Errata

MSP430FR2000 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
CPU46	✓
CS13	✓
CS14	✓
PMM32	✓
RTC15	✓
TB25	✓
USCI42	✓
USCI47	✓
USCI50	✓

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
EEM23	✓

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
CPU21	✓
CPU22	✓
CPU40	✓

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."

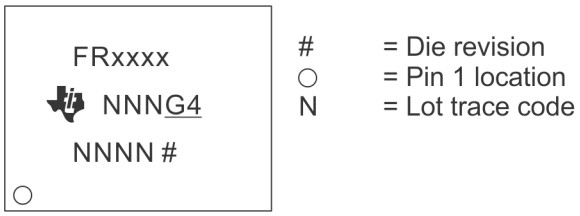
MSP devices have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

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.

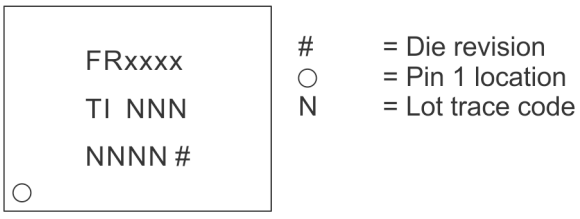
TI device nomenclature also includes a suffix with the device family name. This suffix indicates the temperature range, package type, and distribution format.

5.2 Package Markings

PW16 ***TSSOP (PW), 16 Pin***



RLL24 ***QFN (RLL), 24 pins***



5.3 Memory-Mapped Hardware Revision (TLV Structure)

Die Revision	TLV Hardware Revision
Rev B	11h

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

CPU21 *CPU Module*

Category Compiler-Fixed

Function Using POPM instruction on Status register may result in device hang up

Description When an active interrupt service request is pending and the POPM instruction is used to set the Status Register (SR) and initiate entry into a low power mode , the device may hang up.

Workaround None. It is recommended not to use POPM instruction on the Status Register.

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	Not affected	
TI MSP430 Compiler Tools (Code Composer Studio)	v4.0.x or later	User is required to add the compiler or assembler flag option below. --silicon_errata=CPU21
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 167 or later	

CPU22 *CPU Module*

Category Compiler-Fixed

Function Indirect addressing mode with the Program Counter as the source register may produce unexpected results

Description When using the indirect addressing mode in an instruction with the Program Counter (PC) as the source operand, the instruction that follows immediately does not get executed. For example in the code below, the ADD instruction does not get executed.

```
mov @PC, R7
add #1h, R4
```

Workaround Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	Not affected	
TI MSP430 Compiler Tools (Code Composer Studio)	v4.0.x or later	User is required to add the compiler or assembler flag option below. --silicon_errata=CPU22
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 167 or later	

CPU40 *CPU Module*

Category Compiler-Fixed

CPU40 *CPU Module*

Function

PC is corrupted when executing jump/conditional jump instruction that is followed by instruction with PC as destination register or a data section

Description

If the value at the memory location immediately following a jump/conditional jump instruction is 0X40h or 0X50h (where X = don't care), which could either be an instruction opcode (for instructions like RRCM, RRAM, RLAM, RRUM) with PC as destination register or a data section (const data in flash memory or data variable in RAM), then the PC value is auto-incremented by 2 after the jump instruction is executed; therefore, branching to a wrong address location in code and leading to wrong program execution.

For example, a conditional jump instruction followed by data section (0140h).

@0x8012 Loop DEC.W R6
@0x8014 DEC.W R7
@0x8016 JNZ Loop
@0x8018 Value1 DW 0140h

Workaround

In assembly, insert a NOP between the jump/conditional jump instruction and program code with instruction that contains PC as destination register or the data section.

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v5.51 or later	For the command line version add the following information Compiler: --hw_workaround=CPU40 Assembler:-v1
TI MSP430 Compiler Tools (Code Composer Studio)	v4.0.x or later	User is required to add the compiler or assembler flag option below. --silicon_errata=CPU40
MSP430 GNU Compiler (MSP430-GCC)	Not affected	

CPU46 *CPU Module*

Category

Functional

Function

POPM peforms unexpected memory access and can cause VMAIFG to be set

Description

When the POPM assembly instruction is executed, the last Stack Pointer increment is followed by an unintended read access to the memory. If this read access is performed on vacant memory, the VMAIFG will be set and can trigger the corresponding interrupt (SFRIE1.VMAIE) if it is enabled. This issue occurs if the POPM assembly instruction is performed up to the top of the STACK.

Workaround

If the user is utilizing C, they will not be impacted by this issue. All TI/IAR/GCC pre-built libraries are not impacted by this bug. To ensure that POPM is never executed up to the memory border of the STACK when using assembly it is recommended to either

1. Initialize the SP to
a. TOP of STACK - 4 bytes if POPM.A is used
b. TOP of STACK - 2 bytes if POPM.W is used

CPU46 *CPU Module*

OR

2. Use the POPM instruction for all but the last restore operation. For the the last restore operation use the POP assembly instruction instead.

