Errata

MSPM0L111x Microcontrollers

ABSTRACT

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

Table of Contents

Trademarks

All trademarks are the property of their respective owners.

1 Functional Advisories

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

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

Errata Number	Rev A
ADC_ERR_05	✓
I2C_ERR_05	✓
I2C_ERR_06	✓
I2C_ERR_07	✓
I2C_ERR_08	✓
I2C_ERR_09	✓
I2C_ERR_10	✓
I2C_ERR_11	✓
I2C_ERR_13	✓
PMCU_ERR_11	✓
PMCU_ERR_13	✓
PMCU_ERR_10	✓
SPI_ERR_04	✓
SPI_ERR_05	✓
SPI_ERR_06	✓
SPI_ERR_07	✓
SYSOSC_ERR_01	✓
SYSOSC_ERR_02	✓
TIMER_ERR_04	✓

Errata Number	Rev A
TIMER_ERR_05	✓
TIMER_ERR_06	✓
TIMER_ERR_07	✓
UART_ERR_01	✓
UART_ERR_02	✓
UART_ERR_03	✓
UART_ERR_04	✓
UART_ERR_05	✓
UART_ERR_06	✓
UART_ERR_07	✓
UART_ERR_08	✓
UART_ERR_10	✓
UART_ERR_11	✓
FLASH_ERR_02	✓
FLASH_ERR_03	✓
FLASH_ERR_05	✓
FLASH_ERR_08	✓
SYSCTL_ERR_01	✓
SYSCTL_ERR_02	✓
SYSCTL_ERR_04	✓
CPU_ERR_02	✓
CPU_ERR_03	✓
AES_ERR_01	✓
KEYSTORE_ERR_01	✓
RST_ERR_01	✓

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.

Errata Number	Rev A
GPIO_ERR_03	✓

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. Please note that X marked development samples should be operated in -40°C to 85°C temperature range.

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.

6 Advisory Descriptions

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_13 — *PMCU Module*

Category	Functional
Function	MCU may get stuck while waking up from STOP2 & STANDBY0
Description	If prefetch access is pending when the device transitions to STOP2 or STANDBY, when the device wakes up, the pending prefetch can prevent the device from resuming normal execution. The errata occurs if the WFI instruction is not word aligned, and the flash wait state is 2. In such a case, neither a DMA transfer nor a pending interrupt will be serviced.
Workaround	User should disable prefetch and issue a shutdown store memory read, which prevents a new prefetch from issuing and allows a pending prefetch to complete.

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.

GPIO_ERR_03 *GPIO and DEBUGSS Module*

Category

Debug only

Function

On a debugger read to GPIO EVENT0 IIDX, interrupt is cleared.

Description

EVENT0's IIDX of GPIO, on a debugger read is treated as a CPU read and interrupt is getting cleared.

Workaround

In case GPIO interrupts and events are expected, do not read using IIDX register via the debugger. Instead, utilize a software read of RIS register.

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.

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 may get stuck to zero if we toggle ACTIVE bit during ongoing transaction

Description

If ACTIVE bit is toggled during an ongoing transfer, its state machine will be reset. However, the SDA and SCL output which is driven by the master will not get reset. There is a situation where SDA is 0 and master has gone into IDLE state, here the master won't be able to move forward from the IDLE state or update the SDA value. Slave'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 master 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 I2C Master 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 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.

I2C_ERR_07 (continued)

I2C Module

Workaround

Write all the CTR bits including CTR.START in a single write or wait one clock cycle 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 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 below 100kHz on I2C, 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.

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).

UNICOMMI2CC_ERR_01 *UNICOMMI2CC Module*

Category

Functional

Function

Polling the I2C BUSY bit might not guarantee that the controller transfer has completed

Description

After setting the BUSRTRUN/FRAME_START bit to initiate an I2C controller transfer, it takes approximately 2 I2C functional clock cycles for the BUSY status to be asserted. If polling for the BUSY bit is used immediately after setting BUSRTRUN/FRAME_START to wait for transfer completion, the BUSY status might be checked before it is set. This problem is more likely to occur with high CLKDIV values (resulting in a slower I2C functional clock) or under higher compiler optimization levels.

