TMS470M
Microcontroller
(Less than or equal to 512K Flash)
Silicon Revision 0

Silicon Errata

Literature Number: SPNZ151
December 2011

Contents

List of Figures

List of Tables

TMS470M Microcontroller (Less than or equal to 512K Flash) Silicon Errata **Silicon Revision 0**

1 Introduction

This document describes the known exceptions to the functional specifications for the TMS470M series devices with less than or equal to 512K Flash RAM. For more detailed information on this device, see the device-specific data sheet.

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. 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 TMS470M series device markings. The device revision can be determined by the symbols marked on the top of the package.

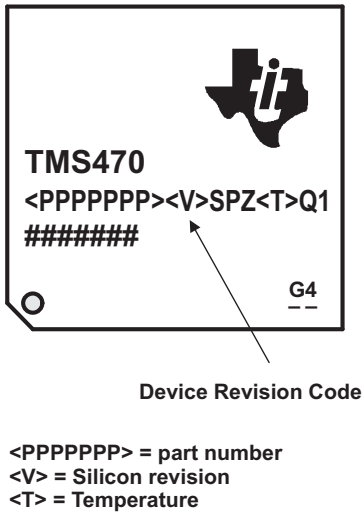

Figure 1. Example, Device Revision Code for TMS470M (PZ Package)

Silicon revision is identified by a device revision code. Table 1 lists the information associated with each silicon revision.

Table 1. TMS470M Device Revision Codes

DEVICE PART NUMBER DEVICE REVISION CODE	SILICON REVISION	PART NUMBERS/COMMENTS
TMS470<PPPPPPP><V>	Initial	**TMS**470<PPPPPPP><V>

2 Silicon Revision 0 Known Design Exceptions to Functional Specifications

This section describes the usage notes and advisories that apply to silicon revision 0 of the TMS470M series devices with less than or equal to 512K Flash RAM.

2.1 Silicon Revision 0 Known Design Exceptions to Functional Specifications

Table 2. Silicon Revision 0 Advisory List

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

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

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*

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#35 *HPROT And MEMATTR Incorrect On Some Unaligned Transactions*

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*

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*

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*

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.

CORTEX-M3#40 *SVC and BusFault/MemManage May Occur Out Of Order*

Details

An SVC is a precise exception and it is generated by executing the SVC instruction. If the fetch for the instruction following an SVC instruction has been faulted, either externally or by the MPU, then the corresponding BusFault or MemManage fault handler should not be entered before the SVC handler as that instruction should not have been attempted to be executed. The SVC handler should be executed first and then if the next instruction is faulted when it is re-fetched it should enter the corresponding fault handler. For Cortex-M3 the BusFault or MemManage handler may be entered before the SVC handler, if the priories allow, otherwise the BusFault or MemManage handler will be tail-chained to. Neither of these events should occur and the instruction following the SVC should be re-fetched once the SVC handler has completed. Only if the instruction is faulted once more should the appropriate handler be entered.

Conditions:

- SVC is executed.
- The following instruction fetch is faulted, either externally or by the MPU.

Implications: The MemManage or BusFault handler may be entered even though the faulted instruction which followed the SVC should not have been executed.

Workaround(s)

A workaround is only required if the SVC handler will not return to the return address that has been stacked for the SVC exception and the instruction access after the SVC will fault. If this is the case then padding can be inserted between the SVC and the faulting area of code, for example, by inserting NOP instructions.

CORTEX-M3#41 *Interrupted Loads to SP Can Cause Erroneous Behavior*

Details

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

- LDR SP,[Rn],#imm
- LDR SP,[Rn,#imm]!
- LDR SP, [Rn,#imm]
- LDR SP,[Rn]
- LDR SP,[Rn,Rm]

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

- LDR SP,[Rn],#imm
- LDR SP,[Rn,#imm]!

Conditions:

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

Implications: Unless the load is being performed to Device or Strongly-Ordered memory, there should be no implications from the repetition of the load. In the unlikely event that the load is being performed to Device or Strongly-Ordered memory, the repeated read can result in the final stack-pointer value being different than had only a single load been performed. Interruption of the two write-back forms of the instruction can result in both the base register value and final stack-pointer value being incorrect. This can result in apparent stack corruption and subsequent unintended modification of memory.

Workaround(s)

Both issues may be worked around by replacing the direct load to the stack-pointer, with an intermediate load to a general-purpose register followed by a move to the stack-pointer. If repeated, reads are acceptable, then the baseupdate issue may be worked around by performing the stack-pointer load without the base increment followed by a subsequent ADD or SUB instruction to perform the appropriate update to the base register.

DCAN#22 *Incorrect Payload Stored In Mailbox*

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) None

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

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

FWM#115 *Flash Clock Is One Cycle Late*

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*

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*

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*

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

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#20 *Captured Period/Pulse Count Is Wrong*

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)*

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*

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*

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*

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*

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*

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*

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*

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*

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*

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*

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*

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*

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*

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*

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*

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*

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*

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*

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).

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

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

SYS#97 *WRITE to ECP_KEY of ECPCNTL1 Register Valid Only In Privileged Mode*

Details A WRITE to ECP_KEY of the ECPCNTL1 register (offset = 0xFFFF E128) is valid only in privileged mode. However, the implementation is different. ECP_KEY is getting updated in USER mode WRITE also.

Workaround(s) None

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

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

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

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

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

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

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