Mask Set Errata for Mask 2N09P

This report applies to mask 2N09P for these products:

- MCIMX7S
- MCIMX7D

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
e3774	AIPS: Unaligned access causes abort on writes to the internal registers
e8305	ARM/MP: 804069-C, Exception mask bits are cleared when an exception is taken in Hyp mode.
e8304	ARM/MP: 805420-C, PMU event counter 0x14 does not increment correctly.
e8303	ARM/MP: 809719-C, PMU events 0x07, 0x0C, and 0x0E do not increment correctly.
e8302	ARM/MP: 814220-B, Cache maintenance by set/way operations can execute out of order.
e10133	ARM: Boot failure after A7 enters into low-power idle mode
e9513	CCM: Violate bit in misc register will be set unexpectedly when writing to debug or target interface.
e9515	CCM; Domain cannot wake up if clock source control is set to 0
e6939	Core: Interrupted loads to SP can cause erroneous behavior
e6940	Core: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used
e4574	CSU: Possibility of incorrect security privileges for memory accesses
e9516	DDR-PHY: Intermittent issue where DDR becomes unstable due to DDR MDLL unlocking.
e9606	ECSPI: In master mode, burst lengths of 32n+1 will transmit incorrect data
e9165	eCSPI: TXFIFO empty flag glitch can cause the current FIFO transfer to be sent twice
e5829	FlexCAN: FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment during the arbitration process.
e7805	I2C: When the I2C clock speed is configured for 400 kHz, the SCL low period violates the I2C spec of 1.3 uS min.
e9514	PCIe address space is not cacheable from A7.
e10728	PCIe: PLL may fail to lock under corner conditions
e9541	PXP: CSC2 does not perform RGB to YCbCr and RGB to YUV conversions correctly
e8244	PXP: pxp_compress step1 FIFO full may lead to data error
e8151	PXP: Rotation Engine alignment and operation combination limitations

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
e8153	PXP: Rotation1 Engine format support limitation
e8335	PXP: store engine is writing data when working in block mode and block size=16
e10787	SIM1/SIM2: On the following part numbers, SIM1 and SIM2 modules are inoperative. MCIMX7D3DVK10SC, MCIMX7D3EVK10SC, MCIMX7S3DVK08SC, MCIMX7S3EVK08SC, MCIMX7D2DVK12SC, MCIMX7D2DVM12SC
e10629	USDHC: EMMC HS200/SD SDR104 tuning process may fail based on delay cell number
e10574	Watchdog: A watchdog timeout or software trigger will not reset the SOC

Table 2. Revision History

Revision	Changes
0	No changes in this revision
0, 05/2016	No changes in this revision
1.0, 05/2017	The following errata were added. • e10728 • e10787 • e10574 • e10629

e3774: AIPS: Unaligned access causes abort on writes to the internal registers

Description: Unaligned access to AIPS can be driven high by SAHARA, DAP, and FEC. If they access the AIPS internal registers during an unaligned access, an ABORT occurs.

Workaround: Make only aligned accesses to the AIPS internal registers.

e8305: ARM/MP: 804069-C, Exception mask bits are cleared when an exception is taken in Hyp mode.

Description: The Cortex-A7 MPCore processor implements the ARM Virtualization Extensions and the ARM Security Extensions. Exceptions can be routed to Monitor mode by setting SCR.{EA, FIQ, IRQ} to 1. Exceptions can be masked by setting corresponding bit CPSR.{A, I, F} to 1.

The ARMv7-A architecture states that an exception taken in Hyp mode does not change the value of the mask bits for exceptions routed to Monitor mode. However, because of this erratum, the corresponding mask bits will be cleared to 0.

Workaround: There is no workaround for this erratum.

e8304: ARM/MP: 805420-C, PMU event counter 0x14 does not increment correctly.

