TMS470MF0660x
Microcontroller
Silicon Revision B

Silicon Errata

Literature Number: SPNZ179A
November 2010–Revised November 2015

Contents

List of Figures

List of Tables

TMS470MF0660x Microcontroller **Silicon Revision B**

1 Introduction

This document describes the known exceptions to the functional specifications for the TMS470MF0660x devices. For more detailed information on this device, see the device-specific data sheet:

- *TMS470MF0660x 16/32-BIT RISC Flash Microcontroller* data manual (Literature Number SPNS157)

1.1 Device Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all devices. Each commercial family member has one of three prefixes: TMX, TMP, or TMS (e.g.,TMS470MF0660xPZ). These prefixes represent evolutionary stages of product development from engineering prototypes (TMX) through fully qualified production devices/tools (TMS).

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.

TMX and TMP devices are shipped against the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

TMS devices 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, PZ), 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.

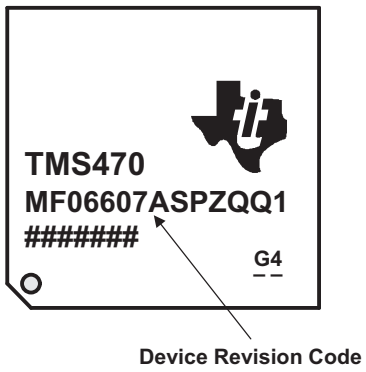

Figure 1. Example, Device Revision Code for TMS470MF0660x (PZ Package)

Silicon revision is identified by a device revision code. The code is of the format TMS470MF06607x, where "x" denotes the silicon revision. If x is "B" in the device part number, it represents silicon version B. Table 1 lists the information associated with each silicon revision.

Table 1. TMS470MF0660x Device Revision Codes

DEVICE PART NUMBER DEVICE REVISION CODE	SILICON REVISION	PART NUMBERS/COMMENTS
TMX470MF0660x	Initial	This silicon revision is available as TMX *only*. **TMX**470MF0660x
TMS470MF0660xA	A	This silicon revision is available as TMX *only*. **TMX**470MF0660x
TMS470MF0660xB	B	This silicon revision is available as TMS **TMS**470MF0660x

2 Known Design Exceptions to Functional Specifications

Table 2. Advisory List

Table 2. Advisory List (continued)

CORTEX-M3#18 *Async Not Generated If No Trace In Previous Session*

Revision(s) Affected Revision B and earlier

Details The ETM is required to generate an alignment synchronisation packet as the first packet every time the programming bit is cleared, in order to allow the tools to synchronize with the protocol stream. If no trace is generated in a session, the A-sync packet will be correctly generated as the progbit is cleared, but if the progbit is then set and cleared again, second and subsequent A-sync packets are not generated. No other aspects of the trace generation are affected by this, and once any other trace packet is generated the A-sync logic will be correctly reset.

Conditions:

This erratum only occurs when the following sequence of events occurs:

- The ETM is programmed and enabled.
- No instructions are traced.
- The ETM programming bit is set.
- The ETM programming bit is cleared

  The absence of any traced instructions can be due either to the Trace Enable conditions not being satisfied, or the core not executing any instructions

Implications: This erratum only affects the trace stream which is generated by the ETM. There is no impact on the normal processing operation of the core. The conditions which cause this erratum are expected to occur whilst debug is being performed. The A-sync packet may be missing at the start of a trace session. All trace packets which are generated will still be correct, and there is no corruption of the trace stream. If the A-sync from a previous session was captured, it can be used for synchronisation. Tools will typically fail to capture the trace relating to any bytes of trace before the next periodic A-sync packet is seen. For short trace sessions, it is possible to lose the whole trace session

Workaround(s) Users may attempt to work around this erratum by forcing the generation of trace packets as part of an initialization sequence. This is only possible if the core is not in debug state, and so may not be applicable in all cases. With this ETM it is also possible to deduce that the first byte of trace captured after the progbit is cleared will be the start of a packet, provided that the formatter in the TPIU is enabled.

CORTEX-M3#19 *Trigger Packets Sometimes Not Inserted In Trace Stream*

Revision(s) Affected Revision B and earlier

Details It is possible to configure a trigger event for the ETM which is used to assist with trace capture and the subsequent analysis of trace by the user. The trigger condition is indicated by a pulse on the ETMTRIGOUT signal, and is also inserted in the trace stream using a special packet. If a trigger condition occurs when there is no data in the ETM's FIFO and there are no instructions yet to be traced, the ETMTRIGOUT signal is pulsed correctly, but the trigger packet is not inserted in the trace stream.

Conditions:

- The ETM is enabled.
- The trace FIFO is empty.
- There are no instructions already executed but not yet entered in the trace FIFO.

Implications: This erratum only affects the trace stream which is generated by the ETM. There is no impact on the normal processing operation of the core. This erratum only affects the inclusion of the trigger packet in the trace stream. It does not affect the visibility of the trigger condition through the ETM's programmers model (bit 2 of the ETM Status Register, register 4). It does not affect the indication of the trigger condition to the trace capture device, and a formatter trigger packet will be inserted if enabled. The erratum does not occur if a trigger is generated using DWT to detect an instruction address if Trace Enable is high. When this erratum occurs, the user will not be able to determine the location in the executed instruction sequence at which the trigger condition occurred. Due to the conditions which are required for this erratum to occur, it is more likely that the erratum occurs when an external input is being used to generate the trigger condition.

Workaround(s) If using the CoreSight formatter protocol, e.g. TPIU/ETB with trigger embedded, then the position of the embedded trigger can be used, however the location of the embedded trigger is approximate.

CORTEX-M3#20 *BKPT In Debug Monitor Mode Can Cause DFSR Mismatch*

Revision(s) Affected Revision B and earlier

Details A BKPT may be executed in debug monitor mode which will cause the debug monitor handler to be run but the Debug Fault Status Register (DFSR) at address 0xE000ED30 will not have bit 1 set to indicate the cause was a BKPT instruction. This will only occur if an interrupt other than the Debug Monitor is already being processed just before the BKPT is executed.

Conditions:

- C_DEBUGEN (bit 0) in the Debug Halting Control and Status Register at address 0xE000EDF0 is 0.
- MON_EN (bit 16) in the Debug Exception and Monitor Control Register at address 0xE000EDFC is 1.
- An enabled interrupt occurs two cycles before the BKPT is executed that causes a preemption.

Implications: The Debug Monitor handler may be entered without the DFSR revealing the cause of the handler being entered.

