Errata

AM65x Processors
Silicon Revision 2.1, 2.0, 1.0

ABSTRACT

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

Table of Contents

1 Usage Notes and Advisories Matrices

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

Table 1-1. Usage Notes Matrix

ID	DESCRIPTION	SILICON REVISIONS AFFECTED		
		AM65x		
		2.1	2.0	1.0
i2033	Section 3.1.1 — Fail-Safe IO's: Latch-up Risk on Fail-Safe IOs	Yes	Yes	Yes
i2082	Section 3.1.2 — ADC: High Input Leakage Current May Impact ADC Accuracy			Yes
i2007	Section 3.1.3 — INTRTR: Spurious Interrupts Generated when Programming Certain Interrupt Routers	Yes	Yes	Yes
i2351	Section 3.1.4 — OSPI: Controller does not support Continuous Read mode with NAND Flash	Yes	Yes	Yes

Advisories Matrix

MODULE	DESCRIPTION	SILICON REVISIONS AFFECTED		
		AM65x		
		2.1	2.0	1.0
ADC	i2151 — ADC: Debounce time control register		Yes	Yes
Boot, USB3SS	i2019 — Boot, USB3SS: Boot ROM Does Not Support USB Host MSC (Mass Storage Class) Boot Mode			Yes
	i2020 — Boot, USB3SS: Boot ROM Does Not Support USB Device Firmware Upgrade (DFU) Boot Mode			Yes
Boot, UART	i2030 — Boot, UART: UART Boot Mode Never Times Out			Yes
Boot	i2038 — Boot: FAT16 Fails When Root Block Resides in More Than One Cluster	Yes	Yes	Yes
	i2307 — Boot: ROM does not properly select OSPI clocking modes based on BOOTMODE	Yes	Yes	Yes
	i2328 — Boot: USB MSC boots intermittently	Yes	Yes	Yes
	i2371 — Boot: ROM code may hang in UART boot mode during data transfer	Yes	Yes	Yes
CBASS	i2207 — CBASS: Command Arbitration Blocking	Yes	Yes	Yes
CC_ARMSS	i2069 — CC_ARMSS: Powering Down CC_ARMSS1 Causes System Data Corruption			Yes
Cortex-R5F	i2099 — Cortex-R5F: Deadlock Might Occur When One or More MPU Regions is Configured for Write Allocate Mode	Yes	Yes	Yes
	i2129 — Cortex-R5F: High Priority Interrupt is Missed by VIM	Yes	Yes	Yes
	i2132 — Cortex-R5F: Interrupt Preemption (Nesting) is Unavailable if Using VIM Vector Interface for Interrupt Handling	Yes	Yes	Yes
CPSW	i2027 — CPSW: CPSW does not support CPPI receive checksum (Host to Ethernet) offload feature			Yes
	i2084 — CPSW: CPSW Does Not Support Intersperced Express Traffic (IET – P802.3br/D2.0) In 10/100Mbps Mode			Yes
	i2139 — CPSW: ALE Incorrectly Routes Packets With CRC Errors			Yes
	i2148 — CPSW: CPSW Directed Frames are Not Observed When Classification Overrides the Destination Port Via the Egress Opcode Feature	Yes	Yes	Yes
	i2184 — CPSW: IET express traffic policing issue	Yes	Yes	Yes
	i2185 — CPSW: Policer color marking issue	Yes	Yes	Yes
CPTS	i2083 — CPTS: GENF (and ESTF) Reconfiguration Issue			Yes
	i2141 — CPTS: GENF and ESTF Nudge Value Not Cleared by Hardware	Yes	Yes	Yes
CSI	i2204 — CSI: Interface Setup/Hold Timing Does Not Meet MIPI DPHY Spec above 600MHz	Yes	Yes	Yes
DCC	i2018 — DCC: Incorrect Counter Values in DCC Operation			Yes
	i2073 — DCC: Suspend Mode Not Functional			Yes
DDR	i2231 — DDR: LPDDR4 and DDR3L are not supported	Yes	Yes	Yes
	i2262 — DDR: Mode Register write busy indicator not cleared after completing software-driven mode register access	Yes	Yes	Yes
	i2264 — DDR: Timing violation from read/write command to Software-Initiated MRW command (LPDDR4 only)	Yes	Yes	Yes
	i2265 — DDR: Issuing MRR/MRW commands in self-refresh state (LPDDR4 only)	Yes	Yes	Yes
	i2266 — DDR: Controller derating logic does not consider system temperature when derating is enabled (LPDDR4 only)	Yes	Yes	Yes

Advisories Matrix (continued)

MODULE	DESCRIPTION	SILICON REVISIONS AFFECTED		
		AM65x		
		2.1	2.0	1.0
	i2268 — DDR: Controller can violate timing from Self-Refresh Exit (SRX) to Power-Down Entry (PDE) for DDR3	Yes	Yes	Yes
DDRSS	i2009 — DDRSS: DDR Controller ECC Scrubbing Feature Can Cause DRAM Data Corruption	Yes	Yes	Yes
	i2022 — DDRSS: Independent Impedance Control for Address/Control and Data Bus Lanes is Not Available			Yes
DMSC	i2245 — DMSC: Firewall Region requires specific configuration	Yes	Yes	Yes
DRU	i2198 — DRU, UTC: Issue with setting ICNT3 to 0 when not being used	Yes	Yes	Yes
DSS	i2000 — DSS: DSS Does Not Support YUV Pixel Data Formats			Yes
	i2032 — DSS: DSS DPI Interface does not support BT.656 and BT.1120 output modes			Yes
	i2039 — DSS: Frame Buffer Flip/Mirror Feature Using RGB24/BGR24 Packed Format Can Result in Pixel Corruption	Yes	Yes	Yes
	i2097 — DSS: Disabling a Layer Connected to Overlay May Result in Synclost During the Next Frame	Yes	Yes	Yes
GIC	i2101 — GIC: ITS Misbehavior			Yes
HyperFlash	i2119 — HyperFlash: HyperFlash is Not Supported	Yes	Yes	Yes
IA	i2196 — IA: Potential deadlock scenarios in IA	Yes	Yes	Yes
IO, MMCSD	i2025 — IO, MMCSD: Incorrect IO Power Supply Connectivity Prevents Dynamic Voltage Change on VDDSHV6 and VDDSHV7			Yes
MCAN	i939 — MCAN: Message Transmitted with Wrong Arbitration and Control Fields (Early Start of Frame)	Yes	Yes	Yes
	i2278 — MCAN: Message Transmit order not guaranteed from dedicated Tx Buffers configured with same Message ID	Yes	Yes	Yes
	i2279 — MCAN: Specification Update for dedicated Tx Buffers and Tx Queues configured with same Message ID	Yes	Yes	Yes
MCU	i2173 — MCU: MCU domain may hang if main domain is issued a reset	Yes	Yes	Yes
MDIO	i2329 — MDIO: MDIO interface corruption (CPSW and PRU-ICSS)	Yes	Yes	Yes
MMCSD	i2024 — MMCSD: MMCSD Peripherals Do Not Support HS400	Yes	Yes	Yes
	i2026 — MMCSD: Negative Current from UHS-I PHY May Create an Over-Voltage Condition on VDDS6 and VDDS7 Which Exposes the Device to a Significant Reliability Risk			Yes
	i2312 — MMCSD: HS200 and SDR104 Command Timeout Window Too Small	Yes	Yes	Yes
MSMC	i2021 — MSMC: Non-Coherent Memory Access to Coherent Memory Can Cause Invalidation of Snoop Filter			Yes
	i2068 — MSMC: SW considerations for HW data coherency due to inconsistent views of memory			Yes
	i2116 — MSMC: Set-hazarding logic withholding RT access waiting on NRT access completion	Yes	Yes	Yes
	i2149 — MSMC: MSMC Scrubber Only Targets Bottom 16 of 32 Ways of SRAM/L3$	Yes	Yes	Yes
	i2187 — MSMC: Cache Resize to 0 Refreshes Tags instead of Updating them	Yes	Yes	Yes
On-chip Debug	i2013 — On-Chip Debug: The Assertion of Warm Reset Coinciding with a Debug Configuration Access Targeting the STM Subsystem May Result in a Hang of Said Debug Configuration Access	Yes	Yes	Yes
	i2015 — On-Chip Debug: CPTracer Bus Probes MAIN_CAL0_0 and MCU_SRAM_SLV_1 are not able to Distinguish between Secure and Non-secure Transactions	Yes	Yes	Yes
OSPI	i2115 — OSPI: OSPI Boot Doesn't Support Some xSPI Modes or xSPI Devices	Yes	Yes	Yes
	i2189 — OSPI: Controller PHY Tuning Algorithm	Yes	Yes	Yes
	i2249 — OSPI: Internal PHY Loopback and Internal Pad Loopback clocking modes with DDR timing inoperable	Yes	Yes	Yes
PCIe	i2037 — PCIe: PCI-Express May Corrupt Inbound Data			Yes
	i2046 — PCIE: Failure to link up in Root Complex mode when automatic lane reversal is performed by downstream port	Yes	Yes	Yes
	i2076 — PCIe: QoS support for outbound transactions across PCIe is not fully functional			Yes
	i2104 — PCIe: GEN3 (8GT/s) Operation Not Supported	Yes	Yes	Yes
	2014 — PCIe: PCIE1 instance does not support outbound address translation bypass	Yes	Yes	Yes

Advisories Matrix (continued)

MODULE	DESCRIPTION	SILICON REVISIONS AFFECTED		
		AM65x		
		2.1	2.0	1.0
	i2040 — PCIe: Root Port does not set the Link Bandwidth Management and Link Autonomous Bandwidth status bits	Yes	Yes	Yes
	i2041 —PCIe: Calculated Negative Round-trip Time Causes Wrong PTM Requester Local time			Yes
	i2043 —PCIE: Compliance test fails with certain lane reversal and lane polarity inversion conditions when using Enter Compliance bit	Yes	Yes	Yes
PRU_ICSSG	i2106 — PRU_ICSSG: 100Mbit/s MII is not supported when the PRU_ICSSG is operating at frequencies < 250MHz	Yes	Yes	Yes
	i2193 — PRU-ICSSG: Erroneous FDB aging behavior during switch operation with bucket size of 8		Yes	Yes
PSIL	i2137 — PSIL: Clock stop operation can result in undefined behavior	Yes	Yes	Yes
	i2138 — PSIL: Configuration accesses and source thread teardowns may cause data corruption	Yes	Yes	Yes
RA	i2095 — RA: Peek to Tail Returns Wrong Data	Yes	Yes	Yes
R5FSS	i2118 — R5FSS: Debug Access in Lock-Step Mode May Result in Failure	Yes	Yes	Yes
	i2161 — Debugger Cannot Access VIM Module While It Is Active	Yes	Yes	Yes
	i2162 — R5FSS: The Same Interrupt Cannot be Nested Back-2-Back Within Another Interrupt	Yes	Yes	Yes
	i2164 — R5FSS: Errors in ECC injection logic are not detected because the pending interrupts are tied low		Yes	Yes
	i2165 — R5FSS: Access to CPU1 TCMs hangs in Lockstep mode			Yes
RINGACC, UDMA	i2023 — RINGACC, UDMA: RINGACC and UDMA Ring State Interoperability Issue after Channel Teardown			Yes
RINGACC	i2054 — RINGACC: Reads from GCFG Region Can Cause Spurious RAM ECC Errors			Yes
	i2177 — RINGACC: The ring accelerator's debug transaction trace stream can be corrupted by certain ring access sequences	Yes	Yes	Yes
SA2_UL	i2098 — SA2_UL: Auth/Decrypt Operations with 2nd Input Thread Does Not Send the DMA Packet Out			Yes
Safety Modules	i2103 — Safety Modules: Incorrect Reporting of ECC_GRP, ECC_BIT and ECC_TYPE Information for Functional Safety Errors	Yes	Yes	Yes
UART	i2096 — UART: Spurious UART Interrupts When Using DMA	Yes	Yes	Yes
	i2310 — USART: Erroneous clear/trigger of timeout interrupt	Yes	Yes	Yes
	i2311 — USART: Spurious DMA Interrupts	Yes	Yes	Yes
UDMA-P	i2004 — UDMA-P: UDMA-P Host Packet Descriptor's "0x3FFFFF" Packet Length Mode not Functional			Yes
	i2163 — UDMA-P: UDMA transfers with ICNTs and/or src/dst addr NOT aligned to 64B fail when used in "event trigger" mode	Yes	Yes	Yes
	i2006 — UDMA-P: UDMA-P Real-Time Remote Peer Registers not Functional Across UDMA-P Domains			Yes
	i2055 — UDMAP: Packet Mode Descriptor Address Space Select Field Restrictions			Yes
	i2143 — UDMAP: TX Channel SA2UL teardown issue	Yes	Yes	Yes
	i2146 — UDMA: Force teardown bitfield readback is masked in realtime TX/RX registers	Yes	Yes	Yes
	i2234 — UDMA: TR15 hangs if ICNT0 is less than 64 bytes	Yes	Yes	Yes
	i2320 — UDMA, UDMAP, BCDMA, PKTDMA: Descriptors and TRs required to be returned unfragmented	Yes	Yes	Yes
USB2PHY	i2075 — USB2PHY: USB2PHY Charger Detect is Enabled by Default Without VBUS Presence			Yes
USB3SS	i2028 — USB3SS: SuperSpeed USB Non-Functional	Yes	Yes	Yes
VTM	i2053 — VTM: Software Reads from On-Die Temperature Sensors Can Be Corrupted			Yes
	i2145 — VTM: Enabled interrupt event status registers incorrectly return raw unmasked values	Yes	Yes	Yes

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, AM6546). 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 AM654x and AM652x devices, see the specific-device Datasheet (SPRSP08, SPRSP31).

