MSP430F5437 Device Erratasheet

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

1 Functional Errata Revision History

Errata impacting device's operation, function or parametrics.

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

Errata Number	Rev N	Rev L
ADC20	✓	✓
ADC21	✓	✓
ADC24	✓	✓
ADC25	✓	✓
ADC27	✓	✓
ADC42	✓	✓
ADC69	✓	✓
CPU36	✓	✓
CPU37	✓	✓
CPU47	✓	✓
DMA4	✓	✓
DMA5	✓	✓
DMA6	✓	✓
DMA7	✓	✓
DMA8	✓	✓
DMA10	✓	✓
FLASH28		✓
FLASH29	✓	✓
FLASH30	✓	✓
FLASH31	✓	✓
FLASH32	✓	✓
FLASH34	✓	✓
FLASH37	✓	✓
MPY1	✓	✓
PMM1	✓	✓
PMM2	✓	✓
PMM3	✓	✓
PMM5	✓	✓
PMM6	✓	✓
PMM7	✓	✓
PMM8	✓	✓
PMM9	✓	✓

Copyright © 2012–2019, Texas Instruments Incorporated

Errata Number	Rev N	Rev L
PMM10	✓	✓
PMM11	✓	✓
PMM12	✓	✓
PMM14	✓	✓
PMM15	✓	✓
PMM18	✓	✓
PMM20	✓	✓
PORT13	✓	✓
PORT14	✓	✓
PORT16	✓	✓
PORT19	✓	✓
RTC2	✓	✓
RTC3	✓	✓
RTC6	✓	✓
SYS1	✓	✓
SYS2	✓	✓
SYS3	✓	✓
SYS4	✓	✓
SYS5	✓	✓
SYS6	✓	✓
SYS7	✓	✓
SYS8		✓
SYS9	✓	✓
SYS12	✓	✓
SYS13	✓	✓
SYS16	✓	✓
TA20	✓	✓
TAB23	✓	✓
TB20	✓	✓
TB21	✓	✓
UCS1	✓	✓
UCS2	✓	✓
UCS4	✓	✓
UCS5	✓	✓
UCS6	✓	✓
UCS7	✓	✓
UCS9	✓	✓
UCS10	✓	✓
UCS11	✓	✓
UCS13	✓	✓
USCI26	✓	✓
USCI30	✓	✓
USCI31	✓	✓
USCI34	✓	✓
USCI35	✓	✓
USCI39	✓	✓
USCI40	✓	✓
WDG4	✓	✓

Copyright © 2012–2019, Texas Instruments Incorporated

2 Preprogrammed Software Errata Revision History

Errata impacting pre-programmed software into the silicon by Texas Instruments.

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

Errata Number	Rev N	Rev L
JTAG20	✓	✓

3 Debug only Errata Revision History

Errata only impacting debug operation.

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

Errata Number	Rev N	Rev L
EEM4	✓	✓
EEM5	✓	✓
EEM6	✓	✓
EEM8	✓	✓
EEM9	✓	✓
EEM11	✓	✓
EEM13	✓	✓
EEM14	✓	✓
EEM16	✓	✓
EEM17	✓	✓
EEM19	✓	✓
EEM21	✓	✓
EEM23	✓	✓
JTAG17	✓	✓
JTAG26	✓	✓
JTAG27	✓	✓

4 Fixed by Compiler Errata Revision History

Errata completely resolved by compiler workaround. Refer to specific erratum for IDE and compiler versions with workaround.

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

Errata Number	Rev N	Rev L
CPU15	✓	✓
CPU16	✓	✓
CPU18	✓	✓
CPU20	✓	✓
CPU21	✓	✓
CPU22	✓	✓
CPU23	✓	✓
CPU24	✓	✓
CPU25	✓	✓

Copyright © 2012–2019, Texas Instruments Incorporated

Errata Number	Rev N	Rev L
CPU26	✓	✓
CPU27	✓	✓
CPU28	✓	✓
CPU29	✓	✓
CPU30	✓	✓
CPU31	✓	✓
CPU32	✓	✓
CPU33	✓	✓
CPU34	✓	✓
CPU35	✓	✓
CPU39	✓	✓
CPU40	✓	✓

Refer to the following MSP430 compiler documentation for more details about the CPU bugs workarounds.

TI MSP430 Compiler Tools (Code Composer Studio IDE)

- MSP430 Optimizing C/C++ Compiler: Check the --silicon_errata option
- MSP430 Assembly Language Tools

MSP430 GNU Compiler (MSP430-GCC)

- MSP430 GCC Options: Check -msilicon-errata= and -msilicon-errata-warn= options
- MSP430 GCC User's Guide

IAR Embedded Workbench

- IAR workarounds for msp430 hardware issues

Copyright © 2012–2019, Texas Instruments Incorporated

5 Package Markings

PN80

LQFP (PN), 80 Pin



\# = Die revision
○ = Pin 1 location
N = Lot trace code



\# = Die revision
○ = Pin 1 location
N = Lot trace code

6 Memory-Mapped Hardware Revision (TLV Structure)

Die Revision	TLV Hardware Revision
Rev N	10h
Rev L	10h

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

Copyright © 2012–2019, Texas Instruments Incorporated

7 Detailed Bug Description

ADC20 *ADC12_A Module*

Category Functional

Function ADC12DF does not work as described in the user's guide

Description The conversion data format (ADC12DF bit setting) must be selected before a conversion. Switching the conversion data format bit after a conversion has begun does not deliver results in the newly selected format.

Workaround Change the conversion data format before starting the conversion.

ADC21 *ADC12_A Module*

Category Functional

Function ADC12ENC is not working correctly

Description Resetting the ADC12ENC bit during an active conversion or within a sequence can lead to an invalid result for both the current and next ADC12 conversions. Subsequent ADC12 conversions are valid.

Workaround Resetting the ADC12ENC bit during an active conversion or within a sequence should be avoided. If the ADC12ENC bit is reset during an active conversion, this and the next conversion result should be discarded.

ADC24 *ADC12_A Module*

Category Functional

Function Unexpected ADC12 current draw when ADC12ENC = 1

Description When set, the ADC12ENC bit issues a clock request to the selected source clock, even before the conversion trigger. This causes some extra current consumption, depending on the selected clock.

Workaround None.

ADC25 *ADC12_A Module*

Category Functional

Function Write to ADC12CTL0 triggers ADC12 when CONSEQ = 00

Description If ADC conversions are triggered by the Timer_B module and the ADC12 is in single-channel single-conversion mode (CONSEQ = 00), ADC sampling is enabled by write access to any bit(s) in the ADC12CTL0 register. This is contrary to the expected behavior that only the ADC12 enable conversion bit (ADC12ENC) triggers a new ADC12 sample.

Workaround When operating the ADC12 in CONSEQ=00 and a Timer_B output is selected as the sample and hold source, temporarily clear the ADC12ENC bit before writing to other bits in the ADC12CTL0 register. The following capture trigger can then be re-enabled by setting ADC12ENC = 1.

Copyright © 2012–2019, Texas Instruments Incorporated

ADC27 *ADC12_A Module*

Category Functional

Function Integral and differential non-linearity exceed specifications

Description The ADC12_A integral and differential non-linearity may exceed the limits specified in the data sheet under the following conditions:

- If the internal voltage reference generator is used

and

- If the reference voltage is not buffered off-chip

and

- If fADC12CLK > 2.7 MHz

The non-linearity can be up to tens of LSBs. This is due to the internal reference buffer providing insufficient drive for the switched capacitor array of the ADC12_A.

Workaround (1) Turn on the output of the internal voltage reference to increase the drive strength of the reference to the ADC_12 core:

Set ADC12REFON bit in ADC12CTL0 = 1 and Set ADC12REFOUT bit in ADC12CTL2 = 1

or

(2) Ensure fADC12CLK < 2.7 MHz. Depending on the frequency of the source of fADC12CLK (ACLK, MCLK, SMCLK, or MODOSC), select the divider bits accordingly.

- If fADC12CLK = MODOSC

(ADC12OSC) ADC12CTL1 |= ADC12DIV_1; // Divide clock by 2

- If fADC12CLK = ACLK/SMCLK/MCLK > 2.7 MHz.

Use ADC12DIVx and/or ADC12PDIVx bits to reduce the selected clock frequency to between 0.45 MHz and 2.7 MHz.

ADC42 *ADC12_A Module*

Category Functional

Function ADC stops converting when successive ADC is triggered before the previous conversion ends

Description Subsequent ADC conversions are halted if a new ADC conversion is triggered while ADC is busy. ADC conversions are triggered manually or by a timer. The affected ADC modes are:

- sequence-of-channels

- repeat-single-channel

- repeat-sequence-of-channels (ADC12CTL1.ADC12CONSEQx)

In addition, the timer overflow flag cannot be used to detect an overflow (ADC12IFGR2.ADC12TOVIFG).

Workaround 1. For manual trigger mode (ADC12CTL0.ADC12SC), ensure each ADC conversion is completed by first checking ADC12CTL1.ADC12BUSY bit before starting a new conversion.

Copyright © 2012–2019, Texas Instruments Incorporated

2. For timer trigger mode (ADC12CTL1.ADC12SHP), ensure the timer period is greater than the ADC sample and conversion time.

To recover the conversion halt:

1. Disable ADC module (ADC12CTL0.ADC12ENC = 0 and ADC12CTL0.ADC12ON = 0)

2. Re-enable ADC module (ADC12CTL0.ADC12ON = 1 and ADC12CTL0.ADC12ENC = 1)

3. Re-enable conversion

ADC69 *ADC12_A Module*

Category Functional

Function ADC stops operating if ADC clock source is changed from SMCLK to another source while SMCLKOFF = 1.

