

## Mask Set Errata for Mask 1N81U

This report applies to mask 1N81U for these products:

- S32V234

### Mask Specific Information

Major mask revision number	0x1
Minor mask revision number	0x0
JTAG identifier	0x098C_901D

**Table 1. Errata and Information Summary**

Erratum ID	Erratum Title
ERR011152	2D-ACE: FIFO_HI and FIFO_LO status flag may be set incorrectly
ERR010327	ADC: Incorrect channel under measure is indicated after sampling phase of conversion until conversion completes
ERR011287	CMU: Sudden loss of clock does not signal the Fault Collection and Control Unit
ERR050920	Core: AArch64 memory streaming performance might be lower than expected when using a 64KB translation granule
ERR050777	Core: Accessing EDPCSR has side-effects when OS Lock is locked
ERR050767	Core: Advanced SIMD integer multiply instructions in IT block might cause data corruption
ERR050772	Core: ATS12NSOPR instruction might incorrectly translate when the HCR.TGE bit is set
ERR050771	Core: Debug not entering Memory Access mode without setting EDSCR.ERR
ERR050779	Core: ETM might output an incorrect exception return address
ERR050773	Core: ETM trace reports incorrect branch target
ERR050877	Core: Fused MAC instructions give incorrect results for rare data combinations
ERR050775	Core: Incorrect fault status codes used for reporting synchronous uncorrectable ECC aborts in the IFSR
ERR050776	Core: Instruction cache parity error might cause an incorrect abort to be taken
ERR006939	Core: Interrupted loads to SP can cause erroneous behavior
ERR009004	Core: ITM can deadlock when global timestamping is enabled
ERR050769	Core: Mismatch between EDPRSR.SR and EDPRSR.R
ERR050878	Core: Processor reset asserted asynchronously could corrupt FPB comparator registers and remap to wrong address

*Table continues on the next page...*



**Table 1. Errata and Information Summary (continued)**

Erratum ID	Erratum Title
ERR050765	Core: Some AT instructions executed from EL3 might incorrectly report a domain fault.
ERR050764	Core: Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation
ERR009005	Core: Store immediate overlapping exception return operation might vector to incorrect interrupt
ERR006940	Core: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used
ERR050778	Core: WFX can cause the PC to jump from lower VA subrange to upper VA subrange
ERR010493	Cortex-A53 MP Core:855873 – B: An eviction might overtake a cache clean operation
ERR010513	Cortex-A53 MPCore 850469 -C: Snoop requests might prevent a store exclusive from passing
ERR010512	Cortex-A53 MPCore 851672-C: ETM might trace an incorrect exception address
ERR010511	Cortex-A53 MPCore 851871-C: ETM might lose counter events while entering wfx mode
ERR010510	Cortex-A53 MPCore 852071-C: Direct branch instructions executed before a trace flush might be output in an atom packet after flush acknowledgement
ERR010509	Cortex-A53 MPCore 852521-C: A64 unconditional branch might jump to incorrect address
ERR010503	Cortex-A53 MPCore 853172 C : ETM might assert AFREADY before all data has been output
ERR010502	Cortex-A53 MPCore 854821 C : A stream of instruction cache invalidate by address operations might cause denial of service
ERR010498	Cortex-A53 MPCore 855831 C: A core might indefinitely delay the response to a DVM Sync message after executing an LDM instruction
ERR010516	Cortex-A53 MPCore 820719-C: Device stores might cause denial of service
ERR010501	Cortex-A53 MPCore 855827 C : PMU counter values might be inaccurate when monitoring certain events
ERR010500	Cortex-A53 MPCore 855829 C : Reads of PMEVCNTRn are not masked by HDCR.HPMN
ERR010499	Cortex-A53 MPCore 855830 C: Loads of mismatched size might not be single-copy atomic
ERR010496	Cortex-A53 MPCore 855872: B: A Store-Exclusive instruction might pass when it should fail
ERR010497	Cortex-A53 MPCore 855874 C: APB data is not masked when PSLVERR is set
ERR010494	Cortex-A53 MPCore- 855871 B: ETM does not report IDLE state when disabled using OSLOCK
ERR008821	Cortex-A53: 836870-C Non-allocating reads might prevent a store exclusive from passing
ERR008823	Cortex-A53: 836919-C Write of JMCR in EL0 does not generate an UNDEFINED exception
ERR009235	Cortex-A53: 845719-B A load might read incorrect data.
ERR010183	Cortex-A53: MPCore 843819-B Memory locations might be accessed speculatively due to instruction fetches when HCR.VM is set
ERR051156	CSE: BOOT_MAC calculation is done only on [DATA] during secure boot
ERR010101	DEC200 : DEC200 generates multiple Flush done interrupt(FLUSH_DN_INT) for a single Frame data transfer from 2D ACE.
ERR050090	DSPI/SPI: Incorrect data may be transmitted in slave mode
ERR009656	DSPI: Frame transfer does not restart in case of SPI parity error in master mode
ERR009976	DSPI: Incorrect data received by master with Modified transfer format enabled when using Continuous serial communication clock mode
ERR050478	ENET: Ethernet RX hang when receiving traffic through multiple queues
ERR009420	FCCU: FOSU may give destructive reset when a hardware recoverable fault of width less than one safe clock occurs
ERR011543	FlexCAN: Nominal Phase SJW incorrectly applied at CRC Delimiter

*Table continues on the next page...*

**Table 1. Errata and Information Summary (continued)**

Erratum ID	Erratum Title
ERR050246	FlexCAN: Receive Message Buffers may have its Code Field corrupted if the Receive FIFO function is used
ERR050119	FlexRay: Disabling of FlexRay Message Buffer during the STARTUP Protocol State takes longer than expected three Slots
ERR009265	FTM: Incorrect match may be generated if intermediate load feature is used in toggle mode
ERR011133	FXOSC: Device will not properly boot if using external oscillator
ERR010820	H264_DEC: H264 Decoder may fail if its configuration registers are programmed while the H264 Decoder clock is enabled.
ERR010206	ISP: Spurious Single-Error-Correction or Double-Error-Detection (SEC/DED) might get reported to the Fault Collection & Control Unit (FCCU) if an IPU initialization procedure is triggered after generation of an actual SEC/DED event from Instruction memory (IMEM).
ERR007274	LINFlexD: Consecutive headers received by LIN Slave triggers the LIN FSM to an unexpected state
ERR009685	MC_RGM: Unexpected reset is observed if DDR Handshake timeout value is set to 1
ERR010598	MEMU: The Memory Error Management Unit may not report a consecutive correctable or uncorrectable error from the same address in FlexRay memory
ERR011172	MIPICSI2: Start of Transmission Error can occur at lower speeds
ERR050070	MMDC: Hardware Write Leveling Calibration Error bits MMDC_MPWLGCR[WL_HW_ERRn] are incorrectly de-asserted
ERR011136	MMDC: HW calibration is not supported in 16 bit DDR configuration
ERR011222	MMDC: I/O pad glitches during power ramp-up which may lead to memory corruption and boot failures when using LPDDR2
ERR011155	MMDC: ZQ calibration issue when interfacing to LPDDR2 memory with two chip selects
ERR010081	OTFAD: MDPC_SRTAR[DMNC] always reads as zero
ERR050518	PAD_RING: Potential pad drive high when I/O power supply is below specification voltage
ERR009196	PCIE: 9000611337-RC Root Error Message Controls Not Fully Implemented.
ERR009193	PCIE: 9000680610-Bus and Device Number not Cleared on Hot Reset
ERR010136	PCIE: 9000783666- AXI Bridge Master (RC): B-Channel Write Response Queue Overflows
ERR009190	PCIE: 9000851378-Gen2: DSP Core Advertising Gen2 Speed Support Does Not Correctly Set Selectable Deemphasis Bit In TS2s Transmitted In Recovery.RcvrCfg State
ERR050130	PIT: Temporary incorrect value reported in LMTR64H register in lifetimer mode
ERR010170	QuadSPI: Insufficient read data may be received in the RX Data Buffer register
ERR009658	SPI: Inconsistent loading of shift register data into the receive FIFO following an overflow event
ERR010400	SSE : Watchdog Error may not get asserted when the Watchdog Counter times out.
ERR010016	TMC: The RSZ register of the Cortex-M4 ETF shows an incorrect RAM size
ERR011151	TPIU: Trace may get corrupted or stalled when functional reset occurs
ERR050593	XRDC: DERRLOCn register may not capture the PAC and the MRC instance correctly in case of error violation
ERR010436	ZipWire: SIPI can have only one initiator with one outstanding write frame at time

**Table 2. Revision History**

Revision	Changes
1	Initial revision
1.1	<p>The following erratum was removed.</p> <ul style="list-style-type: none"> <li>• ERR010481</li> </ul> <p>The following errata were added.</p> <ul style="list-style-type: none"> <li>• ERR011155</li> <li>• ERR011222</li> </ul>
1.2	<p>The following erratum was removed.</p> <ul style="list-style-type: none"> <li>• ERR010542</li> </ul> <p>The following errata were added.</p> <ul style="list-style-type: none"> <li>• ERR010820</li> <li>• ERR011287</li> </ul>
1.3	<p>The following erratum was added.</p> <ul style="list-style-type: none"> <li>• ERR050070</li> </ul>
1.4	<p>The following errata were added.</p> <ul style="list-style-type: none"> <li>• ERR011543</li> <li>• ERR050090</li> <li>• ERR050119</li> </ul> <p>The following errata were revised.</p> <ul style="list-style-type: none"> <li>• ERR006940</li> <li>• ERR008823</li> <li>• ERR010500</li> </ul>
1.5	<p>The following errata were added.</p> <ul style="list-style-type: none"> <li>• ERR050246</li> <li>• ERR050478</li> <li>• ERR050518</li> </ul> <p>The following erratum was revised.</p> <ul style="list-style-type: none"> <li>• ERR050119</li> </ul>
1.6	<p>The following errata were added.</p> <ul style="list-style-type: none"> <li>• ERR050130</li> <li>• ERR050593</li> <li>• ERR050764</li> <li>• ERR050765</li> <li>• ERR050767</li> <li>• ERR050769</li> <li>• ERR050771</li> <li>• ERR050772</li> <li>• ERR050773</li> <li>• ERR050775</li> <li>• ERR050776</li> <li>• ERR050777</li> <li>• ERR050778</li> </ul>

**Table 2. Revision History**

Revision	Changes
	<ul style="list-style-type: none"><li>• ERR050779</li><li>• ERR050877</li><li>• ERR050878</li><li>• ERR050920</li><li>• ERR051156</li></ul>

**ERR011152: 2D-ACE: FIFO\_HI and FIFO\_LO status flag may be set incorrectly**

**Description:** The status flags Pm\_FIFO\_HI and Pm\_FIFO\_LO from the status register DCU\_INT\_STATUS indicate whether the FIFO has reached its upper or lower threshold. These flags may be set even when the condition has not occurred. Monitoring these flags for buffer fill status is not recommended.

**Workaround:** Use DCU\_INT\_STATUS[UNDRUN] flag to monitor if the 2D-ACE is getting sufficient bandwidth to fetch data from memories.

**ERR010327: ADC: Incorrect channel under measure is indicated after sampling phase of conversion until conversion completes**

**Description:** The Main Status Register Channel under measure address field (ADC\_MSR[CHADDR]) indicates which ADC channel is currently performing a conversion. This field indicates the correct channel during the sampling phase of conversion, but will display an incorrect value in the subsequent phases until conversion is complete.

**Workaround:** User must only consider ADC\_MSR[CHADDR] to be valid when the ADC is in the sample phase of conversion. The Main Status Register Status of the ADC field shows when the ADC is in the sample phase (ADC\_MSR[ADCSTATUS] = 0b100).

**ERR011287: CMU: Sudden loss of clock does not signal the Fault Collection and Control Unit**

**Description:** The Clock Monitor Unit (CMU) detects when a monitored clock frequency drops below a programmed threshold through the Frequency Less than Low Threshold (FLL) signal. This FLL signal is routed to the Fault Collection and Control Unit ( FCCU ) providing a mechanism to react to the clock fault. Due to it's implementation, the FLL signal will not be triggered when the monitored clock source suddenly stops.

**Workaround:** The CMU has an internal signal which is designed to give an indication that the monitored clock has dropped below 1/4 of the reference clock CLKMT0\_RMN. This provides an alternative means to detect the sudden loss of clock, however since this internal signal is not routed to the FCCU, the user software must periodically poll bitfield [3] of CMU\_ISR register to detect a sudden loss of clock. Write '0b1' to clear the bitfield [3] of CMU\_ISR after enabling CMU.

## **ERR050920: Core: AArch64 memory streaming performance might be lower than expected when using a 64KB translation granule**

**Description:** Arm errata 859374

Affects: Cortex-A53 MPCore

Fault Type: Programmer Category C

Fault Status: Present in: r0p0, r0p1, r0p2, r0p3, r0p4 Open.

The processor includes an automatic data prefetcher to improve memory streaming performance. Because of this erratum, the data prefetcher does not operate as intended when virtual addresses are translated using a 64KB translation granule.

Configurations Affected:

This erratum affects all configurations of the processor.

Conditions:

- 1) A core is in the AArch64 execution state.
- 2) TCR\_ELx.{TG0 or TG1} is set to 01 for the current Exception level and TTBR combination, indicating a 64KB translation granule.
- 3) One of the following is true:
  1. The current Exception level is Secure EL1, EL2, or EL3.
  2. The current Exception level is Non-secure EL1 or EL0 and Stage 2 translation is not enabled.
  3. The current Exception level is Non-secure EL1 or EL0 and VTCCR\_EL2.TG0 is set to 01, indicating a 64KB translation granule.
  - 4) The core executes a sequence of loads or stores that triggers the automatic data prefetcher. For example, a sequence of loads or stores to consecutive cachelines.

Implications:

The automatic data prefetcher will not prefetch data effectively.

This will result in low AArch64 memory streaming performance for some workloads but it will not cause functional failures.

**Workaround:** AArch64 memory streaming performance can be improved by using the 4KB translation granule.

## **ERR050777: Core: Accessing EDPCSR has side-effects when OS Lock is locked**

**Description:** ARM errata 859075

Affects: Cortex-A53 MPCore

Access to the External Debug Program Counter Sample Register, EDPCSR, should not have side-effects when the OS lock is locked or when the OS Double Lock is locked. Because of this erratum, the side-effects are only disabled when OS Double Lock is locked.

