Stellaris® LM4F120H5QR RevA1/A3 Errata

This document contains known errata at the time of publication for the Stellaris LM4F120H5QR 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. Revision History

Date	Revision	Description
October 2012	2.0	■ Added issue “Non-word-aligned write to SRAM can cause incorrect value to be loaded” on page 11. ■ Added issue “Resets fail while in Deep-sleep when using certain clock configurations” on page 12. ■ Added issue “Deep-sleep clock configuration incorrect if certain resets occur upon entry” on page 12. ■ Added issue “Clearing the BORMIS or the LDORDMIS interrupt status bits require an extra write” on page 13. ■ Added issue “ADC sample sequencers priorities are different than expected” on page 29. ■ Added issue “ADC sample sequencer only samples when using certain clock configurations” on page 29. ■ Added issue “UART transfers fail at certain system clock frequency and baud rate combinations” on page 30. ■ Added issue “Clearing the RXRIS bit when configured for LIN mode causes the UART to not transfer data” on page 31.

Date	Revision	Description
July 2012	1.9	■ Removed issue "Device does not wake from Deep-sleep mode if the hibernation oscillator is the clock source and a reset occurs". ■ Added issue “Wait-for-Trigger mode is not available for PWM mode” on page 24. ■ Added issue “Watchdog Timer 1 module cannot be used without enabling other peripherals first” on page 25. ■ Added issue “Watchdog clear mechanism described in the data sheet does not work for the Watchdog Timer 1 module” on page 25. ■ Added issue “Watchdog Timer 1 module asserts reset signal even if not programmed to reset” on page 25. ■ Added issue “WDTLOAD yields an incorrect value when read back” on page 26. ■ Added issue “WDTMIS register does not indicate an NMI interrupt from WDT0” on page 26. ■ Removed issue "A specific sequence is required when the MOSC is used to clock the ADC module". ■ Added issue “Digital comparator in last step of sequence does not trigger or interrupt” on page 27. ■ Added issue “Digital comparator interrupts do not trigger or interrupt as expected” on page 28. ■ Added issue “Missing trigger or interrupt when multiple sequences configured for processor trigger and different trigger” on page 28. ■ Added issue “When UART LIN or SIR mode enabled, µDMA burst transfer will not occur” on page 30. ■ Added issue “Freescale SPI Mode at low SSIClk frequencies can yield data corruption” on page 31. ■ Added issue “First two ADC samples from the internal temperature sensor must be ignored” on page 32.
May 2012	1.8	■ Added issue “Internal reset supervisors may not prevent incorrect device operation during power transitions” on page 10 which replaces "MCU executes code after BOR before proper power is restored" and "The POR and BOR threshold may vary from the specification". ■ Added issue “Hibernation write corruption on arbitrary power loss” on page 16.
March 2012	1.7	■ Added issue "Device does not wake from Deep-sleep mode if the hibernation oscillator is the clock source and a reset occurs". ■ Added issue “The START bit in the EEPROM Support Control and Status (EESUPP) register does not function” on page 21. ■ Added additional information regarding PD7 and PF0 to issue “Some GPIO register bits default to the incorrect state” on page 21. ■ Added issue “GPIO Port B1 has a leakage path to ground when VDD is removed” on page 23. ■ Added issue "A specific sequence is required when the MOSC is used to clock the ADC module".
December 2011	1.6	■ Noted that issue “DID0 register shows revision A0 for revision A1 devices” on page 7 is fixed on revision A3. ■ Noted that issue “Device may not operate correctly at certain frequencies” on page 8 is fixed on revision A3. ■ Added issue “The MOSC verification circuit does not detect a loss of clock after the clock has been successfully operating” on page 9. ■ Added issue “Device may not wake correctly from Sleep mode under certain circumstances” on page 9. ■ Added issue “RTC match event is missed if it occurs in a certain window” on page 16.

