

## MSP430F2131 Device Erratasheet

---



---



---

### 1 Functional Errata Revision History

Errata impacting device's operation, function or parametrics.

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

Errata Number	Rev L	Rev K	Rev J	Rev I	Rev H	Rev G	Rev E	Rev D	Rev C
BCL6								✓	✓
BCL8								✓	✓
BCL9						✓	✓	✓	✓
BCL10						✓	✓	✓	✓
BCL11						✓	✓	✓	✓
BCL12	✓	✓	✓	✓	✓	✓	✓	✓	✓
BCL13					✓	✓	✓	✓	✓
BCL14	✓	✓	✓	✓	✓				
CPU5									✓
CPU11	✓	✓	✓	✓	✓	✓	✓	✓	✓
CPU14						✓	✓		
CPU45	✓	✓	✓	✓	✓	✓	✓	✓	✓
FLASH16	✓	✓	✓	✓	✓	✓	✓	✓	✓
FLASH17	✓	✓	✓	✓	✓	✓	✓	✓	✓
FLASH18	✓	✓	✓	✓	✓	✓	✓	✓	✓
FLASH19	✓	✓	✓	✓	✓	✓	✓	✓	✓
FLASH20	✓	✓	✓	✓	✓	✓	✓	✓	✓
FLASH22						✓	✓	✓	✓
FLASH24	✓	✓	✓	✓	✓	✓	✓	✓	✓
FLASH27	✓	✓	✓	✓	✓	✓	✓	✓	✓
FLASH36	✓	✓	✓	✓	✓	✓	✓	✓	✓
PORT8								✓	✓
PORT10						✓	✓	✓	✓
SYS15	✓	✓	✓	✓	✓	✓	✓	✓	✓
TA12	✓	✓	✓	✓	✓	✓	✓	✓	✓
TA16	✓	✓	✓	✓	✓	✓	✓	✓	✓
TA21	✓	✓	✓	✓	✓	✓	✓	✓	✓
TAB22	✓	✓	✓	✓	✓	✓	✓	✓	✓
XOSC5	✓	✓	✓	✓	✓	✓	✓	✓	✓
XOSC8			✓	✓	✓	✓	✓	✓	✓

### 2 Preprogrammed Software Errata Revision History

Errata impacting pre-programmed software into the silicon by Texas Instruments.

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

Errata Number	Rev L	Rev K	Rev J	Rev I	Rev H	Rev G	Rev E	Rev D	Rev C
<a href="#">BSL5</a>									✓

### 3 Debug only Errata Revision History

Errata only impacting debug operation.

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

Errata Number	Rev L	Rev K	Rev J	Rev I	Rev H	Rev G	Rev E	Rev D	Rev C
<a href="#">EEM20</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">JTAG15</a>							✓	✓	✓

### 4 Fixed by Compiler Errata Revision History

Errata completely resolved by compiler workaround. Refer to specific erratum for IDE and compiler versions with workaround.

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

Errata Number	Rev L	Rev K	Rev J	Rev I	Rev H	Rev G	Rev E	Rev D	Rev C
<a href="#">CPU4</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">CPU6</a>								✓	✓
<a href="#">CPU12</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">CPU13</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">CPU19</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓

Refer to the following MSP430 compiler documentation for more details about the CPU bugs workarounds.

#### TI MSP430 Compiler Tools (Code Composer Studio IDE)

- [MSP430 Optimizing C/C++ Compiler](#): Check the --silicon\_errata option
- [MSP430 Assembly Language Tools](#)

#### MSP430 GNU Compiler (MSP430-GCC)

- [MSP430 GCC Options](#): Check -msilicon-errata= and -msilicon-errata-warn= options
- [MSP430 GCC User's Guide](#)

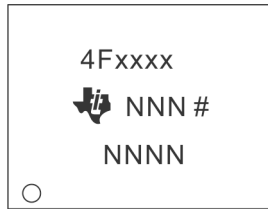
#### IAR Embedded Workbench

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

## 5 Package Markings

### DGV20

#### TVSOP (DGV), 20 Pin



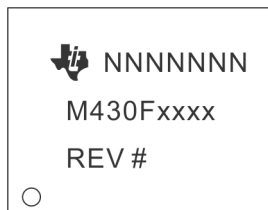
# = Die revision  
 ○ = Pin 1 location  
 N = Lot trace code



