

# Errata

## J721S2/TDA4VE/TDA4AL/TDA4VL/AM68A Processor 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.....	2
2 Nomenclature, Package Symbolization, and Revision Identification.....	4
3 Silicon Revision 1.0 Usage Notes and Advisories.....	6
Revision History.....	42

## 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
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	i2307 — Boot: ROM does not properly select OSPI clocking modes based on BOOTMODE
	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
	i2414 — Ethernet PHY Scan and Bring-Up Flow doesn't work with PHYs that don't support Auto Negotiation
	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
	C7x SE
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.	
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	
CBASS	i2235 — CBASS Null Error Interrupt Not Masked By Enable Register
CP	i2283 — Restrictions on how CP Tracer Debug Probes can be used
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
	i2316 — DDR: MCU_OSPIO_* pins are limited to 1.8V operation when DDR1 is used
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
ECC AGGR	i2049 — ECC AGGR: Potential IP Clockstop/reset sequence hang due to pending ECC Aggregator interrupts
GPMC	i2313 — GPMC: Sub-32-bit read issue with NAND and FPGA/FIFO
I3C	i2197 — I3C: Slave mode is not supported
	i2205 — I3C: Command fetched during pending IBI is not properly processed in some cases

**Table 1-2. Advisories by Modules (continued)**

MODULE	ADVISORY
	i2216 — I3C: Command execution may fail during slave-initiated IBI address byte reception
IA	i2196 — IA: Potential deadlock scenarios in IA
Internal Diagnostics Modules	i2103 — Internal Diagnostics Modules: Incorrect Reporting of ECC_GRP, ECC_BIT and ECC_TYPE Information for Functional Safety Errors
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
MDIO	i2329 — MDIO: MDIO interface corruption (CPSW and PRU-ICSS)
MMCSDB	i2312 — MMCSDB: 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	i2237 — PCIe: SerDes Reference Clock Output does not comply to Vcross, Rise-Fall Matching, and Edge Rate limits
	i2242 — PCIe: The 4-L SerDes PCIe Reference Clock Output is temporarily disabled while changing Data Rates
	i2243 — PCIe: Timing requirement for disabling output refclk during L1.2 substate is not met
	i2308 — PCIe: PCIE_REFCLK mis-wired in package
	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
RINGACC	i2177 — RINGACC: The ring accelerator's debug transaction trace stream can be corrupted by certain ring access sequences
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
	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
	i2234 — UDMA: TR15 hangs if ICNT0 is less than 64 bytes
USART	i2310 — USART: Erroneous clear/trigger of timeout interrupt
	i2311 — USART: Spurious DMA Interrupts
USB	i2091 — 2.0 PHY hangs if received signal amplitude crosses squelch threshold multiple times within the same packet
	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, TDA4AL88TGAALZQ1

). 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 J721S2 device, see the specific-device Datasheet (SPRSP62).

### 2.2 Devices Supported

This document supports the following devices:

- J721S2, TDA4VE, TDA4AL, TDA4VL, AM68

### 2.3 Package Symbolization and Revision Identification

Figure 2-1 shows an example of package symbolization.

Table 2-1 lists the device revision codes.

