RT500_2P43B

Mask Set Errata for Mask 2P43B

Rev. 3.1 — 8 November 2024 **Errata**

Document information

Information	Content
Keywords	MIMXRT595SFFOC, MIMXRT555SFFOC, MIMXRT533SFFOC, MIMXRT595SFFOCR, MIMXRT555SFFOCR, MIMXRT533SFFOCR, MIMXRT533SFAWCR, MIMXRT555SFAWCR, MIMXRT595SFAWCR
Abstract	RT500 errata

This report applies to mask 2P43B (**SILICONREV_ID = 0x000B0002)** for these products:

- MIMXRT595SFFOC
- MIMXRT555SFFOC
- MIMXRT533SFFOC
- MIMXRT595SFFOCR
- MIMXRT555SFFOCR
- MIMXRT533SFFOCR
- MIMXRT533SFAWCR
- MIMXRT555SFAWCR
- MIMXRT595SFAWCR

Errata and Information Summary

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
ERR050638	ADC: ADC misses software trigger when there is no ADC clock
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
ERR011246	Core: Floating-point state can be incorrectly cleared on some exception return faults
ERR011247	Core: Processor might not wake up to a SEVONPEND event when in WIC-based WFE sleep
ERR050503	Core: Non-secure HardFault exception might preempt when disabled by AIRCR.BFHFNMINS
ERR011312	Core: CTI trigger outputs for debug and interrupt requests do not support acknowledge handshake
ERR011249	Core: Secure non-invasive debug might be incorrectly blocked
ERR050458	FlexIO: Shifter Status/Error flag not generated correctly in Logic Mode
ERR011377	FlexSPI: FlexSPI DLL lock status bit not accurate due to timing issue
ERR051426	FlexSPI: FlexSPI DLL does not lock with FRO DIV1 clock source
ERR050610	FlexSPI: TX buffer fill / RX buffer drain by DMA with a single DMA descriptor cannot be performed
ERR052505	FSGPIO: A parametric shift over time is observed on Fail-Safe GPIO (FSGPIO) pin's output driver when it is powered above 1.98 V
ERR050641	GPIO: During initial power-up, a brief pull-up pulse could occur on the port pins
ERR051617	I3C: In I2C compatibility mode, read transaction not terminating correctly
ERR052040	I3C: Data loss in transmission with DMA when transfer size larger than Tx FIFO
ERR052041	I3C: Data corruption in reception with DMA when receive size is greater than Rx FIFO
ERR052148	I3C: I3C private read transfer cannot be performed due to timing issue
ERR011439	MIPI DSI: Checksum is incorrect for DCS command long packet writes with zero-length data payload
ERR050799	Non-Secure Boot ROM: ROM API initializes unused FlexSPI0 IO pins
ERR051427	Non-Secure Boot ROM: SRAM memory address overwritten by Boot ROM
ERR052244	Non-Secure Boot ROM: BOOT_FAIL_PIN does not work
ERR050716	Power Management: Leakage path between VDD1V8 and VDDIO_x
ERR052303	Secure Boot ROM: CRC checking of fuses covered by CRC5 field cannot be used when KEY_SCRAMBLE_SEED feature is enabled
ERR052309	Secure Boot ROM: CRC integrity check of OTP fuses fails during boot
ERR052225	TRNG: Errors at a high rate when generating random data independent of the entropy delay, core frequency, or core voltage
ERR050715	USBHSD: The detection handshaking fails when certain full-speed hubs are connected
ERR050739	USBHSD: Isochronous IN endpoint MaxPacketSize of 1024 byte limitation
ERR051403	USBHSD: In USB high-speed device mode, device writes extra byte(s) to the buffer if the NBytes is not multiple of 8 for OUT transfer

Table 1. Errata and Information Summary*...continued*

Erratum ID	Erratum Title
ERR050742	USBHSH: Transaction limitation for isochronous IN endpoints in High-bandwidth mode
ERR052276	USDHC: Does not support SD Memory Cards

1 Known Errata

1.1 ERR050638: ADC: ADC misses software trigger when there is no ADC clock

Description

ADC command execution can be initiated from up to 16 trigger sources. Those triggers can be generated via either software or hardware. However, when using software triggers, the ADC will not properly capture this event when no ADC clock is present.

