM counters being held and prevents them from outputting a PWM signal. The issue occurs only when the PWM clock divider is enabled in the **Run-Mode Clock Configuration (RCC)** register. When the USEPWMDIV bit is set, any divider (PWMDIV) exhibits the same behavior.

Workaround(s): A software write to the **PWMSYNC** register to manually clear the sync bits releases the PWM counters and they will generate expected output waveforms.

LM3PWM#08 *PWM fault latch does not operate correctly*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: If the LATCH bit is set in the **PWMnCTL** register, the PWM fault condition should be latched until the INTFAULTn bit in the **PWMISC** register is cleared. However, the PWM fault signal is not correctly latched and the PWM resumes programmed signaling after the fault condition is removed, regardless of whether the INTFAULTn bit is cleared.

Workaround(s): Software can effectively address this issue with the addition of a few register writes in the ISR.

1. The PWMnMINFLTPER register can be used to ensure that the fault is asserted for a long enough period such that the ISR can be called to implement the workaround.
2. The PWM output can be disabled manually using the PWMnEN bit in the **PWMENABLE** register.
3. Software can perform computations to determine if the PWM can be restarted.
4. The INTFAULTn bit in the **PWMISC** is cleared by writing a 1 to it.
5. The PWM output can be manually re-enabled using the PWMnEN bit in the **PWMENABLE** register.

Note that when using this workaround, the PWM output is disabled manually, which means it does not go to the "pre-programmed" state from various fault registers but instead goes to 0.

LM3PWM#09 *Under certain circumstances, the PWM load interrupt is triggered as soon as the PWM is enabled*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

Description: A spurious PWM interrupt occurs immediately when the PWM is enabled under the following conditions:

- The PWM Load register contains a nonzero value and
- Either of the PWM Compare registers contains a value less than the value in the PWM Load register and
- PWM interrupts are enabled.

Workaround(s): None.

LM3PWM#10 *Setting the PWMSYNC bits may not synchronize the PWM counters if PWMDIV is used*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

Description: The bits in the **PWM Time Base Sync (PWMSYNC)** register are used to synchronize the counters in the PWM generators. The PWMDIV field in the **PWM Clock Configuration (PWMCC)** register is used to specify a fractional version of the system clock to use for the counters. If the PWMSYNC bits are set when the PWMDIV field is configured to anything other than 0x0, the counters may not be synchronized.

Workaround(s): None.

LM3QEI#01 *QEI index resets position when index is disabled*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

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

LM3QEI#02 *QEI hardware position can be wrong under certain conditions*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

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:

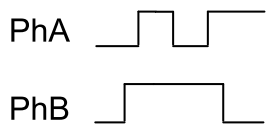


Figure 4. PhA and PhB Transitions

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(s): 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.

LM3QEI#03 *When using the index pulse to reset the counter, a specific initial conditions in the QEI module causes the direction for the first count to be misread*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

Description: When using the index pulse to reset the counter with the following configuration in the **QEI Control (QEICTL)** register:

- SIGMODE is 0 indicating quadrature mode
- CAPMODE is 1 indicating both PhA and PhB edges are counted

and the following initial conditions:

- Both PhA and PhB are 0
- The next quadrature state is in the counterclockwise direction

the QEI interprets the state change as an update in the clockwise direction, which results in a position mismatch of 2.

Workaround(s): None.

LM3SSI#02 *SSI Receive FIFO Time-out interrupt may assert sooner than expected in slave*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

Description: The SSI receive FIFO time-out interrupt may assert sooner than 32 system clock periods in slave mode if the CPSDVSR field in the **SSI Clock Prescale (SSICPSR)** register is set to a value greater than 0x2. Master mode is not affected by this behavior.

Workaround(s): In some cases, software can use the SCR field in the **SSI Control 0 (SSICR0)** register in combination with a CPSDVSR field value of 0x2 to attain the same SSI clock frequency. For example, if the desired serial clock rate is SysClk/48, then CPSDVSR = 0x2 and SCR = 0x17 can be used instead of CPSDVSR = 0x18 and SCR = 0x1 to achieve the same clock rate, using the equation SSInCLK = SysClk / (CPSDVSR * (1 + SCR)). If there is not a value of SCR that can be used with CPSDVSR = 0x2 to attain the required serial clock rate, then the receive FIFO time-out feature cannot be used.

LM3SYSCTL#01 *MOSC verification circuit does not detect all failures of the main oscillator*

Device(s) Affected: Stellaris Sandstorm-class Rev C2

Description: The MOSC verification circuit does not detect all MOSC (main oscillator) failures. In the case where the MOSC clock verification timer function has been enabled by the MOSCVER bit in the **Run-Mode Clock Configuration (RCC)** register, and a MOSC failure is detected, the main clock tree is supposed to immediately switch to a working clock and generate an interrupt to the core. The MOSC verification circuit does not always detect the failure, so the device continues to be clocked by the failed MOSC clock source.

