Errata

Errata AM263Px Sitara™ Microcontroller Silicon Revision 1.0

ABSTRACT

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

Table of Contents

1 Usage Notes and Advisories Matrices

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

Table 1-1. Usage Notes Matrix

Module	DESCRIPTION	SILICON REVISIONS AFFECTED
		AM263Px
		1.0
CLOCKS	i2324 — No synchronizer present between GCM and GCD status signals	YES
CLOCKS	i2488 — CLOCKS : PLL Configuration for precise 50-50 Duty cycle clocks	YES

Table 1-2. Advisories Matrix

MODULE	DESCRIPTION	SILICON REVISIONS AFFECTED
		AM263Px
		1.0
CONTROLSS	i2352 — CONTROLSS-SDFM: Dynamically Changing Threshold Settings (LLT, HLT), Filter Type, or COSR Settings Will Trigger Spurious Comparator Events	YES
CONTROLSS	i2353 — CONTROLSS-SDFM: Dynamically Changing Data Filter Settings (Such as Filter Type or DOSR) Will Trigger Spurious Data Acknowledge Events	YES
CONTROLSS	i2354 — CONTROLSS-SDFM: Two Back-to-Back Writes to SDCPARMx Register Bit Fields CEVT1SEL, CEVT2SEL, and HZEN Within Three SD-Modulator Clock Cycles can Corrupt SDFM State Machine, Resulting in Spurious Comparator Events	YES
CONTROLSS	i2356 — CONTROLSS-ADC: Interrupts may Stop if INTxCONT (Continue-to-Interrupt Mode) is not Set	YES
CONTROLSS	i2357 — CONTROLSS-ePWM: An ePWM Glitch can Occur if a Trip Remains Active at the End of the Blanking Window	YES
CONTROLSS	i2358 — CONTROLSS-ePWM: Trip Events Will Not be Filtered by the Blanking Window for the First 3 Cycles After the Start of a Blanking	YES
CONTROLSS	i2359 — CONTROLSS-CMPSS:Prescaler counter behavior different from spec when DACSOURCE is made 0 or reconfigured as 1	YES
CONTROLSS	i2405 — CONTROLSS: Race condition OUTPUT_XBAR and PWM_XBAR resulting in event miss	YES
CPSW	i2345 — CPSW: Ethernet Packet corruption occurs if CPDMA fetches a packet which spans across memory banks	YES
CPSW	i2401 — CPSW: Host Timestamps Cause CPSW Port to Lock	YES
CPSW	i2438 — CPSW: Host to Ethernet Checksum Generation with VLAN ADD/Remove	YES
CPSW	i2439 — CPSW: Host to Ethernet Timestamp Accuracy Issue	YES
UART	i2310 — USART: Erroneous triggering of timeout interrupt	YES
UART	i2311 USART Spurious DMA Interrupts — USART: Spurious DMA Interrupts	YES
M4 ROM	i2403 — M4 ROM: SBL redundant boot image feature not supported on HSSE devices	YES
MBOX	i2404 — MBOX: Race condition in mailbox registers resulting in events miss	YES
ROM	i2426 — ROM does not support OSPI 8D boot mode for the flashes supporting extended opcode	YES
RAM SEC	i2427 — RAM SEC can cause Spurious RAM writes resulting in L2 & MBOX memory corruption	YES
DTHE	i2428 — AES in DTHE generates extra dma request for data_in at the end of GCM encrypt	YES
SOC CONTROL	i2394 — Race condition in interrupt and error aggregator capture registers resulting in events miss	YES
SOC CONTROL	i2392 — Race condition in mem-init capture registers resulting in events miss	YES
BUS SAFETY	i2393 — Granular error status not logged in BUS_SAFETY_ERR registers for the detected faults	YES
OSPI	i2383 — OSPI: 2-byte address is not supported in PHY DDR mode	YES
OSPI	i2351 — OSPI: Direct Access Controller (DAC) does not support Continuous Read mode with NAND Flash	YES
OSPI	i2189 — OSPI: Controller PHY Tuning Algorithm	YES
PBIST	i2374 — PBIST: PBIST fails if clock frequency of R5SS_CORE_CLK is not same as R5FSS_CLK_SELECTED frequency	YES
ICSS	i2433 — ICSS: Reading the 64-bit IEP timer does not have a lock MSW logic when LSW is read	YES
RESOLVER	i2486 — RESOLVER: Angle inaccuracy and Velocity ripple can occur due to offset issue in arctan lookup table	YES

