Errata

AM62Dx Sitara™ Processors Silicon Errata, Silicon Revision 1.0

ABSTRACT

This document describes the known exceptions to the functional specifications (advisories). This document may also contain usage notes. Usage notes describe situations where the device's behavior may not match presumed or documented behavior. This may include behaviors that affect device performance or functional correctness.

Table of Contents

1 Usage Notes and Advisories Matrices

Table 1-1 lists all usage notes and the applicable silicon revision(s). Table 1-2 lists all advisories, modules affected, and the applicable silicon revision(s).

Table 1-1. Usage Notes Matrix

MODULE	DESCRIPTION	SILICON REVISIONS AFFECTED
		AM62Dx 1.0
Boot	i2372 — ROM doesn't support select multi-plane addressing schemes in Serial NAND boot	YES
DDR	i2330 — DDRSS Register Configuration Tool Updates	YES
OSPI	i2351 — OSPI: Controller does not support Continuous Read mode with NAND Flash	YES

Table 1-2. Advisories Matrix

MODULE	DESCRIPTION	SILICON REVISIONS AFFECTED
		AM62Dx 1.0
BCDMA	i2431 — BCDMA: RX Channel can lockup in certain scenarios	YES
BCDMA	i2436 — BCDMA: BCDMA RX_IGNORE_LONG setting in RX CHAN CFG register doesn't work	YES
Boot	i2366 — Boot: ROM does not comprehend specific JEDEC SFDP features for 8D-8D-8D operation	YES
Boot	i2371 — Boot: ROM code may hang in UART boot mode during data transfer	YES
Boot	i2410 — Boot: ROM may fail to boot due to i2409	YES
Boot	i2413 — Boot: HS-FS ROM boots corrupted ROM boot image	YES
Boot	i2414 — Boot: Ethernet PHY Scan and Bring-Up Flow doesn't work with PHYs that don't support Auto Negotiation	YES
Boot	i2417 — Boot: GPMC NAND configured to slower clock speed	YES
Boot	i2419 — Boot: When disabling deskew calibration, ROM does not check if deskew calibration was enabled	YES
Boot	i2420 — Boot: XSPI Boot time is not consistent in SFDP mode	YES
Boot	i2421 — Boot: fatTiny GPT handling causes data abort	YES
Boot	i2422 — Boot: ROM timeout for MMCSD filesystem boot too long	YES
Boot	i2423 — Boot: HS-FS ROM applies debug access restrictions to all address space covered by the efuse controller firewall	YES
Boot	i2435 — Boot: ROM timeout for eMMC boot too long	YES
Boot	i2482 — Boot: ROM does not provide enough clocks during SD card initialization	YES
Boot	i2464 — Boot: ROM is unable to boot from SD cards with incorrect formatting	YES
C7x	i2199 — C71x: SE returning incorrect data when non-aligned transposed stream crosses AM1 circular buffer boundary	YES
C7x	i2120 — C71x: SE returning incorrect data when non-aligned transposed stream crosses AM1 circular buffer boundary	YES
C7x	i2087— C7x MMA HWA_STATUS reports errors before application starts	
C7x	i2376 — C7x: SE/SA/HWAOPEN receives corrupted template following two back to back VPUT/MVC instructions	YES
C7x	i2399 — C7x: CPU NLC Module Not Clearing State on Interrupt	YES
CPSW	i2208 — CPSW: ALE IET Express Packet Drops	YES
CPSW	i2401 — CPSW: Host Timestamps Cause CPSW Port to Lock up	YES
DDR	i2160 — DDR: Valid VRef range must be defined during LPDDR4 Command Bus Training	YES
ECC_AGGR	i2049 — ECC_AGGR: Potential IP Clockstop/Reset Sequence Hang due to Pending ECC Aggregator Interrupts	YES
Interrupt Aggregator	i2196 — IA: Potential deadlock scenarios in IA	YES
LPM	i2487 — LPM: Low power modes may inadvertently corrupt DDR contents	YES
MCAN	i2278 — MCAN: Message Transmit order is not ensured from dedicated Tx Buffers configured with same Message ID	YES
MCAN	i2279 — MCAN: Specification Update for dedicated Tx Buffers and Tx Queues configured with same Message ID	YES
MMCHS	i2312 — MMCHS HS200 and SDR104 Command Timeout Window Too Small	YES
MMCHS	i2493 — MMCSD: HS200 Write Failures	YES

Table 1-2. Advisories Matrix (continued)

MODULE	DESCRIPTION	SILICON REVISIONS AFFECTED
		AM62Dx 1.0
OSPI	i2189 — OSPI: Controller PHY Tuning Algorithm	YES
OSPI	i2249 — OSPI: Failing OSPI DDR PHY Internal Pad Loopback and No Loopback timing modes	YES
OSPI	i2383 — OSPI: 2-byte address is not supported in PHY DDR mode	YES
PRG	i2253 — PRG: CTRL_MMR STAT registers are unreliable indicators of POK threshold failure	YES
PSIL	i2137 — PSIL: Clock stop operation can result in undefined behavior	YES
RAT	i2062 — RAT: Error Interrupt Triggered Even When Error Logging Disable Is Set	YES
RESET	i2407 — RESET: MCU_RESETSTATz unreliable when MCU_RESETz is asserted low	YES
USART	i2310 — USART: Erroneous triggering of timeout interrupt	YES
USART	i2311 — USART Spurious DMA Interrupts	YES
USB	i2134 — USB: 2.0 Compliance Receive Sensitivity Test Limitation	YES
USB	i2409 — USB: USB2 PHY locks up due to short suspend	YES

1.1 Devices Supported

This document supports the following devices:

- AM62Dx

Reference documents for the supported devices are:

- AM62Dx Processors Technical Reference Manual (SPRUJD4)
- AM62Dx Processors Data Sheet (SPRSPB5)

2 Silicon Usage Notes and Advisories

This section lists the usage notes and advisories for this silicon revision.

