Errata

MSP430F423A 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 B
ESP1	✓
ESP4	✓
FLL3	✓
TA12	✓
TA16	✓
TA21	✓
TAB22	✓
US15	✓
WDG2	✓

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 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 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

PM64 ***LQFP (PM), 64 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

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.

ESP1

ESP Module

Category Functional

Function Suspending the ESP430CE1

Description Suspending the ESP430 may create an invalid interrupt which can lead to a reset-like behavior of the module.

Workaround Set the bit 0x08 together with the ESPSUSP bit:
bis.w #08h+ESPSUSP, &ESPCTL
This bit also must be cleared when the suspend mode is exited.
bic.w #08h+ESPSUSP, &ESPCTL
NOTE:
- After suspending the ESP430CE1 it can take up to 9 MCLK clock cycles before the CPU can access the SD16 registers.
- An interrupt service routine for the SD16 is required.

```
// Shut down ESP (set Embedded Signal Processing into
// "Suspend" mode)
// ensure that it is not in measurement or calibration mode,
ESPCTL |= 0x08 + ESPSUSP;
// Set ESP into Suspend Mode
// incl. Bug Fix for Suspend Mode
// wait 9 clocks until proper access to the SD16 is possible
__delay_cycles(9);

MBCTL &= ~(IN0IFG + IN0IE);
// Clear any Pending MB interrupt and disable
// ESP interrupt
SD16CTL &= ~SD16REFON; // Switch Reference off
```

ESP4 *ESP Module*

Category Functional

Function Suspending the ESP430 activity

Description Due to timing violations between the ESP CPU and the MSP430 CPU, the SD16 converters are not switched off correctly if the ESP CPU is set into suspend mode immediately after the ESP CPU is checked for idle mode. This leads to an higher current consumption in low-power modes.

Workaround Implement an additional wait loop of 16 clock cycles between checking the ESP for idle mode and set the ESP CPU into suspend mode.

```
while ((RET0 & 0x8000) != 0); // Wait for Idle mode
// wait 16 clocks to exclude timing violations between MSP430 CPU
// and ESP CPU
_NOP();_NOP();_NOP();_NOP();_NOP();_NOP();_NOP();_NOP();_NOP();--
_NOP();_NOP();_NOP();_NOP();_NOP();_NOP();_NOP();
// Shut down ESP (set Embedded Signal Processing into "Suspend" mode)
// ensure that it is not in measurement or calibration mode,
if ((RET0 & 0x8000) == 0)
{
ESPCTL |= 0x08 + ESPSUSP; // Set ESP into Suspend Mode
// incl. Bug Fix for Suspend Mode
}
```

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 (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.

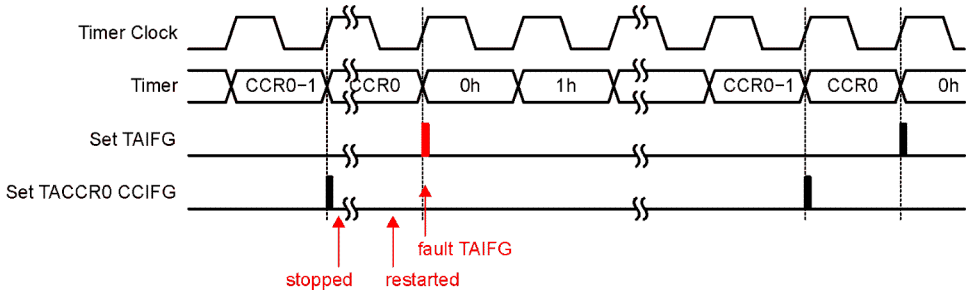
Workaround None

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.

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

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	4

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