The following conditions will cause this behavior:

- System enters low power state (both bus and functional clocks get disabled)

- System receives a temporary wake up and the ADC bus clock starts process to start running

- ADC receives a software trigger before the functional ADC clock completed start up and misses the event

Workaround

When no ADC clock is present, use only the hardware trigger functionality.

1.2 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)

1.3 ERR050505: Core: Access permission faults 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.

1.4 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.

1.5 ERR011246: Core: Floating-point state can be incorrectly cleared on exception return faults

Description

Cortex-M33 937163-C:

The Armv8-M architecture defines integrity checks which are performed before the exception return unstacking occurs.

These check the validity of the EXC_RETURN value and raise a fault if they fail. Because of this erratum it is possible for the floating-point state to be incorrectly cleared when one of these faults occurs.

Conditions

The floating-point state will be incorrectly cleared when all the following conditions are met:

1. One of the following exception return integrity checks fails:

• SFSR.INVER. • UFSR.INVPC (exiting a handler that is not active).

• UFSR.INVPC (EXC_RETURN[1]!=0). • SFSR.LSERR (when attempting to clear because of FPCCR.CLRONRET).

2. The floating-point state would have been unstacked if there had been no fault (that is, EXC_RETURN[4]==0, FPCCR.LSPACT==0 and access is permitted to the FPU).

Implications

The floating-point state can be incorrectly cleared if software causes one of the faults mentioned above. The scenario that could be problematic is when a Secure exception calls a Non-secure function, which in turn attempts to return from the exception. This erratum allows the Non-secure function to clear the Secure floating-point context. Note that doing so will always cause a Secure fault to be raised and no Secure state is ever leaked to Non-secure.

Workaround

There is no workaround for this erratum.

1.6 ERR011247: Core: Processor might not wake up to a SEVONPEND event when in WIC-based WFE sleep

Description

Cortex-M33 1015127-C:

The Armv8-M architecture includes a feature which allows an event to be sent when an interrupt state changes from inactive to pending (SEVONPEND). The Cortex-M33 processor also includes a Wakeup Interrupt Controller (WIC) to enable the processor to enter a low-power state. Because of this erratum, when in WIC-based WFE sleep, it is possible that the processor will fail to wake up as expected because a pended interrupt does not generate the expected event.

Conditions

• WIC-based sleep enabled (SCR.SLEEPDEEP==1, WICENACK==1).

• Only one of the banked SCR.SEVONPEND bits is set.

• The processor enters WFE sleep in the security state where the associated banked SCR.SEVONPEND field is not set.

Implications

The WIC will not wake up to an event generated by a pending interrupt targeting the alternate security state where the associated SCR.SEVONPEND bit is 1. However, it is expected that a system will not be affected by this behavior since software cannot depend on a wake-up event controlled by the alternate security state.

Workaround

This erratum is not expected to require a workaround.

1.7 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.

1.8 ERR011312: Core: CTI trigger outputs for debug and interrupt requests do not support acknowledge

Description

Cortex-M33 1042640-B:

The Cortex-M33 processor includes a Cross-Trigger Interface (CTI) which can generate internal or external events, depending on the CTI programming. Some of the internal events support software handshaking, such that the CTI holds the event until it is cleared by software. Because of this erratum some events are not configured to use software handshaking, which has some implications to debug and interrupt requests.

Workaround

• For debug requests: An additional CTI can be implemented in the system to generate debug requests which behave in the correct manner. This can be connected to the EDBGRQ and HALTED signals on the processor which are used by the internal CTI.

• For interrupt requests: The system should treat the interrupt requests as pulse-sensitive and convert them to level-sensitive if required.

1.9 ERR011249: Core: Secure non-invasive debug might be incorrectly blocked

Description

Cortex-M33 936921-C:

The Cortex-M33 processor supports an external debug authentication interface using the standard CoreSight debug enables (DBGEN, NIDEN, SPNIDEN, and SPIDEN). It also supports the DAUTHCTRL register allowing software to override the external debug authentication interface values. Because of this erratum, it is possible for the value of Secure non-invasive debug within the processor to be incorrectly disabled.

