Errata

MSP430F437 Microcontroller

ABSTRACT

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

Table of Contents

1 Functional Advisories

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

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

Errata Number	Rev J	Rev I	Rev H
ADC9	✓	✓	✓
ADC10	✓	✓	✓
ADC13	✓	✓	✓
ADC18	✓	✓	✓
ADC25	✓	✓	✓
FLL3	✓	✓	✓
PORT3	✓	✓	✓
TA12	✓	✓	✓
TA16	✓	✓	✓
TA21	✓	✓	✓
TAB22	✓	✓	✓
TB2	✓	✓	✓
TB14	✓	✓	✓
TB16	✓	✓	✓
TB24	✓	✓	✓
US13	✓	✓	✓
US14	✓	✓	✓
US15	✓	✓	✓
WDG2	✓	✓	✓
XOSC5	✓		
XOSC9	✓	✓	✓

2 Preprogrammed Software Advisories

Advisories that affect factory-programmed software.

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

The device does not have any errata for this category.

3 Debug Only Advisories

Advisories that affect only debug operation.

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

The device does not have any errata for this category.

4 Fixed by Compiler Advisories

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

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

Errata Number	Rev J	Rev I	Rev H
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 Nomenclature, Package Symbolization, and Revision Identification

The revision of the device can be identified by the revision letter on the Package Markings or by the HW_ID located inside the TLV structure of the device.

5.1 Device Nomenclature

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

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

MSP – Fully qualified production device

Support tool naming prefixes:

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

null: Fully-qualified development-support product.

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

"Developmental product is intended for internal evaluation purposes."

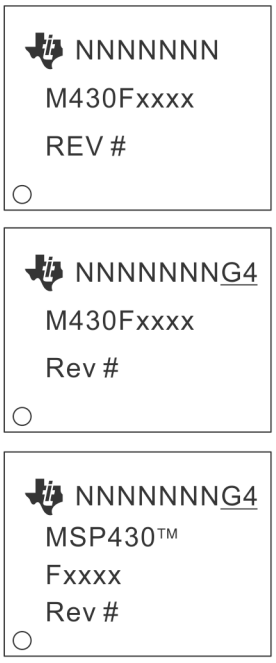
MSP devices have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

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

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

5.2 Package Markings

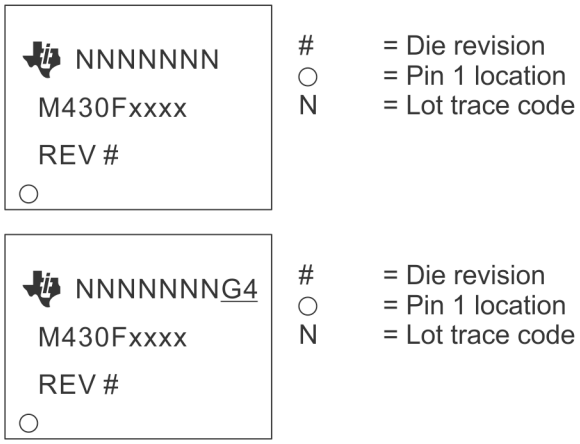
PZ100 ***LQFP (PZ) 100 Pin***



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

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

PN80 ***LQFP (PN), 80 Pin***



5.3 Memory-Mapped Hardware Revision (TLV Structure)

This device does not support reading the hardware revision from memory.

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

6 Advisory Descriptions

ADC9	***ADC Module***
Category	Functional
Function	Interrupt vector register
Description	If the ADC12 uses a different clock than the CPU (MCLK) and more than one ADC interrupt is enabled, the ADC12IV register content may be unpredictable for one clock cycle. This happens if, during the execution of an ADC interrupt, another ADC interrupt with higher priority occurs.
Workaround	- Read out ADC12IV twice and use only when values are equal. or - Use ADC12IFG to determine which interrupt has occurred.

ADC10	***ADC Module***
Category	Functional
Function	Unintended start of conversion
Description	Accessing ADC12OVIE or ADC12TOVIE at the end of an ADC12 conversion with BIS/BIC commands can cause the ADC12SC bit to be set again immediately after it was cleared. This might start another conversion, if ADC12SC is configured to trigger the ADC (SHS = 0).
Workaround	If ADC12SC is configured to trigger the ADC, the control bits ADC12OVIE and ADC12TOVIE should be modified only when the ADC is not busy (ADC12BUSY = 0).

ADC13	***ADC Module***
Category	Functional
Function	Current consumption after clearing ADC12ON while ADC is busy
Description	If the ADC12ON bit is cleared while the ADC is busy, the ADC core might not be completely turned off and still consume current.
Workaround	- Wait until ADC12BUSY is reset before clearing the ADC12ON bit. This is recommended for all protected bits in the ADC12CTLx registers. or - Clear CONSEQx bits. With CONSEQx=0 and ENC=0 the ADC12 is reset.

