STM32H7R3xx STM32H7R7xx
STM32H7S3xx STM32H7S7xx

Errata sheet

STM32H7Rxx/7Sxx device errata

Applicability

This document applies to the part numbers of STM32H7Rxx/7Sxx devices and the device variants as stated in this page.

It gives a summary and a description of the device errata, with respect to the device datasheet and reference manual RM0477.

Deviation of the real device behavior from the intended device behavior is considered to be a device limitation. Deviation of the description in the reference manual or the datasheet from the intended device behavior is considered to be a documentation erratum. The term *"errata"* applies both to limitations and documentation errata.

Table 1. Device summary

Reference	Part numbers
STM32H7R3xx	STM32H7R3A8, STM32H7R3I8, STM32H7R3L8, STM32H7R3N8, STM32H7R3R8, STM32H7R3V8, STM32H7R3Z8
STM32H7R7xx	STM32H7R7A8, STM32H7R7I8, STM32H7R7L8, STM32H7R7Z8
STM32H7S3xx	STM32H7S3A8, STM32H7S3I8, STM32H7S3L8, STM32H7S3R8, STM32H7S3V8, STM32H7S3Z8
STM32H7S7xx	STM32H7S7A8, STM32H7S7I8, STM32H7S7L8, STM32H7S7Z8

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32H7Rxx/Sxx	Y	0x1003

1. *Refer to the device datasheet for how to identify this code on different types of package.*
2. *REV_ID[15:0] bitfield of register.*

1 Summary of device errata

The following table gives a quick reference to the STM32H7Rxx/7Sxx device limitations and their status:

A = limitation present, workaround available

N = limitation present, no workaround available

P = limitation present, partial workaround available

“-” = limitation absent

Applicability of a workaround may depend on specific conditions of target application. Adoption of a workaround may cause restrictions to target application. Workaround for a limitation is deemed partial if it only reduces the rate of occurrence and/or consequences of the limitation, or if it is fully effective for only a subset of instances on the device or in only a subset of operating modes, of the function concerned.

Table 3. Summary of device limitations

Function	Section	Limitation	Status
			Rev. Y
Core	2.1.1	PLD might perform linefill to address that would generate a MemManage Fault	A
	2.1.2	Software programming errors might not be reported for online MBIST access to the ICACHE	N
	2.1.3	ECC error causes data corruption when the data cache error bank registers are locked	A
	2.1.4	Store after cache invalidate without intervening barrier might cause inconsistent memory view	A
System	2.2.1	Boundary scan, PC10 and PC11 are not controllable on TFBGA100 package	P
	2.2.2	Only one configuration available using MCE on the FMC interface	N
	2.2.3	Data read might be corrupted on FMC NOR	A
	2.2.4	Incorrect backup domain reset	P
	2.2.5	SRAM1 AHB limitation if the device is not in OPEN state	N(1)
	2.2.6	LSE crystal oscillator may be disturbed by transitions on PC13	N
	2.2.7	Secure Firmware Install (SFI) is not supported	N(1)
FMC	2.3.1	Dummy read cycles inserted when reading synchronous memories	N
	2.3.2	Wrong data read from a busy NAND memory	A
	2.3.3	Unsupported read access with unaligned address	P
XSPI	2.4.1	Memory-mapped write error response when DQS output is disabled	P
	2.4.2	Deadlock can occur under certain conditions	A
	2.4.3	Memory wrap instruction not enabled when DQS is disabled	N
	2.4.4	Deadlock or write-data corruption after spurious write to a misaligned address in XSPI_AR register	N
	2.4.5	Deadlock on consecutive out-of-range memory-mapped write operations	P
	2.4.6	XSPI deadlock or RAM content corrupted on CSBOUND split during prefetch, when DQS is disabled	A
	2.4.7	Indirect write mode limited to 256 Mbytes	N
	2.4.8	Read-modify-write operation does not clear the MSEL bit	A
	2.4.9	CALMAX bit not set when the PHY reaches the DLL maximum value	N
	2.4.10	Setting the ABORT bit does not generate an error on the AHB bus for undefined-length incremental burst transfers	P
	2.4.11	Read data corruption when a wrap transaction is followed by a linear read to the same MSB address	N

Function	Section	Limitation	Status Rev. Y
XSPI	2.4.12	Transactions are limited to 8 Mbytes in OctaRAM™ memories	N
	2.4.13	Variable latency is not supported when a refresh collision occurs during a write access to some OctaRAM™ memories	P
XSPIM	2.5.1	Certain quad memories may be reset during arbitration while in single-SPI mode	A
SDMMC	2.6.1	Command response and receive data end bits not checked	N
ADC	2.7.1	New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0	A
	2.7.2	Two consecutive context conversions fail when writing new context in ADC_JSQR just after previous context completion with JQDIS = 0 and JQM = 0	A
	2.7.3	Unexpected regular conversion when two consecutive injected conversions are performed in Dual interleaved mode	A
	2.7.4	ADC_AWDy_OUT reset by non-guarded channels	A
	2.7.5	Injected data stored in the wrong ADC_JDRx registers	A
	2.7.6	ADC slave data may be shifted in Dual regular simultaneous mode	A
ADF	2.8.1	In LFM mode ADF_CCK1 clock cannot be selected for SITFx interfaces	A
GPU2D	2.9.1	Occasional writing miss to frame buffer with slow memories	N(1)
SAES	2.10.1	Data transfer from TAMP_BKPxR to key registers must be done only in ascending order when KEYSEL[2:0] is set to 010 or 100	D
LPTIM	2.11.1	Device may remain stuck in LPTIM interrupt when entering Stop mode	A
	2.11.2	ARRM and CMPM flags are not set when APB clock is slower than kernel clock	A
	2.11.3	Interrupt status flag is cleared by hardware upon writing its corresponding bit in LPTIM_DIER register	N
RTC and TAMP	2.12.1	Alarm flag may be repeatedly set when the core is stopped in debug	N
I2C	2.13.1	Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period	P
	2.13.2	Spurious bus error detection in master mode	A
	2.13.3	SDA held low upon SMBus timeout expiry in slave mode	A
I3C	2.14.1	I3C controller: unexpected read data bytes during a legacy I^2C read	A
	2.14.2	I3C controller: SCL clock is not stalled during address ACK/NACK phase following a frame start, when enabled through I3C_TIMINGR2 register	A
	2.14.3	I3C controller: unexpected first frame with a 0x7F address when the I3C peripheral is enabled	A
	2.14.4	I3C controller: no timestamp on IBI acknowledge when timing control is used in Asynchronous mode 0	A
USART	2.15.1	Wrong data received in smartcard mode and 0.5 stop bit configuration	A
	2.15.2	Received data may be corrupted upon clearing the ABREN bit	N
	2.15.3	Noise error flag set while ONEBIT is set	N
LPUART	2.16.1	Possible LPUART transmitter issue when using low BRR[15:0] value	P
SPI	2.17.1	RDY output failure at high serial clock frequency	N
	2.17.2	Truncation of SPI output signals after EOT event	A
	2.17.3	TIFRE flag wrongly set in slave PCM long frame mode if FIXCH = 1	N
	2.17.4	TIFRE flag never set in slave PCM/I2S mode if FIXCH = 0	N
FDCAN	2.18.1	Desynchronization under specific condition with edge filtering enabled	A

Function	Section	Limitation	Status
			Rev. Y
FDCAN	2.18.2	Tx FIFO messages inverted under specific buffer usage and priority setting	A
UCPD	2.19.1	Ordered set with multiple errors in a single K-code is reported as invalid	N
ETH	2.20.1	The MAC does not provide bus access to a higher priority request after a low priority request is serviced	N
	2.20.2	Rx DMA engine may fail to recover upon a restart following a bus error, with Rx timestamping enabled	A
	2.20.3	Tx DMA engine fails to recover correctly or corrupts TSO/USO header data on receiving a bus error response from the AHB DMA slave	N
	2.20.4	Incorrectly weighted round robin arbitration between Tx and Rx DMA channels to access the common host bus	A
	2.20.5	Incorrect L4 inverse filtering results for corrupted packets	N
	2.20.6	IEEE 1588 Timestamp interrupt status bits are incorrectly cleared on write access to the CSR register with similar offset address	A
	2.20.7	Bus error along with Start-of-Packet can corrupt the ongoing transmission of MAC generated packets	N
	2.20.8	Spurious receive watchdog timeout interrupt	A
	2.20.9	Incorrect flexible PPS output interval under specific conditions	A
	2.20.10	Packets dropped in RMII 10 Mbps mode due to fake dribble and CRC error	A
	2.20.11	ARP offload function not effective	A
CEC	2.21.1	Missed CEC messages in normal receiving mode	A
	2.21.2	Unexpected TXERR flag during a message transmission	A

1. *This limitation will be fixed in the next revision*

The following table gives a quick reference to the documentation errata.

Table 4. Summary of device documentation errata

Function	Section	Documentation erratum
FMC	2.3.4	CTB1, CTB2, MODE[2:0] write-only bitfields in FMC_SDCMR incorrectly described as read-write

2 Description of device errata

The following sections describe the errata of the applicable devices with Arm® core and provide workarounds if available. They are grouped by device functions.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

2.1 Core

Reference manual and errata notice for the Arm® Cortex®-M7 core unknown is available from http://infocenter.arm.com.

2.1.1 PLD might perform linefill to address that would generate a MemManage Fault

Description

If the MPU is present and enabled, then it can be programmed so that loads to certain addresses generate a MemManage Fault. This could be because:

- The address is unmapped, that is, it is not in an enabled region and the default memory map is not being used.
- The address cannot be accessed at the current privilege level.
- The address cannot be accessed at any privilege level.

