MSP430FR6989 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 E	Rev D	Rev C
ADC38	✓	✓	✓
ADC42	✓	✓	✓
ADC43	✓	✓	✓
ADC64	✓	✓	✓
ADC66	✓	✓	✓
ADC69	✓	✓	✓
AES1	✓	✓	✓
COMP7	✓	✓	✓
COMP10	✓	✓	✓
CPU46	✓	✓	✓
CPU47	✓	✓	✓
CS7	✓	✓	✓
CS12	✓	✓	✓
DMA7	✓	✓	✓
ESI2	✓	✓	✓
GC1	✓	✓	✓
GC4	✓	✓	✓
GC5	✓	✓	✓
PMM21	✓	✓	✓
PMM24			✓
PMM27		✓	✓
PMM29	✓	✓	✓
PMM31	✓	✓	✓
PMM32	✓	✓	✓
PORT28	✓	✓	✓
REF9	✓	✓	✓
RTC10	✓	✓	✓
RTC12	✓	✓	✓
TA22	✓	✓	✓
USCI41	✓	✓	✓
USCI42	✓	✓	✓
USCI45	✓	✓	✓

Errata Number	Rev E	Rev D	Rev C
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 E	Rev D	Rev C
EEM19	✓	✓	✓
EEM27	✓	✓	✓
EEM28	✓	✓	✓
EEM29			✓
EEM30	✓	✓	✓
EEM31	✓	✓	✓
JTAG27	✓	✓	✓

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 E	Rev D	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

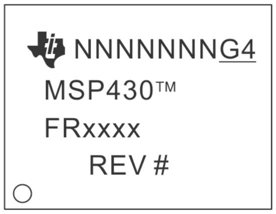
IAR Embedded Workbench

- IAR workarounds for msp430 hardware issues

5 Package Markings

PZ100

LQFP (PZ) 100 Pin



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

PN80

LQFP (PN), 80 Pin

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

6 Memory-Mapped Hardware Revision (TLV Structure)

Die Revision	TLV Hardware Revision
Rev E	30h
Rev D	22h
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

ADC38 — *ADC Module*

Category Functional

Function External ADC trigger without toggling ENC bit might prevent further ADC conversions.

Description The ADC may stop sampling and converting until the module is reset if an external (timer) trigger occurs without toggling the ADC12CTL0.ADC12ENC bit at:

- The end of sequence in the sequence-of-channel mode.

- The end of conversion in single-channel mode.

Workaround Ensure ADC12CTL0.ADC12ENC bit is always toggled before providing any new External Trigger to ADC.

ADC42 — *ADC 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

ADC43 — *ADC Module*

Category Functional

Function DMA does not trigger at the end of an ADC12 sequence of channels

Description The DMA transfer is triggered at the end of every ADC conversion when the ADC is configured to convert in a sequence of channels (ADC12CTL1.CONSEQ = 1 or 3.) This causes the DMA transfer to trigger prematurely after each ADC conversion instead of triggering only at the end of the conversion sequence.

Workaround Design the application to expect the DMA trigger at the end of every ADC conversion. For example, if a block transfer at the end of the sequence is originally desired, configure the DMA in single transfer mode with size = length of the sequence. The DMA transfer occurs at each conversion, but the DMA interrupt will still occur at the end of the sequence.

ADC64 *ADC Module*

Category Functional

Function Incorrect conversion result in extended sample mode in some conditions

Description The ADC12 conversion result can be incorrect if the extended sample mode is selected (ADC12SHP = 0), ADC12VRSEL is set to 0, 2, 4, 6, 12, 14 (VR+ and VR- unbuffered), and the ADC sample time is less than 6 ADC clock cycles.

Workaround 1) Use Pulse sample mode (ADC12SHP=1) if sample time less than 6 ADC clock cycles is needed;

OR

2) In extended sample mode (ADC12SHP = 0) increase the sample time to at least 6 ADC clock cycles;

OR

3) Use reference mode corresponding to ADC12VRSEL =1,3,5,7,9,13,15

ADC66 *ADC Module*

Category Functional

Function ADC stops converting when ADC12ON bit is toggled during conversion

Description Subsequent ADC conversions are halted if the ADC12CTL0.ADC12ON bit is toggled while the ADC is busy. The affected ADC modes are:

- sequence-of-channels

- repeat-single-channel

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

Workaround Stop the ADC conversion by clearing the ADC12CTL0.ADC12ENC bit.

