

MSP430F5152 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
ADC39	✓
ADC42	✓
ADC69	✓
COMP10	✓
CPU46	✓
CPU47	✓
DMA4	✓
DMA7	✓
DMA10	✓
PMAP1	✓
PMM14	✓
PMM15	✓
PMM18	✓
PMM20	✓
PMM26	✓
PORT15	✓
PORT19	✓
PORT21	✓
SYS12	✓
SYS16	✓
TD1	✓
TD2	✓
UCS9	✓
UCS11	✓
USCI26	✓
USCI31	✓
USCI34	✓
USCI35	✓
USCI39	✓
USCI40	✓

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
BSL7	✓

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 C
EEM11	✓
EEM17	✓
EEM19	✓
EEM21	✓
EEM23	✓
JTAG26	✓
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 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](#)

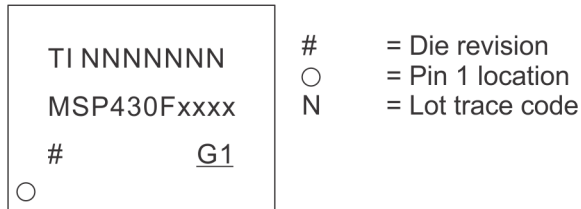
IAR Embedded Workbench

- [IAR workarounds for msp430 hardware issues](#)

5 Package Markings

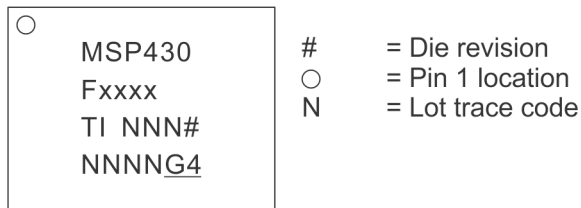
YFF40

DSBGA (YFF), 40 Pin



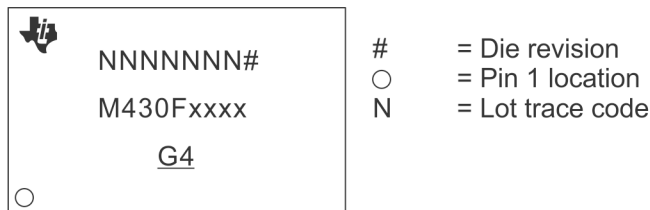
RSB40

QFN (RSB), 40 Pin



DA38

TSSOP (DA), 38 Pin



6 Memory-Mapped Hardware Revision (TLV Structure)

Die Revision	TLV Hardware Revision
Rev C	30h

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

ADC39	<i>ADC10_A Module</i>
Category	Functional
Function	Erroneous ADC10 results in extended sample mode
Description	If the extended sample mode is selected (ADC10SHP = 0) and the ADC10CLK is asynchronous to the SHI signal, the ADC10 may generate erroneous results.
Workaround	<ol style="list-style-type: none"> 1) Use the pulse sample mode (ADC10SHP=1) OR 2) Use a synchronous clock for ADC10 and the SHI signal.
ADC42	<i>ADC10_A Module</i>
Category	Functional
Function	ADC stops converting when successive ADC is triggered before the previous conversion ends
Description	<p>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:</p> <ul style="list-style-type: none"> - sequence-of-channels - repeat-single-channel - repeat-sequence-of-channels (ADC12CTL1.ADC12CONSEQx) <p>In addition, the timer overflow flag cannot be used to detect an overflow (ADC12IFGR2.ADC12TOVIFG).</p>
Workaround	<ol style="list-style-type: none"> 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. <p>To recover the conversion halt:</p> <ol style="list-style-type: none"> 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
ADC69	<i>ADC10_A Module</i>
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

BSL7
BSL Module

Category

Software in ROM

Function

BSL does not start after waking up from LPMx.5

Description

When waking up from LPMx.5 mode, the BSL does not start as it does not clear the Lock I/O bit (LOCKLPM5 bit in PM5CTL0 register) on start-up.

Workaround

1. Upgrade the device BSL to the latest version (see Creating a Custom Flash-Based Bootstrap Loader (BSL) Application Note - SLAA450 for more details)
- OR
2. Do not use LOCKLPM5 bit (LPMx.5) if the BSL is used but cannot be upgraded.

COMP10
COMP_B 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
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 performs 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.