Workaround(s): An external clock verification circuit must be used to guarantee the detection of a main oscillator failure. For example, a general-purpose timer can be used to generate a periodic signal to an external circuit that monitors the oscillation of that signal. If the external circuit detects that the signal has stopped oscillating, the circuit can assert the hardware reset pin of the controller.

LM3SYSCTL#02 *Excessive input pin current when level exceeds V_{DD}*

Device(s) Affected: Stellaris Sandstorm-class Rev C2

Description: When the voltage on an input pin is driven above V_{DD}, excessive current is sunk through the input pin. For example, if a pin is configured as a GPIO input and pulled up to 5 V with a 1K W resistor, the V_{IH} level is only 4.4 V, meaning I_{IH} is 0.64 mA. This violates the 5-V tolerance specification.

Workaround(s): If the device driving the input pin above V_{DD} cannot source enough current to drive the signal to a logic High value, a resistor can be used in series to limit the amount of current being sunk through the pin.

LM3SYSCTL#03 *LDO Current Limit interrupt does not function properly*

Device(s) Affected: Stellaris Sandstorm-class Rev C2

Description: The System Control module can be programmed by software to generate an interrupt when the LDO exceeds its current limit. This feature does not function properly in all over-current conditions.

Workaround(s): The Current Limit interrupt should not be used. The over-current condition should be ignored by the interrupt controller by always masking the Current Limit interrupt. The CLIM bit in the **Interrupt Mask Control (IMC)** register should always be cleared (0).

LM3SYSCTL#04 *LDO Power Unregulated interrupt unpredictable after LDO voltage adjustments*

Device(s) Affected: Stellaris Sandstorm-class Rev C2

Description: The System Control module can be programmed by software to generate an interrupt when the LDO power is unregulated. The interrupt is unpredictable while the LDO voltage is being adjusted. This could incorrectly cause an interrupt or, if the LDOARST bit in the **Allow Unregulated LDO to Reset the Part (LDOARST)** register is set, a system reset.

Workaround(s): The Power Unregulated interrupt should not be used while the LDO voltage is adjusted. While the LDO voltage is adjusted, the unregulated power condition should be ignored by the interrupt controller by always masking the Power Unregulated interrupt. This condition is masked by clearing the LDOIM bit in the **Interrupt Mask Control (IMC)** register.

LM3SYSCTL#05 *PLL Lock Raw Interrupt Status triggers when PLL is powered down*

Device(s) Affected: Stellaris Sandstorm-class Rev C2

Description: The **PLL Lock Raw Interrupt Status (PLLLRIS)** register incorrectly triggers when the PLL is powered down. After reset, by default, the PWRDN bit in the **Run-Mode Clock Configuration (RCC)** register is set, powering down the PLL. This interrupt will not be promoted to the interrupt controller unless the PLL Lock Interrupt Mask (PLLLIM) bit is set in the **Interrupt Mask Control (IMC)** register.

Workaround(s): If the PLL is not being used, mask the PLL Lock interrupt by clearing the PLLIM bit in the **IMC** register. Only unmask the PLLRIS bit after the PLL has been powered on (PWRDN = 0).

LM3SYSCTL#06 *I/O buffer 5-V tolerance issue*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

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

LM3SYSCTL#07 *Standard R-C network cannot be used on $\overline{RST}$ to extend POR timing*

Device(s) Affected: Stellaris Sandstorm-class Rev C2 and Fury-class Rev A2

Description: The standard R-C network on $\overline{RST}$ does not work to extend POR timing beyond the 10 ms on-chip POR. Instead of following the standard capacitor charging curve, $\overline{RST}$ jumps straight to 3 V at power-on. The capacitor is fully charged by current out of the $\overline{RST}$ pin and does not extend or filter the power-on condition. As a result, the reset input is not extended beyond the POR.

Workaround(s): Add a diode to block the output current from $\overline{RST}$. This helps to extend the $\overline{RST}$ pulse, but also means that the R-C is not as effective as a noise filter.

LM3SYSCTL#08 *Clock source incorrect when waking up from Deep Sleep mode in some configurations*

Device(s) Affected: Stellaris Fury-class Rev A2

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(s): 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.

LM3SYSCTL#09 *PLL may not function properly at default LDO setting*

Device(s) Affected: Stellaris Fury-class Rev A2

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 V or below (minimum of 2.25 V). Designs that do not use the PLL module are not affected.

Workaround(s): 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.

LM3SYSCTL#10 *PLL runs fast when using a 3.6864-MHz crystal*

Device(s) Affected: Stellaris Fury-class Rev A2

Description: If the PLL is enabled, and a 3.6864-MHz crystal is used, the PLL runs 4% fast.

