Errata

MSPM0G351x, MSPM0G151x Microcontrollers

ABSTRACT

This document describes the known exceptions to the functional specifications (advisories).

Table of Contents

1 Functional Advisories

Advisories that affect the device operation, function, or parametrics.

✓ The check mark indicates that the issue is present in the specified revision.

Errata Number	Rev A (Prototype X-Marked Products)	Rev C
ADC_ERR_05 ADC Module	✓	✓
ADC_ERR_06 ADC Module	✓	✓
ADC_ERR_07 ADC Module	✓	✓
ADC_ERR_08 ADC Module	✓	✓
AES_ERR_01 AES Module	✓	✓
COMP_ERR_05 COMP Module	✓	✓
CPU_ERR_01 CPU Module	✓	✓
CPU_ERR_02 CPU Module	✓	✓
CPU_ERR_03 CPU Module	✓	✓
FLASH_ERR_01 FLASH Module	✓	✓
FLASH_ERR_02 FLASH Module	✓	✓
FLASH_ERR_04 FLASH Module	✓	✓
FLASH_ERR_05 FLASH Module	✓	✓
I2C_ERR_04 I2C Module	✓	✓
I2C_ERR_06 I2C Module	✓	✓
I2C_ERR_07 I2C Module	✓	✓
I2C_ERR_08 I2C Module	✓	✓
I2C_ERR_09 I2C Module	✓	✓
I2C_ERR_10 I2C Module	✓	✓
MATHACL_ERR_01 MATHACL Module	✓	✓
MATHACL_ERR_02 MATHACL Module	✓	✓

Errata Number	Rev A (Prototype X-Marked Products)	Rev C
PMCU_ERR_10 PMCU Module	✓	
PMCU_ERR_11 PMCU Module	✓	✓
PMCU_ERR_12 PMCU Module	✓	✓
PMCU_ERR_13 PMCU Module	✓	✓
RST_ERR_01 RST Module	✓	✓
RTC_ERR_01 RTC Module	✓	✓
SPI_ERR_04 SPI Module	✓	✓
SPI_ERR_05 SPI Module	✓	✓
SPI_ERR_06 SPI Module	✓	✓
SPI_ERR_07 SPI Module	✓	✓
SRAM_ERR_01 SRAM Module	✓	
SYSOSC_ERR_01 SYSOSC Module	✓	✓
SYSOSC_ERR_02 SYSOSC Module	✓	✓
TIMER_ERR_04 TIMG Module	✓	✓
TIMER_ERR_06 TIMA and TIMG Module	✓	✓
UART_ERR_01 UART Module	✓	✓
UART_ERR_02 UART Module	✓	✓
UART_ERR_04 UART Module	✓	✓
UART_ERR_05 UART Module	✓	✓
I2C_ERR_06 I2C Module	✓	✓
UART_ERR_07 UART Module	✓	✓
UART_ERR_08 UART Module	✓	✓
VREF_ERR_03 VREF Module	✓	✓

2 Preprogrammed Software Advisories

Advisories that affect factory-programmed software.

✓ The check mark indicates that the issue is present in the specified revision.

3 Debug Only Advisories

Advisories that affect only debug operation.

✓ The check mark indicates that the issue is present in the specified revision.

4 Fixed by Compiler Advisories

Advisories that are resolved by compiler workaround. Refer to each advisory for the IDE and compiler versions with a workaround.

✓ The check mark indicates that the issue is present in the specified revision.

5 Device Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all MSP MCU devices. Each MSP MCU commercial family member has one of two prefixes: MSP or XMS. These prefixes represent evolutionary stages of product development from engineering prototypes (XMS) through fully qualified production devices (MSP).

XMS – Experimental device that is not necessarily representative of the final device's electrical specifications

MSP – Fully qualified production device

Support tool naming prefixes:

X: Development-support product that has not yet completed Texas Instruments internal qualification testing.

null: Fully-qualified development-support product.

XMS devices and X development-support tools are shipped against the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

MSP devices have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

Predictions show that prototype devices (XMS) have a greater failure rate than the standard production devices. TI recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are to be used.

TI device nomenclature also includes a suffix with the device family name. This suffix indicates the temperature range, package type, and distribution format.

5.1 Device Symbolization and Revision Identification

The package diagrams below indicate the package symbolization scheme, and Table 5-1 defines the device revision to version ID mapping.

