MSP430F5634 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 E	Rev D
ADC25		✓
ADC42	✓	✓
ADC69	✓	✓
COMP10	✓	✓
CPU37	✓	✓
CPU46	✓	✓
CPU47	✓	✓
DMA4	✓	✓
DMA7	✓	✓
DMA10	✓	✓
MPY1	✓	✓
PMAP1	✓	✓
PMM11	✓	✓
PMM12	✓	✓
PMM14	✓	✓
PMM15	✓	✓
PMM18	✓	✓
PMM20	✓	✓
PMM26	✓	✓
PORT15	✓	✓
PORT17	✓	✓
PORT19	✓	✓
RTC16	✓	✓
SYS16	✓	✓
SYS18	✓	✓
TAB23	✓	✓
UCS9	✓	✓
UCS11	✓	✓
USB9	✓	✓
USB10	✓	✓
USB11		✓
USB12	✓	✓

Errata Number	Rev E	Rev D
USCI26	✓	✓
USCI31	✓	✓
USCI34	✓	✓
USCI35	✓	✓
USCI39	✓	✓
USCI40	✓	✓

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 E	Rev D
BSL6		✓
BSL7		✓
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 E	Rev D
EEM11	✓	✓
EEM16	✓	✓
EEM17	✓	✓
EEM19	✓	✓
EEM21	✓	✓
EEM23	✓	✓
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 E	Rev D
CPU21	✓	✓
CPU22	✓	✓
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

5 Package Markings

PZ100 *LQFP (PZ) 100 Pin*

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

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

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

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

ZQW113 *BGA (ZQW), 113 Pin*

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

6 Memory-Mapped Hardware Revision (TLV Structure)

Die Revision	TLV Hardware Revision
Rev E	23h
Rev D	22h

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

7 Detailed Bug Description

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.

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.

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

BSL6 *BSL Module*

Category Software in ROM

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

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

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

Applications that do not use the USB BSL are unaffected.

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

BSL7 *BSL Module*

Category Software in ROM

Function BSL does not start after waking up from LPMx.5

Description When waking up from LPMx.5 mode, the BSL does not start as it does not clear the Lock I/O bit (LOCKLPM5 bit in PM5CTL0 register) on start-up.

Workaround 1. Upgrade the device BSL to the latest version (see Creating a Custom Flash-Based Bootstrap Loader (BSL) Application Note - SLAA450 for more details)

OR

2. Do not use LOCKLPM5 bit (LPMx.5) if the BSL is used but cannot be upgraded.

COMP10 *COMP_B Module*

Category Functional

Function Comparator port output toggles when entering or leaving LPM3/LPM4

Description The comparator port pin output (CECTL1.CEOUT) erroneously toggles when device enters or leaves LPM3/LPM4 modes under the following conditions:

1) Comparator is disabled (CECTL1.CEON = 0)

AND

2) Output polarity is enabled (CECTL1.CEOUTPOL = 1)

AND

3) The port pin is configured to have CEOUT functionality.

For example, if the CEOUT pin is high when the device is in Active Mode, CEOUT pin becomes low when the device enters LPM3/LPM4 modes.

Workaround When the comparator is disabled, ensure at least one of the following:

1) Output inversion is disabled (CECTL.CEOUTPOL = 0)

OR

2) Change pin configuration from CEOUT to GPIO with output low.

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.

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	

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
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

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 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	

CPU46 *CPUXv2 Module*

Category Functional

Function POPM peforms unexpected memory access and can cause VMAIFG to be set

Description When the POPM assembly instruction is executed, the last Stack Pointer increment is followed by an unintended read access to the memory. If this read access is performed on vacant memory, the VMAIFG will be set and can trigger the corresponding interrupt (SFRIE1.VMAIE) if it is enabled. This issue occurs if the POPM assembly instruction is performed up to the top of the STACK.

Workaround If the user is utilizing C, they will not be impacted by this issue. All TI/IAR/GCC pre-built libraries are not impacted by this bug. To ensure that POPM is never executed up to the memory border of the STACK when using assembly it is recommended to either

1. Initialize the SP to

a. TOP of STACK - 4 bytes if POPM.A is used

b. TOP of STACK - 2 bytes if POPM.W is used

OR

2. Use the POPM instruction for all but the last restore operation. For the the last restore operation use the POP assembly instruction instead.