DMA4

DMA Module

Category

Functional

Function

Corrupted write access to 20-bit DMA registers

Description

When a 20-bit wide write to a DMA address register (DMAxSA or DMAxDA) is interrupted by a DMA transfer, the register contents may be unpredictable.

Workaround

1. Design the application to guarantee that no DMA access interrupts 20-bit wide accesses to the DMA address registers.

OR

2. When accessing the DMA address registers, enable the Read Modify Write disable bit (DMARMWDIS = 1) or temporarily disable all active DMA channels (DMAEN = 0).

OR

3. Use word access for accessing the DMA address registers. Note that this limits the values that can be written to the address registers to 16-bit values (lower 64K of Flash).

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.

DMA10
DMA Module

Category

Functional

Function

DMA access may cause invalid module operation

Description

The peripheral modules MPY, CRC, USB, RF1A and FRAM controller in manual mode can stall the CPU by issuing wait states while in operation. If a DMA access to the module occurs while that module is issuing a wait state, the module may exhibit undefined behavior.

Workaround

Ensure that DMA accesses to the affected modules occur only when the modules are not in operation. For example with the MPY module, ensure that the MPY operation is completed before triggering a DMA access to the MPY module.

EEM11
EEM Module

Category

Debug

Function

Conditional register write trigger fails while executing rotate instructions

Description

A conditional register write trigger will fail to generate the expected breakpoint if the trigger condition is a result of executing one of the following rotate instructions: RRUM, RRCM, RRAM and RLAM.

Workaround

None

NOTE: This erratum applies to debug mode only.

EEM17
EEM Module

Category

Debug

Function

Wrong Breakpoint halt after executing Flash Erase/Write instructions

Description

Hardware breakpoints or Conditional Address triggered breakpoints on instructions that follow Flash Erase/Write instructions, stops the debugger at the actual Flash Erase/Write instruction even though the flash erase/write operation has already been executed. The hardware/conditional address triggered breakpoints that are placed on either the next two single opcode instructions OR the next double opcode instruction that follows the Flash Erase/Write instruction are affected by this erratum.

Workaround

None. Use other conditional/advanced triggered breakpoints to halt the debugger right after Flash erase/write instructions.

NOTE: This erratum affects debug mode only.

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.

EEM21
EEM Module

Category

Debug

Function

LPMx.5 debug limitations

Description

Debugging the device in LPMx.5 mode might wake the device up from LPMx.5 mode inadvertently, and it is possible that the device enters a lock-up condition; that is, the device cannot be accessed by the debugger any more.

Workaround

Follow the debugging steps in Debugging MSP430 LPM4.5 [SLAA424](#) .

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.

JTAG26
JTAG Module

Category

Debug

Function

LPMx.5 Debug Support Limitations

Description

The JTAG connection to the device might fail at device-dependent low or high supply

voltage levels if the LPMx.5 debug support feature is enabled. To avoid a potentially unreliable debug session or general issues with JTAG device connectivity and the resulting bad customer experience Texas Instruments has chosen to remove the LPMx.5 debug support feature from common MSP430 IDEs including TIs Code Composer Studio 6.1.0 with msp430.emu updated to version 6.1.0.7 and IARs Embedded Workbench 6.30.2, which are based on the MSP430 debug stack MSP430.DLL 3.5.0.1
<http://www.ti.com/tool/MSPDS>

TI plans to re-introduce this feature in limited capacity in a future release of the debug stack by providing an IDE override option for customers to selectively re-activate LPMx.5 debug support if needed. Note that the limitations and supply voltage dependencies outlined in this erratum will continue to apply.

For additional information on how the LPMx.5 debug support is handled within the MSP430 IDEs including possible workarounds on how to debug applications using LPMx.5 without toolchain support refer to [Code Composer Studio User's Guide for MSP430 chapter F.4](#) and [IAR Embedded Workbench User's Guide for MSP430 chapter 2.2.5](#).

Workaround

1. If LPMx.5 debug support is deemed functional and required in a given scenario:

a) Do not update the IDE to continue using a previous version of the debug stack such as MSP430.DLL v3.4.3.4.

OR

b) Roll back the debug stack by either performing a clean re-installation of a previous version of the IDE or by manually replacing the debug stack with a prior version such as MSP430.DLL v3.4.3.4 that can be obtained from <http://www.ti.com/tool/MSPDS>.

