



## STM32U3B5xx and STM32U3C5xx device errata

### Applicability

This document applies to the part numbers of STM32U3B5xx and STM32U3C5xx 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 RM0523. 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
STM32U3B5xx	STM32U3B5CI, STM32U3B5CG, STM32U3B5RI, STM32U3B5RG, STM32U3B5JI, STM32U3B5JG, STM32U3B5VI, STM32U3B5VG, STM32U3B5WI, STM32U3B5WG, STM32U3B5QI, STM32U3B5QG, STM32U3B5ZI, STM32U3B5ZG
STM32U3C5xx	STM32U3C5CI, STM32U3C5RI, STM32U3C5JI, STM32U3C5VI, STM32U3C5WI, STM32U3C5QI, STM32U3C5ZI

**Table 2. Device variants**

Reference	Silicon revision codes	
	Device marking <sup>(1)</sup>	REV_ID <sup>(2)</sup>
STM32U3B5xx and STM32U3C5xx	A	0x1000

1. Refer to the device datasheet for how to identify this code on different types of package.
2. REV\_ID[15:0] bitfield of DBGMCU\_IDCODE fuse.

## 1 Summary of device errata

The following table gives a quick reference to the STM32U3B5xx and STM32U3C5xx device limitations and their status:

<b>A</b>	Limitation applicable. Workaround available
<b>N</b>	Limitation applicable. No workaround available
<b>P</b>	Limitation applicable. Partial workaround available
<b>-</b>	Limitation not applicable.

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	USART2 cannot be used in autonomous mode or to wake up from stop mode	N
	2.2.2	PA1 and PA7 limited usage when low-leakage OPAMP1_VINM and OPAMP2_VINM are used	N
	2.2.3	SRAMs retained in Stop 1 mode may be corrupted on exit by system reset	A
	2.2.4	HSI remains enabled after wake-up from Stop mode when debug in Stop mode is enabled	A
	2.2.5	LSE in bypass mode may not be functional	N
	2.2.6	LSE crystal oscillator may be disturbed by transitions on PC13	N
	2.2.7	Incorrect backup domain reset with V <sub>BAT</sub> and V <sub>DD</sub> supplied by different power source	A
HSP	2.3.1	Scheduler error flag wrongly set when executing CNN functions	N
OCTOSPI	2.4.1	Memory-mapped write error response when DQS output is disabled	P
	2.4.2	Byte possibly dropped during an SDR read in clock mode 3 when a transfer gets automatically split	A
	2.4.3	Deadlock or write-data corruption after spurious write to a misaligned address in OCTOSPI_AR register	N
	2.4.4	At least six cycles memory latency must be set when DQS is used for HyperBus™ memories	A
	2.4.5	Data write discarded in memory-mapped mode if a write to a misaligned address is directly followed by a request to the same address	A
	2.4.6	Setting the ABORT bit does not generate an error on the AHB bus for undefined-length incremental burst transfers	P
	2.4.7	Read data corruption when a wrap transaction is followed by a linear read to the same MSB address	N
	2.4.8	Transactions are limited to 8 Mbytes in OctaRAM™ memories	N
	2.4.9	Variable latency is not supported when a refresh collision occurs during a write access to some OctaRAM™ memories	P
	2.4.10	OCTOSPI external memory read issue after exiting Stop 2 or Stop 3 mode when using DQS and delay block	A