Date	Revision	Description
November 2011	1.5	■ Clarified issue “Boundary scan does not function correctly” on page 6. ■ Clarified issue "MCU executes code after BOR before proper power is restored". ■ Clarified issue "The POR and BOR threshold may vary from the specification". ■ Clarified issue “Device may not operate correctly at certain frequencies” on page 8. ■ Added issue “With a specific clock configuration, device may not wake from Deep-sleep mode” on page 8. ■ Clarified issue “Some Hibernation module registers may not have the correct value in two situations” on page 13. ■ Clarified issue “USB boot loader in ROM does not operate correctly” on page 17. ■ Added issue “Reading the HIBRTCC and HIBRTCSS registers may provide incorrect values” on page 14. ■ Added issue “JTAG controller does not ignore transitions on PC0/TCK when it is configured as a GPIO” on page 23.
November 2011	1.4	■ Updated issue “DID0 register shows revision A0 for revision A1 devices” on page 7. ■ Added issue “Precision Internal Oscillator (PIOSC) is untrimmed on devices with date codes prior to January 2012” on page 7. ■ Added issue “Device may not operate correctly at certain frequencies” on page 8. ■ Added issue “GPTMSYNC bits require manual clearing” on page 23. ■ Added issue “The GPTMPP register does not correctly indicate 32/64-bit timer capability” on page 24.
September 2011	1.3	■ Added issue “Boundary scan does not function correctly” on page 6. ■ Added issue "MCU executes code after BOR before proper power is restored". ■ Added issue "The POR and BOR threshold may vary from the specification". ■ Added issue “Flash memory page 0 and 1 may be erased if reset occurs during Flash memory erase operation” on page 18. ■ Added issue “EEPROM blocks must be accessed in alternate pairs to avoid corruption of data” on page 20. ■ Added issue “EEPROM blocks 0 through 3 may be erased if reset occurs during an EEPROM write” on page 20. ■ Added issue “Reset during Flash memory program or erase or an EEPROM write causes Suspend state” on page 18. ■ Added issue “PB1 has permanent internal pull-up resistance” on page 22. ■ Added issue “Retriggering a sample sequencer before it has completed the current sequence results in continuous sampling” on page 27. ■ Added additional information to issue “Higher current than expected is consumed while V_{DD} ramps up until V_{DDC} crosses 1 V” on page 31. ■ Added issue “Nominal current consumption is 650 μA higher than specified” on page 32. ■ Added issue “V_{DD} inrush current of up to 500 mA is seen while V_{DD} ramps up” on page 32.

Date	Revision	Description
June 2011	1.2	■ Added issue “DID0 register shows revision A0 for revision A1 devices” on page 7. ■ Clarified issue “Some Hibernation module registers may not have the correct value in two situations” on page 13. ■ Added issue “USB boot loader in ROM does not operate correctly” on page 17. ■ Added issue “ROM_SysCtlClockSet() does not operate correctly with fractional dividers” on page 17. ■ Added issue “When a 1-kB Flash page is erased, the adjacent page is also erased” on page 17. ■ Added issue “Higher current than expected is consumed while V_{DD} ramps up until V_{DDC} crosses 1 V” on page 31.
May 2011	1.1	■ Added issue “Some GPIO register bits default to the incorrect state” on page 21.
March 2011	1.0	Started tracking revision history.

Table 2. List of Errata

Erratum Number	Erratum Title	Module Affected	Revision(s) Affected
1.1	Boundary scan does not function correctly	JTAG	A1
2.1	DID0 register shows revision A0 for revision A1 devices	System Control	A1
2.2	Precision Internal Oscillator (PIOSC) is untrimmed on devices with date codes prior to January 2012	System Control	A1
2.3	Device may not operate correctly at certain frequencies	System Control	A1
2.4	With a specific clock configuration, device may not wake from Deep-sleep mode	System Control	A1, A3, B0
2.5	The MOSC verification circuit does not detect a loss of clock after the clock has been successfully operating	System Control	A1, A3, B0
2.6	Device may not wake correctly from Sleep mode under certain circumstances	System Control	A1, A3, B0
2.7	Internal reset supervisors may not prevent incorrect device operation during power transitions	System Control	A1, A3
2.8	Non-word-aligned write to SRAM can cause incorrect value to be loaded	System Control	A1, A3
2.9	Resets fail while in Deep-sleep when using certain clock configurations	System Control	A1, A3
2.10	Deep-sleep clock configuration incorrect if certain resets occur upon entry	System Control	A1, A3
2.11	Clearing the BORMIS or the LDORDMIS interrupt status bits require an extra write	System Control	A1, A3, B0
3.1	Some Hibernation module registers may not have the correct value in two situations	Hibernation	A1, A3, B0
3.2	Reading the HIBRTCC and HIBRTCSS registers may provide incorrect values	Hibernation	A1, A3, B0
3.3	Device fails to wake from hibernation within a certain time after hibernation is requested	Hibernation	A1, A3, B0