Description When SMCLK is used as the clock source for the ADC (ADC12CTL1.ADC12SSELx = 11) and CSCTL4.SMCLKOFF = 1, the ADC will stop operating if the ADC clock source is changed by user software (e.g. in the ISR) from SMCLK to a different clock source. This issue appears only for the ADC12CTL1.ADC12DIVx settings /3/5/7. The hang state can be recovered by PUC/POR/BOR/Power cycle.

Workaround 1. Set CSCTL4.SMCLKOFF = 0 before switch ADC clock source.

OR

2. Only use ADC12CTL1.ADC12DIVx as /1, /2, /4, /6, /8

CPU15 *CPUXv2 Module*

Category Compiler-Fixed

Function Modifying the PC behaves differently than in previous devices.

Description When using instructions with immediate or indirect addressing mode to modify the PC, a different value compared to previous devices must be added to get to the same destination.

Example:

Previous device (MSP430F4619)

label_1 ADD.W #Branch1-label_1-4h,PC

MSP430F5438:

label_1 ADD.W #Branch1-label_1-2h,PC

NOTE: The MOV instruction is not affected

Workaround 1) Add NOP after the PC-modifying instruction

or

2) Change the offset value in software

Refer to the table below for compiler-specific fix implementation information.

Copyright © 2012–2019, Texas Instruments Incorporated

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v6.40.1 or later	User is required to add the option below to the linker. Linker:-D?CPU15_OFFSET=2
TI MSP430 Compiler Tools (Code Composer Studio)	v4.0.x or later	User is required to add the compiler or assembler flag option below. --silicon_errata=CPU15
MSP430 GNU Compiler (MSP430-GCC)	Not affected	

CPU16 *CPUXv2 Module*

Category Compiler-Fixed

Function Indexed addressing with instructions calla, mova and bra.

Description With indexed addressing mode and instructions calla, mova, and bra, it is not possible to reach memory above 64k if the register content is < 64k.

Example: Assume R5 = FFFEh. The instruction calla 0004h(R5) will result in a 20-bit call of address 0002h instead of 10002h.

Workaround - Use different addressing mode to reach memory above 64k.

- First use adda [index],[Rx] to calculate address in upper memory and then do a calla [Rx]

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number
IAR Embedded Workbench	IAR EW430 v6.30.1 or later
TI MSP430 Compiler Tools (Code Composer Studio)	Fix not available
MSP430 GNU Compiler (MSP430-GCC)	Fix not available

CPU18 *CPUXv2 Module*

Category Compiler-Fixed

Function LPM instruction can corrupt PC/SR registers

Description The PC and SR registers have the potential to be corrupted when:

- An instruction using register, absolute, indexed, indirect, indirect auto-increment, or symbolic mode is used to set the LPM bits AND (e.g. BIS &xyh, SR)

- This instruction is followed by a CALL or CALLA instruction

Upon servicing an interrupt service routine, the program counter (PC) is pushed twice onto the stack instead of the correct operation where the PC, then the SR registers are pushed onto the stack. This corrupts the SR and possibly the PC on RETI from the ISR.

Workaround Insert a NOP or __no_operation() intrinsic function between the instruction to enter low power mode and the CALL or CALLA instruction.

Refer to the table below for compiler-specific fix implementation information.

Copyright © 2012–2019, Texas Instruments Incorporated

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v6.20.1 until v6.40	User is required to add the compiler or assembler flag option below. --hw_workaround=nop_after_lpm
IAR Embedded Workbench	IAR EW430 v6.40 or later	Workaround is automatically enabled
TI MSP430 Compiler Tools (Code Composer Studio)	v4.0 or later	User is required to add the compiler or assembler flag option below. --silicon_errata=CPU18
MSP430 GNU Compiler (MSP430-GCC)	Not affected	

CPU20 *CPUXv2 Module*

Category Compiler-Fixed

Function An unexpected Vacant Memory Access Flag (VMAIFG) can be triggered due to the CPU autoincrement of the MAB+2 outside the range of a valid memory block.

Description The VMAIFG can be triggered under the following conditions:

1. If an interrupt is requested, fetched by the CPU, but lost before excution of the interrupt service routine.

OR

2. If a PC-modifying instruction (e.g. - ret, push, call, pop, jmp, br) is fetched from the last address of a section of memory (e.g.- FLASH, RAM) that is not continguous to a higher, valid section on the memory map.

Workaround For case 1 - None.

For case 2 - If code is affected, edit the linker command file to make the last four bytes of affected memory sections unavailable.

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number
IAR Embedded Workbench	IAR EW430 v6.40 or later
TI MSP430 Compiler Tools (Code Composer Studio)	Not affected
MSP430 GNU Compiler (MSP430-GCC)	Not affected

CPU21 *CPUXv2 Module*

Category Compiler-Fixed

Function Using POPM instruction on Status register may result in device hang up

Description When an active interrupt service request is pending and the POPM instruction is used to set the Status Register (SR) and initiate entry into a low power mode , the device may hang up.

Workaround None. It is recommended not to use POPM instruction on the Status Register.

Refer to the table below for compiler-specific fix implementation information.

Copyright © 2012–2019, Texas Instruments Incorporated

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	Not affected	
TI MSP430 Compiler Tools (Code Composer Studio)	v4.0.x or later	User is required to add the compiler or assembler flag option below. --silicon_errata=CPU21
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 167 or later	

CPU22 *CPUXv2 Module*

Category Compiler-Fixed

Function Indirect addressing mode with the Program Counter as the source register may produce unexpected results

Description When using the indirect addressing mode in an instruction with the Program Counter (PC) as the source operand, the instruction that follows immediately does not get executed.

For example in the code below, the ADD instruction does not get executed.

```
mov @PC, R7
add #1h, R4
```

Workaround Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	Not affected	
TI MSP430 Compiler Tools (Code Composer Studio)	v4.0.x or later	User is required to add the compiler or assembler flag option below. --silicon_errata=CPU22
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 167 or later	

CPU23 *CPUXv2 Module*

Category Compiler-Fixed

Function Rotate instruction does not function as expected

Description When repeated rotate instructions (rrcm, rram, rrum and rlam) are applied on the Program Counter(PC), unexpected instruction execution may occur.

Workaround Insert a NOP instruction between sequential rotate instructions performed on the PC register.

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	Not affected	
TI MSP430 Compiler Tools (Code Composer Studio)	v4.0.x or later	User is required to add the compiler or assembler flag option below. --silicon_errata=CPU23
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 167 or later	

Copyright © 2012–2019, Texas Instruments Incorporated

CPU24 *CPUXv2 Module*

Category Compiler-Fixed

Function Program counter corruption following entry into low power mode

Description The program counter is corrupted when an interrupt event occurs in the time between (and including) one cycle before and one cycle after the CPUOFF bit is set in the status register. This failure occurs when the BIS instruction is followed by a CALL or CALLA instruction using the following addressing modes:

BIS &, SR

CALLA indir, indir autoinc, reg

BIS INDEX, SR

CALLA indir, indir autoinc, reg

BIS reg, SR

CALLA reg, indir, indir autoinc

NOTE: Due to the instruction emulation, the EINT instruction, as well as the __enable_interrupts() and possibly the __bis_SR_register() intrinsic functions are affected.

Workaround Insert a NOP instruction or __no_operation() intrinsic function call between the BIS and CALL or CALLA instructions.

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v6.20 until v6.40	User is required to add the compiler or assembler flag option below. --hw_workaround=nop_after_lpm
IAR Embedded Workbench	IAR EW430 v6.40 or later	Workaround is automatically enabled
TI MSP430 Compiler Tools (Code Composer Studio)	v4.1.3 or later	
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 167	

CPU25 *CPUXv2 Module*

Category Compiler-Fixed

Function DMA transfer does not execute during low power mode

Description If the following instruction sequence is used ([] denotes an addressing mode) to enter a low-power mode, and the DMARMWDIS bit is set, then DMA transfers are blocked for the duration of the low-power mode.

BIS [register|index|absolute|symbolic],SR

CALLA [register]

Workaround 1. Insert a NOP instruction or __no_operation() intrinsic function call between the BIS and CALLA instructions

OR

Copyright © 2012–2019, Texas Instruments Incorporated

2. Temporarily clear the DMARMWDIS bit when entering low power mode

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v6.20 until v6.40	User is required to add the compiler or assembler flag option below. --hw_workaround=nop_after_lpm
IAR Embedded Workbench	IAR EW430 v6.40 or later	Workaround is automatically enabled
TI MSP430 Compiler Tools (Code Composer Studio)	v4.1.3 or later	
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 167	

CPU26 *CPUXv2 Module*

Category Compiler-Fixed

Function CALL SP instruction does not behave as expected

Description The intention of the CALL SP instruction is to execute code from the stack, instead it skips the first piece of data (instruction) on the stack. The second piece of data at SP+2 is used as the first executable instruction.

Workaround Write the op code for a NOP as the first instruction on the stack. Begin the intended subroutine at address SP + 2.

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number
IAR Embedded Workbench	Not affected
TI MSP430 Compiler Tools (Code Composer Studio)	v4.1.3 or later
MSP430 GNU Compiler (MSP430-GCC)	Not affected

CPU27 *CPUXv2 Module*

Category Compiler-Fixed

Function Program Counter (PC) is corrupted during the context save of a nested interrupt

Description When a low power mode is entered within an interrupt service routine that has enabled nested interrupts (by setting the GIE bit), and the instruction that sets the low power mode is directly followed by a RETI instruction, an incorrect value of PC + 2 is pushed to the stack during the context save. Hence, the RETI instruction is not executed on return from the nested interrupt and the PC becomes corrupted.