Workaround(s): Use a different crystal whose frequency is one of the other allowed crystal frequencies (see the values shown for the XTAL bit in the **RCC** register).

LM3SYSCTL#11 *External reset does not reset the XTAL to PLL Translation (PLLCFG) register*

Device(s) Affected: Stellaris Fury-class Rev A2 and DustDevil-class Rev A0

Description: Performing an external reset (anything but power-on reset) reconfigures the XTAL field in the **Run-Mode Clock Configuration (RCC)** register to the 6 MHz setting, but does not reset the **XTAL to PLL Translation (PLLCFG)** register to the 6 MHz setting.

Consider the following sequence:

1. Performing a power-on reset results in XTAL = 6 MHz and **PLLCFG** = 6 MHz
2. Writing an 8 MHz value to the XTAL field results in XTAL = 8 MHz and **PLLCFG** = 8 MHz
3. Asserting $\overline{\text{RST}}$ results in XTAL = 6 MHz and **PLLCFG** = 8 MHz

In the last step, **PLLCFG** was not reset to its 6-MHz setting. If this step is followed by enabling the PLL to run from an attached 6-MHz crystal, the PLL then operates at 300 MHz instead of 400 MHz. Subsequently configuring the XTAL field with the 8 MHz setting does not change the setting of **PLLCFG**.

Workaround(s): Set XTAL in **PLLCFG** to an incorrect value, and then to the desired value. The second change updates the register correctly. Do not enable the PLL until after the second change.

LM3SYSCTL#12 *Debugger cannot halt processor when in Sleep Mode*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: When the processor is in Sleep mode, the debugger is unable to bring the processor out of Sleep mode in order to initiate a debug session.

Workaround(s): Do not use Sleep mode when attempting to debug the application.

LM3SYSCTL#13 *MOSC valid detect circuit should not be enabled*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: If bit 0 of the **Main Oscillator Control (MOSCCTL)** register is set, the microcontroller is immediately reset and control is transferred to the NMI handler, even if the MOSC is functioning correctly.

Workaround(s): None.

LM3SYSCTL#24 *DSDIVORIDE value of 0x1 does not divide Deep Sleep clock by 2*

Device(s) Affected: Stellaris Fury-class Rev A2 and DustDevil-class Rev A0

Description: A value of 0x1 for the DSDIVORIDE bit field in the **Deep Sleep Clock Configuration (DSLPCLKCFG)** register does not provide divide by two capability for the Deep Sleep clock. The Run-mode clock divider will be used instead. All other DSDIVORIDE values work as expected when entering Deep Sleep.

Workaround(s): Software must program the SYSDIV bit field of the **Run-Mode Clock Configuration (RCC)** register to the desired divider before entering Deep Sleep if Deep Sleep clock divide by 2 was intended for use. Note that when configuring the SYSDIV bit field, this will affect the Run-mode clock divider. Do not configure the clock divider such that the system clock speed is faster than the maximum clock frequency of 80 MHz before entering Deep Sleep.

LM3SYSCTL#25 *Special considerations for power transitions are required to ensure correct device operation*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

Description: The integrated On-chip Low Drop-Out (LDO) voltage regulator, the Power-On Reset (POR) circuitry, and the Brown-Out Reset (BOR) circuitry do not always operate as expected and may result in a risk of incorrect operation during VDD power transitions. Therefore, special system design guidelines must be followed to ensure proper device operation.

The LDO, which supplies power to the microcontroller's internal logic, may fail to initialize properly during the power-up sequence. If the LDO does not initialize correctly, the LDO does not supply enough power to the microcontroller's internal logic, user code is not executed, and the GPIOs are in an indeterminate state. This failure can potentially occur when powering on the device from 0 V, though it most commonly occurs when VDD powers on from a voltage just above 200 mV.

The internal Power-On Reset circuit should hold the internal logic in reset until the microcontroller has reached the minimum operating voltage. In certain cases, the duration of the reset is not enough to protect the device. The POR supervisor can enable internal state machine operation as soon as 6 ms after the VDD supply reaches the minimum POR threshold of 1.75 V. If VDD is still below the minimum operating voltage of 3.0 V after 6 ms, the power-up state machine may not function correctly. The $\overline{\text{RST}}$ pin of the device has no effect on the initialization state machine; therefore a complete power cycle is required to restore the initialization state machine.

The internal Brown-Out Reset circuit is designed to guard against improper operation of logic when VDD is below the minimum operating voltage. However, the brown-out supervisor has a threshold as low as 2.68 V, which is less than the minimum operating voltage on VDD, and it can take several microseconds to respond to a brown-out event. This delay creates a gap in the protection of the device. BOR gaps can be encountered after power up, during steady state operation, if the VDD rail has glitches, and also during power-down.

