# Stellaris® LM4F232H5QD RevA1 Errata

This document contains known errata at the time of publication for the Stellaris LM4F232H5QD 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 |
|---|---|---|
| December 2011 | 1.6 | ■ Noted that issue “DID0 register shows revision A0 for revision A1 devices” on page 4 is fixed on revision A3.<br>■ Noted that issue “Device may not operate correctly at certain frequencies” on page 6 is fixed on revision A3.<br>■ Added issue “The MOSC verification circuit does not detect a loss of clock after the clock has been successfully operating” on page 7.<br>■ Added issue “Device may not wake correctly from Sleep mode under certain circumstances” on page 7.<br>■ Added issue “RTC match event is missed if it occurs in a certain window” on page 11. |
| November 2011 | 1.5 | ■ Clarified issue “Boundary scan does not function correctly” on page 4.<br>■ Clarified issue “MCU executes code after BOR before proper power is restored” on page 5.<br>■ Clarified issue “The POR and BOR threshold may vary from the specification” on page 5.<br>■ Clarified issue “Device may not operate correctly at certain frequencies” on page 6.<br>■ Added issue “With a specific clock configuration, device may not wake from Deep-sleep mode” on page 7.<br>■ Clarified issue “Some Hibernation module registers may not have the correct value in two situations” on page 8.<br>■ Clarified issue “USB boot loader in ROM does not operate correctly” on page 11.<br>■ Added issue “Reading the HIBRTCC and HIBRTCSS registers may provide incorrect values” on page 9.<br>■ Added issue “JTAG controller does not ignore transitions on PC0/TCK when it is configured as a GPIO” on page 17. |
| November 2011 | 1.4 | ■ Updated issue “DID0 register shows revision A0 for revision A1 devices” on page 4.<br>■ Added issue “Precision Internal Oscillator (PIOSC) is untrimmed on devices with date codes prior to January 2012” on page 5.<br>■ Added issue “Device may not operate correctly at certain frequencies” on page 6.<br>■ Added issue “GPTMSYNC bits require manual clearing” on page 17.<br>■ Added issue “The GPTMPP register does not correctly indicate 32/64-bit timer capability” on page 17. |

| Date | Revision | Description |
|---|---|---|
| September 2011 | 1.3 | ■ Added issue “Boundary scan does not function correctly” on page 4.<br>■ Added issue “MCU executes code after BOR before proper power is restored” on page 5.<br>■ Added issue “The POR and BOR threshold may vary from the specification” on page 5.<br>■ Added issue “Flash memory page 0 and 1 may be erased if reset occurs during Flash memory erase operation” on page 12.<br>■ Added issue “EEPROM blocks must be accessed in alternate pairs to avoid corruption of data” on page 12.<br>■ Added issue “EEPROM blocks 0 through 3 may be erased if reset occurs during an EEPROM write” on page 13.<br>■ Added issue “Reset during Flash memory program or erase or an EEPROM write causes Suspend state” on page 13.<br>■ Added issue “PB1 has permanent internal pull-up resistance” on page 16.<br>■ Added issue “Retriggering a sample sequencer before it has completed the current sequence results in continuous sampling” on page 18.<br>■ Added issue “USB controller VBUS signal does not drop to 0 V when 5-V supply is removed” on page 19.<br>■ Added issue “USB Host controller may not be used to communicate with a low-speed Device when connected through a hub” on page 20.<br>■ Added additional information to issue “Higher current than expected is consumed while $V_{DD}$ ramps up until $V_{DDC}$ crosses 1 V” on page 20.<br>■ Added issue “Nominal current consumption is 650 µA higher than specified” on page 20.<br>■ Added issue “$V_{DD}$ inrush current of up to 500 mA is seen while $V_{DD}$ ramps up” on page 21. |
| June 2011 | 1.2 | ■ Added issue “DID0 register shows revision A0 for revision A1 devices” on page 4.<br>■ Clarified issue “Some Hibernation module registers may not have the correct value in two situations” on page 8.<br>■ Added issue “USB boot loader in ROM does not operate correctly” on page 11.<br>■ Added issue “ROM_SysCtlClockSet() does not operate correctly with fractional dividers” on page 11.<br>■ Added issue “When a 1-kB Flash page is erased, the adjacent page is also erased” on page 12.<br>■ Added issue “External reference requires additional configuration” on page 18.<br>■ Added issue “Higher current than expected is consumed while $V_{DD}$ ramps up until $V_{DDC}$ crosses 1 V” on page 20. |
| May 2011 | 1.1 | ■ Added issue “Some GPIO register bits default to the incorrect state” on page 15. |
| 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 |