Workaround Insert a NOP or __no_operation() intrinsic function between the instruction that sets the lower power mode and the RETI instruction.

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v6.20 until v6.40	User is required to add the compiler or assembler flag option below. --hw_workaround=nop_after_lpm

Copyright © 2012–2019, Texas Instruments Incorporated

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v6.40 or later	Workaround is automatically enabled
TI MSP430 Compiler Tools (Code Composer Studio)	v4.1.3 or later	
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 167	

CPU28 *CPUXv2 Module*

Category Compiler-Fixed

Function PC is corrupted when using certain extended addressing mode combinations

Description An extended memory instruction that modifies the program counter executes incorrectly when preceded by an extended memory write-back instruction under the following conditions:

First instruction:

2-operand instruction, extended mode using (register,index), (register,absolute), OR (register,symbolic) addressing modes

Second instruction:

2-operand instruction, extended mode using the (indirect,PC), (indirect auto-increment,PC), OR (indexed [with ind 0], PC) addressing modes

Example:

BISX.A R6,&AABCD

ANDX.A @R4+,PC

Workaround 1. Insert a NOP or a __no_operation() intrinsic function between the two instructions

Or

2. Do not use an extended memory instruction to modify the PC

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number
IAR Embedded Workbench	Not affected
TI MSP430 Compiler Tools (Code Composer Studio)	v4.1.3 or later
MSP430 GNU Compiler (MSP430-GCC)	Not affected

CPU29 *CPUXv2 Module*

Category Compiler-Fixed

Function Using a certain instruction sequence to enter low power mode(s) affects the instruction width of the first instruction in an NMI ISR

Description If there is a pending NMI request when the CPU enters a low power mode (LPMx) using an instruction of Indexed source addressing mode, and that instruction is followed by a 20-bit wide instruction of Register source and destination addressing modes, the first instruction of the ISR is executed as a 20-bit wide instruction.

Example:

main:

...

Copyright © 2012–2019, Texas Instruments Incorporated

MOV.W [indexed],SR ; Enter LPMx

MOVX.A [register],[register] ; 20-bit wide instruction

...

ISR_start:

MOV.B [indexed],[register] ; ERROR - Executed as a 20-bit instruction!

Note: [] indicates addressing mode

Workaround

1. Insert a NOP or a __no_operation() intrinsic function following the instruction that enters the LPMx using indexed addressing mode

OR

2. Use a NOP or a __no_operation() intrinsic function as first instruction in the ISR

OR

3. Do not use the indexed mode to enter LPMx

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v6.20 until v6.40	User is required to add the compiler or assembler flag option below. --hw_workaround=nop_after_lpm
IAR Embedded Workbench	IAR EW430 v6.40 or later	Workaround is automatically enabled
TI MSP430 Compiler Tools (Code Composer Studio)	v4.1.3 or later	
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 167	

CPU30 *CPUXv2 Module*

Category Compiler-Fixed

Function ADDA, SUBA, CMPA [immediate],PC behave as if immediate value were offset by -2

Description

The extended address instructions ADDA, SUBA, CMPA in immediate addressing mode are represented by 4-bytes of opcode (see the MSP430F5xx Family User's Guide MSP430F5xx Family User's Guide for more details). In cases where the program counter (PC) is used as the destination register only 2 bytes of the current instruction's 4-byte opcode are accounted for in the PC value. The resulting operation executes as if the immediate value were offset by a value of -2.

Ideal: ADDA #Immediate-4, PC

...is equivalent to...

Actual: ADDA #Immediate-2, PC

** NOTE: The MOV instruction is not affected **

Workaround

1) Modify immediate value in software to account for the offset of 2.

OR

2) Use extended 20-bit instructions (addx.a, subx.a, cmpx.a).

Refer to the table below for compiler-specific fix implementation information.

Copyright © 2012–2019, Texas Instruments Incorporated

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v5.30 or later	IDE-based usage enables the workaround automatically. When using the command line, user is required to add the option below: Linker: -D?CPU30_OFFSET=2
TI MSP430 Compiler Tools (Code Composer Studio)	v4.0 or later	
MSP430 GNU Compiler (MSP430-GCC)	Not affected	

CPU31 *CPUXv2 Module*

Category Compiler-Fixed

Function SP corruption

Description When the instruction PUSHX.A is executed using the indirect auto-increment mode with the stack pointer (SP) as the source register [PUSHX.A @SP+] the SP is consequently corrupted. Instead of decrementing the value of the SP by four, the value of the SP is replaced with the data pointed to by the SP previous to the PUSHX.A instruction execution.

Workaround None. Note that compilers will not generate a PUSHX.A instruction that involves the SP.

Refer to the table below for compiler-specific information.

IDE/Compiler	Version Number
IAR Embedded Workbench	Not affected
TI MSP430 Compiler Tools (Code Composer Studio)	Not affected
MSP430 GNU Compiler (MSP430-GCC)	Not affected

CPU32 *CPUXv2 Module*

Category Compiler-Fixed

Function CALLA PC executes incorrectly

Description When the instruction CALLA PC is executed, the program counter (PC) that is pushed onto the stack during the context save is incorrectly offset by a value of -2.

Workaround None. Note that compilers will not generate a CALLA PC instruction.

Refer to the table below for compiler-specific information.

IDE/Compiler	Version Number
IAR Embedded Workbench	Not affected
TI MSP430 Compiler Tools (Code Composer Studio)	Not affected
MSP430 GNU Compiler (MSP430-GCC)	Not affected

CPU33 *CPUXv2 Module*

Category Compiler-Fixed

Function CALLA [indexed] may corrupt the program counter

Copyright © 2012–2019, Texas Instruments Incorporated

Description When the Stack Pointer (SP) is used as the destination register in the CALLA index(Rdst) instruction and is preceded by a PUSH or PUSHX instruction in any of the following addressing modes: Absolute, Symbolic, Indexed, Indirect register or Indirect auto increment, the "index" of the CALLA instruction is not sign extended to 20-bits and is always treated as a positive value. This causes the Program Counter to be set to a wrong address location when the index of the CALLA instruction represents a negative offset.

NOTE:

1. This erratum only applies when the instruction sequence is: PUSH or PUSHX followed by CALLA index(SP)

2. This erratum does not apply if the PUSH or PUSHX instruction is used in the Register or Immediate addressing mode

3. This erratum only applies when SP is used as the destination register in the CALLA index(Rdst) instruction

Workaround Place a "NOP" instruction in between the PUSH or PUSHX and the CALLA index(SP) instructions.

NOTE: This bug has no compiler impact as the compiler will not generate a CALLA instruction that uses indexed addressing mode with the SP.

Refer to the table below for compiler-specific information.

IDE/Compiler	Version Number
IAR Embedded Workbench	Not affected
TI MSP430 Compiler Tools (Code Composer Studio)	Not affected
MSP430 GNU Compiler (MSP430-GCC)	Not affected

CPU34 *CPUXv2 Module*

Category Compiler-Fixed

Function CPU may be halted if a conditional jump is followed by a rotate PC instruction

Description If a conditional jump instruction (JZ, JNZ, JC, JNC, JN, JGE, JL) is followed by an Address Rotate instruction on the PC (RRCM, RRAM, RLAM, RRUM) and the jump is not performed, the CPU is halted.

Workaround Insert a NOP between the conditional jump and the rotate PC instructions.

Refer to the table below for compiler-specific information.

IDE/Compiler	Version Number
IAR Embedded Workbench	Not affected
TI MSP430 Compiler Tools (Code Composer Studio)	Not affected
MSP430 GNU Compiler (MSP430-GCC)	Not affected

CPU35 *CPUXv2 Module*

Category Compiler-Fixed

Function Instruction BIT.B @Rx,PC uses the wrong PC value

Description The BIT(.B/.W) instruction in indirect register addressing mode uses the wrong PC value.

Copyright © 2012–2019, Texas Instruments Incorporated

This instruction is represented by 2 bytes of opcode. If the Program Counter (PC) is used as the destination register, the 2 opcode bytes of the current BIT instruction are not accounted for. The resulting operation executes the instruction using the wrong PC value and this affects the results in the Status Register (SR).

Workaround None. Note that compilers will not generate a BIT instruction that uses the PC as an operand.

Refer to the table below for compiler-specific information.

IDE/Compiler	Version Number
IAR Embedded Workbench	Not affected
TI MSP430 Compiler Tools (Code Composer Studio)	Not affected
MSP430 GNU Compiler (MSP430-GCC)	Not affected

CPU36 *CPUXv2 Module*

Category Functional

Function PC corruption when single-stepping through flash erase

Description When single-stepping over code that initiates an INFOD Flash memory erase, the program counter is corrupted.

Workaround None.

NOTE: This erratum applies to debug mode only.

CPU37 *CPUXv2 Module*

Category Functional

Function Wrong program trace display in the debugger while using conditional jump instructions

Description The state storage window displays an incorrect sequence of instructions when:

1. Conditional jump instructions are used to form a software loop

AND

2. A false condition on the jump breaks out of the loop

In such cases the trace buffer incorrectly displays the first instruction of the loop as the instruction that is executed immediately after exiting the loop.

Example:

Actual Code:

```
mov #4,R4
LABEL mov #1,R5
dec R4
jnz LABEL
mov #2,R6
nop
```

State Storage Window Displays:

```
LABEL mov #1,R5
```


Copyright © 2012–2019, Texas Instruments Incorporated

```
dec R4
jnz LABEL
mov #1,R5
nop
```

Workaround None

Note: This erratum affects the trace buffer display only. It does not affect code execution in debugger or free run mode

CPU39 *CPUXv2 Module*

Category Compiler-Fixed

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.

