NOTE: Silicon Revisions AA, AB, and AD use BSL version 1.61. For specific information on this version of the BSL and its proper usage, see the MSP430 Memory Programming User’s Guide (SLAU265).

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></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC1</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ADC5</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ADC7</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ADC8</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ADC9</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ADC10</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ADC11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ADC18</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ADC25</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BCL5</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MPY2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PORT3</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>RES3</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>RES4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TA12</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TA16</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TA21</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TAB22</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TB1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TB2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TB3</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TB4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TB14</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TB16</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TB24</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>US13</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>US14</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>US15</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>WDG2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BSL3</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BSL4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BSL5</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

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></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EEM20</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

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
## Package Markings

<table>
<thead>
<tr>
<th>Package Markings</th>
<th>TQFP (PAG), 64 Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PAG64</strong></td>
<td><img src="image1" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td>NNNNNNN</td>
</tr>
<tr>
<td></td>
<td>M430FxXXX</td>
</tr>
<tr>
<td></td>
<td>REV #</td>
</tr>
<tr>
<td></td>
<td># = Die revision</td>
</tr>
<tr>
<td></td>
<td>O = Pin 1 location</td>
</tr>
<tr>
<td></td>
<td>N = Lot trace code</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Package Markings</th>
<th>LQFP (PM), 64 Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PM64</strong></td>
<td><img src="image2" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td>NNNNNNNN G4</td>
</tr>
<tr>
<td></td>
<td>M430FxXXX</td>
</tr>
<tr>
<td></td>
<td>REV #</td>
</tr>
<tr>
<td></td>
<td># = Die revision</td>
</tr>
<tr>
<td></td>
<td>O = Pin 1 location</td>
</tr>
<tr>
<td></td>
<td>N = Lot trace code</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Package Markings</th>
<th>QFN (RTD), 64 Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RTD64</strong></td>
<td><img src="image3" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td>M430FxXXX</td>
</tr>
<tr>
<td></td>
<td>TI NNN</td>
</tr>
<tr>
<td></td>
<td>NNNN #</td>
</tr>
<tr>
<td></td>
<td># = Die revision</td>
</tr>
<tr>
<td></td>
<td>O = Pin 1 location</td>
</tr>
<tr>
<td></td>
<td>N = Lot trace code</td>
</tr>
</tbody>
</table>
## Detailed Bug Description

### ADC1

**Category:** Functional  
**Function:** Start of conversion  
**Description:** In single conversion/sequence mode (CONSEQ=0/1), the next conversion can be started with ADC12SC. It is not necessary to clear ENC before setting ADC12SC. This is contrary to the specification.  
**Workaround:** None

### ADC5

**Category:** Functional  
**Function:** Interrupt flag register  
**Description:** ADC12 interrupt flag may not be set when the CPU simultaneously accesses the ADC12IFG register.  
**Workaround:** There is no need to access the interrupt flag register to process interrupt situations. Please use the ADC12IV register to identify the interrupt event. The corresponding flag bits will be reset automatically. Additional details are discussed in the device family user's guide.

### ADC7

**Category:** Functional  
**Function:** Conversion time overflow  
**Description:** The timing overflow flag is set when the device is in sequence mode (CONSEQ = 1 or 3) and MSC = 0, even if no overflow has occurred.  
**Workaround:** Verify correct timing and do not enable Conversion-Time Overflow interrupt.

### ADC8

**Category:** Functional  
**Function:** Interrupt flag register  
**Description:** Clearing flags in the interrupt flag register with a CPU instruction will not clear the latest interrupt flag.  
**Workaround:** Clear interrupt flags by accessing the conversion-memory registers.

### ADC9
Function | Interrupt vector register  
---|---  
Description | If the ADC12 uses a different clock than the CPU (MCLK) and more than one ADC interrupt is enabled, the ADC12IV register content may be unpredictable for one clock cycle. This happens if, during the execution of an ADC interrupt, another ADC interrupt with higher priority occurs.  
Workaround | - Read out ADC12IV twice and use only when values are equal.  
| or  
| - Use ADC12IFG to determine which interrupt has occurred.  

