STM32H523xx and STM32H533xx device errata

Applicability

This document applies to the part numbers of STM32H523xx and STM32H533xx 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 RM0481.

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
STM32H523xx	STM32H523CC, STM32H523RC, STM32H523VC, STM32H523ZC, STM32H523CE, STM32H523HE, STM32H523RE, STM32H523VE, STM32H523ZE
STM32H533xx	STM32H533CE, STM32H533HE, STM32H533RE, STM32H533VE, STM32H533ZE

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32H523xx/33xx	A	0x1000

1. *Refer to the device datasheet for how to identify this code on different types of package.*
2. *REV_ID[5:0] bitfield of DBGMCU_IDCODE fuse.*

Note: DBGMCU_IDCODE register is not automatically shadowed with OTP content, so a fuse read sequence must be issued to get the register updated once (clear after reading). Refer to product reference manual - BSEC section "Operations on fuses".

1 Summary of device errata

The following table gives a quick reference to the STM32H523xx and STM32H533xx 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. A
Core	2.1.1	Access permission faults are prioritized over unaligned Device memory faults	N
System	2.2.1	LSE crystal oscillator may be disturbed by transitions on PC13	N
	2.2.2	Peripheral triggers connected to RTC wakeup timer interrupt instead of RTC wakeup timer trigger signal	A
	2.2.3	Incorrect behavior of ICACHE refill from SRAM when an SRAM AHB error occurs	N
	2.2.4	SRAMx_RST option bits have an immediate effect	A
	2.2.5	Full JTAG configuration without NJTRST pin cannot be used	A
	2.2.6	SRAM2 is erased when the backup domain is reset	A
	2.2.7	Clearing WWDG_SW might result in debug authentication or ST-iRoT failure	A
	2.2.8	LSE low drive mode is not functional	N
	2.2.9	Incorrect backup domain reset	A
	2.2.10	Debug not available when TrustZone® security is disabled and the PRODUCT_STATE is iROT-Provisioned	A
	2.2.11	Low-speed external clock in analog bypass mode might not work properly	A
	2.2.12	Octo-SPI memory data failure when using HCLK as kernel clock	A
	2.2.13	PLL1P output can unduly be disabled by software when used as SYSCLK clock	A
	2.2.14	ADC and DAC clock selection cannot be secured when the ADC is nonsecure	A
	2.2.15	Reset vector catch feature cannot be used when debugging is disabled for secure code	A
FMC	2.3.1	Dummy read cycles inserted when reading synchronous memories	N
	2.3.3	Wrong data read from a busy NAND memory	A
OCTOSPI	2.4.1	Memory-mapped write error response when DQS output is disabled	P
	2.4.2	TCF set twice on an abort	A
	2.4.3	Deadlock in clock mode 3 when prefetching the 64 upper memory bytes	A
	2.4.4	Memory wrap instruction not enabled when DQS is disabled	N
	2.4.5	Deadlock or write-data corruption after spurious write to a misaligned address in OCTOSPI_AR register	N
	2.4.6	Deadlock on consecutive out-of-range memory-mapped write operations	P
	2.4.7	Indirect write mode limited to 256 Mbytes	N

