

## MSP430F5632 Device Erratasheet

### 1 Revision History

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

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

Errata Number	Rev D
<a href="#">BSL6</a>	✓
<a href="#">BSL7</a>	✓
<a href="#">BSL12</a>	✓
<a href="#">CPU37</a>	✓
<a href="#">CPU40</a>	✓
<a href="#">CPU43</a>	✓
<a href="#">DMA4</a>	✓
<a href="#">DMA10</a>	✓
<a href="#">EEM11</a>	✓
<a href="#">EEM16</a>	✓
<a href="#">EEM17</a>	✓
<a href="#">EEM19</a>	✓
<a href="#">EEM21</a>	✓
<a href="#">EEM23</a>	✓
<a href="#">JTAG20</a>	✓
<a href="#">MPY1</a>	✓
<a href="#">PMAP1</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="#">PORT17</a>	✓
<a href="#">PORT19</a>	✓
<a href="#">SYS16</a>	✓
<a href="#">SYS18</a>	✓
<a href="#">TAB23</a>	✓
<a href="#">UCS9</a>	✓
<a href="#">UCS11</a>	✓
<a href="#">USB9</a>	✓
<a href="#">USB10</a>	✓
<a href="#">USCI26</a>	✓

Errata Number	Rev D
<a href="#">USCI31</a>	✓
<a href="#">USCI35</a>	✓

## 2 Package Markings

### PZ100

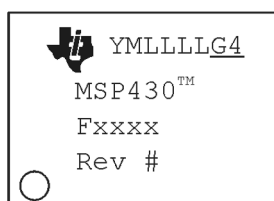
#### LQFP (PZ) 100 Pin



YM = Year and Month Date Code  
 LLLL = LOT Trace Code  
 S = Assembly Site Code  
 # = DIE Revision  
 o = PIN 1



YM = Year and Month Date Code  
 LLLL = LOT Trace Code  
 S = Assembly Site Code  
 # = DIE Revision  
 o = PIN 1

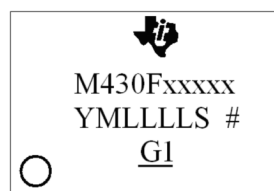


YM = Year and Month Date Code  
 LLLL = LOT Trace Code  
 S = Assembly Site Code  
 # = DIE Revision  
 o = PIN 1

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

### ZQW113

#### BGA (ZQW), 113 Pin



YM = Year and Month Date Code  
 LLLL = LOT Trace Code  
 S = Assembly Site Code  
 # = DIE Revision  
 o = PIN 1

## 3 TLV Hardware Revision

Die Revision	TLV Hardware Revision
Rev D	22h

Further guidance on how to locate the TLV structure and read out the HW\_ID can be found in the device User's Guide.

## 4 Detailed Bug Description

### BSL6

#### **BSL Module**

#### Function

USB BSL does not respond properly to suspend/reset events from the USB host

#### Description

The USB BSL in affected revisions contains an improper configuration of the USB module. As a result, errors might occur in response to suspend/reset events from the USB host. (Since enumeration of the USB device often involves suspend and/or reset events, an enumeration might trigger the failure.) If the failure occurs, the device becomes unresponsive to the USB host.

If the failure occurs, and if application code exists in main flash, a reset (BOR/POR/PUC) can be issued to switch execution away from the BSL, to the application. Given the same USB host/setup circumstances, the problem is likely to occur again on subsequent attempts.

Applications that do not use the USB BSL are unaffected.

#### Workaround

1. The BSL can be updated via JTAG with a version that does not contain this bug. Use the code published in [www.ti.com/lit/pdf/slaa450](http://www.ti.com/lit/pdf/slaa450) BSL documentation starting with version 00.07.85.36.

### BSL7

#### **BSL Module**

#### 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.

### BSL12

#### **BSL Module**

#### Function

BSL invoke

#### Description

When externally invoking BSL according SLAU319 chapter 1.3.3. a critical setup time may not be met. In this case the BSL will not start. The pass/fail condition is temperature-dependent, where if a unit passes at a certain temperature, it will always pass at the same or higher temperature condition.

#### Workaround

1. Invoke the BSL from the application code and ensure VCore is set to level 2 or 3 prior to BSL entry.

OR

2. Update the device BSL. The CustomBSL source code implements the fix for this errata in versions 1.00.05.00 and newer. The CustomBSL package can be download at [Custom BSL package](#)

### CPU37

#### **CPUXv2 Module**

#### Function

Wrong program trace display in the debugger while using conditional jump instructions

#### Description

The state storage window displays an incorrect sequence of instructions when:

1. Conditional jump instructions are used to form a software loop

AND

2. A false condition on the jump breaks out of the loop

In such cases the trace buffer incorrectly displays the first instruction of the loop as the instruction that is executed immediately after exiting the loop.

