Mask Set Errata for Mask 3N69S

This report applies to mask 3N69S for these products:

- K32L3A60VPJ1A

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
e6939	Core: Interrupted loads to SP can cause erroneous behavior
e9004	Core: ITM can deadlock when global timestamping is enabled
e9005	Core: Store immediate overlapping exception return operation might vector to incorrect interrupt
e6940	Core: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used
e50208	LPADC: Glitch on LPADC reset signal when operating in low-power mode
e10752	LPI2C: Slave Busy Flag may incorrectly read as zero when 10-bit address mode enabled in slave mode.
e10792	LPI2C: Slave Transmit Data Flag may incorrectly read as one when TXCFG is zero.
e11097	LPSPI: Command word not loaded correctly when TXMSK=1
e10655	LPSPI: Module Busy Flag may incorrectly read as zero in Master mode
e10653	MU: Individual core reset cannot be performed above 24MHz
e50096	ROM: Registers are not properly restored after exiting Boot ROM
e11096	SAI: Internal bit clock is not generated when RCR2[BCI]=1 or TCR2[BCI]=1
e11150	SAI: Internally generated receive or transmit BCLK cannot be re-enabled if it is first disabled when RCR2[DIV] or TCR2[DIV] > 0
e50118	USB: USB RAM reads immediately following writes return incorrect data
e10654	XRDC: The MRGD_W4_n_m[LK2] lock field cannot transition from 2'b10 to 2'b11 when written with 2'b01

Table 2. Revision History

Revision	Changes
11 JUL 2019	Initial revision

e6939: Core: Interrupted loads to SP can cause erroneous behavior

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

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

Affects: Cortex-M4, Cortex-M4F

Fault Type: Programmer Category B

Fault Status: Present in: r0p0, r0p1 Open.

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

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

1) LDR SP,[Rn],#imm

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

3) LDR SP,[Rn,#imm]

4) LDR SP,[Rn]

5) LDR SP,[Rn,Rm]

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

1) LDR SP,[Rn],#imm

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

Conditions:

1) An LDR is executed, with SP/R13 as the destination.

2) The address for the LDR is successfully issued to the memory system.

3) An interrupt is taken before the data has been returned and written to the stack-pointer.

Implications:

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

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

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

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

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

e9004: Core: ITM can deadlock when global timestamping is enabled

Description: ARM ERRATA 806422

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

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

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

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

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

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

e9005: Core: Store immediate overlapping exception return operation might vector to incorrect interrupt

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

Affects: Cortex-M4, Cortex-M4F

Fault Type: Programmer Category B Rare

Fault Status: Present in: r0p0, r0p1 Open.

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

Configurations Affected

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

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

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

ARMCC:

...

__schedule_barrier();

__asm{DSB};

__schedule_barrier();

}

GCC:

...

__asm volatile (“dsb 0xf” ::: “memory”);

}

e6940: Core: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used

Description: Arm Errata 709718: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used

Affects: Cortex-M4F

Fault Type: Programmer Category B

Fault Status: Present in: r0p0, r0p1 Open.

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

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

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

There are two workarounds:

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

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

e50208: LPADC: Glitch on LPADC reset signal when operating in low-power mode

Description: When operating in low-power modes, a glitch on the LPADC reset signal can occur after the first command of a scan has completed if no other triggers are asserted. The impact this may have on an application is that a scan with multiple commands may not be completed if the trigger was asserted during low-power mode.

Workaround: This errata has one workaround:

A lower priority trigger may be asserted immediately following the trigger of the scan with multiple commands. The lower priority trigger must be asserted after the high priority trigger but before the first command can be completed.

e10752: LPI2C: Slave Busy Flag may incorrectly read as zero when 10-bit address mode enabled in slave mode.

Description: When operating in slave mode with 10-bit addressing enabled and an address match is detected on the first address byte, the Slave Busy Flag, SSR[SBF], may incorrectly read as zero.

Workaround: When using the LPI2C in slave mode when 10-bit addressing is enabled, and the SSR[SBF] bit is read as a one, then the flag is set. If it is read as a zero, it must be read a second time and this second read will be the correct state of the bit.

e10792: LPI2C: Slave Transmit Data Flag may incorrectly read as one when TXCFG is zero.

Description: When SCFGR1[TXCFG] = 0, the slave transmit data ready flag can incorrectly assert for one cycle.