Conditions:

- 1) The OS Lock is locked (EDPRSR.OSLK == 1).

2) The OS Double Lock is unlocked (EDPRSR.DLK == 0).

3) An external debugger reads EDPCSR[31:0].

Impact:

If the erratum conditions are met then EDCIDSR, EDVIDSR, and EDPCSR[63:32] will be updated as if the OS Lock were unlocked.

**Workaround:** There is no workaround.

## **ERR050767: Core: Advanced SIMD integer multiply instructions in IT block might cause data corruption**

**Description:** ARM errata – 859372

Affects: Cortex-A53 MPCore

The Cortex-A53 processor also supports conditional execution of instructions. In the T32 execution state, when the processor executes an IT instruction, up to four instructions at a time can become conditional.

Because of this erratum, if certain Advanced SIMD integer multiply instructions are executed conditionally, the processor might produce the wrong result for subsequent and dependent Advanced SIMD integer multiply-accumulate instructions.

Conditions:

This erratum occurs when the processor executes a sequence of instructions in the T32 execution state. The sequence must comprise of the following instructions:

- 1) An IT instruction.
- 2) An optional instruction, belonging in the IT block.
- 3) An Advanced SIMD producer instruction, belonging in the IT block.
- 4) An optional instruction, belonging in the IT block.
- 5) An Advanced SIMD consumer instruction, which may or may not belong in the IT block.

Additionally, all the following conditions must hold:

- The producer instruction (3) must be one of the following:
  - VMLA (by scalar).
  - VMLA (integer).
  - VMLAL (by scalar).
  - VMLAL (integer).
  - VMLS (by scalar).
  - VMLS (integer).
  - VMLSL (by scalar).
  - VMLSL (integer).
  - VMUL (by scalar).
  - VMUL (integer).
  - VMULL (by scalar).

- VMULL (integer).
  - The consumer instruction (5) must be one of the following:
    - VMLA (by scalar).
    - VMLA (integer).
    - VMLAL (by scalar).
    - VMLAL (integer).
    - VMLS (by scalar).
    - VMLS (integer).
    - VMLSL (by scalar).
    - VMLSL (integer).
  - The <Qd> or <Dd> destination operand of the producer instruction must match the <Qd> or <Dd> source operand of the consumer instruction.
  - The data type of the producer instruction must be compatible with the data type of the consumer instruction.
  - Both the producer instruction and the consumer instructions must be of the integer variant, and not of the floating-point or the polynomial variants.
  - The producer instruction must fail its condition code check.
  - The consumer instruction must either:
    - Not belong in the IT block and therefore execute unconditionally.
    - Belong in the IT block and pass its condition code check, because either:
      - The consumer has the opposite condition code compared to the producer.
      - The consumer has the same condition code compared to the producer, but a flag-setting instruction executes successfully in (4) and changes the condition flags.
- Even though all the conditions above might hold, the erratum does not always occur. Specifically, any of the following conditions might affect its occurrence:
- The timing of the instructions in the processor pipeline.
  - The arrangement of the instructions in dual-issuing pairs.

#### NOTE

Note that the ARM architecture deprecates the conditional execution of any instruction encoding provided by Advanced SIMD that is not also provided by floating-point. The architecture strongly recommends that such T32 Advanced SIMD instructions are never included in an IT block.

#### Impact:

If this erratum occurs, the processor will incorrectly forward parts of the resulting value of the <Qd> or <Dd> operand from the producer instruction to the consumer instruction, even though the producer instruction has failed its condition code check. Therefore, the result of the consumer instruction will be wrong.

**Workaround:** A workaround is not expected to be required in most cases. This is because the erratum conditions require use of deprecated code.



If a workaround is required, the erratum can be avoided by disabling dual-issue. Dual-issue can be disabled by setting CPUACTLR.DIDIS to 0b1. Disabling dual-issue has a significant performance impact so this workaround is not recommended for most systems. If the erratum is encountered, it is preferable to work around it by avoiding the conditions described.

## **ERR050772: Core: ATS12NSOPR instruction might incorrectly translate when the HCR.TGE bit is set**

**Description:** ARM errata – 859082

Affects: Cortex-A53 MPCore

An ATS12NSOPR address translation instruction executed from EL3 might report an incorrect result in the PAR when both the HCR.TGE bit is set and the SCTLR.M is set.

Conditions:

- 1) The core is executing at Exception level 3 in AArch32.
- 2) SCR.NS = 0
- 3) HCR.TGE = 1
- 4) SCTLR(ns).M = 1
- 5) The core executes an ATS12NSOPR instruction.

Impact:

The PAR register will incorrectly report the translation as if the stage 1 MMU was enabled.

Note that the combination of both HCR.TGE and SCTLR.M bits being set was UNPREDICTABLE in earlier versions of the architecture, and was only given a defined behavior to reduce the UNPREDICTABLE space. It is not expected to be a useful combination for software.

**Workaround:** Secure software can clear the SCTLR(ns).M bit before executing the address translation instruction, if the HCR.TGE bit is set. It should restore the previous SCTLR(ns).M value before returning to a lower Exception level.

## **ERR050771: Core: Debug not entering Memory Access mode without setting EDSCR.ERR**

**Description:** ARM errata – 859083

Affects: Cortex-A53 MPCore

The processor supports entering debug Memory Access (MA) mode through the APB interface. This might fail when an instruction previously executed through the APB interface has not yet completed.

Conditions:

- 1) The debugger inserts an instruction through the APB interface by writing the EDITR.
- 2) The debugger writes to the EDSCR to enter MA mode.
- 3) The debugger reads the DTRTX at the same time as the instruction completes.

Impact:

The APB read of the DTRTX does not start up the MA state machine, but EDSCR.TXU and EDSCR.ERR are not set.

**Workaround:** Before executing the instruction, the debugger should write 1 to the EDRCR.CSPA. After executing the instruction, the debugger should poll the EDSCR.PipeAdv bit to determine when the instruction has completed, and only attempt to enter MA mode after the PipeAdv bit is set.

## **ERR050779: Core: ETM might output an incorrect exception return address**

**Description:** ARM errata – 859072

Affects: Cortex-A53 MPCore

The processor supports instruction trace using a per-core ETM. If tracing is enabled, a core's ETM generates an Exception element when the core takes an exception. The Exception element contains an indication that an exception has occurred, the type of exception, and the exception return address. Because of this erratum, the exception return address that is output by the ETM might have bits [63:49] set to 1 when they should be set to 0.

Conditions:

- 1) Trace is enabled.
- 2) Non-invasive debug is permitted.
- 3) A core is in the AArch64 execution state.
- 4) The core executes one of the following instructions at VA 0x0000\_ffff\_ffff\_fffc
  - SVC
  - HVC
  - SMC

Impact:

The exception return address reported by the ETM might be incorrect. For example, an SMC instruction at VA 0x0000\_ffff\_ffff\_fffc might cause the ETM to output an exception return address of 0xffff\_0000\_0000\_0000 instead of 0x0001\_0000\_0000\_0000.

Real code is not expected to meet the conditions in this erratum because, if this erratum did not exist, the core would access a VA that is outside of the implemented virtual address space.

**Workaround:** There is no workaround for this erratum.

## **ERR050773: Core: ETM trace reports incorrect branch target**

**Description:** ARM errata 859080

Affects: Cortex-A53 MPCore

When executing a branch that targets an out of range virtual address, the ETM might trace the target address incorrectly.

Conditions:

- 1) ETM trace is enabled and not prohibited.

2) The core is executing code located in the virtual address range 0hFFFF000000000000-0hFFFFFFFFFFFFFFFF, and bit[48] of the PC is set. If the relevant TCR.TBI bit is set, then bits[63:56] of the virtual address can be any value.

3) The core executes a branch instruction with the target virtual address in the range 0h0002000000000000-0hFFFFFFF000000000, with bit[48] of the target address clear. This could be any kind of instruction that alters the program flow, including immediate branches, register based branches, and returns.

Impact:

The target of the branch will take a translation fault, because it is not in a valid address range, but because the ETM trace decompression will report the branch target incorrectly the reason for the fault might be harder to determine when debugging. Bits[47:0] of the branch target will still be correct, but bits[63:48] will be incorrectly reported as all zero.

**Workaround:** There is no workaround for this erratum.

## **ERR050877: Core: Fused MAC instructions give incorrect results for rare data combinations**

**Description:** Arm errata 839676

Affects: Cortex-M4F

Fault Type: Programmer Category B Rare

Fault Status: Present in: r0p0, r0p1. Open.

The Cortex-M4 processor includes optional floating-point logic which supports the fused MAC instructions (VFNMA, VFNMS, VFMA, VFMS). This erratum causes fused MAC operations on certain combinations of operands to result in either or both of the following:

- A result being generated which is one Unit of Least Precision (ULP) greater than the correct result.
- Inexact or underflow flags written to the Floating-point Status Control Register, FPSCR, incorrectly.

Configurations Affected:

Cortex-M4F configured with floating-point.

Conditions:

The conditions for this erratum are all of the following:

1. Cortex-M4 is configured with floating-point and it is enabled in the Coprocessor Access Control Register, CPACR.

2. Flush-to-Zero (FZ) mode is not enabled, that is, FPSCR.FZ is clear.

3. A fused MAC instruction (VFNMA, VFNMS, VFMA, or VFMS) is executed with all of the following properties:

- The addition part of the operation is Unlike Signed Addition (USA). This implies that the combination of the instruction used and the signs of the operands means that a subtraction is being performed.
- The result of the instruction, before rounding, is subnormal. This implies that the result is smaller in magnitude than  $2^{-126}$ .
- The significance of the product and the addend operand are the same or differ by one.

Implications:

If this erratum occurs, then the processor either produces an incorrect result to a computation or fails to run a floating-point exception routine when it should.

Note:

It is expected that most algorithms only use normalised numbers because:

- Subnormal results have low precision.
- It is easier to avoid underflow.

This erratum does not affect algorithms which only use normalized numbers.

**Workaround:** Enable FZ mode in the FPSCR.

## **ERR050775: Core: Incorrect fault status codes used for reporting synchronous uncorrectable ECC aborts in the IFSR**

**Description:** ARM errata 859077

Affects: Cortex-A53 MPCore

When reporting a synchronous uncorrectable ECC error, an incorrect encoding is written to the fault status code field of the Instruction Fault Status Register.

Configurations Affected

This erratum affects configurations of the processor that have CPU\_CACHE\_PROTECTION or SCU\_CACHE\_PROTECTION set to TRUE.

Conditions:

- 1) A core is in the AArch32 execution state.
- 2) The core is executing in EL0, EL1, or EL3.
- 3) The core is using the Short-descriptor page format.
- 4) An uncorrectable ECC error occurs in one of the following caches:
  - The L1 data cache of the core.
  - The L1 data cache of another core in the processor.
  - The L2 cache.
- 5) An instruction fetch or a translation table walk for an instruction fetch hits the cache line containing the ECC error, triggering a prefetch abort.

If these conditions are met, then IFSR.FS[4] is set to 0 instead of 1.

Impact:

When the short descriptor page format is in use, software cannot make a distinction between a Domain fault, level 1, or a synchronous parity or ECC error on memory access. Both will be reported in the IFSR as 01001.

Similarly, software cannot make a distinction between synchronous external abort on translation table walk and synchronous parity or ECC error on translation table walk. Both will be reported in the IFSR as synchronous external abort on translation table walk (IFSR.FS[4:0] will be 01100 for translation level 1 and 01110 for translation level 2). This is valid behavior because it is IMPLEMENTATION DEFINED whether ECC errors are reported using the

assigned fault code or using another appropriate encoding. However, the behavior is inconsistent with fault code usage in other fault status registers, and in the IFSR when using the Long-descriptor page format.

**Workaround:** When reading the fault status code from the IFSR, software will have to be aware that the value reported might not be accurate. If software reads 01001 for IFSR.FS[4:0], then software will have to determine if a Domain fault could have occurred for this address.

## **ERR050776: Core: Instruction cache parity error might cause an incorrect abort to be taken**

**Description:** ARM errata 859076

Affects: Cortex-A53 MPCore

The core cache protection option allows the processor to detect and correct a 1-bit error in the instruction cache RAMs. If an error is detected on an instruction at the end of a page, and the following page would generate a fault if executed, then under some conditions it is possible for a fault to be taken when no instruction from the faulting page was architecturally executed.

Conditions:

- 1) The core is executing in AArch32 T32 instruction state.
- 2) The program counter points to a 16-bit length instruction which is at the end of a page.
- 3) The related instruction is stored in the L1 instruction cache.
- 4) There is a single bit error on one specific bit of the instruction, which causes the instruction to be incorrectly interpreted as a 32-bit instruction.
- 5) The page which is immediately after the address range of the previous 16-bit instruction would cause a translation fault, access flag fault, or permission fault if it was accessed.

Impact:

If the above conditions are met, then the translation, permission, or access flag fault is incorrectly taken, instead of correcting the error. There is still substantial benefit being gained from the parity logic. There might be a negligible increase in overall system failure rate due to this erratum.

**Workaround:** No workaround is required.

## **ERR006939: Core: Interrupted loads to SP can cause erroneous behavior**

**Description:** Arm Errata 752770: Interrupted loads to SP can cause erroneous behavior

This issue is more prevalent for user code written to manipulate the stack. Most compilers will not be affected by this, but please confirm this with your compiler vendor. MQX™ and FreeRTOS™ are not affected by this issue.

Affects: Cortex-M4, Cortex-M4F

Fault Type: Programmer Category B

Fault Status: Present in: r0p0, r0p1 Open.

If an interrupt occurs during the data-phase of a single word load to the stack-pointer (SP/R13), erroneous behavior can occur. In all cases, returning from the interrupt will result in the load instruction being executed an additional time. For all instructions performing an update to the base register, the base register will be erroneously updated on each execution, resulting in the stack-pointer being loaded from an incorrect memory location.

