MSP430F15x/16x/161x Device Erratasheet

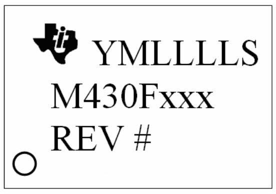
Current Version

Devices	Rev:	ADC18	BCL5	CPU4	I2C7	I2C8	I2C9	I2C10	I2C11	I2C12	I2C13	I2C14	TA12	TA16	TB2	TB16	US15	WDG2
MSP430F155	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F156	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F157	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F167	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F168	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F169	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F1610	B	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F1611	B	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F1612	B	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Note: See Appendix for prior revisions

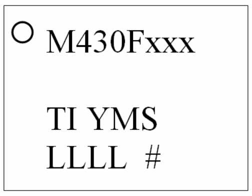
Package Markings

PM64: LQFP(PM) 64-pin



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

RTD64: QFN(RTD) 64-pin



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

Detailed Bug Description

ADC18 ADC18 - Bug description:

Module: ADC12, incorrect conversion result in extended sample mode

The ADC12 conversion result can be incorrect in the case where 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:

1.) 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
2.) The extended sample input signal SHI is synchronous to the clock source used for ADC12CLK and the undivided ADC12 input clock frequency exceeds 6.3 MHz.

Workaround:
1.) Use the pulse sample mode (SHP = 1)
Or
2.) Use the ADC12 internal oscillator as the ADC12 clock source
Or
3.) Limit the undivided ADC12 input clock frequency to 3.15 MHz
Or
4.) Use the same clock source (such as ACLK or SMCLK) to derive both SHI and ADC12CLK in order to achieve synchronous operation, and also limit the undivided ADC12 input clock frequency to 6.3 MHz

BCL5 BCL5 – Bug description:

Module: Basic Clock, Function: RSELx bit modifications can generate high frequency spikes on MCLK

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. Once the RSELx bits are configured as desired, the DIVMx setting can be changed back to the original selection.

CPU4 CPU4 - Bug description:

Module: CPU, Function: PUSH #4, PUSH #8

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 implemented in assembler.
No fix planned.

Detailed Bug Description (continued)

I2C7 I2C7 - Bug description:

Module: USART (I2C Mode): ARDYIFG Interrupt flag generation can fail in I2C slave mode.

When the USART is configured for I2C mode (U0CTL.I2C, SYNC, and I2CEN are set) and the module is configured as an I2C slave (U0CTL.MST = 0), the ARDYIFG interrupt flag generation can fail, even when both the I2C stop condition is received and the receive buffer is empty.
This condition occurs when the I2C clock source selected by I2CSSELx is disabled by the Status Register (SR) control signals OSCOFF or SCG1.
In this configuration, the hardware clock activation is enabled by the I2C module. However, if RXRDYIFG is polled to determine data reception, the I2C hardware clock activation may be disabled before the ARDYIFG is generated.

Workaround:
Solution #1: Use interrupt service routines using the I2C interrupt vector generator feature (I2CIV) to handle all I2C interrupts.
Solution #2: After detection of I2C Own Address (OAIFG), the selected I2C clock source is enabled by clearing the OSCOFF or SCG1 Status Register (SR) bits. When the ARDYIFG is detected, the OSCOFF or SCG1 in the Status Register (SR) can be set to disable the clock source and return to the desired low power mode operation.
Solution #3: For slave only devices, it is normally not necessary to use ARDYIFG.

I2C8 I2C8 - Bug description:

Module: USART (I2C Mode): Master Transmitter transmits 0FFh continuously.

When the USART is configured for I2C mode (U0CTL.I2C, SYNC, and I2CEN are set) and the module is configured as an I2C master (U0CTL.MST = 1), and when in this case I2CNDAT is used to control the number of bytes to transmit, the possibility exists that the master state-machine can become corrupted and start sending 0FFh as data on the I2C bus. Specifically, this error can occur when a long delay occurs between the set of the I2CTXRDY interrupt flag and the loading of I2CDRB (I2CDRW).

