

MSP430F6726 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

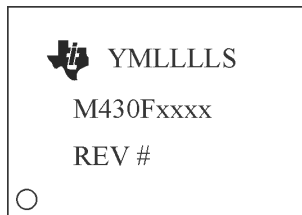
Errata Number	Rev A
ADC39	✓
ADC42	✓
AUXPMM1	✓
AUXPMM2	✓
BSL7	✓
BSL14	✓
CPU21	✓
CPU22	✓
CPU36	✓
CPU40	✓
CPU46	✓
DMA4	✓
DMA7	✓
DMA9	✓
DMA10	✓
EEM8	✓
EEM17	✓
EEM19	✓
EEM23	✓
JTAG26	✓
JTAG27	✓
LCDB5	✓
LCDB6	✓
PMM7	✓
PMM11	✓
PMM12	✓
PMM14	✓
PMM15	✓
PMM18	✓
PMM20	✓
PMM26	✓
PORT15	✓
PORT19	✓
SD3	✓

Errata Number	Rev A
UCS11	✓
USCI36	✓
USCI37	✓
USCI41	✓
USCI42	✓

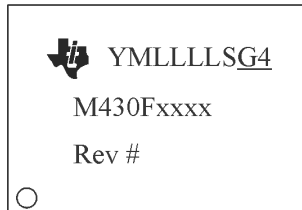
2 Package Markings

PN80

LQFP (PN), 80 Pin



YM = Year and Month Date Code
 LLLL = Assembly Lot Code
 S = Assembly Site Code
 # = Die Revision
 ○ = Pin 1



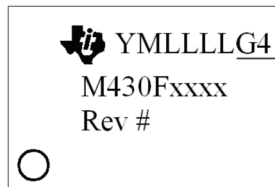
YM = Year and Month Date Code
 LLLL = Assembly Lot Code
 S = Assembly Site Code
 # = Die Revision
 ○ = Pin 1

PZ100

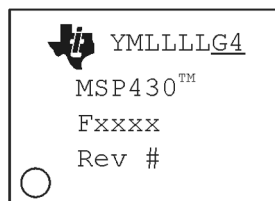
LQFP (PZ) 100 Pin



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



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



YM = Year and Month Date Code
 LLLL = Assembly Lot Code
 S = Assembly Site Code
 # = DIE Revision
 ○ = PIN 1

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

3 Memory-Mapped Hardware Revision (TLV Structure)

Die Revision	TLV Hardware Revision
Rev A	10h

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

ADC39

ADC10_A Module

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

- 1) Use the pulse sample mode (ADC10SHP=1)
- OR
- 2) Use a synchronous clock for ADC10 and the SHI signal.

ADC42

ADC10_A Module

Function

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

Description

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

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

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

Workaround

1. For manual trigger mode (ADC12CTL0.ADC12SC), ensure each ADC conversion is completed by first checking ADC12CTL1.ADC12BUSY bit before starting a new conversion.
 2. For timer trigger mode (ADC12CTL1.ADC12SHP), ensure the timer period is greater than the ADC sample and conversion time.
- To recover the conversion halt:
1. Disable ADC module (ADC12CTL0.ADC12ENC = 0 and ADC12CTL0.ADC12ON = 0)
 2. Re-enable ADC module (ADC12CTL0.ADC12ON = 1 and ADC12CTL0.ADC12ENC = 1)
 3. Re-enable conversion

AUXPMM1

AUXPMM Module

Function

AUXVCC1/AUXVCC2 can not be switched back to DVCC

Description

When the system is running with the AUXVCC1 supply after DVCC/AVCC is lost, if the AUXVCC1 voltage goes lower than SVSH setting for POR and above BORH level, the system can not switch back to DVCC after DVCC ramps back up again.

Similarly, when the system is running with the AUXVCC2 supply after DVCC/AVCC is lost, if the AUXVCC2 voltage goes lower than SVSH setting for POR and above BORH level, the system can not switch back to DVCC after DVCC ramps back up again.

