Mask Set Errata for Mask 2N36B

This report applies to mask 2N36B for these products:

- MK10DX64xxx7, MK10DX128xxx7, MK10DX256xxx7
- MK20DX64xxx7, MK20DX128xxx7, MK20DX256xxx7
- MK30DX64xxx7, MK30DX128xxx7, MK30DX256xxx7
- MK40DX64xxx7, MK40DX128xxx7, MK40DX256xxx7
- MK50DX128xxx7, MK50DX256xxx7
- MK51DX128xxx7, MK51DX256xxx7

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
e6804	CJTAG: Performing a mode change from Standard Protocol to Advanced Protocol may reset the CJTAG.
e6990	CJTAG: possible incorrect TAP state machine advance during Check Packet
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
e4588	DMAMUX: When using PIT with "always enabled" request, DMA request does not deassert correctly
e6933	eDMA: Possible misbehavior of a preempted channel when using continuous link mode
e5751	FTFx: Launching the Read 1's Section command (RD1SEC) on an entire flash block results in access error (ACCER).
e4710	FTM: FTMx_PWMLOAD register does not support 8-/16-bit accesses
e6484	FTM: The process of clearing the FTMx_SC[TOF] bit does not work as expected under a certain condition when the FTM counter reaches FTM_MOD value.
e6573	JTAG: JTAG TDO function on the PTA2 disables the pull resistor
e7214	Low Leakage Stop (LLS) mode non-functional
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
e4590	MCG: Transitioning from VLPS to VLPR low power modes while in BLPI clock mode is not supported.
e6665	Operating requirements: Limitation of the device operating range
e5667	PMC: When used as an input to ADC or CMP modules, the PMC bandgap 1-V voltage reference is not available in VLPx, LLS, or VLLSx modes

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
e5130	SAI: Under certain conditions, the CPU cannot reenter STOP mode via an asynchronous interrupt wakeup event
e4218	SIM/FLEXBUS: SIM_SCGC7[FLEXBUS] bit should be cleared when the FlexBus is not being used.
e5952	SMC: Wakeup via the LLWU from LLS/VLLS to RUN to VLPR incorrectly triggers an immediate wakeup from the next low power mode entry
e2638	TSI: The counter registers are not immediately updated after the EOSF bit is set.
e3926	TSI: The TSI will run several scan cycles during reference clock instead of scanning each electrode once
e4181	TSI: When the overrun flag is set, the TSI scanning sequence will exhibit undefined behavior.
e4935	UART: CEA709.1 features not supported
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
e8184	UART: During ISO-7816 T=0, TC bit set at 12 ETUs may cause loss of characters when UART is switched from transmit to receive mode
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
e3892	UART: ISO-7816 automatic initial character detect feature not working correctly
e4945	UART: ISO-7816 T=1 mode receive data format with a single stop bit is not supported
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
e5928	USBOTG: USBx_USBTRC0[USBRESET] bit does not operate as expected in all cases

Table 2. Revision History

Revision	Changes
12 Feb 2015	Added errata: e6804, e6990, e6939, e6940, e6933, e6484, e6573, e7214, e7993, e7735, e7028, e7027, e8184, e6472, e7090, e7029, e7031, e7091, e7092, e8807, e4647 Removed errata: e3863, e2582, e5666

e6804: CJTAG: Performing a mode change from Standard Protocol to Advanced Protocol may reset the CJTAG.

Description: In extremely rare conditions, when performing a mode change from Standard Protocol to Advanced Protocol on the IEEE 1149.7 (Compact JTAG interface), the CJTAG may reset itself. In this case, all internal CJTAG registers will be reset and the CJTAG will return to the Standard Protocol mode.

Workaround: If the CJTAG resets itself while attempting to change modes from Standard Protocol to Advanced Protocol and Advanced Protocol cannot be enabled after several attempts, perform future accesses in Standard Protocol mode and do not use the Advanced Protocol feature.

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.

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.

e4588: DMAMUX: When using PIT with "always enabled" request, DMA request does not deassert correctly

Description: The PIT module is not assigned as a stand-alone DMA request source in the DMA request mux. Instead, the PIT is used as the trigger for the DMAMUX periodic trigger mode. If you want to use one of the PIT channels for periodic DMA requests, you would use the periodic trigger mode in conjunction with one of the “always enabled” DMA requests. However, the DMA request does not assert correctly in this case.

