## 1 Revision History

✓ 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>
</tr>
</thead>
<tbody>
<tr>
<td>BCL12</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BCL13</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BCL16</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CPU14</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CPU19</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>EEM20</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FLASH21</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FLASH22</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FLASH24</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FLASH26</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FLASH27</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FLASH36</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>JTAG13</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>JTAG14</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PORT10</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SYS15</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>
</tr>
<tr>
<td>TA16</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>
</tr>
<tr>
<td>TAB22</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>
</tr>
<tr>
<td>TB16</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>
</tr>
<tr>
<td>USCI16</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>USCI20</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>USCI21</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>USCI22</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>USCI23</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>USCI24</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>USCI25</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>USCI26</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>USCI27</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>USCI30</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>USCI35</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XOSEC5</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XOSEC8</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
## 2 Package Markings

<table>
<thead>
<tr>
<th>Package Markings</th>
<th>Description</th>
<th>Marking Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DA38</strong> TSSOP (DA), 38 Pin</td>
<td>YM = Year and Month Date Code LLLL = LOT Trace Code S = Assembly Site Code # = DIE Revision o = PIN 1</td>
<td></td>
</tr>
<tr>
<td><strong>RHA40</strong> QFN (RHA), 40 Pin</td>
<td>YM = Year and Month Date Code LLLL = LOT Trace Code S = Assembly Site Code # = DIE Revision o = PIN 1</td>
<td></td>
</tr>
<tr>
<td><strong>YFF49</strong> DSBGA (YFF), 49 Pin</td>
<td>YM = Year and Month Date Code LLLL = LOT Trace Code S = Assembly Site Code # = DIE Revision o = PIN 1</td>
<td></td>
</tr>
</tbody>
</table>
3 Detailed Bug Description

BCL12

Function
Switching RSELx or modifying DCOCCTL 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 DCOCCTL 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.

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 DCOCCTL to modify the DCOx and MODx bits. After the DCOCCTL 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 DCOCCTL, loading the BCSCTL1 target value, and finally loading the DCOCCTL 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 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.

BCL13

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.
BCL16

**BCS Module**

**Function**
SMCLK clock source selection from XT1/VLO to DCO

**Description**
When the MCLK and the SMCLK do not use the DCO, the DCO is off. The DCO does not start if the clock source for SMCLK is changed from XT1/VLO to DCO. As a result, the SMCLK remains high. Note: This is only true for SMCLK. The DCO starts if the clock source of MCLK is set to DCO.

**Workaround**
Set clock source of MCLK to DCO by either:

1) Setting the selection bits SELMx of BCSCTL2 register to '00' or '01'.

OR

2) Setting the OFIFG bit of IFG1 register. Note: This triggers the oscillator fault logic that automatically starts the DCO. Reset the OFIFG bit to further use the XT1/VLO.

For both options, if the XT1/VLO is still required to source MCLK, revert the clock source of MCLK back to XT1/VLO afterwards.

CPU14

**CPU Module**

**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**

**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.

EEM20

**EEM Module**

**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

**FLASH21  FLASH Module**

Function  Info memory read data corrupted

Description  Flash addresses between 0x1080 and 0x10ff (information memory) might not be read correctly. Supply voltages and addressing mode affect the read value.

Workaround  None.

**FLASH22  FLASH Module**

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**

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 of 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.

**FLASH26  FLASH Module**
**Function**
Main memory read data corrupted

**Description**
Flash addresses between 0xC000 and 0xffff might not be read correctly. Supply voltages and addressing mode affect the read value.

**Workaround**
None

---

**FLASH27 FLASH Module**

**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 (FCLTL3).
   Sample code:
   
   MOV R5, 0(R5) ; Dummy write, erase segment
   LOOP: BIT #BUSY, &FCTL3 ; test busy bit
   JMP SUB_RETI ; Force RETI instruction
   JNZ LOOP ; loop while BUSY=1
   SUB_RETI: PUSH SR
   RETI

---

**FLASH36 FLASH Module**

**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.

---

**JTAG13 JTAG Module**

---
**Detailed Bug Description**

**Function**
PSA Checksum generation fails

**Description**
PSA checksum generation gives a wrong result when data_psa is executed during test clock low phase and last address of flash information memory addresses 0x107E or 0x10FE are part of the calculation.