The affected instructions that can result in the load transaction being repeated are:

- 1) LDR SP,[Rn],#imm
- 2) LDR SP,[Rn,#imm]!
- 3) LDR SP,[Rn,#imm]
- 4) LDR SP,[Rn]
- 5) LDR SP,[Rn,Rm]

The affected instructions that can result in the stack-pointer being loaded from an incorrect memory address are:

- 1) LDR SP,[Rn],#imm
- 2) LDR SP,[Rn,#imm]!

Conditions:

- 1) An LDR is executed, with SP/R13 as the destination.
- 2) The address for the LDR is successfully issued to the memory system.
- 3) An interrupt is taken before the data has been returned and written to the stack-pointer.

Implications:

Unless the load is being performed to Device or Strongly-Ordered memory, there should be no implications from the repetition of the load. In the unlikely event that the load is being performed to Device or Strongly-Ordered memory, the repeated read can result in the final stack-pointer value being different than had only a single load been performed.

Interruption of the two write-back forms of the instruction can result in both the base register value and final stack-pointer value being incorrect. This can result in apparent stack corruption and subsequent unintended modification of memory.

**Workaround:** Most compilers are not affected by this, so a workaround is not required.

However, for hand-written assembly code to manipulate the stack, both issues may be worked around by replacing the direct load to the stack-pointer, with an intermediate load to a general-purpose register followed by a move to the stack-pointer.

If repeated reads are acceptable, then the base-update issue may be worked around by performing the stack pointer load without the base increment followed by a subsequent ADD or SUB instruction to perform the appropriate update to the base register.

## **ERR009004: Core: ITM can deadlock when global timestamping is enabled**

**Description:** ARM ERRATA 806422

The Cortex-M4 processor contains an optional Instrumentation Trace Macrocell (ITM). This can be used to generate trace data under software control, and is also used with the Data Watchpoint and Trace (DWT) module which generates event driven trace. The processor supports global timestamping. This allows count values from a system-wide counter to be included in the trace stream.

When connected directly to a CoreSight funnel (or other component which holds ATREADY low in the idle state), the ITM will stop presenting trace data to the ATB bus after generating a timestamp packet. In this condition, the ITM\_TCR.BUSY register will indicate BUSY.

Once this condition occurs, a reset of the Cortex-M4 is necessary before new trace data can be generated by the ITM.

Timestamp packets which require a 5 byte GTS1 packet, or a GTS2 packet do not trigger this erratum. This generally only applies to the first timestamp which is generated.

Devices which use the Cortex-M optimized TPIU (CoreSight ID register values 0x923 and 0x9A1) are not affected by this erratum.

**Workaround:** There is no software workaround for this erratum. If the device being used is susceptible to this erratum, you must not enable global timestamping.

## **ERR050769: Core: Mismatch between EDPRSR.SR and EDPRSR.R**

**Description:** ARM errata 859373

Affects: Cortex-A53 MPCore

The processor provides access to the EDPRSR through the APB interface. If this access is done at the same time as the core leaves a Warm reset, then a subsequent read of the same register will read an incorrect value of the SR field.

Conditions:

- 1) The core is in Warm reset.
- 2) A debugger reads the EDPRSR register over the APB interface.
- 3) The core comes out of Warm reset during the APB read.
- 4) A second APB read is made to the EDPRSR register.

Impact:

The first read of the EDPRSR will read the SR field and R field both as 0b1.

The second read of the EDPRSR.SR field will read 0b0 whereas the previous read of the EDPRSR.SR was 0b1 while in Warm reset. Because the first read took place while in Warm reset, the sticky bit should still be set on the second read.

**Workaround:** If the debugger reads the EDPRSR and sees both the SR and R fields set, then it must remember this result and on the next EDPRSR read treat the SR bit as if it was set.

## **ERR050878: Core: Processor reset asserted asynchronously could corrupt FPB comparator registers and remap to wrong address**

**Description:** Arm errata 1299509

Affects: Cortex-M4, Cortex-M4F

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r0p1. Open.

Normally, the debugger uses the Flash Patch and Breakpoint (FPB) unit for breakpoints or for patching ROM code during debugging. However, you can also use the FPB functionality as a method of in-the-field ROM code patching. On the Cortex-M4 processor, the processor reset

can be asynchronously asserted and this could potentially cause problems if the processor is in the middle of writing to debug components (which are not reset by the processor reset) such as the FPB unit.

Configurations Affected:

A Cortex-M4 processor that is configured with debug. Conditions

- 1) The processor is programming the FPB to remap an address in ROM to fetch a replacement opcode or vector from an area in RAM.
- 2) The processor is asynchronously reset during the programming of the FPB.
- 3) The data is incorrectly written to the FPB register due to metastability.

Implications:

In the unlikely event of this occurring it may cause incorrect functional operation of the processor to occur. If the FPB is programmed with incorrect data it may cause the processor to unintentionally patch instructions.

**Workaround:** The problem does not arise if the FPB is not used to patch instructions. If the FPB is used to patch instructions, a software workaround is to ensure that the FPB is globally disabled before programming any comparators. If code exists which programs the FPB other than at reset, the software workaround should include: 1. Disable the FPB. 2. Program the individual comparators required. 3. Explicitly disable the individual comparators not required. 4. Re-enable the FPB. This sequence prevents any corruptly programmed FPB comparators from being activated.

## **ERR050765: Core: Some AT instructions executed from EL3 might incorrectly report a domain fault.**

**Description:** ARM errata 859081

Affects: Cortex-A53 MPCore

Address translation instructions executed from EL3 and targeting EL1 or EL0 might report an incorrect result in the PAR when the HCR\_EL2.DC bit is set.

Conditions:

- 1) The core is executing at Exception level 3 in AArch64.
- 2) The core executes one of the following address translation instructions:
  - AT S1E0R, AT S1E0W
  - AT S1E1R, AT S1E1W
  - AT S12E0R, AT S12E0W
  - AT S12E1R, AT S12E1W
- 3) The Exception level targeted by the address translation instruction is Non-secure and AArch32.
- 4) HCR\_EL2.DC is set to 1.
- 5) The DACR is programmed so that domain 0 would cause a domain fault if the HCR\_EL2.DC bit had not been set. Note that this is the default value out of reset.

Impact:

The PAR register will incorrectly report that a domain fault occurred.



**Workaround:** Secure software can set the DACR[1:0] to 0b01 before executing the address translation instruction. It should restore the previous DACR value before returning to a lower Exception level.

## **ERR050764: Core: Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation**

**Description:** ARM errata 1530924

Affects: Cortex-A53 MPCore

A speculative Address Translation (AT) instruction translates using registers that are associated with an out-of-context translation regime and caches the resulting translation in the TLB. A subsequent translation request that is generated when the out-of-context translation regime is current uses the previous cached TLB entry producing an incorrect virtual to physical mapping.

Conditions:

- 1) A speculative AT instruction performs a table walk, translating a virtual address to a physical address using registers associated with an out-of-context translation regime.
- 2) Address translation data that is generated during the walk is cached in the TLB.
- 3) The out-of-context translation regime becomes current and a subsequent memory access is translated using previously cached address translation data in the TLB, resulting in an incorrect virtual to physical mapping.

Impact:

If the above conditions are met, the resulting translation would be incorrect.

**Workaround:** When context-switching the register state for an out-of-context translation regime, system software at EL2 or above must ensure that all intermediate states during the context-switch would report a level 0 translation fault in response to an AT instruction targeting the out-of-context translation regime. A workaround is only required if the system software contains an AT instruction as part of an executable page.

## **ERR009005: Core: Store immediate overlapping exception return operation might vector to incorrect interrupt**

**Description:** Arm Errata 838869: Store immediate overlapping exception return operation might vector to incorrect interrupt

Affects: Cortex-M4, Cortex-M4F

Fault Type: Programmer Category B Rare

Fault Status: Present in: r0p0, r0p1 Open.

The Cortex-M4 includes a write buffer that permits execution to continue while a store is waiting on the bus. Under specific timing conditions, during an exception return while this buffer is still in use by a store instruction, a late change in selection of the next interrupt to be taken might result in there being a mismatch between the interrupt acknowledged by the interrupt controller and the vector fetched by the processor.

Configurations Affected

This erratum only affects systems where writeable memory locations can exhibit more than one wait state.

**Workaround:** For software not using the memory protection unit, this erratum can be worked around by setting DISDEFWBUF in the Auxiliary Control Register.

In all other cases, the erratum can be avoided by ensuring a DSB occurs between the store and the BX instruction. For exception handlers written in C, this can be achieved by inserting the appropriate set of intrinsics or inline assembly just before the end of the interrupt function, for example:

ARMCC:

```
...
__schedule_barrier();
__asm{DSB};
__schedule_barrier();
}
```

GCC:

```
...
__asm volatile ("dsb 0xf" ::: "memory");
}
```

## **ERR006940: Core: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used**

**Description:** Arm Errata 776924: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used

Affects: Cortex-M4F

Fault Type: Programmer Category B

Fault Status: Present in: r0p0, r0p1 Open.

On Cortex-M4 with FPU, the VDIV and VSQRT instructions take 14 cycles to execute. When an interrupt is taken a VDIV or VSQRT instruction is not terminated, and completes its execution while the interrupt stacking occurs. If lazy context save of floating point state is enabled then the automatic stacking of the floating point context does not occur until a floating point instruction is executed inside the interrupt service routine.

Lazy context save is enabled by default. When it is enabled, the minimum time for the first instruction in the interrupt service routine to start executing is 12 cycles. In certain timing conditions, and if there is only one or two instructions inside the interrupt service routine, then the VDIV or VSQRT instruction might not write its result to the register bank or to the FPSCR.

**Workaround:** A workaround is only required if the floating point unit is present and enabled. A workaround is not required if the memory system inserts one or more wait states to every stack transaction.

There are two workarounds:

1) Disable lazy context save of floating point state by clearing LSPEN to 0 (bit 30 of the FPCCR at address 0xE000EF34).

- 2) Ensure that every interrupt service routine contains more than 2 instructions in addition to the exception return instruction.

## **ERR050778: Core: WFx can cause the PC to jump from lower VA subrange to upper VA subrange**

**Description:** ARM errata 859074

Affects: Cortex-A53 MPCore

If a core executes a WFI or WFE instruction at VA 0x0000\_ffff\_ffff\_fffc in the AArch64 execution state then, when the core wakes up, it should resume execution from VA 0x0001\_0000\_0000\_0000. This address is outside of the implemented virtual address space and will trigger a Translation fault if accessed.

Because of this erratum, the core will resume execution from VA 0xffff\_0000\_0000\_0000 instead of 0x0001\_0000\_0000\_0000. Address 0xffff\_0000\_0000\_0000 is a valid address in the upper VA subrange for EL0/EL1 translations if TCR\_EL1.T1SZ is set to 16.

Conditions:

- 1) A core is in the AArch64 execution state.
- 2) The core executes a WFI or WFE instruction at VA 0x0000\_ffff\_ffff\_fffc.

Impact:

Real code is not expected to include a WFI or WFE instruction at VA 0x0000\_ffff\_ffff\_fffc because, if this erratum did not exist, wake-up from the WFx would always trigger an access to a VA that is outside of the implemented virtual address space.

**Workaround:** If a workaround is required, code could be inserted at VA 0xffff\_0000\_0000\_0000 to catch unexpected accesses to this address.

## **ERR010493: Cortex-A53 MP Core:855873 – B: An eviction might overtake a cache clean operation**

**Description:** The Cortex-A53 processor supports instructions for cache clean operations. To avoid data corruption, the processor must ensure correct ordering between evictions and cache clean operations for the same address.

Because of this erratum, the processor might issue an eviction and an L2 cache clean operation to the interconnect in the wrong order. The processor might also issue the transactions such that they are outstanding in the interconnect at the same time. This violates the ACE protocol specification and might cause the transactions to be erroneously re-ordered in the interconnect.

Conditions

The erratum can be hit if the following conditions are met under specific timing conditions.

- 1) One or both of the following are true:
  1. L2ACTLR[14] is set to 1. This enables sending of WriteEvict transactions on the ACE interface when the processor evicts data that it holds in the UniqueClean state.
  2. L2ACTLR[3] is set to 0. This enables sending of Evict transactions on the ACE interface when the processor evicts clean data.

- 2) A core executes a cache clean by address operation for a line that is present and dirty in the L2 cache.
- 3) A core performs a memory access to the same set. This could be any type of memory access including a pagewalk, an instruction fetch, a cache maintenance operation, or a data access.
- 4) The instruction in condition (3) triggers an L2 cache eviction.
- 5) The line chosen for eviction from the L2 cache is the same line that was targeted by the cache clean operation in condition (2).

#### Implications

If the processor is connected to an interconnect that has a system cache or a snoop filter then this erratum might cause data corruption.

**Workaround:** The erratum can be avoided by upgrading cache clean by address operations to cache clean and invalidate operations. For Cortex-A53 r0p3 and later releases, this can be achieved by setting CPUACTLR.ENDCCASCI to 1. For earlier releases, software that uses DCCMVAC or DCCMVAU instructions can replace them with DCCIMVAC instructions.

### **ERR010513: Cortex-A53 MPCore 850469 -C: Snoop requests might prevent a store exclusive from passing**

**Description:** If a Cortex-A53 processor is executing a load and store exclusive instruction in a loop, then there are certain conditions that are allowed to cause the store exclusive instruction to repeatedly fail. This includes another processor repeatedly writing to the same cache line.

In addition to these allowed conditions, a ReadOnce snoop request from the same CPU, another CPU, or another master in the system might also cause the store exclusive instruction to repeatedly fail.

#### Conditions

- 1) CPU A executes a loop containing a store exclusive instruction. The loop will continue until the store exclusive instruction succeeds.
- 2) CPU A, another CPU, or another master in the system issues a ReadOnce transaction for an address with the same L1 data cache index as the store exclusive instruction. On Cortex-A53, a CPU might issue a ReadOnce transaction in the following scenarios:
  - . The CPU performs an instruction fetch to cacheable memory.
  - . The CPU executes a load instruction to cacheable memory that does not cause an allocation into the cache. This might be because the memory is marked as no read allocate or transient in the translation tables, or because a non-temporal load instruction is used.
- 3) The cache line of the ReadOnce transaction is present in the L1 cache of CPU A, and is not present in the L2 cache.
- 4) The ReadOnce transaction triggers a snoop request that arrives at CPU A at the same time as the store exclusive instruction is executing. This causes the store exclusive instruction to fail.
- 5) The ReadOnce transaction is repeated at exactly the same frequency as the store exclusive loop, so that every time around the loop, a snoop arrives at CPU A at the same time as the store exclusive instruction is executing.

