Document identifier: MCXNx4x_0P02G
Rev. 1.0, 1/2024

MCXNx4x_0P02G

Mask Set Errata

Mask Set Errata for Mask 0P02G

Revision History

This report applies to mask 0P02G for these products:

- MCXN947VDFT
- MCXN947VNLT
- MCXN946VDFT
- MCXN946VNLT
- MCXN546VDFT
- MCXN546VNLT
- MCXN547VDFT
- MCXN547VNLT

Table 1. Revision History

Revision	Date	Significant Changes
1.0	1/2024	The following errata were added. • ERR052149
0.4	1/2024	The following errata were added. • ERR052108 • ERR052088 • ERR051877 • ERR052122
0.3	9/2023	The following errata were added. • ERR051703 • ERR051162 • ERR051998 • ERR052001 • ERR051689 • ERR051704 • ERR051588 • ERR051684 • ERR052000 • ERR052002 • ERR051713 • ERR051705 • ERR051993

Table continues on the next page...

Table 1. Revision History (continued)

Revision	Date	Significant Changes
		The following errata were revised. • ERR051617 • ERR051204
0.2	4/2023	The following errata were added. • ERR051617 • ERR051629
0.1	2/2023	The following errata were revised. • ERR051421
0	1/2023	Initial Revision

Errata and Information Summary

Table 2. Errata and Information Summary

Erratum ID	Erratum Title
ERR051713	ADC: Extra conversion can occur when moving to low power mode
ERR051051	Core: A partially completed VLLDM might leave Secure floating-point data unprotected
ERR050505	Core: Access permission faults are prioritized over unaligned Device memory faults
ERR050501	Core: DFSR.EXTERNAL is not set correctly when waking up from sleep
ERR050502	Core: Execution priority might be wrong for one cycle after AIRCR is changed
ERR050500	Core: Group priority of a Non-secure interrupt might be incorrect when AIRCR.PRIS is set
ERR050503	Core: Non-secure HardFault exception might preempt when disabled by AIRCR.BFHFNMINS
ERR050504	Core: Sorting of pending interrupts might be wrong when high latency IRQs are pending
ERR050875	CoreSight: AHB-AP can issue transactions where HADDR[1:0] is not aligned to HSIZE on the AHB
ERR051704	DCDC: Failure changing to Low drive-strength mode
ERR051703	ENC: Compare interrupt generation persists when position counter equals to compare value
ERR051204	ENET: MAC Unable to Identify PTP SYNC and Follow_Up Messages with Peer Delay Reserved Multicast Address in the 802.1AS Mixed Mode Operation
ERR051993	FLASH: Flash fails to become ready during asynchronous interrupt event
ERR051877	GPIO: Port 5 interrupt may result in unpredictable behavior
ERR052122	I3C : Data size limitation in Message mode DDR transfer
ERR051617	I3C: In I2C compatibility mode read transaction not terminating correctly
ERR051162	I3C: Slave reset not supported when I3C is in slave mode

Table continues on the next page...

Table 2. Errata and Information Summary (continued)

Erratum ID	Erratum Title
ERR051588	LPSPI:Reset transmit FIFO after FIFO underrun by LPSPI Slave.
ERR051629	LPUART:Transmit Complete bit (STAT[TC]) is not set.
ERR051705	NPX: Error when reading REMAP register
ERR051374	PWM fault may work abnormally when the fault signal is very narrow
ERR051689	PWM: Stretch count prescaler does not work properly
ERR051998	ROM: Command "get-property 12" not supported when using USB interface
ERR052000	ROM: ECDSA P256 certificates not supported
ERR052149	ROM: IFR0 Recovery Boot not supported
ERR052108	ROM: LDO_SYS VDD level not returned to Normal voltage range after programming fuses
ERR052002	ROM: PKC RAM not erased after tamper event
ERR051684	ROM: TZ-M preset data missing registers
ERR052001	ROM: Unable to change ISP mode I2C address
ERR051421	SAI: Synchronous mode with bypass is not supported
ERR052088	SmartDMA: FlexIO_IRQ not correctly routed to SMARTDMAARCHB_INMUX
ERR051379	SRAM: Incorrect data reads when Auto-clock gating and ECC are enabled
ERR051410	TSI: TSICH bit field cannot be read correctly

Known Errata

ERR051713: ADC: Extra conversion can occur when moving to low power mode

Description

When high-priority trigger exceptions are enabled (ADCx->CFG[HPT_EXDI] = 0x1) and the ADC command uses the "Repeat until true" compare option (ADCx->CMDHa[CMPEN] = 0x3), an extra conversion occurs at the end of the conversion cycle if a higher priority trigger is asserted when a low power request is also made. This can result in erroneous extra data in the result FIFO and/or prevent the ADC module from being disabled in the low power mode (even if the Doze enable bit is set - ADCx->CTRL[DOZEN] = 0x1).

Workaround

The ADC workaround is to do ONE of the following:

- Disable the ADC before entering low power mode (ADCx->CTRL[ADCEN] = 0)

- Disable high priority exceptions (ADCx->CFG[HPT_EXDI] = 0x1)

- If high priority exceptions are enabled (ADCx->CFG[HPT_EXDI] = 0x1) and "Repeat until true" compare option is used (ADCx->CMDHa[CMPEN] = 0x3), then the trigger command select (ADCx->TCTRLa[TCMD]) pointing to that command must be the highest priority (ADCx->TCTRLa[TPRI] = 0).

- User software waits for final conversion to be completed before entering low power mode.

ERR051051: Core: A partially completed VLLDM might leave Secure floating-point data unprotected

Description

Arm errata 2219175

Affects: Cortex-M33

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r0p1, r0p2, r0p3, r0p4, r1p0. Open.

The VLLDM instruction allows Secure software to restore a floating-point context from memory. Due to this erratum, if this instruction is interrupted or it faults before it completes, then Secure data might be left unprotected in the floating point register file, including the FPSCR.

Configurations affected:

This erratum affects all configurations of the Cortex-M33 processor configured with the Armv8-M Security Extension and the Floating-point Extension.

