

# MSP430F5232 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
<a href="#">COMP10</a>	✓
<a href="#">CPU47</a>	✓
<a href="#">DMA4</a>	✓
<a href="#">DMA7</a>	✓
<a href="#">DMA10</a>	✓
<a href="#">PMAP1</a>	✓
<a href="#">PMM9</a>	✓
<a href="#">PMM11</a>	✓
<a href="#">PMM12</a>	✓
<a href="#">PMM14</a>	✓
<a href="#">PMM15</a>	✓
<a href="#">PMM18</a>	✓
<a href="#">PMM20</a>	✓
<a href="#">PORT15</a>	✓
<a href="#">PORT19</a>	✓
<a href="#">PORT33</a>	✓
<a href="#">RTC3</a>	✓
<a href="#">RTC6</a>	✓
<a href="#">SYS12</a>	✓
<a href="#">SYS16</a>	✓
<a href="#">UCS7</a>	✓
<a href="#">UCS9</a>	✓
<a href="#">UCS11</a>	✓
<a href="#">USCI26</a>	✓
<a href="#">USCI34</a>	✓
<a href="#">USCI35</a>	✓
<a href="#">USCI39</a>	✓
<a href="#">USCI40</a>	✓

## 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
<a href="#">BSL7</a>	✓

## 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
<a href="#">EEM17</a>	✓
<a href="#">EEM19</a>	✓
<a href="#">EEM21</a>	✓
<a href="#">EEM23</a>	✓

## 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
<a href="#">CPU21</a>	✓
<a href="#">CPU22</a>	✓
<a href="#">CPU40</a>	✓

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

### RGZ48

#### QFN (RGZ), 48 Pin

<div style="text-align: center;">○</div> MSP430 Fxxxx TI NNN # NNNNG4	# = Die revision ○ = Pin 1 location N = Lot trace code
<div style="text-align: center;">○</div> M430 Fxxxx TI NNN # NNNNG4	# = Die revision ○ = Pin 1 location N = Lot trace code
<div style="text-align: center;">○</div> MSP430™ Fxxxx TI NNN # NNNN G4	# = Die revision ○ = Pin 1 location N = Lot trace code

NOTE: Package marking with "TM" applies only to devices released after 2011.

## 6 Memory-Mapped Hardware Revision (TLV Structure)

Die Revision	TLV Hardware Revision
Rev C	12h

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

<b>BSL7</b>	<b><i>BSL Module</i></b>
<b>Category</b>	Software in ROM
<b>Function</b>	BSL does not start after waking up from LPMx.5
<b>Description</b>	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.
<b>Workaround</b>	<ol style="list-style-type: none"> <li>1. Upgrade the device BSL to the latest version (see Creating a Custom Flash-Based Bootstrap Loader (BSL) Application Note - SLAA450 for more details)</li> </ol> <p>OR</p> <ol style="list-style-type: none"> <li>2. Do not use LOCKLPM5 bit (LPMx.5) if the BSL is used but cannot be upgraded.</li> </ol>
<b>COMP10</b>	<b><i>COMP_B Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	Comparator port output toggles when entering or leaving LPM3/LPM4
<b>Description</b>	<p>The comparator port pin output (CECTL1.CEOUT) erroneously toggles when device enters or leaves LPM3/LPM4 modes under the following conditions:</p> <ol style="list-style-type: none"> <li>1) Comparator is disabled (CECTL1.CEON = 0)</li> </ol> <p>AND</p> <ol style="list-style-type: none"> <li>2) Output polarity is enabled (CECTL1.CEOUTPOL = 1)</li> </ol> <p>AND</p> <ol style="list-style-type: none"> <li>3) The port pin is configured to have CEOUT functionality.</li> </ol> <p>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.</p>
<b>Workaround</b>	<p>When the comparator is disabled, ensure at least one of the following:</p> <ol style="list-style-type: none"> <li>1) Output inversion is disabled (CECTL.CEOUTPOL = 0)</li> </ol> <p>OR</p> <ol style="list-style-type: none"> <li>2) Change pin configuration from CEOUT to GPIO with output low.</li> </ol>
<b>CPU21</b>	<b><i>CPUXv2 Module</i></b>
<b>Category</b>	Compiler-Fixed
<b>Function</b>	Using POPM instruction on Status register may result in device hang up
<b>Description</b>	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.
<b>Workaround</b>	<p>None. It is recommended not to use POPM instruction on the Status Register.</p> <p>Refer to the table below for compiler-specific fix implementation information.</p>