Workaround

When the system is running with the AUXVCC1 supply, use SVMH to monitor AUXVCC1 voltage. When AUXVCC1 is lower than the SVMH setting, the program drives the chip

into LPMx.5. After DVCC ramps up back again, trigger one of the wake up pins. The power supply could be switched back to DVCC again.

When the system is running with the AUXVCC2 supply, use SVMH to monitor AUXVCC2 voltage. When AUXVCC2 is lower than the SVMH setting, the program drives the chip into LPMx.5. After DVCC ramps up back again, trigger one of the wake up pins. The power supply could be switched back to DVCC again.

AUXPMM2

AUXPMM Module

Function

Latch-up in AUXPMM

Description

Latch-up current can appear at the AUXPMM module supply pins in the following two scenarios:

Scenario 1: When the AUXPMM is configured for hardware- or software-controlled switching and the module switches from DVCC to AUXVCC2, latch-up current can appear at AUXVCC2 at the switching point defined by SVSMHCTL.SVSMHRRL (or AUXCTL2.AUX0LVLx). The probability for this event to occur depends on:

- a) Operating temperature (higher temperatures increase probability)
- b) External AUXVCC2 voltage level (higher voltages increase probability)
- c) SVSMHRRL level (lower levels increase probability) defining the switching level in hardware-controlled mode
- d) AUX0LVLx level (lower levels increase probability) defining the switching level in software-controlled mode (applicable to DVCC only)

Scenario 2: When a battery is connected to DVCC, AUXVCC1 or AUXVCC2 as the first voltage supply, due to the low internal resistance of the battery a very fast rise time is seen by the AUXPMM and latch-up current can appear at the connected supply if:

- a) Rise times are in the range of 140 kV/s (faster rise times increase probability)
- b) Device operates at temperatures of 75 deg C and above (higher temperatures increase probability)

The latch-up current disappears after complete power cycles of all supply sources.

Workaround

For scenario 1:

- Increase SVSMRRL to a level above maximum external voltage expected on AUXVCC2. SVSMRRL = 6 or 7 (requires VCORE level of 3) is applicable for AUXVCC2 of up to maximum voltage, 3.58V, while a lower SVSMRRL setting can be selected if a lower voltage (e.g. 3.3V) is expected on AUXVCC2.

Or

- Connect all 3 supplies via 3 external diodes to DVCC and realize the switching externally without using the internal AUXPMM switches. See application report ["Implementation of a Three-Phase Electronic Watt-Hour Meter Using the MSP430F471xx"](#) for details.

Or

- Use AUXVCC1 instead of AUXVCC2 for backup supply

For scenario 2:

Limit the supply voltage ramp up time through a series resistor (e.g. 10 Ohm) in the critical supply path. Side effects such as voltage dips due to high current consumption of the device need to be considered.

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	<ol style="list-style-type: none"> 1. Upgrade the device BSL to the latest version (see Creating a Custom Flash-Based Bootstrap Loader (BSL) Application Note - SLAA450 for more details) <p>OR</p> <ol style="list-style-type: none"> 2. Do not use LOCKLPM5 bit (LPMx.5) if the BSL is used but cannot be upgraded.

BSL14 *BSL Module*

Function	BSL request to unlock the JTAG
Description	The feature in the BSL to keep the JTAG unlocked by setting the bit BSL_REQ_JTAG_OPEN in the return value has been disabled in this device.
Workaround	None

CPU21 *CPUXv2 Module*

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	<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*

Function	Indirect addressing mode with the Program Counter as the source register may produce unexpected results
Description	<p>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.</p> <p>For example in the code below, the ADD instruction does not get executed.</p> <pre>mov @PC, R7 add #1h, R4</pre>
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	

CPU36
CPUXv2 Module
Function

PC corruption when single-stepping through flash erase

Description

When single-stepping over code that initiates an INFOD Flash memory erase, the program counter is corrupted.