Conditions:

This erratum occurs when all the following conditions are met:

• There is no active floating-point context, (CONTROL.FPCA==0)

• Secure lazy floating-point state preservation is not active, (FPCCR_S.LSPACT==0)

• The floating-point registers are treated as Secure (FPCCR_S.TS==1)

• Secure floating-point state needs to be restored, (CONTROL_S.SFPA == 1)

• Non-secure state is permitted to access to the floating-point registers, (NSACR.CP10 == 1)

• A VLLDM instruction has loaded at least one register from memory and does not complete due to an interrupt or fault

Implications:

If the floating-point registers contain Secure data, a VLSTM instruction is usually executed before calling a Non-secure function to protect the Secure data. This might cause the data to be transferred to memory (either directly by the VLSTM or indirectly by the triggering of a subsequent lazy state preservation operation). If the data has been transferred to memory, it is restored using VLLDM on return to Secure state. If the VLLDM is interrupted or it faults before it completes and enters a Non-secure handler, the partial register state which has been loaded will be accessible to Non-secure state.

Workaround

To avoid this erratum, software can ensure a floating-point context is active before executing the VLLDM instruction by performing the following sequence:

• Read CONTROL_S.SFPA

• If CONTROL_S.SFPA==1 then execute an instruction which has no functional effect apart from causing context creation (such as VMOV S0, S0)

ERR050505: Core: Access permission faults are prioritized over unaligned Device memory faults

Description

Cortex-M33 1080541-C :

A load or store which causes an unaligned access to Device memory will result in an UNALIGNED UsageFault exception. However, if the region is not accessible because of the MPU access permissions (as specified in MPU_RBAR.AP), then the resulting MemManage fault will be prioritized over the UsageFault.

Workaround

There is no workaround.

However, it is expected that no existing software is relying on this behavior since it was permitted in Armv7-M.

ERR050501: Core: DFSR.EXTERNAL is not set correctly when waking up from sleep

Description

Cortex-M33 1367266-C:

An external debug event which causes the processor to enter Debug state or the debug monitor should set DFSR.EXTERNAL. It has been found that this field is not set if the event occurs while the processor is asleep.

Workaround

There is no workaround.

ERR050502: Core: Execution priority might be wrong for one cycle after AIRCR is changed

Description

Cortex-M33 1435973-C:

AIRCR is used in the NVIC active tree to calculate the execution priority, which in turn is used to determine fault escalation, exception preemption, and other NVIC-related behaviors. When the active tree is pipelined and there are high latency IRQs active, there might be a glitch in the active tree output for one cycle after AIRCR is changed. The glitch results in NVIC producing wrong execution priority that is neither based on the old AIRCR value nor the new one.

Workaround

There is no workaround for this erratum.

ERR050500: Core: Group priority of a Non-secure interrupt might be incorrect when AIRCR.PRIS is set

Description

Cortex-M33 1113997-C:

When the processor is configured with Security extension and AIRCR.PRIS is 1, the Armv8-M architecture requires that the priorities of Non-secure interrupts are modified to ensure that Secure interrupts are prioritized over Non-secure interrupts. The Armv8-M architecture requires that lower priority numbers take precedence over higher priority numbers. Because of this erratum, a Non-secure interrupt with higher priority number might be handled in the wrong order compared to another Non-secure or Secure interrupt.

Workaround

There is no workaround for this erratum.

ERR050503: Core: Non-secure HardFault exception might preempt when disabled by AIRCR.BFHFNMINS

Description

Cortex-M33 1453380-C:

When the processor implements the Security Extension and AIRCR.BFHFNMINS is 1, the Non-secure banked version of SHCSR.HARDFAULTPENDED can be set to 1. This Non-secure pended HardFault might not preempt per architecture because it does not have enough priority (that is, the processor is in HardFault handler mode). If AIRCR.BFHFNMINS is subsequently changed to 0 with the Non-secure HardFault still pending, then the architecture requires that the Nonsecure HardFault should never preempt regardless of execution priority. Because of this erratum, the pended Non-secure HardFault exception preempts when AIRCR.BFHFNMINS is 0 and current execution priority is larger than -1 (Non-secure HardFault having higher priority).

Workaround

There is no workaround for this erratum.

ERR050504: Core: Sorting of pending interrupts might be wrong when high latency IRQs are pending

Description

Cortex-M33 1540599-C:

The NVIC contains a pending tree which sorts all pending and enabled interrupts based on priorities. If DHCSR.C_DEBUGEN and DHCSR.C_MASKINTS are 1, DHCSR.S_SDE is 0 and halting debug is allowed, then Nonsecure PendSV, Non-secure SysTick, and Non-secure IRQs should be masked off and they should not affect the sorting of pending and enabled secure interrupts. If multiple high latency IRQs are pending and enabled with different security targets and priorities, then Non-secure IRQs which should be masked off might cause the pending tree output to be a pending Secure nterrupt without highest priority. This is because of incorrect masking before doing priority comparisons in the tree.

Workaround

There is no workaround for this erratum.

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.

ERR051704: DCDC: Failure changing to Low drive-strength mode

Description

The DCDC output may fail when transitioning from Normal to Low drive-strength, resulting in the DCDC output voltage dropping to the point it is not able to adequately power the VDD_CORE supply, or causes temporary brown-out conditions. This failure may occur when both of these conditions occur:

1) The transition from Normal drive strength (DCDC_VDD_DS = 10b) to Low drive-strength (DCDC_VDD_DS = 01b) occurs when the DCDC is actively switching the output.

2) The voltage level set in the bitfield SPC->LP_CFG[DCDC_VDD_LVL] is greater than or equal to the current output voltage of the DCDC.

Because this failure requires a specific timing to manifest, it may fail very infrequently in an application. The greater the load current of the DCDC, the more likely the failure will occur because the DCDC will spend more time in the active switching period. A higher rate of transitioning to Low drive-strength will also see a higher failure rate.

There are two scenarios when the DCDC drive-strength can transition from Normal to Low drive-strength, and this failure may occur:

1) While the MCU is in Active power mode, and the application changes the drive-strength setting by writing 01b to the bitfield SPC->ACTIVE_CFG[DCDC_VDD_DS]. Writing this bitfield will start the transition to Low drive-strength.

2) When the MCU enters a low-power mode (Deep Sleep, Power Down, or Deep Power Down), and Active mode uses Normal drive-strength with ACTIVE_CFG[DCDC_VDD_DS] = 10b, while the low-power mode uses Low drive-strength with LP_CFG[DCDC_VDD_DS] = 01b.

