Mask Set Errata for Mask 2N96T

This report applies to mask 2N96T for these products:

- MK28FN2M0CAU15R
- MK28FN2M0VMI15
- MK27FN2M0VMI15

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
e8992	AWIC: Early NMI wakeup not detected upon entry to stop mode from VLPR mode
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
e9380	FlexIO: Reading FlexIO register when FlexIO functional clock is disabled results in a bus hang
e9265	FTM: Incorrect match may be generated if intermediate load feature is used in toggle mode
e10856	FTM: Safe state is not removed from channel outputs after fault condition ends if SWOCTRL is being used to control the pin
e9308	I2C: I2C does not hold bus between byte transfers in receive and may result in lost data
e10990	I2S/SAI: I2S1 logic tied to I2S0 clock gate
e10779	Kinetis ROM Bootloader: Programming QuadSPI using bootloader over UART peripheral will fail due to incorrect baud rate detection
e10527	LPUART: Setting and immediately clearing SBK bit can result in transmission of two break characters
e10656	LPUART: The RXD Pin Active Edge Interrupt flag does not assert when an active edge is detected
e9878	MCG: Clock transition may have an issue immediately after writing the OSCSEL or RANGE bit fields in MCG control registers
e7735	MCG: IREFST status bit may set before the IREFS multiplexor switches the FLL reference clock
e10806	OCRAM: Does not support successive accesses to the same address
e10780	OCRAM: Un-aligned INCR burst transfers are not supported for eSDHC/USBHS/USBFS modules
e11033	OCRAM: With certain write/read sequences accessed from 0x3400_0000 to 0x3407_FFFF, data corruption will occur
e9462	QuadSPI: DQS Learning/Calibration does not supports concurrent read transactions

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
e9651	QuadSPI: QuadSPI SDR clock limitation when core clock is greater than 100MHz
e9461	QuadSPI: Read data errors may occur with data learning in 4x sampling method
e3981	SDHC: ADMA fails when data length in the last descriptor is less or equal to 4 bytes
e3982	SDHC: ADMA transfer error when the block size is not a multiple of four
e4624	SDHC: AutoCMD12 and R1b polling problem
e3977	SDHC: Does not support Infinite Block Transfer Mode
e4627	SDHC: Erroneous CMD CRC error and CMD Index error may occur on sending new CMD during data transfer
e3984	SDHC: eSDHC misses SDIO interrupt when CINT is disabled
e3983	SDHC: Problem when ADMA2 last descriptor is LINK or NOP
e3978	SDHC: Software can not clear DMA interrupt status bit after read operation
e8807	USB: In Host mode, transmission errors may occur when communicating with a Low Speed (LS) device through a USB hub
e9646	WDOG:Unexpected watchdog behavior on LLS exit

Table 2. Revision History

Revision	Changes
21 FEB 2017	Initial revision
04 AUG 2017	The following errata were added. • e10527 • e11033 • e10856 • e10990 • e10656

e8992: AWIC: Early NMI wakeup not detected upon entry to stop mode from VLPR mode

Description: Upon entry into VLPS from VLPR, if NMI is asserted before the VLPS entry completes, then the NMI does not generate a wakeup to the MCU. However, the NMI interrupt will occur after the MCU wakes up by another wake-up event.

Workaround: There are two workarounds:

1) First transition from VLPR mode to RUN mode, and then enter into VLPS mode from RUN mode.

2) Assert NMI signal for longer than 16 bus clock cycles.

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.

e9380: FlexIO: Reading FlexIO register when FlexIO functional clock is disabled results in a bus hang

Description: Accessing a FlexIO register when the FlexIO functional clock is disabled (the clock source configured to 0 in PCC_FLEXIO0[PCS], or the selected clock source is disabled) will hang the bus and the access will stall forever.

Workaround: Always enable the FlexIO functional clock before accessing any FlexIO register.

e9265: FTM: Incorrect match may be generated if intermediate load feature is used in toggle mode

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

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

e10856: FTM: Safe state is not removed from channel outputs after fault condition ends if SWOCTRL is being used to control the pin

Description: If an FTM channel output is being controlled using the software output control register (FTM_SWOCTRL) and fault detection is also enabled for the channel, then when a fault is detected the output is forced to its safe value. However, when the fault condition has been cleared, the channel output will stay in the safe state instead of reverting to the value programmed by the FTM_SWOCTRL register.

