Mask Set Errata for Mask 0N14W

This report applies to mask 0N14W for these products:

- MIMX8MQ7DVAJZAA
- MIMX8MQ6DVAJZAA
- MIMX8MD7DVAJZAA
- MIMX8MD6DVAJZAA
- MIMX8MQ5DVAJZAA

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
e3774	AIPS: Unaligned access causes abort on writes to the internal registers
e11174	CA53: Cannot enter WAIT mode
e11171	CA53: Cannot support single-core runtime wakeup
e6939	Core: Interrupted loads to SP can cause erroneous behavior
e9004	Core: ITM can deadlock when global timestamping is enabled
e9005	Core: Store immediate overlapping exception return operation might vector to incorrect interrupt
e6940	Core: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used
e11167	CoreSight: No timestamp in trace data
e11039	DCSS: AYUV to YUV422 issue in Scaler
e10951	DCSS: CTX_LD does not support SB_COUNT.HP_COUNT = 0 if SB_COUNT.LP_COUNT != 0.
e11041	DCSS: DPR-DTRC integration limitation when DTRC is used for decompression/de-tiling
e11042	DCSS: DTRC requires 16 lines when enabled
e9535	ECSPI: Burst completion by SS signal in slave mode is not functional
e9165	eCSPI: TXFIFO empty flag glitch can cause the current FIFO transfer to be sent twice
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.
e10740	QuadSPI: Insufficient read data may be received in the RX Data Buffer register
e8565	QuadSPI: Low performance received when using certain access sequences
e11096	SAI: Internal bit clock is not generated when RCR2[BCI]=1 or TCR2[BCI]=1
e11150	SAI: Internally generated receive or transmit BCLK cannot be re-enabled if it is first disabled when RCR2[DIV] or TCR2[DIV] > 0

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
e11231	USB: Clock must remain on during suspend/resume
e11176	USB: USB 3.0 Host Hot Plug incorrectly enumerates
e11232	USDHC: uSDHC setting requirement for IPG_CLK and AHB_BUS clocks

Table 2. Revision History

Revision	Changes
0, 12/2017	Initial revision
0.1, 01/2018	The following errata were removed: • e10986 The following errata were revised: • e9535

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.

e11174: CA53: Cannot enter WAIT mode

Description: CA53 platform cannot enter WAIT mode if there is a pending interrupt. If the chip enters WAIT (WFI) mode, the chip requires a reboot to recover.

Workaround: No workarounds. SW should not use WAIT mode.

Impact: This mode turns off the power to the SCU (Snoop Control Unit) and the L2 cache. Not having this mode affects only 1 mode of core power savings.

e11171: CA53: Cannot support single-core runtime wakeup

Description: According to the GIC500 specification and the Arm Trusted Firmware design, when a CPU core enters the deepest CPU idle state (power-down), it must disable the GIC500 CPU interface and set the Redistributor register to indicate that this CPU is in sleep state. In such case, if the CPU core is in WFI or power-down with CPU interface disabled, another core cannot wake-up the powered-down core using SGI interrupt.

Workaround: One workaround is to use another A53 core for the IRQ0 which is controlled by the IOMUX GPR to generate an external interrupt to wake-up the powered-down core.

The SW workaround is implemented into default BSP release. The workaround commit tag is “MLK-16804-04 driver: irqchip: Add IPI SW workaround for imx8mq".

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.

e9004: Core: ITM can deadlock when global timestamping is enabled

Description: ARM ERRATA 806422

The Cortex-M4 processor contains an optional Instrumentation Trace Macrocell (ITM). This can be used to generate trace data under software control, and is also used with the Data Watchpoint and Trace (DWT) module which generates event driven trace. The processor supports global timestamping. This allows count values from a system-wide counter to be included in the trace stream.

When connected directly to a CoreSight funnel (or other component which holds ATREADY low in the idle state), the ITM will stop presenting trace data to the ATB bus after generating a timestamp packet. In this condition, the ITM_TCR.BUSY register will indicate BUSY.

Once this condition occurs, a reset of the Cortex-M4 is necessary before new trace data can be generated by the ITM.

