Errata

J784S4 AM69x TDA4VH TDA4AH TDA4VP TDA4AP J742S2 TDA4VPE TDA4APE Processors 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 Modules Affected

Table 1-1 shows the module(s) that are affected by each usage note.

Table 1-1. Usage Note by Modules

MODULE	USAGE NOTE
DDR	i2330 — DDRSS Register Configuration Tool Updates
PLL	i2424 — PLL: Programming Sequence May Introduce PLL Instability
USB	i2134 — USB: 2.0 compliance receive sensitivity test limitation

Table 1-2 shows the module(s) that are affected by each advisory.

Table 1-2. Advisories by Modules

MODULE	ADVISORY
Boot	i2366 — Boot: ROM does not comprehend specific JEDEC SFDP features for 8D-8D-8D operation
	i2371 — Boot: ROM code may hang in UART boot mode during data transfer
	i2372 — Boot: ROM doesn't support select multi-plane addressing schemes in Serial NAND boot
	i2413 — HS-FS ROM boots corrupted ROM boot image
	i2414 — Ethernet PHY Scan and Bring-Up Flow doesn't work with PHYs that don't support Auto Negotiation
	i2415 — UART Backup Boot Authentication Failure w/ xSPI Primary Boot Mode
	i2419 — When disabling deskew calibration, ROM does not check if deskew calibration was enabled
	i2422 — ROM timeout for MMCSD filesystem boot too long
	i2435 — ROM timeout for eMMC boot too long
	i2482 — Boot: ROM does not provide enough clocks during SD card initialization
BCDMA	i2431 — BCDMA: RX Channel can lockup in certain scenarios
	i2436 — BCDMA: RX_IGNORE_LONG setting in RX CHAN CFG register doesn’t work
C7x SE	i2063 — C7x SE: VCOP Aliasing for CPU loads and stores is not supported for non-aligned accesses to the last line in the IBUF buffers.
	i2064 — C7x SE: DMA accesses to L1D SRAM may stall indefinitely in the presence cache mode change or global writeback, in specific conditions.
	i2065 — C7x SE: The C7x memory system and cpu may stall indefinitely, in the presence L1D snoops caused due to streaming engine reads, cache misses from MSMC or DDR, L1D victims, and some other specific conditions in a small time window.
	i2079 — C7x SE: DMA accesses to L1D SRAM may stall indefinitely in the presence of CPU traffic in specific conditions.
	i2087 — C7x MMA HWA_STATUS reports errors before application starts
	i2120 — C71x: SE Hangs on Non-Parity Error Detection in Transposed Streams With LEZR
	i2219 — C7x SE: SE Returning incorrect rstatus for uTLB faults
	i2271 — C7x SE: SE Can Hang on Page Fault/UMC Error Occurring During SEBRK
	i2272 — C7x SE: SE Corrupting First End-of-Stream Reference After SEBRK With FILLVAL Enabled
	i2399 — CPU NLC Module Not Clearing State on Interrupt
CPSW	i2401 — Host Timestamps Cause CPSW Port to Lock up
CSI	i2190 — CSI_RX_IF may enter unknown state following an incomplete frame
DDR	i2157 — DDR: Controller anomaly in setting wakeup time for low power states
	i2159 — DDR: VRCG high current mode must be used during LPDDR4 CBT
	i2160 — DDR: Valid VRef Range Must be Defined During LPDDR4 Command Bus Training
	i2166 — DDR: Entry and exit to/from Deep Sleep low-power state can cause PHY internal clock misalignment
	i2232 — DDR: Controller postpones more than allowed refreshes after frequency change
	i2244 — DDR: Valid stop value must be defined for write DQ VREF training
DRU	i2215 — DRU: TR Submission can be corrupted by C7x writes coming out of order if Non-Atomic TR Submission Mechanism is Used
DSS	i2097 — DSS: Disabling a Layer Connected to Overlay May Result in Synclost During the Next Frame

Table 1-2. Advisories by Modules (continued)

MODULE	ADVISORY
ECC AGGR	i2049 — ECC AGGR: Potential IP Clockstop/reset sequence hang due to pending ECC Aggregator interrupts
I3C	i2197 — I3C: Slave mode is not supported
	i2205 — I3C: Command fetched during pending IBI is not properly processed in some cases
	i2216 — I3C: Command execution may fail during slave-initiated IBI address byte reception
IA	i2196 — IA: Potential deadlock scenarios in IA
MCAN	i2278 — MCAN: Message Transmit order not guaranteed from dedicated Tx Buffers configured with same Message ID
	i2279 — MCAN: Specification Update for dedicated Tx Buffers and Tx Queues configured with same Message ID
MMCSD	i2312 — MMCSD: HS200 and SDR104 Command Timeout Window Too Small
MSMC	i2378 — MSMC: Cache/snoop filter way selection MMRs have incorrect reset values
	i2381 — MSMC: FFI reset allows target port to have backdoor access to the L3 data cache as mapped SRAM
OSPI	i2189 — OSPI: Controller PHY Tuning Algorithm
	i2249 — OSPI: Internal PHY Loopback and Internal Pad Loopback clocking modes with DDR timing inoperable
	i2351 — OSPI: Controller does not support Continuous Read mode with NAND Flash
	i2383 — OSPI: 2-byte address is not supported in PHY DDR mode
PCIe	i2242 — PCIe: The 4-L SerDes PCIe Reference Clock Output is temporarily disabled while changing Data Rates
	i2326 — PCIe: MAIN_PLLx operating in fractional mode, which is required for enabling SSC, is not compliant with PCIe Refclk jitter limits
PRG	i2253 — PRG: CTRL_MMR STAT registers are unreliable indicators of POK threshold failure
PSIL	i2137 — Clock stop operation can result in undefined behavior
R5FSS	i2161 — R5FSS: Debugger cannot access VIM module while it is active
RAT	i2062 — RAT: Error Interrupt Triggered Even When Error Logging Disable Is Set
	i2449 — RAT: R5FSS RAT MMRs Not Parity Protected
RINGACC	i2177 — RINGACC: The ring accelerator’s debug transaction trace stream can be corrupted by certain ring access sequences
Safety	i2103 — Safety Modules: Incorrect Reporting of ECC_GRP, ECC_BIT and ECC_TYPE Information for Functional Safety Errors
SE	i2437 — Clock-Gating Turning Off Too Early
SGMII	i2362 — 10-100M SGMII: Marvell PHY does not ignore the preamble byte resulting in link failure
UDMA	i2146 — UDMA: Force teardown bitfield readback is masked in realtime TX/RX registers
	i2234 — UDMA: TR15 hangs if ICNT0 is less than 64 bytes
	i2320 — UDMA, UDMAP: Descriptors and TRs required to be returned unfragmented
UDMAP	i2163 — UDMAP: UDMA transfers with ICNTs and/or src/dst addr NOT aligned to 64B fail when used in "event trigger" mode
UFS	i2102 — UFS: Auto-Hibernate can cause false entry/exit errors
	i2134 — USB: 2.0 compliance receive sensitivity test limitation
USART	i2310 — USART: Erroneous clear/trigger of timeout interrupt
	i2311 — USART: Spurious DMA Interrupts
USB	i2409 — USB2 PHY locks up due to short suspend

2 Nomenclature, Package Symbolization, and Revision Identification

2.1 Device and Development-Support Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all microprocessors (MPUs) and support tools. Each device has one of three prefixes: X, P, or null (no prefix) (for example, TDA4VH88T5AALYRQ1

). Texas Instruments recommends two of three possible prefix designators for its support tools: TMDX and TMDS. These prefixes represent evolutionary stages of product development from engineering prototypes (TMDX) through fully qualified production devices and tools (TMDS).

