MSP430F247 Device Erratasheet

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	Rev C	Rev B	Rev A
ADC25	✓	✓	✓	✓	✓
BCL12	✓	✓	✓	✓	✓
BCL13			✓	✓	✓
BCL15	✓	✓	✓	✓	✓
COMP2					✓
FLASH19	✓	✓	✓	✓	✓
FLASH24	✓	✓	✓	✓	✓
FLASH25			✓	✓	✓
FLASH27	✓	✓	✓	✓	✓
FLASH36	✓	✓	✓	✓	✓
PORT11				✓	✓
PORT12	✓	✓	✓	✓	✓
TA12	✓	✓	✓	✓	✓
TA16	✓	✓	✓	✓	✓
TA21	✓	✓	✓	✓	✓
TAB22	✓	✓	✓	✓	✓
TB2	✓	✓	✓	✓	✓
TB16	✓	✓	✓	✓	✓
TB24	✓	✓	✓	✓	✓
USCI20	✓	✓	✓	✓	✓
USCI21		✓	✓	✓	✓
USCI22	✓	✓	✓	✓	✓
USCI23	✓	✓	✓	✓	✓
USCI24	✓	✓	✓	✓	✓
USCI25	✓	✓	✓	✓	✓
USCI26	✓	✓	✓	✓	✓
USCI28	✓	✓	✓	✓	✓
USCI30	✓	✓	✓	✓	✓
USCI35	✓	✓	✓	✓	✓
USCI40	✓	✓	✓	✓	✓
XOSC5	✓	✓	✓	✓	✓
XOSC6	✓	✓	✓	✓	✓
XOSC8		✓	✓	✓	✓

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.

The device doesn't have Software in ROM errata.

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	Rev C	Rev B	Rev A
JTAG23	✓	✓	✓	✓	✓

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	Rev C	Rev B	Rev A
CPU19	✓	✓	✓	✓	✓

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

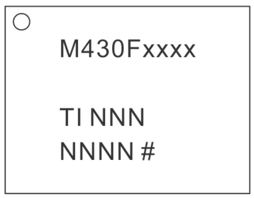
PM64

LQFP (PM), 64 Pin

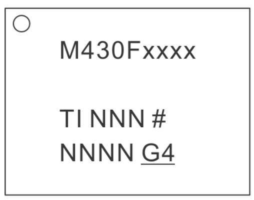
= Die revision
○ = Pin 1 location
N = Lot trace code

RGC64

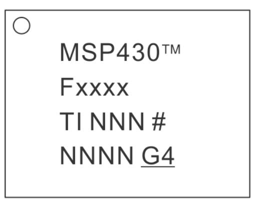
QFN (RGC), 64 pin



= 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.

6 Detailed Bug Description

ADC25 *ADC12 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.

BCL12 *BCS Module*

Category Functional

Function Switching RSELx or modifying DCOCTL can cause DCO dead time or a complete DCO stop

Description After switching RSELx bits (located in register BCSCTL1) from a value of >13 to a value of <12 OR from a value of <12 to a value of >13, the resulting clock delivered by the DCO can stop before the new clock frequency is applied. This dead time is approximately 20 us. In some instances, the DCO may completely stop, requiring a power cycle.

Furthermore, if all of the RSELx bits in the BSCTL1 register are set, modifying the DCOCTL register to change the DCOx or the MODx bits could also result in DCO dead time or DCO hang up.

Workaround - When switching RSEL from >13 to <12, use an intermediate frequency step. The intermediate RSEL value should be 13.

Current RSEL	Target RSEL	Recommended Transition Sequence
15	14	Switch directly to target RSEL
14 or 15	13	Switch directly to target RSEL
14 or 15	0 to 12	Switch to 13 first, and then to target RSEL (two step sequence)
0 to 13	0 to 12	Switch directly to target RSEL

AND

- When switching RSEL from <12 to >13 it's recommended to set RSEL to its default value first (RSEL = 7) before switching to the desired target frequency.