**Workaround**
Calculate PSA sum when test clock is at high level

---

**JTAG14**

**JTAG Module**

**Function**
Releasing JTAG control can corrupt CPU registers during debug

**Description**
During a debug session, on rare occasions, the CPU register contents can get corrupted when JTAG control is released by the debugger. This behavior is exhibited during, but not limited to, the use of the "Use Virtual Breakpoints" and "Force Single Stepping" features in the IAR Embedded Workbench software. This bug does not affect normal device and application operation, such as starting a device out of POR and executing application code.

In order for the bug to occur, both of the following two conditions must be true:
1. The CPU (MCLK) is sourced by the DCO.
2. The "External Resistor (Rosc)" feature of the DCO is not used.

**Workaround**
Use an external crystal or a digital high-speed clock source connected to the LFXT1 oscillator to source the CPU (MCLK) during a debug session. Alternatively, use the on-chip DCO in the "External Resistor (Rosc)" configuration. Note that, in this case, an external resistor connected to the device Rosc pin is mandatory, and that the factory-programmed DCO calibration constants cannot be applied directly.

---

**PORT10**

**PORT Module**

**Function**
Pull-up/down resistor selection when module pin function is selected

**Description**
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).

**Workaround**
None. Do not set PxSEL.y and PxREN.y at the same time.

---

**SYS15**

**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**

**TIMER_A Module**

**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**

**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**

**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 resets 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 showing Timer Clock, Timer, CCR0-1, CCR0, 0h, 1h, CCR0-1, CCR0, 0h, Set TAIFG, Set TACCR0 CCIFG, stopped, fault TAIFG, restarted](image-url)

**Workaround**

None.

TAB22

**TIMER_A/TIMER_B Module**

**Function**

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). TAI/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.

---

**TB2**  
**TIMER_B Module**

**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.

---

**TB16**  
**TIMER_B Module**

**Function**  
First increment of TBR erroneous when IDx > 00

**Description**  
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.

**Workaround**  
None

---

**TB24**  
**TIMER_B Module**

**Function**  
TBIFG Flag is erroneously set after Timer B restarts in Up Mode

**Description**  
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 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.
### USCI16

**USCI Module**

**Function**
UART/IrDA Mode Lost Characters

**Description**
When configured for UART/IrDA mode, the USCI baud rate generator may halt operation under the following conditions:

1. IrDA mode: repeated invalid start bits on the receive line
   
2. UART/IrDA modes: positive pulse on the receive line during break character reception inside the stop bit time slot (the second stop bit time slot in case of UCSPB=1) with a pulse width that passes the deglitch filter but is shorter than half a bit time.

After halting, additional characters will be ignored. Transmit functionality is not affected.

**Workaround**
Check the UCBUSY flag status periodically in software. If the flag is set and no character has been received in the expected time, reset the USCI module in software. To reset the USCI module, toggle UCSWRST and re-enable the USCI interrupts.

### USCI20

**USCI Module**

**Function**
I2C Mode Multi-master transmitter issue

**Description**
When configured for I2C master-transmitter mode, and used in a multi-master environment, the USCI module can cause unpredictable bus behavior if all of the following four conditions are true:

1. Two masters are generating SCL

And

2. The slave is stretching the SCL low phase of an ACK period while outputting NACK on SDA

And

3. The slave drives ACK on SDA after the USCI has already released SCL, and then the SCL bus line gets released

And

4. The transmit buffer has not been loaded before the other master continues communication by driving SCL low

The USCI will remain in the SCL high phase until the transmit buffer is written. After the transmit buffer has been written, the USCI will interfere with the current bus activity and...
Detailed Bug Description

may cause unpredictable bus behavior.

Workaround

1 - Ensure that slave doesn't stretch the SCL low phase of an ACK period
   Or
2 - Ensure that the transmit buffer is loaded in time
   Or
3 - Do not use the multi-master transmitter mode

USCI21

USCI Module

Function

UART IrDA receive filter

Description

The IrDA receive filter can be used to filter pulses with length UCAIRRXFL configured in UCAxIRRCTL register. If UCIRRXFE is set the IrDA receive decoder may filter out pulses longer than the configured filter length depending on frequency of BRCLK. This is resulting in framing errors or corrupted data on the receiver side.

