

# MSP430F135 Device Erratasheet

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.

Errata Number	Rev AE	Rev AD	Rev AB	Rev AA	Rev S	Rev Q	Rev O	Rev N	Rev M	Rev L
<a href="#">ADC1</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">ADC5</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">ADC7</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">ADC8</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">ADC9</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">ADC10</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">ADC11</a>										✓
<a href="#">ADC18</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">ADC25</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">BCL5</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">PORT3</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">RES3</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">RES4</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">TA12</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">TA16</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">TA21</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">TAB22</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">TB1</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">TB2</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">TB3</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">TB4</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">TB14</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">TB16</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">TB24</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">US13</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">US14</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">US15</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">WDG2</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

## 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 AE	Rev AD	Rev AB	Rev AA	Rev S	Rev Q	Rev O	Rev N	Rev M	Rev L
<a href="#">BSL3</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<a href="#">BSL4</a>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<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 AE	Rev AD	Rev AB	Rev AA	Rev S	Rev Q	Rev O	Rev N	Rev M	Rev L
<a href="#">EEM20</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 AE	Rev AD	Rev AB	Rev AA	Rev S	Rev Q	Rev O	Rev N	Rev M	Rev L
<a href="#">CPU4</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

### PAG64



#### TQFP (PAG), 64 Pin

 NNNNNNNN M430Fxxxx REV # 	# = Die revision ○ = Pin 1 location N = Lot trace code
---	--

 NNNNNNNNG4 M430Fxxxx REV # 	# = Die revision ○ = Pin 1 location N = Lot trace code
---	--


### PM64


#### LQFP (PM), 64 Pin

 NNNNNNNN G4 M430Fxxxx REV # 	# = Die revision ○ = Pin 1 location N = Lot trace code
--	--

### RTD64

#### QFN (RTD), 64 Pin

 M430Fxxxx TI NNN NNNN #	# = Die revision ○ = Pin 1 location N = Lot trace code
--	--

 M430Fxxxx TI NNN G3 NNNN #	# = Die revision ○ = Pin 1 location N = Lot trace code
---	--

## 6 Detailed Bug Description

<b>ADC1</b>	<b><i>ADC12 Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	Start of conversion
<b>Description</b>	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.
<b>Workaround</b>	None
<b>ADC5</b>	<b><i>ADC12 Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	Interrupt flag register
<b>Description</b>	ADC12 interrupt flag may not be set when the CPU simultaneously accesses the ADC12IFG register.
<b>Workaround</b>	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.
<b>ADC7</b>	<b><i>ADC12 Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	Conversion time overflow
<b>Description</b>	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.
<b>Workaround</b>	Verify correct timing and do not enable Conversion-Time Overflow interrupt.
<b>ADC8</b>	<b><i>ADC12 Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	Interrupt flag register
<b>Description</b>	Clearing flags in the interrupt flag register with a CPU instruction will not clear the latest interrupt flag.
<b>Workaround</b>	Clear interrupt flags by accessing the conversion-memory registers.
<b>ADC9</b>	<b><i>ADC12 Module</i></b>
<b>Category</b>	Functional

<b>Function</b>	Interrupt vector register
<b>Description</b>	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.
<b>Workaround</b>	- Read out ADC12IV twice and use only when values are equal. or - Use ADC12IFG to determine which interrupt has occurred.
<hr/>	
<b>ADC10</b>	<b><i>ADC12 Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	Unintended start of conversion
<b>Description</b>	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).
<b>Workaround</b>	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).
<hr/>	
<b>ADC11</b>	<b><i>ADC12 Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	Temporary leakage current after conversion
<b>Description</b>	The ADC12 causes temporary leakage current after a completed conversion. Duration and magnitude of the leakage current depends on parasitic effects.
<b>Workaround</b>	None
<hr/>	
<b>ADC18</b>	<b><i>ADC12 Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	Incorrect conversion result in extended sample mode
<b>Description</b>	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 (ADC12SSEL > 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.
<b>Workaround</b>	- 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 framesBug

**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.

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	

**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.

---

**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 I<sub>cc</sub> 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

**Function** 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:

V<sub>cc</sub> = 1.8V: maximum pull down resistor = 12 kohm

V<sub>cc</sub> = 3.0V: maximum pull down resistor = 5 kohm

V<sub>cc</sub> = 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.