**ADC10**  
**ADC12 Module**  
Category | Functional  
Function | Unintended start of conversion  
Description | Accessing ADC12OVIE or ADC12TOVIE at the end of an ADC12 conversion with BIS/BIC commands can cause the ADC12SC bit to be set again immediately after it was cleared. This might start another conversion, if ADC12SC is configured to trigger the ADC (SHS = 0).  
Workaround | If ADC12SC is configured to trigger the ADC, the control bits ADC12OVIE and ADC12TOVIE should be modified only when the ADC is not busy (ADC12BUSY = 0).  

**ADC11**  
**ADC12 Module**  
Category | Functional  
Function | Temporary leakage current after conversion  
Description | The ADC12 causes temporary leakage current after a completed conversion. Duration and magnitude of the leakage current depends on parasitic effects.  
Workaround | None  

**ADC18**  
**ADC12 Module**  
Category | Functional  
Function | Incorrect conversion result in extended sample mode  
Description | The ADC12 conversion result can be incorrect if the extended sample mode is selected (SHP = 0), the conversion clock is not the internal ADC12 oscillator (ADC12SSELE > 0), and one of the following two conditions is true:  
- The extended sample input signal SHI is asynchronous to the clock source used for ADC12CLK and the undivided ADC12 input clock frequency exceeds 3.15 MHz.  
| or  
| - The extended sample input signal SHI is synchronous to the clock source used for ADC12CLK and the undivided ADC12 input clock frequency exceeds 6.3 MHz.  
Workaround | - Use the pulse sample mode (SHP = 1).
or
- Use the ADC12 internal oscillator as the ADC12 clock source.

or
- Limit the undivided ADC12 input clock frequency to 3.15 MHz.

or
- Use the same clock source (such as ACLK or SMCLK) to derive both SHI and ADC12CLK, to achieve synchronous operation, and also limit the undivided ADC12 input clock frequency to 6.3 MHz.

---

### ADC25

**ADC12 Module**

**Category**
Functional

**Function**
Write to ADC12CTL0 triggers ADC12 when CONSEQ = 00

**Description**
If ADC conversions are triggered by the Timer_B module and the ADC12 is in single-channel single-conversion mode (CONSEQ = 00), ADC sampling is enabled by write access to any bit(s) in the ADC12CTL0 register. This is contrary to the expected behavior that only the ADC12 enable conversion bit (ADC12ENC) triggers a new ADC12 sample.

**Workaround**
When operating the ADC12 in CONSEQ=00 and a Timer_B output is selected as the sample and hold source, temporarily clear the ADC12ENC bit before writing to other bits in the ADC12CTL0 register. The following capture trigger can then be re-enabled by setting ADC12ENC = 1.

---

### BCL5

**BCS Module**

**Category**
Functional

**Function**
RSELx bit modifications can generate high frequency spikes on MCLK

**Description**
When DIVMx = 00 or 01 the RSELx bits of the Basic Clock Module are incremented or decremented in steps of 2 or greater, the DCO output may momentarily generate high frequency spikes on MCLK, which may corrupt CPU operation. This is not an issue when DIVMx = 10 or 11.

**Workaround**
Set DIVMx = 10 or 11 to divide the MCLK input prior to modifying RSELx. After the RSELx bits are configured as desired, the DIVMx setting can be changed back to the original selection.

---

### BSL3

**BSL Module**

**Category**
Software in ROM

**Function**
Receiving frames

**Description**
Receiving frames with a checksum value equal to a legal address can change the content of this address or the bootstrap loader may stop operation.

**Workaround**
Software workaround is available as part of BSLDEMO.exe, found as part of the BSL-SCRIPTER download on the page for MSPBSL.
BSL4  
**BSL Module**

**Category**  
Software in ROM

**Function**  
Flash memory can not be programmed

**Description**  
The bootstrap loader software cannot program the flash memory.

**Workaround**  
Software workaround is available as part of BSLDEMO.exe, found as part of the BSL-SCRIPTER download on the page for MSPBSL.

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.

<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.
Detailed Bug Description

MPY2  
**MPY Module**

**Category**  Functional

**Function**  Multiplier Result register corruption

**Description**  Depending on the address of the write instruction, writing to the multiplier result registers (RESHI, RESLO, or SUMEXT) may corrupt the result registers. The address dependency varies between a 2-word and a 3-word instructions.