Erratum Number	Erratum Title	Module Affected	Revision(s) Affected
3.4	RTC match event is missed if it occurs in a certain window	Hibernation	A1, A3, B0
3.5	Hibernation write corruption on arbitrary power loss	Hibernation	A1, A3
4.1	USB boot loader in ROM does not operate correctly	ROM	A1
4.2	ROM_SysCtlClockSet() does not operate correctly with fractional dividers	ROM	A1
5.1	When a 1-kB Flash page is erased, the adjacent page is also erased	Flash memory	A1
5.2	Flash memory page 0 and 1 may be erased if reset occurs during Flash memory erase operation	Flash memory	A1
6.1	Reset during Flash memory program or erase or an EEPROM write causes Suspend state	Flash memory, EEPROM	A1
7.1	EEPROM blocks must be accessed in alternate pairs to avoid corruption of data	EEPROM	A1
7.2	EEPROM blocks 0 through 3 may be erased if reset occurs during an EEPROM write	EEPROM	A1
7.3	The START bit in the EEPROM Support Control and Status (EESUPP) register does not function	EEPROM	A1, A3, B0
8.1	Some GPIO register bits default to the incorrect state	GPIO	A1
8.2	PB1 has permanent internal pull-up resistance	GPIO	A1
8.3	JTAG controller does not ignore transitions on PC0/TCK when it is configured as a GPIO	GPIO	A1, A3, B0
8.4	GPIO Port B1 has a leakage path to ground when VDD is removed	GPIO	A1, A3, B0
9.1	GPTMSYNC bits require manual clearing	General-Purpose Timers	A1, A3, B0
9.2	The GPTMPP register does not correctly indicate 32/64-bit timer capability	General-Purpose Timers	A1, A3, B0
9.3	Wait-for-Trigger mode is not available for PWM mode	General-Purpose Timers	A1, A3, B0
10.1	Watchdog Timer 1 module cannot be used without enabling other peripherals first	Watchdog Timers	A1, A3, B0
10.2	Watchdog clear mechanism described in the data sheet does not work for the Watchdog Timer 1 module	Watchdog Timers	A1, A3, B0
10.3	Watchdog Timer 1 module asserts reset signal even if not programmed to reset	Watchdog Timers	A1, A3, B0
10.4	WDTLOAD yields an incorrect value when read back	Watchdog Timers	A1, A3, B0
10.5	WDTMIS register does not indicate an NMI interrupt from WDT0	Watchdog Timers	A1, A3, B0
11.1	Retriggering a sample sequencer before it has completed the current sequence results in continuous sampling	ADC	A1, A3, B0
11.2	The first ADC sample when using differential mode is incorrect	ADC	A1, A3
11.3	Digital comparator in last step of sequence does not trigger or interrupt	ADC	A1, A3, B0
11.4	Digital comparator interrupts do not trigger or interrupt as expected	ADC	A1, A3, B0

Erratum Number	Erratum Title	Module Affected	Revision(s) Affected
11.5	Missing trigger or interrupt when multiple sequences configured for processor trigger and different trigger	ADC	A1, A3, B0
11.6	ADC sample sequencers priorities are different than expected	ADC	A1, A3, B0
11.7	ADC sample sequencer only samples when using certain clock configurations	ADC	A1, A3, B0
12.1	When UART LIN or SIR mode enabled, μDMA burst transfer will not occur	UART	A1, A3, B0
12.2	UART transfers fail at certain system clock frequency and baud rate combinations	UART	A1, A3, B0
12.3	Clearing the RXRIS bit when configured for LIN mode causes the UART to not transfer data	UART	A1, A3, B0
13.1	Freescale SPI Mode at low SSIClk frequencies can yield data corruption	SSI	A1, A3, B0
14.1	Higher current than expected is consumed while V_{DD} ramps up until V_{DDC} crosses 1 V	Electrical Characteristics	A1
14.2	Nominal current consumption is 650 μA higher than specified	Electrical Characteristics	A1
14.3	V_{DD} inrush current of up to 500 mA is seen while V_{DD} ramps up	Electrical Characteristics	A1
14.4	First two ADC samples from the internal temperature sensor must be ignored	Electrical Characteristics	A1, A3, B0

1 JTAG

1.1 Boundary scan does not function correctly

Description:

Boundary scan does not function correctly and should not be used. This issue does not affect the use of JTAG for programming Flash memory or debug.

Workaround:

None.

Silicon Revision Affected:

A1

Fixed:

Fixed on A3.

2 System Control

2.1 DID0 register shows revision A0 for revision A1 devices

Description:

The **Device Identification 0 (DID0)** register shows the revision of the device. The register should read 0x1005.0001 for A1, but instead it reads 0x1005.0000.

Workaround:

Read the ROM revision at address 0x0100.0010. This value is 0x1a9 on A1 silicon.

Silicon Revision Affected:

A1

Fixed:

Fixed on A3.

2.2 Precision Internal Oscillator (PIOSC) is untrimmed on devices with date codes prior to January 2012

Description:

The Precision Internal Oscillator (PIOSC) is untrimmed on some devices during factory test prior to shipment. The PIOSC on untrimmed devices has an error of up to ±10 %. Normally the PIOSC is trimmed to 16 MHz ± 1% at room temperature and 16 MHz ±3% across the operating temperature range.

In addition, the USB bootloader cannot operate if the PIOSC is not calibrated.

Workaround:

The PIOSC can be trimmed by the user in one of two ways: automatically with the Hibernation module, and manually with a user-defined calibration value based on another clock source.