Workaround

Add software delay before polling BUSY status. Software delay = 3 x I2C functional clock = 3 x clock_divider x (CPU_CLK / selected clock source frequency) For example, with a clock_divider of 8, a clock source of 4 MHz(MFCLK), and CPU_CLK of 32 MHz: Software delay = 3 x 8 x (32 MHz / 4 MHz)= 192 CPU cycles

I2C_ERR_13 *I2C Module*

Category

Functional

Function

Polling the I2C BUSY bit might not guarantee that the controller transfer has completed

Description

After setting the CCTR.BURSTRUN bit to initiate an I2C controller transfer, it takes approximately 3 I2C functional clock cycles for the BUSY status to be asserted. If polling for the BUSY bit is used immediately after setting CCTR.BURSTRUN to wait for transfer completion, the BUSY status might be checked before it is set. This problem is more likely to occur with high CLKDIV values (resulting in a slower I2C functional clock) or under higher compiler optimization levels.

Workaround

Add software delay before polling BUSY status. Software delay = 3 x CPU CLK / I2C functional clock = 3 x CPU CLK / (CLKSEL / CLKDIV) For example, with a clock divider (CLKDIV) of 8, a clock source of 4 MHz(MFCLK), and CPU CLK of 32 MHz: Software delay = 3 x 32 MHz / (4 MHz/ 8)= 192 CPU cycles

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 receiving mode, the IDLE interrupt and
BUSY status are toggling after each frame receive while SPI is receiving data

SPI_ERR_04 (continued) ***SPI Module***

continuously(SPI_PHASE=1). Here there is no data loaded into peripheral(slave) TXFIFO and TXFIFO is empty.

Workaround

Do not use SPI peripheral only receive mode. Set SPI in peripheral(slave) simultaneous transmit and receive mode.

SPI_ERR_05 *SPI Module*

Category

Functional

Function

SPI Peripheral Receive Timeout interrupt is setting irrespective of RXFIFO data

Description

When using SPI timeout interrupt, the RXTIMEOUT counter started decrementing from the point that peripheral is stopped receiving SPI clock and setting the RXTIMEOUT interrupt irrespective of data exists in RXFIFO or not, which does not match the description in the TRM: SPI peripheral receive timeout(RTOUT) interrupt is "asserted when the receive FIFO is not empty, and no further data is received in the specified time at CTL1.RXTIMEOUT.

Workaround

Repeat load RXTIMEROUT counter value while receive FIFO is empty, and start timeout counting only when receive FIFO gets any data.

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

Function

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

SPI_ERR_07 (continued)

SPI Module

Description

When SPI.CTL0.SPH = 0 and the device is configured as the SPI peripheral.

If there is a write to the TXFIFO WHILE there is a read request from the SPI controller, then an underflow event may not be generated as the read/write request is happening simultaneously.

Workaround

Ensure the TXFIFO is not empty when the SPI Controller is addressing the device, this can be done by preloading data to avoid a write and read to the same TXFIFO address. Alternatively, data checking strategies, like CRC, can be used to verify the packets were sent properly, then the data can be resent if the CRC doesn't match.

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 the 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 fails to start up in certain conditions.

Description

MFCLK will not start to toggle in the 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/STANBY1).

3. Now asynchronous clock request is received from some peripheral which uses MFCLK as its functional clock.

Workaround

Avoid the above scenario - check that your system does not use FCL and MFCLK along with the listed low power modes while asynchronous fast clock requests are enabled.

TIMER_ERR_04 *TIMER Module*

Category

Functional

Function

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

Description

When using a GPTIMER 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 *Timer module*

Category

Functional

Function

Writing 0 to CLKEN does not disable the counter.

