Errata

J721E DRA829/TDA4VM Processors Silicon Revision 1.1/1.0

TEXAS INSTRUMENTS

ABSTRACT

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

Table of Contents

1 Modules Affected

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

Table 1-1. Usage Note by Modules

MODULE	USAGE NOTE
USB	i2134 USB: 2.0 Compliance Receive Sensitivity Test Limitation — USB: 2.0 Compliance Receive Sensitivity Test Limitation

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

Table 1-2. Advisories by Modules

MODULE	ADVISORY	SILICON REVISIONS AFFECTED	
		SR 1.0	SR 1.1
Boot	i2038 — Boot: FAT16 Fails When Root Block Resides in More Than One Cluster	YES	YES
	i2081 — Boot: ROM Maximum Timeout per Boot Mode Will Be Half of the Original Value from TRM	YES	NO
C71x	i2063 — C71x: VCOP Aliasing for CPU Loads and Stores Is Not Supported for Non-Aligned Accesses to the Last Line in the IBUF Buffers	YES	YES
	i2064 — C71x: DMA Accesses to L1D SRAM May Stall Indefinitely in the Presence Cache Mode Change or Global Writeback in Specific Conditions	YES	YES
	i2065 — C71x: The C71x Memory System and CPU May Stall Indefinitely in the Presence L1D Snoops	YES	YES
	i2079 — C71x: DMA Accesses to L1D SRAM May Stall Indefinitely in the Presence of CPU Traffic in Specific Conditions	YES	YES
	i2087 — C71x: MMA HWA_STATUS Reports Errors Before Application Starts	YES	YES
	i2117 — C71x: Register Corruption When MMA HWARCV is in Parallel With Load or Store With uTLB Miss	YES	NO
	i2131 — C71x: Memory System May Hang During L2 Writeback Invalidate Operation when L2 Scrubber is Enabled	YES	YES
CPTS	i2083 — CPTS: GENF (and ESTF) Reconfiguration Issue	YES	YES
	i2141 — CPTS: GENF and ESTF Nudge Value Not Cleared by Hardware	YES	YES
CPSW	i2139 — CPSW: ALE Incorrectly Routes Packets With CRC Errors	YES	YES
	i2148 — CPSW: CPSW Directed Frames are Not Observed When Classification Overrides the Destination Port Via the Egress Opcode Feature	YES	YES
CPSW9G	i2179 — CPSW9G: Reset isolation not working correctly	YES	NO
CSI	i2052 — CSI: CSI-Rx to CSI-Tx Retransmit Path Is Unavailable	YES	YES
DDR	i2155 — DDR: Controller DDRSS_CTL_194[9-8] BIST_RESULT Status is Unreliable	YES	YES
	i2158 — DDR: Controller Hangs if Data Traffic is Sent to DRAM After BIST Execution	YES	YES
	i2160 — DDR: Valid VRef Range Must be Defined During LPDDR4 Command Bus Training	YES	YES
DPHY	i2174 — DPHY: Reset sequence issue can lead to undefined module behavior	YES	NO
DSS	i2097 — DSS: Disabling a Layer Connected to Overlay May Result in Synclost During the Next Frame	YES	YES
ECC_AGGR	i2049 — ECC_AGGR: Potential IP Clockstop/Reset Sequence Hang due to Pending ECC Aggregator Interrupts	YES	YES
	i2191 — ECC_AGGR: Erroneous non-correctable parity error assertion for RAM80	YES	NO
eMMC	i2144 — eMMC: VIO Supply Sequencing	YES	YES
FSS	i2048 — FSS: MCU_FSS0_WRT_TYPE Register is Logging Incorrectly	YES	YES
GIC	i2101 — GIC: ITS Misbehavior	YES	YES
HyperBus	i2119 — HyperBus: HyperBus is Not Functional	YES	NO
I3C	i2150 — I3C: SDAPULLEN drives low instead of Hi-Z	YES	YES
	i2197 — I3C: Slave mode is not supported	YES	YES

Table 1-2. Advisories by Modules (continued)

MODULE	ADVISORY	SILICON REVISIONS AFFECTED	
		SR 1.0	SR 1.1
Internal Diagnostics Modules	i2103 — Internal Diagnostics Modules: Incorrect Reporting of ECC_GRP, ECC_BIT and ECC_TYPE Information for Functional Safety Errors	YES	YES
MCU	i2173 — MCU domain may hang if main domain is issued a reset	YES	NO
MMCSD	i2024 — MMCSD: Peripherals Do Not Support HS400	YES	YES
	i2090 — MMCSD: MMCSD1 and MMCSD2 Speed Issue	YES	NO
MSMC	i2149 — MSMC: MSMC Scrubber Only Targets Bottom 16 of 32 Ways of SRAM/L3$	YES	YES
OSPI	i2115 — OSPI: OSPI Boot Doesn't Support Some xSPI Modes or xSPI Devices	YES	NO
	i2189 — OSPI: Controller PHY Tuning Algorithm	YES	YES
PCIe	i2085 — PCIe: Gen2 Capable Endpoint Devices Always Enumerate as Gen1	YES	YES
	i2086 — PCIe: MMA Unsupported Request (UR) or Configuration Request Retry Status (CRS) in Configuration Completion Response Packets Results in External Abort	YES	YES
	i2094 — PCIe: End of Interrupt (EOI) Not Enabled for PCIe Legacy Interrupts	YES	YES
	i2100 — PCIe: Endpoint Destination Select Attribute (ASEL) Based Routing Issue	YES	YES
	i2153 — PCIe: Incorrect Reserved Bit Handling in TS1 Packet	YES	YES
PRU-ICSSG	i2180 — PRU-ICSSG: FDB table corruption during switch operation	YES	NO
PSIL	i2137 — PSIL: Clock stop operation can result in undefined behavior	YES	YES
	i2138 — PSIL: Configuration accesses and source thread teardowns may cause data corruption	YES	YES
R5FSS	i2099 — R5FSS: Deadlock Might Occur When One or More MPU Regions is Configured for Write Allocate Mode	YES	YES
	i2118 — R5FSS: Debug Access in Lock-Step Mode May Result in Failure	YES	YES
	i2129 — R5FSS: High Priority Interrupt is Missed by VIM	YES	YES
	i2132 — R5FSS: Interrupt Preemption (Nesting) is Unavailable if Using VIM Vector Interface for Interrupt Handling	YES	YES
	i2133 — R5FSS: Lock-Step Mode of Operation is Not Functional	YES	NO
	i2161 — Debugger Cannot Access VIM Module While It Is Active	YES	YES
	i2162 — R5FSS: The Same Interrupt Cannot be Nested Back-2-Back Within Another Interrupt	YES	YES
RA	i2054 — RA: Reads from GCFG Region Can Cause Spurious RAM ECC Errors	YES	YES
	i2095 — RA: Peek to Tail Returns Wrong Data	YES	YES
RAT	i2062 — RAT: Error Interrupt Triggered Even When Error Logging Disable Is Set	YES	YES
SA2_UL	i2098 — SA2_UL: Auth/Decrypt Operations with 2nd Input Thread Does Not Send the DMA Packet Out	YES	YES
STOG	i2121 — STOG: Flushing Gasket while there is a write transaction in flight can result in dropped write responses	YES	NO
	i2122 — STOG: Flushing Gasket concurrently with Gasket receiving a write response can cause indefinite non-idleness	YES	YES
	i2124 — STOG: Read command timeout can result in a gasket hang	YES	NO
	i2126 — STOG: Error miscounting when there are two concurrent timeouts or two concurrent unexpected responses	YES	YES
UART	i2096 — UART: Spurious UART Interrupts When Using DMA	YES	YES
UDMAP	i2055 — UDMAP: Packet Mode Descriptor Address Space Select Field Restrictions	YES	YES
	i2143 — UDMAP: TX Channel SA2UL teardown issue	YES	YES
	i2163 — UDMAP: UDMA transfers with ICNTs and/or src/dst addr NOT aligned to 64B fail when used in "event trigger" mode	YES	YES
	i2168 — UDMAP: Spurious ECC errors due to MAIN/MCU NAVSS rofifo_wr_byten issue	YES	YES
USB	i2050 — USB: Endpoint OUT Data Queue is Locked Up Due to a Data Packet for an Endpoint that Does Not Have Associated TRB	YES	YES

Table 1-2. Advisories by Modules (continued)

MODULE	ADVISORY	SILICON REVISIONS AFFECTED	
		SR 1.0	SR 1.1
	i2067 — USB: Race Condition while Reading TRB from System Memory in Device Mode	YES	YES
	i2092 — USB: Invalid Termination of DMA Transfer for Endpoint Following Isochronous Endpoint in SuperSpeed Device Mode	YES	YES
	i2093 — USB: DMA Hangs if USB Reset is Received During DMA Transfer in Device Mode	YES	YES
	i2134 — USB: 2.0 Compliance Receive Sensitivity Test Limitation	YES	YES
VPAC	i2188 — VPAC, DMPAC: UTC ECC writeback on queue memory can cause TR corruption	YES	YES
VTM	i2053 — VTM: Software Reads from On-Die Temperature Sensors Can Be Corrupted	YES	YES
	i2128 — VTM: VTM Temperature Monitors (TEMPSENSORs) Should Use a Software Trimming Method	YES	NO
	i2145 — VTM: Enabled interrupt event status registers incorrectly return raw unmasked values	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, DRA829). 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 DRA829 and TDA4VM devices, see the specific-device Datasheet (SPRSP35 and SPRSP36).