| Erratum Number | Erratum Title | Module Affected | Revision(s) Affected |
|---|---|---|---|
| 2.2 | MCU executes code after BOR before proper power is restored | System Control | A1 |
| 2.3 | The POR and BOR threshold may vary from the specification | System Control | A1 |
| 2.4 | Precision Internal Oscillator (PIOSC) is untrimmed on devices with date codes prior to January 2012 | System Control | A1 |
| 2.5 | Device may not operate correctly at certain frequencies | System Control | A1 |
| 2.6 | With a specific clock configuration, device may not wake from Deep-sleep mode | System Control | A1 |
| 2.7 | The MOSC verification circuit does not detect a loss of clock after the clock has been successfully operating | System Control | A1 |
| 2.8 | Device may not wake correctly from Sleep mode under certain circumstances | System Control | A1 |
| 3.1 | Some Hibernation module registers may not have the correct value in two situations | Hibernation | A1 |
| 3.2 | Reading the HIBRTCC and HIBRTCSS registers may provide incorrect values | Hibernation | A1 |
| 3.3 | Device fails to wake from hibernation within a certain time after hibernation is requested | Hibernation | A1 |
| 3.4 | RTC match event is missed if it occurs in a certain window | Hibernation | A1 |
| 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 | EEPROM blocks must be accessed in alternate pairs to avoid corruption of data | EEPROM | A1 |
| 6.2 | EEPROM blocks 0 through 3 may be erased if reset occurs during an EEPROM write | EEPROM | A1 |
| 7.1 | Reset during Flash memory program or erase or an EEPROM write causes Suspend state | Flash memory, EEPROM | A1 |
| 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 |
| 9.1 | GPTMSYNC bits require manual clearing | General-Purpose Timers | A1 |
| 9.2 | The GPTMPP register does not correctly indicate 32/64-bit timer capability | General-Purpose Timers | A1 |
| 10.1 | External reference requires additional configuration | ADC | A1 |
| 10.2 | Retriggering a sample sequencer before it has completed the current sequence results in continuous sampling | ADC | A1 |

| Erratum Number | Erratum Title | Module Affected | Revision(s) Affected |
|---|---|---|---|
| 10.3 | The first ADC sample when using differential mode is incorrect | ADC | A1 |
| 11.1 | USB controller VBUS signal does not drop to 0 V when 5-V supply is removed | USB | A1 |
| 11.2 | USB Host controller may not be used to communicate with a low-speed Device when connected through a hub | USB | A1 |
| 12.1 | Higher current than expected is consumed while $V_{DD}$ ramps up until $V_{DDC}$ crosses 1 V | Electrical Characteristics | A1 |
| 12.2 | Nominal current consumption is 650 µA higher than specified | Electrical Characteristics | A1 |
| 12.3 | $V_{DD}$ inrush current of up to 500 mA is seen while $V_{DD}$ ramps up | Electrical Characteristics | A1 |

# 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 MCU executes code after BOR before proper power is restored

**Description:**

Following a brown-out reset, the microcontroller comes out of reset and begins executing instructions prior to the restoration of the minimum operating voltage on $V_{DD}$. The device cannot operate reliably below the minimum operating voltage on $V_{DD}$, so unexpected operation may occur.

**Workaround:**

Use a voltage supervisor to ensure that the proper operating voltage is supplied to the device.

