MSP430F5508 Device Erratasheet

1   Revision History

✓ The check mark indicates that the issue is present in the specified revision.

The revision of the device can be identified by the revision letter on the Package Markings or by the HW_ID located inside the TLV structure of the device

Errata Number	Rev E	Rev D	Rev C
ADC30			✓
ADC31			✓
BSL6	✓	✓	
BSL7	✓	✓	✓
CPU39	✓	✓	✓
CPU40	✓	✓	✓
CPU43	✓	✓	✓
DMA4	✓	✓	✓
DMA10	✓	✓	✓
EEM11	✓	✓	✓
EEM13	✓	✓	✓
EEM17	✓	✓	✓
EEM19	✓	✓	✓
EEM21	✓	✓	✓
EEM23	✓	✓	✓
FLASH37			✓
JTAG20	✓	✓	✓
MPY1	✓	✓	✓
PMAP1	✓	✓	✓
PMM9	✓	✓	✓
PMM10			✓
PMM11	✓	✓	✓
PMM12	✓	✓	✓
PMM14	✓	✓	✓
PMM15	✓	✓	✓
PMM17			✓
PMM18	✓	✓	✓
PMM20	✓	✓	✓
PORT15	✓	✓	✓
PORT16	✓	✓	✓
PORT19	✓	✓	✓
RTC3	✓	✓	✓
RTC6	✓	✓	✓
SYS12	✓	✓	✓

Errata Number	Rev E	Rev D	Rev C
SYS16	✓	✓	✓
SYS18	✓	✓	✓
TAB23			✓
UCS6			✓
UCS7	✓	✓	✓
UCS9	✓	✓	✓
UCS10			✓
UCS11	✓	✓	✓
USB8			✓
USB9	✓	✓	✓
USB10	✓	✓	✓
USCI26	✓	✓	✓
USCI30			✓
USCI31	✓	✓	✓
USCI35	✓	✓	✓
WDG4	✓	✓	✓

2   Package Markings

PT48   *LQFP (TP), 48 Pin*

M430Fxxxx
ti YMSG4
LLLL #
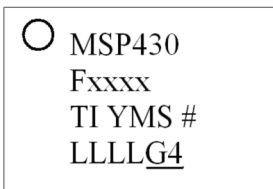○
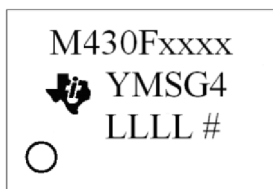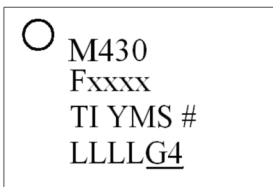
YM   = Year and Month Date Code
LLLL = LOT Trace Code
S   = Assembly Site Code
= DIE Revision
o   = PIN 1

RGZ48   *QFN (RGZ), 48 Pin*

○ MSP430
Fxxxx
TI YMS #
LLLL<u>G4</u>

YM   = Year and Month Date Code
LLLL = LOT Trace Code
S   = Assembly Site Code
= DIE Revision
o   = PIN 1

○ M430
Fxxxx
TI YMS #
LLLL<u>G4</u>

YM   = Year and Month Date Code
LLLL = LOT Trace Code
S   = Assembly Site Code
= DIE Revision
o   = PIN 1

○
MSP430™
Fxxx
TI YMS #
LLLL <u>G4</u>

YM   = Year and Month Date Code
S   = Assembly Site Code
#   = Die Revision
LLLL = Lot Trace Code
○   = Pin 1

Note: Package marking with "TM" applies only to devices released after 2011.

RGC64   *QFN (RGC), 64 pin*

```
┌─────────────────────┐      TI    = TI
│  ○ M430Fxxx         │      YM   = Year and Month Date Code
│                     │      LLLL = LOT Trace Code
│   TI YMS            │      S     = Assembly Site Code
│   LLLL #            │      #     = DIE Revision
│                     │      o     = PIN 1
└─────────────────────┘
┌─────────────────────┐      TI    = TI
│ ○                   │      YM   = Year and Month Date Code
│    M430Fxxxx        │      LLLL = LOT Trace Code
│                     │      S     = Assembly Site Code
│   TI  YMS  #        │      #     = DIE Revision
│   LLLL  G4          │      o     = PIN 1
└─────────────────────┘
┌─────────────────────┐      YM    = Year and Month Date Code
│ ○                   │      S      = Assembly Site Code
│   MSP430™           │      #      = Die Revision
│   Fxxx              │      LLLL = Lot Trace Code
│   TI YMS #          │      ○      = Pin 1
│   LLLL G4           │
└─────────────────────┘
```

Note: Package marking with "TM" applies only to devices released after 2011.

ZQE80   *BGA (ZQE), 80 pin*

```
┌─────────────────────┐      YM    = Year and Month Date Code
│        ti           │      LLLL = LOT Trace Code
│   M430Fxxxxx        │      S      = Assembly Site Code
│   YMLLLLS  #        │      #      = DIE Revision
│     G1              │      o      = PIN 1
│ ○                   │
└─────────────────────┘
┌─────────────────────┐      YM    = Year and Month Date Code
│                     │      LLLL = Lot Trace Code
│   MSP430™           │      S      = Assembly Site Code
│   Fxxx              │      #      = Die Revision
│   YMLLLLS #         │      ○      = Pin 1
│   TI  G1            │
│ ○                   │
└─────────────────────┘
```

Note: Package marking with "TM" applies only to devices released after 2011.

3   TLV Hardware Revision

Die Revision	TLV Hardware Revision
Rev E	14h
Rev D	13h
Rev C	12h

Further guidance on how to locate the TLV structure and read out the HW_ID can be found in the device User's Guide.

4   Detailed Bug Description

ADC30	ADC10_A Module

Function   ADC hangs when a slow ADC clock source is used

Description   When using ADC10SC bit to software trigger ADC conversions, the ADC10 state machine may hang (that is, no further interrupts are generated) if the ADC clock source (ADC10CLK) is significantly slower than the system clock (MCLK). This issue can be observed when ADC10CLK < MCLK/8. If the ADC conversions are re-triggered by setting the ADC10SC bit before eight MCLK cycles have elapsed since the ADCIFG interrupt bit is set or the ADC10BUSY bit is reset, then this behavior can be seen.

Workaround   - Use MODOSC or any clock that is > MCLK/8 as ADC10CLK

or

- When using a clock source <= MCLK/8 as ADC10CLK, ensure the application code provides a delay of at least one ADC10CLK cycle before setting the ADC10SC bit again.

ADC31	ADC10_A Module

Function   Sporadic occurrence of false ADC conversion results

Description   Sporadic errors in the ADC conversion results occur when the input voltage is an integer fraction of the reference voltage Vref. The probability of these sporadic errors is extremely low but possible. The false ADC results deviate from the actual input value depending on which integer fraction of Vref the input voltage is; for example:

- With Vin= 1/2 Vref, the most significant bit of the ADC conversion result might be affected, thus affecting all the bits that follow the most significant bit and, therefore, a completely wrong reading that ranges anywhere in the ADC range might be read.

