KINETIS_K_5N22D

Mask Set Errata

Rev. 2 — 2 September 2025 **Errata**

1 Mask Set Errata for Mask 5N22D

1.1 Revision History

This report applies to mask 5N22D for these products:

- MK40DX256Vxx10
- MK60DX256Vxx10
- MK50DN512Cxx10
- MK50DX256Cxx10
- MK51DN256Cxx10
- MK51DN512Cxx10
- MK51DX256Cxx10
- MK52DN512Cxx10
- MK53DN512Cxx10
- MK53DX256Cxx10
- MK10DN512Vxx10
- MK10DX128Vxx10
- MK10DX256Vxx10
- MK20DN512Vxx10
- MK20DX128Vxx10
- MK20DX256Vxx10
- MK30DN512Vxx10
- MK30DX256Vxx10
- MK40DN512Vxx10
- MK60DN256Vxx10
- MK40DX128Vxx10
- MK60DN512Vxx10

Table 1. Revision History

Revision	Release Date	Significant Changes
2	9/2025	The following errata were added. • ERR052537 The following errata were revised. • ERR003980 • ERR004647 • ERR006939 • ERR004624
1	8/2014	Initial Revision

1.2 Errata and Information Summary

Table 2. Errata and Information Summary

Erratum ID	Erratum Title
ERR052537	ADC: Incorrect ADC conversions when using high speed, long sample time or asynchronous clock configurations
ERR006804	CJTAG: Performing a mode change from Standard Protocol to Advanced Protocol may reset the CJTAG.
ERR006990	CJTAG: possible incorrect TAP state machine advance during Check Packet
ERR006939	Core: Interrupted loads to SP can cause erroneous behavior
ERR004588	DMAMUX: When using PIT with "always enabled" request, DMA request does not deassert correctly
ERR006933	eDMA: Possible misbehavior of a preempted channel when using continuous link mode
ERR006358	ENET: Write to Transmit Descriptor Active Register (ENET_TDAR) is ignored
ERR005641	FlexCAN: Module does not transmit a message that is enabled to be transmitted at a specific moment during the arbitration process.
ERR004710	FTM: FTMx_PWMLOAD register does not support 8-/16-bit accesses
ERR006484	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.
ERR006573	JTAG: JTAG TDO function on the PTA2 disables the pull resistor
ERR004590	MCG: Transitioning from VLPS to VLPR low power modes while in BLPI clock mode is not supported.
ERR005130	SAI: Under certain conditions, the CPU cannot reenter STOP mode via an asynchronous interrupt wakeup event
ERR003981	SDHC: ADMA fails when data length in the last descriptor is less or equal to 4 bytes
ERR003982	SDHC: ADMA transfer error when the block size is not a multiple of four
ERR004624	SDHC: AutoCMD12 and R1b polling problem
ERR003977	SDHC: Does not support Infinite Block Transfer Mode
ERR004627	SDHC: Erroneous CMD CRC error and CMD Index error may occur on sending new CMD during data transfer
ERR003984	SDHC: eSDHC misses SDIO interrupt when CINT is disabled
ERR003980	SDHC: Glitch is generated on card clock with software reset or clock divider change
ERR006934	SDHC: Issues with card removal/insertion detection
ERR003983	SDHC: Problem when ADMA2 last descriptor is LINK or NOP
ERR003978	SDHC: Software can not clear DMA interrupt status bit after read operation
ERR004218	SIM/FLEXBUS: SIM_SCGC7[FLEXBUS] bit should be cleared when the FlexBus is not being used.
ERR005952	SMC: Wakeup via the LLWU from LLS/VLLS to RUN to VLPR incorrectly triggers an immediate wakeup from the next low power mode entry
ERR002638	TSI: The counter registers are not immediately updated after the EOSF bit is set.
ERR004546	TSI: The counter values reported from TSI increase when in low power modes (LLS, VLLS1, VLLS2, VLLS3)