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	

## 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

<b>Function</b>	DMA request may cause the loss of interrupts
<b>Description</b>	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.
<b>Workaround</b>	<ol style="list-style-type: none"> <li>1. Use a read of Interrupt Vector registers to clear interrupt flags and do not use read-modify-write instruction.</li> </ol> <p>OR</p> <ol style="list-style-type: none"> <li>2. Disable all DMA channels during read-modify-write instruction of specific module registers containing interrupts flags while these interrupts are activated.</li> </ol>

---

**DMA10** ***DMA Module***


---

<b>Category</b>	Functional
<b>Function</b>	DMA access may cause invalid module operation
<b>Description</b>	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.
<b>Workaround</b>	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.

---

**EEM17** ***EEM Module***


---

<b>Category</b>	Debug
<b>Function</b>	Wrong Breakpoint halt after executing Flash Erase/Write instructions
<b>Description</b>	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.
<b>Workaround</b>	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***


---

<b>Category</b>	Debug
<b>Function</b>	DMA may corrupt data in debug mode
<b>Description</b>	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.

---

## **PMAP1**

### ***PMAP 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.



**PMM9** *PMM Module*
**Category** Functional

**Function** False SVSxIFG events

**Description** The comparators of the SVS require a certain amount of time to stabilize and output a correct result once re-enabled; this time is different for the Full Performance versus the Normal mode. The time to stabilize the SVS comparators is intended to be accounted for by a built-in event-masking delay of 2 us when Full Performance mode is enabled.

However, the comparators of the SVS in Full Performance mode take longer than 2 us to stabilize so the possibility exists that a false positive will be triggered on the SVSH or SVSL. This results in the SVSxIFG flags being set and depending on the configuration of SVSxPE bit a POR can also be triggered.

Additionally when the SVSxIFGs are set, all GPIOs are tri-stated i.e. floating until the SVSx comparators are settled.

The SVS IFG's are falsely set under the following conditions:

1. Wakeup from LPM2/3/4 when SVSxMD = 0 (default setting) && SVSxFP=1. The SVSx comparators are disabled automatically in LPM2/3/4 and are then re-enabled on return to active mode.
2. SVSx is turned on in full performance mode (SVSxFP=1).
3. A PUC/POR occurs after SVSx is disabled. After a PUC or POR the SVSx are enabled automatically but the settling delay does not get triggered. Based on SVSxPE bit this may lead to POR events until the SVS comparator is fully settled.

**Workaround** For each of the above listed conditions the following workarounds apply:

1. If the Full Performance mode is to be enabled for either the high- or low-side SVS comparators, the respective SVSxMD bits must be set (SVSxMD = 1) such that the SVS comparators are not temporarily shut off in LPM2/3/4. Note that this is equivalent to a 2 uA (typical) adder to the low power mode current, per the device-specific datasheet, for each SVSx that remains enabled.
2. The SVSx must be turned on in normal mode (SVSxFP=0). It can be reconfigured to use full performance mode once the SVSx/SVMx delay has expired.
3. Ensure that SVSH and SVSL are always enabled.

**PMM11** *PMM Module*
**Category** Functional

**Function** MCLK comes up fast on exit from LPM3 and LPM4

**Description** The DCO exceeds the programmed frequency of operation on exit from LPM3 and LPM4 for up to 6 us. This behavior is masked from affecting code execution by default: SVSL and SVMLE run in normal-performance mode and mask CPU execution for 150 us on wakeup from LPM3 and LPM4. However, when the low-side SVS and the SVM are disabled or are operating in full-performance mode (SVMLE = 0 and SVSLE = 0, or SVMLE = 1 and SVSLE = 1) AND MCLK is sourced from the internal DCO running over 4 MHz, 7 MHz, 11 MHz, or 14 MHz at core voltage levels 0, 1, 2, and 3, respectively, the mask lasts only 2 us. MCLK is, therefore, susceptible to run out of spec for 4 us.

**Workaround** Set the MCLK divide bits in the Unified Clock System Control 5 Register (UCSCTL5) to divide MCLK by two prior to entering LPM3 or LPM4 (set DIVMx = 001). This prevents

MCLK from running out of spec when the CPU wakes from the low-power mode. Following the wakeup from the low-power mode, wait 32, 48, 80, or 100 cycles for core voltage levels 0, 1, 2, and 3, respectively, before resetting DIVMx to zero and running MCLK at full speed [for example, `__delay_cycles(100)`].