If the internal logic is active during a POR or BOR gap, unexpected operation may occur that can include brief execution of random sections of user code including ROM functions and random instructions, as well as incorrect power-up initialization. The uncontrolled brief execution of random instructions may result in the undesired erasing or writing of non-volatile memories and GPIO state changes. There is also the possibility that the device may be left in a state where it does not operate correctly until a clean power cycle has been completed.

Workaround(s):

Any processor operation at a VDD level below 3.0 V may result in unexpected code execution and the effects described above. The processor must be halted or the $\overline{\text{RST}}$ signal must be driven Low prior to VDD dropping below 3.0 V and stay in that state until VDD is above 3.0 V.

If VDD falls below 2.4 V, it must continue to fall until it reaches less than 0.2 V. Additionally, the VDD power-up time between 1.75 V and 3.0 V must be less than or equal to 6 ms. If VDD falls below 3.0 V but stays above 2.4 V, it is not necessary for the voltage to continue falling below 2.4 V. VDD can come back up to 3.0 V without any additional timing requirements.

The system designer must ensure that the requirements listed below for power-up, steady state, and power-down are met:

1. The VDD power-up, steady state, and power-down waveform meets the timing requirements shown in Figure 5.
2. The power-up transition of VDD between 1.75 V and 3.0 V must not have any points where it decreases in voltage (must be monotonic).
3. The power-down transition of VDD between 3.0 V and 0.2 V must not have any points where it increases in voltage (must be monotonic).
4. Once steady-state operation between 3.0 V and 3.6 V is achieved, $\overline{\text{RST}}$ must go Low or the CPU execution must be halted prior to VDD falling below 3.0 V.
5. The **Brown-Out Reset Control (PBORCTL)** register must be set so that a brown-out event causes a reset.

Depending on the system environment requirement, items 4 and 5 in the above list may be met by using a voltage supervisor, such as the TLV803M, to monitor a higher voltage rail from which the VDD supply is regulated. Figure 6 shows this implementation with a voltage supervisor monitoring the 5-V rail and a voltage trip point of 4.38 V. A voltage supervisor with a lower voltage trip point can be used to monitor the VDD (3.3-V) rail, however this supervisor must assert reset before VDD reaches 3.0 V. Regardless of the implemented voltage supervisor circuit, the system designer must ensure that there is enough time to assert $\overline{\text{RST}}$ Low prior to VDD falling below 3.0 V. Figure 7 shows the resulting waveform of the circuit shown in Figure 6.

In addition to the power-up, steady-state, and power-down requirements, two diodes in series must be connected between VDD (anode) and the LDO output (cathode) on the microcontroller, as shown in Figure 8. The diodes should have a maximum forward current of greater than 100 mA and a combined forward voltage drop between 1.2 V and 1.8 V.

The reset characteristics parameters that are currently specified in the data sheet are given in Table 4 for Sandstorm and in Table 5 for Fury and DustDevil.

Due to the erratum, the following adjustments to the parameters in Table 4 and Table 5 should be observed in system design:

- R1: Add min of 1.75 V and max of 2.25 V.
- R2: Change min to 2.68 V and max to 3.12 V.
- R3: Add min of 6 ms and max of 15 ms, change nom to 10.5 ms.
- R4: Add min of 320 µs and max of 800 µs, change nom to 560 µs.

Table 4. Sandstorm Reset Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
R1	V_{TH}	Reset threshold	-	2.0	-	V
R2	V_{BTH}	Brown-Out threshold	2.85	2.9	2.95	V
R3	T_{POR}	Power-On Reset timeout	-	10	-	ms
R4	T_{BOR}	Brown-Out timeout	-	500	-	µs
R5	T_{IRPOR}	Internal reset timeout after POR	15	-	30	ms
R6	T_{IRBOR}	Internal reset timeout after BOR (1)	2.5	-	20	µs
R7	T_{IRHWR}	Internal reset timeout after hardware reset ($\overline{RST}$ pin)	2.9	-	29	µs
R8	T_{IRSWR}	Internal reset timeout afer software-initiated system reset(1)	2.5	-	20	µs
R9	T_{IRWDR}	Internal reset timeout after watchdog reset(1)	2.5	-	20	µs
R10	T_{IRLDOR}	Internal reset timeout after LDO reset(1)	2.5	-	20	µs
R11	$T_{VDDRISE}$	Supply voltage (V_{DD}) rise time (0 V-3.3 V)	-	-	100	ms