Conditions

- External authentication is configured as follows:
- Non-secure non-invasive debug allowed (that is, NIDEN==1 or DBGEN==1).
- SPNIDEN == 0, SPIDEN ==1.
- DAUTHCTRL == 0x00000001 overrides Secure invasive debug to be disabled, but does not override Secure non-invasive debug.

Note: *The erratum requires the system to drive the external authentication interface in a non-standard way.*

Implications

The Cortex-M33 processor will behave as if Secure non-invasive debug is disabled even though the external SPIDEN value is 1.

Workaround

When SPIDEN == 1, systems must drive SPNIDEN == 1.

1.10 ERR050458: FlexIO: Shifter Status/Error flag not generated correctly in Logic Mode

Description

Some shifters will not generate status or error flags correctly when configured for logic mode (SHIFTCTLn[SMOD] = 0b111). Shifters 0, 1, 2, and 3 behave correctly. All other shifters are affected.

Workaround

In logic mode, if the Status/Error flags are required, use shifter 0, 1, 2, or 3. If the Status/Error flag is not required, then any shifter could be used in logic mode.

Known Errata

1.11 ERR011377: FlexSPI: FlexSPI DLL lock status bit not accurate due to timing issue

Description

After configuring DLL and the lock status bit is set, the data may be wrong if read/write immediately from FLEXSPI based external flash due to timing issue.

Workaround

Add delay time (100 NOP) again after the DLL lock status is set.

1.12 ERR051426: FlexSPI: FlexSPI DLL does not lock with FRO DIV1 clock source

Description

The FlexSPI DLL is a delay line chain feature, which can be set to a fixed number of delay cells or auto-adjusted to lock on a certain phase delay to the reference clock. One of the reference clocks available to the FlexSPI module is the FRO clock.

When using the FRO Divide-by-1 (FRO_DIV1) clock source option with the FlexSPI module, the DLL does not lock. This is due to the FRO clock not meeting the DLL duty cycle requirement. This does not affect FRO clocks that are divided by 2 or greater.

Workaround

When using the FRO_DIV1 clock source for the FlexSPI, disable the DLL and set the override delay with DLLACR/DLLBCR[OVRDEN] =1 and DLLACR/DLLBCR[OVRDVAL]=0.

1.13 ERR050610: FlexSPI : TX buffer fill / RX buffer drain by DMA with a single DMA descriptor cannot be performed

Description

Using FlexSPI register interface, you cannot perform TX buffer fill / RX buffer drain by DMA with a single DMA descriptor if the transfer size exceeds the FIFO watermark level.

Workaround

The erratum requires a software workaround for maintaining a linked DMA descriptor array. The link array consumes about 2K RAM consumption to support the FLEXSPI TX watermark starting from 8 bytes.

1.14 ERR052505: FSGPIO: A parametric shift over time is observed on Fail-Safe GPIO (FSGPIO) pin's output driver when it is powered above 1.98 V

Description

All GPIO pins are specified to be fail safe up to 3.6 V when VDDIO supply = 0 V except High Speed pads. However, when the FSGPIO pins are powered above 1.98V, a parametric shift over time is observed on its output driver. The output low drive current (IOL) is degraded, leading to a longer fall time and output low voltage level (VOL) is increased. Analog and input functionality is not impacted.

For i.MX RT500, the affected FSGPIO are the IO pins powered by the VDDIO_3 supply.

VDDIO_3:

PIO4_20 to PIO4_31

PIO5_0 to PIO5_3

For new or updated designs, use 1.8 V (1.71-1.98 V) for the FSGPIO power supply. For the legacy designs, refer to Technote TN00188 for IOL and fall time degradation information when operating above 1.98 V. If it is determined that IO does not meet the mission profile requirement of the end application, implement the workaround.

