MSP430F4784 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 J	Rev I	Rev H
COMP3			✓
CPU44	✓	✓	✓
FLASH19	✓	✓	✓
FLASH24	✓	✓	✓
FLASH25			✓
FLASH27	✓	✓	✓
FLASH36	✓	✓	✓
FLL4	✓	✓	✓
FLL5	✓	✓	✓
FLL6	✓	✓	✓
FLL7	✓	✓	✓
LCDA5	✓	✓	✓
LCDA7	✓	✓	✓
SDA4	✓	✓	✓
TA12	✓	✓	✓
TA16	✓	✓	✓
TA21	✓	✓	✓
TAB22	✓	✓	✓
TB2	✓	✓	✓
TB16	✓	✓	✓
TB24	✓	✓	✓
USCI15	✓	✓	✓
USCI19	✓	✓	✓
USCI20	✓	✓	✓
USCI21	✓	✓	✓
USCI22	✓	✓	✓
USCI23	✓	✓	✓
USCI24	✓	✓	✓
USCI25	✓	✓	✓
USCI26	✓	✓	✓
USCI28	✓	✓	✓
USCI30	✓	✓	✓
USCI34	✓	✓	✓
USCI35	✓	✓	✓
USCI40	✓	✓	✓

Errata Number	Rev J	Rev I	Rev H
XOSC5	✓	✓	✓
XOSC8		✓	✓
XOSC9	✓	✓	✓

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 J	Rev I	Rev H
EEM20	✓	✓	✓
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 J	Rev I	Rev H
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

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.

6 Detailed Bug Description

COMP3 *COMP_A Module*

Category Functional

Function Control bits for comparator analog inputs not functional.

Description Bits P2CA0 and P2CA1 in CACTL2 used to connect the CA0 and CA1 device pins to the comparator module, respectively, have no effect when set to "1". The external connections to the comparator inputs are controlled instead by the CAPDx bits for the respective CA0 and CA1 port pins. (By definition, the CAPDx bits are used to disable the GPIO logic for each associated port pin.)

Workaround The CA0 and CA1 external inputs can be connected to the comparator internally by setting the bits CAPD6=1 and CAPD7=1 respectively. To disconnect an external analog signal from the CA0 or CA1 comparator inputs, CAPD6 or CAPD7 are cleared. Additionally in this case, the GPIO logic is no longer disabled and an external analog signal at the port input may cause increased current consumption through the digital input structure on the order of tens of microamps; the input leakage current is not affected.

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

IDE/Compiler	Version Number	Notes
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

CPU44 *CPU Module*

Category Functional

Function Incorrect address fetching during interrupt decoding

Description The CPU uses the default reset address if an interrupt is fired during the same time window as the module interrupt is being disabled. The failure only occurs at high temperature and/or when the frequency is near the maximum allowable range for the current VCC.

Workaround 1) Keep the system frequency lower from the maximum allowable value for the system VCC.

OR

2) Use the DINT before clearing the module interrupt enable (IE) bit.

Example for USCI_A0:

```
__disable_interrupt();    // Workaround
IE2 &= ~(UCA0TXIE);
__enable_interrupt();
```

EEM20 *EEM Module*

Category Debug

Function Debugger might clear interrupt flags

Description During debugging read-sensitive interrupt flags might be cleared as soon as the debugger stops. This is valid in both single-stepping and free run modes.

Workaround None.

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.

FLL4 *FLL+ Module*

Category Functional

Function Unexpected behavior of bit XT20FF

Description If MCLK and/or SMCLK are configured to operate with the external crystal XT2 (SELMx=10b, and/or SELS=1), modification of the bit XT2OFF from 0 to 1 in register FLL_CTL1 thereafter should not turn off XT2. This feature is not functional and the following two conditions occur

1) XT2 is switched OFF, MCLK is set to DCO frequency and SMCLK is turned OFF

And,

2) OFIFG bit in IFG1 register is set to 1

Workaround Oscillator Fault interrupt (OFIE bit in IE1 register) can be enabled. Follow the User's Guide instructions to use the NMI Interrupt Handler to check for OFIFG=1. Additionally check for XT2OFF=1 and clear this bit to re-enable XT2.

FLL5 *FLL+ Module*

Category Functional

Function Incorrect SMCLK request in low power modes

Description If SMCLK is configured to be sourced by XT2 and if bit SCG1=1 in SR (to enter specific low-power modes), SMCLK if not being used by any peripheral, should turn OFF when SMCLKOFF=1 (FLL_CTL1). However, SMCLK continues to remain active with no effect of SMCLKOFF=1.