2.2 Devices Supported

This document supports the following devices:

- DRA829
- TDA4VM

Reference documents for the supported devices are:

- J721E DRA829/TDA4VM Processors Technical Reference Manual (SPRUIJ7)
- Jacinto™ DRA829 Automotive Processors Datasheet (SPRSP35)
- TDA4VM Jacinto™ Automotive Processors for ADAS and Autonomous Vehicles Datasheet (SPRSP36)

2.3 Package Symbolization and Revision Identification

Figure 2-1 shows an example of package symbolization.

Table 2-1 lists the device revision codes.

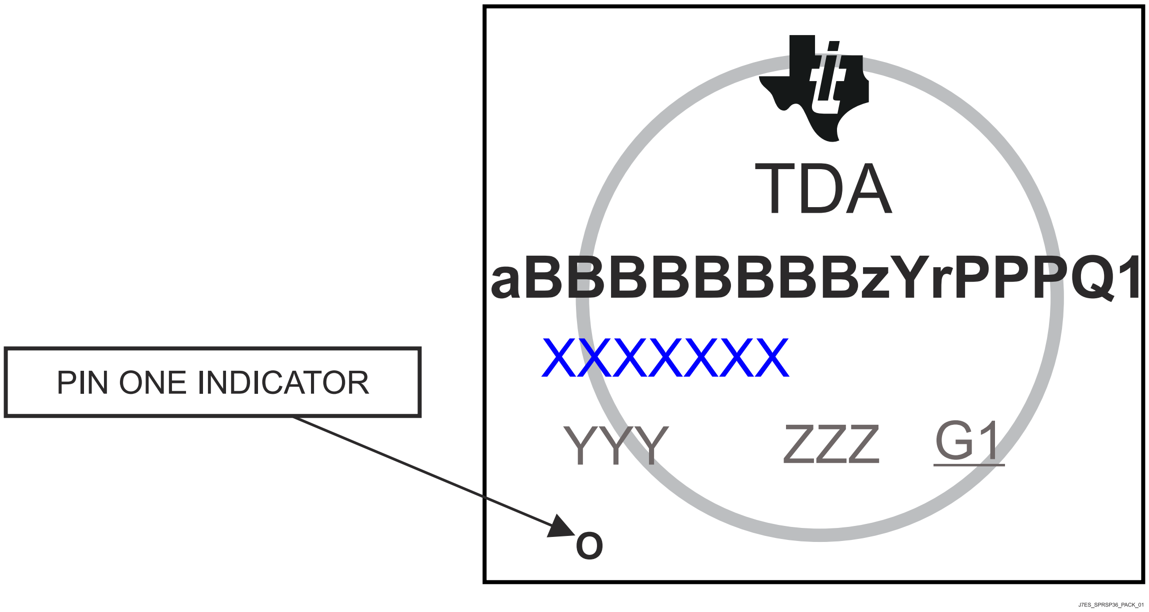


Figure 2-1. Package Symbolization

Table 2-1. Revision Identification

DEVICE REVISION CODE	SILICON REVISION	COMMENTS
A or BLANK	1.0	
B	1.1	

3 Silicon Revision 1.1/1.0 Usage Notes and Advisories

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

3.1 Silicon Revision 1.1/1.0 Usage Notes

No known usage notes for this silicon revision.

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

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

The issue was originally found while performing these tests using automation software, which increased USB signal amplitude while sending packets. The software was sweeping the amplitude from a value less than 100 mV to a value greater than 150 mV while verifying the device under test (DUT) NAK'd all packets below 100 mV and NAK'd no packets above 150 mV. However, increasing the amplitude through the squelch threshold while sending valid packets may lock the PHY as described in Advisory i2091.

Workaround(s): It may be necessary to perform the receive sensitivity test manually by breaking it into two parts. The first part begins the same as described above, with the initial amplitude set to a value less than 100 mV, verify the DUT NAK'd all packets while increasing the amplitude until it reaches 100 mV. The other part of the test begins by setting the amplitude above 150 mV, verify the DUT NAK'd no packets while decreasing the amplitude until it reaches 150 mV. This confirms the squelch threshold lies between 100 mV and 150 mV as required by the USB specification without sweeping the amplitude through the squelch threshold which can lock the PHY.

3.2 Silicon Revision 1.1/1.0 Advisories

i2024 ***MMC/SD Peripherals Do Not Support HS400***

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

Workaround(s): None.

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

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 4k bytes, 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 (maximum size is typically set to 512) the ROM will not find the boot file.

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

i2048 ***FSS: MCU_FSS0_WRT_TYPE Register is Logging Incorrectly***

Details: When programming the flash using the block method with embedded ECC, the FSS errors on any transaction that is not a full 32-byte block quanta.

The write error reporting stack is incorrectly connected to the ECC error stack.

MCU_FSS0_WRT_TYPE[12] WRT_ERR_ADR = top of ECC error stack (DED bit)

MCU_FSS0_WRT_TYPE[13] WRT_ERR_BEN = top of ECC error stack (SEC bit)

MCU_FSS0_WRT_TYPE[11-0] WRT_ERR_ROUTEID = top of ECC error stack

({MCU_FSS0_ECC_BLOCK_ADR[7-0] ECC_ERROR_BLOCK_ADDR, MCU_FSS0_ECC_TYPE[5] ECC_ERR_ADR, MCU_FSS0_ECC_TYPE[4] ECC_ERR_MAC, MCU_FSS0_ECC_TYPE[3] ECC_ERR_DA1, MCU_FSS0_ECC_TYPE MCU_FSS0_ECC_TYPE [2] ECC_ERR_DA0})

If any of the ECC error events is already processed, WRT_ERR_ADR, WRT_ERR_BEN, and WRT_ERR_ROUTEID bit fields of MCU_FSS0_WRT_TYPE register are zero.

Workaround(s): None.

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

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

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

Workaround(s): General Note:

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

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

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

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

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

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

i2050 *USB: Endpoint OUT Data Queue is Locked Up Due to a Data Packet for an Endpoint that Does Not Have Associated TRB*

Details: The USB device controller stores the endpoint OUT data received on the USB bus into a queue data structure. It transfers the data from the queue to system memory if a Transfer Request Block (TRB) is available for the endpoint that owns the data. However, if a TRB is not available for this endpoint, the data stays in the queue and blocks the subsequent data in the queue from being transferred to system memory. This occurs even if the endpoints owning the subsequent data have TRBs available.

One known use case which could be affected by this issue is a composite device that combines ACM class with another class such as MSC. ACM class drivers are known to operate without TRBs for long durations. If the other class receives data after the ACM class, it could potentially be stuck in the queue until the ACM class driver provides its TRB. This issue could be observed in general with classes that do not provide a TRB in advance or may not provide a TRB for long durations after data is received.

Workaround(s): An IRQ[6] interrupt is generated by the Controller when data is received into the queue for an endpoint that does not have a TRB. The corresponding interrupt status bit for this register is called the TRBERR in EP_STS register. Software can potentially use this interrupt to provide a TRB for the blocking endpoint. If the data is not needed immediately, the software must configure the TRB such that the data is transferred from the queue to a temporary buffer in the system memory. The data in the system memory can be consumed at a later time.

i2052 *CSI: CSI-Rx to CSI-Tx Retransmit Path Is Unavailable*

Details: CSI_TX_IF module does not recognize blanking in input data. Given all sensors are expected to have blanking, the re-transmit path between CSI_RX_IF module and CSI_TX_IF module cannot be used.

