

## **MSP430FG461x Device Erratasheet**

### **1 Current Version**

Devices	Rev:	ADC18	ADC25	CPU8	CPU16	CPU19	DMA3	DMA4	FLL3	FLL6	LCDA5	RTC1	TA12	TA16	TA18	TAB22	TB2	TB16	TB18	USCI19	USCI20	USCI21	
MSP430FG4616	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430FG4617	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430FG4618	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430FG4619	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Devices	Rev:	USCI22	USCI23	USCI24	USCI25	USCI26	USCI27	WDG2	XOSC5	XOSC8	XOSC9
MSP430FG4616	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430FG4617	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430FG4618	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430FG4619	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

See [Appendix A](#) for prior revisions.

✓ The checkmark means that the issue is present in that revision

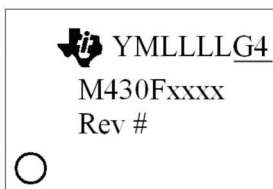
## 2 Package Markings

### PZ100

#### LQFP (PZ) 100 Pin



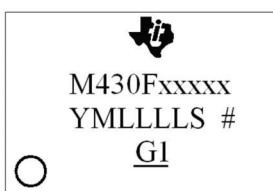
YM = Year and Month Date Code  
 LLLL = LOT Trace Code  
 S = Assembly Site Code  
 # = DIE Revision  
 o = PIN 1



YM = Year and Month Date Code  
 LLLL = LOT Trace Code  
 S = Assembly Site Code  
 # = DIE Revision  
 o = PIN 1

### ZQW113

#### BGA (ZQW), 113 pin



YM = Year and Month Date Code  
 LLLL = LOT Trace Code  
 S = Assembly Site Code  
 # = DIE Revision  
 o = PIN 1

### 3 Detailed Bug Description

<b>ADC18</b>	<b>ADC12 Module</b>
<b>Function</b>	Incorrect conversion result in extended sample mode
<b>Description</b>	<p>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 &gt; 0), and one of the following two conditions is true:</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>or</li> <li>• 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.</li> </ul>
<b>Workaround</b>	<ul style="list-style-type: none"> <li>• Use the pulse sample mode (SHP = 1).</li> <li>or</li> <li>• Use the ADC12 internal oscillator as the ADC12 clock source.</li> <li>or</li> <li>• Limit the undivided ADC12 input clock frequency to 3.15 MHz.</li> <li>or</li> <li>• 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.</li> </ul>
<b>ADC25</b>	<b>ADC12 Module</b>
<b>Function</b>	Write to ADC12CTL0 triggers ADC12 when CONSEQ = 00
<b>Description</b>	<p>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.</p>
<b>Workaround</b>	<p>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.</p>
<b>CPU8</b>	<b>CPU Module</b>
<b>Function</b>	Using odd values in the SP register
<b>Description</b>	<p>The SP can be written with odd values. In the original CPU, an odd SP value could be combined with an odd offset (for example, <code>mov. #value, 5(SP)</code>). In the new CPU, the SP can be written with an odd value, but the first time the SP is used, the LSB is forced to 0.</p>
<b>Workaround</b>	Do not use odd values with the SP.

**CPU16**
***CPU Module***
**Function**

Indexed addressing with instructions `calla`, `movb`, and `bra`
**Description**

With indexed addressing mode and instructions `calla`, `movb`, and `bra`, it is not possible to reach memory above 64k if the register content is < 64k.

Example: Assume R5 = FFFEh. The instruction `calla 0004h(R5)` results in a 20-bit call of address 0002h instead of 10002h.

**Workaround**

- Use different addressing mode to reach memory above 64k.
- First use `adda [index], [Rx]` to calculate address in upper memory and then use `calla [Rx]`.

**CPU19**
***CPU Module***
**Function**

CPUOFF can change register values

**Description**

If a CPUOFF command is followed by an instruction with an indirect addressed operand (for example, `mov @R8, R9`, and `RET`), an unintentional register-read operation can occur during the wakeup of the CPU. If the unintentional read occurs to a read-sensitive register (for example, UCB0RXBUF or TAIV), which changes its value or the value of other registers (IFGs), the bug leads to lost interrupts or wrong register read values.

