Mask Set Errata for Mask 0N74K

This report applies to mask 0N74K for these products:

- MK22FN128VDC10, MK22FN128VLL10, MK22FN128VMP10
- MK22FN128VLH10, MK22FN128CAK10R

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
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
e8096	DAC12: DNL for DAC12 operating in Low-Power Mode larger than specification for some devices
e8011	eDMA: Possible corruption when writing an eDMA descriptor while the eDMA is active
e10121	FTFA: For MCUs prior to work week 14 of 2016, FSEC[MEEN] = 10 disables Mass Erase only when the MCU is secured.
e8162	I2C: IAAS and IICIF bits in the I2C Status Register are not set properly under certain conditions after low power recovery.
e9004	ITM can deadlock when global timestamping is enabled
e10123	Kinetis Flashloader/ ROM Bootloader: For MCUs prior to work week 14 of 2016, the peripheral auto-detect code in bootloader can falsely detect presence of SPI host causing non-responsive bootloader
e10134	Kinetis Flashloader/ROM Bootloader: For MCUs prior to work week 14 of 2016, memory data read can be incorrect.
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
e7986	LPUART: The LPUART_TX pin is tri-stated when the transmitter is disabled
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.
e8361	SMC: Compute Operation cannot be enabled in HSRUN mode

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
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
e4647	UART: Flow control timing issue can result in loss of characters if FIFO is not enabled
e7857	UART: WT timer in T=0 mode and CWT timer in T=1 mode can expire between 0.2 ETU to 0.8 ETU earlier than programmed.
e8807	USB: In Host mode, transmission errors may occur when communicating with a Low Speed (LS) device through a USB hub
e7998	USB: USB host signal crossover voltage higher than specification at low temp or high voltage
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
e9646	WDOG:Unexpected watchdog behavior on LLS exit

Table 2. Revision History

Revision	Changes
22 MAY 2014	Initial revision
20 JULY 2016	The following errata were added. • e10123 • e8184 • e8807 • e8992 • e10134 • e8096 • e9004 • e9005 • e8361 • e10121 • e9646 • e8162

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.

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.

e8096: DAC12: DNL for DAC12 operating in Low-Power Mode larger than specification for some devices

Description: Some devices may have a DNL up to ±2 LSB when DAC is configured to operate in Low-Power Mode (DACx_C0[LPEN] set to 1).

Workaround: To achieve a DNL of ±1 LSB, configure the DAC to operate in High-Power Mode by setting DACx_C0[LPEN] to 0.

e8011: eDMA: Possible corruption when writing an eDMA descriptor while the eDMA is active

Description: If software writes to the SADDR, DADDR, or NBYTES field of an idle eDMA channel at the same time the eDMA engine is writing the same field of an active channel, the eDMA engine write will corrupt the software write.

Workaround: There are two potential workarounds for this issue:

Ensure that the eDMA is idle before writing TCD values by following this procedure:

1) Halt the DMA via the HALT bit or by disabling each channel.

2) Verify DMA is inactive via the ACTIVE bit.

3) Write the TCD information.

4) Re-enable the DMA (clear the HALT bit or enable each channel).

or

Verify all writes to SADDR, DADDR, and NBYTES before starting a DMA channel or enabling the external request:

1) Write SADDR, DADDR, or NBYTES.

2) Read-back and verify contents of each field.

3) If contents of any fields do not match the expected values, return to step 1.

4) After all contents have been verified as written correctly, then the DMA channel can be started using a software request or the external request for the channel can be enabled.

e10121: FTFA: For MCUs prior to work week 14 of 2016, FSEC[MEEN] = 10 disables Mass Erase only when the MCU is secured.

Description: MCUs manufactured prior to work week 14 of 2016 do not have the following features:

1) FSEC[MEEN] = 10 disables Mass Erase in all MCU operations including

a) debug mode using MDM-AP

b) EZPORT

c) internal Erase all block commands

2) Flash commands

a) 0x4A Read 1s All Execute only Segments

b) 0x4B Erase All Execute-only Segments

Workaround: If these features are required, please obtain MCUs manufactured during work week 14 of 2016 or later.

Read the Flash Version ID via the Read Resource Flash command or look at the date code marked on the device to confirm the availability of the features referenced above.

- Flash Version ID 08.01.01.00 indicates the features are available.
- Date Code 1614 (YYWW) or EN (YW) and later indicates the features are available.

Note: The last line marked on the part contains the date code.

- For parts with 8 characters on the last line, the date code is characters 4-7
- For parts with 6 characters on the last line, the date code is characters 4-5
- For parts with 5 characters on the last line, the date code is characters 3-4

e8162: I2C: IAAS and IICIF bits in the I2C Status Register are not set properly under certain conditions after low power recovery.

Description: The IAAS and IICIF bits in the I2C Status Register indicating an address match and a pending interrupt will not be set if recovering from a low power mode and the Bus Clock is less than ¼ of the Core Clock. Only the TCF flag will be set.

Workaround: When recovering from LLSx/VLLSx modes, the Asynchronous Wake-up Interrupt Controller (AWIC) correctly identifies the source of wakeup and can be used to confirm valid I2C addressing. For other low power modes, the TCF flag in conjunction with a software semaphore that establishes recovery from low power modes can be used to determine address match.

e9004: ITM can deadlock when global timestamping is enabled

Description: 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.

e10123: Kinetis Flashloader/ ROM Bootloader: For MCUs prior to work week 14 of 2016, the peripheral auto-detect code in bootloader can falsely detect presence of SPI host causing non-responsive bootloader

Description: During the active peripheral detection process, the bootloader can interpret spurious data on the SPI peripheral as valid data. The spurious data causes the bootloader to shutdown all peripherals except the “falsely detected" SPI and enter the command phase loop using the SPI. After the bootloader enters the command phase loop using the SPI, the other peripherals are ignored, so the desired peripheral is no longer active.