Because of this erratum, a PLD to such an address might incorrectly cause a data cache line-fill.

Conditions:

- The data cache is enabled and the MPU is enabled.
- A PLD is executed, and either:
  - The PLD is to an address not mapped in the MPU, which requires that:
    - The MPU is enabled.
    - The default memory map is not being used.
    - The default memory map is cacheable at that address.
    - The PLD does not hit an enabled MPU region.
  - The PLD is to a region that has permission requirements that the PLD does not meet, which requires that:
    - The MPU is enabled.
    - The default memory map is not being used.
    - The region that the PLD hits is cacheable.
    - The region that the PLD hits would generate a MemManage fault for a load. This requires either:
      - The region cannot be accessed by a read at any privilege level.
      - The region only has read access for privileged code and the PLD is unprivileged.

Note that in rare cases, a PLD instruction can be speculatively executed in the shadow of a mispredicted branch. This can even theoretically be a literal value that decodes to a PLD.

Processor execution is not affected by this erratum. The data returned from the line-fill is not directly consumed by the PLD. Any subsequent load to that address can only access the data if it has permission to do so. This erratum does not permit software to access data that it does not have permissions for.

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

Workaround

Accesses to memory that is not mapped in the MPU can be avoided by using MPU region 0 to cover all unmapped memory and make this region execute-never and inaccessible. That is, MPU_RASR0 must be programmed with:

- The ENABLE bit of the MPU_RASR0 register = 0b1; MPU region 0 enable
- The SIZE bit of the MPU_RASR0 register = 0b11111; MPU region 0 size = 2^32 bytes to cover entire memory
- The SRD bit of the MPU_RASR0 register = 0b0000 0000; All sub-regions enabled
- The XN bit of the MPU_RASR0 register = 0b1; Execute-never to prevent instruction fetch
- The AP bit of the MPU_RASR0 register = 0b000; No read or write access for any privilege level
- The TEX bit of the MPU_RASR0 register = 0b000; Attributes = Strongly-ordered
- The C bit of the MPU_RASR0 register = 0b0; Attributes = Strongly-ordered
- The B bit of the MPU_RASR0 register = 0b0; Attributes = Strongly-ordered

Accesses to memory that is mapped in the MPU, but should not be accessed at the current privilege level can be avoided by making the region noncacheable. That is, MPU_RASR0 should be programmed with:

- The TEX bit of the MPU_RASR0 register = 0b000; Attributes = Strongly-ordered
- The C bit of the MPU_RASR0 register = 0b0; Attributes = Strongly-ordered
- The B bit of the MPU_RASR0 = 0b0; Attributes = Strongly-ordered

2.1.2 Software programming errors might not be reported for online MBIST access to the ICACHE

Description

The online MBIST interface provides access to the cache and TCM RAMs to allow in-field memory testing during normal operation of the processor. Because of this erratum, errors in the software that works with the memory testing might not be indicated on the MBISTERR output signal as intended for ICACHE tests.

Note that this erratum does not affect the detection of faults in the memories under test, but affects only the feature that helps to indicate errors in software used during testing.

There are two online MBIST use cases: software transparent and software assisted.

In the software transparent use case, software running on the processor is not involved in or aware of the memory testing being carried out. See the Cortex®-M7 safety manual for more details. In this case, the target memory is automatically locked by the MBIST controller, which causes the processor pipeline to stall if it attempts to access this memory. Testing is carried out using short bursts of accesses, which last for less than 20 clock cycles and do not corrupt the memory contents. For this reason, the memory is locked only for a very short period of time and the gap between bursts is very large.

In the software-assisted use case, the target memory is still locked by the MBIST controller, but the software running on the processor disables the target memory before testing commences. This prevents any software access to this memory during testing. See the Cortex®-M7 safety manual for more details. For this reason, software accesses go to another memory instead of the target memory and the pipeline does not stall. This is important because the software-assisted use case is intended to be used for production MBIST algorithms, which take a long time to run. For example, if the ICACHE were disabled then software might still execute using the main memory or the TCMs.

This erratum only affects the software-assisted use case, when the ICACHE RAMs are tested. An error indication is sent back to the MBIST controller if software attempts to access the target memory while it is locked for testing. Because of this erratum, an error is not indicated back to the MBIST controller on the MBISTERR[0] output signal when software performs a lookup to the ICACHE during MBIST testing.

The error indication is correctly asserted for all the other types of ICACHE access during MBIST testing:

- A cache line invalidates because of an ECC error.
- A cache invalidates by MVA.
- A cache invalidates all operation.
- A cache line-fill allocation.

Note that this erratum only affects the MBIST software-assisted use case error indication for the ICACHE and the MBISTERR[0] signal functions correctly for the DCACHE, ITCM, and DTCM.

The following conditions are required to cause this erratum:

- The software intends to use the software assisted online MBIST use case.
- The ICACHE is not disabled by software running on the Cortex®-M7 before testing commences.
- The MBIST controller selects an ICACHE memory array for testing, locks the target memory, and testing commences.

This erratum could result in an error not being indicated back to the MBIST controller on the MBISTERR[0] output signal when software assisted use case is used and the ICACHE is not disabled by software before testing commences. This could result in the processor unexpectedly stalling for a long period of time during MBIST testing of the ICACHE memories, without there being a clear indication of the cause of the stall. For this reason, the processor might not make progress as expected, because of the software error, during ICACHE testing.

Workaround

There is no workaround for this erratum.

2.1.3 ECC error causes data corruption when the data cache error bank registers are locked

Description

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

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

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

Conditions:

- DEBR0 and DEBR1 are locked.
- The wanted address has been allocated to the cache.
- A cacheable store to the wanted address looks up in the cache, and an ECC error is found in the cache set that the store addresses.

This erratum can cause data corruption in the data cache.

Workaround

Software must avoid locking both error bank registers.

2.1.4 Store after cache invalidate without intervening barrier might cause inconsistent memory view

Description

If a cache invalidate operation is followed by a write-through store to an address affected by that operation and a line-fill to that address occurs, then the line-fill might allocate to the cache without the data from the store. Subsequently, that store writes to the bus and leaves the cache with stale data.

The following sequence is required for this erratum to occur:

1. The address of interest is in the cache.
2. One of the following data cache maintenance operations that affects the same cache line as the wanted address is performed.
   - DCCIMVAC.
   - DCCISW.
   - DCIMVAC.
   - DCISW.
3. A write-through store is performed to the wanted address
4. A line-fill to the same cache line of the wanted address occurs for any reason.

There must be no DSB or DMB between the maintenance operation and the store.

If this sequence occurs and certain very specific internal timing conditions are met, then the store data is not merged into the line-fill, but it writes out to the bus. After this has occurred, the line-fill buffer or cache contains stale data.

A subsequent load to the same address of the store might observe stale data in the cache.

Workaround

A DMB must be inserted between the cache maintenance operation and the store.

It is expected that all code should already have this DMB or DSB because there is no implicit ordering between cache maintenance operations and stores.

2.2 System

2.2.1 Boundary scan, PC10 and PC11 are not controllable on TFBGA100 package

Description

At board level, when using boundary scan on TFBGA 100 PC10 and PC11 must not be used as they are not controllable, resulting in indeterminate behavior.

Workaround

Mask (ignore) the TFBGA 100 PC10 and PC11 signals when performing boundary scan.

2.2.2 Only one configuration available using MCE on the FMC interface

Description

When using the encryption (MCE) on the FMC interface, only one configuration is available. Each memory connected to the FMC interface inherits the MCE region programming.

Workaround

None.

2.2.3 Data read might be corrupted on FMC NOR

Description

Data read might be corrupted when the write FIFO is disabled.

Workaround

Enable the write FIFO using FMC_WRITE_FIFO_ENABLE variable in the HAL.

2.2.4 Incorrect backup domain reset

Description

The backup domain reset may be missed upon n backup domain power-on following a VBAT power-off in VBAT mode, if the VBAT voltage drops during the power-off phase hitting a window, which is a few mV wide before it starts to rise again. This window is located in the range between 100 mV and 700 mV, the exact position depending mainly on the device and on the temperature.

The missed reset results in unpredictable values of the backup domain registers, which may lead to a wrong device behavior, such as driving the LSCO output pin on PA2, raising an unexpected tamper event preventing the access to SRAM2 and PKA, or influencing any of the backup domain functions.

Workaround

Apply one of the following measures to avoid an incorrect backup domain reset:

- Before performing a new power-on,let the VBAT supply voltage fall to a level below 100 mV for more than 200 ms.
- If none of the previous workarounds can be applied, and the boot follows a backup domain power-on reset, erase the backup domain by software. In order to discriminate the backup domain power-on reset from a power-on reset,at least one backup register (called, for example, Backup Test Register) must be previously programmed with a BKP_REG_VAL value containing 16 bits set and 16 bits cleared. The robustness of this workaround can be significantly improved by using a CRC rather than registers, since the registers are subject to backup domain reset.

The workaround consists in calculating the CRC of the backup domain registers, RCC_BDCR and RTC/ TAMP registers, excluding the bits modified by hardware. The CRC result can be stored in the backup register, instead of a fixed BKP_REG_VAL value. The CRC result needs to be updated for each modification of values covered by the CRC, for example when the CRC peripheral is used.