Workaround

This issue will always be avoided when the voltage level at the low-power low drive-strength is lower than the current output voltage of the DCDC. Before transitioning to Low drive-strength, ensure the voltage level in LP_CFG[DCDC_VDD_LVL] is lower than the voltage level in Normal drive-strength configured by ACTIVE_CFG[DCDC_VDD_LVL]. As part of this workaround, the voltage level used in Low drive-strength configured by LP_CFG[DCDC_VDD_LVL] must not be set to the maximum value 11b for 1.2 V at any time in an application.

If the desired voltage level in LP_CFG[DCDC_VDD_LVL] is the same as the level currently set in ACTIVE_CFG[DCDC_VDD_LVL], a workaround is to temporarily increase the voltage level in ACTIVE_CFG[DCDC_VDD_LVL], and then transition to Low drive-strength with the lower level in LP_CFG[DCDC_VDD_LVL]. Here is the sequence for this workaround:

1) Ensure LP_CFG is configured for Low drive-strength and the desired voltage level in Low drive-strength mode

2) Wait for the SPC bit SC[BUSY] to be clear.

3) Write ACTIVE_CFG[DCDC_VDD_LVL] with the value for the voltage level one step higher than the desired level in LP_CFG[DCDC_VDD_LVL].

4) Start the transition to Low drive-strength

If the workaround sequence above is used when the MCU enters a low-power mode, then when the MCU wakes the DCDC will return to Normal drive-strength with the output voltage level configured in SPC->ACTIVE_CFG[DCDC_VDD_LVL]. If a lower voltage level is preferred, the application can lower DCDC voltage by waiting for the bit SC[BUSY] to be clear and writing the new voltage level to SPC->ACTIVE_CFG[DCDC_VDD_LVL].

ERR051703: ENC: Compare interrupt generation persists when position counter equals to compare value

Description

When CTRL[CMPIE] is set and the position counter (LPOS and UPOS) matches COMP compare registers (LCOMP and UCOMP), the corresponding compare interrupt is constantly generated as long as the QDC counter value is equal to COMP, even if SW clears the interrupt flags.

Workaround

Keep CTRL[CMPIE] cleared and route the POS_MATCH signal using INPUTMUX to another module to either post-process the signal or to trigger a different interrupt. When the position counter equals COMP, POS_MATCH is asserted. You can use INPUTMUX to send the POS_MATCH trigger to a number of trigger inputs, the most typical use would be to route to CTIMER or SCTIMER to trigger a timer measurement which can be used to measure the time between POS_MATCH pulses. Alternatively, these same timers can be configured to interrupt immediately after 1 count, giving an interrupt 1 timer count after compare register matches.

ERR051204: ENET: MAC Unable to Identify PTP SYNC and Follow_Up Messages with Peer Delay Reserved Multicast Address in the 802.1AS Mixed Mode Operation

Description

This defect occurs only when the Ethernet MAC is configured for IEEE 802.1AS mixed mode. That is, when:

MAC_TIMESTAMP_CONTROL[AV8021ASMEN] = 1'b1

and

MAC_TIMESTAMP_CONTROL[SNAPTYPSEL] = 2'b01 and MAC_TIMESTAMP_CONTROL[TSEVNTENA] = 1'b0.

or

MAC_TIMESTAMP_CONTROL[SNAPTYPSEL]= 2'b01, MAC_TIMESTAMP_CONTROL[TSMSTRENA] = 1'b0, and MAC_TIMESTAMP_CONTROL[TSEVNTENA]= 1'b1.

the Ethernet MAC is unable to capture the ingress timestamp for PTP SYNC and Follow_Up messages that are received with PTP Peer Delay Reserved multicast destination address. The slave node is unable to compute and perform the time correction, and this results in inaccuracies in the maintained system time.

Workaround

The IEEE 802.1AS mixed mode is not a general use case. The time correction can be performed by using either Delay Request-Response or Peer Delay mechanism. However, if mixed mode is required the application must program the MAC_TIMESTAMP_CONTROL[TSENALL] = 1'b1, to enable the MAC to capture the timestamp for all the received packets. The software must identify the PTP SYNC and Follow_Up messages and associate the timestamp status provided by the MAC.

ERR051993: FLASH: Flash fails to become ready during asynchronous interrupt event

Description

The flash can fail to become ready on an asynchronous interrupt event resulting in the SOC stalling and the CPU unable to continue code execution.

This condition occurs when the Flash Doze bit is disabled (CMC0->FLASHCR[FLASHDOZE] = 0) and an asynchronous interrupt event occurs when the flash is attempting to move to low power mode due to a WFI / WFE instruction execution.

Workaround

This issue has one workaround:

1) When moving to low power mode, ensure that the Flash Doze bit is set (CMC0->FLASHCR[FLASHDOZE] = 1).

Note that in implementing this workaround, bus masters that can operate during low power modes (such as the DMA engines) will not be able to access the flash.

ERR051877: GPIO: Port 5 interrupt may result in unpredictable behavior

Description

Unpredictable behavior could result when using Port 5 interrupts.

Workaround

There is no workaround for this issue. Port 5 interrupts should not be used.

ERR052122: I3C : Data size limitation in Message mode DDR transfer

Description

The message length in DDR message(DMA) mode is defined in MWMSG_DDR_CONTROL2 [9:0].LEN field. The 2 MSBs [9:8] of this field has no effect on operation. Only [7:0] part of this field is taken as length in number of Half words. So, for max value of 3FF h in this field is taken as FF h (255) by hardware , consequently the maximum amount of actual data gets limited as per the operation type. For Read operation the actual data size will be (255 - 2) = 253 half-words (506 bytes) and for write operation it is (255 -1) = 254 halfword (508 bytes)

Workaround

Application need to limit the data size for Write and Read operation in message(DMA) mode of DDR transfer. Configure MWMSG_DDR_CONTROL2 [9:0].LEN = FF h, For Read frame data receive size will be 506 bytes and for Write frame data transmit size will be 508 bytes for a single frame.