Workaround: If fault control is enabled while the software output control register is also being used (FTM_SWOCTRL), then the FTM should be configured as follows:

-- FTM_MODE[FAULTM] configured for manual fault clearing (0b10)

-- For devices that include the FTM_CONF[NUMTOF] field, it must be cleared to 0b00000 (TOF set for each counter overflow). For FTM versions that don't include the FTM_CONF[NUMTOF] field this doesn't apply.

The procedure below must be used in the TOF interrupt handler when a fault is detected to ensure that the outputs return to the value configured by FTM_SWOCTRL.

1. Check the value of FTM_FMS[FAULTF].

-- If FTM_FMS[FAULTF] = 1 (fault occurred or is occurring), then set a variable to indicate that a fault was detected and continue to step 2.

-- If FTM_FMS[FAULTF] = 0 but the fault variable is set (fault is not active, but was previously detected), continue to step 6.

2. Write the FTM_OUTMASK register to set the bit or bits corresponding to any channels that are controlled by FTM_SWOCTRL to temporarily inactivate the channel output.

3. Clear fault conditions by reading the FTM_FMS register and then writing FTM_FMS with all zeroes.

4. Clear the FTM_SC[TOF] bit by reading the FTM_SC register, then writing a 0 to FTM_SC[TOF].

5. Exit the interrupt handler to skip following steps (they will execute the next time the TOF handler is called).

6. Clear the FTM_SWOCTRL by writing all zeroes to it.

7. Write FTM_SWOCTRL with the desired value again.

8. Clear the FTM_OUTMASK bits that were set in step 2.

9. Clear the fault variable that was set in step 1 when the fault condition was originally detected.

10. Clear the FTM_SC[TOF] bit by reading the FTM_SC register, then writing a 0 to FTM_SC[TOF].

e9308: I2C: I2C does not hold bus between byte transfers in receive and may result in lost data

Description: When the I2C module is in receive mode, the bus is typically held by simply not reading the I2C_D register. However, in devices with this errata, the bus is not held between byte transfers by this action. If the I2C_D register and the data buffer are full, incoming data from an I2C device will overwrite data in the data buffer.

Workaround: When configured to receive data, the delay in processing incoming bytes should be minimized. Delay can be minimized by the use of DMA or increased interrupt priority.

e10990: I2S/SAI: I2S1 logic tied to I2S0 clock gate

Description: After un-gating the I2S1 module (SIM_SCGC3[I2S1]) for the first time after a Power-On-Reset, the I2S1_MCR[MOE] bit is stuck at a value of 1 due to the incorrect state of the MCLK fractional clock divider. This not only affects the MOE bit, but also affects:

- I2S1_MCR[DUF], this bit is a read only.

- I2S1_MCR[MICS] and I2S1_MDR[FRACT,DIVIDE] bits.

- Entry into low-power Stop modes

Workaround: To set the MCLK fractional clock divider to the correct state, set, and then clear the SIM_SCGC6[I2S0] bit prior to enabling the SIM_SCGC3[I2S1] bit. Toggling the SIM_SCGC6[I2S0] bit will allow the I2S1 module to function correctly.

e10779: Kinetis ROM Bootloader: Programming QuadSPI using bootloader over UART peripheral will fail due to incorrect baud rate detection

Description: When programming the QuadSPI using the bootloader with the UART interface after the QuadSPI Configuration Block (QCB) is valid and the FTFE_FOPT[7:6] is set to 0b10, the programming will not be successful due to incorrect baud rate detection.

The UART interface is not affected when the QuadSPI or any other peripheral interface is not programmed.

Workaround: If using the 'blhost' application to establish communication with the ROM Bootloader, the application will automatically resolve the baud rate detection issue.