2.2 Devices Supported

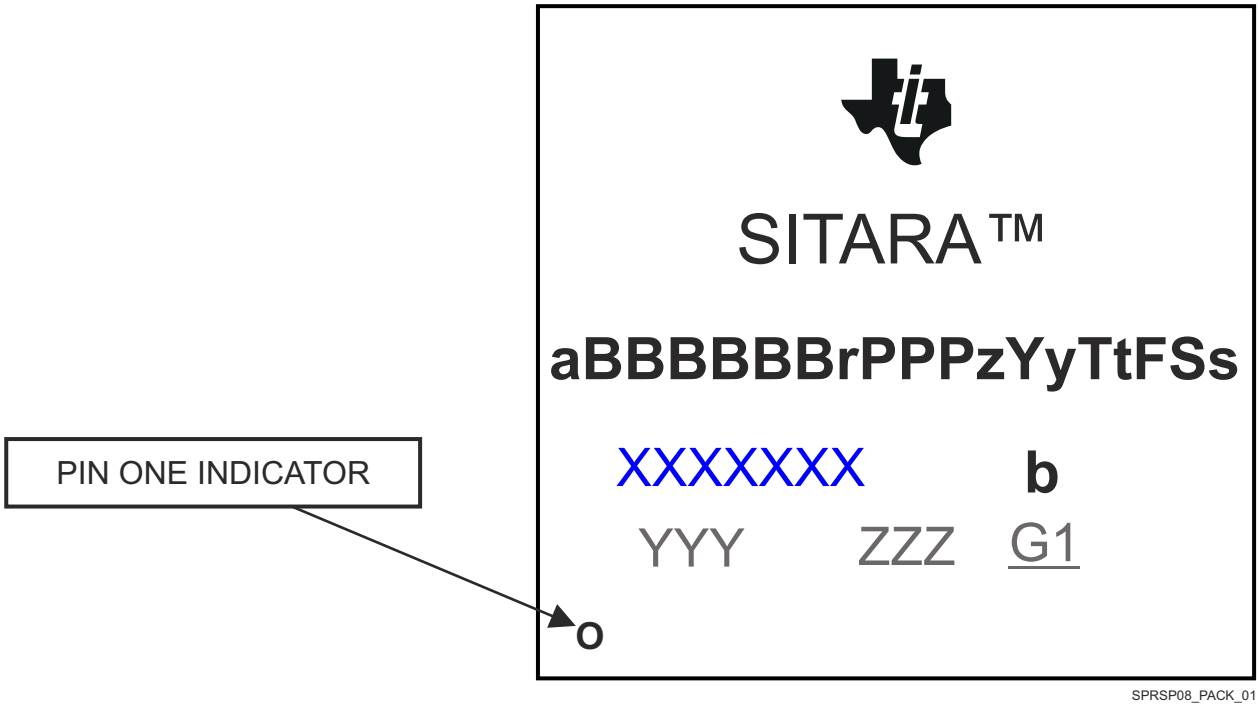
This document supports the following devices:

- AM6526
- AM6528
- AM6546
- AM6548

2.3 Package Symbolization and Revision Identification

Figure 2-1 shows an example of package symbolization.

Table 2-1 lists the device revision codes.



SPRSP08_PACK_01

Figure 2-1. Package Symbolization

Table 2-1. Revision Identification

DEVICE REVISION CODE	SILICON REVISION	COMMENTS
BLANK	1.0	Available as null.
A	2.0	
B	2.1	

3 Silicon Revision 2.1, 2.0, 1.0 Usage Notes and Advisories

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

3.1 Silicon Revision 2.1, 2.0, 1.0 Usage Notes

3.1.1 Fail-Safe IO's: Latch-up Risk on Fail-Safe IOs

AM65x silicon revision 1.0, 2.0, and 2.1 incorporate fail-safe I/O's on several pins (I2C0_SCL, I2C0_SDA, I2C1_SCL, I2C1_SDA, and NMIn). The fail safe I/O's tolerate voltage applied on the pins before their respective I/O supply voltage is ramped up. There is a potential latch up risk based on design reviews of the fail-safe I/O pins when driven high during functional mode. This latch-up risk is not yet confirmed through silicon characterization. To avoid the risk of a latch-up condition, the following steps should be implemented, depending on the mux mode used on the fail safe I/O. If the fail safe I/O is used in an I2C mux mode, then an external pull-up resistor (> 1 kOhm) is required on the signal. If the fail safe I/O is used in any other mux mode, then an external series resistor (> 1 kOhm) should be placed on this signal (close to the SoC).

3.1.2 ADC: High Input Leakage Current May Impact ADC Accuracy

On AM65x silicon revision 1.0, ADC input leakage current may be higher than expected at worst case Process/Voltage/Temperature (PVT) conditions, where process variation and operating temperature are the major contributors. Leakage current is larger for strong process devices operating at elevated temperatures.

There is also a dependency on the potential applied to an ADC input. Leakage current flows out of the ADC when applying a potential equal to VSS, flows into the ADC when applying a potential equal to VDDA_ADC_MCU, and the direction change occurs at approximately 42% of VDDA_ADC_MCU. Magnitude of leakage current has a non-linear function to the applied potential, where it increases exponentially as the applied potential approached VSS or VDDA_ADC_MCU.

Significant error can be introduced in ADC measurements when high impedance sources are connected to inputs with high leakage. This occurs because the input leakage current introduces a voltage drop across the source impedance. For example, the ADC would measure a potential of 1.45 V when measuring a 1.5 V source with 1 kΩ output impedance that is connected to an ADC input with 50 µA of leakage flowing into the ADC input. The error of this measurement would be 50 mV, which is 3.3% lower than the expected value. Reducing the source impedance from 1 kΩ to 100 Ω in this example would reduce the measurement error to 0.33%.

There are design techniques that can be used to minimize the impact of input leakage.

Reduce impedance of sources connected to ADC inputs. For example, it may be necessary to buffer outputs of a high impedance sources with voltage-follower operational amplifier circuits.

Design static DC sources to apply a nominal potential of approximately 42% of VDDA_ADC_MCU. For example, this approach can be used to minimize leakage when monitoring a DC power source via a resistor voltage divider.

3.1.3 INTRTR: Spurious Interrupts Generated when Programming Certain Interrupt Routers

On AM65x silicon revision 1.0, 2.0, and 2.1, programming the MUXCNTL_n registers to configure input-output mapping of interrupt signals may result in a short glitch on the intended output signal, causing a spurious interrupt. Additionally, reprogramming the register to the same value may also cause a glitch.

The *Interrupt Routers* section in the device TRM also describes this behavior, which is applicable to multiple Interrupt Routers (INTRTR) instantiated in the device, including:

- WKUP_GPIOMUX_INTRTR0
- GPIO_INTRTR0
- MAIN2MCU_LVL_INTRTR0
- MAIN2MCU_PLS_INTRTR0
- TIMESYNC_INTRTR0
- CMPEVT_INTRTR0

To prevent system from servicing these spurious interrupts unintentionally, following programming sequence are recommended when configuring INTR interrupt mapping.

In systems where interrupt mapping is static, typically with RTOS or bare-metal programming, the following interrupt configuration sequence shall be followed:

1. Disable interrupt at GIC and VIM for the interrupt sourced from the INTRTR output;
2. Re-configure MUXCNTL_n on the specific INTRTR;
3. Clear spurious interrupt if any by clearing raw status;
4. Enable the interrupt at GIC and/or VIM.

In systems where static interrupt configurations not possible, such as Linux systems with standard GIC drivers, interrupt drivers must detect false interrupt caused by the glitch, and clear the false interrupt. This method can be performed by the following programming sequence:

- If possible to customize interrupt handler, the handler shall first clear any interrupts before the handler being initialized, otherwise,
- The handler must check the interrupt source (such as GPIO status) to check valid event exists, then service the interrupt.

In systems with shared IRQs where multiple INTRTRs map to the same GIC IRQ, the following pseudo code may be used for the global interrupt handler:

```
isr(irq)
{     if (!read_status_reg())        return IRQ_NONE; }
```

In this case, each ISR can check and report to global IRQ handler that it wasn't the cause of IRQ, allowing the global IRQ handler to call the next handler in the list for that IRQ. In case of spurious IRQ, all the handlers (if there are no events) will return IRQ_NONE, which means Linux kernel will report a spurious IRQ on that line as the global handler will report EOI.

i2351	***OSPI: Controller does not support Continuous Read mode with NAND Flash***
Details:	The SoC and OSPI controller doesn't support Continuous Read mode with NAND Flash since the OSPI controller can deassert the CSn signal (by design intent) to the Flash memory between internal DMA bus requests to the OSPI controller. The issue occurs because "Continuous Read" mode offered by some OSPI/QSPI NAND Flash memories requires the Chip Select input to remain asserted for an entire burst transaction. The SoC internal DMA controllers and other initiators are limited to 1023 B or smaller transactions, and arbitration/queuing can happen both inside of the various DMA controllers or in the interconnect between any DMA controller and the OSPI peripheral. This results in delays in bus requests to the OSPI controller that result in the external CSn signal being deasserted. NOR Flash memories are not affected by CSn de-assertion and Continuous Read mode works as expected.
Workaround(s):	Software should use page/buffered read modes to access NAND flash.

3.2 Silicon Revision 2.1, 2.0, 1.0 Advisories

3.2.1 Silicon Revision 2.1, 2.0, 1.0 Advisory List

i939 ***MCAN: Message Transmitted with Wrong Arbitration and Control Fields (Early Start of Frame)***

Details

The erratum is limited to the case when MCAN is in state "Receiver" (MCAN_PSR[4:3] ACT = "10") with no pending transmission (no MCAN_TXBRP bit set) and a new transmission is requested before the 3rd bit of Intermission is reached and this 3rd bit of intermission is seen dominant. This issue occurs only with disturbed CAN bus.

Under the following conditions, a message with wrong ID, format, and DLC is transmitted:

- MCAN is in state "Receiver" (MCAN_PSR[4:3] ACT = "10"), no pending transmission
- A new transmission is requested before the 3rd bit of Intermission is reached
- The CAN bus is sampled dominant at the third bit of Intermission which is treated as SoF (see ISO11898-1:2015 Section 10.4.2.2).

Under the conditions listed above, it may happen, that:

- The shift register is not loaded with ID, format, and DLC of the requested message
- The MCAN will start arbitration with wrong ID, format, and DLC on the next bit
- In case the ID wins arbitration, a CAN message with valid CRC is transmitted
- In case this message is acknowledged, the ID stored in the Tx Event FIFO is the ID of the requested Tx message and not the ID of the message transmitted on the CAN bus, no error is detected by the transmitting MCAN
- If the message loses arbitration or is disturbed by an error, it is retransmitted with correct arbitration and control fields.

Workaround

Workaround 1:

Request a new transmission only if another transmission is already pending or when the MCAN is not in state "Receiver" (when MCAN_PSR[4:3] ACT !=, "10").

To avoid activating the transmission request in the critical time window between the sample points of the 2nd and 3rd bit of Intermission, following can be done:

- Evaluate the Rx Interrupt flags MCAN_IR[19] DRX, MCAN_IR[0] RF0N, MCAN_IR[4] RF1N which are set at the last bit of EoF when a received and accepted message gets valid. Prevent the transmission of any message within 3 bit times after detecting the Rx interrupt flags.

Workaround 2:

A checksum covering arbitration and control fields can be added to the data field of the message to be transmitted, to detect frames transmitted with wrong arbitration and control fields.

Workaround 3:

Set MCAN_CCCR[0] INIT bit, add transmission request for the message and then clear MCAN_CCCR[0] INIT bit. This needs to be done for each message to be transmitted.

Workaround 4:

Keep the number of pending transmissions always at ">0" by frequently requesting a message so that the condition "no pending transmission" is never met. The frequently requested message may be given a low priority, losing arbitration to all other messages.

In case, where all the nodes present on the CAN bus have this workaround implemented, at least two nodes shall have this frequently sent low priority message with different priorities/identifiers. Rest of the nodes can have one of these priorities/identifiers (same) as the priority of this frequently sent message.

Comparison within available Workarounds is shown in Table 3-1.

i939 (continued) ***MCAN: Message Transmitted with Wrong Arbitration and Control Fields (Early Start of Frame)***

Table 3-1. Comparison within available Workarounds

Workaround (WA) #	Advantages	Disadvantages
1	• Easy to implement. • Fix is only limited to a local node with the erratum.	• Only viable in interrupt context. • Effectiveness is vastly dependent on interrupt service latency. • Application becomes complicated.
2	• Simple and easy to implement.	• All nodes on the bus must implement this WA. Restricting use of device for end points and all the existing nodes needs to be updated to support this WA.
3	• Simple and easy to implement.	• Node will miss any messages received during this sequence. • System (including existing nodes) needs to be updated for detecting the dropped/missed message.
4	• DMA can be used to send the frequently sent low priority message to reduce CPU load. • Most effective and viable amongst available workarounds. • Fix is only limited to a local node with the erratum.	• 100% CAN Bus utilization at all times. • Increased CPU load in case CPU is used to send the message.

i2000 ***DSS: DSS Does Not Support YUV Pixel Data Formats***

Revision(s) Affected: AM65x SR 1.0

Details: Table 3-2 lists the pixel data formats that are not supported in the DSS.

Table 3-2. Unsupported Pixel Data Formats

Format #		Pixel Formats	Pipline support		Component Bit Depth
Packed	Planar		Video	Video_Lite	
0x3E	N/A(1)	YUV422-YUV2	X	X	8/10/12
0x3F	N/A	YUV422-UYVY	X	X	8/10/12
N/A	0x3D	YUV422-NV12	X	X	8/10/12
N/A	N/A	YUV422-NV21	X	X	8/10/12

(1) N/A - Not Applicable

Selecting the above formats (see Table 3-2) along with using the color conversion logic inside Video/Video_Lite pipeline will result in incorrect color reproduction at the output of DSS.