- With Vin = 1/4 or 3/4 Vref, the second most significant bit along with all the bits that follow this bit are affected. In this case, the false reading is confined to one-half of the ADC range (as the first most significant bit value is correct).

- With Vin = 1/8, 3/8, 5/8, or 7/8 Vref, the third most significant bit and all the bits that follow this bit are affected. The false reading is confined to one-fourth of the ADC range (as the first two most significant bit values are correct)

- Similarly this behavior continues to the last bit.

Workaround   For measuring dc signals: Depending on the dc signal range being measured, discard the obvious outliers in measurement results and average out the false reading for errors in the lower bits.

For measuring ac signals: If possible, use a higher over-sampling rate to average out the error readings.

BSL6	BSL Module

Function   USB BSL does not respond properly to suspend/reset events from the USB host

Description   The USB BSL in affected revisions contains an improper configuration of the USB module. As a result, errors might occur in response to suspend/reset events from the USB host. (Since enumeration of the USB device often involves suspend and/or reset events, an enumeration might trigger the failure.) If the failure occurs, the device becomes unresponsive to the USB host.

If the failure occurs, and if application code exists in main flash, a reset (BOR/POR/PUC) can be issued to switch execution away from the BSL, to the application. Given the same USB host/setup circumstances, the problem is likely to occur again on subsequent attempts.

Applications that do not use the USB BSL are unaffected.

Workaround	1. The BSL can be updated via JTAG with a version that does not contain this bug. Use the code published in <ulink="www.ti.com/lit/pdf/slaa450">BSL documentation</ulink> starting with version 00.07.85.36.

BSL7   *BSL Module*

Function	BSL does not start after waking up from LPMx.5
Description	When waking up from LPMx.5 mode, the BSL does not start as it does not clear the Lock I/O bit (LOCKLPM5 bit in PM5CTL0 register) on start-up.
Workaround	1. Upgrade the device BSL to the latest version (see Creating a Custom Flash-Based Bootstrap Loader (BSL) Application Note - SLAA450 for more details)
	OR
	2. Do not use LOCKLPM5 bit (LPMx.5) if the BSL is used but cannot be upgraded.

CPU39   *CPUXv2 Module*

Function	PC is corrupted when single-stepping through an instruction that clears the GIE bit
Description	Single-stepping over an instruction that clears the General Interrupt Enable bit (for example DINT or BIC #GIE,SR) when the GIE bit was previously set may corrupt the PC. For example, the DINT or BIC #GIE,SR is a 2-byte instruction. Single stepping through this instruction increments the PC by a value of 4 instead of 2 thus corrupting the next PC value.
	Note: This erratum applies to debug mode only.
Workaround	Insert a NOP or __no_operation() intrinsic immediately after the line of code that clears the GIE bit.

CPU40   *CPUXv2 Module*

Function	PC is corrupted when executing jump/conditional jump instruction that is followed by instruction with PC as destination register or a data section
Description	If the value at the memory location immediately following a jump/conditional jump instruction is 0X40h or 0X50h (where X = don't care), which could either be an instruction opcode (for instructions like RRCM, RRAM, RLAM, RRUM) with PC as destination register or a data section (const data in flash memory or data variable in
	RAM), then the PC value is auto-incremented by 2 after the jump instruction is executed; therefore, branching to a wrong address location in code and leading to wrong program execution.
	For example, a conditional jump instruction followed by data section (0140h).
	@0x8012 Loop DEC.W R6
	@0x8014 DEC.W R7
	@0x8016 JNZ Loop

Copyright © 2012–2013, Texas Instruments Incorporated

@0x8018 Value1 DW 0140h

Workaround	In assembly, insert a NOP between the jump/conditional jump instruction and program code with instruction that contains PC as destination register or the data section.

CPU43   *CPUXv2 Module*

Function	Halt operation in debug mode may cause unintended behavior
Description	In certain cases when using the 'Halt CPU' function available via the IDE (CCS or IAR), on continuing code execution after a halt, the program counter may skip an instruction.
	Pausing and resuming code execution after a breakpoint works as expected and is not affected by the erratum.
	Note: This erratum affects debug mode only.
Workaround	None.

DMA4   *DMA Module*

Function	Corrupted write access to 20-bit DMA registers
Description	When a 20-bit wide write to a DMA address register (DMAxSA or DMAxDA) is interrupted by a DMA transfer, the register contents may be unpredictable.
Workaround	1. Design the application to guarantee that no DMA access interrupts 20-bit wide accesses to the DMA address registers.
	OR
	2. When accessing the DMA address registers, enable the Read Modify Write disable bit (DMARMWDIS = 1) or temporarily disable all active DMA channels (DMAEN = 0).
	OR
	3. Use word access for accessing the DMA address registers. Note that this limits the values that can be written to the address registers to 16-bit values (lower 64K of Flash).

DMA10   *DMA Module*

Function	DMA interrupting CPU wait state might cause peripheral module into unknown state.
Description	When the CPU accesses a module that is capable of stalling the CPU with a wait mechanism, if a DMA interrupts the instruction during the CPU stall, the module might be caused into an unknown state.
	The affected modules (if present on the device) that can stall CPU are: FRAM controller in manual timing mode, MPY, CRC, USB, and RF1A.
	As an example a wrong result can be read by DMA from MPY result register because the DMA does not wait until MPY operation is finished.
Workaround	Disable DMA when using affected modules.

EEM11   *EEM Module*

Function	Conditional register write trigger fails while executing rotate instructions
Description	A conditional register write trigger will fail to generate the expected breakpoint if the

trigger condition is a result of executing one of the following rotate instructions: RRUM,RRCM, RRAM and RLAM.

Workaround	None
	Note: This erratum applies to debug mode only.

EEM13 *EEM Module*

Function	Halting the debugger does not return correct PC value when in LPM
Description	When debugging, if the device is in any low power mode and the debugger is halted, the program counter update by the debugger is corrupted. The debugger is unable to halt at the correct location.
Workaround	None.
	Note: This erratum applies to debug mode only.

EEM17 *EEM Module*

Function	Wrong Breakpoint halt after executing Flash Erase/Write instructions
Description	Hardware breakpoints or Conditional Address triggered breakpoints on instructions that follow Flash Erase/Write instructions, stops the debugger at the actual Flash Erase/Write instruction even though the flash erase/write operation has already been executed. The hardware/conditional address triggered breakpoints that are placed on either the next two single opcode instructions OR the next double opcode instruction that follows the Flash Erase/Write instruction are affected by this erratum.
	Note: This erratum affects debug mode only.
Workaround	None. Use other conditional/advanced triggered breakpoints to halt the debugger right after Flash erase/write instructions.