Function	Section	Limitation	Status
			Rev. A
SDMMC	2.5.1	Command response and receive data end bits not checked	N
ADC	2.6.1	In combined regular simultaneous plus alternate trigger mode, stopping injected conversion may delay regular conversion	A
	2.6.2	In certain dual ADC modes with regular oversampling continued mode enabled, the JEOC flag is not set at the end of the new injected conversion	A
	2.6.3	In certain dual ADC modes with DMA one-shot mode enabled, JADSTART and ADSTART may not be cleared if JADSTP and ADSTP are set at the same time	A
	2.6.4	Wrong injected conversion data in bulb sampling mode	A
	2.6.5	Shifted master and slave sequence in interleaved discontinuous mode	A
	2.6.6	When the ADC clock is derived from the AHB clock, the injected conversion latency is not respected if the injected trigger coincides with the stopping of the regular conversion	A
	2.6.7	When injected conversions are enabled and the ADC clock is slower than the ADC kernel clock, a wrong context can be applied in certain conditions	A
	2.6.8	In certain dual modes, the fixed trigger latency for the injected conversions may not be respected	A
	2.6.9	In simultaneous regular mode, stopping an injected conversion may shift the next regular conversion master and slave timing by one clock cycle	A
DAC	2.7.1	Incorrect DAC output voltage due to DAC kernel clock setting	N
VREFBUF	2.8.1	V <sub>REFBUF_OUT</sub> voltage overshoots in Range 2 or Stop 1 mode	A
RNG	2.9.1	RNG may report continuous seed errors in specific environmental conditions	N
TIM	2.10.1	Unexpected PWM output when using ocref_clr	N
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
	2.12.2	RTC calendar read may return transient incorrect value when BYPSHAD = 0	A
	2.12.3	Parasitic tamper detection when debugger is used in RDP Level 0	A
I2C	2.13.1	Wrong data sampling when data setup time (t <sub>SU, DAT</sub> ) is shorter than one I2C kernel clock period	P
	2.13.2	Spurious bus error detection in controller mode	A
USART	2.14.1	Wrong data received by SPI slave receiver in autonomous mode with CPOL = 1	A
	2.14.2	Received data may be corrupted upon clearing the ABREN bit	A
	2.14.3	Noise error flag set while ONEBIT is set	N
LPUART	2.15.1	Possible LPUART transmitter issue when using low BRR[15:0] value	P
SPI	2.16.1	Possible corruption of last-received data depending on CRCSIZE setting	A
	2.16.2	MODF flag cannot generate interrupt	A
	2.16.3	RDY output failure at high serial clock frequency	N
	2.16.4	Master communication suspension fails in autonomous mode	N
	2.16.5	SPE may not be cleared upon MODF event	A
	2.16.6	SPI slave stalls with masters not providing extra SCK periods upon <i>Not ready</i> signaling	A
	2.16.7	Truncation of SPI output signals after EOT event	A

Function	Section	Limitation	Status
			Rev. A
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

## 2 Description of device errata

The following sections describe the errata of the applicable devices and provide a workaround where available. They are grouped by device functions.

The applicable devices are based on Arm® Cortex® core.

arm

*Note: Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.  
The Arm word and logo are trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved.*

### 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 results 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 is 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 is 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 USART2 cannot be used in autonomous mode or to wake up from stop mode

##### Description

USART2 is intended to operate in stop 0 and stop 1 modes and to wake up from these modes by using the corresponding stop clock enable bits in the RCC peripheral. However, this connection is not functional. USART2 can only operate in run and sleep modes.

##### Workaround

None.

#### 2.2.2 PA1 and PA7 limited usage when low-leakage OPAMP1\_VINM and OPAMP2\_VINM are used

##### Description

On the UFBGA132 package, when the low-leakage dedicated pins OPAMP1\_VINM and OPAMP2\_VINM are connected to OPAMP inputs, PA1 and PA7 GPIOs must be configured in analog mode and cannot be used for other functions.

If security is enabled on the platform, OPAMP1\_VINM and OPAMP2\_VINM cannot be connected to OPAMP inputs if PA1 and PA7 are configured as secure while the OPAMPs are defined as non-secure in the GTZC peripheral.

**Workaround**

None.

**2.2.3 SRAMs retained in Stop 1 mode may be corrupted on exit by system reset**

**Description**

Some SRAM bit values may be lost in retained SRAMs when exiting from Stop 1 mode by a system reset. This issue occurs when one or more SRAM pages or instances are powered down during Stop 1 mode by using the SRAM power-down in stop bits of the PWR\_CR2 registers.

**Workaround**

Keep all SRAMs in retention when entering Stop 1 mode.

**2.2.4 HSI remains enabled after wake-up from Stop mode when debug in Stop mode is enabled**

**Description**

HSI clock remains enabled after wake-up from Stop mode where:

- HSI was used as system clock before entering Stop mode
- MSIS is selected as the wake-up clock
- A debugger is connected
- DBG\_STOP bit is set in the DBGMCU\_CR register

**Workaround**

If not needed, force HSI OFF by software after wake-up from Stop in above condition.

**2.2.5 LSE in bypass mode may not be functional**

**Description**

LSE in bypass mode fails on around 1% of produced parts. The failure rate increases with colder temperature.

**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 in input or output (for example when used for RTC\_OUT1).

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

The WLCSP and UFBGA packages are not impacted by this limitation.

**Workaround**

Avoid toggling PC13 when LSE is used on LQFP and UFQFPN packages.

## 2.2.7 Incorrect backup domain reset with $V_{BAT}$ and $V_{DD}$ supplied by different power source

### Description

The backup domain reset may be missed upon backup domain power-on subsequent to a  $V_{BAT}$  power-off, in  $V_{BAT}$  mode, if the  $V_{BAT}$  voltage during the power-off phase drops to a few mV window before starting to rise again. In this critical window, the flip-flops are no longer able to safely retain the information, and the backup domain reset has not yet been triggered. This window is located in the range between 100 mV and 700 mV, with the exact position depending mainly on the device and on the temperature.