**Workaround**

Insert a NOP instruction after each CPUOFF instruction.

**DMA3**
***DMA Module***
**Function**

Read-modify-write instructions may corrupt DMA address registers

**Description**

When a 16-bit-wide read-modify-write instruction (such as `add.w` and `sub.w`) is directly used on a DMA address register (DMAxSA or DMAxDA), the register contents are corrupted.

**Workaround**

- Do not use 16-bit-wide read-modify-write instructions on DMA address registers. Instead, if address calculations are necessary, do the calculations first, and then assign the result to the DMA address registers.  
or
- Use 20-bit-wide read-modify-write instructions (such as `addx.a`, `subx.a`) on the DMA address registers, if needed.

<b>DMA4</b>	<b><i>DMA Module</i></b>
<b>Function</b>	Corrupted write access to 20-bit DMA registers
<b>Description</b>	When a 20-bit wide write to a DMA address register (DMAxSA or DMAxDA) is interrupted by a DMA transfer, the register contents may be unpredictable.
<b>Workaround</b>	<ul style="list-style-type: none"> <li>• Design the application to ensure that no DMA access interrupts 20-bit wide accesses to the DMA address registers. or</li> <li>• When accessing the DMA address registers, enable the Read Modify Write disable bit (DMARMWDIS = 1) or temporarily disable all active DMA channels (DMAEN = 0). or</li> <li>• Use word access for accessing the DMA address registers. Note that this limits the values that can be written to the address registers to 16-bit values (lower 64K of Flash).</li> </ul>
<b>FLL3</b>	<b><i>FLL+ Module</i></b>
<b>Function</b>	FLLDx = 11 for /8 may generate an unstable MCLK frequency
<b>Description</b>	When setting the FLL to higher frequencies using FLLDx = 11 (/8), the output frequency of the FLL may have a larger frequency variation (for example, averaged over 2 seconds) and a lower average output frequency than expected when compared to the other FLLDx bit settings.
<b>Workaround</b>	None
<b>FLL6</b>	<b><i>FLL+ Module</i></b>
<b>Function</b>	LFXT1DIG readout is wrong
<b>Description</b>	The LFXT1DIG bit is always read as 0, even when the bit has been programmed as 1. The clock bypass implementation is functional, but the bit is always read as 0.
<b>Workaround</b>	None
<b>LCDA5</b>	<b><i>LCD_A Module</i></b>
<b>Function</b>	Wrong cycle time for first cycle of COMx/Sx signals
<b>Description</b>	The time of the first cycle of COMx/Sx signals after enabling the LCD_A module is only half of the selected value. All following cycles are correct.
<b>Workaround</b>	Not required, because it does not influence the LCD function.

**RTC1**
***RTC Module***
**Function**

Incorrect RTCDAY count in BCD mode

**Description**

When using the RTC in BCD mode, RTCDAY counts from 0x29 to 0x31 instead of counting to 0x30 in the month of December (RTCMON = 0x12). Furthermore, due to a malfunction in the leap year detection logic, RTCMON/RTCDAY may incorrectly count from 0x02/0x28 to 0x03/0x01 instead of 0x02/0x29, or it may incorrectly count from 0x02/0x28 to 0x02/0x29 instead of 0x03/0x01.

**Workaround**

Do not operate the RTC module in BCD mode. Use the RTC in hexadecimal format mode (RTCB CD = 0) instead. Convert RTC registers to BCD on demand using software.

---

**NOTE:** The CPU instruction DADD.B/.W can be used to efficiently implement a hex to BCD conversion. An Assembly language example of such an optimized 8-bit conversion is shown below:

---

```

mov.b   #8,R14 // Loop counter, process 8 bits
clr.b   R12    // Result will get assembled in R12
loop   rlc.b   R13 // Get MSB from input variable in R13
      dadd.b   R12,R12
      dec.b   R14
      jnz    loop

```

**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 is lost.

**Workaround**

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

**TA16**
***Timer\_A Module***
**Function**

First increment of TAR erroneous when IDx &gt; 00

**Description**

The first increment of TAR after any timer clear event (POR/TACLK) happens immediately following the first positive edge of the selected clock source (INCLK, SMCLK, ACLK, or TACLK). This is independent of the clock input divider settings (ID0, ID1). All following TAR increments are performed correctly with the selected IDx settings.

