RT600

Errata sheet RT600

Rev. 2.2 — 8 November 2024 **Errata**

Document information

Information	Content
Keywords	MIMXRT685SFFOB, MIMXRT685SFVKB, MIMXRT685SFAWBR, MIMXRT633SFVKB, MIMXRT633SFAWBR
Abstract	RT600 errata

1 Product identification

The MIMXRT6xxSFAWBR WLCSP114 production samples have the following top-side package markings:

- First line: MRT6xxSF
- Second line: AW[R]R
- Third line: xxxxxx xx
- Fourth line: xxxxyyww
  - yyww: Date code with yy = year and ww = week

    Fifth line: xxx-yyy
- Sixth line: NXP

The MIMXRT6xxSFVKB VFBGA176 production samples have the following top-side package markings:

- First line: MRT6xxSFV
- Second line: K[R] xxxxxx
- Third line: xxyyww
- Fourth line: xxxxx
  - yyww: Date code with yy = year and ww = week

The MIMXRT685SFFOB FOWLP249 production samples have the following top-side package markings:

- First line: MRT6xxSFFOB
- Second line: xxxxxx
- Third line: xxxxxx
- Fourth line: xxxxxyyww
  - yyww: Date code with yy = year and ww = week

Table 1. Device revision table

Revision identifier	Revision description [R]
B	Initial device revision

2 Errata overview

Table 2. Functional problems table

Functional problems	Short description	Revision identifier	Detailed description
FlexSPI	FlexSPI DLL lock status bit not accurate due to timing issue.	B	Section 3.1
GPIO.1	During initial power-up, a brief pull-up pulse could occur on the port pins.	B	Section 3.2
ADC.1	ADC misses software and hardware triggers when there is no ADC clock.	B	Section 3.3
USB.1	For the USB high-speed device controller, the detection handshaking fails when certain full-speed hubs are connected.	B	Section 3.4
USB.2	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.	B	Section 3.5
USB.3	In USB high-speed device mode, when device isochronous IN endpoint sends a packet of MaxPacket	B	Section 3.6

Table 2. Functional problems table*...continued*

Functional problems	Short description	Revision identifier	Detailed description
	Size 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.		
USB.4	In USB high-speed host mode, only one transaction per micro-frame is allowed for isochronous IN endpoints.	B	Section 3.7
ROM.1	Non-Secure Boot ROM API initializes unused FlexSPI0 IO pins.	B	Section 3.8
ROM.3	Non-secure boot ROM: SRAM memory address overwritten by boot ROM.	B	Section 3.9
ROM.4	BOOT_FAIL_PIN does not function properly.	B	Section 3.10
I3C.1	In I2C compatibility controller mode, read transaction does not terminate correctly.	B	Section 3.11
I3C.2	Data loss in transmission with DMA when transfer size larger than Tx. FIFO	B	Section 3.12
I3C.3	Data corruption with reception with DMA when receive size is greater than Rx FIFO.	B	Section 3.13
I3C.4	I3C private read transfer cannot be performed due to timing issue.	B	Section 3.14
TRNG	Errors at a high rate when generating random data independent of entropy delay, core frequency, or core voltage.	B	Section 3.15
ROM.5	Non-Secure Boot ROM: CRC checking of fuses covered by CRC5 field cannot be used when KEY_SCRAMBLE_SEED feature is enabled.	B	Section 3.16
GPIO.2	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	B	Section 3.17

Table 3. AC/DC deviations table

AC/DC deviations	Short description	Product version(s)	Detailed description
n/a	n/a	n/a	n/a

Table 4. Errata notes

Errata notes	Short description	Revision identifier	Detailed description
VDD.1	Leakage path between VDD1V8 and VDDIO_x.	B	Section 5.1
ROM.2	Usage of ROM APIs when trust zone is disabled	B	Section 5.2

3 Functional problems detail

3.1 FlexSPI.1: FlexSPI DLL lock status bit not accurate due to timing issue

Introduction

Based on the sample clock source selection, the DLL control register (DLLxCR) can be used to set the delay line chain, which allows a fixed number of delay cells or auto-adjusted to lock on a certain phase delay to the reference clock.

