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.

<table>
<thead>
<tr>
<th>Errata Number</th>
<th>Rev G</th>
<th>Rev F</th>
<th>Rev E</th>
<th>Rev D</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCL12</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BCL14</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FLASH16</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SYS15</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TA12</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TA16</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TA21</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TAB22</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>USI4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>USI5</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XOSC5</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XOSC8</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

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.
The device doesn't have Software in ROM errata.

3 Debug only Errata Revision History

Errata only impacting debug operation.
✓ The check mark indicates that the issue is present in the specified revision.

<table>
<thead>
<tr>
<th>Errata Number</th>
<th>Rev G</th>
<th>Rev F</th>
<th>Rev E</th>
<th>Rev D</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEM20</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

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.
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

<table>
<thead>
<tr>
<th>Package Type</th>
<th>Description</th>
<th>Marking Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>N14 PDIP (N), 14 Pin</td>
<td>MSP430Gxxxx</td>
<td>NNNNNNN # = Die revision, N = Lot trace code</td>
</tr>
<tr>
<td>PW14 TSSOP (PW), 14 Pin</td>
<td>NNNG4</td>
<td>Gxxxx = Die revision, N = Lot trace code</td>
</tr>
<tr>
<td>RSA16 QFN (RSA), 16 Pin</td>
<td></td>
<td>M430Gxxxx = Die revision, NNNN = Lot trace code</td>
</tr>
</tbody>
</table>
### Detailed Bug Description

#### 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 BCSCTL1) 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 BSCTL1 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.

<table>
<thead>
<tr>
<th>Current RSEL</th>
<th>Target RSEL</th>
<th>Recommended Transition Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>Switch directly to target RSEL</td>
</tr>
<tr>
<td>14 or 15</td>
<td>13</td>
<td>Switch directly to target RSEL</td>
</tr>
<tr>
<td>14 or 15</td>
<td>0 to 12</td>
<td>Switch to 13 first, and then to target RSEL (two step sequence)</td>
</tr>
<tr>
<td>0 to 13</td>
<td>0 to 12</td>
<td>Switch directly to target RSEL</td>
</tr>
</tbody>
</table>

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 BCSCTL1 target value, and finally loading the DCOCTL target value as suggested 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 BCSCTL1 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.

#### 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.

---

**CPU4**

**CPU Module**

**Category**

Compiler-Fixed

**Function**

PUSH #4, PUSH #8 - CPU4 - 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.

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

**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
### Detailed Bug Description

**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.

### SYS15

**Category**

SYS Module

**Function**

LPM3 and LPM4 currents exceed specified limits

**Description**

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.

**Workaround**

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

**Category**

Functional

**Function**

Interrupt is lost (slow ACLK)

**Description**

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.

**Workaround**

Switch capture/compare mode to capture mode before the CCRx register increment. Switch back to compare mode afterwards.

### TA16

**Category**

Functional

**Function**

First increment of TAR erroneous when IDx > 00

**Description**

The first increment of TAR after any timer clear event (POR/TACLR) 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.

**Workaround**

None

### TA21

**Category**

Functional

**Function**

First increment of TAR erroneous when IDx > 00

**Description**

The first increment of TAR after any timer clear event (POR/TACLR) 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.

**Workaround**

None
**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 TACLR bit, and finally restarted in Up Mode, the next rising edge of the TACLK will erroneously set the TAIFG flag.

![Diagram of Timer Clock](image)

**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.

---

**USI4**

**USI Module**

**Category**

**Functional**

**Function**

I2C Slave mode can generate a glitch at SCL

**Description**

USI I2C Slave Operation at slower communication rates (less than 20kbps). During I2C bus active operation, if USICNT is written while SCL is high, I2C module will generate a glitch on SCL that can corrupt the I2C bus sequence.
<table>
<thead>
<tr>
<th>Module</th>
<th>Category</th>
<th>Function</th>
<th>Description</th>
<th>Workaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>USI5</td>
<td>Functional</td>
<td>SPI master generates one additional clock after module reset</td>
<td>Initializing the USI in SPI MASTER mode with the USICKPH bit set generates one additional clock pulse than defined by the value in the USICNTx bits on the SCLK pin during the first data transfer after module reset. For example, if the USICNTx bits hold the value eight, nine clock pulses are generated on the SCLK pin for the first transfer only.</td>
<td>Load USICNTx with a count of N-1 bits (where N is the required number of bits) for the first transfer only.</td>
</tr>
<tr>
<td>XOSC5</td>
<td>Functional</td>
<td>LF crystal failures may not be properly detected by the oscillator fault circuitry</td>
<td>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.</td>
<td>None</td>
</tr>
<tr>
<td>XOSC8</td>
<td>Functional</td>
<td>ACLK failure when crystal ESR is below 40 kOhm.</td>
<td>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.</td>
<td>Please refer to &quot;XOSC8 Guidance&quot; found at SLAA423 for information regarding working with this erratum.</td>
</tr>
</tbody>
</table>

Workaround
Verify that SCL is low before writing USICNT register.
Document Revision History

Changes from family erratasheet to device specific erratasheet.
1. Errata BCL14 was added
2. Errata EEM20 was added
3. Errata SYS15 was added
4. Revision F was added
5. Revision G was added
6. Errata TA22 was renamed to TAB22
7. Description for TAB22 was updated

Changes from device specific erratasheet to document Revision A.
1. USI5 Workaround was updated.

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. Function for CPU4 was updated.
2. Workaround for CPU4 was updated.

Changes from document Revision G to Revision H.
1. Erratasheet format update.
2. Added errata category field to "Detailed bug description" section