EEM19 *EEM Module*

Function	DMA may corrupt data in debug mode
Description	When the DMA is enabled and the device is in debug mode, the data transferred by the DMA may be corrupted when a breakpoint is hit or when the debug session is halted.
	NOTE: This erratum applies to debug mode only.
Workaround	None. Do not set a breakpoint during a DMA transfer.

EEM21 *EEM Module*

Function	LPMx.5 debug limitations
Description	Debugging the device in LPMx.5 mode might wake the device up from LPMx.5 mode inadvertently, and it is possible that the device enters a lock-up condition; that is, the device cannot be accessed by the debugger any more.
Workaround	Follow the debugging steps in Debugging MSP430 LPM4.5 SLAA424 .

EEM23	***EEM Module***

Function	EEM functions do not work reliably when modules using wait cycles are enabled

Description   When modules using wait states (USB, MPY,CRC and FRAM controller in manual mode) are enabled the EEM may not perform profile counter and state storage functions reliably.

Note: This erratum affects debug mode only.

Workaround   Do not enable profile counter and state storage functions when modules using wait states are enabled.

FLASH37	***FLASH Module***

Function	Corrupted flash read when SVM low-side flag is triggered

Description   If the SVM low side is enabled, a change in the VCORE voltage level (an increase in the VCORE level) may cause the currently executed read operation from flash to be incorrect and may lead to unexpected code execution or incorrect data. This can happen under any one of the following conditions:

- When the VCORE is changed in application, the SVM low side is used to indicate if the core voltage has settled by using the SVMDLYIFG flag. The failure occurs only when a flash access is concurrent to the expiration of the settling time delay.

- Unexpected changes in the VCORE voltage level

For code examples and detailed guidance on the PMM operation and software APIs for PMM configuration see the driverlib APIs from 430Ware (MSP430Ware).

Workaround   - Execute the procedure to change the VCORE level from RAM.

or

- If executing from flash, follow the procedure below when increasing the VCORE level. Note: To apply this workaround, the SVM low-side comparator must operate in normal mode (SVMLFP = 0 in SVMLCTL).

// Set SVM highside to new level and check if a VCore increase is possible

SVSMHCTL = SVMHE | SVSHE | (SVSMHRRL0 * level);

// Wait until SVM highside is settled

while ((PMMIFG & SVSMHDLYIFG) == 0);

// Clear flag

PMMIFG &= ~SVSMHDLYIFG;

// Set also SVS highside to new level

// Vcc is high enough for a Vcore increase

SVSMHCTL |= (SVSHRVL0 * level);

// Wait until SVM highside is settled

while ((PMMIFG & SVSMHDLYIFG) == 0);

// Clear flag

PMMIFG &= ~SVSMHDLYIFG;

//**************flow change for errata workaround ************

// Set VCore to new level

PMMCTL0_L = PMMCOREV0 * level;

// Set SVM, SVS low side to new level

SVSMLCTL = SVMLE | (SVSMLRRL0 * level)| SVSLE | (SVSLRVL0 * level);

// Wait until SVM, SVS low side is settled

while ((PMMIFG & SVSMLDLYIFG) == 0);

// Clear flag

PMMIFG &= ~SVSMLDLYIFG;

//*************flow change for errata workaround ************

JTAG20 *JTAG Module*

Function BSL does not exit to application code

Description The methods used to exit the BSL per MSP430 Programming Via the Bootstrap Loader (SLAU319) are invalid.

Workaround To exit the BSL one of the following methods must be used.

- A Power cycle

or

- Toggle the TEST pin twice when nRST is high and then pull nRST low.
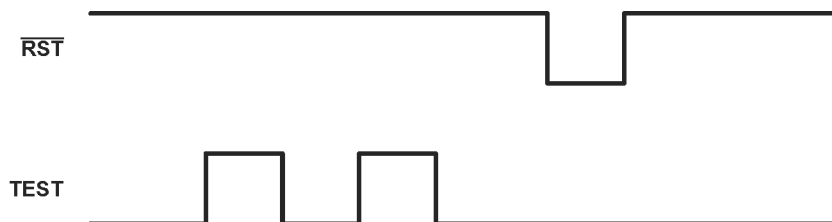
Note: This sequence is not subject to timing constraints and the appropriate level transitions are sufficient to trigger an exit from BSL mode.

MPY1 *MPY Module*

Function Save and Restore feature on MPY32 not functional

Description The MPY32 module uses the Save and Restore method which involves saving the multiplier state by pushing the MPY configuration/operand values to the stack before using the multiplier inside an Interrupt Service Routine (ISR) and then restoring the state by popping the configuration/operand values back to the MPY registers at the end of the ISR. However due to the erratum the Save and Restore operation fails causing the write operation to the OP2H register right after the restore operation to be ignored as it is not preceded by a write to OP2L register resulting in an invalid multiply operation.

Workaround None. Disable interrupts when writing to OP2L and OP2H registers.

Note: When using the C-compiler, the interrupts are automatically disabled while using the MPY32

PMAP1   *PMAP Module*

Function   Port Mapping Controller does not clear unselected inputs to mapped module.

Description   The Port Mapping Controller provides the logical OR of all port mapped inputs to a module (Timer, USCI, etc). If the PSEL bit (PxSEL.y) of a port mapped input is cleared, then the logic level of that port mapped input is latched to the current logic level of the input. If the input is in a logical high state, then this high state is latched into the input of the logical OR. In this case, the input to the module is always a logical 1 regardless of the state of the selected input.

Workaround   1. Drive input to the low state before clearing the PSEL bit of that input and switching to another input source.

or

2. Use the Port Mapping Controller reconfiguration feature, PMAPRECFG, to select inputs to a module and map only one input at a time.

PMM9   *PMM Module*

Function   False SVSxIFG events

Description   The comparators of the SVS require a certain amount of time to stabilize and output a correct result once re-enabled; this time is different for the Full Performance versus the Normal mode. The time to stabilize the SVS comparators is intended to be accounted for by a built-in event-masking delay of 2 us when Full Performance mode is enabled.

However, the comparators of the SVS in Full Performance mode take longer than 2 us to stabilize so the possibility exists that a false positive will be triggered on the SVSH or SVSL. This results in the SVSxIFG flags being set and depending on the configuration of SVSxPE bit a POR can also be triggered.

Additionally when the SVSxIFGs are set, all GPIOs are tri-stated i.e. floating until the SVSx comparators are settled.

The SVS IFG's are falsely set under the following conditions:

1. Wakeup from LPM2/3/4 when SVSxMD = 0 (default setting) && SVSxFP=1. The SVSx comparators are disabled automatically in LPM2/3/4 and are then re-enabled on return to active mode.

2. SVSx is turned on in full performance mode (SVSxFP=1).

3. A PUC/POR occurs after SVSx is disabled. After a PUC or POR the SVSx are enabled automatically but the settling delay does not get triggered. Based on SVSxPE bit this may lead to POR events until the SVS comparator is fully settled.

