MCXE317_0P09C

Mask Set Errata

Rev. 1.1 — 24 April 2025 Errata

1 Mask Set Errata for Mask 0P09C

1.1 Revision History

This report applies to mask 0P09C for these products:

- MCXE317MPA
- MCXE317MPB

Table 1. Revision History

Revision	Release Date	Significant Changes
1.1	4/2025	The following errata were added. • ERR052645 • ERR052558 • ERR052403 The following errata were revised. • ERR052438
1.0	1/2025	Initial Revision

1.2 Errata and Information Summary

Table 2. Errata and Information Summary

Erratum ID	Erratum Title
ERR011573	Cortex-M7: Speculative accesses might be performed to memory unmapped in MPU.
ERR050727	Core: Data corruption for load following Store-Exclusive.
ERR050729	Core: ECC error causes data corruption when the data cache error bank registers are locked.
ERR050763	PIT: RTI_LDVAL_STAT not reliable in Dynamic Loading mode
ERR050875	CoreSight: AHB-AP can issue transactions where HADDR[1:0] is not aligned to HSIZE on the AHB
ERR051040	ITCM/DTCM: On the TCM backdoor accesses, burst termination (via MRC) due to entering protected region within the burst leads to an erroneous update of the protected region accessed by the burst.
ERR051046	Core: CTI might generate interrupts even when DBGENCTRL[CDBGEN] is low.
ERR051061	PFLASH: Read-While-Write to the same block may return incorrect read data
ERR051114	PFLASH: PFCBLK0_LOCKMASTER_SS register provides incorrect status of the super sector program/erase lock bit domain ID owner.
ERR051127	PFLASH: Flash read during array integrity may return incorrect read data
ERR051588	LPSPI:Reset transmit FIFO after FIFO underrun by LPSPI Slave.
ERR051629	LPUART:Transmit Complete bit (STAT[TC]) is not set.

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

Erratum ID	Erratum Title
ERR052121	LPI2C: NACK Detect Flag can be set when IGNACK=1
ERR052226	SWT: Toggling watchdog enable may cause unexpected timeout in some boundary conditions
ERR052277	Cortex-M7: Can halt in an incorrect address when breakpoint and exception occurs simultaneously.
ERR052403	FlexCAN: CAN frame drops in Enhanced RX FIFO when message buffer (MB) is locked for more than 1 CAN frame time (33 us)
ERR052438	FlexCAN: CAN frame may drop when using Enhanced RX FIFO
ERR052460	Cortex-M7: A hang scenario can occur when a reserved read locked memory region is accessed by application cores
ERR052558	FlexCAN: Message buffer (MB) overrun status is cleared when reading Enhanced RX FIFO (ERF)
ERR052645	Flash: Incorrect read data may be returned from flash

2 Known Errata

ERR011573: Cortex-M7: Speculative accesses might be performed to memory unmapped in MPU.

Description

Arm errata 1013783-B

Fault Type: Programmer Cat B

Cortex-M7 can perform speculative memory accesses to Normal memory for various reasons. All other types of memory should never be subject to speculative accesses.

The memory attributes for a given address are defined by the settings of the MPU when it is enabled. Regions that are not mapped in the MPU do not have any explicit attributes and should not be subject to any speculative accesses.

Because of this erratum, Cortex-M7 can incorrectly perform speculative accesses to such unmapped regions.

Conditions:

To trigger this erratum, the data cache must be enabled and the MPU must be enabled with the default memory map disabled. That is:

• CCR.DC = 1; data cache is enabled.

• MPU_CTRL.ENABLE = 1; MPU is enabled.

• If MPU_CTRL.PRIVDEFNA = 1, then this erratum cannot occur from privileged mode.

• If MPU_CTRL.HFNMIENA = 1, then this erratum cannot occur from the NMI or HF handlers or exception handlers when FAULTMASK = 1.

In these situations, a PLD instruction targeting an unmapped region might result in an incorrect speculative access. The PLD instruction itself could be speculative because of branch prediction. Even a literal data value that corresponds to a PLD encoding could theoretically cause this issue. This makes it difficult to scan code to check if these conditions apply.