By using the Hibernation module with a functioning 32.768-kHz clock source, the PIOSC can be automatically calibrated using the following method:

1. Set the `CAL` bit in the **Precision Internal Oscillator Calibration (PIOSCCAL)** register; the results of the calibration are shown in the `RESULT` field in the **Precision Internal Oscillator Statistic (PIOSCSTAT)** register.
2. After calibration is complete, the PIOSC is trimmed using the trimmed value returned in the `CT` field.

If the Hibernation module is not used in the system, the user must program a user-defined calibration value. The user can program the `UT` value in the `PIOSCCAL` register to adjust the PIOSC frequency. As the `UT` value increases, the generated period increases. To commit a new `UT` value, first set the `UTEN` bit, then program the `UT` field, and then set the `UPDATE` bit. The adjustment finishes within a few clock periods and is glitch free. For more information, see the section entitled, "Precision Internal Oscillator Operation (PIOSC)" in the System Control chapter in the data sheet.

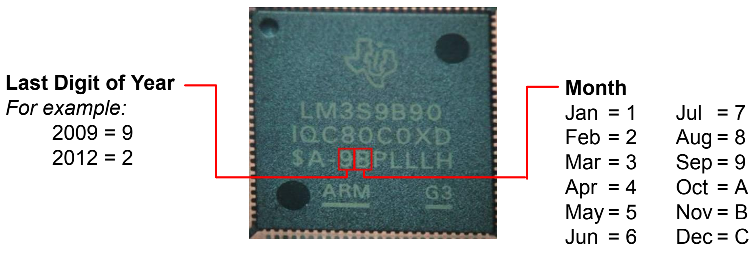
Silicon Revision Affected:

A1

Fixed:

Fixed on devices with date codes of 0x21 (January, 2012) or later. Fixed on all A3 devices.

Note: To determine the date code of your part, look at the first two characters following the dash on the third line of the part markings (highlighted in red in the following figure). The first number after the dash indicates the last decimal digit of the year. The second character indicates the month. Therefore, the following example shows a date code of 9B which indicates November 2009.



2.3 Device may not operate correctly at certain frequencies

Description:

When operating at system clock (SysClk) frequencies such that [35 MHz ≤ SysClk ≤ 45 MHz] or [70 MHz ≤ SysClk ≤ 80 MHz], an error in the digital control logic may result in inverted data causing incorrect program execution.

Workaround:

- When using the PLL, regardless of the clock source to the PLL, do not use SYSDIV values of 2.5, 4.5, 5, or 5.5.
- When not using the PLL and clocking from an external oscillator connected to MOSC, ensure that the system clock is below 35 MHz.

Note that this issue is not a concern when using the PIOSC or an external crystal of any allowed frequency connected to MOSC as the system clock, with the PLL bypassed.

Silicon Revision Affected:

A1

Fixed:

Fixed on A3.

2.4 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 30-kHz IOSC 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

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

2.5 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.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

2.6 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 functions `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
}
```

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

2.7 Internal reset supervisors may not prevent incorrect device operation during power transitions

Description:

The microcontroller incorporates internal Power-On Reset (POR) and Brown-Out Reset (BOR) supervisors to ensure that code only executes when power to the device is within specification. However, gaps in the voltage and timing thresholds of the internal supervisors result in a risk of incorrect operation during V_{DD} power transitions. This also results in a change of the minimum operating voltage to 3.05 V.

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 part may be left in a state where it will not operate correctly until a clean power cycle has been completed.

The Power-On Reset gap occurs because the supervisor can release internal state machine operation as soon as 118.5 µs after the V_{DD} supply reaches 2.70 V. If V_{DD} is still below the minimum operating voltage of 3.05 V after 118.5 µs, the power-up state machine may not function correctly resulting in the effects described above. 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 Brown-Out Reset threshold (V_{BTH}) gap occurs because the brown-out supervisor has a threshold as low as 2.85 V, which is less than the minimum operating voltage on V_{DD}, and also because it can take several microseconds to respond. BOR gaps can be encountered after power up, during steady state operation power-on, if the V_{DD} rail has glitches, and also during power-down.

Workaround:

This issue will be resolved in B0 silicon. If designing for B0 silicon, design for V_{DD} (Min) specification of 3.15 V.

Silicon Revision Affected:

A1, A3

Fixed:

Fixed on B0.

2.8 Non-word-aligned write to SRAM can cause incorrect value to be loaded

Description:

If a word-aligned value is loaded from an SRAM location into a core register, then altered by storing a byte or halfword at an unaligned offset, the altered word-aligned value is not correctly indicated when loaded into a core register. The loaded value from the SRAM location into a core register reflects the original value, not the modified value.

The following assembly sequence causes the altered value loaded into a core register to not load the correct value, even though the correct value is visible in the SRAM memory location.