Insert the following software sequence at the very beginning of the boot code:

1. Check if the BORRSTF flag of the RCC_RSR register is set (the reset is caused by a power-on).
2. If it is set, check that the Backup Test Register content is different from BKP_REG_VAL, or that the new CRC calculated value is different from stored results, depending on the chosen workaround implementation.
3. If this is the case and if no tamper flag is set (when the tamper detection is enabled), the reset is caused by a backup domain power-on. Then apply the following sequence:
   a. Enable backup domain access by setting the DBP bit of the PWR_DBPCR register.
   b. Reset the backup domain by applying the following sequence:
      i. Write 0x00010000 to the RCC_BDCR register, which sets the VSWRST bit and clears the other register bits that may not be cleared.
      ii. Read the RCC_BDCR register to make the reset time long enough.
      iii. Write 0x00000000 to the RCC_BDCR register to clear the VSWRST bit.
   c. Clear the BORRSTF flag by setting the RMVF bit of the RCC_RSR register.

2.2.5 SRAM1 AHB limitation if the device is not in OPEN state

Description

During the RSS processing the SRAM1 AHB is reserved and no longer accessible if the product state is not equal to OPEN.

Workaround

None.

2.2.6 LSE crystal oscillator may be disturbed by transitions on PC13

Description

On LQFP and UFQFPN packages, the LSE crystal oscillator clock frequency can be incorrect when PC13 is toggling as an input or output (for example when used for RTC_OUT1).

The external clock input (LSE bypass) is not impacted by this limitation.

WLCSP and UFBGA packages are not impacted by this limitation.

Workaround

None.
Avoid toggling PC13 when LSE is used on LQFP and UFQFPN packages.

2.2.7 Secure Firmware Install (SFI) is not supported

Description

Secure Firmware Install (SFI) is not supported.

Workaround

None

2.3 FMC

2.3.1 Dummy read cycles inserted when reading synchronous memories

Description

When performing a burst read access from a synchronous memory, two dummy read accesses are performed at the end of the burst cycle whatever the type of burst access.

The extra data values read are not used by the FMC and there is no functional failure.

Workaround

None.

2.3.2 Wrong data read from a busy NAND memory

Description

When a read command is issued to the NAND memory, the R/B signal gets activated upon the de-assertion of the chip select. If a read transaction is pending, the NAND controller might not detect the R/B signal (connected to NWAIT) previously asserted and sample a wrong data. This problem occurs only when the MEMSET timing is configured to 0x00 or when ATTHOLD timing is configured to 0x00 or 0x01.

Workaround

Either configure MEMSET timing to a value greater than 0x00 or ATTHOLD timing to a value greater than 0x01.

2.3.3 Unsupported read access with unaligned address

Description

Read access with unaligned address, such as a half-word read access starting at odd address, is not supported.

Workaround

Compile the software that accesses the fmc region with a compiler option that ensures data alignment, such as –*no_unaligned_access*.

2.3.4 CTB1, CTB2, MODE[2:0] write-only bitfields in FMC_SDCMR incorrectly described as read-write

Description

The CTB1, CTB2, and MODE[2:0] bitfields in FMC_SDCMR are write-only, and always read as zero. Some versions of the device reference manual incorrectly indicate that these bitfields are read-write.

This is a documentation error rather than a device limitation.

Workaround

None.

2.4 XSPI

2.4.1 Memory-mapped write error response when DQS output is disabled

Description

If the DQSE control bit of the XSPI_WCCR register is cleared for memories without DQS pin, it results in an error response for every memory-mapped write request.

Workaround

When doing memory-mapped writes, set the DQSE bit of the XSPI_WCCR register, even for memories that have no DQS pin.

2.4.2 Deadlock can occur under certain conditions

Description

A deadlock can occur when all the following conditions are met:

- The product communicates through an I/O manager in multiplexed mode with an single external memory or an external combo featuring two memories, directly or through a high-speed interface.
- The external memory(ies) is(are) accessed in indirect mode or memory-mapped mode.

The deadlock can happen when the two following conditions occur at the same time:

- The Extended-SPI interface that currently owns the external bus (for example XSPI1) waits for a transfer to occur with the external memory, to complete its transfer on the internal interconnect matrix bus.
- A data transfer request on the internal interconnect matrix bus arrives to the other Extended-SPI interface (for example XSPI2).

This leads to an ownership conflict where:

- XSPI2 cannot get ownership of the external bus which is currently in use by XSPI1.
- XSPI1 cannot get ownership of the internal interconnect matrix bus which is currently in use by XSPI2.

Workaround

Apply one of the following measures:

- If any of the features generating automatic transfer split (MAXTRAN, REFRESH, CSBOUND, TIMEOUT) is set, XSPI1 splits its transfer at some point in time, releasing the bus. XSPI2 can then process its data, and when XSPI1 gets ownership back again, it resumes its transfer thanks to its embedded capability to restart at the address following the last address accessed. In this case, the deadlock is resolved.
  Limitation of the workaround: The automatic resume of the transfer does not work with certain flash memories in write direction only. These memories require an extra "write enable" command before resuming a write transfer. This "write enable" command is not generated by the XSPI.
- The application must ensure that it has sufficient room left in the XSPI internal FIFO for each and every transfer before launching it. The internal interconnect matrix bus activity no longer depends on what happens on external bus side, and the deadlock condition is avoided.

2.4.3 Memory wrap instruction not enabled when DQS is disabled

Description

Memory wrap instruction (as configured in the XSPI_WPxxx registers) is not generated when DQS is disabled. The memory wrap instruction is replaced by two regular successive read instructions to ensure the correct data ordering: this split has very limited impact on performance.

Workaround

None.

2.4.4 Deadlock or write-data corruption after spurious write to a misaligned address in XSPI_AR register

Description

Upon writing a misaligned address to XSPI_AR just before switching to memory-mapped mode (without first triggering the indirect write operation), with the XSPI configured as follows:

- FMODE = 00 in XSPI_CR (indirect write mode)
- DQSE = 1 in XSPI_CCR (DQS active)

then, the XSPI may be deadlocked on the first memory-mapped request or the first memory-mapped write to memory (and any sequential writes after it) may be corrupted.

An address is misaligned if:

- the address is odd and the XSPI is configured to send two bytes of data to the memory every cycle (octal-DTR mode or dual-quad-DTR mode), or
- the address is not a multiple of four when the XSPI is configured to send four bytes of data to the memory (16-bit DTR mode or dual-octal DTR mode).

If the XSPI_AR register is reprogrammed with an aligned address (without triggering the indirect write between the two writes to XSPI register), the data sent to the memory during the indirect write operation are also corrupted.

Workaround

None.

2.4.5 Deadlock on consecutive out-of-range memory-mapped write operations

Description

The DEVSIZE[4:0] bitfield of the XSPI_DCR1 register indicates that the size of the memory is 2 ^ [DEVSIZE + 1] bytes, and thus any memory-mapped access to address 2 ^ [DEVSIZE + 1] or above should get an error response.

However, no error response may be returned and the XSPI may become deadlocked after the following sequence of events:

1. A memory-mapped write operation is ongoing on the AHB bus.
2. A second memory-mapped write is requested to an address close to the end of the memory but not consecutive to the address targeted by the first write operation.
3. A third memory-mapped write operation is requested, this time to an address consecutive to the address targeted by the second write, and the address of this third write is 2 ^ [DEVSIZE + 1] or an address consecutive to 2 ^ [DEVSIZE + 1].
   If the first write command has not completed writing data, then the write to 2 ^ [DEVSIZE + 1] does not return any error response and the next memory-mapped request gets stalled indefinitely.

Workaround

Ensure that no sequences of consecutive memory-mapped write operations pass the memory boundary.

2.4.6 XSPI deadlock or RAM content corrupted on CSBOUND split during prefetch, when DQS is disabled

Description

Depending on the XSPI configuration and sequence of operations, XSPI may be deadlocked after an abort or the RAM content corrupted when a REFRESH, TIMEOUT or MAXTRAN event occurs.

When the XSPI is configured as follows:

- 16-bit DTR mode or dual-octal DTR mode enabled
- DQS input disabled (DQSE bit cleared into XSPI_CCR register)

and the following sequence occurs:

1. The last byte is read from the AMBA interface at an address that is 65-72 bytes before a CSBOUND split.

2. There are no more memory-mapped requests (or no more reads from XSPI_DR register in indirect read mode) for long enough that the prefetch mechanism fills up the FIFO.

then, the issue is encountered if either of the two events occurs:

- An abort is requested before another memory-mapped request is issued.
  In such case, XSPI becomes thoroughly deadlocked, and only a reset enables to recover from deadlock.
- No new memory-mapped requests are issued for so long that the command to the memory can be interrupted (chip select released) due to a REFRESH, TIMEOUT or MAXTRAN event.
  In such case, the chip select is not released and the RAM content may get corrupted (for instance if no refresh is performed).

Workaround

Use the DQS output of the memory (by setting the DQSE bit of the XSPI_CCR register) when using 16-bit memories or octal memories in dual-octal mode.

2.4.7 Indirect write mode limited to 256 Mbytes

Description

In indirect write mode, if the address is greater than 256 Mbytes, the indirect write is not performed at the targeted address, even if it is located inside the allowed memory space configured through the device size (DEVSIZE[4:0] of XSPI_DCR1). Actually, this write operation takes place within the 256-Mbyte memory space, thus corrupting the memory content.

Indirect read operations are not impacted.

Workaround

Indirect write operations have to be performed inside the first 256 Mbytes of the memory space.

2.4.8 Read-modify-write operation does not clear the MSEL bit

Description

When the MSEL bit of the XSPI_CR register is set, it remains set even if the software attempts to clear it by performing a read-modify-write operation.

Workaround

To clear the MSEL bit, clear in a single write access bit 7 and bit 30 of the XSPI_CR register, otherwise, the MSEL bit remains set.

2.4.9 CALMAX bit not set when the PHY reaches the DLL maximum value

Description

The CALMAX bit (bit 31 of the XSPI_CALFCR register) is expected to be set when the PHY reaches the maximum value of the DLL. However, this bit is wrongly cleared at the end of the calibration phase, meaning that it is always read at 0 when the calibration is complete, even if the maximum value of the calibration has been reached.