Description

Writing 0 to the Counter Clock Control (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.

TIMER_ERR_06 *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.

TIMER_ERR_07 *TIMG Module*

Category

Functional

Function

Initial repeat counter has 1 less period than next repeats

Description

When using the timer repeat counter mode, the first repeat will have 1 less count than the subsequent repeats because the following repeat counters will include the transition between 0 and the load value. For example if the TIMx.RCLD = 0x3 then 3 observable zero events would appear on the first repeat counter and 4 observable zero events would appear on the following repeat counter sequences.

Workaround

Set the initial RCLD value to 1 more than the expected RCLD, then in the ISR for the Repeat Counter Zero Event (REPC), set the RCLD to the intended RCLD value.

For example, if intending to have 4 repeats, set the initial RCLD value to RCLD = 0x5, then in the timer ISR for the REPC interrupt, set RCLD = 0x4. Now all timer repeats will have the same number of zero/load events.

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_03 — *UART Module*

Category

Functional

Function

UART RX interrupt erroneously set with 3x oversampling and MFCLK or BUSCLK as clock source

Description

When using BUSCLK or MFCLK for UART and 3x oversampling, the RXINT is getting set incorrectly. When using the UART module with BUSCLK or MFCLK as the source, and 3x oversampling mode, the RX interrupt may be set erroneously. Under these conditions TX data may also be corrupted.

Workaround

Use a higher oversampling rate when using BUSCLK or MFCLK. If 3x oversampling is required, utilize LFCLK.

UART_ERR_04 — *UART Module*

Category

Functional

UART_ERR_04 (continued)

UART Module

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

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.

UART_ERR_06 (continued)

UART Module

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 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.

UNICOMMUART_ERR_09

UNICOMMUART Module

Category

Functional

Function

ISO-7816 Smartcard Mode can't support 9600 baud rate with lower than 57MHz UARTCLK

Description

To achieve a 9600 baud rate in ISO-7816 Smartcard Mode, and considering the limitations below, a UARTCLK frequency exceeding 57 MHz is required. 1. The ISO-7816 standard dictates that 1 bit requires 372 clock cycles. 2. In MSPM0 ISO-7816 mode, the

UNICOMMUART_ERR_09 (continued) *UNICOMMUART Module*

oversampling rate (OVS) is fixed in the UART peripheral at 16x. Here is the minimum UARTCLK calculation: Required UARTCLK = 9600 * 372 * 16 = 57.139 MHz

Workaround

No

UART_ERR_11 *UART Module*

Category

Functional

Function

UART Receive timeout starts counting earlier than expected during the STOP bit transaction

Description

During the STOP bit transaction the Receive timeout will start counting in the middle of the STOP bit transaction, which can cause an unintended RTOUT interrupt if the RXTOSEL setting is too small. For example, if the baud rate was 1Mbps, and RXTOSEL was set to 1, the expected RTOUT should happen 1us after the STOP bit transaction, instead the RTOUT interrupt is getting set at 0.5 us.

Workaround

The UART.IFLS.RXTOSEL register selects the bit time before the Receive Time out (RTOUT) interrupt will fire. The RXTOSEL value needs to be greater than 1 in order to prevent an early interrupt. The receive timeout time can be calculated as: Receive timeout = (RXTOSEL - 0.5) / Baud Rate

FLASH_ERR_02 *FLASH Module*

Category

Functional

Function

Debug disable in NONMAIN can be re-enabled using the password

Description

If debug is disabled in NONMAIN configuration (DEBUGACCESS = 0x5566), the device can still be accessed using the password that is programmed (default password if not explicitly programmed).

Workaround

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.

Workaround 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_03 *FLASH Module*

Category

Functional

Function

Flash access with 2 wait states followed by invalid bootcode access will cause next flash access to also throw a violation

Description

Doing a Flash access followed by a access to BOOTCODE when you have 2 wait states will cause the next flash access to also cause a violation.