Workaround(s): Customer may use RGB mode for image capture or configure the input camera to use RGB mode instead of YUV formats referenced above. If RGB mode not available, it is recommended to use processor core or GPU to convert YUV to RGB.

i2004 ***UDMA-P: UDMA-P Host Packet Descriptor's "0x3FFFFF" Packet Length Mode not Functional***

Revision(s) Affected: AM65x SR 1.0

Details: In the UDMA-P Host Packet Descriptor (PD Word 0), there is a 22-bit packet length field. This field is typically set to a value between 0x0 – 0x3FFFFE, representing the total byte length of the packet. If the packet length is less than the sum of the buffer length, the packet will be truncated to the size specified in the packet length field. Specifying a packet length that is greater than the sum of the buffers will result in an error.

Alternatively, the packet length field can be set to a special value, 0x3FFFFF, which disables truncation and allows the UDMA-P to transmit as much data as is specified in the Buffer Length fields. This "0x3FFFF" mode is not functional.

Workaround(s): There is no workaround. A valid packet length field must be supplied. This only prevents a user from sending a packet larger than 0x3FFFFE bytes, which is more than large enough for any envisioned use case.

i2006 ***UDMA-P: UDMA-P Real-Time Remote Peer Registers not Functional Across UDMA-P Domains***

Revision(s) Affected: AM65x SR 1.0

Details: In the UDMA-P, both RX and TX channels contain a specific set of registers called "Real-time Remote Peer Registers" (UDMA_PEER[0-15]_j registers). These registers provide access to the remote peer device's real time registers within the PSI-L configuration (PSIL_CFG) register space, address 0x400 to 0x40F. The peer device is the device that the UDMA-P channel is communicating with, and is set via the Rx/TX Channel Destination Thread ID Mapping Register (UDMA_THREAD_j register). Once the UDMA_THREAD_j register is configured with the peer's PSI-L thread ID, any access to the UDMA_PEER[0-15]_j registers on the UDMA-P will results in an access to the PSI-L register on the peer corresponding to the offset of the register accessed. For example, peer register 0 maps to peer PSI-L address 0x400, and peer register 1 maps to peer PSI-L address 0x401, and so on. Having these registers allows the UDMA-P driver to access peer registers in the 0x400 to 0x40F range without accessing the configuration proxy IP, which in some environments can be reserved for secure access only.

There are two UDMA-P instances on the device. These instances exist in separate domains called MAIN and MCU. Along with the two UDMA-P instances, individual peripheral devices also reside in both domains. It was originally intended that a UDMA-P instance in one domain would seamlessly work with a peripheral in the other domain. For the most part, this is still the case, but when it comes to using the peer registers, a bug in the internal message routing prevents the UDMA-P peer registers in the MAIN domain from accessing the PSI-L registers of a peripheral in the MCU domain, and prevents the UDMA-P peer registers in the MCU domain from accessing the PSI-L registers of the peripheral in the MAIN domain. Read results across UDMA-P domains will always be invalid. Writes across UDMA-P domains will still take affect but will take longer than normal.

Workaround(s): Avoid using the UDMA-P peer registers in one domain to access PSI-L registers of a peripheral in the other domain. This can be accomplished in one of two ways:

1. Always allocate a TX/RX channel from the UDMA-P residing in the same domain as the target peripheral.
2. Use the PSI-L proxy module to write PSI-L registers 0x400 through 0x40F instead of the using the UDMA-P Real-time Remote Peer registers.

i2006 (continued) ***UDMA-P: UDMA-P Real-Time Remote Peer Registers not Functional Across UDMA-P Domains***

Solution 1 avoids the problem by making sure that the peer register access is never performed “across domains.”

Solution 2 avoids the problem by not making use of the affected registers. The drawback here is that in some environments the PSI-L proxy module is restricted to secure access only. This means that any register access will have to be done through secure code, increasing overhead.

Note, solution 1 is preferred due to secure access limitation for this resource configuration when implementing the workaround in software.

i2009 ***DDRSS: DDR Controller ECC Scrubbing Feature Can Cause DRAM Data Corruption***

Revision(s) Affected: AM65x SR 2.1, SR 2.0, SR 1.0

Details: The DDR controller implements a built-in ECC scrubbing feature that is used for correcting single-bit errors in the DRAM. When this feature is enabled (DDRCTL_ECCCFG0[4] DIS_SCRUB = 0), the controller schedules an ECC scrub operation when a single-bit error is detected. However, in some cases, the ECC scrub operation can lead to accidental auto-correction of two-bit errors in an another DRAM location, thus corrupting the data inside the DRAM.

Workaround(s): The ECC scrubbing feature inside the DDR controller must be kept disabled at all times by setting DIS_SCRUB = 1 in the DDR controller's ECC Configuraion 0 (DDRCTRL_ECCCFG0) register.

i2013 ***On-Chip Debug: The Assertion of Warm Reset Coinciding with a Debug Configuration Access Targeting the STM Subsystem May Result in a Hang of Said Debug Configuration Access***

Revision(s) Affected: AM65x SR 2.1, SR 2.0, SR 1.0

Details: Debug configuration transaction initiators and targets are typically susceptible only to power-on-resets. The STM Subsystem includes the Arm® STM500 and a CoreSight™ CTI that provides triggering support. The orthogonal debug interconnect for the STM Subsystem is affected by warm reset. This results in the possibility of a hang of a debug configuration access that is active when warm reset is active..

Workaround(s): There is no workaround for this issue. The user needs to be aware of the possibility that a debug configuration hang may result in the rare case that the STM Subsystem is being configured while warm reset is active.

i2015 ***On-Chip Debug: CPTracer Bus Probes MAIN_CAL0_0 and MCU_SRAM_SLV_1 are not able to Distinguish between Secure and Non-secure Transactions***

Revision(s) Affected: AM65x SR 2.1, SR 2.0, SR 1.0

Details: The CPTracer Bus Probe supports filtering on the properties of bus transactions including a property 'psecure' that identifies a transaction as being secure or non-secure. In the case of bus probe instances supporting MAIN_CAL0_0 and MCU_SRAM_SLV_1, the 'psecure' property will always indicate that a transaction is non-secure even if it were actually secure.

i2015 (continued)	***On-Chip Debug: CPTracer Bus Probes MAIN_CAL0_0 and MCU_SRAM_SLV_1 are not able to Distinguish between Secure and Non-secure Transactions***
Workaround(s):	There are no workarounds for this issue. The user needs to be aware that secure and non-secure transactions are not distinguishable for MAIN_CAL0_0 and MCU_SRAM_SLV_1 CPTracer Bus Probes.
i2018	***DCC: Incorrect Counter Values in DCC Operation***
Revision(s) Affected:	AM65x SR 1.0
Details:	Due to a potential race condition, if the DCCCNT0, DCCCNT1, DCCCNTSEED0, or DCCCNTSEED1 registers are modified while a DCC module is in operation or immediately before a DCC module is put in to operation, the counter may not expire correctly as per the programmed counter value (expiring either late or early). This could result in an incorrect error signal value, indicating either a false pass or fail.
Workaround(s):	All DCC configuration fields, especially counter seeds and values, should be programmed while the DCC module is disabled. A user may also read back the last written counter value in the DCCCNT0 and DCCCNT1 registers before enabling the DCC module to ensure that counter values have had ample time to settle before the DCC module is enabled.
i2019	***Boot, USB3SS: Boot ROM Does Not Support USB Host MSC (Mass Storage Class) Boot Mode***
Revision(s) Affected:	AM65x SR 1.0
Details:	The Boot ROM does not support USB Host MSC (Mass Storage Class) Boot mode.
Workaround(s):	None.
i2020	***Boot, USB3SS: Boot ROM Does Not Support USB Device Firmware Upgrade (DFU) Boot Mode***
Revision(s) Affected:	AM65x SR 1.0
Details:	The Boot ROM does not support USB Device Firmware Upgrade (DFU) Boot mode.
Workaround(s):	None.
i2021	***MSMC: Non-Coherent Memory Access to Coherent Memory Can Cause Invalidation of Snoop Filter***
Revision(s) Affected:	AM65x SR 1.0
Details:	Snoop filter can be invalidated when different system masters access the same memory location with different coherency attributes. The MSMC snoop filter retains knowledge of memory cached locally by master (for example, Arm Cortex®-A53 local cache). A coherent transaction performed by another system master through the MSMC will check the snoop filter and ensure the locally-cached data is accessed for transaction, and that the memory and cache is kept coherent. If, however, a non-coherent transaction is performed by another master through MSMC, it will invalidate the snoop filter entry for

i2021 (continued) ***MSMC: Non-Coherent Memory Access to Coherent Memory Can Cause Invalidation of Snoop Filter***

the locally-cached data and future transactions will not perform a snoop on the cached contents and the memory and cache are no longer coherent.

Workaround(s): The workaround to avoid this scenario is to control the accesses to cacheable memory. During software initialization (for example, exception level startup) on different A53 clusters, the cache view to memory needs to be controlled such that it is consistent between clusters. When one cluster is performing a non-coherent access to memory, software on the other cluster must ensure that it does not have the memory location in its local cache. This applies to any SoC device variants where there are two A53 clusters in the SoC.

For IO transactions with DMA, it is required that all DMA masters perform coherent transactions only to any memory that is cached locally by an A53 core, unless it is accessing a globally non-coherent memory space. This applies to all SoC device variants, irrespective of the number of A53 clusters in the SoC.

i2022 ***DDRSS: Independent Impedance Control for Address/Control and Data Bus Lanes is Not Available***

Revision(s) Affected: AM65x SR 1.0

Details: The IO impedance calibration control for the address/control segment and the data segment in the DDR PHY are incorrectly mapped to the same set of control registers. This results in requiring a common set of impedance settings for output driver impedance and on-die termination for both the address/control IO signals as well as the data IO signals. As a result, only register DDRPHY_ZQ0PR0 can be used to program output driver impedance and on-die termination for both address/control and data.

Workaround(s): Program DDRPHY_ZQ0PR0 with appropriate values that will satisfy output driver impedance and on-die termination for both address/control and data signals. Use the DDR configuration tool to help determine the most optimal values. This tool will also facilitate the configuration of all DDR controller and PHY registers for optimal performance. Also, ensure that all layout recommendations are met as described in the AM65x DDR Board Design and Layout Guidelines Application Report (SPRACI2).

i2023 ***RINGACC, UDMA: RINGACC and UDMA Ring State Interoperability Issue after Channel Teardown***

Revision(s) Affected: AM65x SR 1.0

Details: The Ring Accelerator (RINGACC) and the Unified DMA Controller (UDMA) each maintain their own state information about rings (queues). However, when a ring is reset in the RINGACC, the UDMA is unaware of this operation and its state information for that ring is not reset. As a result, the UDMA may believe a ring still has valid pending entries, while the RINGACC does not, and will potentially read invalid information from a recently reset and enabled ring.

Resetting a ring is a required operation (following DMA channel teardown operations and before re-enabling the DMA channels) when re-configuring an existing ring for another purpose (that is changing the ring mode, DMA channels, channel type, etc.).

Workaround(s): The software workaround is to use the ring-mode doorbell functionality to cause the UDMA occupancy counter for a given ring to increment and wrap back to 0. Since a ring's UDMA occupancy counter is 21-bits wide, and the highest doorbell ring count that can be

i2023 (continued) ***RINGACC, UDMA: RINGACC and UDMA Ring State Interoperability Issue after Channel Teardown***

written per register write is 127, the workaround requires a maximum of ((2**22) / 127) = 33,027 writes to the doorbell register. Note that a negative value cannot be used as the doorbell ring count value as the UDMA ignores negative values (a negative doorbell ring count value is seen by the UDMA as a ring pop count of abs(value) elements which it ignores in its ring state accounting).

To implement the software workaround, the following sequence should be used:

1. Read the ring occupancy (RINGACC_OCC_j [CNT]). If the ring occupancy is not 0, then steps 2-6 need to be executed to implement the workaround.
2. Reset the ring by writing any value in the RINGACC CFG registers (that is RINGACC_SIZE_j = RINGACC_SIZE_j).
3. Read the ring mode (RINGACC_SIZE_j [QMODE]) to determine the “adjusted ring occupancy count” used in step 5.
    a. If the ring is configured for exposed ring mode or messaging mode, the “adjusted ring occupancy count” is equal to the ring occupancy.
    b. If the ring is configured in credentials mode or queue manager (QM) mode, the “adjusted ring occupancy count” is equal to the ring occupancy divided by 2. This is required because when in credentials mode or QM mode, each ring write increases the ring occupancy by 2 elements (one entry for the credentials, one entry for the data). However, the UDMA-P’s local occupancy counter only records the number of writes, and the ring occupancy, therefore, needs to be divided by 2 to convert back to the number of doorbell rings needed.
4. Setup the ring in exposed ring/doorbell mode, if not already in this mode (RINGACC_SIZE_j [QMODE] = 0).
5. Ring the doorbell (2**22 – (adjusted ring occupancy count)) times. This will wrap the internal UDMA-P ring state occupancy counter (which is 21-bits wide) to 0. (If possible, ring the doorbell with the maximum count each iteration to minimize the total number of writes.
6. Restore the original ring mode (if not exposed ring mode).

This will ultimately reset the UDMA state information for a ring so that the RINGACC and UDMA are both in the same reset state for that ring. This workaround must be executed only when the DMA channels associated with the ring are disabled.

i2024 ***MMCSD: MMCSD Peripherals Do Not Support HS400***

Revision(s) Affected: AM65x SR 2.0, SR 1.0

Details: The MMCSD peripherals do not support the Multimedia Card HS400 mode.

Workaround(s): None

i2025 ***IO, MMCSD: Incorrect IO Power Supply Connectivity Prevents Dynamic Voltage Change on VDDSHV6 and VDDSHV7***

Revision(s) Affected: AM65x SR 1.0

Details: The LVCMOS IOs associated with device pins MMC0_SDCD (A23) and MMC0_SDWP (B23) receive power from VDDSHV6 rather than VDDS_OSC1.