# = Die revision  
 ○ = Pin 1 location  
 N = Lot trace code

### DW20

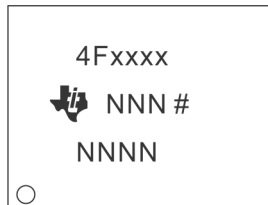
#### SOP (DW), 20 Pin



# = Die revision  
 ○ = Pin 1 location  
 N = Lot trace code

### PW20

#### TSSOP (PW), 20 Pin



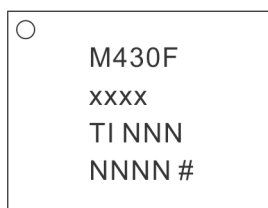
# = Die revision  
 ○ = Pin 1 location  
 N = Lot trace code



# = Die revision  
 ○ = Pin 1 location  
 N = Lot trace code

### RGE24

#### QFN (RGE), 24 Pin



# = Die revision  
 ○ = Pin 1 location  
 N = Lot trace code

## 6 Detailed Bug Description

<b>BCL6</b>	<b><i>BCS Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	DCO calibration error
<b>Description</b>	<p>The DCO calibration values stored in the information memory may differ by up to +- 2%. This applies to all Rev. C and the following Lot-Trace code Rev. D devices:</p> <p>Lot#57FNFCCK          Lot#58E081K          Lot#58E521K          Lot#58F5K9K          Lot#58F5KTK          Lot#58F860K          Lot#58F866K          Lot#58FKL3K          Lot#59C279K</p>
<b>Workaround</b>	Recalibrate the DCO using a reference clock source
<b>BCL8</b>	<b><i>BCS Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	Erroneous ISR DCO DC generator enable
<b>Description</b>	<p>When MCLK and SMCLK are sourced from a clock other than the DCO, the SCG0 DCO control bit is erroneously cleared upon interrupt service routine entry from LPMx by the CPU. This enables the DCO DC generator causing additional current consumption. After executing the RETI instruction, the status register SCG0 bit will return to a set state turning off the DC generator, eliminating this added current. The increased current will only occur during the interrupt service routine, and is in the range of 20 uA at 3V.</p>
<b>Workaround</b>	Set SCG0 in software at the beginning of an interrupt service routine. This will reduce the time that the DC generator is enabled to four MCLK.
<b>BCL9</b>	<b><i>BCS Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	ACLK divider modifications require delay before entering LPM3
<b>Description</b>	<p>After modifying the DIVAx bits, immediately entering LPM3 can cause the modification to be ignored and the divider settings not to take effect. Reading back the DIVAx bits will indicate the intended setting even when the divider has not been correctly applied.</p>
<b>Workaround</b>	When the DIVAx bits are modified, a delay of one complete ACLK (VLO or LFXT1CLK) period must elapse before entering LPM3. The delay is only necessary the first time LPM3 is entered after the DIVAx bits are modified. After the one-period delay, LPM3

may be entered and exited normally without additional delays.

**BCL10**
***BCS Module***
**Category**

Functional

**Function**

MCLK = ACLK and P2SEL control bits

**Description**

When using ACLK as the CPU MCLK clock source, the oscillator failsafe feature does not automatically switch MCLK to the DCO if the P2SEL6 or P2SEL7 bit is cleared. This applies when ACLK = LFXT1 (external low frequency clock source). The CPU will halt operation since no MCLK signal is present.

**Workaround**

None

**BCL11**
***BCS Module***
**Category**

Functional

**Function**

Watchdog failsafe when using ACLK

**Description**

When using ACLK as the WDT+ clock source, the WDT+ oscillator failsafe feature does not automatically switch to the DCO if the P2SEL6 or P2SEL7 bit is cleared. This applies when ACLK = LFXT1 (external low frequency clock source). The WDT+ will halt operation since no clock signal is present.

**Workaround**

None

**BCL12**
***BCS Module***
**Category**

Functional

**Function**

Switching RSELx or modifying DCOCTL can cause DCO dead time or a complete DCO stop

**Description**

After switching RSELx bits (located in register BCSTL1) from a value of >13 to a value of <12 OR from a value of <12 to a value of >13, the resulting clock delivered by the DCO can stop before the new clock frequency is applied. This dead time is approximately 20 us. In some instances, the DCO may completely stop, requiring a power cycle.

Furthermore, if all of the RSELx bits in the BCSTL1 register are set, modifying the DCOCTL register to change the DCOx or the MODx bits could also result in DCO dead time or DCO hang up.