**Workaround**

None

<b>TA18</b>	<b><i>Timer_A Module</i></b>
<b>Function</b>	MOV to TACTL may clear TAR
<b>Description</b>	When TACTL is modified with a MOV instruction, the contents of TAR may be cleared, even when TACLK is not set.
<b>Workaround</b>	Use BIS or BIC instructions to modify TACTL.
	<hr/> <p><b>NOTE:</b> A DMA transfer must not occur while these BIS and BIC instructions execute. This can be prevented by disabling the DMA prior to these instructions, or by using the DMAONFETCH bit to align DMA transfers to instruction fetch boundaries.</p> <hr/>
<b>TAB22</b>	<b><i>Timer_A/Timer_B Module</i></b>
<b>Function</b>	Timer_A/B register modification after Watchdog Timer PUC
<b>Description</b>	Unwanted modification of the Timer_A/B registers TACTL and TAIV can occur when a PUC is generated by the Watchdog Timer (WDT) in watchdog mode and any Timer_A/B counter register TACCRx/TBCCRx is incremented/decremented (Timer_A/B does not need to be running).
<b>Workaround</b>	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><b>Example code:</b></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>
<b>TB2</b>	<b><i>Timer_B Module</i></b>
<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.

<b>TB16</b>	<b><i>Timer_B Module</i></b>
<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
<b>TB18</b>	<b><i>Timer_B Module</i></b>
<b>Function</b>	MOV to TBCTL may clear TBR
<b>Description</b>	When TBCTL is modified with a MOV instruction, the contents of TBR may be cleared, even when TBCLR is not set.
<b>Workaround</b>	Use BIS or BIC instructions to modify TBCTL.
	<hr/> <p><b>NOTE:</b> A DMA transfer must not occur while these BIS and BIC instructions execute. This can be prevented by disabling the DMA prior to these instructions or by using the DMAONFETCH bit to align DMA transfers to instruction fetch boundaries.</p> <hr/>
<b>USCI19</b>	<b><i>USCI Module</i></b>
<b>Function</b>	LPM4 may affect USCI operation
<b>Description</b>	When SMCLK is used as the USCI clock source, and SMCLK is deactivated due to a LPM4 entry, ongoing SPI master, I <sup>2</sup> C master, and UART transmit transaction are interrupted. Also, while in LPM4, UART receive operation is nonfunctional.
<b>Workaround</b>	Do not enter LPM4 while SPI master, I <sup>2</sup> C master, or UART transmit operations are active. Wait for the operation to be completed prior entering LPM4, or enter a different low-power mode. Also, do not use LPM4 during UART receive operation. Instead, use a different low-power mode.

**USCI20**
***USCI Module***
**Function**

I<sup>2</sup>C mode multi-master transmitter issue

**Description**

When configured for I<sup>2</sup>C master-transmitter mode and used in a multi-master environment, the USCI module can cause unpredictable bus behavior if all of the following 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 is released.  
and
4. The transmit buffer has not been loaded before the other master continues communication by driving SCL low.

The USCI remains in the SCL high phase until the transmit buffer is written. After the transmit buffer has been written, the USCI interferes with the current bus activity and may cause unpredictable bus behavior.

**Workaround**

- Ensure that slave does not stretch the SCL low phase of an ACK period.  
or
- Ensure that the transmit buffer is loaded in time.  
or
- Do not use the multi-master transmitter mode.

**USCI21**
***USCI Module***
**Function**

UART IrDA receiver mode

**Description**

IrDA reception does not function correctly at certain baud rates. This occurs only when the IrDA receive filter via the UCAXIRRCTL register is enabled and the USCI source clock (BRCLK) is higher than 6 MHz.