Workaround(s) If the DFSR does not have any bits set when the debug monitor has been entered then the cause must be due to this corner case and that it was the result of a BKPT.

CORTEX-M3#21 *Core May Freeze For SLEEPONEXIT Single Instruction ISR*

Revision(s) Affected Revision B and earlier

Details

The SLEEPONEXIT functionality causes the core to enter the sleep mode when the exit from the sole active interrupt occurs. This means that there are no more interrupts active and the exit would have caused a return to the thread. It is possible for the core to become frozen if the SLEEPONEXIT functionality is used and the interrupt service routine (ISR) concerned only contains a single instruction. This freezing may occur if only one interrupt is active and it is pre-empted by an interrupt whose handler only contains the single instruction. This instruction must be a legal ISR exit instruction that takes one cycle to execute (either a BX or a BLX). In this case the unstacking would occur after the single instruction had been executed as normal to return to the now only active interrupt handler. However, once it has returned no more instructions will be processed and the core will be frozen. Any new preempting interrupt will unfreeze the processor.

Conditions:

- SLEEPONEXIT (bit 1) in the System Control Register at address 0xE000ED10 is set.
- An interrupt occurs that causes a preemption of the current ISR which is the only interrupt that is currently active.
- The interrupt service routine that is entered consists of only one instruction (either BX or BLX) which causes a legal exit from that ISR.

Implications: The core may freeze and stop processing instructions when it returns to the only currently active ISR. Note that a new interrupt that causes a preemption would cause the core to become unfrozen and behave correctly again.

Workaround(s) If the SLEEPONEXIT functionality is required then do not allow an ISR to contain only one instruction. If an empty ISR is used then insert a NOP before the exit instruction.

CORTEX-M3#22 *Unaligned MPU Fault During A Write Causes Wrong Data Written To Successful First Access*

Revision(s) Affected Revision B and earlier

Details

When an unaligned store is executed by Cortex-M3 the transaction is split up into either two or three aligned transactions forming constituent parts of the larger transaction. The MPU will check that these transactions are permitted and will block them if necessary. If an unaligned transaction occurs where it overlaps two MPU regions then each region relating to the part of the transaction that hits that region will be checked. If an unaligned store occurs that crosses an MPU region boundary and has an MPU permission fault for the second region check but not for the first region then it is possible for the second component's data to be written for the first successful transaction in place of the first transaction's data. This can occur for writes to either the D-Code or system bus but will only occur if one or more wait-states are applied for the first component of the store.

Conditions:

- The full MPU is present and enabled with at least one region programmed and enabled.
- An unaligned store is executed by the processor. The store can be to either the D-Code or the System bus.
- The store crosses an MPU region boundary.
- The first region lookup passes, the second region lookup fails. 5. One or more wait states are applied via HREADYS or HREADYD.

Implications:

The wrong data will be stored to a permitted address. However, a MemManage fault will occur immediately pointing to the instruction that caused the fault. This may lead to the instruction being re-executed and the store occurring successfully if it is for non-device memory. This would mean that the previously stored data would be overwritten and the wrong value would never be seen. This may not be true for a shared memory system.

Workaround(s)

A workaround is only required if the MPU is present and enabled. Do one of the following:

- Do not allow accesses to span more than one region
- Do not allow unaligned accesses at all
- Program the MPU correctly if applicable

Secondary workaround:

M3 can be configured to generate a usage fault upon ANY unaligned access by enabling UNALIGN_TRP bit in the M3 Configuration Control Register. If the user does not intend to ever do an unaligned write, then this would be the preferred workaround, as it would prevent incorrect data from being written.

CORTEX-M3#23 *Unaligned MPU Fault During A Write Causes Wrong Data To Be Written*

Revision(s) Affected Revision B and earlier

Details

When an unaligned store is executed by Cortex-M3 the transaction is split up into either two or three aligned transactions forming constituent parts of the larger transaction. The MPU will check that these transactions are permitted and will block them if necessary. If an unaligned transaction occurs where it overlaps two MPU regions then each region relating to the part of the transaction that hits that region will be checked. If an unaligned store occurs that crosses an MPU region boundary and has an MPU permission fault for the second region check but not for the first region then it is possible for the second component's data to be written for the first successful transaction in place of the first transaction's data. This can occur for writes to either the D-Code or system bus but will only occur if one or more wait-states are applied for the first component of the store.

Conditions:

- The full MPU is present and enabled with at least one region programmed and enabled.
- An unaligned store is executed by the processor. The store can be to either the D-Code or the System bus.
- The store crosses an MPU region boundary.
- The first region lookup passes, the second region lookup fails.
- One or more wait states are applied via HREADYS or HREADYD.

Implications: The wrong data will be stored to a permitted address. However, a MemManage fault will occur immediately pointing to the instruction that caused the fault. This may lead to the instruction being re-executed and the store occurring successfully if it is for non-device memory. This would mean that the previously stored data would be overwritten and the wrong value would never be seen. This may not be true for a shared memory system.

Workaround(s)

A workaround is only required if the MPU is present and enabled. Do one of the following:

- Do not allow accesses to span more than one region.
- Do not allow unaligned accesses at all.
- Program the MPU correctly if applicable.

Additional Workaround: M3 can be configured to generate a usage fault upon ANY unaligned access by enabling UNALIGN_TRP bit in the M3Configuration Control Register. If the user does not intend to ever do an unaligned write, then this would be the preferred workaround, as it would prevent incorrect data from being written.

CORTEX-M3#24 *Cortex-M3 Fetches Instructions Using Incorrect Privilege On Exception Return*

Revision(s) Affected Revision B and earlier

Details While unstacking registers on return from an exception to a User-privilege thread, Cortex-M3 attempts to simultaneously prefetch the thread's instruction stream. Before the register unstacking is complete, up to the first three memory transactions used to perform instruction prefetching may be erroneously marked as Privileged. This may allow between three and six instructions from a Privileged-access-only region to be executed by a User-privilege thread. Once fetched, the instructions are executed with User-privilege. Instruction fetches performed after register unstacking has completed will be performed with User-privilege. Both the register unstacking, and any data transactions generated by executing the erroneously executed instructions will be performed and correctly marked as User-privilege.

Conditions:

- Exception return is executed.
- The exception return is to user code.