For instance, instead of using:

```
POPM.W #5,R13
```

Use:

```
POPM.W #4,R12
POP.W R13
```

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	Not affected	C code is not impacted by this bug. User using POPM instruction in assembler is required to implement the above workaround manually.
TI MSP430 Compiler Tools (Code Composer Studio)	Not affected	C code is not impacted by this bug. User using POPM instruction in assembler is required to implement the above workaround manually.
MSP430 GNU Compiler (MSP430-GCC)	Not affected	C code is not impacted by this bug. User using POPM instruction in assembler is required to implement the above workaround manually.

CS13 *CS Module*

Category Functional

Function Device may enter lockup state during transition from AM to LPM3/4 if DCO frequency is above 2 MHz.

Description The device might enter lockup state if DCO frequency is above 2 MHz and two events happen at the same time:

1) The device transitions from AM to LPM3/4 (e.g. during ISR exits or Status Register modifications)

2) An interrupt is requested (e.g. GPIO interrupt).

This condition can be recovered by BOR/Power cycle.

CS13 (continued)	***CS Module***
Workaround	1. Use DCOCLK at 2MHz or lower. OR 2. Use LPM0/x.5 instead of LPM3/4. OR 3. Use external high-frequency crystal if it is available on the device. OR 4. Set DCOCLK to 2MHz or lower before entering LPM3/4, then restore DCOCLK after wake-up. Note using peripherals using clocks derived from DCOCLK might be affected during this interval.

CS14	***CS Module***
Category	Functional
Function	TLV values for 16MHz calibration are shifted beyond datasheet error limits
Description	During setting DCO frequency to 16MHz, 16MHz DCO tap settings value inside of TLV is used as the CSCTL0 setting. In this case after FLL locked, DCO 16MHz frequency can shift outside of datasheet specification of +/- 1% at 25C for a maximum time of 60ms. After the maximum time of 60ms, DCO frequency error will settle back to DCO FLL lock frequency of 16 MHz +/- 1% at 25C specified in device data sheet
Workaround	1. After setting the DCO frequency to 16MHz, set a 300us delay in software, then poll the FLL locked status bits in CSCTL7 Register. Once the FLL is locked, the DCO frequency will match the device data sheet specifications. OR 2. During first power on of MCU: - set DCO frequency to 16MHz, - set a 300us delay in software, - poll FLL locked status bits in CSCTL7 until FLL lock - store DCO tap settings by reading CSCTL0 register value and write to FRAM The stored DCO tap settings can then be loaded into CSCTL0 when setting the DCO to 16MHz in order to reduce FLL lock duration. This value will be used in lieu of TLV values.

EEM23	***EEM Module***
Category	Debug
Function	EEM triggers incorrectly when modules using wait states are enabled
Description	When modules using wait states (USB, MPY, CRC and FRAM controller in manual mode) are enabled, the EEM may trigger incorrectly. This can lead to an incorrect profile counter value or cause issues with the EEMs data watch point, state storage, and breakpoint functionality.
Workaround	None.

Note

This erratum affects debug mode only.

PMM32	*PMM Module*
Category	Functional
Function	Device may enter lockup state or execute unintentional code during transition from AM to LPM3/4
Description	The device might enter lockup state or start executing unintentional code resulting in unpredictable behavior depending on the contents of the address location- if any of the two conditions below occurs: Condition1: The following three events happen at the same time: 1) The device transitions from AM to LPM3/4 (e.g. during ISR exits or Status Register modifications), AND 2) An interrupt is requested (e.g. GPIO interrupt), AND 3) MODCLK is requested (e.g. triggered by ADC) or removed (e.g. end of ADC conversion). Modules which can trigger MODCLK clock requests/removals are ADC, eUSCI and CapTIvate (if exist). If clock events are started by the CPU (e.g. eUSCI during SPI master transmission), they can not occur at the same time as the power mode transition and thus should not be affected. The device should only be affected when the clock event is asynchronous to the power mode transition. The device can recover from this lockup condition by a PUC/BOR/Power cycle (e.g. enable Watchdog to trigger PUC). Condition2: The following events happen at the same time: 1) The device transitions from AM to LPM3/4 (e.g. during ISR exits or Status Register modifications), AND 2) An interrupt is requested (e.g. GPIO interrupt), AND 3) Neither MODCLK nor SMCLK are running (e.g. requested by a peripheral), AND 4) SMCLK is configured with a different frequency than MCLK.

PMM32 (continued) ***PMM Module***

The device can recover from this lockup condition by a BOR/Power cycle.