Check the ADC12CTL1.ADC12BUSY flag for 0 before toggling the ADC12CTL0.ADC12ON bit.

ADC69 *ADC 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

AES1 — *AES Module*

Category Functional

Function Ongoing AES operation cannot be aborted by writing to AESAXIN

Description Writing to AESAXIN register when AESASTAT.AESBUSY bit is set does abort the ongoing AES operation or set the AESACTL0.AESERRFG bit.

Workaround Always let AES operation run to completion (i.e. do not abort). Ignore the encryption/decryption output if AESAXIN is written when AESASTAT.AESBUSY is set.

COMP7 — *COMP Module*

Category Functional

Function Comparator triggers false output at low overdrive levels

Description When the differential voltage on the comparator input pins is smaller than the comparator offset according to the datasheet, the comparator can provide a false output.

Workaround Drive the differential voltage to above the comparator offset according to the datasheet.

COMP10 — *COMP Module*

Category Functional

Function Comparator port output toggles when entering or leaving LPM3/LPM4

Description The comparator port pin output (CECTL1.CEOUT) erroneously toggles when device enters or leaves LPM3/LPM4 modes under the following conditions:

1) Comparator is disabled (CECTL1.CEON = 0)

AND

2) Output polarity is enabled (CECTL1.CEOUTPOL = 1)

AND

3) The port pin is configured to have CEOUT functionality.

For example, if the CEOUT pin is high when the device is in Active Mode, CEOUT pin becomes low when the device enters LPM3/LPM4 modes.

Workaround When the comparator is disabled, ensure at least one of the following:

1) Output inversion is disabled (CECTL.CEOUTPOL = 0)

OR

2) Change pin configuration from CEOUT to GPIO with output low.

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

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

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.

CPU47 — *CPU 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.

CS7 — *CS Module*

Category Functional

Function DCO clock frequency out of specification when returning from LPM2, LPM3 or LPM4

Description When waking up from LPM2, LPM3 or LPM4 the first clocks generated by the DCO are not within the specified frequency range for approximately 13us (independent of the selected frequency). Any observable overshoot of the frequency is not critical for the device functionality. Frequency undershoots can be considered as additional wake-up delay because the frequency is below the target and less clocks are generated than expected. The overall impact of the clock overshoots and undershoots during stabilization is approximately 2us of additional delay.

Workaround Account for frequency undershoots as additional wake-up delay of about 2us.

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
```

DMA7 — *DMA Module*

Category Functional

Function DMA request may cause the loss of interrupts

Description

If a DMA request starts executing during the time when a module register containing an interrupt flags is accessed with a read-modify-write instruction, a newly arriving interrupt from the same module can get lost. An interrupt flag set prior to DMA execution would not be affected and remain set.

Workaround

1. Use a read of Interrupt Vector registers to clear interrupt flags and do not use read-modify-write instruction.

OR

2. Disable all DMA channels during read-modify-write instruction of specific module registers containing interrupts flags while these interrupts are activated.

EEM19 — *EEM Module*

Category Debug

Function DMA may corrupt data in debug mode

Description

When the DMA is enabled and the device is in debug mode, the data written by the DMA may be corrupted when a breakpoint is hit or when the debug session is halted.

Workaround

This erratum has been addressed in MSPDebugStack version 3.5.0.1. It is also available in released IDE EW430 IAR version 6.30.3 and CCS version 6.1.1 or newer.

If using an earlier version of either IDE or MSPDebugStack, do not halt or use breakpoints during a DMA transfer.

NOTE: This erratum applies to debug mode only.

EEM27 *EEM Module*

Category

Debug

Function

Switching off FRAM LDO stalls device during debug access

Description

With the "Enable Ultra Low Power debug/LPMx.5 debug" option disabled in the IDE, if user application switches off the FRAM LDO (FRPWR = 0) and a debug halt is requested during this time, device debug control is lost and the debug session must be restarted. At this point, the code execution is also stalled.

The following error message is observed:

IAR - "Internal error: (State)"

CCS - "MSP430: Trouble Halting Target CPU: Internal error"

Workaround

If IDE error message is observed, restart the debug session or perform a hardware reset. Turn on "Enable Ultra Low Power debug/LPMx.5 debug" option in the IDE debug settings.

EEM28 *EEM Module*

Category

Debug

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

EEM29 *EEM Module*

Category

Debug

Function

A breakpoint after a conditional jump is missed when wait-states are used

Description

A hardware breakpoint set on a code line immediately following a conditional jump will not be hit when the application uses a wait-state. This also affects single-stepping C code through a conditional jump. A conditional jump could be if-else, for-loops, or switch-case statements.

Note: This erratum affects debug mode only.

Workaround 1) Insert a __no_operation() immediately before the intended line of code that a breakpoint will be set on. For example:

```
if (a) {
    __no_operation();   \\ workaround
    \\ your application code -- set breakpoint
}
else {
    __no_operation();   \\ workaround
    \\ your application code -- set breakpoint
}
```

Or

2) Operate the debugger in Free Run mode.

Or

3) Single-step on disassembler level

EEM30 *EEM Module*

Category Debug

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.

EEM31 *EEM Module*

Category Debug

Function Breakpoint trigger may be lost when MPU is enabled

Description A data value written to FRAM can be used as a trigger condition for breakpoints during a debug session. This trigger can be lost if the FRAM access is made to an address that has been write-protected by the MPU.

Workaround None. This issue affects debug mode only.

ESI2 *SCANIF Module*

Category Functional

Function TSM1 register corruption

Description When CPU performs write operations to any ESI register during active TSM (Timing State Machine) sequence, the TSM1 register might be corrupted. The critical scenario is a CPU write access at the end of the TSM sequence.

Workaround Gate ESI write accesses during TSM active phase by reading the TSM register pointer from ESIDEBUG2 to ensure TSM is in IDLE state (TSM_Index = 0).

```
__bic_SR_register(GIE);      // disable interrupts  important
                             // to not interrupt the SW gating
while (ESIDEBUG2_H != 0x00);// check TSM state pointer to
                             // ensure IDLE state before
```

