Tiva™ C Series TM4C123x Microcontrollers Silicon Revisions 6 and 7

Silicon Errata

Literature Number: SPMZ849F
August 2013–Revised April 2016

Contents

Tiva™ C Series TM4C123x Microcontrollers Silicon Revisions 6 and 7

1 Introduction

This document describes known exceptions to the functional specifications for all of the Tiva™ C Series TM4C123x microcontrollers. Note that some features are not available on all devices in the series, so not all errata may apply to your device. See your device-specific data sheet for more details.

For details on ARM® Cortex™-M4F CPU advisories, see the *ARM Core Cortex™-M4 (AT520) and Cortex-M4F (AT521) Errata Notice* (literature number: SPMZ637).

2 Device Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all microcontroller (MCU) devices. Each Tiva C series family member has one of two prefixes: XM4C or TM4C (for example, **XM4C**123GH6PMT7). These prefixes represent evolutionary stages of product development from engineering prototypes (XM4C) through fully qualified production devices (TM4C).

Device development evolutionary flow:

XM4C — Experimental device that is not necessarily representative of the final device's electrical specifications and may not use production assembly flow.

TM4C — Production version of the silicon die that is fully qualified.

XM4C devices are shipped against the following disclaimer:

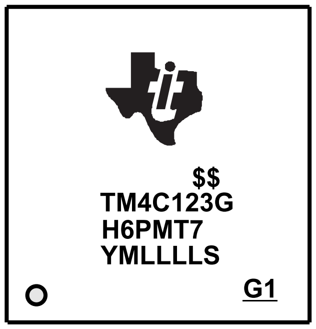
"Developmental product is intended for internal evaluation purposes."

TM4C devices have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

Predictions show that prototype devices (XM4C) have a greater failure rate than the standard production devices. Texas Instruments recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are to be used.

3 Device Markings

Figure 1 shows an example of the Tiva™ C Series TM4C123x microcontroller package symbolization.



Device Revision Code

Figure 1. Example of Device Part Markings

This identifying number contains the following information:

- **Lines 1 and 5:** Internal tracking numbers
- **Lines 2 and 3:** Part number

For example, TM4C123G on the second line followed by H6PMT7 on the third line indicates orderable part number TM4C123GH6PMT7. Note that the first letter in the part number indicates the product status. A T indicates the part is fully qualified and released to production; an X indicates the part is experimental (pre-production) and requires a waiver. The revision number is also included in the part number, for example, **T**M4C123G or **X**M4C123G followed by H6PMT**7** indicates revision 7. The **DID0** register identifies the version of the microcontroller, as shown in Table 1. The MAJOR and MINOR bit fields indicate the die revision number. Combined, the MAJOR and MINOR bit fields indicate the TM4C123x microcontroller silicon revision number.

Table 1. Tiva™ C Series TM4C123x Silicon Revision Codes

MAJOR Bit Field Value	MINOR Bit Field Value	Die Revision	Silicon Revision
0x0	0x0	A0	1
0x0	0x1	A1	2
0x0	0x2	A2	3
0x0	0x3	A3	4
0x1	0x0	B0	5
0x1	0x1	B1	6
0x1	0x2	B2	7

- **Line 4:** Date code The first two characters on the fourth line indicate the date code, followed by internal tracking numbers. The two-digit date code YM indicates the last digit of the year, then the month. For example, a 34 for the first two digits of the fourth line indicates a date code of April 2013.

4 Advisory to Silicon Revision Correlation

Table 2. Advisory to Silicon Revision Matrix

Advisory Number	Advisory Title	Silicon Revision(s) Affected	
		6	7
ADC			
ADC#01	Retriggering a Sample Sequencer Before it has Completed the Current Sequence Results in Continuous Sampling	X	X
ADC#03	Digital Comparator in Last Step of Sequence Does not Trigger or Interrupt	X	X
ADC#04	Digital Comparator Interrupts do not Trigger or Interrupt as Expected	X	X
ADC#07	ADC Sample Sequencers Priorities are Different Than Expected	X	X
ADC#08	ADC Sample Sequencer Only Samples When Using Certain Clock Configurations	X	X
ADC#09	First two ADC Samples From the Internal Temperature Sensor Must be Ignored	X	X
ADC#11	The DITHER bit in the ADC Control (ADCCTL) Register Does not Function	X	X
ADC#13	A Glitch can Occur on pin PE3 When Using any ADC Analog Input Channel to Sample	X	X
ADC#14	The First two ADC Samples may be Incorrect	X	X
ADC#16	Phase Offset does not Delay as Expected if Sample Sequencers are not Triggered at the Same Time	X	X
DMA			
DMA#01	In Three Cases, two Peripherals Cannot Both be Programmed to use µDMA	X	X
DMA#02	µDMA may be Corrupted if Transferred or Received While Entering or Exiting Deep Sleep Mode	X	X
ELEC			
ELEC#02	V_{BAT} Supply pin may be Damaged if the pin Voltage Ramps Faster Than 0.7 V/µs	X	X
ELEC#04	Equation for C_L for HIBXOSC is not Correct	X	X
ELEC#05	PIOSC Frequency Variation more than +/-3%	X	X
GPIO			
GPIO#01	JTAG Controller Does not Ignore Transitions on PC0/TCK When it is Configured as a GPIO	X	X
GPIO#07	GPIO Interrupts Do Not Function Correctly on Ports P and Q	X	X
GPIO#08	Certain GPIOs Have Limited Pin Configurations	X	X
GPIO#10	In Some Cases, Noise Injected Into GPIO Pins can Cause High Current Draw	X	X
General-Purpose Timers			
GPTM#01	GPTMSYNC Bits Require Manual Clearing	X	X
GPTM#02	The GPTMPP Register Does not Correctly Indicate 32/64-bit Timer Capability	X	X
GPTM#04	Wait-for-Trigger Mode is not Available for PWM Mode	X	X
GPTM#09	General-Purpose Timers do not Synchronize When Configured for RTC or Edge Count Mode	X	X
GPTM#10	Writes to Some General-Purpose Timer Registers Cause the Counter to Increment and Decrement in Some Cases	X	X
GPTM#11	The Prescalar Does not Work Properly When Counting up in Input Edge-Time Mode When the GPTM Timer n Interval Load (GPTMTnILR) Register is Written With 0xFFFF	X	X
GPTM#15	Counter Does not Immediately Clear to 0 When MATCH is Reached In Edge Count Up Mode	X	X
Hibernation			
HIB#01	Some Hibernation Module Registers may not Have the Correct Value in two Situations	X	X
HIB#02	Reading the HIBRTCC and HIBRTCSS Registers may Provide Incorrect Values	X	X
HIB#03	Device Fails to Wake From Hibernation Within a Certain Time after Hibernation is Requested	X	X
HIB#04	RTC Match Event is Missed if it Occurs in a Certain Window	X	X
HIB#14	External Wake Interrupt may be Lost When Returning From Hibernation	X	X

Table 2. Advisory to Silicon Revision Matrix (continued)