Implications: User-privileged code may contrive a situation in order to allow execution of up to three words worth of instructions intended to be accessible to Privileged-only execution; however, execution of said instructions will always be performed with User-privilege, thus there are no additional capabilities provided to User-privilege through this erratum. There exists a theoretical possibility that User-privilege code could use this erratum to allow limited extraction of code and or data from Privileged-access only memory. Note that read sensitive Privileged-access only peripherals should always be placed in an XN region either via the default memory map, or via the optional memory-protection-unit. Alternatively such peripherals should ignore transactions with HPROT[0] indicating that the transaction is an instruction fetch.

Workaround(s) None

CORTEX-M3#25 *DWT CPI Counter Increments During Sleep*

Revision(s) Affected Revision B and earlier

Details The DWT contains a number of counters for the profiling of applications. The CPI counter is used to indicate the total number of clock cycles beyond the first cycle of each instruction. The counter is specified to not increment whilst the core is sleeping but for previous revisions it does increment. This results in sleep cycles being counted as program execution cycles.

Conditions:

- The CPI counter in the DWT is enabled.
- Core sleeps during profiling.

Implications: Profiling information could be calculated incorrectly if the following calculation is used:

InstructionCount = CycleCount (CPIcount+LSUcount+INTcount+SLEEPcount) + FOLDcount

Workaround(s) The number of sleep cycles given by SLEEPCNT can be subtracted from the CPI cycle count to obtain the correct CPI cycle information. Use the following equation:

InstructionCount = CycleCount (CPIcount+LSUcount+INTcount) + FOLDcount

CORTEX-M3#26 *Cortex-M3 TPIU Clock Domain Crossing*

Revision(s) Affected Revision B and earlier

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

Conditions:

- FCLK/HCLK is asynchronous to TRACECLKIN

Implications: When FCLK and TRACECLKIN are asynchronous and depending on the silicon implementation of the block, trace data might become corrupted.

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

CORTEX-M3#27 *Internal write buffer Active While In Sleep Mode*

Revision(s) Affected Revision B and earlier

Details

If a store immediate that is marked as not strongly ordered is used immediately before a WFE or WFI then the store may still be in progress when the core has asserted the SLEEPING signal. This will only occur if wait states are applied to the store operation. This will not cause a problem unless the location that the store was accessing was using a free-running clock whilst a clock-gating cell has been used to gate FCLK to form HCLK.

Conditions:

- A store with immediate offset is executed.
- The store operation is allowed to be bufferable
- The store is followed immediately by a WFI or WFE.
- Wait-states are used to delay the data-phase of the store.
- A clock-gate is used to produce HCLK which is gated when SLEEPING is asserted.
- The location the store was to is using a free-running version of the core clock (FCLK).

Implications : An imprecise error response could be missed if it is issued by the peripheral whilst the core is asleep when the peripheral is not using a gated clock but the core is. The stored data will be correct as the core will hold HWDATA at the correct value until it wakes from sleep and completes the transaction.

Workaround(s) Insert DSB instructions before any WFE and WFI instructions in the application code.

CORTEX-M3#30 *SWJ-DP Internal Reset Synchronizer*

Revision(s) Affected Revision B and earlier

Details In Cortex-M3 r0p0 and r1p0, the SWJ-DP has an internal reset synchronizer for the power on reset signal. Version r1p1 was upgraded to a newer version of SWJ-DP and this SWJ-DP did not have the same reset synchronizer.

Workaround(s) The r1p1 CortexM3Integration level can be modified to add the reset synchronizer. The signal which has the problem is the nPOTRST input on SWJ-DP. Add a few new signals: reg nPOTRSTQ; // DFF #1 reg nPOTRSTQQ; // DFF #2 wire inPOTRST; // reset bypass MUX // Add the synchroniser DFFs: // nPOTRST synchroniser always @ (posedge SWCLKTCK or negedge PORESETn) if(!PORESETn) begin nPOTRSTQ <= 1'b0; nPOTRSTQQ M <= 1'b0; end else begin nPOTRSTQ <= 1'b1; nPOTRSTQQ <= nPOTRSTQ; end // And a reset bypass MUX: assign inPOTRST = RSTBYPASS PORESETn : nPOTRSTQQ; // And connect the MUX output inPOTRST to the nPOTRSTinput of SWJ-DP: DAPSWJDP uDAPSWJDP (// Inputs .nPOTRST (inPOTRST)

CORTEX-M3#31 *LDRD With Base In List May Result In Incorrect Base Register When Interrupted Or Faulted*

Revision(s) Affected Revision B and earlier

Details

LDRD with the base register in the list of the form LDRD Ra, Rb, [Ra] with or without an immediate may be interrupted or faulted after the load of the first register but before the completion of the second load. Since the base register has been updated the base register must be restored to its original value before entering the appropriate interrupt or fault handler so that the instruction can restart correctly upon return from the handler. In certain circumstances this may not occur as required. When the LDRD is interrupted in between the two loads then the base register may not be restored as required. This can only happen when the instructions are being executed from the system bus (address 0x20000000 and above) and the loaded data is also being read from the system bus. For the fault case where the second load gets a bus fault or an MPU fault then the base register is never restored and there is no dependence on which bus the instructions are being executed from. When the base register is the second register in the LDRD list then this erratum cannot occur.

Conditions:

Either

- 1. An LDRD is being executed where the base register is in the list and write-back is not used: LDRD Ra, Rb, [Ra, #imm]
- Instructions and data are both not located in the code space and are both being fetched via the system bus. This occurs for locations in memory greater than0x20000000.
- An instruction fetch and the first load for the LDRD are issued internally in parallel to the bus-matrix which arbitrates between them.
- An interrupt occurs in between the two load operations.

Or

- An LDRD is being executed where the base register is in the list and write-back is not used: LDRD Ra, Rb, [Ra, #imm]
- A bus fault or MPU fault occurs for the second load.

Implications: The base register will not be restored as expected preventing the instruction from being restarted correctly.

Workaround(s)

This form of LDRD may be used as an optimization to re-use the base register directly and reduce register use pressure. This instruction can be directly replaced by two LDR instructions which will produce the same functionality. For Cortex-M3 r2p0 it is possible to prevent this behavior for the interrupt case by setting DISMCYCINT (bit[0]) in the Auxiliary Control Register which is located at address 0xE000E008. Setting this bit will increase the interrupt latency of Cortex-M3. Setting DISMCYCINT does not prevent the second load being faulted which means that the base will still be incorrect for bus faults or MPU faults. Alternatively, if the instructions are always executed from the code space and faults cannot occur then a workaround is not required.

