TMS570LS20216, TMS570LS20206, TMS570LS10216, TMS570LS10206, TMS570LS10116, TMS570LS10106 Microcontroller **Silicon Revision A**

Silicon Errata

TEXAS INSTRUMENTS

Literature Number: SPNZ176D
June 2011

Contents

1 Introduction **5**
1.1 Device and Development-Support Tool Nomenclature 5
1.2 Revision Identification 6
2 Silicon Revision A Known Design Exceptions to Functional Specifications **7**
2.1 Silicon Revision A Known Design Exceptions to Functional Specifications 7

List of Figures

1 Example, Device Revision Code for TMS570LS20216 (PGE/ZWT Packages) 6

List of Tables

1 TMS570LS20216 Device Revision Codes 6
2 Silicon Revision A Advisory List 7

Silicon Errata
SPNZ176D–June 2011

TMS570LS Series Microcontroller **Silicon Revision A**

1 Introduction

This document describes the known exceptions to the functional specifications for the TMS570LS20216 device. For more detailed information on this device, see the device-specific data sheet:

- *TMS570LS Series 16/32-BIT RISC Flash Microcontroller* data manual (Literature Number SPNS141)

1.1 Device and Development-Support Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all devices and support tools. Each commercial family member has one of three prefixes: TMX, TMP, or TMS (e.g.,TMS570LS20216PGE). Texas Instruments recommends two of three possible prefix designators for its support tools: TMDX and TMDS. These prefixes represent evolutionary stages of product development from engineering prototypes (TMX/TMDX) through fully qualified production devices/tools (TMS/TMDS).

Device development evolutionary flow:

TMX — Experimental device that is not necessarily representative of the final device's electrical specifications.

TMP — Final silicon die that conforms to the device's electrical specifications but has not completed quality and reliability verification.

TMS — Fully-qualified production device.

Support tool development evolutionary flow:

TMDX — Development-support product that has not yet completed Texas Instruments internal qualification testing.

TMDS — Fully qualified development-support product.

TMX and TMP devices and TMDX development-support tools are shipped against the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

TMS devices and TMDS development-support tools have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

Predictions show that prototype devices (TMX or TMP) have a greater failure rate than the standard production devices. Texas Instruments recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are to be used.

TI device nomenclature also includes a suffix with the device family name. This suffix indicates the package type (for example, PGE), the temperature range (for example, "Blank" is the commercial temperature range), and the device speed range in megahertz.

1.2 Revision Identification

Figure 1 provides an example(s) of the TMS570LS series device markings. The device revision can be determined by the symbols marked on the top of the package.

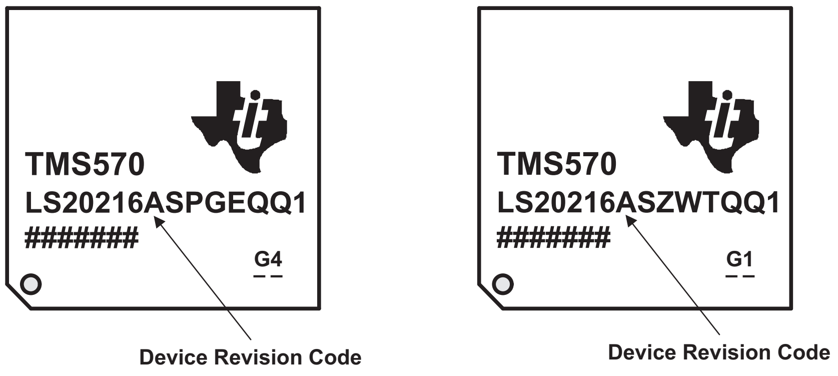

Figure 1. Example, Device Revision Code for TMS570LS20216 (PGE/ZWT Packages)

Silicon revision is identified by a device revision code. The code is of the format TMS570LS20216x, where "x" denotes the silicon revision. If x is "A" in the device part number, it represents silicon version A. Table 1 lists the information associated with each silicon revision.

Table 1. TMS570LS20216 Device Revision Codes

DEVICE PART NUMBER DEVICE REVISION CODE	SILICON REVISION	PART NUMBERS/COMMENTS
TMX570LS20216	Initial	This silicon revision is available as TMX *only*. **TMX**570LS20216
TMS570LS20216A	A	This silicon revision is available as TMX *only*. **TMX**570LS20216

2 Silicon Revision A Known Design Exceptions to Functional Specifications

This section describes the usage notes and advisories that apply to silicon revision A of the TMS570LS20216 device.

2.1 Silicon Revision A Known Design Exceptions to Functional Specifications

Table 2. Silicon Revision A Advisory List

Title **Page**

AHB_ACCES_PORT#3 —DAP AHB Access Port Might Fail To Return Slave Error On AHB Reset. 10
CCMR4#12 —CMPE Flag Not Set 10
CORTEX-R4#24 —In Standby Mode, The Clock May Be Gated Off Spuriously 11
CORTEX-R4#25 —Thumb MOVT, BFI And BFC Gives Wrong Result 12
CORTEX-R4#26 —Thumb STREXD Treated As NOP If Same Register Used For Both Source Operands 13
CORTEX-R4#27 —Debug Reset Does Not Reset DBGDSCR When In Standby Mode 13
CORTEX-R4#33 —Processor Can Deadlock When Debug Mode Enables Cleared 14
CORTEX-R4#46 —CP15 Auxiliary ID And Prefetch Instruction Accesses Are UNDEFINED 15
CORTEX-R4#54 —Instruction Causing A Data Watchpoint Match Incorrectly Traced 16
CORTEX-R4#55 —CPACR.ASEDIS And CPACR.D32DIS Incorrect When Configured With Floating Point 17
CORTEX-R4#56 —Debug Halt Exceptions Always Shown As Cancelling On ETM Interface 18
CORTEX-R4#57 —Conditional VMRS APSR_Nzcv, FPSCR May Evaluate With Incorrect Flags 19
CORTEX-R4#58 —DBGDSCR.Adadiscard Is Wrong When DBGDSCR.Dbgack Set 20
CORTEX-R4#59 —Missing Reset Exception On ETM Interface 21
CORTEX-R4#61 —Latched DTR-Full Flags Not Updated Correctly On DTR Access 22
CORTEX-R4#66 —Register Corruption During A Load-Multiple Instruction At An Exception Vector 23
CORTEX-R4#67 —Watchpoint On A Load Or Store Multiple May Be Missed. 24
CORTEX-R4#68 —Processor Not Waiting Long Enough Before Asserting TCM RAM Enables 25
CORTEX-R4#69 —Incorrect Sequential Access Indication On B0 And B1 TCM Ports 26
DCAN#22 —Incorrect Payload Stored In Mailbox 27
DMA#27 —DMA Requests Lost During Suspend Mode 28
DMM#11 —Control Signal Is Not Being Selected 28
DMM#12 —Programming DMMPC5 Register Is Not Clearing $\overline{\text{nDMMENA}}$ Pin In JTAG Interface 28
DMM#14 —Packet Error Is Not Being Generated 28
DMM#16 —BUSY Flag Not Set When DMM Starts 29
ETM_R4#3 —Accessing ATCLK Registers Causes APB Interface Lockup 30
ETM_R4#4 —AFREADYM Signal Not Asserted 31
ETM_R4#5 —Triggers Might Not Occur 32
ETM_R4#6 —ETM Address Range Comparator Does Not Match 33
ETM_R4#7 —Lock Access Can Be Modified When PADDRBG31 Is High 34
ETM_R4#8 —Power Down Status Register (PDSR) Reads As Zero 35
ETM_R4#9 —Cycle Count Is Up 7 Cycles Greater Than It Should Be On Debug Exit 36
ETM_R4#10 —Address Range Comparator Not Cleared By Coprocessor Access 36
ETM_R4#12 —Multiple Trigger Requests Occur 37
ETM_R4#13 —Consecutive Flushes Not Acknowledged 38
ETM_R4#16 —ETM-R4 Fails To Trace Vnt Packet For The Second Half Of SWP Instruction 39
F035_FLEPBANK#1 —Redundant column shows much higher programmed Vt in compressed programming . 40
F035_FLEPBANK#2 —High source line leakage in F035 platform devices(FLEP bank) 41
FLEXRAY#16 —Detected Coding Error Not Reported 41
FLEXRAY#20 —E-Ray Erroneously Uses Incorrect 42
FLEXRAY#21 —Temporal Deviation Is Different For Channel A And B 42
FLEXRAY#22 —E-Ray Acting As Sync Node Does Ignores Own Sync Frame For Counting Number Of Sync Frames 42

Table 2. Silicon Revision A Advisory List (continued)

FLEXRAY#23 —Temporal Deviation Value Is Set To The Measured Value Of The Previous Non Sync Frame 43
FLEXRAY#24 —Incorrect Secondary Time Reference Point (STRP) Stored 44
FLEXRAY#25 —E-Ray Erroneously Updates Register ACS 44
FLEXRAY#26 —Cycle Counter Value MTCCV.CCV Returns Incorrect Value 44
FLEXRAY#27 —Data Transfer From Message RAM To Transient Buffer Not Started 45
FLEXRAY#28 —CC Unable To Transit From STARTUP To NORMAL_ACTIVE State 45
FLEXRAY#29 —EIR.MHF Error Interrupt Flag Set 45
FLEXRAY#30 —E-Ray Stuck In Startup State 46
FLEXRAY#31 —Protocol Engine Reads Extra Words Of Its Transient Buffer 46
FLEXRAY#32 —Incorrect Content In CCSV.......... 46
FLEXRAY#33 —Wakeup Pattern Transmitted n bit Times Early 47
FLEXRAY#34 —Data In TBF Incorrectly Overwritten 47
FLEXRAY#35 —Command CCSV.RCA and CCSV.PSL Inconsistent 47
FLEXRAY#36 —CLEAR_RAMS Command Does Not Clear The 1st RAM Word 48
FLEXRAY#37 —Wrong Message Buffer Used For Transmission During Next Cycle.......... 48
FLEXRAY#38 —Frame Received Is Corrupted 49
FLEXRAY#39 —Irregular Sync Frame List Exported In State Coldstart_Gap 49
FLEXRAY#40 —After Detecting Low Level Beyond gdWakeupSymbolRxWindow, Node Completes Wakeup State.... 50
FLEXRAY#44 —Register RCV Displays Wrong Value.......... 50
FLEXRAY#45 —Valid Sync Frame Received Is Ignored 50
FLEXRAY#46 —Sync Frame Overflow Flag EIR.SFO Set If Slot Counter Is Greater Than 1024 51
FLEXRAY#47 —Startup Frame Added Erroneously After Reception Of More Than gSyncNodeMax Sync Frames 51
FLEXRAY#48 —Initial Rate Correction Value Of An Integrating Node Is Zero 51
FLEXRAY#49 —Incorrect Rate And/Or Offset Correction 52
FLEXRAY#50 —SFS.MRCS Flag Set Erroneously.......... 52
FLEXRAY#51 —Rate Correction Set To Zero.......... 52
FLEXRAY#52 —WUS Generates Redundant SIR.WUPA/B Events 53
FLEXRAY#53 —Mismatch Between Spec & RTL For POCS[5:0] Value Of CCSV 53
FLEXRAY#55 —FlexRay Timer Interrupts Lost On Level Triggered Interrupt Modules 54
FLEXRAY#58 —Erroneous Cycle Offset During Startup 55
FLEXRAY#59 —First WUS Following Received Valid WUP May Be Ignored 55
FLEXRAY#60 —READY Command Accepted In READY State 56
FLEXRAY#61 —Slot Status Vpoc!SlotMode Is Reset Immediately When Entering HALT State 56
FLASH_WRAPPER#115 —Flash Clock Cycle is Late 57
FLASH WRAPPER#151 —Clearing Error Status Bit Blocks New Errors.......... 57
FMZPLL#17 —FBSLIP And/Or RFSLIP Inadvertently Set On Power Up 58
FMZPLL#18 —PLL May Fail To Lock Or May Show PLL SLIP Immediately After Locking 58
FMZPLL#27 —Duty Cycle Incorrect When Output PLLCLK Divider Ratio Changed.......... 59
FTU#6 —FTU Registers Are Set Before TU_VBM_SDONE Is Asserted 59
FTU#8 —Transfer Unit Not Disabled On MPV Error 60
FTU#14 —Parity Calculation Occurs Even If Parity Protection For Transfer Configuration RAM(TCR) Is Off............ 60
FTU#19 —TCCOx Flag Clearing Masked 61
GCM#47 —PLL Can Be Running Too Fast After Lock Period 61
HTU#52 —VBUSP Register Interface Of HTU Does Not Have EMUDBG Port 61
HTU#53 —Buffer Full Flag Updated Incorrectly 62
LIN#46 —Super-Fractional Divider Table Mismatch For Synch Field And ID Field.......... 62
LIN#47 —Data Corrupted By New Header 62
LIN#48 —Header Not Received By Slave 63
LIN#49 —Data Transmitted Correctly But First Byte Of Next Set Of Data Is Corrupted 63

Table 2. Silicon Revision A Advisory List (continued)

LIN#50, LIN#65 —LIN Slave Mode Issues 63
LIN#51 —Slave Not Transmitting Response To The First Header Following Wakeup 64
LIN#54 —LIN TX Pin Continues Toggling Improperly 64
LIN#57 —LIN Rejects First Byte Followed By Data Byte With A False Start Bit. 65
LIN#58 —Incomplete Synch Field Reception Results In Loss Of Next Header 65
LIN#59 —LIN Transmission Incorrect 66
LIN#60 —Start Bit Transmitted Within One Vclk During Extended Frame Transmission In Single Buffer Mode 66
LIN#61 —TOA3WUP Flagged When New Header Is Received During Wait State 66
LIN#62 —FE Restarts Reception During Checksum Byte 67
LIN#63 —LIN Timeout Counters Should Stop During Power Down Mode 67
MIBSPI#93 —RELOAD Bit In TICKCNT Register Not Cleared Automatically 68
MIBSPI#109 —BITERR Unreliable In Slave mode 68
MIBSPI#110 —Multibuffer Slave In 3 or 4 Pin Mode Transmits Data Incorrectly 68
MIBSPI#111 —Data Length Error Is Generated Repeatedly In Slave Mode 69
MIBSPI#117 —Write To Unimplemented Buffer Overwriting The Corresponding Implemented Buffer 69
MIBSPI#118 —TXPIN Pin Has No Reset In MIBSPI 69
MIBSPI#127 —DMACNTLEN Writable in Privilege and User Modes 70
NHET#31 —MCMP Instruction Branches To Conditional Address When NAF Is Zero 70
NHET#32 —SCMP Instruction Not Updating Cout_Prv Correctly 71
NHET#36 —Z Flag Not Set 72
PBIST#3 —PBIST Module Functionality Affected When VCLK Equals HCLK 72
RTP#35 —RTP Stops Tracing Data 73
SYS#46 —Switching Not Qualified With Clock Source On And Clock Source Valid 74
SYS#103 —Pulse extension needed to ensure device reset. 75
TPIU#2 —TPIU Frame Synchronization Packets Not Output As 76
TPIU#4 —Serializer Not Returned To The Same State As From Reset 77
TPIU#5 —TPIU Formatter Not Outputting Existing ID On Trace Reenable 77
TPIU#6 —CSTPIU Unable To Disable Pattern Generator If Trace Data Is Output On Stop 78
TPIU#7 —TPIU Unsynchronized Clock Domain Crossing In CSTPIU 79
TC RAM WRAPPER#27 —Read To RAMUERRADDR Or RAMPERRADDR Causes Registers To Clear Twice 79
TC RAM WRAPPER#28 —Single Error Generated Even When RAMTHRESHOLD Register Programmed To Zero 80
TMS570LS#20 —BSR Logic Overrides INENA Signal For All IO Buffers Except When Test Logic Reset 80
TMS570LS#23 —Write Access To Illegal Address Region Of MCRC Not Generating CPU Abort 80
TMS570LS#29 —JTAG Connectivity Can Be Lost 81
TMS570LS#30 —BSR Logic Should Be In Bypass Mode 81

AHB_ACCES_PORT#3 *DAP AHB Access Port Might Fail To Return Slave Error On AHB Reset.*

Revision(s) Affected	Revision A and earlier
Details	If an AHB reset occurs ($\overline{\text{HRESETn}}$ is asserted LOW) while the AHB-AP is performing an access on the AHB bus, the AHB-AP should return a slave error back to the SWJ-DP. Instead, the AHB-AP might indicate that the access completed successfully, and return unpredictable data if the access was a read. Conditions: $\overline{\text{HRESETn}}$ is asserted LOW on a specific cycle while the AHB-AP is completing an access on the AHB bus. Implications: This will make it harder to identify cases where the AHB bus is being reset. However, this should not affect most usage models because it is not generally possible to debug over a bus when it is likely to be reset.
Workaround(s)	Avoid performing debug operations while the AHB bus might be reset. Resets usually occur following powerdown, and so this can usually be achieved through the use of features that prevent powerdown during debug

CCMR4#12 *CMPE Flag Not Set*

Revision(s) Affected	Revision A and earlier
Details	The CMPE flag is not set if there is a lock step compare error during Selftest Error Forcing mode. During the 1cycle window of the Selftest Error Forcing mode, if there is a compare error in the lock step compare logic the CMPE flag does not get set. However, the error_compare output signal from CCMR4 is active, the flag in the ESM is set and the ERROR pin is activated. Generally the scenario is unlikely that the signals are not in step for only 1 cycle.
Workaround(s)	None

CORTEX-R4#24 *In Standby Mode, The Clock May Be Gated Off Spuriously*

Revision(s) Affected Revision A and earlier

Details The Cortex-R4 processor implements Standby Mode, entered by the execution of a Wait-For-Interrupt (WFI) instruction. In Standby Mode, the processor stops executing instructions and once quiescent, stops the clock to most of the logic and asserts the STANDBYWFI output. One use for this output is as an indication to a power controller that the processor is quiescent and that a lower power mode can be entered, for example, Dormant Mode or Shutdown Mode. Because of this it is possible for the processor to temporarily gate off its clocks and assert STANDBYWFI for up to three FREECLKIN cycles before the AXIslave interface activity is complete. If the AXIslave interface is signaling a valid response when the clock is turned off, this response can be accepted by the AXI system without the AXI-slave recognizing the handshake. Alternatively, if the power controller responds to STANDBYWFI by removing the power from the processor, any active transaction will be lost. In either case, the loss of an AXI transaction will typically cause the AXI master generating that transaction to deadlock. This occurs when transitions on various signals from CORTEXR4/u_cortexr4noram cause a switching glitch on the CORTEXR4/u_cortexr4clk/can_stop_clk signal which is captured by the synchronizer CORTEXR4/u_cortexr4clk/u_cr4cell_sync0. The source signals are clocked by the gated version of CLKIN, while the synchronizer is clocked by FREECLKIN. These clocks are identical at the periphery of the processor. Therefore, whether or not the glitch is captured depends on the timing of the logic and the difference between the clock insertion delays of FREECLKIN and the gated version of CLKIN.