Advisory Number	Advisory Title	Silicon Revision(s) Affected	
		6	7
	I2C		
I2C#04	I2C Glitch Filter Suppression Width may Differ From the Configured Value	X	X
I2C#06	I2C Slave Alternate Address Disable does not Function as Expected	X	X
I2C#07	I2C Master does not Clear ERROR Status bit	X	X
	Memory		
MEM#02	The START bit in the EEPROM Support Control and Status (EESUPP) Register Does not Function	X	X
MEM#03	EEPROM Data May be Corrupted if an EEPROM Write or Erase is Interrupted	X	
MEM#04	Device may Become Non-functional if an EEPROM Write or Erase is Interrupted	X	
MEM#05	Device may Become Non-functional if Power is Interrupted During an Unlock of the Microcontroller or During Non-volatile Register Commits	X	X
MEM#07	Soft Resets Should not be Asserted During EEPROM Operations		X
MEM#08	Writes and Erases to the EEPROM will not Work if the Three EEPROM Password Registers are Used for Last EEPROM Block	X	X
MEM#10	The START bit in the EESUPP Register may Cause EEPROM Corruption		X
MEM#11	The ROM Version of the TivaWare EEPROMInit API Does not Correctly Initialize the EEPROM		X
MEM#14	Flash Write Operation During Execute from Flash may Result in Wrong Instruction Fetch	X	X
MEM#19	Certain GPIOs Cannot be Configured as boot pins	X	X
	PWM		
PWM#01	Under Certain Circumstances, the PWM Load Interrupt is Triggered as Soon as the PWM is Enabled	X	X
PWM#02	Setting the PWMSYNC Bits May Not Synchronize the PWM Counters if PWMDIV is Used	X	X
PWM#04	PWM Generator Interrupts can only be Cleared 1 PWM Clock Cycle After the Interrupt Occurs	X	X
	QEI		
QEI#01	When Using the Index Pulse to Reset the Counter, a Specific Initial Condition in the QEI Module Causes the Direction for the First Count to be Misread	X	X
	SSI		
SSI#06	SSI Receive FIFO Time-out Interrupt may Assert Sooner than Expected in Slave Mode	X	X
SSI#07	SSI Transmit Interrupt Status Bit is not Latched	X	X
	System Control		
SYSCTL#01	With a Specific Clock Configuration, Device may not Wake From Deep Sleep Mode	X	X
SYSCTL#03	The MOSC Verification Circuit Does not Detect a Loss of Clock After the Clock has been Successfully Operating	X	X
SYSCTL#04	Device May not Wake Correctly From Sleep Mode Under Certain Circumstances	X	X
SYSCTL#06	Resets Fail While in Deep Sleep When Using Certain Clock Configurations	X	X
SYSCTL#07	Deep Sleep Clock Frequency Incorrect if a Watchdog Reset Occurs Upon Entry	X	X
SYSCTL#11	Longer Reset Pulse Needed if Device is in Deep Sleep Mode With the LFIOSC as the Clock Source	X	X
SYSCTL#14	Power Consumption is Higher When MOSC is Used in Single-Ended Mode	X	X
SYSCTL#16	On-Chip LDO may not Start Properly During Power Up	X	X
SYSCTL#17	DSDIVORIDE Value of 0x1 Does not Divide Deep Sleep Clock by 2	X	X
SYSCTL#20	DID1 Register Incorrect	X	
SYSCTL#21	Reset Cause may not be Logged in RESC Register	X	X
	UART		
UART#01	When UART SIR Mode is Enabled, μDMA Burst Transfer Does not Occur	X	X

Table 2. Advisory to Silicon Revision Matrix (continued)

Advisory Number	Advisory Title	Silicon Revision(s) Affected	
		6	7
	USB		
USB#01	USB Host Controller may not be Used to Communicate With a Low-Speed Device When Connected Through a hub	X	X
USB#02	USB Controller Sends EOP at end of Device Remote Wake-Up	X	X
USB#04	Device Sends SE0 in Response to a USB Bus Reset	X	X
USB#05	USB Resume Occasionally does not Wake Device from Deep Sleep	X	X
	Watchdog Timers		
WDT#01	Watchdog Timer 1 Module Cannot be Used Without Enabling Other Peripherals First	X	X
WDT#02	Watchdog Clear Mechanism Described in the Data Sheet Does not Work for the Watchdog Timer 1 Module	X	X
WDT#03	Watchdog Timer 1 Module Asserts Reset Signal Even if not Programmed to Reset	X	X
WDT#05	WDTLOAD Yields an Incorrect Value When Read Back	X	X
WDT#06	WDTMIS Register Does not Indicate an NMI Interrupt From WDT0	X	X
WDT#07	The Watchdog Load (WDTLOAD) Register Cannot be Changed When Using a Debugger While the STALL bit is set	X	X
WDT#08	Reading the WDTVALUE Register may Return Incorrect Values When Using Watchdog Timer 1	X	X

5 Known Design Exceptions to Functional Specifications

Table 3. Advisory List

Table 3. Advisory List (continued)

ADC#01 *Retriggering a Sample Sequencer Before it has Completed the Current Sequence Results in Continuous Sampling*

Revision(s) Affected: 6 and 7.

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.

ADC#03 *Digital Comparator in Last Step of Sequence Does not Trigger or Interrupt*

Revision(s) Affected: 6 and 7.

Description: If a digital comparator that is expected to trigger or interrupt is configured for the last step of a sample sequence with sequence trigger TRIGGER_PROCESSOR, TRIGGER_COMP, TRIGGER_EXTERNAL, TRIGGER_TIMER, or TRIGGER_PWM, the trigger or interrupt does not occur. These sequence trigger parameters should not be used when using a sample sequencer configured with only one step and a digital comparator that is expected to trigger or interrupt. Note that Sample Sequencer 3 can only be configured for a total of one step.

Workaround(s): If an extra sequence step is available in a sample sequencer, a dummy sequence step and a dummy digital comparator can be configured as the last step in the sample sequencer.

ADC#04 *Digital Comparator Interrupts do not Trigger or Interrupt as Expected*

Revision(s) Affected: 6 and 7.

Description: The digital comparator configured for the ADC sample sequence step (n+1) is triggered if the voltage on the AINx input specified for step (n) meets the conditions that trigger the digital comparator for step (n+1). In this case, the conversion results are sent to the digital comparator specified by step (n+1).

Workaround(s): Adjust user code or hardware to account for the fact that the voltage seen at the AINx input specified for sequence step (n) will be handled by sequence step (n+1)’s digital comparator using sequence step (n+1)’s configurations.

ADC#07 *ADC Sample Sequencers Priorities are Different Than Expected*

Revision(s) Affected: 6 and 7.

Description: If sample sequencer 2 (SS2) and sample sequencer 3 (SS3) have been triggered, and sample sequencer 0 (SS0) and sample sequencer 1 (SS1) have not been triggered or have already been triggered, the priority control logic compares the priorities of SS1 and SS2 rather than SS2 and SS3. For example, if SS1's priority is the highest (such as 0) and SS3's priority is higher than SS2's priority (such as SS3 = 1, SS2 = 2), SS2 is incorrectly selected to initiate the sampling conversion after SS1. If SS1's priority is the lowest (such as 3) and SS3's priority is lower than SS2's (such as SS3 = 2, SS2 = 1), SS3 is incorrectly selected as the next sample sequencer, then SS2, then SS1.