AND

- In case RSEL is at 15 (highest setting) it's recommended to set RSEL to its default value first (RSEL = 7) before accessing DCOCTL to modify the DCOx and MODx bits. After the DCOCTL register modification the RSEL bits can be manipulated in an additional step.

In the majority of cases switching directly to intermediate RSEL steps as described above will prevent the occurrence of BCL12. However, a more reliable method can be implemented by changing the RSEL bits step by step in order to guarantee safe function without any dead time of the DCO.

Note that the 3-step clock startup sequence consisting of clearing DCOCTL, loading the BCSCTL1 target value, and finally loading the DCOCTL target value as suggested in the in the "TLV Structure" chapter of the MSP430x2xx Family User's Guide is not affected by BCL12 if (and only if) it is executed after a device reset (PUC) prior to any other modifications being made to BCSCTL1 since in this case RSEL still is at its default value of 7. However any further changes to the DCOx and MODx bits will require the consideration of the workaround outlined above.

BCL13 — *BCS Module*

Category Functional

Function DCO powerup halt

Description When subject to very slow Vcc rise times, the device may enter into a state where the DCO does not oscillate. No JTAG access or program execution is possible and the device will remain in a reset state until the supply voltage is disconnected.

Workaround Apply a Vcc poweron ramp >= 10V/second under all power-on/power-cycle scenarios.

BCL15 — *BCS Module*

Category Functional

Function Unpredictable device behavior if XT2 is sourcing SMCLK or MCLK while operating in LPM3

Description If the MCLK or SMCLK is sourced by the XT2 oscillator, when the device wakes up from LPM3 or the SMCLK is requested by the USCI module an unpredictable glitch might appear. The glitch might appear on the corresponding clock signal with the 1st rising edge of the ACLK after wake-up. This can lead to a frequency violation.

In case of MCLK it can cause the device to hang up or execute code incorrectly.

In case of SMCLK any corresponding module using the clock can behave unpredictably.

Workaround Do not use XT2 clock for MCLK/SMCLK when using LPM3

COMP2 — *COMP_A+ Module*

Category Functional

Function Configuring the port disable register (CAPD)

Description According to the user's guide, each bit in the CAPD register should correspond with its associated port I/O number. For example, when "bit 0" of CAPD is set, the port disable function of pin Px.0 is enabled; "bit 1" controls Px.1, and so on (where Px is the port that contains the comparator inputs). However, on this device, the bits of the CAPD register correspond with the Comparator_A input number. For example, "bit 0" of CAPD controls the CA0 input, "bit 1" controls CA1, etc. This difference matters when the port I/O number is not the same as the comparator input number.

If the wrong CAPD bit is set, the port I/O function for the wrong pin will be disabled. Also, the analog signal applied to the comparator input pin being used may cause a parasitic current to flow from Vcc to GND. See the Comparator_A+ chapter of the MSP430x2xx Family User's Guide (SLAU144) for more information on CAPD.

Workaround None

CPU19 *CPU Module*

Category Compiler-Fixed

Function CPUOFF modification may result in unintentional register read

Description If an instruction that modifies the CPUOFF bit in the Status Register is followed by an instruction with an indirect addressed operand (e.g. MOV @R8, R9, RET, POP, POPM), an unintentional register read operation can occur during the wakeup of the CPU. If the unintentional read occurs to a read sensitive register (e.g. UCB0RXBUF, TAIV), which changes its value or the value of other registers (IFG's), the bug leads to lost interrupts or wrong register read values.

Workaround Insert a NOP instruction after each CPUOFF instruction.