2.1 Silicon Usage Notes

i2351 ***OSPI: Direct Access Controller (DAC) does not support Continuous Read mode with NAND Flash***

Details:

The OSPI Direct Access Controller (DAC) doesn't support Continuous Read mode with NAND Flash since the OSPI controller can deassert the CSn signal (by design intent) to the Flash memory between internal DMA bus requests to the OSPI controller.

The issue occurs because "Continuous Read" mode offered by some OSPI/QSPI NAND Flash memories requires the Chip Select input to remain asserted for an entire burst transaction.

The SoC internal DMA controllers and other initiators are limited to 1023 B or smaller transactions, and arbitration/queuing can happen both inside of the various DMA controllers or in the interconnect between any DMA controller and the OSPI peripheral. This results in delays in bus requests to the OSPI controller that result in the external CSn signal being deasserted.

NOR Flash memories are not affected by CSn de-assertion and Continuous Read mode works as expected.

Workaround(s):

Software can use page/buffered read modes to access NAND flash.

i2330 ***DDRSS Register Configuration Tool Updates***

Details:

The DDR Register Configuration Tool provides custom register settings based on system level details such as the architecture (density, data width, ranks) of the DDR device, frequency of operation, and IO settings determined through board simulations. This tool may be updated over time to support new devices and/or features, fix issues identified with the tool, and most importantly, capture work-arounds of errata or recent updates identified to register calculations which improve performance, signal integrity, or timing relationships between signals.

Workaround(s):

In order to ensure that parameters are set appropriately based on lessons learned and reduce the risk of functional failure, the latest DDR register configuration tool should always be used to generate register values. As the DDR register configuration tool can periodically be updated, the revision history of the tool should be reviewed and evaluated whether tool changes apply to existing systems. When applicable, the configuration of an existing system should be updated appropriately. The latest version of the tool can be found at http://dev.ti.com/sysconfig, and choosing DDR Configuration under Software Product drop down for the applicable device that is being used.

i2372 ***Boot: ROM doesn't support select multi-plane addressing schemes in Serial NAND boot***

Details:

The ROM bootloader does not support certain multi-plane Serial SPI NAND flash memories that require the read from cache/buffer command to comprehend changing the cache/buffer/plane number to access the correct data.

i2372 (continued) ***Boot: ROM doesn't support select multi-plane addressing schemes in Serial NAND boot***

Workaround(s):

Carefully review the addressing requirements of a candidate flash memory for references to a special bit for selecting a plane/buffer/cache in the read from cache/buffer command. Do not use memories that have such a requirement.

2.2 Silicon Advisories

i2049 ***ECC_AGGR: Potential IP Clockstop/Reset Sequence Hang due to Pending ECC Aggregator Interrupts***

Details:

The ECC Aggregator module is used to aggregate safety error occurrences (which are rare) and generate interrupts to notify software. The ECC Aggregator provides software control over the enabling/disabling and clearing of safety errors interrupts.

When software is performing a clockstop/reset sequence on an IP, the sequence can potentially not complete because the IP's associated ECC Aggregator instance is not idle. The ECC Aggregator idle status is dependent upon any pending safety error interrupts either enabled or disabled, which have not been cleared by software. As a result, the IP's clockstop/reset sequence may never complete (hang) if there are any pending safety errors interrupts that remain uncleared.

The affected ECC_AGGRs can be determined by the value listed in the Technical Reference Manual (TRM) for their REV register at Register Offset 0h. The REV register encodes the ECC_AGGR version in its fields as follows:

v[REVMAJ].[REVMIN].[REVRTL]

ECC_AGGR versions before v2.1.1 are affected. ECC_AGGR versions v2.1.1 and later are not affected.

Affected Example:

REVMAJ = 2

REVMIN = 1

REVRTL = 0

The above values decode to ECC_AGGR Version v2.1.0, which is Affected.

Not Affected Example:

REVMAJ = 2

REVMIN = 1

REVRTL = 1

The above values decode ECC_AGGR Version v2.1.1, which is Not Affected.

Workaround(s):

General Note:

Clockstopping the ECC Aggregator is not supported in functional safety use-cases.

Software should use the following workaround for non-functional safety use-cases:

1. Enable all ECC Aggregator interrupts for the IP
2. Service and clear all Pending interrupts
3. Step 3:
   a. Disable all interrupt sources to the ECC Aggregator, followed by performing Clockstop/reset sequence.
   b. Perform Clockstop/reset sequence, while continuing to service/clear pending interrupts.

Due to interrupts being external stimuli, software has two options for step 3:

i2049 (continued) ***ECC_AGGR: Potential IP Clockstop/Reset Sequence Hang due to Pending ECC Aggregator Interrupts***

1. Disable all interrupt sources (EDC CTRL checkers) that can generate pending ECC_AGGR interrupts prior to performing the clockstop/reset sequence
2. Continue to service/clear pending interrupts that occur while performing the clkstop/reset sequence. The sequence would proceed when all interrupts are cleared.

Software in general may need to detect pending interrupts that continuously fire during this entire sequence (ex. in the case of a stuck-at fault scenario), and disable their associated EDC CTRL safety checkers to allow the clockstop/reset sequence to progress towards completion.

i2062 ***RAT: Error Interrupt Triggered Even When Error Logging Disable Is Set***

Details:

If the RAT error logging is programmed to disable logging and enable interrupts, then an error will incorrectly trigger an interrupt but the error log registers will correctly not be updated. The error interrupt should not have been generated.

Workaround(s):

If the RAT error logging is disabled, then the error interrupt should also be disabled by software.

i2087 ***C71x: MMA HWA_STATUS Reports Errors Before Application Starts***

Details:

Due to uninitialized internal state, the Matrix Math Accelerator (MMA) attached to the C71x may report errors in the FirstErrorCode and LastErrorCode fields of the HWA_STATUS register after power-on. Because these fields are sticky, any subsequent HWARCV instruction may throw a C71x exception.

Workaround(s):

After power-on, a short instruction sequence running on the C71x can initialize the internal MMA state before the first execution of normal MMA operation. Only one execution of the sequence is required.