Otherwise, to successfully program the QuadSPI using the bootloader and the UART interface, the baud rate must be twice the desired baud rate and will remain at twice the desired baud rate until a subsequent Power On Reset.

e10527: LPUART: Setting and immediately clearing SBK bit can result in transmission of two break characters

Description: When the LPUART transmitter is idle (LPUART_STAT[TC]=1), two break characters may be sent when using LPUART_CTRL[SBK] to send one break character. Even when LUART_CTRL[SBK] is set to 1 and cleared (set to 0) immediately.

Workaround: To queue a single break character via the transmit FIFO, set LPUART_DATA[FRETSC]=1 with data bits LPUART_DATA[T9:T0]=0.

e10656: LPUART: The RXD Pin Active Edge Interrupt flag does not assert when an active edge is detected

Description: The RXD Pin Active Edge Interrupt (LPUART_STAT[RXEDGIF]) flag in the LPUART Status Register should set when an active edge is detected and the Receiver Enable (LPUART_CTRL[RE]) bit in the LPUART Control Register is set. However, the LPUART_STAT[RXEDGIF] flag will only set if the RX Input Active Edge Interrupt Enable (LPUART_BAUD[RXEDGIE]) bit in the LPUART Baud Rate Register is also set. Note that LPUART_BAUD[RXEDGIE] enables the interrupt for the LPUART_STAT[RXEDGIF] flag.

Workaround: Set LPUART_BAUD[RXEDGIE] bit if using the LPUART_STAT[RXEDGIF] flag.

e9878: MCG: Clock transition may have an issue immediately after writing the OSCSEL or RANGE bit fields in MCG control registers

Description: The clock transition may fail when transitioning to clock mode FEI or FEE and writing to the MCG registers immediately after writing the OSCSEL bit field in MCG_C7 or the RANGE bit field in MCG_C2 register.

Also in some cases, the user's code may fail to transition to FEE clock mode. The problem can occur when all of the following conditions exist:

1) FOPT[BOOTSRC_SEL] is configured to 0'b11, enabling MCU boot from ROM and

2) The 'enabled Peripherals' BCA field (offset address 0x10) enables the USB peripheral.

Workaround: Add a 50us delay after writing the MCG_C7 or MCG_C2 registers.

If ROM bootloader is used, choose one of the following three options for boot setting:

1) Configure FOPT[BOOTSRC_SEL] to 0'b00 or 0'b10;

2) Configure the ‘enabledPeripherals’ BCA field (offset address 0x10) to disable USB if using the FOPT[BOOTSRC_SEL] = 0’b11 option, i.e. if boot from ROM, then ROM code must disable USB;

3) Ensure that the user code does not transition to clock mode to FEE mode if FOPT[BOOTSRC_SEL] = 0’b11 and the ‘enabled Peripherals’ BCA field enables USB, i.e. when booting from ROM and also allow ROM code to enable the USB, afterward, the user code must not transition clock mode to FEE mode.

e7735: MCG: IREFST status bit may set before the IREFS multiplexor switches the FLL reference clock

Description: When transitioning from MCG clock modes FBE or FEE to either FBI or FEI, the MCG_S[IREFST] bit will set to 1 before the IREFS clock multiplexor has actually selected the slow IRC as the reference clock. The delay before the multiplexor actually switches is:

2 cycles of the slow IRC + 2 cycles of OSCERCLK

In the majority of cases this has no effect on the operation of the device.

Workaround: In the majority of applications no workaround is required. If there is a requirement to know when the IREFS clock multiplexor has actually switched, and OSCERCLK is no longer being used by the FLL, then wait the equivalent time of:

2 cycles of the slow IRC + 2 cycles of OSCERCLK

after MCG_S[IREFST] has been set to 1.

e10806: OCRAM: Does not support successive accesses to the same address

Description: The AHB interface to the OCRAM does not support successive accesses to the same address. There is a race-condition in the arbitration mechanism that cannot be overcome with user configuration. The two bus-masters, ARM Core and eDMA, can access the OCRAM at any time as long as they do not attempt to access the same address at the same time, or very quickly in succession.

The ARM Core and eDMA can access the OCRAM at anytime as long they do not access the same address--that is, the addresses accessed by the Core are mutually exclusive to the eDMA address access.