(1) 20 * t_{MOSC_per}

Table 5. Fury and DustDevil Reset Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
R1	V_{TH}	Reset threshold	-	2.0	-	V
R2	V_{BTH}	Brown-Out threshold	2.85	2.9	2.95	V
R3	T_{POR}	Power-On Reset timeout	-	10	-	ms
R4	T_{BOR}	Brown-Out timeout	-	500	-	µs
R5	T_{IRPOR}	Internal reset timeout after POR	6	-	11	ms
R6	T_{IRBOR}	Internal reset timeout after BOR(1)	0	-	1	µs
R7	T_{IRHWR}	Internal reset timeout after hardware reset ($\overline{RST}$ pin)	0	-	1	ms
R8	T_{IRSWR}	Internal reset timeout afer software-initiated system reset(1)	2.5	-	20	µs
R9	T_{IRWDR}	Internal reset timeout after watchdog reset(1)	2.5	-	20	µs
R10	$T_{VDDRISE}$	Supply voltage (V_{DD}) rise time (0 V-3.3 V), power-on reset	-	-	100	ms
		Supply voltage (V_{DD}) rise time (0 V-3.3 V), waking from hibernation	-	-	250	µs
R11	T_{MIN}	Minimum $\overline{RST}$ pulse width	2	-	-	µs