Table 2. Errata and Information Summary*...continued*

Erratum ID	Erratum Title
ERR003926	TSI: The TSI will run several scan cycles during reference clock instead of scanning each electrode once
ERR003927	TSI: TSI will scan continuously if only one electrode is enabled
ERR004935	UART: CEA709.1 features not supported
ERR007028	UART: During ISO-7816 initial character detection the parity, framing, and noise error flags can set
ERR007027	UART: During ISO-7816 T=0 initial character detection invalid initial characters are stored in the RxFIFO
ERR006472	UART: ETU compensation needed for ISO-7816 wait time (WT) and block wait time (BWT)
ERR004647	UART: Flow control timing issue can result in loss of characters if FIFO is not enabled
ERR007090	UART: In ISO-7816 mode, timer interrupts flags do not clear
ERR007029	UART: In ISO-7816 T=1 mode, CWT interrupts assert at both character and block boundaries
ERR007031	UART: In single wire receive mode UART will attempt to transmit if data is written to UART_D
ERR003892	UART: ISO-7816 automatic initial character detect feature not working correctly
ERR004945	UART: ISO-7816 T=1 mode receive data format with a single stop bit is not supported
ERR005704	UART: TC bit in UARTx_S1 register is set before the last character is sent out in ISO7816 T=0 mode
ERR007091	UART: UART_S1[NF] and UART_S1[PE] can set erroneously while UART_S1[FE] is set
ERR007092	UART: UART_S1[TC] is not cleared by queuing a preamble or break character
ERR005928	USBOTG: USBx_USBTRC0[USBRESET] bit does not operate as expected in all cases

2 Known Errata

ERR052537: ADC: Incorrect ADC conversions when using high speed, long sample time or asynchronous clock configurations

Description

When the ADC is configured to use high speed operation (ADHSC=1), long sample time (LSMP=1), or the asynchronous clock (ADICLK=11), the input comparator of the ADC can fail over time leading to incorrect conversion results.

Workaround

For new unused devices, do not use high speed configuration (ADHSC=1), long sample time (LSMP=1) or the asynchronous clock input (ADICLK=11).

ERR006804: 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.

ERR006990: 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.

ERR006939: 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.

ERR004588: 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.

ERR006933: 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.

ERR006358: ENET: Write to Transmit Descriptor Active Register (ENET_TDAR) is ignored

Description

If the ready bit in the transmit buffer descriptor (TxBD[R]) is previously detected as not set during a prior frame transmission, then the ENET_TDAR[TDAR] bit is cleared at a later time, even if additional TxBDs were added to the ring and the ENET_TDAR[TDAR] bit is set. This results in frames not being transmitted until there is a 0-to-1 transition on ENET_TDAR[TDAR].

Workaround

Code can use the transmit frame interrupt flag (ENET_EIR[TXF]) as a method to detect whether the ENET has completed transmission and the ENET_TDAR[TDAR] has been cleared. If ENET_TDAR[TDAR] is detected as cleared when packets are queued and waiting for transmit, then a write to the TDAR bit will restart TxBD processing.

ERR005641: FlexCAN: Module does not transmit a message that is enabled to be transmitted at a specific moment during the arbitration process.

Description

FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment during the arbitration process. The following conditions are necessary to have the issue.

- Only one MB is configured to be transmitted

- The write which enables the MB to be transmitted (write on Control status word) happens during a specific clock during the arbitration process.

After this arbitration process occurs, the bus goes to Idle state and no new message is received on bus.

For example:

1) MB13 is deactivated on RxIntermission (write 0x0 on CODE field from Control Status word) - First write on CODE

2) Reconfigure the ID and data fields

3) Enable the MB13 to be transmitted on BusIdle (write 0xC on Code field) - Second write on code

4) CAN bus keeps in Idle state

5) No write on Control status from any MB happens.

During the second write on code (step 3), the write must happen one clock before the current MB13 is to be scanned by arbitration process. In this case, it does not detect the new code (0xC) and no new arbitration is scheduled.

The problem can be detectable only if the message traffic ceases and the CAN bus enters into Idle state after the described sequence of events.

There is NO ISSUE if any of the conditions below holds:

a) Any MB (either Tx or Rx) is reconfigured (by writing its CS field) just after the Intermission field.

b) There is other configured MB to be transmitted

c) A new incoming message sent by any external node starts just after the Intermission field.

Workaround

To transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the following standard 5 step procedure:

1. Check if the respective interrupt bit is set and clear it.

2. If the MB is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control and Status word to request an abortion of the transmission.Wait for the corresponding IFLAG to be asserted by polling the IFLAG register or by the interrupt request if enabled by the respective IMASK. Then read back the CODE field to check if the transmission was aborted or transmitted. If backwards compatibility is desired (MCR[AEN] bit negated), just write the INACTIVE code (0b1000) to the CODE field to inactivate the MB but then the pending frame may be transmitted without notification.