The bootloader will not falsely detect activity on the I2C, UART, or USB interfaces, so only the SPI interface is affected.

Workaround: If this feature is required, please obtain MCUs manufactured during work week 14 of 2016 or later, otherwise, ensure that there is an external pull-up on the SPI chip-select pin or that the pin is driven high. This will prevent the bootloader from seeing spurious data due to activity on the SPI clock pin.

Read the Flash Version ID via the Read Resource Flash command or look at the date code marked on the device to confirm the availability of the features referenced above.

- Flash Version ID 08.01.01.00 indicates the features are available.

- Date Code 1614 (YYWW) or EN (YW) and later indicates the features are available.

Note: The last line marked on the part contains the date code.

- For parts with 8 characters on the last line, the date code is characters 4-7

- For parts with 6 characters on the last line, the date code is characters 4-5

- For parts with 5 characters on the last line, the date code is characters 3-4

e10134: Kinetis Flashloader/ROM Bootloader: For MCUs prior to work week 14 of 2016, memory data read can be incorrect.

Description: A read data command can return incorrect data due to the lack of a cache clear before the read. The read will return what is in the cache instead of the actual memory contents.

For example in the following sequence, step 4 will return incorrect data

1) Program 10 byte data to 0xA000 --> Response status = 0 (0x0) Success.

2) Read data at 0xA000 --> aa bb cc dd 01 02 03 04 05 06 ff ff ff ff ff ff

3) Erase sector at address: 0xA000 --> Response status = 0 (0x0) Success

4) Read data at 0xA000 --> aa bb cc dd 01 02 03 04 05 06 ff ff ff ff ff ff

when the Read data at 0xA000 --> should be ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

Workaround: If this feature is required, please obtain MCUs manufactured during work week 14 of 2016 or later, otherwise, read a different flash location outside of the cache range then read the desired flash address to read the correct data.

Read the Flash Version ID via the Read Resource Flash command or look at the date code marked on the device to confirm the availability of the feature referenced above.

- Flash Version ID 08.01.01.00 indicates the features are available.

- Date Code 1614 (YYWW) or EN (YW) and later indicates the features are available.

Note: The last line marked on the part contains the date code.

- For parts with 8 characters on the last line, the date code is characters 4-7

- For parts with 6 characters on the last line, the date code is characters 4-5

- For parts with 5 characters on the last line, the date code is characters 3-4

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.

e7986: LPUART: The LPUART_TX pin is tri-stated when the transmitter is disabled

Description: The LPUART transmitter is disabled when the MCU:

- Enters Stop, Wait, or VLPS with the DOZEN bit set
- Enters LLS or VLLS power mode.

The LPUART will tri-state the LPUART_TX pin when the transmitter is disabled, which may result in leakage current.

Workaround: Before the MCU enters Stop, Wait, or VLPS power mode with the DOZEN bit set or enters the LLS or VLLS power mode, enable the pullup resistor on the LPUART_TX pin to ensure the pin does not float. If the TXINV bit is set, enable the pulldown resistor on the LPUART_TX pin.

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.

e8361: SMC: Compute Operation cannot be enabled in HSRUN mode

Description: Compute Operation, which keeps the CPU enabled with full access to the SRAM and Flash read ports while placing all other bus masters and bus slaves into their stop mode, cannot be enabled in HSRUN mode.

Workaround: Transition from HSRUN to RUN mode to enable Compute Operation via the MCM_CPO[CPOREQ]=1. Upon exiting Compute Operation (MCM_CPO[CPOREQ]=0), you can transition from RUN back to HSRUN mode.

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.

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.

e7857: UART: WT timer in T=0 mode and CWT timer in T=1 mode can expire between 0.2 ETU to 0.8 ETU earlier than programmed.

Description: In ISO7816 receive mode, the wait timer (WT) used in T=0 mode and the character wait timer (CWT) used in T=1 mode can expire between 0.2 ETU to 0.8 ETU earlier than programmed. The early expiration of the timers can cause software to discard valid data during communication.

Workaround: To minimize the possibility of discarded data, in T=0 mode, program the WI counter field for the wait timer as WWT + D *480 ETUs for both receive and transmit.

To minimize the possibility of discarded data, in T=1 mode, program the CWI counter field for the character wait timer as CWT + 5 ETUs when receiving and CWT + 4 ETUs when transmitting.

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.

e7998: USB: USB host signal crossover voltage higher than specification at low temp or high voltage

Description: When using the USB in host mode, some of the USB signal timing from the on-chip FS/LS transceiver can be marginal if the ambient temperature is below-20°C or if USBVDD is higher than 3.2 V. The USB signal timing issues can appear as conditional pass conditions for crossover voltage.

Workaround: Operate at temperatures above -20°C or with USBVDD lower than 3.2 V if the marginal crossover voltage is not acceptable.

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.

e9646: WDOG:Unexpected watchdog behavior on LLS exit

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

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

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: KINETIS_K_0N74K
Rev. 20 JULY 2016