(1) 20 * t_{MOSC_per}

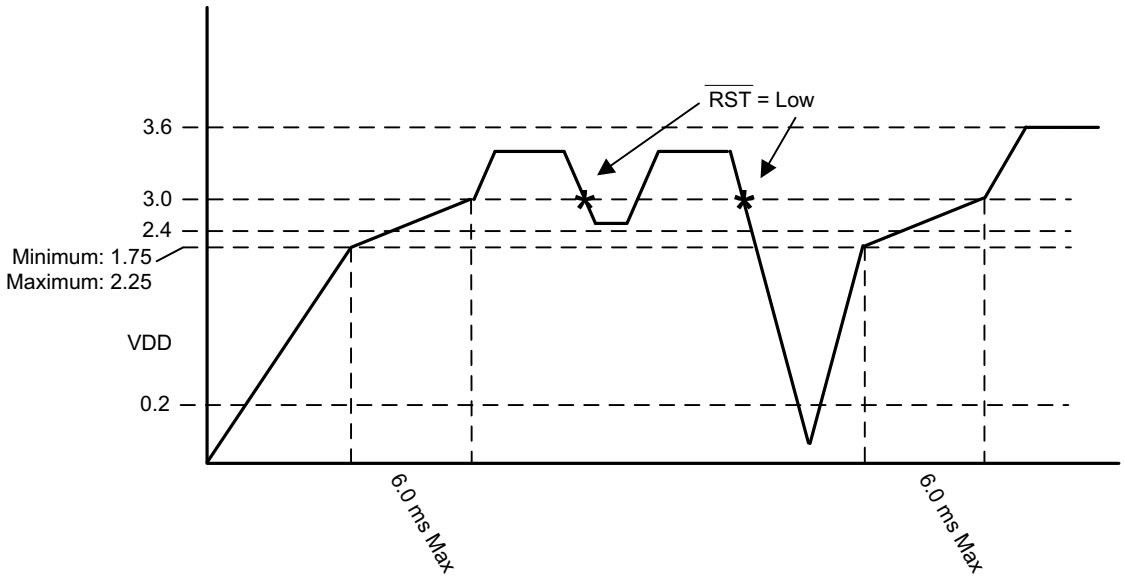


Figure 5. VDD Waveform Signature Limits

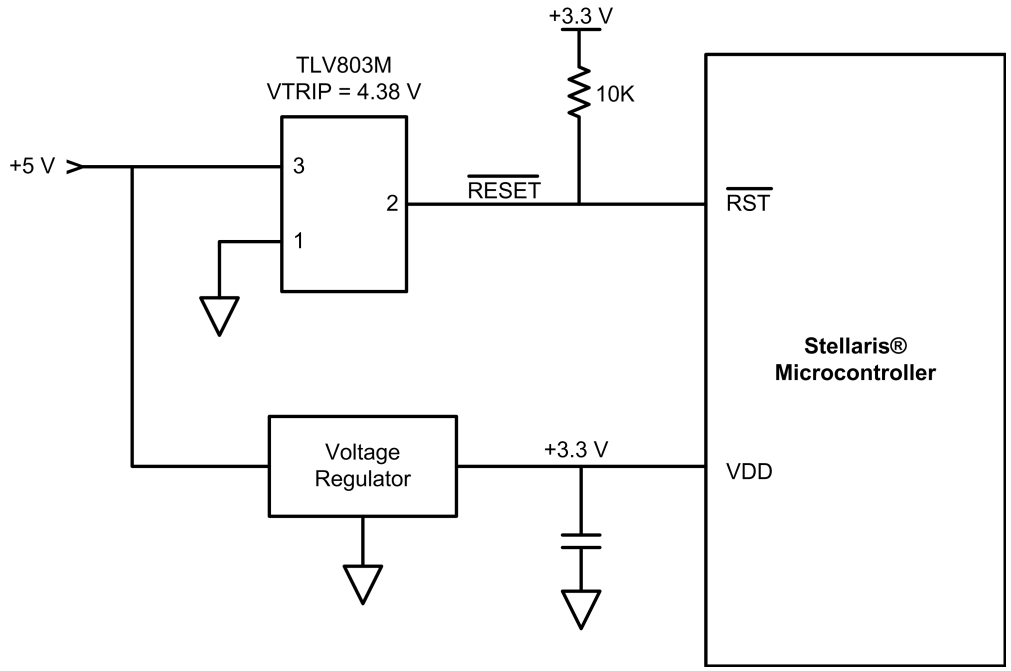


Figure 6. Using a Voltage Supervisor to Monitor the Voltage Rail

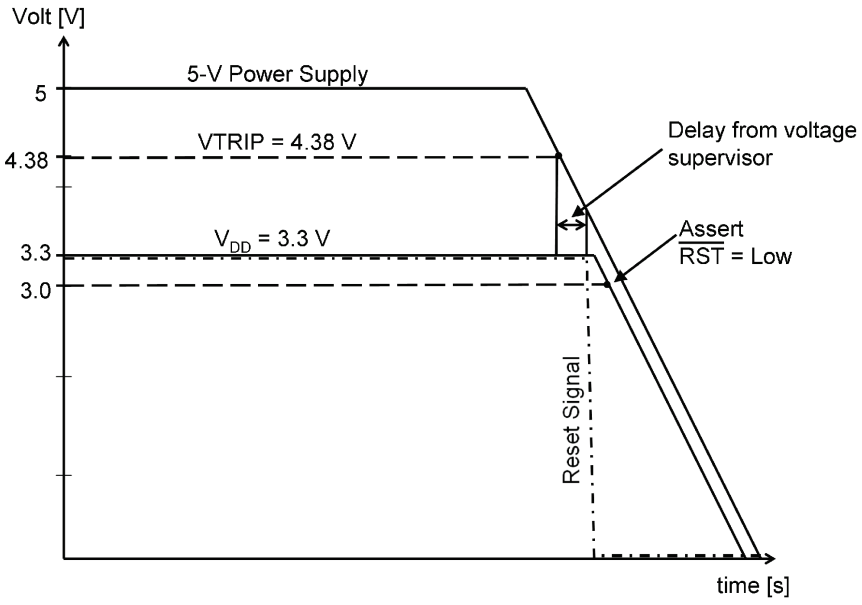


Figure 7. Resulting Waveform Using the Voltage Supervisor Circuit

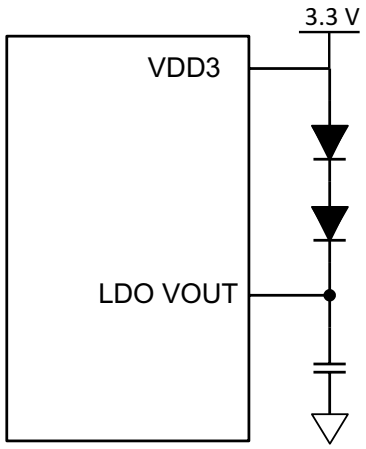


Figure 8. Diode Stack between VDD and LDO

LM3UART#01 *The RTRIS bit in the UARTRIS register is only set when the interrupt is enabled*

Device(s) Affected: Stellaris Sandstorm-class Rev C2, Fury-class Rev A2, and DustDevil-class Rev A0

Description: The RTRIS (UART Receive Time-Out Raw Interrupt Status) bit in the **UART Raw Interrupt Status (UARTRIS)** register should be set when a receive time-out occurs, regardless of the state of the enable RTIM bit in the **UART Interrupt Mask (UARTIM)** register. However, currently the RTIM bit must be set in order for the RTRIS bit to be set when a receive time-out occurs.

Workaround(s): For applications that require polled operation, the RTIM bit can be set while the UART interrupt is disabled in the NVIC using the IntDisable(n) function in the StellarisWare Peripheral Driver Library, where n is 21, 22, or 49 depending whether UART0, UART1 or UART2 is used. With this configuration, software can poll the RTRIS bit, but the interrupt is not reported to the NVIC.

LM3UART#03 *When UART SIR mode is enabled, µDMA burst transfer does not occur*

Device(s) Affected: DustDevil-class Rev A0

Description: If the IrDA Serial Infrared (SIR) mode is enabled in the UART peripheral and the µDMA UARTn RX or UARTn TX channel is configured to do a burst transfer, the burst data transfer does not occur.

Workaround(s): Clear the respective SETn bit in the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register to have the µDMA UART channel respond to single or burst requests to ensure that the data transfer occurs.

LM3USB#01 *Device Capabilities 6 (DC6) register incorrectly specifies USB OTG functionality*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: On the Stellaris controller, the **Device Capabilities 6 (DC6)** register incorrectly indicates that the part has OTG capability when it provides Host or Device connectivity (bits 1:0 read 0x3 when they should read 0x2).