CORTEX-M3#32 *Cortex-M3 Reserved Interface Allowing An External Interrupt Controller Connection*

Revision(s) Affected Revision B and earlier

Details

An issue can arise with this interface when a sequence of events occurs. The occurrence of this issue is dependent on the PC being written with an interrupt service routine exit value the cycle after a new higher priority interrupt was asserted. When this event occurs it means that an ETMINTSTAT of 3 b010 (interrupt exit) is issued in preference to ETMINTSTAT of 3 b100 (new interrupt occurring). The ETMINTSTAT of 3 b100 is issued in the cycle after 3 b010 which is too late to allow the external system to respond as specified. This event by itself does not cause any problems until HREADYI is held low from the last instruction fetch in the ISR until four cycles after the interrupt was asserted. The last instruction fetch will never be used because the ISR has completed but since the fetch is still outstanding HREADYI is held low until the data is returned. With a fixed wait-state system the fetch buffer will need to be in a particular state such that the last fetch of the ISR returns HREADYI four cycles after the interrupt was asserted. This means that producing this issue is very dependent on the sequence of instructions before the ISR exit instruction as well as the number of wait-states used in the system.

Conditions:

For this issue to arise the following conditions must occur.

- VECTADDREN is tied to 1.
- The PC must be written with the interrupt service routine exit code the cycle after the IRQ has been asserted.
- The IRQ that is asserted must be enabled and higher priority than the current interrupt so that it causes a preemption.
- The asserted IRQ must be able to be re-mapped (INTISR[7:0] only).
- The fetch buffer and sequence of instructions must allow the correct conditions for a fetch to be started before the interrupt is asserted.
- IHREADY must be held low and then asserted four cycles after the interrupt was asserted. This means that the minimum number of wait-states before this issue can be seen is four.

Implications: When this event occurs the re-mapped vector table entry will not be used by the core and instead the address located in the vector table will be used. This entry may not be correct and an undesired interrupt service routine may be entered and executed.

Workaround(s)

Both software and hardware fixes are available to prevent this issue from occurring. The software fix consists of inserting a "CPSID f" instruction before the exit instruction of every interrupt service routine except NMI and HARDFAULT. This instruction will set FAULTMASK and prevent a preemption in the same cycle as the exit and stop condition 3 from occurring. When the exit is executed then FAULTMASK will be cleared automatically and the system will return to the same set up as before, i.e. you don't need to manually clear FAULTMASK. The consequence of this is that interrupt latency will be delayed by about 2 cycles but this is dependant on the alignment of the code and wait-states. The delay will be the time between the execution of the CPS (setting FAULTMASK) and the execution of the exit instruction (clearing FAULTMASK). The hardware fix addresses the remapping logic external to Cortex-M3 by asserting VECTADDREN in a different method compared to the original specification. This will always ensure that the remapped vector is always fetched. // Logic instantiated external toCortexM3Integration reg REMAPPING; wire VECTADDREN; // VECTADDREN generation always @(posedge HCLK or negedge HRESETn) if(!HRESETn) REMAPPING <= 1'b0; else if ((ETMINTSTAT == 3'b001) | (ETMINTSTAT == 3'b100)) REMAPPING <= ETMINTSTAT[2];assign VECTADDREN = ETMINTSTAT [2] | REMAPPING; // Inside CortexM3Integration.v assign ETMPWRUP = 1 b1;

CORTEX-M3#34 *ETM Traces BKPT As An Executed Instruction*

Revision(s) Affected Revision B and earlier

Details When tracing program execution using the ETM an extra instruction is traced if an exception is caused by a software or hardware breakpoint or if the processor halts due to a software or hardware breakpoint.

Conditions:

- BKPT instruction is executed and causes the processor to halt or enter the debug monitor, in this case the BKPT instruction is incorrectly traced.
- A hardware breakpoint causes the processor to halt or enter the debug monitor, in this case the instruction which matches the hardware breakpoint is incorrectly traced.
- A hardware watch point matches and causes the processor to halt or enter the debug monitor, in this case the instruction which causes the hardware watch point to match is incorrectly traced If the processor halts due to an External or Internal debug request or a Vector catch, the correct instructions are traced.

Implications: Trace analysis tools will incorrectly consider the extra instruction to have executed.

Workaround(s) If entry to the debug monitor is traced, the instruction traced immediately before the entry to the debug monitor was not executed and must be discarded. If the processor halts due to a hardware or software breakpoint, the instruction traced immediately prior to the halt was not executed and must be discarded. The reason for halting can be determined from the Debug Fault Status Register (DFSR).

CORTEX-M3#35 *HPROT And MEMATTR Incorrect On Some Unaligned Transactions*

Revision(s) Affected Revision B and earlier

Details An unaligned word access starting with the lowest 4 bits being 0xF will be split into three bus transactions by the processor. If these three bus transactions cross a memory region (either defined by the MPU or the default memory map) then incorrect memory attributes may be asserted on the HPROTS and MEMATTRS outputs for the second access if the store is to the SYSTEM bus. The first byte access will correctly have the attributes of the first region. The second half-word access will incorrectly have the attributes of the first region when it should have the attributes of the second. The third byte access will correctly have the attributes of the second region.

Conditions:

- An unaligned word access occurs.
- The access has an address greater than 0x20000000.
- The lower four bits of the address are 0xF.
- The transfer crosses two memory regions (either MPU or default memory map).
- The two regions have different memory attributes 6. The SYSTEM bus MEMATTRS or HPROTS are used externally to Cortex-M3.

Implications: A system utilizing HPROT and MEMATTR (either system or DCODE) may receive the incorrect information for the data being loaded or stored.

Workaround(s) No workaround is required if the HPROT and MEMATTR output pins are not used or if unaligned accesses do not occur. To ensure unaligned accesses do not occur the UNALIGN_TRP bit of the Configuration and Control Register at address 0xE000ED14 can be set. This will cause all unaligned accesses to produce an exception.

CORTEX-M3#36 *Incorrect Core Feature Identification Registers*

Revision(s) Affected Revision B and earlier

Details The following identification register fields return the wrong value when read:

- Instruction Set Attributes Register0 (ID_ISAR0). Bits 19:16 should read as 0 instead of 4 as no coprocessor instructions are supported.
- Instruction Set Attributes Register4 (ID_ISAR4). Bits 7:4 should read as 3 instead of 0 to indicate that constants shifts are supported on loads/stores and some DP operations.
- Memory Model Feature register0 (ID_MMFR0). Bits 23:20 should read as 1 instead of 0 to indicate that an auxiliary control register is available.
- Memory Model Feature register2 (ID_MMFR2). Bits 27:24 should read as 1 instead of 0 to indicate that wait for interrupt is supported.
- Debug Features register0 (ID_DFR0). Bits 23:20 should read as 0 instead of 1 when no debug is present to indicate that the debug model is not supported.

