*Errata*

# *MSP430FR2673 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 A |
|---|---|
| COMP12 | ✓ |
| CPU46 | ✓ |
| TB25 | ✓ |
| USCI42 | ✓ |
| 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 A |
|---|---|
| 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 A |
|---|---|
| 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."

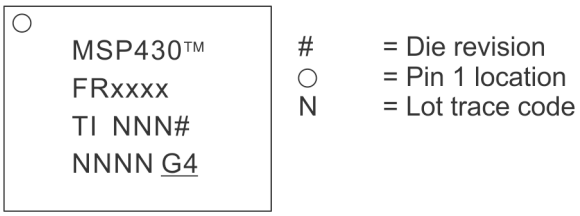
MSP devices have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

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

**RHB32** ***QFN (RHB), 32 Pin***




## 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

## COMP12 *COMP Module*

**Category** Functional

**Function** eCOMP0 output is not connected to timer B capture input channel

**Description** eCOMP0 output can not be selected internally to the Timer0_B7 CCI1B input (TB0CCTL1.CCIS = 01b)

**Workaround** Connect eCOMP0 output and Timer B capture input externally through GPIOs.

## 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 | |

**CPU22** (continued) ***CPU Module***

| IDE/Compiler | Version Number | Notes |
|---|---|---|
| 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

**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

## CPU46 *CPU Module*

**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

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. |

## EEM23 *EEM Module*

**Category**

Debug

**EEM23** (continued) ***EEM Module***

**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.

**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

**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.

**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

**USCI50** (continued) ***USCI Module***

**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 | 4 |
| • TB25 Description was updated | 7 |
| • TB25 Workaround was updated | 7 |

# 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