

Mask Set Errata for Mask 1N83J

This report applies to mask 1N83J for these products:

- MK64FN1M0VMD12, MK64FN1M0VLQ12, MK64FN1M0VDC12, MK64FN1M0VLL12, MK64FN1M0CAJ12
- MK64FX512VMD12, MK64FX512VLQ12, MK64FX512VDC12, MK64FX512VLL12
- MK63FN1M0VMD12, MK63FN1M0VLQ12
- MK24FN1M0VLQ12, MK24FN1M0VDC12, MK24FN1M0VLL12, MK24FN1M0CAJ12

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
e6990	CJTAG: possible incorrect TAP state machine advance during Check Packet
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
e6933	eDMA: Possible misbehavior of a preempted channel when using continuous link mode
e9265	FTM: Incorrect match may be generated if intermediate load feature is used in toggle mode
e6749	I2C: The I2C_C1[MST] bit is not automatically cleared when arbitration is lost
e6573	JTAG: JTAG TDO function on the PTA2 disables the pull resistor
e8010	LLWU: CMP flag in LLWU_Fx register cleared by multiple CMP out toggles when exiting LLSx or VLLSx modes.
e7950	LLWU: When exiting from Low Leakage Stop (LLS) mode using the comparator, the comparator ISR is serviced before the LLWU ISR
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
e7914	PIT: After enabling the Periodic Interrupt Timer (PIT) clock gate, an attempt to immediately enable the PIT module may not be successful.
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
e4627	SDHC: Erroneous CMD CRC error and CMD Index error may occur on sending new CMD during data transfer

Table continues on the next page...



Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
e3983	SDHC: Problem when ADMA2 last descriptor is LINK or NOP
e7534	SUBFAMID reads back incorrect sub-family of the Kinetis device
e7028	UART: During ISO-7816 initial character detection the parity, framing, and noise error flags can set
e7027	UART: During ISO-7816 T=0 initial character detection invalid initial characters are stored in the RxFIFO
e6472	UART: ETU compensation needed for ISO-7816 wait time (WT) and block wait time (BWT)
e4647	UART: Flow control timing issue can result in loss of characters if FIFO is not enabled
e7090	UART: In ISO-7816 mode, timer interrupts flags do not clear
e7029	UART: In ISO-7816 T=1 mode, CWT interrupts assert at both character and block boundaries
e7031	UART: In single wire receive mode UART will attempt to transmit if data is written to UART_D
e5704	UART: TC bit in UARTx_S1 register is set before the last character is sent out in ISO7816 T=0 mode
e7091	UART: UART_S1[NF] and UART_S1[PE] can set erroneously while UART_S1[FE] is set
e7092	UART: UART_S1[TC] is not cleared by queuing a preamble or break character
e8807	USB: In Host mode, transmission errors may occur when communicating with a Low Speed (LS) device through a USB hub
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
e8101	USBOTG: USB host signal crossover voltage higher than specification at low temperature
e9646	WDOG: Unexpected watchdog behavior on LLS exit

Table 2. Revision History

Revision	Changes
13 JAN 2014	Initial revision
1 SEP 2017	The following errata were added. <ul style="list-style-type: none"> • e8807 • e8992 • e8010 • e7993 • e9265 • e7735 • e6573 • e7914 • e7950 • e9005 • e9004 • e7919 • e8101 • e9646

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.

e6990: CJTAG: possible incorrect TAP state machine advance during Check Packet

Description: While processing a Check Packet, the IEEE 1149.7 module (CJTAG) internally gates the TCK clock to the CJTAG Test Access Port (TAP) controller in order to hold the TAP controller in the Run-Test-Idle state until the Check Packet completes. A glitch on the internally gated TCK could occur during the transition from the Preamble element to the first Body element of Check Packet processing that would cause the CJTAG TAP controller to change states instead of remaining held in Run-Test-Idle

If the CJTAG TAP controller changes states during the Check Packet due to the clock glitch, the CJTAG will lose synchronization with the external tool, preventing further communication.