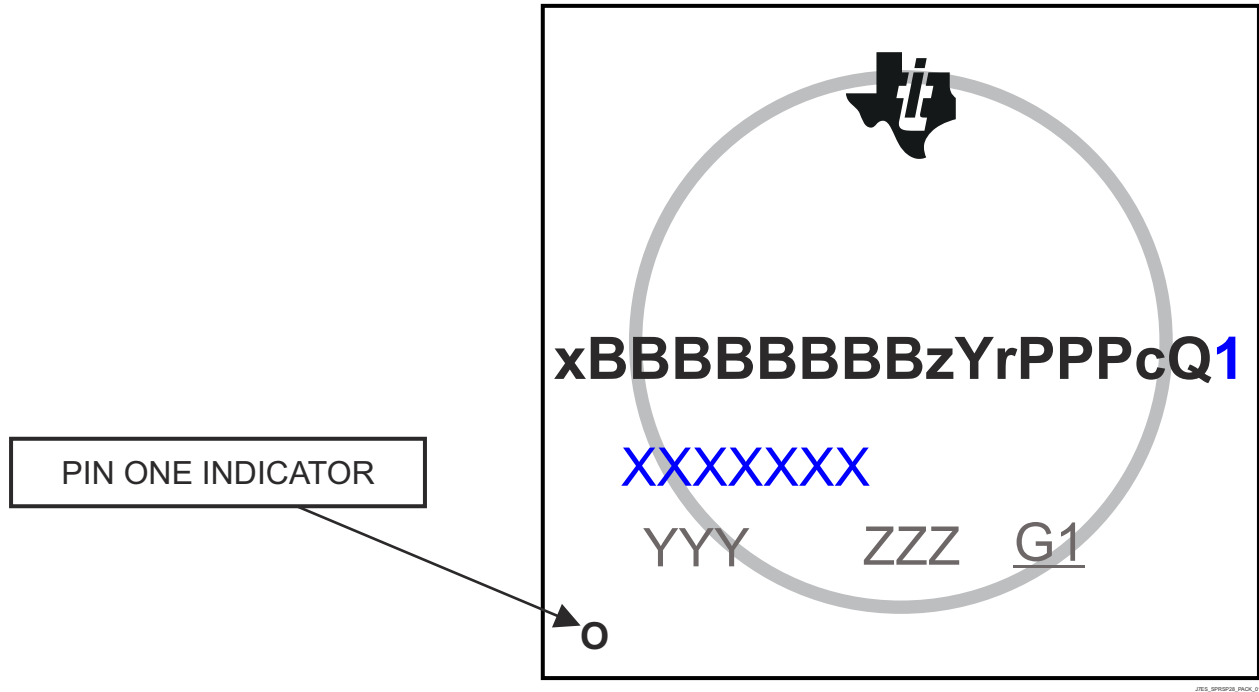


Figure 2-1. Package Symbolization

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[REVM AJ].[REVM IN].[REVRTL]

ECC\_AGGR versions before v2.1.1 are affected. ECC\_AGGR versions v2.1.1 and later are not affected.

Affected Example:

REVM AJ = 2

REVM IN = 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

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

---

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 access to the last line in IBUFLA, where the line spills into IBUHA, both lines will get aliased.
- (2) On a non-aligned access 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.
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).



**i2065** (continued)

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

---

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.

**i2091**

***USB: 2.0 PHY Hangs if Received Signal Amplitude Crosses Squelch Threshold Mmultiple Times Within the Same Packet***

---

**Details:**

USB 2.0 PHY implements a squelch detection circuit on the receiver to ensure noise is not interpreted as valid data when the bus is idle. The squelch circuit blocks invalid data

**i2091 (continued) USB: 2.0 PHY Hangs if Received Signal Amplitude Crosses Squelch Threshold Multiple Times Within the Same Packet**

by disabling the receiver output while the DP/DM differential signal amplitude is less than the squelch threshold.

The PHY may hang if the DP/DM differential signal amplitude drops below the squelch threshold for a brief period of time and increases back above the squelch threshold within the same packet. The issue does not occur if the DP/DM differential signal amplitude crosses the squelch threshold during the idle time between two packets.