Table 1-2. Advisories Matrix (continued)

MODULE	DESCRIPTION	SILICON REVISIONS AFFECTED
		AM263Px
		1.0
TMU	i2485 — TMU: TCM Memory Corruption on R5SS0_CORE1 and R5SS1_CORE1 when writing to TMU Registers	YES

2 Silicon Revision *1.0* Usage Notes and Advisories

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

2.1 Silicon Revision *1.0* Usage Notes

This section lists all the usage notes that are applicable to silicon revision *1.0* [and earlier silicon revisions].

i2324 ***No synchronizer present between GCM and GCD status signals***

Details:

There is no synchronizer in between GCM and GCD, so the clock configuration register reads may be incorrect momentarily.

Severity:

Minor

Workaround(s):

Poll for the status registers change until it reflects the programmed SRC_SEL and DIV values.

i2488 ***CLOCKS : PLL Cofiguration for presice 50-50 Duty cycle clocks***

Details:

The VCO within a PLL can generate output waveforms with varying duty cycles, which can not meet the requirement for a precise 50-50 duty cycle needed by system & peripherals.

Severity:

Minor

Workaround(s):

Configure the PLL to operate at twice (2x) the target frequency. Configure the clock divider (HSDIVIDER) register to divide the PLL output by 2.

2.2 Silicon Revision *1.0* Advisories

The following advisories are known design exceptions to functional specifications. Advisories are numbered in the order in which they were added to this document. Some advisory numbers may be removed in future revisions of this document because the design exception was fixed or documented in the device-specific data manual or technical reference manual. When items are deleted, the remaining advisory numbers are not re-sequenced.

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

Details:

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

Workaround(s):

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

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

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.

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

Details:

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

Workaround(s):

For CPU use-case.

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

For DMA use-case.

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

i2374 ***PBIST fails if clock frequency of R5SS_CORE_CLK is not same as R5FSS_CLK_SELECTED frequency***

Details

The R5SS memories receive the R5SS CPU clock “R5SS_CORE_CLK” which is derived from R5SS_CLOCK_SELECTED root clock using programmable divider. When R5SS memories are tested using PBIST controller, the PBIST controller receives R5SS_CLOCK_SELECTED root clock. PBIST operation fails if different frequencies are chosen for the two clocks.

Workaround

For PBIST to work with R5SS memories the frequency of both clocks need to be same. If application usage requires R5SS_CORE_CLK to be a divided frequency of R5SS_CLOCK_SELECTED, then during PBIST operation of R5SS memories, the

i2374 (continued) ***PBIST fails if clock frequency of R5SS_CORE_CLK is not same as R5FSS_CLK_SELECTED frequency***

application shall ensure the R5SS_CORE_CLK is configured to same frequency as R5SS_CLOCK_SELECTED.

i2311 ***USART Spurious DMA Interrupts***

Details:

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

Workaround(s):

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

i2345 ***CPSW: Ethernet Packet corruption occurs if CPDMA fetches a packet which spans across memory banks***

Details:

Each memory bank in SoC has a separate memory controller. Even though memory addresses are contiguous, each bank is a separate entity with a separate controller.

If a memory bank received a memory request say 32 bytes and address of memory request is 16 bytes before end of memory bank, the behavior of the memory controller will be:

When the memory controller encounters end of memory bank after 16 bytes it will wrap around and give 16 bytes from the start of the memory bank.

This results in the packet corruption.

Workaround(s):

Ensure from application side single ethernet packet does not span across memory banks.

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

Details:

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

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

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

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

Workaround(s):

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

i2352 *CONTROLSS-SDFM: Dynamically Changing Threshold Settings (LLT, HLT), Filter Type, or COSR Settings Will Trigger Spurious Comparator Events*

Details:

When SDFM comparator settings—such as filter type, lower/upper threshold, or comparator OSR (COSR) settings—are dynamically changed during run time, spurious comparator events will be triggered. The spurious comparator event will trigger a corresponding CPU interrupt, CLA task, ePWM X-BAR events, and GPIO output X-BAR events if configured appropriately.