OR

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

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

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v6.20.1 until v6.40	User is required to add the compiler or assembler flag option below. --hw_workaround=nop_after_lpm
IAR Embedded Workbench	IAR EW430 v6.40 or later	Workaround is automatically enabled
TI MSP430 Compiler Tools (Code Composer Studio)	15.12.0.LTS	User is required to add the compiler or assembler flag option below. --silicon_errata=CPU19
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 389 or later	User is required to add the compiler or assembler flag option below. -msilicon-errata=cpu19 -msilicon-errata-warn=cpu19 generates a warning in addition
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 5.x build 14 or later	User is required to add the compiler or assembler flag option below. -msilicon-errata=cpu19 -msilicon-errata-warn=cpu19 generates a warning in addition

FLASH19 *FLASH Module*

Category Functional

Function EEI feature does not work for code execution from RAM

Description When the program is executed from RAM, the flash controller EEI feature does not work. The erase cycle is suspended and the interrupt is serviced, but there is a problem while resuming with the erase cycle.

Addresses applied to flash are different than the actual values while resuming erase cycle after ISR execution.

Workaround None

FLASH24 *FLASH Module*

Category Functional

Function Write or erase emergency exit can cause failures

Description When a flash write or erase is abruptly terminated, the following flash accesses by the CPU may be unreliable resulting in erroneous code execution. The abrupt termination can be the result of one the following events:

1) The flash controller clock is configured to be sourced by an external crystal. An oscillator fault occurs thus stopping this clock abruptly.

or

2) The Emergency Exit bit (EMEX in FCTL3) when set forces a write or an erase operation to be terminated before normal completion.

or

3) The Enable Emergency Interrupt Exit bit (EEIEX in FCTL1) when set with GIE=1 can lead to an interrupt causing an emergency exit during a Flash operation.

Workaround 1) Use the internal DCO as the flash controller clock provided from MCLK or SMCLK.

or

2) After setting EMEX = 1, wait for a sufficient amount of time before Flash is accessed again.

or

3) No Workaround. Do not use EEIEX bit.

FLASH25 *FLASH Module*

Category Functional

Function Marginal Read Mode is not functional

Description The control bits for marginal read mode contained in the FCTL4 register are automatically cleared by any flash access. This prevents the marginal read mode from being used.

Workaround It is possible to read out memory contents in marginal read mode if the indexed addressing mode X(Ry) is used to access the flash memory. In this case, the FCTL4 control bits are not cleared, and the marginal read mode works as expected. It is recommended to write the code for reading the flash memory contents in assembler as this allows full control over the used addressing mode. Note that certain assemblers may optimize an indexed addressing source operation of 0(Ry) to an indirect register mode @Ry operation, which will not work. The following is an example of reading the word memory location 0x4000 in marginal read mode, preventing a possible assembler optimization:

```
mov.w #0x4000,R15 ; Pointer to target address
dec.w R15 ; Decrement pointer
mov.w 1(R15),R12 ; Read memory contents at R15+1, store result in R12
```

FLASH27 *FLASH Module*

Category Functional

Function EEI feature can disrupt segment erase

Description When a flash segment erase operation is active with EEI feature selected (EEI=1 in FLCTL1) and GIE=0, the following can occur:

An interrupt event causes the flash erase to be stopped, and the flash controller expects an RETI to resume the erase. Because GIE=0, interrupts are not serviced and RETI will never happen.

Workaround 1) Do not set bit EEI=1 when GIE = 0.

or,

2) Force an RETI instruction during the erase operation during the check for BUSY=1 (FCLTL3).

Sample code:

MOV R5, 0(R5) ; Dummy write, erase segment

LOOP: BIT #BUSY, &FCTL3 ; test busy bit

JMP SUB_RETI ; Force RETI instruction

JNZ LOOP ; loop while BUSY=1

SUB_RETI: PUSH SR

RETI

FLASH36 *FLASH Module*

Category Functional

Function Flash content may degrade due to aborted page erases

Description If a page erase is aborted by EEIEX, the flash page containing the last instruction before erase operation will start to degrade. This effect is incremental and, after repetitions, may lead to corrupted flash content.