Description: The Cortex-A7 MPCore processor implements version 2 of the Performance Monitor Unit architecture (PMUv2). The PMU can gather statistics on the operation of the processor and memory system during runtime. This event information can be used when debugging or profiling code. When a PMU counter is programmed to count L1 instruction cache accesses (event 0x14), the counter should increment on all L1 instruction cache accesses. Because of this erratum, the counter increments on cache hits but not on cache misses.

All configurations affected.

Workaround: To obtain a better approximation for the number of L1 instruction cache accesses, enable a second PMU counter and program it to count instruction fetches that cause linefills (event 0x01). Add the value returned by this counter to the value returned by the L1 instruction access counter (event 0x14). The result of the addition is a better indication of the number of L1 instruction cache accesses.

e8303: ARM/MP: 809719-C, PMU events 0x07, 0x0C, and 0x0E do not increment correctly.

Description: The Cortex-A7 MPCore processor implements version 2 of the Performance Monitor Unit architecture (PMUv2). The PMU can gather statistics on the operation of the processor and memory system during runtime. This event information can be used when debugging or profiling code.

The PMU can be programmed to count architecturally executed stores (event 0x07), software changes of the PC (event 0x0C), and procedure returns (event 0x0E). However, because of this erratum, these events do not fully adhere to the descriptions in the PMUv2 architecture.

All configurations affected.

Workaround: There is no workaround for this erratum.

e8302: ARM/MP: 814220-B, Cache maintenance by set/way operations can execute out of order.

Description: The v7 ARM ARM states that all cache and branch predictor maintenance operations that do not specify an address execute, relative to each other, in program order. However, because of this erratum, an L2 set/way cache maintenance operation can overtake an L1 set/way cache maintenance operation.

To be affected by this erratum, the processor must be implemented with an L2 cache.

Workaround: Correct ordering between set/way cache maintenance operations can be forced by executing a DSB before changing cache levels.

e10133: ARM: Boot failure after A7 enters into low-power idle mode

Description: Boot failure may happen after the A7 enters into low-power idle mode (no other blocks are powered down except the A7 CPU). When the system resumes from low-power idle, A7 CPU0 runs into exception while A7 CPU1 runs successfully. Such random boot failure happens when the wakeup INT arrives during A7 power down sequence.

Workaround: If both CPU0/CPU1 are IDLE, the last IDLE CPU should disable GIC first, then REG_BYPASS_COUNTER is used to mask wakeup INT, and then execute “wfi” is used to bring the system into power down processing safely. The counter must be enabled as close to the “wfi” state as possible. The following equation can be used to determine the RBC counter value: RBC_COUNT * (1/32K RTC frequency) >= (46 + PDNSCR_SW + PDNSCR_SW2ISO) * (1/IPG_CLK frequency).

e9513: CCM: Violate bit in misc register will be set unexpectedly when writing to debug or target interface.

Description: When writing to debug interface or target interface, normal violate bit in misc register will be set unexpectedly.

Violate bit is located at bit 8 of

0x30388010~0x3038801C,

0x30388110~0x3038811C,

0x30388210~0x3038821C,

0x30388310~0x3038831C,

...

0x3038BE10~0x3038BE1C.

Workaround: Clear this bit before accessing normal interface.

e9515: CCM; Domain cannot wake up if clock source control is set to 0

Description: If clock source control is set to 0 in a domain, the domain cannot wake up. For unused domains, the source setting can be set to all 0 to save power. For in-use domains, source controls can only be set to 1, 2, or 3, but if any are set to 0, the domain will not wake up.

Related registers are CCM_PLL_CTRLn.

Workaround: Ensure for all domains in use, the clock source control register is set to something other than 0.

e6939: Core: Interrupted loads to SP can cause erroneous behavior

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

This issue is more prevalent for user code written to manipulate the stack. Most compilers will not be affected by this, but please confirm this with your compiler vendor. MQX™ and FreeRTOS™ are not affected by this issue.

Affects: Cortex-M4, Cortex-M4F

Fault Type: Programmer Category B

Fault Status: Present in: r0p0, r0p1 Open.