For instance, instead of using:

```
POPM.W #5,R13
```

Use:

```
POPM.W #4,R12
POP.W R13
```

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

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	Not affected	C code is not impacted by this bug. User using POPM instruction in assembler is required to implement the above workaround manually.
TI MSP430 Compiler Tools (Code Composer Studio)	Not affected	C code is not impacted by this bug. User using POPM instruction in assembler is required to implement the above workaround manually.
MSP430 GNU Compiler (MSP430-GCC)	Not affected	C code is not impacted by this bug. User using POPM instruction in assembler is required to implement the above workaround manually.

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.

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).

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.

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.

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.

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.

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.

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.

JTAG20 *JTAG Module*

Category Software in ROM

Function BSL does not exit to application code

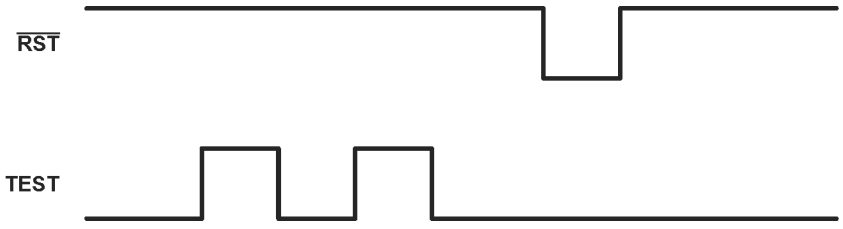
Description The methods used to exit the BSL per MSP430 Programming Via the Bootstrap Loader (SLAU319) are invalid.

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.



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 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

PMAP1 *PMAP Module*

Category Functional

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

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

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

or

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

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 5 MHz, 7.5 MHz, 10 MHz, or 12.5 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, 64, or 80 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(32)].

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.

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, 64, or 80 cycles for core voltage levels 0, 1, 2, and 3, respectively, before re-enabling the module [for example, __delay_cycles(32)].

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.

	SVSLE	SVSLMD	SVSLFP	AM, LPM0/1 SVSL state	Manual SVSMLACE = 0	Automatic SVSMLACE = 1	Wakeup Time LPM2/3/4
SVSL					LPM2/3/4 SVSL State	LPM2/3/4 SVSL State	
	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	Automatic SVSMLACE = 1	Wakeup Time LPM2/3/4
					LPM2/3/4 SVML State	LPM2/3/4 SVML State	
	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	Automatic SVSMHACE = 1
SVSH					LPM2/3/4 SVSH State	LPM2/3/4 SVSH State
	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	Automatic SVSMHACE = 1
					LPM2/3/4 SVMH State	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.

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

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).

PMM26 *PMM Module*

Category Functional

Function Device lock-up if RST pin pulled low during write to SVSMHCTL or SVSMLCTL

Description Device results in lock-up condition under one of the two scenarios below:

1) If RST pin is pulled low during write access to SVSMHCTL, with the RST/NMI pin is configured to reset function and is pulled low (reset event) the device will stop code execution and is continuously held in reset state. RST pin is no longer functional. The only way to come out of the lock-up situation is a power cycle.

OR

2) If RST pin is pulled low during write access to SVSMLCTL and only if the code that checks for SVSMLDLYIFG==1 is implemented without a timeout. The device will be stuck in the polling loop polling since SVSMLDLYIFG will never be cleared.

Workaround Follow the sequence below to prevent the lock-up for both use cases:

1) Disable RST pin reset function and switch to NMI before access SVSMHCTL or SVSMLCTL.

then

2) Activate NMI interrupt and handle reset events in this time by SW (optional if reset functionality required during access SVSMHCTL or SVSMLCTL)

then

3) Enable RST pin reset function after access to SVSMHCTL or SVSMLCTL

To prevent lock-up caused by use case #2 a timeout for the SVSMLDLYIFG flag check should be implemented to 300us.

PORT15 *PORT Module*

Category Functional

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

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

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

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

PORT17 *PORT Module*

Category Functional

Function Certain pins when subject to negative high current pulses may cause latch-up in adjacent pins.