Example:

Actual Code:

```
mov #4,R4
```

```
LABEL mov #1,R5
```

```
dec R4
```

```
jnz LABEL
```

```
mov #2,R6
```

```
nop
```

State Storage Window Displays:

```
LABEL mov #1,R5
```

```
dec R4
```

```
jnz LABEL
```

```
mov #1,R5
```

```
nop
```

#### Workaround

None

Note: This erratum affects the trace buffer display only. It does not affect code execution in debugger or free run mode

## CPU40

### CPUXv2 Module

#### 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.

**CPU43**
**CPUXv2 Module**
**Function**

Halt operation in debug mode may cause unintended behavior

**Description**

In certain cases when using the 'Halt CPU' function available via the IDE (CCS or IAR), on continuing code execution after a halt, the program counter may skip an instruction.

Pausing and resuming code execution after a breakpoint works as expected and is not affected by the erratum.

---

**NOTE:** This erratum only affects debug mode.

---

**Workaround**

None.

**DMA4**
**DMA Module**
**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).

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

DMA interrupting CPU wait state might cause peripheral module into unknown state.

**Description**

When the CPU accesses a module that is capable of stalling the CPU with a wait mechanism, if a DMA interrupts the instruction during the CPU stall, the module might be caused into an unknown state.

The affected modules (if present on the device) that can stall CPU are: FRAM controller in manual timing mode, MPY, CRC, USB, and RF1A.

As an example a wrong result can be read by DMA from MPY result register because the DMA does not wait until MPY operation is finished.

**Workaround**

Disable DMA when using affected modules.

**EEM11**
**EEM Module**
**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.

---

## **EEM16** *EEM Module*

---

**Function** The state storage display does not work reliably when used on instructions with CPU Wait cycles.

**Description** When executing instructions that require wait states; the state storage window updates incorrectly. For example a flash erase instruction causes the CPU to be held until the erase is completed i.e. the flash puts the CPU in a wait state. During this time if the state storage window is enabled it may incorrectly display any previously executed instruction multiple times.

**Workaround** Do not enable the state storage display when executing instructions that require wait states. Instead set a breakpoint after the instruction is completed to view the state storage display.

---

**NOTE:** This erratum affects debug mode only.

---

## **EEM17** *EEM Module*

---

**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*

---

**Function** DMA may corrupt data in debug mode

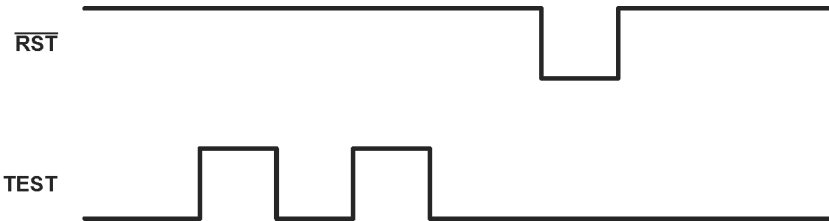
**Description** When the DMA is enabled and the device is in debug mode, the data transferred by the DMA may be corrupted when a breakpoint is hit or when the debug session is halted.