Workaround(s): If only three of the four ADC sample sequencers are needed, SS0 and SS1 can be used with either SS2 or SS3. This ensures that the execution order is as expected. If all four ADC sample sequencers are needed, the highest priority conversions should be programmed into SS0 and SS1. The sequences programmed into SS2 and SS3 occur, but not necessarily in the programmed priority order.

ADC#08 *ADC Sample Sequencer Only Samples When Using Certain Clock Configurations*

Revision(s) Affected: 6 and 7.

Description: The ADC sample sequencer does not sample if using either the MOSC or the PIOSC as both the system clock source and the ADC clock source.

Workaround(s): There are three possible workarounds:

- Enable the PLL and use it as the system clock source.
- Configure the MOSC as the system clock source and the PIOSC as the ADC clock source.
- Enable the PLL, configure the PIOSC as the ADC clock source and as the system clock source, then subsequently disable the PLL using HWREG(0x400fe060) != 0x00000200.

ADC#09 *First two ADC Samples From the Internal Temperature Sensor Must be Ignored*

Revision(s) Affected: 6 and 7.

Description: The analog source resistance (R_S) to the ADC from the internal temperature sensor exceeds the specified amount of 500Ω. This causes a settling time requirement that is longer than the sampling interval to the converter.

Workaround(s): Three consecutive samples from the same channel must be taken to accurately sample the internal temperature sensor using the ADC. The first two consecutive samples should be discarded and the third sample can be kept. These consecutive samples cannot be interrupted by sampling another channel.

ADC#11 *The DITHER bit in the ADC Control (ADCCTL) Register Does not Function*

Revision(s) Affected: 6 and 7.

Description: The DITHER bit in the ADC Control (ADCCTL) register does not function.

Workaround(s): None.

ADC#13 A Glitch can Occur on pin PE3 When Using any ADC Analog Input Channel to Sample

Revision(s) Affected:	6 and 7.
Description	A glitch may occur on PE3 when using any ADC analog input channel (AINx) to sample. This glitch can occur when PE3 is configured as a GPIO input or as a GPIO open drain and happens at the end of the ADC conversion. These glitches will not affect analog measurements on PE3 when configured as AIN0 as long as the specified source resistance is met.
Workaround(s)	A 1kΩ external pull-up or pull-down on PE3 will help to minimize the magnitude of the glitch to 200 mV or less.

ADC#14 *The First two ADC Samples may be Incorrect*

Revision(s) Affected: 6 and 7.

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

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

ADC#16 *Phase Offset does not Delay as Expected if Sample Sequencers are not Triggered at the Same Time*

Revision(s) Affected: 6 and 7.

Description: The phase difference set in the ADC Sample Phase Control (ADCSPC) register does not reference the same starting point in time if the sequencers are configured for a phase offset and are not triggered at the same time.

Workaround(s): Use the same trigger to ensure that the sample sequencers will trigger at the same time. If using processor trigger and both ADC modules with phase offset, use the GSYNC and SYNCWAIT bits in the ADC Processor Sample Sequence Initiate (ADCPSSI) register to ensure that the trigger occurs simultaneously. The phase offsets will not align if triggering using trigger always mode.

DMA#01 *In Three Cases, two Peripherals Cannot Both be Programmed to use µDMA*

Revision(s) Affected: 6 and 7.

Description: For the following pairs of peripherals, both peripherals cannot both be configured to use µDMA:

- SSI0 and SSI1
- UART2 and USB0EP1
- UART0 and UART2

Workaround(s): Configure peripherals such that the combinations of peripherals listed above are not both using µDMA.

DMA#02 *µDMA Data may be Corrupted if Transferred or Received While Entering or Exiting Deep Sleep Mode*

Revision(s) Affected: 6 and 7.

Description: Transferred or received data using the µDMA from either the UART or the SSI peripherals may get corrupted when entering Deep Sleep mode from Run mode or exiting Deep Sleep mode to Run mode if the Run mode clock configuration is not the same as the Deep Sleep mode clock configuration.

Workaround(s): Program the Run mode clock configuration to match the Deep Sleep mode clock configuration right before entering Deep Sleep mode.

ELEC#02 *V_{BAT} Supply pin may be Damaged if the pin Voltage Ramps Faster Than 0.7 V/µs*

Revision(s) Affected: 6 and 7.

Description The V_{BAT} supply pin may be damaged if the pin voltage ramps faster than 0.7 V/µs. Fast V_{BAT} ramps are a concern when a battery is being connected or the V_{BAT} supply is hard switched.

Workaround(s) An RC circuit as shown should be added to the V_{BAT} pin to prevent the damage. The R_1 and C_1 should be placed close to the microcontroller for best protection. In systems that do not require Hibernate when the VDD supply is off, the V_{BAT} pin should be tied to the VDD supply, which typically ramps at a rate slower than 0.7 V/µs. The R1 and C1 components are not required for a V_{BAT} supply ramp less than 0.7 V/µs.

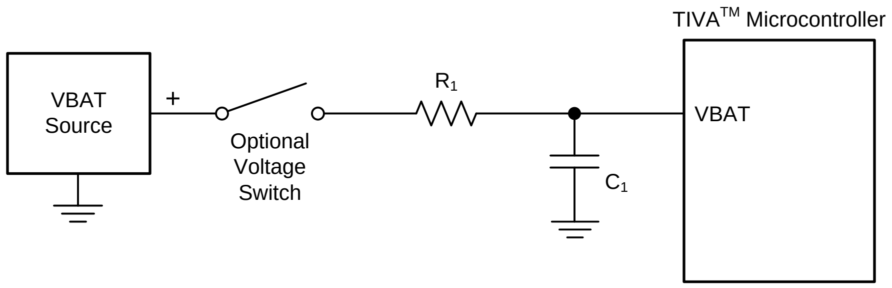


Figure 2. RC Circuit

ELEC#04 *Equation for C_L for HIBXOSC is not Correct*

Revision(s) Affected: 6 and 7.

Description: The Electrical Specification for C_L for HIBXOSC is a function $C_L = (C_1*C_2)/(C_1+C_2) + C_{PKG} + C_{PCB}$ and does not include the C_0 Shunt Capacitance Specified by Crystal Manufactured.

Workaround(s): The correct equation is as follows:

$C_L = (C_1*C_2)/(C_1+C_2) + C_{SHUNT}$

$C_{SHUNT} = C_0 + C_{PKG} + C_{PCB}$

ELEC#05 *PIOSC Frequency Variation more than +/-3%*

Revision(s) Affected: 6 and 7.

Description: The F_{PIOSC} as per TM4C123 device data sheet may have a frequency variation of more than +/-3% for factory calibrated parts.

Workaround(s): The following are the suggested workarounds:

- The user design must instead use the Main Oscillator as source of system clock.

  OR
- The user must add the 32768Hz Hibernate Oscillator to allow periodic recalibration of the PIOSC to be performed by Software.

GPIO#01 *JTAG Controller Does not Ignore Transitions on PC0/TCK When it is Configured as a GPIO*

Revision(s) Affected: 6 and 7.

Description: When PC0/TCK is configured as a GPIO, toggling on the pin may cause the device to execute unexpected JTAG instructions.