Therefore, Arm recommends that any software with the MPU and data cache configured as mentioned in the conditions above uses the workaround below.

Implications:

Processor execution is not directly affected by this erratum. The data returned from the speculative access is never used and if the access is inferred by the program, then an abort will be taken as required.

The only implications of this erratum are the access itself which should not have been performed. This might have an impact on memory regions with side-effects on reads or on memory which never returns a response on the bus.

Workaround

Instead of leaving memory unmapped, software should use MPU region 0 to cover all unmapped memory and make this region execute-never and inaccessible. That is, MPU_RASR0 should be programmed with:

• MPU_RASR0.ENABLE = 1; MPU region 0 enable.

• MPU_RASR0.SIZE = b11111; MPU region 0 size = 2^32 bytes to cover entire memory.

• MPU_RASR0.SRD = b00000000; All sub-regions enabled.

• MPU_RASR0.XN = 1; Execute-never to prevent instruction fetch.

• MPU_RASR0.AP = b000; No read or write access for any privilege level.

• MPU_RASR0.TEX = b000; Attributes = Strongly-ordered.

• MPU_RASR0.C = b0; Attributes = Strongly-ordered.

• MPU_RASR0.B = b0; Attributes = Strongly-ordered.

Note that the MPU supports addressing hitting in multiple regions with the highest numbered region taking priority.

Therefore, use of MPU region 0 in this way does not affect the existing organization and use of MPU regions.

ERR050727: Core: Data corruption for load following Store-Exclusive.

Description

ARM errata 1315869

Affects: Cortex-M7, Cortex-M7 with FPU Fault Type: Programmer Category C

Fault Status: Present in r0p1, r0p2, r1p0, r1p1 and r1p2. Open.

A load that follows a Store-Exclusive to the same address might forward data from an earlier store, situated between the Load-Exclusive and the Store-Exclusive, and not the data from the Store-Exclusive.

Conditions:

The following sequence is required for this erratum to occur:

1. A load exclusive sets the local monitor.

2. A store to the wanted address

3. Any of the following instructions to the wanted address. This instruction must not fail either the local or global monitor check.

• STREXB.

• STREXH.

• STREX.

4. A load to the wanted address.

There must be at most one instruction between the Store-Exclusive and the load. All accesses must be to Shareable memory.

Implications:

Data corruption occurs when the load returns data from the older store instead of the newer Store-Exclusive.

Stores between a Load-Exclusive and Store-Exclusive are not expected in real code because such stores can always clear the local monitor in some implementations.

This impacts Cortex-M7 and Cortex-M7 with FPU.

Configurations Affected

All configurations are affected.

Workaround

No workaround is necessary.

ERR050729: Core: ECC error causes data corruption when the data cache error bank registers are locked.

Description

ARM errata 1267980

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Category C

Fault Status: Present in r0p1, r0p2, r1p0, r1p1 and r1p2. Open.

The data cache contains two error bank registers, DEBR0 and DEBR1. These registers store the locations in the cache that Error Correcting Code (ECC) errors affect and prevent future allocations to those locations.

Software can lock each DEBR and this prevents the DEBR from being automatically updated when a data cache ECC error is detected.

Because of this erratum, if both DEBR0 and DEBR1 are locked and an ECC error is detected on a cacheable store, then the store data is written onto the bus but not written into the data cache. This might result in the data cache containing stale data.

Conditions:

• DEBR0 and DEBR1 are locked.

• The wanted address has been allocated to the cache.

• A cacheable store to the wanted address looks up in the cache, and an ECC error is found in the cache set that the store addresses.

Implications:

This erratum can cause data corruption in the data cache.

Configurations Affected

All configurations with a data cache and ECC are affected.

Workaround

Software must avoid locking both error bank registers.

ERR050763: PIT: RTI_LDVAL_STAT not reliable in Dynamic Loading mode

Description

RTI_LDVAL_STAT register give the RTI timer load synchronization status. In the case of RTI timer load, it will take several cycles until this value is synchronized into the RTI clock domain. This register gives the status of the new loaded in the RTI timer load register. However in case of Dynamic loading of RTI timer load, this value might not be reliable.