Workaround To turn off SMCLK sourced by XT2 for specific low-power modes follow the sequence below:

1) Set SMCLKOFF=1

2) Configure SMCLK to be sourced by DCO (SELS=0)

3) Enter desired low-power mode

FLL6 *FLL+ Module*

Category Functional

Function LFXT1DIG bit is read incorrectly

Description The LFXT1DIG bit always reads as '0' even when the bit is programmed to '1'. This affects only the readout of the bit and not the clock bypass implementation which functions as expected.

Workaround None

FLL7 *FLL+ Module*

Category Functional

Function Flash operations using MCLK as the source may be affected

Description Flash write and erase operations that take place with the flash controller using MCLK as the source are not guaranteed to complete as expected under the following conditions:

1) When MCLK is sourced by an external clock source such as LFXT1 or XT2 and a clock fail condition occurs.

2) When MCLK is sourced from the DCOCLK by setting SELMx = 01 in FLL_CTL1 register.

Workaround 1)If MCLK is used as the clock source for the flash controller (FSSEL_01.FCTL2), always source MCLK from DCO by setting SELMx = 00 in the FLL_CTL1 register.

OR

2)Use ACLK or SMCLK as the clock source for the flash controller. In this case, if the external clock source fails, the clock fail indication works as expected.

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.

LCDA5 *LCD_A Module*

Category Functional

Function Wrong cycle time for first cycle of COMx/Sx signals

Description The time of the first cycle of COMx/Sx signals after enabling the LCD_A module is only half of the selected value. All following cycles are correct

Workaround Not required, because it does not influence the LCD function.

LCDA7 *LCD_A Module*

Category Functional

Function Higher current consumption when using shared LCD ports as fast toggling outputs

Description If a shared LCD pin (segment or com line) is used as digital fast toggling output (f>10kHz) and the VLCD is >0V (BG enabled) the device current consumption increases with higher toggling frequencies.

Workaround 1. Do not use shared LCD pins as fast toggling outputs if an LCD is used.

2. Reduce the toggle frequency of the shared pin to <10kHz.

SDA4 *SD16_A Module*

Category Functional

Function Reduced SINAD performance at certain input voltage levels.

Description The performance of the SD16_A maybe degraded due to reduced SINAD when the level of the analog input is between 20mV and 120mV. This can occur on any channel, irrespective of their PGA settings.

Workaround 1. Avoid the use of any PGA settings less than 16 with the common-mode voltage of zero, which most likely accommodates input signal levels that fall under this range.

or

2. Introduce a common-mode voltage, such as internal reference voltage of 1.2 V, to ensure that the input signal level is outside the range of 20 mV to 120 mV.

The following table shows the SD16_A performance with common mode voltage of zero:

SD16_A, performance (f_{SD16} = 1 MHz, SD16OSRx = 256, SD16REFON = 1)

PARAMETER		TEST CONDITIONS		V_{CC}	MIN	TYP	MAX	UNIT
SINAD	Signal-to-noise + distortion ratio	SD16GAINx = 1, Signal Amplitude V_{PP} = 500 mV	f_{IN} = 50 Hz, 100 Hz (see Note 1)	3 V	82	84		dB
		SD16GAINx = 2, Signal Amplitude V_{PP} = 250 mV			79	82		
		SD16GAINx = 4, Signal Amplitude V_{PP} = 125 mV			64	76		
		SD16GAINx = 8, Signal Amplitude V_{PP} = 62 mV			60	74		
		SD16GAINx = 16, Signal Amplitude V_{PP} = 31 mV			66	70		
		SD16GAINx = 32, Signal Amplitude V_{PP} = 15 mV			62	65		

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

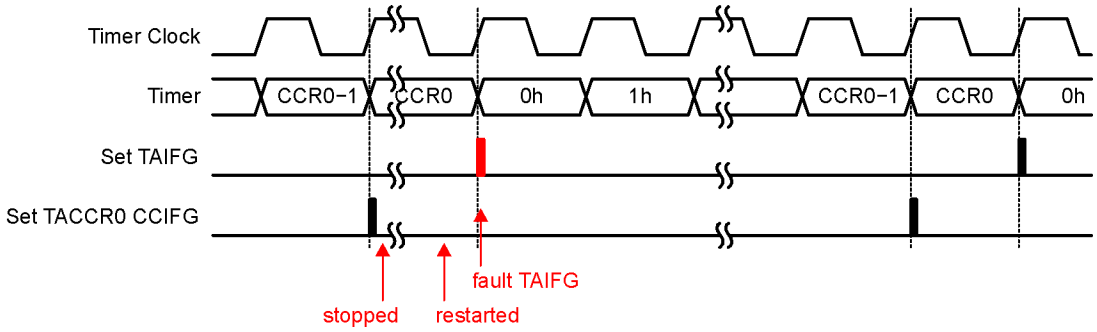
TA21 *TIMER_A Module*

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.