Workaround(s) None

CORTEX-M3#37 *Bit-Band Access Could Read Or Write Wrong Bit In BE8*

Revision(s) Affected Revision B and earlier

Details In some circumstances, if an access to a bit-band alias region is immediately followed by another access to a bit-band alias region which has a different access size compared to the first access then the wrong bit may be returned or modified when the core is used in big-endian mode.

Conditions:

- The core is running in big-endian mode (BIGEND input set at reset).
- Bit-banding is utilized.
- An access occurs to a bit-band alias region.
- A second access occurs to a bit-band alias region immediately after the first one.
- The two accesses have different size attributes.
- The memory transaction inserts at least one wait state.

Implications: For store operations the wrong bit of the bit-band region may be modified. For load operations the wrong bit of the bit-band region may be read

Workaround(s) A workaround is only required if big-endian mode is used, bit-banding is utilized and the memory has wait states. If this is the situation then two workarounds are possible: Either always access the bit-band alias with the same size attributes, or stop consecutive bit-band alias accesses by inserting a NOP in between them.

CORTEX-M3#39 *TBH Will Never Cause An Alignment Fault*

Revision(s) Affected Revision B and earlier

Details For ARM v7M the following data accesses support unaligned addressing, and only generate alignment faults when the CCR.UNALIGN_TRP bit is set in the Configuration and Control Register:

- Non halfword-aligned LDR{S}H{T} and STRH{T}
- Non halfword-aligned TBH
- Non word-aligned LDR{T} and STR{T}

Cortex-M3 does cause an alignment fault for the load and store cause but it does not generate a fault for the TBH case.

Conditions:

- A TBH instruction is executed.
- CCR.UNALIGN_TRP bit is set in the Configuration and Control Register at address 0xE000ED14.
- The address is not halfword aligned.

Implications: An alignment fault will not be generated when it should have been. This will occur when the user is trying to ensure all accesses are aligned and sets the unaligned trap enable. With this errata the user may not spot that a TBH was unaligned.

Workaround(s) This only occurs when the unaligned trap has been enabled and the user is trying to identify all unaligned accesses. Ensure that all TBH instructions are aligned as required.

CSHTM#10 *TraceOff Packet Duplicated At End Of Trace Session*

Revision(s) Affected Revision B and earlier

Details The HTM uses a TraceOff packet to indicate the ending of trace, or stopping of trace capturing if the trace start/stop control inside the HTM is used. It has a value of 0x28, as documented in the chapter 4 of HTM TRM. The TraceOff packet is output by the HTM at the end of each trace session, when the PROG bit in the HTMCONTROL register is set. If the PROG bit is set while the internal pipelined HREADY signal in the AHB being traced is high, the TraceOff packet can be output twice. This is caused by the TraceOff packet generation which can be triggered by two separated mechanisms: setting of PROG bit, and turning off of the Trace Enabling block. These two sources are offset by 1 clock cycle. The second mechanism only triggers when the pipelined HREADY signal is high. So the duplicated TraceOff packet scenario depends on the AHB state.

Conditions: Pipelined HREADY is high when PROG bit is switched from 0 to 1.

Implications: The result trace may result in an extra TraceOff packet at the end of a trace session. However, this does not affect the decoding of trace information.

Workaround(s) Trace decompression software can ignore the extra TraceOff packet.

CSHTM#11 *ASYNC Packet Generation Suppressed By HREADY Highs*

Revision(s) Affected Revision B and earlier

Details The HTM uses a TraceOff packet to indicate the ending of trace, or stopping of trace capturing if the trace start/stop control inside the HTM is used. It has a value of 0x28, as documented in the chapter 4 of HTM TRM. The TraceOff packet is output by the HTM at the end of each trace session, when the PROG bit in the HTMCONTROL register is set. If the PROG bit is set while the internal pipelined HREADY signal in the AHB being traced is high, the TraceOff packet can be output twice. This is caused by the TraceOff packet generation which can be triggered by two separated mechanisms: setting of PROG bit, and turning off of the Trace Enabling block. These two sources are offset by 1 clock cycle. The second mechanism only triggers when the pipelined HREADY signal is high. So the duplicated TraceOff packet scenario depends on the AHB state.

Conditions: Pipelined HREADY is high when PROG bit is switched from 0 to 1.

Implications: For systems with no wait states or wait states of only one cycle, the ASYNC packet will always be suppressed. As a result external trace decompression software cannot decompress the trace. For systems with transfers with two or more wait states the ASYNC packet could be delayed, causing the beginning of the trace to be lost. Depending on t he wait state pattern on the bus being traced, the regular ASYNC packet during the trace can also be delayed and can cause a problem to trace decompression if ASYNC is not output for a long time.

Workaround(s) None

CTAP#21 *CTAP TMS_PSEL Not Getting Reset Values Asynchronously*

Revision(s) Affected Revision B and earlier

Details The CTAP TMS_PSEL is not getting reset values asynchronously . This is not an issue in operating mode and that it is reset on first TCK clock when debugging.

Workaround(s) None

DCAN#18 *Write To A Mailbox Address Beyond Implemented Number Corrupts Implemented Mailbox*

Revision(s) Affected Revision B and earlier

Details A write to a mailbox address beyond the implemented number of mailboxes might corrupt an implemented mailbox.

Workaround(s) Do not write to mailboxes beyond the number described in the device datasheet.

DCAN#21 *In Interrupt Multiplexer Registers, Bit To Message Objects Mapping Is Left Rotated By 1*

Revision(s) Affected Revision B and earlier

Details In the Interrupt Multiplexer Registers, the Bit To Message Objects Mapping Is Left Rotated By 1

The bit mapping for Interrupt Multiplexer register is Implemented as below:

- Bit 0 -> last MsgObj
- Bit 1 -> MsgObj Number 1
- Bit 2 -> MsgObj Number 2
- and so on

No other DCAN registers are affected.

Workaround(s) None

DCAN#22 *Incorrect Payload Stored In Mailbox*

Revision(s) Affected Revision B and earlier

Details The incorrect payload may be stored in the mailbox the DCAN INIT bit is set during ongoing traffic on the bus. If there is ongoing traffic on the bus and the DCAN INIT bit is set, the DCAN state machine may enter a state where for the next and all subsequent matching IDs, incorrect data is stored in the messageRAM. 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, 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.