OR

Refer to the table below for compiler-specific fix implementation information.

Note that compilers implementing the fix may lead to double stack usage when RET/RETA follows the compiler-inserted NOP.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v5.60 until v6.20	User is required to add the compiler flag option below. --hw_workaround=CPU39 For the command line version add the following information Compiler: --core=430 Assembler:-v1
IAR Embedded Workbench	IAR EW430 v6.20 or later	Workaround is automatically enabled
TI MSP430 Compiler Tools (Code Composer Studio)	v4.1.3 or later	
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 167 or later	

CPU40 *CPUXv2 Module*

Category Compiler-Fixed

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

Copyright © 2012–2019, Texas Instruments Incorporated

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

@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.

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v5.51 or later	For the command line version add the following information Compiler: --hw_workaround=CPU40 Assembler:-v1
TI MSP430 Compiler Tools (Code Composer Studio)	v4.0.x or later	User is required to add the compiler or assembler flag option below. --silicon_errata=CPU40
MSP430 GNU Compiler (MSP430-GCC)	Not affected	

CPU47 *CPUXv2 Module*

Category Functional

Function An unexpected Vacant Memory Access Flag (VMAIFG) can be triggered

Description An unexpected Vacant Memory Access Flag (VMAIFG) can be triggered, if a PC-modifying instruction (e.g. - ret, push, call, pop, jmp, br) is fetched from the last addresses (last 4 or 8 byte) of a memory (e.g.- FLASH, RAM, FRAM) that is not contiguous to a higher, valid section on the memory map.

In debug mode using breakpoints the last 8 bytes are affected.

In free running mode the last 4 bytes are affected.

Workaround Edit the linker command file to make the last 4 or 8 bytes of affected memory sections unavailable, to avoid PC-modifying instructions on these locations.

Remaining instructions or data can still be stored on these locations.

DMA4 *DMA Module*

Category Functional

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.

Copyright © 2012–2019, Texas Instruments Incorporated

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).

DMA5 *DMA Module*

Category Functional

Function DMAxSZ is not updated correctly

Description

The total number of DMA transactions to be executed per transfer (stored initially in DMAxSZ) is updated under the following conditions:

1. A DMA transfer has been completed.

- DMAxSZ is refreshed by the value in the temporary register, T_Size, because it has been decremented to zero (ANY MODE).

- DMAxSZ is refreshed by the value in the temporary register, T_Size, because the DMA enable bit has transitioned from Hi -> Lo (ANY NON-REPEAT MODE).

2. A value is written into the DMAxSZ register by the application when the DMA state machine is in the Reset state (DMAEN = 0).

For all transfer modes the second condition is false. Writing to the DMAxSZ register (in an NMI service routine, for instance) immediately affects the DMAxSZ counter and the DMA will execute transfers until the DMAxSZ value decrements to zero, regardless of the current DMA state or DMAxSZ value.

Workaround The application should ensure that the DMAxSZ register is only written when the DMA is in the Reset state (DMAEN = 0).

DMA6 *DMA Module*

Category Functional

Function DMA cannot write to DMA

Description One DMA channel cannot modify the registers of another DMA channel. DMA register modifications must be done by JTAG or by the CPU.

Workaround None

DMA7 *DMA Module*

Category Functional

Function DMA request may cause the loss of interrupts

Description If a DMA request starts executing during the time when a module register containing an interrupt flags is accessed with a read-modify-write instruction, a newly arriving interrupt from the same module can get lost. An interrupt flag set prior to DMA execution would not be affected and remain set.

Copyright © 2012–2019, Texas Instruments Incorporated

Workaround 1. Use a read of Interrupt Vector registers to clear interrupt flags and do not use read-modify-write instruction.

OR

2. Disable all DMA channels during read-modify-write instruction of specific module registers containing interrupts flags while these interrupts are activated.

DMA8 *DMA Module*

Category Functional

Function DMA can corrupt values on write-access to program stack

Description If the DMA controller makes a write access to the stack while executing one of the following instructions, the data that is written may be corrupted.

CALLA [REG | IDX | SYM | ABS | IND | INA | IMM]

PUSHX.A [IDX | SYM | ABS | IND | IMM | INA]

PUSHX.A [REG]

PUSHM.A [REG]

POPM.A [REG]

Note: [...] denotes an addressing mode

Workaround Do not declare function-scope variables. Declare all variables that are intended to be modified by the DMA as global- or file-scope such that they are allocated in the data section of RAM and not on the program stack.

DMA10 *DMA Module*

Category Functional

Function DMA access may cause invalid module operation

Description The peripheral modules MPY, CRC, USB, RF1A and FRAM controller in manual mode can stall the CPU by issuing wait states while in operation. If a DMA access to the module occurs while that module is issuing a wait state, the module may exhibit undefined behavior.

Workaround Ensure that DMA accesses to the affected modules occur only when the modules are not in operation. For example with the MPY module, ensure that the MPY operation is completed before triggering a DMA access to the MPY module.

EEM4 *EEM Module*

Category Debug

Function No breakpoint stop reaction at DMA/CPU switch

Description If an address-related trigger, such as the general IDE breakpoint, is triggered for a CPU instruction one cycle after a DMA transaction, the stop reaction will not be accepted by the EEM (will not halt program execution). A data related trigger set at a DMA transaction one cycle after a CPU instruction results in the same behavior.

Workaround Use trigger configurations that are not dependent on DMA/not-DMA transactions (Read, Write, or Don't Care). The State Storage Block can then be used at program stop to

Copyright © 2012–2019, Texas Instruments Incorporated

determine:

- Whether the trigger-causing event was a DMA or a non-DMA transaction

- The state of the program at the time of the event

EEM5 *EEM Module*

Category Debug

Function Individual timer enable/disable control unavailable

Description The timer control bits work such that the timers cannot be individually enabled or disabled for debug operation. All three timers are sourced by the selected global debug clock.

Workaround None

EEM6 *EEM Module*

Category Debug

Function Return from breakpoint with wrong PC value

Description If a breakpoint occurs on an instruction which has a 20-bit write back to memory, the address offset does not match the debug flow. The PC is offset by 6 rather than 4, meaning it points to the 16-bit word after the op-code of the instruction at which the program was to continue execution.

Workaround The user needs to manually change to PC value to [(displayed_value) - 2] (i.e. via the debugger's CPU register window) before the next single step or run operation is issued

EEM8 *EEM Module*

Category Debug

Function Debugger stops responding when using the DMA

Description In repeated transfer mode, the DMA automatically reloads the size counter (DMAxSZ) once a transfer is complete and immediately continues to execute the next transfer unless the DMA Enable bit (DMAEN) has been previously cleared. In burst-block transfer mode, DMA block transfers are interleaved with CPU activity 80/20% - of ten CPU cycles, eight are allocated to a block transfer and two are allocated for the CPU.

Because the JTAG system must wait for the CPU bus to be clear to halt the device, it can only do so when two conditions are met:

- Three clock cycles after any DMA transfer, the DMA is no longer requesting the bus.

and

- The CPU is not requesting the bus.

Therefore, if the DMA is configured to operate in the repeat burst-block transfer mode, and a breakpoint is set between the line of code that triggers the DMA transfers and the line that clears the DMAEN bit, the DMA always requests the bus and the JTAG system never gains control of the device.

Workaround When operating the DMA in repeat burst-block transfer mode, set breakpoint(s) only when the DMA transfers are not active (before the start or after the end of the DMA transfers).

Copyright © 2012–2019, Texas Instruments Incorporated

EEM9 *EEM Module*

Category Debug

Function Combined triggers on the PUSH instruction may be missed

Description When the PUSH instruction is used in any addressing mode except register or immediate modes, a combined trigger may be missed when its conditions are defined by a PUSH instruction fetch and a successful match of the value being pushed onto stack.

Workaround None

EEM11 *EEM Module*

Category Debug

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*

Category Debug

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.

EEM14 *EEM Module*

Category Debug

Function Single-step or breakpoint on module registers with WAIT capability may not work

Description In debug mode, the CPU clock is driven independently from the wait inputs of device modules (i.e., MULT, USB, RF1A, CRC). As a result, an EEM halt on an access to the module data registers (breakpoint or single-step) may show incorrect results due to incomplete execution.

Workaround Do not single-step through a data register access that holds the CPU to provide a valid result. Place breakpoints after the affected register is accessed and sufficient clock

Copyright © 2012–2019, Texas Instruments Incorporated

cycles have been provided.

NOTE: This erratum applies to debug mode only.

EEM16 *EEM Module*

Category Debug

Function The state storage display does not work reliably when used on instructions with CPU Wait cycles.

Description When executing instructions that require wait states; the state storage window updates incorrectly. For example a flash erase instruction causes the CPU to be held until the erase is completed i.e. the flash puts the CPU in a wait state. During this time if the state storage window is enabled it may incorrectly display any previously executed instruction multiple times.

Workaround Do not enable the state storage display when executing instructions that require wait states. Instead set a breakpoint after the instruction is completed to view the state storage display.

NOTE: This erratum affects debug mode only.

EEM17 *EEM Module*

Category Debug

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.

Workaround None. Use other conditional/advanced triggered breakpoints to halt the debugger right after Flash erase/write instructions.

NOTE: This erratum affects debug mode only.