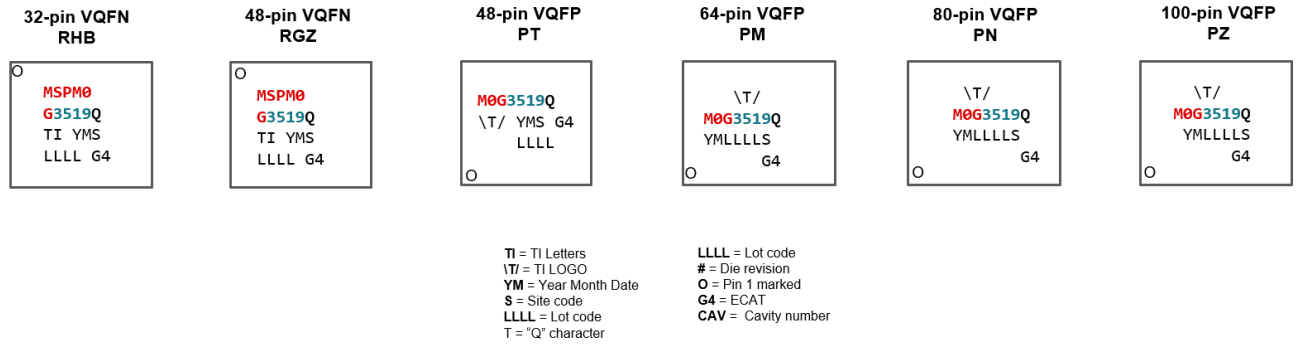

Figure 5-1. Package Symbolization

Table 5-1. Die Revisions

Revision Letter	Version (in the device factory constants memory)
A	1
C	2

The revision letter indicates the product hardware revision. Advisories in this document are marked as applicable or not applicable for a given device based on the revision letter. This letter maps to an integer stored in the memory of the device, which can be used to look up the revision using application software or a connected debug probe.

6 Advisory Descriptions

ADC_ERR_05 *ADC Module*

Category

Functional

Function

HW Event generated before enabling IP, ADC Trigger will stay in queue

Description

When ADC is configured in HW event trigger mode and the trigger is generated before enabling the ADC, the ADC trigger will stay in queue. Once ADC is enabled, it will trigger sampling and conversion.

Workaround

After configuring ADC in HW trigger mode, enable ADC first before giving external trigger.

ADC_ERR_06 *ADC Module*

Category

Functional

Function

ADC Output code jumps degrading DNL/INL specification

Description

The ADC may have errors at a rate as high as 1 in 25M conversions in 12-bit mode. When a conversion error occurs, it will be either +/-1LSB or +/- 64LSB random jump in the digital output of the ADC without a corresponding change in the ADC input voltage. The magnitude of this jump is larger near major transitions in the bit values of the ADC result (more bits transitioning from 1->0, or 0->1), and largest around midscale (2048 or 0x800). Depending on the application needs the best workaround may vary, but the following workarounds in software are proposed. Selection of the best workaround is left to the judgment of the system designer.

Workaround

Workaround 1: Upon ADC result outside of application threshold (via ADC Window Comparator or software thresholding), trigger or wait for another ADC result before making critical system decisions

Workaround 2: During post-processing, discard ADC values which are sufficiently far from the median or expected value. The expected value can be based on the average of real samples taken in the system, and the threshold for rejection can be based on the magnitude of the measured system noise.

Workaround 3: Use ADC sample averaging to minimize the effect of the results of any single incorrect conversion.

ADC_ERR_07 *ADC Module*

Category

Functional

Function

The conversion underflow flag UVIFG not set as expected under certain sequence

ADC_ERR_07
(continued)

ADC Module

Description

Scenario:
1. PWREN and configured adc in single mode.
2. Enabling ADC and waiting for at least 6usec.
3. Put SC bit high and wait for z number of cycles.
4. Read the busy bit and if busy bit is low, read ADC_SVT_MEMRES0 Register
5. Store the read value in sram

Observation:
1. If Optimization is high and z <11 cycles, UVIFG Flag getting set. Data Read is correct when z=10 since it takes few cycle, but z<10 gives data val =0 because MEMRES is not updated by this time when we are trying to read from MEMRES.
2. If Optimization is high and z >= 11, UVIFG Flag not getting set and data Read is correct
3. If Optimization is 0 and z >= 5, UVIFG Flag not getting set and data Read is correct

Workaround

No workarounds.