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

### Workaround

Apply the following measure to avoid the incorrect backup domain reset:

- In the application, let the  $V_{BAT}$  supply voltage fall to a level below 100 mV for more than 200 ms, before a new power-on.

If this workaround is not applicable 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 or exit from Shutdown mode, at least one backup register (called, for example, BackupTestRegister) must be previously programmed with a BKP\_REG\_VAL value with 16 bits set and 16 bits cleared. Robustness of this workaround can be significantly improved by using a CRC rather than registers. 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 bits modified by hardware.

The CRC result can be stored in the backup register instead of a fixed value. This value needs to be updated for each modification of values covered by CRC, such as by using CRC peripheral.

At the very beginning of the boot code, insert the following software sequence:

1. Check if the BORRSTF flag of RCC\_CSR is set (the reset is caused by a power-on).
2. If true, check that the BackupTestRegister content is different from BKP\_REG\_VAL, or the CRC recalculation is different from stored results, accordingly the chosen workaround implementation.
3. If true and if no tamper flag is set (when tamper detection is enabled), the reset is caused by a backup domain power-on. Apply the following sequence:
  - a. Enable the PWR clock in RCC, by setting the PWREN bit of RCC\_AHB1ENR2.
  - b. Enable backup domain access, by setting the DBP bit of PWR\_DBPR.
  - c. Reset the backup domain by:
    - i. Writing 0x0001 0000 to RCC\_BDCR, which sets the BDRST bit and clears other register bits that might not be reset.
    - ii. Reading RCC\_BDCR, to make the reset time long enough.
    - iii. Writing 0x0000 0000 to RCC\_BDCR, to clear the BDRST bit.
  - d. Clear BORRSTF, by setting the RMVF bit of RCC\_CSR.

## 2.3 HSP

### 2.3.1 Scheduler error flag wrongly set when executing CNN functions

#### Description

The SCHERRF (scheduler error) flag in the HSP\_ERR\_ISR register is wrongly set due to a timeout during the execution of a CNN function.

The SCHERRF flag cannot be cleared by the CPU as long as the HSP is running the CNN function.

The execution of the CNN function and the data integrity are not affected by the activation of SCHERRF flag.

### Workaround

When running CNN functions in applications where the hsp\_it global interrupt is required, make sure interrupts due to the scheduler timeout are disabled by clearing the SCHERRIE bit of the HSP\_ERR\_IER register. If the SCHERRF flag has been wrongly set, launch any other function to clear it.

## 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 Byte possibly dropped during an SDR read in clock mode 3 when a transfer gets automatically split

#### Description

When reading a continuous stream of data from sequential addresses in a serial memory, the OCTOSPI can interrupt the transfer and automatically restart it at the next address when features generating transfer splits (CSBOUND, REFRESH, TIMEOUT or MAXTRAN) are active. Thus, a single continuous transfer can effectively be split into multiple smaller transfers.

When the OCTOSPI is configured to use clock mode 3 (CKMODE bit of the OCTOSPI\_DCR1 register set) and a continuous stream of data is read in SDR mode (DDTR bit of the OCTOSPI\_CCR register cleared), the last byte sent by the memory before an automatic split gets dropped, thus causing all the subsequent bytes to be seen one address earlier.

#### Workaround

Use clock mode 0 (CKMODE bit of the OCTOSPI\_DCR1 register cleared) when in SDR mode.

### 2.4.3 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
- 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.

### 2.4.4 At least six cycles memory latency must be set when DQS is used for HyperBus™ memories

#### Description

For HyperBus™ memories, the TACC[7:0] bitfield of the OCTOSPI\_HLCR register enables the setting of the memory latency in number of clock cycles. These dummy cycles are inserted between the address and the data phases during read operations.

When the DQS signal is used for HyperBus™ memories, and the number of latency clock cycles programmed in TACC[7:0] is lower than six, a deadlock occurs during read operations.

#### Workaround

Configure the memory and the octo-SPI controller to have at least six clock cycles of latency.

### 2.4.5 Data write discarded in memory-mapped mode if a write to a misaligned address is directly followed by a request to the same address

#### Description

In memory-mapped mode with DQS enabled, a data write is discarded if the write targets a misaligned address and is directly followed by a request (cycle by cycle on AHB) to the same address.