Workaround - Use the EEI (interrupted erasing) feature instead of EEIEX (abort erasing).

or

- A PSA checksum can be calculated over affected flash page using the marginal read mode (marginal 0). If PSA sum differs from expected PSA value the affected flash page has to be reprogrammed.

or

- Start flash erasing from RAM and limit system frequency to <1MHz (to ensure 6-us delay after EEIEX). If the last instruction before erasing is located in RAM, flash cell degradation does not occur.

JTAG23 *JTAG Module*

Category Debug

Function PSA checksum calculation does not work in marginal read mode.

Description If the PSA checksum is calculated via JTAG interface in marginal read mode the MRG0 and MRG1 bits in the FCTL4 register are reset.

Workaround None.

PORT11 *PORT Module*

Category Functional

Function Pullup for P3.6 controlled by bit 0

Description According to the user's guide, the internal pullup of an I/O should be enabled when a corresponding bit from PxREN and PxOUT are both set. For example, in the case of P3.6, this should be bit 6. However, P3.6 is currently controlled by bit 0 instead. Bit 0 also controls P3.0, as expected. The pulldown resistors operate properly and are not affected by this errata.

Workaround If bit 6 of PxREN is set, bits 0 and 6 of PxOUT should be set/cleared together. If P3.6 is to be configured for pullup/down, P3.0 must have the same configuration. So the workaround options are:

- Configure both P3.0 and P3.6 with pulldowns. (bits 0/6 of PxREN set, bits 0/6 of PxOUT cleared)

- Configure both P3.0 and P3.6 with pullups. (bits 0/6 of PxREN set/cleared, bits 0/6 of PxOUT set)

- Do not use the pullup/pulldown feature on these pins. (bits 0/6 of PxREN cleared)

PORT12 *PORT Module*

Category Functional

Function PxIFG is set on PUC

Description The PxIN register is cleared when a PUC is asserted, and it regains the original value after the PUC is de-asserted. If the PxIN register bits read high, asserting a PUC causes clearing of the register, which results in a high-to-low transition. Once the PUC is de-asserted, the PxIN register is restored to high, which results in a low-to-high transition. This behavior results in the PxIFG being set regardless of the PxIES setting.

Workaround Prior to setting PxIE bits ensure that corresponding PxIFG bits are cleared.

TA12 *TIMER_A Module*

Category Functional

Function Interrupt is lost (slow ACLK)

Description Timer_A counter is running with slow clock (external TACLK or ACLK)compared to MCLK. The compare mode is selected for the capture/compare channel and the CCRx register is incremented by one with the occurring compare interrupt (if TAR = CCRx). Due to the fast MCLK the CCRx register increment (CCRx = CCRx+1) happens before the Timer_A counter has incremented again. Therefore the next compare interrupt should happen at once with the next Timer_A counter increment (if TAR = CCRx + 1). This interrupt gets lost.

Workaround Switch capture/compare mode to capture mode before the CCRx register increment.

Switch back to compare mode afterwards.

TA16 — *TIMER_A Module*

Category Functional

Function First increment of TAR erroneous when IDx > 00

Description The first increment of TAR after any timer clear event (POR/TACLR) happens immediately following the first positive edge of the selected clock source (INCLK, SMCLK, ACLK or TACLK). This is independent of the clock input divider settings (ID0, ID1). All following TAR increments are performed correctly with the selected IDx settings.