```
                              // write access
ESICNT2 = 0x00000;            // example write to any ESI
                              // register
__bis_SR_register(GIE);       // re-enable interrupts
```

Due to this workaround the device stays maximum 1 TSM sequence longer in Active mode.

The CPU access must not extend the Idle time of the TSM.

The gating should be used for each write operation or a sequence of ESI write accesses in a row.

GC1 — *GC Module*

Category Functional

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

Description The GCCTL1.UBDIFG flag is an interrupt flag that gets set if an uncorrectable bit error has been detected in non-volatile memory. Even the GCCTL1.UBDIFG flag is set to 1 (GCCTL0.UBDRSTEN = 0 and GCCTL0.UBDIE = 1), it does not trigger a NMI request. In this case, the application is not notified via a NMI request that an uncorrectable bit error occurred in non-volatile memory (SYSSNIV = 0).

Workaround Set GCCTL0.UBDRSTEN = 1 and GCCTL0.UBDIE = 0 to trigger a PUC and check GCCTL1.UBDIFG = 1 after each PUC for manual interrupt flag handling.

Please consider GC4 errata for side effects.

GC4 — *GC Module*

Category Functional

Function Unexpected PUC is triggered

Description During execution from FRAM a non-existent uncorrectable bit error can be detected and trigger a PUC if the uncorrectable bit error detection flag is set (GCCTL0.UBDRSTEN = 1). This behavior appears only if:

(1) MCLK is sourced from DCO frequency of 16 MHz

OR

(2) MCLK is sourced by external high frequency clock above 12 MHz at pin HFXIN

OR

(3) MCLK is sourced by High-Frequency crystals (HFXT) above 12 MHz.

This PUC will not be recognized by the SYSRSTIV register (SYSRSTIV = 0x00).

A PUC RESET will be executed with not defined reset source.

Also the corresponding bit error detection flag is not set (GCCTL1.UBDIFG = 0).

Workaround 1. Check the reset source for SYSRSTIV = 0 and ignore the reset.

OR

2. Set GCCTL0.UBDRSTEN = 0 to prevent unexpected PUC. NMI event will not be triggered, even if GCCTL0.UBDIE = 1 -> consider GC1 Errata for more details.

OR

3. Set the MCLK to maximum 12MHz to leverage the uncorrectable bit error PUC

feature.

GC5 — *GC Module*

Category Functional

Function Nonexistent FRAM failures can be detected after wake-up from LPM 1/2/3/4

Description The FRAM bit error detection may indicate bit errors, even the memory has no failure, after wakeup from LPM1/2/3/4.

Based on the setting inside the FRAM controller registers (GCCTL0), following behaviors can appear.

1. Unexpected PUC for an uncorrectable FRAM error can be triggered and causing the corresponding value in the SYSRSTIV register.

This happens only if GCCTL0.UBDRSTEN =1.

2. Unexpected NMI for an uncorrectable FRAM error can be triggered and causing the corresponding value in the SYSSNIV register.

This happens only if the GCCTL0.UBDIE = 1.

3. Unexpected NMI for a correctable FRAM error can be triggered and causing the corresponding value in the SYSSNIV register.

This happens only if the GCCTL0.CBDIE =1.

Workaround 1. Disable PUC (GCCTL0.UBDRSTEN=0), UBDIE and CBDIE interrupts (GCCTL0.UBDIE=0 and GCCTL0.CBDIE=0) prior to entering LPM 1/2/3/4.

2. After LPM wake up, clear GCCTL1.UBDIFG and GCCTL1.CBDIFG, and then reinitialize the GCCTL0 register after the first valid FRAM access has been completed. For the valid FRAM access the user has to consider possible cache hits which depends on implementation.

JTAG27 — *JTAG Module*

Category Debug

Function Unintentional code execution after programming via JTAG/SBW

Description The device can unintentionally start executing code from uninitialized RAM addresses 0x0006 or 0x0008 after being programming via the JTAG or SBW interface. This can result in unpredictable behavior depending on the contents of the address location.

Workaround 1. If using programming tools purchased from TI (MSP-FET, LaunchPad), update to CCS version 6.1.3 later or IAR version 6.30 or later to resolve the issue.

2. If using the MSP-GANG Production Programmer, use v1.2.3.0 or later.

3. For custom programming solutions refer to the specification on MSP430 Programming Via the JTAG Interface User's Guide (SLAU320) revision V or newer and use MSPDebugStack v3.7.0.12 or later.

For MSPDebugStack (MSP430.DLL) in CCS or IAR, download the latest version of the development environment or the latest version of the MSPDebugStack

NOTE: This only affects debug mode.

PMM21 — *PMM Module*

Category Functional

Function Long wake-up time from LPM4.5 at -40C when SVS is disabled

Description At -40degC and SVS disabled (SVSHE = 0), the device wake-up time from LPM4.5 to active mode is out of specification and can be up to 50ms.

Workaround None.

PMM24 *PMM Module*

Category Functional

Function Device may enter lockup state during wake-up from LPM3 and LPM4

Description The device may enter a lockup state during an interrupt-triggered wake up from LPM3 or LPM4. The device will remain in lockup state, unable to respond to the interrupt or continue application execution, until a power cycle brings it back to reset state.

LPM3.5 and LPM4.5 are not affected by this behavior.

Workaround 1) Use LPM2 instead of LPM3 or LPM4. Refer to the device specific datasheet for details on LPM2 wake up time and power consumption.

OR

2) If the application only uses RTC or GPIO as a wakeup source, use LPM3.5 or LPM4.5 instead. Refer to the device specific datasheet for details on LPM3.5/LPM4.5 wake up times and power consumption.

Note: When using LPM3.5/LPM4.5, the Compute Through Power Loss (CTPL) utility APIs (part of the FRAM software utilities) can be used to configure device behavior prior to LPM entry and on wake-up.

PMM27 *PMM Module*

Category Functional

Function Device may reset when waking up from LPM2 or LPM3

Description When the device is in LPM2/LPM3 and the eUSCI UART module is enabled and waiting to receive a byte, an unintentional device reset (BOR) may be triggered if the following two conditions are met:

1) There are exactly five other peripherals (excluding the eUSCI UART) that are both active AND requesting ACLK for example Timer_A or RTC

AND

2) Interrupts from other peripherals occur within a 1us time window of the eUSCI UART detecting the start bit of the first received byte

Workaround Do not use exactly five active peripherals requesting ACLK, when the eUSCI UART is enabled in LPM2/LPM3. Instead use less than OR greater than five active peripherals to prevent a BOR from occurring.

PMM29 *PMM Module*

Category Functional

Function Device may enter lockup state during wake-up from LPM2, LPM3, and LPM4

Description In rare cases, the device may enter lockup state during wake up from LPM2, LPM3, or LPM4. The device will remain in lockup state, unable to respond to interrupts or continue application execution, until a BOR reset occurs.

LPM0, LPM1, LPM3.5 and LPM4.5 are not affected by this behavior.

Workaround 1) Use LPM0 or LPM1. See device datasheet for details on wake up time and power consumption.

OR

2) Use LPM3.5 or LPM4.5 Note that only RTC or GPIO can wake from LPM3.5/4.5 and see device datasheet for details on wake up time and power consumption. When using LPM3.5/4.5 the Compute Through Power Loss (CTPL) Utility APIs, found in the FRAM Utilities download, can be used to configure device behavior prior to LPM entry and on wake-up.

OR

3) At the beginning of code, clear the FRLPMPWR bit in the GCCTL0 register, as shown below:

```
// PMM29 workaround.
FRCTL0 = FRCTLPW;
GCCTL0 = FRPWR; //clear FRLPMPWR while keeping FRPWR set
FRCTL0_H = 0; //re-lock FRCTL
// End PMM29 workaround
```

This adds additional latency when waking from LPM to enter the ISR. To calculate the new wake up time with FRLPMPWR bit cleared, take the wake-up time for the low power mode used and add the $t_{\text{wake-up FRAM}}$ value specified in the datasheet.

E.g. $t_{\text{wake-up workaround}} = t_{\text{wake-up LPM3}} + t_{\text{wake-up FRAM}}$

NOTE: For workaround (3), if the WDT triggers a PUC reset during LPM2, 3 or 4 the FRLPMPWR bit will be re-set before the wake-up occurs, meaning the workaround will not be effective and the part could still enter lock-up state. In this case it is recommended to configure the WDT to interval timer mode and trigger a PUC reset via WDT PW violation.

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, eUSCI in UART mode and ESI.

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, ESI internal clock).

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);
}
```

PORT28 — *PORT Module*

Category Functional

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.

REF9 — *REF Module*

Category Functional

Function REFON Feature

Description The Reference module does not not provide REF voltage to Comparator module when the REFON bit is set (REFCTL0.REFON=1).

Workaround 1. Use REFBGOT bit of the REFCTL0 regsiter instead of REFON bit to provide REF voltage to Comparator.

OR

2. Enable the Comparator module with internal REF setting (CEREFL + CERS bits of the CECTL2 register) to request the REF module.

RTC10 — *RTC Module*

Category Functional

Function RTC interrupt flag can be lost during LPMx.5 entry

Description An RTC interrupt flag can get lost if it triggers within a small critical time window of the device's entry into LPM3.5. This results in the RTC interrupt flag not triggering a wake-up from LPM3.5. The subsequent RTC interrupt flag is captured to wake device up from LPM3.5.

Workaround Use LPM3 for timing-critical applications where the device is entering LPM3.5 close to the RTC interrupt flag triggering.

RTC12 — *RTC 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

TA22 — *TIMER_A Module*

Category Functional

Function Timer A0 output toggles upon entry into LPM3/LPM4

Description If the output unit on Timer A0 is enabled for any of the capture/compare blocks and the device enter LPM3 or LPM4, the Timer A0 output toggles. If the Timer A0 output was high, it goes low upon LPM3/4 entry and if the Timer A0 output was low, it goes high upon LPM3/4 entry.

Workaround None. This issue impacts Timer A0 only, all other instances of Timer A operate as specified.

USCI41 *eUSCI Module*

Category Functional

Function UCBUSY bit of eUSCIA module might not work reliable when device is in SPI mode.

Description When eUSCIA is configured in SPI mode, the UCBUSY bit might get stuck to 1 or start toggling after transmission is completed. This happens in all four combinations of Clock Phase and Clock Polarity options (UCAxCTLW0.UCCKPH & UCAxCTLW0.UCCKPL bits) as well as in Master and Slave mode. There is no data loss or corruption. However the UCBUSY cannot be used in its intended function to check if transmission is completed. Because the UCBUSY bit is stuck to 1 or toggles, the clock request stays enabled and this adds additional current consumption in low power mode operation.