```
//
// Load a word-aligned value from an SRAM location into a
// core register (such as R0)
//
LDR        R0, [SP, #+0];

//
// Store byte or halfword from the core register to
// the SRAM location at a non-word-aligned offset
//
STRB     R0, [SP, #+1];
      OR
STRB     R0, [SP, #+2];
      OR
STRB     R0, [SP, #+3];
      OR
STRW     R0, [SP, #+1];

//
// Load the same word-aligned value of the same SRAM location
// into a core register (such as R0)
//
LDR       R0, [SP, #+0];
```

This assembly sequence causes erroneous values only if these three instructions are executed in this order. However, the three instructions do not have to be consecutive, which means that other instructions can be placed in between the first and the second instructions, or the second and the third instructions, and the false value still occurs. Other instructions include, but are not limited to, branches in Flash, accesses to non-SRAM locations such as peripherals, and writes to other SRAM locations.

Pointers, structures, and unions are common C code methods that can be found in user code that may generate this assembly sequence and, therefore, result in incorrect values for variables. If using interrupts, it is possible to continue the assembly sequence in the interrupt handler, which could also return incorrect data.

For more information about this erratum as well as C code examples that may generate this assembly sequence, refer to the document, *Non-Word-Aligned Write to SRAM Additional Information*.

Workaround:

The type of compiler and optimization settings used in your application affects whether the problematic assembly code is generated from your user code. Each compiler behaves a little differently with respect to this erratum. The behavior for each compiler is not guaranteed due to the large number of compiler and tool version combinations.

At the assembly level, loading a volatile 32-bit-aligned word value from a different address in SRAM after storing and before loading in the assembly instruction sequence yields a correct value. A dummy SRAM load of a volatile 32-bit-aligned word from a different SRAM memory location should be inserted after the second assembly instruction (storing a byte or halfword from the core register to the desired SRAM location at a non-word-aligned offset) and before the third assembly instruction (loading the same word-aligned value of the desired SRAM location into a core register). This also means that a dummy SRAM load of a volatile 32-bit-aligned word from a different SRAM memory location should also be placed at the beginning of any interrupt routine, in case the third assembly instruction is executed before leaving the handler.

For more information about this erratum as well as C code examples that may generate this assembly sequence, refer to the document, *Non-Word-Aligned Write to SRAM Additional Information*.

Silicon Revision Affected:

A1, A3

Fixed:

Fixed on B0.

2.9 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.

Silicon Revision Affected:

A1, A3

Fixed:

Fixed on B0.

2.10 Deep-sleep clock configuration incorrect if certain resets occur upon entry

Description:

If an external reset, a brown-out reset, or a watchdog reset occurs when entering Deep-sleep mode with a system clock of any frequency, the clocking configuration for Deep-sleep may be overlooked.

If one of these resets occurs within 10 run-time clock cycles of entering Deep-sleep mode, the first time the device enters Deep-sleep after the reset, the Run mode parameters used for the system clock frequency is used instead of the originally configured Deep-sleep parameters. If the PIOSC was on in Run mode and configured to power-down in Deep-sleep, this is ignored and stays on. The **DCGC** register is used for the peripheral clock enables.

The originally configured Deep-sleep configurations is reapplied after this first time entering Deep-sleep.

Workaround:

If the Run mode configurations do not have a significant impact to the user application, no additional steps are necessary.

If the Run mode configurations are undesirable for Deep-sleep mode, an external reset, a brown-out reset, or a watchdog reset that occurs when entering Deep-sleep should be followed by entering then exiting Deep-sleep mode. This allows the next entry to Deep-sleep to use the originally configured Deep-sleep clocking conditions.

Silicon Revision Affected:

A1, A3

Fixed:

Fixed on B0 for external and brown-out resets.

2.11 Clearing the BORMIS or the LDORDMIS interrupt status bits require an extra write

Description:

Writing a 1 to the `BORMIS` bit or the `LDORDMIS` bit in the **Masked Interrupt Status and Clear (MISC)** register does not immediately clear the `BORRIS` or the `LDORDRIS` raw interrupt bit in the **Raw Interrupt Status (RIS)** register.

Workaround:

Write a 1 twice to the `BORMIS` bit or the `LDORDMIS` bit to successfully clear the `BORRIS` or the `LDORDRIS` raw interrupt bit.

Silicon Revision Affected:

A1, A3, B0

Fixed:

`BORMIS` fixed in B0; `LDORDMIS` not yet fixed.

3 Hibernation

3.1 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.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

3.2 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();
```

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

3.3 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 hibernation 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:

Add a StellarisWare `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)
```

```
{
}
```

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

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

Description:

A 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 3 counts of the RTC sub seconds counter, hold off entering into hibernation until the match has occurred.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

3.5 Hibernation write corruption on arbitrary power loss

Description:

A write to any configuration register in the Hibernation Module can be corrupted if the V_{DD} supply falls below the minimum operating voltage of 3.05 V while a write is in progress.

Workaround:

Use a voltage supervisor to assert an external reset at 3.05 V. The power-down transition between 3.05 V and 2.70 V must be at least 93 µs and must not have any points where it increases in voltage (must be monotonic).

Silicon Revision Affected:

A1, A3

Fixed:

Fixed on B0.

4 ROM

4.1 USB boot loader in ROM does not operate correctly

Description:

The USB boot loader in ROM does not operate correctly.

Workaround:

To use the USB boot loader, load the StellarisWare version of the USB boot loader into Flash memory.

Silicon Revision Affected:

A1

Fixed:

Fixed on A3.

4.2 ROM_SysCtlClockSet() does not operate correctly with fractional dividers

Description:

The `ROM_SysCtlClockSet()` function in ROM does not operate correctly when using fractional dividers (such as SYSDIV_2_5). The function does work correctly with integer dividers.

Workaround:

If fractional clock dividers are used, load the StellarisWare version 8049 or later of `SysCtlClockSet()` into Flash memory and use that version of the function.

Silicon Revision Affected:

A1

Fixed:

Fixed on A3.

5 Flash memory

5.1 When a 1-kB Flash page is erased, the adjacent page is also erased

Description:

When a 1-kB Flash page is erased, the adjacent page in the even/odd pair is also erased. For example, if page 0 is erased, then page 1 is also erased. Similarly, if page 1 is erased, then page 0 is also erased.

Workaround:

None.

Silicon Revision Affected:

A1

Fixed:

Fixed on A3.

5.2 Flash memory page 0 and 1 may be erased if reset occurs during Flash memory erase operation

Description:

If a page erase command is issued to Flash memory and any type of system reset occurs before the erase operation starts, page 0 and 1 may be erased instead of the specified page.

Workaround:

None.

Silicon Revision Affected:

A1

Fixed:

Fixed on A3.

6 Flash memory, EEPROM

6.1 Reset during Flash memory program or erase or an EEPROM write causes Suspend state

Description:

If a non-POR reset ($\overline{\text{RST}}$ signal, brown out, software, watchdog, or MOSC failure) occurs when the Flash memory is being programmed or erased, or when the EEPROM is being written, any subsequent attempts to program or erase Flash memory or write to EEPROM fail. When this situation occurs, the Flash memory or the EEPROM is in the Suspend state. It is possible that a POR does not clear this condition.

Workaround:

The following code checks to see if the Flash memory or the EEPROM is in the Suspend state and clears it if necessary. This code should be run by an application during initialization and before any attempt to program or erase Flash memory or write to EEPROM.

```
tBoolean
FlashClearSuspend(void)
{
    unsigned long ulVal, ulSave;
    tBoolean bRetcode;

    //
    // Wait a while.
    //
    ROM_SysCtlDelay(10);
    ulSave = HWREG(0x400FD0FC);
    HWREG(0x400FD0FC) = 0x01000003;
    ROM_SysCtlDelay(10);
```

```
        //
        // Read flash controller status.
        //
        ulVal = HWREG(0x400AE054);

        //
        // Is the controller in the suspended state?
        //
        if(ulVal & 0x06)
        {
            //
            // Yes - clear the state.
            //
            HWREG(0x400AE288) = 0x05;
            HWREG(0x400AE20C) = 0x18;
            HWREG(0x400AE110) = 0;
            HWREG(0x400AE2B4) = 0x15;

            do
            {
                //
                // Poll for completion.
                //
                ulVal = HWREG(0x400AE054);
            }
            while(ulVal & 0x100);

    // NEW CODE
            HWREG(0x400AE050) = 0;
    // END NEW CODE
            HWREG(0x400AE2A4) = 0;
            HWREG(0x400AE2C0) = 0;
            HWREG(0x400AE2C4) = 0;
            HWREG(0x400AE2C8) = 0;
            HWREG(0x400AE2CC) = 0;
            HWREG(0x400AE2D0) = 0;
            HWREG(0x400AE2D4) = 0;
            HWREG(0x400AE2D8) = 0;
            HWREG(0x400AE2DC) = 0;
    // NEW CODE
            HWREG(0x400AE050) = 1;
            HWREG(0x400AE2C0) = 0;
            HWREG(0x400AE050) = 0;
    // END NEW CODE

            //
            // Tell the caller that we needed to clean up.
            //
            bRetcode = true;
        }
        else
        {
            //
            // No cleaning up was necessary.
            //
            bRetcode = false;
        }