Workaround

Do not attempt to access boot-code region post-boot phase. Otherwise, there will need to be 4 clock cycles in between the bootcode violation and next correct flash access.

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.

FLASH_ERR_08 *FLASH Module*

Category

Functional

Function

Hard fault isn't generated for typical invalid memory region

Description

Hard fault isn't generated while trying to access illegal memory address space as shown below: 1. 0x010053FF - 0x20000000 2. 0x40BFFFFF - 0x41C00000 3. 0x41C007FF - 0x41C40000

Workaround

No

SYSCTL_ERR_01 *SYSCTL Module*

Category

Functional

SYSCTL_ERR_01 (continued) — *SYSCTL Module*

Function	SW-POR functionality is combined with HW-POR
Description	When a user writes to the LFSSRST register with the correct key to generate a software-triggered POR, the RSTCAUSE register will display 0x2 (indicating an NRST-triggered POR) instead of the expected 0x3 (Software-Triggered POR). This occurs because the SW-POR functionality is combined with the HW-POR path.
Workaround	No

SYSCTL_ERR_02 — *SYSCTL Module*

Category	Functional
Function	SYSSTATUS.FLASHSEC is non-zero after a BOOTRST
Description	After BOOTRST/ bootcode completion SYSSTATUS.FLASHSEC is non-zero. This the customer will see after bootcode completion.
Workaround	No

SYSCTL_ERR_04 — *SYSCTL Module*

Category	Functional
Function	SYSSTATUS.FLASHSEC is not cleared after a SYSRESET
Description	SYSSTATUS.FLASHSEC is not cleared after a SYSRESET and is only cleared by writing to the SYSSTATUSCLR register.
Workaround	No

CPU_ERR_02 — *CPU Module*

Category	Functional
Function	Limitation of disabling prefetch for CPUSS
Description	CPU prefetch disable will not take effect if there is a pending flash memory access.

CPU_ERR_02
(continued)

CPU Module

Workaround

Disable the prefetcher, 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.
Example:
CPUSS.CTL.PREFETCH = 0x0; //disables prefetcher
SYSCTL.SOCLOCK.SHUTDNSTORE0; // memory access to shutdown memory

CPU_ERR_03

CPU Module

Category

Functional

Function

Prefetcher can fetch wrong instructions when transitioning into Low power modes

Description

When transitioning into low power modes and there is a pending prefetch, the prefetcher can erroneously fetch incorrect data (all 0's). When the device wakes up, 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. A HW Event wake is another example of a process that will wake the device, but not flush the prefetcher.

Workaround

Disable prefetcher before entering low power modes.

Example:
CPUSS.CTL.PREFETCH = 0x0; // disables prefetcher
SYSCTL.SOCLOCK.SHUTDNSTORE0 // Read from SHUTDOWN Memory
__WFI(); // or __WFE(); this function calls the transition into low power mode
CPUSS.CTL.PREFETCH = 0x1; // enables prefetcher

AES_ERR_01

AES Module

Category

Functional

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.

KEYSTORE_ERR_01	***KEYSTORE Module***
Category	Functional
Function	STATUS.STAT value can be 0 or 1 without key access
Description	STATUS.STAT has a reset value of 1 and turns to 0 under these conditions: 1. After reset, debugger access via the register window returns 0x00. 2. After reset, the first CPU read returns 0x01, while subsequent CPU reads return 0x00. 3) After reset, first reading any other KEYSTORE register and then reading STATUS.STAT return 0x00.
Workaround	STATUS.STAT=0x0 means "No Error" . For checking if a slot is valid or not (Whether key is present), check STATUS.VALID.

RST_ERR_01	***RST Module***
Category	Functional
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 can appear normally.
Workaround	Keep the NRST pulse width higher than 608us to avoid this issue.