Workaround   For each of the above listed conditions the following workarounds apply:

1. If the Full Performance mode is to be enabled for either the high- or low-side SVS comparators, the respective SVSxMD bits must be set (SVSxMD = 1) such that the SVS comparators are not temporarily shut off in LPM2/3/4. Note that this is equivalent to a 2 uA (typical) adder to the low power mode current, per the device-specific datasheet, for each SVSx that remains enabled.

2. The SVSx must be turned on in normal mode (SVSxFP=0). It can be reconfigured to use full performance mode once the SVSx/SVMx delay has expired.

3. Ensure that SVSH and SVSL are always enabled.

PMM10   *PMM Module*

Function   SVS/SVM flags disabled after Power Up Clear reset

Description   SVS/SVM interrupt flag functionality is disabled after a Power Up Clear (PUC) Reset if
                          the SVS was disabled before the PUC reset was applied.

Workaround   A write access to the intended SVSx register after PUC re-enables the SVS & SVM
                         interrupt flags.

PMM11   *PMM Module*

Function   MCLK comes up fast on exit from LPM3 and LPM4

Description   The DCO exceeds the programmed frequency of operation on exit from LPM3 and LPM4
                         for up to 6 us. This behavior is masked from affecting code execution by default: SVSL
                         and SVML run in normal-performance mode and mask CPU execution for 150 us on
                         wakeup from LPM3 and LPM4. However, when the low-side SVS and the SVM are
                         disabled or are operating in full-performance mode (SVMLE = 0 and SVSLE = 0, or
                         SVMLFP = 1 and SVSLFP = 1) AND MCLK is sourced from the internal DCO running
                         over 4 MHz, 7 MHz, 11 MHz, or 14 MHz at core voltage levels 0, 1, 2, and 3,
                         respectively, the mask lasts only 2 us. MCLK is, therefore, susceptible to run out of spec
                         for 4 us.

Workaround   Set the MCLK divide bits in the Unified Clock System Control 5 Register (UCSCTL5) to
                         divide MCLK by two prior to entering LPM3 or LPM4 (set DIVMx = 001). This prevents
                         MCLK from running out of spec when the CPU wakes from the low-power mode.
                         Following the wakeup from the low-power mode, wait 32, 48, 80, or 100 cycles for core
                         voltage levels 0, 1, 2, and 3, respectively, before resetting DIVMx to zero and running
                         MCLK at full speed [for example, __delay_cycles(100)].

PMM12   *PMM Module*

Function   SMCLK comes up fast on exit from LPM3 and LPM4

Description   The DCO exceeds the programmed frequency of operation on exit from LPM3 and LPM4
                         for up to 6 us. When SMCLK is sourced by the DCO, it is not masked on exit from LPM3
                         or LPM4. Therefore, SMCLK exceeds the programmed frequency of operation on exit
                         from LPM3 and LPM4 for up to 6 us. The increased frequency has the potential to
                         change the expected timing behavior of peripherals that select SMCLK as the clock
                         source.

Workaround   - Use XT2 as the SMCLK oscillator source instead of the DCO.

                         or

                         - Do not disable the clock request bit for SMCLKREQEN in the Unified Clock System
                         Control 8 Register (UCSCTL8). This means that all modules that depend on SMCLK to
                         operate successfully should be halted or disabled before entering LPM3 or LPM4. If the
                         increased frequency prevents the proper function of an affected module, wait 32, 48, 80,
                         or 100 cycles for core voltage levels 0, 1, 2, or 3, respectively, before re-enabling the
                         module [for example, __delay_cycles(100)].

PMM14   *PMM Module*

Function   Increasing the core level when SVS/SVM low side is configured in full-performance

mode causes device reset

Description   When the SVS/SVM low side is configured in full performance mode (SVSMLCTL.SVSLFP = 1), the setting time delay for the SVS comparators is ~2us. When increasing the core level in full-performance mode; the core voltage does not settle to the new level before the settling time delay of the SVS/SVM comparator expires. This results in a device reset.

Workaround   When increasing the core level; enable the SVS/SVM low side in normal mode (SVSMLCTL.SVSLFP=0). This provides a settling time delay of approximately 150us allowing the core sufficient time to increase to the expected voltage before the delay expires.

PMM15   *PMM Module*

Function   Device may not wake up from LPM2, LPM3, or LPM4

Description   Device may not wake up from LPM2, LPM3 or LMP4 if an interrupt occurs within 1 us after the entry to the specified LPMx; entry can be caused either by user code or automatically (for example, after a previous ISR is completed). Device can be recovered with an external reset or a power cycle. Additionally, a PUC can also be used to reset the failing condition and bring the device back to normal operation (for example, a PUC caused by the WDT).

This effect is seen when:

- A write to the SVSMHCTL and SVSMLCTL registers is immediately followed by an LPM2, LPM3, LPM4 entry without waiting the requisite settling time ((PMMIFG.SVSMLDLYIFG = 0 and PMMIFG.SVSMHDLYIFG = 0)).

or

The following two conditions are met:

- The SVSL module is configured for a fast wake-up or when the SVSL/SVML module is turned off. The affected SVSMLCTL register settings are shaded in the following table.

	SVSLE	SVSLMD	SVSLFP	AM, LPM0/1 SVSL state	Manual SVSMLACE = 0 LPM2/3/4 SVSL State	Automatic SVSMLACE = 1 LPM2/3/4 SVSL State	Wakeup Time LPM2/3/4
SVSL	0	x	x	OFF	OFF	OFF	$t_{\text{WAKE-UP FAST}}$
	1	0	0	Normal	OFF	OFF	$t_{\text{WAKE-UP SLOW}}$
	1	0	1	Full Performance	OFF	OFF	$t_{\text{WAKE-UP FAST}}$
	1	1	0	Normal	Normal	OFF	$t_{\text{WAKE-UP SLOW}}$
	1	1	1	Full Performance	Full Performance	Normal	$t_{\text{WAKE-UP FAST}}$

	SVMLE	SVMLFP	AM, LPM0/1 SVML state	Manual SVSMLACE = 0 LPM2/3/4 SVML State	Automatic SVSMLACE = 1 LPM2/3/4 SVML State	Wakeup Time LPM2/3/4
SVML	0	x	OFF	OFF	OFF	$t_{\text{WAKE-UP FAST}}$
	1	0	Normal	Normal	OFF	$t_{\text{WAKE-UP SLOW}}$
	1	1	Full Performance	Full Performance	Normal	$t_{\text{WAKE-UP FAST}}$

and

-The SVSH/SVMH module is configured to transition from Normal mode to an OFF state when moving from Active/LPM0/LPM1 into LPM2/LPM3/LPM4 modes. The affected SVSMHCTL register settings are shaded in the following table.