Workaround

There are two options:

1. Do not use Dynamic loading feature of RTI.

2. If user needs to use dynamic loading of RTI, so to ensure the correct loading of RTI timer load register, read the current timer load value (RTI_CVAL) register after the current timer expires.

ERR050875: CoreSight: AHB-AP can issue transactions where HADDR[1:0] is not aligned to HSIZE on the AHB

Description

ARM errata 1624041

This erratum affects the following components:

• AHB Access Port.

The ARM Debug Interface v5 Architecture Specification specifies a TAR (Transfer Address Register) in the MEM-AP that holds the memory address to be accessed.

TAR[1:0] is used to drive HADDR[1:0] when accesses are made using the Data Read/Write register DRW.

When the AHB-AP is programmed to perform a word or half-word sized transaction the AHB-AP does not force HADDR[1:0] to be aligned to the access size. This can result in illegal AHB transactions that are not correctly aligned according to HSIZE if HADDR[1:0] is programmed with an unaligned value.

Conditions:

1) TAR[1:0] programmed with a value that is not aligned with the size programmed in the CSW register of the AHB-AP.

2) An access is initiated by an access to the Data Read/Write Register (DRW) in the AHB-AP.

Implications:

As a result of the programming conditions listed above, AHB-AP erroneously initiates an access on the AHB with HADDR[1:0] not aligned to the size on HSIZE. This might initiate an illegal AHB access.

Workaround

TAR[1:0] must be b00 for word accesses, TAR[0] must be b0 for half-word accesses.

Software program should program TAR with an address value that is aligned to transaction size being made.

ERR051040: ITCM/DTCM: On the TCM backdoor accesses, burst termination (via MRC) due to entering protected region within the burst leads to an erroneous update of the protected region accessed by the burst.

Description

XRDC’s MRC is used to setup R/W protection to system memory including cores’ D-TCM and I-TCM through backdoor access. If core that doesn’t have access into the protected D-TCM/I-TCM region performs an access to that region, using a burst sequence with start address outside of the protected region and end address within the protected region, then D-TCM/I-TCM content within the protected region and overlaid by the burst transfer will be modified. The written values will be random.

The burst termination via the MRC is notified to the master via the bus error. However the un-intended region post the termination gets modified instead of being aborted.

Workaround

There are two possible workarounds:

1. The application should align the start and end addresses of the burst transfer within a region (protected or unprotected).

2. Disable the burst optimization (with impact to performance) by configuring IAHBCFGREG[TCM_DIS_WR_OPT] as 1 to avoid this issue.

ERR051046: Core: CTI might generate interrupts even when DBGENCTRL[CDBGEN] is low.

Description

ARM errata 585224

The Cortex-M7 integration level interrupt request outputs (CTIIRQ) are connected to the CTI outputs TRIGOUT[2:1]. These triggers should not be generated when the DBGENCTRL[CDBGEN] is low. This behavior should be guaranteed by tying off respective bits of the TODBGENSEL mask to 0. This mask has been tied off in the processor integration level to the incorrect value, which can result in the output being triggered regardless of the DBGENCTRL[CDBGEN] value.

Conditions:

• The CTIIRQ outputs are used by the system and enabled by respective bit in IRSPRCn (n refers to the irq number) (Refer to IRQ number from the CTI IRQ number provided in Interrupt map).

• Programming CTI such that it can generate triggers on these outputs (for more information on configuring the trigger outputs, see the ARM CoreSight™ SoC-400 Integration Manual).

• The processor interrupts are enabled and DBGENCTRL[CDBGEN] has to be driven low.

Implications:

Because of this erratum, debug events configured to trigger interrupts on the CTIIRQ output might generate them even if the DBGENCTRL[CDBGEN] is low.

Workaround

If the CTI interrupts are not used and debug is disabled, it is recommended to disable the respected CTI IRQ by software by disabling it in the appropriate bit in IRSPRCn for the respective core.