Function	Section	Limitation	Status Rev. A
OCTOSPI	2.4.8	Read-modify-write operation does not clear the MSEL bit	A
	2.4.9	Setting the ABORT bit does not generate an error on the AHB bus for undefined-length incremental burst transfers	P
	2.4.10	Read data corruption when a wrap transaction is followed by a linear read to the same MSB address	N
	2.4.11	Transactions are limited to 8 Mbytes in OctaRAM™ memories	N
ADC	2.5.1	New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0	A
	2.5.2	Two consecutive context conversions fail when writing new context in ADC_JSQR just after previous context completion with JQDIS = 0 and JQM = 0	A
	2.5.3	Unexpected regular conversion when two consecutive injected conversions are performed in Dual interleaved mode	A
	2.5.4	ADC_AWDy_OUT reset by non-guarded channels	A
	2.5.5	Injected data stored in the wrong ADC_JDRx registers	A
	2.5.6	ADC slave data may be shifted in Dual regular simultaneous mode	A
TIM	2.7.1	Bidirectional break mode not working with short pulses	N
	2.7.2	Timer connection to USB SOF might not work properly	A
LPTIM	2.8.1	Device may remain stuck in LPTIM interrupt when entering Stop mode	A
	2.8.2	ARRM and CMPM flags are not set when APB clock is slower than kernel clock	P
	2.8.3	Interrupt status flag is cleared by hardware upon writing its corresponding bit in LPTIM_DIER register	N
	2.8.4	PLL2 output cannot be used as LPTIM clock source	A
IWDG	2.9.1	Independent watchdog does not wake up the system from Stop mode	N
RTC	2.10.1	Alarm flag may be repeatedly set when the core is stopped in debug	N
	2.10.2	Timestamp flag unexpectedly raised when disabling timestamp	A
I2C	2.11.1	Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period	P
	2.11.2	Spurious bus error detection in master mode	A
	2.11.3	SDA held low upon SMBus timeout expiry in slave mode	A
I3C	2.12.1	I3C controller: unexpected read data bytes during a legacy I^2C read	A
	2.12.2	I3C controller: SCL clock is not stalled during address ACK/NACK phase following a frame start, when enabled through I3C_TIMINGR2 register	A
	2.12.3	I3C controller: unexpected first frame with a 0x7F address when the I3C peripheral is enabled	A
	2.12.4	I3C controller: no timestamp on IBI acknowledge when timing control is used in Asynchronous mode 0	A
USART	2.13.2	USART does not generate DMA requests after setting/clearing DMAT bit	A
	2.13.3	Received data may be corrupted upon clearing the ABREN bit	A
	2.13.4	Noise error flag set while ONEBIT is set	N
	2.13.1	Data corruption due to noisy receive line	A
LPUART	2.14.1	LPUART does not generate DMA requests after setting/clearing DMAT bit	A
	2.14.2	Possible LPUART transmitter issue when using low BRR[15:0] value	P
SPI	2.15.2	Truncation of SPI output signals after EOT event	A
	2.15.1	RDY output failure at high serial clock frequency	N

Function	Section	Limitation	Status Rev. A
SPI	2.15.3	TIFRE flag wrongly set in slave PCM long frame mode if FIXCH = 1	N
	2.15.4	TIFRE flag never set in slave PCM/I2S mode if FIXCH = 0	N
FDCAN	2.17.1	Desynchronization under specific condition with edge filtering enabled	A
	2.17.2	Tx FIFO messages inverted under specific buffer usage and priority setting	A
USB	2.18.1	Buffer description table update completes after CTR interrupt triggers	A
UCPD	2.19.1	TXHRST upon write data underflow corrupting the CRC of the next packet	A
	2.19.2	Ordered set with multiple errors in a single K-code is reported as invalid	N
CEC	2.20.1	Missed CEC messages in normal receiving mode	A
	2.20.2	Unexpected TXERR flag during a message transmission	A
SDMMC	2.16.1	Command response and receive data end bits not checked	N

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

Table 4. Summary of device documentation errata

Function	Section	Documentation erratum
FMC	2.3.2	Missing information on prohibited 0xFF value of NAND transaction wait timing
SAES	2.6.1	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

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®-M33 core revision r0p4 is available from http://infocenter.arm.com.

2.1.1 Access permission faults are prioritized over unaligned Device memory faults

Description

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

The failure occurs when the MPU is enabled and:

- A load/store access occurs to an address which is not aligned to the data type specified in the instruction.
- The memory access hits one region only.
- The region attributes (specified in the MAIR register) mark the location as Device memory.
- The region access permissions prevent the access (that is, unprivileged or write not allowed).

The MemManage fault caused by the access permission violation will be prioritized over the UNALIGNED UsageFault exception because of the memory attributes.

Workaround

None. However, it is expected that no existing software is relying on this behavior since it was permitted in Armv7-M.

2.2 System

2.2.1 LSE crystal oscillator may be disturbed by transitions on PC13

Description

The LSE crystal oscillator clock frequency can be incorrect when PC13 is toggling in input or output (for example when used for RTC_OUT1).

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

Workaround

None.

Avoid toggling PC13 when LSE is used.

2.2.2 Peripheral triggers connected to RTC wakeup timer interrupt instead of RTC wakeup timer trigger signal

Description

GPDMA1 and GPDMA2 triggers are connected to RTC wakeup timer interrupt instead of RTC wakeup timer trigger signal (rtc_wut_trg).