Workaround

None.

2.4.10 Setting the ABORT bit does not generate an error on the AHB bus for undefined-length incremental burst transfers

Description

An AHB error is expected to be generated when the ABORT bit of the XSPI_CR register is set while a request is ongoing.

Instead, the controller does not trigger any AHB error if the ongoing request is an undefined-length incremental burst AHB transfer.

An AHB error is generated for all other transfer types.

Workaround

When possible, wait for the end of the transfer before setting the ABORT bit.

2.4.11 Read data corruption when a wrap transaction is followed by a linear read to the same MSB address

Description

If a wrap transaction is followed by a linear read having the same MSB start address as the wrap (), then the linear read is wrongly considered as a sequential transaction to the previous one, taking back the prefetched data and causing data corruption.

Notice that for a wrap transaction, the prefetch starts after the last address of the wrap window.

Workaround

As prefetch cannot be disabled, there is no workaround. However, the issue is seldom encountered since wrap operations are mostly initiated by the internal cache to refresh its cacheline. All the other masters must avoid retrieving data by using a linear read access to the same MSB address as the wrap, which has been just completed.

2.4.12 Transactions are limited to 8 Mbytes in OctaRAM™ memories

Description

When the controller is configured in Macronix OctaRAM™ mode, by setting the MTYP[2:0] bitfield of the XSPI_DCR1 register to 011, only 13 bits of row address are decoded and sent to the memory, meaning that only 8 K of 1-Kbyte blocks can be accessed (8 Mbytes).

Workaround

None.

This limitation is not present for PSRAMs or HyperRAM™ memories.

2.4.13 Variable latency is not supported when a refresh collision occurs during a write access to some OctaRAM™ memories

Description

When the memory type (MTYP[2:0] bitfield of the XSPI_CR register) is configured to 0b011 to target an OctaRAM™ memory, the host controller does not support the variable latency requested by the external memory if a refresh collision occurs during the write access. For example, some OctaRAM™ memories, such as ISSI memories, request extra latency cycles for write accesses during refresh collision. In this case, the controller does not sample the DQS input signal during the instruction phase, and cannot detect the extra latency requested by the external memory for the refresh operation.This results in data corruption.

Some OctaRAM™ memories do not request any additional latency for write access during refresh cycles. It is required only when the refresh occurs during a read access. In this case, no issue can be observed.

Workaround

When the application targets an OctaRAM™ memory that requests extra latency cycles for write access during refresh collision, force the fixed latency mode in the configuration register of the external memory. There is no constraint about read access, since both variable and fixed latency modes are supported.

2.5 XSPIM

2.5.1 Certain quad memories may be reset during arbitration while in single-SPI mode

Description

The XSPI I/O manager allows two XSPIs to be mapped on the same I/Os, in which case the XSPIs arbitrate for use of the multiplexed port. This arbitration introduces a glitch on the data lines when the arbitration passes the ownership of the port from one XSPI to the other.

External quad memories, having their asynchronous $\overline{\text{RESET}}$ pin (when selected by default by the memory) multiplexed with an SO data line and operating in single-SPI mode, may be asynchronously reset due to the glitch on the data line when the ownership of the port is transferred.

This problem typically occurs when the memory defaults to operate in single-bit mode and the application reconfigures the memory in quad mode, while arbitrating and transferring the port ownership.

Workaround

Ensure that the ownership of the port does not change while an XSPI is configuring its memory to operate in quad mode:

1. Configure the first memory to quad mode, while clearing MUXEN of XSPIM_CR.
2. Switch the ownership of the port by inverting the MODE bit value in the XSPIM_CR register (even if a glitch is present on the corresponding data line, the reset is applied to the memory that is not yet configured)
3. Configure the second memory to quad mode, while clearing MUXEN of XSPIM_CR.
4. Write the MODE bit to the correct value and set the MUXEN bit of the XSPIM_CR register to come back to the multiplexed mode.

2.6 SDMMC

2.6.1 Command response and receive data end bits not checked

Description

The command response and receive data end bits are not checked by the SDMMC. A reception with only a wrong end bit value is not detected. This does not cause a communication failure since the received command response or data is correct.

Workaround

None.

2.7 ADC

2.7.1 New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0

Description

Once an injected conversion sequence is complete, the queue is consumed and the context changes according to the new ADC_JSQR parameters stored in the queue. This new context is applied for the next injected sequence of conversions.

However, the programming of the new context in ADC_JSQR (change of injected trigger selection and/or trigger polarity) may launch the execution of this context without waiting for the trigger if:

- the queue of context is enabled (JQDIS cleared to 0 in ADC_CFGR), and
- the queue is never empty (JQM cleared to 0 in ADC_CFGR), and
- the injected conversion sequence is complete and no conversion from previous context is ongoing

Workaround

Apply one of the following measures:

- Ignore the first conversion.
- Use a queue of context with JQM = 1.
- Use a queue of context with JQM = 0, only change the conversion sequence but never the trigger selection and the polarity.

2.7.2 Two consecutive context conversions fail when writing new context in ADC_JSQR just after previous context completion with JQDIS = 0 and JQM = 0

Description

When an injected conversion sequence is complete and the queue is consumed, writing a new context in ADC_JSQR just after the completion of the previous context and with a length longer that the previous context, may cause both contexts to fail. The two contexts are considered as one single context. As an example, if the first context contains element 1 and the second context elements 2 and 3, the first context is consumed followed by elements 2 and 3 and element 1 is not executed.

This issue may happen if:

- the queue of context is enabled (JQDIS cleared to 0 in ADC_CFGR), and
- the queue is never empty (JQM cleared to 0 in ADC_CFGR), and
- the length of the new context is longer than the previous one

Workaround

If possible, synchronize the writing of the new context with the reception of the new trigger.

2.7.3 Unexpected regular conversion when two consecutive injected conversions are performed in Dual interleaved mode

Description

In Dual ADC mode, an unexpected regular conversion may start at the end of the second injected conversion without a regular trigger being received, if the second injected conversion starts exactly at the same time than the end of the first injected conversion. This issue may happen in the following conditions:

- two consecutive injected conversions performed in Interleaved simultaneous mode (DUAL[4:0] of ADC_CCR = 0b00011), or
- two consecutive injected conversions from master or slave ADC performed in Interleaved mode (DUAL[4:0]of ADC_CCR = 0b00111)

Workaround

- In Interleaved simultaneous injected mode: make sure the time between two injected conversion triggers is longer than the injected conversion time.
- In Interleaved only mode: perform injected conversions from one single ADC (master or slave), making sure the time between two injected triggers is longer than the injected conversion time.

2.7.4 ADC_AWDy_OUT reset by non-guarded channels

Description

ADC_AWDy_OUT is set when a guarded conversion of a regular or injected channel is outside the programmed thresholds. It is reset after the end of the next guarded conversion that is inside the programmed thresholds.

However, the ADC_AWDy_OUT signal is also reset at the end of conversion of non-guarded channels, both regular and injected.

Workaround

When ADC_AWDy_OUT is enabled, it is recommended to use only the ADC channels that are guarded by a watchdog.
If ADC_AWDy_OUT is used with ADC channels that are not guarded by a watchdog, take only ADC_AWDy_OUT rising edge into account.

2.7.5 Injected data stored in the wrong ADC_JDRx registers

Description

When the AHB clock frequency is higher than the ADC clock frequency after the prescaler is applied (ratio > 10), if a JADSTP command is issued to stop the injected conversion (JADSTP bit set to 1 in ADC_CR register) at the end of an injected conversion, exactly when the data are available, then the injected data are stored in ADC_JDR1 register instead of ADC_JDR2/3/4 registers.

Workaround

Before setting JADSTP bit, check that the JEOS flag is set in ADC_ISR register (end of injected channel sequence).

2.7.6 ADC slave data may be shifted in Dual regular simultaneous mode

Description

In Dual regular simultaneous mode, ADC slave data may be shifted when all the following conditions are met:

- A read operation is performed by one DMA channel,
- OVRMOD = 0 in ADC_CFGR register (Overrrun mode enabled).

Workaround

Apply one of the following measures:

- Set OVRMOD = 1 in ADC_CFGR. This disables ADC_DR register FIFO.
- Use two DMA channels to read data: one for slave and one for master.

2.8 ADF

2.8.1 In LFM mode ADF_CCK1 clock cannot be selected for SITFx interfaces

Description

In low-frequency master (LFM) mode, the input clock selectors of the SITFx serial interfaces do not allow selecting the ADF_CCK1 clock. The STIFx clock selector is controlled through the SCKSRC[1:0] bitfield of the corresponding ADF_SITFxCR register. The following table shows the expected and the actual behavior of the device:

Table 5. **SITFx clock selector operation**

SCKSRC[1:0]	Clock selected	
	Expected	Actual
00	ADF_CCK0	ADF_CCK0
01	ADF_CCK1	ADF_CCK0
1x	Reserved	Reserved

As in the LFM mode the ADF_CCK1 cannot be selected and the ADF_CKIx is disabled, ADF_CCK0 is the only applicable clock for the SITFx interfaces.

Workaround

Always enable the ADF_CCK0 clock (by setting the CCK0EN bit of the ADF_CKGCR register) and select it for the SITFx interfaces, even in applications that only use the ADF_CCK1 clock output on the I/Os.

Note: As the ADF_CCK1 and ADF_CCK0 clocks originate from the same clock source, the use of ADF_CCK1 for clocking external microphones and ADF_CCK0 for clocking the SITFx interfaces does not compromise the performance.

