Mask Set Errata for Mask 0N65N

This report applies to mask 0N65N for these products:

- MK26FN2M0xxx18
- MK65FN2M0xxx18, MK65FX1M0xxx18
- MK66FN2M0xxx18, MK66FX1M0xxx18

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
e6940	Core: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used
e7993	MCG: FLL frequency may be incorrect after changing the FLL reference clock
e7735	MCG: IREFST status bit may set before the IREFS multiplexor switches the FLL reference clock
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
e9861	SMC: LLS3/LLS2 low power mode current draw at cold may be higher than specified on some parts.
e4647	UART: Flow control timing issue can result in loss of characters if FIFO is not enabled
e9712	USB-PHY: USB PHY PLL does not lock when MCGC2[EREFS] = 0 (external oscillator clock)
e8807	USB: In Host mode, transmission errors may occur when communicating with a Low Speed (LS) device through a USB hub
e9816	USBHS: The USBHS module can cause non-deterministic Idd value until enabled
e7919	USBOTG: In certain situations, software updates to the Start of Frame Threshold Register (USBx_SOFTHLD) may lead to an End of Frame error condition
e9359	USBReg: In some conditions when using both VREG_INn inputs, the USB voltage regulator current limit can fall below specification

Table 2. Revision History

Revision	Changes
13Apr2015	Initial Release
17Nov2015	The following errata were added. • e9816 • e9712 • e9861

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.

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.

e7993: MCG: FLL frequency may be incorrect after changing the FLL reference clock

Description: When the FLL reference clock is switched between the internal reference clock and the external reference clock, the FLL may jump momentarily or lock at a higher than configured frequency. The higher FLL frequency can affect any peripheral using the FLL clock as its input clock. If the FLL is being used as the system clock source, FLL Engaged Internal (FEI) or FLL Engaged External (FEE), the maximum system clock frequency may be exceeded and can cause indeterminate behavior.

Only transitions from FLL External reference (FBE, FEE) to FLL Internal reference (FBI, FEI) modes and vice versa are affected. Transitions to and from BLPI, BLPE, or PLL clock modes (if supported) are not affected because they disable the FLL. Transitions between the external reference modes or between the internal reference modes are not affected because the reference clock is not changed.

Workaround: To prevent the occurrence of this jump in frequency either the MCG_C4[DMX32] bit must be inverted or the MCG_C4[DRST_DRS] bits must be modified to a different value immediately before the change in reference clock is made and then restored back to their original value after the MCG_S[IREFST] bit reflects the selected reference clock.

If you want to change the MCG_C4[DMX32] or MCG_C4[DRST_DRS] to new values along with the reference clock, the sequence described above must be performed before setting these values to the new value(s).

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.

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.

e9861: SMC: LLS3/LLS2 low power mode current draw at cold may be higher than specified on some parts.

Description: LLS3/LLS2 low power mode current draw at cold may be higher than specified on some parts.

Workaround: VLPS and VLLSx low power modes are unaffected and can be used instead of LLS3/LLS2 power modes across the full operating temperature range. Otherwise, operate at an ambient temperature at or above 0°C.

e4647: UART: Flow control timing issue can result in loss of characters if FIFO is not enabled

Description: On UARTx modules with FIFO depths greater than 1, when the /RTS flow control signal is used in receiver request-to-send mode, the /RTS signal is negated if the number of characters in the Receive FIFO is equal to or greater than the receive watermark. The /RTS signal will not negate until after the last character (the one that makes the condition for /RTS negation true) is completely received and recognized. This creates a delay between the end of the STOP bit and the negation of the /RTS signal. In some cases this delay can be long enough that a transmitter will start transmission of another character before it has a chance to recognize the negation of the /RTS signal (the /CTS input to the transmitter).

Workaround: Always enable the RxFIFO if you are using flow control for UARTx modules with FIFO depths greater than 1. The receive watermark should be set to seven or less. This will ensure that there is space for at least one more character in the FIFO when /RTS negates. So in this case no data would be lost.

Note that only UARTx modules with FIFO depths greater than 1 are affected. The UARTs that do not have the RxFIFO feature are not affected. Check the Reference Manual for your device to determine the FIFO depths that are implemented on the UARTx modules for your device.

e9712: USB-PHY: USB PHY PLL does not lock when MCGC2[EREFS] = 0 (external oscillator clock)

Description: USB PHY PLL does not lock when configuring MCGC2[EREFS] bit to 0 for using external oscillator clock instead of using a crystal with the internal oscillator. This will impact the use case where EXTAL is connected with a valid input clock for PHY PLL while XTAL is saved for GPIO usage.

Workaround: In order to make USB PHY PLL lock when connecting with external clock on EXTAL, configure MCGC2[EREFS] bit to 1 pretending a crystal is connected.

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.

e9816: USBHS: The USBHS module can cause non-deterministic Idd value until enabled

Description: When the HSUSB module is disabled, it can cause non-deterministic Idd value after POR.

Workaround: Workaround for this issue is to enable the HSUSB module and then disable it. This involves enabling the HSUSB PHY PLL and then disabling the HSUSB PHY PLL.

e7919: USBOTG: In certain situations, software updates to the Start of Frame Threshold Register (USBx_SOFTHLD) may lead to an End of Frame error condition

Description: If software updates the Start of Frame Threshold Register (USBx_SOFTHLD) to a value greater than the previous value while the internal SOF countdown counter value is between the previous and updated SOF_THLD value, a new token packet transaction may be initiated, even though it may not complete before the next SOF. This may lead to an End of Frame error condition (CRC5OEF), causing the USB controller to hang.

Workaround: Fix the SOF_THLD to a constant safe or larger value, which is independent of the packet type/ size.

e9359: USBReg: In some conditions when using both VREG_INn inputs, the USB voltage regulator current limit can fall below specification

Description: The USB voltage regulator has an inrush current limiter function which is enabled by default. The inrush current limiter helps to spread out the initial charging current for the load capacitor over time. If the currently selected VREG_INn input (determined by SIM_USBPHYCTL[USBVREGSEL]) is lower than the unselected VREG_INn input and VDD is powered (VDD >= 1.71 V), the current limiter threshold is lowered. With sufficient loading, the decrease in the input current limit can lead to the USB voltage regulator output voltage

collapsing. The value of the current limit becomes abnormally low relative to the datasheet when there is a large delta between the two VREG_INn inputs and the regulator is programmed to specifically use the smaller of the two VREG_INn inputs.

As per the manual, the current limiter was always intended to be disabled after the regulator voltage is established on the output capacitor.

Workaround: If only one VREG_INn input is used, no workaround is required. The erratum only applies if both VREG_INn inputs are used in a system.

When either of the USB ports has an active connection (even if only one is ever being used), then disable the inrush current limit feature by setting the SIM_USBPHYCTL[USBDISILIM] bit. Disabling the inrush current limit feature is recommended in all cases.

If current must be sourced upon startup above the reduced current limit value, consider shorting both VREG _INn inputs together. The current limit threshold is not reduced in this situation, but the current limit should still be disabled after the regulator output is established (method to determine regulator output is established varies based on use case and system hardware).

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Document Number: KINETIS_K_0N65N
Rev. 17Nov2015