Errata

SLAZ150G – October 2012 – Revised May 2018

MSP430F169 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
ADC18	✓	✓	✓	✓
ADC25	✓	✓	✓	✓
BCL5	✓	✓	✓	✓
DAC4			✓	✓
I2C7	✓	✓	✓	✓
I2C8	✓	✓	✓	✓
I2C9	✓	✓	✓	✓
I2C10	✓	✓	✓	✓
I2C11	✓	✓	✓	✓
I2C12	✓	✓	✓	✓
I2C13	✓	✓	✓	✓
I2C14	✓	✓	✓	✓
I2C15	✓	✓	✓	✓
I2C16	✓	✓	✓	✓
MPY2	✓	✓	✓	✓
TA12	✓	✓	✓	✓
TA16	✓	✓	✓	✓
TA21	✓	✓	✓	✓
TAB22	✓	✓	✓	✓
TB2	✓	✓	✓	✓
TB16	✓	✓	✓	✓
TB24	✓	✓	✓	✓
US14			✓	✓
US15	✓	✓	✓	✓
WDG2	✓	✓	✓	✓
XOSC4				✓

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.

The device doesn't have Debug errata.

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
CPU4	✓	✓	✓	✓

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

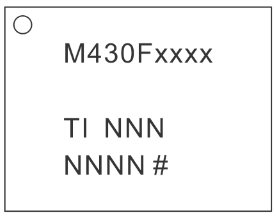
PM64

LQFP (PM), 64 Pin

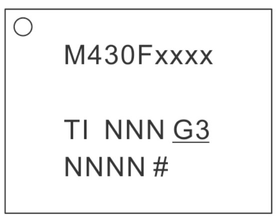
= Die revision
○ = Pin 1 location
N = Lot trace code

RTD64

QFN (RTD), 64 Pin



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



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

6 Detailed Bug Description

ADC18 — *ADC12 Module*

Category Functional

Function Incorrect conversion result in extended sample mode

Description The ADC12 conversion result can be incorrect if the extended sample mode is selected (SHP = 0), the conversion clock is not the internal ADC12 oscillator (ADC12SSEL > 0), and one of the following two conditions is true:

- The extended sample input signal SHI is asynchronous to the clock source used for ADC12CLK and the undivided ADC12 input clock frequency exceeds 3.15 MHz.

or

- The extended sample input signal SHI is synchronous to the clock source used for ADC12CLK and the undivided ADC12 input clock frequency exceeds 6.3 MHz.

Workaround - Use the pulse sample mode (SHP = 1).

or

- Use the ADC12 internal oscillator as the ADC12 clock source.

or

- Limit the undivided ADC12 input clock frequency to 3.15 MHz.

or

- Use the same clock source (such as ACLK or SMCLK) to derive both SHI and ADC12CLK, to achieve synchronous operation, and also limit the undivided ADC12 input clock frequency to 6.3 MHz.

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.

BCL5 — *BCS Module*

Category Functional

Function RSELx bit modifications can generate high frequency spikes on MCLK

Description When DIVMx = 00 or 01 the RSELx bits of the Basic Clock Module are incremented or decremented in steps of 2 or greater, the DCO output may momentarily generate high frequency spikes on MCLK, which may corrupt CPU operation. This is not an issue when

DIVMx = 10 or 11.

Workaround Set DIVMx = 10 or 11 to divide the MCLK input prior to modifying RSELx. After the RSELx bits are configured as desired, the DIVMx setting can be changed back to the original selection.

CPU4 *CPU Module*

Category Compiler-Fixed

Function PUSH #4, PUSH #8CPU4 - Bug

Description The single operand instruction PUSH cannot use the internal constants (CG) 4 and 8. The other internal constants (0, 1, 2, -1) can be used. The number of clock cycles is different:

PUSH #CG uses address mode 00, requiring 3 cycles, 1 word instruction

PUSH #4/#8 uses address mode 11, requiring 5 cycles, 2 word instruction

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

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v2.x until v6.20	User is required to add the compiler flag option below. --hw_workaround=CPU4
IAR Embedded Workbench	IAR EW430 v6.20 or later	Workaround is automatically enabled
TI MSP430 Compiler Tools (Code Composer Studio)	v1.1 or later	
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 167 or later	

DAC4 *DAC12 Module*

Category Functional

Function DAC1 overwrites an input of the SVS comparator

Description DAC1, when enabled (DAC12_1CTL.DAC12AMPx >0), overrides the input of the SVS comparator if SVSCTL.VLDx = 1111 (comparing external input voltage SVSIN to 1.25 V.) This is caused by a conflict between SVS and DAC1 at Port 6.7. This behavior only affects DAC output pins shared with SVSIN function.