An address is misaligned if the address is odd and the OCTOSPI is configured to send two bytes of data to the memory on every cycle that is targeted by one of the following modes:

- Octal DTR
- Dual-quad DTR

#### Workaround

Use one of the following measures:

- Configure the OCTOSPI to issue commands to aligned addresses, that is to an even address when two bytes are transferred during each clock cycle.
- Avoid consecutive back-to-back (AHB cycle by cycle) accesses to the memory after a write to a memory mapped at the same address. Instead, insert a NOP (no operation) or a software delay between the two accesses.

### 2.4.6 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.

#### 2.4.7 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 (ADDR[27:2]), 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.

#### 2.4.8 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

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

#### 2.4.9 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 OCTOSPI\_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.

#### 2.4.10 OCTOSPI external memory read issue after exiting Stop 2 or Stop 3 mode when using DQS and delay block

##### Description

After exiting Stop 2 or Stop 3 mode, the first read in the OCTOSPI external memory can return erroneous data when the following condition is met:

- OCTOSPI DQS is enabled
- Delay block is enabled

### Workaround

After exiting Stop 2 or Stop 3 mode, make a dummy read from the external memory before reading the required data.

## 2.5 SDMMC

### 2.5.1 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.

## 2.6 ADC

### 2.6.1 In combined regular simultaneous plus alternate trigger mode, stopping injected conversion may delay regular conversion

#### Description

In dual ADC combined regular simultaneous plus alternate trigger mode, the resumption of an active regular conversion may be delayed by a few ADC clock cycles when an injected trigger event coincides with stopping, by application, an ongoing injected conversion.

*Note:* The dual ADC combined regular simultaneous plus alternate trigger mode is selected by setting the DUAL[4:0] bitfield of the ADCC\_CCR register to 0x02. To stop an ongoing injected conversion, the software sets the JADSTP bit of the ADC\_CR register.

#### Workaround

- Design the application so as to avoid injected trigger events from coinciding with stopping, by software or DMA, an ongoing regular conversion.
- Stop the regular conversion before stopping the injected conversion.

### 2.6.2 In certain dual ADC modes with regular oversampling continued mode enabled, the JEOC flag is not set at the end of the new injected conversion

#### Description

In dual ADC regular simultaneous or interleaved mode, when the regular oversampling is enabled in continued mode, and injected conversions are ongoing on both ADCs, if an injected conversion is stopped and a new conversion is started for the same ADC, the JEOC flag is not set at the end of the new injected conversion.

*Note:* The regular simultaneous and interleaved modes are selected by setting the DUAL[4:0] bitfield of the ADCC\_CCR register to 0x06 and 0x07, respectively. The regular oversampling continued mode is enabled by setting the ROVSE bit and clearing the ROVSM bit of the ADC\_CFGR2 register. To start a new injected conversion and stop the ongoing one, the software must set the JADSTART and JADSTP bits of the ADC\_CR register, respectively. When set, the JEOC flag of the ADC\_ISR register indicates the end of the injected channel conversion.

### Workaround

Apply one of the following measures:

- Use one of the following modes instead of the regular simultaneous or the interleaved mode:
  - Combined regular simultaneous + injected simultaneous mode (DUAL[4:0] = 0x01)
  - Combined regular simultaneous + alternate trigger mode (DUAL[4:0] = 0x02)
  - Combined interleaved mode + injected simultaneous mode (DUAL[4:0] = 0x03)
- If an injected conversion is already ongoing on one ADC, avoid triggering a new injected conversion on the other ADC after stopping the ongoing conversion.
- Instead of the regular oversampling continued mode, use the resumed mode (ROVSM = 1).

### 2.6.3 In certain dual ADC modes with DMA one-shot mode enabled, JADSTART and ADSTART may not be cleared if JADSTP and ADSTP are set at the same time

#### Description

In certain dual ADC modes (DUAL[4:0] = 0x05, 0x06, 0x07, and 0x09 in the ADCC\_CCR register), ADSTART and JADSTART bits may not be cleared, if the DMA is configured in one-shot mode, and the stopping of the ongoing injected conversion by software coincides with the stopping of the ongoing regular conversion by DMA.

*Note:* The DMA one-shot mode is enabled by setting the DMNGT[1:0] bitfield of the ADC\_CFGR2 register to 0b01. To stop an injected conversion, the software must set the JADSTP bit of the ADC\_CR register.

#### Workaround

Apply one of the following measures:

- Use one of the following modes instead of the injected simultaneous, regular simultaneous or the interleaved mode:
  - Combined regular simultaneous + injected simultaneous mode (DUAL[4:0] = 0x01)
  - Combined regular simultaneous + alternate trigger mode (DUAL[4:0] = 0x02)
  - Combined interleaved mode + injected simultaneous mode (DUAL[4:0] = 0x03)
- Avoid using DMA one-shot mode to manage regular conversions. Instead, use interrupt or polling.