Device development evolutionary flow:

X Experimental device that is not necessarily representative of the final device's electrical specifications and may not use production assembly flow.

P Prototype device that is not necessarily the final silicon die and may not necessarily meet final electrical specifications.

null Production version of the silicon die that is fully qualified.

Support tool development evolutionary flow:

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

TMDS Fully-qualified development-support product.

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

"Developmental product is intended for internal evaluation purposes."

Production devices and TMDS development-support tools 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 (X or P) have a greater failure rate than the standard production devices. Texas Instruments 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.

For additional information how to read the complete device name for any J784S4 device, see the specific-device Datasheet (SPRSP79).

2.2 Devices Supported

This document supports the following devices:

- J784S4, TDA4AP, TDA4VP, TDA4AH, TDA4VH, AM69x
- J742S2, TDA4APE, TDA4VPE,

2.3 Package Symbolization and Revision Identification

Figure 2-1 shows an example of package symbolization.

Table 2-1 lists the device revision codes.

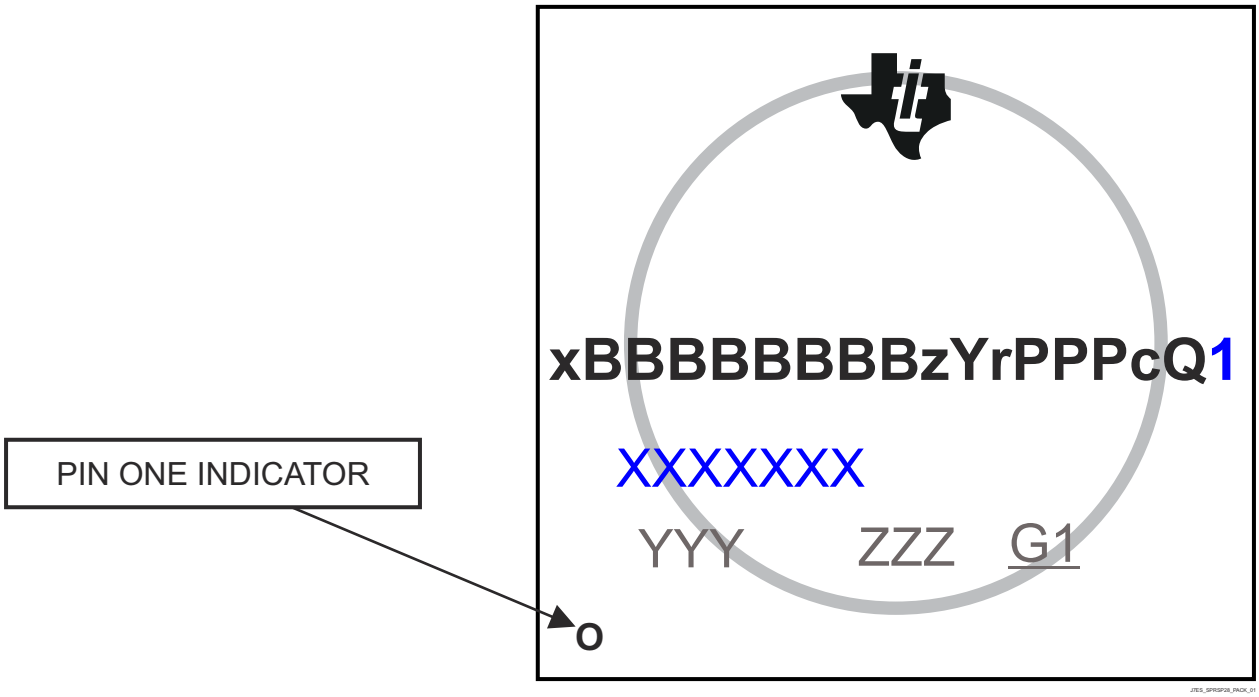

Figure 2-1. Package Symbolization

Table 2-1. Revision Identification

DEVICE REVISION CODE	SILICON REVISION	COMMENTS
A or BLANK	1.0	

3 Silicon Revision 1.0 Usage Notes and Advisories

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

3.1 Silicon Revision 1.0 Usage Notes

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 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):

Enable both of the following hardware workarounds.

Set cdr_eb_wr_reset bit (bit 7) to 1'b1 in UTMI_REG28 register present in USB*_PHY2 region.

Set phyrst_a_enable bit (bit 0) to 1'b1 in PHYRST_CFG register present in USB*_MMR_MMRVBP_USBSS_CMN region. Please note that phyrst_a_value (bits 12:8) in PHYRST_CFG register should be retained at default value of 0xE.

3.2 Silicon Revision 1.0 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

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

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:

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.

i2063 *C71x: VCOP Aliasing for CPU Loads and Stores Is Not Supported for Non-Aligned Accesses to the Last Line in the IBUF Buffers*

Details:

The C71x memory system supports EVE-style VCOP aliasing for CPU loads and stores, in addition to DMAs and accesses made through the streaming engine. When this aliasing is enabled, non-aligned loads and stores to the last line (128 bytes) in the IBUF buffers may not get aliased in some configurations.

Table 3-1 shows the actual behavior.

Table 3-1. Behavior of CPU Aliasing

CPU Aliasing ON					
	IBUFLA	IBUFHA	IBUFLB	IBUFHB	L1D Action
Owned	CPU	CPU	DMA	DMA	No issue
	DMA	DMA	CPU	CPU	No issue
	DMA	CPU	CPU	DMA	See (1)
	CPU	DMA	CPU	DMA	See (2)

(1) On a non-aligned acess to the last line in IBUFLA, where the line spills into IBUHA, both lines will get aliased.
(2) On a non-aligned acess to the last line in IBUFLA, where the line spills into IBUHA, both lines not will get aliased.

Workaround(s):

The IBUF buffers should be sized such that the last lines (128 bytes) for all the four buffers are not used.

i2064 *C71x: DMA Accesses to L1D SRAM May Stall Indefinitely in the Presence Cache Mode Change or Global Writeback in Specific Conditions*

Details:

DMA reads or writes to L1D SRAM may stall indefinitely. These transactions are required to sensitize this condition:

1. L1D Cache Mode Change or Global Writeback/Writeback w/ invalidate. These are initiated by ECR writes to CPU registers.
2. CPU loads while the cache mode change or global Writeback is in progress. This can be due to a CPU transaction that is scheduled in parallel with the MOVC instruction that writes to the ECR register.
3. DMA Reads or Writes to a buffer in L1D SRAM.

