Errata

MSP430FR5723 Microcontroller

ABSTRACT

This document describes the known exceptions to the functional specifications (advisories).

Table of Contents

Copyright © 2021 Texas Instruments Incorporated

1 Functional Advisories

Advisories that affect the device's operation, function, or parametrics.

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

Errata Number	Rev J	Rev H
COMP10	✓	✓
COMP11	✓	✓
CPU46	✓	✓
CPU47	✓	✓
CS12	✓	✓
DMA7	✓	✓
DMA9	✓	✓
DMA10	✓	✓
DMA14	✓	✓
GC3	✓	✓
GC4	✓	✓
MPY1	✓	✓
PORT16	✓	✓
PORT19	✓	✓
PORT26	✓	✓
RTC14	✓	✓
TA23	✓	✓
TB25	✓	✓
USCI36	✓	✓
USCI37	✓	✓
USCI41	✓	✓
USCI42	✓	✓
USCI44	✓	✓
USCI47	✓	✓
USCI50	✓	✓
WDG6	✓	✓
XOSC13	✓	✓

2 Preprogrammed Software Advisories

Advisories that affect factory-programmed software.

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

The device does not have any errata for this category.

3 Debug Only Advisories

Advisories that affect only debug operation.

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

Errata Number	Rev J	Rev H
EEM19	✓	✓
EEM23	✓	✓
EEM25	✓	✓

Copyright © 2021 Texas Instruments Incorporated

Errata Number	Rev J	Rev H
EEM30	✓	✓
EEM31	✓	✓
JTAG27	✓	✓

4 Fixed by Compiler Advisories

Advisories that are resolved by compiler workaround. Refer to each advisory for the IDE and compiler versions with a workaround.

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

Errata Number	Rev J	Rev H
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

Copyright © 2021 Texas Instruments Incorporated

5 Nomenclature, Package Symbolization, and Revision Identification

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.

5.1 Device Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all MSP MCU devices. Each MSP MCU commercial family member has one of two prefixes: MSP or XMS. These prefixes represent evolutionary stages of product development from engineering prototypes (XMS) through fully qualified production devices (MSP).

XMS – Experimental device that is not necessarily representative of the final device's electrical specifications

MSP – Fully qualified production device

Support tool naming prefixes:

X: Development-support product that has not yet completed Texas Instruments internal qualification testing.

null: Fully-qualified development-support product.

XMS devices and X development-support tools are shipped against the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

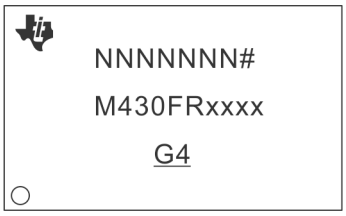
MSP devices have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

Predictions show that prototype devices (XMS) have a greater failure rate than the standard production devices. TI recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are to be used.

TI device nomenclature also includes a suffix with the device family name. This suffix indicates the temperature range, package type, and distribution format.

5.2 Package Markings

DA38 ***TSSOP (DA), 38 Pin***



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

RHA40 ***QFN (RHA), 40 Pin***

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

5.3 Memory-Mapped Hardware Revision (TLV Structure)

Die Revision	TLV Hardware Revision
Rev J	26h

Copyright © 2021 Texas Instruments Incorporated

Die Revision	TLV Hardware Revision
Rev H	24h

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 © 2021 Texas Instruments Incorporated

6 Advisory Descriptions

COMP10	*COMP 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.

COMP11	*COMP Module*
Category	Functional
Function	Polling comparator interrupts may result in a missed interrupt
Description	Polling the comparator output interrupt and the output inverted interrupt flags (CEIFG.CExINT and CEIIFG.CExINT) may result in a missed interrupt if the flags are modified while being read.
Workaround	Using an interrupt service routine to service CEIFG and CEIIFG interrupts significantly reduces the probability of the issue occurring.