### 2.6.4 Wrong injected conversion data in bulb sampling mode

#### Description

In bulb sampling mode, if regular conversions are enabled but not running, and an injected conversion is requested just after the ADSTP bit of the ADC\_CR register is set, the injected conversion data obtained on the selected channel may be wrong.

*Note:* The bulb sampling mode is selected by setting the BULB bit of the ADC\_CR register.

#### Workaround

Avoid triggering injected conversions when the ADSTP bit is set.

### 2.6.5 Shifted master and slave sequence in interleaved discontinuous mode

#### Description

In dual ADC interleaved mode, when the discontinuous mode is enabled on regular channels, and the next trigger arrives before the end of the sequence, the master and slave interleave sequence may be shifted.

*Note:* The dual ADC interleaved modes are selected by setting the DUAL[4:0] bitfield of the ADCC\_CCR register to 0x03 or 0x07. The discontinuous mode is enabled through the DISCEN bit of the ADC\_CFGR1 register.

#### Workaround

Avoid triggering a new conversion when the ongoing conversion is not complete.

## 2.6.6 When the ADC clock is derived from the AHB clock, the injected conversion latency is not respected if the injected trigger coincides with the stopping of the regular conversion

### Description

When the ADC clock is derived from the AHB clock, and the analog-to-digital conversion is triggered by a timer, the latency between the trigger and the start of the ADC sampling is fixed. When both injected and regular conversions are enabled, if the injected trigger coincides with the stopping of regular conversion, the latency of injected conversions becomes one clock cycle shorter than the expected latency.

*Note:* To stop an ongoing regular conversion, the software sets the ADSTP bit of the ADC\_CR register.

### Workaround

Apply one of the following measures:

- Avoid triggering injected conversions when the ADSTP bit is set.
- When an injected trigger is expected, keep the ADSTP bit cleared.

## 2.6.7 When injected conversions are enabled and the ADC clock is slower than the ADC kernel clock, a wrong context can be applied in certain conditions

### Description

When the injected queue (JSQR) is enabled (by clearing the JQM bit of the ADC\_CFGR1 register), if the JSQR queue becomes empty, the old context is maintained when a new trigger occurs, and the new sequence starts with the old context. When this sequence completes, the JEOS flag is set in the ADC\_ISR register.

The JSQR queue can be updated by software, and started by a hardware trigger at any time. If these two actions occur simultaneously with the setting of JEOS, and the `adc_hclk` clock is slower than the ADC kernel clock, the JSQR queue is not updated at the right moment, thus resulting in an unpredictable sequence execution:

- The previous sequence is executed instead of the programmed sequence, or
- The programmed sequence is executed but it is triggered by the previous external event, or
- The injected conversion sequence has started with the previous context, but during the sequence execution, the old context is overridden by the new one.

### Workaround

When the ADC clock is slower than the ADC kernel clock and the injected queue is empty, avoid a hardware trigger and a JSQR update from occurring simultaneously with the setting of JEOS. This can be done by disabling the injected trigger when a JSQR update is performed.

## 2.6.8 In certain dual modes, the fixed trigger latency for the injected conversions may not be respected

### Description

If the application operates in regular simultaneous or interleaved mode (`DUAL[4:0] = 0x06` or `0x07` in the `ADCC_CCR` register), the injected conversions can be activated either on the master or on the slave ADC. When a fixed trigger latency is configured for injected conversions, the regular conversions on both ADCs are interrupted by the injected trigger. In this case, the previous conversion trigger latency and the current conversion trigger latency may differ by one clock cycle.

### Workaround

If the regular simultaneous mode or the interleaved mode is used (`DUAL[4:0]=0x06` or `0x07`), and the application needs injected conversions with a fixed trigger latency for both ADCs, apply one of the following measures:

- Select another dual mode, either `DUAL[4:0] = 0x01` or `0x03`.
- Make sure that no injected trigger event occurs when regular conversions are stopped, by setting the ADSTP bit in the ADC\_CR register.

## 2.6.9 In simultaneous regular mode, stopping an injected conversion may shift the next regular conversion master and slave timing by one clock cycle

### Description

In simultaneous regular mode (DUAL[4:0] = 0x06 in the ADCC\_CCR register), injected conversions can be activated either on ADC1 or ADC2. When regular conversions are ongoing, if an injected conversion stop (JADSTP) occurs at the same time as an injected trigger event, then ADC1 and ADC2 timing may be shifted by one clock cycle.