ADC_ERR_08

ADC Module

Category

Functional

Function

ADCMEMRES swap is seen when PA15 is toggling

Description

When ADC is configured in below settings:
1. ADC is configured in repeat sequence mode.
2. FIFO is disabled.
Set up condition:
1. ADC can use internal/external channels for which data needs to be converted.
2. PA15 is toggled from TB when IO is configured in input mode or PA15 is generating PWM when IO is configured in output mode.
Observation:
1. When the software is starting the ADC conversion, sometimes MEMRES data is being swapped. Means the data which needs to be in MEMRES0 is coming into MEMRES1, MEMRES1 data is coming into MEMRES2..so on.
Root cause:
The toggling signal on PA15 could affect the conversion clock of the ADC. And the increasing in frequency of conversion clock is making ADC to read out faster before the actual data comes then causes MEMRES 1 reads MEMRES0 sometimes and so on.

Workaround

Avoid input/output toggling signal on PA15.

AES_ERR_01

AES Module

Category

Functional

AES_ERR_01
(continued)

AES Module

Function

AES Saved Context Ready interrupt is not generating as expected

Description

Saved Context Ready interrupt is not getting generated. The interrupt is generated if an access (read or write) is made to any AES register.

Workaround

Use polling based mechanism to check the status bit for Saved Context Ready in CTRL register instead of interrupt.

COMP_ERR_05

COMP Module

Category

Functional

Function

Comparator output will set the rising and falling interrupt when comparator is enabled

Description

Comparator will set the rising and falling when the comparator is enabled.

Workaround

1. Clear the CPU interrupts by using ICLR bit.
ICLR will not work for clearing generic events. Follow below steps to clear COMP generic events (below are DriverLib functions, you can see the bits manipulation by looking in the function contents in our MSPM0 SDK)
a. Before COMP is enabled, configure COMP publisher with some dummy ID.
DL_COMP_setPublisherChanID(COMP_0_INST, 0); // remove the actual publisher
b. DL_COMP_enableEvent(COMP_0_INST, (DL_COMP_EVENT_OUTPUT_EDGE)); // Enable the COMP events in IMASK
c. DL_COMP_enable(COMP_0_INST); // Enable the COMP module, this step clearing the events in RIS.
d. DL_COMP_disableEvent(COMP_0_INST, (DL_COMP_EVENT_OUTPUT_EDGE));// Disable the COMP events by clearing in IMASK
e. DL_COMP_setPublisherChanID(COMP_0_INST, COMP_0_INST_PUB_CH); // configure the actual publisher
f. DL_COMP_enableEvent(COMP_0_INST, (DL_COMP_EVENT_OUTPUT_EDGE)); // Re-enable COMP events in IMASK

Or

Read the interrupt after the comparator is enabled, knowing that the first interrupt happened due to comparator being enabled.

CPU_ERR_01

CPU Module

Category

Functional

Function

CPU cache content can get corrupted

CPU_ERR_01 (continued) — ***CPU Module***

Description

Cache corruption can occur when switching between accessing Main flash memory, and other memory regions such as NONMAIN or Calibration data areas.

Workaround

Use the following procedure to access areas outside main memory safely:
1. Disable the cache by setting CTL.ICACHE to "0".
2. Perform needed access to memory area.
3. Re-enable cache by setting CTL.ICACHE to "1".

CPU_ERR_02 — ***CPU Module***

Category

Functional

Function

Limitation of disabling prefetch feature for CPUSS

Description

CPU prefetch disable will not take effect if there is a pending flash memory access.

Workaround

Disable prefetch, then issue a memory access to the shutdown memory (SHUTDNSTORE) in SYSCTL, this can be done with SYSCTL->SOCLOCK.SHUTDNSTORE0; After the memory access completes the prefetcher will be disabled.

CPU_ERR_03 — ***CPU Module***

Category

Functional

Function

Prefetcher can cause data integrity issues when transitioning into SLEEP mode

Description

When transitioning into SLEEP0 the prefetcher can erroneously fetch incorrect data (all 0's). When coming out of sleep mode, if the prefetcher and cache do not get overwritten by ISR code, then the main code execution from flash can get corrupted. For example, if the ISR is in the SRAM, then the incorrect data that was prefetched from Flash does not get overwritten. When the ISR returns the corrupted data in the prefetcher can be fetched by the CPU resulting in incorrect instructions.