CPU A can trigger the stream of ReadOnce snoop requests itself under the following conditions:

- 1) CPU A issues a ReadOnce transaction for an instruction fetch to cacheable memory.
- 2) CPU A issues two or more instruction fetches to cacheable memory with the same L1 instruction cache index as the first instruction fetch. This causes the cache line for the first instruction fetch to be invalidated.
- 3) The sequence is repeated so that the first instruction fetch repeatedly misses in the L1 instruction cache and issues another ReadOnce transaction.

#### Implications

CPU A can be prevented from making progress, resulting in a software livelock. The full set of conditions are unlikely to be met within CPU A because it is unusual for several instruction fetches to compete for the same set in the instruction cache with the frequency required to hit the erratum. However, malicious code executing on another CPU or master might attempt to use this erratum to cause a denial of service attack on CPU A. This is unlikely to be successful because disturbances in the system such as an interrupt or other bus traffic could easily alter the frequency of the loop or the instruction fetch sufficiently to break the livelock.

**Workaround:** A workaround is not expected to be necessary. However, if a workaround is required then a denial of service on a process can be avoided if the OS sets up a timer-based interrupt source to interrupt all snoop generating masters periodically.

## **ERR010512: Cortex-A53 MPCore 851672-C: ETM might trace an incorrect exception address**

**Description:** The address in an exception packet should be the preferred exception return address. Because of this erratum, the address might be equal to the target address of the exception. The trace stream is not corrupted, and decompression can continue after the affected packet.

#### Conditions

The following sequence is required to hit this erratum:

- 1) The ETM must start tracing instructions because of one of the following:
  - . Viewinst goes high.
  - . The security state changes such that trace is now permitted.
  - . The values of the external debug interface (DBGEN, SPIDEN, NIDEN, SPNIDEN) change such that trace is now permitted.
  - . The core exits debug mode.
- 2) Before the core executes any other behavior which would cause trace to be generated, it executes a direct branch instruction, which might be taken or not-taken.

- 3) The next instruction is a load or store that takes a Data Abort or Watchpoint exception.

After this sequence, provided certain timing specific conditions are met, the address in the exception packet might be incorrect.

#### Implications

The trace decompressor might incorrectly infer execution of many instructions from the branch target to the provided address.

**Workaround:** The trace decompressor can detect that this erratum has occurred by checking if the exception address is in the Vector Table and the branch was not expected to be taken to the Vector Table.

A decompressor can infer the correct address of the exception packet. It will be given by the target of the preceding branch (If the branch was taken), or the next instruction after the branch (If the branch was not-taken).

### **ERR010511: Cortex-A53 MPCore 851871-C: ETM might lose counter events while entering wfx mode**

**Description:** If the ETM resources become inactive because of low-power state, there is a one-cycle window during which the counters and the sequencer might ignore counter-at-zero resources.

#### Conditions

The following sequence is required to hit this erratum:

- 1) The core executes a WFI or WFE instruction.
- 2) The ETM enters low-power state because of this.
- 3) In a one-cycle window around this point, either:
  - . A counter in self-reload mode generates a counter-at-zero resource.
  - . A counter in normal mode gets a RLDEVENT on the cycle in which it has just transitioned to zero.
- 4) A counter or sequencer is sensitive to the counter-at-zero resource.

#### Implications

Counters sensitive to a counter-at-zero resource might not reload or decrement. If the sequencer is sensitive to a counter-at-zero resource, it might not change state, or might change to an incorrect state.

**Workaround:** The ETM can be prevented from entering low-power mode by programming LPOVERRIDE bit of TRCEVENTCTL1R to 1. This workaround is only needed if there is a counter or sequencer sensitive to a counter-at-zero resource, and is not normally necessary.

### **ERR010510: Cortex-A53 MPCore 852071-C: Direct branch instructions executed before a trace flush might be output in an atom packet after flush acknowledgement**

**Description:** The Embedded Trace Macrocell (ETMv4) architecture requires that when a trace flush is requested on the AMBA Trace Bus (ATB), a processor must complete any packets that are in the process of being encoded and output them prior to acknowledging the flush request. When trace is enabled, the Cortex-A53 processor attempts to combine multiple direct branch instructions into a single Atom packet. If a direct branch instruction is executed, and an Atom packet is in the process of being generated, Cortex-A53 does not force completion of the packet prior to acknowledging the flush request. This is a violation of the ETMv4 architecture.

#### Conditions

- 1) ETM is enabled.
- 2) Instruction tracing is active.
- 3) One or more direct branch instructions are executed.
- 4) An Atom packet is being encoded but is not complete.
- 5) A trace flush is requested on the AMBA ATB.

#### Implications

When the above conditions occur, the Atom packet being encoded should complete and be output prior to the trace flush request being acknowledged. Because of this erratum, the Atom packet is output after the flush is acknowledged. Therefore, it will appear to software monitoring the trace that the direct branch was executed after that requested flush.

**Workaround:** Enabling the timestamp by setting TRCONFIGR.TS will solve the issue as it will complete the atom packets through the timestamp behavior.

### **ERR010509: Cortex-A53 MPCore 852521-C: A64 unconditional branch might jump to incorrect address**

**Description:** When executing in AArch64 state with address translation disabled, unconditional immediate branch instructions might jump to an incorrect address.

#### Conditions

- 1) The processor is executing in AArch64 state.
- 2) The SCTLR\_ELx.M bit for the current exception level is 0.
- 3) The HCR\_EL2.VM bit is 0.
- 4) A B or BL instruction from the “Unconditional branch (immediate)” encoding class is executed.
- 5) This branch has an imm26 field of 0x1FFFFFFF, encoding a branch target of {pc}+0x7FFFFFFC.

#### Implications

If these conditions are met, then the processor might incorrectly branch to the target {pc}-0x8000004 instead of {pc}+0x7FFFFFFC.

**Workaround:** The workaround for this erratum is to avoid the conditions described.

### **ERR010503: Cortex-A53 MPCore 853172 C : ETM might assert AFREADY before all data has been output**

**Description:** When the AFVALID signal on the ATB interface is asserted, the ETM should immediately start outputting all buffered trace. It should assert the AFREADY output one cycle after all trace that was buffered on the cycle in which AFVALID was first asserted.

Because of this erratum, the AFREADY signal might be asserted before all the necessary trace has been output.

#### Conditions

The ETM must contain buffered trace.

#### Implications

This might result in the ETM containing trace that was generated before the flush request when the rest of the system expects this trace to have been output.

**Workaround:** The system can ensure that all trace has been drained from the ETM by disabling it by setting TRCPRGCTLR.EN to 0. The system should then poll the TRCSTATR.IDLE bit, and once it reads as 1, then the ETM is idle, and all trace that was generated before the write to TRCPRGCTLR has been output.

**ERR010502: Cortex-A53 MPCore 854821 C : A stream of instruction cache invalidate by address operations might cause denial of service**

**Description:** The Cortex-A53 processor supports cache maintenance operations issued as Distributed Virtual Memory (DVM) messages into the processor through a coherent interconnect, and can also broadcast cache maintenance operations between cores within the processor.

Because of this erratum, a continuous stream of instruction cache invalidate by address operations can prevent a core from fetching instructions.

Conditions

1) A core within the processor attempts to execute an instruction from an address which will cause an Instruction Abort.

2) Either:

1. Another core within the processor executes a stream of instruction cache invalidate by address instructions (IC IVAU in AArch64 or ICIMVAU in AArch32), or

2. A coherent master in the system sends a stream of one of the following types of DVM messages:

3. Physical Instruction Cache Invalidate by PA with Virtual Index.

4. Virtual Instruction Cache Invalidate by ASID and VA.

5. Virtual Instruction Cache Invalidate by VA.

If the above conditions are met, the core in condition (1) might delay instruction fetches until the stream of instruction cache maintenance operations has ended.

Implications

Malicious code could be used to prevent instructions from being executed on the core in condition (1) indefinitely.

The security state of the cores has no effect, meaning that a non-secure process issuing DVM messages can stall a secure process on the processor.

**Workaround:** A denial of service on a secure process can be avoided if the secure OS sets up a timer-based interrupt source to interrupt all DVM generating masters periodically

**ERR010498: Cortex-A53 MPCore 855831 C: A core might indefinitely delay the response to a DVM Sync message after executing an LDM instruction**

**Description:** Sometimes, after executing an LDM instruction, a Cortex-A53 core might delay the response to a DVM Sync message indefinitely. This can prevent other cores from making forward progress.

Conditions

1) A Cortex-A53 core, core A, executes an LDM instruction.

. This instruction must load an odd number of registers.



. The address operand of this instruction must have bit [2] set.

2) Core A does not execute any subsequent load, store, or barrier instructions. For example, because it has executed WFI or WFE and suspended execution.

3) Another core in the system, core B, either in the same cluster as core A or another cluster, executes a TLB maintenance instruction.

. Both core A and core B must be in the same inner shareability domain.

. The TLB maintenance instruction must be broadcast to other cores within the inner shareability domain. This is either due to the instruction being one of the \*IS variants or the HCR.FB bit being set.

4) Core A receives a DVM Sync message. This is usually due to another core in the system, which could be different to core B, executing a DSB instruction.

#### Implications

If these conditions are met, then core A might not respond to the DVM Sync message. If another core executing a DSB generates this message, then this DSB instruction would not complete.

**Workaround:** To avoid this erratum, it is necessary to ensure that a Cortex-A53 core cannot have an indefinite period of time after executing an LDM instruction without subsequently executing a single-register load or store or a barrier instruction. This can be achieved by ensuring all Cortex-A53 cores receive periodic wakeup events or by trapping execution of WFE or WFI instructions which would suspend execution and executing a barrier within the handler.

## **ERR010516: Cortex-A53 MPCore 820719-C: Device stores might cause denial of service**

### **Description:** Description

If one or more Cortex-A53 cores execute a stream of store instructions to non-reorderable Device memory, they might prevent a DSB instruction on another core in the processor from making progress.

#### Conditions

1) One or more cores execute a stream of continuous stores to Device-nGnRnE or Device-nGnRE memory. For Cortex-A53 r0p3 and later revisions, this condition must be met by two or more cores.

2) The streams might contain other instructions between the stores, such as data processing instructions. However it must not contain any DMB or DSB instructions or any loads to non-reorderable Device memory.

3) The interconnect or peripheral must take long enough in returning the write responses such that there is always at least one store outstanding on the bus.

4) A different core in the processor executes a DSB instruction.

#### Implications

Malicious code could be used to prevent instructions from being executed on the processor indefinitely.

The security state of the cores has no effect, meaning that a non-secure process issuing device stores can stall a secure process on the processor.

### **Workaround:** Workaround

A denial of service on a secure process can be avoided if the secure OS sets up a timer-based interrupt source to periodically interrupt all masters that could generate a stream of stores. A DMB or DSB instruction should be added to the interrupt handler if it does not already contain one.

## **ERR010501: Cortex-A53 MPCore 855827 C : PMU counter values might be inaccurate when monitoring certain events**

**Description:** The Cortex-A53 processor implements a Performance Monitor Unit (PMU). The PMU allows programmers to gather statistics on the operation of the processor during run time. Because of this erratum, the PMU counter values might be inaccurate when monitoring certain events. Specifically:

- . The INST\_RETIRED event should count architecturally executed instructions. Because of this erratum, it might count more instructions than were architecturally executed.
- . The ST\_RETIRED event should not count Store-Exclusive instructions that fail. Because of this erratum, it does count these instructions.
- . The UNALIGNED\_LDST\_RETIRED event should count loads and stores that fail their alignment check. Because of this erratum, it might also count LDRD and STRD instructions that pass their alignment check.
- . The EXC\_TAKEN and EXC\_RETURN events should be filtered precisely according to the Exception level/Security state they were executed in. Because of this erratum, they will be filtered according to the destination Exception level/Security state

### Conditions

1) A performance counter is enabled and configured to count one of the following events: INST\_RETIRED, ST\_RETIRED, UNALIGNED\_LDST\_RETIRED, EXC\_TAKEN, or EXC\_RETURN.

2) This condition depends on which event the performance counter is configured to monitor:

- . INST\_RETIRED: An immediate branch instruction is executed.
- . ST\_RETIRED: A Store-Exclusive instruction is executed, and fails.
- . UNALIGNED\_LDST\_RETIRED: A LDRD or STRD instruction is executed that is word aligned but not double-word aligned.
- . EXC\_TAKEN: An exception is taken.
- . EXC\_RETURN: An exception return is executed.

3) For INST\_RETIRED, ST\_RETIRED, and UNALIGNED\_LDST\_RETIRED, the filtering settings for the performance counter are set so that an event should be counted if it occurs. For EXC\_TAKEN and EXC\_RETURN, the filtering settings are set so that:

- . Events should be counted in the original Exception level/Security state, but should not be counted in the Exception level/Security state following the exception or the exception return, or
- . Events should not be counted in the original Exception level/Security state, but should be counted in the Exception level/Security state following the exception or the exception return.

### Implications

If the erratum conditions are met, the performance counter might increment when it should not or it might not increment when it should. Specifically:

- . INST\_RETIRED: The counter might erroneously increment by two when only one instruction is executed.

- . ST\_RETIRED: The counter will erroneously increment for the failed Store-Exclusive instruction.
- . UNALIGNED\_LDST\_RETIRED: The counter will erroneously increment for the LDRD or STRD instruction.
- . EXC\_TAKEN: The counter will erroneously increment or erroneously fail to increment.
- . EXC\_RETURN: The counter will erroneously increment or erroneously fail to increment.

**Workaround:** For the EXC\_TAKEN and EXC\_RETURN events, the erratum can be worked around by changing the filtering settings for the performance counter to monitor the destination Exception level/Security state instead of the Exception level/Security state that the exception or exception return are executed in.

There is no workaround for the other PMU events.