2. In case JTAG connectivity fails during the LPMx.5 debug mode, the device supply voltage level needs to be raised or lowered until the connection is working.

Do not enable the LPMx.5 debug support feature during production programming.

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 programmed 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.

PMP1
PMP Module

Category

Functional

Function

Port Mapping Controller does not clear unselected inputs to mapped module.

Description The Port Mapping Controller provides the logical OR of all port mapped inputs to a module (Timer, USCI, etc). If the PSEL bit (PxSEL.y) of a port mapped input is cleared, then the logic level of that port mapped input is latched to the current logic level of the input. If the input is in a logical high state, then this high state is latched into the input of the logical OR. In this case, the input to the module is always a logical 1 regardless of the state of the selected input.

Workaround

1. Drive input to the low state before clearing the PSEL bit of that input and switching to another input source.

or

2. Use the Port Mapping Controller reconfiguration feature, PMAPRECFG, to select inputs to a module and map only one input at a time.

PMM14 *PMM Module*

Category Functional

Function Increasing the core level when SVS/SVM low side is configured in full-performance mode causes device reset

Description When the SVS/SVM low side is configured in full performance mode (SVSMLCTL.SVSLFP = 1), the setting time delay for the SVS comparators is ~2us. When increasing the core level in full-performance mode; the core voltage does not settle to the new level before the settling time delay of the SVS/SVM comparator expires. This results in a device reset.

Workaround When increasing the core level; enable the SVS/SVM low side in normal mode (SVSMLCTL.SVSLFP=0). This provides a settling time delay of approximately 150us allowing the core sufficient time to increase to the expected voltage before the delay expires.

PMM15 *PMM Module*

Category Functional

Function Device may not wake up from LPM2, LPM3, or LPM4

Description Device may not wake up from LPM2, LPM3 or LPM4 if an interrupt occurs within 1 us after the entry to the specified LPMx; entry can be caused either by user code or automatically (for example, after a previous ISR is completed). Device can be recovered with an external reset or a power cycle. Additionally, a PUC can also be used to reset the failing condition and bring the device back to normal operation (for example, a PUC caused by the WDT).

This effect is seen when:

- A write to the SVSMHCTL and SVSMLCTL registers is immediately followed by an LPM2, LPM3, LPM4 entry without waiting the requisite settling time ((PMMIFG.SVSMLDLYIFG = 0 and PMMIFG.SVSMHDLYIFG = 0)).

or

The following two conditions are met:

- The SVSL module is configured for a fast wake-up or when the SVSL/SVML module is turned off. The affected SVSMLCTL register settings are shaded in the following table.

SVSL	SVSLE	SVSLMD	SVSLFP	AM, LPM0/1 SVSL state	Manual SVSMLACE = 0	Automatic SVSMLACE = 1	Wakeup Time LPM2/3/4
					LPM2/3/4 SVSL State	LPM2/3/4 SVSL State	
	0	x	x	OFF	OFF	OFF	$t_{WAKE-UP\ FAST}$
	1	0	0	Normal	OFF	OFF	$t_{WAKE-UP\ SLOW}$
	1	0	1	Full Performance	OFF	OFF	$t_{WAKE-UP\ FAST}$
	1	1	0	Normal	Normal	OFF	$t_{WAKE-UP\ SLOW}$
	1	1	1	Full Performance	Full Performance	Normal	$t_{WAKE-UP\ FAST}$

SVML	SVMLE	SVMLFP	AM, LPM0/1 SVML state	Manual SVSMLACE = 0	Automatic SVSMLACE = 1	Wakeup Time LPM2/3/4
				LPM2/3/4 SVML State	LPM2/3/4 SVML State	
	0	x	OFF	OFF	OFF	$t_{WAKE-UP\ FAST}$
	1	0	Normal	Normal	OFF	$t_{WAKE-UP\ SLOW}$
	1	1	Full Performance	Full Performance	Normal	$t_{WAKE-UP\ FAST}$

and

-The SVSH/SVMH module is configured to transition from Normal mode to an OFF state when moving from Active/LPM0/LPM1 into LPM2/LPM3/LPM4 modes. The affected SVSMHCTL register settings are shaded in the following table.