Workaround

Disable prefetcher before entering SLEEP.

FLASH_ERR_01 — ***FLASH Module***

Category

Functional

Function

Access to FACTORY region will lead to hard fault with flash wait state equal to 2.

FLASH_ERR_01
(continued)

FLASH Module

Description

Access to FACTORY region when the flash wait state is set to 2 will trigger a hard fault. When MCLK is set to beyond 32MHz, the flash wait states needs to be 2.

Workaround

Set MCLK at a lower frequency(with flash wait state as 0 or 1) to access FACTORY region. Access the FACTORY region while the flash wait states is less than 2 (requires MCLK to be 32MHz or less). Cache the data in SRAM, MAIN flash, or DATA flash, if the application needs to access these values during run time. A typical value would be the Temperature Sensor's calibration value.

FLASH_ERR_02

FLASH Module

Category

Functional

Function

Debug disable in NONMAIN can be re-enable using default password

Description

If debug is disabled in NONMAIN configuration (DEBUGACCESS = 0x5566), the device can still be accessed using the default password.

Workaround

1. Set the DEBUGACCESS to Debug Enabled with Password option (DEBUGACCESS = 0xCCDD) and provide a unique password in the PWDDEBUGLOCK field. For higher security, it is recommended to have device-unique passwords that are cryptographically random. This would allow debug access with the right 128-bit password but can still allow some debug commands and access to the CFG-AP and SEC-AP.
2. Disable the physical SW Debug Port entirely by disabling SWDP_MODE. This fully prevents any debug access or requests to the device, but may affect Failure Analysis and return flows.

FLASH_ERR_04 *FLASH Module*

Category

Functional

Function

Wrong Address gets reported in the SYSCTL->DEDERRADDR

Description

When a FLASHDED error appears the data truncates the most significant byte. In the memory limits of the device, the most significant byte does not have an impact to the return address for MAIN flash. For NONMAIN flash or Factory region the MSB can be listed as 0x41xx.xxxx

Workaround

If the return address of the SYSCTL_DEDERRADDR returns a 0x00Cxxxxx, do an OR operation with 0x41000000 to get the proper address for the NONMAIN or Factory region return address. For example, if SYSCTL_DEDERRADDR = 0x00C4013C, the real address is 0x41C4013C.

For MAIN Flash DED, the SYSCTL_DEDERRADDR can be used as is.

FLASH_ERR_05 *FLASH Module*

Category

Functional

Function

DEDERRADDR can have incorrect reset value

Description

The reset value of the SYSCTL->DEDERRADDR can return a 0x00C4013C instead of the correct 0x00000000. The location of the error is in the Factory Trim region and is not indicative of a failure, it can be properly ignored. The reset value tends to change once NONMAIN has been programmed on the device.

Workaround

Accept 0x00C4013C as another reset value, so the default value from boot can be 0x00000000 or 0x00C4013C. The return value is outside of the range of the MAIN flash on the device so there is no potential of this return coming from an actual FLASH DED status.

I2C_ERR_04 *I2C Module*

Category

Functional

Function

When SCL is low and SDA is high the Target i2c is not able to release the stretch.

Description

1: SCL line grounded and released, device indefinitely pulls SCL low.

2: Post clock stretch, timeout, and release; if there is another clock low on the line, device indefinitely pulls SCL low.

Workaround

If the I2C target application does not require data reception in low power mode using Async fast clock request, disabling SWUEN by default is recommended, including during reset or power cycle. In this case, bug description 1 and 2 does not occur.

I2C_ERR_04
(continued)

I2C Module

If the I2C target application requires data reception in low power mode using Async fast clock request, enable SWUEN just before entering low power and clear SWUEN after low power exit. Even in this scenario, bug description 1 and 2 can occur when the I2C target is in low power, it will indefinitely stretch the SCL line if there is a continuous clock stretching or timeout caused by another device on the bus. To recover from this situation, enable the low timeout interrupt on the I2C target device, reset and re-initialize the I2C module within the low timeout ISR.