**Workaround**

- When switching RSEL from >13 to <12, use an intermediate frequency step. The intermediate RSEL value should be 13.

Current RSEL	Target RSEL	Recommended Transition Sequence
15	14	Switch directly to target RSEL
14 or 15	13	Switch directly to target RSEL
14 or 15	0 to 12	Switch to 13 first, and then to target RSEL (two step sequence)
0 to 13	0 to 12	Switch directly to target RSEL

AND

- When switching RSEL from <12 to >13 it's recommended to set RSEL to its default value first (RSEL = 7) before switching to the desired target frequency.

AND

- In case RSEL is at 15 (highest setting) it's recommended to set RSEL to its default value first (RSEL = 7) before accessing DCOCTL to modify the DCOx and MODx bits. After the DCOCTL register modification the RSEL bits can be manipulated in an additional step.

In the majority of cases switching directly to intermediate RSEL steps as described above will prevent the occurrence of BCL12. However, a more reliable method can be implemented by changing the RSEL bits step by step in order to guarantee safe function without any dead time of the DCO.

Note that the 3-step clock startup sequence consisting of clearing DCOCTL, loading the BCSTL1 target value, and finally loading the DCOCTL target value as suggested in the in the "TLV Structure" chapter of the [MSP430x2xx Family User's Guide](#) is not affected by BCL12 if (and only if) it is executed after a device reset (PUC) prior to any other modifications being made to BCSTL1 since in this case RSEL still is at its default value of 7. However any further changes to the DCOx and MODx bits will require the consideration of the workaround outlined above.

### **BCL13**

#### ***BCS Module***

---

**Category**

Functional

**Function**

DCO powerup halt

**Description**

When subject to very slow Vcc rise times, the device may enter into a state where the DCO does not oscillate. No JTAG access or program execution is possible and the device will remain in a reset state until the supply voltage is disconnected.

**Workaround**

Apply a Vcc poweron ramp  $\geq 10\text{V/second}$  under all power-on/power-cycle scenarios.

### **BCL14**

#### ***BCS Module***

---

**Category**

Functional

**Function**

Oscillator fault forced in bypass mode when P2SEL.7 bit is not set

**Description**

When the LFXT1 oscillator is used in bypass mode and P2SEL.7 is not set, the oscillator fault flag (OFIFG) will be forced to set and cannot be cleared. Due to the failsafe logic, LFXT1 cannot be used as MCLK in this case. The bug only affects the behavior of the oscillator fault, the clocking itself works properly.

**Workaround**

Set both P2SEL.6 and P2SEL.7 if the application requires correct function of the oscillator fault flag (e.g. MCLK failsafe logic).

---

**NOTE:** Setting P2SEL.7 bit disables the GPIO functionality and enables the input schmitt trigger of the pin. P2.7 should be tied to a fixed voltage level (VCC or GND) to prevent cross current.

---

### **BSL5**

#### ***BSL Module***

---

**Category**

Software in ROM

**Function**

BSL might not start if RST/NMI pin is configured as NMI input

**Description** If the RST/NMI pin is configured to NMI, the bootstrap loader may not be started. Unpredictable operations will result.

**Workaround** None

## **CPU4** *CPU Module*

---

**Category** Compiler-Fixed

**Function** PUSH #4, PUSH #8CPU4 - Bug

**Description** The single operand instruction PUSH cannot use the internal constants (CG) 4 and 8. The other internal constants (0, 1, 2, -1) can be used. The number of clock cycles is different:

PUSH #CG uses address mode 00, requiring 3 cycles, 1 word instruction

PUSH #4/#8 uses address mode 11, requiring 5 cycles, 2 word instruction

**Workaround** Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v2.x until v6.20	User is required to add the compiler flag option below. --hw_workaround=CPU4
IAR Embedded Workbench	IAR EW430 v6.20 or later	Workaround is automatically enabled
TI MSP430 Compiler Tools (Code Composer Studio)	v1.1 or later	
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 167 or later	

## **CPU5** *CPU Module*

---

**Category** Functional

**Function** Interrupt handler, incorrect interrupt vector fetch

**Description** When an interrupt is accepted and the interrupt vector is applied to the MAB, an interrupt vector

address 32 bytes lower than the expected address may be applied to the address bus. This is true

only when the stack pointer address is in one of the following ranges:

0x02C0 to 0x02DF

0x0280 to 0x029F

0x0240 to 0x025F

0x0200 to 0x021F

**Workaround** Solution #1: Limit stack depth to 32 bytes.

OR

Solution #2 (Recommended, works under all conditions): Copy the interrupt vector table (0xFFE0 to

0xFFFF) in flash to addresses 0xFFC0 to 0xFFDF, respectively.