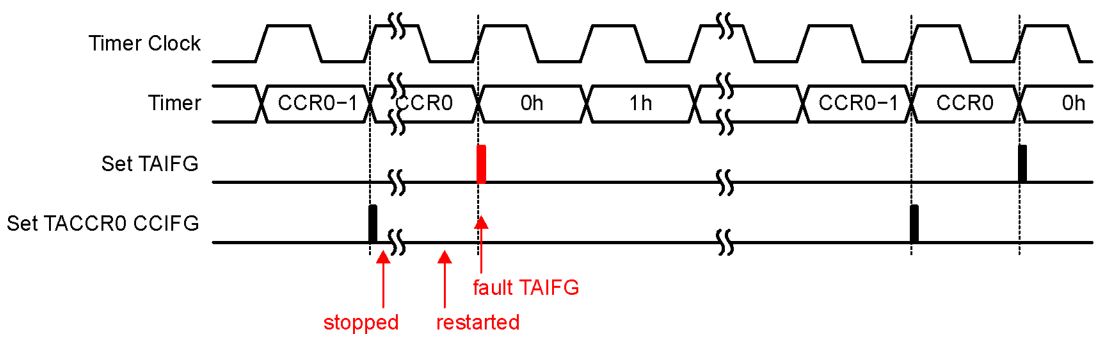
Workaround None

TA21 — *TIMER_A Module*

Category Functional

Function TAIFG Flag is erroneously set after Timer A restarts in Up Mode

Description In Up Mode, the TAIFG flag should only be set when the timer counts from TACCR0 to zero. However, if the Timer A is stopped at TAR = TACCR0, then cleared (TAR=0) by setting the TACLR bit, and finally restarted in Up Mode, the next rising edge of the TACLK will erroneously set the TAIFG flag.



Workaround None.

TAB22 — *TIMER_A/TIMER_B Module*

Category Functional

Function Timer_A/Timer_B register modification after Watchdog Timer PUC

Description Unwanted modification of the Timer_A/Timer_B registers TACTL/TBCTL and TAIV/TBIV can occur when a PUC is generated by the Watchdog Timer(WDT) in Watchdog mode and any Timer_A/Timer_B counter register TACCRx/TBCCRx is incremented/decremented (Timer_A/Timer_B does not need to be running).

Workaround Initialize TACTL/TBCTL register after the reset occurs using a MOV instruction (BIS/BIC may not fully initialize the register). TAIV/TBIV is automatically cleared following this initialization.

Example code:

MOV.W #VAL, &TACTL

or

MOV.W #VAL, &TBCTL

Where, VAL=0, if Timer is not used in application otherwise, user defined per desired function.

TB2 — *TIMER_B Module*

Category Functional

Function Interrupt is lost (slow ACLK)

Description Timer_B counter is running with slow clock (external TBCLK or ACLK) compared to MCLK. The compare mode is selected for the capture/compare channel and the CCRx register is incremented by 1 with the occurring compare interrupt (if TBR = CCRx).

Due to the fast MCLK, the CCRx register increment (CCRx = CCRx + 1) happens before the Timer_B counter has incremented again. Therefore, the next compare interrupt should happen at once with the next Timer_B counter increment (if TBR = CCRx + 1). This interrupt is lost.

Workaround Switch capture/compare mode to capture mode before the CCRx register increment. Switch back to compare mode afterward.

TB16 — *TIMER_B Module*

Category Functional

Function First increment of TBR erroneous when IDx > 00

Description The first increment of TBR after any timer clear event (POR/TBCLR) happens immediately following the first positive edge of the selected clock source (INCLK, SMCLK, ACLK, or TBCLK). This is independent of the clock input divider settings (ID0, ID1). All following TBR increments are performed correctly with the selected IDx settings.