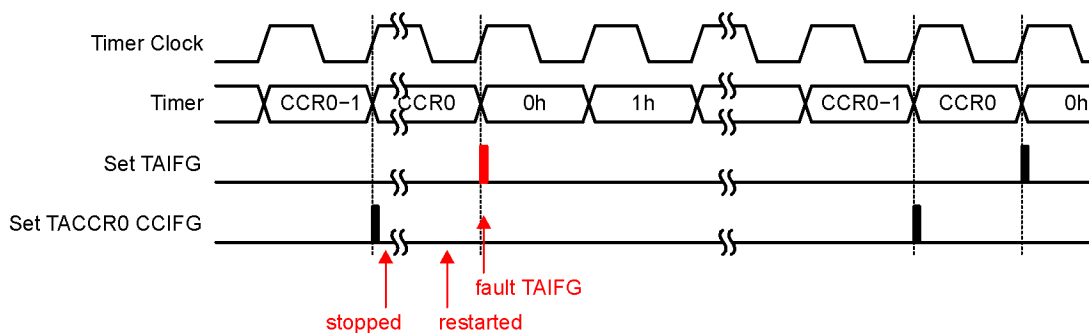
**Workaround** None

**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

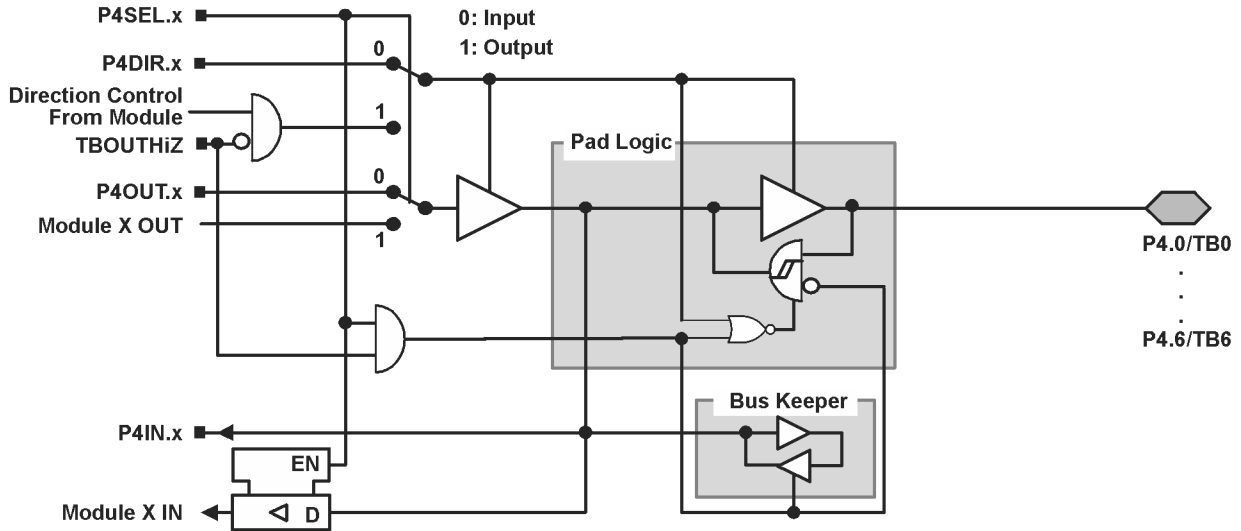
**Description** Unwanted modification of the Timer\_A/Timer\_B registers TACTL/TBCTL and TAIV/TBIV can occur when a PUC is generated by the Watchdog Timer(WDT) in Watchdog mode and any Timer\_A/Timer\_B counter register TACCRx/TBCCRx is incremented/decremented (Timer\_A/Timer\_B does not need to be running).