The sequence generates a valid HWA_CONFIG and HWA_OFFSET value, loads it into the MMA, then clears the sticky error codes.

The sequence, in C71x assembly code is:

```
PROT
    MVK32 .M2 0x0,B0 ; clear low word of VB0
    VDUPW .C2 B0,VB0 ; duplicate word across VB0
    HWAOPEN .L2 VB0,VB0,0 ; clear HWA_CONFIG and HWA_OFFSET
    HWACLOSE .S1 0 ; clear any error conditions
```

i2134 ***USB: 2.0 Compliance Receive Sensitivity Test Limitation***

Details:

Performing receive sensitivity tests (EL_16 and EL_17) as defined in the USB-IF USB 2.0 Electrical Compliance Test Specification may invoke the problem described in Advisory i2091.

The issue was originally found while performing these tests using automation software, which increased USB signal amplitude while sending packets. The software was sweeping the amplitude from a value less than 100 mV to a value greater than 150

i2134 (continued) ***USB: 2.0 Compliance Receive Sensitivity Test Limitation***

mV while verifying the device under test (DUT) NAK'd all packets below 100 mV and NAK'd no packets above 150 mV. However, increasing the amplitude through the squelch threshold while sending valid packets may lock the PHY as described in Advisory i2091.

Workaround(s):

i2189 ***OSPI: Controller PHY Tuning Algorithm***

Details:

The OSPI controller uses a DQS signal to sample data when the PHY Module is enabled. However, there is an issue in the module which requires that this sample must occur within a window defined by the internal clock. Read operations are subject to external delays, which change with temperature. To ensure valid reads at any temperature, a special tuning algorithm must be implemented which selects the most robust TX, RX, and Read Delay values.

Workaround(s):

The workaround for this bug is described in detail in SPRACT2. To sample data under some PVT conditions, users are required to increment the Read Delay field to shift the internal clock sampling window. This allows sampling of the data anywhere within the data eye. However, this has these side effects:

1. PHY Pipeline mode must be enabled for all read operations. Because PHY Pipeline mode must be disabled for writes, reads and writes must be handled separately.
2. Hardware polling of the busy bit is broken when the workaround is in place, so SW polling must be used instead. Writes must occur through DMA accesses, within page boundaries, to prevent interruption from either the host or the flash device. Software must poll the busy bit between page writes. Alternatively, writes can be performed in non-PHY mode with hardware polling enabled.
3. STIG reads must be padded with extra bytes, and the received data must be right-shifted.

i2196 ***IA: Potential deadlock scenarios in IA***

Details:

The interrupt Aggregator (IA) has one main function, which is to convert events arriving on the Event Transport Lane (ETL) bus, can convert them to interrupt status bits which are used to generate level interrupts. The block that performed this function in IA version 1.0 was called the status event block.

In addition to the status event block, there are two other main processing blocks; the multicast event block, and the counted event block. The multicast block really functions as an event splitter. For every event it takes in, it can generate two output events. The counted event block is used to convert high frequency events into a readable count. It counts input events and generates output events on count transitions to/from 0 to/from non-zero count values. Unlike the status event block, the multicast and counted event blocks generate output ETL events that are then mapped to other processing blocks.

An issue was found after design that could cause the IA to deadlock. The issue occurs when event "loops" occur between these three processing blocks. It is possible to create a situation where a processing block can not output an event because the path is blocked, and since it can not output an event, it can not take any new input events. This inability to

i2196 (continued) ***IA: Potential deadlock scenarios in IA***

take input events prevents the output path from being able to unwind, and thus both paths remain blocked.

Workaround(s):

Figure 2-1 shows the conceptual block diagram of IA 1.0. Potential loops are avoided by adopting the policy of not allowing the counted event block to send events to the multicast block. This method was chosen because it is more common to split an event first, and then count one while sending the other elsewhere. With this path blocked by convention, it is not possible for a single event to visit any block more than once and thus not possible for paths to become blocked so long as the outputs remain unblocked.

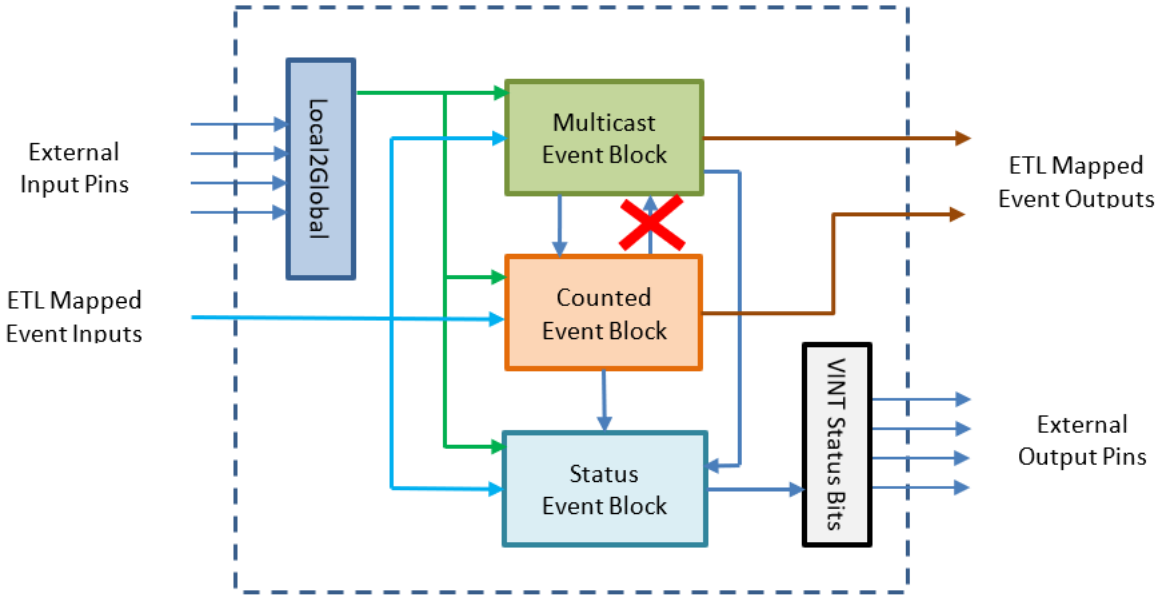

Figure 2-1. Interrupt Aggregator Version 1.0

By following the conventions outlined here, the system is safe from looping hazards that can create a deadlock scenario.

i2199 ***C71x: SE returning incorrect data when non-aligned transposed stream crosses AM1 circular buffer boundary***

Details:

When AM1 refers to a larger circular buffer size than AM0, SE can reuse the wrong 64B line of data during non-aligned transposed streams. This occurs when one of the rows being transposed crosses the AM1 circular buffer boundary, but not the AM0 boundary.