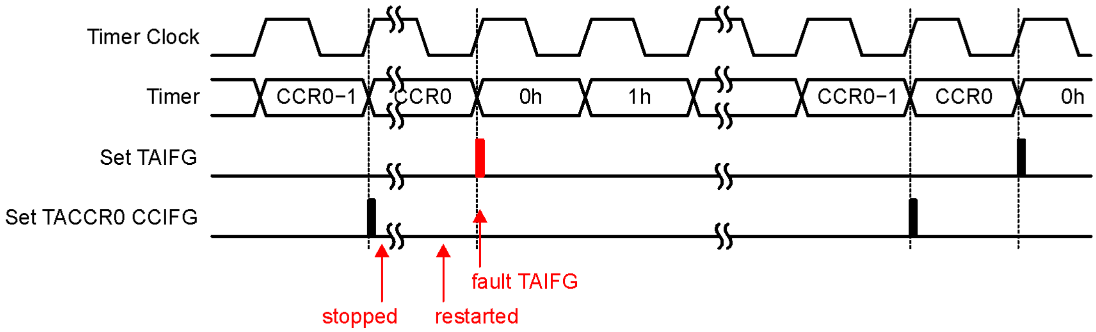
Workaround None

TB24 — *TIMER_B Module*

Category Functional

Function TBIFG Flag is erroneously set after Timer B restarts in Up Mode

Description In Up Mode, the TBIFG flag should only be set when the timer resets from TBCCR0 to zero. However, if the Timer A is stopped at TBR = TBCCR0, then cleared (TBR=0) by setting the TBCLR bit, and finally restarted in Up Mode, the next rising edge of the TBCLK will erroneously set the TBIFG flag.



Workaround None.

USCI20 *USCI Module*

Category Functional

Function I2C Mode Multi-master transmitter issue

Description When configured for I2C master-transmitter mode, and used in a multi-master environment, the USCI module can cause unpredictable bus behavior if all of the following four conditions are true:

1 - Two masters are generating SCL

And

2 - The slave is stretching the SCL low phase of an ACK period while outputting NACK on SDA

And

3 - The slave drives ACK on SDA after the USCI has already released SCL, and then the SCL bus line gets released

And

4 - The transmit buffer has not been loaded before the other master continues communication by driving SCL low

The USCI will remain in the SCL high phase until the transmit buffer is written. After the transmit buffer has been written, the USCI will interfere with the current bus activity and may cause unpredictable bus behavior.

Workaround 1 - Ensure that slave doesn't stretch the SCL low phase of an ACK period

Or

2 - Ensure that the transmit buffer is loaded in time

Or

3 - Do not use the multi-master transmitter mode

USCI21 *USCI Module*

Category Functional

Function UART IrDA receive filter

Description The IrDA receive filter can be used to filter pulses with length UCAIRRXFL configured in

UCAxIRRCTL register. If UCIRRXFE is set the IrDA receive decoder may filter out pulses longer than the configured filter length depending on frequency of BRCLK. This is resulting in framing errors or corrupted data on the receiver side.

Workaround Depending on the used baud rate and the configured filter length a maximum frequency for BRCLK needs to be set to avoid this issue:

For baud rates equal and higher than 115.000 the maximum allowed BRCLK frequency is equal to the max specified system frequency.

$$\text{Max BRCLK} = \frac{\text{Filter Length} + 64}{2} \times \frac{\text{Baud Rate} \times 16}{3 \times 10^6}$$

Baud Rate	Filter Length UCIRRXFL (dec)	Max BRCLK (MHz)
9600	64	3.28
	32	2.46
	16	2.05
	8	1.84
	4	1.74
	2	1.69
	1	1.66
	0	1.64
19200	64	6.55
	32	4.92
	16	4.1
	8	3.69
	4	3.48
	2	3.38
	1	3.33
	0	3.28
38400	64	13.11
	32	9.83
	16	8.19
	8	7.37
	4	6.96
	2	6.76
	1	6.66
	0	6.55
56000	64	19.11
	32	14.34
	16	11.95
	8	10.75
	4	10.15
	2	9.86
	1	9.71
	0	9.56

USCI22 — *USCI Module*

Category Functional

Function I2C Master Receiver with 10-bit slave addressing

Description Unexpected behavior of the USCI_B can occur when configured in I2C master receive mode with 10-bit slave addressing under the following conditions:

1) The USCI sends first byte of slave address, the slave sends an ACK and when second address byte is sent, the slave sends a NACK.

2) Master sends a repeat start condition (If UCTXSTT=1).

3) The first address byte following the repeated start is acknowledged.

However, the second address byte is not sent, instead the Master incorrectly starts to receive data and sets UCBxRXIFG=1.

Workaround Do not use repeated start condition instead set the stop condition UCTXSTP=1 in the NACK ISR prior to the following start condition (USTXSTT=1).