Figure 3-1 shows CSI_RX_IF block diagram.

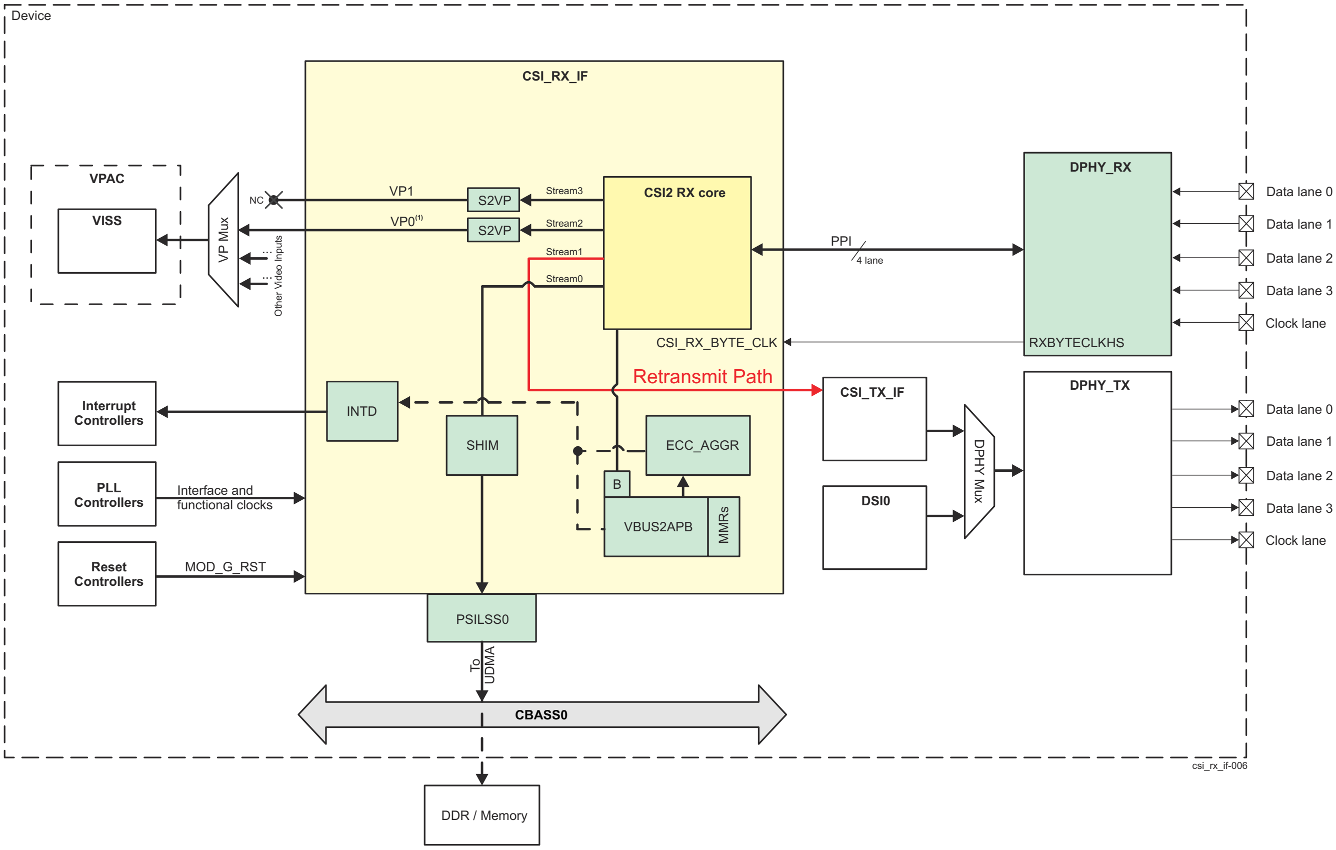


Figure 3-1. CSI_RX_IF Block Diagram

Workaround(s): Loop data through internal or external memory:

- Step 1: Send data from CSI_RX_IF capture to memory via UDMA-P.
- Step 2: Use CSI_TX_IF to read data via UDMA-P and output to external device.

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

Details:

The WKUP_VTM_TMPSENS_STAT_j[9-0] DATA_OUT 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 WKUP_VTM_TMPSENS_STAT_j[9-0] DATA_OUT 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 WKUP_VTM_TMPSENS_STAT_j[9-0] DATA_OUT;
        s1 = Read WKUP_VTM_TMPSENS_STAT_j[9-0] DATA_OUT;
        s2 = Read WKUP_VTM_TMPSENS_STAT_j[9-0] DATA_OUT;
        return get_best_value(s0,s1,s2);
}
```

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

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.

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

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

Workaround(s): If the RAT error logging is disabled, then the error interrupt should also be disabled by software.

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

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

Table 3-1 shows the actual behavior.

Table 3-1. Behavior of CPU Aliasing

CPU Aliasing ON					
	IBUFLA	**IBUFHA**	**IBUFLB**	**IBUFHB**	**L1D Action**
Owned	CPU	CPU	DMA	DMA	No issue
	DMA	DMA	CPU	CPU	No issue
	DMA	CPU	CPU	DMA	See [1]
	CPU	DMA	CPU	DMA	See [2]

[1] On a non-aligned acess to the last line in IBUFLA, where the line spills into IBUHA, both lines will get aliased.

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

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

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

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

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

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

Workaround(s): Avoid doing DMAs to buffers mapped to L1D SRAM.

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

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

Transactions:

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

Conditions/Stalls:

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

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

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

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

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

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

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

i2067 *USB: Race Condition while Reading TRB from System Memory in Device Mode*

Details:

The following sequence will ensure that stale data is not transferred:

1. Software needs to mark the initial TRB in TD as invalid (cycle bit points to software ownership).
2. Software needs to prepare all other TRBs in TD
3. Software will defer making initial TRB in TD as valid until DMA is done transferring all TRBs in existing TD. Software waits for IRQ[6] interrupt with TRBERR flag set before marking this TRB as valid (change cycle bit to indicate hardware ownership).

Workaround(s):

USB device controller uses 12-byte Transfer Request Block (TRB) data structures that are used to form a transfer ring in system memory. TRB contains the pointer to data buffer in memory that contains data to be transferred over USB or the location to store the data received over USB. Transfer ring management uses producer-consumer model where the software is the producer and Controller is the consumer. Ownership of a TRB is transferred between software and hardware using ‘Cycle’ bit field within the TRB. Software write of TRB into memory and hardware read of TRB from memory are expected to be atomic operations.

The issue arises because controller reads the TRB from system memory using two independent DMA transactions (8-byte transaction followed by a 4-byte transaction). As a result, TRB read operation by the controller is not atomic. If the software write to TRB occurs after hardware has read the first 8 bytes, this could lead to stale data transfer on IN transaction and stale data provided to software on OUT transaction.

The ‘Cycle’ bit is the least significant bit, which is read by the Controller in the second DMA transfer. Race condition exists because software write could be interleaved between the two read transactions. The following order of events could lead to corrupted data transfer on the USB bus:

1. Controller reads the 8 most significant bytes of the TRB. The data buffer pointer in this TRB may be old.
2. Software writes the 12-byte TRB in an atomic manner. This updates the data buffer pointer in the TRB to the new location.
3. Controller reads the remaining 4 bytes of the TRB. Since cycle bit was updated by software in previous step, Controller sees this as a valid TRB even though data buffer pointer is incorrect.

This issue only affects Device mode.

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

Details:

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

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

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

Workaround(s): Avoid doing DMAs to buffers mapped to L1D SRAM.

i2081 *Boot: ROM Code Maximum Timeout per Boot Mode Is Half the Original Timeout Value from TRM*

Details: The maximum timeout implemented in the ROM code per boot mode is half the original timeout value. For example, in UART boot mode, the maximum timeout is documented as 120 s, while on the device the actual timeout is 60 s.

This issue applies to all timeout values implemented by all boot modes.

This impacts the maximum image size downloaded by the ROM code in case of UART boot; currently it supports booting of images up to 300KB.

Workaround(s): None. For UART boot, the ROM code supports only booting of image up to 300KB size.

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.

i2085 *PCIe: Gen2 Capable Endpoint Devices Always Enumerate as Gen1*

Details: When a PCIe Gen2 capable End Point (EP) is connected to the SoC configured as a Root Port (RP), the RP fails to enumerate in Gen2 mode and always falls back to Gen1 mode even if autonomous speed change is enabled on both ends of the link.

Workaround(s): Once the link reaches L0 state, software can initiate link re-train by setting the PCIE_CORE_LM_I_LINKWIDTH_CONTROL_REG[31] EPLSCRL bit in the PCIe RP. This will force the RP to re-enumerate and achieve Gen2 speed.

i2086 *PCIe: MMA Unsupported Request (UR) or Configuration Request Retry Status (CRS) in Configuration Completion Response Packets Results in External Abort*

Details: When the PCIe Root Port (RP) is enumerating a PCIe multi-function End Point (EP) device or a PCIe switch, the EP may respond with an Unsupported Request (UR) or a Configuration Request Retry Status (CRS) to a configuration read from the RP. The UR response is returned when the RP tries to access a Bus Device Function (BDF) resource that is non-existent in the EP. This type of configuration access and the UR response is expected during the enumeration process. The UR and the CRS response from the EP results in a bus error in the PCIe RP which causes a data abort to the CPU.

Workaround(s): For multi-function EP devices that support Alternate Routing ID (ARI) capability, software can avoid scanning non-existent functions by using the ARI “Next Function” field, which points to the next physical function in the device. This will prevent UR response from the EP device during enumeration.

There is no feasible workaround identified for PCIe switch cards.

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

Details: Due to uninitialized internal state, the Matrix Math Accelerator (MMA) attached to the C71x may report errors in the FirstErrorCode and LastErrorCode fields of the

HWA_STATUS register after power-on. Because these fields are sticky, any subsequent HWARCV instruction may throw a C71x exception.

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

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

The sequence, in C71x assembly code is:

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

i2090 *MMCSD: MMCSD1 and MMCSD2 Speed Issue*

Details: MMC1/2 data read and write operations fail at SDR104 (200 MHz SDR) due to timing issue on the output DAT and CMD path. This causes erroneous data to be transmitted out in SDR104 mode, and limits proper MMC1/2 data read and write operations to 100 MHz clock frequency.

Workaround(s): Reduce clock frequency when performing data operations to 100 MHz for MMC1 and MMC2.

