MSP430FR2110 Device Erratasheet

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

1 Functional Errata Revision History

Errata impacting device's operation, function or parametrics.

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

Errata Number	Rev B
ADC50	✓
ADC63	✓
CPU46	✓
CS13	✓
PMM32	✓
RTC15	✓
USCI42	✓
USCI47	✓
USCI50	✓

2 Preprogrammed Software Errata Revision History

Errata impacting pre-programmed software into the silicon by Texas Instruments.

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

The device doesn't have Software in ROM errata.

3 Debug only Errata Revision History

Errata only impacting debug operation.

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

Errata Number	Rev B
EEM23	✓

4 Fixed by Compiler Errata Revision History

Errata completely resolved by compiler workaround. Refer to specific erratum for IDE and compiler versions with 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)

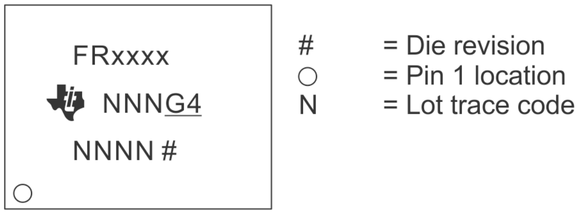
- MSP430 GCC Options: Check -msilicon-errata= and -msilicon-errata-warn= options
- MSP430 GCC User's Guide

IAR Embedded Workbench

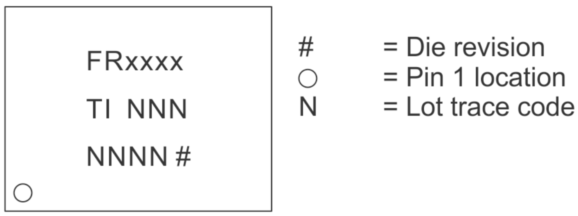
- IAR workarounds for msp430 hardware issues

5 Package Markings

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



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



6 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.

7 Detailed Bug Description

ADC50 — *ADC Module*

Category Functional

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 — *ADC Module*

Category Functional

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*

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 — *CPUXv2 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 *CPUXv2 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 — *CPUXv2 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

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.

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.

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.

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.

USCI42 — *eUSCI 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 — *eUSCI 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

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 — *eUSCI 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.

8 Document Revision History

Changes from device specific erratasheet to document Revision A.

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. Workaround for CPU46 was updated.

Changes from document Revision A to Revision B.

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

Changes from document Revision B to Revision C.

1. USCI45 was removed from the errata documentation.
2. USCI47 was added to the errata documentation.

Changes from document Revision C to Revision D.

1. Function for USCI47 was updated.
2. Description for USCI47 was updated.
3. Workaround for USCI47 was updated.

Changes from document Revision D to Revision E.

1. Workaround for USCI47 was updated.

Changes from document Revision E to Revision F.

1. RTC15 was added to the errata documentation.

Changes from document Revision F to Revision G.

1. USCI50 was added to the errata documentation.

Changes from document Revision G to Revision H.

1. Erratasheet format update.
2. Added errata category field to "Detailed bug description" section

Changes from document Revision H to Revision I.

1. Workaround for CPU40 was updated.

Changes from document Revision I to Revision J.

1. CS13 was added to the errata documentation.

Changes from document Revision J to Revision K.

1. PMM32 was added to the errata documentation.

Changes from document Revision K to Revision L.

1. CPU47 was added to the errata documentation.

Changes from document Revision L to Revision M.

1. CPU47 was removed from the errata documentation.