SVSH	SVSHE	SVSHMD	SVSHFP	AM, LPM0/1 SVSH state	Manual SVSMHACE = 0	Automatic SVSMHACE = 1
					LPM2/3/4 SVSH State	LPM2/3/4 SVSH State
	0	x	x	OFF	OFF	OFF
	1	0	0	Normal	OFF	OFF
	1	0	1	Full Performance	OFF	OFF
	1	1	0	Normal	Normal	OFF
	1	1	1	Full Performance	Full Performance	Normal

SVMH	SVMHE	SVMHFP	AM, LPM0/1 SVMH state	Manual SVSMHACE = 0	Automatic SVSMHACE = 1
				LPM2/3/4 SVMH State	LPM2/3/4 SVMH State
	0	x	OFF	OFF	OFF
	1	0	Normal	Normal	OFF
	1	1	Full Performance	Full Performance	Normal

Workaround

Any write to the SVSMxCTL register must be followed by a settling delay (PMMIFG.SVSMLDLYIFG = 0 and PMMIFG.SVSMHDLYIFG = 0) before entering LPM2, LPM3, LPM4.

and

1. Ensure the SVSx, SVMx are configured to prevent the issue from occurring by the following:

- Configure the SVSL module for slow wake up (SVSLFP = 0). Note that this will increase the wakeup time from LPM2/3/4 to twakeupslow (~150 us).

or

- Do not configure the SVSH/SVMH such that the modules transition from Normal mode to an OFF state on LPM entry and ensure SVSH/SVMH is in manual mode. Instead force the modules to remain ON even in LPMx. Note that this will cause increased power consumption when in LPMx.

Refer to the MSP430 Driver Library([MSPDRIVERLIB](#)) for proper PMM configuration functions.

Use the following function, PMM15Check (void), to determine whether or not the existing PMM configuration is affected by the erratum. The return value of the function is 1 if the configuration is affected, and 0 if the configuration is not affected.

unsigned char PMM15Check (void)

```
{
// First check if SVSL/SVML is configured for fast wake-up
if ( (!(SVSMLCTL & SVSLE)) || ((SVSMLCTL & SVSLE) && (SVSMLCTL & SVSLFP)) ||
(!(SVSMLCTL & SVMLE)) || ((SVSMLCTL & SVMLE) && (SVSMLCTL & SVMLEFP)) )
{ // Next Check SVSH/SVMH settings to see if settings are affected by PMM15
if ((SVSMHCTL & SVSHE) && (!(SVSMHCTL & SVSHFP)))
{
if ( (!(SVSMHCTL & SVSHMD)) || ((SVSMHCTL & SVSHMD) &&
(SVSMHCTL & SVSMHACE)) )
return 1; // SVSH affected configurations
}
if ((SVSMHCTL & SVMHE) && (!(SVSMHCTL & SVMHFP)) && (SVSMHCTL &
SVSMHACE))
return 1; // SVMH affected configurations
}
return 0; // SVS/M settings not affected by PMM15
}
}
```

2. If fast servicing of interrupts is required, add a 150us delay either in the interrupt service routine or before entry into LPM3/LPM4.

PMM18

PMM Module

Category

Functional

Function

PMM supply overvoltage protection falsely triggers POR

Description

The PMM Supply Voltage Monitor (SVM) high side can be configured as overvoltage protection (OVP) using the SVMHOVPE bit of SVSMHCTL register. In this mode a POR should typically be triggered when DVCC reaches ~3.75V.

If the OVP feature of SVM high side is enabled going into LPM234, the SVM might trigger at DVCC voltages below 3.6V (~3.5V) within a few ns after wake-up. This can falsely cause an OVP-triggered POR. The OVP level is temperature sensitive during fail scenario and decreases with higher temperature (85 degC ~3.2V).

Workaround

Use automatic control mode for high-side SVS & SVM (SVSMHCTL.SVSMHACE=1). The SVM high side is inactive in LPM2, LPM3, and LPM4.

PMM20

PMM Module

Category

Functional

Function Unexpected SVSL/SVML event during wakeup from LPM2/3/4 in fast wakeup mode

Description If PMM low side is configured to operate in fast wakeup mode, during wakeup from LPM2/3/4 the internal V_{CORE} voltage can experience voltage drop below the corresponding SVSL and SVML threshold (recommendation according to User's Guide) leading to an unexpected SVSL/SVML event. Depending on PMM configuration, this event triggers a POR or an interrupt.