Workaround(s): Only use PC0/TCK as a JTAG pin. Do not use it as a GPIO. Ensure that this pin is connected to a pull-up to VDD.

GPIO#07 *GPIO Interrupts Do Not Function Correctly on Ports P and Q*

Revision(s) Affected: 6 and 7.

Description: Ports P and Q are designed to provide either a single interrupt where interrupts for all pins on the port are OR'ed together or multiple interrupts where each port pin has an individual interrupt. This function is controlled by the SUM bit in the GPIO Select Interrupt (GPIOSI) register. When SUM is 0 and the interrupt occurs on a pin other than pin 0, an interrupt on pin 0 is triggered in addition to the interrupt on the other pin(s). The interrupt on the pin(s) other than pin 0 is unexpected.

Workaround(s): To configure GPIO ports P or Q for summary interrupt mode, the following additional steps are required:

- For a port pin to be included in the summary interrupt on P0 or Q0 the corresponding bit must be set in the GPIOIM register
- For each port pin other than P0 or Q0 that is enabled in GPIOIM, the corresponding bit(s) must be set in the Interrupt Clear Disable (DISn) register to avoid generating undesired interrupts

For each port pin that is configured as edge-detect in summary interrupt mode, the following additional steps are required in the P0 or Q0 interrupt service routine:

- Write a 1 to the corresponding bit(s) in GPIOICR to clear the interrupt in the GPIO module
- Write a 1 to the corresponding bit(s) in the UNPENDn register to clear the pending interrupt in the NVIC

For each port pin that is configured as level-detect in summary interrupt mode, the following additional steps are required in the P0 or Q0 interrupt service routine:

- Write a 1 to the corresponding bit(s) in the UNPENDn register to clear the pending interrupt in the NVIC

To configure ports P and Q for per-pin interrupt mode, no additional steps are required.

GPIO#08 *Certain GPIOs Have Limited Pin Configurations*

Revision(s) Affected: 6 and 7.

Description: The following pins (which are dependent on pin package) are fixed at a 4 mA pad drive and are not configurable for open drain:

- PL6 and PL7 (157-BGA, 144-LQFP)
- PD4 and PD5 (64-LQFP)
- PJ0 and PJ1 (100-LQFP)

Writes to the GPIODR2R, GPIODR8R, or GPIOODR registers for these two pins have no effect.

Workaround(s): None.

GPIO#10 *In Some Cases, Noise Injected Into GPIO Pins can Cause High Current Draw*

Revision(s) Affected: 6 and 7.

Description: A fast transition on any GPIO pin can switch on a low resistance path between the GPIO pin or the adjacent GPIO pins and ground potentially causing a high current draw.

This condition has been observed when the signal at the device pin has a slew rate such that the rise time or fall time (measured from 10% to 90% of VDD) is faster than 2ns. The condition is more likely to occur at high temperatures or in noisy environments. It can occur when the pin is in input or output mode or with any pin multiplexing options.

If the condition is induced while the pin is configured as an output GPIO, then changing the pin state to low and then returning it to a high state at a lower temperature will resolve the condition.

Workaround(s): Limit the slew rate on the IO so that the fastest rise and fall time is greater than 2 ns. Use an RC filter or series ferrite to limit the rise and fall times at the device pin. The filter capacitor should be placed as close as possible to the device.

GPTM#01 *GPTMSYNC Bits Require Manual Clearing*

Revision(s) Affected: 6 and 7.

Description: The GPTM Synchronize (GPTMSYNC) register allows software to synchronize a number of timers. The bits in this register should be self-clearing after setting bits to synchronize selected timers, but they are not.

Workaround(s): When bits in the GPTMSYNC register are set, software must clear the bits prior to setting them for a subsequent update. When using TivaWare™ APIs, instead of just calling the *TimerSynchronize()* function once, software should call the function a second time with 0 as a parameter, as shown below:

```
TimerSynchronize(TIMER0_BASE, TIMER_0A_SYNC | TIMER_1A_SYNC);
TimerSynchronize(TIMER0_BASE, 0);
```

GPTM#02 *The GPTMPP Register Does not Correctly Indicate 32/64-bit Timer Capability*

Revision(s) Affected:	6 and 7.
Description:	The GPTM Peripheral Properties (GPTMPP) register reads as 0x0 on the 32/64-bit wide timers, which indicates that the timer is a 16/32-bit timer. It should read as 0x1 on these timers, indicating a 32/64-bit wide timer.
Workaround(s):	In situations where code is required to dynamically determine the capabilities of a specific timer, create a look-up table based on the CLASS field of the Device Identification 0 (DID0) register.

GPTM#04 *Wait-for-Trigger Mode is not Available for PWM Mode*

Revision(s) Affected: 6 and 7.

Description: Daisy chaining functionality of the general-purpose timers is only valid for One-shot and Periodic modes. If the TnWOT bit of the GPTM Timer n Mode (GPTMTnMR) register is set, and the nth timer is configured for PWM mode, the nth timer will not wait for the (n-1)th timer to trigger it and will begin counting immediately when enabled. If, instead, the nth timer is configured for One-shot or Periodic mode and the (n-1)th timer is configured for PWM mode, the nth timer would never begin counting as it will never receive a trigger from the (n-1)th timer in the daisy chain.

Workaround(s): None.

GPTM#09 *General-Purpose Timers do not Synchronize When Configured for RTC or Edge Count Mode*

Revision(s) Affected: 6 and 7.

Description: When attempting to synchronize the General-Purpose Timers using the GPTM Synchronize (GPTMSYNC) register, they do not synchronize if any of the timers are configured for RTC or Edge Count mode.

Workaround(s): None.

GPTM#10 *Writes to Some General-Purpose Timer Registers Cause the Counter to Increment and Decrement in Some Cases*

Revision(s) Affected: 6 and 7.

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.

GPTM#11 *The Prescalar Does not Work Properly When Counting up in Input Edge-Time Mode When the GPTM Timer n Interval Load (GPTMTnILR) Register is Written With 0xFFFF*

Revision(s) Affected: 6 and 7.

Description: If the GPTM is configured in Input Edge-Time count-up mode with the GPTM Timer n Interval Load (GPTMTnILR) register equal to 0xFFFF, the prescaler does not work properly.

Workaround(s): Do not load 0xFFFF into the GPTMTnILR register when counting up in Input Edge-Time mode.

GPTM#15 *Counter Does not Immediately Reset to 0 When MATCH is Reached In Edge Count Up Mode*

Revision(s) Affected: 6 and 7.

Description When configured for input edge count mode and count up mode, after counting to the match value, the counter uses one additional edge to reset the timer to 0. As a result, after the first match event, all subsequent match events occur after the programmed number of edge events plus one.

Workaround(s) In software, account for one additional edge in the programmed edge count after the first match interrupt is received.

HIB#01 *Some Hibernation Module Registers may not Have the Correct Value in two Situations*

Revision(s) Affected: 6 and 7.

Description: Some Hibernation module registers may not have the correct value in two different situations:

1. After enabling the hibernation 32-kHz oscillator by setting the CLK32EN bit in the Hibernation Control (HIBCTL) register.
2. When the CLK32EN bit is set, both the RTCEN and PINWEN bits in the HIBCTL register are clear, and any kind of reset occurs.