Problem

After configuring the DLL and setting the lock status bit, data may not be in sync if a

read/write is performed immediately from a FLEXSPI based external flash due to timing

issues.

Work-around

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

3.2 GPIO.1: During initial power-up, a brief pull-up pulse could occur on the port pins

Introduction

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.

Problem

During VDDIO_x power-up, the internal pull-up resistor may not initialize during the early part of the VDDIO_x 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 PIO1_19 to PIO1_31, PIO2_0 to PIO2_8, PIO0_21, PIO0_22, and PIO_23 pins, all fail-safe GPIOs are affected by this issue.

Typically, for a 20 ms power-up ramp, the pulse width of the glitch is approximately 8 ms and the amplitude is about 1.2 V.

Work-around

As a 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.

3.3 ADC.1: ADC misses software and hardware triggers when there is no ADC clock

Introduction

ADC command execution can be initiated from up to 16 trigger sources. These triggers can be generated either via software trigger or hardware trigger.

Problem

When no ADC clock is present, the ADC will not properly capture software or hardware trigger events. The following set of conditions cause this behavior:

- First, the system enters a low-power state (both bus and functional clocks get disabled).
- Then, the system receives a wake-up and, upon exit from the lower power state, the CPU clocks start running.
- Finally, the ADC receives a software or hardware trigger before the functional ADC clock has completed start up and misses the trigger event.

Work-around

To use software or hardware triggers, the ADC must be left powered and the ADC source clock must be kept enabled upon entry into the low-power state. Please note, when exiting the low-power state, a 3 ADC Clock cycle synchronization delay is required between waking up until SWTRIG can be accepted.

3.4 USB.1: For the USB high-speed device controller, the detection handshaking fails when certain full-speed hubs are connected

Introduction

See the USB2.0 specification for details regarding the USB High-speed Detection Handshake protocol.

Problem

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.

Work-around

There are two workarounds:

1. The software work-around below can be implemented in usb_dev_hid_mouse where the API is called "USB_DeviceHsPhyChirpIssueWorkaround()". In the event handler in USB_DeviceCallback(),
   - On "kUSB_DeviceEventBusReset" event, USB_DeviceHsPhyChirpIssueWorkaround() should be called to identify the speed of the host connected to. If a 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 is 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 the 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.

3.5 USB.2: In USB high-speed device mode, the device writes extra byte(s) to the buffer if the NBytes is not multiple of 8 for OUT transfer

Introduction

The RT600 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.

Problem

The RT600 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 byte will be written to the receive data buffer.

Work-around

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. This software work-around is implemented on the SDK software platform.

3.6 USB.3: In USB high-speed device mode, 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

Introduction

The RT600 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.

Problem

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

Work-around

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

3.7 USB.4: In USB high-speed host mode, only one transaction per micro-frame is allowed for isochronous IN endpoints

Introduction

The RT600 device family include 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.

Problem

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.

Work-around

There is no software workaround. Only one transaction can be issued per micro-frame.

3.8 ROM.1: Non-secure Boot ROM API initializes unused FlexSPI0 IO pins

Introduction

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.

Problem

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).

Work-around

None.

3.9 ROM.3: Non-secure Boot ROM: SRAM memory address overwritten by boot ROM

Introduction

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, and Code/Data Bus address ranges.

Problem

During each boot, the Boot ROM writes a 32-bit value, 0x3CC35AA5, to 0x00000FD0 located in SRAM partition 0. This has no adverse effect on Boot ROM operation.

Work-around

SRAM locations 0x00000FD0 – 0x00000FD3 cannot 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).

3.10 ROM.4: BOOT_FAIL_PIN does not function properly

Introduction

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

Problem

The Boot ROM code does not set the BOOT_FAIL_PIN high and as a result, this feature is not available.