Workaround(s):

Have the transposed stream either be fully aligned, meaning that the start address and all scaled DIM values be multiples of 64B, or to not configure AM1 to be a larger circular addressing buffers size than AM0.

i2208 ***CPSW: ALE IET Express Packet Drops***

Details:

This issue impacts the following Module:

i2208 (continued) ***CPSW: ALE IET Express Packet Drops***

[AM62A/D] 3-port CPSW

The issue with ALE is due to CPSW frequency and IET operation with short express traffic and pre-empted packets that get pre-empted between 60-69 bytes on non-10G capable ports.

If an IET pre-emptible packet get interrupted at 60-69 bytes, the lookup will occur when the next chunk arrives. The CPSW only gives the ALE 64 bytes from the pre-emptible MAC.

As a result, a short express traffic lookup will start at the end of a 64 byte express traffic, but when the pre-empted queue continues, the pre-empted traffic will complete the 64 bytes and attempt a lookup for the pre-empt packet. But this lookup is less that 64 clocks from the express lookup start, so the express lookup will be aborted(express traffic dropped) and start the new lookup for the pre-empted traffic.

Rules to induce the issue:

1. You are in IET (Interspersed Express Traffic) mode on ports not capable of 5/10G operation
2. Remote express packets can be preempt packets as low as 60 bytes
3. Pre-empt packet traffic that is 128 bytes or more.
4. Express traffic that interrupts the pre-empt traffic between 60-69 bytes.
5. A short express traffic immediately followed by the continuation of the pre-empt traffic.
   a. Gap between express frame and pre-empt frame be its minimum.
6. The CPSW frequency is at its lowest capability for the speeds required.

Workaround(s):

During IET negotiation, tell the remote to fragment at 128 bytes.

i2249 ***OSPI: Internal PHY Loopback and Internal Pad Loopback clocking modes with DDR timing inoperable***

Details

The OSPI Internal PHY Loopback mode and Internal Pad Loopback mode uses “launch edge as capture edge” (same edge capture, or 0-cycle timing).

The programmable receive delay line (Rx PDL) is used to compensate for the round trip delay (Tx clock to Flash device, Flash clock to output and Flash data to Controller).

In the case of internal and IO loopback modes, the total delay of the Rx PDL is not sufficient to compensate for the round trip delay, and thus these modes cannot be used.

The table below describes the recommended clocking topologies in the OSPI controller. All other modes not described here are affected by the advisory in DDR mode and are not recommended clocking topologies.

Table 2-1. OSPI Clocking Topologies

Clocking Mode Terminology	CONFIG_REG.PHY_MODE_ENABLE	READ_DATA_CAPTURE.BYPASS	READ_DATA_CAPTURE.DQS_EN	Board implementation
No Loopback, no PHY	0 (PHY disabled)	1 (disable adapted loopback clock)	X	None. Relying on internal clock. Max freq 50MHz.
External Board Loopback with PHY	1 (PHY enabled)	0 (enable adapted loopback clock)	0 (DQS disabled)	External Board Loopback (OSPI_LOOPBACK_CLK_SEL = 0)

i2249 (continued) ***OSPI: Internal PHY Loopback and Internal Pad Loopback clocking modes with DDR timing inoperable***

Table 2-1. OSPI Clocking Topologies (continued)

Clocking Mode Terminology	CONFIG_REG.PHY_MODE_ENABLE	READ_DATA_CAPTURE.BYPASS	READ_DATA_CAPTURE.DQS_EN	Board implementation
DQS with PHY	1 (PHY enabled)	X (DQS enable has priority)	1 (DQS enabled)	Memory strobe connected to SOC DQS pin

Workaround

None. Please use one of the unaffected clocking modes based on the table in the description

i2278 ***MCAN: Message Transmit order is not ensured from dedicated Tx Buffers configured with same Message ID***

Details

The erratum is limited to the case when multiple Tx Buffers are configured with the same Message ID (TXBC.NDTB > 1).

Under the following conditions, a message may be transmitted out of order:

- Multiple Tx Buffers configured with the same Message ID
- Tx requests for these Tx Buffers are submitted sequentially with delays between each

Workaround

Workaround #1:

After writing the Tx messages with same Message ID to the Message RAM, request transmission of all these message concurrently by single write access to TXBAR. Make sure none of these messages have a pending Tx request before making the concurrent request.

Workaround #2:

Use the Tx FIFO instead of dedicated Tx Buffers (set bit MCAN_TXBC[30] TFQM = 0 to use Tx FIFO) for the transmission of several messages with the same Message ID in a specific order.

i2279 ***MCAN: Specification Update for dedicated Tx Buffers and Tx Queues configured with same Message ID***

Details

The erratum updates the descriptions in Section 3.5.2 Dedicated Tx Buffers and 3.5.4 Tx Queue of the M_CAN User's Manual related to message transmission from multiple dedicated Tx Buffers configured with the same Message ID.

Workaround

Workaround #1:

After writing the Tx messages with same Message ID to the Message RAM, request transmission of all these message concurrently by single write access to TXBAR. Make sure none of these messages have a pending Tx request before making the concurrent request.