Workaround(s):

When comparator settings need to be changed dynamically, follow the procedure below to ensure spurious comparator events do not generate a CPU interrupt, CLA event, or X-BAR events (ePWM X-BAR/GPIO output X-BAR events):

1. Disable the comparator filter.
2. Delay for at least a latency of the comparator filter + 3 SD-Cx clock cycles.
3. Change comparator filter settings such as filter type, COSR, or lower/upper threshold.
4. Delay for at least a latency of the comparator filter + 5 SD-Cx clock cycles.
5. Enable the comparator filter.

i2353 *CONTROLSS-SDFM: Dynamically Changing Data Filter Settings (Such as Filter Type or DOSR) Will Trigger Spurious Data Acknowledge Events*

Details:

When SDFM data settings—such as filter type or DOSR settings—are dynamically changed during run time, spurious data-filter-ready events will be triggered. The spurious data-ready event will trigger a corresponding CPU interrupt, CLA task, and DMA trigger if configured appropriately.

Workaround(s):

When SDFM data filter settings need to be changed dynamically, follow the procedure below to ensure spurious data-filter-ready events are not generated:

1. Disable the data filter.
2. Delay for at least a latency of the data filter + 3 SD-Cx clock cycles.
3. Change data filter settings such as filter type and DOSR.
4. Delay for at least a latency of the data filter + 5 SD-Cx clock cycles.
5. Enable the data filter.

i2354 *CONTROLSS-SDFM: Two Back-to-Back Writes to SDCPARMx Register Bit Fields CEVT1SEL, CEVT2SEL, and HZEN Within Three SD-Modulator Clock Cycles can Corrupt SDFM State Machine, Resulting in Spurious Comparator Events*

Details:

Back-to-back writes to SDCPARMx register bit fields CEVT1SEL, CEVT2SEL, and HZEN within three SD-modulator clock cycles can potentially corrupt the SDFM state machine, resulting in spurious comparator events, which can potentially trigger CPU interrupts, CLA tasks, ePWM XBAR events, and GPIO output X-BAR events if configured appropriately.

Workaround(s):

Avoid back-to-back writes within three SD-modulator clock cycles or have the SDCPARMx register bit fields configured in one register write.

i2356 *CONTROLSS-ADC: Interrupts may Stop if INTxCONT (Continue-to-Interrupt Mode) is not Set*

Details:

If ADCINTSELxNx[INTxCONT] = 0, then interrupts will stop when the ADCINTFLG is set and no additional ADC interrupts will occur. When an ADC interrupt occurs simultaneously with a software write of the ADCINTFLGCLR register, the ADCINTFLG will unexpectedly remain set, blocking future ADC interrupts.

Workaround(s):

1. Use Continue-to-Interrupt Mode to prevent the ADCINTFLG from blocking additional ADC interrupts:

```
ADCINTSEL1N2[INT1CONT] = 1;
ADCINTSEL1N2[INT2CONT] = 1;
ADCINTSEL3N4[INT3CONT] = 1;
ADCINTSEL3N4[INT4CONT] = 1;
```

2. Ensure there is always sufficient time to service the ADC ISR and clear the ADCINTFLG before the next ADC interrupt occurs to avoid this condition.
3. Check for an overflow condition in the ISR when clearing the ADCINTFLG. Check ADCINTOVF immediately after writing to ADCINTFLGCLR; if it is set, then write ADCINTFLGCLR a second time to ensure the ADCINTFLG is cleared. The ADCINTOVF register will be set, indicating an ADC conversion interrupt was lost.

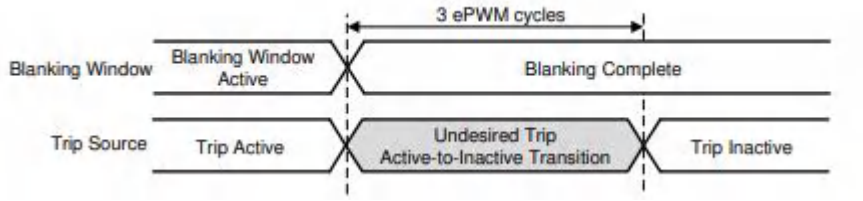
```
AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1; //clear INT1 flag
if(1 == AdcaRegs.ADCINTOVF.bit.ADCINT1) //ADCINT overflow
{
 AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1; //clear INT1 again
// If the ADCINTOVF condition will be ignored by the application
// then clear the flag here by writing 1 to ADCINTOVFCLR.
// If there is a ADCINTOVF handling routine, then either insert
// that code and clear the ADCINTOVF flag here or do not clear
// the ADCINTOVF here so the external routine will detect the
// condition.
// AdcaRegs.ADCINTOVFCLR.bit.ADCINT1 = 1; // clear OVF
```

i2357 *CONTROLSS-ePWM: An ePWM Glitch can Occur if a Trip Remains Active at the End of the Blanking Window*

Details:

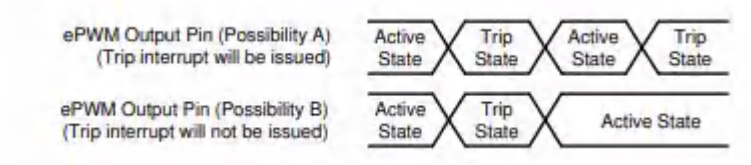
The blanking window is typically used to mask any PWM trip events during transitions which would be false trips to the system. If an ePWM trip event remains active for less than three ePWM clocks after the end of the blanking window cycles, there can be an undesired glitch at the ePWM output.