**CPU6**
**CPU Module**
**Category**

Compiler-Fixed

**Function**

Incorrect execution of ADD instruction

**Description**

When the CPU executes an ADD (.B or .W) instruction using indirect addressing mode with destination R1 or R4 to R15, directly after a RET or RETI instruction, the addition will be executed twice. This bug does not apply in the case that the indirect source working register is R2 or R3 (constant generator access for #2 or #4 are valid).

The instruction word mask that corresponds to the CPU6 condition is as follows:

**Format I Instruction**

(See *MSP430x2xx Family User's Guide*)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
op-code				s-reg				Ad	B/W	As			d-reg			
MSB															LSB	
0	1	0	1	X	X	X	X	0	X	1	0	X	X	X	X	

X = Don't care

s-reg = R2 or R3 are not affected

d-reg = R0, R2, or R3 are not affected

**Workaround**

1. (Preferred) Use indexed addressing with offset 0 instead of indirect addressing:

ADD(.B/.W) 0(R10),R11

Note that some assemblers may convert this instruction to indirect addressing.

Alternatively, when the source is in assembly, making the following changes works around the bug. Perform the following only if Workaround 1 cannot be implemented.

or

2. Place an instruction (for example NOP) between any instances of CALL and ADD(.B/.W) instructions where the ADD immediately follows the CALL.

or

3. Replace all RETI instructions with 'DW 01301h'. The instruction word for RETI is 01300h. By replacing this directly in assembly with 01301h, the bug is avoided and the RETI instruction is executed properly; for example:

WDT\_ISR ; Exit LPM3 on reti

bic.w #LPM3,0(SP) ;

;reti ; Replace RETI with

DW 01301h ; this line of code

Refer to the table below for compiler-specific fix implementation information.



IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v4.x or later	User is required to add the compiler flag option below. --hw_workaround=CPU6 Add -h option for Assembler
TI MSP430 Compiler Tools (Code Composer Studio)	Fix not available	
MSP430 GNU Compiler (MSP430-GCC)	Fix not available	

## CPU11

### *CPU Module*

#### Category

Functional

#### Function

Invalid status register after program counter access

#### Description

When addressing the program counter (PC) in register mode where the PC is the destination, the Status Register (SR) may be erroneous. The instructions BIS, BIC and MOV do not affect SR contents. Only CPU flags are effected: DOES NOT apply to LPMx control bits.

#### Workaround

None

## CPU12

### *CPU Module*

#### Category

Compiler-Fixed

#### Function

CMP or BIT instruction with PC as destination

#### Description

Instructions immediately following a CMP(.B) or BIT(.B) instruction that use the Program Counter (PC) as the destination register will be ignored or erroneously executed. When the following instruction is longer than one word, the second word is fetched by the CPU and decoded as the instruction, leading to unpredictable behavior. Affected source addressing modes are indexed and indirect addressing modes.

Example:

```
cmp &200,PC
```

```
add #4,R8
```

In this example, the add instruction will not be executed.

#### Workaround

Insert a NOP instruction after the BIT or CMP instruction. The NOP will be ignored and program execution will continue as expected.

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	Fix not available	
TI MSP430 Compiler Tools (Code Composer Studio)	15.12.0.LTS	User is required to add the compiler or assembler flag option below. --silicon_errata=CPU12

IDE/Compiler	Version Number	Notes
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 389 or later	User is required to add the compiler or assembler flag option below. -msilicon-errata=cpu12 -msilicon-errata-warn=cpu12 generates a warning in addition
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 5.x build 14 or later	User is required to add the compiler or assembler flag option below. -msilicon-errata=cpu12 -msilicon-errata-warn=cpu12 generates a warning in addition

## CPU13

### *CPU Module*

#### Category

Compiler-Fixed

#### Function

Arithmetic operations on the Status Register may lead to erroneous results

#### Description

Performing arithmetic operations with the Status Register (SR) as the destination address will not update the SR as intended. The result in SR may be invalid, leading to erroneous low power mode entry. Arithmetic operations are defined as all instructions that modify the SR flag bits (RRA, SUB, XOR and ADD for example).