ADC18	***ADC 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

ADC 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.

CPU4

CPU Module

Category

Compiler-Fixed

Function

PUSH #4, PUSH #8

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	

FLL3	*FLL Module*
Category	Functional
Function	FLLDx = 11 for /8 may generate an unstable MCLK frequency
Description	When setting the FLL to higher frequencies using FLLDx = 11 (/8) the output frequency of the FLL may have a larger frequency variation (e.g. averaged over 2sec) as well as a lower average output frequency than expected when compared to the other FLLDx bit settings.
Workaround	None

PORT3	*PORT Module*
Category	Functional
Function	Port interrupts can get lost
Description	Port interrupts can get lost if they occur during CPU access of the P1IFG and P2IFG registers.
Workaround	None

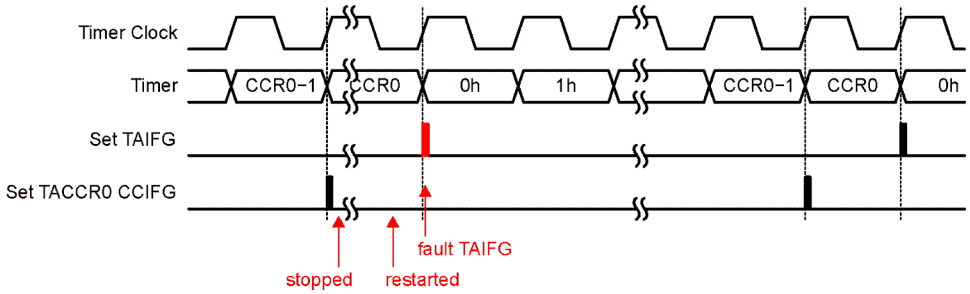
TA12	*TA Module*
Category	Functional
Function	Interrupt is lost (slow ACLK)
Description	Timer_A counter is running with slow clock (external TACLK or ACLK)compared to MCLK. The compare mode is selected for the capture/compare channel and the CCRx register is incremented by one with the occurring compare 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	*TA 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	*TA 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 *TAB 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 *TB 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.

TB14 *TB Module*

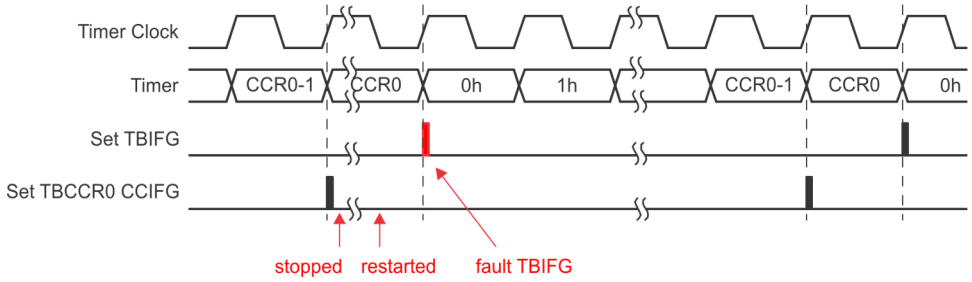
Category	Functional
Function	PWM output
Description	The PWM output unit may behave erroneously if the condition for changing the PWM output (EQUx or EQU0) and the condition for loading the shadow register TBCLx happen at the same time. Depending on the load condition for the shadow registers (CLLD bits in TBCCTLx), there are four possible error conditions: 1. Change CCRx register from any value to CCRx = 0 (for example, sequence for CCRx = 4 3 2 0 0 0) 2. Change CCRx register from CCRx = 0 to any value (for example, sequence for CCRx = 0 0 0 2 3 4) 3. Change CCRx register from any value to current SHD0 (CCR0) value (for example, sequence for CCRx = 4 2 5 SHD0 3 8) 4. Change CCRx register from current SHD0 (CCR0) value to any value (for example, sequence for CCRx = 4 2 SHD0 5 3 8)
Workaround	No general workaround available.

TB16 *TB 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 *TB 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.

US13 *USART Module*

Category Functional

Function Unpredictable program execution

Description USART interrupts requested by URXS can result in unpredictable program execution if this request is not served within two bit times of the received data.

Workaround Ensure that the interrupt service routine is entered within two bit times of the received data.

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 *WDG 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

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

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 Revision History

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

Changes from October 9, 2019 to May 11, 2021	Page
• Changed the document format and structure; updated the numbering format for tables, figures, and cross references throughout the document	6

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), 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, regulatory 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 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.

TI objects to and rejects any additional or different terms you may have proposed.

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