The following picture illustrates the time period which could result in an undesired ePWM output.



The following picture illustrates the two potential ePWM outputs possible if the trip event ends within 1 cycle before or 3 cycles after the blanking window closes.

i2357 (continued) ***CONTROLSS-ePWM: An ePWM Glitch can Occur if a Trip Remains Active at the End of the Blanking Window***



Workaround(s):

Avoid configuration of blanking window such that the trip input would fall in this range (1 cycle before and 3 cycles after the blanking window closure).

i2358 ***CONTROLSS-ePWM: Trip Events Will Not be Filtered by the Blanking Window for the First 3 Cycles After the Start of a Blanking***

Details:

The Blanking Window will not blank trip events for the first 3 cycles after the start of a Blanking Window. DCEVTFILT may continue to reflect changes in the DCxEVTy signals. If DCEVTFILT is enabled, this may impact subsequent subsystems that are configured (for example, the Trip Zone submodule, TZ interrupts, ADC SOC, or the PWM output).

Workaround(s):

Start the Blanking Window 3 cycles before blanking is required. If a Blanking Window is needed at a period boundary, start the Blanking Window 3 cycles before the beginning of the next period. This works because Blanking Windows persist across period boundaries.

i2359 ***CONTROLSS-CMPSS:Prescaler counter behavior different from spec when DACSOURCE is made 0 or reconfigured as 1***

Details:

While the prescaler is running, if we make DACSOURCE = 0 the prescale counter will not reset, if the enable condition is LOW the value stays, and when the DACSOURCE is again configured as 1 the counter starts from the previous value which was retained. This bug is present only when DACSOURCE is configured during the prescale counter running.

Workaround(s):

Issue a soft reset between DACSOURCE configuration which is not a dynamic configuration.

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

Details:

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

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

Workaround(s):

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

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

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

i2392 *Race condition in mem-init capture registers resulting in events miss*

Details:

Potential race condition in capture registers resulting in events getting lost while other events in the same register are being cleared by writing to the register. Following registers are impacted by this issue:

Workaround(s):

Any of the following Workarounds can be used:

Sequentially trigger the mem-init and clear the status before triggering the new mem-init. This is needed if both the status are in the same register.

(OR)

If parallel triggers are must then poll for the all status-bits that got triggered to be 1'b1 and then go and clear the DONE status register

(OR)

Check the MEM_INIT_STATUS register after starting the mem-init and wait the status to go -low by checking it in regular interval and finally clear the DONE status register when the status goes low

i2393 *Granular error status not logged in BUS_SAFETY_ERR registers for the detected faults*

Details:

Granular error status not logged correctly for detected faults in COMP_CHECK and COMP_ERR fields of MSS_CRTL:*_BUS_SAFETY_ERR registers.

The error signal err_comp and err_comp_signals are used to detect any faults on the diagnostic circuit. The AND'ed output of these two signals are used to report the fault. However they are sampled at different edges of the clocks resulting in loss of the error signal getting generated. and hence is not getting logged in the MSS_CTRL MMRs.

i2393 (continued) ***Granular error status not logged in BUS_SAFETY_ERR registers for the detected faults***

There are two possible scenario:

Case 1: Log registers have non-zero values

Here the granular logs are captured correctly and appropriate action can be taken for a given fault.

Case 2: Log Registers have all zeros

Here the granular logs are not captured correctly and possible entities affected are R5F and L2 memory.