Workaround

None.

NOTE: This erratum applies to debug mode only.

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.

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	
MSP430 GNU Compiler (MSP430-GCC)	Not affected	

CPU46 *CPUXv2 Module*

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.
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.

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).

DMA7

DMA Module

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.

DMA9

DMA Module

Function

DMA stops transferring bytes unexpectedly

Description

When the DMA is configured to transfer bytes from the eUSCI_A or eUSCI_B transmit or receive buffers, the transmit or receive triggers (TXIFG and RXIFG) may not be seen by the DMA module and the transfer of the bytes is missed. Once the first byte in a transfer sequence is missed, all the following bytes are missed as well. All eUSCI_A modes (UART, SPI, and IrDA) and all eUSCI_B modes (SPI and I2C) are affected.

Workaround

1) Use Interrupt Service Routines to transfer data to and from the eUSCI_A or eUSCI_B.

OR

2) When using DMA channel 0 for transferring data to and from the eUSCI_A or eUSCI_B, use DMA channel 2 (lower priority than DMA channel 0) to read the same register of the eUSCI_A or eUSCI_B that DMA channel 0 is working with. Use the same USCIFG (e.g. UCA0RXIFG) as trigger source for these both DMA channels.

DMA10

DMA Module

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.

EEM8

EEM Module

Function	Debugger stops responding when using the DMA
Description	<p>In repeated transfer mode, the DMA automatically reloads the size counter (DMAxSZ) once a transfer is complete and immediately continues to execute the next transfer unless the DMA Enable bit (DMAEN) has been previously cleared. In burst-block transfer mode, DMA block transfers are interleaved with CPU activity 80/20% - of ten CPU cycles, eight are allocated to a block transfer and two are allocated for the CPU.</p> <p>Because the JTAG system must wait for the CPU bus to be clear to halt the device, it can only do so when two conditions are met:</p> <ul style="list-style-type: none"> - Three clock cycles after any DMA transfer, the DMA is no longer requesting the bus. <p>and</p> <ul style="list-style-type: none"> - The CPU is not requesting the bus. <p>Therefore, if the DMA is configured to operate in the repeat burst-block transfer mode, and a breakpoint is set between the line of code that triggers the DMA transfers and the line that clears the DMAEN bit, the DMA always requests the bus and the JTAG system never gains control of the device.</p>
Workaround	When operating the DMA in repeat burst-block transfer mode, set breakpoint(s) only when the DMA transfers are not active (before the start or after the end of the DMA transfers).

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 written by the DMA may be corrupted when a breakpoint is hit or when the debug session is halted.
Workaround	<p>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.</p> <p>If using an earlier version of either IDE or MSPDebugStack, do not halt or use breakpoints during a DMA transfer.</p>

NOTE: This erratum applies to debug mode only.