ERR051061: PFLASH: Read-While-Write to the same block may return incorrect read data

Description

While flash memory write (program or erase) is ongoing to a given block, other masters system can simultaneously perform a read from any other block. If Read-While-Write is performed to the same block, then system bus gets error response. However, when read occurs to the same block as being written, and this read occurs while there is previous system bus read outstanding, bus error response is not returned to the system. The scenario when this problem would occur is:

1. Master 1 initiates program/erase, considering this master won't initiate read to same block

2. Master 2 initiates read to same block while program/erase is in progress

3. Master 3 initiates read to same block while AHB read request from master 2 is accepted but pending at flash controller

Workaround

There are 2 workaround options:

1) The software can read the Read-While-Write Event Error bit in the Module Configuration Status register (MCRS[RWE]) after system bus reads to make sure there's no error. In case of error, discard system read data and perform reads again.

2) Avoid Read-While-Write to the same block at the first place. The software shall use synchronization when sharing flash block by multiple masters similar to sharing other hardware resources.

ERR051114: PFLASH: PFCBLK0_LOCKMASTER_SS register provides incorrect status of the super sector program/erase lock bit domain ID owner.

Description

PFCBLK0_SSETSLOCK register provides mechanism for the masters to gain ownership of the corresponding PFCBLK0_SSPELOCK register based on domain ID. If set lock register bit is set to 1, the lock bit is owned by the master with its domain ID. This domain ID value is stored in PFCBLK0_LOCKMASTER_SS register.

The PFCBLK0_LOCKMASTER_SS register incorrectly stores the value of the PFCBLK1_LOCKMASTER_SS register, thus indicating incorrect lock bit domain ID owner value of the corresponding PFCBLK0_SSPELOCK. This affects only the status register, locking functionality is not affected.

Workaround

There is no software workaround for this erratum. Master can find out if it owns the lock bit by toggling PFCBLK0_SSPELOCK register lock bit. If it owns lock bit then PFCBLK0_SSPELOCK register lock bit will be toggled.

ERR051127: PFLASH: Flash read during array integrity may return incorrect read data

Description

The Array integrity self-check can be performed to check the integrity of the embedded flash memory when it is in UTest mode. The Flash controller normally terminates AHB reads with an error response when array integrity self-check is in progress. However, the Flash controller doesn't send AHB error response for the following scenario:

1. Multiple AHB reads to flash are initiated on different AHB ports concurrently. This results in flash read from one of the AHB ports. The request is issued to flash but remains to wait for a response, while other AHB requests remain pending.

2. Meanwhile, a flash array integrity self-check is initiated by a core. This results in an incorrect AHB OK response for the first pending of the AHB requests and getting an incorrect read data from the embedded flash.

Workaround

An Array integrity self-check to the embedded flash memory must be initiated by code executed from SRAM. In a multicore environment, other cores' code should be moved to SRAM as well before initiating array integrity self-check.

ERR051588: LPSPI:Reset transmit FIFO after FIFO underrun by LPSPI Slave.

Description

Transmit FIFO pointers are corrupted when a transmit FIFO underrun occurs (SR[TEF]) in slave mode.

Workaround

When clearing the transmit error flag (SR[TEF] = 0b1) following a transmit FIFO underrun, reset the transmit FIFO (CR[RTF] = 0b1) before writing any new data to the transmit FIFO.

ERR051629: LPUART:Transmit Complete bit (STAT[TC]) is not set.

Description

When the CTS pin is negated and the CTS feature is enabled (MODIR[TXCTSE] = 0b1) and the TX FIFO is flushed by software then, the Transmit Complete (STAT[TC]) flag is not set.

Workaround

Clear (MODIR[TXCTSE]) bit and reset the transmit FIFO (FIFO[TXFLUSH] = 0b1) when flushing the FIFO with CTS enabled(MODIR[TXCTSE] = 0b1).

ERR052121: LPI2C: NACK Detect Flag can be set when IGNACK=1

Description

The NACK detect flag (MSR[NDF]) can be set even when the Controller Configuration 1 (MCFGR1[IGNACK]=0b1).