Workaround

Enable the RTC wakeup timer interrupt. The wakeup timer flag must be cleared in the interrupt subroutine before getting a new trigger.

To avoid serving an interrupt, use triggers from RTC alarm or from LPTIM instead of RTC wakeup timer.

2.2.3 Incorrect behavior of ICACHE refill from SRAM when an SRAM AHB error occurs

Description

If an AHB error is returned by the SRAM memory protection controller (MPCBB) during an ICACHE refill operation from SRAM, the next access may not be correctly handled and thus corrupted.

The SRAM (MPCBB) returns an AHB error during a CPU fetch operation if the one of the following conditions is met:

- a secure access to a non-secure area is ongoing (only when TrustZone® is enabled),
- a non-secure access to a secure area is ongoing (only when TrustZone® is enabled),
- a non-privileged access to a privilege area is ongoing, or
- a non-mapped address is accessed.

Workaround

None.

2.2.4 SRAMx_RST option bits have an immediate effect

Description

Clearing the SRAMx_RST (x = 1, 2) option bit immediately triggers an SRAMx erase operation. This might lead to a CPU exception or unpredictable behavior if the erased SRAMx is being used by the application.

Workaround

The application must be reset immediately after SRAMx_RST clear.

2.2.5 Full JTAG configuration without NJTRST pin cannot be used

Description

When using the JTAG debug port in Debug mode, the connection with the debugger is lost if the NJTRST pin (PB4) is used as a GPIO or for an alternate function other than NJTRST. Only the 4-wire JTAG port configuration is impacted.

Workaround

Use the SWD debug port instead of the full 4-wire JTAG port.

2.2.6 SRAM2 is erased when the backup domain is reset

Description

When the VSWRST bit of the RCC_BDCR register (entire VSW domain reset) and the DBP bit of the PWR_DBPCR register (write access to backup domain enabled) are both set, a backup domain reset also erases the SRAM2 content.

Workaround

Apply the following sequence to reset the backup domain and safely erase SRAM2:

1. After the device exits the reset state, check that no SRAM erase is ongoing in *SystemInit()*.
2. Reset the backup domain. This also erases the SRAM2 content.
3. Wait until SRAM2 erase is complete.
4. Proceed with normal application execution, and store data into the SRAM2.

2.2.7 Clearing WWDG_SW might result in debug authentication or ST-iRoT failure

Description

If the WWDG_SW configuration option bit is cleared (window watchdog controlled by hardware), the WWDG is always enabled after a reset, and cannot be disabled. As a result:

- The ST-DA debug authentication sequence might fail, preventing any possibility to securely reopen the debug or launch regressions on secure products.
- The ST-iROT (immutable root of trust) execution might fail during the boot sequence.

Workaround

Do not enable the *"WWDG controlled by hardware"* configuration. Instead, use the *"WWDG controlled by software"* configuration (WWDG_SW configuration option bit set).

2.2.8 LSE low drive mode is not functional

Description

The LSE oscillator may not start or may stop in low drive mode (LSEDRV = 00). Using this mode is forbidden.

Workaround

None.

2.2.9 Incorrect backup domain reset

Description

The backup domain reset may be missed upon a backup domain power-on following a V_{BAT} power-off in VBAT mode, if the V_{BAT} 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 V_{BAT} 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, BackupTestRegister) 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 BackupTestRegister 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 0x0001 0000 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 0x0000 0000 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.

2.2.10 Debug not available when TrustZone® security is disabled and the PRODUCT_STATE is iROT-Provisioned

Description

When the TrustZone® security is disabled, and the PRODUCT_STATE is iROT-Provisioned, the debug must be available for a code executed from an HDPL 3 area. However, in this case, the code cannot be debugged.

Workaround

If PRODUCT_STATE = iROT-Provisioned, force the debug opening in the first stage of the boot sequence software.

Below an example of code:

```
/* This code ensures that in PRODUCT_STATE = IROT_PROVISIONED, the Debug is opened for HDPL3
when TZEN = disabled. */
 /* This code must be integrated in a HDPL1 code */
if (READ_BIT(FLASH->OPTSR_CUR, FLASH_OPTSR_PRODUCT_STATE_Msk) == OB_PROD_STATE_IROT_PROVISION
ED)
 {
 __HAL_RCC_SBS_CLK_ENABLE();
 ((SBS_TypeDef *)SBS_BASE)->DBGCR = 0x006FB4B4U; // 6F value to open debug when HDPL3 is reac
hed
 }
 /* To be able to attach the debugger, the code to debug must be executed in HDPL3. */
 /* The code must call the below function twice before reaching the code to debug: */
 /* HAL_SBS_IncrementHDPLValue(); */
```

2.2.11 Low-speed external clock in analog bypass mode might not work properly

Description

While the LSE bypass in analog mode is selected (LSEBYP = 1 and LSEEXT = 0 in the RCC_BDCR register), the LSE clock might not work properly.