2.9 GPU2D

2.9.1 Occasional writing miss to frame buffer with slow memories

Description

The GPU operating slowly due to slow memories may occasionally fail to write isolated single-pixel data to the frame buffer.

Workaround

None

2.10 SAES

2.10.1 Data transfer from TAMP_BKPxR to key registers must be done only in ascending order when KEYSEL[2:0] is set to 010 or 100

Description

The KEYSEL[2:0] bitfield of the SAES_CR register defines the source of the key information to use in the SAES cryptographic core:

- When KEYSEL[2:0] is set to 010, the boot hardware key (BHK), stored in tamper-resistant secure backup registers, is entirely transferred into the key registers upon a secure application performing a single read of all TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE = 0, x = 0 to 7 for KEYSIZE = 1).
- When KEYSEL[2:0] is set to 100, the XOR combination of DHUK and BHK is entirely transferred into the key registers upon a secure application performing a single read of all TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE = 0, x = 0 to 7 for KEYSIZE = 1).

Some revisions of the reference manual may wrongly specify that the read operation can be performed either in ascending or descending order, while it must be performed always in **ascending** order.
This is a documentation issue rather than a product limitation.

Workaround

No application workaround is required, provided that the read operation to the TAMP_BKPxR registers is always done in ascending order.

2.11 LPTIM

2.11.1 Device may remain stuck in LPTIM interrupt when entering Stop mode

Description

This limitation occurs when disabling the low-power timer (LPTIM).
When the user application clears the ENABLE bit in the LPTIM_CR register within a small time window around one LPTIM interrupt occurrence, then the LPTIM interrupt signal used to wake up the device from Stop mode may be frozen in active state. Consequently, when trying to enter Stop mode, this limitation prevents the device from entering low-power mode and the firmware remains stuck in the LPTIM interrupt routine.
This limitation applies to all Stop modes and to all instances of the LPTIM. Note that the occurrence of this issue is very low.

Workaround

In order to disable a low power timer (LPTIMx) peripheral, do not clear its ENABLE bit in its respective LPTIM_CR register. Instead, reset the whole LPTIMx peripheral via the RCC controller by setting and resetting its respective LPTIMxRST bit in the relevant RCC register.

2.11.2 ARRM and CMPM flags are not set when APB clock is slower than kernel clock

Description

When LPTIM is configured in one shot mode and APB clock is lower than kernel clock, there is a chance that ARRM and CMPM flags are not set at the end of the counting cycle defined by the repetition value REP[7:0]. This issue can only occur when the repetition counter is configured with an odd repetition value.

Workaround

To avoid this issue the following formula must be respected:

{ARR, CMP} ≥ KER_CLK / (2* APB_CLK),

where APB_CLK is the LPTIM APB clock frequency, and KER_CLK is the LPTIM kernel clock frequency. ARR and CMP are expressed in decimal value.

Example: The following example illustrates a configuration where the issue can occur:

- APB clock source (MSI) = 1 MHz , Kernel clock source (HSI) = 16 MHz
- Repetition counter is set with REP[7:0] = 0x3 (odd value)

The above example is subject to issue, unless the user respects:

{CMP, ARR} ≥ 16 MHz / (2 * 1 MHz)

→ ARR must be ≥ 8 and CMP must be ≥ 8

Note: REP set to 0x3 means that effective repetition is REP+1 (= 4) but the user must consider the parity of the value loaded in LPTIM_RCR register (=3, odd) to assess the risk of issue.

2.11.3 Interrupt status flag is cleared by hardware upon writing its corresponding bit in LPTIM_DIER register

Description

When any interrupt bit of the LPTIM_DIER register is modified, the corresponding flag of the LPTIM_ISR register is cleared by hardware.

Workaround

None.

2.12 RTC and TAMP

2.12.1 Alarm flag may be repeatedly set when the core is stopped in debug

Description

When the core is stopped in debug mode, the clock is supplied to subsecond RTC alarm downcounter even when the device is configured to stop the RTC in debug.

As a consequence, when the subsecond counter is used for alarm condition (the MASKSS[3:0] bitfield of the RTC_ALRMASSR and/or RTC_ALRMBSSR register set to a non-zero value) and the alarm condition is met just before entering a breakpoint or printf, the ALRAF and/or ALRBF flag of the RTC_SR register is repeatedly set by hardware during the breakpoint or printf, which makes any attempt to clear the flag(s) ineffective.

Workaround

None.

2.13 I2C

2.13.1 Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period

Description

The I^2C-bus specification and user manual specify a minimum data setup time ($t_{SU;DAT}$) as:

- 250 ns in Standard mode
- 100 ns in Fast mode
- 50 ns in Fast mode Plus

The device does not correctly sample the I^2C-bus SDA line when $t_{SU;DAT}$ is smaller than one I2C kernel clock (I^2C-bus peripheral clock) period: the previous SDA value is sampled instead of the current one. This can result in a wrong receipt of slave address, data byte, or acknowledge bit.

Workaround

Increase the I2C kernel clock frequency to get I2C kernel clock period within the transmitter minimum data setup time. Alternatively, increase transmitter's minimum data setup time. If the transmitter setup time minimum value corresponds to the minimum value provided in the I^2C-bus standard, the minimum I2CCLK frequencies are as follows:

- In Standard mode, if the transmitter minimum setup time is 250 ns, the I2CCLK frequency must be at least 4 MHz.
- In Fast mode, if the transmitter minimum setup time is 100 ns, the I2CCLK frequency must be at least 10 MHz.
- In Fast-mode Plus, if the transmitter minimum setup time is 50 ns, the I2CCLK frequency must be at least 20 MHz.

2.13.2 Spurious bus error detection in master mode

Description

In master mode, a bus error can be detected spuriously, with the consequence of setting the BERR flag of the I2C_SR register and generating bus error interrupt if such interrupt is enabled. Detection of bus error has no effect on the I^2C-bus transfer in master mode and any such transfer continues normally.

Workaround

If a bus error interrupt is generated in master mode, the BERR flag must be cleared by software. No other action is required and the ongoing transfer can be handled normally.

2.13.3 SDA held low upon SMBus timeout expiry in slave mode

Description

For the slave mode, the SMBus specification defines $t_{TIMEOUT}$ (detect clock low timeout) and $t_{LOW:SEXT}$ (cumulative clock low extend time) timeouts. When one of them expires while the I2C peripheral in slave mode drives SDA low to acknowledge either its address or a data transmitted by the master, the device is expected to report such an expiry and release the SDA line.

However, although the device duly reports the timeout expiry, it fails to release SDA. This stalls the I^2C bus and prevents the master from generating RESTART or STOP condition.

Workaround

When a timeout is reported in slave mode (TIMEOUT bit of the I2C_ISR register is set), apply this sequence:

1. Wait until the frame is expected to end.
2. Read the STOPF bit of the I2C_ISR register. If it is low, reset the I2C kernel by clearing the PE bit of the I2C_CR1 register.
3. Wait for at least three APB clock cycles before enabling again the I2C peripheral.

2.14 I3C

2.14.1 I3C controller: unexpected read data bytes during a legacy I²C read

Description

Under specific conditions, unexpected data bytes are read during a legacy I²C read transfer.

The issue occurs when all the following conditions are met:

- I3C acts as controller
- a legacy I²C read message is generated
- the STALLT bit of I3C_TIMINGR2 register is set to request the SCL clock to be stalled at low level on the 9th T-bit phase of data bytes (also known as ACK/NACK phase)
- instead of releasing the SDA line, the I²C target incorrectly drives SDA low on the 9th T-bit phase of the end of read from the I3C controller

To end a legacy I²C read, the I3C controller is supposed not to drive SDA low on the 9th T-bit, and to emit a NACK. If the STALLT bit of I3C_TIMINGR2 is set, the controller does not NACK for the purpose of ending the data read transfer.

During the same clock cycle, if the I²C target, instead of releasing the SDA line, incorrectly drives SDA low on this 9th T-bit phase of the end of read from the controller, then the controller detects an incorrect ACK on the I3C bus and keeps SCL clock running.

After 8 clock cycles, the I3C controller generates again an ACK instead of a NACK, and an unexpected dummy data byte is transferred to the RX-FIFO.

Then the target continues transferring data or releases the SDA line, thus causing additional dummy bytes to be received. The transfer can be stopped only when an overrun error occurs.

Workaround

Apply the following measures:

- If the I3C controller is configured with S-FIFO mode enabled (SMODE bit set in I3C_CFGR), the transfer goes on until RX-FIFO is full. Then ERRF = 1 in I3C_EVR (an error occurred), PERR = 1 in I3C_SER (protocol error), DOVR = 1 in I3C_SER (RX-FIFO overrun), and CODERR[3:0] = 001 in I3C_SER (CE1 error).
  It is recommended to enable the error interrupt by setting ERRIE in I3C_IER. When DOVR = 1 and CODERR[3:0] = 0001, flush the RX-FIFO inside the error interrupt service routine by setting RXFLUSH in I3C_CFGR, then clear the CERRF error flag.
- If the I3C controller is configured with S-FIFO mode disabled (SMODE bit cleared in I3C_CFGR), the I3C status register (I3C_SR) may be overwritten by the hardware if unread, thus failing to report any status overrun. An overrun can occur only as a data overrun if the DMA or the software stops reading the RX-FIFO during enough time for the RX-FIFO to be full with dummy bytes. Then both CE1 and DOVR flags are set and an error is reported (ERRF = 1, PERR = 1, CODERR[3:0] = 0001 and DOVR = 1).

