MSP430FG6626 Device Erratasheet

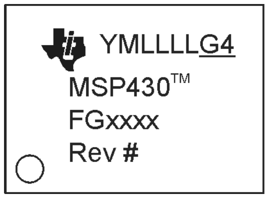
1 Revision History

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

Errata Number	Rev A
BSL14	✓
CPU37	✓
CPU40	✓
CTSD1	✓
DMA4	✓
DMA7	✓
DMA10	✓
EEM16	✓
EEM17	✓
EEM19	✓
EEM23	✓
LCDB5	✓
LCDB6	✓
PMM11	✓
PMM12	✓
PMM14	✓
PMM15	✓
PMM18	✓
PMM20	✓
PORT15	✓
PORT19	✓
UCS11	✓
USCI26	✓
USCI35	✓
USCI39	✓
USCI40	✓

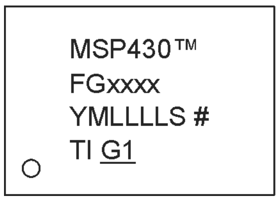
2 Package Markings

PZ100 ***LQFP (PZ) 100 Pin***



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

ZQW113 ***BGA (ZQW), 113 Pin***



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

3 Detailed Bug Description

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

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.

In C, no workaround is necessary since the compiler automatically generates the necessary NOPs.

CTSD1 *CTSD16 Module*

Function

CTSD16OFFG bit erroneously set while CTSD16 module is inactive

Description

The CTSD16CTL.CTSD16OFFG bit is erroneously set when the CTSD16 module is disabled and not actively converting (CTSD16CCTLx.CTSD16SC = 0). This CTSD16CTL.CTSD16OFFG bit can only be cleared once the CTSD16 module is enabled and actively converting (CTSD16CCTLx.CTSD16SC = 1).

This errata effectively nullifies the ability to trigger NMI interrupts in response to oscillator faults, unless CTSD16 is kept enabled.

Workaround

1) If CTSD16 is enabled, and the fault condition is ensured not to be present, then CTSD16OFFG and OFIFG can function normally. The only way to keep CTSD16 enabled indefinitely is by setting CTSD16SC.

2) While CTSD16 is not enabled, the OFIFG bit cannot be used. The other bits sourcing into it besides CTSD16OFFG (that is, XT1LFOFFG, XT1HFOFFG, XT2OFFG, and DCOFFG) can be polled or checked by software individually; but the ability to trigger the NMI upon OFIFG becoming set is no longer possible.

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.

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.

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 written by the DMA may be corrupted when a breakpoint is hit or when the debug session is halted.

Workaround Do not halt or use breakpoints during a DMA transfer.

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.

LCDB5 — *LCD_B Module*

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.

LCDB6 — *LCD_B Module*

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 LCDC 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'

PMM11 *PMM 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 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 SVMLFP = 1 and SVSLFP = 1) AND MCLK is sourced from the internal DCO running over 4 MHz, 7 MHz, 11 MHz, or 14 MHz at core voltage levels 0, 1, 2, and 3, respectively, the mask lasts only 2 us. MCLK is, therefore, susceptible to run out of spec for 4 us.

Workaround Set the MCLK divide bits in the Unified Clock System Control 5 Register (UCSCTL5) to divide MCLK by two prior to entering LPM3 or LPM4 (set DIVMx = 001). This prevents MCLK from running out of spec when the CPU wakes from the low-power mode. Following the wakeup from the low-power mode, wait 32, 48, 80, or 100 cycles for core voltage levels 0, 1, 2, and 3, respectively, before resetting DIVMx to zero and running MCLK at full speed [for example, __delay_cycles(100)].

PMM12 *PMM Module*

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 LMP4 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.

	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
SVSL	0	x	x	OFF	OFF	OFF	$t_{WAKE-UP\ FAST}$
	1	0	0	Normal	OFF	OFF	$t_{WAKE-UP\ SLOW}$
	1	0	1	Full Performance	OFF	OFF	$t_{WAKE-UP\ FAST}$
	1	1	0	Normal	Normal	OFF	$t_{WAKE-UP\ SLOW}$
	1	1	1	Full Performance	Full Performance	Normal	$t_{WAKE-UP\ FAST}$

	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
SVML	0	x	OFF	OFF	OFF	$t_{WAKE-UP\ FAST}$
	1	0	Normal	Normal	OFF	$t_{WAKE-UP\ SLOW}$
	1	1	Full Performance	Full Performance	Normal	$t_{WAKE-UP\ FAST}$

and

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

					Manual SVSMHACE = 0 LPM2/3/4 SVSH State	Manual SVSMHACE = 1 LPM2/3/4 SVSH State
SVSH	SVSHE	SVSHMD	SVSHFP	AM, LPM0/1 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 & SVMLFP)) )
{ // 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 VCORE voltage can experience voltage drop below the corresponding SVSL and SVML threshold (recommendation according to User's Guide) leading to an unexpected SVSL/SVML event. Depending on PMM configuration, this event triggers a POR or an interrupt.

> **NOTE:** As soon the SVSL or the SVML is enabled in Normal performance mode the device is in slow wakeup mode and this erratum does not apply.
>
> In addition, this erratum has sporadic characteristic due to an internal asynchronous circuit. The drop of Vcore does not have an impact on specified device performance.

Workaround

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

PORT15 — *PORT Module*

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

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.

USCI26 *USCI Module*

Function Tbuf parameter violation in I2C multi-master mode

Description 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.

Note: It is recommended to check if UCBBUSY bit is cleared before setting UCTXSTT=1.

Workaround None

USCI35 *USCI Module*

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

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

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

USCI39 — *USCI Module*

Function
USCI I2C IFGs UCSTTIFG, UCSTPIFG, UCNACKIFG

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

1. Pending Interrupt: One of the UCxIFG=1 AND UCxIE=1 while GIE=0

2. The GIE is set by software (e.g. EINT)

3. The pending interrupt is cleared by hardware (external I2C event) in a time window of 1 MCLK clock cycle after the "EINT" instruction is executed.

Workaround
Disable the UCSTTIFG, UCSTPIFG and UCNACKIFG before the GIE is set. After GIE is set, the local interrupt enable flags can be set again.

Assembly example:

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

NOP

EINT

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

USCI40 — *USCI Module*

Function
SPI Slave Transmit with clock phase select = 1

Description
In SPI slave mode with clock phase select set to 1 (UCAxCTLW0.UCCKPH=1), after the first TX byte, all following bytes are shifted by one bit with shift direction dependent on UCMSB. This is due to the internal shift register getting pre-loaded asynchronously when writing to the USCIA TXBUF register. TX data in the internal buffer is shifted by one bit after the RX data is received.

Workaround
Reinitialize TXBUF before using SPI and after each transmission.

If transmit data needs to be repeated with the next transmission, then write back previously read value:

```
UCAxTXBUF = UCAxTXBUF;
```

4 Document Revision History

Changes from device specific erratasheet to document Revision A.

1. Errata BSL14 was added to the errata documentation.