Workaround:
After detection of the I2CTXRDY interrupt flag, verify that the I2CTXUDF bit in I2CDCTL is set before loading I2CDRB (I2CDRW).

Detailed Bug Description (continued)

I2C9 I2C9 - Bug description:

Module: USART (I2C Mode): Master Transmitter Repeat Mode I2CSTP setting error.

When the USART is configured for I2C mode (U0CTL.I2C, SYNC, and I2CEN are set) and the module is configured as an I2C master (U0CTL.MST = 1), and when in this case repeat mode operation is selected (I2CTCTL.I2CRM = 1), the timing of the I2CSTP bit can result in lost data or extra requested transmitted bytes.
Specifically, if interrupts are active during the following two cases:
1) During the time between the setting of the I2CSTP bit and loading of I2CDRB (I2CDRW).
2) For transmitting slave address only, during the time between checking for I2CSTT cleared and setting I2CSTP.
Note: In the above case #2, the SCL line will be held low until the I2CDRB (I2CDRW) is loaded and then shifted out.

Workaround:
Solution for case #1: disable all interrupts (DINT) before setting I2CSTP then re-enabling after loading of I2CDRB.
Solution for case #2: disable all interrupts (DINT) before setting I2CSTT bit then re-enabling after setting I2CSTP bit.

I2C10 I2C10 - Bug description:

Module: USART (I2C Mode): Master stop bit SCL low phase does not match I2CSCLL setting.

When the USART is configured for I2C mode (U0CTL.I2C, SYNC, and I2CEN are set) and the module is configured as an I2C master (U0CTL.MST = 1), the hardware control of the SCL low phase before stop generation is equal to a single I2CCLK period. This is particularly noticeable with large I2CSCLL settings or large I2CPSC settings.

Workaround:
No software workaround.

I2C11 I2C11 - Bug description:

Module: USART (I2C Mode): Master state machine requires reset before new sequence can proceed.

When the USART is configured for I2C mode (U0CTL.I2C, SYNC, and I2CEN are set) and the module is configured as an I2C master (U0CTL.MST = 1), the master state-machine does not properly reset between execution cycles.

Workaround:
Before starting the new master sequence, clear and then re-set the I2CEN bit in the U0CTL register.

```
    bic.b    #I2CEN,&U0CTL
    bis.b    #I2CEN,&U0CTL
```

Detailed Bug Description (continued)

I2C12 I2C12 - Bug description:

Module: USART (I2C Mode): Master/Slave looses data on reception (lost RXRDYIFG).

If the I2C data register I2CDRB (I2CDRW) is read the same moment as data is loaded from the internal I2C shift register into I2CRB (I2CDRW), the received data is lost and no corresponding receive ready interrupt (RXRDYIFG) is generated. Following RXRDYIFG interrupts will be processed, but the missed byte cannot be recovered.

Workaround:
Do not read the I2CDRB (I2CDRW) register while data is being loaded into this register. This can be ensured by reading this register in a timely manner. For this any of the following 3 methods could be used:
1) Handle RXRDYIFG events with all other interrupt sources being disabled.
2) Use the DMA for storing incoming I2C data. The DMA interrupt or ARDYIFG interrupt can be used to initiate further processing of received data.
3) Enable nested interrupts to allow immediate processing of RXRDYIFG interrupts. (Caution must be taken to avoid stack overflows).

I2C13 I2C13 - Bug description:

Module: USART (I2C Mode): Glitch on SCL between I2C communication cycles can corrupt the state machine in I2C master mode.