**Workaround**

1. Set the filter length (UCIRRFLx in UCAXIRRCTL) to the maximum value 0x3F to achieve proper functionality.
2. Reduce the frequency of the USCI source clock (BRCLK) to the IrDA module to be less than 6 MHz.

<b>USCI22</b>	<b><i>USCI Module</i></b>
<b>Function</b>	I <sup>2</sup> C master receiver with 10-bit slave addressing
<b>Description</b>	<p>Unexpected behavior of the USCI_B can occur when configured in I<sup>2</sup>C master receive mode with 10-bit slave addressing under the following conditions:</p> <ol style="list-style-type: none"> <li>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.</li> <li>2. Master sends a repeat start condition (if UCTXSTT = 1).</li> <li>3. The first address byte following the repeated start is acknowledged.</li> </ol> <p>However, the second address byte is not sent; instead, the master incorrectly starts to receive data and sets UCBxRXIFG = 1.</p>
<b>Workaround</b>	Do not use a repeated start condition; instead, set the stop condition UCTXSTP = 1 in the NACK ISR prior to the following start condition (USTXSTT = 1).
<b>USCI23</b>	<b><i>USCI Module</i></b>
<b>Function</b>	UART transmit mode with automatic baud rate detection
<b>Description</b>	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 does not deliver a stop bit of logic high; instead, it sends a logic low during the subsequent synch period.
<b>Workaround</b>	<ul style="list-style-type: none"> <li>• Follow user's guide instructions for transmitting a break/synch field following UCSWRST = 1.</li> <li>or</li> <li>• Set UCTXBRK = 1 before an active transmission; that is, check for bit UCBUSY = 0 and then set UCTXBRK = 1.</li> </ul>
<b>USCI24</b>	<b><i>USCI Module</i></b>
<b>Function</b>	Incorrect baud rate information during UART automatic baud rate detection mode
<b>Description</b>	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 UCxBR0 and UCxBR1. Under oversampling mode (UCOS16 = 1), for baud rates that should result in UCxBRx = 0x0002, the UART incorrectly reports it as UCxBRx = 0x5555.
<b>Workaround</b>	When break/synch is detected following the automatic baud rate detection, the flag UCBRK flag is set to 1. Check if UCxBRx = 0x5555 and correct it to 0x0002.
<b>USCI25</b>	<b><i>USCI Module</i></b>
<b>Function</b>	TXIFG is not reset when NACK is received in I <sup>2</sup> C mode
<b>Description</b>	When the USCI_B module is configured as an I <sup>2</sup> C master transmitter, the TXIFG is not reset after a NACK is received if the master is configured to send a restart (UCTXSTT = 1 and UCTXSTP = 0).
<b>Workaround</b>	Reset TXIFG in software within the NACKIFG interrupt service routine.

<b>USCI26</b>	<b><i>USCI Module</i></b>
<b>Function</b>	$t_{buf}$ parameter violation in I <sup>2</sup> C multi-master mode
<b>Description</b>	<p>In multi-master I<sup>2</sup>C systems, the timing parameter <math>t_{buf}</math> (bus free time between a stop condition and the following start) is not ensured to match the I<sup>2</sup>C specification of 4.7 <math>\mu</math>s in standard mode and 1.3 <math>\mu</math>s in fast mode. If the UCTXSTT bit is set during a running I<sup>2</sup>C transaction, the USCI module waits and issues the start condition on bus release, causing the violation to occur.</p> <hr/> <p><b>NOTE:</b> It is recommended to check if UCBBUSY bit is cleared before setting UCTXSTT = 1.</p> <hr/>
<b>Workaround</b>	None
<b>USCI27</b>	<b><i>USCI Module</i></b>
<b>Function</b>	Timing of USCI I <sup>2</sup> C interrupts may cause device reset due to automatic clear of an IFG.
<b>Description</b>	<p>When certain USCI I<sup>2</sup>C interrupt flags (IFG) are set and an automatic flag-clearing event on the I<sup>2</sup>C bus occurs, the program counter may become corrupted. This happens only 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.</p> <p>The automatic flag-clearing scenarios are described in the following situations:</p> <ul style="list-style-type: none"> <li>• A pending UCBxTXIFG interrupt request is cleared on the falling SCL clock edge following a NACK.</li> <li>• A pending UCSTPIFG, UCSTTIFG, or UCNACKIFG interrupt request is cleared by a following Start condition.</li> </ul>
<b>Workaround</b>	<ul style="list-style-type: none"> <li>• Polling the affected flags instead of enabling the interrupts or</li> <li>• 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.</li> </ul>
<b>WDG2</b>	<b><i>Watchdog Module</i></b>
<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 a correctly generated PUC.
<b>Workaround</b>	None
<b>XOSC5</b>	<b><i>LFXT1 Module</i></b>
<b>Function</b>	LF crystal failures may not be properly detected by the oscillator fault circuitry
<b>Description</b>	The oscillator fault error detection of the LFXT1 oscillator in low-frequency mode (XTS = 0) may not work reliably, causing a failing crystal to go undetected by the CPU; that is, OFIFG is not set.
<b>Workaround</b>	None