If an interrupt occurs during the data-phase of a single word load to the stack-pointer (SP/R13), erroneous behavior can occur. In all cases, returning from the interrupt will result in the load instruction being executed an additional time. For all instructions performing an update to the base register, the base register will be erroneously updated on each execution, resulting in the stack-pointer being loaded from an incorrect memory location.

The affected instructions that can result in the load transaction being repeated are:

1) LDR SP,[Rn],#imm

2) LDR SP,[Rn,#imm]!

3) LDR SP,[Rn,#imm]

4) LDR SP,[Rn]

5) LDR SP,[Rn,Rm]

The affected instructions that can result in the stack-pointer being loaded from an incorrect memory address are:

1) LDR SP,[Rn],#imm

2) LDR SP,[Rn,#imm]!

Conditions:

1) An LDR is executed, with SP/R13 as the destination.

2) The address for the LDR is successfully issued to the memory system.

3) An interrupt is taken before the data has been returned and written to the stack-pointer.

Implications:

Unless the load is being performed to Device or Strongly-Ordered memory, there should be no implications from the repetition of the load. In the unlikely event that the load is being performed to Device or Strongly-Ordered memory, the repeated read can result in the final stack-pointer value being different than had only a single load been performed.

Interruption of the two write-back forms of the instruction can result in both the base register value and final stack-pointer value being incorrect. This can result in apparent stack corruption and subsequent unintended modification of memory.

Workaround: Most compilers are not affected by this, so a workaround is not required.

However, for hand-written assembly code to manipulate the stack, both issues may be worked around by replacing the direct load to the stack-pointer, with an intermediate load to a general-purpose register followed by a move to the stack-pointer.

If repeated reads are acceptable, then the base-update issue may be worked around by performing the stack pointer load without the base increment followed by a subsequent ADD or SUB instruction to perform the appropriate update to the base register.

e6940: Core: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used

Description: ARM Errata 709718: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used

Affects: Cortex-M4F

Fault Type: Programmer Category B

Fault Status: Present in: r0p0, r0p1 Open.

On Cortex-M4 with FPU, the VDIV and VSQRT instructions take 14 cycles to execute. When an interrupt is taken a VDIV or VSQRT instruction is not terminated, and completes its execution while the interrupt stacking occurs. If lazy context save of floating point state is enabled then the automatic stacking of the floating point context does not occur until a floating point instruction is executed inside the interrupt service routine.

Lazy context save is enabled by default. When it is enabled, the minimum time for the first instruction in the interrupt service routine to start executing is 12 cycles. In certain timing conditions, and if there is only one or two instructions inside the interrupt service routine, then the VDIV or VSQRT instruction might not write its result to the register bank or to the FPSCR.

Workaround: A workaround is only required if the floating point unit is present and enabled. A workaround is not required if the memory system inserts one or more wait states to every stack transaction.

There are two workarounds:

1) Disable lazy context save of floating point state by clearing LSPEN to 0 (bit 30 of the FPCCR at address 0xE000EF34).

2) Ensure that every interrupt service routine contains more than 2 instructions in addition to the exception return instruction.

e4574: CSU: Possibility of incorrect security privileges for memory accesses

Description: The CSU configuration signals are not synchronized to the OCRAM, OCRAM L2 cache, and WEIM so modifications of the security protection privileges to these memory locations during memory access may have unintended bus privileges.

Workaround: Program the CSU during boot prior to runtime memory access.

e9516: DDR-PHY: Intermittent issue where DDR becomes unstable due to DDR MDLL unlocking.

Description: Intermittent issue where DDR becomes unstable due to DDR MDLL unlocking.

Workaround: Set the ctrl_ref in register DDR_PHY_MDLL_CON0 [4:1] to 4’b1111 to disable the de-asserting of dfi_init_complete until rst_n is asserted.

e9606: ECSPI: In master mode, burst lengths of 32n+1 will transmit incorrect data

Description: When the ECSPI is configured in master mode and the burst length is configured to a value 32n+1 (where n=0,1, 2,…), the ECSPI will transmit the portions of the first word in the FIFO twice.