        ROM_SysCtlDelay(10);
        HWREG(0x400FD0FC) = ulSave;
        ROM_SysCtlDelay(10);

        return(bRetcode);
    }
```

Silicon Revision Affected:

A1

Fixed:

Fixed on A3.

7 EEPROM

7.1 EEPROM blocks must be accessed in alternate pairs to avoid corruption of data

Description:

When the words in a pair of EEPROM blocks are repetitively written, the words of the next pair of blocks get corrupted. In a given group of four blocks of EEPROM, for example, 0, 1, 2 and 3, repeated writes to either block 0 or block 1 cause the data in blocks 2 and 3 to be corrupted.

Workaround:

The EEPROM should be used only in alternate pairs of blocks 0,1,4,5,8,9, and so on, or 2,3,6,7,10,11, and so on.

Silicon Revision Affected:

A1

Fixed:

Fixed on A3.

7.2 EEPROM blocks 0 through 3 may be erased if reset occurs during an EEPROM write

Description:

If a write is issued to EEPROM and any type of system reset occurs before the write starts, blocks 0 through 3 may be erased.

Workaround:

Do not use blocks 0 through 3 in the EEPROM. Blocks 4 through 31 are available for EEPROM use.

Silicon Revision Affected:

A1

Fixed:

Fixed on A3.

7.3 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.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

8 GPIO

8.1 Some GPIO register bits default to the incorrect state

Description:

The AFSEL bits for the following pins are set at reset, resulting in the pins defaulting to their alternate function:

- Port A[1:0]
- Port A[5:2]
- Port B[3:2]

This error in pin functionality may create pin conflict during any type of reset with the following signals:

Signal	Function	I/O	Level
PA0	U0Rx	Input	Tristate
PA1	U0Tx	Output	High
PA2	SSI0Clk	Output	Low
PA3	SSI0Fss	Output	High
PA4	SSI0Rx	Input	Tristate
PA5	SSI0Tx	Output	Low
PB2	I2C0SCL	Indeterminate[a]	Indeterminate[a]

Signal	Function	I/O	Level
PB3	I2C0SDA	Input	Tristate

a. While the pin is in an indeterminate state, it may be driving High or Low. When powering up, this pin is in an indeterminate state for 100 µs after V_{DD} reaches 3.0 V, at which point, PB2 is configured as an input and the level is tristate. If the pin has been operating in I^2C mode and any type of reset occurs, this pin holds its last driven state for 1 PIOSC clock after reset asserts, at which point, PB2 is configured as an input and the level is tristate.

In addition, the PMCx fields in the **GPIOPCTL** register for PD7 and PF0 default to 0x3.

Workaround:

To reconfigure the pins to their intended reset state (GPIO Input, GPIODEN =0), software must clear the corresponding bits in the **GPIOAFSEL** and **GPIODEN** registers for the associated pins. For pins PD7 and PF0, software must clear the corresponding AFSEL bits using the register commit control procedures described in the Commit Control section in the General-Purpose Input/Outputs chapter in the data sheet.

Note that PD7 and PF0 should be grounded, if possible, to prevent triggering an NMI. If that is not possible, an NMI handler must be implemented in case a High level is applied to PD7 or PF0 before they can be reconfigured.

For PD7, the PMC7 assignment of 0x3 is not valid, so it does not cause any issues. However, for PF0, the PMC0 assignment of 0x3 specifies CAN0Rx. If the system design requires CAN0Rx to be on another pin, the PMC0 field for Port F must be assigned to another function or cleared.

Silicon Revision Affected:

A1

Fixed:

Fixed on A3.

8.2 PB1 has permanent internal pull-up resistance

Description:

Regardless of its configuration, PB1 has an internal pull-up resistance that turns on when the voltage on the pin reaches approximately 3.3 V. Once turned on, the resistance remains in place even if the pin is driven Low.

Workaround:

When this pin is configured as an input, the external circuit must drive with an impedance less than or equal to 20 kΩ to provide enough drive strength to over-drive the internal pull-up and achieve the necessary V_{IL} voltage level.

If this pin is configured as an output, be aware that if the output was driven High and a non-POR reset occurs, the output may be driven High after reset instead of defaulting to an input. If a logic Low level is required after reset, a pull-down resistor of 20-kΩ or less should be connected. After reset, once the pin is re-configured as an output, the pin drives the programmed level.

Silicon Revision Affected:

A1

Fixed:

Fixed on A3.

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

Description:

When PC0/TCK is configured as 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.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

8.4 GPIO Port B1 has a leakage path to ground when VDD is removed

Description:

When the device is unpowered and a voltage is applied to PB1, there is a leakage path to ground that results in 45 µA of leakage current. Note that this leakage can also occur during hibernation when not using the VDD3ON mode.

Workaround:

None.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

9 General-Purpose Timers

9.1 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 in the **GPTMSYNC** register prior to setting them for a subsequent update. Using StellarisWare 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);
```

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

9.2 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 lookup table based on the `CLASS` field of the **Device Identification 0 (DID0)** register.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

9.3 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.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

10 Watchdog Timers

10.1 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 ADCn by setting the respective bit(s) in the **RCGUART**, **RCGCSSI**, or the **RCGCADC** registers.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

10.2 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.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

10.3 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.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

10.4 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.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

10.5 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.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

11 ADC

11.1 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.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

11.2 The first ADC sample when using differential mode is incorrect

Description:

The first sample taken after the ADC is configured to operate in differential mode is incorrect. When using the continuous trigger, only the first sample is incorrect. When using other trigger sources, the first value after every trigger is incorrect.