Workaround For correct functional implementation 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.

To reduce the additional current it is recommended to either reset the SPI module (UCAxCTLW0.UCSWRST) in the UCBxCTLW0 or send a dummy byte 0x00 after the intended SPI transmission is completed.

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 CS3 was added to the errata documentation.
2. Errata EEM26 was added to the errata documentation.
3. Errata USCI37 was added to the errata documentation.
4. Errata CS4 was added to the errata documentation.
5. PORT16 Function was updated.
6. CPU43 Description was updated.
7. BSL9 Workaround was updated.
8. Errata TAB25 was added to the errata documentation.
9. Errata SYS20 was added to the errata documentation.
10. CPU43 Workaround was updated.
11. EEM22 Function was updated.
12. EEM23 Function was updated.
13. Errata PMM21 was added to the errata documentation.
14. EEM19 Workaround was updated.
15. EEM23 Workaround was updated.
16. Errata ADC32 was added to the errata documentation.
17. EEM23 Description was updated.
18. EEM25 Function was updated.
19. Errata USCI36 was added to the errata documentation.
20. CPU43 Function was updated.
21. Errata PMM22 was added to the errata documentation.
22. Errata BSL9 was added to the errata documentation.
23. EEM22 Description was updated.
24. PORT16 Workaround was updated.
25. EEM22 Workaround was updated.
26. Errata ESIF1 was added to the errata documentation.
27. EEM25 Workaround was updated.
28. EEM25 Description was updated.
29. EEM19 Function was updated.
30. PORT16 Description was updated.

Changes from document Revision A to Revision B.

1. Errata ADC40 was added to the errata documentation.
2. Errata EEM28 was added to the errata documentation.
3. Errata EEM27 was added to the errata documentation.
4. Silicon Revision B was added to the errata documentation.

Changes from document Revision B to Revision C.

1. EEM26 Workaround was updated.
2. EEM19 Workaround was updated.
3. Errata PORT19 was added to the errata documentation.
4. EEM23 Workaround was updated.
5. ADC38 Function was updated.
6. EEM23 Description was updated.
7. ADC32 Workaround was updated.

8. PMM22 Description was updated.
9. Errata ADC41 was added to the errata documentation.
10. ADC38 Description was updated.
11. CPU40 Workaround was updated.
12. ADC38 Workaround was updated.
13. ADC40 Workaround was updated.
14. ADC40 Description was updated.
15. Errata ADC36 was added to the errata documentation.
16. EEM26 Description was updated.
17. Errata COMP7 was added to the errata documentation.
18. ADC32 Function was updated.
19. Errata REF2 was added to the errata documentation.
20. PMM22 Function was updated.
21. EEM23 Function was updated.
22. ADC32 Description was updated.
23. EEM19 Description was updated.

Changes from document Revision C to Revision D.

1. COMP7 Workaround was updated.
2. Errata PMM22 was removed from the errata documentation.
3. Errata ADC42 was added to the errata documentation.
4. COMP7 Description was updated.
5. Errata GC1 was added to the errata documentation.

Changes from document Revision D to Revision E.

1. Errata MPU2 was added to the errata documentation.
2. Errata ADC43 was added to the errata documentation.
3. Errata JTAG24 was added to the errata documentation.

Changes from document Revision E to Revision F.

1. Errata USCI37 was removed from the errata documentation.
2. Errata USCI36 was removed from the errata documentation.
3. Errata EEM22 was removed from the errata documentation.
4. Errata ADC40 was removed from the errata documentation.
5. Errata REF2 was removed from the errata documentation.
6. Errata BSL8 was removed from the errata documentation.
7. Errata RTC9 was removed from the errata documentation.
8. Errata PMM21 was removed from the errata documentation.
9. Errata EEM25 was removed from the errata documentation.
10. Errata PORT19 was removed from the errata documentation.
11. Errata WDG5 was removed from the errata documentation.
12. Package Markings section was updated.
13. Errata CS3 was removed from the errata documentation.
14. Errata ADC41 was removed from the errata documentation.
15. Errata SYS20 was removed from the errata documentation.
16. Errata EEM26 was removed from the errata documentation.
17. Module name for ESIF1 was modified.
18. Errata ADC36 was removed from the errata documentation.