Conditions:

- There are one or more active transactions to the Cortex-R4 AXI-slave interface
- A WFI instruction is executed to quiesce the Cortex-R4 processor and put it into Standby Mode
- When STANDBYWFI is asserted, the power controller removes power from the Cortex-R4 processor logic, e.g. for Dormant Mode, but the master device for the active transactions remains powered up
- When the processor clock is gated, a response to the outstanding transaction is made by the AXI-slave interface (RVALIDS or BVALIDS is asserted) which is accepted by the AXI system within three cycles (RREADYS or BREADYS is asserted)

Implications: If this occurs, it will either lead to AXI transaction handshakes occurring when the processor cannot respond to them, or, if the power controller responds quickly to STANDBYWFI, the power being removed from the processor while an AXI transaction is still in flight. In either case the transaction will be lost, which will typically cause the requesting master to deadlock.

Workaround(s) None

CORTEX-R4#25 *Thumb MOVT, BFI And BFC Gives Wrong Result*

Revision(s) Affected Revision A and earlier

Details The Cortex-R4F processor can, in some cases, issue and execute two instructions simultaneously for improved throughput. This is known as dual-issuing. Because of this erratum, some BFC, BFI and MOVT instructions will give an incorrect result when dual-issued with a preceding VMOV instruction which transfers 32-bits of data from the VFP register file to an ARM core register.

Conditions:

- The Cortex-R4 processor is implemented with floating point logic, i.e., Cortex-R4F
- The VFP logic (cp10 and cp11) is enabled in the Coprocessor Access Register for the current privilege level
- The VFP logic is enabled by means of the EN bit in the FPEXC register
- The processor is in Thumb state
- The program contains a VMOV instruction which transfers either one single-precision register or half of a double-precision register from the VFP register file to the ARM core register file
- Immediately following the VMOV instruction is a MOVT, BFI or BFC instruction
- The destination register, Rd, for the VMOV instruction is the same as the destination register, Rd, for the MOVT, BFI or BFC instruction
- When the VMOV instruction is being decoded, the second instruction has also been fetched so that it can be decoded and issued simultaneously
- Dual-issue case F3 is enabled, that is, bit [18] of the Secondary Auxiliary Control Register is 0

Implications: If the occurs, the result of the MOVT, BFI or BFC instruction is incorrect. The result is computed as if the VMOV instruction had not been executed.

Workaround(s) This can be avoided by disabling case F3 dual-issuing. Set bit [18] of the Secondary Auxiliary Control Register. Note: this workaround will have a small impact on the instruction throughput of the processor. The impact will depend on what code is being executed, but ARM has not been able to measure any noticeable impact during internal trials.

CORTEX-R4#26 *Thumb STREXD Treated As NOP If Same Register Used For Both Source Operands*

Revision(s) Affected Revision A and earlier

Details The ARM Architecture permits the Thumb STREXD instruction to be encoded with the same register used for both transfer registers (Rt and Rt2). Because of this erratum, the Cortex-R4 processor treats such an encoding as UNPREDICTABLE and executes it as a NOP. Conditions This occurs when the processor is in Thumb state and a STREXD instruction is executed which has Rt = Rt2. This instruction was new in ARM Architecture version 7 (ARMv7). It is not present in ARMv6T2 or other earlier architecture versions.

Implications: If this occurs the destination register, Rd, which indicates the status of the instruction, is not updated and no memory transaction takes place. If the software is attempting to perform an exclusive read-modify-write sequence, then it might either incorrectly complete without memory being written, or loop forever attempting to complete the sequence.

Workaround(s) This can be avoided by using two different registers for the data to be transferred by a STREXD instruction. This may involve copying the data in the transfer register to a second, different register for use by the STREXD.

CORTEX-R4#27 *Debug Reset Does Not Reset DBGDSCR When In Standby Mode*

Revision(s) Affected Revision A and earlier

Details The debug reset input, PRESETDBGn, resets the processor's debug registers as specified in the ARMv7R Architecture. Because of this erratum, when the processor is in Standby Mode and the clock has been gated off, PRESETDBGn fails to reset the Debug Status and Control Register (DBGDSCR).

Conditions:

- The DBGDSCR register has been written so that its contents differ from the reset values (most fields in this register reset to zero, though a few are UNKNOWN at reset)
- The processor is in Standby Mode, and the clocks have been gated off, that is STANDBYWFI is asserted
- The debug reset, PRESETDBGn, is asserted and deasserted while the processor clocks remain gated off

The debug reset is commonly used to set the debug registers to a known state when a debugger is attached to the target processor.

Implications: If this occurs, then after the reset the DBGDSCR register contains the values that it had before reset rather than the reset values. If the debugger relies on the reset values, then it may cause erroneous debug of the processor. For example, the DBGDSCR contains the ExtDCCmode field which controls the Data Communications Channel (DCC) access mode. If this field was previously set to Fast mode but the debugger assumes that it is in Non-blocking mode (the reset value) then debugger accesses to the DCC will cause the processor to execute instructions which were not expected.

Workaround(s) This can be avoided by a workaround in the debug control software. Whenever the debugger (or other software) generates a debug reset, follow this with a write of zero to the DBGDSCR which forces all the fields to their reset values.

CORTEX-R4#33 ***Processor Can Deadlock When Debug Mode Enables Cleared***

Revision(s) Affected Revision A and earlier

Details The Cortex-R4 processor supports two different debugging modes: Halt-mode and Monitor-mode, or both modes can be disabled. Bits [15:14] in the Debug Status and Control Register (DBGDSCR) control which, if any, mode is enabled. Debug events can be generated by the breakpoint unit, matching instructions executed. If there are active breakpoints at the time when the debugging mode is changed, this can cause the processor to deadlock.

Conditions:

- At least one breakpoint is programmed and active
- Either halt-mode debugging or monitor mode debugging is enabled and the debug enable input, DBGEN is HIGH
- An instruction which matches a breakpoint is fetched and executed
- After the instruction is fetched, but before it has completed execution, both halt-mode and monitor-mode debugging are disabled by means of a write to DBGDSCR

Implications: Changing the debugging mode while breakpoints or watchpoints are active is UNPREDICTABLE. Therefore the above conditions cannot be met in normal (recommended) operation of the processor. However, if the conditions do occur, then the processor will deadlock, which is outside the bounds of permitted UNPREDICTABLE behavior.

Workaround(s) None

CORTEX-R4#46 *CP15 Auxiliary ID And Prefetch Instruction Accesses Are UNDEFINED*

Revision(s) Affected Revision A and earlier

Details The ARMv7-R architecture requires implementation in CP15 of the following two features:

1. An Auxiliary ID Register (AIDR), which can be read in privileged modes, and the contents and format of which are IMPLEMENTATION DEFINED.
2. The operation to prefetch an instruction by MVA, as defined in the ARMv6 architecture, to be executed as a NOP.

Because of this erratum, both of these CP15 accesses generate an UNDEFINED exception on Cortex-R4.

Conditions: Either of the following instructions is executed in a privileged mode:

- - MRC p15,1,<Rt>,c0,c0,7 ; Read IMPLEMENTATION DEFINED Auxiliary ID Register
- - MCR p15,0,<Rt>,c7,c13,1 ; NOP, was Prefetch instruction by MVA in ARMv6

Implications: If software attempts to read the AIDR as part of its process of identifying the processor on which it is running, it will encounter an unexpected UNDEFINED exception. If software attempts to execute an instruction prefetch using the ARMv6 CP15 prefetch operation, it will encounter an unexpected UNDEFINED exception.

Workaround(s) In the first case, because the contents of the AIDR are IMPLEMENTATION DEFINED, it must always be read in conjunction with the Main ID Register (MIDR). A simple way of avoiding this would be to read and analyze the MIDR first and, if this indicates that the processor is Cortex-R4 skip the read of the AIDR. In the second case, any instruction to prefetch an instruction by MVA should be removed or replaced by a true NOP instruction.

CORTEX-R4#54 *Instruction Causing A Data Watchpoint Match Incorrectly Traced*

Revision(s) Affected Revision A and earlier

Details When tracing a program execution using the ETM, an extra instruction is traced if a data watchpoint matches and causes a debug exception. The extra instruction that is traced is the instruction which caused the data watchpoint to match.

Conditions: The extra instruction is traced if the following occurs:

- A hardware watchpoint matches
- DBGEN is asserted
- Monitor debug-mode is enabled

Implications: Trace analysis tools will incorrectly consider the instruction causing a data watchpoint match to have executed. If any of the ETM address comparators are configured to match on address of the instruction and the exact match bit is set, the comparator will incorrectly fire. This might cause an unexpected trigger or change in any ETM resources which are configured to be sensitive to the address comparator.

Workaround(s) If a data abort exception is taken and the cause was a data watchpoint, the instruction traced immediately before the entry to the exception handler was not executed and must be discarded.

CORTEX-R4#55 *CPACR.ASEDIS And CPACR.D32DIS Incorrect When Configured With Floating Point*

Revision(s) Affected

Revision A and earlier

Details

In implementations of the VFPv3-D16 architecture that do not also implement the Advanced SIMD functionality, the ASEDIS and D32DIS bits, (bits [31] and [30] respectively of the Coprocessor Access Control Register (CPACR)) are read-as-one/writes ignored (RAO/WI). This indicates that Advanced SIMD functionality and registers D16-D31 are disabled. Because of this erratum, these bits read zero in implementations of Cortex- R4F which include the floating-point unit. When the floating point unit is not included, all bits of the CPACR correctly read-as-zero (RAZ).

Conditions: In an implementation of Cortex-R4F with the floating-point unit included, the CPACR is read.

Implications: If software uses the CPACR to determine if Advanced SIMD functionality and registers D16-D32 are available, it will incorrectly determine that they are. Any attempt to use these features will lead to an unexpected UNDEFINED exception.

Workaround(s)

ecause the Cortex-R4F processor never includes Advanced SIMD functionality or registers D16-D31, this can be correctly determined by reading the part number from one of the ID registers rather than examining ASEDIS and D32DIS in the CPACR.

CORTEX-R4#56 *Debug Halt Exceptions Always Shown As Cancelling On ETM Interface*

Revision(s) Affected Revision A and earlier

Details When tracing program execution using the ETM, the instruction executed immediately before an external debug halt request will not be traced. If this instruction is a partially-executed multi-cycle instruction, for example a load-multiple which has transferred some, but not all of its registers, then this is correct. However, when the instruction has completed execution, this is erroneous.

Conditions: The erratum occurs if:

- Tracing is enabled
- The processor enters debug halt state either because EDBGRQ was asserted or because of a write to the DRCR,
- At the time it stopped executing instructions in order to enter debug halt state, the processor was not part-way through executing a load or store multiple instruction

Implications: Trace analysis tools will incorrectly consider the instruction executed immediately before the debug halt state entry to have not executed. If any of the ETM address comparators are configured to match on address of the instruction and the exact match bit is set, the comparator will fail to fire. This might have a knock-on effect to an expected trigger event or change in any ETM resources which are configured to be sensitive to the address comparator.

Workaround(s) None

CORTEX-R4#57 *Conditional VMRS APSR_Nzcv, FPSCR May Evaluate With Incorrect Flags*

Revision(s) Affected

Revision A and earlier

Details

Under certain circumstances, a conditional VMRS APSR_nzcv, FPSCR instruction may evaluate its condition codes using the wrong flags and incorrectly execute or not execute.

Conditions: The erratum requires the following sequence of instructions in ARMstate: - VMRS<c> APSR_nzcv, FPSCR (formerly FMSTAT<c>), where the condition on the instruction is not always.

- A flag-setting integer multiply or multiply and accumulate instruction (e.g. MULS)
- A single-precision floating-point multiply-accumulate (FP-MAC) instruction (e.g. VMLA), timed such that the accumulate operation is inserted into the pipeline in the cycle in which the VMRS instruction is first attempted to be issued.

  To meet the above timing requirements, the VMRS instruction must be three pipeline stages behind the FPMAC. Depending on the rate in which the instructions are fetched, interlocks within this sequence and dual-issuing, this can be up to three other instructions between this pair, plus the multiply. Out-of-order completion of FP-MAC instructions must be enabled.

Implications: If this erratum occurs, the VMRS instruction will pass or fail its condition codes incorrectly, and this will appear in any trace produced by the ETM. This can corrupt the N, Z, C, V flag values in the CPSR which will typically affect the program flow.

Workaround(s)

This erratum can be avoided by disabling out-of-order single-precision floating point multiply-accumulate (SPMAC) instruction completion. Set DOOFMACS, bit [16] in the Secondary Auxiliary Control Register. This will have the side-effect of reducing the performance of SP-MAC operations, though the impact will depend on how these instructions are used in your code.

CORTEX-R4#58 *DBGDSCR.Adadiscard Is Wrong When DBGDSCR.Dbgack Set*

Revision(s) Affected Revision A and earlier

Details When the DBGDSCR.ADAdiscard bit is set, asynchronous data aborts are discarded, except for setting the DBGDSCR.ADAbort sticky flag. The Cortex-R4 processor ensures that all possible outstanding asynchronous data aborts have been recognised before it enters debug halt state. The flag is immediately on entry to debug halt state to indicate that the debugger does not need to take any further action to determine whether all possible outstanding asynchronous aborts have been recognized. Because of this erratum, the Cortex-R4 processor also sets the DBGDSCR.ADAdiscard bit when the DBGDSCR.DBGack bit is set. This can cause the DBGDSCR.ADAbort bit to become set when the processor is not in debug halt state, and it is not cleared when the processor enters debug halt state. However, the processor does not discard the abort. It is pending or generates an exception as normal.

Conditions:

- The processor is not in debug halt state
- The DBGDSCR.DBGack bit is set
- An asynchronous data abort (for example, SLVERR response to a store to Normal-type memory) is recognized

NOTE: it is not expected that DBGDSCR.DBGack will be set in any Cortex-R4 system

If this erratum occurs, and the processor subsequently enters debug halt state, the DBGDSCR.ADAbort bit will be set, when in fact no asynchronous data abort has occurred in debug state. Before exiting debug state, the debugger will check this bit and will typically treat it as an error. If no other asynchronous data abort has occurred in debug state, this is a false error.

Workaround(s) None

CORTEX-R4#59 *Missing Reset Exception On ETM Interface*

Revision(s) Affected Revision A and earlier

Details When tracing program execution through soft reset using the ETM, a reset exception might not be traced. Two sets of conditions cause this erratum to occur.

Conditions set 1: The first instruction at the reset vector is a load-store multiple instruction. This should not happen because after reset there is no base register whose contents can be guaranteed.