### **ERR010500: Cortex-A53 MPCore 855829 C : Reads of PMEVCNTRn are not masked by HDCR.HPMN**

**Description:** The HDCR.HPMN/MDCR\_EL2.HPMN field can be set by software executing at EL2 to restrict access to the performance counters by Non-secure EL0/EL1 software to a subset of the counters. If n is greater than HPMN, then performance counter n is not accessible to Non-secure EL0/EL1. Because of this erratum, on Cortex-A53 reads of inaccessible registers might not return zero as expected, but instead might read the actual contents of the counter registers.

Conditions

- 1) The processor is executing at Non-secure EL1 or EL0.
- 2) HDCR.HPMN/MDCR\_EL2.HPMN is set to a value n, where n less than 6.
- 3) A read of PMEVCNTRn/PMEVCNTRn\_EL0 is executed.

Implications

If the conditions described above are met, then Non-secure EL0/EL1 software can read the values of performance counters reserved for EL2.

**Workaround:** Software executing at EL2 can set the HDCR.TPM/MDCR\_EL2.TPM bit. This will cause all Non-secure EL0 and EL1 accesses to performance monitor registers to trap to EL2. The EL2 software can then emulate accesses as required, and can mask out accesses to reserved registers.

### **ERR010499: Cortex-A53 MPCore 855830 C: Loads of mismatched size might not be single-copy atomic**

**Description:** The Cortex-A53 processor supports single-copy atomic load and store accesses as described in the ARM architecture. However, in some unusual code sequences, this erratum can cause the CPU executing a store and later a load to the same address but with a different access size to load data that does not meet the requirements of a single-copy atomic load.

Conditions

On one CPU, the following sequence must occur:

1) A store instruction is executed. This could be any halfword, word, or doubleword store instruction that is not a store release, and the address must be aligned to the access size.

On a second CPU, which can be within the same cluster or in a different cluster, the following sequence must occur:

1) A store instruction is executed. This store must be a smaller access size to the store on the first CPU, and must be to an address with the bytes accessed by the first CPU. The address must also be aligned to the access size.

2) The store instruction must not allocate into the cache. This could be because:

. The memory address is marked as transient, or

. The write allocate hint in the translation table is not set, or The memory is marked as non-cacheable, or

. The CPU has recently executed a stream of stores and so has dynamically switched into a no write allocate mode.

3) A load instruction is executed. The load must be a larger access size than the store from the same CPU, and at least some bytes of the load must be to the same address as the store. The address must also be aligned to the access size.

The ARM architecture requires that the load is single-copy atomic. However, in the conditions described, the load might observe a combination of the two stores, indicating that the store on the first CPU was serialized first. However, if the load is repeated, then the second time it might see just the data from the store from the first CPU indicating that the store on the first CPU was serialized second.

#### Implications

Concurrent, unordered stores are not common in multi-threaded code. In the C11 standard, they are restricted to the family of “relaxed” atomics. In addition, using different size load and store instructions to access the same data is unusual. Therefore the majority of multi-threaded software is not going to meet the conditions for this erratum.

**Workaround:** Most multi-threaded software is not expected to meet the conditions for this erratum and therefore will not require a workaround. If a workaround is required, then the store on the second CPU should be replaced with a store release instruction.

## **ERR010496: Cortex-A53 MPCore 855872: B: A Store-Exclusive instruction might pass when it should fail**

**Description:** The processor implements an internal exclusive monitor to manage Load-Exclusive, Store-Exclusive, and Clear-Exclusive instructions. Because of this erratum, a Load-Exclusive instruction to cacheable memory might set the monitor to the exclusive state when the processor does not have exclusive access to the line. A subsequent Store-Exclusive instruction might pass when it should fail.

The erratum affects all Load-Exclusive and Store-Exclusive instructions, including Load-Acquire Exclusive and Store-Release Exclusive instructions.

#### Conditions

1) A core executes a store to memory that is marked as both inner-writeback and outer-writeback.

2) The store is not a Store-Exclusive (or a Store-Release Exclusive) or a Store-Release instruction.

- 3) The store is not followed by a DMB SY or DSB.
- 4) The store misses in the L1 data cache.
- 5) The store does not trigger a linefill. This requires one or more of the following to be true:
  - . The core is in read-allocate mode.
  - . The memory is marked as no-write-allocate.
  - . The memory is marked as transient.
  - . The store is a STNP instruction.
  - . The store is triggered by a DC ZVA instruction.
- 6) The core starts a linefill to the same address as the store. The linefill is started for one of the following:
  - . A PRFM, PLD, or PLDW instruction.
  - . An automatic data prefetch.
  - . A pagewalk.
- 7) The core executes a Load-Exclusive (or a Load-Acquire Exclusive) instruction to the same address as the store.
- 8) The store data is forwarded to the Load-Exclusive instruction.
- 9) The Load-Exclusive instruction retires before the linefill in condition (6) is serialised.

If the above conditions are met then the processor might set the internal exclusive monitor. This is not correct because the processor is not guaranteed to have exclusive access to the line.

#### Implications

If another core or master executes a Load-Exclusive instruction to the same address then both cores or masters might gain access to an exclusive region of code at the same time.

Most of the code sequences that can hit this erratum are not expected to exist commonly in real code. The scenario involving read-allocate mode in condition (5) is the most plausible.

The erratum has not been observed in the field. This indicates that the code sequences and timing conditions required to hit the erratum are rare.

**Workaround:** If a workaround is required then the only workaround is to avoid the conditions described. Disabling read-allocate mode by setting CPUACTLR.RADIS to 0b11 degrades write-stream performance. Therefore, the preferred workaround is to avoid conditions (2) or (3) using an appropriate Store-Release or DMB instruction.

## **ERR010497: Cortex-A53 MPCore 855874 C: APB data is not masked when PSLVERR is set**

**Description:** The Cortex-A53 processor implements an AMBA 4 APB interface to provide external debuggers access to some memory-mapped registers. External debugger access to performance monitor registers and breakpoint and watchpoint registers can be disabled by the processor using SDCR in the AArch32 execution state or MDCR\_EL3 in the AArch64 execution state. When access is disabled, an APB read request to one of these registers will receive an error response. However, because of this erratum, the contents of the register will be included in the APB response.

#### Conditions

There are two sequences that can trigger this erratum.

Sequence 1:

1) External debugger access to performance monitor registers is disabled by setting SDCR.EPMAD or MDCR\_EL3.EPMAD to 1.

2) An external debugger initiates an APB read to a performance monitor register.

Sequence 2:

1) External debugger access to breakpoint and watchpoint registers is disabled by setting SDCR.EDAD or MDCR\_EL3.EDAD to 1.

2) An external debugger initiates an APB read to a breakpoint or watchpoint register.

Implications

An external debugger might gain read access to registers that it should not have access to.

**Workaround:** If a workaround is required, PRDATDBG can be masked outside of the processor when PSLVERRDBG is asserted.

### **ERR010494: Cortex-A53 MPCore- 855871 B: ETM does not report IDLE state when disabled using OSLOCK**

**Description:** The OS Lock feature in the ETM allows software running on a processor to disable external debug access and then save the register state before powering down the ETM. There is a defined sequence which must be followed to ensure that the register state is stable and that all trace has been output before the system is powered-down. Because of this erratum, when the OS Lock mechanism is used, the ETM will never indicate that it is safe to be powered-off.

Conditions

1) The ETM is enabled using TRCPRGCTLR.EN == 1

2) The OS Lock feature is used to disable the ETM using TRCOSLAR.OSLK == 1

Implications

Software which follows the defined save and restore sequence will poll TRCSTATR.IDLE, but this will remain HIGH even though the ETM will be disabled and will drain. If the ETM was already disabled with TRCPRGCTLR.EN == 0 then the sequence will behave correctly.

**Workaround:** After the ETM has been disabled using TRCOSLAR.OSLK and the state of TRCPRGCTLR.EN has been recorded, TRCPRGCTLR.EN can be written LOW. This will allow the disable sequence to complete correctly.

### **ERR008821: Cortex-A53: 836870-C Non-allocating reads might prevent a store exclusive from passing**

**Description:** If a Cortex-A53 processor is executing a load and store exclusive instruction in a loop then there are certain conditions that are allowed to cause the store exclusive instruction to repeatedly fail. This includes another processor repeatedly writing to the same cache line.

In addition to these allowed conditions, a non-allocating load from another processor might also cause the store exclusive instruction to repeatedly fail.

Conditions

- 1) One CPU executes a loop containing a load exclusive and a store exclusive instruction. The loop will continue until the store exclusive instruction succeeds.
- 2) The cache line containing the address of the load and store exclusive is present in the L1 cache of the CPU, and is not present in the L2 cache.
- 3) Another CPU, or master in the system, repeatedly executes a load to the same cache line as the load and store exclusive.
- 4) The load does not cause an allocation into the cache of the CPU or master executing the load. This must result in a snoop arriving at the CPU holding the cache line every time the load is performed.
- 5) The load is repeated at exactly the same frequency as the load and store exclusive loop, such that every time around the loop, the snoop arrives at the same time as the store exclusive instruction is executing.

A load can be non-allocating in Cortex-A53 in the following conditions:

- 1) A load instruction is executed on a CPU, while the CPUACTLR.DTAH bit is not set, with one of the following conditions:
  - a. The memory address is marked as writeback cacheable, with no read allocate, in the pagetables.
  - b. The memory address is marked as writeback cacheable, transient, in the pagetables.
  - c. The memory address is marked as writeback cacheable, and an LDNP non-temporal load instruction is used.
- 2) A read transaction is made on the ACP interface.

#### Implications

If a CPU or other master is polling a location to determine when the value changes, then it can prevent another CPU from updating that location, causing a software livelock. However most polling routines would use memory that could be allocated into their cache, and for improved efficiency it is generally recommended to use a load exclusive and WFE instruction to avoid repeated polling. In addition, the frequency of the polling loop must match the frequency of the load store exclusive loop, so it is unlikely to get into this situation, and any other disturbance in the system such as an interrupt or other bus traffic could easily alter the frequency of either loop sufficiently to break the livelock.

**Workaround:** If the repeating load is being executed on a Cortex-A53 CPU then this erratum can be worked around by setting the CPUACTLR.DTAH bit. Note that from r0p4 and onwards, this bit is set by default.

If the repeating load is being executed by another master in the system connected to the ACP interface, then the erratum can be worked around by changing the frequency of the polling so that it no longer aligns with the frequency of the load and store exclusive loop.

If the repeating load is being executed by another master in the system connected through the coherent interconnect, then the erratum can be worked around either by ensuring that the other master allocates the line into its cache, or by changing the frequency of the polling so that it no longer aligns with the frequency of the load and store exclusive loop.

## **ERR008823: Cortex-A53: 836919-C Write of JMCR in EL0 does not generate an UNDEFINED exception**

**Description:** When EL0 is using AArch32 register width state and a write is performed in EL0 to JMCR, the write will be permitted but ignored. The intended behavior is that the instruction should be UNDEFINED.

Conditions

- 1) The processor is executing in AArch32 User mode.
- 2) A write to the JMCR is executed, using the instruction MCR p14,7,Rt,c2,c0,0.

Implications

Rather than treating the MCR instruction as UNDEFINED, the write to the JMCR is ignored.

**Workaround:** It should be expected that a write to the JMCR register will be ignored in AArch32 User mode.

## **ERR009235: Cortex-A53: 845719-B A load might read incorrect data.**

**Description:** When executing in AArch32 state at EL0, a load to the same offset within a different 4GB region as a recent previous load in AArch64 state might read incorrect data.

Conditions

The following sequence must occur for this erratum to be triggered:

Note the sequence requires execution in both AArch32 and AArch64, therefore at least one exception level change is

required in this sequence.

- 1) At EL0 or EL1, in AArch32 or AArch64 state, a load, store, preload or data- or instruction-cache maintenance by

MVA instruction is executed.

- 2) At EL0 or EL1 in AArch64 state, a load is executed with VA[63:32] != 32'h00000000.

- 3) At EL0 in AArch32 state, a load is executed to the same 4KB region as the instruction in position 1, with VA[31:6]

matching the VA from the instruction in position 2.

Between position 1 and position 3 in the sequence, the following must not occur:

- A write to any of the following registers:
  - Any SCTLR.
  - Any TTBR.
  - Any TCR.
  - Any MAIR.
  - CONTEXTIDR.
- A TLB maintenance instruction is executed, followed by a DSB.

Between position 2 and position 3 in the sequence, the following must not occur:

- An address translation instruction is executed.

Implications



If the above conditions are met, then data corruption could occur. The load at EL0 could access data written at EL1 for

which it does not have read permissions, however a load in non-secure state cannot access secure data.

**Workaround:** The erratum can be avoided by causing one of the conditions given in the list above to occur between positions 2 and 3 in the sequence. Because there must be an exception return from AArch64 to AArch32 between positions 2 and 3 in the above sequence, the recommended workaround is to insert a write of the CONTEXTIDR\_EL1 register into operating system exception return sequences.

### **ERR010183: Cortex-A53: MPCore 843819-B Memory locations might be accessed speculatively due to instruction fetches when HCR.VM is set**

**Description:** The ARMv8 architecture requires that when all associated stages of translation are disabled for the current exception level, memory locations are only accessed due to instruction fetches within the same or next translation granule as an instruction which has been or will be fetched due to sequential execution. In the conditions detailed below, the Cortex-A53 MPCore processor might access other locations speculatively due to instruction fetches.

Configurations affected

All configurations of Cortex-A53 are affected.

Conditions

- 1) The processor must be executing at EL3, EL2 or Secure EL1.
- 2) The processor can be in either AArch32 or AArch64 execution state.
- 3) Address translation is disabled for the current exception level (by clearing the appropriate SCTLR.M, HSCTLR.M or SCTLR\_ELx.M bit).
- 4) The HCR.VM/HCR\_EL2.VM bit is set.

Implications

If these conditions are met, then speculative instruction fetches might be made to memory locations not permitted by the architecture.

**Workaround:** Because the HCR.VM bit is reset low, this situation is most likely to arise in powerdown code, if EL2 or EL3 software disables address translation before the core is powered down. To work around this erratum, software should ensure that HCR.VM is cleared before disabling address translation at EL3, EL2 or Secure EL1.