The following Hibernation module registers are affected:

- HIBRTCLD
- HIBRTCM0
- HIBRTCSS
- HIBRTCT
- HIBIM

Note that the register values may or may not be correct, but software cannot assume that these registers have any specific values following the occurrence of the situations described above.

Workaround(s): Ensure that every bit in these registers is correctly initialized in application software following the occurrence of the situations described above.

HIB#02 *Reading the HIBRTCC and HIBRTCSS Registers may Provide Incorrect Values*

Revision(s) Affected: 6 and 7.

Description: Reads from the Hibernation RTC Counter (HIBRTCC) and Hibernation RTC Sub Seconds (HIBRTCSS) registers may not be correct.

Workaround(s): Use the following code sequence to read from the HIBRTCC and HIBRTCSS registers:

```
//
// Disable Interrupts
//
IntMasterDisable();
//
// A) For HIB_RTCC or HIB_RTCSS individual register reads
//
do
{
    ulRTC = HWREG(HIB_RTCC);
}   while (ulRTC != HWREG(HIB_RTCC));
//
// B) For synchronized reads of both the HIB_RTCC and HIB_RTCSS
//
do {
    ulRTC = HWREG(HIB_RTCC);
    ulRTCSS = HWREG(HIB_RTCSS);
    ulRTCSS2 = HWREG(HIB_RTCSS);
    ulRTC1 = HWREG(HIB_RTCC);
}   while ((ulRTC != ulRTC1) || (ulRTCSS != ulRTCSS2));
//
// Re-enable interrupts
//
IntMasterEnable();
```

HIB#03 *Device Fails to Wake From Hibernation Within a Certain Time after Hibernation is Requested*

Revision(s) Affected: 6 and 7.

Description: If a wake event occurs during a small window after the device enters Hibernate mode, the device cannot wake from hibernation. The window in which this issue occurs extends from 31 µs before the $\overline{\text{HIB}}$ signal is asserted until V_{DD} drops below the BOR threshold, if BOR is enabled, or the POR falling edge threshold. Note that this erratum does not apply when using the VDD3ON mode because V_{DD} does not drop in this mode.

Workaround(s): Add a TivaWare™ SysCtlReset() function after the hibernation request in the following manner:

```
HibernateRequest();
//
// Wait till the isolation has been applied
//
while ((HWREG(HIB_CTL) & HIB_CTL_CLK32EN) == HIB_CTL_CLK32EN)
{
}
SysCtlReset();
```

In addition, add the following code to the reset handler

```
//
// Halt code execution if in Hibernate as supplies decay
//
while( HWREG(HIBCTL) == 0x80000000)
{
}
```

HIB#04 *RTC Match Event is Missed if it Occurs in a Certain Window*

Revision(s) Affected:	6 and 7.
Description:	An RTC match event is missed if the match occurs within three 32.768-kHz clocks (92 µs) after setting the HIBREQ bit in the Hibernation Control (HIBCTL) register.
Workaround(s):	Compare the RTC counter value before going into hibernation with the RTC match value and if the match is within three counts of the RTC sub seconds counter, hold off entering into hibernation until the match has occurred.

HIB#14 *External Wake Interrupt may be Lost When Returning From Hibernation*

Revision(s) Affected: 6 and 7.

Description

A $\overline{\text{WAKE}}$ pin interrupt, EXTW, may be lost while returning from non-VDD3ON hibernation. The sequence begins as $\overline{\text{WAKE}}$ is asserted to trigger the exit of hibernation. The $\overline{\text{HIB}}$ is de-asserted to enable the VDD supply. If the VDD supply drops below the Power-On Reset threshold after being above the threshold for 1-2 hibernate clock cycles (typically 30-60 µs), the device continues to wake, but the EXTW interrupt will not occur.

Workaround(s)

For systems which require all EXTW events to be accounted for, one of two methods may be used to ensure an EXTW interrupt is not missed.

- Do not generate a wake event until the VDD supply has dropped below the minimum Power-On Reset threshold.
- Ensure the VDD supply begins to rise less than 2 hibernate clock cycles (typically 60 µs) from when the $\overline{\text{HIB}}$ signal has been de-asserted.

Once the supply is above the Power-On Reset threshold for 1-2 hibernate clock cycles (typically 30-60 µs), the supply should not drop below the Power-On Reset threshold to retain the EXTW interrupt.

I2C#04 *I²C Glitch Filter Suppression Width may Differ From the Configured Value*

Revision(s) Affected: 6 and 7.

Description: The I²C glitch filter pulse width is configured using the GFPW bit field in the I²C Master Configuration 2 (I2CMCR2) register. Due to a logic error in the initialization of the glitch filter, the actual pulse width may differ from what is expected. This issue rarely occurs, but is not predictable in software. The following table outlines the different pulse widths that may occur with respect to the value written to the GFPW bit field.

Table 4. Expected vs. Actual I²C Glitch Suppression Pulse Width

GFPW[6:4]	Glitch Suppression Pulse Width	
	Expected Value [system clocks]	Actual Value [system clocks]
0x0	Bypass	Bypass
0x1	1 clock	0-1 clock
0x2	2 clocks	0-3 clocks
0x3	3 clocks	0-3 clocks
0x4	4 clocks	0-7 clocks
0x5	8 clocks	0-15 clocks
0x6	16 clocks	0-31 clocks
0x7	31 clocks	0-31 clocks

Workaround(s): None.

I2C#06 *I2C Slave Alternate Address Disable does not Function as Expected*

Revision(s) Affected: 6 and 7.

Description: I^{2}C slave control bit OAR2EN can be used to enable or disable the alternate address for the I^{2}C slave device. The I^{2}C slave will acknowledge a disabled alternate address access when:

- The alternate address feature is enabled and subsequently disabled.

  AND
- An access is made to the alternate address prior to the primary address

Workaround(s): The I^{2}C master must access the I^{2}C slave through the primary address to disable the alternate address.

I2C#07 *I2C Master does not Clear ERROR Status bit*

Revision(s) Affected: 6 and 7.

Description: The ERROR status bit is an OR of ADRACK, DATACK and ARBLST status bit and the application is expected to check the ERROR status bit in I2CMCS register after every transaction command. During master write, if the slave address does not get acknowledged on the bus, ADRACK and DATACK bits are set. However, if the subsequent transaction is a master read to a valid slave, the DATACK bit is not cleared causing the ERROR status bit to remain set.

Workaround(s):

- The application code must perform a valid master write to clear the DATACK bit.

  OR
- The application code when performing a master read should only evaluate ARBLST and ADRACK when ERROR status bit is set.

MEM#02 *The START bit in the EEPROM Support Control and Status (EESUPP) Register Does not Function*

Revision(s) Affected: 6 and 7.

Description: Setting the START bit should begin error recovery if the PRETRY or ERETRY bit in the EESUPP register is set. However, setting this bit does not perform any function.

Workaround(s): Execute the EEPROMInit() function and then manually retry the failed operation.

MEM#03 *EEPROM Data May be Corrupted if an EEPROM Write is Interrupted*

Revision(s) Affected: 6 only.