Workaround

Do not use the LSE bypass in analog mode. Instead, use the digital LSE bypass mode (LSEBYP = 1 and LSEEXT = 1 in the RCC_BDCR register).

2.2.12 Octo-SPI memory data failure when using HCLK as kernel clock

Description

When the HCLK clock (rcc_hclk4) frequency is lower than the SYSCLK clock frequency, the HCLK clock duty cycle is not 50 %. In this condition, selecting HCLK as Octo-SPI kernel clock may lead to memory data failure.

Note: The HCLK clock is prescaled by a ratio controlled with the HPRE[3:0] bitfield of the RCC_CFGR2 register. Any value exceeding 0b0111 makes the HCLK frequency lower than the SYSCLK frequency.
The Octo-SPI kernel clock is selected with the OCTOSPI1SEL[1:0] bitfield of the RCC_CCIPR4 register.

Workaround

When the HCLK and SYSCLK frequencies differ, do not select rcc_hclk4 as Octo-SPI kernel clock. Instead, select another clock such as pll1_q_ck or pll2_r_ck.

2.2.13 PLL1P output can unduly be disabled by software when used as SYSCLK clock

Description

The PLL1P output (pll1_p_ck clock signal) is expected to be protected against disabling by software while selected for SYSCLK (sys_ck clock signal). Unduly, it is on the PLL1Q output that this protection acts, instead on PLL1P. As a consequence, the PLL1P output selected as SYSCLK can be disabled by software, which leads to a system deadlock.

Note: The PLL1P output (pll1_p_ck clock signal) is selected as SYSCLK (sys_ck clock signal) by setting the SW[1:0] bitfield of the RCC_CFGR1 register to 0b11. The PLL1P and PLL1Q outputs of the PLL1 are enabled and disabled through, respectively, the bits PLL1PEN and PLL1QEN of the RCC_PLL1CFGR register.

Workaround

Do not clear the PLL1PEN bit while the PLL1P output is used as SYSCLK clock.

2.2.14 ADC and DAC clock selection cannot be secured when the ADC is nonsecure

Description

The ADC and DAC kernel clock source selection is expected to be secure whenever the ADC or the DAC is secure. It is duly secured when the ADC is secure. However, it unduly remains nonsecure when the DAC is secure but the ADC nonsecure.

Note: The ADC and DAC kernel clock source is selected with the ADCDACSEL[2:0] bitfield of the RCC_CCIPR5 register.

Workaround

To secure the ADC and DAC kernel clock source selection bitfield, always make the ADC secure.

2.2.15 Reset vector catch feature cannot be used when debugging is disabled for secure code

Description

If the product state (PRODUCT_STATE[7:0] bitfield of the FLASH_OPTSR_CUR register) is different from the open state, debugging is not allowed except by resetting the CPU by mean of a reset vector catch. This may lead to the CPU not halting before executing the first instruction of the nonsecure exception handler.

Consequently, nonsecure code execution may not stop after switching to the nonsecure state, and the CPU may proceed executing code in the nonsecure area.

Workaround

To halt the CPU in the reset handler, insert a software break point in the first nonsecure instruction of the user application. The address of the reset handler can be read from the NSBOOTADD[23:8] bitfield of the FLASH_NSBOOTR_CUR register.

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 Missing information on prohibited 0xFF value of NAND transaction wait timing

Description

Some reference manual revisions may omit the information that the value 0xFF is prohibited for the wait timing of NAND transactions in their corresponding memory space (common or attribute).

Whatever the setting of the PWAITEN bit of the FMC_PCRx register, the wait timing set to 0xFF would cause a NAND transaction to stall the system with no fault generated.

This is a documentation error rather than a device limitation.

Workaround

No application workaround required provided that the 0xFF wait timing value is duly avoided.

2.3.3 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.