### **ERR051156: CSE: BOOT\_MAC calculation is done only on [DATA] during secure boot**

**Description:** When the BOOT\_MAC is calculated according to the SHE specification the SECURE\_BOOT command fails. The reason is that the specification defines the MAC calculation over ["0..0"96b SIZE|DATA] but the CSE calculates the MAC during the secure boot process only over the [DATA].

When the first secure boot is executed and the BOOT\_MAC is not programmed on the device then the CSE will calculate it (only over the [DATA]) and store it automatically into the BOOT\_MAC slot.

**Workaround:** BOOT\_MAC must be calculated only over the [DATA].

**ERR010101: DEC200 : DEC200 generates multiple Flush done interrupt(FLUSH\_DN\_INT) for a single Frame data transfer from 2D ACE.**

**Description:** DEC200 decompresses the data read from 2D ACE and requires separate frame ID information per master ID. The 2D ACE generates frame end ID signal irrespective of the number of planes enabled. So even if a single layer is enabled, the 2D ACE generates 3 separate frame ID signals for DEC200. The DEC200 sees this as separate IDs and generates multiple interrupts to the core.

**Workaround:** In case DEC200 is configured for reading only single ID, the software should respond to only the first FLUSH\_DN\_INT from DEC200 and ignore the remaining interrupts.

**ERR050090: DSPI/SPI: Incorrect data may be transmitted in slave mode**

**Description:** If the Serial Peripheral Interface (SPI or the Deserial/Serial Peripheral Interface) is operating in slave mode, incorrect or stale data may be transmitted in next transaction without underflow interrupt generation if the set up time of the Peripheral Chip Select (PCS) to the SPI Serial Clock (SCLK) is short and the transmit FIFO may become empty after one transaction.

This can occur if the PCS to SCK is less than:

$$4 \times \text{IPG\_CLOCK\_PERIOD} + 4 \times \text{DSPI\_CLOCK\_PERIOD} + 0.5 \times \text{SCK\_CLOCK\_PERIOD}$$

Where:

IPG\_CLOCK is the internal bus clock ("system" clock)

DSPI\_CLOCK is the protocol clock.

SCK\_CLOCK is the Line-Side Serial Communication Clock.

**Workaround:** When operating in slave mode, software must ensure that the time interval between PCS assertion to start of SCK Clock is greater than  $4 \times \text{IPG\_CLOCK\_PERIOD} + 4 \times \text{DSPI\_CLOCK\_PERIOD} + 0.5 \times \text{SCK\_CLOCK\_PERIOD}$ .

To meet this requirement, the Master SPI can either lengthen the PCS to SCK assertion time or decrease the frequency of the communication interface, or both.

**ERR009656: DSPI: Frame transfer does not restart in case of SPI parity error in master mode**

**Description:** In the Deserial Serial Peripheral Interface (DSPI) module, in the scenario when:

1. Master/slave mode select bit (MSTR) of Module Configuration register (MCR) is set (MCR[MSTR]=0b1) to configure the module in master mode
2. SPI communication is selected via DSPI Configuration field (DCONF) in MCR (MCR[DCONF] = 0b00)
3. Parity reception check on received frame is enabled by setting the Parity Enable or Mask tASC delay (PE\_MASC) bit of DSPI PUSH FIFO Register In Master Mode (PUSHR), i.e. PUSHR[PE]=0b1.
4. Parity Error Stop bit (PES) of MCR is set (MCR[PES]=0b1) which stops SPI frame transfer in case of parity error

5. Parity error is detected on received frame.

Then the next frame transfer is stopped, the SPI Parity Error Flag bit (SPEF) of the DSPI Status Register (DSPI\_SR) is set (SR[SPEF] = 0b1) and the corresponding SPI parity error interrupt is asserted. Even after the interrupt is serviced and SR[SPEF] is reset, the frame transfer does not restart.

**Workaround:** Do not use SPI frame transfer stop in case of parity error detection for SPI transmission in master mode. For this, keep the Parity Error Stop bit of Module Configuration Register de-asserted (MCR[PES] = 0b0).

### **ERR009976: DSPI: Incorrect data received by master with Modified transfer format enabled when using Continuous serial communication clock mode**

**Description:** When the Deserial Serial Peripheral Interface (DSPI) module is configured as follows:

1. Master mode is enabled (Master/Slave Mode Select bit in Module Configuration Register is set (DSPI\_MCR [MSTR] = 0b1))
2. Modified transfer format is enabled (Modified Transfer Format Enable bit in Module Configuration Register is set (DSPI\_MCR [MTFE] = 0b1))
3. Continuous serial communication clock mode is enabled (Continuous SCK Enable bit in Module Configuration Register is set (DSPI\_MCR [CONT\_SCKE] = 0b1))

In this configuration if the frame size of the current frame is greater than the frame size of the next received frame, corrupt frames are received in two scenarios:

- a) Continuous Peripheral Chip Select Enable bit in PUSH TX FIFO Register is set (DSPI\_PUSHR [CONT] = 0b1)
- b) DSPI\_PUSHR [CONT] = 0b0 and lower significant bit of the frame is transferred first (LSB first bit in Clock and Transfer Attributes Register is set (DSPI\_CTAR [LSBFE] = 0b1))

**Workaround:** To receive correct frames:

- a) When DSPI\_PUSHR [CONT] = 0b1, configure the frame size of the current frame less than or equal to the frame size of the next frame (for all frames).
- b) When DSPI\_PUSHR [CONT] = 0b0, configure DSPI\_CTAR [LSBFE] = 0b0. Alternatively, configure the frame size of the current frame less than or equal to the frame size of the next frame (for all frames).

Make sure that for all received frames, the bits are read equal to their respective frame sizes and any extra bits during POP operation are masked.

### **ERR050478: ENET: Ethernet RX hang when receiving traffic through multiple queues**

**Description:** When two or more applications are accessing ENET module using two or more queues configured in receive mode with flushing enabled (RX\_FLUSH), a lock up of the receive path can happen when the queues become full. When this occurs, the buffer descriptor for the last received packet contains an incorrect packet size (equal to the maximum buffer size) and packets cannot be received anymore

**Workaround:** To recover the receive path lock up in such scenario, the ENET hardware block must be reset and re-configured.

To prevent the lock up scenario the RX flushing needs to be disabled (RX\_FLUSHn = 0)

## **ERR009420: FCCU: FOSU may give destructive reset when a hardware recoverable fault of width less than one safe clock occurs**

**Description:** The Fault Collection and Control Unit Output Supervision Unit (FOSU) may issue a destructive reset in the following conditions:

- An input fault is programmed as hardware recoverable (FCCU\_RF\_CFG0, FCCU\_RF\_CFG1)
- The reaction programmed is only Error Output pin (EOUT) signaling (FCCU\_EOUT\_SIG\_EN0, FCCU\_EOUT\_SIG\_EN1)
- The source fault coming on the Recoverable Fault (RF) line is asserted only for less than one safe clock duration

**Workaround:** 1) If a fault must be configured as hardware recoverable, may last less than one safe clock cycle and requires EOUT signalling, program long/short functional reset for that HW recoverable fault besides eout signalling.

2) Alternatively if a fault assertion may last less than one safe clock cycle and only EOUT signaling is preferred as reaction, the said fault shall be configured as software recoverable

## **ERR011543: FlexCAN: Nominal Phase SJW incorrectly applied at CRC Delimiter**

**Description:** During the reception of a CAN-FD frame when the Bit Rate Switch (BRS) is enabled, the Synchronization Jump Width (SJW) for the CRC Delimiter bit is incorrectly defined by the Nominal Phase SJW. The CAN specification stipulates that the CRC Delimiter bit should have a SJW set by the Data Phase SJW.

When a resynchronization event is triggered for the CRC delimiter bit (recessive in correct operation), the sample point will be adjusted by an amount as defined by the Nominal Phase SJW rather than the specified Data Phase SJW. This may result in the incorrect detection of a dominant bit leading to a CAN error frame. However, as the CRC delimiter bit position will only apply the SJW upon the detection of an unexpected dominant bit on the CAN bus, an error frame is already likely. For the case the SJW is applied at the CRC delimiter and a recessive bit is not detected, the receiving node will issue an error frame.

The CAN protocol is designed to handle resynchronization errors and hence the CAN bus will recover from the insertion of the incorrect SJW at the CRC delimiter. Upon detecting the error frame the transmitting node will re-transmit the frame.

The following FlexCAN configurations are not affected:

- Classical CAN frames (CAN 2.0B)
- CAN FD frames with bit rate switch disabled (BRS = 0)
- CAN FD frames with Nominal Phase SJW equal to Data Phase SJW
- CAN FD transmissions

Configuration for the FlexCAN:

- Nominal Phase SJW is configured by the Resync Jump Width bit in the CAN Control Register 1 (CAN\_CTRL1[RJW]) or by the Extended Resync Jump Width bit in the CAN Bit Timing Register (CAN\_CBT[ERJW])
- Data Phase SJW is configured by the Fast Resync Jump Width bit in the CAN FD Bit Timing Register (CAN\_FDCBT[FRJW])

**Workaround:** The robustness of the CAN protocol ensures that the receiver automatically recovers from the application of the incorrect SJW. The CAN protocol is designed to recover from resynchronization errors and hence any frame that is not correctly received will be re-sent by the transmitting node.

## **ERR050246: FlexCAN: Receive Message Buffers may have its Code Field corrupted if the Receive FIFO function is used**

**Description:** If the Code Field of a Receive Message Buffer is corrupted it may deactivate the Message Buffer, so it is unable to receive new messages. It may also turn a Receive Message Buffer into any type of Message Buffer as defined in the Message buffer structure section in the device documentation.

The Code Field of the FlexCAN Receive Message Buffers (MB) may get corrupted if the following sequence occurs.

- 1- A message is received and transferred to an MB (i.e. MBx)
- 2- MBx is locked by software for more than 20 CAN bit times (time determines the probability of erratum to manifest).
- 3- SMB0 (Serial Message Buffer 0) receives a message (i.e. message1) intended for MBx, but destination is locked by the software (as depicted in point 2 above) and therefore NOT transferred to MBx.
- 4- A subsequent incoming message (i.e. message2) is being loaded into SMB1 (as SMB0 is full) and is evaluated by the FlexCAN hardware as being for the FIFO.
- 5- During the message2, the MBx is unlocked. Then, the content of SMB0 is transferred to MBx and the CODE field is updated with an incorrect value.

The problem does not occur in cases when only Rx FIFO or only a dedicated MB is used (i.e. either RX MB or Rx FIFO is used). The problem also does not occur when the Enhanced Rx FIFO and dedicated MB are used in the same application. The problem only occurs if the FlexCAN is programmed to receive in the Legacy FIFO and dedicated MB at the same application.

**Workaround:** This defect only applies if the Receive FIFO (Legacy Rx FIFO) is used. This feature is enabled by RFEN bit in the Module Control Register (MCR). If the Rx FIFO is not used, the Receive Message Buffer Code Field is not corrupted.

If available on the device, use the enhanced Rx FIFO feature instead of the Legacy Rx FIFO. The Enhanced Rx FIFO is enabled by the ERFEN bit in the Enhanced Rx FIFO Control Register (ERFCR).

The defect does not occur if the Receive Message Buffer lock time is less than or equal to the time equivalent to 20 x CAN bit time.

The recommended way for the CPU to service (read) the frame received in a mailbox is by the following procedure:

1. Read the Control and Status word of that mailbox.
2. Check if the BUSY bit is deasserted, indicating that the mailbox is not locked. Repeat step 1) while it is asserted.
3. Read the contents of the mailbox.
4. Clear the proper flag in the IFLAG register.

5. Read the Free Running Timer register (TIMER) to unlock the mailbox

In order to guarantee that this procedure occurs in less than 20 CAN bit times, the MB receive handling process in software (step 1 to step 5 above) should be performed as a 'critical code section' (interrupts disabled before execution) and should ensure that the MB receive handling occurs in a deterministic number of cycles.

### **ERR050119: FlexRay: Disabling of FlexRay Message Buffer during the STARTUP Protocol State takes longer than expected three Slots**

**Description:** Disabling of FlexRay Message Buffer takes longer than the expected three Slots. This is observed, when software application tries to disable the Message Buffer during the FlexRay STARTUP protocol state (vPOC!State = POC:startup) when vPOC!StartupState = "initialize schedule" or "integration consistency check".

In this scenario, FlexRay Communication Controller keeps the specific Message buffer search results until the availability of next cycle start/segment start/slot start events and therefore prevent the disabling of Message Buffer.

Note:

1.All Message Buffers can be disabled immediately if FlexRay protocol state (vPOC!State) is in following States: "POC:default config", "POC:config", "POC:wakeup", "POC:ready", "POC:halt", "POC:startup" and (vPOC!StartupState = "POC:integration listen" or "POC: ColdStart-Listen").

2.All Message Buffers can be disabled within three slots, if FlexRay protocol state (vPOC!State) is in following states: "POC: Normal-Active" or "POC: Normal-Passive".

**Workaround:** Do not disable Message Buffer, while FlexRay is in STARTUP protocol State

### **ERR009265: FTM: Incorrect match may be generated if intermediate load feature is used in toggle mode**

**Description:** When a channel (n) match is used as an intermediate reload, an incorrect second match may occur immediately following the correct match. The issue is problematic only if channel (n) is configured for output compare with the output configured to toggle mode. In this scenario, channel (n) toggles on the correct match and again on the incorrect match. The issue may also occur if a certain channel has a match which is coincident with an intermediate reload point of any other channel.

**Workaround:** If any channel is configured for output compare mode with the output set for toggle mode, the intermediate reload feature must not be used.

### **ERR011133: FXOSC: Device will not properly boot if using external oscillator**

**Description:** Boot ROM code does not have a mechanism to select the FXOSC oscillator mode: Enable or Bypass. By default, Boot ROM will assume the XOSC is using an external crystal therefore XOSC is configured in Enable mode. When the device is connected to an external Oscillator with the intention of using the XOSC in Bypass mode the device mode transition may hang during Boot ROM execution.

**Workaround:** When the device is using an external oscillator and bypass mode is required, XTAL pin cannot be left floating. The XTAL pin needs to be connected to a 500mV reference using a voltage divider. For example: two series resistor of 10K 5% connected to the 1V core voltage supply. In addition to this, the value of the trans-conductance must be set to 0 (FXOSC\_CTL[GM\_SEL] = 000b).