For example, if the transmit FIFO is loaded with:

[0] 0x00000001

[1] 0xAAAAAAAA

And the burst length is configured for 33 bits (ECSPIx_CONREG[BURST_LENGTH]=0x020), the ECSPI will transmit the first bit of word [0] followed by the entire word [0], then transmit the data as expected.

The transmitted sequence in this example will be:

[0] 0x00000001

[1] 0x00000001

[2] 0x00000000

[3] 0xAAAAAAAA

Workaround: Do not use burst lengths of 32n+1 (where n=0,1, 2,…).

e9165: eCSPI: TXFIFO empty flag glitch can cause the current FIFO transfer to be sent twice

Description: When using DMA to transfer data to the TXFIFO, if the data is written to the TXFIFO during an active eCSPI data exchange, this can cause a glitch in the TXFIFO empty signal, resulting in the TXFIFO read pointer (TXCNT) not updating correctly, which in turn results in the current transfer getting resent a second time.

Workaround: This errata is only seen when the SMC (Start Mode Control) bit is set. A modified SDMA script with TX_THRESHOLD = 0 and using only the XCH (SPI Exchange) bit to initiate transfers prevents this errata from occurring. There is an associated performance impact with this workaround. Testing transfers to a SPI-NOR flash showed approximately a 5% drop in write data rates and a 25% drop in read data rates.

e5829: FlexCAN: FlexCAN does not transmit a message that is enabled to be transmitted in 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 for the issue to occur:

• Only one message buffer is configured to be transmitted

• The write which enables the message buffer 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 the Idle state and no new message is received on the bus.

For example:

1. Message buffer 13 is deactivated on RxIntermission (write 0x0 to the CODE field from the Control/Status word) [First write to CODE]

2. Reconfigure the ID and data fields

3. Enable the message buffer 13 to be transmitted on BusIdle (write 0xC on CODE field) [Second write to CODE]

4. CAN bus keeps in Idle state

5. No write on the Control/Status from any message buffer happens.

During the second write to CODE (step 3), the write must happen one clock before the current message buffer 13 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 detected 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:

• Any message buffer (either Tx or Rx) is reconfigured (by writing to its CS field) just after the Intermission field.

• There are other configured message buffers to be transmitted

• 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 message buffer is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control/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 message buffer, 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/Status word to activate the message buffer.

6. The workaround consists of executing two extra steps:

7. Reserve the first valid mailbox as an inactive mailbox (CODE=0b1000). If RX FIFO is disabled, this mailbox must be message buffer 0. Otherwise, the first valid mailbox can be found using the “RX FIFO filters” table in the FlexCAN chapter of the chip reference manual.

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

NOTE

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

e7805: I2C: When the I2C clock speed is configured for 400 kHz, the SCL low period violates the I2C spec of 1.3 uS min.

Description: When the I2C module is programmed to operate at the maximum clock speed of 400 kHz (as defined by the I2C spec), the SCL clock low period violates the I2C spec of 1.3 uS min. The user needs to reduce the clock speed to get the SCL low time to meet the 1.3us I2C minimum required. This behavior means the SoC is not compliant to the I2C spec at 400kHz.

Workaround: In order to exactly meet the clock low period requirement at fast speed mode, SCL must be configured to 384KHz or less.

The following clock configuration meets the I2C specification requirements for SCL low for i.MX6 products:

I2C parent clock PERCLK_ROOT = 24M OSC

perclk_podf = 1

PERCLK_ROOT = 24M OSC/perclk_podf = 24MHz

I2C_IFDR = 0x2A

I2C clock frequency = 24MHz/64 = 375KHz

e9514: PCIe address space is not cacheable from A7.

Description: PCIe address doesn't support cacheable access from ARM.