**XOSC8**
***LFXT1 Module***
**Function**

ACLK failure when crystal ESR is below 40 kΩ

**Description**

When ACLK is sourced by a low-frequency crystal with an ESR below 40 kΩ, the duty cycle of ACLK may fall below the specification; the OFIFG may become set or, in some instances, ACLK may stop completely.

**Workaround**

See the application report *XOSC8 Guidance* ([SLAA423](#)) for information regarding working with this erratum.

**XOSC9**
***LFXT1 Module***
**Function**

XT1 oscillator may not function as expected in high-frequency (HF) mode

**Description**

XT1 oscillator does not work correctly in high-frequency mode at supply voltages below 2 V with crystal frequency > 4 MHz.

**Workaround**

None. When XT1 oscillator is used in HF mode with crystal frequency > 4 MHz, ensure a supply voltage > 2.2 V.

## Appendix A Prior Revisions

Devices	Rev:	ADC18	ADC25	CPU8	CPU16	CPU19	DMA3	DMA4	FLL3	FLL6	LCDA5	RTC1	TA12	TA16	TA18	TAB22	TB2	TB16	TB18	USCI16	USCI19	USCI20
MSP430FG4616	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
	F	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430FG4617	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
	F	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430FG4618	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
	F	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430FG4619	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
	F	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Devices	Rev:	USCI21	USCI22	USCI23	USCI24	USCI25	USCI26	USCI27	WDG2	XOSC5	XOSC8	XOSC9
MSP430FG4616	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	F	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430FG4617	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	F	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430FG4618	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	F	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430FG4619	G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	F	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

✓ The checkmark means that the issue is present in that revision

## **A.1 Detailed Bug Description**

<b>USCI16</b>	<b>USCI Module</b>
<b>Function</b>	UART/IrDA mode lost characters
<b>Description</b>	<p>When configured for UART/IrDA mode, the USCI baud rate generator may halt operation under the following conditions:</p> <ul style="list-style-type: none"> <li>• IrDA mode: Repeated invalid start bits on the receive line or</li> <li>• 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 if UCSPB = 1) with a pulse width that passes the deglitch filter but is shorter than one-half of a bit time.</li> </ul> <p>After halting, additional characters are ignored. Transmit functionality is not affected.</p>
<b>Workaround</b>	<p>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 reenale the USCI interrupts.</p>

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

<b>Products</b>		<b>Applications</b>	
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>	Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>	Automotive	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>	Communications and Telecom	<a href="http://www.ti.com/communications">www.ti.com/communications</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>	Computers and Peripherals	<a href="http://www.ti.com/computers">www.ti.com/computers</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>	Consumer Electronics	<a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>	Energy	<a href="http://www.ti.com/energy">www.ti.com/energy</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>	Industrial	<a href="http://www.ti.com/industrial">www.ti.com/industrial</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>	Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>	Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>	Space, Avionics & Defense	<a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a>
RF/IF and ZigBee® Solutions	<a href="http://www.ti.com/lprf">www.ti.com/lprf</a>	Video and Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>
		Wireless	<a href="http://www.ti.com/wireless-apps">www.ti.com/wireless-apps</a>

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2010, Texas Instruments Incorporated