The LVCMOS IOs associated with device pins MMC1_SDCD (B24) and MMC1_SDWP (C24) receive power from VDDSHV7 rather than VDDS_OSC1.

i2025 (continued) ***IO, MMCSD: Incorrect IO Power Supply Connectivity Prevents Dynamic Voltage Change on VDDSHV6 and VDDSHV7***

VDDSHV6 and VDDSHV7 also provides IO power to MMCSD0 UHS-I PHY and MMCSD1 UHS-I PHY, respectively, which may require dynamically voltage change for some use cases.

LVCMOS IOs detect the voltage applied to their respective VDDSHV power rail while MCU_PORz or PORz reset inputs are asserted to determine if they should operate in 1.8 V mode or 3.3 V mode. The IO voltage operating mode is latched on the rising edge of PORz.

After PORz is de-asserted, VDDSHV6 and VDDSHV7 must remain within the recommended operating range of the applied voltage to avoid potential long term reliability issues.

Since VDDSHV6 and VDDSHV7 shall not change after PORz is released, it is not possible to implement dynamic IO voltage change on the IOs associated with MMCSD0 and MMCSD1.

Advisory i2026 describes a reliability issue when operating VDDSHV6 and VDDSHV7 at 3.3 V, which limits the operation of these IO supplies to 1.8 V.

These four LVCMOS IOs associated with VDDSHV6 and VDDSHV7 should have been powered from VDDS_OSC1, which is a fixed 1.8 V supply.

Workaround(s):

For most applications where MMC card detect and/or write protect function is implemented on these pins, the external pull-up resistors associated with these signals need to be powered from the same power source as the respective VDDSHV6 or VDDSHV7 power pins.

For these use cases, PCBs designers should consider including resistor installation options that allow external pull-ups to be powered from the respective VDDSHV6 / VDDSHV7 supply or the VDDS_OSC1 supply. The resistor installation option allows a common PCB design to support current silicon revision and future silicon revisions where these IOs will be powered from VDDS_OSC1.

For use cases other than MMC card detect and write protect, the effect of this IO being sourced from a different supply rail needs to be evaluated/considered for any connectivity to these pins.

i2026 ***MMCSD: Negative Current from UHS-I PHY May Create an Over-Voltage Condition on VDDS6 and VDDS7 Which Exposes the Device to a Significant Reliability Risk***

Revision(s) Affected:

AM65x SR 1.0

Details:

The MMCSD0 UHS-I PHY and MMCSD1 UHS-I PHY receives 1.8 V bias power from device pins VDDS6 and VDDV7 respectively. Unexpected paths through the UHS-I PHY allows current to flow from VDDSHV6 to VDDS6 and VDDSHV7 to VDDS7 when the VDDSHV IO supply is operating at 3.3 V.

The UHS-I PHY typically consumes power from its VDDS bias supply. However, the unexpected current flowing from the 3.3 V VDDSHV IO supply may exceed the bias current consumed by the UHS-I PHY. When this occurs, current may flow out of the 1.8 V VDDS bias supply. This negative current may cause the voltage applied to VDDS6 and VDDS7 to increase above the recommended operating range in some operating conditions.

This issue has been observed on a system where SDIO LDO was sourcing the UHS-I PHY bias supply while its VDDSHV IO supply was 3.3 V. This occurs because SDIO LDO was not designed to sink current. Therefore, it is not able to shunt any negative current to

i2026 (continued) ***MMCSD: Negative Current from UHS-I PHY May Create an Over-Voltage Condition on VDDS6 and VDDS7 Which Exposes the Device to a Significant Reliability Risk***

VSS. Negative current causes the VDDS bias supply to increase above the recommended bias supply voltage where the UHS-I PHY enters a non-functional state that can only be cleared when the respective MMCSD subsystem is reset.

The negative current is a function of device operating temperature, device process variation, and UHS-I PHY output toggle rate.

This issue can also occur when the IOs associated with MMCSD0 UHS-I PHY and MMCSD1 UHS-I PHY are operating at 3.3 V while configured to one of the other MUXMODES defined in the Datasheet Pin Multiplexing Table (when IOMUX_ENABLE bit in the respective MMCSD0_SS_PHY_CTRL_1_REG or MMCSD1_SS_PHY_CTRL_1_REG register is set to 1 to enable alternate MUXMODES).

Further analysis has indicated a significant reliability risk when operating VDDSHV6 and VDDSHV7 at 3.3 V.

This issue does not occur when the VDDSHV IO supply is operating at 1.8 V.

Workaround(s): There is no workaround which prevents the negative current from producing an overvoltage condition. Therefore, there is no support for operating VDDSHV6 and VDDSHV7 at 3.3 V.

i2027 ***CPSW: CPSW Does Not Support CPPI Receive Checksum (Host to Ethernet) Offload Feature***

Revision(s) Affected: AM65x SR 1.0

Details: CPSW does not support the CPPI receive checksum (Host to Ethernet) offload feature. As such, the CPSW0_P0_CONTROL_REG [0] RX_CHECKSUM_EN bit must remain at the default value of zero, disabling the receive checksum feature.

Workaround(s): Use software to implement receive checksum operations.

i2028 ***USB3SS: USB Host and Device Non-Functional***

Revision(s) Affected: AM65x SR 2.1, SR 2.0, SR 1.0

Details: The USB3.1 GEN1 (5Gbps) portion of USB3 Subsystem 0 (USB3SS0) is non-functional in both SuperSpeed Host and SuperSpeed Device mode.

The USB2.0 portion of USB3SS0 is unaffected.

Workaround(s): Implement USB2.0 on USB3SS0 for non-functional SuperSpeed modes. USB3SS0 supports USB2.0 High-Speed (480Mbps), Full-Speed (12Mbps), and Low-Speed (1.5Mbps) modes of operation when operating in USB Host mode, and USB2.0 High-Speed and Full-Speed modes of operation when operating as a USB Device.

i2030 ***Boot, UART: UART Boot Mode Never Times Out***

Revision(s) Affected: AM65x SR 1.0

Details: When UART is configured as the primary boot mode and its 180-second timeout expires, the ROM code is expected to switch to the selected backup boot mode. However, this transition to the backup boot mode never occurs.

i2030 (continued) ***Boot, UART: UART Boot Mode Never Times Out***

Instead, the SoC device goes through a warm reset because the watchdog timer expires before the UART boot mode times out. After the warm reset, the SoC device will continue to attempt booting from UART, instead of switching to the backup boot mode.

Workaround(s): Do not select a backup boot mode when UART is the primary boot mode.

i2032 ***DSS: DSS DPI Interface Does Not Support BT.656 and BT.1120 Output Modes***

Revision(s) Affected: AM65x SR 1.0

Details: The BT.656 and BT.1120 output modes are not supported on the DSS DPI interface. Selecting the above output modes will result in incorrect color reproduction on DSS DPI interface.

Workaround(s): No workaround is available if BT.656 or BT.1120 output modes are required. Alternatively, MIPI DPI 2.0 RGB output mode could be used instead of BT.656 and BT.1120.

i2037 ***PCIe: PCI-Express May Corrupt Inbound Data***

Revision(s) Affected: AM65x SR 1.0

Details: When an inbound PCIe TLP spans more than two internal AXI 128-byte bursts, the bus may corrupt the packet payload. This issue affects inbound reads only. Outbound transactions are not affected. No PCIe error is flagged as the protocol itself is correct and only the contained data is corrupted. This corrupt data may cause associated applications or the processor to hang.

Workaround(s): Limit the PCIe MAX_READ_REQUEST_SIZE (MRRS) and the PCIe MAX_PAYLOAD_SIZE (MPS) to 128 bytes by setting:

- PCIE_EP_DEVICE_CONTROL_DEVICE_STATUS / PCIE_RC_DEVICE_CONTROL_DEVICE_STATUS[14-12] PCIE_CAP_MAX_READ_REQ_SIZE register field to 0h, and
- PCIE_EP_DEVICE_CONTROL_DEVICE_STATUS / PCIE_RC_DEVICE_CONTROL_DEVICE_STATUS[7-5] PCIE_CAP_MAX_PAYLOAD_SIZE_CS register field to 0h.

This ensures that no more than two AXI burst transactions will be needed to complete any single PCIe TLP.

i2038 ***Boot: FAT16 Fails When Root Block Resides in More Than One Cluster***

Revision(s) Affected: AM65x SR 1.0

Details: Boot ROM will not find the boot file on a FAT16 file system if the file system uses multiple clusters for the boot block. Boot fails if the boot file does not reside on the first cluster. This has been observed when using Ubuntu to create a small FAT16 partition. In this case the cluster size is 4KB, so only 128 entries reside in the first root cluster (each directory entry is 32 bytes). If the boot file resides in file index 128 or later (max size is typically set to 512) the ROM will not find the boot file.

Workaround(s): Use FAT32 partition, instead of FAT16 partition.

i2039 ***DSS: Frame Buffer Flip/Mirror Feature Using RGB24/BGR24 Packed Format Can Result in Pixel Corruption***

Revision(s) Affected: AM65x SR 2.1, SR 2.0, SR 1.0

Details: The frame buffer flip/mirror feature in the video pipeline can result in pixel corruption on the display output when the RGB24 or BGR24 packed format is used and the following conditions are met simultaneously:

- Frame buffer flip/mirror feature enabled (DSS0_VID_ATTRIBUTES[12] FLIP = 0x1)
- RGB24 or BGR24 packed format selected (DSS0_VID_ATTRIBUTES[6-1] FORMAT = 0x0B or 0x0C)
- Scaler inside VID pipe is enabled (DSS0_VID_ATTRIBUTES[8-7] RESIZEENABLE = 0x1, 0x2, or 0x3)
- Scale ratio < 0.5, or down-scaling by more than half, this scale ratio is configured by DSS0_VID_FIRH, DSS0_VID_FIRH2, DSS0_VID_FIRV, and DSS0_VID_FIRV2 registers.

These conditions can cause one pixel to be corrupted at the start of a line, for some lines, in a frame.

Workaround(s): If the RGB24 or BGR24 packed format is selected, then use the GPU to implement the flip/mirror operation.

i2046 ***PCIE: Failure to link up in Root Complex mode when automatic lane reversal is performed by downstream port***

Details: Link up failure can occur if downstream port (Root Complex) is required to perform automatic lane reversal.

Usually, upstream port (Endpoint or Switch) detects lane reversal first and automatically performs the reversal. However, upstream port may not support lane reversal since it is optional. In this case, PCIE subsystem performs the following sequence:

1. Downstream port transitions from Configuration.Lanenum.Wait to Configuration.Lanenum.Accept state after 1ms and other conditions for this transition are met.
2. Downstream port performs lane reversal in Configuration.Lanenum.Accept state and waits for 1ms before transitioning to Configuration.Complete state.

This 1ms wait violates PCIe specification requirement, which requires downstream port to transition to Configuration.Complete immediately. In the meantime, upstream port's 2ms timeout in Configuration.Lanenum.Wait state may have elapsed and it may have transitioned to Detect state. If this occurs, link up does not succeed and downstream port also returns back to Detect state.

There is a possibility that upstream port adds timeout margin (instead of waiting for 2ms, it may wait for 2048us). This issue does not cause link up failure if such a margin is present in upstream port.

Workaround(s): Lane reversal can be manually performed by setting TX_LANE_FLIP_EN=1 and RX_LANE_FLIP_EN=1 in PCIE_RC_CMD_STATUS register. This has to be done during initialization, before link training is enabled by setting LTSSM_EN bit in PCIE_RC_CMD_STATUS register.

This requires system level mechanism for software to be notified if lane reversal is required. There are four possibilities to consider:

1. Lanes are not reversed in both RC and EP/switch board designs.
2. Lanes are reversed in RC board design and not reversed in EP/switch board design.
3. Lanes are not reversed in RC board design and reversed in EP/switch board design.

i2046 (continued) ***PCIE: Failure to link up in Root Complex mode when automatic lane reversal is performed by downstream port***

4. Lanes are reversed in both RC and EP/switch board designs.

If it is known whether lanes are reversed at both RC and EP/switch sides, then software can perform lane reversal manually if it is case 2 or case 3.

If it is only known whether RC board has lanes reversed, then software may perform lane reversal considering only the RC side. However, cases 3 and 4 will not be covered by this approach. Since EP/switch board is reversing lanes, it is likely that EP/switch automatically corrects for lane reversal at its end.

There is no workaround for the case where all of the below are true:

- EP/Switch may reverse its lanes.
- There is no method for informing RC software whether EP/Switch has reversed lanes.
- EP/Switch does not support automatic lane reversal in its LTSSM and it does not manually correct lane reversal done by its board.

i2053 ***VTM: Software Reads from On-Die Temperature Sensors Can Be Corrupted***

Revision(s) Affected: AM65x SR 1.0

Details: The VTM_TMPSENSj_STAT[9-0] DTEMP, where j = 0-7, registers can be read in software to determine the last sampled temperature of each of the '*j*' number of on die temp sensors on the SoC. If a read happens on the same exact cycle that a temperature sample is updated then there is a chance that the read data can be corrupted due to incorrect resynchronization between clock domains.

Workaround(s): Software should perform three reads from the VTM_TMPSENSj_STAT[9-0] DTEMP, where j = 0-7, registers. Software should then compute the temperature to be used based on the average of the two samples that are closest to each other.

The software pseudo code is as follows:

```
#define abs(x) (((x)<0)?-(x):(x))
unsigned int get_best_value(unsigned int s0, unsigned int s1, unsigned int s2)
{
        int d01 = abs(s0 - s1);
        int d02 = abs(s0 - s2);
        int d12 = abs(s1 - s2);

        // if delta 01 is least, take 0 and 1
        if ((d01 <= d02) && (d01 <=d12)) {
                return (s0+s1)/2;
        }
        // if delta 02 is least, take 0 and 2
        if ((d02 <= d01) && (d02 <=d12)) {
                return (s0+s2)/2;
        }
        /* in all other cases, take 1 and 2 */
        return (s1+s2)/2;
}

unsigned int get_temp()
{
        unsigned int s0,s1,s2;
        s0 = Read VTM_TMPSENSj_STAT[9-0] DTEMP;
        s1 = Read VTM_TMPSENSj_STAT[9-0] DTEMP;
        s2 = Read VTM_TMPSENSj_STAT[9-0] DTEMP;
        return get_best_value(s0,s1,s2);
}
```

i2054 ***RINGACC: Reads from GCFG Region Can Cause Spurious RAM ECC Errors***

Revision(s) Affected: AM65x SR 2.1, SR 2.0, SR 1.0

Details: A read to the Ring Accelerator (RA) Global Config Region (GCFG) can cause a read of a RAM with an illegal address. This causes the RAM to read random data which will fail the RAM ECC check. This will cause a log and interrupt to be created. The data itself is not used, so there is no functional failure, but the interrupt will make it appear there was a RAM failure.