The value of the GM\_SEL bit field is set at boot by the device that reads the RCON pins or Fuses depending on the configured boot mode.

Once the device has finished execution of bootrom code, the FXOSC module must be properly configured by the user code to bypass mode and a mode change is required. After that, it is allowed to configure the PLLs to use the external oscillator as clock source.

**ERR010820: H264\_DEC: H264 Decoder may fail if its configuration registers are programmed while the H264 Decoder clock is enabled.**

**Description:** The H264 Decoder has various registers that configure its operational state. If the decoder is being actively clocked then writing to some of these registers could cause the internal state machine to enter an invalid state and fail during various H264 decoder activities such as context switch mode and high intra mode.

**Workaround:** Follow the below sequence before programming the noted internal registers:

1. Disable the H264 Decoder clock using the SoC clock configuration module – for example the MC\_ME module

2. Program the following registers as required:

H264\_DEC\_MCR, H264\_DEC\_VCR, H264\_DEC\_VO\_STRn\_Y\_ADDR,  
H264\_DEC\_VO\_STRn\_CB\_ADDR, H264\_DEC\_VO\_STRn\_CR\_ADDR,  
H264\_DEC\_VO\_STRn\_NRLINES, H264\_DEC\_RATE\_FLOW\_CNTRL

3. Enable the H264 Decoder clock using the SoC clock configuration module

**ERR010206: ISP: Spurious Single-Error-Correction or Double-Error-Detection (SEC/DED) might get reported to the Fault Collection & Control Unit (FCCU) if an IPU initialization procedure is triggered after generation of an actual SEC/DED event from Instruction memory (IMEM).**

**Description:** Spurious SEC/DED might get reported to the FCCU if an IPU initialization procedure is triggered after generation of an actual SEC/DED event from IMEM.

**Workaround:** At occurrence of SEC/DED event from IMEM of the IPU, if the software triggers IPU initialization procedure then it should ignore any further SEC/DED event reported from IPU until the IPU initialization procedure is complete. SEC and DED event reporting do not effect each other i.e. reporting of one cannot lead to a spurious reported of another.

## **ERR007274: LINFlexD: Consecutive headers received by LIN Slave triggers the LIN FSM to an unexpected state**

**Description:** As per the Local Interconnect Network (LIN) specification, the processing of one frame should be aborted by the detection of a new header sequence and the LIN Finite State Machine (FSM) should move to the protected identifier (PID) state. In the PID state, the LIN FSM waits for the detection of an eight bit frame identifier value.

In LINFlexD, if the LIN Slave receives a new header instead of data response corresponding to a previous header received, it triggers a framing error during the new header's reception and returns to IDLE state.

**Workaround:** The following three steps should be followed -

- 1) Configure slave to Set the MODE bit in the LIN Time-Out Control Status Register (LINTCSR[MODE]) to '0'.
- 2) Configure slave to Set Idle on Timeout in the LINTCSR[IOT] register to '1'. This causes the LIN Slave to go to an IDLE state before the next header arrives, which will be accepted without any framing error.
- 3) Configure master to wait for Frame maximum time (T Frame\_Maximum as per LIN specifications) before sending the next header.

Note:

$T_{Header\_Nominal} = 34 * T_{Bit}$

$T_{Response\_Nominal} = 10 * (N_{Data} + 1) * T_{Bit}$

$T_{Header\_Maximum} = 1.4 * T_{Header\_Nominal}$

$T_{Response\_Maximum} = 1.4 * T_{Response\_Nominal}$

$T_{Frame\_Maximum} = T_{Header\_Maximum} + T_{Response\_Maximum}$

where TBit is the nominal time required to transmit a bit and NData is number of bits sent.

## **ERR009685: MC\_RGM: Unexpected reset is observed if DDR Handshake timeout value is set to 1**

**Description:** The DDR handshake duration when enabled is controlled by the HNDSHK\_TO\_VAL field in the MC\_RGM DDR Handshake Enable register (MC\_RGM\_DDR\_HE). When the DDR handshake is triggered, its duration in terms of number of clock cycles is bounded by the value held in HNDSHK\_TO\_VAL. Whenever, the count specified in HNDSHK\_TO\_VAL field expires the reset cycle is initiated. However, if the field HNDSHK\_TO\_VAL is written with a value of 1, then no DDR handshake is performed and reset cycle is initiated right away without any reset event

**Workaround:** Whenever DDR handshake feature is enabled and used for a particular reset, the value of the timeout event in the MC\_RGM\_DDR\_HE[HNDSHK\_TO\_VAL] field should be greater than 1.



## **ERR010598: MEMU: The Memory Error Management Unit may not report a consecutive correctable or uncorrectable error from the same address in FlexRay memory**

**Description:** The Memory Error Management Unit (MEMU) blocks the reporting of duplicate errors (correctable/un-correctable) coming from the same address if the valid bit (VLD) has not been cleared in reporting table. If the VLD bit is cleared then the error should be logged and reported.

However, if an entry in the FlexRay data RAM (DRAM) is logged in the reporting table and the valid bit is cleared and if the next error occurs from the same address, then the MEMU will not log this error in the reporting table or report it. This is only the case where consecutive errors occur at the same address in FlexRay DRAM. If an error from a different address occurs in between errors from the same address then these will all be correctly reported.

**Workaround:** Application software should note the location of the initial error reported by the MEMU module and should not rely on subsequent reporting of consecutive errors from the same address.

It is possible to reset the error reporting process for this error address as follows: inject another error at a suitable alternative address in FlexRay DRAM and then clear the VLD bit for this injected error. In this case an error at the original address would be reported however a second error at the injected address would not be reported.

Alternatively, the application software could perform a reset if an error is found however it is likely that an error in FlexRay DRAM would be detected in the data transmission protocol.

## **ERR011172: MIPICSI2: Start of Transmission Error can occur at lower speeds**

**Description:** At data rates < 333Mbps per lane, the high speed data plus high speed trail (Ths\_trail) duration must be > 4 bytes (per lane). I.E., if a single byte of HS data is sent per lane, then the Ths\_trail must be of duration >  $3 \cdot 8 \cdot UI$ , where UI is the Unit Interval at the associated data rate. If this condition is not met, a start of transmission error can occur. The start of transmission error may happen on the current high speed burst or subsequent bursts.

**Workaround:** The Ths\_trail duration is typically a configurable value in the transmitter. When using data rates < 333Mbps, configure the transmitter in such way that the Ths\_trail duration is always >  $3 \cdot 8 \cdot UI$ .

## **ERR050070: MMDC: Hardware Write Leveling Calibration Error bits MMDC\_MPWLGCR[WL\_HW\_ERRn] are incorrectly de-asserted**

**Description:** During Auto Hardware Write Leveling, the error status bits (MMDC\_MPWLGCR[WL\_HW\_ERRn]) should be set if an error occurs. These bits are set when an error occurs but then are cleared automatically before software can capture the status. Consequently, the error status bits are not a reliable indication whether a hardware write leveling error has occurred.

**Workaround:** If the hardware write leveling was successful, the MMDC PHY Write Leveling HW Error Register (MMDC\_MPWLHWERR) will contain non-zero values for each byte lane used. A zero value for an active byte lane indicates that a hardware write leveling error occurred. Software should use MMDC\_MPWLHWERR as the error indication instead of MMDC\_MPWLGCR.

## **ERR011136: MMDC: HW calibration is not supported in 16 bit DDR configuration**

**Description:** Device HW calibration is performed on all 4 byte lanes. For 16 bit mode, unused byte lanes will result in calibration error and will stop the automatic hardware calibration.

**Workaround:** For DDR3/DDR3L 16 bit mode, software calibration process should be followed as described in the Reference Manual 'Calibration Process' section in the MMDC chapter.  
Using LPDDR2 in 16 bit mode is not supported.

## **ERR011222: MMDC: I/O pad glitches during power ramp-up which may lead to memory corruption and boot failures when using LPDDR2**

**Description:** During the ramp-up of the power supply for the DDR I/O (VDD\_DDR\_IO), some glitches can appear on the DDR I/O pads. These glitches can force the LPDDR2 memory into a non-idle state and result in memory corruption and boot failures.

This erratum does not apply to DDR3/DDR3L because the memory chip has a separate dedicated RESET signal, which is not present in LPDDR2. The assertion of the RESET signal guarantees that the DDR memory is in the correct state for initialization, regardless of the possible DDR IO Pad glitches.

**Workaround:** If the system is using LPDDR2, the following software work-around must be implemented.

In the initialization sequence of the Multi Mode DDR Controller (MMDC), a "Pre-Charge All" command must be added prior to issuing a "Mode Register Write" (MRW) command to reset the LPDDR2 memory. It must be done for each channel, and each chip select within a channel. For example, in the case where 2 MMDC channels are used and that each of them is connected to a LPDDR2 chips with 2 chip selects, then a total of 4 "Pre-Charge All" commands must be configured with the following steps:

- 1) Set the MMDC\_MDSCR of the first MMDC channel to 0x00008010 in order to issue the pre-charge command to CS0.
- 2) Set the MMDC\_MDSCR of the first MMDC channel to 0x00008018 in order to issue the pre-charge command to CS1.
- 3) Repeat step 1 and 2 for the second MMDC channel.
- 4) Issue the "Mode Register Write" command and complete the initialization sequence normally.

## **ERR011155: MMDC: ZQ calibration issue when interfacing to LPDDR2 memory with two chip selects**

**Description:** This issue is relevant to processors using the MMDC DDR controller, when attempting to connect to LPDDR2 memories that are single channel (x32)/dual die. When using these memory devices, the drive strengths of the READ DQS and DQ pins coming from the LPDDR2 devices becomes degraded after a short period of time, resulting in corrupted READ operations. The degraded drive strengths that result from the MMDC issuing the ZQ calibration commands to both Chip Selects (CS) of the LPDDR2 occur nearly simultaneously, which causes a shared ZQ calibration resistor to give incorrect results.

- This issue can cause data read by the MMDC DDR controller from the DRAM memory to be corrupted due to the incorrect drive strengths.
- This issue only impacts certain LPDDR2 memory configurations using multiple chip selects and one ZQ resistor. It does not impact devices with a single CS.
- This issue does not impact DDR3/DDR3L memories.

Memory vendors connect the ZQ calibration pin for two dies internally to their parts (for higher densities) forcing the two memory dies to share a single calibration resistor. This is allowed by JEDEC standards, with a caveat that the DDR controller never attempts to issue ZQ command requests to the two memory die at the same time for ZQ calibration. The affected MMDC does the calibration in parallel (at the same time) which causes the calibration to be incorrect. The affected MMDC does not support a mode to run the calibrations serially (one after the other).

Please note that having two ZQ resistors connected on the memory is not sufficient because the memory vendor can have them configured such that both CS/multiple dies can still access the same ZQ resistor at the same time (resulting in the degraded drive strength).

**Workaround:** For functional safety critical applications and to completely avoid this issue, use Single CS (RANK) devices only.

Customers can use LPDDR2 devices that have a single CS or are compatible with the MMDC. To achieve higher densities customers can use multiple single-CS (RANK) LPDDR2 devices, however this requires additional board space and routing.

For non functional safety critical applications, two options are available:

1) Connect the ZQ calibration pin to a fixed supply:

The JEDEC standards for LPDDR2 give the option to remove the ZQ calibration resistor and simply connect the ZQ calibration resistor pin(s) to VDDCA. The JEDEC specification also specifies the allowed drive strengths of the LPDDR2 device must be achieved over the entire operating temperature range. NXP recommends programming the LPDDR2 memory device to the maximum drive strength in conjunction with the connection of the ZQ calibration resistor pin(s) to the VDDCA on the customer board.

2) Use ZQ SW Calibration:

Users can disable the default automatic ZQ calibrations, both long (ZQCL) and short (ZQCS) calibration, and implement a software patch that triggers ZQCL and ZQCS commands to be sent at staggered times to either chip select. This requires the software to momentarily block memory access to the DRAM before issuing a ZQ calibration command. After the command is issued, the MMDC will prevent data from being passed to/from the LPDDR2/3 until the required time interval has passed.

## **ERR010081: OTFAD: MDPC\_SRTAR[DMNC] always reads as zero**

**Description:** The OTFAD\_SRTAR[DMNC] = OTFAD\_MDPC\_SRTAR[8] control register bit always reads as zero even when asserted. The functionality associated with the DMNC bit (disabling the master number in the region hit logic) is not affected by the register read defect. Even though this bit is documented as R/W, it operates as if Read-as-Zero/Write.

**Workaround:** Always expect to read the OTFAD\_SRTAR[DMNC] = OTFAD\_MDPC\_SRTAR[8] control register bit as zero. Note: the write of 1'b1 to this bit has full functionality. Therefore, ignore this bit's documented R/W accessibility, and instead operate as Read-as-Zero/Write.

## **ERR050518: PAD\_RING: Potential pad drive high when I/O power supply is below specification voltage**

**Description:** When the I/O pad supply is below the specified minimum supply voltage as defined in the Data Sheet, the pad may unexpectedly drive high instead of maintaining a Hi-Z or weak internal pull state. The pad may drive high with a maximum drive strength defined in the Data Sheet. The pad maximum output voltage level is equal to the corresponding I/O pad supply voltage during the I/O pad supply ramp up or ramp down. An external 10K ohm pulldown resistor will not guarantee to keep the pad state low. Once the I/O pad supply level is above the minimum voltage level specifications, then the corresponding pad will function as expected to the defined reset state.

The affected I/O pads include all the GPIO pads of Pad Type “MultiVolt GPIO PAD” as defined in the IO Signal Table tab of the Signal Description Input Multiplexing Table (Excel file attached with the Reference Manual) for the I/O Power Supply Segments (VDD\_GPIO0, VDD\_GPIO1, VDD\_GPIO2, VDD\_HV\_IO\_ETH, VDD\_HV\_IO\_VIU0, VDD\_HV\_IO\_VIU1, VDD\_HV\_IO\_DIS, VDD\_HV\_IO\_FLTA) and also including the other Misc pads on VDD\_GPIO0: RESET, JCOMP, TMS, TCK, NMI, FCCU\_F0, and FCCU\_F1.

