KINETIS_K_0N87R
Rev. 03DEC2015

Mask Set Errata for Mask 0N87R

This report applies to mask 0N87R for these products:

- MKS22FN256VLH12
- MKS22FN256VLL12
- MKS22FN128VLH12
- MKS22FN128VLL12
- MKS22FN256VFT12
- MKS22FN128VFT12

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
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
e9878	MCG: Clock transition may have an issue immediately after writing the OSCSEL or RANGE bit fields in MCG control registers
e7735	MCG: IREFST status bit may set before the IREFS multiplexor switches the FLL reference clock
e9682	SPI: Inconsistent loading of shift register data into the receive FIFO following an overflow event
e4647	UART: Flow control timing issue can result in loss of characters if FIFO is not enabled
e2580	UART: Start bit sampling not compliant with LIN 2.1 specification

Table 2. Revision History

Revision	Changes
03DEC2015	Initial revision

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.

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

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

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

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

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

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

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

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

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

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

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

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

2 cycles of the slow IRC + 2 cycles of OSCERCLK

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

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

2 cycles of the slow IRC + 2 cycles of OSCERCLK

after MCG_S[IREFST] has been set to 1.

e9682: SPI: Inconsistent loading of shift register data into the receive FIFO following an overflow event

Description: In the Serial Peripheral Interface (SPI) module, when both the receive FIFO and shift register are full (Receive FIFO Overflow Flag bit in Status Register is set (SR [RFOF] = 0b1)) and then the Clear Rx FIFO bit in Module Configuration Register (MCR [CLR_RXF]) is asserted to clear the receive FIFO, shift register data is loaded into the receive FIFO after the clear operation completes.

Workaround: 1. Avoid a receive FIFO overflow condition (SR[RFOF] should never be 0b1). To do this, monitor the RX FIFO Counter field of the Status Register (SR[RXCTR]) which indicates the number of entries in receive FIFO and clear before the counter equals the FIFO depth.

2. Alternatively, after every receive FIFO clear operation (MCR[CLR_RXF] = 0b1) following a receive FIFO overflow (SR[RFOF] = 0b1) scenario, perform a single read from receive FIFO and discard the read data.

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.

e2580: UART: Start bit sampling not compliant with LIN 2.1 specification

Description: The LIN 2.1 specification states that start bits should be checked at sample 7, 8, 9, and 10. The UART module checks the start bit at samples 3, 5, and 7 instead.

Workaround: Start bits longer than 5/16 of a bit time are guaranteed to be recognized. Start bits shorter than this should not be used with this version of the UART because they might not be recognized.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: Kinetis_L_0N87R
Rev. 03DEC2015