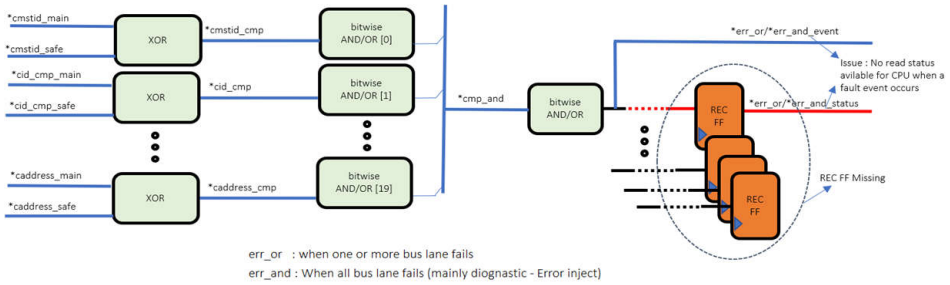


Figure 2-1.

Workaround(s):

No Workaround.

As Granular Error Status is not logged, the bus safety fault is detected only as an aggregated error event and the Granularity of diagnostics information will not be captured correctly.

Case 1: In the case where the logs are captured correctly the log results can be used to take appropriate action.

Case 2: In the case where the logs are not captured correctly then for a) diagnostics no action needed and b) in case of actual failure in the application WarmResetn should be used.

i2394 ***Race condition in interrupt and error aggregator capture registers resulting in events miss***

Details:

Potential race condition in capture registers resulting in events getting lost while other events in the same register are being cleared by writing to the register. Following registers are impacted by this issue:

MSS_CTRL: *INTAGG_STATUS_REG, *TPCC_ERR/INTAGG_STATUS_RAW

Workaround(s):

Follow below steps in ISR:

1) Before exiting the ISR read the *_ERRAGG_RAW and check the bit-validity by "anding" with *_ERRAGG_MASK.

2) If any bit is set that-implies there is a interrupt/Error which got missed while clearing the *_ERRAGG_STATUS.

3) Service the corresponding bit in ISR and then exit the ISR. So ISR should be exited after both STATUS and "RAW&MASK" are zero

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

Details:

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

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

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

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

Workaround(s):

Ethernet to host timestamps must be disabled.

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

i2403 *M4 ROM: SBL redundant boot image feature not supported on HSSE devices*

Details:

SBL redundant boot image feature not supported on HSSE devices

Any corruption on the primary image at the following locations , SBL boot fails to boot from redundant flash region

- Image corruption at middle of the certificate
- Image corruption at end of the certificate
- Image corruption at start of the sbl binary
- Image corruption at middle of the sbl binary
- Image corruption at End of the sbl binary

Workaround(s):

None.

i2404 *MBOX: Race condition in mailbox registers resulting in events miss*

Details:

Potential race condition in capture registers resulting in events getting lost while other events in the same register are being cleared by writing to the register. Following registers are impacted by this issue:

MSS_CTRL: *_MBOX_READ_REQ

MSS_CTRL: *_MBOX_READ_DONE

Workaround(s):

Read the status(READ DONE / READ_DONE_REQ) of the other processor to check any interrupt is in flight before setting up the trigger (WRITE DONE /READ ACK) event.

(OR)

i2404 (continued) ***MBOX: Race condition in mailbox registers resulting in events miss***

Re-trigger the (WRITE DONE /READ ACK) event if the status (READ DONE / READ_DONE_REQ) is not received within the given time.

i2405 ***CONTROLSS: Race condition OUTPUT_XBAR and PWM_XBAR resulting in event miss***

Details:

Potential race condition in capture registers resulting in events getting lost while other events in the same register are being cleared by writing to the register. Following registers are impacted by this issue:

C2K_PWMXBAR:PWMXBAR_STATUS

C2K_OUTPUTXBAR:OUTPUTXBAR_STATUS

Workaround(s):

WA -1 (For event widths > 50ns):

By default, level events (width >50ns) will be captured in “STATUS” Register, while performing “Clr Flag”, if any new event from hardware is asserted at the same time, it will be missed in FLAG Register, However, STATUS register does capture such events missed in FLAG register. After completing “Clr FLAG”, reading the “STATUS” register allows to capture/process any missed event based on “STATUS” read.