**Workaround:** Any external logic attached to the GPIO pads will need to be implemented such that the GPIO pad driven to a high level when the I/O power supply is below specification voltage does not cause any undesired effects. The EXT\_POR will only be negated once all the I/O power supply levels are within specification and thus may be used to indicate when the I/O pad functions are valid.

## **ERR009196: PCIE: 9000611337-RC Root Error Message Controls Not Fully Implemented.**

**Description:** The following error message controls of a Root Complex are not compliant with the PCIe Specification in “6.2.6. Error Message Controls”.

1) Signaled System Error bit in Type1 Configuration Space Headers.

- This register is never set when core (PCIe Controller) is Root Complex

2) Root Error Status Register field in Advanced Error Reporting Capability.

If receiving an Error Message, the core (PCIe Controller) lacks the following registers to control Root Error Status Register setting.

-SERR# Enable in Command register of Type1 Configuration Space Headers

-Correctable Error Reporting Enable in Device Control Register of PCI Express Capability

-Non-Fatal Error Reporting Enable in Device Control Register of PCI Express Capability

-Fatal Error Reporting Enable in Device Control Register of PCI Express Capability

3) Asserting Error Interrupt (cfg\_aer\_rc\_err\_int and cfg\_aer\_rc\_err\_msi of User I/F signals).

- If receiving an Error Message, the core (PCIe Controller) lacks the following registers to control cfg\_aer\_rc\_err\_int and cfg\_aer\_rc\_err\_msi.

-SERR# Enable in Command register of Type1 Configuration Space Headers

-Correctable Error Reporting Enable in Device Control Register of PCI Express Capability

-Non-Fatal Error Reporting Enable in Device Control Register of PCI Express Capability

-Fatal Error Reporting Enable in Device Control Register of PCI Express Capability

4) Asserting System Error (cfg\_sys\_err\_rc of User I/F signal).

- The core(PCIe Controller) lacks the following registers to control cfg\_sys\_err\_rc of User I/F signal.

-SERR# Enable in Command register of Type1 Configuration Space Headers -Correctable Error Reporting Enable in Device Control Register of PCI Express Capability -Non-Fatal Error Reporting Enable in Device Control Register of PCI Express Capability -Fatal Error Reporting Enable in Device Control Register of PCI Express Capability -Unsupported Request Reporting Enable in Device Control Register of PCI Express Capability -Each Mask field in Uncorrectable/Correctable Error Mask Register of Advanced Error Reporting Capability

**Workaround:** 1) Signaled System Error: If PCIe is configured as Root Complex (RC) and connected to a switch, then a system error will not be detected even though "Correctable/Non-Correctable, Fatal error bits of the PCIE\_RC\_RECR register are enabled. Consider alternate methods of signalling a system error if required.

2) Root Error Status Register field: Use only the "SERR Enable" bit of the "PCIE\_RC\_Interrupt\_Line\_Register" to enable or disable Root Error Status Register updates.

3) Asserting Error Interrupt: Use only the "SERR Enable" bit of the "PCIE\_RC\_Interrupt\_Line\_Register" to enable or disable asserting Error Interrupt.

4) Asserting System Error: use only the Correctable/Non-Correctable, Fatal error enables of the PCIE\_RC\_RECR register to enable or disable the System error.

### **ERR009193: PCIe: 9000680610-Bus and Device Number not Cleared on Hot Reset**

**Description:** The PCIe standard allows the system to perform a "hot reset" on request – this is also known as an In-band reset. This reset is generated under software control, can be propagated downstream only and is only generated by the Root Complex.

When the PCIe module is configured as Root Complex and a hot-reset occurs, the PCIe module does not clear the bus and device number registers and as a result, a completion generated by the core prior to the initial device Configuration Write Request may not contain zeroes in the Bus Number and Device Number fields. This violates the Completion Rule stated in the base spec: "If a Function must generate a Completion prior to the initial device Configuration Write Request, 0's must be entered into the Bus Number and Device Number fields".

**Workaround:** The PCIe base spec states that "The Completion ID field is not meaningful prior to the software initialization and configuration of the completing device (using at least one Configuration Write Request), and for this case the Requester must ignore the value returned in the Completer ID field." Since the Requester will be aware of the hot reset it can ignore the non-zero value of bus and device number in the completion before the Configuration Write.

### **ERR010136: PCIe: 9000783666- AXI Bridge Master (RC): B-Channel Write Response Queue Overflows**

**Description:** The PCIe standard has an ordering rule such that "posted must not pass posted" when it comes to messages. However, a condition can arise in the PCIe module where Message signaled interrupt (MSI) transactions can cause the internal write response queue to overflow and cause a subsequent MSI interrupt to be signaled before the write messages are complete. This error will only occur if the PCIe module is configured as Root Complex.

**Workaround:** If the PCIe module is configured in Root Complex mode then ensure that the PCIe EP device sends fewer than 8 TLP packets at a time to the RC. This will ensure that the MSI interrupt is received in accordance with PCIe ordering rules.

### **ERR009190: PCIe: 9000851378-Gen2: DSP Core Advertising Gen2 Speed Support Does Not Correctly Set Selectable Deemphasis Bit In TS2s Transmitted In Recovery.RcvrCfg State**

**Description:** In Recovery.RcvrCfg state the PCIe specification requires a downstream port (DSP) to:

- 1) Set the `select_deemphasis` variable equal to the Selectable De-emphasis field in the Link Control 2 register, and
- 2) If advertising Gen2 data rate; to set the Selectable De-emphasis bit in its transmitted TS2s identical to the `select_deemphasis` variable.

When all of the following conditions are true:

- 1) Link is in L0 state,
- 2) The Target Link Speed field (`PCIE_CAP_TARGET_LINK_SPEED`) of the Link Control 2 register (`LINK_CONTROL2_LINK_STATUS2_REG`) is set to Gen2 or Gen3
- 3) The core has entered Recovery state
- 4) Software overwrites the Target Link Speed field to Gen1,

The core will advertise Gen2 supported data rate but will not advertise the correct Selectable Deemphasis bit in its transmitted TS2s in Recovery.RcvrCfg state.

**Workaround:** Do not change the Target Link Speed field from Gen 2 rate to Gen1 rate after the PCIe module is configured. It is possible to use Gen 1 rate after a reset.

### **ERR050130: PIT: Temporary incorrect value reported in LMTR64H register in lifier mode**

**Description:** When the Programmable interrupt timer (PIT) module is used in lifier mode, timer 0 and timer 1 are chained and the timer load start value (`LDVAL0[TSV]` and `LDVAL1[TSV]`) are set according to the application need for both timers. When timer 0 current time value (`CVAL0[TVL]`) reaches 0x0 and subsequently reloads to `LDVAL0[TSV]`, then timer 1 `CVAL1[TVL]` should decrement by 0x1.

However this decrement does not occur until one cycle later, therefore a read of the PIT upper lifetime timer register (`LTMR64H`) is followed by a read of the PIT lower lifetime timer register (`LTMR64L`) at the instant when timer 0 has reloaded to `LDVAL0[TSV]` and timer 1 is yet to be decremented in next cycle then an incorrect timer value in `LTMR64H[LTH]` is expected.

**Workaround:** In lifier mode if the read value of `LTMR64L[LTL]` is equal to `LDVAL0[TSV]` then read both `LTMR64H` and `LTMR64L` registers one additional time to obtain the correct lifetime value.

## **ERR010170: QuadSPI: Insufficient read data may be received in the RX Data Buffer register**

**Description:** Data read from flash through QuadSPI using Peripheral Bus Interface (IPS) may return insufficient data in the RX Buffer Data register (QuadSPI\_RBDRn) when the read data size of a flash transaction is programmed to be greater than 32 bytes.

**Workaround:** For data size greater than 32 bytes, program the IP data transfer size in the IP configuration register (QuadSPI\_IPCR[IDATSZ]) to be in multiples of 8 bytes.

## **ERR009658: SPI: Inconsistent loading of shift register data into the receive FIFO following an overflow event**

**Description:** In the Serial Peripheral Interface (SPI) module, when both the receive FIFO and shift register are full (Receive FIFO Overflow Flag bit in Status Register is set (SR[RFOF] = 0b1)) and then the Clear Rx FIFO bit in Module Configuration Register (MCR[CLR\_RXF]) is asserted to clear the receive FIFO, shift register data is sometimes loaded into the receive FIFO after the clear operation completes.

**Workaround:** 1. Avoid a receive FIFO overflow condition (SR[RFOF] should never be 0b1). To do this, monitor the RX FIFO Counter field of the Status Register (SR[RXCTR]) which indicates the number of entries in receive FIFO and clear before the counter equals the FIFO depth.

2. Alternatively, after every receive FIFO clear operation (MCR[CLR\_RXF] = 0b1) following a receive FIFO overflow (SR[RFOF] = 0b1) scenario, perform a single read from receive FIFO and discard the read data.

## **ERR010400: SSE : Watchdog Error may not get asserted when the Watchdog Counter times out.**

**Description:** Safe State Engine (SSE) watchdog error interrupt flag SSE\_STATE [WD\_ERR] and SSE watchdog error interrupt line OUT3 may not assert when the watchdog counter expires.

**Workaround:** When the SSE\_STATE[WD\_ERR] occurs follow the below steps to ensure correct assertion of interrupt next time:

- 1) Disable the Watchdog Error interrupt : Write SSE\_CFG [WD\_EN] = 0
- 2) Clear the interrupt – Write 1 to clear on SSE\_STATE [WD\_ERR]
- 3) Read SSE\_STATE [WD\_ERR] to confirm correct clearing of the interrupt
- 4) Enable the Watchdog interrupt again : Write SSE\_CFG [WD\_EN] = 1
- 5) Exit the ISR

## **ERR010016: TMC: The RSZ register of the Cortex-M4 ETF shows an incorrect RAM size**

**Description:** The Trace Memory Controller (TMC) implements the Embedded Trace FIFO (ETF) configuration. The RAM Size register (RSZ) for the Cortex-M4 ETF incorrectly shows the size of the ETF RAM as 8KB, when actually it is 4KB. This also causes overflows during offline trace on Cortex-M4.

**Workaround:** Do not use offline trace capability for the Cortex-M4 since it might show overflows. No impact on online trace capability.

### **ERR011151: TPIU: Trace may get corrupted or stalled when functional reset occurs**

**Description:** If the TPIU\_TRACECLKIN source is programmed to a clock other than FIRC in MC\_CGM\_2 AUX4 Clock Selector and the clock divider value is configured to be  $\geq 2$ , there is a chance that the TPIU\_TRACECLKIN clock may receive a glitch while coming out of functional reset. If a trace is ongoing at that point of time, then there is a chance that the ongoing trace may get corrupted or the trace path may get stalled.

**Workaround:** 1. Select a clock source that does not require a divider (Divider value = 1) and runs at a speed equal or less than the maximum TPIU allowed frequency.  
2. A divided clock source may be used but in case the trace is stalled, the user will need to restart the debug session.

### **ERR050593: XRDC: DERRLOCn register may not capture the PAC and the MRC instance correctly in case of error violation**

**Description:** XRDC provides access protection mechanism to protect all on-chip resources (registers, volatile- and non volatile memory areas and all other on-chip peripherals) against manipulation and/or unauthorized access from external or internal.

A write attempt by a non core master outside the defined ranges shall lead to an exception in case XRDC region is defined to prevent non core master access. The chip will generate bus error when XRDC policies are violated.

When either a memory region controller or a peripheral access controller detects a domain access violation, the address and attribute information of the offending access is captured. The DERRLOCn read-only registers provide additional information by signaling the instance number of the submodule for which the access violation(s) occurred.

It is recommended that the exception handler for XRDC policy violation begins by reading the HWCFG1 register to determine its domainID. Next, it should use the just-retrieved domainID to index into the DERRLOCn array. The resulting DERRLOCn value is then examined to determine

the instance number of the reporting MRC and/or PAC submodule.

XRDC implements access control and signals violations correctly but if there are multiple violation after the first violation successively then the DERRLOCn register is not able to provide the correct instance number of reporting MRC and/or PAC submodule.

**Workaround:** To ascertain the correct instance reporting MRC and/or PAC submodule that was source of access violation, follow the below software sequence

Read DID from HWCFG1 register to know masters DID

Read all DERR\_W1\_n registers to determine which all submodules failed (check EST bit)

Read DERR\_W0\_n registers for which DERR\_W1\_n[EST] bit is set

Write to DERR\_W3\_n corresponding to failing submodules to clear error and rearm them for subsequent captures



## **ERR010436: ZipWire: SIPI can have only one initiator with one outstanding write frame at time**

**Description:** The Serial Inter-processor Interface (SIPI) module of the Zipwire interface only supports one initiator and one outstanding write frame at a time.

If a new write is initiated (by setting `SIPI_CCRn[WRT] = 0b1`, where `n` is the respective channel number for the transmission), or a new streaming write is initiated (by setting `SIPI_CCRn[ST] = 0b1`) with acknowledgement of a previous frame pending, then the initiator node may get a timeout error (indicated by `SIPI_ERR[TOEn]=0b1`). The previous write frame last byte may also be corrupted at the target node.

This also means that the target node cannot initiate a write transfer while the initiator node is in the process of a write transfer.

**Workaround:** The initiator should maintain only one outstanding write/streaming write frame to the target node at any one time.

The user must ensure that before initiating a new write request or initiating a new streaming write that it has received an acknowledgement for the previous write transaction (indicated by `SIPI_CSRn[ACKR] = 0b1`). The write acknowledgement interrupt can be enabled by setting `SIPI_CIRn[WAIE]=0b1`.

Implement a protocol that ensures both sides of the link cannot initiate a transfer at the same time. For example, a token-passing protocol could be implemented using the SIPI trigger command feature. Send a trigger command to pass the token to the other end of the link. Upon receipt of the trigger command, either initiate a write transfer if one is pending, or pass the token back by sending a trigger command. If a write transfer is initiated, wait until ACK is received and then send a trigger command to pass the token back. In this manner, if each side agrees only to initiate a transfer when it obtains the token, there will be no simultaneous transfers that can cause the problem described.

**How to Reach Us:****Home Page:**[nxp.com](http://nxp.com)**Web Support:**[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro,  $\mu$ Vision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2022 NXP B.V.