CPU21	*CPU 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 © 2021 Texas Instruments Incorporated

CPU21 (continued) ***CPU Module***

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 *CPU 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	

CPU40 *CPU 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

Copyright © 2021 Texas Instruments Incorporated

CPU40 ***CPU Module***

@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 ***CPU 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
```


Copyright © 2021 Texas Instruments Incorporated

CPU46 *CPU Module*

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 *CPU 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.

CS12 *CS Module*

Category Functional

Function DCO overshoot at frequency change

Description When changing frequencies (CSCTL1.DCOFSEL), the DCO frequency may overshoot and exceed the datasheet specification. After a time period of 10us has elapsed, the frequency overshoot settles down to the expected range as specified in the datasheet. The overshoot occur when switching to and from any DCOFSEL setting and impacts all peripherals using the DCO as a clock source. A potential impact can also be seen on

Copyright © 2021 Texas Instruments Incorporated

CS12 (continued) ***CS Module***

FRAM accesses, since the overshoot may cause a temporary violation of FRAM access and cycle time requirements.

Workaround When changing the DCO settings, use the following procedure:

1) Store the existing CSCTL3 divider into a temporary unsigned 16-bit variable

2) Set CSCTL3 to divide all corresponding clock sources by 4 or higher

3) Change DCO frequency

4) Wait ~10us

5) Restore the divider in CSCTL3 to the setting stored in the temporary variable.

The following code example shows how to increase DCO to 16MHz.