Workaround: To prevent the possible loss of JTAG synchronization, when processing a Check Packet, provide a logic 0 value on the TMS pin during the Preamble element to avoid a possible glitch on the internally gated TCK clock.

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.

e6933: eDMA: Possible misbehavior of a preempted channel when using continuous link mode

Description: When using continuous link mode ($\text{DMA_CR[CLM]} = 1$) with a high priority channel linking to itself, if the high priority channel preempts a lower priority channel on the cycle before its last read/write sequence, the counters for the preempted channel (the lower priority channel) are corrupted. When the preempted channel is restored, it runs past its “done” point instead of performing a single read/write sequence and retiring.

The preempting channel (the higher priority channel) will execute as expected.

Workaround: Disable continuous link mode ($\text{DMA_CR[CLM]}=0$) if a high priority channel is using minor loop channel linking to itself and preemption is enabled. The second activation of the preempting channel will experience the normal startup latency (one read/write sequence + startup) instead of the shortened latency (startup only) provided by continuous link mode.

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.

e6749: I2C: The I2C_C1[MST] bit is not automatically cleared when arbitration is lost

Description: When the I2C module is used as a master device and loses bus arbitration, it correctly switches to be a slave device. The I2C_C1[MST] bit is not automatically cleared when this occurs but it does correctly operate as a slave.

Workaround: When the I2C module has been configured as a master device and the I2C_S[ARB] bit is set, indicating arbitration has been lost, the I2C_C1[MST] bit must be cleared by software before the I2C_S[ARB] bit is cleared.

e6573: JTAG: JTAG TDO function on the PTA2 disables the pull resistor

Description: The JTAG TDO function on the PTA2 pin disables the pull resistor, but keeps the input buffer enabled. Because the JTAG will tri-state this pin during JTAG reset (or other conditions), this pin will float with the input buffer enabled. If the pin is unconnected in the circuit, there can be increased power consumption in low power modes for some devices.

Workaround: Disable JTAG TDO functionality when the JTAG interface is not needed and left floating in a circuit. Modify the PORTA_PCR2 mux before entering low power modes. Set the mux to a pin function other than ALT7. If set up as a digital input and left unconnected in the circuit, then a pull-up or pull-down should be enabled. Alternatively, an external pull device or external source can be added to the pin.

Note: Enabling the pull resistor on the JTAG TDO function violates the JTAG specification.

e8010: LLWU: CMP flag in LLWU_Fx register cleared by multiple CMP out toggles when exiting LLSx or VLLSx modes.

Description: The comparator's corresponding wakeup flag in the LLWU_Fx register is cleared prematurely if:

1. The CMP output is toggled more than one time during the LLSx wakeup sequence and the comparator's corresponding flag in the LLWU_Fx register is cleared.

Or

2. The CMP output is toggled more than one time during the VLLSx wakeup sequence, PMC_REGSC[ACKISO] is cleared, and the comparator's corresponding flag in the LLWU_Fx register is cleared.

Workaround: When MCU is waking up from LLS, code can implement a software flag to retain the wakeup source, if required by software.

When MCU is waking up from VLLSx, code can implement a software flag prior to clearing PMC_REGSC[ACKISO] to retain the wakeup source, if required by software.

e7950: LLWU: When exiting from Low Leakage Stop (LLS) mode using the comparator, the comparator ISR is serviced before the LLWU ISR

Description: The comparator's interrupt service routine when exiting from LLS mode is serviced before the LLWU ISR. Clearing the comparator flag in CMPx_SCR clears the corresponding comparator flag in the LLWU_Fx register which may be used to determine wakeup source in the LLWU ISR.

Workaround: Code can implement a software flag in the CMP ISR to retain wakeup source if required by software.

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.

e7914: PIT: After enabling the Periodic Interrupt Timer (PIT) clock gate, an attempt to immediately enable the PIT module may not be successful.

Description: If a write to the PIT module enable bit (PIT_MCR[MDIS]) occurs within two bus clock cycles of enabling the PIT clock gate in the SIM_CG register, the write will be ignored and the PIT will fail to enable.