Workaround #2:

i2279 (continued) ***MCAN: Specification Update for dedicated Tx Buffers and Tx Queues configured with same Message ID***

Use the Tx FIFO instead of dedicated Tx Buffers (set bit MCAN_TXBC[30] TFQM = 0 to use Tx FIFO) for the transmission of several messages with the same Message ID in a specific order.

i2310 ***USART: Erroneous clear/trigger of timeout interrupt***

Details:

The USART may erroneously clear or trigger the timeout interrupt when RHR/MSR/LSR registers are read.

Workaround(s):

For CPU use-case.

- If the timeout interrupt is erroneously cleared:
  - This is Valid since the pending data inside the FIFO retriggers the timeout interrupt
- If timeout interrupt is erroneously set, and the FIFO is empty, use the following SW workaround to clear the interrupt:
  - Set a high value of timeout counter in TIMEOUTH and TIMEOUTL registers
  - Set EFR2 bit 6 to 1 to change timeout mode to periodic
  - Read the IIR register to clear the interrupt
  - Set EFR2 bit 6 back to 0 to change timeout mode back to the original mode

For DMA use-case.

- If timeout interrupt is erroneously cleared:
  - This is valid since the next periodic event retriggers the timeout interrupt
  - User must ensure that RX timeout behavior is in periodic mode by setting EFR2 bit6 to 1
- If timeout interrupt is erroneously set:
  - This causes DMA to be torn down by the SW driver
  - Valid since next incoming data causes SW to setup DMA again

i2311 ***USART Spurious DMA Interrupts***

Details:

Spurious DMA interrupts may occur when DMA is used to access TX/RX FIFO with a non-power-of-2 trigger level in the TLR register.

Workaround(s):

Use power of 2 values for TX/RX FIFO trigger levels (1, 2, 4, 8, 16, and 32).

i2312 ***MMCSD: HS200 and SDR104 Command Timeout Window Too Small***

Details:

Under high speed HS200 and SDR104 modes, the functional clock for MMC modules will reach up to 192 MHz. At this frequency, the maximum obtainable timeout through of MMC host controller using MMCSD_SYSCTL[19:16] DTO = 0xE is (1/192MHz)*2^27 = 700ms. Commands taking longer than 700ms may be affected by this small window frame.

Workaround(s):

i2312 (continued) ***MMCSD: HS200 and SDR104 Command Timeout Window Too Small***

If the command requires a timeout longer than 700ms, then the MMC host controller command timeout can be disabled (MMCSD_CON[6] MIT=0x1) and a software implementation may be used in its place. Detailed steps as follows (in Linux):

1. During MMC host controller probe function (omap_hsmmc.c:omap_hsmmc_probe()), inform processor that the host controller is incapable of supporting all the necessary timeouts.

2. Modify the MMC core software layer functionality so the core times out on its own when the underlying MMC host controller is unable to support the required timeout.

i2366 ***Boot: ROM does not comprehend specific JEDEC SFDP features for 8D-8D-8D operation***

Details:

JEDEC spec JESD216 - SERIAL FLASH DISCOVERABLE PARAMETERS (SFDP) details the parameter table used in certain serial flash devices to describe features and how to communicate/configure the device. The ROM interprets relevant portions of the SFDP for a device's features (such as a how to change from 1S-1S-1S to 8D-8D-8D mode), but does not properly comprehend a flash device that requires:

- A swapped byte order in 8D-8D-8D mode compared to 1S-1S-1S mode
- A command extension that in 8D-8D-8D mode that requires a different command than the first byte sent (such as an inversion of the opcode or another unique byte)

Workaround(s):

Review the SFDP table of any candidate flash memory that is compliant with JEDEC JESD216; in most cases vendors do not publish this table and can instead be requested from the flash vendor. If the 18th DWORD of the JEDEC Basic Flash Parameter table has bit 31 with a value of "1b", then the memory must be programmed with a swapped byte order from the factory or programmed with the SoC. If bits [30:29] have a value other than "00b" then it will not work with any bootmodes in 8D-8D-8D mode. Avoid using any 8D-8D-8D bootmodes with that flash device as a result.

i2371 ***Boot: ROM code may hang in UART boot mode during data transfer***

Details:

Due to advisory i2310, it is possible for ROM code execution to hang during UART boot. The software workaround presented in i2310 is not implemented in ROM, and thus an erroneous timeout interrupt can be triggered in an unexpected state. This can prevent the ROM from being able to clear this interrupt and therefore hang.

This can manifest any time UART boot mode is used or when UART is used as the boot interface to enable production flows such as UniFlash or programing eFuses with OTP Keywriter.

Workaround(s):

None. Another boot interface should be used.

i2120 ***C71x: SE Hangs on Non-Parity Error Detection in Transposed Streams With LEZR***

Details:

The C71x Streaming Engine's (SE) pipeline for returning formatted data and return report internal error information is always monitoring the tags for the data that it is working on. When an error is detected for a line of data used to format data back to the CPU, all

i2120 (continued) ***C71x: SE Hangs on Non-Parity Error Detection in Transposed Streams With LEZR***

fetching side execution for queuing up commands to go to UMC, uTLB, and the formatting pipeline back to CPU is halted.

In general operation, the only tags monitored for errors are the ones being used for the current command. For transposed mode, this is all tags touched by the current array column. A gap in suppressing internal tag monitoring causes the formatting pipeline to monitor tags that it is not currently working on while creating zero vectors for the LEZR feature. If the SE's fetching side encounters and records an error for a future column, the formatting side may notice it and halt the fetching side before the command for that column has been committed for formatting.

Errors are only reported back to the CPU for commands that are internally committed for formatting, thus halting internal execution before committing the column results in no error being reported to the CPU. Because the SE has halted fetching operations without reporting an error, the CPU proceeds to hang, waiting for either return data or an error from the SE, until an unrelated external event or interrupt occurs.

Workaround(s):

The only 100% workaround is to not use stream templates with both LEZR and transposed mode enabled.

i2137 ***PSIL: Clock stop operation can result in undefined behavior***

Details:

The clock stop interface is a request/acknowledge interface used to coordinate the handshaking of properly stopping the main clock to the module. Attempting a clock stop on the module without first performing the channel teardowns or clearing of global enable bits will result in module-specific behavior that may be undefined.

