



**ABSTRACT**

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

---

**Table of Contents**

<b>1 Functional Advisories</b> .....	<b>2</b>
<b>2 Preprogrammed Software Advisories</b> .....	<b>2</b>
<b>3 Debug Only Advisories</b> .....	<b>2</b>
<b>4 Fixed by Compiler Advisories</b> .....	<b>2</b>
<b>5 Nomenclature, Package Symbolization, and Revision Identification</b> .....	<b>4</b>
5.1 Device Nomenclature.....	4
5.2 Package Markings.....	4
5.3 Memory-Mapped Hardware Revision (TLV Structure).....	4
<b>6 Advisory Descriptions</b> .....	<b>5</b>
<b>7 Revision History</b> .....	<b>11</b>

## 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 E	Rev D	Rev C	Rev B
<a href="#">ADC18</a>	✓	✓	✓	✓
<a href="#">ADC25</a>	✓	✓	✓	✓
<a href="#">DAC4</a>				✓
<a href="#">FLL3</a>	✓	✓	✓	✓
<a href="#">TA12</a>	✓	✓	✓	✓
<a href="#">TA16</a>	✓	✓	✓	✓
<a href="#">TA21</a>	✓	✓	✓	✓
<a href="#">TAB22</a>	✓	✓	✓	✓
<a href="#">TB2</a>	✓	✓	✓	✓
<a href="#">TB16</a>	✓	✓	✓	✓
<a href="#">TB24</a>	✓	✓	✓	✓
<a href="#">US14</a>				✓
<a href="#">US15</a>	✓	✓	✓	✓
<a href="#">WDG2</a>	✓	✓	✓	✓
<a href="#">XOSC5</a>	✓			
<a href="#">XOSC9</a>	✓	✓	✓	✓

## 2 Preprogrammed Software Advisories

Advisories that affect factory-programmed software.

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

The device does not have any errata for this category.

## 3 Debug Only Advisories

Advisories that affect only debug operation.

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

Errata Number	Rev E	Rev D	Rev C	Rev B
<a href="#">EEM20</a>	✓	✓	✓	✓

## 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 E	Rev D	Rev C	Rev B
<a href="#">CPU4</a>	✓	✓	✓	✓

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

**PN80**

**LQFP (PN), 80 Pin**

 NNNNNNN M430Fxxxx REV # ○	# = Die revision ○ = Pin 1 location N = Lot trace code
 NNNNNNNG4 M430Fxxxx REV # ○	# = Die revision ○ = Pin 1 location N = Lot trace code

### 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

<b>ADC18</b>	<b>ADC Module</b>
<b>Category</b>	Functional
<b>Function</b>	Incorrect conversion result in extended sample mode
<b>Description</b>	<p>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 &gt; 0), and one of the following two conditions is true:</p> <ul style="list-style-type: none"> <li>- 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.</li> <li>or</li> <li>- 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.</li> </ul>
<b>Workaround</b>	<ul style="list-style-type: none"> <li>- Use the pulse sample mode (SHP = 1).</li> <li>or</li> <li>- Use the ADC12 internal oscillator as the ADC12 clock source.</li> <li>or</li> <li>- Limit the undivided ADC12 input clock frequency to 3.15 MHz.</li> <li>or</li> <li>- 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.</li> </ul>
<b>ADC25</b>	<b>ADC Module</b>
<b>Category</b>	Functional
<b>Function</b>	Write to ADC12CTL0 triggers ADC12 when CONSEQ = 00
<b>Description</b>	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.
<b>Workaround</b>	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.
<b>CPU4</b>	<b>CPU Module</b>
<b>Category</b>	Compiler-Fixed
<b>Function</b>	PUSH #4, PUSH #8
<b>Description</b>	<p>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:</p> <p>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</p>
<b>Workaround</b>	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*****DAC 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

**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.

**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

**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 ( $CCR_x = CCR_x + 1$ ) 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  $TAR = CCR_x + 1$ ). 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  $ID_x > 00$

**Description** The first increment of TAR after any timer clear event (POR/TACLK) 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  $ID_x$  settings.