Workaround: Insert a read of the PIT_MCR register before writing to the PIT_MCR register. This guarantees a minimum delay of two bus clocks to guarantee the write is not ignored.

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
2 Bytes valid data + 2 Byte dummy 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.

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.

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.

e7534: SUBFAMID reads back incorrect sub-family of the Kinetis device

Description: The current Kinetis Sub-Family ID (SUBFAMID) field of SIM_SDID reads back as 0x2 if the device is K24/K64 or reads back as 0x1 if the device is K63. The value should be read as 0x4 for K24/K64 and 0x3 for K63.

Workaround: Software workaround is to read the value of RAMSIZE in System Options Register 1 (SIM_SOPT1[RAMSIZE] = 1011b) and DIEID bits 11-7 in System Device Identification Register (SIM_SDID[DIEID] = 00110b).

e7028: UART: During ISO-7816 initial character detection the parity, framing, and noise error flags can set

Description: When performing initial character detection (UART_C7816[INIT] = 1) in ISO-7816 mode the UART should not set error flags for any receive traffic before a valid initial character is detected, but the UART will still set these error flags if any of the conditions are true.

Workaround: After a valid initial character is detected (UART_IS7816[INITD] sets), check the UART_S1[NF, FE, and PF] flags. If any of them are set, then clear them.

e7027: UART: During ISO-7816 T=0 initial character detection invalid initial characters are stored in the RxFIFO

Description: When performing initial character detection (UART_C7816[INIT] = 1) in ISO-7816 T=0 mode with UART_C7816[ANACK] cleared, the UART samples incoming traffic looking for a valid initial character. Instead of discarding any invalid initial characters that are received, the UART will store them in the receive FIFO.

Workaround: After a valid initial character is detected (UART_IS7816[INITD] sets), flush the RxFIFO to discard any invalid initial characters that might have been received before the valid initial character.

e6472: UART: ETU compensation needed for ISO-7816 wait time (WT) and block wait time (BWT)

Description: When using the default ISO-7816 values for wait time integer (UARTx_WP7816T0[WI]), guard time FD multiplier (UARTx_WF7816[GTFD]), and block wait time integer (UARTx_WP7816T1[BWI]), the calculated values for Wait Time (WT) and Block Wait Time (BWT) as defined in the Reference Manual will be 1 ETU less than the ISO-7816-3 requirement.

Workaround: To comply with ISO-7816 requirements, compensation for the extra 1 ETU is needed. This compensation can be achieved by using a timer, such as the low-power timer (LPTMR), to introduce a 1 ETU delay after the WT or BWT expires.

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.

e7090: UART: In ISO-7816 mode, timer interrupts flags do not clear

Description: In ISO-7816, when any of the timer counter expires, the corresponding interrupt status register bits gets set. The timer register bits cannot be cleared by software without additional steps, because the counter expired signal remains asserted internally. Therefore, these bits can be cleared only after forcing the counters to reload.

Workaround: Follow these steps to clear the UART_IS7816 WT, CWT, or BWT bits:

1. Clear the UART_C7816[ISO_7816E] bit, to temporarily disable ISO-7816 mode.
2. Write 1 to the WT, CWT, or BWT bits that need to be cleared.
3. Set UART_C7816[ISO_7816E] to re-enable ISO-7816 mode.

Note that the timers will start counting again as soon as the ISO_7816E bit is set. To avoid unwanted timeouts, software might need to wait until new transmit or receive traffic is expected or desired before re-enabling ISO-7816 mode.

e7029: UART: In ISO-7816 T=1 mode, CWT interrupts assert at both character and block boundaries

Description: When operating in ISO-7816 T=1 mode and switching from transmission to reception block, the character wait time interrupt flag (UART_IS7816[CWT]) should not be set, only block type interrupts should be valid. However, the UART can set the CWT flag while switching from transmit to receive block and at the start of transmit blocks.

Workaround: If a CWT interrupt is detected at a block boundary instead of a character boundary, then the interrupt flag should be cleared and otherwise ignored.

e7031: UART: In single wire receive mode UART will attempt to transmit if data is written to UART_D

Description: If transmit data is loaded into the UART_D register while the UART is configured for single wire receive mode, the UART will attempt to send the data. The data will not be driven on the pin, but it will be shifted out of the FIFO and the UART_S1[TDRE] bit will set when the character shifting is complete.

