Errata
SLAZ550N–September 2013–Revised March 2017

MSP430FR4133 Device Erratasheet

1 Revision History

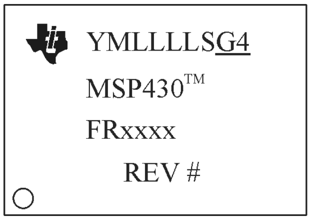
✓ The check mark indicates that the issue is present in the specified revision.

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

Errata Number	Rev B
ADC39	✓
ADC50	✓
ADC63	✓
CPU21	✓
CPU22	✓
CPU40	✓
CPU46	✓
CS11	✓
EEM23	✓
EEM28	✓
EEM30	✓
GC1	✓
PORT28	✓
SYS23	✓
USCI41	✓
USCI42	✓
USCI45	✓

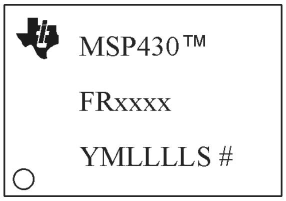
2 Package Markings

PM64 *LQFP (PM), 64 Pin*



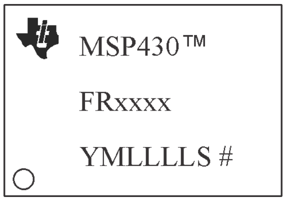
YM	=	Year and Month Date Code
LLLL	=	Assembly Lot Code
S	=	Assembly Site Code
#	=	Die Revision
○	=	Pin 1

DGG56 *(DGG), 56 Pin*



YM	=	Year and Month Date Code
LLLL	=	Assembly Lot Code
S	=	Assembly Site Code
#	=	Die Revision
○	=	Pin 1

DGG48 *(DGG), 48 Pin*



YM	=	Year and Month Date Code
LLLL	=	Assembly Lot Code
S	=	Assembly Site Code
#	=	Die Revision
○	=	Pin 1

3 Memory-Mapped Hardware Revision (TLV Structure)

Die Revision	TLV Hardware Revision
Rev B	20h

Further guidance on how to locate the TLV structure and read out the HW_ID can be found in the device User's Guide.

4 Detailed Bug Description

ADC39 *ADC10 Module*

Function Erroneous ADC results in extended sample mode

Description If the extended sample mode is selected (ADCSHP = 0) and the ADCCLK is asynchronous to the SHI signal, the ADC may generate erroneous results.

Workaround 1) Use the pulse sample mode (ADCSHP=1)

OR

2) Use a synchronous clock for ADC and the SHI signal. For example, if SMCLK is used to source Timer to trigger SHI, ADCCLK should also be sourced by SMCLK.

ADC50 *ADC10 Module*

Function Erroneous ADC conversion result for internal temperature sensor in LPM3 mode

Description When ACLK is used as ADC clock source and device is in LPM3 mode while sampling the on-chip temperature sensor, the ADC may generate erroneous conversion results.

Workaround 1) Use SMCLK or MODCLK as the ADC clock source. A 100us sampling time is required if triggering ADC conversion from LPM3.

OR

2) Use LPM0 or Active Mode.

ADC63 *ADC10 Module*

Function ADCHI/ADCLO may be reset unexpectedly when ADCCTL2 high byte is written byte-wise

Description ADCHI/ADCLO may be reset unexpectedly when ADCCTL2 high byte is written byte-wise.

Workaround Write to ADCCTL2 high byte in word-wise method.

CPU21 *CPUXv2 Module*

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

IDE/Compiler	Version Number	Notes
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 167 or later	

CPU22 *CPUXv2 Module*

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 *CPUXv2 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	
MSP430 GNU Compiler (MSP430-GCC)	Not affected	

CPU46 *CPUXv2 Module*

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

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.

CS11 — *CS Module*

Function FLL cannot lock at low or high temperature

Description The FLL may fail to lock under the following conditions:

(1) at low temperature -40C if DCOTAP (CSCTL0.DCO) value is too close to 0x1FF when FLL is locked at room temperature.

OR

(2) The FLL may fail to lock at high temperature 85C if DCOTAP (CSCTL0.DCO) value is too close to 0x0 when FLL is locked at room temperature.

Workaround 1. Calibration once at room temperature

(a) Check the DCOTAP value when FLL is locked at room temperature 22C-28C.

(b) If DCOTAP is between 0x190 and 0x70, go to step (d).

(c) If DCOTAP is greater than 0x190, set register DCOFTRIMEN=1 and increase register DCOFTRIM by 1; If DCOTAP is less than 0x70, set register DCOFTRIMEN=1 and decrease register DCOFTRIM by 1. Then repeat from step (a).

(d) Save calibrated DCOFTRIM value into FRAM.

(e) Set calibration flag and save it into FRAM.

AND

2. When user code boot up again, and if the calibration flag in FRAM is set, configure register DCOFTRIMEN=1 and load calibrated DCOFTRIM value from FRAM into register DCOFTRIM before enabling FLL to lock.

EEM23 — *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.

EEM28 — *EEM Module*

Function Clock outputs observed on port module during LPMx in debug mode

Description When the device is in LPMx mode, if a debug halt is requested and if the port pin is configured as MCLK, SMCLK, or ACLK output, these clocks are observed on the port pin. Depending on the LPM mode (see Device User's Guide), peripherals that are clocked from MCLK, SMCLK, or ACLK are still halted during debug halt state.

For example, if the device is in debug halt in LPM3 mode and a port pin is configured as SMCLK output, SMCLK can be observed on the pin. But the peripherals sourced from SMCLK are still halted as expected.