Timestamp packets which require a 5 byte GTS1 packet, or a GTS2 packet do not trigger this erratum. This generally only applies to the first timestamp which is generated.

Devices which use the Cortex-M optimized TPIU (CoreSight ID register values 0x923 and 0x9A1) are not affected by this erratum.

Workaround: There is no software workaround for this erratum. If the device being used is susceptible to this erratum, you must not enable global timestamping.

e9005: Core: Store immediate overlapping exception return operation might vector to incorrect interrupt

Description: Arm Errata 838869: Store immediate overlapping exception return operation might vector to incorrect interrupt

Affects: Cortex-M4, Cortex-M4F

Fault Type: Programmer Category B Rare

Fault Status: Present in: r0p0, r0p1 Open.

The Cortex-M4 includes a write buffer that permits execution to continue while a store is waiting on the bus. Under specific timing conditions, during an exception return while this buffer is still in use by a store instruction, a late change in selection of the next interrupt to be taken might result in there being a mismatch between the interrupt acknowledged by the interrupt controller and the vector fetched by the processor.

Configurations Affected

This erratum only affects systems where writeable memory locations can exhibit more than one wait state.

Workaround: For software not using the memory protection unit, this erratum can be worked around by setting DISDEFWBUF in the Auxiliary Control Register.

In all other cases, the erratum can be avoided by ensuring a DSB occurs between the store and the BX instruction. For exception handlers written in C, this can be achieved by inserting the appropriate set of intrinsics or inline assembly just before the end of the interrupt function, for example:

ARMCC:

...

__schedule_barrier();

__asm{DSB};

__schedule_barrier();

}

GCC:

...

__asm volatile (“dsb 0xf” ::: “memory”);

}

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

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

Affects: Cortex-M4F

Fault Type: Programmer Category B

Fault Status: Present in: r0p0, r0p1 Open.

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

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

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

There are two workarounds:

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

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

e11167: CoreSight: No timestamp in trace data

Description: During the JTAG debug with trace function, there is no timestamp in the trace data. Because of the lack of a timestamp, the debug with trace function is not enabled by CoreSight.

Workaround: No workarounds, SoC does not support this feature.

e11039: DCSS: AYUV to YUV422 issue in Scaler

Description: Scaler uses 5 tap filtering for Graphics pipe. When the input data format in the graphics pipe is AYUV444 and dolby path is used, scaler is programmed to output YUV422 and src_is_video is set to 1. In this case, the module fails to replicate the last row when the vertical tap filter crosses the bottom boundary of the source picture. It results in incorrect pixels being output for the last 2 rows of the image.

Workaround: When the Input image format is AYUV444 and the Dolby Path is used, i.e. scaler converting AYUV444 to YUV422, program the src_is_video to 1, and set the Source Picture Height Register field to (input_image_height – 2).

No artifacts are found by this work around. The change is to be done only to the register programming, i.e. the input height to be used for determining the scaling coefficients should remain the same.

If the input image height is 256, and the Scaler is configured to convert AYUV444 to YUV422, source picture height register should be programmed to 254,

in all the other cases, it should be programmed to 255.

e10951: DCSS: CTX_LD does not support SB_COUNT.HP_COUNT = 0 if SB_COUNT.LP_COUNT != 0.

Description: The CTX_LD does not support single buffer high priority (SB.HP) count of 0 if the single buffer low priority (SB.LP) count is not 0. So, if any single buffered registers need to be programmed, the SB.HP count must be > 0.

Workaround: SW must always program at least one entry for the SB.HP region when there are entries in the SB.LP region. If the SB.LP region is 0, it is ok for both regions to be =0.

e11041: DCSS: DPR-DTRC integration limitation when DTRC is used for decompression/de-tiling

Description: The DTRC does not allow re-read of pixels. DTRC requires DPR to read exactly full scan line pixels when DTRC is used for decompression/de-tiling. The AXI interface on DTRC is a 128-bit which requires picture size x bitdepth/8 to be an integer multiple of 16. On further restriction, DPR requires X-WIDE to be integer multiple of 64bytes, which does not allow any picture size.