Workaround: If both the ARM Core and eDMA are required to access the same address in the OCRAM array, this can only be supported if these accesses are mutually exclusive. For example, by implementing a semaphore in the TCRAM space, application software can ensure that the same address within the OCRAM is not accessed simultaneously by the eDMA and the ARM Core.

e10780: OCRAM: Un-aligned INCR burst transfers are not supported for eSDHC/ USBHS/USBFS modules

Description: When the USBHS/USBFS/eSHDC controller reads buffer data from the OCRAM via module’s built-in DMA and the start address ends in 0x10, 0x30, 0x50 and so on, the transferred data bytes 16-20 are always read incorrectly. Therefore, AHB INCR burst transfers (a burst of one or more transfers with addresses consecutive to the first transfer) are not supported.

Workaround: Memory for the eSDHC/USBHS/USBFS modules must be allocated in the Tightly Coupled Memory (TCM) SRAM to avoid this issue. Allocating memory for the Core in the upper TCM SRAM and the eSDHC/USBHS/USBFS modules in the lower TCM SRAM will provide maximum MIPS performance for the Core.

e11033: OCRAM: With certain write/read sequences accessed from 0x3400_0000 to 0x3407_FFFF, data corruption will occur

Description: When a read immediately follows a write access, the data returned will be corrupted.

Workaround: Limit the access to the OCRAM to a single master such as the core and insert a Data Synchronization Barrier instruction between consecutive store and load instructions:

.

.

.

STR

DSB

LDR

.

.

.

This issue only affects the OCRAM accessed from 0x3400_0000 to 0x3407_FFFF and does not affect the TCRAM accessed from 0x1FFC_0000 to 0x2003_FFFF.

Fix plan: 3N96T mask set addresses the data corruption issue.

e9462: QuadSPI: DQS Learning/Calibration does not supports concurrent read transactions

Description: Learning/calibration in DQS sampling method is semi-automated. Coarse and fine delay values (configured using QuadSPI_MCR[SCLKCFG] and QuadSPI_SOCCR respectively) are changed to test whether the learning patterns are passing or failing.

During this time if concurrent read transactions from DMA or others master occurs, it might result in incorrect read data from flash.

Workaround: It must be ensured that while this calibration is ongoing no other accesses to QuadSPI must be done

e9651: QuadSPI: QuadSPI SDR clock limitation when core clock is greater than 100MHz

Description: The MCGPLL 2x clock cannot be used as the QuadSPI source clock (selected by QuadSPIx_SOCCR[QSPISRC]) when the MCGPLL clock is over 100MHz and QuadSPIx_SOCCR[SCLKCFG] is non-zero.

This means that when running the core clock at above 100MHz, the QuadSPI SDR clock cannot be 100MHz because it requires the use of the MCGPLL 2x clock to derive that QuadSPI clock speed.

Workaround: To run the QuadSPI SDR clock at 100MHz, the MCGPLL/core clock will also need to run at 100MHz.

If the core clock is run above 100MHz, the SDR clock can be generated by dividing the MCGPLL clock by 2, meaning that a 75MHz maximum SDR clock speed is possible with the core clock at 150MHz.

e9461: QuadSPI: Read data errors may occur with data learning in 4x sampling method

Description: Data learning using 4x Sampling method may select a sampling point which is marginal. A marginal sampling point occurs when the sampling point is located on the edge of the valid sampling window. A marginal sampling point may return a positive comparison of the data learning pattern but small variations in voltage and temperature during the same read transaction may result in data errors, since the sampling point is not properly located inside the valid sampling window.