**PMM12**
***PMM Module***


---

**Category**

Functional

**Function**

SMCLK comes up fast on exit from LPM3 and LPM4

**Description**

The DCO exceeds the programmed frequency of operation on exit from LPM3 and LPM4 for up to 6 us. When SMCLK is sourced by the DCO, it is not masked on exit from LPM3 or LPM4. Therefore, SMCLK exceeds the programmed frequency of operation on exit from LPM3 and LPM4 for up to 6 us. The increased frequency has the potential to change the expected timing behavior of peripherals that select SMCLK as the clock source.

**Workaround**

- Use XT2 as the SMCLK oscillator source instead of the DCO.

or

- Do not disable the clock request bit for SMCLKREQEN in the Unified Clock System Control 8 Register (UCSCTL8). This means that all modules that depend on SMCLK to operate successfully should be halted or disabled before entering LPM3 or LPM4. If the increased frequency prevents the proper function of an affected module, wait 32, 48, 80, or 100 cycles for core voltage levels 0, 1, 2, or 3, respectively, before re-enabling the module [for example, `__delay_cycles(100)`].

**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 <sub>WAKE-UP FAST</sub>
	1	0	0	Normal	OFF	OFF	t <sub>WAKE-UP SLOW</sub>
	1	0	1	Full Performance	OFF	OFF	t <sub>WAKE-UP FAST</sub>
	1	1	0	Normal	Normal	OFF	t <sub>WAKE-UP SLOW</sub>
	1	1	1	Full Performance	Full Performance	Normal	t <sub>WAKE-UP FAST</sub>
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 <sub>WAKE-UP FAST</sub>
	1	0		Normal	Normal	OFF	t <sub>WAKE-UP SLOW</sub>
	1	1		Full Performance	Full Performance	Normal	t <sub>WAKE-UP FAST</sub>

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 VCORE 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 Vcore does not have an impact on specified device performance.

---

**Workaround** If SVSL or SVML is required for application (to observe external disruptive events at Vcore 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).

## **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.

<b>Workaround</b>	None
<b>PORT33</b>	<b><i>PORT Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	P4.6 is not set to input floating if device is in RESET state
<b>Description</b>	P4.6 is driven active low while the MSP430 device is held in a reset state. This can influence components connected externally that are expecting the pin to act as a floating input during MSP430 reset.
<b>Workaround</b>	Use any other GPIO if input floating state is required while the device is in a reset state.
<b>RTC3</b>	<b><i>RTC Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	Unreliable write to RTC register
<b>Description</b>	A write access to the RTC registers (SEC, MIN, HOUR, DATE, MON, YEAR, DOW) may result in unexpected results. As a consequence the addressed register might not contain the written data, or some data can be accidentally written to other RTC registers.
<b>Workaround</b>	Use the RTC library routines, available as F541x/F543x code examples on the MSP430 Code Examples page ( <a href="http://www.ti.com/msp430">www.ti.com/msp430</a> > Software > Code Examples), which use carefully aligned MOV instructions. Library is listed as RTC_Workaround.zip and includes both CCE and IAR example projects that show proper usage. Using this library, full access to RTC registers is possible.
<b>RTC6</b>	<b><i>RTC Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	the step size of the RTC frequency adjustment is twice the specified size.
<b>Description</b>	In BCD mode of operation, the step size of the RTC frequency adjustment is =+8ppm/-4ppm. This is twice the size specified in the User's Guide.  In BCD mode, for up calibration this results in a step size per step of 8ppm (1024 cycles) instead of 4ppm (512 cycles). For down calibration this results in a step size per step of 4ppm (512 cycles) instead of 2ppm (256 cycles).  In Binary mode, the step size = +4ppm/-2ppm as per the spec.
<b>Workaround</b>	In BCD mode of operation, half the calibration value could be written into RTCCAL register to compensate the doubled step size.
<b>SYS12</b>	<b><i>SYS Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	Invalid ACCVIFG when DVcc in the range of 2.4 to 2.6V
<b>Description</b>	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.