DCAN#23 *WRITE Accesses From IFx Registers To Message RAM Lost*

Revision(s) Affected Revision B and earlier

Details

Write accesses of message object configuration and control bits from the IFx registers to the message RAM may be lost if they are performed while MsgVal = 1 for the message object which is accessed and CAN communication is ongoing. This occurs if the IFx register write to the message RAM occurs in between a read-modify-write access of the Host Message Handler when it is in the process of receiving a message for the same message object.

Reconfiguration of message objects for the reception of frames:

It is necessary to reset MsgVal to not valid before changing any of the following configuration and control bits:

- Id28-0
- Xtd
- Dir
- DLC3-0
- RxIE
- TxIE
- RmtEn
- EoB
- Umask
- Msk28-0
- MXtd
- MDir

These parts of a message object may be changed without clearing MsgVal:

- Data7-0
- TxRqst
- NewDat
- MsgLst
- IntPnd

Reconfiguration of message objects for the transmission of data frames:

It is necessary to reset MsgVal to not valid before changing any of the following configuration and control bits:

- Dir
- RxIE
- TxIE
- RmtEn
- EoB
- Umask
- Msk28-0
- MXtd
- MDir

These parts of a message object may be changed without clearing MsgVal:

- Id28-0
- Xtd
- DLC3-0
- Data7-0

- TxRqst
- NewDat
- MsgLst
- IntPnd

Workaround(s) Reset the MSGVAL before re-configuring the message object, this avoids the corner condition in which this problem may happen. It is mandatory to reset the MSGVAL before modifying certain fields, as listed under the erratum description.

ESRAMW#22 *DERR Address Register Cleared While Being Read*

Revision(s) Affected Revision B and earlier

Details The DERR address register may be cleared when this register is being read during VBUSP emulation debug mode. This failure will be seen only when a single read to DERR address register is done, or when DERR address register is the last register being read.

Workaround(s) None

ESRAMW#32 *AHB Transaction Initiated During Auto Initialization Mode Fails*

Revision(s) Affected Revision B and earlier

Details When AHB transaction is triggered during auto initialization, the wrapper asserts wait state to master, but fails to drive the address to the RAM on the completion of auto initialization mode and hops to the next AHB address. The master is being held. The issue occurs when auto init is completed, the transaction that started during the auto init (the one that's being held) receives the wrong data (because the wrong address is sent to RAM after the auto init).

Workaround(s) None

FWM#87 *IFLUSH Mode Hangs Or Returns Incorrect Data*

Revision(s) Affected Revision B and earlier

Details In some cases IFLUSH mode either hangs or returns the wrong data This can occur in two cases and both occur during an IFLUSH, DIS_PREEMPT=1 operation and after a single bit error. DIS_PREMPT cannot be used with IFLUSH mode.

Workaround(s) Always use with DIS_PREEMPT=0 . Do not disable DIS_PREEMPT in order to use IFLUSH.

FWM#90 *New Algorithm Required To Meet Flash Bank's Read Margin Test Requirements*

Revision(s) Affected

Revision B and earlier

Details

There are two issues in the flash bank test.

- There is a requirement for the flash bank to see the TEZ and TCR change and then receive two Flash clocks and then wait 1 us before a valid read time can begin.
- The Bank cannot power up or wake up in read margin mode.

Currently the FWM does not address these two issues. It can change TEZ/TCR on every access to the BANK. A new algorithm is needed to ensure the above conditions are met.

Workaround(s)

To enter Read Margin Mode:

1. Set the banks to remain powered up. Write 0505FFFFh to register 40h FBFALLBACK.
2. Do one dummy read from each bank to turn on the banks.
3. Start execution out of RAM.
4. Turn on Read Margin 0 or 1. Write 1 or 2 to register 04h FSPRD.
5. Flush the data buffer by reading two Ram locations separated by at least 32 bytes.
6. Do two dummy reads from each bank.
7. Wait 1 us Any reads will now be with the selected read margins.

You can now return to Flash if desired

To change Read margin mode to the other :

1. Start execution out of RAM.
2. Switch Read Margin mode to the other. Write 2 or 1 to register 04h FSPRD.
3. Flush the data buffer by reading two Ram locations separated by at least 32 bytes.
4. Do two dummy reads from each bank.
5. Wait 1 us Any reads will now be with the selected read margins.

You can now return to Flash if desired

To leave Read Margin Mode:

1. Start execution out of RAM.
2. Turn off Read Margin 0 and 1. Write 0 to register 04h FSPRD.
3. Flush the data buffer by reading two Ram locations separated by at least 32 bytes.
4. Do two dummy reads from each bank.
5. Wait 1 us Any reads will now be a standard read.

You can now return to Flash if desired. You can also set the FBFALLBACK register to your original values.

FWM#115 *Flash Clock Is One Cycle Late*

Revision(s) Affected Revision B and earlier

Details The Flash Clock is coming one cycle late for some Flash Reads if the test case enters the software interface (SW_INTF) mode in pipeline mode. The Flash API does not use SW_INTF, it is a test-only issue.

Workaround(s) Enter non-pipeline mode before changing to SW_INTF mode.

FWM#123 *Bad Data Read At End Of FSM Operation*

Revision(s) Affected Revision B and earlier

Details After the FSM finishes there is a window where the CPU can read bad data.

Workaround(s) Do not read until the FSM_BUSY is off.

FWM#135 *Reads To ECC Space Fail*

Revision(s) Affected Revision B and earlier

Details Reads to the ECC space are failing. This condition only occurs when DIS_PREEMPT bit in FSPRD register is set. This bit is not set by default.

Workaround(s) Do not set the DIS_PREEMPT bit.

FWM#138 *Data Read Corrupted*

Revision(s) Affected Revision B and earlier

Details If program code is doing program/erase on one bank, and then a speculative access occurs on that same bank during a critical 5 cycle window, it is possible that the data read during the speculative read would be corrupted, resulting in an ECC error being generated on data that is actually correct. This will not affect the program/erase occurring on the bank.

Workaround(s) Do not program/erase with ECC enabled.

FWM#141 *Single Bit Error Address Register Updated With Incorrect Address*

Revision(s) Affected Revision B and earlier

Details When IFLUSH is enabled, the single bit error address register(0x14) gets updated with the incorrect address.

Workaround(s) None

FWM#142 *Uncorrectable Error Address Register Updated With Incorrect Address*

Revision(s) Affected Revision B and earlier

Details When IFLUSH is enabled for uncorrectable error (FWM_UERR), the Uncorrectable Error Address register (0x20) is updated with the incorrect address.