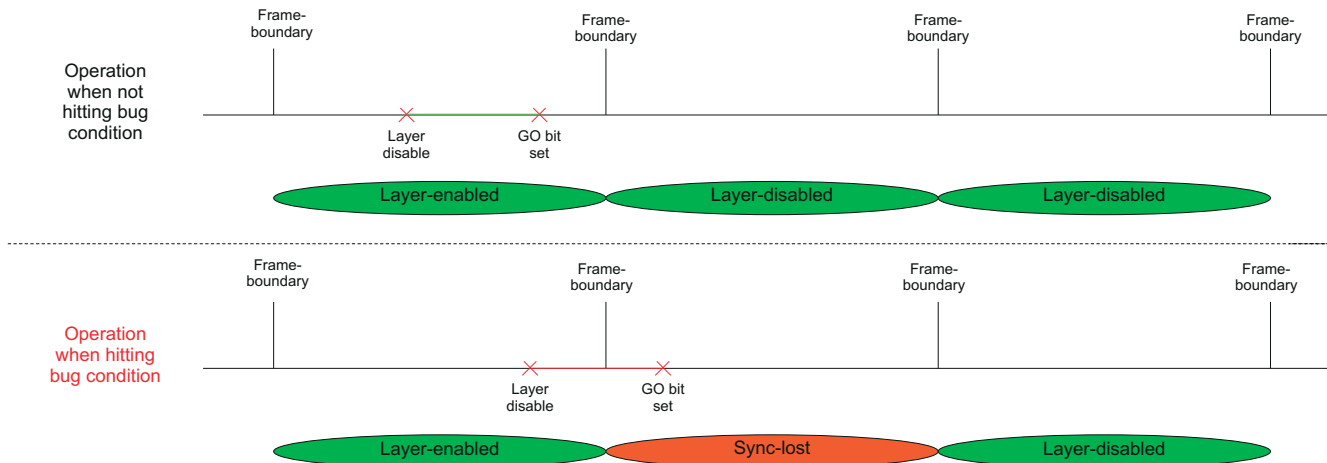
**Workaround(s):**

The issue can be avoided by ensuring the DP/DM differential signal amplitude applied to the receiver input remains above the squelch threshold during valid data transfers.