Whatever S-FIFO configuration, implement a software timeout to inform that neither FCF nor ERRF error bit was raised during an acceptable time. Then, stop reading RX-FIFO to cause a data overrun to be reported. When an error is reported, if both CE1 and DOVR flags are set (ERRF = 1, PERR = 1, CODERR[3:0] = 0001 and DOVR = 1), flush the RX-FIFO.

2.14.2 I3C controller: SCL clock is not stalled during address ACK/NACK phase following a frame start, when enabled through I3C_TIMINGR2 register

Description

Under specific conditions, the I3C controller does not stall the SCL clock during the address ACK/NACK phase when this feature is configured through I3C_TIMINGR2 register.

The issue occurs when all the following conditions are met:

- I3C acts as controller
- I3C is programmed to stall the SCL clock low during the address ACK/NACK phase (STALLA bit of I3C_TIMINGR2 set to 1 and STALL[7:0] bitfield of I3C_TIMINGR2 set to a non-null value)
- the address emitted by the controller follows a frame start and not a repeated start

The purpose of this programmed SCL clock stall time is to add an additional duration for the I3C target(s) to respond on the address ACK/NACK phase. However, the SCL clock is not stalled on this address ACK/NACK phase.

Workaround

Set NOARBH = 0 in I3C_CFGR in order to insert the arbitrable header between the frame start and the emitted address.

If the I^2C/I3C target has still not enough time to respond to the emitted static/dynamic address, increase the SCL low duration for any open-drain phase by increasing SCLL_OD[7:0] value in I3C_TIMINGR0.

2.14.3 I3C controller: unexpected first frame with a 0x7F address when the I3C peripheral is enabled

Description

After I3C has been initialized as controller, an unexpected frame is generated when the I3C peripheral is enabled. The issue occurs after the following sequence:

1. I3C is initialized as I3C controller (CRINIT bit is set in I3C_CFGR whereas EN bit is kept cleared in I3C_CFGR).
2. I3C is enabled (EN bit set in I3C_CFGR).

As a result, the I3C controller can incorrectly detect that the SDA line has been driven low by a target, interpret it as a start request, activate the SCL clock, and generate a 0x7F address followed by RNW bit = 1 that is not acknowledged.

This first frame completes without any other impact than this unexpected I3C bus activity.

Workaround

Respect the sequence below during I3C controller initialization:

1. Instead of configuring the alternate GPIO of the SDA line without any pull-up, temporary enable the GPIO pull-up.
2. After a delay of 1 ms, disable GPIO pull-up.
3. Initialize I3C as I3C controller by setting CRINIT in I3C_CFGR whereas EN bit is kept cleared in I3C_CFGR.
4. Enable I3C by setting EN bit in I3C_CFGR.

As a result the I3C controller does not detect SDA low when it is enabled, and no unexpected frame is generated.

2.14.4 I3C controller: no timestamp on IBI acknowledge when timing control is used in Asynchronous mode 0

Description

When I3C acts as controller, it cannot provide a timestamp on an IBI acknowledge (named C_REF in MIPI I3C v1.1 specification).

As a result, when timing control is used in Asynchronous mode 0, the controller software cannot calculate the timestamp of the sampled data of the target(s) following a received and acknowledged IBI using payload data for timing control (T_C1 and T_C2) (see MIPI formula: $\mathrm{C\_TS} = \mathrm{C\_REF} - \mathrm{C\_C2} \times \mathrm{T\_C1} / \mathrm{T\_C2}$), despite the fact that the controller software can compute the duration C_C2 by using the formula:

$$\mathrm{C\_C2} = 9 \times (\mathrm{I3C\_TIMINGR0.SCLL\_PP[7{:}0]} + 1 + \mathrm{I3C\_TIMINGR0.SCLH\_I3C[7{:}0]} + 1) \times T_{I3CCLK}$$

When operating in Asynchronous mode 0, the sampled data received from the target(s) cannot be associated with a computed timestamp, and on controller side, they can not be time-correlated.

Workaround

Follow the sequence below:

1. Allocate an available product timer by software and approximate the IBI acknowledge moment by when the timer is notified by an interrupt of a received and complete IBI.
2. Program a broadcast/direct SETXTIME CCC with subcommand byte 0xDF to enter Asynchronous mode 0.
3. After being notified of the command completion by the flag and/or the related interrupt (FCF flag is set in I3C_EVR), reset and enable the timer to start the counter.
4. After being notified that an IBI is complete by the flag and/or the related interrupt (IBIF flag is set in I3C_EVR), read the value of the timer as C_TIM. The timestamp of the sampled data can then be approximated by using the formula:
   $$\mathrm{C\_TS} = \mathrm{C\_TIM} - \mathrm{C\_C2} \times (\mathrm{T\_C1}/\mathrm{T\_C2} + 4)$$
   knowing that
   $$\mathrm{C\_C2} = 9 \times (\mathrm{I3C\_TIMINGR0.SCLL\_PP[7{:}0]} + 1 + \mathrm{I3C\_TIMINGR0.SCLH\_I3C[7{:}0]} + 1) \times \mathrm{T}_{I3CCLK}$$
   and that the IBI is complete after a 4-byte payload.
5. Generate a broadcast/direct SETXTIME CCC with subcommand byte 0xFF to exit Asynchronous mode 0 to disable/deallocate the timer resource.

2.15 USART

2.15.1 Wrong data received in smartcard mode and 0.5 stop bit configuration

Description

The USART receiver reads wrong data in smartcard mode and 0.5 stop bit configuration.

Workaround

Use the 1.5 stop bit configuration.

2.15.2 Received data may be corrupted upon clearing the ABREN bit

Description

The USART receiver may miss data or receive corrupted data when the auto baud rate feature is disabled by software (ABREN bit cleared in the USART_CR2 register) after an auto baud rate detection, while a reception is ongoing.

Workaround

Do not clear the ABREN bit.

2.15.3 Noise error flag set while ONEBIT is set

Description

When the ONEBIT bit is set in the USART_CR3 register (one sample bit method is used), the noise error (NE) flag must remain cleared. Instead, this flag is set upon noise detection on the START bit.

Workaround

None.

Note: Having noise on the START bit is contradictory with the fact that the one sample bit method is used in a noise free environment.

2.16 LPUART

2.16.1 Possible LPUART transmitter issue when using low BRR[15:0] value

Description

The LPUART transmitter bit length sequence is not reset between consecutive bytes, which could result in a jitter that cannot be handled by the receiver device. As a result, depending on the receiver device bit sampling sequence, a desynchronization between the LPUART transmitter and the receiver device may occur resulting in data corruption on the receiver side.

This happens when the ratio between the LPUART kernel clock and the baud rate programmed in the LPUART_BRR register (BRR[15:0]) is not an integer, and is in the three to four range. A typical example is when the 32.768 kHz clock is used as kernel clock and the baud rate is equal to 9600 baud, resulting in a ratio of 3.41.

Workaround

Apply one of the following measures:

- On the transmitter side, increase the ratio between the LPUART kernel clock and the baud rate. To do so:
  - Increase the LPUART kernel clock frequency, or
  - Decrease the baud rate.
- On the receiver side, generate the baud rate by using a higher frequency and applying oversampling techniques if supported.

2.17 SPI

2.17.1 RDY output failure at high serial clock frequency

Description

When acting as slave with RDY alternate function enabled through setting the RDIOM bit of the SPI_CFG2 register, the device may fail to indicate its *Not ready* status in time through the RDY output signal to suspend communication. This may then lead to data overrun and/or underrun on the device side. The failure occurs when the serial clock frequency exceeds:

- twice the APB clock frequency, with data sizes from 8 to 15 bits
- six times the APB clock frequency, with data sizes from 16 to 23 bits
- fourteen times the APB clock frequency, with data sizes from 24 to 32 bits

Workaround

None.

2.17.2 Truncation of SPI output signals after EOT event

Description

After an EOT event signaling the end of a non-zero transfer size transaction (TSIZE > 0) upon sampling the last data bit, the software may disable the SPI peripheral. As expected, disabling SPI deactivates the SPI outputs (SCK, MOSI and SS when the SPI operates as a master, MISO when as a slave), by making them float or statically output their by-default levels, according to the AFCNTR bit of the SPI_CFG2 register.

With fast software execution (high PCLK frequency) and slow SPI (low SCK frequency), the SPI disable occurring too fast may result in truncating the SPI output signals. For example, the device operating as a master then generates an asymmetric last SCK pulse (with CPHA = 0), which may prevent the correct last data bit reception by the other node involved in the communication.

Workaround

Apply one of the following measures or their combination:

- Add a delay between the EOT event and SPI disable action.
- Decrease the ratio between PCLK and SCK frequencies.

2.17.3 TIFRE flag wrongly set in slave PCM long frame mode if FIXCH = 1

Description

When FIXCH = 1, the flag TIFRE indicates an error when channel length indicated by WS does not last as expected. In slave PCM long frame mode, TIFRE is wrongly set, indicating a frame error even if it did not occur.

This issue occurs when all the following conditions are met:

- I2SMOD[1:0] = 1 (I2S/PCM mode) in the SPI_I2SCFGR register
- I2SCFG[2:0] = 000 or 001 or 100 (slave modes) in the SPI_I2SCFGR register
- I2SSTD[1:0] = 11 (PCM) and PCMSYNC=1 (PCM long) in the SPI_I2SCFGR register
- FIXCH[1:0] = 1 (channel length given by CHLEN) in the SPI_I2SCFGR register

Workaround

None. Ignore the TIFRE flag.

2.17.4 TIFRE flag never set in slave PCM/I2S mode if FIXCH = 0

Description

When FIXCH = 0, the TIFRE flag in the SPI_SR register is set to indicate a frame error if a new frame synchronization is received while the shift-in or shift-out of the previous data is not complete (early frame error). Instead, this flag is not set, and no frame error is detected.