ERR051617: I3C: In I2C compatibility mode read transaction not terminating correctly

Description

The I3C module can operate in I2C compatibility mode to support I2C devices. However when operating in this mode, the end of any read transaction may terminate with a repeated START followed by the STOP instead of only a STOP.

Workaround

In I2C compatibility mode, the use of no skew should be avoided and must set to MCONFIG[SKEW] = 1.

ERR051162: I3C: Slave reset not supported when I3C is in slave mode

Description

When operating in slave mode, the I3C module is unable to reliably detect a slave reset from an external master. As a result this feature is not supported on this device and should be disabled in application software.

Workaround

When I3C is operating in slave mode, the application must mask the slave reset interrupt by writing '1' to the SLVRST field in the I3C Slave Interrupt Clear (SINTCLR) register.

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).

ERR051705: NPX: Error when reading REMAP register

Description

Reading of the LIM data field (NPX0->REMAP[LIM]) returns incorrect results. Instead of returning the LIM field value, the LIM_DP field value is returned. Writes to the LIM data field are not affected.

Workaround

There is no workaround to this issue. Customer software should write the NPX0->REMAP[LIM] field and assume this write occurred correctly.

ERR051374: PWM fault may work abnormally when the fault signal is very narrow

Description

If the fault signal pulse width is narrower than a certain threshold, the protected PWM channels may generate a glitch, which occurs after the PWM channel outputs become inactive.

Workaround

(1) When FCTRL2[NOCOMB] = 0, FFILT [GSTR]= 0, and FFILT[FILT_PER]=0, pulse width of fault signals must be larger than 6 PWM clock periods, otherwise a glitch may be generated on the protected PWM channels.

(2) When FCTRL2[NOCOMB] = 0, FFILT [GSTR]= 1, and FFILT[FILT_PER]=0, pulse width of fault signals must be larger than 3 PWM clock periods, otherwise a glitch may be generated on the protected PWM channels.

(3) When FCTRL2[NOCOMB] = 0, FFILT [GSTR]= 1, and FFILT[FILT_PER] has non-zero values, pulse width of fault signals must be larger than FILT_PER*(FILT_CNT+3)+6 PWM clock periods, otherwise a glitch may be generated on the protected PWM channels.

(4) When FCTRL2[NOCOMB] = 0, FFILT [GSTR]= 0, and FFILT[FILT_PER] has non-zero values, pulse width of fault signals must be larger than FILT_PER*(FILT_CNT+3)+9 PWM clock periods, otherwise a glitch may be generated on the protected PWM channels.

ERR051689: PWM: Stretch count prescaler does not work properly

Description

PWM MCTRL2[STRETCH_CNT_PRSC] register bit field is intended to stretch the trigger pulse width to allow slower speed peripherals to capture the trigger. Due to this defect, however, this bit field is ineffective and output triggers are only able to be one clock width wide. This prevents the following peripherals from capturing PWM triggers:

- SCTIMER

- CTIMER

- CMP

- FlexIO

- SINC

Workaround

There is one workaround for this defect. The EVTG module can be used to stretch the PWM trigger pulse. To do this,

- Connect PWMa_SMb_MUX_TRIG0 to EVTG_AOI0_1 (INPUTMUX0[EVTG_TRIGx] = 0byy_yyy0, where x is the EVTG AOI input desired and 0byy_yyy0 is the Trig0 connection of the corresponding PWM instance and sub-module).

- Connect PWMa_SMb_MUX_TRIG1 to EVTG_AOI0_0 (INPUTMUX0[EVTG_TRIGx] = 0byy_yyy1, where x is the EVTG AOI input desired and 0byy_yyy1 is the Trig1 connection of the corresponding PWM instance and sub-module).

- Configure EVTG_OUT0A to the peripheral to be triggered in the INPUTMUX registers.

- Configure the EVTGx AOI to RS trigger mode (EVTGx[CTRL] = 0x4).

- Configure the EVTGx AOI_0 to pass the PWMa_SMb_MUX_TRIG1 signal directly. Configure all other signals to “Input Logic One”.

- Configure the EVTGx AOI_1 engine to pass the PWMa_SMb_MUX_TRIG0 signal directly. Configure all other signals to “Input Logic One”.

- Configure the PWMa[SMbTCTRL]->OUT_TRIG_EN bit field to route the TRIG0 and TRIG1 outputs to the desired VALz registers.

ERR051998: ROM: Command "get-property 12" not supported when using USB interface

Description

When using the USB interface to access the device in ISP mode, command "get-property 12" returns a fail result. This applies to both Full-Speed and High-Speed USB interfaces.

Workaround

There is no workaround for this issue. Customers should not use the "get-property 12" command when using USB as the ISP mode interface.

ERR052000: ROM: ECDSA P256 certificates not supported

Description

Authentication by ECDSA P256 certificates is not supported.

Workaround

Use ECDSA P384 authentication instead of ECDSA P256.

ERR052149: ROM: IFR0 Recovery Boot not supported

Description

The IFR0 Recovery Boot does not function.

Workaround

No workaround to this issue exists. Customers requiring this function should use rev A1 devices (mask number 1P02G).

ERR052108: ROM: LDO_SYS VDD level not returned to Normal voltage range after programming fuses

Description

When programming any fuse using the ROM API, the voltage level of the LDO_SYS is not returned to Normal Voltage level (1.8V). That is, SPC0->ACTIVE_CFG[SYSLDO_VDD_LVL] = 1.

Workaround

User software should return the LDO_SYS voltage level to normal level immediately after programming fuses (SPC0->ACTIVE_CFG &= SPC_ACTIVE_CFG_SYSLDO_VDD_LVL_MASK;).

Note that the SDK functions which program fuses already account for this errata.

ERR052002: ROM: PKC RAM not erased after tamper event

Description

When a tamper event occurs and the device is configured to erase the PKC RAM, the PKC RAM will not be erased.

Workaround

On a RAM_ZEROIZE ITRC event, the PKC RAM should be erased by the application. The ROM will erase RAMA.

ERR051684: ROM: TZ-M preset data missing registers

Description

The TrustZone preset data uses an earlier version with some changes from the documented struct in the Security Reference Manual.