### Workaround

Apply one of the following measures:

- Avoid the following events from occurring simultaneously:
  - Triggering injected conversions
  - Stopping injected conversions
  - Enabling regular conversion in simultaneous regular mode (DUAL[4:0] = 0x06)
- Stop regular conversions before stopping injected conversions.
- Stop injected trigger source before stopping injected conversions.
- Configure DUAL[4:0] to 0x01 or 0x02.

## 2.6.10 The OVR flag is set if EOC is cleared by writing 1 to the EOC bit of ADC\_ISR instead of reading the data register

### Description

When a regular ADC conversion completes, the EOC (end of conversion) flag is set. Reading the data register (ADC\_DR, ADC\_CDR, or ADC\_CDR2) clears the EOC flag. However, the CPU can also clear the EOC flag by writing 1 to the EOC bit in the ADC\_ISR register without reading the data register.

If eight conversions complete, but the EOC flag is cleared only by this write operation (without reading the data register), the OVR (overrun) flag may be set.

### Workaround

To discard ADC data when processed by the CPU, perform a dummy read of the data register instead of clearing the EOC flag by writing to ADC\_ISR.

## 2.7 DAC

### 2.7.1 Incorrect DAC output voltage due to DAC kernel clock setting

#### Description

The DAC output voltage might be incorrect when HCLK is selected as DAC kernel clock, and a prescaler is applied. This configuration occurs when the ADCDACSEL[1:0] and ADCDACPRE[3:0] bitfields of the RCC\_CCIPR2 register are set to select HCLK as the clock source and to a division factor other than 1, respectively.

*Note:* The ADC and DAC share a common clock source. If the ADC is used in the same application as the DAC, set the clock frequency so as to respect the ADC constraints specified in the datasheet.

#### Workaround

1. Disable the prescaler by writing, respectively, 00 and 0000 to the ADCDACSEL[1:0] and ADCDACPRE[3:0] bitfields of the RCC\_CCIPR2 register.
2. Select HSE or MSIK as the DAC kernel clock by writing 01 or 11 to the ADCDACSEL[1:0] bitfield of the RCC\_CCIPR2 register.

## 2.8 VREFBUF

### 2.8.1 VREFBUF\_OUT voltage overshoots in Range 2 or Stop 1 mode

#### Description

The VREFBUF\_OUT output voltage overshoots when started:

- While the regulator is in Range 2 or
- While the device is in Stop 1 mode.

#### Workaround

Modify the regulator voltage Range to Range 1 before enabling the voltage reference buffer, then go back to Range 2 or Stop 1 mode.

## 2.9 RNG

### 2.9.1 RNG may report continuous seed errors in specific environmental conditions

#### Description

Under certain combinations of supply voltage, temperature, and process variation, one of the RNG analog noise sources may provide insufficient entropy. As a consequence, the RNG generates seed errors continuously.

If the application correctly implements the error handling procedure described in the “Error Management” section of the RNG chapter, it can reliably detect this condition and recover from continuous seed error generation.

#### Workaround

No application workaround is required or applicable as it is handled by the standard error management procedure.

## 2.10 TIM

### 2.10.1 Unexpected PWM output when using ocref\_clr

#### Description

In combined PWM mode 1, asymmetric PWM mode 1, or asymmetric PWM mode 2, using ocref\_clr can cause the tim\_ocxrefc output to be unexpectedly re-enabled or disabled. This behavior depends on the timing of when ocref\_clr is activated and deactivated.

#### Workaround

None.

## 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:

$$\{ARR, CMP\} \geq KER\_CLK / (2 * APB\_CLK),$$

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
- The repetition counter is set with REP[7:0] = 0x3 (odd value)

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

$$\{CMP, ARR\} \geq 16 \text{ MHz} / (2 * 1 \text{ MHz})$$

→ ARR must be  $\geq 8$  and CMP must be  $\geq 8$

*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 the 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.

## 2.12.2 RTC calendar read may return transient incorrect value when BYPSHAD = 0

### Description

When the BYPSHAD control bit in the RTC\_CR register is cleared, reading the RTC calendar registers (RTC\_SSR, RTC\_TR, and RTC\_DR) may seldom produce incorrect values. This issue occurs because internal timing can cause the asynchronous RTC calendar registers to be copied into their shadow registers during a transition of the asynchronous registers.

Since this copying process occurs every RTC kernel clock cycle, any erroneous value persists for only one RTC kernel clock cycle. The likelihood of this failure is very low and depends on several factors, including:

- RTC software configuration, such as the RTC kernel clock source, asynchronous prescaler factor, and calibration settings.
- APB clock frequency.
- Operating voltage and temperature.
- The timing of the register read operation.