Conditions set 2: The processor enters debug halt state before executing the instruction at the reset vector due to the following conditions being true when the processor comes out of soft reset with nCPUHALTm HIGH or when nCPUHALTm is first de-asserted after soft reset:

- The Halting mode debug enable bit (bit[14] in the DBGDSCR) is set
  AND
- The DBGENm input pin is asserted
  AND
- A debug event is triggered due to: - The Halt request bit in the DBGDRCR being set
  OR
- The EDBGRQm input pin being asserted

Implications:
For conditions set 1:

if the last instruction before the reset was an indirect branch instruction, a branch to the reset vector will be traced but not marked with a reset exception. If the last instruction before the reset was not an indirect branch, a trace analysis tool might incorrectly infer the execution of one or more instructions after the instruction before the reset until the processor executes an indirect branch. Typically, the instruction at the reset vector is an indirect branch and therefore this error is limited to one or two instructions. The address comparators in the ETM are unaffected unless an address comparison was set on the address of the instruction just before the reset occurs and the exact match bit was set for comparison. In this case the instruction is always considered to be executed.

For conditions set 2:

The reset exception is not traced and only the debug exception is traced. Any ETM address comparator configured to match on the instruction at the reset vector will not match.

Workaround(s) None

CORTEX-R4#61 *Latched DTR-Full Flags Not Updated Correctly On DTR Access*

Revision(s) Affected Revision A and earlier

Details When the debug Data Transfer Register (DTR) is in non-blocking mode, the latched DTR-full flags (RXfull_l and TXfull_l) record the state of the DTR registers as observed by the debugger and control the flow of data to and from the debugger to prevent race hazards. For example, when the target reads data from DBGDTRRXint, the associated flag RXfull is cleared to indicate that the register has been drained, but the latched value Rxfull_l remains set. Subsequent debugger writes to DBGDTRRXext are ignored because RXfull_l is set. RXfull_l is updated from RXfull when the debugger reads DBGDSCRext such that a debugger write to DBGDTRRXext will only succeed after the debugger has observed that the register is empty. The ARMv7 debug architecture requires that RXfull_l be updated when the debugger reads DBGDSCRext and when it writes DBGDTRRXext. Similarly, TXfull_l must be updated when the debugger reads DBGDSCRext and when it reads DBGDTRTXext. In Cortex-R4, because of this erratum, RXfull_l and TXfull_l are only updated when the debugger reads DBGDSCRext.

onditions: The DTR is in non-blocking mode, that is, DBGDSCR.ExtDCCmode is set to 0b00 and EITHER:

- The debugger reads DBGDSCRext which shows that RXfull is zero, that is, DBGDTRRX is empty.
- The debugger writes data to DBGDTRRXext.
- Without first reading the DBGDSCRext, and before the processor has read from DBGDTRRXint, the debugger performs another write to DBGDTRRXext.

OR

- The debugger reads DBGDSCRext which shows that TXfull is one, that is, DBGDTRTX is full.
- The debugger reads data from DBGDTRTXext,
- The processor writes new data into DBGDTRTXint,
- Without first reading the DBGDSCRext, the debugger performs another read from DBGDTRTXext.

Implications: The ARMv7 debug architecture requires the debugger to read the DBGDSCRext before attempting to transfer data via the DTR when in non-blocking mode. This erratum only has implications for debuggers that violate this requirement. If the erratum occurs via data transfer, data loss may occur. The architecture requires that data transfer never occur.

Workaround(s) None

CORTEX-R4#66 *Register Corruption During A Load-Multiple Instruction At An Exception Vector*

Revision(s) Affected Revision A and earlier

Details Under certain circumstances, a load multiple instruction can cause corruption of a general purpose register.

Conditions: All the following conditions are required for this erratum to occur:

- A UDIV or SDIV instruction is executed with out-of-order completion of divides enabled
- A multi-cycle instruction is partially executed before being interrupted by either an IRQ, FIQ or imprecise abort. In this case, a multi-cycle instruction can be any of the following:
  - LDM/STM that transfers 3 or more registers
  - LDM/STM that transfers 2 registers to an unaligned address without write back
  - LDM/STM that transfers 2 registers to an aligned address with write back
  - TBB/TBH
- A load multiple instruction is executed as the first instruction of the exception handler
- The load multiple instruction itself is interrupted either by an IRQ, FIQ, imprecise abort or external debug halt request. This erratum is very timing sensitive and requires the UDIV or SDIV to complete when the load multiple is in the Issue stage of the CPU pipeline. The register that is corrupted is not necessarily related to the load-multiple instruction and will depend on the state in the CPU store pipeline when the UDIV or SDIV completes.

Implications: For practical systems, it is not expected that an interruptible LDM will be executed as the first instruction of an exception handler, because the handler is usually required to save the registers of the interrupted context. Therefore, it is not expected that this erratum has any implications for practical systems. If the situation of the erratum occurs it will result in the corruption of the register bank state and could cause a fatal failure if the corrupted register is subsequently read before being written.

Workaround(s) To work around this erratum, set bit [7] of the Auxiliary Control Register to disable out-of-order completion for divide instructions. Code performance may be reduced depending on how often divide operations are used.

CORTEX-R4#67 *Watchpoint On A Load Or Store Multiple May Be Missed.*

Revision(s) Affected Revision A and earlier

Details The Cortex-R4 supports synchronous watchpoints. This implies that for load and store multiples, a watchpoint on any memory access will generate a debug event on the instruction itself. Due to this erratum, certain watchpoint hits on multiples will not generate a debug event.

Conditions: All the following conditions are required for this erratum to occur:

- A load or store multiple instruction is executed with at least 5 registers in the register list
- The address range accessed corresponds to Strongly-Ordered or Device memory
- A watchpoint match is generated for an access that does not correspond to either the first two or the last two registers in the list.

Under these conditions the processor will lose the watchpoint. Note that for a "store multiple" instruction, the conditions are also affected by pipeline state making them timing sensitive.

Implications: Due to this erratum, a debugger may not be able to correctly watch accesses made to Device or Strongly-ordered memory. The ARM architecture recommends that watchpoints should not be set on individual Device or Strongly-ordered addresses that can be accessed as part of a load or store multiple. Instead, it recommends the use of the address range masking functionality provided to set watchpoints on an entire region, ensuring that the watchpoint event will be seen on the first access of a load or store multiple to this region. If this recommendation is followed, this erratum will not occur.

Workaround(s) None

CORTEX-R4#68 *Processor Not Waiting Long Enough Before Asserting TCM RAM Enables*

Revision(s) Affected Revision A and earlier

Details The Cortex-R4 processor supports low-power implementations by providing a sleep mode that can be entered on execution of a Wait-For-Interrupt (WFI) instruction. In this sleep mode the internal processor clock is gated and the STANDBYWFI output signal is asserted to indicate to the external system that the processor is in sleep mode. On receipt of an appropriate wake-up event, the processor will de-assert STANDBYWFI, turn on its internal clock and resume execution. The Cortex-R4 processor was specified to ensure that after deasserting STANDBYWFI on wake-up, no accesses will be performed to the TCM memories for 3 clock cycles. This enables these memories to be put into low-power modes when the processor is sleeping and affords them adequate time on wake-up to prepare for the first processor access. Due to this erratum, this guarantee is not met and the processor may perform TCM accesses without waiting for three cycles after waking up from sleep mode.

Conditions: The conditions for this erratum are very timing dependent and require all of the following:

- The processor must be configured to include TCMs
- The processor enters sleep mode due to the execution of the WFI instruction
- The instruction after the WFI is located in TCM
- Any of the following events are raised to the processor in the 2-3 cycle window between the processor pipeline committing to enter sleep state and the STANDBYWFI signal being asserted:
  - An interrupt raised on the IRQ pin or FIQ pin
  - An external debug request raised on the EDBGRQ pin
  - A request to enter halt state by an external debugger setting bit[0] of the DBGDRCR

Implications: The implications of this erratum are entirely system dependent. If the TCMs RAMs implement low-power states that require three cycles or more of wake-up time, then this erratum may be stimulated. The first processor access after wake-up may be missed or return incorrect data causing fatal system failure. If no such low-power modes are implemented, then this erratum will not have any effect.

Workaround(s) To avoid this erratum, put the TCM RAMs into a low-power state only after observing STANDBYWFI asserted for at least 2 consecutive clock cycles.

CORTEX-R4#69 *Incorrect Sequential Access Indication On B0 And B1 TCM Ports*

Revision(s) Affected Revision A and earlier

Details Cortex-R4 implements two TCM interfaces (ATCM and BTCM) to support the integration of local memories. The BTCM interface can support two separate ports (B0 and B1) with limited address filtering to choose which port a given BTCM access is performed on. This filtering can either be based on the most significant bit of the address or bit[3] of the address. The latter of these is referred to as interleaved BTCM. Each TCM port includes a sequential signal which indicates that the address of the current access is sequential to the address of the last access made to the same port. This signal is provided to allow memory controllers to avoid having to perform a full address check for sequential accesses. Due to this erratum, the sequential signal on the B0 and B1 ports (B0TCSEQ and B1TCSEQ respectively) may be spuriously asserted for accesses to the same address as the previous access. This occurs when a waited data phase to one BTCM port is pipelined with a sequential address phase to the other BTCM port. The sequential signal is held asserted for the entirety of the delayed address phase which means that the TCM currently being accessed may see multiple accesses to the same address, each being marked as sequential.

Conditions: All of the following are required for this erratum to occur:

- The processor must be configured to include both B0 and B1 TCM ports
- The ENTCM1IF configuration input signal must be tied HIGH to enable the B1 port
- The SLBTCMSB configuration input signal must be tied LOW to enable interleaved BTCM
- The processor must be fetching from BTCM
- At least one of the TCM ports must be implemented with non-zero wait states

Implications: The implications of this erratum are entirely dependent on how the BTCM is implemented. If a TCM memory controller relies on the sequential bit to determine the address of an access, then this erratum is likely to cause instruction fetches to return data from the wrong address.

Workaround(s) The following workarounds are possible:

- The BTCM memory controller must not use the sequential bit
- The BTCM memory controller must qualify the sequential bit as follows:
  – Register bit[4] of the address when accepting an access
  – On an access marked as sequential, compare bit[4] of the current address with the copy of bit[4] registered from the address of the previous access
  – Ignore the sequential flag for the current access if these bits are the same

DCAN#22 *Incorrect Payload Stored In Mailbox*

Revision(s) Affected Revision A and earlier

Details If there is ongoing traffic on the bus and DCAN INIT bit is set, the DCAN state machine may enter a state where incorrect data is stored in the message RAM. The incorrect payload is actually from the previous message on the bus, which may not even have a matching ID configured in any receive mailbox. In this case, the Arbitration bits (ID and DLC etc.) are correct, but the payload is incorrect. This is a corner condition which depends on the timing of INIT bit set with respect to the last message frame reception. Coming out of a power-on-reset, the DCAN module has INIT bit set by default. There is no issue in this case.

Workaround(s) There are multiple workarounds available.

- After software sets the INIT bit, generate a software reset to DCAN using SWR bit in CAN control register. After INIT bit is set:
  - Disable all interrupts inside the DCAN module, by clearing IE0 and IE1 bits (can be done together with setting the INIT bit)
  - Wait for at least 6 CAN clock cycles and check for the pending DCAN interrupts to avoid phantom interrupts
  - Set SWR bit. This will reset the state machines and get them back in sync
  - Setup DCAN configuration: Setup Bit timing, Setup MSG_VAL bits, Set CAN pins as functional, Other DCAN configuration
  - Clear INIT bit when DCAN needs to be active
- Software can disable and re-enable peripherals (includes DCAN) using global peripheral enable bit (PENA) in system module register CLK CNTL. This would require reconfiguration of DCAN again and other peripherals as needed. Note this would reset all peripherals.
- Set the DCAN module into local Power Down Mode (by setting bit "PDR" in the CAN Control Register) before setting the INIT bit. This will cause the DCAN to wait for the end of all CAN activity before setting the INIT bit. Note that this workaround applies only to DCAN module versions which have local power down feature.

DMA#27 — *DMA Requests Lost During Suspend Mode*

Revision(s) Affected Revision A and earlier

Details Peripheral may keep generating DMA requests and after two requests other will be lost during suspend mode. This result in the data transfer loss. Subsequent DMA requests may be lost in suspend mode. This results in data transfer loss.

Workaround(s) Either use TTYPE = Block transfer when DMA DEBUG MODE is '01' (Finish Current Block Transfer) or use DMA DEBUG MODE = '00' (Ignore suspend) when using TTYPE = Frame transfer to complete block transfer even after suspend/halt is asserted.

DMM#11 — *Control Signal Is Not Being Selected*

Revision(s) Affected Revision A and earlier

Details During JTAG read, JTAG_RDATA for the DEST1BL2 and DEST2BL2 registers is not obtained at the output as a result of the control signal is not being selected.

Workaround(s) None

DMM#12 — *Programming DMMPC5 Register Is Not Clearing $\overline{\text{nDMMENA}}$ Pin In JTAG Interface*

Revision(s) Affected Revision A and earlier

Details When a write operation is performed on DMMPC5 register, it should clear the corresponding pin set or having output state as logic high. When the write to DMMPC5 register is done through JTAG interface, it clears all the pin except the $\overline{\text{nDMMENA}}$ pin.

Workaround(s) None

DMM#14 — *Packet Error Is Not Being Generated*

Revision(s) Affected Revision A and earlier

Details A packet error is not being generated by DMM when appropriate. This occurs in non continuous clock mode when the received data is greater than the programmed width and the received data is obtained in a single clock cycle.

Workaround(s) None

DMM#16 — *BUSY Flag Not Set When DMM Starts*

Revision(s) Affected	Revision A and earlier
Details	The BUSY flag is not set when DMM starts receiving a packet. It is set only after the packet has been received and deserialized.
Workaround(s)	Wait for 95 DMM clock cycles before checking the status of the busy flag after the DMM ON/OFF register field has been programmed to OFF. Turn off the clock to DMM when the BUSY flag is observed to be low after this time.

ETM_R4#3 *Accessing ATCLK Registers Causes APB Interface Lockup*

Revision(s) Affected Revision A and earlier

Details The CoreSight ETMR4 performs internal clock gating to decrease power consumption. Many of the internal registers are clock gated when any of the following conditions exist:

- The ETMPWRDOWN (bit [0] of the ETM Control register, 0x000) is HIGH
- The input signals NIDEN and DBGEN are both LOW

When the internal registers are clock gated, accesses to the programmer's model registers in the ATCLK domain cause the output signal PREADYDBG to remain LOW, thereby causing the APB interface to lock up and preventing further accesses to the ETM and other devices on the same APB bus until the ETM is reset.

One of the following registers is accessed:

- CoreSight Trace ID register (register 0x080, offset 0x200)
- Integration Register ITATBCTR0 (register 0x3BE, offset 0xEF8)
- Integration Register ITATBCTR1 (register 0x3BD, offset 0xEF4)
- Integration Register ITATBCTR2 (register 0x3BC, offset 0xEF0)
- Integration Register ITATBDATA0 (register 0x3BB, offset 0xEEC)
- Integration Register ITTRIGGERREQ (register 0x3BA, offset 0xEE8)
- Integration Register ITTRIGGERACK (register 0x3B9, offset 0xEE4)

Implications When this occurs, the PREADYDBG output from the CoreSight ETM-R4 is driven low until PRESETDBGn is asserted LOW. This means the APB bus is locked up and no further accesses can be made to the CoreSight ETM-R4 or any other peripherals on the same APB bus. This is highly unlikely to occur, because debug tools must always power up the ETM before accessing the affected registers. If NIDEN and DBGEN are both LOW, tools must inspect the Authentication Status register (register 0x3EE, offset 0xFB8) to determine if non-invasive debug is enabled before accessing the affected registers. The only other likely reason to access these registers is by faulty software, for example where a corrupted pointer causes the ETM registers to be accessed by accident.

Workaround(s) This is a workaround for tools vendors. Tools should ensure that the proper mechanisms are used to detect if the ETM is powered up and non-invasive debug is enabled before accessing the affected registers. The ETMPWRDOWN bit (bit [0] of the ETM Control register, register 0x000) must be LOW and the Authentication Status register (register 0x3EE, offset 0xFB8) must indicate noninvasive debug is enabled. There is no workaround when faulty software accidentally accesses the ETM. Do not allow CCS to access affected memory range before trace tool is initialized.

ETM_R4#4 *AFREADYM Signal Not Asserted*

Revision(s) Affected Revision A and earlier

Details On the AMBA Trace Port (ATB) of the CoreSight ETM-R4, two signals are provided to implement a trace data flushing mechanism. AFVALIDM is an input signal which requests a flush of all trace currently stored in the CoreSight ETMR4. AFREADYM is an output signal which indicates when all stored data has been output by the CoreSight ETM-R4 and is asserted in response to an assertion of AFVALIDM. This flushing mechanism allows trace capture devices to dynamically request all stored data to be output. On CoreSight ETM-R4 the AFREADYM signal might not be asserted under certain conditions, causing the ETM to never acknowledge the flush request. Two sets of conditions can cause this to occur, described below in Conditions 1 and Conditions 2.