Note

This affects the MCU NAVSS RA only, as the MAIN NAVSS RA has an aligned size so there are no illegal RAM addresses.

Workaround(s): The software that handles the RAM ECC interrupts for MCU NAVSS RA can check the address in the log registers and ignore the error if the address is beyond the limit of the RAM (which is the number of rings supported by the RA). The software can just clear the error.

i2055 ***UDMAP: Packet Mode Descriptor Address Space Select Field Restrictions***

Details: The UDMAP is used to perform several different types of data transfer including block copy and packet mode.

Packet mode transfers are designed to be used when the application requires support for true, unlimited fragment count scatter/gather type operations.

The Address Space Select field of the packet descriptor is used by the infrastructure as an identifier for which address space this particular memory region is located within. Address space 0 is the default unified address space for a given device. Address spaces 1-15 are used for alternate address maps which may be external to the device (PCIe/ Hyperlink) or in other ‘tiles’ on large devices.

SW is recommended to avoid non-zero Address Space Select values in the descriptors used in packet mode transfers only. Block copy and other transfer types supported by UDMAP are unaffected.

Use of non-zero Address Space Select values in packet mode transfers can result in unintended memory accesses.

Workaround(s): SW is recommended to avoid non-zero Address Space Select values in the descriptors used in packet mode transfers.

Use of non-zero Address Space Select values in packet mode transfers can result in unintended memory accesses.

i2068 ***MSMC: SW considerations for HW data coherency due to inconsistent views of memory***

Details: Hardware-Based Cache coherency and IO coherency is supported for both On-Chip Shared SRAM and DDR in the Keystone 3 Platform. To maintain proper data coherency, all data requestors are required to maintain a common consistent view of memory for all transactions to the same memory location (ie. memory type and shareability). A mismatch of these memory attributes can result in a loss of data coherency.

Workaround(s): In the absence of a common, consistent view of memory for all accesses to memory, software is required to use cache maintenance operations, barrier operations, and IPC

i2068 (continued) ***MSMC: SW considerations for HW data coherency due to inconsistent views of memory***

messages to maintain data coherency. Caching requestors are required to use clean and invalidate cache maintenance operations before and after each memory access in order to move data from local caches back out to main memory. Barrier operations are needed to ensure all memory write accesses have reached main memory and therefore observable to all requestors. Messages to other requestors are required to signal when these operations have completed. The memory granularity for Shared SRAM and DDR is 64B. Overlapping accesses from different processing entities to a memory location should be avoided to prevent potential race conditions. Software is required to manage data ownership and manually push data out back to main memory prior to transferring data ownership and/or enabling data observability by all memory requestors in the system.

i2069 ***CC_ARMSS: Powering Down CC_ARMSS1 Causes System Data Corruption***

Details: When attempting to power down CC_ARMSS1, programming sequences defined in Power Modes Section of ID062414 are followed as:

- Cluster shutdown mode without system driven L2 flush, or,
- Cluster shutdown mode with system driven L2 flush.

During the execution of these sequences, data corruption can be observed in the rest of the device subsystems, causing device, operating system(s), and application software to hang or malfunction.

Workaround(s): Cluster power down shall not be used. Instead, system may program individual cores to be shut down, by following the programing sequence defined in Power Modes Section of ID062414, but leave the CC_ARMSS1 cluster power domain ON.

i2073 ***DCC: Suspend Mode Not Functional***

Details: When DCC is programmed to work in suspend mode (also referred to as continuous mode), a DCC error event may be correctly generated on any of the following conditions:

- Clock1 expires before the COUNT0 reaches 0
- Clock1 expires after both COUNT0 and VALID0 reach 0
- Clock1 not present
- Clock0 not present.

Due to an IP level issue, a false DCC error event may be generated without any of the above conditions. This false event may trigger an unwanted interrupt in the system. Further, the false event condition will prevent DCC counters to reload after the first comparison, instead of continuously reload as expected in this mode. These two issues effectively render the suspend mode clock comparison non-usable.

Workaround(s): Suspend mode operation may not be used for any DCCs in the devices. Only single-shot mode is supported.

i2075 ***USB2PHY: USB2PHY Charger Detect is Enabled by Default Without VBUS Presence***

Details: The USB2PHY Charger Detect function is enabled by default after power on without VBUS presence. This causes D+ to be pulled up, violating the Universal Serial Bus Revision 2.0 Specification.

This violation could cause some USB hubs to not respond to the SETUP packet from the USB host and fail in enumeration if the hub is attached to the SoC's USB host port prior to the SoC being powered on.

i2075 (continued) ***USB2PHY: USB2PHY Charger Detect is Enabled by Default Without VBUS Presence***

Workaround(s): Disable the USB2PHY Charger Detect function by setting the PHY register bit USB2PHY_CHRG_DET[28] MEM_DIS_CHG_DET = 1 prior starting the USB host controller.

i2076 ***PCIe: QoS support for outbound transactions across PCIe is not fully functional***

Details: Quality of Service (QoS) support is available within the PCIeSS for the outbound transactions but is not properly utilized, with transactions following a strict ordering of all transactions.

Within the PCIeSS, traffic classes can be separated to four streams based on OrderID for the outbound transaction. The command order across the PCIe interface is based on priority between classes. Read response between classes should be independent, however with this errata the responses to high-priority transactions are serialized behind those belonging to low-priority transactions issued earlier.

The remote PCIe device operation is independent of this errata and may follow it's QoS servicing rules. Inbound PCIe transactions are not affected by this errata.

Workaround(s): None. Responses for outbound transactions are serviced in order.

i2083 ***CPTS: GENF (and ESTF) Reconfiguration Issue***

Details: Re-configuring a GENF/ESTF function after having been previously configured has an issue. Issue details:

If GENF re-configuration occurs when the GENF output is logic one then the re-configuration comparison time will be a half-count instead of the full count, and the GENF output will be off by 1/2 cycle. The re-configured cycle will be correct if the GENF output is logic zero when the re-configuration occurs.

Workaround(s): GENF reconfiguration can only happen after a SOC hardware reset.

i2084 ***CPSW: CPSW Does Not Support Intersperced Express Traffic (IET – P802.3br/D2.0) In 10/100Mbps Mode***

Details: The CPSW peripheral does not support Intersperced Express Traffic (IET – P802.3br/D2.0) in 10/100Mbps mode.

IET is only supported in 1000Mbps mode.

Workaround(s): None.

i2095 ***RA: Peek to Tail Returns Wrong Data***

Details: The Ring Accelerator (RA) peek from tail function does not work correctly. The RA will read the wrong location instead of the tail element, so the data will not match the real tail element.

Workaround(s): Users should not use this function as it is not reliable, as there is no workaround.

i2096 ***UART: Spurious UART Interrupts When Using DMA***

Details: Spurious UART interrupts may occur when DMA mode (UART_FCR[3] DMA_MODE) is enabled and DMA is used to read data from RX FIFO. The Interrupt Controller flags that a

i2096 (continued) ***UART: Spurious UART Interrupts When Using DMA***

UART interrupt has occurred; however, the associated UART_IIR_UART[0] IT_PENDING bit remains set to 1, indicating that no interrupt is pending.

Workaround(s): Acknowledge the spurious interrupts for every occurrence. The issue can be avoided by disabling Receive Data Interrupt (RDI) using the UART_IER_UART[0] RHR_IT bit; however, be aware that this also disables RX timeout interrupts, which may not be practical for all use cases.

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.

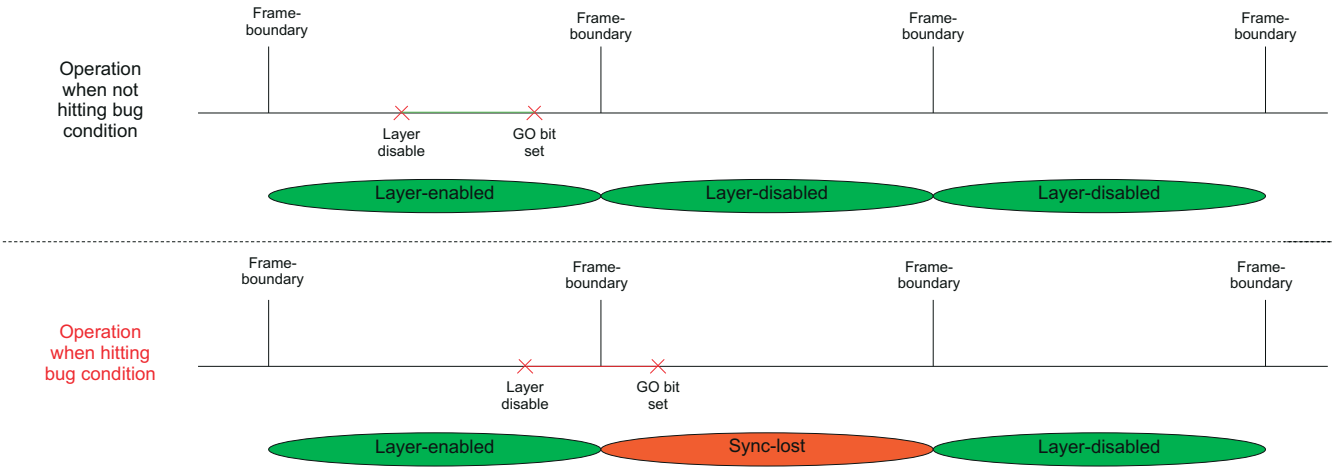


Figure 3-1. Bug Condition

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].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

i2098 ***SA2_UL: Auth/Decrypt Operations with 2nd Input Thread Does Not Send the DMA Packet Out***

Details: Thread muxing mode in ETYPE=5 (SA2_UL) can have unpredictable results ranging from lost data to crediting overflow on the UDMAP. This prevents use of destination thread 1, and source threads 2 and 3 of ETYPE=5 (SA2_UL).

Workaround(s): None. Destination thread 1, and source threads 2 and 3 cannot be used on SA2_UL.

i2099 ***Cortex-R5F: Deadlock Might Occur When One or More MPU Regions is Configured for Write Allocate Mode***

Details: There are two conditions where R5F can deadlock:

- When software is performing series of store operations to cacheable write back/write allocate memory region and later on software execute barrier operation (DSB or DMB). R5F may hang at the barrier instruction.
- When software is performing a mix of load and store operations within a tight loop and store operations are all writing to cacheable write back/write allocates memory regions, R5F may hang at one of the load instruction.

Workaround(s): Disabling linefill optimization inside Cortex-R5F will eliminate deadlock condition.

To disable the linefill optimization, the software needs to set bit 13 (DLFO) of Auxiliary Control Register (See Cortex-R5F Technical Reference Manual for how to update Auxiliary Control Register).

i2101 ***GIC: ITS Misbehavior***

Details: GIC AXI master traffic goes through protocol conversion bridge to access memory. Because of the misconfiguration of this protocol conversion bridge AXI read request generated on one particular ARID will not be returned by the bridge.

This will cause all ITS requests on that particular Device ID, for which read access was requested to fail.

i2101 (continued) ***GIC: ITS Misbehavior***

Impact of this is if more than 4 or more Device IDs are used in ITS, GIC can use ARID for AXI read which is not supported by protocol conversion bridge and that will result in missed interrupts.

Workaround(s): In the boot sequence, software needs to setup 5 dummy device IDs starting from 0, which are then reserved and can't be used for application, and then send ITS request for the first 2 device IDs. This sequence should use up unsupported ARID which will not be used by GIC during application and no ITS misbehavior would be seen.

Other combination of workaround can be setup 6 dummy device IDs starting from 0 and send ITS request to first device ID, setup 7 dummy device IDs starting from 0 and send ITS request to first 4 device IDs.

Note: This workaround is not implemented in Processor SDK Linux since it needs modifications to Arm® provided / maintained generic driver for GIC in Linux.

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.

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.

i2104 ***PCIe: GEN3 (8GT/s) Operation Not Supported***

Details: PCI-Express (PCIe) GEN3 (8GT/s) operation is not supported. This restriction applies to both Root Complex and Endpoint modes of operation. PCIe GEN1 (2.5GT/s) and PCIe GEN2 (5GT/s) operation is unaffected.

Workaround(s): No workaround is available. Implement PCIe as GEN1 or GEN2 only.

i2106 ***PRU_ICSSG: 100Mbit/s MII is not supported when the PRU_ICSSG is operating at frequencies < 250MHz***

Details: 100Mbit/s MII is not supported when ICSSG_TXCFG0/1[12] TX_IPG_WIRE_CLK_EN = 0 and the PRU_ICSSG CORE functional clock is operating at frequencies < 250MHz.

Workaround(s): For [M4_SR1.0], there is no workaround.

For [M4_SR2.0 and all other KS3 devices], the workaround to set ICSSG_TXCFG0/1[12] TX_IPG_WIRE_CLK_EN = 1 and program the minimum inter packet gap (ICSSG_TX_IPG0/1[15:0] TX_IPG0/1) as described in the device TRM. With this configuration and ICSSG_TXCFG0/1 [30:28] TX_CLK_DELAY0/1 set to the value defined

i2106 (continued) ***PRU_ICSSG: 100Mbit/s MII is not supported when the PRU_ICSSG is operating at frequencies < 250MHz***

in the device datasheet, 100Mbit/s MII support is not limited by the PRU_ICSSG CORE functional clock frequency.

i2115 ***OSPI: OSPI Boot Doesn't Support Some xSPI Modes or xSPI Devices***

Details:

For background, the various OSPI and xSPI protocols are described according to bit-width (1 or 8) and data rate (S or D for *S*ingle Data rate or *D*ouble Data rate) for the Command/Address/Data segments of the protocol.