The LPI2C will not automatically generate a STOP or repeated START if the NACK detect flag (MSR[NDF]=0b1) and the ignore NACK are set (MCFGR1[IGNACK]=0b1). Thus, the transfer will continue as if the (MSR[NDF]) had not been set.

The LPI2C will continue to block a new START condition if (MSR[NDF]=0b1).

Workaround

When (MCFGR1[IGNACK]=0b1), the (MSR[NDF]) must be cleared by software, writing (MSR[NDF]=0b1) to allow new I2C transfers to start.

ERR052226: SWT: Toggling watchdog enable may cause unexpected timeout in some boundary conditions

Description

The Software Watchdog Timer (SWT) may timeout unexpectedly when loading a new timeout value. This can occur when the SWT is paused (CR[WEN]=0b0) to update the TO[WTO], while the counter is less than 0x14 (CO[CNT] = 0x14). When SWT is re-enabled (CR[WEN]=0b1), the SWT resumes the cycle count, but the counter is not updated (CO[CNT]) with the new timeout value before the cycle counter reaches zero.

Workaround

Before setting a new timeout value (TO[WTO]) the SWT must be updated with the watchdog keys ((SR[WSC]= 0xA602) and then (SR[WSC] = 0xB480)) to restart the counter value and have the timeout change being made within the appropriate time window, preventing the counter from reaching zero.

ERR052277: Cortex-M7: Can halt in an incorrect address when breakpoint and exception occurs simultaneously.

Description

Arm Errata 3092511

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Category C

When an asynchronous exception occurs at the same time as a breakpoint event (either hardware breakpoint or software breakpoint), it is possible for the processor to halt at the beginning of the exception handler instead of the instruction address pointed by the breakpoint.

Configurations Affected

This erratum affects all configurations of Cortex-M7.

When this happens:

- The BKPT bit in Debug Fault Status Register (DFSR) is set, indicating that a breakpoint event has occurred.
- The return address of the exception is the breakpoint address. As a result, if the debugger clears the halting control bit in the processor at this point, the processor will reach the breakpoint again after servicing the exception.

The correct behavior should be one of the followings:

1. Execute BKPT instruction and halt at BKPT before taking the asynchronous exception.

2. Take the asynchronous exception before BKPT and return to BKPT instruction and then halt on BKPT instruction.

In both cases, the debugger should see the processor halt on the BKPT instruction.

Workaround

This issue only affects the debugger's operation. The debugger could report the halting reason as an unknown breakpoint, and optionally resume operation. If the processor's operation is resumed, it is likely to be halted again immediately after the interrupt is serviced and returns to the breakpoint address.

ERR052403: FlexCAN: CAN frame drops in Enhanced RX FIFO when message buffer (MB) is locked for more than 1 CAN frame time (33 us)

Description

If Message Buffer (MB) and Enhanced RX FIFO both are configured for reception, and FlexCAN Message Buffer is locked for a long time (more than 1 CAN frame, 33us), FlexCAN receives some frames in FIFO and then start dropping the frames.

Workaround

There are two possible workarounds:

1) Core must read the message buffer within the 33 us after locking the MB.

2) Avoid using a few specific Message Buffers (MBs) as listed below:

ERR052438: FlexCAN: CAN frame may drop when using Enhanced RX FIFO

Description

An incoming CAN frame will be lost (i.e not latched into its expected Enhanced Rx FIFO data element), if both following two conditions are met simultaneously. There will be no indication that the frame was lost.

Conditions:

1. A write access is made to the message buffer Control and Status word (MB_CS) of a specific message buffer corresponding to the expected Enhanced Rx FIFO data element. Each Enhanced Rx FIFO data element corresponds to different message buffers impacted by this erratum and cannot be determined by software.

2. Depending on the timestamp configuration, the write access is made when receiving a frame at one certain Controller Host Interface (CHI) clock cycle either:

a. Around the time between the seventh bit of EOF and the second bit of IFS if timestamp is disabled (CTRL2[TSTAMPCAP] = 00b) or

b. Around the time between the fifth bit of EOF and seventh bit of EOF if timestamp is enabled (CTRL2[TSTAMPCAP] = 01b or 10b or 11b)

Workaround

To avoid the potential for dropped CAN frames, one of the following options may be implemented:

Workaround #1 : Disable Enhanced RX FIFO feature.