This issue occurs when all the following conditions are met:

- I2SMOD[1:0] = 1 (I2S/PCM mode) in the SPI_I2SCFGR register
- I2SCFG[2:0] = 000 or 001 or 100 (slave modes) in the SPI_I2SCFGR register
- FIXCH[1:0] = 0 (CHLEN different from 16 or 32) in the SPI_I2SCFGR register

Workaround

None. Ignore the TIFRE flag.

2.18 FDCAN

2.18.1 Desynchronization under specific condition with edge filtering enabled

Description

FDCAN may desynchronize and incorrectly receive the first bit of the frame if:

- the edge filtering is enabled (the EFBI bit of the FDCAN_CCCR register is set), and
- the end of the integration phase coincides with a falling edge detected on the FDCAN_Rx input pin

If this occurs, the CRC detects that the first bit of the received frame is incorrect, flags the received frame as faulty and responds with an error frame.

Note: This issue does not affect the reception of standard frames.

Workaround

Disable edge filtering or wait for frame retransmission.

2.18.2 Tx FIFO messages inverted under specific buffer usage and priority setting

Description

Two consecutive messages from the Tx FIFO may be inverted in the transmit sequence if:

- FDCAN uses both a dedicated Tx buffer and a Tx FIFO (the TFQM bit of the FDCAN_TXBC register is cleared), and
- the messages contained in the Tx buffer have a higher internal CAN priority than the messages in the Tx FIFO.

Workaround

Apply one of the following measures:

- Ensure that only one Tx FIFO element is pending for transmission at any time:
  The Tx FIFO elements may be filled at any time with messages to be transmitted, but their transmission requests are handled separately. Each time a Tx FIFO transmission has completed and the Tx FIFO gets empty (TFE bit of FDACN_IR set to 1) the next Tx FIFO element is requested.
- Use only a Tx FIFO:
  Send both messages from a Tx FIFO, including the message with the higher priority. This message has to wait until the preceding messages in the Tx FIFO have been sent.
- Use two dedicated Tx buffers (for example, use Tx buffer 4 and 5 instead of the Tx FIFO). The following pseudo-code replaces the function in charge of filling the Tx FIFO:

```
Write message to Tx Buffer 4
Transmit Loop:
    Request Tx Buffer 4 - write AR4 bit in FDCAN_TXBAR
    Write message to Tx Buffer 5
    Wait until transmission of Tx Buffer 4 complete (IR bit in FDCAN_IR),
    read TO4 bit in FDCAN_TXBTO
    Request Tx Buffer 5 - write AR5 bit of FDCAN_TXBAR
    Write message to Tx Buffer 4
    Wait until transmission of Tx Buffer 5 complete (IR bit in FDCAN_IR),
    read TO5 bit in FDCAN_TXBTO
```

2.19 UCPD

2.19.1 Ordered set with multiple errors in a single K-code is reported as invalid

Description

The Power Delivery standard allows considering a received ordered set as valid even if it contains errors, provided that they only affect a single K-code of the ordered set.

In the reference manual, the RXSOP3OF4 flag is specified to signal errors affecting a single K-code, the RXERR flag to signal errors in multiple K-codes.

However, the behaviour does not conform with the reference manual. The RXSOP3OF4 flag is only raised in the case of a single error. The RXERR flag is raised in the case of multiple errors, regardless of whether they affect a single K-code or multiple K-codes. As a consequence, ordered sets with multiple errors in a single K-code are reported by the device as invalid although the Power Delivery standard allows considering them as valid.

Despite this non-conformity versus its reference manual, the device remains compliant with the Power Delivery standard.

Workaround

None.

2.20 ETH

2.20.1 The MAC does not provide bus access to a higher priority request after a low priority request is serviced

Description

The ETH_DMAMR DMA mode register in the MAC can be programmed to arbitrate between the DMA channels to access the system bus:

- Use a weighted round robin (WRR) algorithm for selecting between transmit or receive DMA channels by clearing DA bit
- Give higher priority to transmit or receive DMA channels by programming the TXPR bit of the ETH_DMAMR register
- Select the priority ratio of TX over RX or vice versa (as per TXPR) by programming the PR[2:0] field

For the WRR algorithm, the MAC provides bus access to a higher priority request provided it is within the priority ratio. It services a lower priority request only when higher priority requests have been serviced as per priority ratio or when there are no higher priority requests.

However, in the WRR algorithm operation, when there are requests pending from both Tx DMA engine and Rx DMA engine after a lower priority request gets serviced, the MAC incorrectly selects the lower priority request, thus violating the PR ratio. The MAC continues to service all the subsequent low priority requests until there are no low priority requests, before servicing any high priority request.

This results in a delay in servicing the higher priority requests. If the high priority request is programmed for receive DMA channels (TXPR is cleared), the receive queue can overflow with a resulting loss of packets. If the high priority request is programmed for transmit DMA (TXPR is set) channels, the transmit queue can get starved in store and forward mode resulting in low throughput. Otherwise when operating in threshold mode, the transmit queue can underflow, resulting in discarding of packet by remote end. In both cases the quality of service or throughput may be affected.

Also, when priority ratio of 8:1 is programmed, the serviced request count rolls over to 0 after reaching 7 and does not reach maximum value which is 8. So, if the higher priority request is being serviced, lower priority request does not get serviced until there is no higher priority request.

These issues do not affect the functionality but impacts the performance.

Workaround

None.

2.20.2 Rx DMA engine may fail to recover upon a restart following a bus error, with Rx timestamping enabled

Description

When the timestamping of the Rx packets is enabled, some or all of the received packets can have an Rx timestamp which is written into a descriptor upon the completion of the Rx packet/status transfer.

However, when a bus error occurs during the descriptor read (that is subsequently used as context descriptor to update the Rx timestamp), the context descriptor write is skipped by the DMA engine. Also, the Rx DMA engine does not flush the Rx timestamp stored in the intermediate buffers during the error recovery process and enters stop state. Due to this residual timestamp in the intermediate buffer remaining after the restart, the Rx DMA engine does not transfer any packets.

Workaround

Issue a soft reset to drop all Tx packets and Rx packets present inside the controller at the time of a bus error. After the soft reset, reconfigure the controller and re-create the descriptors.

Note: The workaround introduces additional latency.

2.20.3 Tx DMA engine fails to recover correctly or corrupts TSO/USO header data on receiving a bus error response from the AHB DMA slave

Description

When a bus error is received from the AHB DMA slave, the controller generates an interrupt by setting the FBE bit of the ETH_DMACSR register. This stops the corresponding DMA channel by resetting the ST bit of the ETH_DMACTXCR register after recovering from the error. The software recreates the list of descriptors and restarts the DMA engine by setting the ST bit 0 of the ETH_DMACTXCR register without issuing the software reset to the controller.

However, the Tx DMA engine fails to recover or corrupts the TSO/USO header data when the TSO/USO segmentation is enabled in the Tx Descriptor and if either:

- a bus error is detected while transferring the header data from the system memory
- a bus error occurs for the intermediate beat transfer of the header data

In this case the first packet (with TSO/USO enabled after re-starts) gets corrupted after the DMA engine restarts.

Workaround

Issue a soft reset to recover from this scenario. Issuing a soft reset results in loss of all Tx packets and Rx packets present inside the controller at the time of bus-error. Also, the software must reconfigure the controller and re-create the descriptors. This is an overhead which introduces additional latency.

2.20.4 Incorrectly weighted round robin arbitration between Tx and Rx DMA channels to access the common host bus

Description

The Ethernet peripheral has independent transmit (Tx) and receive (Rx) DMA engines. The transaction requests from the Tx and Rx DMA engines are arbitrated to allow access to the common DMA master interface. The following two types of arbitrations are supported by programming Bit DA of the ETH_DMAMR register:

- Weighted round-robin arbitration
- Fixed-priority arbitration

The PR[2:0] bit field controls the ratio of the weights between the Tx DMA and Rx DMA engines in the weighted round robin scheme.

However, the programmed polarity ratio PR[2:0] in the weighted round-robin scheme is not adhered to, when there is a priority difference between Rx and Tx. In other words when Rx DMA engine is given higher priority over Tx DMA engine or vice-versa.

The defect occurs in the following conditions:

- The weighted round robin arbitration scheme is selected by clearing the DA bit of the ETH_DMAMR
- Programming different weights in the TXPR and PR fields of ETH_DMAMR
- Both Tx and Rx DMA engines are simultaneously requesting for access.

As a consequence, the expected quality of service (QoS) requirement between Tx and Rx DMA channels for host bus bandwidth allocation might not get adhered to. This defect might have an impact only if the host bus bandwidth is limited and close to or above the total Ethernet line rate traffic. The impact can be in terms of buffer underflow (for Tx in cut-through mode) or Buffer overflows (for Rx). If the host side bandwidth is much more than the Ethernet line rate traffic, then this bandwidth allocation of WRR scheme is of no consequence.

Workaround

Operate in fixed priority arbitration mode where the DA bit of the ETH_DMAMR is set with Rx DMA engine having a higher priority over Tx clearing the TXPR bit. Operate the Tx buffers in Store-and-Forward mode to avoid any buffer underflows/overflows.

2.20.5 Incorrect L4 inverse filtering results for corrupted packets

Description

Received corrupted IP packets with payload (for IPv4) or total (IPv6) length of less than two bytes for L4 source port (SP) filtering or less than four bytes for L4 destination port (DP) filtering are expected to cause a mismatch. However, the inverse filtering unduly flags a match and the corrupted packets are forwarded to the software application. The L4 stack gets incomplete packet and drops it.