Instead of sending a single DMA request every time the PIT expires, the first time the PIT triggers a DMA transfer the “always enabled” source will not negate its request. This results in the DMA request remaining asserted continuously after the first trigger.

Workaround: Use of the PIT to trigger DMA channels where the major loop count is greater than one is not recommended. For periodic triggering of DMA requests with major loop counts greater than one, we recommended using another timer module instead of the PIT.

If using the PIT to trigger a DMA channel where the major loop count is set to one, then in order to get the desired periodic triggering, the DMA must do the following in the interrupt service routine for the DMA_DONE interrupt:

1. Set the DMA_TCDn_CSR[DREQ] bit and configure DMAMUX_CHCFGn[ENBL] = 0

2. Then again DMAMUX_CHCFGn[ENBL] = 1, DMASREQ=channel in your DMA DONE interrupt service routine so that “always enabled” source could negate its request then DMA request could be negated.

This will allow the desired periodic triggering to function as expected.

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

Description: When using continuous link mode (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 (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.

e5751: FTFx: Launching the Read 1's Section command (RD1SEC) on an entire flash block results in access error (ACCER).

Description: FTFx: Launching the Read 1's Section command on an entire flash block (i.e. with flash address = flash block base address & number of longwords = total number of longwords in the flash block) results in an incorrectly asserted access error (ACCER).

Workaround: To verify an entire flash block, use the Read 1's Block command. Use the Read 1's Section command only to verify sections that are smaller than an entire flash block.

e4710: FTM: FTMx_PWMLOAD register does not support 8-/16-bit accesses

Description: The FTM PWM Load register should support 8-bit and 16-bit accesses. However, the FTMx_PWMLOAD[LDOK] bit is cleared automatically by FTM with these sized accesses, thus disabling the loading of the FTMx_MOD, FTMx_CNTIN, and FTMx_CnV registers.

Workaround: Always use a 32-bit write access to modify contents of the FTMx_PWMLOAD register.

e6484: FTM: The process of clearing the FTMx_SC[TOF] bit does not work as expected under a certain condition when the FTM counter reaches FTM_MOD value.

Description: The process of clearing the TOF bit does not work as expected when FTMx_CONF[NUMTOF] != 0 and the current TOF count is less than FTMx_CONF[NUMTOF], if the FTM counter reaches the FTM_MOD value between the reading of the TOF bit and the writing of 0 to the TOF bit. If the above condition is met, the TOF bit remains set, and if the TOF interrupt is enabled (FTMx_SC[TOIE] = 1), the TOF interrupt also remains asserted.

Workaround: Two possible workarounds exist for this erratum and the decision on which one to use is based on the requirements of your particular application.

1) Repeat the clearing sequence mechanism until the TOF bit is cleared.

Below is a pseudo-code snippet that would need to be included in the TOF interrupt routine.

while (FTM_SC[TOF]!=0) { void FTM_SC() ; // Read SC register FTM_SC[TOF]=0 ; // Write 0 to TOF bit }

2) With FTMx_CONF[TOFNUM] = 0 and a variable in the software, count the number of times that the TOF bit is set. In the TOF interrupt routine, clear the TOF bit and increment the variable that counts the number of times that the TOF bit was set.

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.

e7214: Low Leakage Stop (LLS) mode non-functional

Description: On some devices, system and peripheral memories may be corrupted when a device exits the Low Leakage Stop (LLS) mode.

Workaround: All other low power modes are not affected. Use VLPS or VLLSx mode.

A silicon revision to correct the errata is planned.

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.

e4590: MCG: Transitioning from VLPS to VLPR low power modes while in BLPI clock mode is not supported.

Description: Transitioning from VLPS mode back to VLPR (LPWUI control bit = 0) while using BLPI clock mode only, is not supported. During Fast IRC startup, the output clock frequency may exceed the maximum VLPR operating frequency. This does not apply to the BLPE clock mode.

Workaround: There are two options for workarounds

a) Exit to Run instead of VLPR. Before entering VLPR set the LPWUI bit so that when exiting VLPS mode the MCU exits to RUN mode instead of VLPR mode. With LPWUI set any interrupt will exit VLPR or VLPS back into RUN mode. To minimize the impact of the higher RUN current re-enter VLPR quickly.

or

b) Utilize MCG clock mode BLPE when transitioning from VLPS to VLPR modes.

e6665: Operating requirements: Limitation of the device operating range

Description: Some devices, when power is applied, may not consistently begin to execute code under certain voltage and temperature conditions. Applications that power up with either VDD >= 2.0 V or temperature >= -20C are not impacted. Entry and exit of low-power modes is not impacted.