Workaround:
- DPR requires all transactions to be 64B, which forces 10b buffers to be constrained to (or padded to) a multiple of 256 pixels wide.
- There are a some hardware constraints on how the blocks access the buffers:

• There is no stride in G2/G1 compressed/uncompressed tiled output, so we cannot pad dummy data directly in those formats.

• G2/G1 raster scan output also does not support stride, so we cannot bypass DTRC to directly read raster scan output from G2 to solve the issue.

• DTRC raster scan output is consistent with the G2 raster scan output and, consequently, also does not support stride.

SW Work around:

• The DTRC crop feature can be used to simulate ‘padding’ in some instances. By setting the parameters to a negative crop value on the right edge, the DTRC will respond by padding with some limitations.

• This ‘padding’ will always work for 8-bit uncompressed G1/G2 output, and 10-bit uncompressed G2 output.

• This ‘padding’ will only work for 8-bit/10-bit compressed G2 output on cases that the padded picture width is <= (128 * ceiling(original_picture_width_in_pixels / 128))

Notes:

Almost all 8b compressed surfaces should work fine due to the ability to pad up to 128 pixel multiples. For 10b compressed surfaces that this does not work with (e.g. 1080p), NXP recommends that the reference frame compression be disabled on G2. The uncompressed tile format should be readable and possible to pad on the DTRC.

That said, NXP confirmed that only 4K formats should be requiring 10b (all of which should work compressed). This means that all primary use cases should be possible to support with compressed buffers.

Programming Model to use Padding feature (highlighted only registers which requires or have programming difference)

DTRC programming:

F0SIZE – The picture size still needs to be set to the original picture size.

F0CROPORIG – Cropped picture top left origin is set to (0, 0).

F0CROPSIZE – For 8bit crop picture width should be (orig picture width + 63) & 0xffff_ffC0; For 10bit crop picture width should be (orig picture width + 255) 0xffff_ff00 (make picture width 256 pixels aligned)

F0DCTL – Set bit 18 to 1’b, Set bit 17 to 0 for compressed source frame buffer and 1 for uncompressed source frame buffer.

DPR programming -

FRAME CTRL0[PITCH] = Cropped width (programmed in DTRC F0CROPSIZE reg) * bit_depth/8

FRAME_1P_PIX_X_CTRL[NUM_X_PIX_WIDE] = Cropped width (programmed in DTRC F0CROPSIZE reg) * bit_depth/8

FRAME_2P_PIX_X_CTRL[NUM_X_PIX_WIDE] = Cropped width (programmed in DTRC F0CROPSIZE reg) * bit_depth/8

Scaler shall be programmed with Original picture size.

e11042: DCSS: DTRC requires 16 lines when enabled

Description: On video channels, the DPR requires both the chroma and luma height to be multiples of 8. If not a multiple of 8, it can cause chroma height to be different than half the luma height.

Because there are not separate DTRC registers for the luma and chroma height and there is only one value, the chroma will always be exactly half the luma.

When the chroma height or luma height does not match between the DTRC and DPR, the DPR will request data that the DTRC doesn't have, therefore, the DTRC will not respond to the request. This will cause an incomplete transaction and stall the bus after enough transactions occur.

Example: For a 360 line image. DTRC = 360 lines, DPR Luma = 360, DPR chroma = 184 (it must be a multiple of 8). Because the DPR chroma and DTRC lines/2 are not the same, this issue will arise.

Workaround: Bypassing the DTRC will not cause this issue to arise. Padding the frame to be a multiple of 16 will work as well.

e9535: ECSPI: Burst completion by SS signal in slave mode is not functional

Description: According to the eCSPI specifications, when eCSPI is set to operate in the Slave mode (CHANNEL_MODE[x] = 0), the SS_CTL[x] bit controls the behavior of burst completion.

In the Slave mode, the SS_CTL bit should control the behavior of SPI burst completion as follows:

- 0—SPI burst completed when (BURST_LENGTH + 1) bits are received
- 1—SPI burst completed when the SS input is negated

Also, in BURST_LENGTH definition, it is stated "In the Slave mode, this field takes effect in SPI transfer only when SS_CTL is cleared."

However, the mode SS_CTL[x] = 1 is not functional in Slave mode. Currently, BURST_LENGTH always defines the burst length.