Description Pins subject to negative high current pulses may cause latch-up in adjacent pins. The latch-up condition exists only if the adjacent pin configurations also referred to as 'affected-pin' configuration are one of the following:

(1) GPIO input driven high by an external source

(2) GPIO output driven high with Full Drive strength OR Reduced Drive strength settings

(3) Peripheral configuration where the peripheral drives pin high or causes pin to be driven high externally

The following affected-pin configurations will not sustain latch-up:

(1) GPIO input driven low

(2) GPIO output driven low

(3) Peripheral configuration where the peripheral drives pin low or causes pin to be driven low externally

(4) Peripheral configuration as LCD pin

Note that for affected-pin configurations with LCD functionality, the window of latch-up when the pin is driven being high still exists but is of extremely short duration and hence there is a low probability of latch-up occurrence.

Workaround All affected pins must be driven low when not in use. If the affected pins are not driven low, then connecting a series resistor of 330 ohms to limit the latch-up current is recommended.

For more details on trigger currents, affected pin configurations and workarounds refer to the document PORT17 Guidance SLAA562

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

RTC16 *RTC_B Module*

Category Functional

Function RTC_B module can seem stuck or function abnormally (jumping RTC)

Description If VBAT and DVCC (VPRIM) power up slowly and cross around the VBAK switching threshold, internal functions may not reset properly.

This can lead to a stuck RTC_B module or to unexpected functionality e.g. RTC_B is running faster which causes the observed time value to jump or skip forward.

Workaround Prevent DVCC (VPRIM) and VBAT from crossing each other below 2V during power up. It does not matter which signal comes up first.

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.

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

SYS18 *USB Module*

Category Functional

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

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

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

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

The correct line of code should read:

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

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

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

OR

2. Load a non-USB or custom BSL

OR

3. Erase the BSL

OR

4. Clear the access violation flag at the beginning of the application code with the following C code (or its assembly equivalent):

```
USBKEYPID = 0;       // Lock USB correctly
FCTL3 = 0xA558;      // Clear violation flag
```

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.

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.

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).

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.

USB9 *USB Module*

Category

Functional

Function

VBUS detection may fail after powerup

Description

In rare cases, some USB-equipped MSP430 devices may experience a failure in the bandgap that aids in detecting the presence of 5 V on the VBUS pin. Two primary effects of this are:

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

and

- The USB LDOs fail to start.

Workaround

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

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

However, if the integrated 3.3-V USB LDO (the output of the VUSB pin) is used to power the device's DVCC pin, as in many bus-powered applications, and if the rare bandgap error occurs, the CPU fails to power up, because the USB LDO fails to operate. The problem might be resolved by cycling power to the VBUS pin; for example, if the end user responds to the failure by unplugging and replugging the USB cable. The bandgap failure is also known to occur more often with slow DVCC ramps > 200 ms; for example, when there is excessive capacitance on the DVCC pin, in excess of what the USB specification allows. However, the only sure way to prevent the problem from occurring in the first place is to avoid making DVCC power reliant on VUSB.

USB10 *USB Module*

Category

Functional

Function

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

Description

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

When the SETUP packet has been decoded and the IN data prepared, the USB ISR

clears the SETUPIFG bit. But if it happens to do so within the 2nd CRC bit of an IN packet from the host, the USB module enters an errant state and can begin to endlessly transmit to the host, irrespective of the protocol. The errant state can be cleared by resetting the module with the USB_EN bit; but there's no way for software to reliably detect the condition.

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

Workaround

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

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

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

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

USB11 *USB Module*

Category Functional

Function USB BSL invoke

Description For devices with USB BSL, when externally invoking BSL according SLAU319 chapter 1.3.3. a critical setup time may not be met. In this case the BSL will not start. The pass/fail condition is temperature-dependent, where if a unit passes at a certain temperature, it will always pass at the same or higher temperature condition.

Workaround

1. Invoke the BSL from the application code and ensure VCore is set to level 2 or 3 prior to BSL entry.

OR

2. Update the device BSL. The CustomBSL source code implements the fix for this errata in versions 1.00.05.00 and newer. The CustomBSL package can be download at Custom BSL package

USB12 *USB Module*

Category Functional

Function The 2nd byte of a slave-to-host transmission is sent twice.