The SoC's ROM OSPI boot mode supports 1S-1S-1S mode and 1S-1S-8S mode.

The xSPI protocol defines 1S-1S-1S mode for general backwards compatibility, and 8D-8D-8D for maximum throughput. The ROM OSPI boot mode is compatible with 1S-1S-1S mode, but is not compatible with 8D-8D-8D mode.

Some SPI Flash memory devices also offer the legacy 1S-1S-8S mode, which is compatible with the ROM OSPI boot mode.

Note that the OSPI IP can in general support 8D-8D-8D mode with an appropriate software driver. The limitation is only for ROM boot which hard codes the 1S-1S-1S and 1S-1S-8S modes.

Workaround(s):

If 8-bit data rate is required for boot, a SPI Flash memory device should be carefully selected that is compatible with 1S-1S-8S mode of operation.

If 1-bit data is sufficient for boot, an xSPI Flash memory device should be chosen that explicitly supports the 1S-1S-1S mode at boot. Different memory vendors may only support this mode on specific part variants.

i2116 ***MSMC: Set-hazarding logic withholding RT access waiting on NRT access completion***

Details:

The DDR controller prioritizes writes over reads to the same page. Additionally, MSMC hazards transactions on the same set regardless of the real-time attribute. Due to these two facts, a stream of writes to the same page followed by a non real-time read to the same page can effectively block out a real-time access command indefinitely.

Example sequence:

1. Stream of Writes to page A sent from MSMC to DDR Controller
2. Non Real-Time Read to page A sent from MSMC to DDR Controller
   - This command will be stalled in the DDR Controller behind the completion of the 1) Stream of Writes
3. Real-Time Access to same set as the 2) Non Real-Time Read will be stalled inside MSMC due to Set Hazarding

Workaround(s):

Software should attempt the following workarounds in order of least to most impact to SW.

1. Cadence DDR controller prioritizes writes to the same page over a read from another page causing a delay in returning the read. Try reducing the DDR controller command_age_count from 0xto 0xF - corresponding to reducing the command age count from 16 DDR refresh cycles (62 us) to 1 refresh cycle (3.9 us). In most of the cases issue is resolved with this setting, but in some cases there are still some underflows. In that case SW may require either 2 or 3 workaround.
2. If possible set the ARM MMU attribute to configure DDR as "Normal memory" instead of "Device memory" type. This makes ARM to DDR access to be more efficient and helps to alleviate the problem. This is the observation based on test results so far,

i2116 (continued) ***MSMC: Set-hazarding logic withholding RT access waiting on NRT access completion***

but it may need more analysis and further system testing. If this workaround is not possible in the system, SW may require workaround 3).

3. If possible make the Real-Time access as non IO-coherent. Set the RT access ATYPE = 3 for non-virtualized cases, and set ATYPE=1 & MEMTYPE=0 for PVU specific cases. This forces the RT traffic to bypass the MSMC set-hazarding logic. SW will have to do the cache operations.

i2118 ***R5FSS: Debug Access in Lock-Step Mode May Result in Failure***

Details: Debug accesses may result in R5FSS going out of lock-step. The debug access could be any debug operation where the debug subsystem is controlling the following R5FSS inputs - cpuhalt, dbgen, niden, dbgnoclkstop. This issue happens only rarely. As a result lock-step miscompare interrupt will fire from R5FSS towards the ESM (Error Signaling Module) in the SoC. Also, the second core (R5FSS_CORE1) will no longer be functional.

Workaround(s): User can disable R5FSS interrupts in the ESM and continue with the debug operation of first core (R5FSS_CORE0).

Another workaround is to do all code debug in Split (non lock-step) mode. From a debug perspective, there is no difference in Split mode vs Lock-step mode – it is the same code which runs on both the cores.

i2119 ***HyperFlash: HyperFlash is Not Supported***

Details: The Hyperflash interface is not supported.

Workaround(s): None. HyperFlash should not be used.

i2129 ***Cortex-R5F: High Priority Interrupt is Missed by VIM***

Details: The VIM will not always interrupt the currently active interrupt when a higher priority interrupt arrives immediately afterwards. In these cases, the higher priority interrupt will only be taken after the completion of the current lower priority interrupt or when an even higher priority interrupt arrives. The impact of this issue is higher than expected interrupt latency for the high priority interrupt. Both Vector Interface (VIC) servicing and MMR Interface servicing modes of VIM are affected.

Workaround(s): This is a problem which affects applications which are latency critical and wants pre-empting of low priority interrupts with higher priority interrupts. If the application is not latency critical, then the behavior may be acceptable (the high priority interrupt will be eventually taken after the low priority interrupt completes).

Alternatively, user can implement a completely SW managed interrupt servicing scheme, where every ISR (Interrupt Service Routine) shall check for the presence of an active higher priority interrupt (by reading Interrupt Raw Status registers in VIM) and jumping to the ISR corresponding to that interrupt.

i2132 ***Cortex-R5F: Interrupt Preemption (Nesting) is Unavailable if Using VIM Vector Interface for Interrupt Handling***

Details: Interrupt preemption, which is the nesting of high priority interrupts inside a low priority interrupt, is unavailable if using VIM Vector Interface for interrupt handling. Nesting of a high priority interrupt within a low priority interrupt will result in corrupted operation of the processor. The issue only impacts Vector Interface method of interrupt handling provided

i2132 (continued) ***Cortex-R5F: Interrupt Preemption (Nesting) is Unavailable if Using VIM Vector Interface for Interrupt Handling***

by VIM. It does not impact MMR interface method of interrupt handling. Issues impact both FIQ and IRQ interrupts.

Workaround(s): If using Vector Interface method, user should not set the I/F bit (to enable nesting of interrupts) in CPSR.

If interrupt nesting is required then user should only use MMR interface method for interrupt handling. Note that, MMR interface method incurs an additional latency for Interrupt Service Routine (ISR) entry compared to Vector Interface method.

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).

i2138 ***PSIL: Configuration accesses and source thread teardowns may cause data corruption***

Details: When performing a tear down on one source thread, a single data phase on a different source thread may be lost. This affects all PSIL_ENDPT modules that have more than 1 source thread (ICSSG/CSI/SA2UL)

Also, if a configuration response is sent out by a PSIL Endpoint Gasket while data is also being sent out, the response will cause data corruption. This can affect data transmitted long after the configuration response occurs. This affects all PSIL_ENDPT users that require width adaption where their PSIL port is less than 128-bit (SA2UL).

Workaround(s): All PSIL source threads must be idled by disabling the source of Rx traffic before attempting a configuration access or a teardown of any source thread. For ICSSG/CSI, idling of PSIL source threads can also be done by pausing all source threads.

i2139 ***CPSW: ALE Incorrectly Routes Packets With CRC Errors***

Details: For InterVLAN, OAM, or packets routed with the ALE egress opcode feature, the Address Lookup Engine (ALE) incorrectly routes received (CPSW ingress) packets with CRC errors when errored packets should have been dropped. The routed packet egresses with a CRC error which is allowed but is not preferred.

This only affects InterVLAN, OAM, and packets using the ALE egress opcode feature in a non-cut-thru CPSW.

Workaround(s): None.

i2141 ***CPTS: GENF and ESTF Nudge Value Not Cleared by Hardware***

Details: The GENF and ESTF nudge value in TS_GENFn_Nudge is not cleared when the nudge occurs. This is generally acceptable as software typically does not need to know exactly when the nudge occurs.

Workaround(s): The software workaround for this issue is:

1. Write a zero value to the CPTS_TS_GENF_NUDGE_REG_j[7-0] NUDGE bit field.
2. Write the desired 2's complement nudge value to the CPTS_TS_GENF_NUDGE_REG_j[7-0] NUDGE bit field.
3. The nudge will occur in CPTS_GENFn_LENGTH[31-0]/2 CPTS_REF clocks.

i2143 ***UDMAP: TX Channel SA2UL teardown issue***

Details: Performing a UDMAP TX channel teardown of SA2UL can result in undefined behavior on the PSIL channel.

Workaround(s): There are two software workarounds:

1. Follow TX Channel Teardown by a teardown of the pairing registers (including clearing the enable bit in PSIL register 0x2), and re-pairing of the channel.
2. Suppress teardown packet generation of the channel via the Tx Channel N Configuration Registers.

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.

i2148 ***CPSW: CPSW Directed Frames are Not Observed When Classification Overrides the Destination Port Via the Egress Opcode Feature***

Details: Directed frames sent via software with the 802.1CB header are incorrectly redirected back to the host port. Directed frames should not be overridden by the ALE Egress Op logic.

Workaround(s): Ensure that the Host traffic is excluded from the classifier.

i2149 ***MSMC: MSMC Scrubber Only Targets Bottom 16 of 32 Ways of SRAM/L3$***

Details: MSMC Scrubber periodically scans through MSMC SRAM/L3$, Snoop Filter, and Tags for correctable 1-bit errors and then corrects them. This is to reduce the probability of multiple 1-bit errors accumulating over time and becoming non-correctable 2-bit errors.

Due to an error in the address decoding, MSMC Scrub transactions only access the lower half of the L3$ Tag ways (0-15). Ways 16-31 are never accessed. The corresponding L3$ Data RAMs will also not be accessed by Scrubber.

Customers will see an increase in probability of accumulating 2-bit detectable/non-correctable errors in upper half (upper 16 ways) of MSMC L3$ Tag and corresponding Data.

This issue does not affect the MSMC SRAM and only applies to L3 Cache.

Workaround(s): There is no complete software workaround.

i2149 (continued) ***MSMC: MSMC Scrubber Only Targets Bottom 16 of 32 Ways of SRAM/L3$***

Software can attempt to periodically flush the L2$ to allow MSMC EDC to be refreshed. This is not a complete workaround, however, since Arm® can silently evict cache lines without alerting MSMC.

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.

i2162 ***R5FSS: The Same Interrupt Cannot be Nested Back-2-Back Within Another Interrupt***

Details: The nesting (preemption) of the same high priority interrupt inside a low priority interrupt is not possible for the second and subsequent times. The second occurrence of the high priority interrupt has to wait until the program exits the lower priority interrupt service routine (ISR). The issue only occurs if the high priority interrupt following a current preemption is the same as the one which caused the original preemption. If a different interrupt preempts the low priority ISR before the second occurrence of the original higher priority interrupt then there is no issue. This issue impacts both Vector Interface Method and MMR Interface Method of interrupt handling in VIM. The issue impacts both FIQ and IRQ interrupts.

Workaround(s): A software workaround exists. The objective of the SW workaround is to prevent back-2-back activation of the same interrupt, thereby removing the necessary condition of the bug. This can be achieved by reserving the highest priority level (Priority-0), and using that priority for a dummy interrupt (any one out of 512 interrupts available in R5FSS), and calling this dummy interrupt inside every ISR. Further, the R5FSS core itself need not enter this dummy ISR (it can be masked), only the handshake with VIM around this dummy ISR needs to happen.

A sample pseudo-code is shown below. If required, TI can provide the necessary drivers which implement this workaround.

```
any_isr_routine {
...
1:      set I/F bit in CPSR ; //so R5FSS cannot be interrupted again. I for
irq, F for fiq
2:      Trigger dummy_intr; //writing 1'b1 to Interrupt RAW Status/Set Register
bit in VIM corresponding to the chosen dummy_intr
3:       rd_irqvec; //Read IRQVEC register in VIM to acknowledge dummy_isr
4:       clear dummy_isr; //writing 1'b0 to Interrupt RAW Status/Set Register
bit in VIM corresponding to the chosen dummy_intr
5:       wr_irqvec;//Write to IRQVEC register in VIM to denote end of interrupt
6:       clear I/F bit in CPSR;
…
}
Note: Depending on where the workaround code is inserted in the ISR, step 1 & 6
may not be needed.
```

The draw-backs with this workaround are, Priority-0 cannot be used (only Priority 1-15 are available), and the added latency in ISR execution.

i2164 ***R5FSS: Errors in ECC injection logic are not detected because the pending interrupts are tied low***

Details: The device has the ability to intentionally introduce ECC errors on memory reads to test that the ECC checking logic is working (a test-for-diagnostic). The logic also contains the ability to detect faults in this injection logic (latent faults). However, there is an issue that causes these errors not to be reported. The result is a slight reduction in latent fault coverage reflected in all FMEDA calculations. The diagnostic (ECC) and the ability to test the diagnostic are not affected.

Workaround(s): None.

i2165 ***R5FSS: Access to CPU1 TCMs hangs in Lockstep mode***

Details: When R5FSS is operating in Lock-step mode, only CPU0 peripherals (TCMs and other components) are used. CPU1 peripherals are unused and are kept disabled. Then, any external access (from SoC) to CPU1 peripherals will not get a response back and will result in system hang.

Workaround(s): When R5FSS is in Lock-step mode, the software must program the firewall for the CPU1 TCM memory access port and CPU1 config port to block any transactions from the SoC level. These firewalls are present in the SoC level CBASS which connects to R5FSS.

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.

i2184 ***CPSW: IET express traffic policing issue***

Details: This applies to 9-port CPSW, 5-port CPSW, 3-port CPSW, and 2-port CPSW IET traffic.

In IET (Interspersed Express traffic), if a preempted packet was interrupted by an express packet, two things can occur:

1. If the express traffic is policed, the frame size for the preempted packets is applied to the express traffic policer. Assuming a policer was set up to rate schedule an express traffic stream, it would take a hit by the preempted packet size it interrupted. The preempted packet also takes on the express traffic policer status. As a result, preempted packets could get dropped along with other express traffic due to the express traffic policer.
2. If the express traffic was not policed, the interrupted preempt packet would not get its packet size applied to the preempted policer.