**Silicon Revision Affected:**

A1

**Fixed:**

Fixed on A3.

## 2.3 The POR and BOR threshold may vary from the specification

**Description:**

The power-on reset threshold ($V_{TH}$) and the brown-out reset threshold ($V_{BTH}$) voltages may vary from the values specified in the data sheet by up to ±120 mV. As a result, the device may come out of reset before $V_{DD}$ rises to the minimum operating voltage of 2.97 V or it may reset after $V_{DD}$ falls below the minimum operating voltage. The device cannot operate reliably below the minimum operating voltage on $V_{DD}$, so unexpected operation may occur.

**Workaround:**

Use a voltage supervisor to ensure that the proper operating voltage is supplied to the device.

**Silicon Revision Affected:**

A1

**Fixed:**

Fixed on A3.

## 2.4 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.

## 2.5 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.

## 2.6 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

**Fixed:**

Not yet fixed.

## 2.7 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

**Fixed:**

Not yet fixed.

## 2.8 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

**Fixed:**

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

**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

**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 $\overline{HIB}$ 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

**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

**Fixed:**

Not yet fixed.

# 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 EEPROM

## 6.1 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.

## 6.2 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 Flash memory, EEPROM

## 7.1 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.

# 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]
- Port D[7]
- Port F[0]

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 |

| Signal | Function | I/O | Level |
|---|---|---|---|
| PA4 | SSI0Rx | Input | Tristate |
| PA5 | SSI0Tx | Output | Low |
| PB2 | I2C0SCL | Indeterminate[a] | Indeterminate[a] |
| PB3 | I2C0SDA | Input | Tristate |
| PD7 | NMI | Input | Tristate |
| PF0 | NMI | 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.

**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.

**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

**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

**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

**Fixed:**

Not yet fixed.

# 10 ADC

## 10.1 External reference requires additional configuration

**Description:**

Setting the VREF bit of the **ADC Control (ADCCTL)** register has no effect; VDDA and GNDA are used as the voltage reference regardless of the status of this bit.

**Workaround:**

Setting the AMSEL bit for PB6 enables the use of the external voltage reference. Setting this AMSEL bit has no effect on the functionality of PB6 as PB6 is not pinned out on this device. For future compatibility, software should set both the VREF bit in the **ADCCTL** register and the PB6 AMSEL bit.

**Silicon Revision Affected:**

A1

**Fixed:**

Fixed on A3.

## 10.2 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

**Fixed:**

Not yet fixed.

## 10.3 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

**Fixed:**

Not yet fixed.

# 11 USB

## 11.1 USB controller VBUS signal does not drop to 0 V when 5-V supply is removed

**Description:**

When the USB0VBUS pin is configured to be used by the USB controller, because of “PB1 has permanent internal pull-up resistance” on page 16 an internal current path causes the voltage on VBUS to drop only from 5 V to 3.3 V, instead of 0 V, when the 5 V supply to the USB0VBUS pin is removed. This issue occurs regardless of the USB mode of operation.

The DEVMODOTG and DEVMOD bits in the **USB General-Purpose Control and Status (USBGPCS)** register can be used to configure the USB controller to operate only in Host mode or Device mode and allowing PB0 and PB1 to be used as GPIOs. If both the DEVMODOTG and DEVMOD bits are set, indicating Device mode, the controller fails to detect a disconnection event. If the DEVMODOTG bit is set and the DEVMOD bit is clear, indicating Host mode, the VBUS voltage is held at 3.3V until a device is connected. When the DEVMODOTG bit is clear indicating OTG mode, the controller fails to switch modes from Device mode to Host mode.

**Workaround:**

Use an external 20-kΩ pull-down on the USB0VBUS pin.

**Silicon Revision Affected:**

A1

**Fixed:**

Fixed on A3.

## 11.2 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.

**Silicon Revision Affected:**

A1

**Fixed:**

Not yet fixed.

# 12 Electrical Characteristics

## 12.1 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.

## 12.2 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.

## 12.3 $V_{DD}$ 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.

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