NOTE: As soon the SVSL or the SVML is enabled in Normal performance mode the device is in slow wakeup mode and this erratum does not apply.

In addition, this erratum has sporadic characteristic due to an internal asynchronous circuit. The drop of V_{core} does not have an impact on specified device performance.

Workaround If SVSL or SVML is required for application (to observe external disruptive events at V_{core} pin) the slow wakeup mode has to be used to avoid unexpected SVSL/SVML events. This is achieved if the SVSL or the SVML is configured in "Normal" performance mode (not disabled and not in "Full" Performance Mode).

PMM26 *PMM Module*

Category Functional

Function Device lock-up if RST pin pulled low during write to SVSMHCTL or SVSMLCTL

Description Device results in lock-up condition under one of the two scenarios below:

- 1) If RST pin is pulled low during write access to SVSMHCTL, with the RST/NMI pin is configured to reset function and is pulled low (reset event) the device will stop code execution and is continuously held in reset state. RST pin is no longer functional. The only way to come out of the lock-up situation is a power cycle.

OR

- 2) If RST pin is pulled low during write access to SVSMLCTL and only if the code that checks for SVSMLDLYIFG==1 is implemented without a timeout. The device will be stuck in the polling loop polling since SVSMLDLYIFG will never be cleared.

Workaround Follow the sequence below to prevent the lock-up for both use cases:

- 1) Disable RST pin reset function and switch to NMI before access SVSMHCTL or SVSMLCTL.

then

- 2) Activate NMI interrupt and handle reset events in this time by SW (optional if reset functionality required during access SVSMHCTL or SVSMLCTL)

then

- 3) Enable RST pin reset function after access to SVSMHCTL or SVSMLCTL

To prevent lock-up caused by use case #2 a timeout for the SVSMLDLYIFG flag check should be implemented to 300us.

PORT15 *PORT Module*

Category Functional

Function In-system debugging causes the PMALOCKED bit to be always set

Description The port mapping controller registers cannot be modified when single-stepping or halting at break points between a valid password write to the PMAPWD register and the expected lock of the port mapping (PMAP) registers. This causes the PMAPLOCKED bit to remain set and not clear as expected.

Note: This erratum only applies to in-system debugging and is not applicable when operating in free-running mode.

Workaround Do not single step through or place break points in the port mapping configuration section of code.

PORT19 ***PORT Module***

Category Functional

Function Port interrupt may be missed on entry to LPMx.5

Description If a port interrupt occurs within a small timing window (~1MCLK cycle) of the device entry into LPM3.5 or LPM4.5, it is possible that the interrupt is lost. Hence this interrupt will not trigger a wakeup from LPMx.5.

Workaround None

PORT21 ***PORT Module***

Category Functional

Function Setting PxSEL bit for XTAL pins

Description Setting the PxSEL bit of XIN pin does not disable the digital function of the XOUT pin (in non-bypass mode). The primary port function will still be active on the XOUT pin.

Workaround Set the PxSEL bit of XOUT pin explicitly to disable the port function of the XOUT pin.

SYS12 ***SYS Module***

Category Functional

Function Invalid ACCVIFG when DVcc in the range of 2.4 to 2.6V

Description A Flash Access Violation Interrupt Flag (ACCVIFG) may be triggered by the Voltage Changed During Program Error bit (VPE) when DVcc is in the range of 2.4 to 2.6V. However the VPE does not signify an invalid flash operation has occurred.

If the ACCVIE bit is set and a flash operation is executed in the affected voltage range, an unnecessary interrupt is requested. The bootstrap loader also cannot be used to execute write/erase flash operations in this voltage range, because it exits the flash operation and returns an error on an ACCVIFG event.

Workaround None

SYS16 ***SYS Module***

Category Functional

Function Fast Vcc ramp after device power up may cause a reset

Description At initial power-up, after Vcc crosses the brownout threshold and reaches a constant level, an abrupt ramp of Vcc at a rate $dV/dT > 1V/100\mu s$ can cause a brownout condition to be incorrectly detected even though Vcc does not fall below the brownout threshold. This causes the device to undergo a reset.

Workaround Use a controlled Vcc ramp to power up the device.