Workaround: Do not queue up characters to transmit while the UART is in receive mode. Always write UART_C3[TXDIR] = 1 before writing to UART_D in single wire mode.

e5704: UART: TC bit in UARTx_S1 register is set before the last character is sent out in ISO7816 T=0 mode

Description: When using the UART in ISO-7816 mode, the UARTx_S1[TC] flag sets after a NACK is received, but before guard time expires.

Workaround: If using the UART in ISO-7816 mode with T=0 and a guard time of 12 ETU, check the UARTn_S1[TC] bit after each byte is transmitted. If a NACK is detected, then the transmitter should be reset.

The recommended code sequence is:

```
UART0_C2 &= ~UART_C2_TE_MASK; //make sure the transmitter is disabled at first
UART0_C3 |= UART_C3_TXDIR_MASK; //set the TX pin as output
UART0_C2 |= UART_C2_TE_MASK; //enable TX
UART0_C2 |= UART_C2_RE_MASK; //enable RX to detect NACK
for(i=0;i<length;i++) { while(!(UART0_S1&UART_S1_TDRE_MASK)){}
UART0_D = data[i]; while(!(UART0_S1&UART_S1_TC_MASK)){} //check for NACK
if(UART0_IS7816 & UART_IS7816_TXT_MASK) //check if TXT flag set { /* Disable transmit to clear the internal NACK detection counter */
UART0_C2 &= ~UART_C2_TE_MASK;
UART0_IS7816 = UART_IS7816_TXT_MASK; // write one to clear TXT
UART0_C2 |= UART_C2_TE_MASK; // re-enable transmit } }
UART0_C2 &= ~UART_C2_TE_MASK; //disable after transmit
```

e7091: UART: UART_S1[NF] and UART_S1[PE] can set erroneously while UART_S1[FE] is set

Description: While the UART_S1[FE] framing error flag is set the UART will discard any received data. Even though the data is discarded, if characters are received that include noise or parity errors, then the UART_S1[NF] or UART_S1[PE] bits can still set. This can lead to triggering of unwanted interrupts if the parity or noise error interrupts are enabled and framing error interrupts are disabled.

Workaround: If a framing error is detected (UART_S1[FE] = 1), then the noise and parity error flags can be ignored until the FE flag is cleared. Note: the process to clear the FE bit will also clear the NF and PE bits.

e7092: UART: UART_S1[TC] is not cleared by queuing a preamble or break character

Description: The UART_S1[TC] flag can be cleared by first reading UART_S1 with TC set and then performing one of the following: writing to UART_D, queuing a preamble, or queuing a break character. If the TC flag is cleared by queuing a preamble or break character, then the flag will clear as expected the first time. When TC sets again, the flag can be cleared by any of the three clearing mechanisms without reading the UART_S1 register first. This can cause a TC flag occurrence to be missed.

Workaround: If preamble and break characters are never used to clear the TC flag, then no workaround is required.

If a preamble or break character is used to clear TC, then write UART_D immediately after queuing the preamble or break character.

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[USBFAC] and SIM_CLKDIV2[USBDIV] bit fields.

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.

e8101: USBOTG: USB host signal crossover voltage higher than specification at low temperature

Description: When using the USB in host mode, some of the USB signal timing from the on-chip FS transceiver can be marginal if the ambient temperature is below -20°C. The USB signal timing issues can appear as a conditional pass conditions for crossover voltage.

Workaround: Operate at temperatures above -20°C if the marginal crossover voltage is not acceptable.

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

Information in this document is provided solely to enable system and software implementers to use NXP 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. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP 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 NXP 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. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, Freescale, the Freescale logo, and Kinetis are trademarks of NXP B.V. All other product or service names are the property of their respective owners. Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

© 2014 - 2017 NXP B.V.

Document Number: KINETIS_K_1N88J
Revision 01 SEP 2017