Workaround(s): Do not attempt to use USB OTG functionality on this part. USB Host and Device capabilities are unaffected.

LM3USB#02 *Violating USB VBUS in-rush current specifications may cause USB controller to hang*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: Care should be taken to ensure that third-party USB devices used do not violate the USB specifications for maximum in-rush VBUS current. Connecting an out-of-spec USB device to the Stellaris USB controller may cause undesirable voltage glitches on the VBUS pin resulting in a controller hung condition.

Workaround(s): In the event of a VBUS glitch or error on the PB1 GPIO, a VBUS error interrupt is generated, so software is capable of detecting the glitch. As a guideline, the following strategy is suggested.

For USB OTG Mode:

1. Disable the external VBUS regulator.
2. Wait until the VBUS field in the **USB Device Control (USBDEVCTL)** register is 0x0.
3. Enable the external VBUS regulator.

For USB Device or Host Mode:

1. Disable the external VBUS regulator.
2. Reset the USB controller (for example, with the Stellaris Peripheral Driver Library function SysCtlPeripheral Reset(SYSCTL_PERIPH_USB0);).
3. Reinitialize the USB controller.
4. Enable the external VBUS regulator.

In experiments, success is most likely if steps 2 and 3 are as close to each other as possible.

NOTE: This erratum affects the USB controller if GPIO PB1 is connected to VBUS rather than an independent 5-V supply. If an independent 5-V supply is used, the glitch is unlikely to affect the PB1 voltage.

LM3USB#03 *PB0 and PB1 pins may not be used for GPIO or peripheral functions if USB Host or Device mode functionality is used*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: If the Stellaris microcontroller is used as a USB Host or USB Device controller, then the PB0 and PB1 pins must be tied to appropriate voltage levels.

Workaround(s): In order to use the USB controller in Host or Device modes, the PB1 pin must be tied to 5 V (4.75-5.25 V). In addition, the PB0 pin must be tied Low for USB Host operation or tied High for USB Device operation. Note that when tying PB0 Low through a resistor, the value of the resistor should not exceed the $R_{A\_PLUG\_ID}$ specification of 10 Ω for the USB cable. The PB0 and PB1 pins cannot be used for GPIO or peripheral functions if USB functionality is used.

LM3USB#04 *Transfer may stall when size of µDMA transfer does not match USB FIFO*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: When µDMA is used with USB to transfer data to or from the USB FIFO, and the size of the µDMA transfer does not match the size of the USB FIFO, the transfer can be stalled.

Workaround(s): The µDMA channel should be configured with an arbitration size that matches the size of the USB FIFO. The size of the µDMA transfer should be restricted to whole multiples of the size of the USB FIFO. This applies for both read and write transfers of the USB FIFOs using µDMA.

For example, the USB endpoint is configured with a FIFO size of 64 bytes. The µDMA channel should be configured with an arbitration size of 64. The µDMA channel can be used to transfer 64 bytes to or from the endpoint FIFO. If the number of bytes to transfer is less than 64, then a programmed I/O method should be used to copy the data to or from the FIFO.

LM3USB#05 *USB Host controller may not be used to communicate with a low-speed Device when connected through a hub*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: Occasionally when the USB controller is operating as a Host and a low-speed packet is sent to a Device when connected through a hub, the subsequent Start-of-Frame will be corrupted. After a period of time, this corruption causes the USB controller to lose synchronization with the hub, resulting in data corruption.

Workaround(s): None.

Fixed on devices with date codes of 0x1A or later. In addition, the system clock on the MCU must be at least 30 MHz. See Section 2, Device Date Code, for more information.

LM3USB#06 *The USB PLL may fail to lock after reset*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: The USB PLL may fail to lock after any type of reset approximately 1 in every 100 times. Once the lock failure has occurred, a software reset or core reset does not correct it. An external reset or a power-on reset is required to correct the condition. As a result of the lock failure, the USB controller fails to enumerate onto the USB bus. Failure to enumerate can be detected by reading the values of bits [4:3] in the **USB Device Control (USBDEVCTL)** register. If both of these bits are 0, the device has failed to enumerate.

Workaround(s): There is a workaround for software reset. There are two types of software reset, SYSRESREQ and VECTRESET. SYSRESREQ resets the entire device, whereas VECTRESET only resets the core. This problem can be eliminated by performing a VECTRESET using this code:

```
HWREG(NVIC_APINT) = NVIC_APINT_VECTKEY | NVIC_APINT_VECT_RESET;
```

VECTRESET only resets the core, so if any peripherals must be reset, use the peripheral software reset function, SysCtlPeripheralReset(). Note that if the USB module is reset using SysCtlPeripheralReset(), the USB PLL may fail to lock.