When the USART is configured for I2C communication (U0CTL.I2C, SYNC, and I2CEN are set) and the module is configured as an I2C master (U0CTL.MST = 1), the I2C module is automatically switched to slave mode following the I2C master’s generation of a stop condition. If SCL is then pulled low and released again, the following device behavior can be observed:

1) When SCL is pulled low after the stop condition is generated, but before ARDYIFG is set, ARDYIFG will never get set and ALIFG is set. SCL is released high. See Note 1.
2) When SCL is pulled low as ARDYIFG is set, ALIFG is set. SCL is released high. Subsequent communication can result in an immediate ALIFG generation. See Note 2.
3) When SCL is pulled low after ARDYIFG is set but before ARDYIFG is cleared, ALIFG will not get set, but SCL will be held low by the master. An SCL hang-up condition occurs. See Note 3.
4) When SCL is pulled low after ARDYIFG is cleared, the module operates as intended. The ALIFG flag will not get set, and SCL is released high.

Workaround:
Note 1. ALIFG must be processed. All data communication can be correct here.
Note 2. ALIFG must be processed. All data communication can be correct here. To avoid a second ALIFG, clear I2CEN and reset I2CEN before new communication begins.
Note 3. Clear I2CEN and reset I2CEN before new communication begins to clear the SCL hangup.

Detailed Bug Description (continued)

I2C14 I2C14 - Bug description:

Module: USART (I2C Mode): Master SCL phases don't match I2CSCLx settings.

When the USART is configured for I2C mode (U0CTL.I2C, SYNC, and I2CEN are set) and the module is used as an I2C master (U0CTL.MST = 1), the generated I2C shift clock (SCL) high and low phases may be one or more I2CIN clock periods longer than defined by I2CSCLH and I2CSCLL. High I2CIN frequencies, large external pull-up resistors, and a large capacitive bus loading on SCL increase the likelihood for this to occur.

Workaround:
If possible, use an I2CIN input frequency of 1 MHz or less. Additionally, use low-impedance I2C pull-up resistors, preferably in the lower single-digit k-Ohm range, and minimize capacitive load on SCL.

TA12 TA12 - Bug description:

Module: Timer_A, Function: Interrupt is lost (slow ACLK)

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 1 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 TA16 - Bug description:

Module: Timer_A, Function: First increment of TAR erroneous when IDx > 00

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

Detailed Bug Description (continued)

TB2 TB2 - Bug description:

Module: Timer_B, Interrupt is lost (slow ACLK)

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 gets lost.

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

TB16 TB16 - Bug description:

Module: Timer_B, Function: First increment of TBR erroneous when IDx > 00

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

US15 US15 - Bug description:

Module: USART0, USART1, Function: UART receive with two stop bits

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 WDG2 - Bug description:

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.

Workaround:
None

Appendix: Prior Versions

Devices	Rev:	ADC18	BCL5	CPU4	I2C7	I2C8	I2C9	I2C10	I2C11	I2C12	I2C13	I2C14	SVS2	TA12	TA16	TB2	TB16	US14	US15	WDG2	XOSC4
MSP430F155	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	
	D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	
	C	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
	B	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F156	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	
	D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	
	C	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
	B	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F157	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	
	D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	
	C	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
	B	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F167	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	
	D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	
	C	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
	B	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F168	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	
	D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	
	C	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
	B	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F169	E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	
	D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	
	C	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
	B	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F1610	B	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	
	A	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
MSP430F1611	B	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	
	A	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
MSP430F1612	B	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	
	A	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

Detailed Bug Description

SVS2 SVS2 - Bug description:

Module: SVS, DAC1: DAC1 overwrites an input of the SVS comparator

DAC1 overrides the input of the SVS comparator. This is caused by a conflict between SVS and DAC1 at Port 6.7. DAC1 is enabled when DAC12AMPx is > 0.

Workaround:
Do not enable DAC1 when SVS is used.

Detailed Bug Description (continued)

US14 US14 - Bug description:

Module: USART0, USART1, UART Mode: Lost character start edge

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

XOSC4 XOSC4 - Bug description:

Module: XT1: XT1 high frequency oscillator low power wake-up error

The XT1 high frequency oscillator wake-up from low power mode operation is not functional.

Workaround:
If using the XT1 high frequency oscillator circuitry (BCSCTL1.XTS = 1), the OSCOFF bit in the Status Register (SR) must always be 0.