TD1 *TIMER_D Module*

Category Functional

Function Timer halt on EXTCLR event

Description When the TEC module is configured to enable external asynchronous signals on the TECxCLR (TEC external clear) pin to clear the timer counter on the selected edge and the timer is halted after an external clear event but before next positive edge of the timer clock, then due to the erratum, it is not possible to write to the timer counter (TDxR) until the next positive timer clock edge occurrence. Halting of the timer right after an external clear event is possible if the system clock (MCLK) much greater than the Timer_D clock and the application halts the timer in the external clear interrupt service routine or if the application is doing random timer halts. This erratum does not cause any dead-lock conditions and the timer counter can be written into when the next positive timer clock edge is available.

Workaround If required to halt the timer and change the timer counter value on an external clear event, wait until one timer clock period has elapsed to change the timer counter value.

TD2 *TIMER_D Module*

Category Functional

Function Up/Down mode with high resolution enabled

Description The Timer_D Up/Down mode is not functional in high-resolution mode. The PWM signals are generated incorrectly in the down phase.

Timer_D can only be used in Up mode in high-resolution mode.

Workaround None

UCS9 *UCS Module*

Category Functional

Function Digital Bypass mode prevents entry into LPM4

Description When entering LPM4, if an external digital input applied to XT1 in HF mode or XT2 is not turned off, the PMM does not switch to low-current mode causing higher than expected power consumption.

Workaround Before entering LPM4:

- (1) Switch to a clock source other than external bypass digital input.

OR

- (2) Turn off external bypass mode (UCSCTL6.XT1BYPASS = 0).

UCS11	<i>UCS Module</i>
Category	Functional
Function	Modifying UCSCTL4 clock control register triggers an additional erroneous clock request
Description	<p>Changing the SELM/SELS/SELA bits in the UCSCTL4 register will correctly configure the respective clock to use the intended clock source but might also erroneously set XT1/XT2 fault flag if the crystals are not present at XT1/XT2 or not configured in the application firmware. If the NMI interrupt for the OFIFG is enabled, an unintentional NMI interrupt will be triggered and needs to be handled.</p> <hr/> <p>NOTE: The XT1/XT2 fault flag can be set regardless of which SELM/SELS/SELA bit combinations are being changed.</p> <hr/>
Workaround	<p>Clear all the fault flags in UCSCTL7 register once after changing any of the SELM/SELS/SELA bits in the UCSCTL4 register.</p> <p>If OFIFG-NMI is enabled during clock switching, disable OFIFG-NMI interrupt during changing the SELM/SELS/SELA bits in the UCSCTL4 register to prevent unintended NMI.</p> <p>Alternatively it can be handled accordingly (clear falsely set fault flags) in the Interrupt Service Routine to ensure proper OFIFG clearing.</p>
USCI26	<i>USCI Module</i>
Category	Functional
Function	Tbuf parameter violation in I2C multi-master mode
Description	<p>In multi-master I2C systems the timing parameter Tbuf (bus free time between a stop condition and the following start) is not guaranteed to match the I2C specification of 4.7us in standard mode and 1.3us in fast mode. If the UCTXSTT bit is set during a running I2C transaction, the USCI module waits and issues the start condition on bus release causing the violation to occur.</p> <p>Note: It is recommended to check if UCBBUSY bit is cleared before setting UCTXSTT=1.</p>
Workaround	None
USCI31	<i>USCI Module</i>
Category	Functional
Function	Framing Error after USCI SW Reset (UCSWRST)
Description	While receiving a byte over USCI-UART (with UCBUSY bit set), if the application resets the USCI module (software reset via UCSWRST), then a framing error is reported for the next receiving byte.
Workaround	<ol style="list-style-type: none"> 1. If possible, do not reset USCI-UART during an ongoing receive operation; that is, when UCBUSY bit is set. 2. If the application software resets the USCI module (via the UCSWRST bit) during an ongoing receive operation, then set and reset the UCSYNC bit before releasing the

```
software USCI reset.
Workaround code sequence:
bis #UCSWRST, &UCAxCTL1 ; USCI SW reset
;Workaround begins
bis #UCSYNC, &UCAxCTL0 ; set synchronous mode
bic #UCSYNC, &UCAxCTL0 ; reset synchronous mode
;Workaround ends
bic #UCSWRST, &UCAxCTL1 ; release USCI reset
```

USCI34
USCI Module
Category

Functional

Function

I2C multi-master transmit may lose first few bytes.