Copyright © 2012–2013, Texas Instruments Incorporated

	SVSHE	SVSHMD	SVSHFP	AM, LPM0/1 SVSH state	Manual SVSMHACE = 0	Manual SVSMHACE = 1
					LPM2/3/4 SVSH State	LPM2/3/4 SVSH State
SVSH	0	x	x	OFF	OFF	OFF
	1	0	0	Normal	OFF	OFF
	1	0	1	Full Performance	OFF	OFF
	1	1	0	Normal	Normal	OFF
	1	1	1	Full Performance	Full Performance	Normal
	SVSHE	SVMHFP		AM, LPM0/1 SVSH state	Manual SVSMHACE = 0	Manual SVSMHACE = 1
SVMH					LPM2/3/4 SVSH State	LPM2/3/4 SVSH State
	0	x		OFF	OFF	OFF
	1	0		Normal	Normal	OFF
	1	1		Full Performance	Full Performance	Normal

Workaround   Any write to the SVSMxCTL register must be followed by a settling delay (PMMIFG.SVSMLDLYIFG = 0 and PMMIFG.SVSMHDLYIFG = 0) before entering LPM2, LPM3, LPM4.

and

1. Ensure the SVSx, SVMx are configured to prevent the issue from occurring by the following:

- Configure the SVSL module for slow wake up (SVSLFP = 0). Note that this will increase the wakeup time from LPM2/3/4 to twakeupslow (~150 us).

or

- Do not configure the SVSH/SVMH such that the modules transition from Normal mode to an OFF state on LPM entry. Instead force the modules to remain ON even in LPMx. Note that this will cause increased power consumption when in LPMx.

Refer to the MSP430F5xx and MSP430F6xx Core Libraries (SLAA448) for proper PMM configuration functions.

Use the following function, PMM15Check (void), to determine whether or not the existing PMM configuration is affected by the erratum. The return value of the function is 1 if the configuration is affected, and 0 if the configuration is not affected.

unsigned char PMM15Check (void)

{

// First check if SVSL/SVML is configured for fast wake-up

if ((!(SVSMLCTL & SVSLE)) || ((SVSMLCTL & SVSLE) && (SVSMLCTL & SVSLFP)) ||

(!(SVSMLCTL & SVMLE)) || ((SVSMLCTL & SVMLE) && (SVSMLCTL & SVMLFP)))

{ // Next Check SVSH/SVMH settings to see if settings are affected by PMM15

if ((SVSMHCTL & SVSHE) && (!(SVSMHCTL & SVSHFP)))

{

if ((!(SVSMHCTL & SVSHMD)) || ((SVSMHCTL & SVSHMD) &&

(SVSMHCTL & SVSMHACE)))

return 1; // SVSH affected configurations

}

if ((SVSMHCTL & SVMHE) && (!(SVSMHCTL & SVMHFP)) && (SVSMHCTL & SVSMHACE))

return 1; // SVMH affected configurations

}

return 0; // SVS/M settings not affected by PMM15

}

}

2. If fast servicing of interrupts is required, add a 150us delay either in the interrupt service routine or before entry into LPM3/LPM4.

PMM17   *PMM Module*

Function   Vcore exceed maximum limit of 2.0V.

Description   If the device is switching between active mode and LPM2/3/4 with very high frequency, the core voltage of the device, VCORE, may rise incrementally until it is beyond 2.0 V, which is the maximum allowable limit for digital circuitry internal to the MSP430. This increase may remain undetected in an application with no functional impact but could potentially result in decreased endurance and increased wear over the lifetime of the device, because the digital circuitry is continually subjected to overvoltage.

The accumulation of Vcore affects only older lot trace codes of mentioned revisions.

Workaround   The VCORE accumulation is fixed by enabling the prolongation mechanism in software. The following lines of code need to be implemented before periodic execution of LPM-to-AM-LPM. It is recommended to execute the code at program start:

ASM code:

mov.w #0x9602, &0110h;

bis.w #0x0800, &0112h;

C code:

(unsigned int)(0x0110)=0x9602;

(unsigned int)(0x0112)|=0x0800;

The automatic prolongation mechanism is disabled with a BOR and must be enabled after each boot code execution.

For detailed background information, affected LTCs and possible workaround(s) see Vcore Accumulation documentation in SLAA505.

PMM18   *PMM Module*

Function   PMM supply overvoltage protection falsely triggers POR

Description   The PMM Supply Voltage Monitor (SVM) high side can be configured as overvoltage protection (OVP) using the SVMHOVPE bit of SVSMHCTL register. In this mode a POR should typically be triggered when DVCC reaches ~3.75V.

If the OVP feature of SVM high side is enabled going into LPM234, the SVM might trigger at DVCC voltages below 3.6V (~3.5V) within a few ns after wake-up. This can falsely cause an OVP-triggered POR. The OVP level is temperature sensitive during fail scenario and decreases with higher temperature (85 degC ~3.2V).

Workaround   Use Adaptive mode (SVMACE=1). The SVM high side is inactive in LPM234.

PMM20 *PMM Module*

Function Unexpected SVSL/SVML event during wakeup from LPM2/3/4 in fast wakeup mode

Description If PMM low side is configured to operate in fast wakeup mode, during wakeup from LPM2/3/4 the internal VCORE voltage can experience voltage drop below the corresponding SVSL and SVML threshold (recommendation according to User's Guide) leading to an unexpected SVSL/SVML event. Depending on PMM configuration, this event triggers a POR or an interrupt.

> **NOTE:** As soon the SVSL or the SVML is enabled in Normal performance mode the device is in slow wakeup mode and this erratum does not apply.
>
> In addition, this erratum has sporadic characteristic due to an internal asynchronous circuit. The drop of Vcore does not have an impact on specified device performance.

Workaround If SVSL or SVML is required for application (to observe external disruptive events at Vcore pin) the slow wakeup mode has to be used to avoid unexpected SVSL/SVML events. This is achieved if the SVSL or the SVML is configured in "Normal" performance mode (not disabled and not in "Full" Performance Mode).

PORT15 *PORT Module*

Function In-system debugging causes the PMALOCKED bit to be always set

Description The port mapping controller registers cannot be modified when single-stepping or halting at break points between a valid password write to the PMAPWD register and the expected lock of the port mapping (PMAP) registers. This causes the PMAPLOCKED bit to remain set and not clear as expected.

Note: This erratum only applies to in-system debugging and is not applicable when operating in free-running mode.

Workaround Do not single step through or place break points in the port mapping configuration section of code.

PORT16 *PORT Module*

Function GPIO pins are driven low during device start-up

Description During device start-up, all of the GPIO pins are expected to be in the floating input state. Due to this erratum, some of the GPIO pins are driven low for the duration of boot code execution during device start-up, if an external reset event (via the RST pin) interrupted the previous boot code execution. Boot code is always executed after a BOR, and the duration of this boot code execution is approximately 500us.

For a given device family, this erratum affects only the GPIO pins that are not available in the smallest package device family member, but that are present on its larger package variants.

Note: This erratum does not affect the smallest package device variants in a particular device family.

Workaround Ensure that no external reset is applied via the RST pin during boot code execution of the device, which occurs 1us after device start-up.