---

## **UCS7** *UCS Module*

**Category** Functional

**Function** DCO drifts when servicing short ISRs when in LPM0 or exiting active from ISRs for short periods of time

**Description** The FLL uses two rising edges of the reference clock to compare against the DCO frequency and decide on the required modifications to the DCOx and MODx bits. If the device is in a low power mode with FLL disabled (LPM0 with DCO not sourcing ACLK/SMCLK or LPM2, LPM3, LPM4 where SCG1 bit is set) and enters a state which enables FLL (enter ISR from LPM0/LPM2 or exit active from ISRs) for a period less than 3x reference clock cycles, then the FLL will cause the DCO to drift.

This occurs because the FLL immediately begins comparing an active DCO with its reference clock and making the respective modifications to the DCOx and MODx bits. If the FLL is not given sufficient time to capture a full reference clock cycle (2 x reference clock periods) and adjust accordingly (1 x reference clock period), then the DCO will keep drifting each time the FLL is enabled.

**Workaround** (1) If DCO is not sourcing ACLK or SMCLK in the application, use LPM1 instead of LPM0 to make sure FLL is disabled when interrupt service routine is serviced.

(2) When exiting active from ISRs, insert a delay of at least 3 x reference clock periods. To save on power budget, the 3 x reference clock periods could also be spent in LPM0 with TimerA or TimerB using ACLK/SMCLK sourced from DCO. This way, the FLL and DCO are still active in LPM0.

---

## **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.



<b>Workaround</b>	<p>Before entering LPM4:</p> <p>(1) Switch to a clock source other than external bypass digital input.</p> <p>OR</p> <p>(2) Turn off external bypass mode (UCSCTL6.XT1BYPASS = 0).</p>
<hr/>	
<b>UCS11</b>	<b><i>UCS Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	Modifying UCSCTL4 clock control register triggers an additional erroneous clock request
<b>Description</b>	<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><b>NOTE:</b> The XT1/XT2 fault flag can be set regardless of which SELM/SELS/SELA bit combinations are being changed.</p> <hr/>
<b>Workaround</b>	<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>
<hr/>	
<b>USCI26</b>	<b><i>USCI Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	Tbuf parameter violation in I2C multi-master mode
<b>Description</b>	<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>
<b>Workaround</b>	None
<hr/>	
<b>USCI34</b>	<b><i>USCI Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	I2C multi-master transmit may lose first few bytes.
<b>Description</b>	<p>In an I2C multi-master system (UCMM =1), under the following conditions:</p> <p>(1)the master is configured as a transmitter (UCTR =1)</p>



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:

```
UCAxTXBUF = UCAxTXBUF;
```

## 8 Document Revision History

Changes from device specific erratasheet to document Revision A.

1. EEM19 Workaround was updated.
2. EEM17 Workaround was updated.
3. EEM23 Workaround was updated.
4. EEM17 Description was updated.
5. EEM23 Description was updated.
6. EEM19 Description was updated.

Changes from document Revision A to Revision B.

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

Changes from document Revision B to Revision C.

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. EEM23 Function was updated.
8. EEM19 Description was updated.

Changes from document Revision C to Revision D.

1. Errata USCI40 was added to the errata documentation.
2. Errata DMA7 was added to the errata documentation.
3. PMM18 Workaround was updated.

Changes from document Revision D to Revision E.

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

Changes from document Revision E to Revision F.

1. USCI39 Description was updated.

Changes from document Revision F to Revision G.

1. EEM19 Workaround was updated.

Changes from document Revision G to Revision H.

1. RTC6 Description was updated.
2. UCS11 Workaround was updated.
3. UCS11 Description was updated.
4. UCS11 Function was updated.

Changes from document Revision H to Revision I.

1. Errata SYS12 was removed from the errata documentation.
2. Errata COMP10 was added to the errata documentation.

Changes from document Revision I to Revision J.

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

Changes from document Revision J to Revision K.

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

Changes from document Revision K to Revision L.

1. TLV Hardware Revision section was added to the documentation.

Changes from document Revision L to Revision M.

1. Workaround for PMM15 was updated.

Changes from document Revision M to Revision N.

1. PORT33 was added to the errata documentation.

Changes from document Revision N to Revision O.

1. Description for RTC6 was updated.
2. Workaround for RTC6 was updated.

Changes from document Revision O to Revision P.

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

Changes from document Revision P to Revision Q.

1. Workaround for CPU40 was updated.

Changes from document Revision Q to Revision R.

1. CPU47 was added to the errata documentation.

Changes from document Revision R to Revision S.

1. USCI34 was added to the 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](http://www.ti.com/legal/termsofsale.html)) or other applicable terms available either on [ti.com](http://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 © 2019, Texas Instruments Incorporated