Description: Corrupted EEPROM data can occur if an EEPROM write is interrupted with any of the following power events:

- Power failure
- External reset ($\overline{\text{RST}}$)
- Brown-out (BOR) event

When the WORKING bit of the EEDONE register is set, an EEPROM program or erase operation is occuring. The corrupted EEPROM data that can result from this sequence is not limited to the current word being written. If these events do not apply to your system, then normal EEPROM operation is expected. If a failure occurs, there will not be any indication of the failed erase or corrupted data (for example in the PRETRY and the ERETRY bits in the EEPROM Support Control and Status (EESUPP) register.

Workaround(s): Configure the external reset ($\overline{\text{RST}}$) and the watchdog reset to issue a system reset (EXTRES = 0x2 and $\overline{\text{WDOG}}$ = 0x2 in the RESBEHAVCTL register). Additionally, a BOR event should be configured to trigger an interrupt or issue a system reset (BOR = 0x2 in the RESBEHAVCTL register).

Depending on the system, there are a few potential workarounds:

1. Program the EEPROM only when the device is guaranteed to not have power removed and when a brown-out reset and an external reset will not occur. There are no restrictions on EEPROM reads.
2. Use the Flash memory with application software to store data instead of the EEPROM controller.
3. Limit the number of lifetime EEPROM writes to 7 writes per word.
4. Use an external EEPROM.

MEM#04 *Device may Become Non-functional if an EEPROM Write or Erase is Interrupted*

Revision(s) Affected: 6 only.

Description: The device may not function if power is removed or if an external reset ($\overline{\text{RST}}$) or a brown-out reset (BOR) occurs during an EEPROM write or erase operation. When the WORKING bit of the EEDONE register is set, an EEPROM program or erase operation is occurring. A reset will not recover the device.

If these events do not apply to your system, then normal EEPROM operation is expected.

Workaround(s): Depending on the system, there are a few potential workarounds to this issue:

- Program and erase the EEPROM only when the device is guaranteed to not have power removed and when a brown-out reset and an external reset will not occur. There are no restrictions on EEPROM reads.
- Use the Flash memory with application software to store data instead of the EEPROM controller.
- Limit the number of lifetime EEPROM writes to 7 writes per word.
- Use an external EEPROM.

MEM#05 *Device may Become Non-functional if Power is Interrupted During an Unlock of the Microcontroller or During Non-volatile Register Commits*

Revision(s) Affected: 6 and 7.

Description: The device may not function if power is removed or if an external reset ($\overline{\text{RST}}$) or a brown-out reset (BOR) occurs while executing the debug port unlock sequence or while committing the contents of a non-volatile register. The debug port unlock sequence is described in the Recovering a "Locked" Microcontroller section in the JTAG chapter of the data sheet. Non-volatile registers are described in the Non-Volatile Register Programming section in the Internal Memory chapter of the data sheet.

Workaround(s): None.

MEM#07 *Soft Resets Should not be Asserted During EEPROM Operations*

Revision(s) Affected: 7 only.

Description: EEPROM data may be corrupted if any of the following soft resets are asserted during an EEPROM program or erase operation:

- Software reset (SYSRESREQ)
- Software peripheral reset of the EEPROM module
- Watchdog reset
- MOSC failure reset

Workaround(s): Ensure that any of the above soft resets are not asserted during an EEPROM program or erase operation. The WORKING bit of the EEDONE register can be checked before the reset is asserted to see if an EEPROM program or erase operation is occurring. Soft resets may occur when using a debugger and should be avoided during an EEPROM operation. A reset such as the Watchdog reset can be mapped to an external reset using a GPIO or Hibernate can be entered, if time is not a concern.

MEM#08 *Writes and Erases to the EEPROM will not Work if the Three EEPROM Password Registers are Used for Last EEPROM Block*

Revision(s) Affected: 6 and 7.

Description: Writes and erases to the EEPROM controller data and registers will not work if all three EEPROM password registers are used to configure a password for the last EEPROM block.

Workaround(s): The password for the last EEPROM block should not exceed 64-bits.

MEM#10 *The START bit in the EESUPP Register may Cause EEPROM Corruption*

Revision(s) Affected: 7 only.

Description: Corrupted EEPROM data cannot be recovered and EEPROM data may become corrupted for the lifetime of the device if the START bit in the EEPROM Support Control and Status (EESUPP) register is set.

Workaround(s): Do not use the START bit for error recovery. If either the PRETRY or ERETRY bits are set, the EEPROM was unable to recover its state. If power is stable when either of these bits are set, this indicates a fatal error and is likely an indication that the EEPROM memory has exceeded its specified lifetime write or erase specification. If power is unstable when either of these bits are set retry the operation once the voltage is stabilized to clear the error.

To recover partially programmed or partially erased EEPROM data, one of the following resets must be performed followed by a call to the TivaWare API EEPROMInit() (TivaWare version 2.1 or later):

- Complete power-off and power-on of the device
- External reset (RSTn)
- Brown-out reset (BOR)

MEM#11 *The ROM Version of the TivaWare EEPROMInit API Does not Correctly Initialize the EEPROM*

Revision(s) Affected: 7 only.

Description: The ROM_EEPROMInit API in TivaWare does not correctly initialize the EEPROM module as described in the data sheet. It should not be used to initialize the EEPROM.

Workaround(s): Use the Flash version of the EEPROMInit API in TivaWare version 2.1 or later.

MEM#14 *Flash Write Operation During Execute from Flash may Result in Wrong Instruction Fetch*

Revision(s) Affected: 6 and 7.

Description: Executing code from FLASH while executing FLASH program/Erase when the system clock is configured for >40MHz operation may result in the wrong instruction fetch. If the CPU has not put the next address on the ICODE or DCODE bus at the start of Flash Operation, then the IDMEM would not return the corresponding data for the address on the ICODE or DCODE bus when the Flash operation is completed.

Workaround(s): Following are the two workarounds that can be used to allow Flash Program or Erase to be done while code execution is required. Both options require the *IntMasterDisable* be called to disable Interrupts before the flash operation-1.

- Option-1: Reduce the System Clock Frequency to 40MHz to disable the Prefetch Buffer during the Flash Operation which will allow the correct data to be returned to the CPU after the stall condition is completed.
- Copy the Flash Program/Erase code to SRAM and CPU executes the code from SRAM.

MEM#19 *Certain GPIOs Cannot be Configured as boot pins*

Revision(s) Affected: 6, and 7.

Description: Pins PC0-3, PD7 and PF0 cannot be selected as boot pins. These pins are locked on the TM4C123x devices, and the ROM does not unlock them before checking for polarity. As a result, the devices may not go into ROM boot loader or may remain in ROM boot loader on a POR reset.

Workaround(s): Use other GPIOs as boot pin. Refer to the BOOTCFG register description for details.

PWM#01 *Under Certain Circumstances, the PWM Load Interrupt is Triggered as Soon as the PWM is Enabled*

Revision(s) Affected: 6 and 7.

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.

PWM#02 *Setting the PWMSYNC Bits May Not Synchronize the PWM Counters if PWMDIV is Used*

Revision(s) Affected: 6 and 7.

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.

PWM#04 *PWM Generator Interrupts can only be Cleared 1 PWM Clock Cycle After the Interrupt Occurs*

Revision(s) Affected: 6 and 7.