Workaround: There are two options:

- Internal DQS method allows to perform data learning as described on the Reference Manual.
- If 4x Sampling method is used, data learning should not be used and a fixed sampling point must be selected.

e3981: SDHC: ADMA fails when data length in the last descriptor is less or equal to 4 bytes

Description: A possible data corruption or incorrect bus transactions on the internal AHB bus, causing possible system corruption or a stall, can occur under the combination of the following conditions:

1. ADMA2 or ADMA1 type descriptor

2. TRANS descriptor with END flag

3. Data length is less than or equal to 4 bytes (the length field of the corresponding descriptor is set to 1, 2, 3, or 4) and the ADMA transfers one 32-bit word on the bus

4. Block Count Enable mode

Workaround: The software should avoid setting ADMA type last descriptor (TRANS descriptor with END flag) to data length less than or equal to 4 bytes. In ADMA1 mode, if needed, a last NOP descriptor can be appended to the descriptors list. In ADMA2 mode this workaround is not feasible due to ERR003983.

e3982: SDHC: ADMA transfer error when the block size is not a multiple of four

Description: Issue in eSDHC ADMA mode operation. The eSDHC read transfer is not completed when block size is not a multiple of 4 in transfer mode ADMA1 or ADMA2. The eSDHC DMA controller is stuck waiting for the IRQSTAT[TC] bit in the interrupt status register.

The following examples trigger this issue:

1. Working with an SD card while setting ADMA1 mode in the eSDHC

2. Performing partial block read

3. Writing one block of length 0x200

4. Reading two blocks of length 0x22 each. Reading from the address where the write operation is performed. Start address is 0x512 aligned. Watermark is set as one word during read. This read is performed using only one ADMA1 descriptor in which the total size of the transfer is programmed as 0x44 (2 blocks of 0x22).

Workaround: When the ADMA1 or ADMA2 mode is used and the block size is not a multiple of 4, the block size should be rounded to the next multiple of 4 bytes via software. In case of write, the software should add the corresponding number of bytes at each block end, before the write is initialized. In case of read, the software should remove the dummy bytes after the read is completed.

For example, if the original block length is 22 bytes, and there are several blocks to transfer, the software should set the block size to 24. The following data is written/stored in the external memory:

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

2 Bytes valid data + 2 Byte dummy data

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

2 Bytes valid data + 2 Byte dummy data

In this example, 48 (24 × 2) bytes are transferred instead of 44 bytes. The software should remove the dummy data.

e4624: SDHC: AutoCMD12 and R1b polling problem

Description: Occurs when a pending command which issues busy is completed. For a command with R1b response, the proper software sequence is to poll the DLA for R1b commands to determine busy state completion. The DLA polling is not working properly for the ESDHC module and thus the DLA bit in PRSSTAT register cannot be polled to wait for busy state completion. This is relevant for all eSDHC ports (eSDHC1-4 ports).

Workaround: Poll bit 24 in PRSSTAT register (DLSL[0] bit) to check that wait busy state is over.

e3977: SDHC: Does not support Infinite Block Transfer Mode

Description: The eSDHC does not support infinite data transfers, if the Block Count register is set to one, even when block count enable is not set.

Workaround: The following software workaround can be used instead of the infinite block mode:

1. Set BCEN bit to one and enable block count

2. Set the BLKCNT to the maximum value in Block Attributes Register (BLKATTR) (0xFFFFfor 65535 blocks)

e4627: SDHC: Erroneous CMD CRC error and CMD Index error may occur on sending new CMD during data transfer

Description: When sending new, non data CMD during data transfer between the eSDHC and EMMC card, the module may return an erroneous CMD CRC error and CMD Index error. This occurs when the CMD response has arrived at the moment the FIFO clock is stopped. The following bits after the start bit of the response are wrongly interpreted as index, generating the CRC and Index errors.

The data transfer itself is not impacted.

The rate of occurrence of the issue is very small, as there is a need for the following combination of conditions to occur at the same cycle:

- The FIFO clock is stopped due to FIFO full or FIFO empty
- The CMD response start bit is received

Workaround: The recommendation is to not set FIFO watermark level to a too small value in order to reduce frequency of clock pauses.

The problem is identified by receiving the CMD CRC error and CMD Index error. Once this issue occurs, one can send the same CMD again until operation is successful.

e3984: SDHC: eSDHC misses SDIO interrupt when CINT is disabled

Description: An issue is identified when interfacing the SDIO card. There is a case where an SDIO interrupt from the card is not recognized by the hardware, resulting in a hang.