I2C_ERR_05

I2C Module

Category

Functional

Function

I2C SDA can get stuck to zero if we toggle ACTIVE bit during ongoing transaction

Description

If ACTIVE bit is toggled during an ongoing transfer, the state machine will be reset. However, the SDA and SCL output which is driven by the controller will not get reset. There is a situation where SDA is 0 and the controller has gone into IDLE state, here the controller won't be able to move forward from the IDLE state or update the SDA value. The target's BUSBUSY is set (toggling of the ACTIVE bit is leading to a start being detected on the line) and the BUSBUSY won't be cleared as the controller will not be able to drive a STOP to clear it.

Workaround

Do not toggle the ACTIVE bit during an ongoing transaction.

I2C_ERR_06

I2C Module

Category

Functional

Function

SMBus High timeout feature fails at I2C clock less than 24KHz onwards

Description

SMBus High timeout feature is failing at I2C clock rate less than 24KHz onwards (20KHz, 10KHz). From SMBUS Spec, the upper limit on SCL high time during active transaction is 50us. Total time taken from writing of START MMR bit to SCL low is 60us, which is >50us. It will trigger the timeout event and let the I2C controller goes into IDLE without completing the transaction at the start of transfer itself. Below is detailed explanation. For SCL is configured as 20KHz, SCL low and high period is 30us and 20us respectively. First, START MMR bit write at the same time high timeout counter starts decrementing. Then, it takes one SCL low period (30us) from START MMR bit write to SDA goes low (start condition). Next, it takes another SCL low period (30us) from SDA goes low (start condition) to SCL goes low (data transfer starts) which should stop the high timeout counter at this point. As a total, it takes 60us from counter start to end. However, due to the upper limit(50us) of the high timeout counter, the timeout event will still be triggered although the I2C transaction is working fine without issue.

Workaround

Do not use SMBus High timeout feature when I2C clock is less than 24KHz onwards.

I2C_ERR_07	*I2C Module*
Category	Functional
Function	Back to back controller control register writes will cause I2C to not start.
Description	Back-to-Back CTR register writes will cause the next CTR.START to not properly cause the start condition.
Workaround	Write all the CTR bits including CTR.START in a single write or wait between the CTR writes and CTR.START write.

I2C_ERR_08	*I2C Module*
Category	Functional
Function	FIFO Read directly after RXDONE interrupt causes erroneous data to be read
Description	When the RXDONE interrupt happens the FIFO is not always updated for the latest data.
Workaround	Wait 2 I2C CLK cycles for the FIFO to make sure to have the latest data. I2C CLK is based on the CLKSEL register in the I2C registers.

I2C_ERR_09	*I2C Module*
Category	Functional
Function	Start address match status might not be updated in time for a read through the ISR if running I2C at slow speeds.
Description	If running at atypical I2C speeds (less than 100kHz) then the ADDRMATCH bit (address match in the TSR register) might not be set in time for the read through an interrupt.
Workaround	If running at atypical I2C speeds, wait at least 1 I2C CLK cycle before reading the ADDRMATCH bit.

I2C_ERR_10	*I2C Module*
Category	Functional
Function	I2C Busy status is enabled preventing low power entry
Description	When in I2C Target mode, the I2C Busy Status stays high after a transaction if there is no STOP bit.

I2C_ERR_10 (continued)	*I2C Module*
Workaround	Program the I2C controller to send the STOP bit and don't send a NACK for the last byte. Terminate any I2C transfer with a STOP condition to maintain proper BUSY status and asynchronous clock request behavior (for low power mode reentry).

MATHACL_ERR_01	*MATHACL Module*
Category	Functional
Function	MATHACL status error bit does not get cleared
Description	If there is a status error generated by the mathacl (ex. divide by 0), then the status register never gets cleared.
Workaround	Reset the peripheral to clear the status bit.

MATHACL_ERR_02	*MATHACL Module*
Category	Functional
Function	MATHACL COS(-180) gives 1 instead of -1, SIN(-90) will give 1 instead of -1
Description	MATHACL will return a 1 instead of a -1 when performing COS(-180) or SIN(-90)
Workaround	No workaround, make the result negative in software.

PMCU_ERR_10	*PMCU Module*
Category	Functional
Function	VBOOST might have larger delay under certain operating conditions
Description	VBOOST for analog MUX has large delay at VDD<1.8V, which delays settling time of other modules like HFXT, COMP, SYSOSC(FCL-external R),OPA and GPAMP.
Workaround	Keep VDD>=1.8V and use VBOOST in ONALWAYS mode using GENCLKCFG[23:22]=0x2.

