MSP430FR5994 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 C
ADC42	✓
ADC65	✓
ADC69	✓
CPU46	✓
CPU47	✓
CS12	✓
PMM31	✓
PMM32	✓
RTC12	✓
USCI42	✓
USCI45	✓
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.

Errata Number	Rev C
ADC67	✓

3 Debug only Errata Revision History

Errata only impacting debug operation.

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

The device doesn't have Debug errata.

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 C
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

PN80

LQFP (PN), 80 Pin

NNNNNNNG4
MSP430™
FRxxxx
REV #

= Die revision
○ = Pin 1 location
N = Lot trace code

ZVW87

NFBGA (ZVW), 87 pin

MSP430™
FRxxxx
NNNNNNN #
TI G1

= Die revision
○ = Pin 1 location
N = Lot trace code

PM64

LQFP (PM), 64 Pin

NNNNNNNG4
MSP430™
FRxxxx
REV #

= Die revision
○ = Pin 1 location
N = Lot trace code

RGZ48

QFN (RGZ), 48 Pin

MSP430™
FRxxxx
TI NNN #
NNNN G4

= Die revision
○ = Pin 1 location
N = Lot trace code

6 Memory-Mapped Hardware Revision (TLV Structure)

Die Revision	TLV Hardware Revision
Rev C	21h

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

ADC42 *ADC12_B Module*

Category Functional

Function ADC stops converting when successive ADC is triggered before the previous conversion ends

Description Subsequent ADC conversions are halted if a new ADC conversion is triggered while ADC is busy. ADC conversions are triggered manually or by a timer. The affected ADC modes are:

- sequence-of-channels

- repeat-single-channel

- repeat-sequence-of-channels (ADC12CTL1.ADC12CONSEQx)

In addition, the timer overflow flag cannot be used to detect an overflow (ADC12IFGR2.ADC12TOVIFG).

Workaround 1. For manual trigger mode (ADC12CTL0.ADC12SC), ensure each ADC conversion is completed by first checking ADC12CTL1.ADC12BUSY bit before starting a new conversion.

2. For timer trigger mode (ADC12CTL1.ADC12SHP), ensure the timer period is greater than the ADC sample and conversion time.

To recover the conversion halt:

1. Disable ADC module (ADC12CTL0.ADC12ENC = 0 and ADC12CTL0.ADC12ON = 0)

2. Re-enable ADC module (ADC12CTL0.ADC12ON = 1 and ADC12CTL0.ADC12ENC = 1)

3. Re-enable conversion

ADC65 *ADC12_B Module*

Category Functional

Function ADC12_B clock stays on between conversions in sequence-of-channels or repeated sequence-of-channels mode

Description When using the ADC in sequence-of-channels or repeat-sequence-of-channels mode (ADC12CONSEQx = 01 or 11), the ADC12_B always requests the ADC clock even between conversions. In this scenario, although the device may still enter LPM0, LPM1, LPM2 or LPM3, the selected ADC12_B clock source will always remain on, resulting in increased current consumption between ADC conversions.

Workaround To avoid the additional current consumption impact, different options will be needed depending on use case:

1. Configure ADC to Repeated-Single-Channel mode (ADC12CONSEQx = 10). Use the DMA or software to change the selected ADC12INCHx between conversions. With this option, the timing between conversions of different channels remains the same as normal ADC12 usage.

OR

2. Configure ADC to Sequence-of-Channels mode (ADC12CONSEQx = 01) with sequence of channels in Multiple Sample and Convert mode (ADC12CTL0.ADC12MSC = 1), then toggle the ADC12ENC bit by DMA or software after completing of each

conversion sequence. With this option, the conversions of each channel in the sequence will happen immediately after the previous channel instead of waiting for the next trigger. This needs to be considered if timing between the sampling of different channels in the sequence matters for the application.

ADC67 — *ADC12_B Module*

Category Software in ROM

Function Invalid ADC12 temperature sensor calibration data

Description The ADC12 reference temperature sensor calibration data stored in the TLV data structure (0x1A1A - 0x1A25) can be incorrect depending on the production lot trace code.

As a result the temperature measurement when using these data can be wrong.

Devices with lot trace code > 87XXXXX are not affected by this issue.

Workaround Record the calibration data by taking ADC measurements of the temperature sensor at 30C and 85C for the required reference voltage. The calibration data in the TLV section (0x1A1A - 0x1A25) can't be overwritten but the new calibration data can be stored in user FRAM or info memory for further temperature calculations.

ADC69 — *ADC12_B Module*

Category Functional

Function ADC stops operating if ADC clock source is changed from SMCLK to another source while SMCLKOFF = 1.