Work-around

None.

3.11 I3C.1: In I2C compatibility controller mode, the read transaction does not terminate correctly

Introduction

The I3C module can operate in I2C compatibility mode to support I2C devices.

Problem

When operating in I2C compatibility controller mode, the end of any read transaction may terminate with a repeated START followed by the STOP instead of only a STOP.

Work-around

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

3.12 I3C.2: Data loss in transmission with DMA when transfer size larger than Tx FIFO

Introduction

The I3C has an 8 byte Tx FIFO used in DMA mode transmission.

Problem

The 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.

Work-around

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.

3.13 I3C.3: Data corruption with reception with DMA when receive size is greater than Rx FIFO

Introduction

The I3C has a 6 byte Rx FIFO used in DMA mode reception.

Problem

The 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.

Work-around

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.

3.14 I3C.4: I3C private read transfer cannot be performed due to timing issue

Introduction

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.

Problem

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

Work-around

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.

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

Introduction

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.

Problem

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.

Work-around

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.

3.16 ROM.5 Non-Secure Boot ROM: CRC checking of fuses covered by CRC5 field cannot be used when KEY_SCRAMBLE_SEED feature is enabled

Introduction

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.

- CRC_DISABLE (b'00): CRC checking of OTP words on startup is disabled.
- CRC_ENABLE (b'01): CRC check of all OTP words is enabled. CRC0-6 are checked fuse groups are integrity checked.
- 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.
- 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.

Problem

Due to the 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.

Work-around

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.

3.17 GPIO.2: 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 RT600, the affected FSGPIO are the IO pins powered by the VDDIO_0, VDDIO_1, and VDDIO_2 supplies.

VDDIO_0:

PIO0_0 to PIO0_13

PIO1_11 to PIO1_17

PIO2_12 to PIO2_23

PIO3_25 to PIO3_31

PIO4_0 to PIO4_10

PIO7_24 to PIO7_31

VDDIO_1:

PIO0_14 to PIO0_31

PIO1_0 to PIO1_10

PIO2_24 to PIO2_31

PIO3_0 to PIO3_24

PMIC_I2C_SCL

PMIC_I2C_SDA

VDDIO_2:

PIO2_9 to PIO2_11

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_0, VDDIO_1, and VDDIO_2, should be connected to 1.8V (1.71-1.98 V) and the related IO bank supply voltage range selection bit fields in PADVRANGE register should be configured as wide voltage range (00) or low voltage range (01). Both wide 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 1.8V (1.71-1.98 V).

4 AC/DC deviations detail

5 Errata notes detail

5.1 Leakage path between VDD1V8 and VDDIO_x

On RT600, 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.

5.2 ROM.2: Usage of ROM APIs when trust zone is disabled

On RT600, ROM APIs are not available only when the trust zone is disabled using OTP setting or image header, and when the user image boots in non-secure mode. TrustZone can be disabled via OTP (BOOT_CFG,

word 96, bits 14:13 = 0b01), or by setting the image header type, bit 14. Under these settings, execution of the ROM API functions will result in the device being locked.

Note: *This errata applies to devices with boot ROM patch revision 6 or greater. The boot ROM rev can be read using the ISP get-property 24 command. If the device responds with "Target Version = T2.0.6" or greater, the device is affected by the ROM API restriction when TrustZone is disabled. See the user manual for further details on the target version number format.*