Additionally, process variations may cause timing differences between samples, which can also influence the probability of this failure.

### Workaround

To avoid this condition and ensure consistent calendar readings:

- Set the BYPSHAD control bit to 1 in the RTC\_CR register to bypass shadow registers.
- Read the RTC calendar registers twice.
- Compare the two readings. If the values differ, repeat the read and compare sequence until two consecutive readings match.

This software method ensures a reliable calendar value under all operating conditions. For further details, see section *Reading the calendar – When the BYPSHAD control bit is set in the RTC\_CR register (bypass shadow registers)* in the reference manual.

## 2.12.3 Parasitic tamper detection when debugger is used in RDP Level 0

### Description

The internal tamper 6 flag (ITAMP6F) can be unexpectedly set in the TAMP status register (TAMP\_SR) when a debugger is connected in RDP Level 0, in case a switch to  $V_{BAT}$  occurs ( $V_{DD}$  is below the BOR0 threshold).

### Workaround

Keep internal tamper 6 flag disabled as long as debug is needed, and enable it once development phase is complete. The tamper flag cannot be set if no debug access is done.

## 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<sup>2</sup>C-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<sup>2</sup>C-bus SDA line when  $t_{SU;DAT}$  is smaller than one I2C kernel clock (I<sup>2</sup>C-bus peripheral clock) period: the previous SDA value is sampled instead of the current one. This can result in a wrong receipt of target 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<sup>2</sup>C-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 controller mode

### Description

In controller 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<sup>2</sup>C-bus transfer in controller mode and any such transfer continues normally.

### Workaround

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

## 2.14 USART

### 2.14.1 Wrong data received by SPI slave receiver in autonomous mode with CPOL = 1

#### Description

The SPI slave receiver device receives wrong data when all the following conditions are met:

- The USART is used in SPI master transmitter mode
- The autonomous mode is used
- The CPOL bit of the USART\_CR2 register is set

#### Workaround

When the autonomous mode is used, do not set the CPOL bit in USART\_CR2.

### 2.14.2 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.

### 2.14.3 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.*

## 2.15 LPUART

### 2.15.1 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.

## 2.16 SPI

### 2.16.1 Possible corruption of last-received data depending on CRCSIZE setting

#### Description

With the CRC calculation disabled (CRCEN = 0), the transfer size bitfield set to a value greater than zero (TSIZE[15:0] > 0), and the length of CRC frame set to less than 8 bits (CRCSIZE[4:0] < 00111), the last data received in the RxFIFO may be corrupted.

#### Workaround

Keep the CRCSIZE[4:0] bitfield at its default setting (00111) during the data reception if CRCEN = 0 and TSIZE[15:0] > 0.

### 2.16.2 MODF flag cannot generate interrupt

#### Description

Mode fault detection results in disabling SPI. With the MODFIE bit of the SPI\_IER register set, the mode fault flag (MODF) going high is expected to trigger an interrupt. However, disabling SPI unduly blocks this interrupt request.

#### Workaround

To detect a mode fault event, poll the MODF flag by software.

### 2.16.3 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.

### 2.16.4 Master communication suspension fails in autonomous mode

#### Description

The SPI peripheral is blocked regardless of the completion of the ongoing data frame transaction, and the SUPSF flag is never set, when:

- The master provides a communication triggered in autonomous mode (TRIGEN=1 of the SPI\_AUTOOCR register), and
- The suspension of the ongoing transaction is applied by setting the CSUSP bit through the smart DMA in Stop mode.

#### Workaround

None.

*Note:* The user software must avoid any master suspension in Stop mode while the master operates in autonomous mode and waits for EOT if TSIZE is greater than 0. If an endless transaction is applied (TSIZE = 0), the suspension is the only way to stop the ongoing transaction. Then to unblock the peripheral, the software must disable SPI then apply the hardware reset. Otherwise, the system cannot proceed to the next transaction.

### 2.16.5 SPE may not be cleared upon MODF event

#### Description

The failure described applies to multi-master topology when the device is configured to monitor the SS input signal by hardware (SSM = 0, SSOE = 0 of the SPI\_CFG2 register).

If the software sets the SPE (SPI enable) bit of the SPI\_CR1 register at the instant of the SS signal transiting to its active logical level, the resulting MODF event duly switches the SPI into slave mode, but it fails to clear the SPE bit and thus disable the SPI.

*Note:* The SS active logical level is the one that matches the SSIOP bit of the SPI\_CFG2 register.

#### Workaround

Whenever MODF event fails to clear the SPE bit, do it by software.