USCI23 — *USCI Module*

Category Functional

Function UART transmit mode with automatic baud rate detection

Description Erroneous behavior of the USCI_A can occur when configured in UART transmit mode with automatic baud rate detection. During transmission if a "Transmit break" is initiated (UCTXBRK=1), the USCI_A will not deliver a stop bit of logic high, instead, it will send a logic low during the subsequent synch period.

Workaround 1) Follow User's Guide instructions for transmitting a break/synch field following UCSWRST=1.

Or,

2) Set UCTXBRK=1 before an active transmission, i.e. check for bit UCBUSY=0 and then set UCTXBRK=1.

USCI24 — *USCI Module*

Category Functional

Function Incorrect baud rate information during UART automatic baud rate detection mode

Description Erroneous behavior of the USCI_A can occur when configured in UART mode with automatic baud rate detection. After automatic baud rate measurement is complete, the UART updates UCAxBR0 and UCAxBR1. Under Oversampling mode (UCOS16=1), for baud rates that should result in UCAxBRx=0x0002, the UART incorrectly reports it as UCAxBRx=0x5555.

Workaround When break/synch is detected following the automatic baud rate detection, the flag UCBRK flag is set to 1. Check if UCAxBRx=0x5555 and correct it to 0x0002.

USCI25 — *USCI Module*

Category Functional

Function TXIFG is not reset when NACK is received in I2C mode

Description When the USCI_B module is configured as an I2C master transmitter the TXIFG is not reset after a NACK is received if the master is configured to send a restart (UCTXSTT=1 & UCTXSTP=0).

Workaround Reset TXIFG in software within the NACKIFG interrupt service routine

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

USCI28 *USCI Module*

Category Functional

Function Timing of USCI I2C interrupts may cause device reset due to automatic clear of an IFG.

Description When certain USCI I2C interrupt flags (IFG) are set and an automatic flag-clearing event on the I2C bus occurs, it results in an errant ISR call to the reset vector. This will only happen when the IFG is cleared within a critical time window (~6 CPU clock cycles) after a USCI interrupt request occurs and before the interrupt servicing is initiated. The affected interrupts are UCBxTXIFG, UCSTPIFG, UCSTTIFG and UCNACKIFG.

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

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

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

Workaround (1) Polling the affected flags instead of enabling the interrupts.

or

(2) Ensuring the above mentioned flag-clearing events occur after a time delay of 6 CPU clock cycles has elapsed since the interrupt request occurred and was accepted.

or

(3) At program start, check any applicable enabled IE bits such as UCBxTXIE, UCBxRXIE, UCSTTIE, UCSTPIE or UCNACKIE for a reset (A PUC will clear all of the IE bits of interest). If no PUC occurred then the device ran into the above mentioned errant condition and the program counter will need to be restored using an RETI instruction.

; ------- Workaround (3) example for TXIFG ------------

Note: For assembly code use code snippet shown below and insert prior to user code main

bit.b #UCBxTXIE ,&IE2 ; if TXIE is set, errant call occurred

jz start_normal ; if not start main program

reti ; else return from interrupt call

start_normal

... ; Application code continues

Note: For C code the workaround will need to be executed prior to the CSTARTUP routine. The steps for modifying the CSTARTUP routine are IDE dependent.

Examples for Code Composer and IAR Embedded Workbench are shown below.

IAR Embedded Workbench:

1) The file cstartup.s43 is found at: ...\IAR Systems\<Current Embedded Workbench Version>\430\src\lib\430

2) Create a local copy of this file and link it to the project. Do not rename the file.

3) In the copy insert the following code prior to stack pointer initialization as shown:

#define IE2 (0x0001)

BIT.B #0x08,&IE2 ; if TXIE is set, errant call occurred

JZ Start_Normal ; if not start main program

RETI ; else return from interrupt call

// Initialize SP to point to the top of the stack.