Workaround

The power supply pins, VDDIO_3, should be connected to 1.8 V (1.71-1.98 V) and the related IO bank supply voltage range selection bit field in PADVRANGE register should be configured as continuous voltage range (00) or low voltage range (01). Both continuous voltage range and low voltage range are allowed. It is recommended that the low voltage range is used for better delay performance and power consumption when the power supply is 1.8 V (1.71-1.98 V).

1.15 ERR050641 GPIO: During initial power-up, a brief pull-up pulse could occur on the port and dedicated output pins

Description

By default (reset state), the GPIO pins are in the high Z state and typically stays high Z until the application code changes its state. The internal pull-up and internal pull-down resistors are disabled by default.

During power-up, the internal pull-up resistor may not initialize during the early part of the IO ramp-up, resulting in a brief pull-up current pulse on some port pins that drops to zero before the VDDIO_x supplies reach the minimum operating voltage. Except for GPIO with the high speed pads, all fail safe GPIOs are affected by this issue. This issue also affects the dedicated output pins, PMIC_MODE0/1, associated with the VDD_AO1V8 supply rail.

Workaround

A pulldown resistor (~10K) can be added to the GPIO pin(s) to minimize the peak voltage where the application is sensitive to potential pulses. A 10nF shunt capacitor instead of a pulldown resistor can be added to the dedicated output pins for the PMIC_MODE0/1 pins.

1.16 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.

1.17 ERR052040: I3C: Data loss in transmission with DMA when transfer size larger than Tx

Description

The I3C has an 8 byte Tx FIFO used in DMA mode transmission. This FIFO is overflown if a single DMA loop size is larger than the FIFO size. This results in frame data loss at the end due to unexpected DMA requests.

Workaround

When the transfer size is larger than 8 bytes, set the trigger level for the Tx FIFO (MDATACTRL[TXTRIG]) to 0b00 (Trigger on empty). Set the DMA destination address in the MWDATAH register and set the DMAWIDTH bit field of the MDMACTRL register to half-word.

Add additional 0s to the source data to arrange the data array in a 32-bit format for the DMA write transfer. Treat the data as a 32-bit value with the upper 16-bit equal to 0 and the 32-bit write transfer between the DMA and the I3C modules. The source data will be stored like:

0x(00)(00)(byte1)(byte0)

0x(00)(00)(byte3)(byte2)

0x(00)(00)(byte5)(byte4)

0x(00)(00)(byte7)(byte6)

…

For each DMA descriptor, set the transfer width to 32-bit and the transfer size to 16 bytes. With this configuration, each DMA descriptor will transfer 8 bytes actual data(16 bytes processed data).

1. Using DMA linked descriptor

Set up MA linked descriptors. Each DMA descriptor exhausts 8 actual bytes then links to another descriptor. The number of descriptors depends on the transfer size. When transferring 32 bytes, it requires 4 descriptors. When transferring 34 bytes, it requires 5 descriptors where the last descriptor exhausts 2 bytes.

2. Configure the next descriptor in the DMA callback

Enable the DMA interrupt. Setup one DMA transfer descriptor and set transfer size to 8 actual bytes. In the DMA callback, configure the next 8 actual bytes in descriptor transfer until the transfer is complete. If the remaining actual size is less than 8, set the remaining byte descriptor.

1.18 ERR052041: I3C : Data corruption with reception with DMA when receive size is greater than Rx FIFO

Description

The I3C has a 6 byte Rx FIFO used in DMA mode reception. This FIFO underflows if it is read for more than 6 bytes when receiving larger frames using DMA via read of the MRDATAB/ MRDATAH registers. This results in incorrect frame data at the end.

Workaround

When the transfer size is larger than 6 bytes, set the initial trigger level for the Rx FIFO (MDATACTRL[RXTRIG]) to 0b11 (Trigger on 3/4 or more full). Set the DMA destination address in the MRDATAB register and set the DMAWIDTH bit field of the MDMACTRL register to one byte.

1. Using DMA linked descriptor

Set up DMA linked descriptors. Each DMA descriptor exhausts 6 bytes then links to another descriptor. The number of descriptors depends on the transfer size. When transferring 30 bytes, it requires 5 descriptors . When transferring 34 bytes, it requires 6 descriptors where the last descriptor exhausts 4 bytes.