Workaround(s) None

FWM#143 *FWM Issuing Spurious Single Bit Error Interrupt (CE_INT)*

Revision(s) Affected Revision B and earlier

Details When FWM is configured with IFLUSH (IDL_EN) enabled, FWM is issuing spurious single bit error interrupt (CE_INT).

Workaround(s) None

FWM#144 *Wrong Read Data Given To FWM*

Revision(s) Affected Revision B and earlier

Details When IFlush is enabled, the wrong read data is given to the FWM when the FRDCNTRL Register contents are changed dynamically.

Workaround(s) None

FWM#145 *Single Bit Or Double Bit Error Not Generated*

Revision(s) Affected Revision B and earlier

Details When IFlush is enabled, a single bit or double bit error is not generated when the IFLUSH cycle occurs in the same cycle as a pipeline hit.

Conditions:

- Zero->one correctable errors when enabled.
- One->zero correctable errors when enabled.
- Multi-bit errors
- Detect_only mode errors; single and multi-bit.
- FCOR_ERR_CNT count values.

Workaround(s) There is no workaround if IDL_EN is set. This situation will not occur if IDL_EN is cleared.

FWM#148 *When Iflush Enabled Error Count Increments Incorrectly*

Revision(s) Affected Revision B and earlier

Details Reading a pipeline hit with a single bit error and an IFLUSH access with an error in the same cycle increments the error counter once instead of the correct twice.

Workaround(s) None

FWM#149 *No Magnitude Comparator For FCOR_ERR_CNT Register*

Revision(s) Affected Revision B and earlier

Details When IFLUSH is enabled, the current implementation does not have a magnitude comparator but instead uses a simpler equal compare. The counter resets to zero when it is equal to the threshold value. This only affects the time when the threshold value changes to a value lower than the current FCOR_ERR_CNT value.

Workaround(s) Clear the FCOR_ERR_CNT register before changing the SEC_THRESHOLD field in the FEDACCTRL2 register.

FWM#151 *Clearing An Error Status Bit Blocks New Errors*

Revision(s) Affected Revision B and earlier

Details if a new error occurs in the same cycle where the CPU is clearing another error in the FEDACSTATUS register, the new error is lost. This occurs most often when the CPU is clearing extra bits like the MULIT_ERR bit when it only needs to clear the single bit error status. Here the multi-bit error would be lost but the address register would set. As a result, he 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.

HET#19 *Incorrect Software Interrupt Generated*

Revision(s) Affected Revision B and earlier

Details All the APCNT instructions with IRQ = ON generate software interrupt for edge selection without checking the condition.

Workaround(s) This instruction is only applicable to use on HET channel 24.

HET#20 *Captured Period/Pulse Count Is Wrong*

Revision(s) Affected Revision B and earlier

Details Captured period/pulse count is wrong after sequence of LR instructions followed by a PCNT instruction. Control Field CF (11:7) is used by HR instructions like ECMP for pinsel functionality. This occurs when:

- When LR instructions like MOV32, which has CF (11:7) reserved and hence the default value is "00000", is executed in a loop before a PCNT instruction which uses pin 0 for capture and if there is a reset edge occurring in the time duration between these instructions and the PCNT instruction then this reset edge will not be registered and the value in the HR counter will be wrong.
- When an instruction like ADCNST, CF (11:7)is not reserved but used for some other functionality other than pinsel. Here if any value between 0 and 31 is programmed in the CF (11:7) field for eg: 6,and the PCNT instruction following this happens to use the corresponding pin number value for capturing (pin 6 in this example), then the reset edge on this pin will not get detected in the duration during this instruction execution and the PCNT instruction and HR counter value would be incorrect.

Workaround(s) Ensure that PCNT executes before any other instruction that has its control field bits 12 - 8 matching the channel used by PCNT. Once PCNT executes on a channel, the PCNT_S flag for that channel is blocked from being reset by another instruction within the same HET loop .

LIN#46 *Super-Fractional Divider Table Mismatch For Synch Field And ID Field Transmission (Master Mode)*

Revision(s) Affected Revision B and earlier

Details In the case of a super-fractional divider, the bits transmitted (for synch and ID fields) with and extra VCLK period are different from the bits specified in the super-fractional-divider table, of BLIN specification.

Workaround(s) None

LIN#48 *Header Not Received By Slave*

Revision(s) Affected Revision B and earlier

Details 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 occurs when the incomplete header is a Sync Break only.

Workaround(s) None

LIN#49 *First Byte Of Data Is Corrupted*

Revision(s) Affected Revision B and earlier

Details When LIN node is in SCI multi-buffered mode and if the buffer length is configured as less than 8, the first set of data (i.e. programmed number 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 *Timeout Flag Not Set*

Revision(s) Affected Revision B and earlier

Details The Timeout flag not set when an incomplete header is followed by bus dominant for more than 4 seconds in slave mode. When the LIN is in slave mode, if a Sync Break and Sync Field are received, and then the bus is recessive for a short time, followed by a bus dominant condition for more than 4 seconds, the timeout flag is not set to indicate a bus timeout. However, the Parity Error flag is set.

Workaround(s) None

LIN#51 *Slave Not Transmitting Response*

Revision(s) Affected Revision B and earlier

Details When a slave receives a wakeup requests with length > 500us, it is not transmitting a response to the first header following the wakeup. The LIN is put into powerdown mode, and then some time later (in 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. What is observed is that 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#57 *LIN Rejects Data*

Revision(s) Affected Revision B and earlier

Details The LIN rejects the first byte followed by a data byte with a false start bit.

Workaround(s) These 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, nextdata 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 re-synchronizes 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 re-synchronizes to the incoming header.

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

Revision(s) Affected Revision B and earlier

Details Details

Workaround(s) This workaround is for Master mode. In Master mode, before triggering the next header:

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

LIN#59 *LIN transmission Impacted*

Revision(s) Affected Revision B and earlier

Details If the LIN is in non-multibuffer mode and a Bit Error has occurred during the transmission of checksum byte, then after the reception of the next header, there will be an idle period of 10 TBIT times after the first data byte transmission. This impacts 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 B 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, then the next data byte transmission will be corrupted (i.e. start bit of next data byte will be corrupted). Writing 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_TXpulse and when the 3rd NEXT_BIT_TX pulse is received, the transmit logic switches to data byte (8bits) transmission. This results in start bit getting transmitted for just 1 VCLK.

Implications: Non-Critical This issue is applicable only when Extended Frame mode is enabled and Single buffer mode transmission is selected. This problem will occur on both Master and Slave nodes and all LIN versions.