3. Write the ID word.

4. Write the data bytes.

5. Write the DLC, Control and CODE fields of the Control and Status word to activate the MB.

The workaround consists of executing two extra steps:

6. Reserve the first valid mailbox as an inactive mailbox (CODE=0b1000). If RX FIFO is disabled, this mailbox must be MB0. Otherwise, the first valid mailbox can be found by using table "RX FIFO filters" on FlexCAN3 chapter.

7. Write twice INACTIVE code (0b1000) into the first valid mailbox.

Note: The first mailbox cannot be used for reception or transmission process.

ERR004710: 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.

ERR006484: 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.

ERR006573: 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.

ERR004590: 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.

ERR005130: 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.

ERR003981: SDHC: ADMA fails when data length in the last descriptor is less or equal to 4 bytes

Description

A possible data corruption or incorrect bus transactions on the internal AHB bus, causing possible system corruption or a stall, can occur under the combination of the following conditions:

1. ADMA2 or ADMA1 type descriptor

2. TRANS descriptor with END flag

3. Data length is less than or equal to 4 bytes (the length field of the corresponding descriptor is set to 1, 2, 3, or 4) and the ADMA transfers one 32-bit word on the bus

4. Block Count Enable mode

Workaround

The software should avoid setting ADMA type last descriptor (TRANS descriptor with END flag) to data length less than or equal to 4 bytes. In ADMA1 mode, if needed, a last NOP descriptor can be appended to the descriptors list. In ADMA2 mode this workaround is not feasible due to ERR003983.

ERR003982: SDHC: ADMA transfer error when the block size is not a multiple of four

Description

Issue in eSDHC ADMA mode operation. The eSDHC read transfer is not completed when block size is not a multiple of 4 in transfer mode ADMA1 or ADMA2. The eSDHC DMA controller is stuck waiting for the IRQSTAT[TC] bit in the interrupt status register.

The following examples trigger this issue:

1. Working with an SD card while setting ADMA1 mode in the eSDHC

2. Performing partial block read

3. Writing one block of length 0x200

4. Reading two blocks of length 0x22 each. Reading from the address where the write operation is performed. Start address is 0x512 aligned. Watermark is set as one word during read. This read is performed using only one ADMA1 descriptor in which the total size of the transfer is programmed as 0x44 (2 blocks of 0x22).

Workaround

When the ADMA1 or ADMA2 mode is used and the block size is not a multiple of 4, the block size should be rounded to the next multiple of 4 bytes via software. In case of write, the software should add the corresponding number of bytes at each block end, before the write is initialized. In case of read, the software should remove the dummy bytes after the read is completed.

For example, if the original block length is 22 bytes, and there are several blocks to transfer, the software should set the block size to 24. The following data is written/stored in the external memory:

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

2 Bytes valid data + 2 Byte dummy data

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

4 Bytes valid data

2 Bytes valid data + 2 Byte dummy data

In this example, 48 (24 x 2) bytes are transferred instead of 44 bytes. The software should remove the dummy data.

ERR004624: SDHC: AutoCMD12 and R1b polling problem

Description

Occurs when a pending command which issues busy is completed. For a command with R1b response, the proper software sequence is to poll the DLA for R1b commands to determine busy state completion. The DLA polling is not working properly for the ESDHC module and thus the DLA bit in PRSSTAT register cannot be polled to wait for busy state completion. This is relevant for all eSDHC ports (eSDHC1-4 ports).

Workaround

Poll bit 24 in PRSSTAT register (DLSL[0] bit) to check that wait busy state is over.

ERR003977: SDHC: Does not support Infinite Block Transfer Mode

Description

The eSDHC does not support infinite data transfers, if the Block Count register is set to one, even when block count enable is not set.