EEM19 *EEM Module*

Category Debug

Function DMA may corrupt data in debug mode

Description When the DMA is enabled and the device is in debug mode, the data written by the DMA may be corrupted when a breakpoint is hit or when the debug session is halted.

Copyright © 2012–2019, Texas Instruments Incorporated

Workaround This erratum has been addressed in MSPDebugStack version 3.5.0.1. It is also available in released IDE EW430 IAR version 6.30.3 and CCS version 6.1.1 or newer.

If using an earlier version of either IDE or MSPDebugStack, do not halt or use breakpoints during a DMA transfer.

NOTE: This erratum applies to debug mode only.

EEM21 *EEM Module*

Category Debug

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*

Category Debug

Function EEM triggers incorrectly when modules using wait states are enabled

Description When modules using wait states (USB, MPY, CRC and FRAM controller in manual mode) are enabled, the EEM may trigger incorrectly. This can lead to an incorrect profile counter value or cause issues with the EEMs data watch point, state storage, and breakpoint functionality.

Workaround None.

NOTE: This erratum affects debug mode only.

FLASH28 *FLASH Module*

Category Functional

Function Read disturb due to PMM level < 2

Description Flash access with default PMMCOREV settings (PMMCOREVx = 00) or PMM level 1 (PMMCOREVx = 01) is not reliable over the full operating range of the device and is not recommended.

Workaround None.

Note: The default PMM setting has been modified to PMM level 2. The affected registers and the new default register settings are shown below. The PMM level setting should not be changed in application code.

PMMCTL0 = 0x02

SVSMHCTL = 0x4602

SVSMLCTL = 0x4602

Copyright © 2012–2019, Texas Instruments Incorporated

FLASH29 *FLASH Module*

Category Functional

Function Read disturb due to emergency exit from write/erase Flash operation

Description When a Flash write or erase is abruptly terminated, any further reliable reads from Flash are not guaranteed. The abrupt termination can occur as a result of the Emergency Exit bit (EMEX in FCTL3) being set. This forces a write or an erase operation to be terminated before normal completion.

Workaround After setting EMEX = 1, wait for at least 100 us after a bank or mass erase and at least 6 us after a segment erase before Flash is accessed again.

FLASH30 *FLASH Module*

Category Functional

Function Read disturb

Description There is a problem that affects how addresses are decoded for info memory, creating read disturb issues for code executed out of information memory.

Workaround No code should be executed from INFO memory. In addition, data stored in INFO memory can only be accessed using code executed from the lower 64K of the memory map. Such accesses to INFO memory are unaffected by the problem.

FLASH31 *FLASH Module*

Category Functional

Function Interrupts not disabled during FLASH erase operation

Description When a flash erase operation is in progress, interrupts are not automatically disabled. The CPU will always try to service the interrupt request, whether or not the flash is busy.

Workaround Disable interrupts using the GIE bit before erasing flash in another bank of memory. Note that all interrupts during this period of time will remain pending until GIE = 1.

FLASH32 *FLASH Module*

Category Functional

Function Bank erase problems

Description A Bank Erase operation can also lead to the erase of the information memory of the selected bank, when there was a read from any information memory address before the dummy write.

Workaround Insert a dummy read from any MAIN memory address before the dummy write, which triggers the Bank Erase operation.

FLASH34 *FLASH Module*

Copyright © 2012–2019, Texas Instruments Incorporated

Category Functional

Function Concurrent flash read during bank erase fails

Description Code residing in flash cannot be executed during a bank erase.

Workaround Place the code to be executed during bank erase in RAM.

FLASH37 *FLASH Module*

Category Functional

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;
```


Copyright © 2012–2019, Texas Instruments Incorporated