For any other type of reset, after detecting the enumeration failure, use a GPIO or an external watchdog device to issue an external reset.

Not yet fixed.

LM3USB#07 *USB OTG signaling does not function correctly*

Device(s) Affected: Stellaris DustDevil-class Rev A0

Description: The OTG signal levels for Session Request Protocol (SRP) and Host Negotiation Protocol (HNP) are not properly detected by the USB controller, which causes the USB controller to not be able to respond to OTG signaling properly. The USB controller can still use USB0ID pin detection to detect and control whether the controller enters Host or Device mode, but cannot properly respond to the OTG signaling from other devices.

Workaround(s): OTG USB0ID pin detection can be used to determine if the USB controller is functioning as Host or Device, but cannot be used to automatically switch from Host to Device or Device to Host.

LM3USB#15 *USB controller sends EOP at end of device Remote Wake-Up*

Devices(s) Affected: Stellaris DustDevil-class Rev A0

Description: When the USB controller is operating as a Device and is suspended by the Host, and the USB controller issues a remote wake-up, an end of packet (EOP) is sent to the Host at the end of the Device's remote wake-up signal. Although this EOP is not expected, issues related to remote wake-up have not been observed. This does not affect USB certification.

Workaround(s): None.

LM3USB#16 *Device sends SE0 in response to a USB bus reset*

Devices(s) Affected: Stellaris DustDevil-class Rev A0

Description: The USB Device (Tiva C MCU) will send an Single Ended Zero (SE0) bus state (USB0DP and USB0DM driven low) in response to a USB bus reset from the Host. Per USB specification, the Device should not drive these pins in the event of a USB bus reset. This does not affect USB certification.

Workaround(s): None.

LM3USB#17 *USB Resume occasionally does not wake device from Deep Sleep*

Devices(s) Affected: Stellaris DustDevil-class Rev A0

Description: If configured to wake from Deep Sleep mode using a USB Resume signal, the device may remain in Deep Sleep mode if the Host tries to resume the Device from Suspend with a USB bus reset before it can enter Deep Sleep. There is a finite window of time where the RESUME interrupt is not realized. This window is from when the RESUME bit in the **USB Device RESUME Interrupt Status and Clear (USBDRISC)** register is cleared (write a 1) to before the Device enters Deep Sleep. During this time, if a bus reset or wake-up signal is issued by the Host, then the **USBDRISC** status bit clearing causes the valid USB bus operation to be lost.

Workaround(s): To prevent this from occurring, perform one of the two options:

- Ensure that the USB Suspend handler is exited only after a WFI instruction is processed by the core.
- Use Sleep mode instead of Deep Sleep mode and keep the USB module enabled in Sleep mode.

To minimize the window of time when the RESUME interrupt can be lost and reduce the risk of this issue occurring, clear the RESUME bit as close as possible to entering Deep Sleep.

NOTE: If using Sleep mode with the USB module enabled (second workaround), MOSC must be the clock source, using the PLL, and the system clock must be at least 30 MHz. As a result of the higher system clock and using Sleep mode instead of Deep Sleep mode, the current consumption will be higher with this workaround.

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

This silicon errata revision history highlight the technical changes made to this document.

SEE	ADDITIONS/MODIFICATIONS/DELETIONS
Revision * (June 2014) Changes Below:	
Global	• Combined individual LM3Sxxx device errata into one errata document and formatted to TI style. • Created new issue numbers for easier tracking; number will stay the same in each revision going forward.
Section 4 Known Design Exceptions to Functional Specifications	Added the following advisories: • LM3ADC#18: Data may not be present in the FIFO at the time of the sequence interrupt or trigger • LM3ADC#19: The first two ADC samples may be incorrect • LM3GPTM#14: Writes to some General-Purpose Timer registers cause the counter to increment and decrement in some cases • LM3PWM#09: Under certain circumstances, the PWM load interrupt is triggered as soon as the PWM is enabled • LM3PWM#10: Setting the PWMSYNC bits may not synchronize the PWM counters if PWMDIV is used • LM3QEI#03: When using the index pulse to reset the counter, a specific initial conditions in the QEI module causes the direction for the first count to be misread • LM3SSI#02: SSI Receive FIFO Time-out interrupt may assert sooner than expected in slave • LM3SYSCTL#24: DSDIVORIDE value of 0x1 does not divide Deep Sleep clock by 2 • LM3SYSCTL#25: Special considerations for power transitions are required to ensure correct device operation • LM3UART#03: When UART SIR mode is enabled, µDMA burst transfer does not occur • LM3USB#15: USB controller sends EOP at end of device Remote Wake-Up • LM3USB#16: Device sends SE0 in response to a USB bus reset • LM3USB#17: USB Resume occasionally does not wake device from Deep Sleep

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265