Errata

MSP430F2013 Microcontroller

ABSTRACT

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

Table of Contents

1 Functional Advisories.......... 2
2 Preprogrammed Software Advisories.......... 2
3 Debug Only Advisories.......... 2
4 Fixed by Compiler Advisories.......... 2
5 Nomenclature, Package Symbolization, and Revision Identification.......... 4
5.1 Device Nomenclature.......... 4
5.2 Package Markings.......... 4
5.3 Memory-Mapped Hardware Revision (TLV Structure).......... 5
6 Advisory Descriptions.......... 6
7 Revision History.......... 14

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 G	Rev F	Rev E	Rev D	Rev C	Rev B
BCL9				✓	✓	✓
BCL10				✓	✓	✓
BCL11				✓	✓	✓
BCL12	✓	✓	✓	✓	✓	✓
BCL13				✓	✓	✓
BCL14	✓	✓	✓			
FLASH16	✓	✓	✓	✓	✓	✓
FLASH22				✓	✓	✓
PORT10				✓	✓	✓
SDA2						✓
SDA3	✓	✓	✓	✓	✓	✓
SYS15	✓	✓	✓	✓	✓	✓
TA12	✓	✓	✓	✓	✓	✓
TA16	✓	✓	✓	✓	✓	✓
TA17						✓
TA21	✓	✓	✓	✓	✓	✓
TAB22	✓	✓	✓	✓	✓	✓
USI1						✓
USI2						✓
USI3						✓
USI4	✓	✓	✓	✓	✓	✓
USI5	✓	✓	✓	✓	✓	✓
XOSC5	✓	✓	✓	✓	✓	✓
XOSC8			✓	✓	✓	✓

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 G	Rev F	Rev E	Rev D	Rev C	Rev B
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 G	Rev F	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 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

PW14

TSSOP (PW), 14 Pin

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

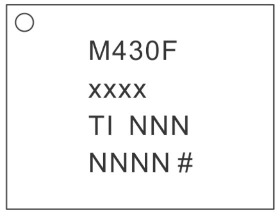
N14

PDIP (N), 14 Pin

# = Die revision
N = Lot trace code

RSA16

QFN (RSA), 16 Pin



= 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

BCL9 — *BCL Module*

Category Functional

Function ACLK divider modifications require delay before entering LPM3

Description After modifying the DIVAx bits, immediately entering LPM3 can cause the modification to be ignored and the divider settings not to take effect. Reading back the DIVAx bits will indicate the intended setting even when the divider has not been correctly applied.

Workaround When the DIVAx bits are modified, a delay of one complete ACLK (VLO or LFXT1CLK) period must elapse before entering LPM3. The delay is only necessary the first time LPM3 is entered after the DIVAx bits are modified. After the one-period delay, LPM3 may be entered and exited normally without additional delays.

BCL10 — *BCL Module*

Category Functional

Function MCLK = ACLK and P2SEL control bits

Description When using ACLK as the CPU MCLK clock source, the oscillator failsafe feature does not automatically switch MCLK to the DCO if the P2SEL6 or P2SEL7 bit is cleared. This applies when ACLK = LFXT1 (external low frequency clock source). The CPU will halt operation since no MCLK signal is present.

Workaround None

BCL11 — *BCL Module*

Category Functional

Function Watchdog failsafe when using ACLK

Description When using ACLK as the WDT+ clock source, the WDT+ oscillator failsafe feature does not automatically switch to the DCO if the P2SEL6 or P2SEL7 bit is cleared. This applies when ACLK = LFXT1 (external low frequency clock source). The WDT+ will halt operation since no clock signal is present.

Workaround None

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.

BCL13 *BCL Module*

Category Functional

Function DCO powerup halt

Description When subject to very slow Vcc rise times, the device may enter into a state where the DCO does not oscillate. No JTAG access or program execution is possible and the device will remain in a reset state until the supply voltage is disconnected.

Workaround Apply a Vcc poweron ramp >= 10V/second under all power-on/power-cycle scenarios.

BCL14 *BCL Module*

Category Functional

Function Oscillator fault forced in bypass mode when P2SEL.7 bit is not set

Description When the LFXT1 oscillator is used in bypass mode and P2SEL.7 is not set, the oscillator fault flag (OFIFG) will be forced to set and cannot be cleared. Due to the failsafe logic, LFXT1 cannot be used as MCLK in this case. The bug only affects the behavior of the oscillator fault, the clocking itself works properly.

Workaround Set both P2SEL.6 and P2SEL.7 if the application requires correct function of the oscillator fault flag (e.g. MCLK failsafe logic).

Note

Setting P2SEL.7 bit disables the GPIO functionality and enables the input schmitt trigger of the pin. P2.7 should be tied to a fixed voltage level (VCC or GND) to prevent cross current.

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.

FLASH16 — *FLASH Module*

Category Functional

Function Modifying INFOA addresses when LOCKA = 1 will modify main flash memory

Description When attempting to write to an address location or perform a segment erase of INFOA while the LOCKA bit is set, flash memory beginning at main memory location 0xFC40 and extending for 64 bytes to address 0xFC7F will be modified erroneously. These 64 bytes are addressed and modified in place of the INFOA addresses when writes or erases are performed within the INFOA address space and LOCKA = 1.

Workaround Prior to modifying (writing or erasing) any address within the INFOA Flash memory segment, properly clear the LOCKA control bit as described in the MSP430x2xx User's Guide (SLAU144) to unlock the segment. Once the modification is complete, setting the LOCKA bit is recommended.