#### Workaround

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	Fix not available	
TI MSP430 Compiler Tools (Code Composer Studio)	15.12.0.LTS	User is required to add the compiler or assembler flag option below. --silicon_errata=CPU13
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 389 or later	User is required to add the compiler or assembler flag option below. -msilicon-errata=cpu13 -msilicon-errata-warn=cpu13 generates a warning in addition
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 5.x build 14 or later	User is required to add the compiler or assembler flag option below. -msilicon-errata=cpu13 -msilicon-errata-warn=cpu13 generates a warning in addition

## CPU14

### *CPU Module*

#### Category

Functional

#### Function

Erroneous setting of SCG0 after reset

#### Description

The SCG0 bit in the CPU status register (SR) is set after any reset (PUC or POR) if bit #6 in the reset vector destination address is set. Setting SCG0 turns off the DCO dc generator when DCOCLK is not used for MCLK or SMCLK.

#### Workaround

1) As the error only occurs after PUC or POR, it is sufficient to clear the SCG0 bit at the beginning of the program code; for example:

bic.w #SCG0, SR

OR

2) Avoid using reset destination addresses where bit #6 is set. Allowed reset vector destination addresses are: xx0xh, xx1xh, xx2xh, xx3xh, xx8xh, xx9xh, xxAxh, xxBxh.

## CPU19

### CPU Module

#### Category

Compiler-Fixed

#### Function

CPUOFF modification may result in unintentional register read

#### Description

If an instruction that modifies the CPUOFF bit in the Status Register is followed by an instruction with an indirect addressed operand (e.g. MOV @R8, R9, RET, POP, POPM), an unintentional register read operation can occur during the wakeup of the CPU. If the unintentional read occurs to a read sensitive register (e.g. UCB0RXBUF, TAIV), which changes its value or the value of other registers (IFG's), the bug leads to lost interrupts or wrong register read values.

#### Workaround

Insert a NOP instruction after each CPUOFF instruction.

OR

Refer to the table below for compiler-specific fix implementation information.

Note that compilers implementing the fix may lead to double stack usage when RET/RETA follows the compiler-inserted NOP.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v6.20.1 until v6.40	User is required to add the compiler or assembler flag option below. --hw_workaround=nop_after_lpm
IAR Embedded Workbench	IAR EW430 v6.40 or later	Workaround is automatically enabled
TI MSP430 Compiler Tools (Code Composer Studio)	15.12.0.LTS	User is required to add the compiler or assembler flag option below. --silicon_errata=CPU19
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 389 or later	User is required to add the compiler or assembler flag option below. -msilicon-errata=cpu19 -msilicon-errata-warn=cpu19 generates a warning in addition
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 5.x build 14 or later	User is required to add the compiler or assembler flag option below. -msilicon-errata=cpu19 -msilicon-errata-warn=cpu19 generates a warning in addition

## CPU45

### CPU Module

#### Category

Functional

#### Function

CPU speed performance limitation

#### Description

The CPU register contents may become unpredictable during CPU register operations if the device operates at minimum Vcc required for system speed performance above 6MHz when using LFXT1 in HF mode (BCSCTL1.XTS = 1) and sourcing MCLK to clock

the CPU under certain conditions. This is dependent on voltage and CPU clock (MCLK) frequency and duty-cycle.

**Workaround**

With respect to the system speed performance above 6MHz versus minimum required Vcc

1. Use external clocks with 50% positive duty cycle when sourced to MCLK

OR

2. Use internally divided clock for MCLK (BCSCTL2.DIVMx > 00)

OR

3. Reduce LFXT1 (used in HF mode) or external clock frequency by 20% when sourced to MCLK

**EEM20**
***EEM Module***


---

**Category**

Debug

**Function**

Debugger might clear interrupt flags

**Description**

During debugging read-sensitive interrupt flags might be cleared as soon as the debugger stops. This is valid in both single-stepping and free run modes.

**Workaround**

None.

**FLASH16**
***FLASH Module***


---

**Category**

Functional

**Function**

Modifying INFOA addresses when LOCKA = 1 will modify main flash memory

**Description**

When attempting to write to an address location or perform a segment erase of INFOA while the LOCKA bit is set, flash memory beginning at main memory location 0xFC40 and extending for 64 bytes to address 0xFC7F will be modified erroneously. These 64 bytes are addressed and modified in place of the INFOA addresses when writes or erases are performed within the INFOA address space and LOCKA = 1.