Conditions 1:

The following conditions must occur in the following order for this to occur:

- The CoreSight ETM-R4 is reset using the $\overline{\text{nPORESET}}$ input.
- The ETMPWRDOWN bit (bit [0] of the ETM Control Register 0x000) is cleared.
- The ETMPWRDOWN bit is set 4. AFVALIDM is asserted

Conditions 2

The following conditions must occur in the following order for this to occur:

- The ETMPWRDOWN bit (bit [0] of the ETM Control Register 0x000) is cleared.
- The ETM is programmed and tracing is enabled.
- Both the NIDEN and DBGEN input signals are driven LOW while trace data is still in the ETM's FIFO.
- AFVALIDM is asserted.

Implications: Trace capture devices will usually use the flush mechanism to request all data from the system before stopping trace capture. When this occurs these trace capture devices will never receive a flush completion acknowledgment via AFREADYM and therefore might never stop capturing trace. For example, if the trace capture device is the CoreSight TPIU or CoreSight ETB then trace decompression tools might continue polling the TPIU or ETB waiting for trace capture to stop and it never stops, thereby causing an infinite loop until the device is reset, or a timeout in the tools is triggered.

Workaround(s) This is a workaround for tools vendors. If a trace capture device is configured to issue a flush before trace capture stops then: If the CoreSight ETM-R4 is connected to a trace funnel, then the CoreSight ETM-R4 should be configured to be the lowest priority ATB source. This ensures that the ETM is flushed last. This might impact the number of FIFO overflows observed in a bandwidth limited system. The tools should monitor the status of the flush operation. When using a CoreSight TPIU or CoreSight ETB this can be done using the Formatter and Flush Status Register. If a flush is observed to continue for a long period of time (for example, 100ms or more) then trace capture should be disabled immediately without waiting for the flush to complete. This might result in some of the trace which was present in the CoreSight ETM-R4 to not be captured by the trace capture device. Trace tools can avoid this by configuring the trace capture devices to never request a flush when the CoreSight ETM-R4 is connected to the trace capture device. This might result in some of the trace which was present in all trace sources to not be captured by the trace capture device.

ETM_R4#5 *Triggers Might Not Occur*

Revision(s) Affected Revision A and earlier

Details If an A-Sync packet is generated at the same time as a trigger occurs, the Trigger packet might not appear in the trace stream. Additionally, the TRIGGER output signal from the ETM might not be asserted. Conditions The following conditions must occur in the same clock cycle:

1. An A-Sync packet is generated.
2. A Trigger occurs A-Sync packets are generated periodically in the trace stream and the period is dependent on the value of the Synchronization Frequency Register (register 0x078), which can take a value of between 72 and 4096.

Implications: The TRIGGER output signal might not be asserted. This means that the trigger condition might not occur on the trace port or be embedded in the trace stream by a Trace Port Interface Unit. This means trace capture and analysis tools might not stop capturing trace which means that the desired trace might be overwritten in the trace capture device. Additionally, any cross triggering in the system which is configured to trigger on the ETM's TRIGGER signal might not trigger. It should be noted that this is unlikely to occur. This is because A-Sync packets are usually generated quite rarely, depending on the Synchronization Frequency. For example, the default value of the Synchronization Frequency Register is 1024, which means that an A-Sync packet is generated every 1024 bytes of trace. Since the ETM can generate up to 4 bytes of trace per cycle, this would imply a 1 in 256 chance of this occurring in a single trace run. However, since the trace output is bursty, with an average of around 2 bytes per cycle for full data trace, this is likely to occur significantly less than once in 256 separate trace runs with the default Synchronization Frequency Register value (1024).

Workaround(s) The probability of this occurring can be reduced by increasing the value of the Synchronization Frequency Register. This issue will not be seen if trigger function is not used.

ETM_R4#6 *ETM Address Range Comparator Does Not Match*

Revision(s) Affected Revision A and earlier

Details

An address range comparator configured for data addresses must match if a data transfer accesses any byte within the range. If an unaligned transfer occurs at the top of memory which wraps around to the bottom of memory, the bytes accessed at the bottom of memory will be ignored by address range comparators. This only occurs when tracing code which performs Unpredictable operations. The ARM Architecture does not permit data accesses to wrap around the top of memory. These accesses are Unpredictable. However, the ETM is sometimes used to help find Unpredictable operations in software running on the ARM processor. It is therefore useful for the ETM to support this behavior. This only affects word and halfword transfers which wrap around the top of memory. A word-aligned multi-word transfer which wraps around the top of memory, for example and LDM or LDRD, is Unpredictable but is not affected by this erratum. This does not apply if the address range comparator has data value comparisons enabled. This is because unaligned accesses never match if data value comparisons are enabled. Single address comparators are not affected.

Conditions:

- An address range comparator is configured for data addresses, with data value comparisons disabled.
- A halfword access is performed at address 0xFFFFFFFF or a word access is performed at 0xFFFFFFFD or higher.
- The access accesses bytes at the bottom of memory greater than or equal to the lower bound of the address range.
- The access meets all other requirements for the address range comparator to match.

Example Address range comparator 1 is configured as follows:

- match on loads or stores - lower bound is 0x00000001 - upper bound is 0x00000002

TraceEnable is configured to start tracing whenever the address range comparator 1 matches. A data transfer occurs of size word at address 0xFFFFFFFF. This accesses the following bytes: - 0xFFFFFFFF - 0x00000000 - 0x00000001 - 0x00000002 The access of the byte at address 0x00000001 should cause address range comparator 1 to match. Because of this erratum, it does not match, and the transfer is not traced.

Implications: The comparator does not match. If software is performing Unpredictable data transfers which wrap around the top of memory, accesses to addresses 0x00000000 to 0x00000002 might be missed.

Workaround(s)

It might not always be possible to consider the single address comparator in the event programming. For example, if the trigger condition is (Address Range 1 AND Sequencer state 1) then it is not possible to work around this erratum, because an event can only consider two conditions. Address range comparators with the 'exact match' bit clear, hold their value between data transfers. Single address comparators do not hold their value between data transfers. For example, if an event is configured, using this workaround, for (Address Range 1 OR Single address 1), then the event output will not hold its value after a wraparound access as described above, because it relies on the Single Address Comparator matching. This might cause unexpected results, depending on the event being controlled. This issue is avoided if you do not use address range comparators.

ETM_R4#7 *Lock Access Can Be Modified When PADDRBG31 Is High*

Revision(s) Affected Revision A and earlier

Details

When PADDRDBG31 is HIGH, the CoreSight ETM-R4 should consider all programming accesses to have been initiated by an external debugger. When PADDRDBG31 is LOW, the CoreSight ETM-R4 should consider all programming accesses to have been initiated by software running on the system. The lock access mechanism consists of the Lock Access Register (register 0x3EC) and the Lock Status Register (register 0x3ED). The Lock Status Register should indicate that no lock access mechanism exists if programming accesses are initiated from an external debugger, i.e., when PADDRDBG31 is HIGH. Writes to the Lock Access register should be ignored by the ETM when PADDRDBG31 is HIGH. Due to this issue on CoreSight ETM-R4, when PADDRDBG31 is HIGH, the Lock Status Register correctly indicates that no lock access mechanism is present on accesses from an external debugger. However the Lock Access Register can be used to mistakenly allow accesses by software running on the system. Also, the Lock Access Register can be used to mistakenly prevent accesses by software running on the system.

Conditions:

- PADDRDBG31 is HIGH, indicating an access from an external debugger
- Debugger software writes to the Lock Access Register without checking the Lock Status Register

Implications: It is possible for target resident software to be given access to the ETM registers when accesses should be prevented using the lock access mechanism. Conversely, it is possible for target resident software to be prevented from accessing the ETM registers when accesses should be allowed by the lock access mechanism. If an external debugger writes the lock access key, 0xC5ACCE55, to the Lock Access Register, the software lock is unlocked and software running on the system can write to the ETM registers. Additionally, if an external debugger was to write any value apart from 0xC5ACCE55 to the Lock Access Register, the software lock is locked, preventing any software running on the system from writing to the ETM registers. Since an external debugger should read the Lock Status Register to determine if the lock mechanism is present before writing to the Lock Access Register, it is expected that no external debugger will write to the Lock Access Register thereby never causing this to have any adverse effects.

Workaround(s)

This is a workaround for tools vendors. Tools must follow the recommended behavior for operating the lock access mechanism. Tools must read the Lock Status register before writing to the Lock Access Register. If the Lock Status Register indicates that no lock access mechanism exists, then the debugger must not write to the Lock Access Register. This is the normal recommended behavior for tools, therefore in most cases no workaround is necessary.

ETM_R4#8 *Power Down Status Register (PDSR) Reads As Zero*

Revision(s) Affected	Revision A and earlier
Details	The ETM Power Down Status Register (register 0xC5, offset 0x314) is used to indicate the current status of the ETM. This register should read as 0x00000001 in CoreSight ETM-R4 indicating the ETM is always powered up, but it incorrectly reads as 0x00000000. Implications Tools read this register to determine if the ETM is accessible and whether register state has been lost due to a power down. The ETM-R4 does not support multiple power domains and this register should always read as 0x00000001, indicating the ETM is powered up and the registers are accessible. The read value of 0x00000000 incorrectly indicates the ETM is powered down and no registers are accessible. Tools might attempt to poll this register waiting for the ETM to be powered up but this condition will never be satisfied.
Workaround(s)	This is a workaround for tools vendors. When reading the Power Down Status Register on CoreSight ETM-R4, if the read is successful and no error response is detected ignore the value read and assume it was 0x00000001.

ETM_R4#9 *Cycle Count Is Up 7 Cycles Greater Than It Should Be On Debug Exit*

Revision(s) Affected Revision A and earlier

Details When the processor is in debug state tracing is automatically disabled on the ETM and so an I-Sync packet is generated by the ETM before the first item traced after the processor exits from debug state. This erratum, which is unlikely to occur, causes the cycle count in the I-Sync packet generated by the ETM after the processor exits debug state to be up to 7 cycles greater than it should be. This can occur when tracing is enabled on the ETM before and after the processor enters debug state and the ETM is configured in cycle accurate mode and the ETM FIFO overflows just before debug mode is entered.

Conditions:

The specific conditions that cause this to occur are as follows:

1. The ETM must be configured in cycle accurate mode and instruction tracing must be enabled (bit [12] set and bit [20] cleared of the ETM Control Register, 0x00).
2. The ETM is programmed and tracing is enabled.
3. There must be several consecutive cycles in which the processor executes an instruction that pass its condition codes. These cause the ETM to generate a format 1 P-header with a WE value greater than 1.
4. Then one cycle in which no instruction is executed. The ETM generates a format 3 P-header for this cycle. Then the processor enters debug state and a debug exception is signaled to the ETM.
5. The ETM's FIFO overflows due to the format 3 P-header generated in step 4.

Implications: If this occurs the processor will appear to be in debug state for up to 7 cycles longer than it is in reality. All other trace information is unaffected.

Workaround(s) None

ETM_R4#10 *Address Range Comparator Not Cleared By Coprocessor Access*

Revision(s) Affected Revision A and earlier

Details Under normal conditions, Address Range comparators configured for data address comparisons hold their state between accesses. If a coprocessor access occurs, the comparator should stop firing because there is no address to compare against. This causes the ETM to ignore the coprocessor access and the address range comparator will continue to fire.

Conditions:

The following sequence must occur in order:

1. A data address range comparator is configured with the exact match bit clear.
2. A data transfer occurs inside the range, causing the comparator to fire.
3. A coprocessor access occurs.

   The coprocessor access should clear the Address Range comparator, however this causes the Address Range comparator to continue firing.

Implications: Any ETM resource which is configured to be sensitive to the comparator might behave unexpectedly, in that a counter might count for longer than expected or tracing might be disabled or enabled for longer than expected. Trace is not corrupted by this erratum.

Workaround(s) None

ETM_R4#12 *Multiple Trigger Requests Occur*

Revision(s) Affected Revision A and earlier

Details The TRIGGER output signal from the ETM might be asserted multiple times for one trigger event.

Conditions:

For this issue to occur, the following conditions must occur:

- The TRIGSBYPASS input signal is tied LOW
- A Trigger occurs
- The ATB transaction which contains the Trigger packet is stalled by the CoreSight system for at least 5 cycles

The ATB transaction may stall because the trace port cannot output the data quickly enough. For example, if a trace port size of 4 bits is used, the ATB transaction might be delayed for up to 8 cycles. Other trace sources in the system might also affect the available bandwidth on the ATB interface.

Implications: The TRIGGER output signal from the ETM is asserted multiple times. This might cause the CoreSight trace sink to indicate multiple triggers in the trace stream or on the trace port. If the TRIGGER output signal is used in a cross trigger system then multiple trigger events might be signaled to other devices which are connected to the cross trigger system and configured to receive the trigger indication. This does not affect the ETM's trace stream. In most trace capture devices, this erratum should not be a problem.

Workaround(s) This is a workaround for tools vendors. If the ETM trigger is being used to embed triggers into a formatted trace stream using the CoreSight ETB or CoreSight TPIU then multiple triggers might be embedded in the formatted trace stream. Only the first of these should be used and the others can be safely discarded.

ETM_R4#13 *Consecutive Flushes Not Acknowledged*

Revision(s) Affected Revision A and earlier

Details Consecutive flushes occur when two flushes are performed on the ATB interface, one after the other, with no cycles between them. The flush request and acknowledge signals on the ATB interface are AFVALID and AFREADY respectively. The AMBA 3 ATB Protocol Specification (ARM IHI 0032A) states that, when AFREADY is asserted, then AFVALID must be deasserted on the following cycle, unless an additional flush is intended. The consequence of this is that the second flush, of a consecutive flush sequence, will not complete if it is requested when TraceEnable is HIGH. This will not occur under any of the following conditions:

- The ProgBit is HIGH TraceEnable is LOW and the FIFO is empty
- The power down bit is HIGH
- The NIDEN and DBGEN inputs are LOW
- The Cortex-R4 is in the wait for interrupt state

Conditions:

All of the following must occur:

1. A CoreSight system does not contain a trace funnel.
2. The trace sinks are configured such that multiple flushes can occur in a trace session.
3. A flush request is generated when another flush request is still outstanding.

This is unlikely to occur because normally only one flush request is outstanding at a time.

Implications: Trace sinks will usually use the flush mechanism to drain all data from the system before stopping trace capture. When this occurs these trace sinks will never receive a flush completion acknowledgment and therefore might never stop capturing trace. For example, if the trace sink is either the CoreSight TPIU or CoreSight ETB, then trace debug tools might continue polling the TPIU or ETB waiting for trace capture to stop, causing an infinite loop until one of the conditions in the description that prevents this from occurring.

Workaround(s) This is a workaround for trace tool vendors. Normally a workaround is not required, but if it is, ensure that only one flush is generated per trace session.

ETM_R4#16 *ETM-R4 Fails To Trace Vnt Packet For The Second Half Of SWP Instruction*

Revision(s) Affected Revision A and earlier

Details When tracing SWP or SWPB instructions, the load and store parts of the SWP or SWPB instruction should be traced with separate data packets to the same address. If the load transfer is traced and the store transfer is not traced, the store transfer should be traced with a “Value Not Traced” packet. When this erratum occurs, erroneously, no trace is generated for the store transfer of the SWP or SWPB.

Conditions: The following conditions must occur:

- The ETM is enabled and is tracing
- A SWP or SWPB instruction is executed
- ViewData is configured to only trace the load part of the SWP or SWPB instruction

Implications: If the ETM traces any data transfer, a data packet must be traced for every subsequent data transfer for that same instruction. This allows trace analysis tools to determine which registers were used or updated by the traced data items. When this erratum occurs, only the load part of the SWP or SWPB instruction is traced and therefore analysis tools cannot determine if the transfer is the load or store part of the SWP or SWPB instruction. This might cause misinterpretation of the execution of the processor by the analysis tool. The trace stream is not corrupted.

Workaround(s) The following workarounds are for users or tool vendors:

- Ensure that for all SWP and SWPB instructions in your code ViewData is not configured to trace load data only
- If ViewData has been configured to trace only the load transfer of a SWP or SWPB instruction and a single transfer has been traced, the trace analyser can assume that this corresponds to the load part of the instruction

F035_FLEPBANK#1 *Redundant column shows much higher programmed Vt in compressed programming .*

Revision(s) Affected	Revision 0
Details	Due to program info(DATAIN) failing to pass on to WRTDRV for redundant columns, redundant column shows much higher programmed Vt in compressed programming . Another symptom is column repaired unit failed to program IO checkerboard pattern. This problem should affect all products with FLEP bank with ACESTAMP less than V4.1.
Workaround(s)	Applies to Factory test only.