i2091 *USB: 2.0 PHY Hangs if Received Signal Amplitude Crosses Squelch Threshold Mmultiple Times Within the Same Packet*

Details: USB 2.0 PHY implements a squelch detection circuit on the receiver to ensure noise is not interpreted as valid data when the bus is idle. The squelch circuit blocks invalid data by disabling the receiver output while the DP/DM differential signal amplitude is less than the squelch threshold.

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

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

i2092 *USB: Invalid Termination of DMA Transfer for Endpoint Following Isochronous Endpoint in SuperSpeed Device Mode*

Details: SuperSpeed Isochronous OUT transaction from host uses Last Packet Flag (LPF) field to indicate that the current packet is the last packet for this service interval. DMA uses this flag to stop handling transfer descriptor for this endpoint for the current service interval.

Hardware bug causes LPF to be incorrectly applied to the endpoints being serviced after the Isochronous endpoint that received LPF set. This causes invalid termination of transfer descriptor processing by the DMA for the subsequent endpoints. As a result, only one TRB will be processed in each transfer descriptor for these endpoints. Subsequent endpoints can be of any type including Control (EP0) or Bulk. This issue violates rules for handling transfer descriptor.

There is no workaround for this issue. As a result, Isochronous OUT endpoint for SuperSpeed device mode is not supported.

Workaround(s): None.

i2093 *USB: DMA Hangs if USB Reset is Received During DMA Transfer in Device Mode*

Details: USB Controller contains a DMA master port used for transferring data to and from the system memory. This DMA master port may not properly terminate the transfer if bus reset (2.0 reset or warm/hot reset for Superspeed) is received while the DMA transfer is active. This can lock up the DMA master.

USB 2.0 bus reset or SuperSpeed Warm/Hot reset may be issued by the USB host in response to abnormal operation by the USB device. USB host will first try to recover from the error condition (for example CRC error in data packet) through transaction retries before issuing reset. During this time, the USB data buffers in the controller will be temporarily unavailable for new DMA transfers. This gives opportunity to finish pending DMA transfer. The following aspects affect the probability of receiving reset while DMA is active:

- Time required for host to detect abnormal behavior of the device.
- Time required for host to retry transactions and time required for device to respond to retries.
- Time required for host to initiate reset signaling.
- Time required for device to finish pending transfers for all available on-chip buffers.

If the system latency is extremely high, the reset may be received while the DMA transfer is still in progress or pending.

This issue impacts all speeds, but the probability of occurrence is very low.

This issue only affects device mode.

Workaround(s): The following two options can be used to recover from this very unlikely scenario. USB subsystem reset is required to recover once a DMA lockup occurs.

Option1: Use other system level mechanism to detect lockup and reset USB subsystem. If DMA locks up after reset, host will not be able to enumerate the device. Host may retry resetting the device again, but subsequent retries will fail since bus reset is not sufficient to recover from DMA lockup. After unsuccessful enumeration, host software has to raise the issue to system level so that alternate mechanism can be used to convey lockup to device.

Option2: Following software workaround can be used to detect DMA lock up and reset the subsystem. The following is the workaround procedure.

1. Software checks if AXI is idle when the first descriptor missing interrupt occurs after bus reset. AXI status can be checked by reading AXI_IDLE MMR bit in DMA_AXI_CAP register. If AXI is idle, software proceeds to step 2. If AXI is not idle, software proceeds to step 6.
2. Initiate a dummy DMA transfer by following below steps.
   - Configure a dummy IN1 endpoint.
   - Prepare TRB and data packet for IN1. Enable Interrupt-on-Completion (IOC) for this transfer.
   - Ring doorbell for IN1
   - Start T2 timer where T2 < 50ms. Suggested T2 time is 40ms in order to allow sufficient time for DMA to finish dummy transfer if DMA is not hung
3. Wait for IOC interrupt. If IOC interrupt is received, proceed to step 3. If T2 timer elapses and IOC was not received, then proceed to step 6.

4. Check that IN1 data is correct by reading BUF_ADDR, BUF_DATA, and BUF_CTRL MMR in Controller register space. If data is correct, proceed to step 5. If data is incorrect, proceed to step 6.
5. DMA master is not hung if this step is reached. Software can proceed with further programming to service the SETUP packet. Workaround flow can be exited.
6. DMA master is hung if this step is reached. Software performs the following steps to recover USB subsystem from the hang:
   - Force device disconnect.
   - Start T1 timer where T1 >= 200ms. T1 needs to be greater than 200ms in order to allow host to recognize device disconnect. A higher T1 delay can be used if the system can tolerate more downtime. In the highly unlikely scenario that any USB transfers are still pending after this delay, resetting the subsystem can potentially lockup the system bus and may lead to full chip reset. Higher T1 delay provides more time for any pending system bus transfers to finish.
   - Software initiates a force reset of USB subsystem using respective LPSC.
   - Wait until T1 timer elapses.
   - Restart USB subsystem by following same steps performed after power on reset to setup USB in device mode.

i2094 *PCIe: End of Interrupt (EOI) Not Enabled for PCIe Legacy Interrupts*

Details:

A PCIe End Point (EP) can signal a legacy interrupt at the PCIe Root Port (RP) by issuing an ASSERT_INTx/DEASSERT_INTx message. The ASSERT_INTx message causes a level output signal at the boundary of the PCIe RP controller to go high and the DEASSERT_INTx message causes the same output signal to go low. This level output signal from the controller is converted to a pulse for signaling an interrupt to the SoC interrupt controller.

The EP can issue a single ASSERT_INTx message and maintain the level output of the RP controller high without issuing a DEASSERT_INTx message if there is pending work to be done. The End of Interrupt (EOI) feature in the interrupt logic is used to re-trigger the pulse interrupt to the SoC interrupt controller from a level signal that remains asserted. The EOI feature has not been enabled for the PCIe legacy interrupts. This will result in only a single pulse interrupt to be generated to the SoC interrupt controller even if the level output signal from the PCIe RP remains asserted high.

As a result of this issue, legacy interrupt in RP mode cannot be used if EPs attached to this RP cannot guarantee DEASSERT_INTx message for each interrupt event.

Workaround(s):

PCIe EP can use MSI/MSI-X to signal interrupts to the PCIe RP in lieu of legacy interrupts.

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

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 3-2.

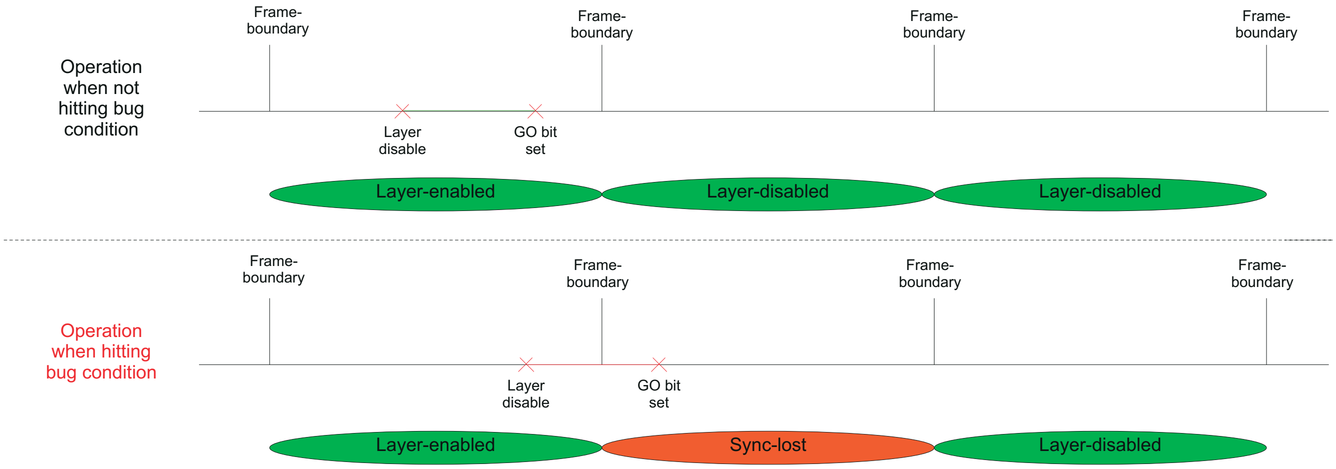
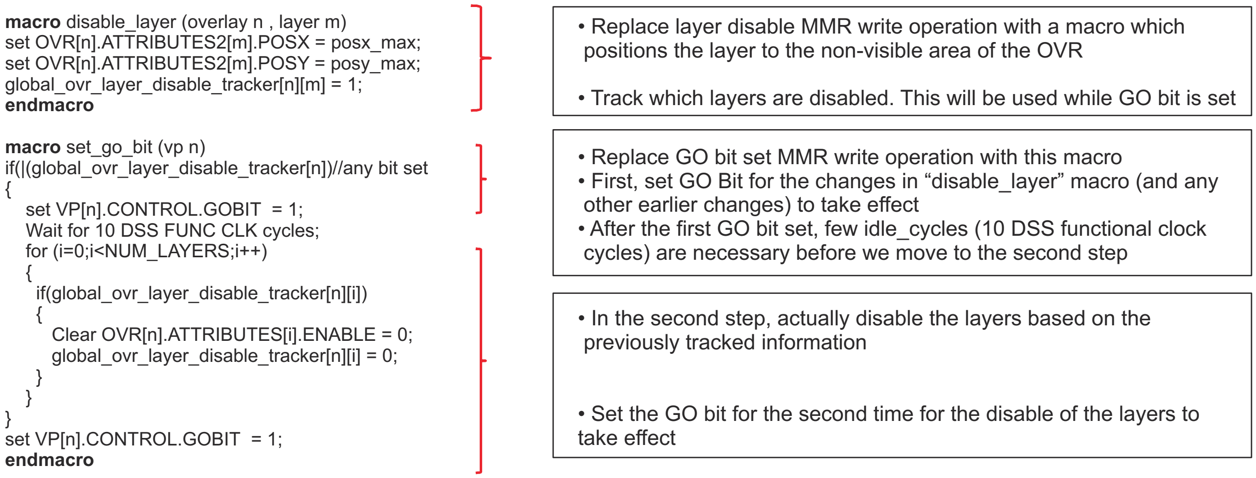


Figure 3-2. Bug Condition

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 3-3. In this case, the regular “disable layer” MMR write operation and “set GO bit set” MMR write operation are replaced with macros which implement the software workaround.



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

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

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

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

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

Figure 3-3. 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 *R5FSS: Deadlock Might Occur When One or More MPU Regions are Configured for Write Allocate Mode*

Details: There are two conditions where R5FSS 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). R5FSS 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, R5FSS may hang at one of the load instruction.

Workaround(s): Disabling linefill optimization inside R5FSS 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).