Workaround: To avoid this unwanted behavior, one or both of these conditions must be met:

a) Perform power on reset of the device with a supply voltage (VDD) equal-to or greater-than 2.0 V , or

b) Perform power on reset of the device at a temperature at or above -20 C.

e5667: PMC: When used as an input to ADC or CMP modules, the PMC bandgap 1-V voltage reference is not available in VLPx, LLS, or VLLSx modes

Description: The Power Management Controller (PMC) bandgap 1-V reference is not available as an input to the Analog-to-Digital Converter (ADC) module (using ADC input channel AD27) or the Comparator (CMP) module (using CMP input IN6) in Very Low Power Run (VLPR), Very Low Power Wait (VLPW), Very Low Power Stop (VLPS), Low Leakage Stop (LLS), Very Low Leakage Stop3 (VLLS3), Very Low Leakage Stop2 (VLLS2), Very Low Leakage Stop1 (VLLS1), or Very Low Leakage Stop0 (VLLS0) modes.

This erratum does not apply to the VREF module 1.2 V reference voltage.

Workaround: Use of the PMC bandgap 1-V reference voltage as an input to the ADC and CMP modules requires the MCU to be in Run, Wait, or Stop modes.

e5130: SAI: Under certain conditions, the CPU cannot reenter STOP mode via an asynchronous interrupt wakeup event

Description: If the SAI generates an asynchronous interrupt to wake the core and it attempts to reenter STOP mode, then under certain conditions the STOP mode entry is blocked and the asynchronous interrupt will remain set.

This issue applies to interrupt wakeups due to the FIFO request flags or FIFO warning flags and then only if the time between the STOP mode exit and subsequent STOP mode reentry is less than 3 asynchronous bit clock cycles.

Workaround: Ensure that at least 3 bit clock cycles elapse following an asynchronous interrupt wakeup event, before STOP mode is reentered.

e4218: SIM/FLEXBUS: SIM_SCGC7[FLEXBUS] bit should be cleared when the FlexBus is not being used.

Description: The SIM_SCGC7[FLEXBUS] bit is set by default. This means that the FlexBus will be enabled and come up in global chip select mode.

With some code sequence and register value combinations the core could attempt to prefetch from the FlexBus even though it might not actually use the value it prefetched. In the case where the FlexBus is unconfigured, this can result in a hung bus cycle on the FlexBus.

Workaround: If the FlexBus is not being used, disabled the clock to the FlexBus during chip initialization by clearing the SIM_SCGC7[FLEXBUS] bit.

If the FlexBus will be used, then enable at least one chip select as early in the chip initialization process as possible.

e5952: SMC: Wakeup via the LLWU from LLS/VLLS to RUN to VLPR incorrectly triggers an immediate wakeup from the next low power mode entry

Description: Entering VLPR immediately after an LLWU wakeup event from LLS/VLLS, will cause any subsequent entry into LLS/VLLS to fail if entry into VLPR mode occurs before clearing the pending LLWU interrupt.

Workaround: After an LLWU wakeup event from LLS/VLLS, the user must clear the LLWU interrupt prior to entering VLPR mode.

e2638: TSI: The counter registers are not immediately updated after the EOSF bit is set.

Description: The counter registers are not immediately updated after the end of scan event (EOSF is set). The counter registers will become available 0.25 ms after the EOSF flag is set. This also applies for the end-of-scan interrupt, as it is triggered with the EOSF flag. This behavior will occur both in continuous scan and in software triggered scan modes.

Workaround: Insert a delay of 0.25 ms or greater prior to accessing the counter registers after an end of scan event or an end of scan interrupt that is triggered by the EOSF flag. This delay does not need to be a blocking delay, so it can be executing other actions before reading the counter registers. Notice that the out-of-range flag (OUTRGF) and interrupt occur after the counters have been updated, so if the OUTRGF flag is polled or the out-of-range interrupt is used, the workaround is not necessary.

e3926: TSI: The TSI will run several scan cycles during reference clock instead of scanning each electrode once

Description: The TSI will run several scan cycles during reference clock instead of scanning each electrode once. For each automatic scanning period determined by AMCLKS (clock source), AMPSC (prescaler) and SMOD (period modulo), TSI will scan during one reference clock cycle divided by the AMPSC prescaler.