```
uint16_t tempCSCTL3 = 0;
CSCTL0_H = CSKEY_H;         // Unlock CS registers
/* Assuming SMCLK and MCLK are sourced from DCO */
/* Store CSCTL3 settings to recover later */
tempCSCTL3 = CSCTL3;
/* Keep overshoot transient within specification by setting clk sources to
divide by 4*/
/* Clear the DIVS & DIVM masks (~0x77)and set both fields to 4 divider */
CSCTL3 = CSCTL3 & (~(0x77)) | DIVS__4 | DIVM__4;
CSCTL1 = DCOFSEL_4 | DCORSEL;        // Set DCO to 16MHz
/* Delay by ~10us to let DCO settle. 60 cycles = 20 cycles buffer + (10us /
(1/4MHz)) */
__delay_cycles(60);
CSCTL3 =  tempCSCTL3;       // Set all dividers
CSCTL0_H = 0;                     // Lock CS registers
```

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.

DMA9 ***DMA Module***

Category Functional

Function DMA stops transferring bytes unexpectedly

Copyright © 2021 Texas Instruments Incorporated

DMA9 (continued)	***DMA Module***
Description	When the DMA is configured to transfer bytes from the eUSCI_A or eUSCI_B transmit or receive buffers, the transmit or receive triggers (TXIFG and RXIFG) may not be seen by the DMA module and the transfer of the bytes is missed. Once the first byte in a transfer sequence is missed, all the following bytes are missed as well. All eUSCI_A modes (UART, SPI, and IrDA) and all eUSCI_B modes (SPI and I2C) are affected.
Workaround	1) Use Interrupt Service Routines to transfer data to and from the eUSCI_A or eUSCI_B. OR 2) When using DMA channel 0 for transferring data to and from the eUSCI_A or eUSCI_B, use DMA channel 2 (lower priority than DMA channel 0) to read the same register of the eUSCI_A or eUSCI_B that DMA channel 0 is working with. Use the same USCI IFG (e.g. UCA0RXIFG) as trigger source for these both DMA channels.

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.

DMA14	***DMA Module***
Category	Functional
Function	DMA misses trigger from asynchronous source and all subsequent transfers/triggers
Description	If DMA module is used in edge-triggered mode AND the trigger source (e.g. TimerA, TimerB, USCI or ADC10) is running with an asynchronous clock (e.g. MODOSC, VLO or external crystal clock) versus DMA clock (MCLK), AND the DMA trigger occurs while the CPU is in active mode, then the DMA module might miss an edge trigger and then all subsequent edge triggers. This leads to a missing DMA transfer cycle. The DMA size address register (DMAxSZ) will note decrement, and the trigger source (corresponding module flag) is not cleared. Due to this, the expected DMA interrupt is not executed and the DMA hangs.
Workaround	To prevent the issue entirely, run DMA and module from the same clock source(e.g. MCLK, SMCLK with same source as MCLK, etc) to prevent asynchronous events. To detect the issue: Use overflow flags found in many modules (e.g. ADC10OVIFG) to detect DMA lockup OR Use the WDT at the system-level to detect a hang-up. After detection, recover operation by clearing the DMA trigger source interrupt flag, clearing the data to be transferred, and re-initializing both the DMA and the module for the

Copyright © 2021 Texas Instruments Incorporated

DMA14 (continued) ***DMA Module***

trigger source. If the trigger source is unknown, trigger a software BOR reset by setting the PMMSWBOR bit in the PMMCTL0 register.

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.

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.

EEM25 ***EEM Module***

Category Debug

Function Unexpected wakeup from LPMx.5

Description When the device is in LPMx.5, debugging in SBW mode can cause an unexpected wakeup from the low power mode.

Workaround Use 4-wire JTAG when debugging.

Note
This issue only affects debug mode.

Copyright © 2021 Texas Instruments Incorporated

EEM30	*EEM Module*
Category	Debug
Function	Missed breakpoint if FRAM power supply is disabled
Description	The FRAM power supply can be disabled (GCCTL0.FRPWR = 0) prior to LPM entry to save power. Upon wakeup, if a breakpoint is set on an the first instruction that accesses FRAM, the breakpoint may be missed.
Workaround	None. This issue affects debug mode only.

EEM31	*EEM Module*
Category	Debug
Function	Breakpoint trigger may be lost when MPU is enabled
Description	A data value written to FRAM can be used as a trigger condition for breakpoints during a debug session. This trigger can be lost if the FRAM access is made to an address that has been write-protected by the MPU.
Workaround	None. This issue affects debug mode only.

GC3	*GC Module*
Category	Functional
Function	Error flags set incorrectly after reset or wake-up from LPM
Description	The error flags GCCTL1.UBDIFG and GCCTL1.CBDIFG are incorrectly set after the first power-up or wake up from LPM2/LPM3/LPM4/LPM3.5/LPM4.5. If the corresponding interrupt enable bits are set, then an erroneous interrupt can be generated.
Workaround	Disable the UBDIEN and CBDIEN interrupts (GCCTL0.UBDIEN=0 and GCCTL0.CBDIEN=0) prior to entering LPM2/LPM3/LPM4/LPM4.5 . After LPM exit, re-initialize the interrupt flags only after the first FRAM access has been completed.

GC4	*GC Module*
Category	Functional
Function	Unexpected PUC is triggered
Description	During execution from FRAM a non-existent uncorrectable bit error can be detected and trigger a PUC if the uncorrectable bit error detection flag is set (GCCTL0.UBDRSTEN = 1). This behavior appears only if: (1) MCLK is sourced from DCO frequency of 16 MHz OR (2) MCLK is sourced by external high frequency clock above 12 MHz at pin HFXIN OR (3) MCLK is sourced by High-Frequency crystals (HFXT) above 12 MHz. This PUC will not be recognized by the SYSRSTIV register (SYSRSTIV = 0x00).