These transactions do not need to be to the same address, but #2 and #3 have to be in flight when #1 is in progress. In this case, the DMAs stall indefinitely even after the cache mode change or global Writeback finishes.

Workaround(s):

Avoid doing DMAs to buffers mapped to L1D SRAM.

i2065 *C71x: The C71x Memory System and CPU May Stall Indefinitely in the Presence L1D Snoops*

Details:

These are transactions and conditions that need to happen in a small time window.

Transactions:

1. Streaming engine reads to MSMC or DDR, which miss L2 cache, and go out as a read to MSMC for a line fill.

i2065 (continued) ***C71x: The C71x Memory System and CPU May Stall Indefinitely in the Presence L1D Snoops***

2. Streaming engine reads to MSMC or DDR, which miss L2 cache, but may be cached in L1D. These reads generate snoops to L1D.
3. CPU loads miss L1D and L1D sends them to L2 for cache line fills (multiple reads).
4. CPU loads or stores cause L1D to evict lines from its cache, resulting in victims to L2 (multiple victims).
5. L1D is responding to snoops, with snoop data.
6. MSMC is responding to the L2 misses with read response data.
7. Snoop responses from L1D (#5) and read response from MSMC (#6) are being routed to streaming engine.

Conditions/Stalls:

1. The L1D victims and snoop responses fill up the entire L1D pipeline and the buffers in L1D and L2, with the result that L1D is unable to send down any more victims or snoop responses to L2.
2. L2 is processing the read misses from L1D, but is unable to send back any more read response data to L1D since the L1D pipeline is full.

In this situation, the memory system stops servicing streaming engine reads. This can cause the CPU to stall indefinitely.

Workaround(s):

There are multiple ways in which this can be avoided. Removing any one transaction prevents this stall from happening. Any of these workarounds can be used. They are independent of each other, and applying even one workaround will avoid this condition.

Workaround 1: Flush the buffer from the L1D cache, before reading from the streaming engine, which eliminates L1D snoops.

Workaround 2: Prevents L1D snoops by not sharing buffers between L1D and Streaming engine.

Workaround 3: Flush the L1D victim cache to prevent L1D victims.

Workaround 4: Map either the streaming engine reads or the CPU loads to L2, instead of MSMC or DDR, thus avoiding cache misses.

i2079 ***C71x: DMA Accesses to L1D SRAM May Stall Indefinitely in the Presence of CPU Traffic in Specific Conditions***

Details:

DMA reads or writes to L1D SRAM may stall indefinitely. These transactions are required to sensitize this condition:

1. Buffer/line ‘A’ previously allocated in L1D cache.
2. CPU reads miss L1D cache to buffer/line ’A’.
3. Streaming engine reads to buffer/line ‘A’.
4. DMA reads or writes to a buffer in L1D SRAM.

Note that transactions #1, #2 and #3 are to the same buffer/line, while #4 is to a different buffer/line. This can encounter a condition which causes the DMAs to stall indefinitely.

Workaround(s):

Avoid doing DMAs to buffers mapped to L1D SRAM.

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
```

i2097 *DSS: Disabling a Layer Connected to Overlay May Result in Synclost During the Next Frame*

Details:

Disabling a layer (for example VID1) connected to an OVR (that is toggling DSS_VID_ATTRIBUTESx[0] ENABLE from 1 to 0) may result in synclost during the next frame. The synclost may result in a corrupted or blank frame (all pixel data sent out of DSS during the frame is 0x0). The occurrence of synclost is dependent on the timing of setting the GO bit (that is DSS_VP_CONTROL[5] GOBIT to 1) vis-à-vis the disabling of the layer. If the “disable layer” MMR write operation and “set GO bit” MMR write operation happens within the same frame boundary, no synclost occurs. If the operations happen across the frame boundary, then synclost occurs (for one frame). The design automatically recovers and returns to normal operation from the next frame after GO bit is set, see Figure 3-1.

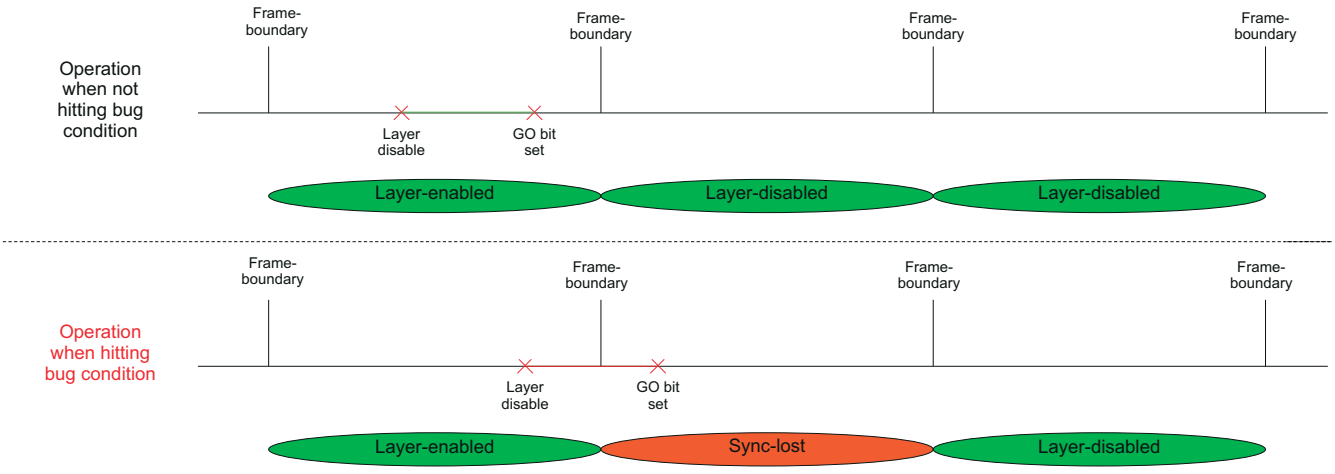

Figure 3-1. Bug Condition

Workaround(s):

A simple software workaround exists. In the workaround, prior to disabling a layer on the OVR, it is moved to the “non-visible” area of the OVR

i2097 (continued) ***DSS: Disabling a Layer Connected to Overlay May Result in Synclost During the Next Frame***

(for example: DSS_OVR_ATTRIBUTES_x[17-6] POSX = max_value_of_posx or DSS_OVR_ATTRIBUTES_x[30-19] POSY = max_value_of_posy). This avoids the synclost when the layer is disabled.

A sample software workaround pseudo-code is shown on Figure 3-2. In this case, the regular “disable layer” MMR write operation and “set GO bit set” MMR write operation are replaced with macros which implement the software workaround.