<b>Workaround</b>	<p>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.</p> <p>Example code:</p> <pre>MOV.W #VAL, &amp;TACTL</pre> <p>or</p> <pre>MOV.W #VAL, &amp;TBCTL</pre> <p>Where, VAL=0, if Timer is not used in application otherwise, user defined per desired function.</p>
<hr/>	
<b>TB1</b>	<b><i>TIMER_B Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	"Equal mode" when grouping compare latches
<b>Description</b>	The "equal mode" for loading the compare latches (CLLD = 3) cannot be used when compare latches are grouped (TBCLGRP > 0).
<b>Workaround</b>	None
<hr/>	
<b>TB2</b>	<b><i>TIMER_B Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	Interrupt is lost (slow ACLK)
<b>Description</b>	<p>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).</p> <p>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.</p>
<b>Workaround</b>	Switch capture/compare mode to capture mode before the CCRx register increment. Switch back to compare mode afterward.
<hr/>	
<b>TB3</b>	<b><i>TIMER_B Module</i></b>
<b>Category</b>	Functional
<b>Function</b>	Port is switched to 3-state independent of selected function
<b>Description</b>	<p>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.</p>

**Workaround** No workaround.

Port function as specified

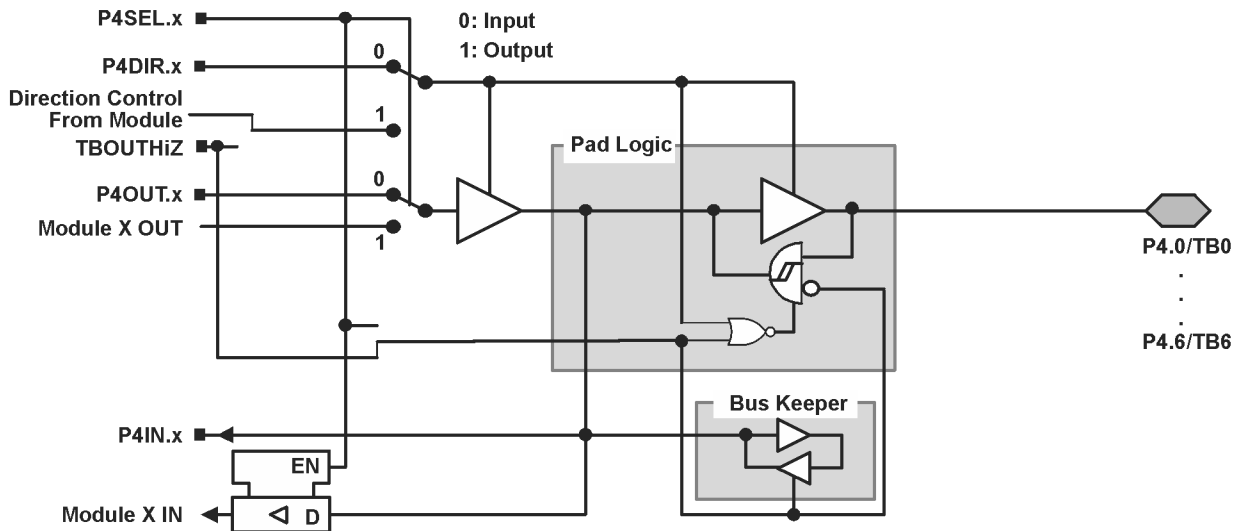
**port P4, P4.0 to P4.6, input/output with Schmitt-trigger**



x: bit identifier, 0 to 6 for Port P4

**Port Realization With TB3 Bug**

**port P4, P4.0 to P4.6, input/output with Schmitt-trigger**



x: bit identifier, 0 to 6 for Port P4

**TB4** *TIMER\_B Module*

**Category** Functional

<b>Function</b>	Group function
<b>Description</b>	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).
<b>Workaround</b>	Ensure that all CCRx registers within a group are loaded before the shadow register load event occurs.