**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 (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.

**i2097** (continued)

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

```

macro disable_layer (overlay n , layer m)
set OVR[n].ATTRIBUTES2[m].PO SX = posx_max;
set OVR[n].ATTRIBUTES2[m].PO SY = 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;<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**

**i2103**

**Internal Diagnostics 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 Internal Diagnostics Module instances and their sub-banks.

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.

**Workaround(s):**

None. For these specific safety checkers, software is limited to knowing whether a correctable or uncorrectable error occurred and which Internal Diagnostics 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.

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


---

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.

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

---

**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 versus 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: DDRSS\_PI\_199[6-0] PI\_CALVL\_VREF\_INITIAL\_START\_POINT\_F0 and DDRSS\_PI\_199[14-8] PI\_CALVL\_VREF\_INITIAL\_STOP\_POINT\_F0 bit fields.

For frequency set 1: DDRSS\_PI\_199[22-16] PI\_CALVL\_VREF\_INITIAL\_START\_POINT\_F1 and DDRSS\_PI\_199[30-24] PI\_CALVL\_VREF\_INITIAL\_STOP\_POINT\_F1 bit fields.

For frequency set 2: DDRSS\_PI\_200[6-0] PI\_CALVL\_VREF\_INITIAL\_START\_POINT\_F2 and DDRSS\_PI\_200[14-8] PI\_CALVL\_VREF\_INITIAL\_STOP\_POINT\_F2 bit fields.

Recommendation is to use the nominal VRef value (based on the device programming of VDDQ/3 or VDDQ/2.5 along with the drive/termination settings used) +/- 4%.

**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.

**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: ICNT0xICNT1 is NOT a multiple of 64.
- Condition 2: src or dst is NOT a multiple of 64.
- Condition 3: ICNT0xICNT1 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.
- 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

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

---

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.

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.



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

---

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 field 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. In order to guarantee 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, it is necessary 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.

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


---

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:**

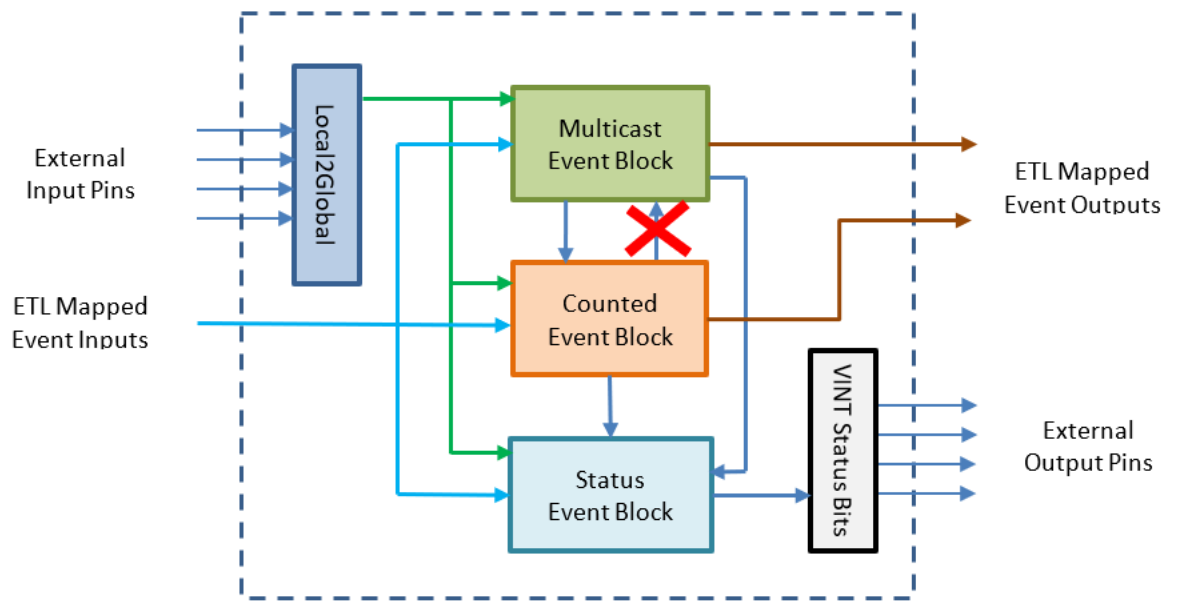
The interrupt Aggregator (IA) has one main function, which is to convert events arriving on the Event Transport Lane (ETL) bus, can convert them to interrupt status bits which are used to generate level interrupts. The block that performed this function in IA version 1.0 was called the status event block.

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 programming 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

**i2235****CBASS Null Error Interrupt Not Masked By Enable Register**

---

**Details**

There is optional feature in CBASS that adds the null error reporting MMR and interrupt source. When the feature is present and the interrupt is enabled, these two output ports: "err\_intr\_intr" (level interrupt source) and "err\_intr\_pls\_intr" (pulse interrupt source) will be asserted when an access to a null region occurs. The enable for the interrupt is in the ERR\_INTR\_ENABLE\_SET register (address offset 0x58).

The issue is CBASS ignores this enable bit, and as a result any null access always produces the interrupt sources/events.

**Workaround**

There is no spurious event due to this bug because of the default disable status of processor events. At system level, processors don't receive any event unless it's enabled in the associated GIC/VIM interrupt controller.

When the interrupt is enabled, and an interrupt does occur, write to the following registers at cbass level to clear it:

write 0x1 to the err\_intr\_enabled\_stat register, then write 0x1 to the err\_eoi register.

**i2237**

**PCIe: SERDES0\_REFCLK\_P/N SerDes Reference Clock Output does not comply to Vcross, Rise-Fall Matching, and Edge Rate limits**

---