If the receive size is an odd number, then the DMA should receive even-numbered bytes, and the last byte should be read by the CM33. If the transfer size is not an integer multiple of 6, then the trigger level for Rx FIFO should be changed to the last descriptor. This results in two cases to take into account 1) a remainder of 2 case and 2) a remainder of 4 case. For the remainder of 2 case, change the trigger level to 0b00 (kI3C_RxTriggerOnNotEmpty) for the last descriptor. And for the remainder or 4 case, change the trigger level to 0b10 (kI3C_RxTriggerUntilOneHalfOrMore) for the last descriptor.

2. Configure the next descriptor in the DMA callback

Enable the DMA interrupt. If the receive size is an odd number, then use DMA to receive even-numbered bytes, and the last byte should be read by the CM33.

Set up one DMA receive descriptor with receive size of 6 bytes. In the DMA callback, configure the next 6 byte descriptor until the receive is completed. For the remainder of 2 case, if the remaining size is less than 6, change the trigger level to 0b00 (kI3C_RxTriggerOnNotEmpty). For the remainder of 4 case, change the trigger level to 0b10 (kI3C_RxTriggerUntilOneHalfOrMore), then set the remaining byte descriptor.

1.19 ERR052148: I3C : I3C private read transfer cannot be performed due to timing issue

Description

As per the MIPI I3C specification, the I3C protocol Controller can start a private read transaction with a specified target in a specified format. Please see section "A2 I3C Private Write and Read Transfers" (MIPI I3C standard Version 1.1.1 specification) for further details.

The I3C Controller cannot produce the specified sequence as stated above for SDR Private Read when attempted via MCTRL register (offset 0x84) interface.

Workaround

I3C SDR private read can be performed without 7E step, as defined in the section "5.1.2 Bus Communication" of MIPI I3C standard Version 1.1.1 specification. The SDR private read must be started directly by using

Target's Dynamic Address. During private read, skip the step of sending Address 7E /W via MCTRL register and instead start with Address of Target DA /R in MCTRL register.

1.20 ERR011439: MIPI DSI: Checksum is incorrect for DCS command long packet writes with zero-length data payload

Description

According to the MIPI DSI specification, long packets are comprised of a Packet Header and a payload of 0 to 2^16-1 bytes. For the special case of a zero-length payload, the specification requires the checksum must be set to 0xFFFF.

The MIPI DSI controller produces an incorrect checksum for DCS commands issued via long packets with zero length payloads in LP (DSI Low-Power mode). There is no such issue for similar commands issued in HP (DSI High-Power mode).

This issue should not affect normal application operation because packets with zero data length would normally be sent using the short packet format. However, since the MIPI DSI spec specifically states this behavior, MIPI DSI certification would fail on this issue.

Workaround

Use short packet format to send DCS commands with zero length data payloads.

1.21 ERR050799: Non-Secure Boot ROM: ROM API initializes unused FlexSPI0 IO pins

Description

Each port IO pin has a dedicated control register in the IOPCTL module that allows control of various functions and characteristics. By default, the port IO pins have their input buffer disabled. This keeps pins that may be left floating from causing excess current leakage.

During the FlexSPI boot flow, the ROM API IAP_FlexspiNorAutoConfig() is called, for initialization of the FlexSPI module before accessing the Flash memory. The ROM configures the FLEXSPI0 pins (PIO1_18 - PIO1_28) enabling the input buffers for those respective pins regardless of what memory device is used.

Workaround

If the application does not use these FlexSPI0 pins as inputs, it should disable the input buffers for these pins in the IOPCTL registers via bit 6 (IBENA).

1.22 ERR051427: Non-Secure Boot ROM: SRAM memory address overwritten by Boot ROM

Description

SRAM memory address partition 0 is the only partition that remains powered through a reset and therefore is the only partition that supports RAM retention. The addresses for partition 0 include Non-secure/Secure address ranges, as well as Code/Data Bus address ranges.

During each boot,

- The Boot ROM will write a 32-bit value, 0x3CC35AA5, to 0x00000FD0 located in SRAM partition 0. This has no adverse effect on Boot ROM operation.