```
macro disable_layer (overlay n , layer m)
set OVR[n].ATTRIBUTES2[m].POSX = posx_max;
set OVR[n].ATTRIBUTES2[m].POSY = posy_max;
global_ovr_layer_disable_tracker[n][m] = 1;
endmacro

macro set_go_bit (vp n)
if(|(global_ovr_layer_disable_tracker[n])//any bit set
{
   set VP[n].CONTROL.GOBIT  = 1;
   Wait for 10 DSS FUNC CLK cycles;
   for (i=0;i<NUM_LAYERS;i++)
   {
     if(global_ovr_layer_disable_tracker[n][i])
     {
        Clear OVR[n].ATTRIBUTES[i].ENABLE = 0;
        global_ovr_layer_disable_tracker[n][i] = 0;
     }
   }
}
set VP[n].CONTROL.GOBIT  = 1;
endmacro
```

- Replace layer disable MMR write operation with a macro which positions the layer to the non-visible area of the OVR
- Track which layers are disabled. This will be used while GO bit is set

- Replace GO bit set MMR write operation with this macro
- First, set GO Bit for the changes in “disable_layer” macro (and any other earlier changes) to take effect
- After the first GO bit set, few idle_cycles (10 DSS functional clock cycles) are necessary before we move to the second step

- In the second step, actually disable the layers based on the previously tracked information
- Set the GO bit for the second time for the disable of the layers to take effect

Figure 3-2. Workaround Pseudo-code

i2102 ***UFS: Auto-Hibernate can cause false entry/exit errors***

Details:

The UFS module can falsely report that Hibernate entry/exit was unsuccessful during a successful Auto-Hibernate entry/exit process. These errors will be reported in the UFS_IS[6].UHES and UFS_IS[5].UHXS registers.

Workaround(s):

Software should disable the Auto-Hibernate feature permanently by setting the Auto-Hibernate Idle Time Value to zero via register field UFS_AHIT[9:0].AH8ITV.

i2103 ***Safety Modules: Incorrect Reporting of ECC_GRP, ECC_BIT and ECC_TYPE Information for Functional Safety Errors***

Details:

For functional safety errors, the logged information - ECC_GRP, ECC_BIT, and ECC_TYPE in the Error Status Registers may be incorrect for certain safety checkers. This only applies to safety checkers that map to ECC_GRP = 0,15,31,47,63...(N*16-1). In the case for the DDR Bridge/Controller, the issue only applies to the safety checkers where ECC_GRP = 0,31,63...(N*32-1).

This issue affects all Safety Module instances and their sub-banks. Refer to section Safety Modules of the device TRM.

Note: The detection and interrupt signaling of these safety errors is unaffected. Only the logging of the aforementioned fields of the Error Status Registers are affected.

i2103 (continued) ***Safety Modules: Incorrect Reporting of ECC_GRP, ECC_BIT and ECC_TYPE Information for Functional Safety Errors***

Workaround(s):

None. For these specific safety checkers, software is limited to knowing whether a correctable or uncorrectable error occurred and which Safety Module instance had the error (thus knowing the IP module), but not which exact safety checker encountered the error.

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 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.

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 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):

Enable both of the following hardware workarounds.

Set cdr_eb_wr_reset bit (bit 7) to 1'b1 in UTMI_REG28 register present in USB*_PHY2 region.

Set phyrst_a_enable bit (bit 0) to 1'b1 in PHYRST_CFG register present in USB*_MMR_MMRVBP_USBSS_CMN region. Please note that phyrst_a_value (bits 12:8) in PHYRST_CFG register should be retained at default value of 0xE.

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).

i2146 *UDMA: Force teardown bitfield readback is masked in realtime TX/RX registers*

Details:

The force teardown bit field will not remain set in the read back of the realtime TX/RX registers after a force teardown is initiated.

Workaround(s):

The Force Teardown operation is only used by software to intervene to address a catastrophic system condition, so software should separately track when it initiates a forced teardown verses a normal teardown, and thus not depend on the readback value of the force teardown bitfield to obtain this information.

i2157 *DDR: Controller Anomaly in Setting Wakeup Time for Low Power States*

Details:

The DDR controller may erroneously decrease the wakeup time for the present low power state if the wakeup time for the next deeper power state is either disabled, or set to a lower value.

Workaround(s):

If a particular low power state is enabled by setting a bit in the DDRSS_CTL_139[29-24] LPI_WAKEUP_EN bit field, all deeper power state bits must also be enabled. From bit 0 through 4, low power states go deeper and deeper as the bit number increases. For example, if bit 0 is set, all bits from 1 through 4 must also be set. Similarly, if bit 2 is set, bit 3 and 4 must also be set.

In addition, the following wakeup values must be programmed in increasing order:

1. LPI_CTRL_IDLE_WAKEUP_FN related to LPI_WAKEUP_EN[0] -> value should be less than all fields below
2. LPI_PD_WAKEUP_FN related to LPI_WAKEUP_EN[1] -> value should be less than all fields below
3. LPI_SR_SHORT_WAKEUP_FN, LPI_SR_LONG_WAKEUP_FN, LPI_SRPD_SHORT_WAKEUP_FN, LPI_SRPD_LONG_WAKEUP_FN related to LPI_WAKEUP_EN[2] -> value should be less than all fields below
4. LPI_SR_LONG_MCCLK_GATE_WAKEUP_FN, LPI_SRPD_LONG_MCCLK_GATE_WAKEUP_FN related to LPI_WAKEUP_EN[3] -> value should be less than all fields below
5. LPI_TIMER_WAKEUP_FN related to LPI_WAKEUP_EN[4] -> highest value,

where FN = F0, F1, and F2 for different frequency set points.

i2159 *DDR: VRCG High Current Mode Must be Used During LPDDR4 CBT*

Details:

The DDR PHY updates VREFca for the command/address bus during LPDDR4 Command Bus Training (CBT). Bit 3 in LPDDR4 Mode Register 13 (MR13) defines the VRef Current Generator (VRCG) mode inside the LPDDR4 device. If this bit is set to 0, the VREFca settling time is too long for subsequent operations to work properly. To ensure proper operation of CBT, bit 3 in MR13 must be set to 1 (VRef Fast Response high current mode) during CBT.

Workaround(s):

To ensure proper operation, VRef Fast Response high current mode should be enabled during both Command Bus Training (CBT) and Write DQ Vref Training. This can be done by setting the following fields to 1:

For chip select 0: PI_MR13_DATA_0[3] in the DDRSS_PI_259 register

For chip select 1: PI_MR13_DATA_1[3] in the DDRSS_PI_261 register

For chip select 2: PI_MR13_DATA_2[3] in the DDRSS_PI_263 register

For chip select 3: PI_MR13_DATA_3[3] in the DDRSS_PI_265 register

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.

i2161 *R5FSS: Debugger Cannot Access VIM Module While It Is Active*

Details:

This issue impacts the Vectored Interrupt Module (VIM) inside R5FSS. There are registers inside VIM which change the state of the IP when they are read (such as VIM_IRQVEC). The expected behavior is that only functional reads should cause the state change. Debug reads (generated by TI debug tools such as CCS) to these registers should leave the state as it is. An issue exists currently where VIM treats debug register reads in the same way as functional register reads. This can cause a debug operation (such as opening a VIM register memory window in CCS) to inadvertently change the state of the VIM IP, making debug ineffective.

i2161 (continued) ***R5FSS: Debugger Cannot Access VIM Module While It Is Active***

Workaround(s):

There is no work-around for this issue. The user should avoid accessing VIM registers while debugging.

i2163 ***UDMAP: UDMA transfers with ICNTs and/or src/dst addr NOT aligned to 64B fail when used in "event trigger" mode***

Details:

Note

The following description uses an example a C7x DSP core, but it applies to any other processing cores which can program the UDMA.

For DSP algorithm processing on C6x/C7x, the software often uses UDMA in NavSS or DRU in MSMC. In many cases, UDMA is used instead of DRU, because DRU channels are reserved in many use-cases for C7x/MMA deep learning operations. In a typical DSP algorithm processing, data is DMA'ed block by block to L2 memory for DSP, and DSP operates on the data in L2 memory instead of operating from DDR (through the cache). The typical DMA setup and event trigger for this operation is as below; this is referred to as "2D trigger and wait" in the following example.

For each "frame":

1. Setup a TR typically 3 or 4 dimension TR.
   a. Set TYPE = 4D_BLOCK_MOVE_REPACKING_INDIRECTION
   b. Set EVENT_SIZE = ICNT2_DEC
   c. Set TRIGGER0 = GLOBAL0
   d. Set TRIGGER0_TYPE = ICNT2_DEC
   e. Set TRIGGER1 = NONE
   f. ICNT0 x ICNT1 is block width x block height
   g. ICNT2 = number of blocks
   h. ICNT3 = 1
   i. src addr = DDR
   j. dst addr = C6x L2 memory
2. Submit this TR
   a. This TR starts a transfer on GLOBAL TRIGGER0 and transfers ICNT0xICNT1 bytes, then raises an event
3. For each block do the following:
   a. Trigger DMA by setting GLOBAL TRIGGER0
   b. Wait for the event that indicates that the block is transferred
   c. Do DSP processing

This sequence is a simplified sequence; in the actual algorithm, there can be multiple channels doing DDR to L2 or L2 DDR transfer in a "ping-pong" manner, such that DSP processing and DMA runs in parallel. The event itself is programmed appropriately at the channel OES registers, and the event status check is done using a free bit in IA for UDMA.

When the following conditions occur, the event in step 3.2 is not received for the first trigger:

- Condition 1: ICNT0xICT1 is NOT a multiple of 64.
- Condition 2: src or dst is NOT a multiple of 64.
- Condition 3: ICNT0xICT1 is NOT a multiple of 64 and src/dst address not a multiple of 64

Multiple of 16B or 32B for ICNT0xICNT1 and src/dst addr also has the same issue, where the event is not received. Only alignment of 64B makes it work.

Conditions in which it works:

- If ICNT0xICNT1 is made a multiple of 64 and src/dst address a multiple of 64, the test case passes.

i2163 (continued) ***UDMAP: UDMA transfers with ICNTs and/or src/dst addr NOT aligned to 64B fail when used in "event trigger" mode***

- If DRU is used instead of UDMA, then the test passes. You must submit the TR to DRU through the UDMA DRU external channel. With DRU and with ICNTs and src/dst addr unaligned, the user can trigger and get events as expected when TR is programmed such that the number of events and number of triggers in a frame is 1, i.e ICNT2 = 1 in above case or EVENT_SIZE = COMPLETION and trigger is NONE. Then the completion event occurs as expected. This is not feasible to be used by the use-cases in question.

Above is a example for "2D trigger and wait", the same constraint applies for "1D trigger and wait" and "3D trigger and wait":

- For "1D trigger and wait", ICNT0 MUST be multiple of 64
- For "3D trigger and wait", ICNT0xICNT1xICNT2 MUST be multiple of 64

Workaround(s):

Set the EOL flag in TR for UDMAP as shown in following example:

- 1D trigger and wait
  - TR.FLAGS |= CSL_FMK(UDMAP_TR_FLAGS_EOL, CSL_UDMAP_TR_FLAGS_EOL_ICNT0);
- 2D trigger and wait
  - TR.FLAGS |= CSL_FMK(UDMAP_TR_FLAGS_EOL, CSL_UDMAP_TR_FLAGS_EOL_ICNT0_ICNT1);
- 3D trigger and wait
  - TR.FLAGS |= CSL_FMK(UDMAP_TR_FLAGS_EOL,CSL_UDMAP_TR_FLAGS_EOL_ICNT0_ICNT1_ICNT2);

There is no performance impact due to this workaround.

i2166 ***DDR: Entry and exit to/from Deep Sleep low-power state can cause PHY internal clock misalignment***

Details:

When DDR PHY enters the Deep Sleep low-power state, there is a delay before the PHY PLL is disabled and gated off. If exit from Deep Sleep occurs before the PHY PLL is disabled, the PHY internal clocks can get misaligned with respect to each other, resulting in timing failures inside the PHY.

Workaround(s):

If using software-initiated low-power mode by writing to LP_CMD in the DENALI_CTL_132 register, ensure that when entry into low-power mode has been acknowledged, wait for a minimum of 160 DDR clock cycles before requesting an exit from low-power mode. Another option is to use the following workaround.

If using PSC to disable the DDR interface, ensure that after disabling of DDR interface has been acknowledged, wait for a minimum of 160 DDR clock cycles before sending a request to enable it. Another option is to use the following workaround.

If using the controller's automatic mechanism for low power entry/exit using LP_AUTO_ENTRY_EN in the DENALI_CTL_141 register, use the following workaround.

Workaround: Ensure that DDR PHY does not enter Deep Sleep low-power state.

i2166 (continued) ***DDR: Entry and exit to/from Deep Sleep low-power state can cause PHY internal clock misalignment***