F035_FLEPBANK#2 *High source line leakage in F035 platform devices(FLEP bank)*

Revision(s) Affected Revision A and earlier

Details In TCR33 the GBL stays at VSS instead of floating. Since the word line in dummy column stick is shorted to array source to prevent over-erase, the word line will be raised as the stress voltage at array source. If the GBL is grounded, the bits will conduct until the bits are programmed. This is why the leakage will reduce after high voltage stress at array source.

Workaround(s) Applies to Factory test only. For TCR33, Set a valid address with AIN[3]=1 and keep OTP=ENGR=0 For TCR34 with a non OTP sector, set a valid address in that sector with AIN[3] =1 For TCR34 with OTP sector, need to get SLL from TCR33 and substract all SLLs from non OTP sectors from it.

FLEXRAY#16 *Detected Coding Error Not Reported*

Revision(s) Affected Revision A and earlier

Details In case eray_bclk is below eray_sclk/2, TEST1.CERA/B may fail to report a detected coding error. All detected coding errors should be reported in the Test Register 1, at TEST1.CERA/B. If eray_bclk is below eray_sclk/2, it may happen, depending on phase difference of eray_bclk and eray_sclk, that TEST1.CERA/B are not updated in case of a detected coding error and remain in the state "0000" = "No coding error detected". This is limited to the case where eray_bclk is below eray_sclk/2. The frame decoding is not affected.

Workaround(s) None

FLEXRAY#20 *E-Ray Erroneously Uses Incorrect*

Revision(s) Affected Revision A and earlier

Details When receiving gSyncNodeMax or more sync frames in an even cycle, only frames with the same sync frame IDs, as received in the even cycle, may be used for offset correction term calculation in the following odd cycle. The E-Ray erroneously uses the first gSyncNodeMax sync frames for offset correction term calculation in the odd cycle, regardless whether they have been also received in the previous even cycle. The occurs where more than gSyncNodeMax nodes are configured to transmit sync frames and where different sets of sync frames are transmitted in even and odd cycle. The offset correction term may base on a different set of sync frames than the rate correction term. Registers ESIDn / OSIDn hold the IDs of the first received sync frames up to gSyncNodeMax used for offset correction term calculation.

Workaround(s) Avoid faulty configurations with more than gSyncNodeMax nodes configured to be transmitter of sync frames.

FLEXRAY#21 *Temporal Deviation Is Different For Channel A And B*

Revision(s) Affected Revision A and earlier

Details Temporal deviation (= time between primary time reference point and action point offset) is different for channel A and B and the values have the following combination: greater than or equal zero on one channel and negative on the other channel the channel with a relative deviation value greater than or equal zero is chosen for offset correction term calculation instead of the negative value. This occurs when both channels are used and if there is a large difference in the propagation delays on channel A and B. I the relation between measured deviation values on channel A and B, the calculated offset correction term may have an error of maximum the difference between the two deviation values. As a result, the error of the local time of the node is limited to the difference of the temporal deviation values of both channels.

Workaround(s) For dual channel FlexRay systems sync frames must be transmitted on both channels. The propagation delay between the two nodes is expected to be nearly the same on both channels. If this is not the case, the channel depending parameter pDelayCompensation[A/B] (GTUC5.DCA,B) must be used to compensate for the different propagation delays. With a correct adjustment of the parameter the difference of the deviation values on both channels is very low. As a result, the error of the local time caused by the implementation error is also minimized.

FLEXRAY#22 *E-Ray Acting As Sync Node Does Ignores Own Sync Frame For Counting Number Of Sync Frames*

Revision(s) Affected Revision A and earlier

Details When the E-Ray acts as sync node it does not consider its own sync frame for counting number of sync frames sets. This results in an error interrupt flag EIR.SFO if it receives more than gSyncNodeMax (GTUC2.SNM[3:0]) sync frames. This occurs when a sync node receives exactly gSyncNodeMax sync frames from other sync nodes during one communication cycle. When this occurs the error interrupt flag EIR.SFO is not set.

Workaround(s) Avoid faulty configurations with more than gSyncNodeMax nodes configured to be transmitter of sync frames.

FLEXRAY#23 *Temporal Deviation Value Is Set To The Measured Value Of The Previous Non Sync Frame*

Revision(s) Affected Revision A and earlier

Details When a non sync frame is detected to be valid simultaneously with the slot boundary, and in the following slot a sync frame is received, the temporal deviation value (= time between primary time reference point and action point) of the received sync frame is erroneously set to the measured value of the previous non sync frame. This occurs when a non sync frame is received in one slot and a sync frame is received in the following slot. Additionally, the detection of 'valid frame' (fsp_val_frame) for the non sync frame must be exactly at the slot boundary. As a result, the temporal deviation value of the sync frame is set to the deviation value of the previous non sync frame. The positive shift of the non sync frame in direction of slot end results in a relative large positive deviation value (order of magnitude of the parameter gdActionPoint- Offset). With more than two sync frame senders within the cluster, this large value is most probably not taken into account for correction term calculation because of the nature of the fault-tolerant midpoint algorithm. If the faulty value is used for offset correction term calculation, the resulting offset shift will be corrected in the following cycles.

Workaround(s) None

FLEXRAY#24 *Incorrect Secondary Time Reference Point (STRP) Stored*

Revision(s) Affected Revision A and earlier

Details When noise or an aborted frame leads to the detection of a secondary time reference point (STRP), and subsequently, a valid sync frame is detected in the same slot and the STRP of the valid sync frame occurs simultaneously with the action point, the temporal deviation value of the first detected STRP is stored instead of the value of the correct STRP. This occurs when noise occurs before reception of a valid sync frame or a frame reception starting before action point is aborted and then a valid sync frame is received in the same slot. As a result the wrong deviation value is used for correction term calculation. Depending on the number of sync frames and other measured temporal deviation values it may lead to an incorrect rate or offset correction value.

Workaround(s) None

FLEXRAY#25 *E-Ray Erroneously Updates Register ACS*

Revision(s) Affected Revision A and earlier

Details When the slot counter has reached ID 2047 and the end of dynamic segment is not reached, the slot counter wraps around to 0 and stays there until the end of the dynamic segment. In this state (slot counter = 0) the E-Ray erroneously updates register ACS every minislot. When correct, the slot status is only updated at the end of the dynamic segment. This occurs in the following case: gNumberOfStaticSlots + gNumberOfMinislots > cSlotIDMax = 2047 As a result the register ACS is updated every minislot until the end of the dynamic segment. This update may also lead to an update of error interrupt flags.

Workaround(s) Avoid configurations with gNumberOfStaticSlots + gNumberOfMinislots > cSlotIDMax.

FLEXRAY#26 *Cycle Counter Value MTCCV.CCV Returns Incorrect Value*

Revision(s) Affected Revision A and earlier

Details During startup the reading cycle counter value MTCCV.CCV may return the value of '1' or '63' instead of the CHI default value of '0'. This can occur in one of the following dedicated startup states: - Leading coldstart node: cycle 'no schedule' at POC-state 'coldstart collision resolution' - Following coldstart node: cycle 1 at POC-state 'integration coldstart check' - Integrating node: cycle 1 at POC-state 'integration consistency check' MTCCV.CCV is updated with the correct internal value of 1 or 63, although this is not required by the FlexRay Protocol Specification v2.1.

Workaround(s) Ignore MTCCV.CCV during startup.

FLEXRAY#27 *Data Transfer From Message RAM To Transient Buffer Not Started*

Revision(s) Affected	Revision A and earlier
Details	When a parity error occurs and the message handler transfers data from the Message RAM to the Transient Buffer the transmission may not start. This occurs during cancelled transmissions of dynamic frames when a parity error occurs during data transfer from Message RAM to the Transient Buffer. When the described condition occurs, the slot counter value is captured and LDTS.LDTA,B[10:0] is updated with this value at the end of the dynamic segment.
Workaround(s)	None

FLEXRAY#28 *CC Unable To Transit From STARTUP To NORMAL_ACTIVE State*

Revision(s) Affected	Revision A and earlier
Details	When a $\overline{\text{CAS}}$ symbol is received during startup exactly at the beginning of cycle 0, the detection of following startup frames is not possible. When this occurs the CC is not able to transit from STARTUP to NORMAL_ACTIVE state.
Workaround(s)	Leave and reenter STARTUP state by Host commands READY and RUN.

FLEXRAY#29 *EIR.MHF Error Interrupt Flag Set*

Revision(s) Affected	Revision A and earlier
Details	When the baud rate prescaler (PRTC1.BRP[1:0]) is configured to 5 or 2.5 MBit/s, and at least one loop-back test is performed at channels A, B, or both, and then a startup without a preceding hardware reset is initiated, the E-Ray will not store a non-null frame received on the channel(s) on which a loop-back test was performed into its message memory and will raise the EIR.MHF error interrupt flag. The issue persists until the E-Ray has transmitted its first non-null frame on the same channel(s). This occurs when loop-back is used with the baud rate prescaler (PRTC1.BRP[1:0]) configured to 5 or 2.5 MBit/s and where there is no hardware reset before a startup is initiated. The error interrupt flag EIR.MHF is raised and the contents of the first received non-null frames will be lost.
Workaround(s)	Run loop-back tests with 10 MBit/s (PRTC1.BRP[1:0] = "00") or perform a hardware reset after a loop-back test.

FLEXRAY#30 *E-Ray Stuck In Startup State*

Revision(s) Affected Revision A and earlier

Details in event of 'integration successful on X' and ' integration abort on Y' at the same point in time, the SUC prioritizes the input event of successful integration, leaves the state POC:initialize schedule and enters, depending of its startup role, either POC:integration coldstart check or POC:integration consistency check. The reaction of the GTU depends on the related channels and the actual state of clock synchronisation startup process. If the GTU is in state CSP:A active. For the combination of 'integration successful on A' and 'integration abort on B' the GTU stops the macrotick generation process and terminates both clock synchronisation startup processes. In this case the CC is stuck in POC:integration coldstart check or POC:integration consistency check. For the other combination of 'integration successful on B' and 'integration abort on A' the GTU stops the macrotick generation process but keeps the clock synchronisation startup process of channel A running. This leads to a delayed (best case two cycles) successful startup of the E-Ray. If the GTU is in state CSP:B active, the same is true with the swapped channels. The combination of 'integration successful on B' and 'integration abort on A' leads to the stuck condition and the combination of 'integration successful on A' and 'integration abort on B' leads to a delayed startup. This occurs during the simultaneous generation of internal signals 'integration successful on X' on one channel and 'integration abort Y' on the other channel. As a result, the E-Ray either is stuck in startup states or extends the startup by at least two cycles.

Workaround(s) Use a timer to measure how long the E-Ray stays in state POC:integration coldstart check or POC:integration consistency. If the timeout is reached, re-enter the startup state by using the CHI commands READY and RUN.

FLEXRAY#31 *Protocol Engine Reads Extra Words Of Its Transient Buffer*

Revision(s) Affected Revision A and earlier

Details The protocol engine reads at least the first two words of its Transient Buffer and one word more than required by the payload count for each transmitted message, even if no data is needed (null frame or payload count is zero). If a parity error is detected when the protocol engine reads from the Transient Buffer, the transmitted message is invalidated by setting its CRC code to zero. This occurs when a parity error occurs in one of the words in the Transient Buffer that are read but not needed for a particular message. As a result, the transmitted message is invalidated.

Workaround(s) None, to be treated as if the parity error is in any other word of the Transient Buffer.

FLEXRAY#32 *Incorrect Content In CCSV*

Revision(s) Affected Revision A and earlier

Details When the POC state changes between WAKEUP and READY the content of register CCSV may show a slight discontinuity, i.e., CCSV.SLM [1:0] may be updated late. This is limited to configurations with SUCC1.TSM = 0 (ALL Slot Mode).

Workaround(s) Ignore CCSV.SLM [1:0] in states READY and WAKEUP.

FLEXRAY#33 *Wakeup Pattern Transmitted n bit Times Early*

Revision(s) Affected Revision A and earlier

Details If the protocol engine is in the state WAKEUP_LISTEN and if the parameter gdWakeup-SymbolRxLow is programmed to a value 11-n, then the channel idle recognition CHIRP comes n bit times early. This is limited to configurations with gdWakeupSymbolRxLow <11. For bit rates of 10/5/2.5 MBit/s, Protocol Version 2.1 Rev A requires a minimum gdWakeupSymbolRx- Low of 46/23/11 gdBit. As a result, the wakeup pattern is transmitted n bit times early.

Workaround(s) Set gdWakeupSymbolRxLow to value >= 11.

FLEXRAY#34 *Data In TBF Incorrectly Overwritten*

Revision(s) Affected Revision A and earlier

Details If after reception of a valid frame in slot N the reception of a second frame (transmitted by a mis-synchronized node) starts in the same slot and the second frame's payload is stored into the TBF shortly after the slot boundary, the first 32 bit payload data of the first received frame in the TBF is overwritten by the payload data from the second frame. This is limited to cases where the complete header and a part of the payload of another frame is received in slot N. The transfer of a received valid frame from TBF to MBF is initiated by the end of the actual slot. Execution is started not later than 40 bclk periods after the slot change. 40 bclk periods equate 10 bit times, when bclk=40MHz. Corruption of payload occurs when a transfer from PRT to TBF happens in the red marked time window between start of new slot and start of transfer from TBF to MBF. As a result, the first two words (4 byte) of the received valid frame's payload are corrupted.

Workaround(s) Workaround: To avoid this do the following:

- Ensure that the static slot length is configured suited to the static payload length.
- Check Message Buffer Status for boundary violation.

FLEXRAY#35 *Command CCSV.RCA and CCSV.PSL Inconsistent*

Revision(s) Affected Revision A and earlier

Details When the FREEZE command is applied during STARTUP the command CCSV.RCA and CCSV.PSL may be inconsistent. This can occur when the FREEZE command coincides with a protocol triggered state change where CCSV.RCA[4:0] is decremented. CCSV.RCA [4:0] is decremented but CCSV.PSL[5:0] does not show state COLDSTART_COLLISION_RESOLUTION.

Workaround(s) Ignore CCSV.RCA [4:0] after CHI command FREEZE.

FLEXRAY#36 *CLEAR_RAMS Command Does Not Clear The 1st RAM Word*

Revision(s) Affected Revision A and earlier

Details After execution of the CLEAR_RAMS command, the 1st RAM word holds the last data written to IBF. Reason is that the registers to support byte access to the RAM are cleared one clock cycle after the CLEAR_RAMS command was started. After execution of the CLEAR_RAMS command, the 1st RAM word holds the last data written to IBF. This occurs because the registers to support byte access to the RAM are cleared one clock cycle after the CLEAR_RAMS command was started. This occurs when the CLEAR_RAMS command is applied during ERay operation. The execution of the CLEAR_RAMS sequence after hard reset is not affected. As a result, execution of the CLEAR_RAMS command does not reset the first word of an E-Ray internal RAM to zero.

Workaround(s) Write 0x00000000 to any address of IBF directly before applying CLEAR_RAMS command.

FLEXRAY#37 *Wrong Message Buffer Used For Transmission During Next Cycle*

Revision(s) Affected Revision A and earlier

Details The message buffer for slot 1 is searched in parallel to every scan in the previous cycle. A message buffer scan is started every 8th slot. A running scan is aborted when the NIT is reached. The message buffer used for slot 1 depends on the cycle filter configuration and the bus activity in the dynamic segment. If the scan is aborted between the first and the last message buffer assigned to slot 1, it is unpredictable, if the correct message buffer is used. This occurs when two (or more) message buffers are configured for slot 1 and cycle filtering is used. If a running message buffer scan is interrupted by the NIT it cannot be guaranteed that the correct message buffer is used for transmission in slot 1of the next cycle.

Workaround(s) If cycle filtering is used, assign all message buffers configured for slot 1 to the static buffers section. I.e., message buffer number < MRC.FDB[4:0].

FLEXRAY#38 *Frame Received Is Corrupted*

Revision(s) Affected Revision A and earlier

Details If a receive slot N is followed by a transmit slot N+1, and if between end of frame reception in slot N and start of frame transmission in slot N+1 the reception of a frame (transmitted by a mis-synchronized node) is started, the header and/or payload of the valid frame received in slot N may be corrupted. This can occur in the following cases:

- Frame reception completed in slot N
- Frame reception across slot boundary between slot N and slot N+1
- Frame reception starts in slot N+1

This may occur when a receive slot is followed by a transmit slot and where at least the complete header of another frame is received before the frame received in slot N has been stored into the respective message buffer. This can have one of the following effects:

- Syntax error signaled for slot N, correct behavior, no corruption
- Boundary violation signaled for slot N and slot N+1. Header and/or payload of the message buffer assigned to slot N may be corrupted or frame received in slot N may be completely rejected.
- No error signaled for slot N. Frame received in slot N may be not stored or header of the message buffer assigned to slot N may be corrupted.

Corruption of the assigned message buffer s payload may only occur when eray_bclk is below 25 MHz.