Workaround

The following software workaround can be used instead of the infinite block mode:

1. Set BCEN bit to one and enable block count

2. Set the BLKCNT to the maximum value in Block Attributes Register (BLKATTR) (0xFFFFfor 65535 blocks)

ERR004627: SDHC: Erroneous CMD CRC error and CMD Index error may occur on sending new CMD during data transfer

Description

When sending new, non data CMD during data transfer between the eSDHC and EMMC card, the module may return an erroneous CMD CRC error and CMD Index error. This occurs when the CMD response has arrived at the moment the FIFO clock is stopped. The following bits after the start bit of the response are wrongly interpreted as index, generating the CRC and Index errors.

The data transfer itself is not impacted.

The rate of occurrence of the issue is very small, as there is a need for the following combination of conditions to occur at the same cycle:

• The FIFO clock is stopped due to FIFO full or FIFO empty

• The CMD response start bit is received

Workaround

The recommendation is to not set FIFO watermark level to a too small value in order to reduce frequency of clock pauses.

The problem is identified by receiving the CMD CRC error and CMD Index error. Once this issue occurs, one can send the same CMD again until operation is successful.

ERR003984: SDHC: eSDHC misses SDIO interrupt when CINT is disabled

Description

An issue is identified when interfacing the SDIO card. There is a case where an SDIO interrupt from the card is not recognized by the hardware, resulting in a hang.

If the SDIO card lowers the DAT1 line (which indicates an interrupt) when the SDIO interrupt is disabled in the eSDHC registers (that is, CINTEN bits in IRQSTATEN and IRQSIGEN are set to zero), then, after the SDIO interrupt is enabled (by setting the CINTEN bits in IRQSTATEN and IRQSIGEN registers), the eSDHC does not sense that the DAT1 line is low. Therefore, it fails to set the CINT interrupt in IRQSTAT even if DAT1 is low.

Generally, CINTEN bit is disabled in interrupt service.

The SDIO interrupt service steps are as follows:

1. Clear CINTEN bit in IRQSTATEN and IRQSIGEN.

2. Reset the interrupt factors in the SDIO card and write 1 to clear the CINT interrupt in IRQSTAT.

3. Re-enable CINTEN bit in IRQSTATEN and IRQSIGEN.

If a new SDIO interrupt from the card occurs between step 2 and step 3, the eSDHC skips it.

Workaround

The workaround interrupt service steps are as follows:

1. Clear CINTEN bit in IRQSTATEN and IRQSIGEN.

2. Reset the interrupt factors in the SDIO card and write 1 to clear CINT interrupt in IRQSTAT.

3. Clear and then set D3CD bit in the PROCTL register. Clearing D3CD bit sets the reverse signal of DAT1 to low, even if DAT1 is low. After D3CD bit is re-enabled, the eSDHC can catch the posedge of the reversed DAT1 signal, if the DAT1 line is still low.

4. Re-enable CINTEN bit in IRQSTATEN and IRQSIGEN.

ERR003980: SDHC: Glitch is generated on card clock with software reset or clock divider change

Description

A glitch may occur on the SDHC card clock when the software sets the RSTA bit (software reset) in the system control register. A glitch can also occur when the clock divider value is set. Either glitch that is produced can cause the external card to switch to an unknown state. This occurrence is not deterministic.

Workaround

Disable the SD card clock before the software reset, then enable it when the module resumes its normal operation. The Host and the SD card are in a master-slave relationship. The Host provides clock and control transfer across the interface. Therefore, any existing operation is discarded when the Host controller is reset.

The recommended flow is as follows:

1. Software disable bit[3], SDCLKEN, of the System Control Register
2. Trigger software reset and/or set clock divider
3. Check bit[3], SDSTB, of the Present State Register for stable clock
4. Enable bit[3], SDCLKEN, of the System Control Register.

Using the above method, the eSDHC cannot send command or transfer data at the time a glitch in the clock line might occur, so this possible glitch cannot cause an issue.

ERR006934: SDHC: Issues with card removal/insertion detection

Description

SD cards include a 50K pullup that is used to detect when a card is inserted. A weak (500K) pulldown on the line is sufficient to allow for the first card insertion to be detected (the 50K pullup on the card will dominate the pulldown and cause the signal to go high).

The 5 V tolerant I/O cells used on this processor require a pulldown resistor stronger than 50K to ensure that a signal will reach a proper low voltage level after the pin has been high. A stronger pulldown cannot be used on the SD card detect signal because it would prevent the card detection from working correctly. Because of this, the automatic card removal detection will not work reliably.

Workaround

A software workaround can be used to check for card removal or verify that a card is still present by redetecting it.