Workaround

Depending on the used baud rate and the configured filter length a maximum frequency for BRCLK needs to be set to avoid this issue:

For baud rates equal and higher than 115.000 the maximum allowed BRCLK frequency is equal to the max specified system frequency.

\[
\text{Max BRCLK} = \frac{\text{Filter Length} + 64}{2} \times \frac{\text{Baud Rate} \times 16}{3 \times 10^6}
\]
<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>Filter Length UCIRRXFL (dec)</th>
<th>Max BRCLK (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9600</td>
<td>64</td>
<td>3.28</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>2.46</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>2.05</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1.84</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.74</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.69</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.66</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1.64</td>
</tr>
<tr>
<td>19200</td>
<td>64</td>
<td>6.55</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>4.92</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>4.1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>3.69</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3.48</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3.38</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3.33</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>3.28</td>
</tr>
<tr>
<td>38400</td>
<td>64</td>
<td>13.11</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>9.83</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>8.19</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>7.37</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>6.96</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6.76</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>6.66</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>6.55</td>
</tr>
<tr>
<td>56000</td>
<td>64</td>
<td>19.11</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>14.34</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>11.95</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>10.75</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>10.15</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>9.86</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>9.71</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>9.56</td>
</tr>
</tbody>
</table>
USCI22  
**USCI Module**

**Function**
I2C Master Receiver with 10-bit slave addressing

**Description**
Unexpected behavior of the USCI_B can occur when configured in I2C master receive mode with 10-bit slave addressing under the following conditions:

1) The USCI sends first byte of slave address, the slave sends an ACK and when second address byte is sent, the slave sends a NACK.

2) Master sends a repeat start condition (If UCTXSTT=1).

3) The first address byte following the repeated start is acknowledged. However, the second address byte is not sent, instead the Master incorrectly starts to receive data and sets UCBxRXIFG=1.

**Workaround**
Do not use repeated start condition instead set the stop condition UCTXSTP=1 in the NACK ISR prior to the following start condition (USTXSTT=1).

USCI23  
**USCI Module**

**Function**
UART transmit mode with automatic baud rate detection

**Description**
Erroneous behavior of the USCI_A can occur when configured in UART transmit mode with automatic baud rate detection. During transmission if a “Transmit break” is initiated (UCTXBRK=1), the USCI_A will not deliver a stop bit of logic high, instead, it will send a logic low during the subsequent synch period.

**Workaround**
1) Follow User’s Guide instructions for transmitting a break/synch field following UCWRST=1.

Or,

2) Set UCTXBRK=1 before an active transmission, i.e. check for bit UCBUSY=0 and then set UCTXBRK=1.

USCI24  
**USCI Module**

**Function**
Incorrect baud rate information during UART automatic baud rate detection mode

**Description**
Erroneous behavior of the USCI_A can occur when configured in UART mode with automatic baud rate detection. After automatic baud rate measurement is complete, the UART updates UCAxBR0 and UCAxBR1. Under Oversampling mode (UCOS16=1), for baud rates that should result in UCAxBRx=0x0002, the UART incorrectly reports it as UCAxBRx=0x5555.

**Workaround**
When break/synch is detected following the automatic baud rate detection, the flag UCBRK flag is set to 1. Check if UCAxBRx=0x5555 and correct it to 0x0002.

USCI25  
**USCI Module**

**Function**
TXIFG is not reset when NACK is received in I2C mode

**Description**
When the USCI_B module is configured as an I2C master transmitter the TXIFG is not reset after a NACK is received if the master is configured to send a restart (UCTXSTT=1 & UCTXSTP=0).
### Workaround

- **USCI26**
  - **USCI Module**: Tbuf parameter violation in I2C multi-master mode
  - **Function**: In multi-master I2C systems the timing parameter Tbuf (bus free time between a stop condition and the following start) is not guaranteed to match the I2C specification of 4.7us in standard mode and 1.3us in fast mode. If the UCTXSTT bit is set during a running I2C transaction, the USCI module waits and issues the start condition on bus release causing the violation to occur.
  - **Description**: Note: It is recommended to check if UCBBUSY bit is cleared before setting UCTXSTT=1.
  - **Workaround**: None

- **USCI27**
  - **USCI Module**: Timing of USCI I2C interrupts may cause device reset due to automatic clear of an IFG.
  - **Function**: When certain USCI I2C interrupt flags (IFG) are set and an automatic flag-clearing event on the I2C bus occurs, the program counter may become corrupted. This will only happen when the IFG is cleared within a critical time window (~6 CPU clock cycles) after a USCI interrupt request occurs and before the interrupt servicing is initiated. The affected interrupts are UCBxTXIFG, UCSTPIFG, UCSTTIFG and UCNACKIFG.
  - **Description**: The automatic flag-clearing scenarios are described in the following situations:
    1) A pending UCBxTXIFG interrupt request is cleared on the falling SCL clock edge following a NACK.
    2) A pending UCSTPIFG, UCSTTIFG, or UCNACKIFG interrupt request is cleared by a following Start condition.
  - **Workaround**: (1) Polling the affected flags instead of enabling the interrupts.
  - (2) Ensuring the above mentioned flag-clearing events occur after a time delay of 6 CPU clock cycles has elapsed since the interrupt request occurred and was accepted.