When enable cach accesses PCIe axi address, ARM A7 issues wrap burst with 64 bytes un-aligned access (shift 0x10). PCIe axi slave takes it as incr burst (it doesn't support wrap on axi bridge), causing the return data and address not to match.

Workaround: None

e10728: PCIe: PLL may fail to lock under corner conditions

Description: Initial VCO oscillation may fail under corner conditions such as cold temperature which cause the PCIe PLL fail to lock in initialization phase.

Workaround: Use the following sequence to toggle the internal PLL_PD signal to make the VCO oscillate after the G_RST signal is de-asserted:

1. De-assert the G_RST signal by clearing SRC_PCIEPHY_RCR[PCIEPHY_G_RST].

2. Toggle internal PLL_PD signal by writing data "0x04", "0xA4", "0x04" to the register address 0x306D_0054 in sequence. The description of the affected register fields are:

Register Address: 0x306D_0054

Bit: 5

Field Name: PLL_PD

Description: Power-down of PLL (valid only when OVRD_PLL_PD = 1)

Register Address: 0x306D_0054

Bit: 7

Field Name: OVRD_PLL_PD

Description: Override enable for PLL power-down (0 – Override disabled; 1 – Override enabled

3. De-assert the CMN_RST signal by clearing register bit SRC_PCIEPHY_RCR[PCIEPHY_BTN].

4. Check whether PLL locks. If PLL remains in unlocked status, retry PLL_PD toggle until the PLL locks by using the following sequence. The maximum amount of retry is 10 attempts:

a. Assert CMN _RST signal by setting register bit SRC_PCIEPHY_RCR[PCIEPHY_BTN].

b. Toggle the internal PLL_PD signal by writing data “0x04”, “0xA4”, “0x04” to register address 0x306D_0054 in sequence.

c. De-assert the CMN_RST signal by clearing register bit SRC_PCIEPHY_RCR[PCIEPHY_BTN].

e9541: PXP: CSC2 does not perform RGB to YCbCr and RGB to YUV conversions correctly

Description: When performing RGB-to-YUV conversions, CSC2 can only output a result from 0 to 255 (8 bits), which does not meet the YUV range requirements (9 bits required) and the parameters d0/d1/d2 cannot meet the YCbCr requirements (17 bits required).

Workaround: Perform RGB-to-YUV or RGB-to-YCbCr conversions outside the PXP (either through a GPU, IPU, or CPU depending on the resources available on the SoC).

e8244: PXP: pxp_compress step1 FIFO full may lead to data error

Description: Due to a pxp compression bug, when pxp_compress step1 FIFO is full there is a Read source data channel error.

Workaround: Set OT=1 and check the flag output from: pxp_compress module, flag fifo_full, and error_prone.

e8151: PXP: Rotation Engine alignment and operation combination limitations

Description: Rotation Engine Position 0:

When processing ‘ps’ and ‘as’ buffers that are unaligned (buffers are not aligned to block boundaries) then a rotation operation that is combined with a flip, decimation, or scaling operation will not execute correctly. Rotation operations must be done in separate passes. These combination operations execute correctly for aligned buffers.

Rotation Engine Position 1:

Unaligned buffer rotation does not execute correctly. Rotation operations combined with flip, scaling, or decimation do not execute correctly. Only simple aligned rotation is supported.

Workaround: Rotation Engine Position 0:

When processing ‘ps’ and ‘as’ buffers that are unaligned (buffers are not aligned to block boundaries) then a rotation operation that is combined with a flip, decimation, or scaling operation will not execute correctly. Rotation operations must be done in separate passes. These combination operations execute correctly for aligned buffers.

Rotation Engine Position 1:

Unaligned buffer rotation does not execute correctly. Rotation operations combined with flip, scaling, or decimation do not execute correctly. Only simple aligned rotation is supported.

e8153: PXP: Rotation1 Engine format support limitation

Description: Rotation of data in YUV420 format does not work correctly.