1. Temporarily change the appropriate PORTx_PCRn to configure the SD card detect pin as a GPIO.

2. Configure the GPIO as an output driving low.

3. Leave the pin in this state for several us. The exact amount of time required can vary depending on pin loading.

4. Change the PORTx_PCRn to switch the pin back to its SDHC function.

When this sequence is complete the SDHC will be ready to detect card insertion again. The sequence above should be run periodically or if an unexpected response is received from the SD card. The procedure will discharge the I/O cell so that the card detect pin is low and an SD card (if present) can be detected.

ERR003983: SDHC: Problem when ADMA2 last descriptor is LINK or NOP

Description

ADMA2 mode in the eSDHC is used for transfers to/from the SD card. There are three types of ADMA2 descriptors: TRANS, LINK or NOP. The eSDHC has a problem when the last descriptor (which has the End bit '1') is a LINK descriptor or a NOP descriptor.

In this case, the eSDHC completes the transfers associated with this descriptor set, whereas it does not even start the transfers associated with the new data command. For example, if a WRITE transfer operation is performed on the card using ADMA2, and the last descriptor of the WRITE descriptor set is a LINK descriptor, then the WRITE is successfully finished. Now, if a READ transfer is programmed from the SD card using ADMA2, then this transfer does not go through.

Workaround

Software workaround is to always program TRANS descriptor as the last descriptor.

ERR003978: SDHC: Software can not clear DMA interrupt status bit after read operation

Description

After DMA read operation, if the SDHC System Clock is automatically gated off, the DINT status can not be cleared by software.

Workaround

Set HCKEN bit before starting DMA read operation, to disable SDHC System Clock auto-gating feature; after the DINT and TC bit received when read operation is done, clear HCKEN bit to re-enable the SDHC System Clock auto-gating feature.

ERR004218: 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.

ERR005952: 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.

ERR002638: 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.

ERR004546: TSI: The counter values reported from TSI increase when in low power modes (LLS, VLLS1, VLLS2, VLLS3)

Description

When the MCU goes into LLS or VLLSx modes, with the TSI enabled for wakeup, the counter value reported by the TSI increases with respect to what was reported in active mode. Because the wakeup threshold is calculated in active mode, it is highly likely that MCU will wakeup immediately after going to low power.

Workaround

1. Use Wait, Stop, or VLPS. These modes do not require any wakeup threshold calibration as TSI remains in active mode and wakes up each end of scan so that normal baseline tracking algorithm can be used.

2. To use LLS or VLLSx modes with the TSI as a wakeup source, calibrate the wakeup threshold using the desired low power mode. During application initialization, configure the TSI to exit low power via the LLWUI (low-leakage wake-up interrupt) with an End of Scan using the desired wakeup electrode. For example enter LLS mode with automatic scanning enabled so that after the first scan the TSI module causes an exit from low power at the end of scan. After the wakeup event, read the TSIx_WUCNTR Register, this register will have the value for the count during low power mode. Use this value to calculate THRSHLD register value.

ERR003926: 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.

ERR003927: TSI: TSI will scan continuously if only one electrode is enabled

Description

TSI will scan continuously if only one electrode is enabled even if a scan period using AMCLKS, AMPSC and SMOD has been defined.

Workaround

1. If the system has an unused TSI pin, enable that electrode as a dummy electrode and configure the scanning period to half of the desired period. This will give half of the period for the single used electrode and half for the dummy electrode.

2. If no extra TSI pin is available to use as a "dummy" electrode, trigger scans with software using SWTS bit and control timing with an additional timer (RTC, MTIM, etc.)

ERR004935: 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.

ERR007028: 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.

ERR007027: 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.

ERR006472: 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.

ERR004647: 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.

ERR007090: 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.

ERR007029: 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.

ERR007031: 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.

ERR003892: 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].

ERR004945: 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.

ERR005704: 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
```

ERR007091: 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.

ERR007092: 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.

ERR005928: 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.

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

Quick reference data — The Quick reference data is an extract of the product data given in the Limiting values and Characteristics sections of this document, and as such is not complete, exhaustive or legally binding.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Trademarks

KINETIS_K_5N22D

Mask Set Errata

Contents

Date of release: 2 September 2025
Document identifier: KINETIS_K_5N22D