Note: System application needs to account for this erratum in to ensure there is no

increased current draw by the external components or damage to the external
components in the system during device start-up

PORT19   *PORT Module*

Function   Port interrupt may be missed on entry to LPMx.5

Description   If a port interrupt occurs within a small timing window (~1MCLK cycle) of the device entry
into LPM3.5 or LPM4.5, it is possible that the interrupt is lost. Hence this interrupt will not
trigger a wakeup from LPMx.5.

Workaround   None

RTC3   *RTC Module*

Function   Unreliable write to RTC register

Description   A write access to the RTC registers (SEC, MIN, HOUR, DATE, MON, YEAR, DOW) may
result in unexpected results. As a consequence the addressed register might not contain
the written data, or some data can be accidentally written to other RTC registers.

Workaround   Use the RTC library routines, available as F541x/F543x code examples on the MSP430
Code Examples page (www.ti.com/msp430 > Software > Code Examples), which use
carefully aligned MOV instructions. Library is listed as RTC_Workaround.zip and
includes both CCE and IAR example projects that show proper usage. Using this library,
full access to RTC registers is possible.

RTC6   *RTC Module*

Function   the step size of the RTC frequency adjustment is twice the specified size.

Description   The step size of the RTC frequency adjustment is =4ppm/-8ppm. This is twice the size
specified in the User's Guide.

For up calibration this results in a step size per step of 8ppm (1024 cycles) instead of
4ppm (512 cycles). For down calibration this results in a step size per step of 4ppm (512
cycles) instead of 2ppm (256 cycles).

Workaround   Half the calibration value written into RTCCAL register to compensate the doubled step
size.

SYS12   *SYS Module*

Function   Invalid ACCVIFG when DVcc in the range of 2.4 to 2.6V

Description   A Flash Access Violation Interrupt Flag (ACCVIFG) may be triggered by the Voltage
Changed During Program Error bit (VPE) when DVcc is in the range of 2.4 to 2.6V.
Although this behavior is expected according to the user's guide, the VPE does not
signify an invalid flash operation has occured.

If the ACCVIE bit is set and a flash operation is executed in the affected voltage range,
an unnecessary interrupt is requested. The bootstrap loader also cannot be used to
execute write/erase flash operations in this voltage range, because it exits the flash
operation and returns an error on an ACCVIFG event.

Workaround   None

SYS16 *SYS Module*

Function Fast Vcc ramp after device power up may cause a reset

Description At initial power-up, after Vcc crosses the brownout threshold and reaches a constant level, an abrupt ramp of Vcc at a rate dV/dT > 1V/100us can cause a brownout condition to be incorrectly detected even though Vcc does not fall below the brownout threshold. This causes the device to undergo a reset.

Workaround Use a controlled Vcc ramp to power up the device.

SYS18 *SYS Module*

Function USB registers are unlocked and ACCVIFG is set at start-up

Description During device start-up, an incorrect line of code in the start-up code causes the USB registers to remain unlocked and causes an access violation, setting ACCVIFG bit.

In the BSL430_Low_Level_Init code, the following line of code accesses USBKEY (incorrect register address) instead of USBKEYPID, causing an access violation setting ACCVIFG bit, and leaving the USB registers unlocked.

mov.w #0x0000, &USBKEY ; lock USB

The correct line of code should read:

mov.w #0x0000, &USBKEYPID ; lock USB correctly

Note: This code does not run when using the JTAG debugger - the behavior only appears when running standalone.

Workaround 1. Load the latest version of the USB BSL from Custom BSL Download

OR

2. Load a non-USB or custom BSL

OR

3. Erase the BSL

TAB23 *TIMER_A/TIMER_B Module*

Function TAxR/TBxR read can be corrupted when TAxR/TBxR = TAxCCR0/TBxCCR0

Description When a timer in Up mode is stopped and the counter register (TAxR/TBxR) is equal to the TAxCCR0/TBxCCR0 value, a read of the TAR/TBR register may return an unexpected result.

Workaround 1. Use 'Up/Down' mode instead of 'Up' mode

OR

2. In 'Up' mode, use the timer interrupt instead of halting the counter and reading out the value in TAxR/TBxR

OR

3. When halting the timer counter in 'Up' mode, reinitialize the timer before starting to run again.

UCS6 *UCS Module*

Function	USCI source clock does not turn off in LPM3/4 when UART is idle
Description	The USCI clock source (ACLK/SMCLK) remains enabled in LPM3 and LPM4 when the USCI is configured in UART mode and the communication is idle (UCSWRST = 0 but no TX or RX currently executing). This is contrary to the expected automatic clock activation described in the User's Guide and can lead to higher current consumption in low power modes, depending on the oscillator that feeds ACLK / SMCLK.
Workaround	Use the oscillator that is already active in LPM3 (ACLK) to source the USCI and utilize the low-power baud rate generator (UCOS16 = 0). For UART baud rates where a fast SMCLK sourced by the internal DCO is required use LPM0 instead of LPM3.

UCS7   *UCS Module*

Function	DCO drifts when servicing short ISRs when in LPM0 or exiting active from ISRs for short periods of time
Description	The FLL uses two rising edges of the reference clock to compare against the DCO frequency and decide on the required modifications to the DCOx and MODx bits. If the device is in a low power mode with FLL disabled (LPM0 with DCO not sourcing ACLK/SMCLK or LPM2, LPM3, LPM4 where SCG1 bit is set) and enters a state which enables FLL (enter ISR from LPM0/LPM2 or exit active from ISRs) for a period less than 3x reference clock cycles, then the FLL will cause the DCO to drift.
	This occurs because the FLL immediately begins comparing an active DCO with its reference clock and making the respective modifications to the DCOx and MODx bits. If the FLL is not given sufficient time to capture a full reference clock cycle (2 x reference clock periods) and adjust accordingly (1 x reference clock period), then the DCO will keep drifting each time the FLL is enabled.
Workaround	(1) If DCO is not sourcing ACLK or SMCLK in the application, use LPM1 instead of LPM0 to make sure FLL is disabled when interrupt service routine is serviced.
	(2) When exiting active from ISRs, insert a delay of at least 3 x reference clock periods. To save on power budget, the 3 x reference clock periods could also be spent in LPM0 with TimerA or TimerB using ACLK/SMCLK sourced from DCO. This way, the FLL and DCO are still active in LPM0.

UCS9   *UCS Module*

Function	Digital Bypass mode prevents entry into LPM4
Description	When entering LPM4, if an external digital input applied to XT1 in HF mode or XT2 is not turned off, the PMM does not switch to low-current mode causing higher than expected power consumption.
Workaround	Before entering LPM4:
	(1) Switch to a clock source other than external bypass digital input.
	OR
	(2) Turn off external bypass mode (UCSCTL6.XT1BYPASS = 0).

UCS10   *UCS Module*

Function	Modulation causes shift in DCO frequency
Description	When the FLL is enabled, the DCO frequency can be tracked automatically by modifying

the DCOx and MODx bits. The MODx bits switch between the frequency selected by the DCO bits and the next-higher frequency set by (DCO + 1). The erroneous behavior is seen when the FLL is tracking close to a DCO step boundary and the MOD counter is expected to rollover, but instead the DCO bits increment and the MOD bits decrement. This causes the DCO to shift by up to 12% and remain at an increased frequency until approximately 15 REFCLK cycles have elapsed. The frequency reverts to the expected value immediately afterward.