Workaround: Set SCFGR1[TXCFG] = 1.

e11097: LPSPI: Command word not loaded correctly when TXMSK=1

Description: When the Transmit Command Register is written with TCR[TXMSK]=1 and the next write to the TX FIFO is another command, then the first command may not load correctly.

Workaround: When writing the Transmit Command Register with TCR[TXMSK]=1, wait for the TX FIFO to go empty (FSR[TXCOUNT] = 0) before writing another command to the Transmit Command Register.

e10655: LPSPI: Module Busy Flag may incorrectly read as zero in Master mode

Description: When operating in master mode, the Module Busy Flag, SR[MBF], may incorrectly read as zero. This applies only to master mode and not to slave mode.

Workaround: When using the LPSPI in master mode and the SR[MBF] bit is read as a one then the flag is set. If it is read as a zero, it must be read a second time and this second read will be the correct state of the bit.

e10653: MU: Individual core reset cannot be performed above 24MHz

Description: When core A or core B uses the corresponding CCR[HR] bit to reset core B or core A respectively, unpredictable behavior may result if the core clock is greater than 24MHz.

Workaround: When using the messaging unit CCR[HR] bit to reset one of the cores, the core that is writing this bit must first set the system core clock to 24MHz or less.

e50096: ROM: Registers are not properly restored after exiting Boot ROM

Description: Upon exiting bootloader mode, some peripheral registers are not restored to their reset default state. The peripherals (or registers) affected are:

LPTMR0->CSR

LPTMR0->PSR

SCB->AIRCR

The modification of the SCB->AIRCR is of particular note as this register change can cause unrecoverable hard fault errors when an RTOS initiates the first task of an application.

Workaround: The workaround is to simply check the value of these registers and write them to the reset value in the application startup code. The recommended code is shown below.

if(LPTMR0->CSR != 0)

{ LPTMR0->CSR = 0; }

if(LPTMR0->PSR != 0) { LPTMR0->PSR = 0; }if((SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) != 0x0) { SCB->AIRCR = 0x05FA0000; }

e11096: SAI: Internal bit clock is not generated when RCR2[BCI]=1 or TCR2[BCI]=1

Description: When the SAI transmitter or receiver is configured for internal bit clock with BCI = 1, the bit clock is not generated for either of the following two configurations:

a) SYNC = 00 and BCS = 0

b) SYNC = 01 and BCS = 1

Workaround: When the SAI transmitter or receiver is configured for internal bit clock with BCI=1, use only one of the following two configurations:

a) SYNC = 01 and BCS = 0

b) SYNC = 00 and BCS = 1

e11150: SAI: Internally generated receive or transmit BCLK cannot be re-enabled if it is first disabled when RCR2[DIV] or TCR2[DIV] > 0

Description: If the receive or transmit bit clock (BCLK) is internally generated, enabled with DIV > 0 and is then disabled, due to software or Stop mode entry, and the BCLK is enabled again, the clock is not generated.

Workaround: If the receive or transmit BCLK is internally generated and a DIV value greater than 0 is used, the SAI must be reset before the BCLK is re-enabled. This is achieved by writing the SR bit in the respective RCSR or TCSR register first to 1 and then immediately to 0.

e50118: USB: USB RAM reads immediately following writes return incorrect data

Description: When reading a USB RAM location immediately after a write to that location, incorrect results may be returned. This only occurs when 8-bit accesses to the RAM are performed.

Workaround: The workaround to this errata is to not use the USB RAM.

If the USB RAM must be used, users must ensure one of the following:

1) Do not perform 8-bit accesses to the USB RAM.

2) Insert a no-operation (“NOP”) instruction or other operation between the write to the location and the read to the location. Note that this does not need to be (and is not recommended to be) coded at the assembly level. This can and should be done at higher programming language levels (such as C, or C++).

e10654: XRDC: The MRGD_W4_n_m[LK2] lock field cannot transition from 2'b10 to 2'b11 when written with 2'b01

Description: It should be possible to individually set the bits in the MRGD_W4_n_m[LK2] lock field to 1, and each bit should individually remain asserted after it is set (cannot be written back to 0). In the case where this field has first been written to a value of 2’b10 it cannot be changed to a value of 2’b11 by writing 2’b01. Writing a value of 2’b01 will not change the field value in this case. There are no issues with changing this field from a value of 2’b00 to 2’b11 by means of a single write.

Workaround: If it is required to change the value of MRGD_W4_n_m[LK2] from 2’b10 to 2’b11, the write value must be 2’b11.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support