According to the SPI protocol, negation of SSB always causes completion of the burst. However, due to the above issue, the data is not sampled correctly in RxFIFO when {BURST_LENGTH+1}mod32 is not equal to {actual burst length}mod32.

Therefore, setting the BURST_LENGTH parameter to a value greater than the actual burst does not resolve the issue.

Workaround: Do not use the SS_CTL[x] = 1 option in the Slave mode. The accurate burst length should always be specified using the BURST_LENGTH parameter.

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.

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

e10740: QuadSPI: Insufficient read data may be received in the RX Data Buffer register

Description: Data read from flash through QuadSPI using Peripheral Bus Interface (IPS) may return insufficient data in the RX Buffer Data register (QuadSPI_RBDRn) when the read data size of a flash transaction is programmed to be greater than 32 bytes.

Workaround: For data size greater than 32 bytes, program the IP data transfer size in the IP configuration register (QuadSPI_IPCR[IDATSZ]) to be in multiples of 8 bytes.

e8565: QuadSPI: Low performance received when using certain access sequences

Description: When sequential accesses are made to QuadSPI with prefetch enabled and the memory difference between two subsequent access bursts is very small then the performance observed may vary with different flash frequency. As an example this may occur if a pre-fetch is requested when an ongoing burst would already fetch the required data.

The reduced performance can occur when using the DMA or cores and when performing execute-in-place in particular.

No data corruption is observed but effective performance can be reduced by more than half.

Workaround: The impact can be limited by reducing the number of overlapping QuadSPI requests through optimizing the size of the pre-fetch buffer but cannot be completely eliminated.

e11096: SAI: Internal bit clock is not generated when RCR2[BCI]=1 or TCR2[BCI]=1

Description: When the SAI transmitter or receiver is configured for internal bit clock with BCI = 1, the bit clock is not generated for either of the following two configurations:

a) SYNC = 00 and BCS = 0

b) SYNC = 01 and BCS = 1

Workaround: When the SAI transmitter or receiver is configured for internal bit clock with BCI=1, use only one of the following two configurations:

a) SYNC = 01 and BCS = 0

b) SYNC = 00 and BCS = 1

e11150: SAI: Internally generated receive or transmit BCLK cannot be re-enabled if it is first disabled when RCR2[DIV] or TCR2[DIV] > 0

Description: If the receive or transmit bit clock (BCLK) is internally generated, enabled with DIV > 0 and is then disabled, due to software or Stop mode entry, and the BCLK is enabled again, the clock is not generated.

Workaround: If the receive or transmit BCLK is internally generated and a DIV value greater than 0 is used, the SAI must be reset before the BCLK is re-enabled. This is achieved by writing the SR bit in the respective RCSR or TCSR register first to 1 and then immediately to 0.

e11231: USB: Clock must remain on during suspend/resume

Description: USB IP requires clock on during suspend/resume, to maintain RUN/STOP = 1.

This results in two impacts:

1. USB cannot be used as a wakeup source.

2. USB device mode will require extra power consumption as the clock is on.

Workaround: SW Workarounds:

Requires RUN/STOP bit to be set to 1 during suspend mode.

e11176: USB: USB 3.0 Host Hot Plug incorrectly enumerates

Description: The USB host hot plug causes the USB3 device to incorrectly enumerate.

Workaround: SW workarounds:

Two workarounds are possible to update the USB3.0 register settings within the Linux Driver:

Solution 1: Set GUSB2PHYCFG0.SUSPENDUSB20=0 when COMMONN=0

Solution 2: Set GUSB2PHYCFG.U2_FREECLK_EXISTS as 0

Solution 1 is the preferred workaround.

e11232: USDHC: uSDHC setting requirement for IPG_CLK and AHB_BUS clocks

Description: uSDHC AHB_BUS and IPG_CLK clocks must be synchronized.

Due to current physical design implementation, AHB_BUS and IPG_CLK must come from same clock source to maintain clock sync.

Workaround: Set AHB_BUS and IPG_CLK to clock source from PLL1.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: IMX8MDQLQ_0N14W
Rev. 0.1, 01/2018