PMCU_ERR_11 *PMCU Module*

Category

Functional

Function

NRST<1sec pulse giving wrong rstcause in shutdown mode

Description

The rstcause value is wrong under the following condition. Though the expected rstcause is 0x05.
(i) Device is configured for shutdown mode
(ii) WFI() is called
(iii) Give NRST<1sec pulse to bring device out from shutdown mode

Workaround

No workaround.

PMCU_ERR_12 *PMCU Module*

Category

Functional

Function

BOR cold boot spec violation

Description

The BOR cold boot spec could be violated, the max spec at is 1.65V.

Workaround

No Workaround. While cold booting the device operate the device above 1.65V.

PMCU_ERR_13 *PMCU Module*

Category

Functional

Function

MCU may get stuck while waking up from STOP2 & STANDBY0 at certain scenario

Description

When there is a pending prefetch access before device transitions to STOP2 & STANDBY0. A pending prefetch access such as a timer has just run to completion and DMA has received the event from the GPIO, in that scenario neither DMA transfer happens nor timer ISR execution takes place as well as CPU gets stuck. This issue arises when WFI instruction is half word aligned, wait state of device is 2 and there is a pending prefetch access before device transitions to LPM.

Workaround

Before going to LPM, customer can disable the prefetch and run some dummy instructions (like shutdown register read or any peripheral read) which doesn't access prefetch so that prefetch can get disabled and doesn't cause the device to hang when waking up from LPM as no prefetch access will be pending.

RST_ERR_01 *RST Module*

Category

Functional

RST_ERR_01 (continued) — ***RST Module***

Function

NRST release doesn't get detected when LFCLK_IN is LFCLK source and LFCLK_IN gets disabled

Description

When LFCLK = LFCLK_IN and we disable the LFCLK_IN, then comes a corner scenario where NRST pulse edge detection is missed and the device doesn't come out of reset. This issue is seen if the NRST pulse width is below 608us. NRST pulse above 608us, the reset should appear normally.

Workaround

Keep the NRST pulse width higher than 608us to avoid this issue.

RTC_ERR_01 — ***RTC Module***

Category

Functional

Function

Some RTC Interrupts are not available in STANDBY1

Description

When in STANDBY1, the RTCRDY and RTC_PRESCALER1 interrupts cannot wakeup the device.

Workaround

When waking up the device from STANDBY1 with the RTC, use other available interrupts such as RTC_ALARM and RTC_PRESCALER0.

SPI_ERR_02 — ***SPI Module***

Category

Functional

Function

Missing SPI Clock and data bytes after wake-up from low power mode (LPM)

Description

After device wake-up from a low power state, the SPI module can not properly propagate the first few clock cycles and data bits of the first byte sent out.

Workaround

To maintain SPI data integrity after a wakeup, use the following sequence when entering and exiting LPMs:

1.Disable SPI module
2.Wait for Interrupt(WFI)- enter LPM
3.Wake up from LPM (any source).
4.Enable the SPI module.

SPI_ERR_04	*SPI Module*
Category	Functional
Function	IDLE/BUSY status toggle after each frame receive when SPI peripheral is in only receive mode.
Description	In case of SPI peripheral in only receive mode, the IDLE interrupt and BUSY status are toggling after each frame receive while SPI is receiving data continuously(SPI_PHASE=1). Here there is no data loaded into peripheral TXFIFO and TXFIFO is empty.
Workaround	Do not use SPI peripheral only receive mode. Set SPI peripheral in transmit and receive mode. You do not need to set any data in the TX FIFO for SPI.

SPI_ERR_05	*SPI Module*
Category	Functional
Function	SPI Peripheral Receive Timeout interrupt is setting irrespective of RXFIFO data
Description	When using the SPI timeout interrupt the RXTIMEOUT can continue decrementing even after the final SPI CLK is received, which can cause a false RXTIMEOUT.
Workaround	Disable the RXTIMEOUT after the last packet is received (this can be done in the ISR) and re-enable when SPI communication starts again.