This can be ensured by programming the value of PHY_LP_WAKEUP[3:0] in the DENALI_PHY_1318 register is greater than the values of all the following thresholds in DDR controller registers.

LPI_CTRL_IDLE_WAKEUP_FN, LPI_PD_WAKEUP_FN, LPI_SR_SHORT_WAKEUP_FN, LPI_SR_LONG_WAKEUP_FN, LPI_SRPD_SHORT_WAKEUP_FN, LPI_SRPD_LONG_WAKEUP_FN, LPI_SR_LONG_MCCLK_GATE_WAKEUP_FN, LPI_SRPD_LONG_MCCLK_GATE_WAKEUP_FN, and LPI_TIMER_WAKEUP_FN

where FN = F0, F1, and F2 for different frequency set points.

i2177 ***RINGACC: The ring accelerator's debug transaction trace stream can be corrupted by certain ring access sequences***

Details:

The Ring Accelerator allows for hardware assisted debug through direct debugger access of its memory space and by the ability to export a trace stream of its transactions out to the cptracer network. Typically this debug information is enabled, collected and analyzed using a JTAG based debugger which interfaces with the ring accelerator through the SOC debug fabric. An errata exists which can result in a corruption or a hang of the ring debug trace information. This failure can be triggered by normal ring peek operation or if the debugger is used to initiate a ring pop operation. The corruption signature for this errata is a peek wrongly being reported as a pop in the trace. Additionally during non-ring modes (message or credential) a normal ring pop operation can result in incorrect information in the trace's empty filed or a debug pop operation can result in incorrect destination address.

Workaround(s):

To use the Ring Accelerator's hardware trace features for development, code should avoid using ring peek operations and debugger initiated pop operations.

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

i2189 (continued) ***OSPI: Controller PHY Tuning Algorithm***

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.

i2190 ***CSI: CSI_RX_IF may enter unknown state following an incomplete frame***

Details:

When an incomplete frame with potential CRC error is received by the CSI2 interface, the module may enter an unknown state. In which case all the subsequent image frames will not be captured.

Workaround(s):

Reset the CSI_RX_IF module.

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

Workaround(s):

Figure 3-3 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.

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

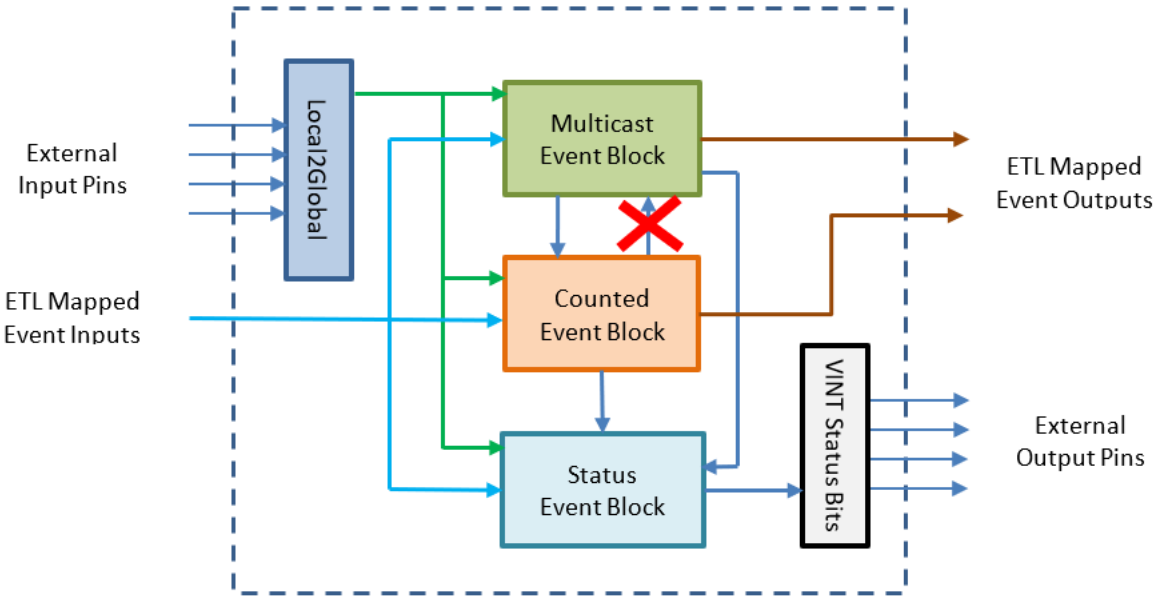

Figure 3-3. Interrupt Aggregator Version 1.0

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

i2197 ***I3C: Slave mode is not supported***

Details:

I3C Slave mode is not available. Only Master role on a single-master bus should be used.

Workaround(s):

None. Only Master role on a single-master bus should be used.

i2205 ***I3C: Command fetched during pending IBI is not properly processed in some cases***

Details:

Writing command by host during target-initiated IBI address byte reception may lead to improper command execution by controller, including incorrect frame generation.

Workaround(s):

Host must disable IBI by sending Broadcast DISEC CCC before sending commands to the controller.

i2215 ***DRU: TR Submission can be corrupted by C7x writes coming out of order if Non-Atomic TR Submission Mechanism is Used***

Details:

The C7x can allow writes to come out of order from how they are sent by the CPU. The Non-Atomic TR register in the DRU requires that the lowest byte of the TR is written last as that forces the other fields to be pushed into the TR queue. The out of order write will cause the wrong TR fields to be used if the last write does not come last causing for unexpected behavior from the DRU.

i2215 (continued) ***DRU: TR Submission can be corrupted by C7x writes coming out of order if Non-Atomic TR Submission Mechanism is Used***

Workaround(s):

The C7x should only use the Atomic TR submission method as this only requires a single 64 byte write for the TR submission.

i2216 ***I3C: Command execution may fail during slave-initiated IBI address byte reception***

Details:

An SoC host command to the I3C controller may lead to improper command execution by the controller, including incorrect frame generation, if the command was written while a slave-initiated IBI address byte reception is in progress.

In such case, the command response queue is incorrectly filled with responses. Additionally, if received IBI has no payload and is ACKed by Master, then slave fetched command causes incorrect frame issued over bus.

Workaround(s):

Host needs to disable IBI by sending Broadcast DISEC CCC before sending commands to the controller.

i2219 ***C7x SE: SE Returning incorrect rstatus for uTLB faults***

Details:

SE can overwrite previously recorded page faults before reporting them to the C7x CPU. This results in SE reporting page faults to the CPU with potentially corrupted error syndromes. While the accompanying error syndrome may be corrupted, when page faults occur they will always be reported with the correct failing virtual address.

Workaround(s):

If a page fault is returned by SE (IERR = 0x1, IESR[19:16] = 0x3), the user must analyze the system/software setup to determine the exact cause of failure without referencing the page fault syndrome (IESR[15:0]).

i2232 *DDR: Controller postpones more than allowed refreshes after frequency change*

Details

When dynamically switching from a higher to lower clock frequency, the rolling window counters that control the postponing of refresh commands are not loaded correctly to scale to the lower clock frequency. This will result in controller postponing more refresh commands than allowed by the DRAM specification, thus violating refresh requirement for the DRAM.

Workaround

Workaround 1:Disable dynamic frequency change by programing DFS_ENABLE = 0

Workaround 2:If switching frequency, program the register field values based on the pseudo code listed below.Note that the controller requires AREF_*_THRESHOLD values to be programmed before triggering initialization. Their values cannot be changed during mission mode after initialization . Therefore, the value of these parameters must be the lowest of all values needed for every frequency change transition planned to be used.