Workaround #2 : If the Enhanced RX FIFO feature is enabled, restrictions apply to certain Message Buffer (MB) numbers for both RX and TX. Either do not use these MB’s at all or at minimum, avoid updating the Control and Status word of these MBs when any reception to the Enhanced Rx FIFO could occur. This means, it would be safe to update the Control and Status word of these MBs when the FlexCAN is for example in Freeze mode or when it is ensured against frame reception from the CAN bus. Below are the MB numbers that have these restrictions:

ERR052460: Cortex-M7: A hang scenario can occur when a reserved read locked memory region is accessed by application cores

Description

Out of reset, by default, the Cortex-M7 core can perform speculative memory accesses to any region defined as Normal Device Type (Code, SRAM, RAM) and the memory attributes for a given address are defined by the settings of the device when the MPU is enabled.

A memory region in a reserved area is located between 0x1B10_0000 and 0x1B10_1FFF which lies within the Normal Code region (0x0000_0000 to 0x1FFF_FFFF) as defined by the default ARM memory map. This region is not intended to be accessed by users, but if accessed through a speculative or an explicit read, the region will not generate an error response and the core will enter into a hung state.

To reproduce the hang scenario, the user can perform an explicit read to this region. Additionally, the hang scenario can occur if the Cortex-M7 performs a speculative fetch to this region without user knowledge.

Workaround

To prevent a core hang scenario, read access to this region must be granted to the user by performing two back to back 32-bit writes to location 0x402A_C0F0. The data to be written firstly is 0x1CB0_499D followed by 0xB992_0D38. These writes should be performed as soon as possible out of reset so as to avoid a hang scenario due to a speculative read to this region. The data in this region is considered reserved and is subject to change.

ERR052558: FlexCAN: Message buffer (MB) overrun status is cleared when reading Enhanced RX FIFO (ERF)

Description

Message buffer status becomes "full" when a frame arrives, and status becomes "overrun" when a second message arrives in the same message buffer, if first message has still not been read. If frame reception is happening in ERF and the frame is being read from ERF, these reads could incorrectly clear the MB overrun status. As a result, the overrun event can be missed by the application.

Workaround

Use one of the following workarounds:

Workaround #1: Don’t use Enhanced RX FIFO (ERF).

Workaround #2: Don’t use any of the message buffers from MB0 to MB7 for reception if ERF is enabled. MB0 to MB7 can be used for transmission.

ERR052645: Flash: Incorrect read data may be returned from flash

Description

If flash prefetch is enabled when the Cortex-M7 caches are disabled, incorrect read data may be returned from the flash to the requesting Cortex-M7. The incorrect read may be returned in response to data or instruction fetches. This issue may be rarely encountered and requires a very specific data or instruction fetch sequence associated with a flash buffer miss followed by a scheduled prefetch and a new flash request to the page associated with the prefetch. When the issue does occur, incorrect data will be returned to the Cortex-M7 which will be from a valid flash location, but not from the address requested by the core and there is no direct indication that incorrect data has been returned.

Workaround

Avoid Cortex-M7 flash accesses when flash prefetching is enabled (PFCRn[Pn_mPFEN] = 1) and the Cortex-M7 instruction and data caches are disabled. Additionally, the code and data buffers can be controlled directly and independently through the PFCRn[Pn_mBFEN] bits. Therefore, if the data buffers or the code buffers have been enabled, the data or instruction caches must be enabled in all requesting cores accessing the flash.

Legal information

Definitions

Disclaimers

Trademarks

Contents

1 Mask Set Errata for Mask 0P09C 1
1.1 Revision History 1
1.2 Errata and Information Summary 1
2 Known Errata 3
Legal information 14

Document feedback

Date of release: 24 April 2025
Document identifier: MCXE317_0P09C