SPI_ERR_06	*SPI Module*
Category	Functional
Function	IDLE/BUSY status does not reflect the correct status of SPI IP when debug halt is asserted
Description	IDLE/BUSY is independent of halt, it is only gating the RXFIFO/TXFIFO writing/reading strobes. So, if controller is sending data, although it's not latched in FIFO but the BUSY is getting set. The POCI line transmits the previously transmitted data on the line during halt
Workaround	Don't use IDLE/BUSY status when SPI IP is halted.

SPI_ERR_07	*SPI Module*
Category	Functional

SPI_ERR_07
(continued)

SPI Module

Function

SPI underflow event can not generate if read/write to TXFIFO happen at the same time for SPI peripheral

Description

When SPH = 0 and device is configured as the SPI peripheral: if there is a write to the TXFIFO WHILE there is a read request, then an underflow event can not be generated as the read/write request is happening simultaneously.

Workaround

Customer must verify that TXFIFO on peripheral can never be empty when the controller is addressing the peripheral. Additionally, data checking strategies, like CRC, can be used to verify the packets were sent properly.

SRAM_ERR_01

SRAM Module

Category

Functional

Function

SRAM Parity and ECC function is not supported on Rev A devices

Description

SRAM Parity and ECC function is not supported on Rev A devices. Please do not use SRAM Parity and ECC function on Rev A devices.

Workaround

None.

SYSOSC_ERR_01

SYSOSC Module

Category

Functional

Function

MFCLK drift when using SYSOSC FCL together with STOP1 mode

Description

If MFCLK is enabled AND SYSOSC is using the frequency correction loop (FCL) mode AND STOP1 low power operating mode is used, then the MFCLK may drift by two cycles when SYSOSC shifts from 4MHz back to 32MHz (either upon exit from STOP1 to RUN mode or upon an asynchronous fast clock request that forces SYSOSC to 32MHz).

Workaround

Use STOP0 mode instead of STOP1 mode, there is no MFCLK drift when STOP0 mode is used.

OR

Do not use SYSOSC in the FCL mode (leave FCL disabled) when using STOP1.

SYSOSC_ERR_02 *SYSOSC Module*

Category

Functional

Function

MFCLK does not work when Async clock request is received in an LPM where SYSOSC was disabled in FCL mode

Description

MFCLK will not start to toggle in below scenario:
1. FCL mode is enabled and then MFCLK is enabled
2. Enter a low power mode where SYSOSC is disabled (SLEEP2/STOP2/STANDBY0/ STANDBY1).
3. Async request is received from some peripherals which use MFCLK as functional clock. On receiving async request, SYSOSC gets enabled and ulpclk becomes 32MHz. But MFCLK is gated off and it does not toggle at all as the device is still set to the LPM.

Workaround

If SYSOSC is using the FCL mode - Do not enable the MFCLK for a peripheral when you're entering a LPM mode which would typically turn off the SYSOSC.

TIMER_ERR_04 *TIMG Module*

Category

Functional

Function

TIMER re-enable may be missed if done close to zero event

Description

When using a GP TIMER in one shot mode and CLKDIV.RATIO is not 0, TIMER re-enable may be missed if done close to zero event.

Workaround

TIMER can be disabled first before re-enabling.

TIMER_ERR_06 *TIMA and TIMG Module*

Category

Functional

Function

Writing 0 to CLKEN bit does not disable counter

Description

Writing 0 to the Counter Clock Control Register(CCLKCTL) Clock Enable bit(CLKEN) does not stop the timer.

Workaround

Stop the timer by writing 0 to the Counter Control(CTRCTL) Enable(EN) bit.

UART_ERR_01 *UART Module*

Category

Functional

UART_ERR_01 (continued) — ***UART Module***

Function

UART start condition not detected when transitioning to STANDBY1 Mode

Description

After servicing an asynchronous fast clock request that was initiated by a UART transmission while the device was in STANDBY1 mode, the device will return to STANDBY1 mode. If another UART transmission begins during the transition back to STANDBY1 mode, the data is not correctly detected and received by the device.

Workaround

Use STANDBY0 mode or higher low power mode when expecting repeated UART start conditions.

UART_ERR_02 — ***UART Module***

Category

Functional

Function

UART End of Transmission interrupt not set when only TXE is enabled

Description

UART End Of Transmission (EOT) interrupt does not trigger when the device is set for transmit only (CTL0.TXE = 1, CTL0.RXE = 0). EOT successfully triggers when device is set for transmit and receive (CTL0.TXE = 1, CTL0.RXE = 1)

Workaround

Set both CTL0.TXE and CTL0.RXE bits when utilizing the UART end of transmission interrupt. Note that you do not need to assign a pin as UART receive.