Note: The perfect filtering correctly reports a mismatch.

Workaround

None.

2.20.6 IEEE 1588 Timestamp interrupt status bits are incorrectly cleared on write access to the CSR register with similar offset address

Description

When RCWE bit of the ETH_MACCSRSWCR register is set, all interrupt status bits (events) are cleared only when the specific status bits are set.

However, the status bits[3:0] of the ETH_MACTSSR register at address 0x0B20 are unintentionally cleared when 1 is written to the corresponding bit positions in any CSR register with address offset [7:0] = 0x20. The Status bits[3:0] correspond to the following events:

- Timestamp seconds register overflow interrupt TSSOVF
- Auxiliary timestamp trigger snapshot AUXTSTRIG
- Target time interrupt TSTARGT0
- Target time programming error interrupt TSTRGTERR0

This defect occurs only when the software enables the write 1 to clear interrupt status bits, by setting RCWE of the ETH_MACCSRSWCR register.

As a consequence, when any of the target time interrupts or timestamp seconds overflow events occur, the software might inadvertently clear the corresponding status bits and as a concequence de-assert the interrupt, if it first writes to any CSR register at the shadow address (0x0_xx20 or 0x1_xx20). Consequently, the interrupt service routine might not identify the source of these interrupt events, as the corresponding status bits are already cleared.

Note: The timestamp seconds register overflow event is extremely rare (once in ~137 years) and the target time error interrupt can be avoided by appropriate programming. The frequency of target time reached interrupt events depends on the application usage.

Workaround

When RCWE is set and the timestamp event interrupts are enabled, process and clear the MAC timestamp interrupt events first in the interrupt service routine software, so that write operations to other shadow CSR registers are avoided.

2.20.7 Bus error along with Start-of-Packet can corrupt the ongoing transmission of MAC generated packets

Description

If a bus error is asserted along with the start of a new packet while the MAC is transmitting an internally generated packet such as: ARP, PTO or Pause, the error indication aborts the ongoing transmission prematurely and corrupts the MAC generated packet being transmitted.

As a consequence, the MAC generated packet is sent on the line as a runt frame with corrupted FCS. The aborted packet is not retransmitted and can cause:

- Failure of the intended flow control in case of a Pause/PFC packet corruption.
- Delay in ARP handshake from ARP offload engine; the ARP stack recovers because it sends ARP requests periodically
- Delay in PTP response/SYNC packets generated by PTP offload engine; the PTP stack recovers because it sends request packets periodically.

The probability of occurrence of an bus error on the first beat of data and coinciding with a MAC generated packet transmission is very low.

Workaround

None.

2.20.8 Spurious receive watchdog timeout interrupt

Description

Setting the RWTU[1:0] bitfield of the ETH_DMACRXIWTR register to a non-zero value while the RWT[7:0] bitfield is at zero leads to a spurious receive watchdog timeout interrupt (if enabled) and, as a consequence, to executing an unnecessary interrupt service routine with no packets to process.

Workaround

Ensure that the RWTU[1:0] bitfield is not set to a non-zero value while the RWT[7:0] bitfield is at zero. For setting RWT[7:0] and RWTU[1:0] bitfields each to a non-zero value, perform two successive writes. The first is either a byte-wide write to the byte containing the RWT[7:0] bitfield, or a 32-bit write that only sets the RWT[7:0] bitfield and keeps the RWTU[1:0] bitfield at zero. The second is either a byte-wide write to the RWTU[1:0] bitfield or a 32-bit write that sets the RWTU[1:0] bitfield while keeping the RWT[7:0] bitfield unchanged.

2.20.9 Incorrect flexible PPS output interval under specific conditions

Description

The use of the fine correction method for correcting the IEEE 1588 internal time reference, combined with a large frequency drift of the driving clock from the grandmaster source clock, leads to an incorrect interval of the flexible PPS output used in Pulse train mode. As a consequence, external devices synchronized with the flexible PPS output of the device can go out of synchronization.

Workaround

Use the coarse method for correcting the IEEE 1588 internal time reference.

2.20.10 Packets dropped in RMII 10 Mbps mode due to fake dribble and CRC error

Description

When operating with the RMII interface at 10 Mbps, the Ethernet peripheral may generate a fake extra nibble of data repeating the last packet (nibble) of the data received from the PHY interface. This results in an odd number of nibbles and is flagged as a dribble error. As the RMII only forwards to the system completed bytes of data, the fake nibble would be ignored and the issue would have no consequence. However, as the CRC error is also flagged when this occurs, the error-packet drop mechanism (if enabled) discards the packets.

Note: Real dribble errors are rare. They may result from synchronization issues due to faulty clock recovery.

Workaround

When using the RMII 10 MHz mode, disable the error-packet drop mechanism by setting the FEP bit of the ETH_MTLRXQOMR register. Accept packets of transactions flagging both dribble and CRC errors.

2.20.11 ARP offload function not effective

Description

When the Target Protocol Address of a received ARP request packet matches the device IP address set in the ETH_MACARPAR register, the source MAC address in the SHA field of the ARP request packet is compared with the device MAC address in ETH_MACA0LR and ETH_MACA0HR registers (Address0), to filter out ARP packets that are looping back.

Instead, a byte-swapped comparison is performed by the device. As a consequence, the packet is forwarded to the application as a normal packet with no ARP indication in the packet status, and the device does not generate an ARP response.

For example, with the Address0 set to 0x6655 4433 2211:

- If the SHA field of the received ARP packet is 0x6655 4433 2211, the ARP response is generated while it should not.
- If the SHA field of the received ARP packet is 0x1122 3344 5566, the ARP response not is generated while it should.

Workaround

Parse the received frame by software and send the ARP response if the source MAC address matches the byte-swapped Address0.

2.21 CEC

2.21.1 Missed CEC messages in normal receiving mode

Description

In normal receiving mode, any CEC message with destination address different from the own address should normally be ignored and have no effect to the CEC peripheral. Instead, such a message is unduly written into the reception buffer and sets the CEC peripheral to a state in which any subsequent message with the destination address equal to the own address is rejected (NACK), although it sets RXOVR flag (because the reception buffer is considered full) and generates (if enabled) an interrupt. This failure can only occur in a multi-node CEC framework where messages with addresses other than own address can appear on the CEC line.

The listen mode operates correctly.

Workaround

Use listen mode (set LSTEN bit) instead of normal receiving mode. Discard messages to single listeners with destination address different from the own address of the CEC peripheral.

2.21.2 Unexpected TXERR flag during a message transmission

Description

During the transmission of a 0 or a 1, the HDMI-CEC drives the open-drain output to high-Z, so that the external pull-up implements a voltage rising ramp on the CEC line.

In some load conditions, with several powered-off devices connected to the HDMI-CEC line, the rising voltage may not drive the HDMI-CEC GPIO input buffer to V_{IH} within two HDMI-CEC clock cycles from the high-Z activation to TXERR flag assertion.

Workaround

Limit the maximum number of devices connected to the HDMI-CEC line to ensure the GPIO V_{IH} threshold is reached within a time of two HDMI-CEC clock cycles (~61 µs).

The maximum equivalent 10%-90% rise time for the HDMI-CEC line is 111.5 µs, considering a V_{IH} threshold equal to 0.7 x V_{DD}.

Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on www.st.com for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

Revision history

Table 6. **Document revision history**

Date	Version	Changes
29-Aug-2023	1	Initial release.
28-Jan-2024	2	Added errata: • Section 2.1.1: PLD might perform linefill to address that would generate a MemManage Fault • Section 2.1.2: Software programming errors might not be reported for online MBIST access to the ICACHE • Section 2.1.4: Store after cache invalidate without intervening barrier might cause inconsistent memory view • Store after cache invalidate without intervening barrier might cause inconsistent memory view • Section 2.4.12: Transactions are limited to 8 Mbytes in OctaRAM™ memories • Section 2.4.9: CALMAX bit not set when the PHY reaches the DLL maximum value • Section 2.4.10: Setting the ABORT bit does not generate an error on the AHB bus for undefined-length incremental burst transfers • Section 2.4.11: Read data corruption when a wrap transaction is followed by a linear read to the same MSB address • Section 2.5.1: Certain quad memories may be reset during arbitration while in single-SPI mode • Section 2.6.1: Command response and receive data end bits not checked • Section 2.9.1: Occasional writing miss to frame buffer with slow memories • Section 2.10.1: Data transfer from TAMP_BKPxR to key registers must be done only in ascending order when KEYSEL[2:0] is set to 010 or 100 • Section 2.16.1: Possible LPUART transmitter issue when using low BRR[15:0] value • Section 2.17.2: Truncation of SPI output signals after EOT event • Section 2.17.3: TIFRE flag wrongly set in slave PCM long frame mode if FIXCH = 1 • Section 2.17.4: TIFRE flag never set in slave PCM/I2S mode if FIXCH = 0 • Section 2.21.1: Missed CEC messages in normal receiving mode • Section 2.21.2: Unexpected TXERR flag during a message transmission Updated errata: • Section 2.2.1: Boundary scan, PC10 and PC11 are not controllable on TFBGA100 package
11-Mar-2024	3	Updated scope to cover silicon revision Y only.
03-Jul-2024	4	Added: • Section 2.2.5: SRAM1 AHB limitation if the device is not in OPEN state • Section 2.2.6: LSE crystal oscillator may be disturbed by transitions on PC13 • Section 2.2.7: Secure Firmware Install (SFI) is not supported • Section 2.3.4: CTB1, CTB2, MODE[2:0] write-only bitfields in FMC_SDCMR incorrectly described as read-write • Section 2.4.13: Variable latency is not supported when a refresh collision occurs during a write access to some OctaRAM™ memories

Contents