**Workaround** None. Do not set a breakpoint during a DMA transfer.

---

**NOTE:** This erratum applies to debug mode only.

---

<b>EEM21</b>	<b><i>EEM Module</i></b>
<b>Function</b>	LPMx.5 debug limitations
<b>Description</b>	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.
<b>Workaround</b>	Follow the debugging steps in Debugging MSP430 LPM4.5 <a href="#">SLAA424</a> .
<b>EEM23</b>	<b><i>EEM Module</i></b>
<b>Function</b>	EEM functions do not work reliably when modules using wait cycles are enabled
<b>Description</b>	When modules using wait states (USB, MPY,CRC and FRAM controller in manual mode) are enabled the EEM may not perform profile counter and state storage functions reliably.
<b>Workaround</b>	Do not enable profile counter and state storage functions when modules using wait states are enabled.
	<hr/> <b>NOTE:</b> This erratum affects debug mode only. <hr/>
<b>JTAG20</b>	<b><i>JTAG Module</i></b>
<b>Function</b>	BSL does not exit to application code
<b>Description</b>	The methods used to exit the BSL per MSP430 Programming Via the Bootstrap Loader ( <a href="#">SLAU319</a> ) are invalid.
<b>Workaround</b>	<p>To exit the BSL one of the following methods must be used.</p> <ul style="list-style-type: none"> <li>- A Power cycle</li> <li>or</li> <li>- Toggle the TEST pin twice when nRST is high and then pull nRST low.</li> </ul>
	 <p>The diagram shows two digital signals: RST and TEST. RST is a high-level signal that transitions from high to low and then back to high. TEST is a low-level signal that transitions from low to high and back to low twice while RST is high.</p>
	Note: This sequence is not subject to timing constraints and the appropriate level transitions are sufficient to trigger an exit from BSL mode.
<b>MPY1</b>	<b><i>MPY Module</i></b>
<b>Function</b>	Save and Restore feature on MPY32 not functional
<b>Description</b>	The MPY32 module uses the Save and Restore method which involves saving the multiplier state by pushing the MPY configuration/operand values to the stack before using the multiplier inside an Interrupt Service Routine (ISR) and then restoring the state



by popping the configuration/operand values back to the MPY registers at the end of the ISR. However due to the erratum the Save and Restore operation fails causing the write operation to the OP2H register right after the restore operation to be ignored as it is not preceded by a write to OP2L register resulting in an invalid multiply operation.

**Workaround**

None. Disable interrupts when writing to OP2L and OP2H registers.

Note: When using the C-compiler, the interrupts are automatically disabled while using the MPY32

**PMP1**
**PMP Module**
**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, PMPRECFG, to select inputs to a module and map only one input at a time.

**PMP11**
**PMP Module**
**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 5 MHz, 7.5 MHz, 10 MHz, or 12.5 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, 64, or 80 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(32)].

**PMP12**
**PMP Module**
**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, 64, or 80 cycles for core voltage levels 0, 1, 2, and 3, respectively, before re-enabling the module [for example, `__delay_cycles(32)`].

**PMM14**
**PMM Module**
**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**
**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 LPM2/3/4 SVSL State	Automatic SVSMLACE = 1 LPM2/3/4 SVSL State	Wakeup Time LPM2/3/4
	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 LPM2/3/4 SVML State	Automatic SVSMLACE = 1 LPM2/3/4 SVML State	Wakeup Time LPM2/3/4
	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 LPM2/3/4 SVSH State	Manual SVSMHACE = 1 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	SVSHE	SVMHFP		AM, LPM0/1 SVSH state	Manual SVSMHACE = 0 LPM2/3/4 SVSH State	Manual SVSMHACE = 1 LPM2/3/4 SVSH 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. Instead force the modules to remain ON even in LPMx. Note that this will cause increased power consumption when in LPMx.

Refer to the MSP430F5xx and MSP430F6xx Core Libraries ([SLAA448](#)) 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**

#### **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 Adaptive mode (SVMACE=1). The SVM high side is inactive in LPM234.

## PMM20

### **PMM Module**

#### **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 SVMML 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 SVMML 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 SVMML is configured in "Normal" performance mode (not disabled and not in "Full" Performance Mode).

### PORT15

#### PORT Module

#### 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.

### PORT17

#### PORT Module

#### Function

Certain pins when subject to negative high current pulses may cause latch-up in adjacent pins.

#### Description

Pins subject to negative high current pulses may cause latch-up in adjacent pins. The latch-up condition exists only if the adjacent pin configurations also referred to as 'affected-pin' configuration are one of the following:

- (1) GPIO input driven high by an external source
- (2) GPIO output driven high with Full Drive strength OR Reduced Drive strength settings
- (3) Peripheral configuration where the peripheral drives pin high or causes pin to be driven high externally

The following affected-pin configurations will not sustain latch-up:

- (1) GPIO input driven low
- (2) GPIO output driven low
- (3) Peripheral configuration where the peripheral drives pin low or causes pin to be driven low externally
- (4) Peripheral configuration as LCD pin

Note that for affected-pin configurations with LCD functionality, the window of latch-up when the pin is driven being high still exists but is of extremely short duration and hence there is a low probability of latch-up occurrence.

#### Workaround

All affected pins must be driven low when not in use. If the affected pins are not driven low, then connecting a series resistor of 330 ohms to limit the latch-up current is recommended.