Copyright © 2021 Texas Instruments Incorporated

GC4 (continued)	***GC Module***
	A PUC RESET will be executed with not defined reset source. Also the corresponding bit error detection flag is not set (GCCTL1.UBDIFG = 0).
Workaround	1. Check the reset source for SYSRSTIV = 0 and ignore the reset. OR 2. Set GCCTL0.UBDRSTEN = 0 to prevent unexpected PUC. OR 3. Set the MCLK to maximum 12MHz to leverage the uncorrectable bit error PUC feature.
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

Copyright © 2021 Texas Instruments Incorporated

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.

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

PORT26 ***PORT Module***

Category Functional

Function Incorrect values for P1.3 / P1.4 input pins during power-up

Description If P1.3/P1.4 is pulled up externally to DVCC during power-up the logical HIGH value might not be read correct by the device (ZERO is read instead of ONE).

Workaround 1) Switch the P1.3/P1.4 Port to logical ZERO after power cycle by:
a) Switch critical GPIO to output-low (with series resistance to limit current) or
b) Remove external pull up connection to pull GPIO via internal pull-down

OR

Copyright © 2021 Texas Instruments Incorporated

PORT26 (continued)	***PORT Module***
	2) Use different GPIOs (not P1.3 & P1.4) OR 3) Change the polarity of the logical check in SW (enable internal pull-up resistor for the GPIO and pull the external pin to DVSS)
RTC14	***RTC Module***
Category	Functional
Function	Polling RTC interrupts may result in a lost interrupt flag
Description	Polling any RTC interrupt flag mapped to the RTCIV register may result in a missed interrupt if the flags are modified while being read.
Workaround	Using an interrupt service routine to service flags mapped to the RTCIV register significantly reduces the probability of the issue occurring.
TA23	***TA Module***
Category	Functional
Function	Polling timer interrupts may result in a lost interrupt flag
Description	Polling any Timer interrupt flag may result in a missed interrupt if the flags are modified while being read.
Workaround	Using an interrupt service routine to service timer interrupt flags significantly reduces the probability of the issue occurring.
TB25	***TB Module***
Category	Functional
Function	In up mode, TBxCCRn value is immediately transferred to TBxCLn when TBxCCTLn.CLLD bits are set or 0x01 or 0x10
Description	IF Timer B is configured for Up mode, AND the compare latch load event (TBxCCTLn.CLLD bits) setting is configured to update TBxCCRn when TBxR reaches 0, THEN TBxCCRn will update immediately instead of the described condition. This is contrary to the user guide description of TBxCCTLn.CLLD = 0x01 or 0x10 modes.
Workaround	If user needs to update TBxCCRn value when TBxR counts to 0 in Timer B up mode: 1. Set TBxCCTLn. CLLD = 0x00 2. Enable the Timer B interrupt (TBIE) in TBxCTL 3. Update TBxCCRn value within interrupt routine. Timer B Interrupt would need to be serviced in a timely manner to mitigate disruption or unintended timer output if an output mode is used.

Copyright © 2021 Texas Instruments Incorporated

USCI36	*USCI Module*
Category	Functional
Function	UCLKI not usable in I2C master mode
Description	When EUSCIB is configured as I2C Master with the external UCLKI as clock source, the UCLKI signal is not available and cannot be used to source I2C clock.
Workaround	Use LFXTCLK via ACLK or HFXTCLK via SMCLK as clock source (BRCLK) for I2C in master mode with external clock source.

USCI37	*USCI Module*
Category	Functional
Function	Reading RXBUF during an active I2C communication might result in unintended bus stalls.
Description	The falling edge of SCL bus line is used to set an internal RXBUF-written flag register, which is used to detect a potential RXBUF overflow. If this flag is cleared with a read access from the RXBUF register during a falling edge of SCL, the clear condition might be missed. This could result in an I2C bus stall at the next received byte.
Workaround	(1) Execute two consecutive reads of RXBUF, if $t_{SCL} > 4 \times t_{MCLK}$. or (2) Provoke an I2C bus stall before reading RXBUF. A bus stall can be verified by checking if the clock line low status indicator bit UCSCLLOW is set for at least three USCI bit clock cycles i.e. $3 \times t_{BitClock}$.

USCI41	*USCI Module*
Category	Functional
Function	UCBUSY bit of eUSCIA module might not work reliable when device is in SPI mode.
Description	When eUSCIA is configured in SPI mode, the UCBUSY bit might get stuck to 1 or start toggling after transmission is completed. This happens in all four combinations of Clock Phase and Clock Polarity options (UCAxCTLW0.UCCKPH & UCAxCTLW0.UCCKPL bits) as well as in Master and Slave mode. There is no data loss or corruption. However the UCBUSY cannot be used in its intended function to check if transmission is completed. Because the UCBUSY bit is stuck to 1 or toggles, the clock request stays enabled and this adds additional current consumption in low power mode operation.
Workaround	For correct functional implementation check on transmit or receive interrupt flag UCTXIFG/UCRXIFG instead of UCBUSY to know if the UCAxTXBUF buffer is empty or ready for the next complete character. To reduce the additional current it is recommended to either reset the SPI module (UCAxCTLW0.UCSWRST) in the UCBxCTLW0 or send a dummy byte 0x00 after the intended SPI transmission is completed.