EEM23
EEM Module
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
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
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	<ol style="list-style-type: none"> 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. <p>For MSPDebugStack (MSP430.DLL) in CCS or IAR, download the latest version of the development environment or the latest version of the MSPDebugStack</p> <p>NOTE: This only affects debug mode.</p>
<hr/>	
LCDB5	<i>LCD_C Module</i>
Function	Static DC charge can built up on dedicated COMx pins.
Description	If the device is set into LPMx.5, its dedicated COMx pins (not shared with GPIO function) are floating. External leakage paths to these pins can result in dedicated COMx pins being charged. This can lead to static DC voltages being applied to the external LCD display. This might cause long term over-stress to the LCD display and/or cause certain LCD segments to flare up when device wakes up from LPMx.5 mode.
Workaround	Connect a high-resistance resistor between the dedicated COM pins and Vss to permanently discharge the affected pins.
<hr/>	
LCDB6	<i>LCD_C Module</i>
Function	LCD outputs may be corrupted by modifying register fields VLCDx and/or LCDCPEN of LCDCVCTL register while LCDON (LCDCCTL0) is set
Description	Writing to VLCDx and/or LCDCPEN register bits in LCDCVCTL register while LCDON is enabled (LCDON = '1' in LCDCCTL0 register) may corrupt the LCD output due to incorrect start-up of LCD-controller and internal voltage generation.
Workaround	Do not modify VLCDx and/or LCDCPEN bits in LCDCVCTL register while LCDON = '1'
<hr/>	
PMM7	<i>PMM Module</i>
Function	PMMRIE default conditions different than user guide
Description	The user guide specifies that, after a BOR reset condition, the SVS will not be configured to trigger a POR signal in the condition that the monitored voltages fall below the SVS level(s). This is not true for this device. The SVS Low and SVS High Side POR Enable bits (SVSLPE/SVSHPE) in the Power Management System Reset Enable and Interrupt Enable register are set by default (PMMRIE = 0x1100).
Workaround	If this behavior is not desired, reset the SVSLPE/SVSHPE bits in the PMMRIE register at the beginning of the application.
<hr/>	
PMM11	<i>PMM Module</i>
Function	DCO 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 SVML 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 5.5 MHz, 8 MHz, 13 MHz, or 16.5 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, 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

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

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
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 LPM2/3/4 SVML State	Automatic SVSMLACE = 1 LPM2/3/4 SVML State	Wakeup Time LPM2/3/4
	0		x	OFF	OFF	OFF
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 LPM2/3/4 SVSH State	Manual SVSMHACE = 1 LPM2/3/4 SVSH State
	0	x	x		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
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 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

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*

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*

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
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
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

SD3
SD24_B Module
Function

Incorrect conversion result in twos complement mode when -VFS is applied

Description

When the SD converter is configured in twos complement mode with left or right alignment and any OSR setting, applying the -VFS voltage at the input will result in an erroneous output.

Workaround

None.

UCS11
UCS Module
Function

Modifying UCSCTL4 clock control register triggers an additional erroneous clock request

Description

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.

NOTE: The XT1/XT2 fault flag can be set regardless of which SELM/SELS/SELA bit combinations are being changed.

Workaround

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

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.

Alternatively it can be handled accordingly (clear falsely set fault flags) in the Interrupt

Service Routine to ensure proper OFIFG clearing.

USCI36
eUSCI Module

Function

UCLKI not usable in I2C master mode

Description

When EUSCIB is configured as I2C Master with the external UCLKI as clock source, the UCLKI signal is not available and cannot be used to source I2C clock.

Workaround

Use LFXTCLK via ACLK or HFXTCLK via SMCLK as clock source (BRCLK) for I2C in master mode with external clock source.

USCI37
eUSCI Module

Function

Reading RXBUF during an active I2C communication might result in unintended bus stalls.

Description

The falling edge of SCL bus line is used to set an internal RXBUF-written flag register, which is used to detect a potential RXBUF overflow. If this flag is cleared with a read access from the RXBUF register during a falling edge of SCL, the clear condition might be missed. This could result in an I2C bus stall at the next received byte.

Workaround

(1) Execute two consecutive reads of RXBUF, if $t_{SCL} > 4 \times t_{MCLK}$.

or

(2) Provoke an I2C bus stall before reading RXBUF. A bus stall can be verified by checking if the clock line low status indicator bit UCSCLOW is set for at least three USCI bit clock cycles i.e. $3 \times t_{BitClock}$.

USCI41
eUSCI Module

Function

UCBUSY bit of eUSCIA module stuck to 1 when device is in SPI mode.

Description

When eUSCIA is configured in SPI mode, and the last transfer bit changes from 0 to 1, the UCBUSY bit gets stuck to 1. This happens in all four combinations of Clock Phase and Clock Polarity options (UCAxCTLW0.UCCKPH & UCAxCTLW0.UCCKPL bits). There is no data loss or corruption. Because the UCBUSY bit is stuck to 1, the clock request stays enabled and adds additional current consumption in low power mode operation.