- For devices with the DSP disabled, the Boot ROM will also write 32-bit values 0x00004136 to address 0x00000000 and 0xF01D0071 to address 0x00000004, respectively, located in SRAM partition 0. This has no adverse effect on Boot ROM operation.

Workaround

SRAM locations 0x00000FD0 – 0x00000FD3 cannot be used for retention RAM.

For devices with the DSP disabled, SRAM locations 0x00000000 – 0x00000007 can also not be used for retention RAM.

These address locations can be used as standard SRAM memory but should not be used to store any data/ code that needs to be retained beyond warm resets (SYSRESET, WDT_RESET).

1.23 ERR052244: Non-Secure Boot ROM: BOOT_FAIL_PIN does not function properly

Description

The Boot ROM has provides the ability to define a GPIO as a BOOT_FAIL_PIN where this pin can be used to power cycle the system. It will be driven high to indicate boot failure prior to locking up the chip on error conditions. This functionality is controlled through OTP Fuse Map.

However, the Boot ROM code does not set the BOOT_FAIL_PIN high properly.

Workaround

This feature is not recommended for use.

1.24 ERR050716: Power Management: Leakage path between VDD1V8 and VDDIO_x

Description

The power sequencing specification in the datasheet mentions that the VDDIO_x rail can be optionally powered after the VDD1V8 and the delta voltage between VDDIO_x and VDD1V8 must be 1.89 V or less.

Before the VDDIO_x is powered, there is a leakage path between the VDD1V8 and VDDIO_x domain. The leakage is approximately 1.5 mA (VDD1V8 - VDDIO / 800 ohm). This leakage does not cause any reliability issues. There is no leakage once the VDDIO_x rail is above VDD1V8 - 0.4 V.

Workaround

In order to avoid the leakage path, the VDDIO_x rail should not be powered after the VDD1V8.

1.25 ERR052303: Secure Boot ROM: CRC checking of fuses covered by CRC5 field cannot be used when KEY_SCRAMBLE_SEED feature is enabled

Description

The device supports integrity checking of OTP fuses. Since the fuses are programmed in different life cycle stages of the SoC. The fuses are divided in to eight groups and their corresponding CRC32 values are programmed in CRC0 – CRC7 fuse words.

The Boot ROM supports OTP fuse integrity checking on every boot based on SEC_BOOT_CFG[5].ENABLE_CRC_CHECK field.

1. CRC_DISABLE (b’00): CRC checking of OTP words on startup is disabled.

2. CRC_ENABLE (b’01): CRC check of all OTP words is enabled. CRC0-6 are checked fuse groups are integrity checked.

3. CRC_NXPONLY (b’10): CRC check is enabled only for NXP programmed OTP words. CRC0/1/2/3 fuse groups programmed by NXP factory are integrity checked.

4. CRC_ENABLE2 (b’11): CRC check of all OTP words is enabled. CRC0-6 are checked fuse groups are integrity checked.

The Boot ROM provides API to calculate the CRC for the fuse groups. During manufacturing, this API can be used to calculate and program the CRC4-7 fused words.

The device supports scrambling of OTP_MASTER_KEY in device unique way when KEY_SCRAMBLE_SEED is programmed with non-zero value. In addition, when key scramble feature is enabled the OTP_MASTER_KEY fuse field is not readable by software but on startup descrambled key is delivered to AES engine by hardware logic. Thus protecting OTP_MASTER_KEY read out from runtime software.

Due to this key scramble feature, computing CRC for CRC5 group is not possible when the KEY_SCRAMBLE_SEED is set a non-zero value. The CRC5 value returned by API is inaccurate. Hence, CRC_ENABLE (b’01) and CRC_ENABLE2 (b’11) options are not usable when key scramble feature is used.

Workaround

When setting the KEY_SCRAMBLE_SEED to a non-zero value, use CRC_NXPONLY (b’10) or CRC_DISABLE (b’00) options for SEC_BOOT_CFG[5].ENABLE_CRC_CHECK field. Do CRC4 and CRC6 integrity check in application.