For example, the modulator moves from DCOx = n and MODx = 31 to DCOx = n + 1

and MODx = 30, causing a large increase in the DCO frequency.

Applications could be impacted as follows:

When using the DCO frequency for asynchronous serial communication and timer operation, the effect can be seen as corrupted data or incorrect timing events.

Workaround   (1) Turn off the FLL.

Or

(2) Implement a Software FLL, comparing the DCO frequency to a known reference such as REFO or LFXT1 using a timer capture and tuning the value of the DCO and MOD bits periodically.

Or

(3) Execute the following sequence in periodic intervals.

1. Disable peripherals sourced by the DCO such as UART and Timer.

2. Turn on the FLL.

3. Wait the worst case settling time of 32 X 32 X fFLLREFCLK to allow it to lock to the target frequency.

4. Turn off the FLL.

5. Compare the DCO frequency to a known reference such as REFO or LFXT1 using a timer capture.

- If the DCO frequency is higher than expected, repeat from step (2) until the

frequency reaches to the expected range.

- Else proceed with code execution.

See the application report UCS10 Guidance SLAA489 for more detailed information regarding working with this erratum. This erratum does not affect proper operation of the CPU when MCLK = DCO/FLL and is set to the maximum clock frequency specified in the device datasheet.

UCS11   *UCS Module*

Function   Modifying UCSCTL4 clock control register triggers an erroneous clock source request

Description   Changing the SELM/SELS/SELA bits in the UCSCTL4 register might trigger the respective clocks to select an incorrect clock source which requests the XT1/XT2 clock. If the crystals are not present at XT1/XT2 or present but not yet configured in the application firmware, then the respective XT1/XT2 fault flag is falsely set.

Workaround   Clear all the fault flags in UCSCTL7 register once after changing any of the SELM/SELS/SELA bits in the UCSCTL4 register.

USB8   *USB Module*

Function	USB PLL may fail to initialize when DCO is not used
Description	If the DCO is not used or does not source any clock in the system, the PLL may not initialize properly, leading to a "bus error" NMI when the CPU attempts to access USB memory. Under certain conditions, the PLL requires the DCO to be active in order to initialize properly. If any of the system clocks (MCLK, SMCLK, ACLK) are derived from the DCO during PLL initialization, this condition is satisfied and the PLL initializes in a stable manner.
Workaround	- Configure one of the system clocks to use the DCO for normal operation. (In most applications, it's convenient to source MCLK from the DCO.)

or

- If it's necessary to use a clock configuration that doesn't include the DCO, then briefly activate it, then de-activate it. This should be done after setting the UPLLEN bit but before polling the USBPLLIR flags. The code below shows a simplified PLL initialization procedure. (In the MSP430 USB API Stacks, this is performed in USB_enable().)

USBPLLCTL |= UPLLEN; // Enable PLL

// Add the following code to enable the DCO, then immediately revert back

// to the original clock settings for the application. This is long

// enough to allow PLL initialization to proceed. The actual UCSCTL4

// settings depend on the application's clock requirements.

UCSCTL4 = SELA__REFOCLK + SELS__XT2CLK + SELM__DCOCLK; // Enable the DCO

UCSCTL4 = SELA__REFOCLK + SELS__XT2CLK + SELM__XT2CLK; // Revert back

//Wait for the PLL to settle

do {

USBPLLIR = 0x0000; // Clear the flags

for (i =0; i < 400; i++); // Wait for flags to set if not stable yet

}while (USBPLLIR != 0);

// PLL is now stable

USB9	***USB Module***
Function	VBUS detection may fail after powerup
Description	In rare cases, some USB-equipped MSP430 devices may experience a failure in the bandgap that aids in detecting the presence of 5 V on the VBUS pin. Two primary effects of this are:

- The USBBGVBV bit fails to show the presence of a valid voltage on the VBUS pin.

and

- The USB LDOs fail to start.

Workaround	This error state can be "reset" by clearing all of the bits in the USBPWRCTL register, which disables the USB LDOs, among other actions. The bits can then be set again normally, and the device functions properly.

This has been added to the USB_Init() function in v3.10 and later of the MSP430 USB API. Therefore, this problem is automatically addressed in applications that use the API.

However, if the integrated 3.3-V USB LDO (the output of the VUSB pin) is used to power

the device's DVCC pin, as in many bus-powered applications, and if the rare bandgap error occurs, the CPU fails to power up, because the USB LDO fails to operate. The problem might be resolved by cycling power to the VBUS pin; for example, if the end user responds to the failure by unplugging and replugging the USB cable. The bandgap failure is also known to occur more often with slow DVCC ramps > 200 ms; for example, when there is excessive capacitance on the DVCC pin, in excess of what the USB specification allows. However, the only sure way to prevent the problem from occurring in the first place is to avoid making DVCC power reliant on VUSB.

USB10   *USB Module*

Function   USB interface may begin to babble when a rare timing event occurs between the USB host and MSP430 software execution

Description   When the host sends a SETUP packet for an IN transaction, the SETUPIFG bit always gets set by hardware, and the USB ISR is triggered. While SETUPIFG is high, the host's attempts to continue the transaction with IN packets are automatically NAKed.

When the SETUP packet has been decoded and the IN data prepared, the USB ISR clears the SETUPIFG bit. But if it happens to do so within the 2nd CRC bit of an IN packet from the host, the USB module enters an errant state and can begin to "babble" (endless transmission to the host, irrespective of the protocol). The errant state can be cleared by resetting the module with the USB_EN bit; but there's no way for software to reliably detect the condition.

Since the 2nd CRC bit is only an 83ns window, the problem is extremely rare. However, since the timing of IN packets relative to their preceding SETUP packets can vary according to the host's timing, there's no way to ensure for certain that it will never happen.

Workaround   If the problem behavior occurs, and if the MSP430 is bus-powered, the user may naturally unplug/re-plug the device's USB connection. If this occurs, the behavior will be corrected because power to the MSP430 will be cycled. After this, it's unlikely the problem will occur again soon, since the failure is usually rare.

The behavior can be prevented altogether by clearing the UBME bit immediately before clearing SETUPIFG, and setting it again immediately after:

```
        USBIEPCNF_0 &= ~EPCNF_UBME; // Clear ME to gate off SETUPIFG
clear event
        USBOEPCNF_0 &= ~EPCNF_UBME; // Clear ME to gate off SETUPIFG
clear event
        USBIFG &= ~SETUPIFG; // clear the interrupt bit
        USBIEPCNF_0 |= EPCNF_UBME; // Set ME to continue with normal
operation
        USBOEPCNF_0 |= EPCNF_UBME; // Set ME to continue with normal
operation
```

This workaround is reliable and effective. However, as a side effect, it results in the creation of orphan tokens on the USB interface. Although the workaround is field-tested, and no problems have been reported with these orphan packets, it is recommended to use the workaround only if the errata behavior is problematic for the application in question.