2.4 OCTOSPI

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

Description

If the DQSE control bit of the OCTOSPI_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 OCTOSPI_WCCR register, even for memories that have no DQS pin.

2.4.2 TCF set twice on an abort

Description

The TCF (total-count flag) of the OCTOSPI_SR register is set twice when a memory-mapped write is aborted (when the software sets the ABORT bit of the OCTOSPI_CR register), and a memory-mapped burst request is received on the same cycle that the BUSY falls. TCF is set once when BUSY falls, and again when the burst transfer finishes. The burst request gets an error response (HardFault) as expected.

The software does not notice that TCF gets set twice unless the software clears TCF before the burst transfer finishes.

Workaround

Apply one of the following measures:

- Ensure that no memory-mapped write requests are being sent after requesting an abort.
- When a HardFault is generated after a requested abort, ensure TCF is cleared prior to restart transfers.

2.4.3 Deadlock in clock mode 3 when prefetching the 64 upper memory bytes

Description

When CKMOD of the OCTOSPI_DCR1 register is set (clock mode 3 is enabled), a deadlock may occur if a memory-mapped request to a non-sequential address is received one memory clock-cycle after an access (normal or prefetch) to any of the 64 upper memory bytes.

Workaround

Apply one of the following measures:

- Recommended: Use clock mode 0, by setting the bit CKMODE of the OCTOSPI_DCR1 register (all memories known to date support clock mode 0).
- In memory-mapped mode, do not access the 64 upper memory bytes.
- Set DEVSIZE bitfield of the OCTOSPI_DCR1 register to allocate more space than the actual size of the memory.

2.4.4 Memory wrap instruction not enabled when DQS is disabled

Description

Memory wrap instruction (as configured in the OCTOSPI_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.

2.4.5 Deadlock or write-data corruption after spurious write to a misaligned address in OCTOSPI_AR register

Description

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

- FMODE = 00 in OCTOSPI_CR (indirect write mode)
- DQSE = 1 in OCTOSPI_CCR (DQS active)

then, the OCTOSPI 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 OCTOSPI 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 OCTOSPI is configured to send four bytes of data to the memory (16-bit DTR mode or dual-octal DTR mode).

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

Workaround

None.

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

Description

The DEVSIZE[4:0] bitfield of the OCTOSPI_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 OCTOSPI 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.

2.4.7 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 OCTOSPI_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 OCTOSPI_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 OCTOSPI_CR register, otherwise, the MSEL bit remains set.

2.4.9 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 OCTOSPI_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.

2.4.10 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.

2.4.11 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 OCTOSPI_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.

2.5 ADC

2.5.1 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.

2.5.2 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.

2.5.3 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.

2.5.4 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.

2.5.5 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).

2.5.6 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.

2.6 SAES

2.6.1 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.

2.7 TIM

2.7.1 Bidirectional break mode not working with short pulses

Description

The TIM_BKIN and TIM_BKIN2 I/Os can be configured in bidirectional mode using the BKBID and BK2BID bits in the TIMx_BDTR register, to be forced to 0 when a break/break2 event occurs. The bidirectional break/break2 mode is not functional when the pulse width on break/break2 input is lower than two tim_ker_clk periods.

This limitation is also valid when software break events are generated (the break event is correctly generated internally but not reflected on break inputs).

Workaround

None.

For applications that can afford some latency in bidirectional break mode, the break interrupt can eventually be enabled, for the CPU to verify the break input state and force it to zero when a break/break2 event occurred.

2.7.2 Timer connection to USB SOF might not work properly

Description

USB SOF signal might not be properly synchronized with TIM2 and TIM5 ITR12.

Workaround

Connect externally USB SOF (PA8) and the timer ETR input pin.

2.8 LPTIM

2.8.1 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.

2.8.2 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.

2.8.3 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.

2.8.4 PLL2 output cannot be used as LPTIM clock source

Description

When pll2_p_ck is selected as LPTIMx clock source by setting LPTIMxSEL[2:0] bitfield of the RCC_CCIPR2 register to 0b001, LPTIMx does not receive its kernel clock, and does not properly operate.

Workaround

Avoid selecting pll2_p_ck as clock source for LPTIMx.

Select any other clock source by programming LPTIMxSEL[2:0] to a value different from 0b001. Refer to the device reference manual for the list of possible values.