The impacted modules are PDMA, SA2UL, Ethernet SW, CSI, UDMAP, ICSS, and CAL.

Workaround(s):

Before attempting to perform a clock stop operation, software is required to teardown all active channels (via UDMAP “real time” registers in the UDMAP, or PSIL register 0x408 in PSIL based modules), and after this is complete, also clear the global enable bit for all channels (via PSIL register 0x2 in both the UDMAP and PSIL based modules).

i2253 ***PRG: CTRL_MMR STAT registers are unreliable indicators of POK threshold failure***

Details

The POK overvoltage and undervoltage flags in the CTRL_MMR PRG STAT registers are unreliable indicators of whether the POK has seen a failure. As a result, they are being marked as Reserved in the device Technical Reference Manual (TRM).

Workaround

The filtered POK output updates ESM flags.

Upon POK initialization (i.e. enable), the ESM flags should be cleared (due to comparisons carried out during the bandgap and / or the POK settling time). After this initial clear, the ESM flags can be used as a reliable indicator of failure (or no failure) from the POKs.

i2383 *OSPI: 2-byte address is not supported in PHY DDR mode*

Details:

When the OSPI controller is configured for 2-byte addressing in PHY DDR Mode, an internal state machine mis-compares the number of address bytes transmitted to a value of 1 (instead of 2). This results in a state machine lockup in the address phase, rendering PHY DDR mode non-operable.

This issue does not occur when using any Tap mode or PHY SDR mode. This issue also doesn't occur when using 4 byte addressing in PHY DDR mode.

Workaround(s):

For compatible OSPI memories that have programmable address byte settings, set the amount of address bytes required from 2 to 4 on the flash. This may involve sending a specific command to change address bytes and/or writing a configuration register on the flash. Once done, update the amount of address bytes sent in the controller settings from 2 to 4.

For compatible OSPI memories that only support 2-byte addressing and cannot be re-programmed, PHY DDR mode will not be compatible with that memory. Alternative modes include:

- PHY SDR mode
- TAP (no-PHY) DDR mode
- TAP (no-PHY) SDR mode

i2401 *CPSW: Host Timestamps Cause CPSW Port to Lock up*

Details:

The CPSW offers two mechanisms for communicating packet ingress timestamp information to the host.

The first mechanism is via the CPTS Event FIFO which records timestamps when triggered by certain events. One such event is the reception of an Ethernet packet with a specified EtherType field. Most commonly this is used to capture ingress timestamps for PTP packets. With this mechanism the host must read the timestamp (from the CPTS FIFO) separately from the packet payload which is delivered via DMA. This mode is supported and is not affected by this errata.

The second mechanism is to enable receive timestamps for all packets, not just PTP packets. With this mechanism the timestamp is delivered alongside the packet payload via DMA. This second mechanism is the subject of this errata.

When the CPTS host timestamp is enabled, every packet to the internal CPSW port FIFO requires a timestamp from the CPTS. When the packet preamble is corrupted due to EMI or any other corruption mechanism a timestamp request may not be sent to the CPTS. In this case the CPTS will not produce the timestamp which causes a lockup condition in the CPSW port FIFO. When the CPTS host timestamp is disabled by clearing the tstamp_en bit in the CPTS_CONTROL register the lockup condition is prevented from occurring.

Workaround(s):

Ethernet to host timestamps must be disabled.

CPTS Event FIFO timestamping can be used instead of CPTS host timestamps.

i2407 *RESET: MCU_RESETSTATz unreliable when MCU_RESETz is asserted low*

Details:

MCU_RESETSTATz goes high periodically for a short duration and then low again while MCU_RESETz is still asserted low. This issue is seen only when MCU_RESETz is

i2407 (continued) ***RESET: MCU_RESETSTATz unreliable when MCU_RESETz is asserted low***

asserted low for greater than 100us. The device remains in reset while MCU_RESETz is low; the advisory only applies to the signal MCU_RESETSTATz.

Workaround(s):

Any one of the following could be used as a workaround for this advisory

- Do not use MCU_RESETz in a functional system. MCU_RESETz can still be used for debug, realizing the errata limitation.
- Limit the maximum low duration of MCU_RESETz to less than 100us.
- Use Main Domain RESETSTATz instead of MCU_RESETSTATz. MCU_RESETz also causes a Main reset, so Main Domain RESETSTATz could be used for device reset observation. Consult the datasheet for RESETSTATz timing specifications.
- For new designs, the circuits which produce Main Domain reset and MCU Domain reset should be combined with an AND gate as an input to RESETz. Also connect MCU Domain reset circuit to the MCU_RESETz input. This will provide full functionality of MCU warm reset using MCU_RESETz and MCU_RESETSTATz to indicate status of MCU domain reset. RESETz will be triggered on either a MAIN domain reset or MCU domain reset by using the AND gate.

i2409 ***USB: USB2 PHY locks up due to short suspend***

Details:

The USB 2.0 PHY may hang in response to a USB wake-up event that occurs within 3 microseconds of the USB controller entering suspend. This PHY hang can only be recovered via a power cycle as warm reset is ineffectual.

Workaround(s):

Note: this workaround is only applicable if USB is not the primary boot mode. If USB is the primary boot mode, no workaround is available.

In order to prevent this issue from occurring, a specific order of operations must be observed during the USB controller initialization process:

1. Remove USB controller reset via the LPSC.
2. Set PLL_REG12.pll_ldo_ref_en field (bit 5) in PHY2 region to '1'.
3. Set PLL_REG12.pll_ldo_ref_en_en field (bit 4) in PHY2 region to '1'.
4. Proceed with normal USB controller initialization.

i2410 ***Boot: ROM may fail to boot due to i2409***

Details:

Due to i2409, the ROM may fail to boot in USB boot mode after a warm reset. If the USB 2.0 PHY locks up, the ROM does not implement any of the workarounds listed in i2409, and thus the ROM will hang and fail to boot.