WA-1: ISR Sequence:

Read FLAG Event[x]

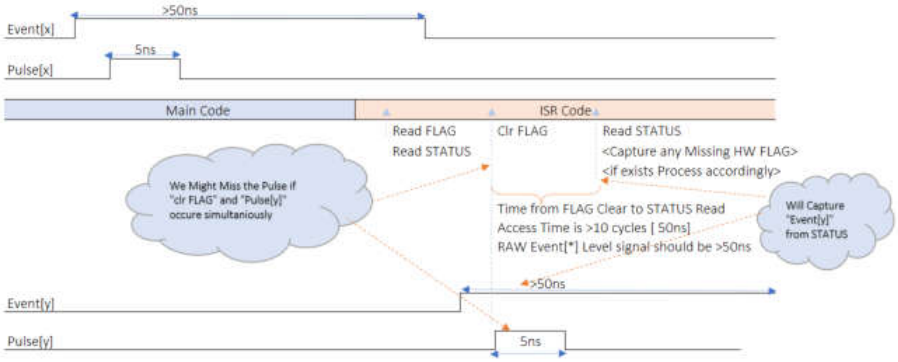
Read STATUS, All events

Clr FLAG, Event[x]

Read STATUS, All events

Capture any missing HW event FLAG

If exists, process accordingly



WA -2 (For any event widths):

Enable OUTPUTXBAR with the same events in the ISR and then “Clr PWMXBAR FLAG”.

Any missed Hardware event during the same window will be captured in OUTPUTXBAR FLAG”. Read the OUTPUTXBAR FLAG and process accordingly

“Clr OUTPUTXBAR FLAG” followed by disable of OUTPUTXBAR in the ISR.

i2405 (continued) ***CONTROLSS: Race condition OUTPUT_XBAR and PWM_XBAR resulting in event miss***

WA-2: ISR Sequence:

Read FLAG Event[x]

Read STATUS, All events

Enable OutPutxBAR

- Map Same Events

Clr FLAG, Event[x] PWMXBAR

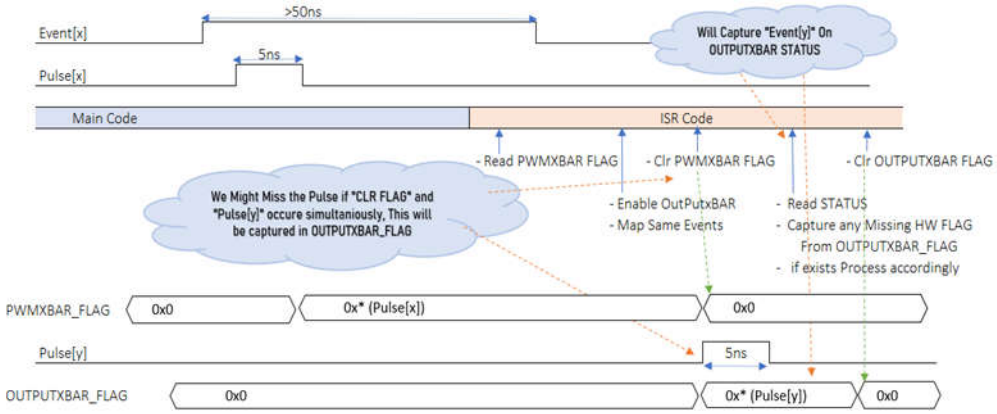
Read STATUS

- Capture any Missing HW event FLAG

from OUTPUTXBAR_FLAG

if exists Process accordingly

- Clr FLAG, Event[y] OutputXBAR



i2426 ***ROM does not support OSPI 8D boot mode for the flashes supporting extended opcode***

Details:

Boot ROM reads SFDP header from flash memory after switching to 8D mode for adjusting the read tap delays. A5h command is used to read the SFDP header. Flashes that support extended Opcode expect A5h followed by 5Ah or A5h based on inverse or repeat to be sent by ROM. This is not supported in ROM code.

This issue applies only for the flash memories that need extended opcode support in 8D boot mode. 1S and 8S boot modes are not impacted by this issue.

Workaround(s):

None. Use the Flash memories that do not require extended OPCode

i2427 ***RAM SEC can cause Spurious RAM writes resulting in L2 & MBOX memory corruption***

Details:

In case when a memory encounters a single-bit error during a RAM read data either due to a read or a partial write transaction, the RAM will enter a state which could lead to a later spurious write to the RAM if the next "memory read" is due to a subsequent

i2427 (continued) ***RAM SEC can cause Spurious RAM writes resulting in L2 & MBOX memory corruption***

partial write transaction. If the "memory read" is instead due to an actual memory read transaction, then the lingering bad internal state would be cleared and there wouldn't be any possibility of a later spurious write. The spurious write would be to the last memory address written prior to the partial write transaction which triggers the spurious write. The issue is only applicable to MBOX & L2.