```
// 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 ************
```

JTAG17 *JTAG Module*

Category Debug

Function JTAG mailbox handshake mechanism can go out of sync.

Description The JTAG mailbox handshake mechanism can go out of sync, if the CPU accesses the Mailbox (read or write on JMB data registers) while the JMBOUTCTL/JMBIBCTL register is accessed via the JTAG interface.

Workaround - Avoid reading or writing on JMB data registers by CPU while the JMBOUTCTL/JMBIBCTL register is accessed via the JTAG interface.

or

- Send a request (set OUTREQ or INPREQ bit in JMBINCTL register) only when it can be served immediately. This means in detail:

INPUT: Wait until the the input registers are empty (IN0RDY = IN1RDY = 1) before sending an input request (INPREQ = 1) to the JMBINCTL register.

OUTPUT 16 Bit: Wait until the CPU has written Register JMBOUT0 (OUT0RDY = 1) before sending an output request (OUTREQ = 1) to the JMBINCTL register.

OUTPUT 32 Bit: Wait until the CPU has written Registers JMBOUT0 and JMBOUT1 (OUT0RDY = OUT1RDY = 1) before sending an output request (OUTREQ = 1) to the JMBINCTL register.

JTAG20 *JTAG Module*

Category Software in ROM

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.

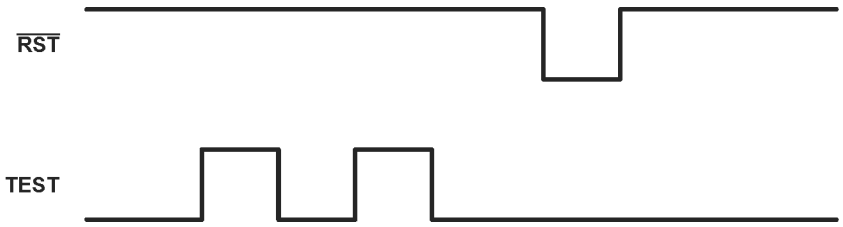
Workaround To exit the BSL one of the following methods must be used.

- A Power cycle

or

- Toggle the TEST pin twice when nRST is high and after 50us pull nRST low.

Copyright © 2012–2019, Texas Instruments Incorporated



Note: This toggling of TEST pins is not subject to timing constraints. The appropriate level transitions on TEST pin, followed by a RST pulse after 50us, are sufficient to trigger an exit from BSL mode.

JTAG26 *JTAG Module*

Category Debug

Function LPMx.5 Debug Support Limitations

Description The JTAG connection to the device might fail at device-dependent low or high supply voltage levels if the LPMx.5 debug support feature is enabled. To avoid a potentially unreliable debug session or general issues with JTAG device connectivity and the resulting bad customer experience Texas Instruments has chosen to remove the LPMx.5 debug support feature from common MSP430 IDEs including TIs Code Composer Studio 6.1.0 with msp430.emu updated to version 6.1.0.7 and IARs Embedded Workbench 6.30.2, which are based on the MSP430 debug stack MSP430.DLL 3.5.0.1 http://www.ti.com/tool/MSPDS

TI plans to re-introduce this feature in limited capacity in a future release of the debug stack by providing an IDE override option for customers to selectively re-activate LPMx.5 debug support if needed. Note that the limitations and supply voltage dependencies outlined in this erratum will continue to apply.

For additional information on how the LPMx.5 debug support is handled within the MSP430 IDEs including possible workarounds on how to debug applications using LPMx.5 without toolchain support refer to Code Composer Studio User's Guide for MSP430 chapter F.4 and IAR Embedded Workbench User's Guide for MSP430 chapter 2.2.5.

Workaround 1. If LPMx.5 debug support is deemed functional and required in a given scenario:

a) Do not update the IDE to continue using a previous version of the debug stack such as MSP430.DLL v3.4.3.4.

OR

b) Roll back the debug stack by either performing a clean re-installation of a previous version of the IDE or by manually replacing the debug stack with a prior version such as MSP430.DLL v3.4.3.4 that can be obtained from http://www.ti.com/tool/MSPDS.

2. In case JTAG connectivity fails during the LPMx.5 debug mode, the device supply voltage level needs to be raised or lowered until the connection is working.

Do not enable the LPMx.5 debug support feature during production programming.

JTAG27 *JTAG Module*

Category Debug

Function Unintentional code execution after programming via JTAG/SBW

Description The device can unintentionally start executing code from uninitialized RAM addresses 0x0006 or 0x0008 after being programming via the JTAG or SBW interface. This can

Copyright © 2012–2019, Texas Instruments Incorporated

result in unpredictable behavior depending on the contents of the address location.

Workaround

1. If using programming tools purchased from TI (MSP-FET, LaunchPad), update to CCS version 6.1.3 later or IAR version 6.30 or later to resolve the issue.

2. If using the MSP-GANG Production Programmer, use v1.2.3.0 or later.

3. For custom programming solutions refer to the specification on MSP430 Programming Via the JTAG Interface User's Guide (SLAU320) revision V or newer and use MSPDebugStack v3.7.0.12 or later.

For MSPDebugStack (MSP430.DLL) in CCS or IAR, download the latest version of the development environment or the latest version of the MSPDebugStack

NOTE: This only affects debug mode.

MPY1 *MPY Module*

Category Functional

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

PMM1 *PMM Module*

Category Functional

Function Interrupt or POR events from SVSx or SVMx are delayed until end of DMA transfer.

Description If a DMA transfer is active, any interrupt or POR event triggered by SVSx or SVMx is delayed until end of DMA transfer. No event gets lost, but it is delayed.

Workaround Avoid long durations of DMA transfers.

PMM2 *PMM Module*

Category Functional

Function Manual change of LDO voltage reference from 'switched' to 'static' mode may trigger an incorrect set event for SVSxIFG's (where 'x' stands for 'H' or 'L')

Description The incorrect set event can occur under the following PMM configuration and conditions (please refer to PMM figures titled "High-Side/Low-Side SVS and SVM"):

- SVSxMD is reset to 0, disabling the set of SVSxIFG when LDO is in 'switched' mode

+ LDO voltage reference automatically set to 'switched' mode if:

Copyright © 2012–2019, Texas Instruments Incorporated

- Device enters LPM2, LPM3, LPM4

- PMMCTL1 register bits PMMCMDx = 10b

- While LDO voltage reference is in 'switched' mode, the supervised voltage level is violated

- LDO voltage reference is changed back to 'static' mode (either by return from LPM mode or through the use of PMMCMDx bits) and the supervised voltage level is still being violated

Workaround SVSx events are not triggered when the LDO voltage reference is in 'switched' mode and the SVSx high-side mode is disabled (SVSHMD = 0). Under these conditions, the SVSx (SVSxE = 0) should be temporarily disabled for any entry into 'switched' mode.

PMM3 *PMM Module*

Category Functional

Function Bit PMMLPM5IFG in PMMIFG register is not readable

Description Bit PMMLPM5IFG in PMMIFG register is not readable. User software read value is always 0; however, the bit functions as expected. Write access is working, and the reset of the bit by reading the respective SYSRSTIV vector word operates as described in the user's guide.

Workaround Check the status of PMMLPM5IFG flag by reading the register SYSRSTIV vector word.

PMM5 *PMM Module*

Category Functional

Function Unexpected SVMxIFG flag

Description The SVMxIFG flags can be unexpectedly set (where 'x' stands for 'H' or 'L'), if the following conditions are met:

SVMxE = 1 ; SVM is enabled

SVMxOVPE = 1 ; Overvoltage protection is on

SVSMxACE = 1 ; Automatic performance mode selection is on

SVMLFP = 0 ; Performance mode set to 'normal'

Workaround None.

PMM6 *PMM Module*

Category Functional

Function Brownout startup problem at low temp

Description At low temp (less than -20C) the device is likely to not work correctly, after brownout condition occurred. A brownout condition is defined as drop on the DVDD supply to DVSS level and shorter than 2 seconds.

Workaround None

Copyright © 2012–2019, Texas Instruments Incorporated

PMM7 *PMM Module*

Category Functional

Function PMMRIE default conditions different than user guide

Description The user guide specifies that, after a BOR reset condition, the SVS will not be configured to trigger a POR signal in the condition that the monitored voltages fall below the SVS level(s). This is not true for this device. The SVS Low and SVS High Side POR Enable bits (SVSLPE/SVSHPE) in the Power Management System Reset Enable and Interrupt Enable register are set by default (PMMRIE = 0x1100).

Workaround If this behavior is not desired, reset the SVSLPE/SVSHPE bits in the PMMRIE register at the beggining of the application.

PMM8 *PMM Module*

Category Functional

Function Supply current in LPM4.5 is unpredictable

Description Due to an unpredictable value of the supply current in LPM4.5, the mode should not be used.

Workaround None.

PMM9 *PMM Module*

Category Functional

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

Copyright © 2012–2019, Texas Instruments Incorporated

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*

Category Functional

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*

Category Functional

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*

Category Functional

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.

Copyright © 2012–2019, Texas Instruments Incorporated

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*

Category Functional

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*

Category Functional

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.

Copyright © 2012–2019, Texas Instruments Incorporated

	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	$t_{WAKE-UP\ FAST}$
	1	0	0	Normal	OFF	OFF	$t_{WAKE-UP\ SLOW}$
	1	0	1	Full Performance	OFF	OFF	$t_{WAKE-UP\ FAST}$
	1	1	0	Normal	Normal	OFF	$t_{WAKE-UP\ SLOW}$
	1	1	1	Full Performance	Full Performance	Normal	$t_{WAKE-UP\ FAST}$
SVML	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
	0	x		OFF	OFF	OFF	$t_{WAKE-UP\ FAST}$
	1	0		Normal	Normal	OFF	$t_{WAKE-UP\ SLOW}$
	1	1		Full Performance	Full Performance	Normal	$t_{WAKE-UP\ FAST}$

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.

	SVSHE	SVSHMD	SVSHFP	AM, LPM0/1 SVSH state	Manual SVSMHACE = 0 LPM2/3/4 SVSH State	Automatic SVSMHACE = 1 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
SVMH	SVMHE	SVMHFP		AM, LPM0/1 SVMH state	Manual SVSMHACE = 0 LPM2/3/4 SVMH State	Automatic SVSMHACE = 1 LPM2/3/4 SVMH 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 and ensure SVSH/SVMH is in manual mode. Instead force the modules to remain ON even in LPMx. Note that this will cause increased power consumption when in LPMx.

Copyright © 2012–2019, Texas Instruments Incorporated

Refer to the MSP430 Driver Library(MSPDRIVERLIB) 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.

PMM18 *PMM Module*

Category Functional

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 automatic control mode for high-side SVS & SVM (SVSMHCTL.SVSMHACE=1). The SVM high side is inactive in LPM2, LPM3, and LPM4.

PMM20 *PMM Module*

Category Functional

Copyright © 2012–2019, Texas Instruments Incorporated

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).

PORT13 *PORT Module*

Category Functional

Function PxIV register is cleared by debugger, affecting debug program flow

Description When the PxIV register is read via the debugger the register gets cleared. This can cause lost port interrupts during the debug sessions.

Workaround The wrong clearing of the register during debugging itself cannot be avoided. The debug software should write back by the interrupt flag after it has been read and cleared.

PORT14 *PORT Module*

Category Functional

Function GPIO pins set to high impedance on exit from LPM2/3/4

Description When automatic SVS control is enabled (SVSMHACE or SVSMLACEare set), all I/O's are set to high impedance state after wakeup from LPM 2/3/4. The duration of this high impedance state is between 10 and 100us. All input, output and pull resistor settings are not applied during this time.

Workaround None.

PORT16 *PORT Module*

Category Functional

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.

Copyright © 2012–2019, Texas Instruments Incorporated

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*

Category Functional

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

RTC2 *RTC Module*

Category Functional

Function RTC base address mirrored to another address location

Description Per datasheet specification, the range 0x0480 - 0x049F should be vacant space in the memory map, and any attempts to access it should create an NMI interrupt with a "vacant memory access" violation. However, the RTC module, which has a base address of 0x4A0, has been mirrored to this range instead, and therefore no violation is generated if code attempts to access the space. The RTC base address specified in the datasheet (0x049F) should be used when addressing RTC registers.

Workaround None

RTC3 *RTC Module*

Category Functional

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

Copyright © 2012–2019, Texas Instruments Incorporated

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*

Category Functional

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

Description In BCD mode of operation, the step size of the RTC frequency adjustment is =+12ppm/-4ppm. This is thrice the size specified in the User's Guide for up-calibration and twice the size for down-calibratin.

In Binary mode of operation, the step size of the RTC frequency adjustment is =+6ppm/-2ppm. This is 1.5x the step size specified in the User's Guide for up-calibration and is as per spec for down-calibration.

In BCD mode, for up calibration this results in a step size per step of 12ppm (1536 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).

In Binary mode, for up calibration this results in a step size per step of 6ppm (768 cycles) instead of 4ppm (512 cycles). For down calibration this results in a step size per step of 2ppm (256 cycles) as in spec.

Workaround To compensate for the erratic step size, the RTCCAL register can be written with

1/3 of the calibration value for BCD up mode, 1/2 of the calibration value for BCD down mode and 1/1.5 of the calibration value for Binary up mode.

SYS1 *SYS Module*

Category Functional

Function User NMI can disturb the higher priority system NMI (priority inversion)

Description A lower priority user NMI (UNMI) can disturb a higher priority system NMI (SNMI), leading to a priority inversion. Note that an interrupt request will still not disturb a higher priority NMI.

Workaround None

SYS2 *SYS Module*

Category Functional

Function Protection weakness in BSL flash address range

Description If the BSL Protection is enabled, just the lowest 16 bytes of the secured BSL part should be usable as entry area (jump table). Instead, this entry window (not secured from readout by CPU) can be also be seen on other addresses - address locations affected are dependent on the selected size of the BSL.

Example: for BSLSIZE = 0, memory is reflected into 0x01600 to 0x0160F & 0x01700 to 0x0170F

Similarly, if RAM space is assigned as BSL area and secured, the mechanism can be observed. Only the area 0x01C00 to 0x01C0F should be protected, but the following RAM areas are also secured and no longer accessible: 0x01D00 to 0x01D0F, 0x01E00

Copyright © 2012–2019, Texas Instruments Incorporated

to 0x01E0F, 0x01F00 to 0x01F0F, 0x02000 to 0x0200F... 0x02B00 to 0x02B0F.

Any access to locations affected by errata leads to a security reset.

Workaround - Do not use RAM as secured area for BSL

or

- Do not put sensitive user code/data in the additional BSL entry address windows.

SYS3 *SYS Module*

Category Functional

Function OFIFG is wrongly cleared by a SYSUNIV read

Description In a situation where more than one NMI flag is set, and software reads the SYSUNIV register, and then the very next memory access is also to the SYS module, then it is possible that an additional, lower priority NMI flag will also be cleared.

Workaround After reading the SYSUNIV register, insert an access to an address outside of the SYS module address space before making another access to the SYS module.

SYS4 *SYS Module*

Category Functional

Function BSL is not programmable

Description Due to other errata negatively affecting the expected behavior of code executed from non-MAIN segments of Flash, the BSL is not programmable for this device.

Workaround None.

SYS5 *SYS Module*

Category Functional

Function Higher priority NMI interrupt lost during execution of lower priority NMI ISR

Description If a higher prioritized NMI occurs during the execution of a lower priority NMI ISR (read out of NMI vector word) the higher IFG can get lost (but lower interrupt is executed), which leads to a second execution of the lower NMI. The bug only occurs if the NMIs are of the same type (UNMI or SMNI).

Workaround None.

SYS6 *SYS Module*

Category Functional

Function CPU skips execution of one additional instruction after returning from an SNMI ISR and before servicing a pending IRQ.

Description If an IRQ is activated during a SNMI ISR, after leaving the SNMI ISR, the CPU services the IRQ ISR immediately without first returning to the main program or the previously running ISR to execute one additional instruction (as defined in the SYS chapter of the

Copyright © 2012–2019, Texas Instruments Incorporated

device user's guide).

Workaround None.

SYS7 *SYS Module*

Category Functional

Function Timing of USCI interrupts may cause PC corruption due to automatic clear of an IFG

Description When certain USCI I2C interrupt flags (IFG) are set and an automatic flag-clearing event on the I2C bus occurs, the address bus will default to FFFEh (Reset Vector) and the data bus to the opcode for the RETI instruction. This means the program counter defaults to FFFEh (Reset Vector) during this period of time. This will only happen when the IFG is cleared within a critical time window (~6 CPU clock cycles) after a USCI interrupt request occurs and before the interrupt servicing is initiated. The affected interrupts are UCBxTXIFG, UCSTPIFG, UCSTTIFG and UCNACKIFG.

The automatic flag-clearing scenarios are described in the following situations:

- A pending UCBxTXIFG interrupt request is cleared on the falling SCL clock edge following a NACK.

- A pending UCSTPIFG, UCSTTIFG, or UCNACKIFG interrupt request is cleared by a following Start condition.

If an NMI request occurs during the time that the PC holds the default value FFFEh before the default RETI instruction is executed, the PC will be corrupted on return from the NMI ISR. This is because the previous PC contents of FFFEh were stored onto the stack as the address of the next instruction to be executed. The CPU consequently interprets the address in FFFEh as an instruction (errant behavior) and because the default RETI was never executed, the stack pointer remains decreased by four (errant behavior).

Workaround Prevent the flag-clearing event from interrupting the servicing of the affected USCI IFG's.

- Poll the affected USCI flags instead of enabling the interrupts.

or

- Ensure the above mentioned flag-clearing events occur after a time delay of 6 CPU clock cycles after the interrupt requests occur and are accepted.

SYS8 *SYS Module*

Category Functional

Function Sporadic device crash on LPMx

Description If an interrupt occurs within 5 ns of the last rising edge of the MCLK prior to entering low-power mode (LPMx), the flash controller experiences a glitch that latches the last address on the address bus. This is the address that contains the opcode following the entry to LPMx.

When the core then requests the address for the interrupt handler routine, the flash instead returns the the data at the address that was latched. This data is interpreted as the address for the interrupt handler, resulting in errant code execution and an eventual reset condition.

Workaround None.

Copyright © 2012–2019, Texas Instruments Incorporated

SYS9 *SYS Module*

Category Functional

Function UNMI ISR executes twice

Description If a running User NMI interrupt service routine is interrupted by a SYSNMI interrupt before the respective USERNMI flag is cleared (NMIIFG, OFIFG, or ACCVIFG) the SYSNMI ISR will be executed, the UNMI ISR will complete, then the UNMI ISR will be executed a second time.

Workaround A common NMI handler can be written where the SNMI and UNMI Interrupt Vectors point to the same function. This common NMI handler must manually clear both types of pending NMI sources (User and System) to avoid double execution of the UNMI ISR.

SYS12 *SYS Module*

Category Functional

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. However the VPE does not signify an invalid flash operation has occurred.

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

SYS13 *SYS Module*

Category Functional

Function Device hangs with slow Vcc rise time

Description If Vcc with a slow rise time is applied to the device, the BOR releases the device at approximately 1.8V. After device release, the CPU executes boot code from flash, but Vcc might not be in the specified voltage range (that is, it might be below 2.2 V). This can cause the device to hang.

Workaround - Hold reset low until the supply voltage applied to the device is in the valid range (2.2V to 3.6V).

- Apply an additional external reset after the supply voltage reaches the valid range.

SYS16 *SYS Module*

Category Functional

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.

Copyright © 2012–2019, Texas Instruments Incorporated

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

TA20 *TIMER_A Module*

Category Functional

Function TA0 output connection to ADC12 is incompatible with previous device families

Description The Timer_A output signal, TA0, is connected to the ADC12. To be compatible with previous device families, should be connected to TA1.

Workaround Modify any existing code to use TA0 as opposed to TA1. In addition, Timer_B7 now supports TB0 or TB1 for usage with the ADC12.

TAB23 *TIMER_A/TIMER_B Module*

Category Functional

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.

TB20 *TIMER_B Module*

Category Functional

Function ACLK cannot be used as an internal capture input

Description The capture compare input of Timer_B7, CCI6B, is not connected internally to ACLK, as in previous device families.

Workaround ACLK can be used indirectly by exporting the ACLK signal through its respective output port pin and feeding it back into TB6.

TB21 *TIMER_B Module*

Category Functional

Function Timer_B7 outputs not set to Hi-Z state

Description The Timer_B7 TBOUTH pin function does not work as described in the user's guide. When the TBOUTH pin is pulled high, the Timer_B7 outputs do not enter a Hi-Z state.

Workaround None

Copyright © 2012–2019, Texas Instruments Incorporated

UCS1 *UCS Module*

Category Functional

Function WDT sourced by VLO is not halted at breakpoint condition

Description When using VLO Clock as WDT source clock, the WDT does not stop in a Breakpoint condition. The VLO clock is not controllable in Debug mode (no standard MSP430 module source clock), resulting in unexpected resets or counter values from the WDT.

Workaround Do not use the VLO clock as the WDT source for debug.

UCS2 *UCS Module*

Category Functional

Function LPM modes not entered properly when MCLK < ACLK

Description When MCLK is sourced by XT1, XT2 or REFO clock sources and MCLK is slower than ACLK the low power modes are not entered properly.

This applies when the MCLK clock divider is greater than the ACLK clock divider setting.

Workaround None

UCS4 *UCS Module*

Category Functional

Function Non-monotonic behavior of DCO

Description When FLL modulation is enabled, the modification of DCO & MOD bits in the UCSCTL0 register can result in an unexpected increment or decrement of the DCO tap. This leads to the non-monotonic behavior of the DCO.

Workaround Modify the DCO & MOD values when modulation is disabled (DISMOD = 1).

BIS.W #DISMOD, &UCSCTL1

MOV.W #-YOUR_DCO_MOD_SETTING-, &UCSCTL0

BIC.W #DISMOD, &UCSCTL1

UCS5 *UCS Module*

Category Functional

Function Writes to UCSCTL0, UCSCTL1, UCSCTL2 and UCSCTL3 registers do not get updated

Description During FLL operation, if two subsequent writes occur to the UCSCTL0-2 registers within one cycle of the DCO, the results are not updated and do not impact the FLL as expected.

Similarly, if two subsequent writes are performed to the UCSCTL3 register within one cycle of FLLREFCLK the results do not get updated.

Workaround When writing to the UCSCTL0-2 registers, drive the CPU from a clock derived from DCO.

Copyright © 2012–2019, Texas Instruments Incorporated

When writing to UCSCTL3, either avoid making subsequent writes, or insert a delay loop between the accesses to prevent them from occurring within the same FLLREFCLK cycle.

UCS6 *UCS Module*

Category Functional

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*

Category Functional

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*

Category Functional

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.

Copyright © 2012–2019, Texas Instruments Incorporated

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*

Category Functional

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.

Copyright © 2012–2019, Texas Instruments Incorporated

UCS11 *UCS Module*

Category Functional

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

Description Changing the SELM/SELS/SELA bits in the UCSCTL4 register will correctly configure the respective clock to use the intended clock source but might also erroneously set XT1/XT2 fault flag if the crystals are not present at XT1/XT2 or not configured in the application firmware. If the NMI interrupt for the OFIFG is enabled, an unintentional NMI interrupt will be triggered and needs to be handled.

NOTE: The XT1/XT2 fault flag can be set regardless of which SELM/SELS/SELA bit combinations are being changed.

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

If OFIFG-NMI is enabled during clock switching, disable OFIFG-NMI interrupt during changing the SELM/SELS/SELA bits in the UCSCTL4 register to prevent unintended NMI.

Alternatively it can be handled accordingly (clear falsely set fault flags) in the Interrupt Service Routine to ensure proper OFIFG clearing.

UCS13 *UCS Module*

Category Functional

Function MODOSC can stop if deactivation time is short

Description If modules like the FLASH controller (default setting and not configurable) or the ADC12 (configurable) use the MODOSC as clock source and perform a de-activation followed by activation, the MODOSC can become stopped.

This can lead to a non-responsive device that can be released with a PUC (e.g. triggered by WDT) or any higher prioritized reset source.

The FLASH controller is waiting on its clock to perform a programming operation.

The FLASH controller example is as follows:

If two bytes were programmed after each other at dedicated frequencies with a dedicated instruction set, such as the following FLASH write routine:

```
MOV.W #FWKEY,&FCTL3    ; Unlock the flash
MOV.W #FWKEY + WRT,&FCTL1    ; Enable single byte write
MOVA &fpointer, R14    ; Configure flash destination pointer
MOVA #buffer, R13    ; configure source buffer
MOV.B @R13+, 0(R14)    ; R14 pointing to flash address and CPU operates
above 12.5 MHz
ADDA #0x01, R14\n
MOV.B @R13+, 0(R14)
MOV.W #FWKEY,&FCTL1    ; Clear Write
MOV.W #FWKEY + LOCK,&FCTL1    ; Lock the flash
```

Then, in this case, the device might be un-responsive due to MODOSC might stop and the FLASH controller is waiting on its clock to perform a programming operation. In this case, the device draws higher current because flash charge pump is enabled as the FLASH controller is in a programming loop.

Copyright © 2012–2019, Texas Instruments Incorporated

Workaround In general, action needs to be taken to prevent MODOSC from a deactivation followed by an activation within 4 MODOSC clock cycles. This will ensure the MODOSC is properly disabled before being re-enabled.

For the FLASH use case above, this can be achieved by polling the BUSY flag located inside FCTL3 after each byte or word during programming.

In addition, no ADC12 sampling should be triggered within 4 MODOSC cycles after flash programming is finished. This can be achieved by disabling the ADC12 trigger during flash programming.

USCI26 *USCI Module*

Category Functional

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*

Category Functional

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

Copyright © 2012–2019, Texas Instruments Incorporated

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*

Category Functional

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
```

USCI34 *USCI Module*

Category Functional

Function I2C multi-master transmit may lose first few bytes.