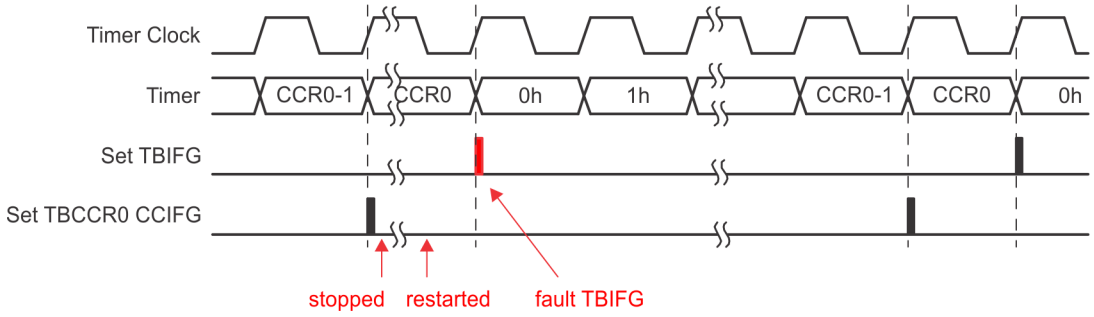
Workaround None

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 B 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.

USCI15 *USCI Module*

Category Functional

Function Receive buffer overrun undetected

Description When a new character is being loaded into RXBUF by the USCI hardware, the previous character may be overwritten. This can occur when the USCI hardware updates RXBUF with the new character and in the same instant the CPU accesses RXBUF to read the old character. In this case, the old character is lost and the new one is read out. No receive overrun error will be detected and UCOE will not be set.

Workaround Running the CPU at an adequate speed in order to guarantee access of RXBUF of received characters prior to new character receive completion will minimize the potential that simultaneous access of RXBUF may happen.

USCI19 *USCI Module*

Category Functional

Function LPM4 may affect USCI operation

Description When SMCLK is used as the USCI clock source, and SMCLK gets deactivated due to a LPM4 entry, ongoing SPI master, I2C master, and UART transmit transaction will be interrupted. Also, while in LPM4, UART receive operation is non-functional.

Workaround Do not enter LPM4 while SPI master, I2C master, or UART transmit operations are active. Wait for the operation to be completed prior entering LPM4, or enter a different low-power mode instead. Also, do not use LPM4 in case of UART receive operation. Instead, use a different low-power mode.

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

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).

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

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.

XOSC9 *XOSC Module*

Category Functional

Function XT1 Oscillator may not function as expected in HF mode

Description XT1 oscillator does not work correctly in high frequency mode at supply voltages below 2.0V with crystal frequency > 4MHz.

Workaround None. When XT1 oscillator is used in HF mode with crystal frequency > 4MHz ensure a supply voltage > 2.2V.

7 Document Revision History

Changes from family erratasheet to device specific erratasheet.

1. Errata FLASH36 was added
2. Errata XOSC9 was added
3. Errata LCDA7 was added
4. PZ100 package markings have been updated

Changes from device specific erratasheet to document Revision A.

1. Errata EEM20 was added to the errata documentation.

Changes from document Revision A to Revision B.

1. Errata TA21 was added to the errata documentation.

Changes from document Revision B to Revision C.

1. Errata TB24 was added to the errata documentation.

Changes from document Revision C to Revision D.

1. Errata USCI35 was added to the errata documentation.

Changes from document Revision D to Revision E.

1. Errata JTAG23 was added to the errata documentation.

Changes from document Revision E to Revision F.

1. Package Markings section was updated.

Changes from document Revision F to Revision G.

1. Errata CPU44 was added to the errata documentation.

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

Changes from document Revision L to Revision M.

1. USCI34 was added to the errata documentation.

Changes from document Revision M to Revision N.

1. Description for TB24 was updated.

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