1.26 ERR052309: Secure Boot ROM: CRC integrity check of OTP fuses fails during boot

Description

The Boot ROM supports OTP fuse integrity checking on every boot based on SEC_BOOT_CFG[5].ENABLE_CRC_CHECK field. Since the fuses are programmed in different life cycle stages of the SoC. The fuses are divided in to eight groups and their corresponding CRC32 values are programmed in CRC0 – CRC7 fuse words. The CRC 0/1/2 correspond to the OTP Fuses programmed by NXP during different stages of manufacturing. However in certain batches CRC0 value is not programmed due to which the overall CRC checks will fail.

However, CRC0, an NXP internal value, is not programmed at production and the overall CRC checks will fail.

Workaround

In order to use CRC integrity checking feature, program CRC0, located at index 488, with a value of 0x9D070512.

CRC0 will be programmed for devices with following Date Codes WWYY or later for each package:

FOWLP : 2339

WLCSP : 2332

1.27 ERR052225: TRNG: Errors at a high rate when generating random data independent of entropy delay, core frequency, or core voltage

Description

The TRNG executes hardware health tests to validate the random bits on-chip. These health tests are a battery of statistical tests that are applied to the generated random bits and validate that the TRNG result has sufficient quality.

The HW health tests (quality tests) consist of internal test logic that is not functioning correctly. When the TRNG begins generating random data, this internal test logic should validate the correctness of the TRNG health tests implementation. However, there is an issue in this self-checking which causes the TRNG health checks to erroneously report errors based on timing violations by these self-checks.

Workaround

Disable the hardware health tests and use software health tests. NXP confirms that the TRNG produces data of high quality and behaves exactly as designed and expected. The quality of logged data can be validated with the NIST SP800-90B (non-IID) test suite which shows that the data has a per-bit min-entropy of ~0.82 bits per TRNG bit.

1.28 ERR050715: USBHSD: The detection handshaking fails when certain full-speed hubs are connected

Description

As a high-speed device, when certain full-speed hubs are connected, the USB device does not detect the HOST KJ sequence correctly and, as a result, does not recognize the speed of the connected host. In this case, the USB device can act erratically due to the wrong speed detection.

Workaround

There are two workarounds:

1. The software workaround below can be implemented in usb_dev_hid_mouse where API is called "USB_ DeviceHsPhyChirpIssueWorkaround()". In event handler in USB_DeviceCallback(),

– On "kUSB_DeviceEventBusReset" event, USB_DeviceHsPhyChirpIssueWorkaround() should be called to identify the speed of the host connected to. If full-speed host is connected or

"isConnectedToFsHostFlag" is set, FORCE_FS (bit 21) of DEVCMDSTAT register should be set to force the device operating in full-speed mode.

– On "kUSB_DeviceEventDetach" event, FORCE_FS (bit 21) of DEVCMDSTAT register should be cleared.

2. The software workaround below is available in tech note (TN00071) In event handler in USB_DeviceCallback(),

– On "kUSB_DeviceEventAttach" event, set PHY_RX register trip-level voltage to

the highest. USBPHY->RX &= ~(USBPHY_RX_ENVADJ_MASK);USBPHY->RX |= 2;.

– On "kUSB_DeviceEventBusReset" event, check the DEVCMDSTAT[SPEED] to determine the connected bus speed. (SPEED are bits 22 and 23). If DEVCMDSTAT[SPEED]=FS, FORCE_FS (bit 21) of DEVCMDSTAT should be set to force the device operating in full-speed mode.

– On "kUSB_DeviceEventGetDeviceDescriptor" event, or first SETUP packet has arrived, Set the USBPHY_RX[ENVADJ] field back to default 0. Otherwise, USBPHY_RX[ENVADJ] field will remain as 2 unless a disconnect event occurs.

– On "kUSB_DeviceEventDetach" event, Clear FORCE_FS (bit 21) of DEVCMDSTAT register to zero. Reset USBPHY_RX[ENVADJ] field back to default 0.

1.29 ERR050739: USBHSD: Isochronous IN endpoint MaxPacketSize of 1024 byte limitation

Description

The RT500 device family include a USB high-speed interface (USB1) that can operate in device mode at high-speed. The isochronous IN endpoint supports a MaxPacketSize of 1024 bytes.