Workaround(s): Do not police IET express traffic.

i2185 ***CPSW: Policer color marking issue***

Details: Only applies to CPSW9G and CPSW5G.

When packets from two different ports hit the same policer such that one port has a large packet and the other has a short packet and the short packet arrives just after the large packet starts, the short packet will stop the backlog counting, resulting in potentially flagging the next frame for this policer as yellow when it should have been green. Because the policer is normally set up to not drop yellow, it should not cause an issue. This is only true of packets that arrive on different ports that share the same policer index.

Workaround(s): Ensure policers are unique to ports.

i2187 ***MSMC: Cache Resize to 0 Refreshes Tags instead of Updating them***

Details: Data corruption (MSMC returning all 0's) occurs upon changing MSMC L3$ Size from non-zero to zero and back to non-zero for lines that previously had cached dirty data in MSMC's L3$ (DDR). A 0->N configuration directly after release of MSMC reset is not impacted by this issue.

MSMC internal cache resize transactions are always marked as *non-allocating* misses. Tags are only updated with new values on *allocating* misses and hits. This results in cache resize operations leaving the tags unchanged, while zeroing out the underlying data.

Because all existing TAGs remain in MSMC when changing L3 Cache Size but data is zeroed, subsequent reads to these previously cached lines will see all 0's returned for data.

Workaround(s): Reset MSMC after L3 Cache is resized from N to 0 and prior to resizing L3 from 0 to X. This workaround preserves data because the L3 Cache Size N -> 0 transition forces data into DDR allowing DDR (in self refresh) to contain valid data.

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 the application note spract2 (link: https://www.ti.com/lit/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.
3. STIG reads must be padded with extra bytes, and the received data must be right-shifted.

i2193 ***PRU-ICSSG: Erroneous FDB aging behavior during switch operation with bucket size of 8***

Details: When the PRU-ICSSG is configured as a 1Gbps Ethernet switch, and the FDB is configured with a bucket size of 8, during an FDB lookup there the "touch bit" indicating the entry lookup match may be incorrectly set.

The FDB lookup can be initiated by either port or by host action. Each row within the FDB has 4 “buckets,” or 32 Bytes, and in 8-bucket configuration there are two rows that hold the entries corresponding to the hardware lookup. The error is caused by the touch bit in the even row being updated regardless of whether the specific match is to the even or odd row. For example, entry #4 was an SA hit, but entry #0 has its touch bit set; or entry #5 was an SA hit, but entry #1 has its touch bit set.

Workaround(s): Use bucket size of 2 or 4 to avoid. If a bucket size of 8 is desired, dynamic entries must be populated only in the even rows of the table.

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***

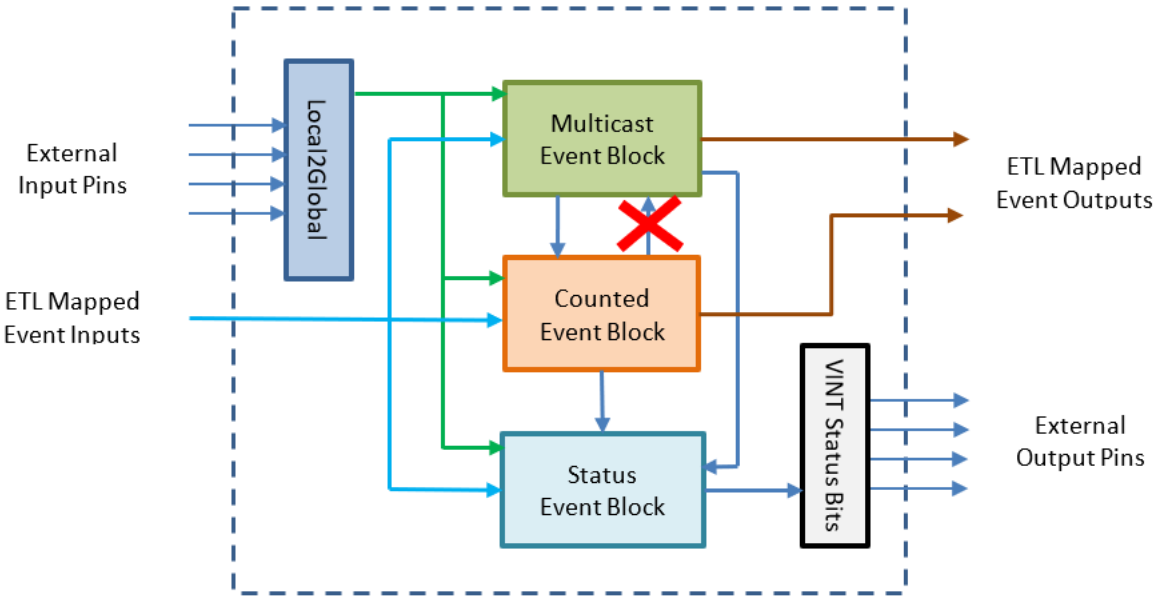


Figure 3-3. Interrupt Aggregator Version 1.0

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

i2198 ***DRU, UTC: Issue with setting ICNT3 to 0 when not being used***

Details: DRU/UTC won't recognize a TR as one dimensional if ICNT3 is set as 0. For the event generation or triggering hold of a TR on ICNT1 decrement. If ICNT2 and ICNT3 are not used they can usually be set as either 0 or 1 with the same effect. But in the case of doing the 1D trigger or event when pushing the TR to the queue a value of 0 on ICNT3 will not be seen as a 1D TR causing the TR to not remove the previous trigger or send the event on the end of the TR.

Workaround(s): The TR should always set ICNT 3 to 1 if it is not being used.

i2204 ***CSI: Interface Setup/Hold Timing Does Not Meet MIPI DPHY Spec above 600MHz***

Details: When running the CSI2 interface at greater than 600MHz (1.2Gbps per lane), setup/ hold times are not compliant with limits required by the MIPI CSI2 DPHY specification. Systems using the CSI2 interface at less than or equal to 600MHz are not affected.

Workaround(s): Since the CSI2 interface includes up to 4 data lanes (plus 1 clock lane), data can be distributed across multiple lanes in order to keep the clock rate lower.

i2207 ***CBASS: Command Arbitration Blocking***

Details: When the interconnect arbitrates commands from multiple sources, the higher priority request always takes precedence. Requests that are at the same priority level are arbitrated in round-robin fashion. The issue is after the higher priority request goes idle and there are two or more pending requests that are at the same priority level, the hardware selects one of them arbitrarily. A potential issue may arise when software polls from multiple sources to the same endpoint: after servicing the higher priority source, the hardware may repeatedly select the same lower priority source for access. That means

i2207 (continued) ***CBASS: Command Arbitration Blocking***

other same-lower priority requests may be blocked for a long time, and in the worst case if there are dependencies between the polling sequences, the software may run into a livelock state.

This issue only affects certain interconnect where in one switch module there are at least three sources that can access the same target simultaneously. Also note that when all requests are at the same priority level, the issue does not apply.

Workaround(s): When multiple sources are simultaneously polling from the same endpoint, and there is expected dependency based on the read data, ensure that all sources are sending the read commands at the same priority level. The source that breaks the dependency should be at equal or higher priority than other dependent sources.

i2231 ***DDR: LPDDR4 and DDR3L are not supported***

Details DDR3L and LPDDR4 are not supported.

Workaround Use DDR4.

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

i2245 ***DMSC: Firewall Region requires specific configuration***

Details The ECC Aggregator inside DMSC (DMSC0_ECC_AGGR) has an endpoint firewall which is used to protect this region. By default, this firewall blocks all the transactions except from the M3 core inside DMSC.

Workaround If another processor or endpoint needs to access DMSC0_ECC_AGGR region, software shall configure the firewall region with starting address 0x0 and end address 0xFFFF_FFFF, using the CBASS_FW_REGION_i_START_ADDRESS and END_ADDRESS registers associated with the DMSC0_ECC_AGGR region. This is the only allowable address configuration for this region.

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 Iclk 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.

i2014	***PCIE: PCIE1 instance does not support outbound address translation bypass***
Details	PCIe module supports an outbound address translation bypass feature by using a CBA interface signal called ASEL. This allows SoC reads and writes through PCIe link to use a PCIe address directly instead of using an SoC level address. Transactions are routed internally to SoC using an ASEL value instead of an SoC address. There are two PCIe instances, PCIE0 and PCIE1. Each PCIe outbound link is assigned a unique ASEL value: PCIE0 is assigned an ASEL value of 1 and PCIE1 is assigned an ASEL value of 2. Due to an integration issue related to ASEL routing, PCIE1 does not support this capability and all outbound transactions for PCIE1 must use an SoC level address with address translation performed by the PCIe module. As a result of this issue, DMA is not programmed to send any transactions with ASEL set to 2. PCIE0 is not affected by this issue and does support outbound address translation bypass capability.
Workaround	None

i2145	***VTM: Enabled interrupt event status registers incorrectly return raw unmasked values***
Details:	Reads of the Enabled interrupt event status registers VTM_LT_TH0_INT_EN_STAT_CLR, VTM_GT_TH1_INT_EN_STAT_CLR, and VTM_GT_TH2_INT_EN_STAT_CLR incorrectly return the raw unmasked pending interrupt values for each voltage domain.
Workaround(s):	Software should read each threshold's INT_EN_STAT_CLR and associated INT_EN_SET/CLR registers, then manually bit-wise mask the int_vd bitfield to get the correct masked view of the threshold's INT_EN_STAT_CLR read result.

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.
- 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

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

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.

i2173 ***MCU: MCU domain may hang if main domain is issued a reset***

Details: The MCU domain is designed to be able to work completely independently from the main domain of the device. If the main domain is put in to reset, the MCU domain should continue to function uninterrupted, even if there are pending transactions from MCU masters to Main domain slaves. The purpose of this feature is to ensure that if the main domain must be put in to reset because of a fault, the MCU domain continues to operate uninterrupted. The Main domain could subsequently be brought out of reset and re-started. The issue is that sometimes, if there is a transaction outstanding from MCU to Main domain and Main is put in to reset (unexpectedly or intentionally because of a fault) it may lead to a hang in the MCU domain interconnect. This can in turn cause MCU masters to become unresponsive.

Workaround(s): The first workaround is to ensure that there are no MCU transactions outstanding from MCU to Main when Main is in reset. This must be enforced at the system level. It is possible to use this workaround when the Main domain reset is done in an orderly fashion, but may not be possible if the main domain reset is unexpected or driven by a fault.

The second workaround is to not put the Main domain in to reset in the case of a fault. The Main domain may be otherwise ‘taken off line’, but the system does not actually assert warm reset to the Main domain. In this case, the MCU will continue to function properly, including unwinding any pending transactions from the Main domain.

Neither of these workarounds is robust to an unexpected reset or a fault that requires reset to prevent propagation or damage.

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 3-3. 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

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).

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.

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 Iclk 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.

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 OK 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 - 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 OK since the next periodic event will retrigger the timeout interrupt

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

-OK 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).

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

i2328 ***Boot: USB MSC boots intermittently***

Details: USB MSC Host boot may fail due to a protocol timing violation present in the ROM USB device enumeration process. USB DFU boot is unaffected.

Workaround(s): No workaround is available. Some USB MSC devices may tolerate this protocol violation and function as expected. Due to the internal component variability of broad-market MSC devices, a list of tolerant devices cannot be provided.

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 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.

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

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.

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.

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

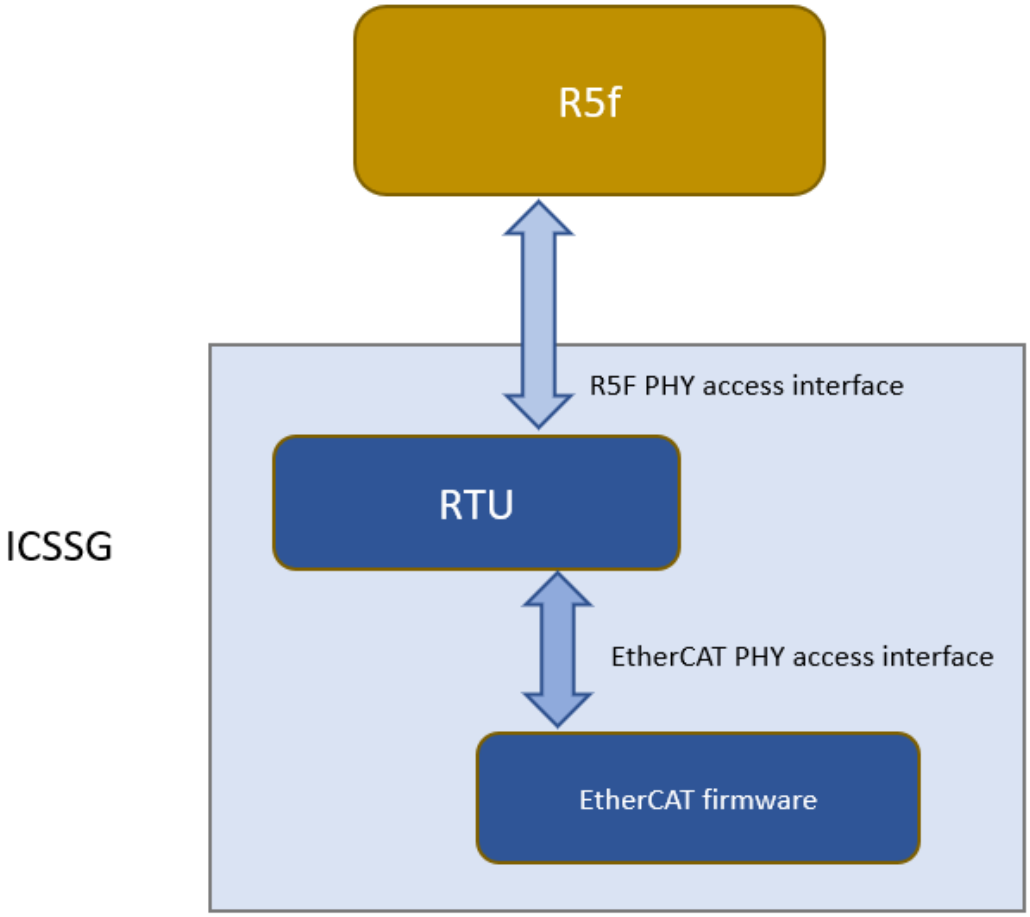


Figure 3-4. MDIO Emulation via Manual Mode using PRU Core

i2040 ***PCIe: Root Port does not set the Link Bandwidth Management and Link Autonomous Bandwidth status bits***

Details: PCIe defines two status fields called ‘Link Bandwidth Management’ and ‘Link Autonomous Bandwidth’ as part of the ‘Link Status Register’ for Downstream port. Hardware is expected to set one of these status fields when a link speed change event occurs depending on the reason for this change.