Description

In an I2C multi-master system (UCMM =1), under the following conditions:

(1)the master is configured as a transmitter (UCTR =1)

AND

(2)the start bit is set (UCTXSTT =1);

if the I2C bus is unavailable, then the USCI module enters an idle state where it waits and checks for bus release. While in the idle state it is possible that the USCI master updates its TXIFG based on clock line activity due to other master/slave communication on the bus. The data byte(s) loaded in TXBUF while in idle state are lost and transmit pointers initialized by the user in the transmit ISR are updated incorrectly.

Workaround

Verify that the START condition has been sent (UCTXSTT =0) before loading TXBUF with data.

Example:

```
#pragma vector = USCIAB0TX_VECTOR
__interrupt void USCIAB0TX_ISR(void)
{
// Workaround for USCI34
if(UCB0CTL1&UCTXSTT)
{
// TXData = pointer to the transmit buffer start
// PTxData = pointer to transmit in the ISR
PTxData = TXData; // restore the transmit buffer pointer if the Start bit is set
}
//
if(IFG2&UCB0TXIFG)
{
if (PTxData<=PTxDataEnd) // Check TX byte counter
{
UCB0TXBUF = *PTxData++; // Load TX buffer
```

```

}
else
{
UCB0CTL1 |= UCTXSTP; // I2C stop condition
IFG2 &= ~UCB0TXIFG; // Clear USCI_B0 TX int flag
__bic_SR_register_on_exit(CPUOFF); // Exit LPM0
}
}
}

```

USCI35
USCI Module

Category

Functional

Function

Violation of setup and hold times for (repeated) start in I2C master mode

Description

In I2C master mode, the setup and hold times for a (repeated) START, $t_{SU,STA}$ and $t_{HD,STA}$ respectively, can be violated if SCL clock frequency is greater than 50kHz in standard mode (100kbps). As a result, a slave can receive incorrect data or the I2C bus can be stalled due to clock stretching by the slave.

Workaround

If using repeated start, ensure SCL clock frequencies is < 50kHz in I2C standard mode (100 kbps).

USCI39
USCI Module

Category

Functional

Function

USCI I2C IFGs UCSTTIFG, UCSTPIFG, UCNACKIFG

Description

Unpredictable code execution can occur if one of the hardware-clear-able IFGs UCSTTIFG, UCSTPIFG or UCNACKIFG is set while the global interrupt enable is set by software (GIE=1). This erratum is triggered if ALL of the following events occur in following order:

1. Pending Interrupt: One of the UCxIFG=1 AND UCxIE=1 while GIE=0
2. The GIE is set by software (e.g. EINT)
3. The pending interrupt is cleared by hardware (external I2C event) in a time window of 1 MCLK clock cycle after the "EINT" instruction is executed.

Workaround

Disable the UCSTTIE, UCSTPIE and UCNACKIE before the GIE is set. After GIE is set, the local interrupt enable flags can be set again.

Assembly example:

```
bic #UCNACKIE+UCSTPIE+UCSTTIE, UCBxIE ; disable all self-clearing interrupts
```

```
NOP
```

```
EINT
```

```
bis #UCNACKIE+UCSTPIE+UCSTTIE, UCBxIE ; enable all self-clearing interrupts
```

USCI40
USCI Module

Category	Functional
Function	SPI Slave Transmit with clock phase select = 1
Description	In SPI slave mode with clock phase select set to 1 (UCAxCTLW0.UCCKPH=1), after the first TX byte, all following bytes are shifted by one bit with shift direction dependent on UCMSB. This is due to the internal shift register getting pre-loaded asynchronously when writing to the USCIA TXBUF register. TX data in the internal buffer is shifted by one bit after the RX data is received.
Workaround	Reinitialize TXBUF before using SPI and after each transmission. If transmit data needs to be repeated with the next transmission, then write back previously read value: <code>UCAxTXBUF = UCAxTXBUF;</code>

8 Document Revision History

Changes from family erratasheet to device specific erratasheet.