Workaround

1. Use LPM0/1/x.5 instead of LPM3/4.

OR

2. Place the FRAM in INACTIVE mode before any entry to LPM3/4 by clearing the FRPWR bit and FRLPMPWR bit (if exist) in the GCCTL0 register. This must be performed from RAM as shown below:

```
// define a function in RAM
#pragma CODE_SECTION(enterLpModeFromRAM,".TI.ramfunc")
void enterLpModeFromRAM(unsigned short lowPowerMode);

//call this function before any entry to LPM3/4
void enterLpModeFromRAM(unsigned short lowPowerMode)
{
FRCTL0 = FRCTLPW;
GCCTL0 &= ~(FRPWR+FRLPMPWR); //clear FRPWR and FRLPMPWR
FRCTL0_H = 0; //re-lock FRCTL
__bis_SR_register(lowPowerMode);
}
```

RTC15 ***RTC Module***

Category Functional

Function RTC Counter stops operating if RTC Counter clock source is changed from XT1CLK to another source while XT1CLK is stopped

Description If XT1CLK is used as the clock source for the RTC Counter and XT1CLK stops (e.g. oscillator fault), if the RTC Counter clock source is changed by user software (e.g. in the clock fault handling ISR) from XT1CLK to a different clock source while XT1CLK is stopped the RTC Counter hangs. In this hang state, the RTC Counter stops operating and cannot be restarted without a device reset via the hardware RST pin, a power-cycle of the device, or recovery of XT1CLK oscillation.

Workaround To change the RTC Counter clock source due to an oscillator fault, in the ISR for handling the OFIFG fault, use this software sequence:

1) Change the RTC Counter clock source away from XT1CLK normally

2) Reconfigure the XIN pin as a GPIO output, then toggle the GPIO twice with at least 2 rising or falling edges.

At this point the RTC Counter will be able to resume operation.

TB25 ***TB Module***

Category Functional

Function In up mode, TBxCCRn value is immediately transferred to TBxCLn when TBxCCTLn.CLLD bits are set or 0x01 or 0x10

TB25 (continued) ***TB Module***

Description

IF Timer B is configured for Up mode,
AND
the compare latch load event (TBxCCTLn.CLLD bits) setting is configured to update TBxCCRn when TBxR reaches 0,
THEN
TBxCCRn will update immediately instead of the described condition.

This is contrary to the user guide description of TBxCCTLn.CLLD = 0x01 or 0x10 modes.

Workaround

If user needs to update TBxCCRn value when TBxR counts to 0 in Timer B up mode:

1. Set TBxCCTLn. CLLD = 0x00
2. Enable the Timer B interrupt (TBIE) in TBxCTL
3. Update TBxCCRn value within interrupt routine.

Timer B Interrupt would need to be serviced in a timely manner to mitigate disruption or unintended timer output if an output mode is used.

USCI42 ***USCI Module***

Category

Functional

Function

UART asserts UCTXCPTIFG after each byte in multi-byte transmission

Description

UCTXCPTIFG flag is triggered at the last stop bit of every UART byte transmission, independently of an empty buffer, when transmitting multiple byte sequences via UART. The erroneous UART behavior occurs with and without DMA transfer.

Workaround

None.

USCI47 ***USCI Module***

Category

Functional

Function

eUSCI SPI slave with clock phase UCCKPH = 1

Description

The eUSCI SPI operates incorrectly under the following conditions:

1. The eUSCI_A or eUSCI_B module is configured as a SPI slave with clock phase mode UCCKPH = 1

AND

2. The SPI clock pin is not at the appropriate idle level (low for UCCKPL = 0, high for UCCKPL = 1) when the UCSWRST bit in the UCxxCTLW0 register is cleared.

If both of the above conditions are satisfied, then the following will occur:
eUSCI_A: the SPI will not be able to receive a byte (UCAxRXBUF will not be filled and UCRXIFG will not be set) and SPI slave output data will be wrong (first bit will be missed and data will be shifted).
eUSCI_B: the SPI receives data correctly but the SPI slave output data will be wrong (first byte will be duplicated or replaced by second byte).

Workaround

Use clock phase mode UCCKPH = 0 for MSP SPI slave if allowed by the application.

OR

USCI47 (continued) ***USCI Module***

The SPI master must set the clock pin at the appropriate idle level (low for UCCKPL = 0, high for UCCKPL = 1) before SPI slave is reset (UCSWRST bit is cleared).

OR

For eUSCI_A: to detect communication failure condition where UCRXIFG is not set, check both UCRXIFG and UCTXIFG. If UCTXIFG is set twice but UCRXIFG is not set, reset the MSP SPI slave by setting and then clearing the UCSWRST bit, and inform the SPI master to resend the data.

USCI50 ***USCI Module***

Category Functional

Function Data may not be transmitted correctly from the eUSCI when operating in SPI 4-pin master mode with UCSTEM = 0

Description When the eUSCI is used in SPI 4-pin master mode with UCSTEM = 0 (STE pin used as an input to prevent conflicts with other SPI masters), data that is moved into UCxTXBUF while the UCxSTE input is in the inactive state may not be transmitted correctly. If the eUSCI is used with UCSTEM = 1 (STE pin used to output an enable signal), data is transmitted correctly.

Workaround When using the STE pin in conflict prevention mode (UCSTEM = 0), only move data into UCxTXBUF when UCxSTE is in the active state. If an active transfer is aborted by UCxSTE transitioning to the master-inactive state, the data must be rewritten into UCxTXBUF to be transferred when UCxSTE transitions back to the master-active state.

7 Revision History

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

Changes from July 14, 2021 to August 27, 2021	Page
• TB25 was added to the errata documentation	6
• TB25 Description was updated	11
• TB25 Workaround was updated	11

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), 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, 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 (https:www.ti.com/legal/termsofsale.html) 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.

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