Description When SMCLK is used as the clock source for the ADC (ADC12CTL1.ADC12SSELx = 11) and CSCTL4.SMCLKOFF = 1, the ADC will stop operating if the ADC clock source is changed by user software (e.g. in the ISR) from SMCLK to a different clock source. This issue appears only for the ADC12CTL1.ADC12DIVx settings /3/5/7. The hang state can be recovered by PUC/POR/BOR/Power cycle.

Workaround 1. Set CSCTL4.SMCLKOFF = 0 before switch ADC clock source.

OR

2. Only use ADC12CTL1.ADC12DIVx as /1, /2, /4, /6, /8

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.

IDE/Compiler	Version Number	Notes
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.

CPU47 — *CPUXv2 Module*

Category Functional

Function An unexpected Vacant Memory Access Flag (VMAIFG) can be triggered

Description An unexpected Vacant Memory Access Flag (VMAIFG) can be triggered, if a PC-modifying instruction (e.g. - ret, push, call, pop, jmp, br) is fetched from the last addresses (last 4 or 8 byte) of a memory (e.g.- FLASH, RAM, FRAM) that is not contiguous to a higher, valid section on the memory map.

In debug mode using breakpoints the last 8 bytes are affected.

In free running mode the last 4 bytes are affected.

Workaround Edit the linker command file to make the last 4 or 8 bytes of affected memory sections unavailable, to avoid PC-modifying instructions on these locations.

Remaining instructions or data can still be stored on these locations.

CS12 — *CS Module*

Category Functional

Function DCO overshoot at frequency change

Description When changing frequencies (CSCTL1.DCOFSEL), the DCO frequency may overshoot and exceed the datasheet specification. After a time period of 10us has elapsed, the frequency overshoot settles down to the expected range as specified in the datasheet. The overshoot occur when switching to and from any DCOFSEL setting and impacts all peripherals using the DCO as a clock source. A potential impact can also be seen on FRAM accesses, since the overshoot may cause a temporary violation of FRAM access and cycle time requirements.

Workaround When changing the DCO settings, use the following procedure:

1) Store the existing CSCTL3 divider into a temporary unsigned 16-bit variable

2) Set CSCTL3 to divide all corresponding clock sources by 4 or higher

3) Change DCO frequency

4) Wait ~10us

5) Restore the divider in CSCTL3 to the setting stored in the temporary variable.

The following code example shows how to increase DCO to 16MHz.

```
uint16_t tempCSCTL3 = 0;
CSCTL0_H = CSKEY_H;         // Unlock CS registers
/* Assuming SMCLK and MCLK are sourced from DCO */
```

```
/* Store CSCTL3 settings to recover later */
tempCSCTL3 = CSCTL3;
/* Keep overshoot transient within specification by setting clk sources
to divide by 4*/
/* Clear the DIVS & DIVM masks (~0x77)and set both fields to 4 divider */
CSCTL3 = CSCTL3 & (~(0x77)) | DIVS__4 | DIVM__4;
CSCTL1 = DCOFSEL_4 | DCORSEL;        // Set DCO to 16MHz
/* Delay by ~10us to let DCO settle. 60 cycles = 20 cycles buffer +
(10us / (1/4MHz)) */
__delay_cycles(60);
CSCTL3 =  tempCSCTL3;      // Set all dividers
CSCTL0_H = 0;                    // Lock CS registers
```

PMM31 *PMM Module*

Category Functional

Function Device may enter lockup state during transition from AM to LPM2/3/4

Description

The device might enter lockup state if the MODOSC is requested (e.g. triggered by ADC) or removed (e.g. end of ADC conversion) during a power mode transition from AM to LPM2/3/4 (e.g. during ISR exits or Status Register modifications).

The same behavior can appear when SMCLK is requested during a power mode transition from AM to LPM3/4.

The device will remain in a lockup state unable to respond to interrupts or continue application execution until a power cycle or external reset brings it back to reset state.

Modules which can trigger MODCLK clock requests/removals are ADC and eUSCI in I2C mode using the clock low timeout feature (e.g. SMBus, PMBus).

Modules which can trigger SMCLK clock requests are ADC, eUSCI in I2C Master mode, eUSCI in SPI Master mode and eUSCI in UART mode.

If clock requests are started by the CPU/DMA (e.g. eUSCI during SPI master transmission), they can't 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 request is asynchronous to the power mode transition.

Workaround

1. Avoid using the aforementioned combinations of clock requests and power mode transitions:

Use LPM0/1 instead of LPM2/3/4 when expecting asynchronous MODCLK requests and removals.