Workaround(s) None

FLEXRAY#39 *Irregular Sync Frame List Exported In State Coldstart_Gap*

Revision(s) Affected Revision A and earlier

Details If the protocol engine is in the state Coldstart_Gap, it stops transmitting its own startup frame (according to the protocol specification), but the status data exported to the CHI (SFS, OSIDn register) lists its own startup frame as transmitted. This occurs with leading coldstart nodes and results in misleading status data.

Workaround(s) Ignore misleading status data in state Coldstart_Gap.

FLEXRAY#40 *After Detecting Low Level Beyond gdWakeupSymbolRxWindow, Node Completes Wakeup State*

Revision(s) Affected Revision A and earlier

Details In the event of a WUP that starts with a duration at the LOW level that is longer then n*gdWakeupSymbolRxWindow but shorter then (n*gdWakeupSymbolRxWindow + gdWakeupSymbolRxLow), followed by a duration of at least gdWakeupSymbolRxIdle at the HIGH level and followed by a duration of at least gdWakeupSymbolRxLow at the LOW level, the last part of this received within a window with a duration of at most gdWakeup- SymbolRxWindow, then this WUP is detected as valid while it should be considered invalid according to figure 3-39 of the FlexRay protocol specification. This only occurs in cluster wakeup with disturbances on a channel. This could result in the detection of WUP independent of the WakeupSymbolRxWindow s phase.

Workaround(s) None

FLEXRAY#44 *Register RCV Displays Wrong Value*

Revision(s) Affected Revision A and earlier

Details If the calculated rate correction value is in the range of [-pClusterDriftDamping .. +pClusterDriftDamping], vRateCorrection of the CSP process is set to zero. If register RCV is updated with this value RCV.RCV[11:0] holds the calculated value in the range [-pClusterDriftDamping .. +pClusterDriftDamping] instead of zero.

Workaround(s) A value of RCV.RCV[11:0] in the range of [-pClusterDriftDamping .. +pClusterDriftDamping] has to be interpreted as zero.

FLEXRAY#45 *Valid Sync Frame Received Is Ignored*

Revision(s) Affected Revision A and earlier

Details After reception of a valid sync frame followed by a valid non-sync frame in the same static slot the received sync frame may be ignored. If in a static slot of an even cycle a valid sync frame followed by a valid non-sync frame is received, and the frame valid detection (prt_frame_decoded_on_X) of the DEC process occurs one sclk after valid frame detection of FSP process (fsp_val_syncfr_chx), the syncframe is not taken into account by the CSP process (devte_xxs_reg).

Workaround(s) Avoid static slot configurations long enough to receive two valid frames.

FLEXRAY#46 *Sync Frame Overflow Flag EIR.SFO Set If Slot Counter Is Greater Than 1024*

Revision(s) Affected Revision A and earlier

Details If in the static segment the number of transmitted and received sync frames reaches gSyncNodeMax and the slot counter in the dynamic segment reaches the value cStaticSlotIDMax + gSyncNodeMax = 1023 + gSyncNodeMax, the sync frame overflow flag EIR.SFO is set erroneously. As a result the sync frame overflow flag EIR.SFO is set erroneously. This has no effect to the POC state. This is limited to configurations where the number of transmitted and received sync frames equals to gSyncNodeMax and the number of static slots plus the number of dynamic slots is greater or equal than 1023 + gSyncNodeMax.

Workaround(s) Configure gSyncNodeMax to number of transmitted and received sync frames plus one or avoid configurations where the total of static and dynamic slots is greater than cStaticSlotIDMax.

FLEXRAY#47 *Startup Frame Added Erroneously After Reception Of More Than gSyncNodeMax Sync Frames*

Revision(s) Affected Revision A and earlier

Details If a node receives in an even cycle a startup frame after it has received more than gSyncNodeMax sync frames, this startup frame is added erroneously by process CSP to the number of valid startup frames (zStartupNodes). The faulty number of startup frames is delivered to the process POC. As a consequence this node may integrate erroneously to the running cluster because it assumes that it has received the required number of startup frames. This occurs when there are more than gSyncNodeMax sync frames. As a result, a node may erroneously integrate successfully into a running cluster.

Workaround(s) Use correct configuration with all startup frames placed in the first static slots.

FLEXRAY#48 *Initial Rate Correction Value Of An Integrating Node Is Zero*

Revision(s) Affected Revision A and earlier

Details The initial rate correction value is zero if parameter pMicroInitialOffsetA,B is configured to be zero. Starting with an initial rate correction value of zero leads to an adjustment of the rate correction earliest 3 cycles later. In a worst case scenario, if the whole cluster is drifting away too fast, the integrating node would not be able to follow and therefore abort integration.

Workaround(s) Do not configure pMicroInitialOffsetA,B to 0x01instead of the calculated value zero. A configuration of pMicroInitialOffsetA,B =zero will delay the start of the node by only one microtick but leads to a correct initial rate correction value.

FLEXRAY#49 *Incorrect Rate And/Or Offset Correction*

Revision(s) Affected Revision A and earlier

Details If a valid sync frame is received before the action point and noise or a second frame leads to a STRP coinciding with the action point, an incorrect deviation value of zero is used for further calculations of rate and/or offset correction values. This may lead to an incorrect rate and/or offset correction of the node.

Workaround(s) ***None***

FLEXRAY#50 *SFS.MRCS Flag Set Erroneously*

Revision(s) Affected Revision A and earlier

Details If in an odd cycle 2c+1 after reception of a sync frame in slot n the total number of different sync frames per double cycle has exceeded gSyncNodeMax and the node receives in slot n+1 a sync frame that matches with a sync frame received in the even cycle 2c, the sync frame pair is not taken into account by CSP process. This may cause the flags SFS.MRCS and EIR.CCF to be set erroneously. This occurs in a faulty cluster configuration where different sets of sync frames are transmitted in even and odd cycles and the total number of different sync frames is greater than gSyncNodeMax. As a result, the error interrupt flag EIR.CCF is set and the node may enter either the POC state NORMAL_PASSIVE or HALT.

Workaround(s) Correct the configuration of Sync NodeMax.

FLEXRAY#51 *Rate Correction Set To Zero*

Revision(s) Affected Revision A and earlier

Details When a node receives too few sync frames for rate correction calculation and signals a SyncCalcResult of MISSING_TERM, the rate correction value is set to zero instead to the last calculated value. This occurs when a node enters NORMAL_PASSIVE state (pAllow- HaltDueToclock=false) because of receiving too few sync frames for rate correction calculation (SyncCalcResult=MISSING_TERM). As a result, a rate correction value of zero is applied in NORMAL_PASSIVE state instead of the last rate correction value calculated in NORMAL_ACTIVE state. This decreases the probability to re-enter a NORMAL_ACTIVE state.

Workaround(s) If NORMAL_PASSIVE state is entered due to missing sync frames, use higher level application software to leave this state and to initiate a reintegration into the cluster.

FLEXRAY#52 *WUS Generates Redundant SIR.WUPA/B Events*

Revision(s) Affected Revision A and earlier

Details If a sequence of wakeup symbols (WUS) is received, all separated by appropriate idle phases, a valid wakeup pattern (WUP) should be detected after every second WUS. The E-Ray detects a valid wakeup pattern after the second WUS and then after each following WUS. When an application program frequently resets the appropriate SIR.WUPA/B bits more SIR.WUPA/B events are seen than expected.

Workaround(s) Ignore redundant SIR.WUPA/B events.

FLEXRAY#53 *Mismatch Between Spec & RTL For POCS[5:0] Value Of CCSV*

Revision(s) Affected Revision A and earlier

Details According to the spec the POCS[5-0] of CCSV Register will show 001001 while the Flexray module is in Loop back Mode. However, in RTL the value is 1101.

Workaround(s) None

FLEXRAY#55 *FlexRay Timer Interrupts Lost On Level Triggered Interrupt Modules*

Revision(s) Affected Revision A and earlier

Details The FlexRay Timers (timer 0 and timer 1) are hooked up with direct interrupt request lines (FlexRay T0C interrupt, FlexRay T1C interrupt) to the Vector Interrupt Module (VIM). When the timer interrupt occurs, the interrupt will be generated via the VIM module, but VIM information, like contents of VIM offset registers, VIM Pending Interrupt Read Location (INTREQ) registers or VIM IRQ and FIQ Interrupt Vector registers disappear after a short amount of time (1 macro tick, which is usually 1us). Even the interrupt request to the VIM disappears. The issue does not impact the FlexRay Protocol functionality, but can impact timer events based on the FlexRay communication cycle. Those timer events can optionally be generated and used by the application software. Description of the Issue: The FlexRay Timers (timer 0 and timer 1) are hooked up with direct interrupt request lines (FlexRay T0C interrupt, FlexRay T1C interrupt) to the Vector Interrupt Module (VIM). When the timer interrupt occurs, the interrupt will be generated via the VIM module, but VIM information, like contents of VIM offset registers, VIM Pending Interrupt Read Location (INTREQ) registers or VIM IRQ and FIQ Interrupt Vector registers disappear after a short amount of time (1 macro tick, which is usually 1us). Even the interrupt request to the VIM disappears.

Conditions: When a FlexRay timer interrupt, i.e. the timer 0 interrupt, is asserted, the E-Ray timer output signal eray_tint0 is set to '1' for the duration of one macro tick and SIR.TI0 is set to '1'. The signal eray_tint0 is used to hook up the timer interrupt to the VIM. Since the VIM is level triggered, the timer interrupt will only be active for the time the eray_tint0 signal is set. Same for timer 1 interrupt and the according eray_tint1 signal.

Implications:

- When the software is reading the VIM information too late, the information might no longer be valid and can lead to phantom interrupts.
- If a interrupt service routine is in execution while a FlexRay timer interrupt occurs, the timer interrupt request can disappear before the execution of the interrupt service routine has finished. In this case the FlexRay timer interrupt would be lost.

Workaround(s) Both FlexRay Timer interrupts (timer 0 and timer 1) are available also as status flags (SIR.TI0 and SIR.TI1) in the Status Interrupt Register of the E-Ray. Each status flag can trigger either the FlexRay Level 0 or FlexRay Level 1 interrupt signal to the VIM. Those interrupt signals behave as expected and do not disappear after one micro tick. In case more status interrupts are used (up to 19 status interrupt sources are possible), the software must handle the identification of the interrupt source.

Hardware Workaround: For a level triggered interrupt module like VIM, those E-Ray signals (eray_tint0, eray_tint1) can not be used for interrupt generation. The status signals SIR.TI0 and SIR.TI1 need to be used instead. Both status signals must be cleared by writing a '1' to the according flag in the Status Interrupt Register (SIR) of the ERay, which would be the expected behavior.

FLEXRAY#58 *Erroneous Cycle Offset During Startup*

Revision(s) Affected Revision A and earlier

Details An abort of startup or normal operation by a READY command near the macrotick border may lead to the effect that the state INITIALIZE_SCHEDULE is one macrotick too short during the first following integration attempt. This leads to an early cycle start instate INTEGRATION_COLDSTART_CHECK or INTEGRATION_CONSISTENCY_CHECK. As a result the integrating node calculates a cycle offset of one macrotick at the end of the first even/odd cycle pair in the states INTEGRATION_COLDSTART_CHECK or INTEGRATION_CONSISTENCY_CHECK and tries to correct this offset. If the node is able to correct the offset of one macrotick (pOffsetCorrectionOut >> gdMacrotick), the node enters NORMAL_ACTIVE with the first startup attempt. If the node is not able to correct the offset error because pOffsetCorrectionOut is too small (pOffsetCorrectionOut ? gdMacrotick), the node enters ABORT_STARTUP and is ready to try startup again. The next (second) startup attempt is not effected by this erratum.

Scope: The erratum is limited to applications where READY command is used to leave STARTUP, NORMAL_ACTIVE, or NORMAL_PASSIVE state.

Implications: In the described case the integrating node tries to correct an erroneous cycle offset of one macrotick during startup.

Workaround(s) With a configuration ofpOffsetCorrectionOut >> gdMacrotick*(1+cClockDeviationMax) the node will be able to correct the offset and therefore also be able to successfully integrate.

FLEXRAY#59 *First WUS Following Received Valid WUP May Be Ignored*

Revision(s) Affected Revision A and earlier

Details When the protocol engine is in state WAKEUP_LISTEN and receives a valid wakeup pattern (WUP), ittransfers into state READY and updates the wakeup status vector CCSV.WSV[2:0] as well as the status interrupt flags SIR.WST and SIR.WUPA/B. If the received wakeup pattern continues, the protocol engine may ignore the first wakeup symbol (WUS) following the state transition and signals the next SIR.WUPA/B at the third instead of the second WUS.

Scope: The erratum is limited to the reception of redundant wakeup patterns.

Implications: Delayed setting of status interrupt flags SIR.WUPA/B for redundant wakeup patterns.

Workaround(s) None

FLEXRAY#60 *READY Command Accepted In READY State*

Revision(s) Affected Revision A and earlier

Details The E-Ray module does not ignore a READY command while in READY state.

Scope: The erratum is limited to the READY state.

Implications: Flag CCSV.CSI is set. Cold starting needs to be enabled by POC command ALLOW_COLDSTART (SUCC1.CMD = "1001").

Workaround(s) None

FLEXRAY#61 *Slot Status Vpoc!SlotMode Is Reset Immediately When Entering HALT State*

Revision(s) Affected Revision A and earlier

Details When the protocol engine is in the states NORMAL_ACTIVE orNORMAL_PASSIVE, a HALT or FREEZE command issued by the Host resets vPOC!SlotMode immediately to SINGLE slot mode (CCSV.SLM[1:0] = "00"). According to the FlexRay protocol specification, the slot mode should not be reset to SINGLE slot mode before the following state transition from HALT to DEFAULT_CONFIG state.

Scope: The erratum is limited to the HALT state.

Implications: The slot status vPOC!SlotMode is reset to SINGLE when entering HALT state.

Workaround(s) None

FLASH_WRAPPER#115 *Flash Clock Cycle is Late*

Revision(s) Affected	Revision A and earlier
Details	Flash Clock is one cycle late for some Flash Reads if test case enters software interface (SW_INTF) mode in pipeline mode. Flash API does not use SW_INTF, it is a test-only issue.
Workaround(s)	Applies to Factory test only. Enter non-pipeline mode before changing to SW_INTF mode.

FLASH WRAPPER#151 *Clearing Error Status Bit Blocks New Errors*

Revision(s) Affected	Revision A and earlier
Details	If a new error came in at the exact cycle where the CPU was clearing another error in the FEDACSTATUS register, then the new error would be lost. This may especially be evident if the CPU was clearing extra bits like the MULIT_ERR bit when it only needed to clear the single bit error status. Here the multi-bit error would be lost but the address register could set, and the freeze bit sets and blocks new interrupts. The UERR to the ESM would trigger but when the CPU investigates the source of the error it would see nothing in FEDACSTATUS.
Workaround(s)	Read and clear all bits in the status register, before doing any clear on the uncorrectable error address register.

FMZPLL#17 *FBSLIP And/Or RFSLIP Inadvertently Set On Power Up*

Revision(s) Affected Revision A and earlier

Details During power on, the FBSLIP and/or the RFSLIP bits in the system module (0xFFFFFFEC) may be inadvertently set along with the CLK source valid bit for the PLL (0xFFFFFF54). The signal to the SYS SLIP reset and the error signaling module is not affected.

Workaround(s) Write PLLCTL1[30:29] to binary 10. This clears the PLL clock source valid signal. Restore PLLCTL1 [30:29] to binary 01. Clear the slip bits in GLBSTAT register (0xFFFFFFEC).

FMZPLL#18 *PLL May Fail To Lock Or May Show PLL SLIP Immediately After Locking*

Revision(s) Affected Revision 0 only

Details The PLL may fail to lock or may show PLL SLIP immediately after locking. To fix this behavior the PLL wrapper will be changed to lock with FASTEN=0 and BWADJ=1. With the fixed wrapper, if PLLCNTL1[22] is set, the PLL will lock with the original settings of FASTEN=1 and BWADJ=7.

Workaround(s) None

FMZPLL#27 *Duty Cycle Incorrect When Output PLLCLK Divider Ratio Changed*

Revision(s) Affected Revision A and earlier

Details When the output pllclock divider ratio is changed (R divider), the duty cycle is not 50% (frequency is correct).

Conditions:

- When switching from odd to even ratio, the high phase is larger by half a pllclkcycle. (problem if low phase is less than spec freq allows).
- When switching from even to odd ratio, the high phase is smaller by half a pllclkcycle. (problem if high phase is less than spec freq allows)
- When switching to divide by one from either odd/even ratio, the duty cycle may not be 50%. It depends on the time MUX_SEL changes. If the MUX_SEL changes after the falling edge of the internal /1 clock, the high phase would be larger by that amount.
  - No clock phases (high or low) are less than the /1 clock phase.
  - No issue when switching from /2 to /1

Workaround(s) None

FTU#6 *FTU Registers Are Set Before TU_VBM_SDONE Is Asserted*

Revision(s) Affected Revision A and earlier

Details FTU registers, e.g., TSMOx (Transfer to System Memory Occurred) registers are set before TU_VBM_SDONE is asserted. The transfer to system memory may still be in the system (in the bridges or in CortexR4) when TSMOx flags are set. Any master accesses to system RAM before sdone is completed may receive incorrect data.