19. Errata ADC32 was removed from the errata documentation.
20. Device name changed from "XMS" to "MSP430"
21. Errata EEM23 was removed from the errata documentation.
22. Errata MPU2 was removed from the errata documentation.
23. Errata SYS19 was removed from the errata documentation.
24. Errata PORT16 was removed from the errata documentation.
25. Errata CS4 was removed from the errata documentation.
26. Errata XOSC12 was removed from the errata documentation.
27. Errata TAB25 was removed from the errata documentation.
28. Errata PORT18 was removed from the errata documentation.
29. Errata BSL9 was removed from the errata documentation.

Changes from document Revision F to Revision G.

1. Errata CPU43 was removed from the errata documentation.

Changes from document Revision G to Revision H.

1. Errata RTC12 was added to the errata documentation.
2. DMA7 Workaround was updated.
3. Errata PORT28 was added to the errata documentation.
4. DMA7 Description was updated.

Changes from document Revision H to Revision I.

1. ADC38 Function was updated.
2. Errata AES1 was added to the errata documentation.
3. ADC38 Description was updated.
4. ADC38 Workaround was updated.

Changes from document Revision I to Revision J.

1. Errata USCI41 was added to the errata documentation.

Changes from document Revision J to Revision K.

1. Errata PMM24 was added to the errata documentation.

Changes from document Revision K to Revision L.

1. EEM19 Workaround was updated.
2. Errata REF9 was added to the errata documentation.

Changes from document Revision L to Revision M.

1. Silicon Revision D was added to the errata documentation.

Changes from document Revision M to Revision N.

1. Errata CS12 was added to the errata documentation.
2. Errata TA22 was added to the errata documentation.
3. Errata USCI42 was added to the errata documentation.
4. Errata EEM30 was added to the errata documentation.
5. Errata JTAG27 was added to the errata documentation.
6. Errata COMP10 was added to the errata documentation.

Changes from document Revision N to Revision O.

1. Errata PMM27 was added to the errata documentation.
2. Errata EEM31 was added to the errata documentation.
3. Silicon Revision E was added to the errata documentation.
4. Errata CPU46 was added to the errata documentation.

Changes from document Revision O to Revision P.

1. PMM27 is no longer impacting silicon Revision E

Changes from document Revision P to Revision Q.

1. CPU21 was added to the errata documentation.
2. CPU22 was added to the errata documentation.
3. USCI45 was added to the errata documentation.
4. Workaround for RTC10 was updated.
5. Workaround for CPU40 was updated.
6. Workaround for CPU46 was updated.
7. Description for USCI41 was updated.

Changes from document Revision Q to Revision R.

1. ADC64 was added to the errata documentation.
2. ESI2 was added to the errata documentation.
3. TLV Hardware Revision section was added to the documentation.
4. Workaround for RTC12 was updated.
5. Workaround for CPU46 was updated.

Changes from document Revision R to Revision S.

1. USCI47 was added to the errata documentation.
2. Workaround for ESI2 was updated.

Changes from document Revision S to Revision T.

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

Changes from document Revision T to Revision U.

1. Workaround for USCI47 was updated.

Changes from document Revision U to Revision V.

1. PMM29 was added to the errata documentation.

Changes from document Revision V to Revision W.

1. Workaround for PMM29 was updated.

Changes from document Revision W to Revision X.

1. USCI50 was added to the errata documentation.
2. ADC66 was added to the errata documentation.
3. Function for USCI45 was updated.

Changes from document Revision X to Revision Y.

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

Changes from document Revision Y to Revision Z.

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

Changes from document Revision Z to Revision AA.

1. GC4 was added to the errata documentation.
2. ADC67 was added to the errata documentation.
3. Description for GC1 was updated.
4. Workaround for GC1 was updated.

Changes from document Revision AA to Revision AB.

1. ADC67 was removed from the errata documentation.
2. PMM32 was added to the errata documentation.

Changes from document Revision AB to Revision AC.

1. GC5 was added to the errata documentation.
2. CPU47 was added to the errata documentation.
3. ADC69 was added to the errata documentation.

Changes from document Revision AC to Revision AD.

1. Function for USCI41 was updated.
2. Description for USCI41 was updated.
3. Workaround for USCI41 was updated.

Changes from document Revision AD to Revision AE.

1. Workaround for GC5 was updated.

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