USCI42	*USCI Module*
Category	Functional
Function	UART asserts UCTXCPTIFG after each byte in multi-byte transmission

Copyright © 2021 Texas Instruments Incorporated

USCI42 (continued) ***USCI Module***

Description UCTXCPTIFG flag is triggered at the last stop bit of every UART byte transmission, independently of an empty buffer, when transmitting multiple byte sequences via UART. The erroneous UART behavior occurs with and without DMA transfer.

Workaround None.

USCI44 ***USCI Module***

Category Functional

Function Differing clock sources may cause UART communication failure

Description When using the USCI_A UART module with differing clock sources for the system clock (MCLK) and the UART source clock (BRCLK), any read or write of the UCAxIFG, UCAxCTLW0, UCAxSTATW, UCAxRXBUF, UCAxTXBUF, UCAxABCTL & UCAxIV registers, while the UCATXIFG or UCARXIFG flag is being set by a UART event could unintentionally clear the UCATXIFG or UCARXIFG. This may result in the UART communication being stalled.

Workaround Workaround 1: Use synchronous clocks to source BRCLK and MCLK. Note that the clock frequencies need not be identical and dividers may be used as long as they are using the same clock source.

Workaround 2: Avoid polling UCAxTXIFG and UCAxRXIFG. Using the standard interrupt service routine to service the interrupt flag significantly reduces access to the USCI registers & hence reduces the probability of this errata occurring.
Also, limit all accesses of the UCAxCTLW0, UCAxSTATW, UCAxRXBUF, UCAxTXBUF, UCAxABCTL, UCAxIFG, & UCAxIV registers while transmit or receive operation is ongoing (and UCAxRXIFG or UCAxTXIFG is expected to be set) as this can further reduce the probability of this errata occurring.

USCI47 ***USCI Module***

Category Functional

Function eUSCI SPI slave with clock phase UCCKPH = 1

Description The eUSCI SPI operates incorrectly under the following conditions:

1. The eUSCI_A or eUSCI_B module is configured as a SPI slave with clock phase mode UCCKPH = 1

AND

2. The SPI clock pin is not at the appropriate idle level (low for UCCKPL = 0, high for UCCKPL = 1) when the UCSWRST bit in the UCxxCTLW0 register is cleared.

If both of the above conditions are satisfied, then the following will occur:
eUSCI_A: the SPI will not be able to receive a byte (UCAxRXBUF will not be filled and UCRXIFG will not be set) and SPI slave output data will be wrong (first bit will be missed and data will be shifted).
eUSCI_B: the SPI receives data correctly but the SPI slave output data will be wrong (first byte will be duplicated or replaced by second byte).

Workaround Use clock phase mode UCCKPH = 0 for MSP SPI slave if allowed by the application.

Copyright © 2021 Texas Instruments Incorporated

USCI47 (continued) ***USCI Module***

OR

The SPI master must set the clock pin at the appropriate idle level (low for UCCKPL = 0, high for UCCKPL = 1) before SPI slave is reset (UCSWRST bit is cleared).

OR

For eUSCI_A: to detect communication failure condition where UCRXIFG is not set, check both UCRXIFG and UCTXIFG. If UCTXIFG is set twice but UCRXIFG is not set, reset the MSP SPI slave by setting and then clearing the UCSWRST bit, and inform the SPI master to resend the data.

USCI50 ***USCI Module***

Category Functional

Function Data may not be transmitted correctly from the eUSCI when operating in SPI 4-pin master mode with UCSTEM = 0

Description When the eUSCI is used in SPI 4-pin master mode with UCSTEM = 0 (STE pin used as an input to prevent conflicts with other SPI masters), data that is moved into UCxTXBUF while the UCxSTE input is in the inactive state may not be transmitted correctly. If the eUSCI is used with UCSTEM = 1 (STE pin used to output an enable signal), data is transmitted correctly.

Workaround When using the STE pin in conflict prevention mode (UCSTEM = 0), only move data into UCxTXBUF when UCxSTE is in the active state. If an active transfer is aborted by UCxSTE transitioning to the master-inactive state, the data must be rewritten into UCxTXBUF to be transferred when UCxSTE transitions back to the master-active state.

WDG6 ***WDG Module***

Category Functional

Function Clock Fail-Safe feature in LPMx.5

Description The watchdog clock fail-safe feature does not prevent the device from going into LPMx.5. As a result, the device enters LPMx.5 state independently of running the watchdog. Note that the watchdog is off in LPMx.5.

Workaround None.

XOSC13 ***XOSC Module***

Category Functional

Function XT1 in bypass mode does not work with RTC enabled

Description When the RTC module is enabled and XT1 is configured in bypass mode to be sourced by an external digital signal, the XT1OFFG flag is set indefinitely, resulting in ACLK defaulting its clock source to VLO instead of XT1.

Workaround Do not use RTC module with XT1 in bypass mode with external digital clock source.

Copyright © 2021 Texas Instruments Incorporated

7 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from July 14, 2021 to August 27, 2021	Page
• TB25 was added to the errata documentation	6
• TB25 Description was updated	16
• TB25 Workaround was updated	16

Copyright © 2021 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 (https: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 © 2021, Texas Instruments Incorporated