**Details**

The SERDES0\_REFCLK\_P/N PCIe Reference Clock Output of the SerDes does not comply with the PCI-SIG specifications for VCROSS and Edge Rate limits. Therefore, some external PCIe components may have an issue receiving and using the Reference Clock. However, the SerDes in this Device family does not have an issue accepting this non-compliant Reference Clock. 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**

The PCIE\_REFCLK1\_P/N should instead be used to output the PCIe Refclk.

**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.



**i2243**

***PCIe: Timing requirement for disabling output refclk during L1.2 substate is not met***

---

**Details**

PCIe base specification requires Refclk to reach idle electrical state within 100ns of CLKREQ# deassertion when entering L1.2 substate (please refer TL10\_REFCLK\_OFF parameter).

This timing requirement cannot be met when sourcing Refclk from the device since hardware does not automatically gate Refclk. Refclk gating has to be performed by software by writing to PHY\_EN\_REFCLK field in SERDES\_RST register.

As a result, Refclk cannot be gated in L1.2 substate. Generally, allowing Refclk to run in L1.2 substate is not expected to cause any functional issues. However, if gating Refclk within 100ns is required by the system, L1.2 substate cannot be supported.

**Workaround**

Use an external Refclk generator to supply the PCIe reference clock

**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:

$$PI\_WDQLVL\_VREF\_INITIAL\_STOP = (\text{multiple of } PI\_WDQLVL\_VREF\_INITIAL\_STEPSIZE) + PI\_WDQLVL\_VREF\_INITIAL\_START$$
**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 not guaranteed 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 not guaranteed 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.

**i2283 *Restrictions on how CP Tracer Debug Probes can be used***

---

**Details**

Some CP Tracer bus probes do not receive the full SoC physical address but only a minimal set that was relevant to endpoint being monitored. This limits the usefulness of the probe in the SoC Analysis > Traffic Profiling feature in CCS.

1) Address Filtering / Matching : User would typically input the full 36b/40b (depending on device) address for any address-qualified bus probe jobs.

2) Decoding of transaction trace : User would expect that the address provided in the decoded stream was the full 36b/40b physical address of the transaction.

Affected probes:

J7AEP: SoC Onchip SRAM, MCU FlashSS

**i2283** (continued) ***Restrictions on how CP Tracer Debug Probes can be used***

---

**Workaround** none

**i2307** ***Boot: ROM does not properly select OSPI clocking modes based on BOOTMODE***

---

**Details** The ROM bootloader only selects an internal loopback mode for SPI/QSPI/OSPI/xSPI boot, regardless of the lclk field value selected by the BOOTMODE pins (see the device specific TRM for BOOTMODE pin mappings), which is intended to allow the user to choose an internal or external clocking method. This results in less flexibility in board topology in customers designs. Customers intending to use the external board loopback mode could see timing issues in ROM boot because the external loopback clock is not being used.

**Workaround** The topology of the OSPI design must not use "External Board Loopback" if planning to use OSPI as a boot source. All other clocking topologies (including internal loopback or DQS) can be used. Refer to the device specific datasheet, section "Applications, Implementation, and Layout" for supported clocking topologies using OSPI.

**i2308** ***PCIe: PCIE\_REFCLK mis-wired in package***

---

**Details** The die supports 4 pairs of PCIe Refclk output pads, PCIE\_REFCLK[3:0]. However, the package only supports one output, PCIE\_REFCLK1. The PCIE\_REFCLK3 pads are incorrectly connected to the PCIE\_REFCLK1\_P/N\_OUT package pins instead of the PCIE\_REFCLK1 pads.

**Workaround** Enable the PCIE\_REFCLK1\_P/N\_OUT pins and select the desired source by setting the PCIE\_REFCLK3\_CLKSEL register instead of the PCIE\_REFCLK1\_CLKSEL register.

This will be fixed as a rolling change to the package substrate. The package will be modified to connect the die PCIE\_REFCLK1 pads to the package PCIE\_REFCLK1\_P/N\_OUT output pins.