Description: A write of 1 to the PWMxISC register is expected to clear the corresponding generator interrupt status on the next system clock. However, the write will clear the generator interrupt status on the next PWM clock. Any write to the PWMxISC to clear the interrupt before the next PWM clock will be ignored and the interrupt will be re-asserted.

Workaround(s): After the interrupt is asserted, the CPU must wait for one PWM clock cycle before writing 1 to the PWMxISC to clear the corresponding generator interrupt status. The larger the PWM clock divider value, the longer the system delay to clear the interrupt.

QEI#01 *When Using the Index Pulse to Reset the Counter, a Specific Initial Condition in the QEI Module Causes the Direction for the First Count to be Misread*

Revision(s) Affected: 6 and 7.

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.

SSI#06 *SSI Receive FIFO Time-out Interrupt may Assert Sooner than Expected in Slave Mode*

Revision(s) Affected: 6 and 7.

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.

SSI#07 *SSI Transmit Interrupt Status Bit is not Latched*

Revision(s) Affected: 6 and 7.

Description: SSI Transmit Interrupt Status Bit does not work correctly.

- Condition-1: Master Mode with interrupts when transmit FIFO is half full or less (SSICR1.EOT bit is clear). SSIMIS will be asserted every time the TXFIFO drops below half FIFO threshold causing the interrupt to be asserted all the time even if SSIICR register is used to clear the Interrupt condition
- Condition-2: Master mode with interrupts when transmit FIFO is empty. (SSICR1.EOT bit is set) SSITXRIS Is not latched when the transmit buffer is empty. SSIMIS and SSIRIS registers may be read as 0 by the CPU.

Workaround(s): SSI must be initialized with SSICR1.EOT bit clear (Condition-1). In the interrupt handler on completion of the transfer from the buffer to SSIDR clear the SSIIM.TXIM to stop any further interrupts. When the next set of interrupts are required for transmission then set the SSIIM.TXIM bit.

SYSCTL#01 *With a Specific Clock Configuration, Device may not Wake From Deep Sleep Mode*

Revision(s) Affected: 6 and 7.

Description: With the following specific clock configuration, the device fails to wake from Deep Sleep mode approximately 1 out of 1500 times. The configuration that may cause the issue is as follows:

- The PLL is using MOSC as the clock source, AND
- The PLL is the system clock source before going in to Deep Sleep mode, AND
- The Low-Frequency Internal Oscillator (LFIOSC) is the clock source during Deep Sleep

Workaround(s): Either:

- Use the PIOSC as the clock source for the PLL, OR
- Manually disable the PLL before entering Deep Sleep mode, OR
- Use the PIOSC as the clock source during Deep Sleep

SYSCTL#03 ***The MOSC Verification Circuit Does not Detect a Loss of Clock After the Clock has been Successfully Operating***

Revision(s) Affected: 6 and 7.

Description: If the MOSC clock source has been powered up and operating correctly and is subsequently removed or flatlines, the MOSC verification circuit does not indicate an error condition.

Workaround(s): Use Watchdog module 1, which runs off of PIOSC, to reset the system if the MOSC fails.

SYSCTL#04 *Device May not Wake Correctly From Sleep Mode Under Certain Circumstances*

Revision(s) Affected: 6 and 7.

Description: With a certain configuration, the device may not wake correctly from Sleep mode because invalid data may be fetched from the prefetch buffer. The configuration that causes this issue is as follows:

- The system clock must be at least 40 MHz
- Interrupts must be disabled

Workaround(s): Use following code instead of the ROM-based function *ROM_SysCtlSleep()* to put the device into Sleep mode:

```
__asm int
CPUwfi_safe(void) {
//
// Wait for the next interrupt.
//
wfi
mov r0,#0 // force bx lr to not start until after clocks back on
bx lr
}
```

SYSCTL#06 *Resets Fail While in Deep Sleep When Using Certain Clock Configurations*

Revision(s) Affected: 6 and 7.

Description: If a system reset occurs while in Deep Sleep mode when the MOSC is configured as the clock source for both Run mode and Deep Sleep mode and the PIOSC is configured to power down in Deep Sleep, the MOSC is immediately disabled. The system cannot be clocked because the PIOSC is configured to be off. A power-on reset (POR) is required to get the system out of this state.

Workaround(s): Use the PIOSC during Deep Sleep or use a system clock other than the MOSC.

SYSCTL#07 *Deep Sleep Clock Frequency Incorrect if a Watchdog Reset Occurs Upon Entry*

Revision(s) Affected: 6 and 7.

Description: If a watchdog reset occurs within 10 run-time clock cycles of entering Deep Sleep mode, the clocking configuration for Deep Sleep may be overlooked. If this occurs, the first time the device enters Deep Sleep after the reset, the Run mode parameters used for the system clock frequency are used instead.

The originally configured Deep Sleep clock configuration is reapplied after this first time entering Deep Sleep.

Workaround(s): If the Run mode clock frequency does not have a significant impact to the user application, no additional steps are necessary. If the Run mode clock frequency is undesirable for Deep Sleep mode, the watchdog module should be powered down in Run mode before entering Deep Sleep to ensure that a watchdog event does not occur during the entry into Deep Sleep.

SYSCTL#11 *Longer Reset Pulse Needed if Device is in Deep Sleep Mode With the LFIOSC as the Clock Source*

Revision(s) Affected:	6 and 7.
Description:	If the device is in Deep Sleep mode with the LFIOSC as the clock source, the specified reset pulse is not sufficient to reset the part in all cases.
Workaround(s):	Ensure that the reset pulse is at least 30 ms if the part may be in Deep Sleep mode with the LFIOSC as the clock source.

SYSCTL#14 *Power Consumption is Higher When MOSC is Used in Single-Ended Mode*

Revision(s) Affected: 6 and 7.

Description: The MOSC internal oscillator continues to run, even when a single-ended clock source is attached to OSC0. This issue does not affect proper operation but does result in additional power consumption of up to 3.5 mA.

Workaround(s): None

SYSCTL#16 *On-Chip LDO may not Start Properly During Power Up*

Revision(s) Affected: 6 and 7.

Description: In very rare cases, a non-monotonic voltage rise of VDDA between the minimum and maximum Power-On Reset Threshold (V_{POR}) voltage range, 2.0 V and 2.6 V, can cause the on-chip LDO to not start up. Because the LDO controls the core voltage (VDDC), the device cannot start up correctly in this situation. If the LDO fails to start, power cycle the device until a successful power up occurs. A software or hardware reset cannot restart the LDO.

Workaround(s): A monotonic voltage rise of VDDA prevents this issue from occurring; however, a perfect monotonic ramp is difficult to achieve, particularly during LDO inrush. The risk of encountering this issue can be minimized by performing one of the following:

- If the VDD and VDDA pins are connected directly to the same power source, at every possible point A and point B along the VDDA waveform between 2.0 V and 2.6 V, point B must never fall below point A after 15 µs, as shown in Figure 3.
- Use a separate power supply for VDDA to reduce noise and isolate it from the effects of LDO inrush. At every possible point A and point B along the VDDA waveform between 2.0 V and 2.6 V, point B must never fall below point A after 15 µs, as shown in Figure 3. The separate power supply will make it easier to avoid this condition.

