Errata
MSP430AFE232 Microcontroller

![Texas Instruments logo]

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 A
BCL12	✓
FLASH24	✓
FLASH27	✓
FLASH36	✓
TA12	✓
TA16	✓
TA18	✓
TA21	✓
TAB22	✓
US15	✓

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 A
EEM20	✓

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 A
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

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

PW24   *TSSOP (PW), 24 Pin*
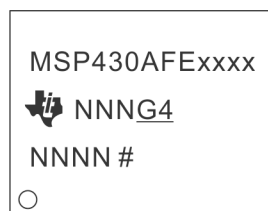
MSP430AFExxxx	#	= Die revision
⫶ NNN<u>G4</u>	○	= Pin 1 location
NNNN #	N	= Lot trace code
○		

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

BCL12	***BCL Module***

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

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

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

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

AND

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

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

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

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

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	

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.

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.

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.

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

TA18	*TA Module*
Category	Functional
Function	MOV to TACTL may clear TAR
Description	When TACTL is modified with a MOV instruction, the contents of TAR may be cleared, even when TACLR is not set.
Workaround	Use BIS or BIC instructions to modify TACTL.

> **Note**
> A DMA transfer must not occur while these BIS and BIC instructions execute. This can be prevented by disabling the DMA prior to these instructions, or by using the DMAONFETCH bit to align DMA transfers to instruction fetch boundaries.

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.
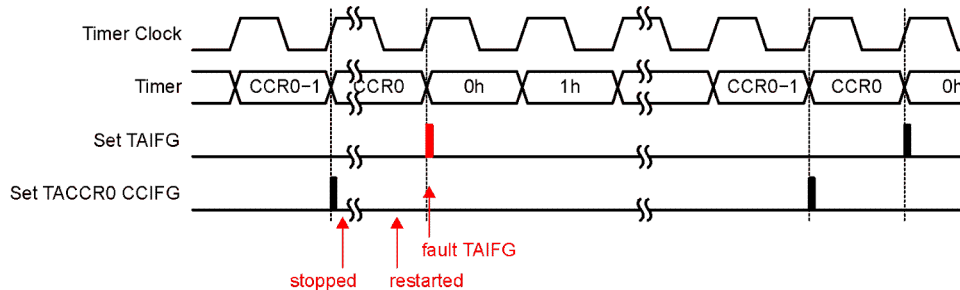
Workaround   None.

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.

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)

7 Revision History

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

IMPORTANT NOTICE AND DISCLAIMER