Start_Normal

MOV #SFE(CSTACK), SP

// Ensure that main is called.

Code Composer:

1) The file boot.c is found at ...\Texas Instruments\<Current Code Composer Version>\tools\compiler\MSP430\lib\rtssrc.zip

2) Extract the file from rtssrc.zip and create a local copy. Link the copy to the project. Do not rename this file.

3) In the copy insert the following code prior to stack pointer initialization as shown:

__asm("\t BIT.B\t #0x08,&0x0001"); // if TXIE is set, errant call occurred

__asm("\t JZ\t Start_Normal"); // if not start main program

__asm("\t RETI"); // else return from interrupt call

__asm("Start_Normal");

/*-- */

/* Initialize stack pointer. Stack grows toward lower memory. */

/*--*/

Insert the code here:

/***/

/* C_INT00() - C ENVIRONMENT ENTRY POINT */

/***/

```
#pragma CLINK(_c_int00)
extern void __interrupt _c_int00()
{
// <-- INSERT USCI28 WORKAROUND HERE
STACK_INIT();
```

USCI30 *USCI Module*

Category Functional

Function I2C mode master receiver / slave receiver

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

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

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

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

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

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

OR

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

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

Code flow for workaround

(1) Enter RX ISR for reading receiving bytes

(2) Check if UCSCLLOW.UCBxSTAT == 1

(3) If no, repeat step 2 until set

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

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

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).

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;
```

XOSC5 — *XOSC Module*

Category Functional

Function LF crystal failures may not be properly detected by the oscillator fault circuitry

Description The oscillator fault error detection of the LFXT1 oscillator in low frequency mode (XTS = 0) may not work reliably causing a failing crystal to go undetected by the CPU, i.e. OFIFG will not be set.

Workaround None

XOSC6 — *XOSC Module*

Category Functional

Function XT2 crystal failures may not be properly detected by the oscillator fault circuitry

Description The XT2OF flag should be set if the XT2 frequency falls below 30kHz. If there is no oscillation at all, the flag will still operate properly. However, 0-30kHz produces an undefined state on XT2OF. When this occurs, OFIFG will not be set.

Workaround Do not depend on the fault detection circuitry to accurately detect all failures.

XOSC8 *XOSC Module*

Category Functional

Function ACLK failure when crystal ESR is below 40 kOhm.

Description When ACLK is sourced by a low frequency crystal with an ESR below 40 kOhm, the duty cycle of ACLK may fall below the specification; the OFIFG may become set or in some instances, ACLK may stop completely.

Workaround Please refer to "XOSC8 Guidance" found at SLAA423 for information regarding working with this erratum.

7 Document Revision History

Changes from family erratasheet to device specific erratasheet.

1. Errata CPU8 was removed
2. RGC64 package markings have been updated

Changes from device specific erratasheet to document Revision A.

1. Errata BCL15 was added to the errata documentation.

Changes from document Revision A to Revision B.

1. BCL12 Workaround was updated.

Changes from document Revision B to Revision C.

1. Errata TA21 was added to the errata documentation.

Changes from document Revision C to Revision D.

1. Errata TB24 was added to the errata documentation.

Changes from document Revision D to Revision E.

1. Errata USCI35 was added to the errata documentation.

Changes from document Revision E to Revision F.

1. Errata JTAG23 was added to the errata documentation.

Changes from document Revision F to Revision G.

1. Package Markings section was updated.

Changes from document Revision G to Revision H.

1. Errata USCI40 was added to the errata documentation.

Changes from document Revision H to Revision I.

1. TA21 Description was updated.

Changes from document Revision I to Revision J.

1. USCI28 Workaround was updated.

Changes from document Revision J to Revision K.

1. Workaround for CPU19 was updated.

Changes from document Revision K to Revision L.

1. Function for BCL15 was updated.
2. Description for BCL15 was updated.
3. Workaround for BCL15 was updated.

Changes from document Revision L to Revision M.

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

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265