Workaround(s) To avoid this errata use the FTU mirroring feature.

FTU#8 *Transfer Unit Not Disabled On MPV Error*

Revision(s) Affected Revision A and earlier

Details On a memory protection error, the GC.TUE bit is reset. In some transfer scenarios, the MPVI error is set, but GC.TUE is not cleared. As a result, the FTU is not disabled.

- SM2CC - HDR+PL - Error during HDR transfer
  When an MPV error is signaled, the ongoing burst is stopped, but TUE is not cleared. The FSM remains in a "CIF_STATE_TTCC_WRHS" state and "ONGOING_BURST" will be toggling, but there will not be any transfer on the bus. (TU_VBM_REQ will not be activated). The safer option is to clear the TUE bit manually by software.
- CC2SM - HDR+PL - Error during Payload transfer
  When an MPV error is signaled, the ongoing burst is stopped, but TUE is not cleared. The FSM goes back to IDLE after an MPV error, but TUE is not cleared.
- CC2SM - Only Payload
  When an MPV error is signaled, the ongoing burst is stopped but TUE is not cleared. The FSM goes back to IDLE after an MPV error, but TUE is not cleared.

Workaround(s) This errata can be avoided in the following ways:

- In transfers from System Memory to Communication Controller, Transfer Payload only. Setup the header w/o using the FTU.
- Always on an MPV error, generate an interrupt and clear the GC.TUE by writing '1' to GCR.TUE register bit.

FTU#14 *Parity Calculation Occurs Even If Parity Protection For Transfer Configuration RAM(TCR) Is Off*

Revision(s) Affected Revision A and earlier

Details In test mode parity Calculation happens even if the Parity Protection for Transfer Configuration RAM(TCR) is switched off.

Workaround(s) None

FTU#19 *TCCOx Flag Clearing Masked*

Revision(s) Affected Revision A and earlier

Details According to the documentation, when TOOFF (Transfer Occurred Offset) is read, the corresponding message flag in TCCOx must be cleared. There may be a timing condition when TCCOx flag clearing could be masked due to the state machine clearing of TTCCx (Trigger transfer to communication controller) within the same cycle as software reading TOOFF. Due to the TCCOx flag not being cleared, the read of TOOFF register would not be up to date and would not reflect the last buffer completed.

Workaround(s) After reading the TOOFF to determine the highest buffer completed, read clear the corresponding flag in TCCOx.

GCM#47 *PLL Can Be Running Too Fast After Lock Period*

Revision(s) Affected Revision 0 only

Details In the devices or device revisions that do not have FMZPLL#18 or FMPLL.1 fix, PLL can be running too fast after lock period. When this happens, PLL cannot be disabled reliability and therefore can not restart PLL lock reliably.

Workaround(s) None

HTU#52 *VBUSP Register Interface Of HTU Does Not Have EMUDBG Port*

Revision(s) Affected Revision A and earlier

Details If COS is enabled, HTU should continue with the transfer even if there are debug accesses by CPU or DAP. As a result, HTU transfers may get halted for about 3 cycles.

Workaround(s) Enable the DEBM bit in HTU Global Control register.

HTU#53 *Buffer Full Flag Updated Incorrectly*

Revision(s) Affected Revision A and earlier

Details In some cases the buffer full flag is updated even if the current double control packet (DCP) has a bus error. The Stop bit is used to stop the transmission of the DCP containing the error. In the case when VCLK2=HCLK and VCLK=HCLK/2, the stop bit is getting asserted with a delay of four VCLK2 cycles. Due to this delay, the transmission is continuing with the remaining frames, despite the DCP bus error, and updating the buffer full flag with the current erroneous DCP value. The buffer full flag should not be updated if the current DCP has a bus error.

Workaround(s) None

LIN#46 *Super-Fractional Divider Table Mismatch For Synch Field And ID Field*

Revision(s) Affected Revision A and earlier

Details In case of super-fractional divider, the bits which will be transmitted (for synch and ID fields) with extra vclk period are different from the bits specified in super-fractional divider table, of BLIN spec. (Master mode)

Workaround(s) None

LIN#47 *Data Corrupted By New Header*

Revision(s) Affected Revision A and earlier

Details Slave response transmission interrupted by new header causes next response data to be corrupted.

Workaround(s) The software workaround is to check for a Bit Error flag during LIN data response transmission:

- During the response transmission, if "Bit Error" flag is set, this indicates that a collision has happened on the LIN bus.
- In the Bit Error ISR, configure the TD0 and TD1 registers with the next set of data to be transmitted (which will be transmitted on a TX Match for the incoming ID). Note that the rest of the code should check to see if there was an update to TD0/TD1 due to a Bit Error, and only write to TD0/TD1 if no previous update due to BE was done; otherwise the TX buffers might be written twice for an ID.
- Once the complete ID is received, based on the match, the newly configured data will be transmitted by the node.

LIN#48 *Header Not Received By Slave*

Revision(s) Affected Revision A and earlier

Details An incomplete frame header consisting of only Sync Break will cause the following complete header to not be received by slave. If the slave receives a header that is only a Sync Break (with no Sync Field), and the next incoming header is a complete header, then the slave gets stuck and is not able to receive the latest incoming header (and the receive buffer does not show the latest incoming header's ID). If the incomplete header consists of a Sync Break + Sync Field, then the next incoming header can be received with no errors. The issue is when the incomplete header is a Sync Break only.

Workaround(s) None

LIN#49 *Data Transmitted Correctly But First Byte Of Next Set Of Data Is Corrupted*

Revision(s) Affected Revision A and earlier

Details When LIN node is in SCI multi-buffered mode and if the buffer length is configured as less than 8, then the first set of data (i.e., programmed no. of bytes) is transmitted correctly but the transmission of the first byte of next set of data is corrupted. At the end of transmission of first set of data, the transmit buffer counter is cleared and then the internal TX Buffer will be loaded with the next byte to be transmitted. In the next cycle, this updated byte in TX buffer will be shifted to SCI TX SHF register which will be transmitted in the following cycle. If the buffer length is less than 8 (say x), then after the transmission of first set of “x” data, the transmit buffer counter is incremented to “x+1” before getting cleared to “0” in the next cycle. In this cycle (when TX Buffer counter has the incremented value “x+1”), internal TX Buffer will get the value of this non-selected byte i.e., “x+1” which will be shifted to TX shift register resulting in a corrupted/unwanted byte transmission. Also TX EMPTY flag is getting set at the start of STOP bit transmission instead of setting at the end of STOP bit. This issue is applicable only for Buffered SCI mode and for buffer length less than 8 bytes.

Workaround(s) When SCI Buffered mode is selected, always configure the buffer length to “8”.

LIN#50, LIN#65 *LIN Slave Mode Issues*

Revision(s) Affected Revision A and earlier

Details The LIN Slave Mode on this device requires multiple workarounds and are documented separately. If slave mode is required for your application, please contact TI for additional information.

Workaround(s) Please contact TI.

LIN#51 *Slave Not Transmitting Response To The First Header Following Wakeup*

Revision(s) Affected	Revision A and earlier
Details	The LIN is put in powerdown, and then some time later (in bench testing it was 1 second), a wakeup request of > 500us length is received. Roughly within 100ms after the wakeup request, the slave receives a header that it should respond to. When this occurs, the slave is waking up but is either not transmitting the response to the first header after wakeup, or the response is garbage.
Workaround(s)	None

LIN#54 *LIN TX Pin Continues Toggling Improperly*

Revision(s) Affected	Revision A and earlier
Details	During emulation mode LIN TX pin keeps toggling. LIN TX pin keeps toggling if we enter emulation mode during synch field generation.
Workaround(s)	None

LIN#57 *LIN Rejects First Byte Followed By Data Byte With A False Start Bit.*

Revision(s) Affected Revision A and earlier

Details LIN rejects the first byte followed upon after a data byte with a false start bit.

Workaround(s) The following are the scenarios which can occur in this situation:

- False start bit occurred during the data byte reception. In this case, along with the existing byte, next data byte also will get rejected. An NRE will occur in this case and suggested sequence for the SW. -- In the NRE ISR, write '0' to SW_nRESET bit (SCIGCR1.7) -- Write '1' to SW_nRESET (SCIGCR1.7).
- False start bit occurred during the checksum byte. In this case, the next valid start bit evaluation will happen on reception of a falling edge or on reception of an incoming synch break. Checksum byte will be rejected in this case and CE flag may get set depending on the expected checkbyte. In either case, the next incoming header will be received correctly and the module resynchronizes to the incoming header. An NRE will occur in this case and suggested sequence for the SW. -- In the NRE ISR, write '0' to SW_nRESET bit (SCIGCR1.7) -- Write '1' to SW_nRESET (SCIGCR1.7). If the next header is received before servicing NRE, then a frame error (FE) occurs (Since the next sync break is treated as a checksum byte) This FE should be ignored by the SW. SW has to ensure that all the LIN pending interrupts are serviced before the reception of next header.
- False start bit occurred during Identifier field. In this case, the incoming data byte will be treated as ID and ID parity error occurs. -- In the ID PE ISR, write '0' to SW_nRESET bit (SCIGCR1.7) -- Write '1' to SW_nRESET (SCIGCR1.7). In all the above cases, we'll never miss an incoming header and always resynchronizes to the incoming header.

LIN#58 *Incomplete Synch Field Reception Results In Loss Of Next Header*

Revision(s) Affected Revision A and earlier

Details When the node is receiving header and if the synch field reception is incomplete, then the next header will be missed. The node will flag an ISFE when the next header is received and discards the header. The following headers will be received correctly. So overall the node will miss one header after the faulty synch field reception.

Workaround(s) In Master mode, before triggering next header:

- Write '0' to SW_nRESET bit (SCIGCR1.7)
- Write '1' to SW_nRESET bit (SCIGCR1.7)

LIN#59 *LIN Transmission Incorrect*

Revision(s) Affected Revision A and earlier

Details If LIN is in non-multibuffer mode and if a Bit Error has occurred during the transmission of checksum byte, then after the reception of next header, there will be an idle period of 10 TBIT times after the first data byte transmission. This will impact the slave.

Workaround(s) On a Bit Error:

- Write '0' to SW_nRESET bit (SCIGCR1.7). (Enter SW Reset mode so that LIN logic is reset to IDLE state).
- Write '1' to SW_nRESET bit (SCIGCR1.7). (Exit from SW Reset mode)

LIN#60 *Start Bit Transmitted Within One Vclk During Extended Frame Transmission In Single Buffer Mode*

Revision(s) Affected Revision A and earlier

Details In extended frame mode, during single buffer mode transmission, when the TX Buffer (TD0) is loaded with new data during the STOP bit transmission of previous data byte, the next data byte transmission will be corrupted (i.e., start bit of next data byte will be corrupted). A write to TD0 (TX Buffer with new data) during ongoing transmission is recognized only during the transmission of current byte field (8-bits). Once the update is recognized, after the completion of STOP bit transmission (of current data byte), FSM will move from TX_STOP to TX_SHIFT state and start the transmission of newly written data byte. In the current failing scenario, when the TD0 buffer is written during STOP bit transmission (of current data byte), FSM will not recognize this and shift to TX_LOAD state from TX_STOP state. During this shift, the TX_VCLK_CNT counter gets cleared thrice resulting in 3 NEXT_BIT_TX pulses. This in turn triggers the transmit logic to start the transmission of start bit of next data byte on the 2nd NEXT_BIT_TX pulse and when the 3rd NEXT_BIT_TX pulse is received, the transmit logic switches to data byte (8-bits) transmission. This results in start bit getting transmitted for just 1 VCLK. This issue is applicable only when Extended Frame mode is enabled and Single buffer mode transmission is selected. This issue will occur on both Master and Slave nodes and all LIN versions.

Workaround(s) When in extended frame mode, during single buffer mode transmission, wait for the TX_EMPTY flag before loading TD0 buffer with the new data.

LIN#61 *TOA3WUP Flagged When New Header Is Received During Wait State*

Revision(s) Affected Revision A and earlier

Details This issue occurs whenever the new header is received during the 1.5sec suspend period and only when the module sleep/low power mode feature is used in application.

Workaround(s) One solution can be to ignore TOA3WUP interrupt. As the current issue will not impact any of the LIN communications, it is safe to ignore TOA3WUP interrupt. This is applicable to Master and Slave mode.

LIN#62 *FE Restarts Reception During Checksum Byte*

Revision(s) Affected Revision A and earlier

Details A frame error is flagged if a new header is received. When a frame error occurs during checksum byte reception, then the next incoming header is treated as data and another frame error will be flagged. The incoming header will be received correctly and the communication happens correctly.

Workaround(s) Following are the scenarios which can occur in this situation:

- Frame error occurred during checksum byte reception followed by an NRE. Suggested sequence for the SW.
  – In the NRE ISR, write '0' to SW_nRESET bit (SCIGCR1.7)
  – Write '1' to SW_nRESET (SCIGCR1.7)
- Frame error occurred during checksum byte reception followed by NRE and if the next header comes before the assigned slot (i.e., before servicing FE and NRE), then another FE will occur.

In this scenario, SW has to ignore the 2nd Frame Error. There will not be any impact on the incoming header reception or communication. This is applicable to Master and Slave mode.

LIN#63 *LIN Timeout Counters Should Stop During Power Down Mode*

Revision(s) Affected Revision A and earlier

Details During powerdown mode whenever there is a register access, clock to the module will be active until the access is complete. Since clock to the LIN registers and logic is same, clock to the timeout counters also will be active. This results in increment of the timeout counters and hence there will be a timeout interrupt after certain time (based on the frequency of the register accesses). Non-Critical Polling of registers continuously during powerdown mode is not expected. This issue is applicable if the module is in powerdown mode and if there is a continuous access to the registers. This issue will occur on both Master and Slave nodes and all LIN versions.

Workaround(s) To avoid this do the following:

- Don't poll the registers continuously if the module is in powerdown mode.
- SW has to ensure that the frequency of register accesses is less when the node is in powerdown mode.
- If polling of registers during powerdown mode is required, then ignore TIMEOUT interrupt during powerdown mode.

As the current issue will not impact any of the LIN communications, it is safe to ignore TIMEOUT interrupt in this case.

MIBSPI#93 *RELOAD Bit In TICKCNT Register Not Cleared Automatically*

Revision(s) Affected Revision A and earlier

Details As a result, the reload and count-down operation does not happen. This issue does not affect the normal functionality of the TICK COUNTER. It's just that RELOAD functionality will not work as expected. The RELOAD functionality is generally used when the TICK COUNTER is changed and to be synchronized to some event in the application. Otherwise, the internal reload will happen automatically.

Workaround(s) The hardware needs the rising edge of RELOAD bit values to start the reload process of pushing the new tick register value to tick counter. To do this the software needs to write 1, followed by write 0 every time the software wants to update current tick counter with new tick values.

MIBSPI#109 *BITERR Unreliable In Slave mode*

Revision(s) Affected Revision A and earlier

Details BITERR has a limitation in Slave mode that it will only work reliably at a ratio of SPICLK=VCLK/4 or slower. No other features are affected by BITERR. It's just that if BITERR is set for a transfer, the SPIBUF will reflect the same in the status field (SPIBUF(28)). This needs to be masked by the software. Same would be the case in Multibuffer Mode (if you use MibSPI), the RXBUF(28) needs to be masked for all the RXBUF locations.

Workaround(s) Use a ratio of SPICLK=VCLK/4 or slower for BITERR in Slave mode.

MIBSPI#110 *Multibuffer Slave In 3 or 4 Pin Mode Transmits Data Incorrectly*

Revision(s) Affected Revision A and earlier

Details Multibuffer slave in 3 or 4 pin mode transmits data incorrectly when there are back to back transactions at slow SPICLK ratios SPICLK < VCLK/12 with SPICLK PHASE =1 configuration

Workaround(s) Setting CSHOLD bit of Control field in TXRAM (even though it's not applicable to 3pin or 4pin $\overline{\text{ENA}}$ configurations) will avoid this issue.

MIBSPI#111 *Data Length Error Is Generated Repeatedly In Slave Mode*

Revision(s) Affected Revision A and earlier

Details A DLEN error is created in Slave mode of the SPI using nSCS pins in IO LoopBack Test mode. After the DLEN error is generated and the interrupt is serviced the SPI does not abort the ongoing transfer but continues generating DLEN error. The DLEN ERR is generated only once if the nENA pin of SPI is made as functional in SPIPC2 register. This is only an issue for a IOLPBK mode Slave in Analog Loopback configuration, when the intentional error generation feature is triggered using CTRL_DLENERR(IOLPBKTSTCR.16).