Workaround(s) This applies to 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 Waitstate*

Revision(s) Affected Revision B 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) This erratum will not impact any of the LIN communications. it is safe to ignore TOA3WUP interrupt. This applies to moth Master and Slave modes.

LIN#62 *FE During Checksum Byte Restarts Reception*

Revision(s) Affected Revision B 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) The following can occur in this erratum:

- Frame error occurred during checksum byte reception followed by an NRE. Suggested sequence for the software:
  - 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 applies to both Master and Slave modes.

LIN#63 *LIN Timeout Counters Need To Stop Incrementing During Power Down Mode*

Revision(s) Affected Revision B and earlier

Details During powerdown mode whenever there is a register access, clock to the module will be active until the access is complete. Because the clock to the LIN registers and the logic is same, clock to the timeout counters also will be active. This results in an increment of the timeout counters. As a result there will be a timeout interrupt after a certain time (based on the frequency of the register accesses).

Implications: 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 applies to both Master and Slave modes and all LIN versions.

Workaround(s) The handle this erratum do the following:

- Don't poll the registers continuously if the module is in powerdown mode.
- Software 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 the TIMEOUT interrupt in this scenario.

LIN#65 *NRE/Timeout Not Flagged*

Revision(s) Affected Revision B and earlier

Details The NRE/Timeout might not get flagged at Tframe_max and 4s respectively, when LIN slave adapts itself to a faster incoming header. Both NRE and Timeout will get flagged after a long delay in this case. This applies to Slave mode and Adaptive mode only.

Workaround(s) Program the BRSR register to a faster baud rate than the expected bus baud rate value.

MIBSPI#109 *BITERR Limitation*

Revision(s) Affected Revision B and earlier

Details BITERR has a limitation in Slave mode that it may work reliably only at a ratio of SPICLK=VCLK/4 or slower. No other features will get 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.

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

Revision(s) Affected Revision B and earlier

Details The multibuffer slave in 3 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) Set the CSHOLD bit of Control field in TXRAM (even though it's not applicable to 3pin or 4pin nENA configurations).

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

Revision(s) Affected Revision B and earlier

Details After a DLEN error is generated in Slave mode of the SPI using nSCS pins in IO LoopBack Test mode and an interrupt is serviced, the SPI does not abort the ongoing transfer. It continues generating a DLEN error. The DLEN ERR is generated only once if the nENA pin of SPI is made as functional in SPIPC2 register. This is an erratum only applies to 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 for once and set again.

MIBSPI#118 *TXPIN_P Register Has No Reset In MIBSPI*

Revision(s) Affected Revision B 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. There is no functional issue.

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.

PCR#9 *Wakeup Interrupt Generated Twice*

Revision(s) Affected Revision B and earlier

Details When a communication peripheral (CAN/DCAN/LIN/SCI etc..,) is put into powerdown mode by writing into the PCR PS_PWRDN register, the module generates a wakeup interrupt once a dominant value is detected on the RX pin. When the wakeup interrupt is serviced, the module is ready for the communication. Currently, the wakeup interrupt is generated twice before the module gets ready for the communication.

Workaround(s) Clear the corresponding bit in PS_PWRDN register on a wakeup interrupt from respective slave. This will ensure that the CLK_STOP_REQ is de-asserted automatically on a wakeup and avoids the double interrupt scenarios. For DCAN this standard sequence must be followed in the ISR:

- After the ISR is entered, read the CAN status register to identify the source for the interrupt.
- If the source is WakeupPnd flag, clear the corresponding bit in PS_PWRDN register in PCR. This will turn on the VCLK to DCAN.
- Clear the CAN INIT bit in CAN control register.
- Exit the ISR.

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

Revision(s) Affected Revision B and earlier

Details During power on, the FBSLIP and/or the RFSLIP bits in the system module (0xFFFFFFEC) maybe inadvertently be 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).

SYS#103 *SYS_Nrst Requires Extra VCLK Cycles To Guarantee Complete Device Reset*

Revision(s) Affected Revision B and earlier

Details SYS_nRST requires extra VCLK cycles to guarantee complete device reset. The nRST pulse is extended 7-8 VCLK cycles to meet this requirement. However, in certain conditions when nRST is very 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) None

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

Revision(s) Affected Revision B 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 and 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] == 2'b00 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:

- The TPIU is stopped (indicated by bit[1] in the Formatter & Flush Status Register at offset 0x300 is set LOW).
- The formatter is not in Continuous Mode (bit [1] of the Formatter and Flush Control Register at offset 0x304 is set LOW.

Then 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 *Unsynchronized Clock Domain Crossing In CSTPIU*

Revision(s) Affected Revision B 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) 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.

TMS470Mx#6	***HET Data-In Register Shows Value Of TDI & TDO Pins***
Revision(s) Affected	Revision B and earlier
Details	The HET Data-in register shows the value of TDI & TDO pins.
Workaround(s)	None

TMS470Mx#8	***GIO Slew Rate Select Register Inaccessible For Read Or Write***
Revision(s) Affected	Revision B and earlier
Details	As a result of the peripheral select definition for the GIO module the GIO Slew Rate Select Register (GIOSRSx) is inaccessible for reads or writes. The memory map should have included 2 quadrants within the GIO peripheral select to include the GIOSRS register at offset 0x134 when only one was included.
Workaround(s)	None

TMS470Mx#10 ***Receive Buffers In Peripherals Will Receive The Data Even When INENA Disabled***

Revision(s) Affected Revision B and earlier

Details The receive buffers in peripherals will receive the data even when INENA is disabled by the peripheral. The input buffer will consume more current when the input is floating.

Workaround(s) None

TMS470Mx#11 *Die-ID Not Read Properly After Power-On Reset*

Revision(s) Affected Revision B and earlier

Details The Die-ID are not read properly after power-on reset.

Workaround(s) None

TMS470Mx#12 *No Abort Generated When Accessing Illegal Location*

Revision(s) Affected Revision B and earlier

Details An abort is not getting generated when accessing an illegal location between RAM and flash.

Workaround(s) None

3 Revision History

This silicon errata revision history highlights the technical changes made from the previous revision of this document to the current revision.

Table 3. Revision History from Initial Errata Document Revision to Revision A

Advisory Changes in Advisory List	Advisory ID
Added advisory(s)	None
Removed advisory(s)	DCAN#19 (1)
Modified advisory(s)	DCAN#22 (2)
Other	None

(1) DCAN#19 was an empty advisory with no content that was inadvertently included in the previous revision of this document.
(2) Added a workaround section to DCAN#22.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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