Figure 2-2 lists possible scenarios where the issue is applicable (Example 1,2,3) and not applicable (Example 4,5,6) for more clarity. Transaction# are for illustration and doesn't necessarily represent the exact cycle each operation occurs. [SEC – Single bit Error Correction, DED – Double Bit Error Detection]

Ex #	Transaction 1	Transaction 2+N N=0,1,2,3..	Transaction 2+N+1	Transaction 2+N+2
1	Read or Partial Write Addr A (SEC) ← read with SEC	Full Write Addr X ← last write prior to partial write Note: N=0	Partial Write ← Triggers spurious write	Spurious write to Addr X with Transaction 1 corrected read data of Addr A
2	Read or Partial Write Addr A (SEC) ← read with SEC	Full Write Addr B Full Write Addr C Full Write Addr D ← last write prior to partial write Note: N=2	Partial Write ← Triggers spurious write	Spurious write to Addr D with Transaction 1 corrected read data of Addr A
3	Read or Partial Write Addr A (SEC)	Partial Write Addr B Note: N=0	Spurious write to Addr A with Transaction 1 corrected read data of Addr A (Addr A is overwritten with the RAM content prior to the Transaction 1 Partial write)	
4	Read Addr A (SEC)	Partial Write Addr B Note: N=0	No Spurious write to Addr A with Transaction 1 corrected read data of address A (no data corruption)	
5	Read or Partial Write Addr A (SEC)	Read ← Clears bad internal state Note: N=0	No spurious writes with all command combinations in subsequent cycles	
6	Read or Partial Write Addr A (SEC)	Full Write Addr B Note: N=0	Read ← Clears bad internal state	No spurious writes will all command combinations in subsequent cycles

Figure 2-2.

Workaround(s):

One of the below Options can be used as workaround.

Option 1:

Disable ECC, Applicable only for non-safety application.

Option 2:

Disallow Partial writes to the memory (only perform full line writes)

In case of L2, if the L2 space is cacheable the core will perform only full line writes and this issue is not applicable.

Option 3:

The application can treat all SEC errors like a DED (no correction only detection even in case of single bit error) since there is a possibility of RAM data corruption if application can't control the transactions immediately after a single bit error on a read or partial write transaction.

Note

Prior statements about using the ECC CTRL - SEC Counter as an indicator of normal SEC issue vs spurious write are NOT VALID. After a spurious write, the ECC CTRL SEC Counter can still be 1.

i2428 *AES in DTHE generates extra dma request for data_in at the end of GCM encrypt*

Details:

The AES Engine produces an additional dma request for data input at the end of GCM cipher mode of Encryption. This issue only applies to Encryption with AES-GCM mode and it does not apply to AES-GCM Decryption or any other block cipher modes (for example CBC).

The extra DMA request goes away (deasserts) by itself after few cycles without any data written to it.

Depending on how the DMA in the system is set up for AES-GCM mode , the extra DMA request at the end of a packet transfer may cause unintended data transfer on the next packet.

Workaround(s):

None

i2433 *ICSS: Reading the 64-bit IEP timer does not have a lock MSW logic when LSW is read*

Details:

IEPx 64-bit timestamp can be incorrect when lower 32-bit data is 0xFFFFFFFC or above (at 250MHz). In this case the upper 32-bit value is updated but lower value is the old number. The issue is seen when IEP counter (IEP_COUNT_REG1 : IEP_COUNT_REG0) is read back-to-back from ICSS PRU cores.

Example 1:

1st read : 0x000000D0(Upper):0xFFFFFFFC(lower)

2nd read : 0x000000D0(Upper):0x00000028(lower)

Example 2:

1st read : 0x000000D7(Upper):0xFFFFFFFC(lower)

2nd read : 0x000000D7(Upper):0x0000002C(lower)

Example 3:

1st read : 0x000000D6(Upper):0xFFFFFFF0(lower)

2nd read : 0x000000D7(Upper):0xFFFFFFFC((lower)

As shown above, this leads to timer increment behavior that is non-monotonic or timer differences to be unusually large as in Example 3 . This is due to 1 cycle race condition when loading 64-bit value from IEPx counter.