Workaround

The correct TrustZone preset struct definition for this silicon revision is:

typedef struct **<b>tzmsecureconfig**

{

uint32_t tzm_magic; /*!< It contains four letters "TZ-M" to identify start of block*/

uint32_t cm33_vtor_addr; /*!< CM33 Secure vector table address */

uint32_t cm33_vtor_ns_addr; /*!< CM33 Non-secure vector table address */

uint32_t cm33_nvic_itns0; /*!< CM33 Interrupt target non-secure register 0 */

uint32_t cm33_nvic_itns1; /*!< CM33 Interrupt target non-secure register 1 */

uint32_t cm33_nvic_itns2; /*!< CM33 Interrupt target non-secure register 2 */

uint32_t cm33_nvic_itns3; /*!< CM33 Interrupt target non-secure register 3 */

uint32_t cm33_nvic_itns4; /*!< CM33 Interrupt target non-secure register 4 */

uint32_t cm33_misc_ctrl; /*!< Miscellaneous CM33 settings:

AIRCR.SYSRESETREQ

AIRCR.BFHFNMINS

AIRCR.PRIS

SCR.SLEEPDEEPS

SHCSR.SECUREFAULTENA */

uint32_t cm33_nsacr; /*!< CM33 Non-secure Access Control Register */

uint32_t cm33_cppwr; /*!< CM33 Coprocessor Power Control Register */

uint32_t cm33_cpacr; /*!< CM33 Coprocessor Access Control Register */

uint32_t cm33_mpu_ctrl; /*!< MPU Control Register.*/

uint32_t cm33_mpu_mair0; /*!< MPU Memory Attribute Indirection Register 0 */

uint32_t cm33_mpu_mair1; /*!< MPU Memory Attribute Indirection Register 1 */

uint32_t cm33_mpu_rbar0; /*!< MPU Region 0 Base Address Register */

uint32_t cm33_mpu_rlar0; /*!< MPU Region 0 Limit Address Register */

uint32_t cm33_mpu_rbar1; /*!< MPU Region 1 Base Address Register */

uint32_t cm33_mpu_rlar1; /*!< MPU Region 1 Limit Address Register */

uint32_t cm33_mpu_rbar2; /*!< MPU Region 2 Base Address Register */

uint32_t cm33_mpu_rlar2; /*!< MPU Region 2 Limit Address Register */

uint32_t cm33_mpu_rbar3; /*!< MPU Region 3 Base Address Register */

uint32_t cm33_mpu_rlar3; /*!< MPU Region 3 Limit Address Register */

uint32_t cm33_mpu_rbar4; /*!< MPU Region 4 Base Address Register */

uint32_t cm33_mpu_rlar4; /*!< MPU Region 4 Limit Address Register */

uint32_t cm33_mpu_rbar5; /*!< MPU Region 5 Base Address Register */

uint32_t cm33_mpu_rlar5; /*!< MPU Region 5 Limit Address Register */

uint32_t cm33_mpu_rbar6; /*!< MPU Region 6 Base Address Register */

uint32_t cm33_mpu_rlar6; /*!< MPU Region 6 Limit Address Register */

uint32_t cm33_mpu_rbar7; /*!< MPU Region 7 Base Address Register */

uint32_t cm33_mpu_rlar7; /*!< MPU Region 7 Limit Address Register */

uint32_t cm33_mpu_ctrl_ns; /*!< Non-secure MPU Control Register.*/

uint32_t cm33_mpu_mair0_ns; /*!< Non-secure MPU Memory Attribute Indirection Register 0 */

uint32_t cm33_mpu_mair1_ns; /*!< Non-secure MPU Memory Attribute Indirection Register 1 */

uint32_t cm33_mpu_rbar0_ns; /*!< Non-secure MPU Region 0 Base Address Register */

uint32_t cm33_mpu_rlar0_ns; /*!< Non-secure MPU Region 0 Limit Address Register */

uint32_t cm33_mpu_rbar1_ns; /*!< Non-secure MPU Region 1 Base Address Register */

uint32_t cm33_mpu_rlar1_ns; /*!< Non-secure MPU Region 1 Limit Address Register */

uint32_t cm33_mpu_rbar2_ns; /*!< Non-secure MPU Region 2 Base Address Register */

uint32_t cm33_mpu_rlar2_ns; /*!< Non-secure MPU Region 2 Limit Address Register */

uint32_t cm33_mpu_rbar3_ns; /*!< Non-secure MPU Region 3 Base Address Register */

uint32_t cm33_mpu_rlar3_ns; /*!< Non-secure MPU Region 3 Limit Address Register */

uint32_t cm33_mpu_rbar4_ns; /*!< Non-secure MPU Region 4 Base Address Register */

uint32_t cm33_mpu_rlar4_ns; /*!< Non-secure MPU Region 4 Limit Address Register */

uint32_t cm33_mpu_rbar5_ns; /*!< Non-secure MPU Region 5 Base Address Register */

uint32_t cm33_mpu_rlar5_ns; /*!< Non-secure MPU Region 5 Limit Address Register */

uint32_t cm33_mpu_rbar6_ns; /*!< Non-secure MPU Region 6 Base Address Register */

uint32_t cm33_mpu_rlar6_ns; /*!< Non-secure MPU Region 6 Limit Address Register */

uint32_t cm33_mpu_rbar7_ns; /*!< Non-secure MPU Region 7 Base Address Register */

uint32_t cm33_mpu_rlar7_ns; /*!< Non-secure MPU Region 7 Limit Address Register */