When device isochronous IN endpoint sends a packet of MaxPacketSize of 1024 bytes in response to IN token from host, the isochronous IN endpoint interrupt is not set and the endpoint command/status list entry for the isochronous IN endpoint is not updated.

Workaround

Restrict the isochronous IN endpoint MaxPacketSize to 1023 bytes in device descriptor.

1.30 ERR051403: USBHSD: USBHSD: In USB high-speed device mode, device writes extra byte(s) to the buffer if the NBytes is not multiple of 8 for OUT

Description

The RT500 device family include a USB high-speed interface (USB1) that can operate in device mode at high-speed. The NBytes value represents the number of bytes that can be received in the buffer.

The RT500 USB device controller writes extra bytes to the receive data buffer if the size of the transfer is not a multiple of 8 bytes since the USB device controller always writes 8 bytes. For example, if the transfer length is 1 byte, 7 extra bytes will be written to the receive data buffer. If the transfer length is 7 bytes, 1 extra bytes will be written to the receive data buffer.

Workaround

Reserve an additional, intermediary buffer along with the buffer used by the application for USB data. After the USB data transfer into the intermediary buffer has been completed, use memcpy to move the data from the intermediary buffer into the application buffer, skipping the extraneous extra byte.

1.31 ERR050742: USBHSH: Transaction limitation for isochronous IN endpoints in High-bandwidth mode

Description

The RT500 device family includes a USB high-speed interface which can operate in host mode. Up to three high-speed transactions are allowed in a single micro-frame to support high-bandwidth endpoints. This mode is enabled by setting the MULT (Multiple) field in the Proprietary Transfer Descriptor (PTD) and is used to indicate to the host controller the number of transactions that should be executed per micro-frame. The allowed bit settings are:

• 00b Reserved. A zero in this field yields undefined results.

• 01b One transaction to be issued for this endpoint per micro-frame

• 10b Two transactions to be issued for this endpoint per micro-frame

• 11b Three transactions to be issued for this endpoint per micro-frame

However, for High-bandwidth mode, using multiple packets (MULT = 10b or 11b) in a frame causes unreliable operation. Only one transaction (MULT = 01b) can be issued per micro-frame.

Workaround

For isochronous IN endpoints, transactions should be limited to only one transaction (MULT = 01b) per micro-frame.

1.32 ERR052276: uSDHC: Does not support SD Memory Cards

Description

The uSDHC provides the interface between the host system and the SD/SDIO/MMC and should support the following features:

- Conforms to the SD Host Controller Standard Specification version 2.0/3.0

- Compatible with the MMC System Specification version 4.2/4.3/4.4/4.41/4.5/5.0

- Compatible with the SD Memory Card Specification version 3.0 and supports the Extended Capacity SD Memory Card

- Compatible with the SDIO Card Specification version 2.0/3.0

- Designed to work with SD Memory, miniSD Memory, SDIO, miniSDIO, SD Combo, MMC, MMC plus, and MMC RS cards

However, due to lack of 3.3V support on the High-speed pads, the uSDHC module is not compatible with the SD Memory Card Specification version 3.0 and does not support the Extended Capacity SD Memory Card. Therefore, SD Memory and miniSD Memory cards are not supported. This does not impact the other specifications.

Workaround

MMC Cards should be used as the SD Memory Cards are not supported.

2 Revision history

Table 2. Revision history

Document ID	Release date	Description
3.1	11/2024	The following errata added. • ERR052505
3.0	8/2024	The following errata were added. • ERR011249, ERR051051, ERR050505 • ERR050501, ERR011312, ERR051426 • ERR051617, ERR052040, ERR052041 • ERR051427, ERR052244, ERR052303 • ERR052309, ERR052225, ERR051403 • ERR052276, ERR052148
2.0	4/2021	The following errata added. • ERR050799
1.0	3/2021	The following errata were added. ERR050739, ERR050742
0.0	2/2021	Initial revision

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

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

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

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

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

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

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.

Contents

Date of release: 8 November 2024
Document identifier: RT500_2P43B