Workaround

Check on transmit or receive interrupt flag UCTXIFG/UCRXIFG instead of UCBUSY to know if the UCAxTXBUF buffer is empty or ready for the next complete character.

USCI42
eUSCI Module

Function

UART asserts UCTXCPITIFG after each byte in multi-byte transmission

Description

UCTXCPTIFG flag is triggered at the last stop bit of every UART byte transmission, independently of an empty buffer, when transmitting multiple byte sequences via UART. The erroneous UART behavior occurs with and without DMA transfer.

Workaround

None.

5 Document Revision History

Changes from family erratasheet to device specific erratasheet.

1. Errata USCI26 was removed
2. Package PN80 was added
3. Package PZ80 was removed
4. 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.

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.
3. Errata LCDB5 was added to the errata documentation.
4. Errata LCDB6 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. Errata AUXPMM1 was added to the errata documentation.

Changes from document Revision D to Revision E.

1. AUXPMM1 Description was updated.
2. AUXPMM1 Function was updated.
3. DMA10 Description was updated.
4. AUXPMM1 Workaround was updated.
5. Errata EEM23 was added to the errata documentation.
6. Errata CPU43 was added to the errata documentation.

Changes from document Revision E to Revision F.

1. CPU43 Description was updated.
2. Errata DMA9 was added to the errata documentation.
3. Device TLV Hardware Revision information added to erratasheet.

Changes from document Revision F to Revision G.

1. Errata PMM20 was added to the errata documentation.

Changes from document Revision G to Revision H.

1. BSL7 Workaround was updated.
2. BSL7 Function was updated.
3. Errata USCI37 was added to the errata documentation.

Changes from document Revision H to Revision I.

1. EEM19 Workaround was updated.
2. EEM17 Workaround was updated.
3. CPU43 Description was updated.
4. EEM23 Workaround was updated.
5. EEM17 Description was updated.
6. EEM23 Description was updated.
7. Errata ADC39 was added to the errata documentation.

8. Errata USCI36 was added to the errata documentation.
9. EEM19 Description was updated.

Changes from document Revision I to Revision J.

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

Changes from document Revision J to Revision K.

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

Changes from document Revision K to Revision L.

1. DMA9 Description was updated.
2. Errata CPU43 was removed from the errata documentation.
3. Errata AUXPMM2 was added to the errata documentation.
4. DMA9 Workaround was updated.
5. PMM18 Workaround was updated.

Changes from document Revision L to Revision M.

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

Changes from document Revision M to Revision N.

1. DMA9 Workaround was updated.

Changes from document Revision N to Revision O.

1. Errata BSL14 was added to the errata documentation.
2. Errata JTAG26 was added to the errata documentation.
3. Errata USCI41 was added to the errata documentation.

Changes from document Revision O to Revision P.

1. USCI37 Workaround was updated.

Changes from document Revision P to Revision Q.

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

Changes from document Revision Q to Revision R.

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

Changes from document Revision R to Revision S.

1. Errata USCI42 was added to the errata documentation.
2. Errata JTAG27 was added to the errata documentation.

Changes from document Revision S to Revision T.

1. Errata CPU46 was added to the errata documentation.

Changes from document Revision T to Revision U.

1. SD3 was added to the errata documentation.
2. CPU21 was added to the errata documentation.
3. CPU22 was added to the errata documentation.
4. Workaround for CPU40 was updated.
5. Workaround for CPU46 was updated.
6. Workaround for PMM15 was updated.
7. Description for USCI41 was updated.

Changes from document Revision U to Revision V.

1. Workaround for CPU46 was updated.

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ("TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources 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.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products (<http://www.ti.com/sc/docs/stdterms.htm>), [evaluation modules](#), and [samples](http://www.ti.com/sc/docs/sampterm.htm) (<http://www.ti.com/sc/docs/sampterm.htm>).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2017, Texas Instruments Incorporated