Workaround(s):

The advisory described in i2409 should be avoided by implementing one of the workarounds described in the advisory in software.

i2376 *C7x: SE/SA/HWAOPEN receives corrupted template following two back to back VPUT/MVC instructions*

Details

On the C7604 CPU, programming information is sent to the Streaming Engines via SEOPEN, the Streaming Address Generators via SAOPEN, and to the MMA via the HWAOPEN instructions. This programming information is sourced from CUCR registers in the CPU that are populated and updated via the MVC and VPUT family of instructions. Data being written into these CUCR registers can be incorrectly forwarded to the SEOPEN, SAOPEN, and HWAOPEN instructions if in a sequence of three CPU cycles a CUCR register is updated on the first two cycles and then used by an SEOPEN, SAOPEN, or HWAOPEN instruction on the third.

For example, this sequence will cause a forwarding error to occur in the execution of the SAOPEN instruction, corrupting SA0's programming:

```
MVC .C2 VB0, CUCR0
VPUTD .C2 B1, 0, CUCR0
SAOPEN .C2 CUCR1:CUCR0, 0
```

Workaround

The issue can be avoided by avoiding the sequence of three execute packets where the first two contain MVC/VPUT instructions that write to the same CUCR and the third contains an SE/SA/HWAOPEN instruction that reads from the same. The workaround is included automatically by the compiler when compiling C7504 code.

i2399 *C7x: CPU NLC Module Not Clearing State on Interrupt*

Details:

Data corruption will occur when:

1. An application is running that involves task switching. In this case there are at least 2 tasks that may use NLC.
2. There is a NLCINIT issued that would be followed by a TICK when an interrupt comes in for Task A. This action ends up setting some internal state in the NLC module that says we need to reload the ILCNT_INIT value to ILCNT on the next TICK since the forwarded case it computed was flushed. This state is not being properly cleared when the interrupt is taken.
3. The ISR performs a task switch to Task B, which is also running NLC code. The NLC code being returned to needs to be in-progress and have a different ILCNT_INIT value than the NLC loop in the original task.
4. After returning from the ISR, the next TICK will end up setting ILCNT to the wrong value (ILCNT_INIT - 2) due to the corrupted state.

At this point the ILCNT is corrupted and the NLC loop will execute the wrong number of iterations, leading to data corruption.

Workaround(s):

Issue a NLCINIT (parameters don't matter and there's no need for TICK's/BNL afterwards) in ISR's as part of the context saving. There is no performance impact due to the work-around.

i2413 *Boot: HS-FS ROM boots corrupted ROM boot image*

Details:

ROM supports an image format in which both boot loader and TIFS images are present. This is called a combined image.

i2413 (continued) ***Boot: HS-FS ROM boots corrupted ROM boot image***

On HS-FS devices, when the combined image is signed with an RSA key, ROM is expected to:

- Skip the integrity check on the boot loader components
- Perform integrity check and signature verification on TIFS components.

Due to a bug in ROM, ROM is skipping the integrity check on the TIFS components on an HS-FS device when a non-degenerate RSA key is used.

Workaround(s):

Sign the X509 certificate with the RSA degenerate key for enabling the integrity check of all the components (bootloader and TIFS)

i2414 ***Boot: Ethernet PHY Scan and Bring-Up Flow doesn't work with PHYs that don't support Auto Negotiation***

Details:

ROM Ethernet (either RGMII or RMII) boot mode relies on PHY auto-negotiation to complete before checking for link status. Hence PHY that do not support auto-negotiation cannot work with this boot mode.

Workaround(s):

None, a PHY supporting auto-negotiation is required.

i2417 ***Boot: GPMC NAND configured to slower clock speed***

Details:

When using GPMC NAND boot mode the GPMCFCLKDIVIDER field of the GPMC_CONFIG1 register bit [1:0] (i.e. GPMCFCLKDIVIDER) gets set to 1 which causes a divide by 2 for the GPMC_FCLK.

The ROM uses very conservative CONFIG timing values anyways so end result may not really adversely affect throughput.

Workaround(s):

None.

i2419 ***Boot: When disabling deskew calibration, ROM does not check if deskew calibration was enabled***

Details:

If PLL Deskew calibration is being disabled, the ROM driver code intends to check if deskew calibration is enabled and if lock has failed. However the current code has an assignment in an if condition. As a result, it does not check if deskew calibration is enabled before clearing the config bit. There is no functional issue.

Workaround(s):

None

i2420 ***Boot: XSPI Boot time is not consistent in SFDP mode***

Details:

When using xSPI boot with SFDP enabled (i.e. booting in DDR mode at 25Mhz) there is boot time variation across cold or warm boot. The issue is related to asynch bridge crossing in the OSPI subsystem, it is causing a race condition between:

1. OSPI IP finishing its prefetch of data
2. The next read transaction being submitted to the OSPI IP by the TI OSPI wrapper.

This introduces enough of a delay to cause the OSPI controller to de-assert chip select hence slowing down the overall transfer.

Workaround(s):

None

i2421 ***Boot: fatTiny GPT handling causes data abort***

Details:

Attempting to boot from a GPT formatted filesystem causes the Public ROM (R5) to go into a Data Abort.At which point boot would hang until the Watchdog timer kicks in.

Workaround(s):

This GPT partition table type is not supported, make sure the partition table is MBR and the boot partition type is FAT.

i2422 ***Boot: ROM timeout for MMCSD filesystem boot too long***

Details:

Due to a bug in ROM if attempting to boot in SD/MMC boot (filesystem mode) from an eMMC device that is empty or erased (or factory fresh) the normal boot timeout to switch to backup boot mode will not occur as the boot gets stuck in an infinite loop until the watchdog timer reset kick in.

Workaround(s):

Need to boot from another primary boot mode to program the eMMC flash.

i2423 ***Boot: HS-FS ROM applies debug access restrictions to all address space covered by the efuse controller firewall***

Details:

On HS-FS device ROM applies debug restrictions to FWL 33 and 66 which contains secure assets. The debug access restriction was applied to the entire firewall region and not just to the region which applies to the secure asset. This prevent the use of an external emulator to be able to perform initial flash programming for example without requiring TIFS SW to in the picture.