**Workaround**

Prior to modifying (writing or erasing) any address within the INFOA Flash memory segment, properly clear the LOCKA control bit as described in the MSP430x2xx User's Guide ([SLAU144](#)) to unlock the segment. Once the modification is complete, setting the LOCKA bit is recommended.

**FLASH17**
***FLASH Module***


---

**Category**

Functional

**Function**

Flash read when EEI = 1

**Description**

When a segment erase in FLASH is initialized and the erase cycle is interrupted, any read from FLASH may be erroneous until after the erase cycle is completed.

**Workaround**

When EEI = 1, initiate all FLASH segment erases using a byte access to an even address within the segment to be erased. For example:

(Assembly)

```
mov.b #0FFh, &1000h
```

(C)  
\*(char\*)0x1000 = 0; //ERASE

**FLASH18**
***FLASH Module***


---

**Category**

Functional

**Function**

Information flash access when EEI = 1

**Description**

When a segment erase to any flash location has been initiated and the erase cycle has been interrupted, it is not possible to read from the information flash memory locations until the erase cycle is completed.

**Workaround**

None

**FLASH19**
***FLASH Module***


---

**Category**

Functional

**Function**

EEI feature does not work for code execution from RAM

**Description**

When the program is executed from RAM, the flash controller EEI feature does not work. The erase cycle is suspended and the interrupt is serviced, but there is a problem while resuming with the erase cycle.

Addresses applied to flash are different than the actual values while resuming erase cycle after ISR execution.

**Workaround**

None

**FLASH20**
***FLASH Module***


---

**Category**

Functional

**Function**

Flash erase is not correctly executed when EEI is enabled and interrupts occur

**Description**

Flash erase is not executed correctly when EEI is enabled and an interrupt occurs in a certain time window. The BUSY signal from the flash controller and the "JMP \$" feature, that keeps the program counter at the current position, disappear shortly after a flash erase cycle has been initialized.

**Workaround**

None

**FLASH22**
***FLASH Module***


---

**Category**

Functional

**Function**

Flash controller may prevent correct LPM entry

**Description**

When ACLK (or SMCLK) is used as the flash controller clock source, and this clock source gets deactivated due to a low-power mode entry while a flash erase or write operation is pending, the flash controller will keep ACLK (or SMCLK) active even after the flash operation has been completed. This will result in an incorrect LPM entry and increased current consumption. Note that this issue can only occur when the Flash

operation and the low-power mode entry are initiated from code located in RAM.

**Workaround** Do not enter low-power modes while flash erase or write operations are active. Wait for the operation to be completed before entering a low-power mode.

## **FLASH24** *FLASH Module*

---

**Category** Functional

**Function** Write or erase emergency exit can cause failures

**Description** When a flash write or erase is abruptly terminated, the following flash accesses by the CPU may be unreliable resulting in erroneous code execution. The abrupt termination can be the result of one the following events:

1) The flash controller clock is configured to be sourced by an external crystal. An oscillator fault occurs thus stopping this clock abruptly.

or

2) The Emergency Exit bit (EMEX in FCTL3) when set forces a write or an erase operation to be terminated before normal completion.

or

3) The Enable Emergency Interrupt Exit bit (EEIEX in FCTL1) when set with GIE=1 can lead to an interrupt causing an emergency exit during a Flash operation.

**Workaround**

- 1) Use the internal DCO as the flash controller clock provided from MCLK or SMCLK.

or

- 2) After setting EMEX = 1, wait for a sufficient amount of time before Flash is accessed again.

or

- 3) No Workaround. Do not use EEIEX bit.

## **FLASH27** *FLASH Module*

---

**Category** Functional

**Function** EEI feature can disrupt segment erase

**Description** When a flash segment erase operation is active with EEI feature selected (EEI=1 in FLCTL1) and GIE=0, the following can occur:

An interrupt event causes the flash erase to be stopped, and the flash controller expects an RETI to resume the erase. Because GIE=0, interrupts are not serviced and RETI will never happen.

**Workaround**

- 1) Do not set bit EEI=1 when GIE = 0.

or,

- 2) Force an RETI instruction during the erase operation during the check for BUSY=1 (FCTL3).

Sample code:

```
MOV R5, 0(R5) ; Dummy write, erase segment
LOOP: BIT #BUSY, &FCTL3 ; test busy bit
```

```
JMP SUB_RET1 ; Force RETI instruction
JNZ LOOP ; loop while BUSY=1
SUB_RET1: PUSH SR
RETI
```

**FLASH36**
***FLASH Module***


---

**Category**

Functional

**Function**

Flash content may degrade due to aborted page erases

**Description**

If a page erase is aborted by EEIEX, the flash page containing the last instruction before erase operation will start to degrade. This effect is incremental and, after repetitions, may lead to corrupted flash content.