Workaround 1) Do not enable DAC1 when SVS is used with VLDx = 1111

OR

2) Use DAC output pin not shared with SVSIN function

I2C7 *USART Module*

Category Functional

Function ARDYIFG Interrupt flag generation can fail in I2C slave mode.

Description When the USART is configured for I2C mode (U0CTL.I2C, SYNC, and I2CEN are set) and the module is configured as an I2C slave (U0CTL.MST=0), the ARDYIFG interrupt flag generation can fail, even when both the I2C stop condition is received and the receive buffer is empty.

This condition occurs when the I2C clock source selected by I2CSSELx is disabled by the Status Register (SR) control signals OSCOFF or SCG1.

In this configuration, the hardware clock activation is enabled by the I2C module. However, if RXRDYIFG is polled to determine data reception, the I2C hardware clock activation may be disabled before the ARDYIFG is generated.

Workaround

(1)Use interrupt service routines using the I2C interrupt vector generator feature (I2CIV) to handle all I2C interrupts.

OR

(2)After detection of I2C Own Address (OAIFG), the selected I2C clock source is enabled by clearing the OSCOFF or SCG1 Status Register (SR) bits. When the ARDYIFG is detected, the OSCOFF or SCG1 in the Status Register (SR) can be set to disable the clock source and return to the desired low power mode operation.

OR

(3)For slave only devices, it is normally not necessary to use ARDYIFG.

I2C8 *USART Module*

Category Functional

Function Master Transmitter transmits 0FFh continuously.

Description When the USART is configured for I2C mode (U0CTL.I2C, SYNC, and I2CEN are set) and the module is configured as an I2C master (U0CTL.MST=1) and I2CNDAT is used to control the number of bytes to transmit, the possibility exists that the master state-machine can become corrupted and start sending 0FFh as data on the I2C bus. Specifically, this error can occur when a long delay occurs between the set of the I2CTXRDY interrupt flag and the loading of I2CDRB (I2CDRW).

Workaround After detection of the I2CTXRDY interrupt flag, verify that the I2CTXUDF bit in I2CDCTL is set before loading I2CDRB (I2CDRW).

I2C9 *USART Module*

Category Functional

Function Master Transmitter Repeat Mode I2CSTP setting error.

Description When the USART is configured for I2C mode (U0CTL.I2C, SYNC, and I2CEN are set) and the module is configured as an I2C master (U0CTL.MST=1) and repeat mode operation is selected (I2CTCTL.I2CRM=1), the timing of the I2CSTP bit can result in lost data or extra requested transmitted bytes.

Specifically, if interrupts are active during the following two cases:

1) During the time between the setting of the I2CSTP bit and loading of I2CDRB (I2CDRW).

2) For transmitting slave address only, during the time between checking for I2CSTT cleared and setting I2CSTP.

Note: In the second case, the SCL line will be held low until the I2CDRB (I2CDRW) is loaded and then shifted out.

Workaround Solution for case #1: disable all interrupts (DINT) before setting I2CSTP then re-enabling after loading of I2CDRB.

Solution for case #2: disable all interrupts (DINT) before setting I2CSTT bit then re-enabling after setting I2CSTP bit.

I2C10 *USART Module*

Category Functional

Function Master stop bit SCL low phase does not match I2CSCLL setting.

Description When the USART is configured for I2C mode (U0CTL.I2C, SYNC, and I2CEN are set) and the module is configured as an I2C master (U0CTL.MST=1), the hardware control of the SCL low phase before stop generation is equal to a single I2CCLK period. This is particularly noticeable with large I2CSCLL settings or large I2CPSC settings.

Workaround None.

I2C11 *USART Module*

Category Functional

Function Master state machine requires reset before new sequence can proceed.

Description When the USART is configured for I2C mode (U0CTL.I2C, SYNC, and I2CEN are set) and the module is configured as an I2C master (U0CTL.MST=1), the master state-machine does not properly reset between execution cycles.

Workaround Before starting the new master sequence, clear and then re-set the I2CEN bit in the U0CTL register.

bic.b #I2CEN,&U0CTL

bis.b #I2CEN,&U0CTL

I2C12 *USART Module*

Category Functional

Function Master/Slave looses data on reception (lost RXRDYIFG).

Description If the I2C data register I2CDRB (I2CDRW) is read at the same time that data is loaded from the internal I2C shift register into I2CDRB (I2CDRW), then the received data is lost and no corresponding receive ready interrupt (RXRDYIFG) is generated. Following RXRDYIFG interrupts are processed but the missing byte cannot be recovered.