---

**TB14** ***TIMER\_B Module***


---

<b>Category</b>	Functional
<b>Function</b>	PWM output
<b>Description</b>	<p>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:</p> <ol style="list-style-type: none"> <li>1. Change CCRx register from any value to CCRx = 0 (for example, sequence for CCRx = 4 3 2 0 0 0)</li> <li>2. Change CCRx register from CCRx = 0 to any value (for example, sequence for CCRx = 0 0 0 2 3 4)</li> <li>3. Change CCRx register from any value to current SHD0 (CCR0) value (for example, sequence for CCRx = 4 2 5 SHD0 3 8)</li> <li>4. Change CCRx register from current SHD0 (CCR0) value to any value (for example, sequence for CCRx = 4 2 SHD0 5 3 8)</li> </ol>
<b>Workaround</b>	No general workaround available.

---

**TB16** ***TIMER\_B Module***


---

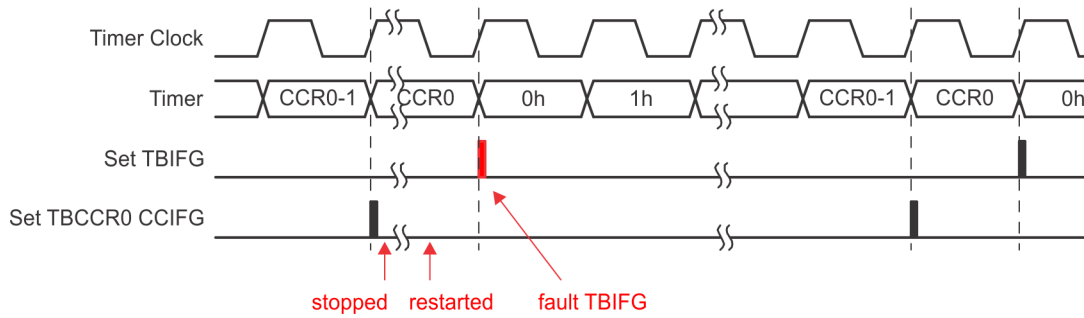
<b>Category</b>	Functional
<b>Function</b>	First increment of TBR erroneous when IDx > 00
<b>Description</b>	The first increment of TBR after any timer clear event (POR/TBCLR) happens immediately following the first positive edge of the selected clock source (INCLK, SMCLK, ACLK, or TBCLK). This is independent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

---

**TB24** ***TIMER\_B Module***


---

<b>Category</b>	Functional
<b>Function</b>	TBIFG Flag is erroneously set after Timer B restarts in Up Mode
<b>Description</b>	In Up Mode, the TBIFG flag should only be set when the timer resets from TBCCR0 to zero. However, if the Timer B is stopped at TBR = TBCCR0, then cleared (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.



**Workaround** None.

**US13** *USART Module*

**Category** Functional

**Function** Unpredictable program execution

**Description** 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.

**Workaround** Ensure that the interrupt service routine is entered within two bit times of the received data.

**US14** *USART Module*

**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** *USART Module*

**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** *WDT Module*

**Category** Functional

<b>Function</b>	Incorrectly accessing a flash control register
<b>Description</b>	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.
<b>Workaround</b>	None

## 7 Document Revision History

Changes from family erratasheet to device specific erratasheet.

Initial release

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

Changes from document Revision H to Revision I.

1. Description for TB24 was updated.

## IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale ([www.ti.com/legal/termsofsale.html](http://www.ti.com/legal/termsofsale.html)) or other applicable terms available either on [ti.com](http://ti.com) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2019, Texas Instruments Incorporated