**Workaround**  Ensure that a write instruction to an MPY result register (for example, mov.w #200, &RESHI) is not located at an address with the four least significant bits shown in Table 1:

<table>
<thead>
<tr>
<th>RESLOW 013Ah</th>
<th>RESHI 013Ch</th>
<th>SUMEXT 013Eh</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>C</td>
</tr>
</tbody>
</table>

Table 1. Sensitive Addresses for Write Access to MPY Result Registers MAB[3:0]

PORT3  
**PORT Module**

**Category**  Functional

**Function**  Port interrupts can get lost

**Description**  Port interrupts can get lost if they occur during CPU access of the P1IFG and P2IFG registers.

**Workaround**  None

RES3  
**RESET Module**

**Category**  Functional

**Function**  Reset

**Description**  When RST/NMI is held low during power up of VCC, some internal drivers are not reset correctly. This may result in a high Icc current until the internal power-on signal has generated one clock cycle to reset the internal drivers. This limits the time when the excess current can occur to the time the power-up circuit is active.

**Workaround**  None

RES4  
**RESET Module**
Category: Functional

No reset if external resistor exceeds certain value

Description: No reset of the device is performed if the external pull down resistor on RST/NMI pin is above a certain limit. The limits are:

- Vcc = 1.8V: maximum pull down resistor = 12 kohm
- Vcc = 3.0V: maximum pull down resistor = 5 kohm
- Vcc = 3.6V: maximum pull down resistor = 2.5 kohm

In addition, a higher current consumption occurs during high/low RST/NMI signal transition when using improper resistors.

Workaround: Use external pulldown resistors below the listed values or directly drive RST/NMI low to generate a reset.

TA12  
**TIMER_A Module**

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  
**TIMER_A Module**

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  
**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 TACLR 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.

**TB1**
**TIMER_B Module**

**Category**
Functional

**Function**
"Equal mode" when grouping compare latches

**Description**
The "equal mode" for loading the compare latches (CLLD = 3) cannot be used when compare latches are grouped (TBCLGRP > 0).

**Workaround**
None

**TB2**
**TIMER_B Module**

**Category**
Functional
**Function**

Interrupt is lost (slow ACLK)

**Description**

Timer_B counter is running with slow clock (external TBCLK or ACLK) compared to MCLK. The compare mode is selected for the capture/compare channel and the CCRx register is incremented by 1 with the occurring compare interrupt (if TBR = CCRx).

Due to the fast MCLK, the CCRx register increment (CCRx = CCRx + 1) happens before the Timer_B counter has incremented again. Therefore, the next compare interrupt should happen at once with the next Timer_B counter increment (if TBR = CCRx + 1). This interrupt is lost.

**Workaround**

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

---

**TB3**

**TIMER_B Module**

**Category**

Functional

**Function**

Port is switched to 3-state independent of selected function

**Description**

Incorrect 3-state function of Ports P4.0/TB0 through P4.6/TB6 (TBoutHiZ control). If TBoutHiZ is set to high, all ports P4.0/TB0 through P4.6/TB6 are set to 3-state, independent of the P4SEL.x control signals. This means a port P4.x is switched to 3-state with TBoutHiZ, even if it is not selected for Timer_B function. In addition, the ports P4.0/TB0 through P4.6/TB6 are switched to 3-state with TBoutHiZ, even if the port direction (direction control from module) is set to input. This is in accordance with the specification description but, nevertheless, is an unexpected behavior.

**Workaround**

No workaround.

Port function as specified

---

**Port Realization With TB3 Bug**

---

*Port function as specified*
Detailed Bug Description

TB4

**Category**: Functional  
**Function**: Group function  
**Description**: If the shadow registers are organized in groups (SHR = 1, 2, or 3), one shadow register is not loaded correctly. This happens when the last CCRx register within a group is loaded at exactly the same time that the timer counter reaches the event for loading the shadow registers (TBR = 0 or TBR = CCR0).