Workaround:

When using the continuous or processor trigger, there is no workaround.

When using other trigger sources, configure Sample Sequencer 3 and an alternate Sample Sequencer in the same manner, but set the priority for SS3 to a higher level. In this configuration, SS3 captures the first, erroneous sample, and the alternate sample sequencer captures correct data for the sequence.

Silicon Revision Affected:

A1, A3

Fixed:

Fixed on B0.

11.3 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: 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.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

11.4 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.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

11.5 Missing trigger or interrupt when multiple sequences configured for processor trigger and different trigger

Description:

If a sample sequence is configured to trigger or interrupt using a processor event and a different, consecutive sample sequence is configured to trigger or interrupt using any other event, the interrupt or trigger for the processor-triggered sample sequence will occasionally not occur, even if the processor-triggered sample sequence is configured with a higher priority.

Workaround:

None.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

11.6 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.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

11.7 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.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

12 UART

12.1 When UART LIN or SIR mode enabled, µDMA burst transfer will not occur

Description:

If the LIN or the IrDA Serial Infrared (SIR) mode is enabled in the UART peripheral and the uDMA is mapped to either UARTn RX or UARTn TX and is configured to do a burst transfer, the burst data transfer will not occur.

Workaround:

Clear the SETn bit in the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register to have the uDMA channel mapped to the UART to respond to single or burst requests to ensure that the data transfer will occur.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

12.2 UART transfers fail at certain system clock frequency and baud rate combinations

Description:

UART data transfers using the TXRIS and RXRIS interrupt bits and FIFOs fail for certain combinations of the system clock frequency and baud rate.

System Clock Freq [MHz]	32	24	16	10	8	5	4	2	1
Failing Baud Rate [bps]	<460800	<460800	<230400	<460800	<115200	<230400	<57600	<38400	<19200

Workaround:

Use a system clock frequency above 32MHz if using the UART with the raw interrupt status bits or use µDMA UART data transfers instead of the TXRIS and RXRIS bits. When using µDMA UART data transfers, there are no system clock frequency and baud rate conflicts.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

12.3 Clearing the RXRIS bit when configured for LIN mode causes the UART to not transfer data

Description:

If LIN mode is enabled in the UART peripheral, clearing the RXRIS bit of the **UART Raw Interrupt Status (UARTRIS)** register during a transfer causes the UART to not receive the rest of the transfer. The RXRIS bit can be cleared by setting the RXIC bit of the **UART Interrupt Clear (UARTICR)** register or by reading data from the receive FIFO until it becomes less than the trigger level (if the FIFO is enabled), or by reading a single byte (if the FIFO is disabled). Any subsequent UART transfers using LIN mode are transferred correctly.

Workaround:

None.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

13 SSI

13.1 Freescale SPI Mode at low SSIClk frequencies can yield data corruption

Description:

Data transmitted by the SPI slave may be corrupted when using Freescale SPI Mode 0 at an SSIClk frequency between 0.5 MHz to 1.1 MHz and a system clock frequency of 33 MHz or lower.

Workaround:

Operate the Freescale SPI Mode 0 at an SSIClk frequency above 1.1 MHz and use a system clock frequency above 33 MHz or use a different mode.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

14 Electrical Characteristics

14.1 Higher current than expected is consumed while V_{DD} ramps up until V_{DDC} crosses 1 V

Description:

While V_{DD} is ramping up, an excess 50 mA of current is consumed until V_{DDC} crosses 1 V. During this time, the output voltage on GPIO pins can go as high as 0.7 V.

Workaround:

None.

Silicon Revision Affected:

A1

Fixed:

Fixed on A3.

14.2 Nominal current consumption is 650 µA higher than specified

Description:

The POR oscillator is always enabled, causing higher current consumption than specified. This issue is noticeable primarily in Deep-sleep operation.

Workaround:

None.

Silicon Revision Affected:

A1

Fixed:

Fixed on A3.

14.3 V_{DD} inrush current of up to 500 mA is seen while V_{DD} ramps up

Description:

V_{DD} inrush current of up to 500 mA is seen while V_{DD} ramps up due to the on-chip LDO regulator charging the LDO and V_{DDC} capacitors. Expected inrush current should be between 50 and 250 mA.

Workaround:

Ensure that the V_{DD} power supply has sufficient output capacitance to supply up to 500 mA for approximately 100 µs.

Silicon Revision Affected:

A1

Fixed:

Fixed on A3.

14.4 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.

Silicon Revision Affected:

A1, A3, B0

Fixed:

Not yet fixed.

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