USCI26   *USCI Module*

Function   Tbuf parameter violation in I2C multi-master mode

Description   In multi-master I2C systems the timing parameter Tbuf (bus free time between a stop condition and the following start) is not guaranteed to match the I2C specification of

4.7us in standard mode and 1.3us in fast mode. If the UCTXSTT bit is set during a running I2C transaction, the USCI module waits and issues the start condition on bus release causing the violation to occur.

Note: It is recommended to check if UCBBUSY bit is cleared before setting UCTXSTT=1.

Workaround   None

USCI30   *USCI Module*

Function   I2C mode master receiver / slave receiver

Description   When the USCI I2C module is configured as a receiver (master or slave), it performs a double-buffered receive operation. In a transaction of two bytes, once the first byte is moved from the receive shift register to the receive buffer the byte is acknowledged and the state machine allows the reception of the next byte.

If the receive buffer has not been cleared of its contents by reading the UCBxRXBUF register while the 7th bit of the following data byte is being received, an error condition may occur on the I2C bus. Depending on the USCI configuration the following may occur:

1) If the USCI is configured as an I2C master receiver, an unintentional repeated start condition can be triggered or the master switches into an idle state (I2C communication aborted). The reception of the current data byte is not successful in this case.

2) If the USCI is configured as I2C slave receiver, the slave can switch to an idle state stalling I2C communication. The reception of the current data byte is not successful in this case. The USCI I2C state machine will notify the master of the aborted reception with a NACK.

Note that the error condition described above occurs only within a limited window of the 7th bit of the current byte being received. If the receive buffer is read outside of this window (before or after), then the error condition will not occur.

Workaround   a) The error condition can be avoided altogether by servicing the UCBxRXIFG in a timely manner. This can be done by (a) servicing the interrupt and ensuring UCBxRXBUF is read promptly or (b) Using the DMA to automatically read bytes from receive buffer upon UCBxRXIFG being set.

OR

b) In case the receive buffer cannot be read out in time, test the I2C clock line before the UCBxRXBUF is read out to ensure that the critical window has elapsed. This is done by checking if the clock line low status indicator bit UCSCLLOW is set for atleast three USCI bit clock cycles i.e. 3 X t(BitClock).

Note that the last byte of the transaction must be read directly from UCBxRXBUF. For all other bytes follow the workaround:

Code flow for workaround

(1) Enter RX ISR for reading receiving bytes

(2) Check if UCSCLLOW.UCBxSTAT == 1

(3) If no, repeat step 2 until set

(4) If yes, repeat step 2 for a time period > 3 x t (BitClock) where t (BitClock) = 1/ f (BitClock)

(5) If window of 3 x t(BitClock) cycles has elapsed, it is safe to read UCBxRXBUF

USCI31   *USCI Module*

Function

Framing Error after USCI SW Reset (UCSWRST)

Description

While receiving a byte over USCI-UART (with UCBUSY bit set), if the application resets the USCI module (software reset via UCSWRST), then a framing error is reported for the next receiving byte.

Workaround

1. If possible, do not reset USCI-UART during an ongoing receive operation; that is, when UCBUSY bit is set.

2. If the application software resets the USCI module (via the UCSWRST bit) during an ongoing receive operation, then set and reset the UCSYNC bit before releasing the software USCI reset.

Workaround code sequence:

bis #UCSWRST, &UCAxCTL1 ; USCI SW reset

;Workaround begins

bis #UCSYNC, &UCAxCTL0 ; set synchronous mode

bic #UCSYNC, &UCAxCTL0 ; reset synchronous mode

;Workaround ends

bic #UCSWRST, &UCAxCTL1 ; release USCI reset

USCI35   *USCI Module*

Function

Violation of setup and hold times for (repeated) start in I2C master mode

Description

In I2C master mode, the setup and hold times for a (repeated) START, $t_{SU,STA}$ and $t_{HD,STA}$ respectively, can be violated if SCL clock frequency is greater than 50kHz in standard mode (100kbps). As a result, a slave can receive incorrect data or the I2C bus can be stalled due to clock stretching by the slave.

Workaround

If using repeated start, ensure SCL clock frequencies is < 50kHz in I2C standard mode (100 kbps).

WDG4   *WDT Module*

Function

The WDT failsafe can be disabled

Description

The UCS is capable of masking clock requests (ACLK, SMCLK, MCLK) from peripheral modules; see request enable (REQEN) bits in the UCS control register, UCSCTL8.

The clock request logic of the UCS is used by the WDT module to ensure a fail-safe clock source in all low-power modes. Therefore, de-asserting the request enable bit of the watchdog clock source (xCLKREQEN = 0) allows the respective clock to be disabled upon entry into a low-power mode. Without an active clock source, the WDT timer stops incrementing and a watchdog event will not occur.

Workaround

None

5   Document Revision History

Changes from family erratasheet to device specific erratasheet.

1. Errata ADC26 was removed
2. Errata RTC4 was removed
3. Errata SYS10 was removed
4. Errata USCI12_5510 was removed
5. Errata SYS12 was added
6. Revision A was removed
7. Revision B was removed
8. PMM10 is now impacting silicon revision C
9. RGC64 package markings have been updated
10. ZQE80 package markings have been updated
11. RGZ48 package markings have been updated

Changes from device specific erratasheet to document Revision A.

1. Errata PORT19 was added to the errata documentation.
2. Errata PMM18 was added to the errata documentation.
3. Errata RTC6 was added to the errata documentation.
4. Errata SYS18 was added to the errata documentation.

Changes from document Revision A to Revision B.

1. Errata DMA10 was added to the errata documentation.
2. Errata BSL7 was added to the errata documentation.
3. Errata RTC3 was added to the errata documentation.

Changes from document Revision B to Revision C.

1. Errata BSL6 was added to the errata documentation.
2. DMA10 Description was updated.
3. DMA10 Function was updated.

Changes from document Revision C to Revision D.

1. DMA10 Description was updated.
2. BSL6 Workaround was updated.
3. MPY1 Description was updated.
4. Errata EEM23 was added to the errata documentation.
5. Errata CPU43 was added to the errata documentation.

Changes from document Revision D to Revision E.

1. Silicon Revision E was added to the errata documentation.
2. SYS16 Description was updated.
3. CPU43 Description was updated.
4. Device TLV Hardware Revision information added to erratasheet.

Changes from document Revision E to Revision F.

1. Errata PMM20 was added to the errata documentation.
2. Errata USCI35 was added to the errata documentation.

Changes from document Revision F to Revision G.

1. BSL7 Workaround was updated.
2. BSL7 Function was updated.
3. Errata USB10 was added to the errata documentation.

IMPORTANT NOTICE

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

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

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

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

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

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

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

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

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

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

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

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