2.9 IWDG

2.9.1 Independent watchdog does not wake up the system from Stop mode

Description

The independent watchdog early wakeup interrupt does not wake up the system from Stop mode.

Workaround

None.

2.10 RTC

2.10.1 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.

2.10.2 Timestamp flag unexpectedly raised when disabling timestamp

Description

The TSF flag of RTC_SR is wrongly set when disabling the timestamp. This issue occurs when the following conditions are met:

- timestamp negative edge detection is requested, and
- no edge has occurred

The other detection configurations are not impacted.

Workaround

After clearing the TSE bit of the RTC_CR register:

- in polling mode, wait for 7 ms to allow the TSF flag to be set. Then clear it.
- in interrupt mode: clear the TSF flag as soon as it is set.

2.11 I2C

2.11.1 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 I²C-bus SDA line when $t_{SU;DAT}$ is smaller than one I2C kernel clock (I²C-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 I²C-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.

2.11.2 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 I²C-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.

2.11.3 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 I²C 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.

2.12 I3C

2.12.1 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 I^2C 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 I^2C target incorrectly drives SDA low on the 9th T-bit phase of the end of read from the I3C controller

To end a legacy I^2C 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 I^2C 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.

2.12.2 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 I²C/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.

2.12.3 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.

2.12.4 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.

2.13 USART

2.13.1 Data corruption due to noisy receive line

Description

In all modes, except synchronous slave mode, the received data may be corrupted if a glitch to zero shorter than the half-bit occurs on the receive line within the second half of the stop bit.

Workaround

Apply one of the following measures:

- Either use a noiseless receive line, or
- add a filter to remove the glitches if the receive line is noisy.

2.13.2 USART does not generate DMA requests after setting/clearing DMAT bit

Description

If the DMA is used for data transmission (DMAT = 1 in USART_CR3 register), and the software clears DMAT bit and sets it again to prepare the next transmission, then the peripheral does not generate DMA requests anymore. As a result, data are not transmitted.

Workaround

- Avoid clearing DMAT.
- If clearing DMAT is needed after the end of DMA transfers, once DMAT is cleared, disable and reenable the peripheral through UE bit of USART_CR1 register. This workaround is acceptable only if the peripheral is not used in receiver mode.
- DMAT can be cleared if the next transmission is based on polling/interrupt.

2.13.3 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.

2.13.4 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.

2.14 LPUART

2.14.1 LPUART does not generate DMA requests after setting/clearing DMAT bit

Description

If the DMA is used for data transmission (DMAT = 1 in LPUART_CR3 register), and the software clears DMAT bit and sets it again to prepare the next transmission, then the peripheral does not generate DMA requests anymore. As a result, data are not transmitted.

Workaround

- Avoid clearing DMAT.
- If clearing DMAT is needed after the end of DMA transfers, once DMAT is cleared, disable and reenable the peripheral through UE bit of LPUART_CR1 register. This workaround is acceptable only if the peripheral is not used in receiver mode.
- DMAT can be cleared if the next transmission is based on polling/interrupt.

2.14.2 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.

2.15 SPI

2.15.1 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.

2.15.2 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.

2.15.3 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.

2.15.4 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.

2.16 SDMMC

2.16.1 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.

2.17 FDCAN

2.17.1 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.

2.17.2 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
```

2.18 USB

2.18.1 Buffer description table update completes after CTR interrupt triggers

Description

During OUT transfers, the correct transfer interrupt (CTR) is triggered a little before the last USB SRAM accesses have completed. If the software responds quickly to the interrupt, the full buffer contents may not be correct.

Workaround

Software should ensure that a small delay is included before accessing the SRAM contents. This delay should be 800 ns in Full Speed mode and 6.4 µs in Low Speed mode.

2.19 UCPD

2.19.1 TXHRST upon write data underflow corrupting the CRC of the next packet

Description

TXHRST command issued at the instant of detecting write data underflow during a packet transmission can cause a corrupt CRC of the following packet.

Workaround

Use DMA (TXDMAEN) rather than software writing to UCPD_TXDR. Normally, this prevents write data underflow. Should a corrupt CRC event still occur, the DMA transfer method retransmits the packet until the CRC is correct and the packet acknowledged by the receiver.

2.19.2 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 CEC

2.20.1 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.

2.20.2 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 5. **Document revision history**

Date	Version	Changes
29-Mar-2024	1	Initial release.

Contents