1. Errata CPU39 was removed
2. Errata EEM8 was removed
3. Errata EEM13 was removed
4. Errata FLASH37 was removed
5. Errata JTAG21 was removed
6. Errata PORT16 was removed
7. Errata MPY1 was removed
8. Errata PMM11 was removed
9. Errata PMM12 was removed
10. Errata SYS10 was removed
11. Errata UCS7 was removed
12. Errata UCS10 was removed
13. Errata USCI30 was removed
14. Revision A was removed
15. Revision B was removed

Changes from device specific erratasheet to document Revision A.

1. Errata PORT19 was added to the errata documentation.
2. Errata PMM18 was added to the errata documentation.

Changes from document Revision A to Revision B.

1. Errata DMA10 was added to the errata documentation.
2. Errata BSL7 was added to the errata documentation.

Changes from document Revision B to Revision C.

1. DMA10 Description was updated.
2. DMA10 Function was updated.

Changes from document Revision C to Revision D.

1. DMA10 Description was updated.
2. Errata EEM23 was added to the errata documentation.
3. Errata CPU43 was added to the errata documentation.

Changes from document Revision D to Revision E.

1. SYS16 Description was updated.
2. Errata TD1 was added to the errata documentation.
3. CPU43 Description was updated.
4. Errata PORT21 was added to the errata documentation.
5. Device TLV Hardware Revision information added to erratasheet.

Changes from document Revision E to Revision F.

1. Errata PMM20 was added to the errata documentation.
2. Errata USCI35 was added to the errata documentation.

Changes from document Revision F to Revision G.

1. BSL7 Workaround was updated.
2. BSL7 Function was updated.

Changes from document Revision G to Revision H.

1. EEM19 Workaround was updated.

2. EEM17 Workaround was updated.
3. Errata TD2 was added to the errata documentation.
4. CPU43 Description was updated.
5. EEM11 Workaround was updated.
6. EEM23 Workaround was updated.
7. EEM17 Description was updated.
8. EEM23 Description was updated.
9. Errata ADC39 was added to the errata documentation.
10. EEM19 Description was updated.

Changes from document Revision H to Revision I.

1. DMA10 Workaround was updated.
2. DMA10 Description was updated.
3. DMA10 Function was updated.

Changes from document Revision I to Revision J.

1. CPU40 Workaround was updated.
2. EEM19 Workaround was updated.
3. Errata USCI39 was added to the errata documentation.
4. Package Markings section was updated.
5. EEM23 Workaround was updated.
6. EEM23 Description was updated.
7. Errata ADC42 was added to the errata documentation.
8. EEM23 Function was updated.
9. EEM19 Description was updated.

Changes from document Revision J to Revision K.

1. Errata USCI40 was added to the errata documentation.
2. Errata CPU43 was removed from the errata documentation.
3. PMM18 Workaround was updated.

Changes from document Revision K to Revision L.

1. DMA7 Workaround was updated.
2. EEM23 Description was updated.
3. DMA7 Description was updated.

Changes from document Revision L to Revision M.

1. USCI39 Description was updated.

Changes from document Revision M to Revision N.

1. Errata JTAG26 was added to the errata documentation.

Changes from document Revision N to Revision O.

1. EEM19 Workaround was updated.
2. Errata PMM26 was added to the errata documentation.

Changes from document Revision O to Revision P.

1. UCS11 Workaround was updated.
2. UCS11 Description was updated.
3. UCS11 Function was updated.

Changes from document Revision P to Revision Q.

1. Errata JTAG27 was added to the errata documentation.

2. Errata COMP10 was added to the errata documentation.

Changes from document Revision Q to Revision R.

1. USCI39 Workaround was updated.
2. SYS12 Description was updated.
3. Errata CPU46 was added to the errata documentation.

Changes from document Revision R to Revision S.

1. CPU21 was added to the errata documentation.
2. CPU22 was added to the errata documentation.
3. Workaround for CPU40 was updated.
4. Workaround for CPU46 was updated.
5. Workaround for PMM15 was updated.

Changes from document Revision S to Revision T.

1. Workaround for CPU46 was updated.

Changes from document Revision T to Revision U.

1. Workaround for PMM15 was updated.

Changes from document Revision U to Revision V.

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

Changes from document Revision V to Revision W.

1. Workaround for CPU40 was updated.

Changes from document Revision W to Revision X.

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

Changes from document Revision X to Revision Y.

1. USCI34 was added to the errata documentation.
2. YFF40 was added to errata documentation

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2020, Texas Instruments Incorporated