Workaround Do not read the I2CDRB(I2CDRW) register while data is being loaded into it. This can be ensured by reading this register in a timely manner using any one of the following methods:

1) Handle RXRDYIFG events with all other interrupt sources disabled.

2) Use the DMA for receiving incoming I2C data. The DMA interrupt or ARDYIFG interrupt can be used to initiate further processing of received data.

3) Enable nested interrupts to allow immediate processing of RXRDYIFG interrupts. (Care must be taken to avoid stack overflows).

I2C13 — *USART Module*

Category Functional

Function Glitch on SCL between I2C communication cycles can corrupt the state machine in I2C master mode.

Description When the USART is configured for I2C communication (U0CTL.I2C, SYNC, and I2CEN are set) and the module is configured as an I2C master (U0CTL.MST=1), the I2C module is automatically switched to slave mode following the I2C master's generation of a stop condition. If SCL is then pulled low and released again, the following device behavior can be observed:

1) When SCL is pulled low after the stop condition is generated and while ARDYIFG is not yet set, then ARDYIFG is not set as expected and ALIFG is set. SCL is released. See workaround 1 for details on how to handle this condition.

2) When SCL is pulled low at the same time as ARDYIFG is being set, ALIFG is set and SCL is released. Subsequent communication can result in an immediate ALIFG generation. See workaround 2 for details on how to handle this condition.

3) When SCL is pulled low after ARDYIFG is set but before ARDYIFG is cleared, ALIFG is not set, but SCL is held low by the master. An SCL hang-up condition occurs. See workaround 3 for details on how to handle this condition.

4) When SCL is pulled low after ARDYIFG is cleared, the module operates as intended. The ALIFG flag is not set and SCL is released.

Workaround 1. ALIFG must be processed. Data bytes are not affected.

2. ALIFG must be processed. Data bytes are not affected. To avoid a second ALIFG, clear I2CEN and re-set I2CEN before new communication begins.

3. Clear I2CEN and re-set I2CEN before new communication begins to clear the SCL hang-up.

I2C14 — *USART Module*

Category Functional

Function Master SCL phases do not match I2CSCLx settings.

Description When the USART is configured for I2C mode (U0CTL.I2C, SYNC, and I2CEN are set) and the module is used as an I2C master (U0CTL.MST=1), the generated I2C shift clock (SCL) high and low phases may be one or more I2CIN clock periods longer than defined by I2CSCLH and I2CSCLL. High I2CIN frequencies, large external pull-up resistors, and a large capacitive bus loading on SCL increase the likelihood for this to occur.

Workaround If possible, use an I2CIN input frequency of 1MHz or less. Additionally, use low-impedance I2C pull-up resistors, preferably in the lower single-digit k-Ohm range, and minimize capacitive load on SCL.

I2C15 — *USART Module*

Category Functional

Function I2CBUSY flag may clear before stop condition

Description The I2CBUSY flag may already be cleared before the Stop condition on the bus is seen.

Workaround Use the I2CBB flag instead of the I2CBUSY flag.

I2C16 *USART Module*

Category Functional

Function I2C Slave may not detect own address correctly

Description When an interrupt occurs between ACK and stop conditions of a slave transmission, the slave may not acknowledge the slave address byte if all below conditions are fulfilled:

- STT interrupt is enabled

- Device is in LPMx during start condition.

If the failure occurs, the I2C state machine switches into IDLE state.

Workaround (1)Do not use the STT interrupt for slave transmission.

Or

(2)Disable all interrupts between ACK and stop condition on I2C

MPY2 *MPY Module*

Category Functional

Function Multiplier Result register corruption

Description Depending on the address of the write instruction, writing to the multiplier result registers (RESHI, RESLO, or SUMEXT) may corrupt the result registers. The address dependency varies between a 2-word and a 3-word instructions.

Workaround Ensure that a write instruction to an MPY result register (for example, mov.w #200, &RESHI) is not located at an address with the four least significant bits shown in Table 1:

Table 1. Sensitive Addresses for Write Access to MPY Result Registers MAB[3:0]

RESLOW 013Ah		RESHI 013Ch		SUMEXT 013Eh	
3 Word	2 Word	3 Word	2 Word	3 Word	2 Word
2	4	2	4	2	4
6	8	4	6	6	8
A	C	A	C	A	C
E	0	C	E	–	–

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.