uint32_t cm33_sau_ctrl; /*!< SAU Control Register.*/

uint32_t cm33_sau_rbar0; /*!< SAU Region 0 Base Address Register */

uint32_t cm33_sau_rlar0; /*!< SAU Region 0 Limit Address Register */

uint32_t cm33_sau_rbar1; /*!< SAU Region 1 Base Address Register */

uint32_t cm33_sau_rlar1; /*!< SAU Region 1 Limit Address Register */

uint32_t cm33_sau_rbar2; /*!< SAU Region 2 Base Address Register */

uint32_t cm33_sau_rlar2; /*!< SAU Region 2 Limit Address Register */

uint32_t cm33_sau_rbar3; /*!< SAU Region 3 Base Address Register */

uint32_t cm33_sau_rlar3; /*!< SAU Region 3 Limit Address Register */

uint32_t cm33_sau_rbar4; /*!< SAU Region 4 Base Address Register */

uint32_t cm33_sau_rlar4; /*!< SAU Region 4 Limit Address Register */

uint32_t cm33_sau_rbar5; /*!< SAU Region 5 Base Address Register */

uint32_t cm33_sau_rlar5; /*!< SAU Region 5 Limit Address Register */

uint32_t cm33_sau_rbar6; /*!< SAU Region 6 Base Address Register */

uint32_t cm33_sau_rlar6; /*!< SAU Region 6 Limit Address Register */

uint32_t cm33_sau_rbar7; /*!< SAU Region 7 Base Address Register */

uint32_t cm33_sau_rlar7; /*!< SAU Region 7 Limit Address Register */

uint32_t ahbsc_flash00_mem_rule0; /*!< FLASH 00 Memory Rule Register 0 */

uint32_t ahbsc_flash00_mem_rule1; /*!< FLASH 00 Memory Rule Register 1 */

uint32_t ahbsc_flash00_mem_rule2; /*!< FLASH 00 Memory Rule Register 2 */

uint32_t ahbsc_flash00_mem_rule3; /*!< FLASH 00 Memory Rule Register 3 */

uint32_t ahbsc_flash01_mem_rule0; /*!< FLASH 01 Memory Rule Register 0 */

uint32_t ahbsc_flash01_mem_rule1; /*!< FLASH 01 Memory Rule Register 1 */

uint32_t ahbsc_flash01_mem_rule2; /*!< FLASH 01 Memory Rule Register 2 */

uint32_t ahbsc_flash01_mem_rule3; /*!< FLASH 01 Memory Rule Register 3 */

uint32_t ahbsc_flash02_mem_rule; /*!< FLASH 02 Memory Rule Register */

uint32_t ahbsc_flash03_mem_rule; /*!< FLASH 03 Memory Rule Register */

uint32_t ahbsc_rom_mem_rule0; /*!< ROM Memory Rule Register 0 */

uint32_t ahbsc_rom_mem_rule1; /*!< ROM Memory Rule Register 1 */

uint32_t ahbsc_rom_mem_rule2; /*!< ROM Memory Rule Register 2 */

uint32_t ahbsc_rom_mem_rule3; /*!< ROM Memory Rule Register 3 */

uint32_t ahbsc_ramx_mem_rule0; /*!< RAMX Memory Rule Register 0 */

uint32_t ahbsc_ramx_mem_rule1; /*!< RAMX Memory Rule Register 1 */

uint32_t ahbsc_ramx_mem_rule2; /*!< RAMX Memory Rule Register 2 */

uint32_t ahbsc_ramx_mem_rule3; /*!< RAMX Memory Rule Register 3 */

uint32_t ahbsc_rama_mem_rule; /*!< RAMA Memory Rule Register */

uint32_t ahbsc_ramb_mem_rule; /*!< RAMB Memory Rule Register */

uint32_t ahbsc_ramc_mem_rule0; /*!< RAMC Memory Rule Register 0 */

uint32_t ahbsc_ramc_mem_rule1; /*!< RAMC Memory Rule Register 1 */

uint32_t ahbsc_ramd_mem_rule0; /*!< RAMD Memory Rule Register 0 */

uint32_t ahbsc_ramd_mem_rule1; /*!< RAMD Memory Rule Register 1 */

uint32_t ahbsc_rame_mem_rule0; /*!< RAME Memory Rule Register 0 */

uint32_t ahbsc_rame_mem_rule1; /*!< RAME Memory Rule Register 1 */

uint32_t ahbsc_ramf_mem_rule0; /*!< RAMF Memory Rule Register 0 */

uint32_t ahbsc_ramf_mem_rule1; /*!< RAMF Memory Rule Register 1 */

uint32_t ahbsc_ramg_mem_rule0; /*!< RAMG Memory Rule Register 0 */

uint32_t ahbsc_ramg_mem_rule1; /*!< RAMG Memory Rule Register 1 */

uint32_t ahbsc_ramh_mem_rule0; /*!< RAMH Memory Rule Register 0 */

uint32_t ahbsc_apb_bridge0_mem_rule0; /*!< APB Bridge 0 Memory Rule Register 0 */

uint32_t ahbsc_apb_bridge0_mem_rule1; /*!< APB Bridge 0 Memory Rule Register 1 */

uint32_t ahbsc_apb_bridge0_mem_rule2; /*!< APB Bridge 0 Memory Rule Register 2 */

uint32_t ahbsc_apb_bridge0_mem_rule3; /*!< APB Bridge 0 Memory Rule Register 3 */

uint32_t ahbsc_apb_bridge1_mem_rule0; /*!< APB Bridge 1 Memory Rule Register 0 */

uint32_t ahbsc_apb_bridge1_mem_rule1; /*!< APB Bridge 1 Memory Rule Register 1 */

uint32_t ahbsc_apb_bridge1_mem_rule2; /*!< APB Bridge 1 Memory Rule Register 2 */

uint32_t ahbsc_aips_bridge0_mem_rule0; /*!< AIPS BRIDGE 0 Peripherals Memory Rule Register 0 */

uint32_t ahbsc_aips_bridge0_mem_rule1; /*!< AIPS BRIDGE 0 Peripherals Memory Rule Register 1 */

uint32_t ahbsc_aips_bridge0_mem_rule2; /*!< AIPS BRIDGE 0 Peripherals Memory Rule Register 2 */

uint32_t ahbsc_aips_bridge0_mem_rule3; /*!< AIPS BRIDGE 0 Peripherals Memory Rule Register 3 */

uint32_t ahbsc_ahb_periph0_mem_rule0; /*!< AHB Peripherals 0 Memory Rule Register 0 */

uint32_t ahbsc_ahb_periph0_mem_rule1; /*!< AHB Peripherals 0 Memory Rule Register 1 */

uint32_t ahbsc_ahb_periph0_mem_rule2; /*!< AHB Peripherals 0 Memory Rule Register 2 */

uint32_t ahbsc_aips_bridge1_mem_rule0; /*!< AIPS BRIDGE 1 Peripherals Memory Rule Register 0 */

uint32_t ahbsc_aips_bridge1_mem_rule1; /*!< AIPS BRIDGE 1 Peripherals Memory Rule Register 1 */