Software can future proof for this change by setting both the PCIE\_REFCLK1\_CLKSEL and PCIE\_REFCLK3\_CLKSEL registers. Production software on the updated package should only set PCIE\_REFCLK1\_CLKSEL.

**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 will retrigger 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

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


---

- 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 will retrigger 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 will cause DMA to be torn down by the SW driver
  - Valid since next incoming data will cause 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      *MMCSDB: 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 MMCSDB\_SYSCTL[19:16] DTO = 0xE is  $(1/192\text{MHz}) \times 2^{27} = 700\text{ms}$ . 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 (MMCSDB\_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.

**i2313      *GPMC: Sub-32-bit read issue with NAND and FPGA/FIFO***


---

**Details:**

Sub-32-bit reads on the GPMC interface will miss portions of the data, which will result in incorrect read data. This includes 8-bit or 16-bit reads from a NAND device or from an FPGA or FIFO interface. Note that 3-byte accesses are not allowed on the GPMC interface.

**Workaround(s):**

**i2313 (continued) *GPMC: Sub-32-bit read issue with NAND and FPGA/FIFO***

---

Read accesses on the GPMC interface must be performed as 32-bit reads. Writes are not affected by this erratum.

**i2316 *DDR: MCU\_OSPI0\_\* pins are limited to 1.8V operation when DDR1 is used***

---

**Details**

The MCU\_OSPI0\_\* pins are limited to 1.8V operation when DDR1 is used, due to an issue with crosstalk in the package. The MCU\_OSPI0\_\* pins are powered by the VDDSHV1\_MCU rail, which also powers the MCU\_OSPI1\_\* pins.

**Workaround**

There are two options to Workaround this issue if DDR1 is used:

- Power the VDDSHV1\_MCU rail at 1.8V. All of the MCU\_OSPI0/1\_\* pins can be operated as normal.
- Power the VDDSHV1\_MCU rail at 3.3V. The MCU\_OSPI1\_\* pins can be operated as normal, but the MCU\_OSPI0\_\* pins should be held static with no activity/toggling.

**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

**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.

**i2329 *MDIO: MDIO interface corruption (CPSW and PRU-ICSS)***

---

**Details:**

It is possible that the MDIO interface of all instances of CPSW and PRU-ICSS peripherals (if present) returns corrupt read data on MDIO reads (e.g. returning stale or previous data), or sends incorrect data on MDIO writes. It is also possible that the MDIO interface



**i2329 (continued)      MDIO: MDIO interface corruption (CPSW and PRU-ICSS)**


---

becomes unavailable until the next peripheral reset (either by LPSC reset or global device reset with reset isolation disabled in case of CPSW).

Possible system level manifestations of this issue could be (1) erroneous ethernet PHY link down status (2) inability to properly configure an ethernet PHY over MDIO (3) incorrect PHY detection (e.g. wrong address) (4) read or write timeouts when attempting to configure PHY over MDIO.

For boot mode (only CPSW if supported), there is no workaround to guarantee the primary ethernet boot is successful. If this exception occurs during primary boot, the boot may possibly initiate retries which may or may not be successful. If the retries are unsuccessful, this would result in an eventual timeout and transition to the backup boot mode (if one is selected). If no backup boot mode is selected, then such failure will result in a timeout and force device reset via chip watchdog after which the complete boot process will restart again.

To select a backup boot option (if supported), populate the appropriate pull resistors on the boot mode pins. See boot documentation for each specific device options, but the typical timeout for primary boot attempts over ethernet is 60 seconds.

**Workaround(s):**

On affected devices, following workaround should be used:

**MDIO manual mode: applicable for PRU-ICSS and for CPSW.**

MDIO protocol can be emulated by reading and writing to the appropriate bits within the MDIO\_MANUAL\_IF\_REG register of the MDIO peripheral to directly manipulate the MDIO clock and data pins. Refer to TRM for full details of manual mode register bits and their function.