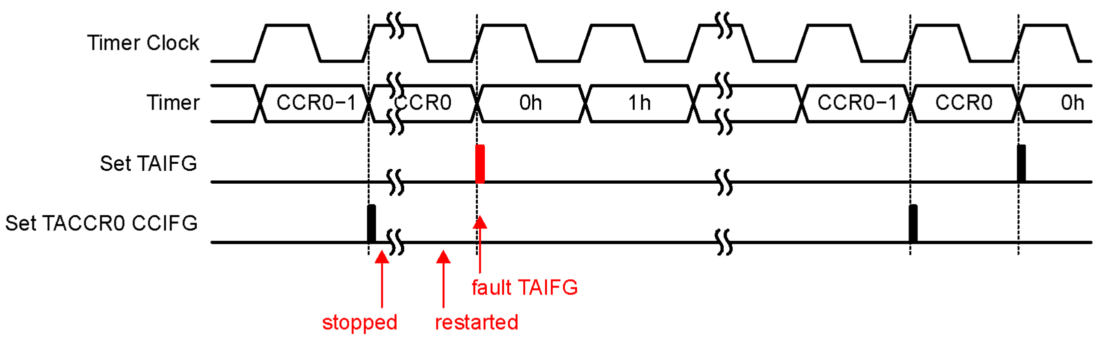
Workaround None

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.

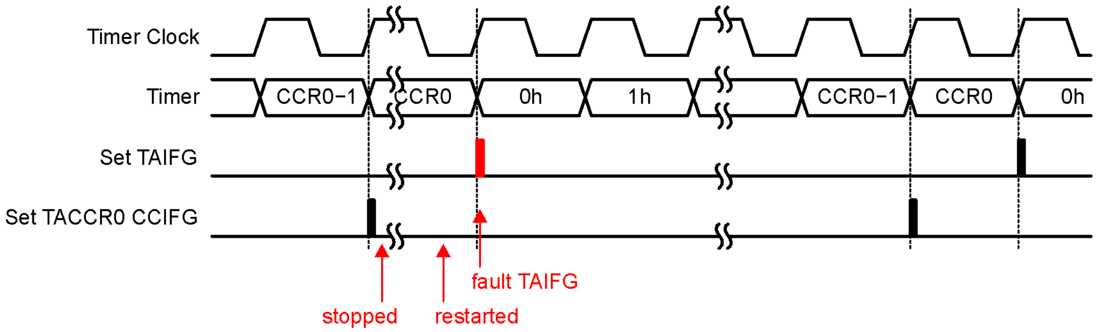
Workaround None

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.

US14 *USART Module*

Category Functional

Function Start edge of received characters may be ignored

Description When using the USART in UART mode with UxBR0 = 0x03 and UxBR1 = 0x00, the start edge of received characters may be ignored due to internal timing conflicts within the UART state machine. This condition does not apply when UxBR0 is > 0x03.

Workaround None

US15 *USART Module*

Category Functional

Function UART receive with two stop bits

Description USART hardware does not detect a missing second stop bit when SPB = 1.

The Framing Error Flag (FE) will not be set under this condition and erroneous data reception may occur.

Workaround None (Configure USART for a single stop bit, SPB = 0)

WDG2 *WDT Module*

Category Functional

Function Incorrectly accessing a flash control register

Description If a key violation is caused by incorrectly accessing a flash control register, the watchdog interrupt flag is set in addition to the expected PUC.

Workaround None

XOSC4 *XOSC Module*

Category Functional

Function	XT1 high frequency oscillator low power wake-up error
Description	The XT1 high frequency oscillator wake-up from low power mode operation is not functional.
Workaround	If using the XT1 high frequency oscillator circuitry (BCSCTL1.XTS = 1), the OSCOFF bit in the Status Register (SR) must always be 0.

7 Document Revision History

Changes from family erratasheet to device specific erratasheet.

1. Errata MPY2 was added
2. Errata I2C16 was added
3. Description for TAB22 was updated

Changes from device specific erratasheet to document Revision A.

1. Errata TA21 was added to the errata documentation.

Changes from document Revision A to Revision B.

1. Errata TB24 was added to the errata documentation.

Changes from document Revision B to Revision C.

1. Package Markings section was updated.
2. Errata SVS2 was removed from the errata documentation.
3. Errata DAC4 was added to the errata documentation.

Changes from document Revision C to Revision D.

1. DAC4 Workaround was updated.
2. DAC4 Function was updated.
3. DAC4 Description was updated.

Changes from document Revision D to Revision E.

1. TA21 Description was updated.

Changes from document Revision E to Revision F.

1. Function for CPU4 was updated.
2. Workaround for CPU4 was updated.

Changes from document Revision F to Revision G.

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