uint32_t ahbsc_ahb_periph1_mem_rule0; /*!< AHB Peripherals 1 Memory Rule Register 0 */

uint32_t ahbsc_ahb_periph1_mem_rule1; /*!< AHB Peripherals 1 Memory Rule Register 1 */

uint32_t ahbsc_ahb_periph1_mem_rule2; /*!< AHB Peripherals 1 Memory Rule Register 2 */

uint32_t ahbsc_aips_bridge2_mem_rule0; /*!< AIPS BRIDGE 2 Peripherals Memory Rule Register 0 */

uint32_t ahbsc_aips_bridge2_mem_rule1; /*!< AIPS BRIDGE 2 Peripherals Memory Rule Register 1 */

uint32_t ahbsc_aips_bridge3_mem_rule0; /*!< AIPS BRIDGE 3 Peripherals Memory Rule Register 0 */

uint32_t ahbsc_aips_bridge3_mem_rule1; /*!< AIPS BRIDGE 3 Peripherals Memory Rule Register 1 */

uint32_t ahbsc_aips_bridge3_mem_rule2; /*!< AIPS BRIDGE 3 Peripherals Memory Rule Register 2 */

uint32_t ahbsc_aips_bridge3_mem_rule3; /*!< AIPS BRIDGE 3 Peripherals Memory Rule Register 3 */

uint32_t ahbsc_aips_bridge4_mem_rule0; /*!< AIPS BRIDGE 4 Peripherals Memory Rule Register 0 */

uint32_t ahbsc_aips_bridge4_mem_rule1; /*!< AIPS BRIDGE 4 Peripherals Memory Rule Register 1 */

uint32_t ahbsc_aips_bridge4_mem_rule2; /*!< AIPS BRIDGE 4 Peripherals Memory Rule Register 2 */

uint32_t ahbsc_aips_bridge4_mem_rule3; /*!< AIPS BRIDGE 4 Peripherals Memory Rule Register 3 */

uint32_t ahbsc_ahb_sec_ctrl_periph_rule; /*!< AHB Secure Controller Peripheral Rule Register */

uint32_t ahbsc_flexspi0_reg0_mem_rule0; /*!< FLEXSPI0 Region 0 Memory Rule Register 0 */

uint32_t ahbsc_flexspi0_reg0_mem_rule1; /*!< FLEXSPI0 Region 0 Memory Rule Register 1 */

uint32_t ahbsc_flexspi0_reg0_mem_rule2; /*!< FLEXSPI0 Region 0 Memory Rule Register 2 */

uint32_t ahbsc_flexspi0_reg0_mem_rule3; /*!< FLEXSPI0 Region 0 Memory Rule Register 3 */

uint32_t ahbsc_flexspi0_reg1_mem_rule; /*!< FLEXSPI0 Region 1 Memory Rule Register */

uint32_t ahbsc_flexspi0_reg2_mem_rule; /*!< FLEXSPI0 Region 2 Memory Rule Register */

uint32_t ahbsc_flexspi0_reg3_mem_rule; /*!< FLEXSPI0 Region 3 Memory Rule Register */

uint32_t ahbsc_flexspi0_reg4_mem_rule; /*!< FLEXSPI0 Region 4 Memory Rule Register */

uint32_t ahbsc_flexspi0_reg5_mem_rule; /*!< FLEXSPI0 Region 5 Memory Rule Register */

uint32_t ahbsc_flexspi0_reg6_mem_rule; /*!< FLEXSPI0 Region 6 Memory Rule Register */

uint32_t ahbsc_flexspi0_reg7_mem_rule0; /*!< FLEXSPI0 Region 7 Memory Rule Register 0 */

uint32_t ahbsc_flexspi0_reg7_mem_rule1; /*!< FLEXSPI0 Region 7 Memory Rule Register 1 */

uint32_t ahbsc_flexspi0_reg7_mem_rule2; /*!< FLEXSPI0 Region 7 Memory Rule Register 2 */

uint32_t ahbsc_flexspi0_reg7_mem_rule3; /*!< FLEXSPI0 Region 7 Memory Rule Register 3 */

uint32_t ahbsc_flexspi0_reg8_mem_rule; /*!< FLEXSPI0 Region 8 Memory Rule Register */

uint32_t ahbsc_flexspi0_reg9_mem_rule; /*!< FLEXSPI0 Region 9 Memory Rule Register */

uint32_t ahbsc_flexspi0_reg10_mem_rule; /*!< FLEXSPI0 Region 10 Memory Rule Register */

uint32_t ahbsc_flexspi0_reg11_mem_rule; /*!< FLEXSPI0 Region 11 Memory Rule Register */

uint32_t ahbsc_flexspi0_reg12_mem_rule; /*!< FLEXSPI0 Region 12 Memory Rule Register */

uint32_t ahbsc_flexspi0_reg13_mem_rule; /*!< FLEXSPI0 Region 13 Memory Rule Register */

uint32_t ahbsc_sec_CPU1_int_mask0; /*!< Secure GPIO Mask Register 0 */

uint32_t ahbsc_sec_CPU1_int_mask1; /*!< Secure GPIO Mask Register 1 */

uint32_t ahbsc_sec_CPU1_int_mask2; /*!< Secure GPIO Mask Register 2 */

uint32_t ahbsc_sec_CPU1_int_mask3; /*!< Secure GPIO Mask Register 3 */

uint32_t ahbsc_sec_CPU1_int_mask4; /*!< Secure GPIO Mask Register 4 */

uint32_t ahbsc_sec_int_reg0; /*!< Secure Interrupt Mask for CPU1 Register 0 */

uint32_t ahbsc_sec_int_reg1; /*!< Secure Interrupt Mask for CPU1 Register 1 */

uint32_t ahbsc_sec_int_reg2; /*!< Secure Interrupt Mask for CPU1 Register 2 */

uint32_t ahbsc_sec_int_reg3; /*!< Secure Interrupt Mask for CPU1 Register 3 */

uint32_t ahbsc_sec_int_reg4; /*!< Secure Interrupt Mask for CPU1 Register 4 */

uint32_t ahbsc_sec_mask_lock; /*!< Secure GPIO Mask Lock Register 2 */

uint32_t ahbsc_master_sec_reg; /*!< Master Secure Level Register */

uint32_t ahbsc_master_sec_anti_pol_reg; /*!< Master Secure Level Anti-pole Register */