7 Revision History

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

Changes from November 30, 2024 to November 30, 2025 (from Revision * (November 2024) to Revision A (November 2025)) **Page**

- Removed PMCU_ERR_12.. 1
- Removed VREF_ERR_03.. 1
- PMCU_ERR_11 Description was updated.. 4
- PMCU_ERR_13 Category was updated... 4
- PMCU_ERR_13 Module was updated... 4
- PMCU_ERR_13 Function was updated... 4
- PMCU_ERR_13 Description was updated.. 4
- PMCU_ERR_13 Workaround was updated... 4
- Description and Workaround was updated... 5
- I2C_ERR_07 Category was updated.. 6
- I2C_ERR_07 Module was updated.. 6
- I2C_ERR_07 Function was updated.. 6
- I2C_ERR_07 Description was updated... 6
- I2C_ERR_07 Workaround was updated.. 6

- I2C_ERR_08 Category was updated....7
- I2C_ERR_08 Module was updated....7
- I2C_ERR_08 Function was updated....7
- I2C_ERR_08 Workaround was updated....7
- I2C_ERR_08 Description was updated....7
- I2C_ERR_09 Category was updated....7
- I2C_ERR_09 Module was updated....7
- I2C_ERR_09 Function was updated....7
- I2C_ERR_09 Description was updated....7
- I2C_ERR_09 Workaround was updated....7
- I2C_ERR_10 Category was updated....7
- I2C_ERR_10 Module was updated....7
- I2C_ERR_10 Function was updated....7
- I2C_ERR_10 Description was updated....7
- I2C_ERR_10 Workaround was updated....7
- UNICOMMI2CC_ERR_1 Category was updated....8
- UNICOMMI2CC_ERR_1 Function was updated....8
- UNICOMMI2CC_ERR_1 Description was updated....8
- UNICOMMI2CC_ERR_1 Workaround was updated....8
- UNICOMMI2CC_ERR_1 Module was updated....8
- I2C_ERR_13 Category was updated....8
- I2C_ERR_13 Module was updated....8
- I2C_ERR_13 Function was updated....8
- I2C_ERR_13 Workaround was updated....8
- I2C_ERR_13 Description was updated....8
- SPI_ERR_06 Category was updated....9
- SPI_ERR_06 Module was updated....9
- SPI_ERR_06 Function was updated....9
- SPI_ERR_06 Workaround was updated....9
- SPI_ERR_06 Description was updated....9
- SPI_ERR_07 Category was updated....9
- SPI_ERR_07 Module was updated....9
- SPI_ERR_07 Function was updated....9
- SPI_ERR_07 Description was updated....9
- SPI_ERR_07 Workaround was updated....9
- UART_ERR_04 Category was updated....13
- UART_ERR_04 Module was updated....13
- UART_ERR_04 Function was updated....13
- UART_ERR_04 Description was updated....13
- UART_ERR_04 Workaround was updated....13
- UART_ERR_05 Category was updated....14
- UART_ERR_05 Module was updated....14
- UART_ERR_05 Function was updated....14
- UART_ERR_05 Description was updated....14
- UART_ERR_05 Workaround was updated....14
- UART_ERR_06 Category was updated....14
- UART_ERR_06 Module was updated....14
- UART_ERR_06 Function was updated....14
- UART_ERR_06 Workaround was updated....14
- UART_ERR_06 Description was updated....14
- UART_ERR_07 Category was updated....15
- UART_ERR_07 Module was updated....15
- UART_ERR_07 Function was updated....15
- UART_ERR_07 Workaround was updated....15
- UART_ERR_07 Description was updated....15