UART_ERR_04 — ***UART Module***

Category

Functional

Function

Incorrect UART data received with the fast clock request is disabled when clock transitions from SYSOSC to LFOSC

Description

Scenario:
1. LFCLK selected as functional clock for UART
2. Baud rate of 9600 configured with 3x oversampling
3. UART fast clock request has been disabled
If the ULPCLK changes from SYSOSC to LFOSC in the middle of a UART RX transfer, it is observed that one bit is read incorrectly

Workaround

Enable UART fast clock request while using UART in LPM modes.

UART_ERR_05 — ***UART Module***

Category

Functional

UART_ERR_05
(continued)

UART Module

Function

Limitation of debug halt feature in UART module

Description

All Tx FIFO elements are sent out before the communication comes to a halt against the expectation of completing the existing frame and halt.

Workaround

Please make sure data is not written into the TX FIFO after debug halt is asserted.

UART_ERR_06

UART Module

Category

Functional

Function

Unexpected behavior RTOUT/Busy/Async in UART 9-bit mode

Description

UART receive timeout (RTOUT) is not working correctly in multi node scenario, where one UART will act as controller and other UART nodes as peripherals , each peripheral is configured with different address in 9-bit UART mode.
First UART controller communicated with UART peripheral1, by sending peripheral1's address as a first byte and then data, peripheral1 has seen the address match and received the data. Once controller is done with peripheral1, peripheral1 is not setting the RTOUT after the configured timeout period, if controller immediately starts the communication with another UART peripheral (peripheral2) which is configured with different address on the bus. The peripheral1 RTOUT counter is resetting while communication ongoing with peripheral2 and peripheral1 setting it's RTOUT only after UART controller is completed the communication with peripheral2 .
Similar behavior observed with BUSY and Async request. Busy and Async request is setting even if address does not match while controller communicating with other peripheral on the bus.

Workaround

Do not use RTOUT/ BUSY /Async clock request behavior in multi node UART communication where single controller is tied to multiple peripherals.

UART_ERR_07

UART Module

Category

Functional

Function

RTOUT counter not counting as per expectation in IDLE LINE MODE

Description

In IDLE LINE MODE in UART, RTOUT counter gets stuck, even when the line is IDLE and FIFO has some elements. This means that RTOUT interrupts will not work in IDLE LINE MODE.
In case of an address mismatch, RTOUT counter is reloaded when it sees toggles on the Rx line.
In case of a multi-responder scenario this could lead to an indefinite delay in getting

UART_ERR_07 *UART Module*
(continued)

an RTOUT event when communication is happening between the commander and some other responder.

Workaround

Do not enable RTOUT feature when UART module is used either in IDLELINE mode/ multi-node UART application.

UART_ERR_08 *UART Module*

Category

Functional

Function

STAT BUSY does not represent the correct status of UART module

Description

STAT BUSY is staying high even if UART module is disabled and there is data available in TXFIFO.

Workaround

Poll TXFIFO status and the CTL0.ENABLE register bit to identify BUSY status.

VREF_ERR_03 *VREF Module*

Category

Functional

Function

Large inrush current from VDD to charge external 1uF cap.

Description

During power-up there is a large inrush current of ≅70mA from VDD to charge the external 1uF cap on VREF. If there are frequent power cycling of the VREF then damage cam occur.

Workaround

Pre-charge the VREF+ capacitor by using the GPIO before enabling VREF. This provides the capacitor is charged more than required. Then enable the VREF module and disable the GPIO pull-up.

7 Trademarks

All trademarks are the property of their respective owners.

8 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from March 30, 2025 to July 30, 2025 (from Revision A (March 2025) to Revision B (July 2025)) **Page**

- Added ADC_ERR_07, ADC_ERR_08, CPU_ERR_02, FLASH_ERR_02, PMCU_ERR_13, SPI_ERR_06, SPI_ERR_07, UART_ERR_04, UART_ERR_05, UART_ERR_06, UART_ERR_07 and UART_ERR_08. 1
- AES_ERR_01 Category was updated..... 5

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), 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, regulatory 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 or other applicable terms available either on 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.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265