Tiva™ C Series TM4C123GH6PGE Errata

This document contains known errata at the time of publication for the TM4C123GH6PGE microcontroller. The table below summarizes the errata and lists the affected revisions. See the data sheet for more details.

See also the ARM® Cortex™-M4F errata, ARM publication number PRD40-PRDC-013029.

Table 1. List of Errata

Erratum Number	Erratum Title	Module Affected
1.1	With a specific clock configuration, device may not wake from Deep-sleep mode	System Control
1.2	The MOSC verification circuit does not detect a loss of clock after the clock has been successfully operating	System Control
1.3	Device may not wake correctly from Sleep mode under certain circumstances	System Control
1.4	Resets fail while in Deep-sleep when using certain clock configurations	System Control
1.5	Deep-sleep clock frequency incorrect if a watchdog reset occurs upon entry	System Control
1.6	Longer reset pulse needed if device is in Deep-Sleep mode with the LFIOSC as the clock source	System Control
2.1	Some Hibernation module registers may not have the correct value in two situations	Hibernation
2.2	Reading the HIBRTCC and HIBRTCSS registers may provide incorrect values	Hibernation
2.3	Device fails to wake from hibernation within a certain time after hibernation is requested	Hibernation
2.4	RTC match event is missed if it occurs in a certain window	Hibernation
3.1	The START bit in the EEPROM Support Control and Status (EESUPP) register does not function	EEPROM
4.1	In three cases, two peripherals cannot both be programmed to use µDMA	µDMA
5.1	JTAG controller does not ignore transitions on PC0/TCK when it is configured as a GPIO	GPIO
6.1	GPTMSYNC bits require manual clearing	General-Purpose Timers
6.2	The GPTMPP register does not correctly indicate 32/64-bit timer capability	General-Purpose Timers
6.3	Wait-for-Trigger mode is not available for PWM mode	General-Purpose Timers
6.4	Writes to some General-Purpose Timers registers cause the counter to increment or decrement	General-Purpose Timers
6.5	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	General-Purpose Timers
7.1	Watchdog Timer 1 module cannot be used without enabling other peripherals first	Watchdog Timers

Erratum Number	Erratum Title	Module Affected
7.2	Watchdog clear mechanism described in the data sheet does not work for the Watchdog Timer 1 module	Watchdog Timers
7.3	Watchdog Timer 1 module asserts reset signal even if not programmed to reset	Watchdog Timers
7.4	WDTLOAD yields an incorrect value when read back	Watchdog Timers
7.5	WDTMIS register does not indicate an NMI interrupt from WDT0	Watchdog Timers
7.6	The Watchdog Load (WDTLOAD) register cannot be changed when using a debugger while the STALL bit is set	Watchdog Timers
8.1	Retriggering a sample sequencer before it has completed the current sequence results in continuous sampling	ADC
8.2	Digital comparator in last step of sequence does not trigger or interrupt	ADC
8.3	Digital comparator interrupts do not trigger or interrupt as expected	ADC
8.4	ADC sample sequencers priorities are different than expected	ADC
8.5	ADC sample sequencer only samples when using certain clock configurations	ADC
8.6	First two ADC samples from the internal temperature sensor must be ignored	ADC
9.1	When UART SIR mode is enabled, µDMA burst transfer does not occur	UART
10.1	I^2C glitch filter suppression width may differ from the configured value	I2C
11.1	USB Host controller may not be used to communicate with a low-speed Device when connected through a hub	USB
11.2	USB controller sends EOP at end of device remote wake-up	USB
12.1	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	QEI

1 System Control

1.1 With a specific clock configuration, device may not wake from Deep-sleep mode

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:

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

1.2 The MOSC verification circuit does not detect a loss of clock after the clock has been successfully operating

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:

Use Watchdog module 1, which runs off of PIOSC, to reset the system if the MOSC fails.

1.3 Device may not wake correctly from Sleep mode under certain circumstances

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:

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
}
```

1.4 Resets fail while in Deep-sleep when using certain clock configurations

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:

Use the PIOSC during Deep-sleep or use a system clock other than the MOSC.

1.5 Deep-sleep clock frequency incorrect if a watchdog reset occurs upon entry

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:

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.

1.6 Longer reset pulse needed if device is in Deep-Sleep mode with the LFIOSC as the clock source

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:

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.

2 Hibernation

2.1 Some Hibernation module registers may not have the correct value in two situations

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:

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

2.2 Reading the HIBRTCC and HIBRTCSS registers may provide incorrect values

Description:

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

Workaround:

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 != HIBREG(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();
```

2.3 Device fails to wake from hibernation within a certain time after hibernation is requested

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 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:

Add a TivaWare™ for C Series `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)
{
}
```

2.4 RTC match event is missed if it occurs in a certain window

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:

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.

3 EEPROM

3.1 The START bit in the EEPROM Support Control and Status (EESUPP) register does not function

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:

Execute the `EEPROMInit()` function and then manually retry the failed operation.

4 µDMA

4.1 In three cases, two peripherals cannot both be programmed to use µDMA

Description:

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

- SSI0 and SSI1
- UART2 and USBEP1
- UART0 and UART2

Workaround:

Configure peripherals such that the combinations of peripherals listed above are not both using µDMA.

5 GPIO

5.1 JTAG controller does not ignore transitions on PC0/TCK when it is configured as a GPIO

Description:

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

Workaround:

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.

6 General-Purpose Timers

6.1 GPTMSYNC bits require manual clearing

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:

When bits in the **GPTMSYNC** register are set, software must clear the bits prior to setting them for a subsequent update. When using TivaWare™ for C Series 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);
```

6.2 The GPTMPP register does not correctly indicate 32/64-bit timer capability

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:

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.

6.3 Wait-for-Trigger mode is not available for PWM mode

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:

None.

6.4 Writes to some General-Purpose Timers registers cause the counter to increment or decrement

Description:

Writes to the following registers cause the counter to increment in up count mode and decrement in down count mode:

- **GPTM Timer n Interval Load (GPTMTnILR)**
- **GPTM Timer n Match (GPTMTnMATCHR)**
- **GPTM Timer n Prescale (GPTMTnPR)**
- **GPTM Timer n Value (GPTMTnV)**

Workaround:

None.

6.5 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

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:

Do not load 0xFFFF into the **GPTMTnILR** register when counting up in Input Edge-Time mode.

7 Watchdog Timers

7.1 Watchdog Timer 1 module cannot be used without enabling other peripherals first

Description:

The Watchdog Timer 1 module is not fully enabled by setting the `WDT1` bit in the **Run Mode Clock Gating Control Register 0 (RCGC0n)** register and, therefore, the module cannot be used unless a different peripheral is enabled first.

Workaround:

Enable at least one of the following peripherals before enabling the Watchdog Timer 1 module—UARTn, SSIn, or ADC—by setting the respective bit(s) in the **RCGUART**, **RCGCSSI**, or **RCGCADC** registers.

7.2 Watchdog clear mechanism described in the data sheet does not work for the Watchdog Timer 1 module

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:

Disable the Watchdog Timer 1 module 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.

7.3 Watchdog Timer 1 module asserts reset signal even if not programmed to reset

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:

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.

7.4 WDTLOAD yields an incorrect value when read back

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:

None.

7.5 WDTMIS register does not indicate an NMI interrupt from WDT0

Description:

The WDTMIS bit of the **Watchdog Masked Interrupt Status (WDTMIS)** register does not get set if a watchdog time-out NMI interrupt from Watchdog Timer Module 0 has been signaled to the interrupt controller. A watchdog interrupt can be programmed to be a non-maskable interrupt (NMI) by setting the INTTYPE bit in the **Watchdog Control (WDTCTL)** register.

Workaround:

None.

7.6 The Watchdog Load (WDTLOAD) register cannot be changed when using a debugger while the STALL bit is set

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:

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

8 ADC

8.1 Retriggering a sample sequencer before it has completed the current sequence results in continuous sampling

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:

Ensure that a sample sequence has completed before triggering a new sequence using the same type of trigger.

8.2 Digital comparator in last step of sequence does not trigger or interrupt

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_COMPn, TRIGGER_EXTERNAL, TRIGGER_TIMER, or TRIGGER_PWMn, 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:

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.

8.3 Digital comparator interrupts do not trigger or interrupt as expected

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:

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.

8.4 ADC sample sequencers priorities are different than expected

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:

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.

8.5 ADC sample sequencer only samples when using certain clock configurations

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:

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.

8.6 First two ADC samples from the internal temperature sensor must be ignored

Description:

The analog source resistance (Rs) 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:

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.

9 UART

9.1 When UART SIR mode is enabled, µDMA burst transfer does not occur

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:

Clear the 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.

10 I2C

10.1 I²C glitch filter suppression width may differ from the configured value

Description:

Due to an error in the initialization of the glitch filter, the actual pulse width may differ from what is expected. This issue rarely occurs. The following table outlines the different pulse widths that may occur with respect to the value written to the GFPW bit field.

Table 2. 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:

None.

11 USB

11.1 USB Host controller may not be used to communicate with a low-speed Device when connected through a hub

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:

None.

11.2 USB controller sends EOP at end of device remote wake-up

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 witnessed. This does not affect USB certification.

Workaround:

None.

12 QEI

12.1 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

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:

None.

Texas Instruments Incorporated
108 Wild Basin, Suite 350
Austin, TX 78746
http://www.ti.com/tiva-c
http://www-k.ext.ti.com/sc/technical-support/customer-support-centers.htm