i2100 *PCIe: Endpoint Destination Select Attribute (ASEL) Based Routing Issue*

Details: The system DMA in a PCIe End Point (EP) can issue outbound PCIe requests with the destination select attribute (ASEL) set to a non-zero value. This will enable the PCIe controller to bypass the Address Translation Unit (ATU) and the address issued by the system DMA will be used as the outbound PCIe address.

The function number used in the outbound PCIe Transaction Level Packet (TLP) is incorrectly tied off to 0 when bypassing ATU. As a result, multi-function EP cannot use non-zero ASEL to bypass ATU. All multi-function EP transactions have to be translated by ATU to ensure correct function number in TLP.

If a multi-function EP issues non-zero ASEL transaction, it may result in an Unsupported Request (UR) at the PCIe Root Port (RP).

This issue does not affect single-function EP as function number is always zero.

Workaround(s): There is no workaround identified for supporting non-zero ASEL outbound transactions in a multi-function EP.

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.

Workaround(s): In the boot sequence, software needs to setup 5 dummy device IDs, which can be marked as reserved in TRM, and then sends 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 and send ITS request to first device ID, setup 7 dummy device IDs and send ITS request to first 4 device IDs.

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.

TI REFERENCES, APPROVERS MAXWELLAPPS-3499

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

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

This issue affects all Internal Diagnostics Module instances and their sub-banks.

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

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

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.

TI has identified that Micron’s Xccella OSPI flash is compatible with 1S-1S-8S mode. Cypress Semper Flash does not support 1S-1S-8S mode, and device part numbers should be chosen that explicitly support the 1b boot mode.

i2117 — *C71x: Register Corruption When MMA HWARCV is in Parallel With Load or Store With uTLB Miss*

Details: The C71x has the possibility of data corruption for certain circumstances on an HWARCV instruction. The symptom is that every 8th byte of the 64Bytes of data from the HWARCV data is corrupted with an arbitrary result (every 8th starting from the least significant byte of the vector). The value seen in the corrupting bytes is likely all zeros unless floating point instructions were executed previously on the .S2 unit.The condition under which this can occur has to do with three cases of resolving whether certain instructions that are in parallel with the HWARCV instruction must take extra time to determine whether the execution of that instruction will result in an exception, AND in the execute packet after this HWARCV instruction there is no instruction on the .S2 unit which will write a result into a 64 byte register.

Workaround(s): When adding the compiler switch --silicon_errata_i2117 to the command line of the C71x compiler, the compiler automatically ensures that there is an instruction on the .S2 unit that writes to a 64 byte vector register following all HWARCV .S2 instructions. This action ensures that one of the required conditions for encountering this issue is not met. If there is no useful work to be performed, the compiler inserts a dummy instruction which writes to an unused register, and is effectively a NOP instruction.

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 *HyperBus: HyperBus is Not Functional*

Details: Due to an internal timing violation, the HyperBus™ interface is not functional.

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

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

Details: The C71x Streaming Engine's (SE) pipeline for returning formatted data and return report internal error information is always monitoring the tags for the data that it is working on. When an error is detected for a line of data used to format data back to the CPU, all fetching side execution for queuing up commands to go to UMC, uTLB, and the formatting pipeline back to CPU is halted.

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

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

Workaround(s): The only 100% workaround is to not use stream templates with both LEZR and transposed mode enabled.

i2121 *STOG: Flushing Gasket while there is a write transaction in flight can result in dropped write responses*

Details:

Flushing the Slave Timeout Gasket while there is a write transaction in-flight can result in some auto-generated write responses (by the gasket) to not be generated. The in-flight write transaction must be accepted by the gasket, but not yet progressed fully through the gasket to the destination side interface. The end result is that a Master IP may be waiting on write responses which are dropped and never returned, thus hanging the Master IP. The gasket's internal scoreboard is also corrupted as a result.

By flushing the gasket, Master IPs that send write transactions through the gasket can hang, breaking the FFI System Solution. The gasket never reaches the idle state and thus cannot be clock stopped.

Workaround(s): STOG should be left in disabled/bypass mode.

i2122 — *STOG: Flushing Gasket concurrently with Gasket receiving a write response can cause indefinite non-idleness*

Details:

Flushing the gasket around the time a write response is also received by the gasket can result in the gasket's internal scoreboard getting corrupted. The corruption can prevent in the gasket from ever returning to the idle state.

The issue can also occur with a transaction timeout, but it is unlikely that a response would return after a sufficiently large timeout period.

Workaround(s):

The software should only flush the gasket in response to a timeout occurrence/interrupt. The gasket should not be arbitrarily flushed.

i2124 — *STOG: Read command timeout can result in a gasket hang.*

Details: When there is a read command timeout on the destination side interface (i.e. the interface is hung) and there are write transactions already outstanding, the Slave Timeout Gasket hangs and some auto-generated read/write responses may be dropped, thus hanging any Master IPs waiting on those responses.

Workaround(s): STOG should be left in disabled/bypass mode.

i2126 *STOG: Error miscounting when there are two concurrent timeouts or two concurrent unexpected responses*

Details: When there is a read command and write command that timeout in the same cycle, the timeout counter will only increment by 1 instead of 2 in this situation. Likewise, if an unexpected read response and an unexpected write response both arrive in the same cycle, the unexpected response counter will only increment by 1 instead of 2.

Workaround(s): The error counters are primarily supplemental information for software debug. Only one timeout error command/transaction info is recorded. The counters saturate at a count of 3, so the software should primarily focus on the error counter value being non-zero vs the exact counter value. The same approach should be applied to the unexpected response counter. Note: unexpected responses are dropped by the flush gasket.

i2128 *VTM: VTM Temperature Monitors (TEMPSENSORs) Should Use a Software Trimming Method*

Details: The VTM Temperature Monitors (TEMPSENSORs) are trimmed during production, with the resulting values stored in software-readable registers. Software should use these register values when translating the Temperature Monitor output codes to temperature values.

Workaround(s): A software trimming procedure should be applied when reading the Temperature Monitors. The spare registers listed below are written on each device during production for software to use in the trimming procedure. Workaround software provided by Texas Instruments (PROCESSOR-SDK RTOS 7.00.00 or later) should be used.

J7ES WKUP_CTRL_MMR base address

The J7ES MMRs that we are using for the PVT workaround are inside the WKUP_CTRL_MMR0 module address space

Here is the base address for that module in J7ES:

```
Ij7_wkup_ctrl_mmr.wkup_0.vbusp.cfg0 = 0x43000000; // size: 0x20000,
```

As a reference here is the base address for VTM in J7ES:

```
Ik3vtm_n16ffc.wkup_0.vbusp.mmr__vbusp__cfg1 = 0x42040000; // size: 0x400,
Ik3vtm_n16ffc.wkup_0.vbusp.mmr__vbusp__cfg2 = 0x42050000; // size: 0x400,
Ik3vtm_n16ffc.wkup_0.vbusp.__eccaggr_cfg__regs = 0x42810000; // size: 0x400,
```

The MMR field mapping of the compressed values for the 5 PVT sensors is described next.

PVT Workaround MMR Field Mapping

Table 3-2. WKUP_SPARE_FUSE0

Address Offset Proxy0, Proxy1	0x0000 0300, 0x0000 2300
Description	
Type	R/W

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0										

RESERVED	PVT4_RDN40C_B5	PVT4_RDN40C_B4	PVT4_RDN40C_B3	PVT4_RDN40C_B2	PVT4_RDN40C_B1	PVT4_RDN40C_B0	PVT3_RDN40C_B5	PVT3_RDN40C_B4	PVT3_RDN40C_B3	PVT3_RDN40C_B2	PVT3_RDN40C_B1	PVT3_RDN40C_B0	PVT2_RDN40C_B5	PVT2_RDN40C_B4	PVT2_RDN40C_B3	PVT2_RDN40C_B2	PVT2_RDN40C_B1	PVT2_RDN40C_B0	PVT1_RDN40C_B5	PVT1_RDN40C_B4	PVT1_RDN40C_B3	PVT1_RDN40C_B2	PVT1_RDN40C_B1	PVT1_RDN40C_B0	PVT0_RDN40C_B5	PVT0_RDN40C_B4	PVT0_RDN40C_B3	PVT0_RDN40C_B2	PVT0_RDN40C_B1	PVT0_RDN40C_B0

Table 3-3. WKUP_SPARE_FUSE1

Address Offset Proxy0, Proxy1	0x0000 0304 , 0x0000 2304
Description	
Type	R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PVT4_RDN125C_B3	PVT3_RDN125C_B3	PVT3_RDN125C_B2	PVT3_RDN125C_B1	PVT3_RDN125C_B0	PVT2_RDN125C_B8	PVT2_RDN125C_B7	PVT2_RDN125C_B6	PVT2_RDN125C_B5	PVT2_RDN125C_B4	PVT2_RDN125C_B3	PVT2_RDN125C_B2	PVT2_RDN125C_B1	PVT2_RDN125C_B0	PVT1_RDN125C_B8	PVT1_RDN125C_B7	PVT1_RDN125C_B6	PVT1_RDN125C_B5	PVT1_RDN125C_B4	PVT1_RDN125C_B3	PVT1_RDN125C_B2	PVT1_RDN125C_B1	PVT1_RDN125C_B0	PVT0_RDN125C_B8	PVT0_RDN125C_B7	PVT0_RDN125C_B6	PVT0_RDN125C_B5	PVT0_RDN125C_B4	PVT0_RDN125C_B3	PVT0_RDN125C_B2	PVT0_RDN125C_B1	PVT0_RDN125C_B0

Table 3-4. WKUP_SPARE_FUSE2

Address Offset Proxy0, Proxy1	0x0000 0308 , 0x0000 2308
Description	
Type	R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PVT2_RDN30C_B2	PVT2_RDN30C_B1	PVT2_RDN30C_B0	PVT1_RDN30C_B7	PVT1_RDN30C_B6	PVT1_RDN30C_B5	PVT1_RDN30C_B4	PVT1_RDN30C_B3	PVT1_RDN30C_B2	PVT1_RDN30C_B1	PVT1_RDN30C_B0	PVT0_RDN30C_B7	PVT0_RDN30C_B6	PVT0_RDN30C_B5	PVT0_RDN30C_B4	PVT0_RDN30C_B3	PVT0_RDN30C_B2	PVT0_RDN30C_B1	PVT0_RDN30C_B0	PVT4_RDN125C_B8	PVT4_RDN125C_B7	PVT4_RDN125C_B6	PVT4_RDN125C_B5	PVT4_RDN125C_B4	PVT4_RDN125C_B3	PVT4_RDN125C_B2	PVT4_RDN125C_B1	PVT4_RDN125C_B0	PVT3_RDN125C_B8	PVT3_RDN125C_B7	PVT3_RDN125C_B6	PVT3_RDN125C_B5

Table 3-5. WKUP_SPARE_FUSE3

Address Offset Proxy0, Proxy1	0x0000 030C , 0x0000 230C
Description	

Table 3-5. WKUP_SPARE_FUSE3 (continued)

Type	R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PVT4_RD30C_B7	PVT4_RD30C_B6	PVT4_RD30C_B5	PVT4_RD30C_B4	PVT4_RD30C_B3	PVT4_RD30C_B2	PVT4_RD30C_B1	PVT4_RD30C_B0	PVT3_RD30C_B7	PVT3_RD30C_B6	PVT3_RD30C_B5	PVT3_RD30C_B4	PVT3_RD30C_B3	PVT3_RD30C_B2	PVT3_RD30C_B1	PVT3_RD30C_B0	PVT2_RD30C_B7	PVT2_RD30C_B6	PVT2_RD30C_B5	PVT2_RD30C_B4	PVT2_RD30C_B3

i2129 *R5FSS: 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.

i2131 *C71x: Memory System May Hang During L2 Writeback Invalidate Operation when L2 Scrubber is Enabled*

Details:

When the C71x L2 Scrubber is enabled, the memory system may hang indefinitely if the C71x CPU issues an L2 Writeback Invalidate command by writing a "1" to the L2WBINV register. This occurs as a result of an interaction between the L2 Scrubber mechanism and the Writeback Invalidate state machine logic if the L2 Scrubber happens to be active during the L2 Writeback Invalidate operation.

Note that other cores on this SoC are not affected, as this issue pertains to only the C71x L2 controller.

Workaround(s):

There are two ways to prevent this issue from occurring:

OPTION A: Software guarantees that L2WBINV will never be set in any context by any privilege level at any time during the operation of the C71x.

OPTION B: In the case that software may use L2WBINV functionality, or cannot guarantee that it will not be used, the C71x L2 Scrubber (which is enabled automatically out of reset) should be disabled by the programmer upon boot-up. This can be done by writing a 0 to the SCEN field (bit 0) of the L2EDCFG register inside the C71x. Once this bit is cleared, the scrubber will be disabled and remain disabled until the C71x memory system is reset.

L2 scrubber functionality is provided if the application expects the L2 memory to hold static data and/or code for longer (>24 hours) periods of time. In these cases, the L2 scrubber can be periodically enabled by the Secure Supervisor with the following:

1. Setting L2EDCFG.SCEN (bit 0) to 1
2. Setting L2EDCFG.BTDELAY (bits 31:16) to 1
3. Setting L2EDCFG.SCDELAY (bits 63:32) to 1

This will guarantee the L2 Scrubber will initiate a scrub of the entire L2 memory in under 1ms. Once complete, the L2Scrubber should be disabled before returning to normal thread execution and/or initiating any L2 Writeback Invalidate operations.

i2132 *R5FSS: 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 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.

i2133 *R5FSS: Lock-Step Mode of Operation is Not Functional*

Details: R5FSS cannot operate in lock-step mode due to a hardware issue with the VIM (Vectored Interrupt Manager) module inside it. If the VIM module is active in lock-step mode, it may go out of lock-step erroneously, causing the generation of spurious lock-step miscompare interrupts towards the Safety monitor in the SoC. Additionally, this can cause the second core in the R5FSS to lock-up completely. The first core will continue to execute and is unaffected by this issue.

Workaround(s): There is no workaround to enable lock-step operation. R5FSS can only operate in split mode.

i2134 *USB: 2.0 Compliance Receive Sensitivity Test Limitation*

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

The issue was originally found while performing these tests using automation software, which increased USB signal amplitude while sending packets. The software was sweeping the amplitude from a value less than 100 mV to a value greater than 150 mV while verifying the device under test (DUT) NAK'd all packets below 100 mV and NAK'd no packets above 150 mV. However, increasing the amplitude through the squelch threshold while sending valid packets may lock the PHY as described in Advisory i2091.

Workaround(s): It may be necessary to perform the receive sensitivity test manually by breaking it into two parts. The first part begins the same as described above, with the initial amplitude set to a value less than 100 mV, verify the DUT NAK'd all packets while increasing the amplitude until it reaches 100 mV. The other part of the test begins by setting the amplitude above 150 mV, verify the DUT NAK'd no packets while decreasing the amplitude until it reaches 150 mV. This confirms the squelch threshold lies between 100 mV and 150 mV as required by the USB specification without sweeping the amplitude through the squelch threshold which can lock the PHY.

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.

i2144 *eMMC: VIO Supply Sequencing*

Details:

Device power up sequencing typically enables higher voltage domains followed by lower voltage domains, unless otherwise specified by power sequence timing diagrams. The power down sequence follows the reverse order for disabling voltage domains. During device power sequencing, IO signals are held in a safe state whenever core logic is not energized, and enabled only after core logic is operational. Enabling IO signaling whenever core logic is not operational may cause functional and reliability issues due to unintended current paths. An eMMC memory interfaces to the device’s MCC0 8-bit data and control signals that are referenced to a VDDS_MMC0 digital voltage domain supplied by a 1.8-V power resource. The MMC0 interface signals are not held in a safe state when core logic is not energized.

This issue has not resulted in any known system issues or failures.

If an eMMC memory component is not connected to the device, the original power sequencing that enables all 1.8-V domains before 0.8-V core domain (VDD_CORE) during power up and disables 1.8-V after 0.8-V core domain during power down can still be applied because the MMC0 signaling interfaces are not used in this type of system. Grouping VDDS_MMC0 into a common power rail with other digital 1.8-V domains and supplying from a common power resource by a VDD_IO_1V8 power rail is a valid power distribution (PDN) scheme for these systems.

Workaround(s):

If eMMC memory is used in a system connected to the device, use the following hardware changes for new board designs:

1. A new power sequence that shifts a VDDS_MMC0 power up ramp to occur after a VDD_CORE, and a VDDS_MMC0 power down ramp to occur before a VDD_CORE to avoid potential long term functional and reliability issues.
2. A new power resource (i.e. low cost, LDO, TLV73318P-Q1) and power rail (VDD_MMC0_1V8) for the PDN to allow power sequencing to be different than the common VDD_IO_1V8 power rail.
3. A PMIC PN that generates a new EN_MMC0_LDO control signal to synchronize and shift power sequencing. PMIC PN varies depending upon the PDN scheme used, as listed below:
   1. Legacy Pre-Silicon Dual Leo PDN (not recommended for new designs) would use new PN: TPS659411FXRWERQ1 (X = new NVM ID, yet to be defined) instead of orig PN: TPS659411F0RWERQ1
   2. Peak-Modified Dual Leo PDN is a recommended PDN for new designs enabling independent MCU and Main domains that support new a control signal.

c. Leo + Hera PDN is a recommended PDN for new designs combining MCU and Main domains that support a new control signal.

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.

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.

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.

i2150 — *I3C: SDAPULLEN drives low instead of Hi-Z*

Details:

The SDAPULLEN pin incorrectly drives low instead of Hi-Z when the I3C interface is operating in push-pull mode. This erroneously pulls the SDA pin low via the external active strong pull resistor between SDAPULLEN and SDA.