FLASH22 — *FLASH Module*

Category Functional

Function Flash controller may prevent correct LPM entry

Description When ACLK (or SMCLK) is used as the flash controller clock source, and this clock source gets deactivated due to a low-power mode entry while a flash erase or write operating is pending, the flash controller will keep ACLK (or SMCLK) active even after the flash operation has been completed. This will result in an incorrect LPM entry and increased current consumption. Note that this issue can only occur when the Flash operation and the low-power mode entry are initiated from code located in RAM.

Workaround Do not enter low-power modes while flash erase or write operations are active. Wait for the operation to be completed before entering a low-power mode.

PORT10 — *PORT Module*

Category Functional

Function Pull-up/down resistor selection when module pin function is selected

Description When the pull-up/down resistor for a certain port pin is enabled (PxREN.y=1) and the module port pin function is selected (PxSEL.y=1), the pull-up/down resistor configuration of this pin is controlled by the respective module output signal (Module X OUT) instead of the port output register (PxOUT.y).

Workaround None. Do not set PxSEL.y and PxREN.y at the same time.

SDA2 — *SDA Module*

Category Functional

Function Internal reference generator performance is beyond the specification limits

Description The SD16_A reference generator may not meet the maximum temperature coefficient specification of 50 ppm/degC.

Workaround The SD16_A internal reference can be adjusted to operate within the specification by writing 0x61 to memory location 0xBF. This corrects the temperature coefficient of the internal reference and centers the typical voltage to 1.20 V.

SDA3 — *SDA Module*

Category Functional

Function The interrupt delay function can result in incorrect conversion data

Description The interrupt delay operation can result in incorrect conversion data when SD16INTDLYx = 01, 10 or 11.

Workaround Use SD16INTDLYx = 00 setting (interrupt generated after fourth conversion). This applies to the first conversion in Continuous mode and to each conversion in Single mode.

SYS15 — *SYS Module*

Category Functional

Function LPM3 and LPM4 currents exceed specified limits

Description LPM3 and LPM4 currents may exceed specified limits if the SMCLK source is switched from DCO to VLO or LFXT1 just before the instruction to enter LPM3 or LPM4 mode.

Workaround After clock switching, a delay of at least four new clock cycles (VLO or LFXT1) must be implemented to complete the clock synchronization before going into LPM3 or LPM4.

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

TA17 — *TA Module*

Category Functional

Function Capture Input CCIOB missing ACLK connection

Description The Timer_A Capture Input CCI0B is not internally connected to the ACLK signal.

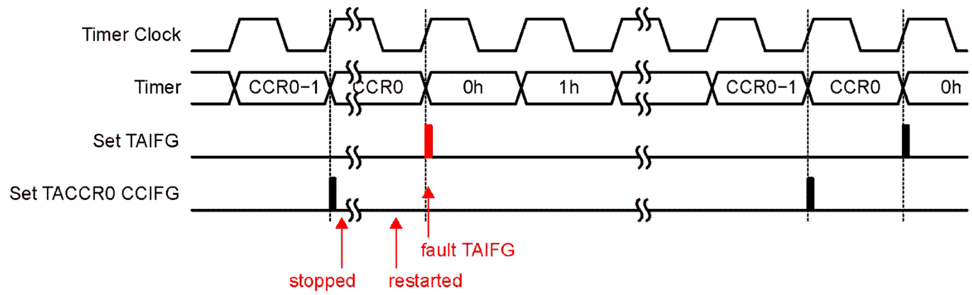
Workaround The ACLK signal can be output on P1.0 and externally input on a Timer_A capture input pin.

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.

USI1 — *USI Module*

Category Functional

Function USICKCTL cannot be written

Description When the USI is in active operation mode (that is when USICNTx < > 0), the USICKCTL cannot be written. If written, the USICNTx is cleared and the USIIFG is set. Operation using the USISWCLK is not possible.

Workaround None

USI2	***USI Module***
Category	Functional
Function	I2C slave mode erroneously pulls SCL low
Description	When the USI is configured in I2C slave mode, SCL is incorrectly pulled low when USICNTx is written with a value of 1.
Workaround	None

USI3	***USI Module***
Category	Functional
Function	I2C slave mode does not hold SCL low
Description	When the USI is configured in I2C slave mode, the module does not hold SCL low while USISTTIFG = 1 following a start condition.
Workaround	None

USI4	***USI Module***
Category	Functional
Function	I2C Slave mode can generate a glitch at SCL
Description	USI I2C Slave Operation at slower communication rates (less than 20kbps). During I2C bus active operation, if USICNT is written while SCL is high, I2C module will generate a glitch on SCL that can corrupt the I2C bus sequence.
Workaround	Verify that SCL is low before writing USICNT register. //STOP ---END--- For HP8/G2MICRO/MSP430V334 device, the erratum is listed as I2C1. I2C16: MSP430V334 I2C Slave Mode Function: MSP430V334 I2C Slave Device can generate a glitch on SCL Description: MSP430V334 I2C slave operation can generate glitches when operated at slow communication rates (less than 20 kbps) and this can corrupt the I2C bus sequence. Workaround: None

USI5	***USI Module***
Category	Functional
Function	SPI master generates one additional clock after module reset Bug
Description	Initalizing the USI in SPI MASTER mode with the USICKPH bit set generates one additional clock pulse than defined by the value in the USICNTx bits on the SCLK pin during the first data transfer after module reset. For example, if the USICNTx bits hold the value eight, nine clock pulses are generated on the SCLK pin for the first transfer only.

Workaround Load USICNTx with a count of N-1 bits (where N is the required number of bits) for the first transfer only.

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.

7 Revision History

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

Changes from May 29, 2018 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