Description In an I2C multi-master system (UCMM =1), under the following conditions:

(1)the master is configured as a transmitter (UCTR =1)

Copyright © 2012–2019, Texas Instruments Incorporated

AND

(2)the start bit is set (UCTXSTT =1);

if the I2C bus is unavailable, then the USCI module enters an idle state where it waits and checks for bus release. While in the idle state it is possible that the USCI master updates its TXIFG based on clock line activity due to other master/slave communication on the bus. The data byte(s) loaded in TXBUF while in idle state are lost and transmit pointers initialized by the user in the transmit ISR are updated incorrectly.

Workaround Verify that the START condition has been sent (UCTXSTT =0) before loading TXBUF with data.

Example:

```
#pragma vector = USCIAB0TX_VECTOR
__interrupt void USCIAB0TX_ISR(void)
{
// Workaround for USCI34
if(UCB0CTL1&UCTXSTT)
{
// TXData = pointer to the transmit buffer start
// PTxData = pointer to transmit in the ISR
PTxData = TXData; // restore the transmit buffer pointer if the Start bit is set
}
//
if(IFG2&UCB0TXIFG)
{
if (PTxData<=PTxDataEnd) // Check TX byte counter
{
UCB0TXBUF = *PTxData++; // Load TX buffer
}
else
{
UCB0CTL1 |= UCTXSTP; // I2C stop condition
IFG2 &= ~UCB0TXIFG; // Clear USCI_B0 TX int flag
__bic_SR_register_on_exit(CPUOFF); // Exit LPM0
}
}
}
```

USCI35 *USCI Module*

Category Functional

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 $t_{HD,STA}$

Copyright © 2012–2019, Texas Instruments Incorporated

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).

USCI39 *USCI Module*

Category Functional

Function USCI I2C IFGs UCSTTIFG, UCSTPIFG, UCNACKIFG

Description Unpredictable code execution can occur if one of the hardware-clear-able IFGs UCSTTIFG, UCSTPIFG or UCNACKIFG is set while the global interrupt enable is set by software (GIE=1). This erratum is triggered if ALL of the following events occur in following order:

1. Pending Interrupt: One of the UCxIFG=1 AND UCxIE=1 while GIE=0

2. The GIE is set by software (e.g. EINT)

3. The pending interrupt is cleared by hardware (external I2C event) in a time window of 1 MCLK clock cycle after the "EINT" instruction is executed.

Workaround Disable the UCSTTIE, UCSTPIE and UCNACKIE before the GIE is set. After GIE is set, the local interrupt enable flags can be set again.

Assembly example:

```
bic #UCNACKIE+UCSTPIE+UCSTTIE, UCBxIE ; disable all self-clearing interrupts
NOP
EINT
bis #UCNACKIE+UCSTPIE+UCSTTIE, UCBxIE ; enable all self-clearing interrupts
```

USCI40 *USCI Module*

Category Functional

Function SPI Slave Transmit with clock phase select = 1

Description In SPI slave mode with clock phase select set to 1 (UCAxCTLW0.UCCKPH=1), after the first TX byte, all following bytes are shifted by one bit with shift direction dependent on UCMSB. This is due to the internal shift register getting pre-loaded asynchronously when writing to the USCIA TXBUF register. TX data in the internal buffer is shifted by one bit after the RX data is received.

Workaround Reinitialize TXBUF before using SPI and after each transmission.

If transmit data needs to be repeated with the next transmission, then write back previously read value:

```
UCAxTXBUF = UCAxTXBUF;
```

WDG4 *WDT Module*

Category Functional

Function The WDT failsafe can be disabled

Copyright © 2012–2019, Texas Instruments Incorporated

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

Copyright © 2012–2019, Texas Instruments Incorporated

8 Document Revision History

Changes from family erratasheet to device specific erratasheet.

1. Errata RTC4 was removed
2. FLASH28 is not impacting silicon revision N

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. RTC3 Description was updated.
4. RTC3 Workaround was updated.
5. RTC3 Function was updated.

Changes from document Revision A to Revision B.

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

Changes from document Revision B to Revision C.

1. DMA10 Description was updated.
2. DMA10 Function was updated.

Changes from document Revision C to Revision D.

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

Changes from document Revision D to Revision E.

1. SYS16 Description was updated.
2. CPU43 Description was updated.
3. Errata USCI34 was added to the errata documentation.
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. Errata UCS11 was added to the errata documentation.

Changes from document Revision G to Revision H.

1. EEM19 Workaround was updated.
2. EEM13 Workaround was updated.
3. EEM23 Workaround was updated.
4. EEM17 Description was updated.
5. EEM23 Description was updated.
6. EEM17 Workaround was updated.
7. PORT16 Workaround was updated.
8. CPU43 Description was updated.
9. EEM11 Workaround was updated.
10. EEM14 Workaround was updated.
11. EEM16 Description was updated.
12. PORT16 Description was updated.

Copyright © 2012–2019, Texas Instruments Incorporated

13. EEM16 Workaround was updated.
14. EEM19 Description was updated.

Changes from document Revision H to Revision I.

1. DMA10 Workaround was updated.
2. DMA10 Description was updated.
3. DMA10 Function was updated.

Changes from document Revision I to Revision J.

1. CPU40 Workaround was updated.
2. EEM19 Workaround was updated.
3. Errata USCI39 was added to the errata documentation.
4. Package Markings section was updated.
5. EEM23 Workaround was updated.
6. EEM23 Description was updated.
7. Errata ADC42 was added to the errata documentation.
8. EEM23 Function was updated.
9. EEM19 Description was updated.

Changes from document Revision J to Revision K.

1. Errata USCI40 was added to the errata documentation.
2. Errata CPU43 was removed from the errata documentation.
3. DMA7 Workaround was updated.
4. DMA7 Description was updated.
5. PMM18 Workaround was updated.

Changes from document Revision K to Revision L.

1. DMA7 Workaround was updated.
2. EEM23 Description was updated.
3. DMA7 Description was updated.

Changes from document Revision L to Revision M.

1. USCI39 Description was updated.

Changes from document Revision M to Revision N.

1. Errata JTAG26 was added to the errata documentation.

Changes from document Revision N to Revision O.

1. EEM19 Workaround was updated.

Changes from document Revision O to Revision P.

1. UCS11 Workaround was updated.
2. UCS11 Description was updated.
3. UCS11 Function was updated.

Changes from document Revision P to Revision Q.

1. Errata JTAG27 was added to the errata documentation.

Changes from document Revision Q to Revision R.

1. JTAG20 Workaround was updated.
2. USCI39 Workaround was updated.
3. SYS12 Description was updated.

Changes from document Revision R to Revision S.

1. CPU21 was added to the errata documentation.

Copyright © 2012–2019, Texas Instruments Incorporated

2. CPU23 was added to the errata documentation.
3. CPU22 was added to the errata documentation.
4. Workaround for CPU20 was updated.
5. Workaround for CPU27 was updated.
6. Function for CPU26 was updated.
7. Description for CPU26 was updated.
8. Workaround for CPU26 was updated.
9. Workaround for CPU29 was updated.
10. Workaround for CPU28 was updated.
11. Workaround for CPU25 was updated.
12. Workaround for CPU24 was updated.
13. Workaround for CPU30 was updated.
14. Workaround for CPU32 was updated.
15. Workaround for CPU31 was updated.
16. Workaround for CPU39 was updated.
17. Workaround for CPU34 was updated.
18. Workaround for CPU33 was updated.
19. Workaround for CPU35 was updated.
20. Workaround for CPU16 was updated.
21. Workaround for CPU15 was updated.
22. Workaround for CPU18 was updated.
23. Workaround for PMM15 was updated.
24. Workaround for CPU40 was updated.

Changes from document Revision S to Revision T.

1. Workaround for PMM15 was updated.

Changes from document Revision T to Revision U.

1. Description for RTC6 was updated.
2. Workaround for RTC6 was updated.

Changes from document Revision U to Revision V.

1. Erratasheet format update.
2. Added errata category field to "Detailed bug description" section

Changes from document Revision V to Revision W.

1. Workaround for CPU40 was updated.

Changes from document Revision W to Revision X.

1. CPU47 was added to the errata documentation.
2. ADC69 was added to the errata documentation.

Changes from document Revision X to Revision Y.

1. Workaround for USCI34 was updated.

Changes from document Revision Y to Revision Z.

1. UCS13 was added to the errata documentation.

Copyright © 2012–2019, Texas Instruments Incorporated

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2019, Texas Instruments Incorporated