- UART_ERR_08 Category was updated... 15
- UART_ERR_08 Module was updated.. 15
- UART_ERR_08 Function was updated.. 15
- UART_ERR_08 Description was updated.. 15
- UART_ERR_08 Workaround was updated.. 15
- UNICOMMUART_EER_9 Category was updated.. 15
- UNICOMMUART_EER_9 Function was updated.. 15
- UNICOMMUART_EER_9 Description was updated.. 15
- UNICOMMUART_EER_9 Workaround was updated.. 15
- UNICOMMUART_EER_9 Module was updated... 15
- UART_ERR_11 Category was updated.. 16
- UART_ERR_11 Module was updated... 16
- UART_ERR_11 Function was updated... 16
- UART_ERR_11 Description was updated... 16
- UART_ERR_11 Workaround was updated... 16
- FLASH_ERR_02 Category was updated.. 16
- FLASH_ERR_02 Module was updated... 16
- FLASH_ERR_02 Function was updated... 16
- FLASH_ERR_02 Description was updated... 16
- FLASH_ERR_02 Workaround was updated... 16
- FLASH_ERR_03 Category was updated.. 17
- FLASH_ERR_03 Module was updated... 17
- FLASH_ERR_03 Function was updated... 17
- FLASH_ERR_03 Description was updated... 17
- FLASH_ERR_03 Workaround was updated... 17
- FLASH_ERR_05 Category was updated.. 17
- FLASH_ERR_05 Module was updated... 17
- FLASH_ERR_05 Function was updated... 17
- FLASH_ERR_05 Description was updated... 17
- FLASH_ERR_05 Workaround was updated... 17
- FLASH_ERR_08 Category was updated.. 17
- FLASH_ERR_08 Module was updated... 17
- FLASH_ERR_08 Function was updated... 17
- FLASH_ERR_08 Description was updated... 17
- FLASH_ERR_08 Workaround was updated... 17
- SYSCTL_ERR_01 Category was updated... 17
- SYSCTL_ERR_01 Module was updated.. 17
- SYSCTL_ERR_01 Function was updated.. 17
- SYSCTL_ERR_01 Description was updated.. 17
- SYSCTL_ERR_01 Workaround was updated.. 17
- SYSCTL_ERR_02 Category was updated... 18
- SYSCTL_ERR_02 Module was updated.. 18
- SYSCTL_ERR_02 Function was updated.. 18
- SYSCTL_ERR_02 Description was updated.. 18
- SYSCTL_ERR_02 Workaround was updated.. 18
- SYSCTL_ERR_04 Category was updated... 18
- SYSCTL_ERR_04 Workaround was updated.. 18
- SYSCTL_ERR_04 Module was updated.. 18
- SYSCTL_ERR_04 Function was updated.. 18
- SYSCTL_ERR_04 Description was updated.. 18
- CPU_ERR_02 Category was updated.. 18
- CPU_ERR_02 Module was updated... 18
- CPU_ERR_02 Description was updated.. 18
- CPU_ERR_02 Function was updated.. 18
- CPU_ERR_02 Workaround was updated.. 18

- CPU_ERR_03 Category was updated.......... 19
- CPU_ERR_03 Module was updated.......... 19
- CPU_ERR_03 Function was updated.......... 19
- CPU_ERR_03 Description was updated.......... 19
- CPU_ERR_03 Workaround was updated.......... 19
- AES_ERR_01 Category was updated.......... 19
- AES_ERR_01 Module was updated.......... 19
- AES_ERR_01 Function was updated.......... 19
- AES_ERR_01 Description was updated.......... 19
- AES_ERR_01 Workaround was updated.......... 19
- KEYSTORE_ERR_01 Category was updated.......... 20
- KEYSTORE_ERR_01 Module was updated.......... 20
- KEYSTORE_ERR_01 Function was updated.......... 20
- KEYSTORE_ERR_01 Description was updated.......... 20
- KEYSTORE_ERR_01 Workaround was updated.......... 20
- RST_ERR_01 Category was updated.......... 20
- RST_ERR_01 Module was updated.......... 20
- RST_ERR_01 Workaround was updated.......... 20
- RST_ERR_01 Function was updated.......... 20
- RST_ERR_01 Description was updated.......... 20