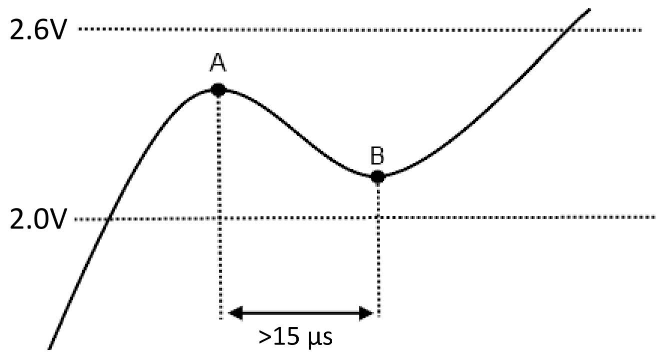


Figure 3. VDDA Waveform to Avoid Between V_{POR} min and V_{POR} max

SYSCTL#17 *DSDIVORIDE Value of 0x1 Does not Divide Deep Sleep Clock by 2*

Revision(s) Affected: 6 and 7.

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.

SYSCTL#20 *DID1 Register Incorrect*

Revision(s) Affected: 6 only.

Description: The DID1.QUAL bits shows the value 0x0 which indicates that the devices are not qualified.

Workaround(s): Nothing required.

SYSCTL#21 *Reset Cause may not be Logged in RESC Register*

Revision(s) Affected: 6 and 7.

Description: The system control register RESC logs the cause of the device reset. In some cases the RESC may not correctly reflect the source of reset.

Workaround(s): None.

UART#01 *When UART SIR Mode is Enabled, µDMA Burst Transfer Does not Occur*

Revision(s) Affected: 6 and 7.

Description: If the IrDA Serial Infrared (SIR) mode is enabled in the UART peripheral and the µDMA is mapped to either UARTn RX or UARTn TX and 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 channel mapped to the UART to respond to single or burst requests to ensure that the data transfer occurs.

USB#01 *USB Host Controller may not be Used to Communicate With a Low-Speed Device When Connected Through a hub*

Revision(s) Affected: 6 and 7.

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.

USB#02 *USB Controller Sends EOP at end of Device Remote Wake-Up*

Revision(s) Affected: 6 and 7.

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.

USB#04 *Device Sends SE0 in Response to a USB Bus Reset*

Revision(s) Affected: 6 and 7.

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.

USB#05 *USB Resume Occasionally does not Wake Device from Deep Sleep*

Revision(s) Affected: 6 and 7.

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, with or without using the PLL, and the system clock must be at least 20 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.

WDT#01 *Watchdog Timer 1 Module Cannot be Used Without Enabling Other Peripherals First*

Revision(s) Affected: 6 and 7.

Description: The Watchdog Timer 1 module is not fully enabled in Run, Sleep, or Deep-Sleep mode by just setting

- the R1 bit in the Watchdog Timer Run Mode Clock Gating Control (RCGCWD) register,
- the S1 bit in the Watchdog Timer Sleep Mode Clock Gating Control (SCGCWD) register,
- the D0 bit in the Watchdog Timer Deep Sleep Mode Clock Gating Control (DCGCWD) register,
- the WDT1 bit in the Run Mode Clock Gating Control Register 0 (RCGC0),
- the WDT1 bit in the Sleep Mode Clock Gating Control Register 0 (SCGC0), or
- the WDT1 bit in the Deep Sleep Mode Clock Gating Control Register 0 (DCGC0)

and, therefore, the module cannot be used unless a different peripheral is enabled first.

Workaround(s): Enable at least one of the following peripherals before enabling the Watchdog Timer 1 module—UARTn, SSIn, or ADCn—by setting the respective bit(s) appropriate system control registers.

WDT#02 *Watchdog Clear Mechanism Described in the Data Sheet Does not Work for the Watchdog Timer 1 Module*

Revision(s) Affected: 6 and 7.

Description: Periodically reloading the count value into the Watchdog Timer Load (WDTLOAD) register of the Watchdog Timer 1 module will not restart the count, as specified in the data sheet.

Workaround(s): Disable the Watchdog Timer 1 module by setting the appropriate bit in the Watchdog Timer Software Reset (SRWD) register before reprogramming the counter. Alternatively, clear the watchdog interrupt status periodically outside of the interrupt handler by writing any value to the Watchdog Interrupt Clear (WDTICR) register.

WDT#03 *Watchdog Timer 1 Module Asserts Reset Signal Even if not Programmed to Reset*

Revision(s) Affected: 6 and 7.

Description: Even if the reset signal is not enabled (the RESEN bit of the Watchdog Control (WDTCTL) register is clear), the Watchdog Timer 1 module will assert a reset signal to the system when the time-out value is reached for a second time.

Workaround(s): Clear the Watchdog Timer 1 interrupt once the time-out value is reached for the first time by writing any value to the Watchdog Interrupt Clear (WDTICR) register.

WDT#05 *WDTLOAD Yields an Incorrect Value When Read Back*

Revision(s) Affected: 6 and 7.

Description: If the Watchdog Timer 1 module is enabled and configured to run off the PIOSC, writes to the Watchdog Load (WDTLOAD) register yield an incorrect value when read back.

Workaround(s): None.

WDT#06 *WDTMIS Register Does not Indicate an NMI Interrupt From WDT0*

Revision(s) Affected: 6 and 7.

Description: The WDTMIS bit of the Watchdog Masked Interrupt Status (WDTMIS) register does not get set if a watchdog time-out non-maskable interrupt (NMI) interrupt from Watchdog Timer Module 0 has been signaled to the interrupt controller. This does not impact operation of the NMI interrupt. The NMI interrupt is still sent to the interrupt controller when a WDT timeout occurs.

Workaround(s): None.

WDT#07 *The Watchdog Load (WDTLOAD) Register Cannot be Changed When Using a Debugger While the STALL bit is set*

Revision(s) Affected: 6 and 7.

Description: The Watchdog Load (WDTLOAD) register cannot be changed when using a debugger with the STALL bit in the Watchdog Test (WDTTEST) register set.

Workaround(s): Avoid changing the Watchdog Load (WDTLOAD) register with the debugger connected when the STALL bit is set.

WDT#08 *Reading the WDTVALUE Register may Return Incorrect Values When Using Watchdog Timer 1*

Revision(s) Affected:	6 and 7.
Description	Incorrect values may be read from the Watchdog Value (WDTVALUE) register at the Watchdog Timer 1 base address when using Watchdog Timer 1.
Workaround(s)	None.

Revision History

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

Changes from March 31, 2015 to April 30, 2016	Page
• Updated/Changed GPIO#10 Description and Workaround	28
• Added new advisory: I2C#06, *I2C Slave Alternate Address Disable does not Function as Expected*	42
• Added new advisory: I2C#07, *I2C Master does not Clear ERROR Status bit*	43
• Added new advisory: MEM#19, *Certain GPIOs Cannot be Configured as boot pins*	53
• Added new advisory: PWM#04, *PWM Generator Interrupts can only be Cleared 1 PWM Clock Cycle After the Interrupt Occurs*	56
• Added new advisory: SYSCTL#21, *Reset Cause may not be Logged in RESC Register*	70

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		