**Workaround**

- Use the EEI (interrupted erasing) feature instead of EEIEX (abort erasing).

or

- A PSA checksum can be calculated over affected flash page using the marginal read mode (marginal 0). If PSA sum differs from expected PSA value the affected flash page has to be reprogrammed.

or

- Start flash erasing from RAM and limit system frequency to <1MHz (to ensure 6-us delay after EEIEX). If the last instruction before erasing is located in RAM, flash cell degradation does not occur.

**JTAG15**
***JTAG Module***


---

**Category**

Debug

**Function**

Using TDO as input for TDI does not work when JTAG is used to drive the MAB

**Description**

When the TDO is used as TDI, the MAB is not correctly driven by JTAG. This causes problems when using the function GangProgramTarget() of the GANG430.DLL. This function is usually used for programming individual serial numbers, for programming calibration data into devices, and for restoration of DCO calibration data after erasing of segment A.

**Workaround**

Upgrade to GANG430.DLL version 1.42 or higher.

**PORT8**
***PORT Module***


---

**Category**

Functional

**Function**

Increased leakage current

**Description**

The LFXT1 oscillator circuit is not disabled when P2SEL.6 = 0 or P2SEL.7=0. This can result in a higher leakage current on P2.7 XOUT when P2DIR.7 = 0 and P2SEL.7=0.

**Workaround**

If using P2.7 as an input, select the bypass mode by setting the LFXT1Sx = 11

**PORT10**
***PORT Module***


---

<b>Category</b>	Functional
<b>Function</b>	Pull-up/down resistor selection when module pin function is selected
<b>Description</b>	When the pull-up/down resistor for a certain port pin is enabled (PxREN.y=1) and the module port pin function is selected (PxSEL.y=1), the pull-up/down resistor configuration of this pin is controlled by the respective module output signal (Module X OUT) instead of the port output register (PxOUT.y).
<b>Workaround</b>	None. Do not set PxSEL.y and PxREN.y at the same time.

---

**SYS15**                      ***SYS Module***


---

<b>Category</b>	Functional
<b>Function</b>	LPM3 and LPM4 currents exceed specified limits
<b>Description</b>	LPM3 and LPM4 currents may exceed specified limits if the SMCLK source is switched from DCO to VLO or LFXT1 just before the instruction to enter LPM3 or LPM4 mode.
<b>Workaround</b>	After clock switching, a delay of at least four new clock cycles (VLO or LFXT1) must be implemented to complete the clock synchronization before going into LPM3 or LPM4.

---

**TA12**                      ***TIMER\_A Module***


---

<b>Category</b>	Functional
<b>Function</b>	Interrupt is lost (slow ACLK)
<b>Description</b>	Timer_A counter is running with slow clock (external TACLK or ACLK) compared to MCLK. The compare mode is selected for the capture/compare channel and the CCRx register is incremented by one with the occurring compare interrupt (if TAR = CCRx). Due to the fast MCLK the CCRx register increment (CCRx = CCRx+1) happens before the Timer_A counter has incremented again. Therefore the next compare interrupt should happen at once with the next Timer_A counter increment (if TAR = CCRx + 1). This interrupt gets lost.
<b>Workaround</b>	Switch capture/compare mode to capture mode before the CCRx register increment. Switch back to compare mode afterwards.

---

**TA16**                      ***TIMER\_A Module***


---

<b>Category</b>	Functional
<b>Function</b>	First increment of TAR erroneous when IDx > 00
<b>Description</b>	The first increment of TAR after any timer clear event (POR/TACLK) happens immediately following the first positive edge of the selected clock source (INCLK, SMCLK, ACLK or TACLK). This is independent of the clock input divider settings (ID0, ID1). All following TAR increments are performed correctly with the selected IDx settings.
<b>Workaround</b>	None