In this case the device pin multiplexing should be configured to allow the IO to be controlled by the CPSW or PRU-ICSS peripherals (same as in normal intended operation), but the MDIO state machine must be disabled by ensuring MDIO\_CONTROL\_REG.ENABLE bit is 0 in the MDIO\_CONTROL\_REG and enable manual mode by setting MDIO\_POLL\_REG.MANUALMODE bit to 1.

*Contact TI regarding implementation of software workaround.*

---

**Note**

If using Ethernet DLR (Device Level Ring) (on CPSW or PRU-ICSS) or EtherCat protocol (on PRU-ICSS) there may be significant CPU or PRU loading impact to implement the run-time workaround 1 due to required polling interval for link status checks. Resulting system impact should be considered.

---

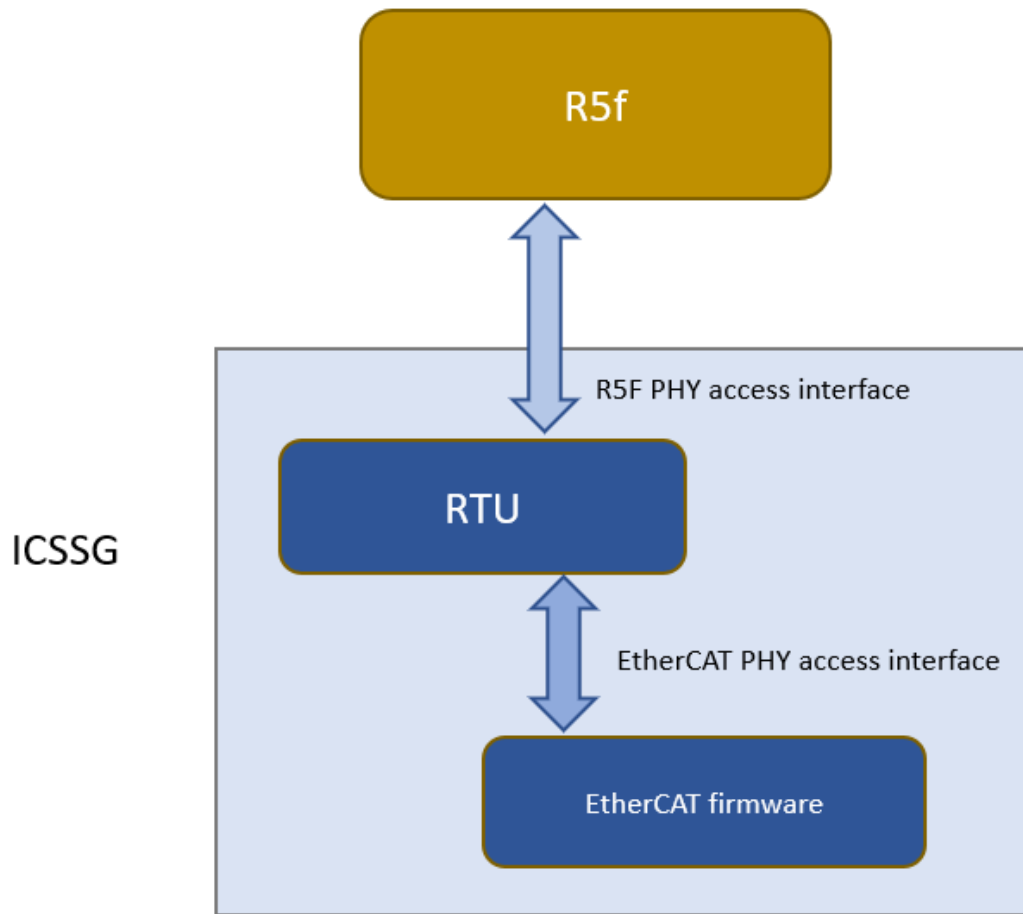
In case of PRU-ICSS, the loading of the software workaround may be reduced by using the MLINK feature of MDIO to do automatic polling of link status via the MIIx\_RXLINK input pin to PRU-ICSS which must be connected to a status output from the external PHY which does not toggle while the link is active. Depending on the specified behavior of the external PHY device, this PHY status output may be LED\_LINK or LED\_SPEED or the logic OR of LED\_LINK and LED SPEED. Refer to the MDIO section of TRM for details on using the MLINK feature of MDIO. This feature is not available on the CPSW peripheral.

For EtherCAT implementation on PRU-ICSS, the software workaround will be done in RTUx/ TX\_PRUx Core. The core will have to be dedicated for workaround, which means this can't be used for other purpose. The implementation will support two user access channels for MDIO access. This provides option for R5f core and PRU core to have independent access channel. The APIs will be similar to the ones we will have in RTOS Workaround implementation.



**i2329 (continued) MDIO: MDIO interface corruption (CPSW and PRU-ICSS)**

EtherCAT will continue to use PHY fast link detection via MDIO MLINK bypassing state m/c for link status (as this path is not affected by errata). This makes sure that cable redundancy related latency requirements are still met.



**Figure 3-4. MDIO Emulation via Manual Mode using PRU Core**

**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.

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

---

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 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 programming 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:

**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.

**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.

**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 will take 10 seconds.

**Workaround(s):**

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

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



## Trademarks

All trademarks are the property of their respective owners.

## Revision History

### Changes from June 10, 2023 to July 31, 2024 (from Revision B (June 2023) to Revision C (July 2024))

	Page
• Removed Advisory i2245; DMSC: Firewall Region requires specific configuration.....	2
• Updated Devices Supported section.....	4
• Updated Workaround for Advisory i2159, DDR: VRCG High Current Mode Must be Used During LPDDR4 CBT.....	13
• Updated Details and Workaround for i2242, PCIe: The SerDes PCIe Reference Clock Output is temporarily disabled while changing Data Rates.....	24
• Updated Workaround for Advisory i2326; PCIe: MAIN_PLLx operating in fractional mode, which is required for enabling SSC, is not compliant with PCIe Refclk jitter limits.....	31
• Added Advisory i2399: C7x: CPU NLC Module Not Clearing State on Interrupt.....	38
• Added Advisory i2401: CPSW: Host Timestamps Cause CPSW Port to Lock up.....	38
• Added Advisory i2409: USB2 PHY locks up due to short suspend.....	39
• Added Advisory i2414: Boot: Ethernet PHY Scan and Bring-Up Flow doesn't work with PHYs that don't support Auto Negotiation.....	39
• Added Advisory i2419: Boot: When disabling deskew calibration, ROM does not check if deskew calibration was enabled.....	39
• Added Advisory i2422: Boot: ROM timeout for MMCSD filesystem boot too long.....	40
• Added Advisory i2435: Boot: ROM timeout for eMMC boot too long.....	40
• Added Advisory i2437: SE Clock-Gating Turning Off Too Early.....	40

## 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](https://www.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  
Copyright © 2024, Texas Instruments Incorporated