OR

Use LPM0/1/2 instead of LPM3/4 when expecting asynchronous SMCLK requests.

OR

Use LPMx.5 instead of LPM2/3/4.

OR

Use a clock different than MODCLK/SMCLK when applicable (e.g. ACLK).

2. Prevent the power mode transition from happening when an asynchronous clock request/removal is expected:

Wake-up device before a UART byte is received.

AND

Wake-up device before an asynchronous ADC trigger and stay in Active Mode until conversion is completed.

AND

Keep device in AM/LPM0/LPM1 during ADC measurement.

PMM32 ***PMM Module***

Category Functional

Function Device may enter lockup state or execute unintentional code during transition from AM to LPM2/3/4

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 LPM2/3/4 (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 and eUSCI.

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 LPM2/3/4 (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 LPM2/3/4.

OR

2. Place the FRAM in INACTIVE mode before any entry to LPM2/3/4 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 LPM2/3/4
void enterLpModeFromRAM(unsigned short lowPowerMode)
{
FRCTL0 = FRCTLPW;
GCCTL0 &= ~(FRPWR+FRLPMPWR); //clear FRPWR and FRLPMPWR
FRCTL0_H = 0; //re-lock FRCTL
__bis_SR_register(lowPowerMode);
}
```

RTC12 — *RTC_C Module*

Category Functional

Function Real-time clock temperature compensation RTCTCOK bit not retained after LPM3.5 wake up

Description The RTC real-time clock temperature compensation write OK bit (RTCTCMP.RTCTCOK) is reset on wake up from LPM3.5 mode and does not get retained.

Workaround Store the RTCTCMP register content into FRAM for retention after wake up from LPM3.5

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.

USCI45 — *eUSCI Module*

Category Functional

Function Unexpected SPI clock stretching possible when UCxCLK is asynchronous to MCLK

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.

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. Errata ADC38 was removed from the errata documentation.
2. Errata CS12 was added to the errata documentation.
3. Module name for ADC42 was modified.
4. Errata USCI42 was added to the errata documentation.
5. Errata JTAG27 was added to the errata documentation.

Changes from document Revision A to Revision B.

1. LEA1 was added to the errata documentation.
2. RTC10 was added to the errata documentation.
3. ADC43 was added to the errata documentation.
4. ADC38 was added to the errata documentation.
5. USCI45 was added to the errata documentation.
6. PMM28 was added to the errata documentation.
7. PMM27 was added to the errata documentation.
8. USCI43 was added to the errata documentation.
9. CPU46 was added to the errata documentation.

Changes from document Revision B to Revision C.

1. Device name changed from "XMS" to "MSP430"
2. LEA1 was removed from the errata documentation.
3. RTC10 was removed from the errata documentation.
4. ADC43 was removed from the errata documentation.
5. JTAG27 was removed from the errata documentation.
6. ADC38 was removed from the errata documentation.
7. PMM25 was removed from the errata documentation.
8. COMP10 was removed from the errata documentation.
9. PMM28 was removed from the errata documentation.
10. PMM27 was removed from the errata documentation.
11. USCI43 was removed from the errata documentation.
12. CPU21 was added to the errata documentation.
13. CPU22 was added to the errata documentation.
14. Silicon Revision A was removed from the errata documentation.
15. Silicon Revision C was added to the errata documentation.
16. ZVW87 was added to errata documentation
17. RGZ48 was added to errata documentation
18. PM64 was added to errata documentation
19. Workaround for CPU40 was updated.

Changes from document Revision C to Revision D.

1. TLV Hardware Revision section was added to the documentation.
2. Workaround for RTC12 was updated.
3. Workaround for CPU46 was updated.

Changes from document Revision D to Revision E.

1. USCI47 was added to the errata documentation.

Changes from document Revision E to Revision F.

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

Changes from document Revision F to Revision G.

1. Workaround for USCI47 was updated.

Changes from document Revision G to Revision H.

1. USCI50 was added to the errata documentation.
2. ADC65 was added to the errata documentation.
3. Function for USCI45 was updated.

Changes from document Revision H to Revision I.

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

Changes from document Revision I to Revision J.

1. PMM31 was added to the errata documentation.
2. Workaround for CPU40 was updated.

Changes from document Revision J to Revision K.

1. ADC67 was added to the errata documentation.

Changes from document Revision K to Revision L.

1. PMM32 was added to the errata documentation.

Changes from document Revision L to Revision M.

1. ADC69 was added to the errata documentation.
2. CPU47 was added to the errata documentation.

IMPORTANT NOTICE AND DISCLAIMER