If the SDIO card lowers the DAT1 line (which indicates an interrupt) when the SDIO interrupt is disabled in the eSDHC registers (that is, CINTEN bits in IRQSTATEN and IRQSIGEN are set to zero), then, after the SDIO interrupt is enabled (by setting the CINTEN bits in IRQSTATEN and IRQSIGEN registers), the eSDHC does not sense that the DAT1 line is low. Therefore, it fails to set the CINT interrupt in IRQSTAT even if DAT1 is low.

Generally, CINTEN bit is disabled in interrupt service.

The SDIO interrupt service steps are as follows:

1. Clear CINTEN bit in IRQSTATEN and IRQSIGEN.

2. Reset the interrupt factors in the SDIO card and write 1 to clear the CINT interrupt in IRQSTAT.

3. Re-enable CINTEN bit in IRQSTATEN and IRQSIGEN.

If a new SDIO interrupt from the card occurs between step 2 and step 3, the eSDHC skips it.

Workaround: The workaround interrupt service steps are as follows:

1. Clear CINTEN bit in IRQSTATEN and IRQSIGEN.

2. Reset the interrupt factors in the SDIO card and write 1 to clear CINT interrupt in IRQSTAT.

3. Clear and then set D3CD bit in the PROCTL register. Clearing D3CD bit sets the reverse signal of DAT1 to low, even if DAT1 is low. After D3CD bit is re-enabled, the eSDHC can catch the posedge of the reversed DAT1 signal, if the DAT1 line is still low.

4. Re-enable CINTEN bit in IRQSTATEN and IRQSIGEN.

e3983: SDHC: Problem when ADMA2 last descriptor is LINK or NOP

Description: ADMA2 mode in the eSDHC is used for transfers to/from the SD card. There are three types of ADMA2 descriptors: TRANS, LINK or NOP. The eSDHC has a problem when the last descriptor (which has the End bit '1') is a LINK descriptor or a NOP descriptor.

In this case, the eSDHC completes the transfers associated with this descriptor set, whereas it does not even start the transfers associated with the new data command. For example, if a WRITE transfer operation is performed on the card using ADMA2, and the last descriptor of the WRITE descriptor set is a LINK descriptor, then the WRITE is successfully finished. Now, if a READ transfer is programmed from the SD card using ADMA2, then this transfer does not go through.

Workaround: Software workaround is to always program TRANS descriptor as the last descriptor.

e3978: SDHC: Software can not clear DMA interrupt status bit after read operation

Description: After DMA read operation, if the SDHC System Clock is automatically gated off, the DINT status can not be cleared by software.

Workaround: Set HCKEN bit before starting DMA read operation, to disable SDHC System Clock auto-gating feature; after the DINT and TC bit received when read operation is done, clear HCKEN bit to re-enable the SDHC System Clock auto-gating feature.

e8807: USB: In Host mode, transmission errors may occur when communicating with a Low Speed (LS) device through a USB hub

Description: In Host mode, if the required 48 MHz USB clock is not derived from the same clock source used by the core, transmission errors may occur when communicating with a Low Speed (LS) device through a USB hub. A typical example that causes this issue is when an external 48 MHz clock is used for the USB module via the USB_CLKIN pin, and a separate external clock on XTAL/EXTAL is used to generate the system/core clock.

This issue does not occur when in USB Device mode or if the LS device is not connected through a USB hub.

Workaround: In Host mode, ensure the 48 MHz USB clock is derived from the same clock source that the system clock uses. The two clocks, while they do not need to be the same frequency, both need to come from the same source so that they are in sync. For example, generate the 48 MHz USB clock by dividing down the PLL clock used by the core/system via the SIM_CLKDIV2[USBFRAC] and SIM_CLKDIV2[USBDIV] bit fields.

e9646: WDOG:Unexpected watchdog behavior on LLS exit

Description: When exiting LLS mode, the watchdog counter can increment in some cases. This can cause the watchdog to timeout earlier than expected in applications where the watchdog is enabled and LLS mode is used.

Workaround: When entering or exiting from LLS mode, refresh watchdog to avoid triggering timeout event.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: KINETIS_K_2N96T
04AUG2017