Workaround None

EEM30 *EEM Module*

Function Missed breakpoint if FRAM power supply is disabled

Description The FRAM power supply can be disabled (GCCTL0.FRPWR = 0) prior to LPM entry to save power. Upon wakeup, if a breakpoint is set on an the first instruction that accesses FRAM, the breakpoint may be missed.

Workaround None. This issue affects debug mode only.

GC1 *GC Module*

Function Uncorrectable memory bit error flag (GCCTL1.UBDIFG) does not trigger NMI

Description GCCTL1.UBDIFG flag is an interrupt flag that gets set if an uncorrectable bit error has been detected in non-volatile memory. The GCCTL1.UBDIFG flag does not trigger a NMI request even if GCCTL0.UBDRSTEN = 0. In this case, the application is not notified via a NMI request if an uncorrectable bit error occurred in non-volatile memory.

Workaround Set GCCTL0.UBDRSTEN = 1 to trigger a PUC. Check GCCTL1.UBDIFG flag after a PUC has been initiated.

PORT28 *PORT Module*

Function Pull-down resistor of TEST/SBWTCK pin

Description The device's internal pull-down resistor on the TEST/SBWTCK pin gets disabled if the SYS control bit SFRRPCR.SYSRSTRE is cleared. This can lead to increased current consumption and unintentionally-enabled JTAG access to the device.

Workaround 1) Do not clear the SFRRPCR.SYSRSTRE bit, use the SFRRPCR.SYSRSTRUP bit to define direction of the internal resistor on RST/NMI/SBWTDIO pin instead.

OR

2) Ensure a zero voltage level of TEST/SBWTCK pin by connecting the pin to an external component (e.g. external pull-down resistor) on the PCB.

SYS23 *SYS Module*

Function RAM not preserved after BOR

Description RAM content at addresses 0x20FE and 0x20FF is not preserved on BOR following one of the following conditions:

- RST/NMI pin reset (RST/NMI BOR)

- Software BOR (PMMSWBOR)

- Access security area (Security violation BOR)

- SVSH low condition event

Workaround None.

USCI41 *eUSCI Module*

Function UCBUSY bit of eUSCIA module stuck to 1 when device is in SPI mode.

Description When eUSCIA is configured in SPI mode, and the last transfer bit changes from 0 to 1, the UCBUSY bit gets stuck to 1. This happens in all four combinations of Clock Phase and Clock Polarity options (UCAxCTLW0.UCCKPH & UCAxCTLW0.UCCKPL bits). There is no data loss or corruption. Because the USCBUSY bit is stuck to 1, the clock request stays enabled and adds additional current consumption in low power mode operation.

Workaround Check on transmit or receive interrupt flag UCTXIFG/UCRXIFG instead of UCBUSY to know if the UCAxTXBUF buffer is empty or ready for the next complete character.

USCI42 *eUSCI Module*

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.

USCI45 *eUSCI Module*

Function Unexpected SPI clock stretching possible

Description In rare cases, during SPI communication, the clock high phase of the first data bit may be stretched significantly. The SPI operation completes as expected with no data loss. This issue only occurs when the USCI SPI module clock (UCxCLK) is asynchronous to the system clock (MCLK).

Workaround Ensure that the USCI SPI module clock (UCxCLK) and the CPU clock (MCLK) are synchronous to each other.

5 Document Revision History

Changes from device specific erratasheet to document Revision A.

1. CPU40 Workaround was updated.
2. ADC39 Workaround was updated.
3. Package Markings section was updated.
4. EEM23 Workaround was updated.
5. Silicon Revision B was added to the errata documentation.
6. Errata PMM23 was added to the errata documentation.
7. EEM23 Description was updated.
8. PORT23 Description was updated.
9. EEM23 Function was updated.
10. Errata USCI38 was added to the errata documentation.

Changes from document Revision A to Revision B.

1. Errata USCI38 was added to the errata documentation.

Changes from document Revision B to Revision C.

1. Errata SYS23 was added to the errata documentation.

Changes from document Revision C to Revision D.

1. Package Markings section was updated.
2. Errata ADC50 was added to the errata documentation.
3. Errata GC1 was added to the errata documentation.

Changes from document Revision D to Revision E.

1. Errata LCDE1 was removed from the errata documentation.
2. Errata PORT23 was removed from the errata documentation.
3. Errata USCI38 was removed from the errata documentation.
4. Device name changed from "XMS" to "MSP430"
5. Errata CPU43 was removed from the errata documentation.
6. Errata PMM23 was removed from the errata documentation.

Changes from document Revision E to Revision F.

1. EEM23 Description was updated.

Changes from document Revision F to Revision G.

1. Errata PORT28 was added to the errata documentation.

Changes from document Revision G to Revision H.

1. Errata USCI41 was added to the errata documentation.

Changes from document Revision H to Revision I.

1. Errata CS11 was added to the errata documentation.

Changes from document Revision I to Revision J.

1. Errata ADC63 was added to the errata documentation.

Changes from document Revision J to Revision K.

1. Errata USCI42 was added to the errata documentation.
2. Errata EEM30 was added to the errata documentation.

Changes from document Revision K to Revision L.

1. Errata CPU46 was added to the errata documentation.

Changes from document Revision L to Revision M.

1. CPU21 was added to the errata documentation.
2. USCI45 was added to the errata documentation.
3. CPU22 was added to the errata documentation.
4. Workaround for CPU40 was updated.
5. Description for USCI41 was updated.
6. Workaround for CPU46 was updated.

Changes from document Revision M to Revision N.

1. TLV Hardware Revision section was added to the documentation.
2. Workaround for CPU46 was updated.

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

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