- **USCI30**
  - **USCI Module**: I2C mode master receiver / slave receiver
  - **Function**: When the USCI I2C module is configured as a receiver (master or slave), it performs a double-buffered receive operation. In a transaction of two bytes, once the first byte is moved from the receive shift register to the receive buffer the byte is acknowledged and the state machine allows the reception of the next byte.
  - **Description**: If the receive buffer has not been cleared of its contents by reading the UCBxRXBUF register while the 7th bit of the following data byte is being received, an error condition may occur on the I2C bus. Depending on the USCI configuration the following may occur:
    1) If the USCI is configured as an I2C master receiver, an unintentional repeated start condition can be triggered or the master switches into an idle state (I2C communication aborted). The reception of the current data byte is not successful in this case.
    2) If the USCI is configured as I2C slave receiver, the slave can switch to an idle state
stalling I2C communication. The reception of the current data byte is not successful in this case. The USCI I2C state machine will notify the master of the aborted reception with a NACK.

Note that the error condition described above occurs only within a limited window of the 7th bit of the current byte being received. If the receive buffer is read outside of this window (before or after), then the error condition will not occur.

Workaround

a) The error condition can be avoided altogether by servicing the UCBxRXIFG in a timely manner. This can be done by (a) servicing the interrupt and ensuring UCBxRXBUF is read promptly or (b) Using the DMA to automatically read bytes from receive buffer upon UCBxRXIFG being set.

OR

b) In case the receive buffer cannot be read out in time, test the I2C clock line before the UCBxRXBUF is read out to ensure that the critical window has elapsed. This is done by checking if the clock line low status indicator bit UCSCLLOW is set for atleast three USCI bit clock cycles i.e. 3 x t(BitClock).

Note that the last byte of the transaction must be read directly from UCBxRXBUF. For all other bytes follow the workaround:

Code flow for workaround

1. Enter RX ISR for reading receiving bytes
2. Check if UCSCLLOW.UCBxSTAT == 1
3. If no, repeat step 2 until set
4. If yes, repeat step 2 for a time period > 3 x t (BitClock) where t (BitClock) = 1/ f (BitClock)
5. If window of 3 x t(BitClock) cycles has elapsed, it is safe to read UCBxRXBUF
| **Description** | 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. |
| **Workaround** | Please refer to “XOSC8 Guidance” found at [SLAA423](https://www.ti.com) for information regarding working with this erratum. |
4 Document Revision History

Changes from family erratasheet to device specific erratasheet.
1. Errata FLASH26 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. 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. Errata TB24 was added to the errata documentation.

Changes from document Revision D to Revision E.
1. Errata USCI35 was added to the errata documentation.

Changes from document Revision E to Revision F.
1. Errata BCL16 was added to the errata documentation.
IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information 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.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have not been so designated is solely at the Buyer’s risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products

Audio
www.ti.com/audio

Amplifiers
amplifier.ti.com

Data Converters
dataconverter.ti.com

DLP® Products
www.dlp.com

DSP
dsp.ti.com

Clocks and Timers
www.ti.com/clocks

Interface
interface.ti.com

Logic
logic.ti.com

Power Mgmt
power.ti.com

Microcontrollers
microcontroller.ti.com

RFID
www.ti-rfid.com

OMAP Applications Processors
www.ti.com/omap

Wireless Connectivity
www.ti.com/wirelessconnectivity

Applications

Automotive and Transportation
www.ti.com/automotive

Communications and Telecom
www.ti.com/communications

Computers and Peripherals
www.ti.com/computers

Consumer Electronics
www.ti.com/consumer-apps

Energy and Lighting
www.ti.com/energy

Industrial
www.ti.com/industrial

Medical
www.ti.com/medical

Security
www.ti.com/security

Space, Avionics and Defense
www.ti.com/space-avionics-defense

Video and Imaging
www.ti.com/video

TI E2E Community
e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2013, Texas Instruments Incorporated