Workaround: Do not use rotation of data in YUV420 format

e8335: PXP: store engine is writing data when working in block mode and block size=16

Description: The PXP store engine writes data when working in block mode and the block size=16.

As a result, stale data overwrites valid data and the valid data becomes corrupted.

Workaround: Restrict block size to size=8 when in block mode.

e10787: SIM1/SIM2: On the following part numbers, SIM1 and SIM2 modules are inoperative. MCIMX7D3DVK10SC, MCIMX7D3EVK10SC, MCIMX7S3DVK08SC, MCIMX7S3EVK08SC, MCIMX7D2DVK12SC, MCIMX7D2DVM12SC

Description: SIM1 and SIM2 Modules are non functional on certain part numbers.

The affected parts are:

MCIMX7D3DVK10SC

MCIMX7D3EVK10SC

MCIMX7S3DVK08SC

MCIMX7S3EVK08SC

MCIMX7D2DVK12SC

MCIMX7D2DVM12SC

Workaround: Use the following alternate part numbers where the SIM modules are functional:

For MCIMX7D3DVK10SC, use MCIMX7D7DVK10SC.

For MCIMX7D3EVK10SC, there is no industrial substitute available. Use MCIMX7D7DVK10SC for consumer qualification tier during development and use silicon revision 1.3 part for production, when it becomes available.

For MCIMX7S3DVK08SC, use MCIMX7S5EVK08SC.

For MCIMX7S3EVK08SC, use MCIMX7S5EVK08SC.

For part numbers MCIMX7D2DVK12SC and MCIMX7D2DVM12SC, there are no substitutions available. Please use rev D silicon when it becomes available. For development purposes and room temperature operation, you may use MCIMX7D7DVK10SC and MCIMX7D7DVM10SC, respectively.

e10629: USDHC: EMMC HS200/SD SDR104 tuning process may fail based on delay cell number

Description: In the eMMC HS200/SD SDR104 tuning process, the tuning process may fail if the first pass delay cell number specified is <10 during the tuning process.

If an internal clock is used as the tuning clock (recommended), then the delay cell number for first pass tuning command will be more than 10. In this case, there is no issue. If a feedback clock from the pad is used as a tuning clock (not recommended), then it is possible that the delay cell number for first pass tuning command will be less than 10. In this case, the tuning process will fail because the tuning state will judge the tuning window one cycle before the tuning window is calculated.

Workaround: If the tuning process fails because the first pass delay cell number specified is less than 10, the USDHC_TUNING_CTRL[TUNING_START_TAP] field should be set to 10.

e10574: Watchdog: A watchdog timeout or software trigger will not reset the SOC

Description: When the watchdog reset is asserted by software or a timeout, the chip reset sequence is started but does not complete.

Workaround: Option 1: Hardware implementation of power-on-reset (POR)

Use the pin muxing capability to route the desired WDOG_B signal to an external signal. That external signal must then be connected at the board level to an active-low power-on control (PWRON) of the PMIC. When WDOG_B is driven low (by a watchdog timeout or by software), this will cause the Power Management IC (PMIC) to cycle power causing a system-level power-on-reset.

Option 2: Use SRC_A7RCR0[A7_CORE_POR_RESET0] to reset the ARM A7.

This workaround works well for DDR3/DDR3L, but does work with LPDDR2. LPDDR2 access may fail after the ARM reboots because the LPDDR2 has no dedicated reset pin as with DDR3/DDR3L.

Option 3: Use the SNVS LPCR register to turn off the system power

Set the SNVS_LPCR[TOP] bit through software. Asserting this bit causes a signal to be sent to the PMIC to turn off the system power. This bit will clear after power is off. This bit is only valid when the Dumb PMIC mode is enabled (SNVS_LPCR[DP_EN]=1).

This option will work even if the WDOG_B is not connected to the PMIC power-on request (PWRON). This option will work with either DDR3 or LPDDR2 because the board-level power is maintained.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support