Workaround(s):

An external circuit should be used to connect an active strong pull up resistor from SDA to VDD when SDAPULLEN is high and to disconnect the pull-up resistor from SDA when SDAPULLEN is low.

Figure 3-4 shows an example implementation of the Workaround.

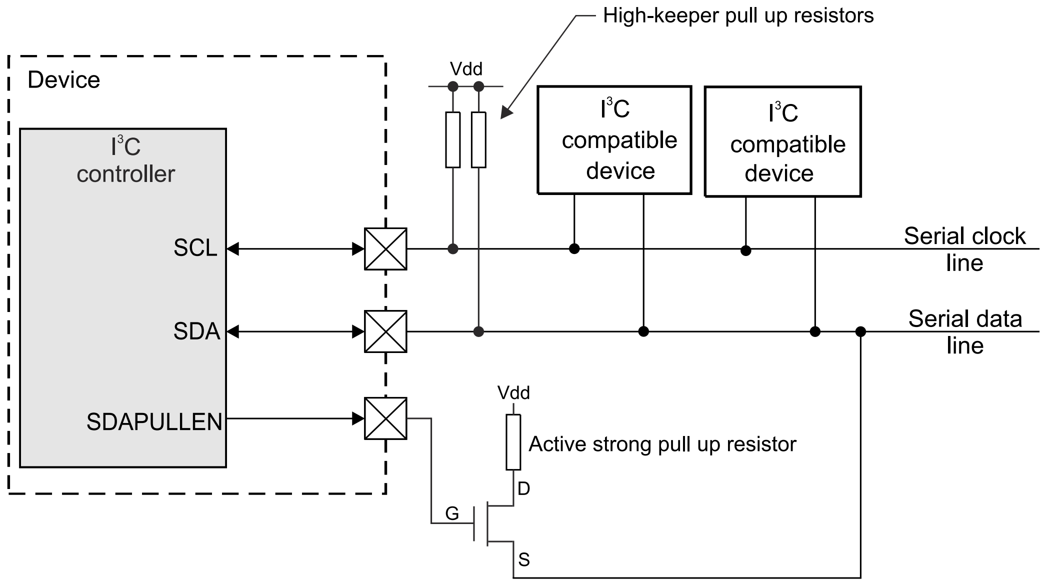


Figure 3-4. I3C SDAPULLEN Block Diagram

i2153 — *PCIe: Incorrect Reserved Bit Handling in TS1 Packet*

Details:

As per PCIe specification, reserved bits in TS1 and TS2 packets should be set to 0 by transmitter and ignored by receiver. However, PCIe controller invalidates TS1 packets if reserved bit 6 in symbol 7 is received as 1 for Gen3/Gen4 operation.

This issue only occurs if both the following conditions are true in addition to symbol 7 bit 6 being set:

- Symbol 7 bits 5 through 0 is equal to 0x5 or 0xA.
- Symbol 6 bits 6 through 0 is equal to 0x45 or 0x4A.

This issue may affect compliance if PCI-SIG adds a test to check for reserved bit handling in future. This is not expected to cause link training issues because transmitters are expected to set reserved bit to 0 and symbol 7 bit 6 is still reserved in 5.0 specification including Gen5 operation.

Workaround(s):

None.

i2155 — *DDR: Controller DDRSS_CTL_194[9-8] BIST_RESULT Status is Unreliable*

Details:

The DDR controller has a built-in self-test (BIST) feature that can be used to test the DDR interface to external DRAM. Upon completion of the BIST, the controller automatically clears DDRSS_CTL_194[9-8] BIST_RESULT to 0, instead of waiting for the user to first clear DDRSS_CTL_194[0] BIST_GO to 0. This could result in a false negative being reported, that is, BIST test actually passed but DDRSS_CTL_194[9-8] BIST_RESULT indicates it failed.

Workaround(s): Software workaround to correctly report the status of BIST.

1. Read the following BIST status fields before triggering the test.

DDRSS_CTL_310[31-0] BIST_FAIL_ADDR_0

DDRSS_CTL_311[2-0] BIST_FAIL_ADDR_1

DDRSS_CTL_306[31-0] BIST_FAIL_DATA_0

DDRSS_CTL_307[31-0] BIST_FAIL_DATA_1

DDRSS_CTL_308[31-0] BIST_FAIL_DATA_2

DDRSS_CTL_309[31-0] BIST_FAIL_DATA_3

DDRSS_CTL_302[31-0] BIST_EXP_DATA_0

DDRSS_CTL_303[31-0] BIST_EXP_DATA_1

DDRSS_CTL_304[31-0] BIST_EXP_DATA_2

DDRSS_CTL_305[31-0] BIST_EXP_DATA_3

DDRSS_CTL_206[11-0] BIST_ERR_COUNT (only valid if

DDRSS_CTL_200[2-0] BIST_TEST_MODE = 1, 2, 3 or 4)

2. Program desired BIST control fields and trigger BIST by setting the DDRSS_CTL_194[0] BIST_GO = 1

3. Poll for the BIST completed interrupt, indicated by DDRSS_CTL_293[11] INT_STATUS_0 bit.

4. Re-read BIST status fields listed in step (1).

If the values are different from step (1) then BIST has failed.

If the values are the same as step (1) then BIST has passed.

i2158 *DDR: Controller Hangs if Data Traffic is Sent to DRAM After BIST Execution*

Details: The DDR controller has a Built In Self Test (BIST) feature that can be used to test the DDR interface to external DRAM. After completion of BIST, the controller will not process/ execute any commands to the DRAM until it is reset.

Workaround(s): The controller must be reset after completion of BIST and before using the DDR interface to access the DRAM address space.

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

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

Workaround(s): Set the following fields to known valid working values before enabling CBT.

For frequency set 0: DDRSS_PI_199[6-0] PI_CALVL_VREF_INITIAL_START_POINT_F0 and DDRSS_PI_199[14-8] PI_CALVL_VREF_INITIAL_STOP_POINT_F0 bit fields.

For frequency set 1: DDRSS_PI_199[22-16] PI_CALVL_VREF_INITIAL_START_POINT_F1 and DDRSS_PI_199[30-24] PI_CALVL_VREF_INITIAL_STOP_POINT_F1 bit fields.

For frequency set 2: DDRSS_PI_200[6-0] PI_CALVL_VREF_INITIAL_START_POINT_F2 and DDRSS_PI_200[14-8] PI_CALVL_VREF_INITIAL_STOP_POINT_F2 bit fields.

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

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

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

Workaround(s): There is no work-around for this issue. The user should avoid accessing VIM registers while debugging.

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.

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

Details:

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 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.

i2168 *UDMAP: Spurious ECC errors due to MAIN/MCU NAVSS rofifo_wr_byten issue*

Details: Packet Starvation can cause a spurious ECC error. If a packet is received and there is no current descriptor to send the packet, the UDMAP sends out a single byte memory read to a predefined memory address to allow for update scoreboards. The received read data will update the buffer memory for the channel without updating the ECC signature stored in the channel FIFO memory. When the channel FIFO does the read to reclaim the buffer, an ECC error is generated by the hardware.

Workaround(s): If all the flows set rx_error_handling mode to 1 in the RX Flow Configuration Register, this will disable the dummy read as the logic now waits for a descriptor instead of generating an error. If it is required that error handling mode is 0 and packet drops are reported, then the software must clear the ECC error after receiving a dropped packet count increment.

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.

i2174 *DPHY: Reset sequence issue can lead to undefined module behavior*

Details:

The DPHY RX module utilizes four different resets: CSI_RX_RST (hardware controlled), common module reset (RSTB_CMN, hardware controlled), data lane reset (CSI_RX_IF_VBUS2APB_DPHY_LANE_CONTROL[15:12] DLx_RESET), and clock lane reset (CSI_RX_IF_VBUS2APB_DPHY_LANE_CONTROL [16] CL_RESET). The module expects these resets to be released in a specific order that can potentially be violated due to RSTB_CMN being internally tied to CSI_RX_RST. This can result in undefined behavior during software configuration and operation of the module.

Workaround(s):

None. Reset the DPHY RX module if issues are observed on the interface.

i2179 *CPSW9G: Reset isolation not working correctly*

Details:

Reset isolation for CPSW9G does not work correctly, causing two issues:

1. The SerDes clock muxes controlled by registers SERDESx_CLKSEL go into reset state.
2. CPSW9G will not be aware of the chip-level reset, and can potentially hang or have its state be corrupted depending on traffic going to/from Host.

Workaround(s):

CPSW9G reset isolation should not be enabled. It can be disabled by setting the following register fields:

PSC0_MDCTL_64[12].RESETISO = 0b0

PSC0_MDCTL_63[12].RESETISO = 0b0

CPSW_SS_SERDES_RESET_ISO_REG[7:0].SERDES_RESET_ISO = 0x00

i2180 *PRU-ICSSG: FDB table corruption during switch operation*

Details:

When the PRU-ICSSG is configured as a 1Gbps Ethernet switch, and the FDB is used, during an FDB lookup there is a one PRU clock cycle window during which the FDB can be corrupted if there is a broadside access by PRU0.

The FDB lookup can be initiated by either port or by host action. Each row within the FDB has 4 "buckets," or 32 Bytes, resulting in up to 4 buckets being corrupted at the given SA Hash index (depending on the PRU0 byte enables during the concurrent broadside access).

Broadside accesses by PRUs other than PRU0 have no affect.

Workaround(s):

A workaround within firmware to avoid PRU0 broadside access during the FDB lookup is possible but complex and not planned.

i2188 *VPAC, DMPAC: UTC ECC writeback on queue memory can cause TR corruption*

Details:

For the following queue buffers in VPAC and DMPAC UTC, UTC may hang or experience data corruption when ECC error injection is enabled through software control to collect diagnostics.

Memory Name of affected queue buffers:

- dru_utc_vpac_tpram_dru_queue_buffer
- dru_utc_vpac_tpram_dru_queue_buffer2
- dru_utc_dmpac_tpram_dru_queue_buffer
- dru_utc_dmpac_tpram_dru_queue_buffer2

Workaround(s): Software must disable ECC injection on the aforementioned QUEUE memories to avoid causing UTC to hang.

Note: ECC check will still be functional during normal operation.

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.

i2191 *ECC_AGGR: Erroneous non-correctable parity error assertion for RAM80*

Details: There are a set of signals on a system bus that require level shifters between voltage domains within the SOC. When the main voltage domain is not powered active, the level shifters maintain a default value to the downstream logic in the MCU domain.

One of these level shifters is driving an inverted value in that situation.

If the ecc_aggregator checks are enabled for that input source (ram_ecc80) before the main domain is powered active, an incorrect "uncorrectable ecc parity" error assertion is generated in the MCU voltage domain and recorded in the error signaling module.

Workaround(s): Do not enable the impacted input source checking in the ECC aggregator until all voltage domains are in a functional state.

Use a source IP from the main domain to enable that particular source to ensure the value is not impacted by the inversion.

Before enabling in the ECC aggregator, the error interrupt must be cleared, as it will always occur in the conditions listed above.

For any situation where the main voltage domain will be disabled/low power state, the input source checking in the ECC aggregator must be disabled as part of the sequence.

i2197 *I3C: Slave mode is not supported*

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

Workaround(s): None. Only Master role on a single-master bus should be used.

Trademarks

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

Revision History

DATE	REVISION	NOTES
December 2020	*	Inital version

IMPORTANT NOTICE AND DISCLAIMER

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

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, 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 (www.ti.com/legal/termsofsale.html) 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.

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