Workaround(s) After the DLEN_ERR interrupt is detected in IOLPBK mode, the SPIEN (SPIGCR.24) bit can be cleared once and set again.

MIBSPI#117 *Write To Unimplemented Buffer Overwriting The Corresponding Implemented Buffer*

Revision(s) Affected Revision A and earlier

Details In MIBSPI, if the RAM_SIZE specified is 32 buffers, write to 33rd buffer overwrites 1st buffer, write to 34th buffer overwrites 2nd buffer and so on. That is writes to unimplemented buffers are altering the implemented buffers.

Workaround(s) None

MIBSPI#118 *TXPIN Pin Has No Reset In MIBSPI*

Revision(s) Affected Revision A and earlier

Details The TXPIN (SIMO in case of Master, SOMI in case of Slave) pin comes up with undefined value after a power-up.

Workaround(s) Switch SIMO / SOMI to output mode after power up so that TXPIN_P reg will now reflect the value on either SIMO / SOMI pins.

MIBSPI#127 *DMACNTLEN Writable in Privilege and User Modes*

Revision(s) Affected	Revision A and earlier
Details	DMACNTLEN register does not behave according the defined access permissions. The documentation defines this register to be writable only in Privilege mode, but in silicon implementation this bit can be written in User mode as well.
Workaround(s)	None.

NHET#31 *MCMP Instruction Branches To Conditional Address When NAF Is Zero*

Revision(s) Affected	Revision A and earlier
Details	As per the pseudocode, if in angle mode, the MCMP instruction should branch to conditional address only when the magnitude compare is satisfied and the NAF flag is set. However the MCMP instruction in Angle mode branches to conditional address even when NAF is zero.
Workaround(s)	None

NHET#32 *SCMP Instruction Not Updating Cout_Prv Correctly*

Revision(s) Affected	Revision A and earlier
Details	The SCMP instruction is not updating cout_prv (i.e CF[25]) correctly when the selected register is lower than DF, and restart is disabled. Cout, Cout_prv, and CF[25] bit will be cleared when it should be set.
Workaround(s)	None

NHET#36 *Z Flag Not Set*

Revision(s) Affected Revision A and earlier

Details Z flag is not set in CNT instruction when data field is greater than the maximum field in GE mode.

Workaround(s) Do not write a value greater than the maximum count value, to the DF field. In time mode, the count is incremented by one, so the equality condition will get hit before the Greater than Equal to condition. This will be an issue only if the application writes a new value into the DF field which is greater than the already programmed MAX count value.

PBIST#3 *PBIST Module Functionality Affected When VCLK Equals HCLK*

Revision(s) Affected Revision A and earlier

Details The PBIST module functionality is affected under the following conditions:

- VCLK = HCLK
- PBIST ROM clock divide ratio is set after the PLL configuration

In the above mentioned condition the PBIST fails for STC ROM group.

Workaround(s) The PBIST ROM clock divide ratio should be set before the PLL setting and once the PLL setting is done the PBIST ROM clock divide ratio should never be altered.

RTP#35 *RTP Stops Tracing Data*

Revision(s) Affected Revision A and earlier

Details Synchronizing the RTP with an external tracer through the RTP_nENA pin with unaligned accesses can cause the RTP to stop tracing out data. This issue is applicable to unaligned-access transfers. An unaligned access is decoded and broken down by the RTP into aligned sub-access, i.e, a word(4bytes) starting at address 0x1, will be broken down to 3 aligned sub-accesses, a byte at address 1, halfword at address 2, and a byte at address 4(assuming BE-32 mode). All three aligned sub-accesses will then be traced out one after the other. If RTP_nENA is low, the RTP behaves correctly. RTP_nENA is sometimes driven high by the external tracer to signal it can no longer take data from the RTP, and requests the RTP to suspend tracing. If RTP_nENA is asserted at the end of the 2nd aligned sub-access, the 3rd sub-access can not start until RTP_nENA goes low. When RTP_nENA goes low, the RTP is not able to trace out the 3rd sub-access. The RTP is not able to recover.

Workaround(s) RTP transfers correctly with aligned accesses with toggling RTPnENA. There are two ways to remove unaligned accesses to the RTP. One is to remove unaligned accesses in software. The second is by setting BTCMRMW=1 in the R4 secondary auxiliary control register. In this case, R4 will issue all writes in 64-bit. RTP will transfer all writes as 64-bit data, so the number of transfer cycles would be different from when the BTCMRMW bit is cleared (i.e., R4 issues unaligned accesses).

SYS#46 *Switching Not Qualified With Clock Source On And Clock Source Valid*

Revision A and earlier

Details If user switches clock by writing to GHVSRC bits of GHVSRC register, the switching is not qualified with clock source on (CSDIS register) and clock source valid (CSVSTAT register).

Workaround(s) Always check the CSDIS register to make sure the clock source is turned on and the CSVSTA register to make sure the clock source is valid, and then write to GHVSRC to switch the clock.

SYS#103 *Pulse extension needed to ensure device reset.*

Revision(s) Affected	Revision A and earlier
Details	SYS_nRST requires additional VCLK cycles to ensure complete device reset. As a result, the nRST pulse is extended 7-8 VCLK cycles to meet this requirement. In certain conditions, when nRST is close to VCLK rising edge, the digital pulse extension logic may not get activated. When this happens, narrow nRST pulse is propagated to SYS_nRST, causing unpredictable device behavior.
Workaround(s)	Assert nRST for 7 to 8 VCLK cycles plus glitch filter value to ensure proper device reset.

TPIU#2 *TPIU Frame Synchronization Packets Not Output As*

Revision(s) Affected Revision A and earlier

Details In Normal and Continuous mode, the TPIU will not output frame synchronization packets when the frame synchronization counter reaches zero if there is formatted trace data awaiting output. This will depend on the bandwidth ratio between the trace port and that required by the trace sources. If the total bandwidth required by the trace sources is higher than that offered by the trace port, synchronization packets are unlikely to occur.

Conditions: This defect will be seen when the following conditions occur:

- The TPIU is programmed in Normal or Continuous modes
- Trace data packets are awaiting output

Trace packets will become stalled, awaiting output, when the bandwidth requirements of the input (ATB) exceed the output (the Trace Port). An example of when this is less likely to occur is when a 16-bit trace port is operating at greater than twice the clock rate of the 32-bit ATB (i.e., TRACECLKIN > 2xATCLK).

Implications: Tools should be aware that synchronization packets might not occur depending on the exact configuration and frequency of the TPIU. If the bandwidth offered by the trace port is not known or is slower than that required by the trace sources, normal and continuous mode should not be used, as synchronization packets output cannot be guaranteed. Instead bypass mode should be used and a single trace source selected. This does not affect the indication of triggers or idle cycles (no data present) when in normal mode as they are independent of the trace port data stream. However, when in continuous mode, synchronisation to the data stream is required to detect trigger and idle markers within the formatted data and under the above conditions, synchronisation to the data stream might be at risk. Also, when in continuous mode, the TPIU outputs synchronisation packets during idle cycles in the data stream provided that the idle cycles occur after the end of a frame and before the start of the next frame. These occurrences may limit the severity of this by providing some synchronisation packets within a trace session. This is dependent upon the trace source configuration and clock ratios present within the design and cannot be guaranteed to occur regularly

Workaround(s) None

TPIU#4 *Serializer Not Returned To The Same State As From Reset*

Revision(s) Affected Revision A and earlier

Details When ending a trace session, the serializer might not be returned to the same state as from reset. If starting a new session without resetting the debug logic, there may be extra data still within the serializer logic from the previous session. Also when changing the trace port width between sessions, the change will not initially be acknowledged by the serializing logic until after the first packet has been output. If the port width has increased then there will be a number of extra bits equal to the increase in port width; when decreasing port width, a number of bits equal to the reduction in port width will be lost (captured as zero).

Conditions:

- Increasing selected port size, (not from reset)
- Previous session used a stopping mode in port sizes other than 32, 16, 8, 4, 2 or 1 bit

  Data might be lost within the first word when decreasing the selected port size.

Implications: The beginning of a trace session (within the first 32 bits) might contain corrupt trace. Tools should not be affected by this, as no decompression should be attempted until synchronization has been detected within the trace stream.

Workaround(s) Do not attempt to decompress any trace data until the first synchronization sequence has been found.

TPIU#5 *TPIU Formatter Not Outputting Existing ID On Trace Reenable*

Revision(s) Affected Revision A and earlier

Details When trace capture is re-enabled and the source ID has not changed, the formatter does not output the current ID. The TPIU will still output the ID value on ID changes and periodically if it does not change.

Conditions:

- Trace capture is re-enabled (no reset between sessions)
- The first transaction logged has the same ATB ID as the last transaction before trace capture was disabled

Implications: If the ATB ID has not changed between trace sessions, and tools are not aware of this, up to 8 frames of trace. Will be lost when trace is re-enabled on the TPIU. No more than 8 frames of data will be lost (120 bytes), as the implementation of the formatter refreshes the ID if it has not changed every 8 frames.

Workaround(s) Decompressing the trace data stored in the TPIU will have to remember the last ID value output from one trace session to the next and use this as the default ID. Alternatively, an ATB ID change could be forced before trace capture is reenabled in the TPIU (e.g. by reprogramming trace sources to output different IDs to the previous session).

TPIU#6 *CSTPIU Unable To Disable Pattern Generator If Trace Data Is Output On Stop*

Revision(s) Affected Revision A and earlier

Details The Pattern Generator does not turn off when trace data is awaiting output following the disabling of the Pattern Generator. Trace data will become corrupted if this situation occurs until the trace port becomes idle, i.e., no trace data to be output. The Trace Port will not be idle when in Continuous mode or if the trace data input bandwidth is greater than the output data bandwidth for a 32-bit Trace Port (regardless of currently selected Trace Port width).

Conditions:

The following sequence must occur:

- The Pattern Generator is enabled (bit [16] or bit[17] is HIGH in the Current Test Pattern/Mode register at offset 0x204)
- Trace Data is ready to be output, either by Bit[1] HIGH in the Formatter & Flush Control Register at offset 0x304 OR Any trace source has been enabled and is generating trace data on the trace bus)
- The Pattern Generator is disabled (bits [17:16] == 00 at offset 0x204)

  The defect stops if there is a break in the trace data (indicated by the TRACECTL pin being HIGH) which cannot occur when in Continuous mode (bit [1] is HIGH in the Formatter & Flush Control Register at offset 0x304).

Implications: If the pattern generator is used in Continuous Mode, then no valid trace data will be output. If after the pattern generator is disabled when not in Continuous Mode, then some trace data might be lost. This does not affect use of the TPIU when the pattern generator has not been enabled (default from reset is disabled).

Workaround(s) It is recommended that when the tool does not require the use of the Pattern Generator that it disabled (default state at reset). When the use of the Pattern Generator is required it is recommended that:

(A) The TPIU is stopped (indicated by bit[1] in the Formatter & Flush Status Register at offset 0x300 is set LOW).

(B) The formatter is not in Continuous Mode (bit[1] of the Formatter & Flush Control Register at offset 0x304 is set LOW.

Then when use of the Pattern Generator has been completed the following sequence is recommended before enabling trace:

- The Pattern Generator is manually disabled (writing 0x00000 to the Current Test Pattern/Mode register at offset 0x204).
- The Current Test Pattern/Mode register is confirmed to be clear (a read of offset 0x204).
- Trace output is enabled as normal.

TPIU#7 *TPIU Unsynchronized Clock Domain Crossing In CSTPIU*

Revision(s) Affected Revision A and earlier

Details Combinatorial paths exist in control signals crossing the asynchronous clock boundary between ATCLK and TRACECLKIN. These signals control the reading and writing of data in the trace data FIFO on both sides of the ATCLK to TRACECLKIN clock boundary and therefore could cause old data to be repeated or new data to be lost.

Conditions:

ATCLK is asynchronous to TRACECLKIN

Implications: It is possible that race hazards could occur in the multi-bit one-hot encoded combinatorial path between the ATCLK and TRACECLKIN clock domains. If a race hazard occurs trace data might become corrupted, it is not possible to detect and correct any corrupt data.

Workaround(s) System implementors should make ATCLK and TRACECLKIN operate synchronously. To avoid the possibility of corrupted trace data, the Trace Port must be fed with a clock synchronous to ATCLK. Any crossing to an asynchronous TRACECLKIN domain should be done externally before the CSTPIU via a separate ATB asynchronous bridge.

TC RAM WRAPPER#27 *Read To RAMUERRADDR Or RAMPERRADDR Causes Registers To Clear Twice*

Revision(s) Affected Revision A and earlier

Details A read to RAMUERRADDR or RAMPERRADDR will cause these registers to clear twice. This might cause the address of a subsequent error which occurs during the read of the RAMxERRADDR register to be lost. The main issue is that a read from the RAMUERRADDRESS register causes two read-clear pulses to be generated. The gap between the two pulses is dependent on the HCLK/VCLK ratio programmed in the CLKCNTL register. For HCLK/VCLK = 2, there is 1 HCLK cycle between the two read-clear pulses, For HCLK/VCLK = 3, there are 2 HCLK cycles between the two read-clear pulses, and so on. The RAMUERRADDRESS register is frozen once an error address is captured. A read from this register unfreezes it and allows it to be overwritten by the address of the next uncorrectable error. Suppose there was an uncorrectable error. The CPU clears the DERR flag first and then reads from the RAMUERRADDRESS register. This generates two read-clear pulses. If the CPU signals two continuous multi-bit errors, one before and one after the second read-clear pulse, then the error address captured for the first multi-bit error is overwritten by the second multi-bit error address. The same also applies to the RAMPERRADDRESS register.

Workaround(s) None

TC RAM WRAPPER#28 *Single Error Generated Even When RAMTHRESHOLD Register Programmed To Zero*

Revision(s) Affected Revision A and earlier

Details According to the device specification a single error interrupt should not be generated when the RAMTHRESHOLD register is programmed to zero. However, a single error will be generated even when the RAMTHRESHOLD register is programmed to zero, while the internal counter over runs from maximum value to zero.

Workaround(s) To avoid this errata do the following:

- Clear SERREN in TCRAMW RAMINTCTRL reg bit when setting THRRSHOLD register value to zero. Single error interrupt generation is blocked when SERREN is zero.
- Disable the Interrupt line in ESM if this particular interrupt needs to be suppressed, instead of programming the Threshold register to zero.

TMS570LS#20 *BSR Logic Overrides INENA Signal For All IO Buffers Except When Test Logic Reset*

Revision(s) Affected Revision A and earlier

Details The INENA signal to the IO buffers were always enabled when debugger was connected, independent of the peripheral register configurations. The current BSR implementation makes the INENA as 1 for all other states than test logic reset. BSR tap runs in parallel with the ICEPICK tap. INENA signal will go HIGH as soon as TMS is asserted LOW by the debugger Impact in a Debug Environment.

- The peripherals will receive the input data even if INENA is disabled in the peripheral.
- Input buffer will consume more current if the input is floating.

Workaround(s) None

TMS570LS#23 *Write Access To Illegal Address Region Of MCRC Not Generating CPU Abort*

Revision(s) Affected Revision A and earlier

Details A normal mode Write Access to an illegal Address region of MCRC followed by a Debug Mode access does not generate CPU Abort because VBUSMSCR blocks the Error Response from MCRC.

Workaround(s) None

TMS570LS#29 *JTAG Connectivity Can Be Lost*

Revision(s) Affected Revision A and earlier

Details JTAG connectivity can be lost when the RTP/DMM scan path is enabled if HCLK is between 2.6x (8:3 ratio) and 8x (8:1 ratio) faster than TCK. The device design implements a 8 to 1 TCK/HCLK synchronizer in the RTP/DMM IcePick subpath. If HCLK is not 8x faster than TCK, the synchronizer will cause the RTCK provided to the Ice Pick and JTAG emulator to slow down. If this happens the JTAG emulator will provide data at a slower rate than TCK. The BSR (boundary scan module) was implemented to always sample data using the TCK input rather than RTCK. When TCK and RTCK are at different frequencies, the BSR could scan in incorrect data and cause all output buffers to be disabled, causing a loss of JTAG connectivity. If HCLK is less than 2.6x faster than TCK, the RTCK will slow the data rate to a point where the BSR will clock in only '1's or '0's. The Cortex R4 IcePick subpath does not have this limitation.

Workaround(s) To avoid this do the following:

- Use a JTAG emulator or RTP/DMM trace tool that supports adaptive clocking (TCK will slow down if RTCK frequency decreases).
- When using the RTP/DMM IcePick subpath, ensure that HCLK is never between 2.6x and 8x faster than TCK.

TMS570LS#30 *BSR Logic Should Be In Bypass Mode*

Revision(s) Affected Revision A and earlier

Details The BSR logic should be in bypass mode when the device is in functional mode. This can be done by gating the instruction register inside BSR with (ICEPICK_CONDEXT OR ICEPICK_CONDBYPASS).

Workaround(s) None