For more details on trigger currents, affected pin configurations and workarounds refer to the document [PORT17 Guidance SLAA562](#)

## PORT19

### **PORT Module**

#### 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

## SYS16

### **SYS Module**

#### 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.

## SYS18

### **SYS Module**

#### Function

USB registers are unlocked and ACCVIFG is set at start-up

#### Description

During device start-up, an incorrect line of code in the start-up code causes the USB registers to remain unlocked and causes an access violation, setting ACCVIFG bit.

In the BSL430\_Low\_Level\_Init code, the following line of code accesses USBKEY (incorrect register address) instead of USBKEYPID, causing an access violation setting ACCVIFG bit, and leaving the USB registers unlocked.

```
mov.w #0x0000, &USBKEY ; lock USB
```

The correct line of code should read:

```
mov.w #0x0000, &USBKEYPID ; lock USB correctly
```

Note: This code does not run when using the JTAG debugger - the behavior only appears when running standalone.

#### Workaround

1. Load the latest version of the USB BSL from [Custom BSL Download](#)
- OR
2. Load a non-USB or custom BSL
- OR
3. Erase the BSL

## TAB23

### **TIMER\_A/TIMER\_B Module**

#### Function

TAxR/TBxR read can be corrupted when TAxR/TBxR = TAxCCR0/TBxCCR0

#### Description

When a timer in Up mode is stopped and the counter register (TAxR/TBxR) is equal to the TAxCCR0/TBxCCR0 value, a read of the TAR/TBR register may return an

unexpected result.

**Workaround**

1. Use 'Up/Down' mode instead of 'Up' mode  
OR
2. In 'Up' mode, use the timer interrupt instead of halting the counter and reading out the value in TAxR/TBxR  
OR
3. When halting the timer counter in 'Up' mode, reinitialize the timer before starting to run again.

**UCS9**
**UCS Module**
**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**
**UCS Module**
**Function**

Modifying UCSCTL4 clock control register triggers an erroneous clock source request

**Description**

Changing the SELM/SELS/SELA bits in the UCSCTL4 register might trigger the respective clocks to select an incorrect clock source which requests the XT1/XT2 clock. If the crystals are not present at XT1/XT2 or present but not yet configured in the application firmware, then the respective XT1/XT2 fault flag is falsely set.

**Workaround**

Clear all the fault flags in UCSCTL7 register once after changing any of the SELM/SELS/SELA bits in the UCSCTL4 register.

**USB9**
**USB Module**
**Function**

VBUS detection may fail after powerup

**Description**

In rare cases, some USB-equipped MSP430 devices may experience a failure in the bandgap that aids in detecting the presence of 5 V on the VBUS pin. Two primary effects of this are:

- The USBBGVBV bit fails to show the presence of a valid voltage on the VBUS pin.
- and
- The USB LDOs fail to start.

**Workaround**

This error state can be "reset" by clearing all of the bits in the USBPWRCTL register, which disables the USB LDOs, among other actions. The bits can then be set again normally, and the device functions properly.

This has been added to the USB\_Init() function in v3.10 and later of the MSP430 USB



API. Therefore, this problem is automatically addressed in applications that use the API.

However, if the integrated 3.3-V USB LDO (the output of the VUSB pin) is used to power the device's DVCC pin, as in many bus-powered applications, and if the rare bandgap error occurs, the CPU fails to power up, because the USB LDO fails to operate. The problem might be resolved by cycling power to the VBUS pin; for example, if the end user responds to the failure by unplugging and replugging the USB cable. The bandgap failure is also known to occur more often with slow DVCC ramps > 200 ms; for example, when there is excessive capacitance on the DVCC pin, in excess of what the USB specification allows. However, the only sure way to prevent the problem from occurring in the first place is to avoid making DVCC power reliant on VUSB.

## USB10

### USB Module

#### Function

USB interface may begin to babble when a rare timing event occurs between the USB host and MSP430 software execution

#### Description

When the host sends a SETUP packet for an IN transaction, the SETUPIFG bit always gets set by hardware, and the USB ISR is triggered. While SETUPIFG is high, the host's attempts to continue the transaction with IN packets are automatically NAKed.

When the SETUP packet has been decoded and the IN data prepared, the USB ISR clears the SETUPIFG bit. But if it happens to do so within the 2nd CRC bit of an IN packet from the host, the USB module enters an errant state and can begin to "babble" (endless transmission to the host, irrespective of the protocol). The errant state can be cleared by resetting the module with the USB\_EN bit; but there's no way for software to reliably detect the condition.