Workaround(s):

TIFS SW is needed to be able to request the needed firewall to be open.

i2431 ***BCDMA: RX Channel can lockup in certain scenarios***

Details:

BCDMA RX chan Teardown can lockup channel and cannot be used for subsequent transfers if none of the TRs have EOP flag set in configuration specific flags field. Subsequently when channel is re-enabled, transfer does not not complete and terminates with various errors in TR response.

Workaround(s):

a) When receiving data from a PSIL/PDMA peripheral, EOP flag needs to be set in the each TR's configuration specific flag field and PDMA's 1 X-Y FIFO Mode Static TR "Z" parameter should be set to non zero value for channel teardown to function properly and cleanup the internal state memory. Otherwise it leads to channel lockups on subsequent runs. The PDMA Z count should also match the TR size, so that PDMA delineates each transfer as an individual packet. This is especially problematic in cases like where TRPD has infinite reload count set to perform cyclic transfer using a single set of TRs in streaming mode, in which case each TR could potentially be the last one.

b) If the usecase doesn't allow for PDMA Z count to be set in advance or packet EOP cannot be set then alternate is to use PKTDMA in single buffer mode instead of BCDMA.

i2435 ***Boot: ROM timeout for eMMC boot too long***

Details:

Due to a bug in ROM, if attempting to boot in eMMC boot mode (ie, from eMMC boot partitions, sometimes referred to as eMMC alternative mode) from an eMMC device that is empty or erased (or factory fresh), the normal boot timeout to switch to backup boot mode takes 10 seconds.

Workaround(s):

Need to boot from another boot mode if this timeout considered too long in the system.

i2160 ***DDR: Valid VRef Range Must be Defined During LPDDR4 Command Bus Training***

Details:

The DDR PHY updates VREF(ca) for the command/address bus during LPDDR4 Command Bus Training (CBT). If VREF(ca) search range is set to invalid values such as no working settings can be found during CBT, the training process could fail or hang.

Workaround(s):

Set the following fields to known valid working values before enabling CBT.

For frequency set 0: PI_CALVL_VREF_INITIAL_START_POINT_F0 and PI_CALVL_VREF_INITIAL_STOP_POINT_F0

For frequency set 1: PI_CALVL_VREF_INITIAL_START_POINT_F1 and PI_CALVL_VREF_INITIAL_STOP_POINT_F1

For frequency set 2: PI_CALVL_VREF_INITIAL_START_POINT_F2 and PI_CALVL_VREF_INITIAL_STOP_POINT_F2

Recommendation is to use the nominal VRef value (based on the device programming of drive strength on the processor and termination in the memory) +/- 4%. Please use the online DDR Register Configuration Tool at http://dev.ti.com/sysconfig to program these registers and check the Revision History to ensure this workaround has been addressed in the version of the tool being used.

i2436 ***BCDMA: BCDMA RX_IGNORE_LONG setting in RX CHAN CFG register doesn't work***

Details:

RX_IGNORE_LONG flag in RXCHAN CFG register of BCDMA gets ignored and BCDMA reports errors in TR response when remote endpoints don't send EOP to match TR boundary.

Workaround(s):

RX_IGNORE_LONG is unusable, so remote endpoint such as PDMA should close packet by sending EOP to match TR boundary (PDMA X*Y*Z should match TR ICNT0*ICNT1*ICNT2*ICNT3)

If infinite stream is desired (PDMA Z=0) then switch to PKTDMA and use Single Buffer Mode

i2482 ***Boot: ROM does not provide enough clocks during SD card initialization***

Details:

ROM code is not providing 74 clocks before sending first command as specified in the SD Card Physical Layer Specification ver. 2.00. This may cause SD card boot to fail, however, a failure has never been observed due to this errata on affected devices.

Workaround(s):

None

i2464 ***Boot: ROM is unable to boot from SD cards with incorrect formatting***

Details:

MMCSD boot mode may be unable to boot from SD cards if the filesystem is formatted incorrectly during the creation of the filesystem.

Workaround(s):

When creating the SD from Ubuntu 22.04 system, add "-a" argument to the mkfs.vfat command to generate correct boot partitioning.

i2487 ***LPM: Low power modes may inadvertently corrupt DDR contents***

Details:

During entry to the Deep Sleep or RTC+IO+DDR low-power modes, the device may not properly transition the attached DDR into retention mode, which will lead to corruption of the DDR data.

Affected devices should not utilize these low power modes without applying the software workaround.

Workaround(s):

A workaround to this issue is provided in SDK versions 11.1 and later. The workaround ensures that an internal retention latch which controls DDR self-refresh is in the correct state through various reset scenarios.

i2493 ***MMCSD: HS200 write failures***

Details:

The MMC0 interface has the potential for write failures when issuing multiple block writes operating in HS200 mode with excessive IO supply noise.

i2493 (continued) ***MMCSD: HS200 write failures***

To minimize IO supply noise follow below best practices and refer to the linked application note:

- Use wide power planes/pours adjacent to ground layers with a thin dielectric between them.
- Place power planes/pours and adjacent ground planes as close to the surface of the powered components as possible.
- Use a wide variety of decoupling capacitor values and place low ESL capacitors as close to the decoupled device as possible.
- Use one decoupling capacitor per power pin.
- Use short and wide traces to decoupling capacitors and power/ground vias.
- Sitara™ Processor Power Distribution Networks: Implementation and Analysis

Workaround(s):

Implement a software recovery mechanism that re-issues the failed multiple block write with minimum 5µs delay between blocks to reduce noise. One way to achieve this delay is using single block writes for the failed multiple block write.

Trademarks

All trademarks are the property of their respective owners.

Revision History

Changes from December 11, 2024 to October 31, 2025 (from Revision * (December 2024) to Revision A (October 2025)) **Page**

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