**Workaround**: Ensure that all CCRx registers within a group are loaded before the shadow register load event occurs.

TB14

**Category**: Functional  
**Function**: PWM output  
**Description**: The PWM output unit may behave erroneously if the condition for changing the PWM output (EQUx or EQU0) and the condition for loading the shadow register TBCLx happen at the same time. Depending on the load condition for the shadow registers (CLLD bits in TBCCTLx), there are four possible error conditions:

1. Change CCRx register from any value to CCRx = 0 (for example, sequence for CCRx = 4 3 2 0 0 0)
2. Change CCRx register from CCRx = 0 to any value (for example, sequence for CCRx = 0 0 0 2 3 4)
3. Change CCRx register from any value to current SHD0 (CCR0) value (for example, sequence for CCRx = 4 2 5 SHD0 3 8)
4. Change CCRx register from current SHD0 (CCR0) value to any value (for example,
sequence for $\text{CCR}x = 4\ 2\ \text{SHD}0\ 5\ 3\ 8$)

**Workaround**

No general workaround available.

---

**TB16**

**TIMER_B Module**

<table>
<thead>
<tr>
<th>Category</th>
<th>Functional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>First increment of TBR erroneous when IDx &gt; 00</td>
</tr>
<tr>
<td>Description</td>
<td>The first increment of TBR after any timer clear event (POR/TBCLR) happens immediately following the first positive edge of the selected clock source (INCLK, SMCLK, ACLK, or TBCLK). This is independent of the clock input divider settings (ID0, ID1). All following TBR increments are performed correctly with the selected IDx settings.</td>
</tr>
</tbody>
</table>

**Workaround**

None

---

**TB24**

**TIMER_B Module**

<table>
<thead>
<tr>
<th>Category</th>
<th>Functional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>TBIFG Flag is erroneously set after Timer B restarts in Up Mode</td>
</tr>
<tr>
<td>Description</td>
<td>In Up Mode, the TBIFG flag should only be set when the timer resets from TBCCR0 to zero. However, if the Timer A is stopped at $\text{TBR} = \text{TBCCR0}$, then cleared ($\text{TBR}=0$) by setting the TBCLR bit, and finally restarted in Up Mode, the next rising edge of the TBCLK will erroneously set the TBIFG flag.</td>
</tr>
</tbody>
</table>

---

**US13**

**USART Module**

<table>
<thead>
<tr>
<th>Category</th>
<th>Functional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>Unpredictable program execution</td>
</tr>
<tr>
<td>Description</td>
<td>USART interrupts requested by URXS can result in unpredictable program execution if this request is not served within two bit times of the received data.</td>
</tr>
</tbody>
</table>

**Workaround**

Ensure that the interrupt service routine is entered within two bit times of the received data.
### US14
**Category**: Functional  
**Function**: Start edge of received characters may be ignored  
**Description**: When using the USART in UART mode with UxBR0 = 0x03 and UxBR1 = 0x00, the start edge of received characters may be ignored due to internal timing conflicts within the UART state machine. This condition does not apply when UxBR0 is > 0x03.  
**Workaround**: None

### US15
**Category**: Functional  
**Function**: UART receive with two stop bits  
**Description**: USART hardware does not detect a missing second stop bit when SPB = 1. The Framing Error Flag (FE) will not be set under this condition and erroneous data reception may occur.  
**Workaround**: None (Configure USART for a single stop bit, SPB = 0)

### WDG2
**Category**: Functional  
**Function**: Incorrectly accessing a flash control register  
**Description**: If a key violation is caused by incorrectly accessing a flash control register, the watchdog interrupt flag is set in addition to the expected PUC.  
**Workaround**: None
Document Revision History

Changes from family erratasheet to device specific erratasheet.
1. Errata MPY2 was added

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. Errata TA21 was added to the errata documentation.

Changes from document Revision B to Revision C.
1. Errata TB24 was added to the errata documentation.

Changes from document Revision C to Revision D.
1. Package Markings section was updated.

Changes from document Revision D to Revision E.
1. TA21 Description was updated.

Changes from document Revision E to Revision F.
1. Function for CPU4 was updated.
2. Workaround for CPU4 was updated.

Changes from document Revision F to Revision G.
1. Workaround for BSL4 was updated.
2. Function for BSL3 was updated.
3. Workaround for BSL3 was updated.

Changes from document Revision G to Revision H.
1. Erratasheet format update.
2. Added errata category field to "Detailed bug description" section
3. ADC11 is now impacting silicon Revision L
4. ADC11 is no longer impacting silicon Revision AA
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/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2018, Texas Instruments Incorporated