These status bits may not be set even for a valid link speed change. This can happen if the PHY communicates the link speed change event to MAC after a high delay (> 800ns).

As a result of this issue, software may not be able to differentiate between autonomous link speed change, directed link speed change through ‘Retrain Link’ field in 'Link Control Register', or speed change due to unreliable link.

This issue only applies to Root Complex mode.

Workaround(s): None. Software can periodically read the ‘Current Link Speed’ field in ‘Link Status Register’ to detect link speed change. But, there is no workaround to obtain additional information on whether link speed change was autonomous, directed or due to unreliable link.

i2041 ***PCIe: Calculated Negative Round-trip Time Causes Wrong PTM Requester Local time***

Details:

PTM Master Time is calculated by the Requester using the round trip time for PTM request and response (Requester to Responder and back). If the calculated round trip time is negative, it causes the Requester local clock to have a huge offset of 2^63 nanoseconds.

The following is the equation used to calculate PTM Master Time (same as PTM Requester local clock):

PTM Master Time at t1' = t2' – round_trip/2

Where round_trip = ((t4-t1) – (t3-t2))

Please note that t1 is the estimated time for PTM Request at the Requester's upstream port. This includes the local time added with the programmed MAC and PHY latencies for transmit. Similarly t4 is the estimated time for PTM Response and the Requester's upstream port. This includes local time subtracted by the programmed MAC and PHY latencies for receive.

PTM Master Time calculated after receiving ResponseD message is then used to update the Requester's local clock.

Round trip time can become negative if (t3-t2) is greater than (t4-t1). This can happen if either or both of below scenarios happen:

1. PTM Requester's transmit and receive latency registers are programmed with values that are close to or greater than actual PHY and MAC latency.

2. The clock frequencies between Requester's Upstream port and Responder's Downstream port have significant difference and the actual wire delays between these two ports are short (few nanoseconds).

The issue arises because the round trip time is stored as an unsigned number and is right shifted to create the divide by two value. As a result, round_trip/2 component in the PTM Master Time equation becomes a very large positive number. This leads to the following issues:

1. 2^63 nanoseconds offset in the adjusted Requester's local clock going to CPTS for hardware push.

2. The VSEC PTM registers have this offset as well.
The affected PTM registers are PCIE_EP_PTM_REQ_LOCAL_LSB_OFF, PCIE_EP_PTM_REQ_LOCAL_MSB_OFF, PCIE_EP_PTM_REQ_T1_LSB_OFF, PCIE_EP_PTM_REQ_T1_MSB_OFF, PCIE_EP_PTM_REQ_T1P_LSB_OFF, PCIE_EP_PTM_REQ_T1P_MSB_OFF, PCIE_EP_PTM_REQ_T4_LSB_OFF, PCIE_EP_PTM_REQ_T4_MSB_OFF, PCIE_EP_PTM_REQ_T4P_LSB_OFF, PCIE_EP_PTM_REQ_T4P_MSB_OFF.

Workaround(s):

One of the following workarounds have to be implemented:

1. Program the transmit latency (PCIE_EP_PTM_REQ_TX_LATENCY_OFF register) and receive latency (PCIE_EP_PTM_REQ_RX_LATENCY_OFF) to zero. This will eliminate the issue as long as clock drift between Requester/EP and Responder/RC is less than 4100pm. However, this workaround will increase the deviation for Requester clock with respect to Responder's PTM Master Time.

2. Detect and correct for this 2^63 nanoseconds offset: -

- Ignore bits 63 and 62 of the local clock and timestamps in VSEC PTM register space.
- Do not set PTM_CLK_SEL field in PTMCFG register to values of 62 or 63. This will prevent the CPTS hardware push logic from using the two most significant bits of the PTM local clock. This limits maximum PTM time to less than 136 years.

i2043 ***PCIE: Compliance test fails with certain lane reversal and lane polarity inversion conditions when using Enter Compliance bit***

Details: Compliance test fails because modified Compliance Pattern is not detected in the following cases when compliance is entered using Enter Compliance bit in Link Control 2 register:

1. Using 8b/10b encoding (2.5 GT/s and 5 GT/s) and lane is polarity inverted.

2. Using 128b/130b encoding (8 GT/s) and lane is both polarity inverted and reversed (negotiated during previous link training).

Workaround(s): Do not use Enter Compliance bit in Link Control 2 register for forcing entry to Compliance state if this issue applies (based on lane reversal and polarity inversion combinations). Instead, compliance entry has to be initiated using Compliance Receive bit in training sequence packets.

i2151 ***ADC: Debounce time control register***

Details: CTRLMMR_WKUP_PADCONFIG76.DEBOUNCE_SEL controls the debounce time for MCU_ADC0_AIN0:7 and CTRLMMR_WKUP_PADCONFIG84.DEBOUNCE_SEL controls the debounce time for MCU_ADC1_AIN0:7. These registers set the debounce period for all of the input channels on the respective ADC whether or not the specific input (e.g. MCU0_ADC0_AIN0 or MCU_ADC1_AIN0) is used.

Workaround(s): None

i2262 ***DDR: Mode Register write busy indicator not cleared after completing software-driven mode register access***

Details: If the hardware-driven mode register access (automatic DDR4 Mode-Register-Set or periodic LPDDR4 Mode-Register-Read for temperature derating) starts just before sending software-driven mode register access (setting the MRCTRL0.mr_wr register to 1), the MRSTAT.mr_wr_busy field is not cleared to 0 until the next hardware-driven mode-register access starts.

Workaround(s): Avoid conflict between software-driven and hardware-driven mode register accesses. For DDR4, disable the sources of hardware-driven mode register accesses by writing the following registers before initiating the software-driven mode register access.

PWRCTL.mpsm_en = 0

PWRCTL.selfref_sw = 0

PWRCTL.selfref_en = 0

HWLPCTL.hw_lp_en = 0

DFIPHYMSTR.dfi_phymstr_en = 0

For LPDDR4, disable the temperature derating feature (set register DERATEEN.derate_enable to 0) before initiating the software-driven MR access.

i2264 ***DDR: Timing violation from read/write command to Software-Initiated MRW command (LPDDR4 only)***

Details: In LPDDR4 mode, if software sends a Mode Register Write (MRW) command using the MRCTRL0 and MRCTRL1 registers while read/write transactions are being executed, a read/write command can be followed by a MRW command resulting in a read to MRW, or a write to MRW command timing violation.

Workaround(s): Perform software initiated MRW commands after entering self-refresh mode by performing the following sequence:

1. Disable PHY master interface
   - Set DFIPHYMSTR.dfi_phymstr_en to 0 (note that this is documented as static, but implemented as dynamic register)
2. Disable the following automatic self-refresh entry (if using)
   - Set PWRCTL.selfref_en to 0
3. Ensure that DDR is not in self-refresh or SR-Powerdown.
   - Wait for STAT.operating_mode to be not equal to 3'b011 (Self-Refresh/SR-Powerdown mode)
4. Enter self-refresh 1 state
   - Set PWRCTL.stay_in_selfref to 1
   - Set PWRCTL.selfref_sw to 1

i2264 (continued) ***DDR: Timing violation from read/write command to Software-Initiated MRW command (LPDDR4 only)***

- Wait for STAT.selfref_type to be not equal to 2'b01 (Self-Refresh caused by PHY master request)
- If STAT.selfref_state is 2'b10 (SR-Powerdown) or 2'b11(Self-Refresh 2), set PWRCTL.stay_in_selfref to 0 and PWRCTL.selfref_sw to 0, then go back to step 2
- Wait for STAT.selfref_state to become 2'b01 (Self-Refresh 1)

5. Send MRW commands using MRCTRL0 and MRCTRL1 registers

6. Enter SR-Powerdown mode.
- Set PWRCTL.stay_in_selfref to 0
- Wait for STAT.selfref_state to become 2'b10 (SR-Powerdown)

7. Exit self-refresh mode.
- Set PWRCTL.selfref_sw to 0
- Wait for STAT.selfref_state to become 2'b00 (Not Self-Refresh)

8. Enable PHY master interface
- Set DFIPHYMSTR.dfi_phymstr_en to 1

i2265 ***DDR: Issuing MRR/MRW commands in self-refresh state (LPDDR4 only)***

Details: PWRCTL.stay_in_selfref=1'b1 allows user to pause in self-refresh 1 or self-refresh 2 states for software to perform Mode Register Read (MRR) and Mode Register Write (MRW) commands. If controller-PHY hardware handshake occurs while in Self Refresh Power Down (SRPD), then PWRCTL.stay_in_selfref=1 will not pause in self-refresh states as expected. The MRR/MRW commands could be sent in normal mode. Depending on if the MRR/MRW commands being sent needs the DRAM to be in Self Refresh, this may violate JEDEC requirements for LPDDR4.

Workaround(s): Use the following sequence to issue software initiated MRR/MRW commands during entering self-refresh:

1. Disable PHY master interface
- Set DFIPHYMSTR.dfi_phymstr_en to 0 (note that this is documented as static, but implemented as dynamic register)

2. Disable the following automatic self-refresh entry (if using)
- Set PWRCTL.selfref_en to 0

3. Ensure that DDR is not in self-refresh or SR-Powerdown.
- Wait for STAT.operating_mode to be not equal to 3'b011 (Self-Refresh/SR-Powerdown mode)

4. Enter self-refresh 1 state

- Set PWRCTL.stay_in_selfref to 1
- Set PWRCTL.selfref_sw to 1
- Wait for STAT.selfref_type to be not equal to 2'b01 (Self-Refresh caused by PHY master request)
- If STAT.selfref_state is 2'b10 (SR-Powerdown) or 2'b11(Self-Refresh 2), set PWRCTL.stay_in_selfref to 0 and PWRCTL.selfref_sw to 0, then go back to step 2
- Wait for STAT.selfref_state to become 2'b01 (Self-Refresh 1)

5. Send MRR/MRW commands using MRCTRL0 and MRCTRL1 registers

6. Enter SR-Powerdown mode.
- Set PWRCTL.stay_in_selfref to 0
- Wait for STAT.selfref_state to become 2'b10 (SR-Powerdown)

i2265 (continued) ***DDR: Issuing MRR/MRW commands in self-refresh state (LPDDR4 only)***

7. Exit self-refresh 2 state
- Set PWRCTL.stay_in_selfref to 1
- Set PWRCTL.selfref_sw to 0
- Wait for STAT.selfref_state to become 2'b11 (Self-Refresh 2)

8. Send MRR/MRW commands using MRCTRL0 and MRCTRL1 registers

9. Exit self-refresh mode
- Set PWRCTL.selfref_sw to 0
- Wait for STAT.selfref_state to become 2'b00 (Not Self-Refresh)

10. Enable PHY master interface
- Set DFIPHYMSTR.dfi_phymstr_en to 1

i2266 ***DDR: Controller derating logic does not consider system temperature when derating is enabled (LPDDR4 only)***

Details: Controller derating logic samples Mode Register 4 (MR4) of the LPDDR4 only if MR4[7], the Temperature Update Flag (TUF) is equal to 1. However, if the system is hot or cold prior to DERATEEN.derate_enable = 1, then there is no guarantee that MR4[7] = 1 will be set even though the MR4[2:0] != 3'b011 (not 1x tREFI case).

As the logic does not sample MR4[2:0] correctly it can result in incorrect refresh period being used, or incorrect derated timing parameters (tRCD, tRAS, tRP, tRRD) being used. This can result in to functional impact in normal operation such as protocol violation leading to potentially unexpected behavior of the memory system and/or possible data loss/corruption in SDRAM.

Workaround(s): When enabling controller derating logic:1. Set DERATEEN.derate_enable = 12. Perform a read of MR4 register via software registers MRCTRL*/MRSTAT3. If MR4[7] = 0 and MR4[2:0] = 3'b011, then it is safe to use controller derating logic, otherwise set DERATEEN.derate_enable=0 and temperature derating should be performed via software.

1. Set DERATEEN.derate_enable = 1
2. Perform a read of MR4 register via software registers MRCTRL*/MRSTAT
3. If MR4[7] = 0 and MR4[2:0] = 3'b011, then it is safe to use controller derating logic, otherwise set DERATEEN.derate_enable=0 and temperature derating should be performed via software.

i2268 ***DDR: Controller can violate timing from Self-Refresh Exit (SRX) to Power-Down Entry (PDE) for DDR3***

Details: The DDR3 JEDEC specification states: "CKE must remain HIGH for the entire Self-Refresh exit period (tXSDLL) for proper operation except for Self-Refresh re-entry". However, if a Power-Down Entry (PDE) is requested immediately after a Self-Refresh Exit (SRX), the controller can issue the PDE command tXS time after the SRX command, and thereby violating the DDR3 JEDEC specification.

Workaround(s): Set DRAMTMG8.t_xs_x32 = DRAMTMG8.t_xs_dll_x32.Please note that this will result in increased timing for commands that do not require a locked DLL from self-refresh exit, i.e., commands other than reads.

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

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

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.

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.

Trademarks

CoreSight™ is a trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
Arm® and Cortex® are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
All trademarks are the property of their respective owners.

Revision History

Changes from November 1, 2022 to May 5, 2023 (from Revision H (November 2022) to Revision I (May 2023)) **Page**

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

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

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