Description In extremely rare cases, when the USB module's PLL is disabled (by clearing the UPLLEN bit), the USB module can be placed into an undetermined state, resulting in an extra byte being sent to the host over the bus. The PLL is usually disabled by software when the USB module detects that the USB device has been suspended by the host. Suspend events can occur at any time, but are typically invoked during periods of inactivity.

Workaround Once this error occurs, the USB module needs to be reset (by clearing the USBEN bit), and then the module can be re-initialized. For example, software can call the MSP430 USB API USB_disable() followed by USB_enable(). These actions are taken by the USB APIs when the user unplugs and replugs the USB cable, which is likely to happen when the user realizes the bus is no longer working.

If automatic detection of the error is required, then software on the host and device could implement a CRC check on the data payload (above the USB API) to detect the extra byte. If detected, software could then disable/re-enable the USB module. (The CRC inherent in the USB protocol calculates over the data packet, and thus cannot detect the erroneously added byte.)

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

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)

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 $t_{HD,STA}$ 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;
```

8 Document Revision History

Changes from family erratasheet to device specific erratasheet.

1. Errata JTAG21 was removed
2. Revision A was removed
3. Revision B was removed
4. Revision C was removed
5. Errata FLASH38 was removed from Revision D
6. PZ100 package markings have been updated

Changes from device specific erratasheet to document Revision A.

1. Errata PORT19 was added to the errata documentation.
2. Errata PMM18 was added to the errata documentation.
3. Errata SYS18 was added to the errata documentation.
4. Errata PORT17 was added to the errata documentation.

Changes from document Revision A to Revision B.

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

Changes from document Revision B to Revision C.

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

Changes from document Revision C to Revision D.

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

Changes from document Revision D to Revision E.

1. SYS16 Description was updated.
2. CPU43 Description was updated.
3. Device TLV Hardware Revision information added to erratasheet.

Changes from document Revision E to Revision F.

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

Changes from document Revision F to Revision G.

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

Changes from document Revision G to Revision H.

1. EEM19 Workaround was updated.
2. EEM17 Workaround was updated.
3. Errata BSL12 was added to the errata documentation.
4. CPU43 Description was updated.
5. EEM11 Workaround was updated.
6. EEM23 Workaround was updated.

7. EEM17 Description was updated.
8. EEM16 Description was updated.
9. PORT17 Workaround was updated.
10. EEM23 Description was updated.
11. EEM19 Description was updated.
12. EEM16 Workaround was updated.

Changes from document Revision H to Revision I.

1. DMA10 Workaround was updated.
2. DMA10 Description was updated.
3. Errata BSL12 was removed from the errata documentation.
4. DMA10 Function was updated.
5. Errata USB11 was added to the errata documentation.

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. Errata USB12 was added to the errata documentation.
10. 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. Module name for SYS18 was modified.
4. SYS18 Workaround was updated.
5. PMM18 Workaround was updated.

Changes from document Revision K to Revision L.

1. Silicon Revision E was added to the errata documentation.
2. DMA7 Workaround was updated.
3. EEM23 Description was updated.
4. 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.
2. Errata PMM26 was added to the errata documentation.

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.
2. Errata COMP10 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. Errata CPU46 was added to the errata documentation.

Changes from document Revision R to Revision S.

1. CPU21 was added to the errata documentation.
2. CPU22 was added to the errata documentation.
3. ADC25 is no longer impacting silicon Revision E
4. Workaround for CPU40 was updated.
5. Workaround for CPU46 was updated.
6. Workaround for PMM15 was updated.

Changes from document Revision S to Revision T.

1. TLV hardware revision ID for Rev E was updated.
2. Workaround for CPU46 was updated.

Changes from document Revision T to Revision U.

1. Workaround for PMM15 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. Function for USB10 was updated.
2. Description for USB10 was updated.

Changes from document Revision X to Revision Y.

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

Changes from document Revision Y to Revision Z.

1. USCI34 was added to the errata documentation.

Changes from document Revision Z to Revision AA.

1. RTC16 was added to the errata documentation.

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