Since the 2nd CRC bit is only an 83ns window, the problem is extremely rare. However, since the timing of IN packets relative to their preceding SETUP packets can vary according to the host's timing, there's no way to ensure for certain that it will never happen.

#### Workaround

If the problem behavior occurs, and if the MSP430 is bus-powered, the user may naturally unplug/re-plug the device's USB connection. If this occurs, the behavior will be corrected because power to the MSP430 will be cycled. After this, it's unlikely the problem will occur again soon, since the failure is usually rare.

The behavior can be prevented altogether by clearing the UBME bit immediately before clearing SETUPIFG, and setting it again immediately after:

```

        USBIEPCNF_0 &= ~EPCNF_UBME; // Clear ME to gate off SETUPIFG
clear event
        USBOEPCNF_0 &= ~EPCNF_UBME; // Clear ME to gate off SETUPIFG
clear event
        USBIFG &= ~SETUPIFG; // clear the interrupt bit
        USBIEPCNF_0 |= EPCNF_UBME; // Set ME to continue with normal
operation
        USBOEPCNF_0 |= EPCNF_UBME; // Set ME to continue with normal
operation

```

This workaround is reliable and effective. However, as a side effect, it results in the creation of orphan tokens on the USB interface. Although the workaround is field-tested, and no problems have been reported with these orphan packets, it is recommended to use the workaround only if the errata behavior is problematic for the application in question.

## USCI26

### USCI Module

#### Function

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
<b>USCI31</b>	<b>USCI Module</b>
<b>Function</b>	Framing Error after USCI SW Reset (UCSWRST)
<b>Description</b>	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.
<b>Workaround</b>	<ol style="list-style-type: none"> <li>1. If possible, do not reset USCI-UART during an ongoing receive operation; that is, when UCBUSY bit is set.</li> <li>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.</li> </ol> <p>Workaround code sequence:</p> <pre>bis #UCSWRST, &amp;UCAxCTL1 ; USCI SW reset ;Workaround begins bis #UCSYNC, &amp;UCAxCTL0 ; set synchronous mode bic #UCSYNC, &amp;UCAxCTL0 ; reset synchronous mode ;Workaround ends bic #UCSWRST, &amp;UCAxCTL1 ; release USCI reset</pre>
<b>USCI35</b>	<b>USCI Module</b>
<b>Function</b>	Violation of setup and hold times for (repeated) start in I2C master mode
<b>Description</b>	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.
<b>Workaround</b>	If using repeated start, ensure SCL clock frequencies is < 50kHz in I2C standard mode (100 kbps).

## 5 Document Revision History

Changes from family erratasheet to device specific erratasheet.

1. Errata JTAG21 was removed
2. Revision A was removed
3. Revision B was removed
4. Revision C was removed
5. Errata FLASH38 was removed from Revision D
6. PZ100 package markings have been updated

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.
3. Errata SYS18 was added to the errata documentation.
4. Errata PORT17 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. Errata BSL6 was added to the errata documentation.
2. DMA10 Description was updated.
3. DMA10 Function was updated.

Changes from document Revision C to Revision D.

1. DMA10 Description was updated.
2. BSL6 Workaround was updated.
3. MPY1 Description was updated.
4. Errata EEM23 was added to the errata documentation.
5. Errata CPU43 was added to the errata documentation.

Changes from document Revision D to Revision E.

1. SYS16 Description was updated.
2. CPU43 Description was updated.
3. 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.
3. Errata USB10 was added to the errata documentation.

Changes from document Revision G to Revision H.

1. EEM19 Workaround was updated.
2. EEM17 Workaround was updated.
3. Errata BSL12 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. EEM16 Description was updated.
9. PORT17 Workaround was updated.
10. EEM23 Description was updated.
11. EEM19 Description was updated.
12. EEM16 Workaround was updated.

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

### Products

Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>
OMAP Applications Processors	<a href="http://www.ti.com/omap">www.ti.com/omap</a>
Wireless Connectivity	<a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a>

### Applications

Automotive and Transportation	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Communications and Telecom	<a href="http://www.ti.com/communications">www.ti.com/communications</a>
Computers and Peripherals	<a href="http://www.ti.com/computers">www.ti.com/computers</a>
Consumer Electronics	<a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a>
Energy and Lighting	<a href="http://www.ti.com/energy">www.ti.com/energy</a>
Industrial	<a href="http://www.ti.com/industrial">www.ti.com/industrial</a>
Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Space, Avionics and Defense	<a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a>
Video and Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>

### TI E2E Community

[e2e.ti.com](http://e2e.ti.com)