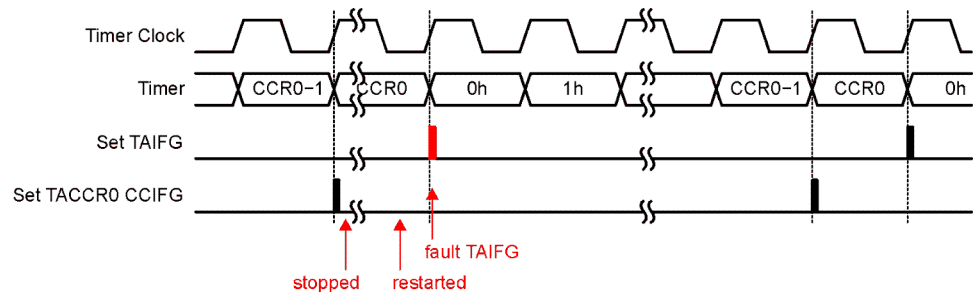
**Workaround** None

**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 TACLK 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/TBCCR<sub>x</sub> 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 CCR<sub>x</sub> register is incremented by 1 with the occurring compare interrupt (if TBR = CCR<sub>x</sub>). Due to the fast MCLK, the CCR<sub>x</sub> register increment (CCR<sub>x</sub> = CCR<sub>x</sub> + 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 = CCR<sub>x</sub> + 1). This interrupt is lost.

#### **Workaround**

Switch capture/compare mode to capture mode before the CCR<sub>x</sub> register increment. Switch back to compare mode afterward.

## TB16

### ***TB Module***

---

#### **Category**

Functional

#### **Function**

First increment of TBR erroneous when ID<sub>x</sub> > 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 (ID<sub>0</sub>, ID<sub>1</sub>). All following TBR increments are performed correctly with the selected ID<sub>x</sub> settings.

#### **Workaround**

None

## TB24

### ***TB Module***

---

#### **Category**

Functional

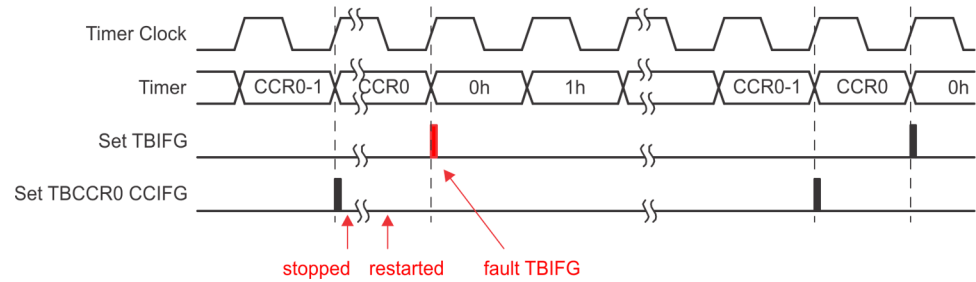
#### **Function**

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 TBCCR<sub>0</sub> to zero. However, if the Timer B is stopped at TBR = TBCCR<sub>0</sub>, 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** *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

<b>XOSC9</b>	<b><i>XOSC Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	XT1 Oscillator may not function as expected in HF mode
<b>Description</b>	XT1 oscillator does not work correctly in high frequency mode at supply voltages below 2.0V with crystal frequency > 4MHz.
<b>Workaround</b>	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.

<b>Changes from May 12, 2021 to May 17, 2021</b>	<b>Page</b>
<ul style="list-style-type: none"><li>Changed the document format and structure; updated the numbering format for tables, figures, and cross references throughout the document.....</li></ul>	<b>5</b>

## 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](http://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  
Copyright © 2022, Texas Instruments Incorporated