uint32_t cpu0_lock_reg; /*!< CPU0 Lock Control Register */

uint32_t cpu1_lock_reg; /*!< CPU1 Lock Control Register */

uint32_t misc_ctrl_dp_reg; /*!< Secure Control Duplicate Register */

uint32_t misc_ctrl_reg; /*!< Secure Control Register */

uint32_t mbc0_memn_glbac0; /*!< MBC Global Access Control Register 0 */

uint32_t mbc0_memn_glbac4; /*!< MBC Global Access Control Register 4 */

uint32_t mbc0_memn_glbac5; /*!< MBC Global Access Control Register 5 */

uint32_t mbc0_dom0_mem0_blk_cfg_w0; /*!< MBC Memory Block Configuration Word 0 */

uint32_t mbc0_dom0_mem0_blk_cfg_w1; /*!< MBC Memory Block Configuration Word 1 */

uint32_t mbc0_dom0_mem0_blk_cfg_w2; /*!< MBC Memory Block Configuration Word 2 */

uint32_t mbc0_dom0_mem0_blk_cfg_w3; /*!< MBC Memory Block Configuration Word 3 */

uint32_t mbc0_dom0_mem0_blk_cfg_w4; /*!< MBC Memory Block Configuration Word 4 */

uint32_t mbc0_dom0_mem0_blk_cfg_w5; /*!< MBC Memory Block Configuration Word 5 */

uint32_t mbc0_dom0_mem0_blk_cfg_w6; /*!< MBC Memory Block Configuration Word 6 */

uint32_t mbc0_dom0_mem0_blk_cfg_w7; /*!< MBC Memory Block Configuration Word 7 */

uint32_t itrc_out0_sel0; /*!< ITRC IRQ Trigger source selector register 0 */

uint32_t itrc_out0_sel1; /*!< ITRC IRQ Trigger source selector register 1 */

uint32_t itrc_css_reset_out1_sel0; /*!< ITRC CSS_RESET Trigger source selector register 0 */

uint32_t itrc_css_reset_out1_sel1; /*!< ITRC CSS_RESET Trigger source selector register 1 */

uint32_t itrc_puf_zeroize_out2_sel0; /*!< ITRC PUF_ZEROIZE Trigger source selector register 0 */

uint32_t itrc_puf_zeroize_out2_sel1; /*!< ITRC PUF_ZEROIZE Trigger source selector register 1 */

uint32_t itrc_ram_zeroize_out3_sel0; /*!< ITRC RAM_ZEROIZE Trigger source selector register 0 */

uint32_t itrc_ram_zeroize_out3_sel1; /*!< ITRC RAM_ZEROIZE Trigger source selector register 1 */

uint32_t itrc_chip_reset_out4_sel0; /*!< ITRC CHIP_RESET Trigger source selector register 0 */

uint32_t itrc_chip_reset_out4_sel1; /*!< ITRC CHIP_RESET Trigger source selector register 1 */

uint32_t itrc_itr_out_out5_sel0; /*!< ITRC ITR_OUT Trigger source selector register 0 */

uint32_t itrc_itr_out_out5_sel1; /*!< ITRC ITR_OUT Trigger source selector register 1 */

} tzm_secure_config_t;

ERR052001: ROM: Unable to change ISP mode I2C address

Description

Reconfiguration of the I2C slave address by changing the CMPA value (CMPA[I2C_SLAVE_ADDRESS]) does not work.

Workaround

There is no workaround for this issue. The default I2C address should be used when using the I2C interface in ISP mode.

ERR051421: SAI: Synchronous mode with bypass is not supported

Description

The SAI does not receive or transmit when:

Scenario 1. The transmitter is configured for synchronous mode (TCR2[SYNC] = 0b1), in the Transmit Configuration 2 register, and the receiver is in bypass (RCR2[BYP]=0b1), in the Receiver Configuration 2 register, then there will not be a bit clock as it is the source of the BCLK.

Scenario 2. The receiver is configured for synchronous mode (RCR2[SYNC] = 0b1) in the Receiver Configuration 2 register and the transmitter is in bypass (TCR2[BYP]=0b1), in the Transmit Configuration 2 register, then there will not be a bit clock as it is the source of the BCLK.

Workaround

If scenario 1, then set the TCR2[BCI] = 0b1, in the Transmit Configuration 2 register.

If scenario 2, then set the RCR2[BCI] = 0b1, in the Receiver Configuration 2 register.

ERR052088: SmartDMA: FlexIO_IRQ not correctly routed to SMARTDMAARCHB_INMUX

Description

The INP[0] selection for the SmartDMA INMUX (INPUTMUX0->SMARTDMAARCHB_INMUXx[INP] = 0) is incorrectly assigned to GPIO P0_0 instead of FLEXIO_IRQ.

Workaround

There is no workaround for this errata.

ERR051379: SRAM: Incorrect data reads when Auto-clock gating and ECC are enabled

Description

When Auto clock gating and ECC are both enabled for a given SRAM block, misaligned reads across block boundaries within that RAM block may return incorrect data.

Workaround

There are two workarounds for this errata:

1) If ECC and Auto-clock gating are required, ensure that misaligned accesses do not occur in your software

2) If either ECC or Auto-clock gating is not required, disable ECC or Auto-clock gating.

ERR051410: TSI: TSICH bit field cannot be read correctly

Description

When reading TSI0->TSI_CONFIG[TSICH] bitfield, the upper most bit (MSB) will always read as zero ('0'). This will result in software incorrectly reading back a channel number less than or equal to 15 even if a channel greater than 15 was written to the bit field.

Workaround

There is no workaround for this errata. However, functionality of the TSI module is not affected. If a channel greater than 15 is desired to be written to the bit field, software should write the desired channel number to the bit field and assume the write was successful.

Note that the TSICH bitfield should be configured after all other bit fields have been configured. Writing other bit fields after writing the TSICH bit field will overwrite the TSICH configuration

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.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://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.

Suitability for use in automotive applications — This NXP product has been qualified for use in automotive applications. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as "Critical Applications"), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

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.

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.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 1/2024
Document identifier: MCXNx4x_0P02G