Workaround(s):

Note: these workarounds exist in SDK9.2 and later

Workaround in C for PRU:

```
uint64_t timestamp = (uint64_t) (0x2E0010);
```

/* Workaround starts here */

```
if ((timestamp & 0xFFFFFFFF) >= 0xFFFFFFFC)
{     timestamp = *(uint64_t*) (0x2E0010); }
```

/* Workaround ends here */

i2433 (continued) ***ICSS: Reading the 64-bit IEP timer does not have a lock MSW logic when LSW is read***

Workaround in assembly for PRU:

```
 ldi32 r4, 0xFFFFFFFC ; 0-4 for 250MHz clock
    ;load 64-bit timestamp to r2:r3
    lbco &r2, c26, 0x10, 8
    qbgt skip_iep_read_errata. r2, r4
    ;re-read IEP if IEP_COUNTER_LOW >= 0xFFFF_FFFC
    lbco &r2, c26, 0x10, 8
skip_iep_read_errata:
```

Workaround in C for R5F, A53:

```
uint64_t getIepTimeStamp64 (void)
{
    uint64_t u64Timestamp1 = (volatile uint64_t)(0x300AE010);
    uint64_t u64Timestamp2 = (volatile uint64_t)(0x300AE010);
    if (u64Timestamp2 > u64Timestamp1)
    {
#ifdef __DEBUG
        if (((u64Timestamp2 >> 32)-(u64Timestamp1 >> 32)) == 1)
        {
            /* HW errata fixed due to picking u64Timestamp1*/
            if ((u64Timestamp2 & 0xFFFFFFFF) >= (u64Timestamp1 & 0xFFFFFFFF))


{                 DebugP_log ("Errata fixed (1): %llx : %llx\r\n",
        u64Timestamp1, u64Timestamp2);             }

        }
#endif
        return u64Timestamp1;
    }
    else
    {
#ifdef __DEBUG
        if ((u64Timestamp2 & 0xFFFFFFFF) < (u64Timestamp1 & 0xFFFFFFFF))


{             /* Adjust the IEP MSW in the case running into HW errata
*/
    DebugP_log ("Errata fixed (2): %llx : %llx\r\n", u64Timestamp1,
u64Timestamp2);         }

#endif
        /* HW errata fixed due to picking u64Timestamp2*/
        return u64Timestamp2;
    }
}
```

i2438 ***CPSW: Host to Ethernet Checksum Generation with VLAN ADD/Remove***

Details:

When the CPSW host to ethernet checksum generation is enabled on HW and a VLAN tag is added or removed on Ethernet egress, a packet from host to ethernet is corrupted and sent as garbage with a GOOD CRC – which is not acceptable.

Workaround(s):

VLAN tags must not be added or removed on Ethernet egress for packets that have a generated checksum.

i2439 ***CPSW: Host to Ethernet Timestamp Accuracy Issue***

Details:

When a packet is sent from the Host to Ethernet with a timestamp to be generated on Ethernet egress, a packet length with 0xD5 in the lower 8-bits results in a timestamp error.

Using timestamp for PTP messages should not be impacted as the PTP messages are usually much shorter than 0xD5 packet length.

Workaround(s):

Ethernet timestamp should be enabled only for PTP messages on Host Tx.

i2485 ***[TMU] TCM Memory Corruption on R5SS0_CORE1 and R5SS1_CORE1 when writing to TMU Registers***

Details:

R5 access to internal TMU space are also accessing ATC Memory location.

CPU1 access to TMU1 memory map through TCM bus are also initiating accesses to ATCM1 Bank0 RAM (Impacted location are 0x40-0x280) in dual core mode of Cluster configuration.

WR TXN: CORE1 writes to TMU1 is corrupting ATCM1 Bank0 memories contents because valid signal to memory are not blocked

RD TXN: CORE1 reads to TMU1 are not corrupted because of ATC_WAIT being asserted which samples the correct read data from TMU even though ATCM memory accesses are made (No Impact)

Workaround(s):

Use one of the Workarounds:

WA1: Do not use initial 576 bytes (0x40-0x280) of ATCM from CPU1 allocation

WA2: Use CPU0 TMU alone for computation. Don’t use CPU1 TMU

i2486 ***[Resolver] Angle inaccuracy and Velocity ripple can occur due to offset issue in arctan lookup table***

Details:

RTL bug causes offset between half quadrants in arctan lookup table. The offset causes angle inaccuracy and velocity ripple.

Workaround(s):

Use Software to find Angle and Velocity resolution

3 Trademarks

All trademarks are the property of their respective owners.

4 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from May 31, 2024 to May 31, 2025 (from Revision A (May 2024) to Revision B (May 2025)) **Page**

- Added Advisory i2485; [TMU] TCM Memory Corruption on R5SS0_CORE1 and R5SS1_CORE1 when writing to TMU Registers.....18
- Added Advisory i2486 ; [Resolver] Angle inaccuracy and Velocity ripple can occur due to offset issue in arctan lookup table..... 18

IMPORTANT NOTICE AND DISCLAIMER

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

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

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

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

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

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