6 Revision history

Table 5. Revision history

Document ID	Release date	Description
ES_RT600 v. 2.2	09 Nov 2024	Added Section 3.17
ES_RT600 v. 2.1	14 August 2024	• Added Section 3.12 , Section 3.13, Section 3.14, Section 3.16, and Section 3.15
ES_RT600 v. 2.0	18 January 2024	• Added remark to ROM.2 errata,Section 5.2 • Added Section 3.9, Section 3.10, Section 3.11
ES_RT600 v. 1.9	31 March 2022	Added Section 5.2"ROM.2: Usage of ROM APIs when trust zone is disabled"
ES_RT600 v. 1.8	11 November 2021	Added ROM.1 errata, Section 3.8 "ROM.1: Non-secure Boot ROM API initializes unused FlexSPI0 IO pins"
ES_RT600 v. 1.7	20 April 2021	• Added USB.3 errata, Section 3.6 "USB.3: In USB high-speed device mode, 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" • Added USB.4 errata, Section 3.7 "USB.4: In USB high-speed host mode, only one transaction per micro-frame is allowed for isochronous IN endpoints"
ES_RT600 v. 1.6	25 February 2021	Added USB.2 errata, Section 3.5 "USB.2: In USB high-speed device mode, the device writes extra byte(s) to the buffer if the NBytes is not multiple of 8 for OUT transfer"
ES_RT600 v. 1.5	18 December 2020	Includes Section 5.1 "Leakage path between VDD1V8 and VDDIO_x"
ES_RT600 v. 1.4	14 December 2020	Includes Section 3.4 "USB.1: For the USB high-speed device controller, the detection handshaking fails when certain full-speed hubs are connected"
ES_RT600 v. 1.3	29 July 2020	Updates Section 3.2 "GPIO.1: During initial power-up, a brief pull-up pulse could occur on the port pins" and adds Section 3.3 "ADC.1: ADC misses software and hardware triggerswhen there is no ADC clock"
ES_RT600 v. 1.2	6 July 2020	Adds Section 3.2 "GPIO.1: During initial power-up, a brief pull-up pulse could occur on the port pins"
ES_RT600 v. 1.1	8 May 2020	Added FlexSPI DLL lock status timing issue and addressed part marking.
ES_RT600 v. 1.0	13 February 2020	Initial version.

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.

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

1 Product identification 2
2 Errata overview 2
3 Functional problems detail 4
3.1 FlexSPI.1: FlexSPI DLL lock status bit not accurate due to timing issue 4
3.2 GPIO.1: During initial power-up, a brief pull-up pulse could occur on the port pins 4
3.3 ADC.1: ADC misses software and hardware triggers when there is no ADC clock 4
3.4 USB.1: For the USB high-speed device controller, the detection handshaking fails when certain full-speed hubs are connected 5
3.5 USB.2: In USB high-speed device mode, the device writes extra byte(s) to the buffer if the NBytes is not multiple of 8 for OUT transfer 6
3.6 USB.3: In USB high-speed device mode, 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 6
3.7 USB.4: In USB high-speed host mode, only one transaction per micro-frame is allowed for isochronous IN endpoints 6
3.8 ROM.1: Non-secure Boot ROM API initializes unused FlexSPI0 IO pins 7
3.9 ROM.3: Non-secure Boot ROM: SRAM memory address overwritten by boot ROM 7
3.10 ROM.4: BOOT_FAIL_PIN does not function properly 8
3.11 I3C.1: In I2C compatibility controller mode, the read transaction does not terminate correctly 8
3.12 I3C.2: Data loss in transmission with DMA when transfer size larger than Tx FIFO 8
3.13 I3C.3: Data corruption with reception with DMA when receive size is greater than Rx FIFO 9
3.14 I3C.4: I3C private read transfer cannot be performed due to timing issue 10
3.15 TRNG: Errors at a high rate when generating random data independent of entropy delay, core frequency, or core voltage 10
3.16 ROM.5 Non-Secure Boot ROM: CRC checking of fuses covered by CRC5 field cannot be used when KEY_SCRAMBLE_SEED feature is enabled 11
3.17 GPIO.2: 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 11
4 AC/DC deviations detail 12
5 Errata notes detail 12
5.1 Leakage path between VDD1V8 and VDDIO_x 12
5.2 ROM.2: Usage of ROM APIs when trust zone is disabled 12
6 Revision history 13
Legal information 14