---

**TA21**                      ***TIMER\_A Module***

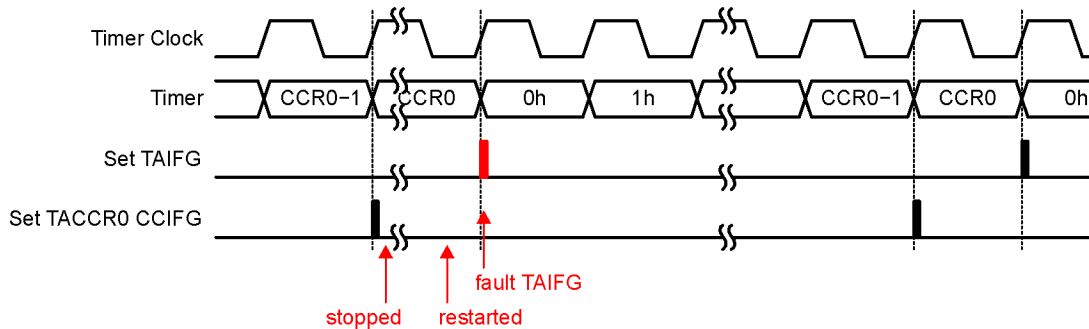

---



**Category** Functional

**Function** TAIFG Flag is erroneously set after Timer A restarts in Up Mode

**Description** In Up Mode, the TAIFG flag should only be set when the timer counts from TACCR0 to zero. However, if the Timer A is stopped at TAR = TACCR0, then cleared (TAR=0) by setting the TACLRL bit, and finally restarted in Up Mode, the next rising edge of the TACLK will erroneously set the TAIFG flag.



**Workaround** None.

**TAB22** ***TIMER\_A/TIMER\_B Module***

**Category** Functional

**Function** Timer\_A/Timer\_B register modification after Watchdog Timer PUC

**Description** Unwanted modification of the Timer\_A/Timer\_B registers TACTL/TBCTL and TAIV/TBIV can occur when a PUC is generated by the Watchdog Timer(WDT) in Watchdog mode and any Timer\_A/Timer\_B counter register TACCRx/TBCCRx is incremented/decremented (Timer\_A/Timer\_B does not need to be running).

**Workaround** Initialize TACTL/TBCTL register after the reset occurs using a MOV instruction (BIS/BIC may not fully initialize the register). TAIV/TBIV is automatically cleared following this initialization.

Example code:

```
MOV.W #VAL, &TACTL
```

or

```
MOV.W #VAL, &TBCTL
```

Where, VAL=0, if Timer is not used in application otherwise, user defined per desired function.

**XOSC5** ***XOSC Module***

**Category** Functional

**Function** LF crystal failures may not be properly detected by the oscillator fault circuitry

**Description** The oscillator fault error detection of the LFXT1 oscillator in low frequency mode (XTS = 0) may not work reliably causing a failing crystal to go undetected by the CPU, i.e. OFIFG will not be set.

---

<b>Workaround</b>	None
<b>XOSC8</b>	<b><i>XOSC Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	ACLK failure when crystal ESR is below 40 kOhm.
<b>Description</b>	When ACLK is sourced by a low frequency crystal with an ESR below 40 kOhm, the duty cycle of ACLK may fall below the specification; the OFIFG may become set or in some instances, ACLK may stop completely.
<b>Workaround</b>	Please refer to "XOSC8 Guidance" found at <a href="#">SLAA423</a> for information regarding working with this erratum.

## 7 Document Revision History

Changes from family erratasheet to device specific erratasheet.

1. Errata TA22 was renamed to TAB22
2. Description for TAB22 was updated

Changes from device specific erratasheet to document Revision A.

1. Errata EEM20 was added to the errata documentation.

Changes from document Revision A to Revision B.

1. BCL12 Workaround was updated.

Changes from document Revision B to Revision C.

1. Errata TA21 was added to the errata documentation.

Changes from document Revision C to Revision D.

1. BCL14 Workaround was updated.

Changes from document Revision D to Revision E.

1. Package Markings section was updated.

Changes from document Revision E to Revision F.

1. TA21 Description was updated.

Changes from document Revision F to Revision G.

1. Errata CPU45 was added to the errata documentation.

Changes from document Revision G to Revision H.

1. Workaround for CPU19 was updated.
2. Function for CPU6 was updated.
3. Workaround for CPU6 was updated.
4. Function for CPU4 was updated.
5. Workaround for CPU4 was updated.
6. Function for CPU12 was updated.
7. Description for CPU12 was updated.
8. Workaround for CPU12 was updated.
9. Function for CPU13 was updated.
10. Description for CPU13 was updated.
11. Workaround for CPU13 was updated.

Changes from document Revision H to Revision I.

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

## 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 © 2018, Texas Instruments Incorporated