This does not affect the count result from TSI because TSI counters keep the last scan result.

Workaround: 1. Because counter results are not affected, a simple workaround is to use the smallest prescaler possible and use a bigger SMOD value, this will minimize the number of extra scans, thus also minimizing the amount of average extra current used by the module.

2. If strict control of number of scan cycles is needed, trigger scans with software control (using the SWTS bit) and control time between scans with a separate timer. This solution is only recommended if strict control of scan cycles is needed, if not, recommendation is to use workaround 1.

e4181: TSI: When the overrun flag is set, the TSI scanning sequence will exhibit undefined behavior.

Description: When the overrun flag is set, the TSI scanning sequence will exhibit undefined behavior, so the results of measurements are invalid at this point. In order to continue reading valid measurements, disable the TSI module and reconfigure it.

Workaround: During development make sure to measure the required scanning time for all the electrodes in your system and configure the scanning time with AMCLKS, AMPSC and SMOD so that an overrun will not happen. Consider adding about 30 to 70% more time as headroom to make sure overrun is not triggered. If scanning time is critical and added scan time is not acceptable, detect the overrun condition either by polling the overrun flag in a loop or through the TSI interrupt. Once overrun is detected, disable the TSI module, clear all flags and reconfigure. During reconfiguration, SMOD can be increased by 10% or more of the current value to reduce the number of overrun occurrences.

e4935: UART: CEA709.1 features not supported

Description: Due to some issues that affect compliance with the specification, the CEA709.1 features of the UART module are not supported. Normal UART mode, IrDA, and ISO-7816 are unaffected.

Workaround: Do not use the UART in CEA709.1 mode.

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 charcter 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 charcter 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.

e8184: UART: During ISO-7816 T=0, TC bit set at 12 ETUs may cause loss of characters when UART is switched from transmit to receive mode

Description: In ISO-7816 T=0 mode, if S1[TC] is set at 12 ETUs to indicate end of transmission and software then switches the UART to receive mode by setting C2[RE], the first received character may be lost.

Workaround: For EMV card applications, no workaround is required since the maximum turnaround time for EMV-compliant cards is 15 ETUs, per the EMV L1 test specification (1CF.004.00).

No workaround is available for ISO-7816-compliant cards.

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.

e3892: UART: ISO-7816 automatic initial character detect feature not working correctly

Description: The ISO-7816 automatic initial character detection feature does not work. The direct convention initial character can be detected correctly, but the inverse convention initial character will only be detected if the S2[MSBF] and S2[RXINV] bits are set. This defeats the purpose of the initial character detection and automatic configuration of the S2[MSBF], S2[RXINV], and C3[TXINV] bits.

Workaround: Use software to manually detect initial characters. Configure the UART with S2[MSBF] and S2[RXINV] cleared. Then check UART receive characters looking for 0x3B or 0x03. If 0x3B is received, then the connected card is direct convention. If 0x03 is received, then the connected card is inverse convention. If an inverse convention card is detected, then software should set S2[MSBF], S2[RXINV], and C3[TXINV].

e4945: UART: ISO-7816 T=1 mode receive data format with a single stop bit is not supported

Description: Transmission of ISO-7816 data frames with single stop bit is supported in T=1 mode. Currently in order to receive a frame, two or more stop bits are required. This means that 11 ETU reception based on T=1 protocol is not supported. T=0 protocol is unaffected.

Workaround: Do not send T=1, 11 ETU frames to the UART in ISO-7816 mode. Use 12 ETU transmissions for T=1 protocol instead.

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[USBFRAC] and SIM_CLKDIV2[USBDIV] bit fields.

e5928: USBOTG: USBx_USBTRC0[USBRESET] bit does not operate as expected in all cases

Description: The USBx_USBTCR0[USBRESET] bit is not properly synchronized. In some cases using the bit can cause the USB module to enter an undefined state.

Workaround: Do not use the USBx_USBTCR0[USBRESET] bit. If USB registers need to be written to their reset states, then write those registers manually instead of using the module reset bit.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: KINETIS_K_2N36B
Rev. 12FEB2015