```
if (old_freq/new_freq >= 7){
    if (PBR_EN==1) { // Per-bank refresh is enabled
        AREF_HIGH_THRESHOLD = 19
        AREF_NORM_THRESHOLD = 18
        AREF_PBR_CONT_EN_THRESHOLD = 17
        AREF_CMD_MAX_PER_TREF = 8
    }
    else { // Per-bank refresh is disabled
        AREF_HIGH_THRESHOLD = 18
        AREF_NORM_THRESHOLD = 17
        // AREF_PBR_CONT_EN_THRESHOLD <=== don't care, PBR not enabled
        AREF_CMD_MAX_PER_TREF = 8
    }
}
else {
    AREF_HIGH_THRESHOLD = 21
    AREF_NORM_THRESHOLD //<=== keep AREF_NORM_THRESHOLD < AREF_HIGH_THRESHOLD
    AREF_CMD_MAX_PER_TREF = 8
    if (PBR_EN==1) { // Per-bank refresh is enabled
    //keep AREF_PBR_CONT_EN_THRESHOLD<AREF_NORM_THRESHOLD<AREF_HIGH_THRESHOLD
        AREF_PBR_CONT_EN_THRESHOLD
    }
}
```

i2234 *UDMA: TR15 hangs if ICNT0 is less than 64 bytes*

Details

The UDMA always attempts to send the burst size for a transaction. If the actual ICNT0 is less than the minimum burst size of 64 the UDMA will wait for data that is never coming and will hang. If the EOL is set in the TR then the UDMA always sends the data for the last data regardless of the size allowing for the transfer to be sent.

Workaround

This can be worked around by setting the EOL to 1 in the TR

i2242 — *PCIe: The SerDes PCIe Reference Clock Output is temporarily disabled while changing Data Rates*

Details

The SerDes PCIe Reference Clock Output will be temporarily disabled when changing Data Rates to or from 8.0 GT/s in Derived Refclk mode (as opposed to Received Refclk mode) and using a single SerDes PLL to generate the PCIe TX and RX clocks. This is due to the PLL reprogramming which must be performed when changing the data rate from 2.5 GT/s or 5.0 GT/s to 8.0 GT/s in this mode.

Some external PCIe components that are using the PCIe Reference Clock may not tolerate the disabling of the clock when changing data rates. However, the SerDes in this Device family does not have an issue accepting this Reference Clock behavior. This means that a link that connects the SerDes in one Device to the SerDes in a second Device will not have an issue when one Device generates the Reference Clock and the other Device receives the Reference Clock.

Workaround

Option 1:

Configure the SerDes to use one PLL to generate the clocks for 2.5 GT/s and 5.0 GT/s data rates, and a second PLL to generate the clocks for 8.0 GT/s data rate. This option imposes some limitations:

A) If Internal SSC mode is used, the two PLLs will not spread in sync with each other. This could result in up to 5000ppm difference between frequency of the two PLLs, and therefore between the TX and RX of the link partners. Because of this, Internal SSC mode is not recommended.

B) Protocols used simultaneously with PCIe on different Lanes of the SerDes must be compatible with sharing the PLL configuration of at least one of the two PLLs used for PCIe.

Option 2:

Use Received Refclk mode. Note that this mode is impacted by the separate Output Refclk jitter errata advisory (i2241)

Option 3:

Do not operate the PCIe interface at the 8.0 GT/s Data Rate

Option 4:

Use an external clock source to supply the PCIe Reference Clock to both the Root Complex and End Point Devices of the Link.

i2244 *DDR: Valid stop value must be defined for write DQ VREF training*

Details

The DDR PHY uses start, stop, and step-size values for write DQ VREF training. If the stop value is not equal to the start value + a multiple of the step-size, then the final VREF setting can go beyond the maximum VREF range, causing the training to hang.

Workaround

Program the stop value as follows:

PI_WDQLVL_VREF_INITIAL_STOP = (multiple of PI_WDQLVL_VREF_INITIAL_STEPSIZE) + PI_WDQLVL_VREF_INITIAL_START

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 3-2. 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)
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

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

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

initial clear, the ESM flags can be used as a reliable indicator of failure (or no failure) from the POKs.

i2271 ***C7x SE: SE Can Hang on Page Fault/UMC Error Occurring During SEBRK***

Details

When SE receives an error response from either the uTLB (page fault) or from UMC (2-bit error, addressing error, permissions error, etc.) for an active tag it halts execution of SE's fetching FSM. The final step in handling an SEBRK is to restart execution of that same FSM.

If both of these events occur with specific timing, then SE will not properly restart execution of the fetching FSM and SE will hang. This will result in the C7x CPU hanging on the next SE reference.

Workaround

Once hung, the only resolution is to reset the C7x corepac.

i2272 ***C7x SE: SE Corrupting First End-of-Stream Reference After SEBRK With FILLVAL Enabled***

Details

SE sends errors to the C7x CPU with accompanying zeroed out data. It zeroes out the data by de-asserting the clr_n pin of the interface-driving register for the data via a sticky "error is present to go to CPU" bit. After an SEBRK SE clears that error bit in preparation to send valid data again, but, in the case of SEBRK prematurely ending the stream, it ends up doing so after pushing the first end-of-stream reference to the interface.

Because of this, the first end-of-stream reference will be created with zeroed out data. When FILLVAL is enabled (set to non-zero fill mode), then that non-zero fill data is supposed to be sent after the end of the stream, but this bug corrupts it for the first reference by forcing the data to actual zero for a single cycle.

Workaround

There is no true workaround for this issue, but the scenario can be avoided by doing any of the following in a given stream:

Not using stream breaks that will end the stream prematurely

Not using end-of-stream references for computation

Specifically not using the first end-of-stream reference for computation

Not using the FILLVAL feature

This issue has been discussed with MMALIB/TIDL teams, who are currently the only users of the FILLVAL feature and the requestors to have it added for J7AEP and J7AHP. End-of-stream references are not utilized for computation, making this bug un-hittable, so performing an ECO to fix it is not currently planned.

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).

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

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:

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

i2310 (continued) ***USART: Erroneous clear/trigger of timeout interrupt***

- 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):

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.

i2320 ***UDMA and UDMAP : Descriptors and TRs required to be returned unfragmented***

Details

The UDMA and UDMAP require that the descriptors and TRs are placed in a memory subsystem that returns the descriptor or TR without any fragmenting of the descriptors. However, there are some memories that contain a fragmentation bridge, which makes them not available for holding the descriptors and TRs.

For this device, the R5 TCM memory cannot hold descriptors or TRs for UDMA or UDMAP

i2320 (continued) ***UDMA and UDMAP : Descriptors and TRs required to be returned unfragmented***

Workaround

None

i2326 ***PCIe: MAIN_PLLx operating in fractional mode, which is required for enabling SSC, is not compliant with PCIe Refclk jitter limits***

Details:

The MAIN_PLLx, which optionally supplies the 100MHz PCIe Refclk for SERDES and external components, does not comply to the PCIe Refclk jitter limits when configured in fractional mode. Fractional mode is required for enabling SSC, therefore SSC mode is not compliant to the PCIe Refclk jitter limits.

Workaround(s):

When sourcing the 100MHz PCIe Refclk from the MAIN_PLLx, the MAIN_PLLx should be configured in integer mode only (DACEN = 0, DSMEN = 0). This prevents the use of SSC for PCIe Refclk, which requires the PLL to operate in fractional mode. If SSC is required on the PCIe interface, an external Refclk generator with SSC should be used to provide the SERDES 100MHz Refclk.

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.

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.

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

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.

i2362 ***10-100M SGMII: Marvell PHY does not ignore the preamble byte resulting in link failure***

Details:

The CPSW SGMII module outputs up to 5 bytes of 0x50 preamble data when in 10/100 mode and there is an odd number of clocks between packets. All bytes should be 0x55. In 1000Mbps mode, which does not have the issue, there are seven 0x55's in the preamble previous to the SFD. In 100Mbps mode there are 70 bytes in the preamble before the SFD (because the data is replicated 10 times from 1000Mbps mode). The first five bytes of the seventy can be 0x50 when the issue occurs. This issue has been undetected until now due to testing only with the PHYs that allow the preamble to be eroded and don't care about the actual data in the first number of bytes However, this issue was recently detected with a Marvel PHY (88Q1111 or similar) that looks at the preamble data and makes packet keep/discard decisions based on preamble data of 0x50.

Workaround(s):

The workaround options are:

1. Use 1000M mode which does not have the issue.

OR

2. Use a TI PHY (DP83869 or similar) or any other PHY which can erode/ignore the preamble data in 10/100/1000M mode.

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

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

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.

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.

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.

i2378 ***MSMC: Cache/snoop filter way selection MMRs have incorrect reset values***

Details:

The shadow copies of the following two MSMC MMRs have the wrong reset value. The main copy that SW can read have the correct reset value, but MSMC functionality uses the value in the shadow copies. The incorrect reset values results in reduced DDR system performance. More specifically MSMC L3 data cache and snoop filter to be under-utilized for DDR accesses (utilization drops to 25% of expectations). The under-utilization extends to the L2 cache of the A72 [include in J7AEP-only: "and C7x"].

RT_WAY_SELECT [Address = 0x6E00_8000]

NRT_WAY_SELECT [Address = 0x6E00_8008]

Workaround(s):

SW needs to write the value (0x0000_0303) to both MSMC MMRs after reset. This is needed even if the MMRs appear to already have the correct value. Writing to the MMRs ensure that their shadow copies have the correct post-reset value.

i2381 ***MSMC: FFI reset allows target port to have backdoor access to the L3 data cache as mapped SRAM***

Details:

A target port on MSMC, after undergoing a FFI reset, can gain backdoor access to the L3 data cache content via the MSMC L3 SRAM memory mapped address for that location.

Workaround(s):

After completing the FFI reset sequence on the MSMC target port, software should re-write the current L3 cache size setting to the MSMC MMR - CACHE_CTRL.CACHE_SIZE field to prevent the HW from allowing backdoor accesses to the L3 data cache content.

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

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.

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

i2401 (continued) ***CPSW: Host Timestamps Cause CPSW Port to Lock up***

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.

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 USB controller suspend_residency_enable field in SUSP_CTRL to '1'.
3. Proceed with normal USB controller initialization

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.

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):

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

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.

i2415 ***Boot: UART Backup Boot Authentication Failure w/ xSPI Primary Boot Mode***

Details:

On HS-SE device type using a flash based primary boot mode which support redundant boot address like OSPI boot mode and a secondary boot mode like UART. Under the following condition:

Boot a valid image from backup boot media (UART) with below configuration:

1. Primary Image at 0x0 => Bad Image (fails authentication)
2. Redundant Image at 0x40_0000 => Valid TIFS image but not a ROM boot (fails authentication)
3. Backup boot mode => Valid Image (Expected Image to boot)

ROM is not able to boot the Valid image from the secondary boot mode, like UART boot media.

Under normal circumstance, after each time an image failed to boot, Secure ROM has to reset all the internal state machine for the next Retry operation.

When trying operate on the TIFS certificate, Secure ROM doesn’t reset all the necessary variables after the Image failed at Redundant offset.

Hence, during Backup boot flow Secure ROM was not able to authenticate the Certificate/ Image binary.

Due to this, Boot fails at UART backup boot for a Valid Image binary as well.

Workaround(s):

None, other than making sure the image located at the redundant offset is a complete boot certificate and not just a TIFS/SYSFW certificate.

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

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

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

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.

i2424 ***PLL: PLL Programming Sequence May Introduce PLL Instability***

Details:

PLL programming sequence has been changed to ensure that, if used, all calibration fields are configured prior to enabling the PLL calibration. In addition to the change to the control of the calibration logic, other changes are implemented so that PLL parameters are unchanged while the PLL is enabled.

When in integer mode, the software enables the PLL calibration feature on calibration-capable PLLs. The previous software adjusted calibration modes after CAL_LOCK was asserted. These writes have been observed to cause a loss of PLL lock on some devices. Additionally, even on susceptible devices, the loss of lock is intermittent, but when the loss occurs, dependent circuitry runs at an incorrect frequency; this wrong frequency can show up as slow algorithm execution or communication failures.

Limit on the impact: The calibration logic cannot be used when the PLL is in fractional mode. Therefore, PLLs that are programmed to use fractional mode should not see a failure related to the calibration programmation. Nevertheless, because of the change to the full PLL sequence, the new software is recommended for all users.

Workaround(s):

Do not use clk_pll_16fft_cal_option4() in SYSFW. Ensure to use updated PLL programming sequences in SDK v10.0 or later when performing any PLL configuration change.

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.

i2431 (continued) ***BCDMA: RX Channel can lockup in certain scenarios***

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.

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

i2437 ***SE Clock-Gating Turning Off Too Early***

Details:

A hardware bug is present in the C7120 Streaming Engine top level clock gating logic that can lead to the C7120 CPU hanging.

Hanging can occur regardless of Streaming Engine Programming and can only be avoided by overriding the top level clock gating to prevent Streaming Engine and other C7120 Corepac components from going idle.

Workaround(s):

The DSP_<COREID>_DEBUG_CLKEN_OVERRIDE fields of the COMPUTE_CLUSTER_CFG_WRAP_0_CC_CNTRL register (where COREID is the

i2437 (continued) ***SE Clock-Gating Turning Off Too Early***

name of the specific C7120 core) must be enabled before power-up of the C7120 core to override all clock-gating.

i2449 ***RAT: R5FSS RAT MMRs Not Parity Protected***

Details:

Values stored in R5FSS RAT MMRs are not parity protected while stored. This means a bit flip in the MMR, even when parity protection is enabled, would not be detected, so there is no protection from permanent or transient errors. Initiator parity checking (where the parity is calculated dynamically from values stored in the MMR at the time of read) only covers errors introduced over the interconnect. It does not cover errors that may exist in the stored MMR values themselves.

Workaround:

User need to perform a Software readback of MMR values during runtime.

i2459 ***Boot: PCIe Boot Mode is not supported***

Details:

PCIe Boot Mode is not supported and should not be used. It will be marked as Reserved in future revisions of the TRM.

Workaround(s):

None. An alternative Boot Mode should be selected.

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

Trademarks

All trademarks are the property of their respective owners.

Revision History

Changes from July 31, 2024 to June 15, 2026 (from Revision B (July 2024) to Revision C (June 2026)) **Page**

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), 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 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, TI’s General Quality Guidelines, 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. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

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

Last updated 10/2025