Mask Set Errata for Mask 0N62J

This document contains errata information for Kinetis Mask Set 0N62J but excludes any information on selected security-related modules.

A nondisclosure agreement (NDA) is required for any security-related module information.

For more information on obtaining an NDA, please contact your local Freescale sales representative.

Mask Set Errata for Mask 0N62J

Introduction

This report applies to mask 0N62J for these products:

- KINETIS

Errata ID	Errata Title
6804	CJTAG: Performing a mode change from Standard Protocol to Advanced Protocol may reset the CJTAG.
6990	CJTAG: possible incorrect TAP state machine advance during Check Packet
6939	Core: Interrupted loads to SP can cause erroneous behavior
4588	DMAMUX: When using PIT with "always enabled" request, DMA request does not deassert correctly
4710	FTM: FTMx_PWMLOAD register does not support 8-/16-bit accesses
6484	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.
6573	JTAG: JTAG TDO function on the PTA2 disables the pull resistor
5130	SAI: Under certain conditions, the CPU cannot reenter STOP mode via an asynchronous interrupt wakeup event
7027	UART: During ISO-7816 T=0 initial character detection invalid initial characters are stored in the RxFIFO
7028	UART: During ISO-7816 initial character detection the parity, framing, and noise error flags can set
6472	UART: ETU compensation needed for ISO-7816 wait time (WT) and block wait time (BWT)
4647	UART: Flow control timing issue can result in loss of characters if FIFO is not enabled
7029	UART: In ISO-7816 T=1 mode, CWT interrupts assert at both character and block boundaries
7090	UART: In ISO-7816 mode, timer interrupts flags do not clear
7031	UART: In single wire receive mode UART will attempt to transmit if data is written to UART_D
5704	UART: TC bit in UARTx_S1 register is set before the last character is sent out in ISO7816 T=0 mode
7091	UART: UART_S1[NF] and UART_S1[PE] can set erroneously while UART_S1[FE] is set
7092	UART: UART_S1[TC] is not cleared by queuing a preamble or break character
6933	eDMA: Possible misbehavior of a preempted channel when using continuous link mode

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

Errata type: Errata

Description: In extremely rare conditions, when performing a mode change from Standard Protocol to Advanced Protocol on trhe 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

Errata type: Errata

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

Errata type: Errata

Description: ARM Errata 752770: Interrupted loads to SP can cause erroneous behavior

Affects: Cortex-M4, Cortex-M4F

Fault Type: Programmer Category B

Fault Status: Present in: r0p0, r0p1 Open.

Description

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: 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.

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

Errata type: Errata

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.

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

Errata type: Errata

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.

Errata type: Errata

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

Errata type: Errata

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.

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

Errata type: Errata

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.

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

Errata type: Errata

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.

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

Errata type: Errata

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.

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

Errata type: Errata

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

Errata type: Errata

Description: On UART0 and UART1 when /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 UART0 or UART1. 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 UART0 and UART1 are affected. The UARTs that do not have the RxFIFO feature are not affected.

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

Errata type: Errata

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.

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

Errata type: Errata

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.

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

Errata type: Errata

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

Errata type: Errata

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

Errata type: Errata

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

Errata type: Errata

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.

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

Errata type: Errata

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.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: KINETIS_0N62J
Rev. 10 DEC 2013
02/2014