### 2.16.6 SPI slave stalls with masters not providing extra SCK periods upon *Not ready* signaling

#### Description

In Stop mode, the device SPI operating as slave with the *Ready* signaling enabled (the RDIOM of the SPI\_CFG2 register set) may stall and never retrieve the *Ready* state. This occurs when SCK stops immediately after *Not ready* status.

*Note:* STM32 devices supporting the *Ready* signaling and operating as SPI master provide some extra SCK periods upon detecting *Not ready* signal, thus allowing the SPI slaves to operate correctly.

### Workaround

If in the application, there is an SPI master that stops SCK immediately upon *Not ready* signal, without providing some extra SCK periods, do not enable the *Ready* signaling.

## 2.16.7 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 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.

## 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](http://www.psacertified.org)) and/or Security Evaluation standard for IoT Platforms ([www.trustcb.com](http://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](http://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 4. Document revision history**

Date	Version	Changes
23-Feb-2026	1	Initial release

## Contents

<b>1</b>	<b>Summary of device errata</b>	<b>2</b>
<b>2</b>	<b>Description of device errata</b>	<b>5</b>
2.1	Core	5
2.1.1	Access permission faults are prioritized over unaligned device memory faults	5
2.2	System	5
2.2.1	USART2 cannot be used in autonomous mode or to wake up from stop mode	5
2.2.2	PA1 and PA7 limited usage when low-leakage OPAMP1_VINM and OPAMP2_VINM are used	5
2.2.3	SRAMs retained in Stop 1 mode may be corrupted on exit by system reset	6
2.2.4	HSI remains enabled after wake-up from Stop mode when debug in Stop mode is enabled	6
2.2.5	LSE in bypass mode may not be functional	6
2.2.6	LSE crystal oscillator may be disturbed by transitions on PC13	6
2.2.7	Incorrect backup domain reset with $V_{BAT}$ and $V_{DD}$ supplied by different power source	7
2.3	HSP	7
2.3.1	Scheduler error flag wrongly set when executing CNN functions	7
2.4	OCTOSPI	8
2.4.1	Memory-mapped write error response when DQS output is disabled	8
2.4.2	Byte possibly dropped during an SDR read in clock mode 3 when a transfer gets automatically split	8
2.4.3	Deadlock or write-data corruption after spurious write to a misaligned address in OCTOSPI_AR register	8
2.4.4	At least six cycles memory latency must be set when DQS is used for HyperBus™ memories	9
2.4.5	Data write discarded in memory-mapped mode if a write to a misaligned address is directly followed by a request to the same address	9
2.4.6	Setting the ABORT bit does not generate an error on the AHB bus for undefined-length incremental burst transfers	9
2.4.7	Read data corruption when a wrap transaction is followed by a linear read to the same MSB address	10
2.4.8	Transactions are limited to 8 Mbytes in OctaRAM™ memories	10
2.4.9	Variable latency is not supported when a refresh collision occurs during a write access to some OctaRAM™ memories	10
2.4.10	OCTOSPI external memory read issue after exiting Stop 2 or Stop 3 mode when using DQS and delay block	10
2.5	SDMMC	11
2.5.1	Command response and receive data end bits not checked	11
2.6	ADC	11
2.6.1	In combined regular simultaneous plus alternate trigger mode, stopping injected conversion may delay regular conversion	11

2.6.2	In certain dual ADC modes with regular oversampling continued mode enabled, the JEOP flag is not set at the end of the new injected conversion . . . . .	11
2.6.3	In certain dual ADC modes with DMA one-shot mode enabled, JADSTART and ADSTART may not be cleared if JADSTP and ADSTP are set at the same time . . . . .	12
2.6.4	Wrong injected conversion data in bulb sampling mode . . . . .	12
2.6.5	Shifted master and slave sequence in interleaved discontinuous mode . . . . .	12
2.6.6	When the ADC clock is derived from the AHB clock, the injected conversion latency is not respected if the injected trigger coincides with the stopping of the regular conversion . . . . .	13
2.6.7	When injected conversions are enabled and the ADC clock is slower than the ADC kernel clock, a wrong context can be applied in certain conditions . . . . .	13
2.6.8	In certain dual modes, the fixed trigger latency for the injected conversions may not be respected . . . . .	13
2.6.9	In simultaneous regular mode, stopping an injected conversion may shift the next regular conversion master and slave timing by one clock cycle. . . . .	14
2.6.10	The OVR flag is set if EOC is cleared by writing 1 to the EOC bit of ADC_ISR instead of reading the data register. . . . .	14
2.7	DAC . . . . .	14
2.7.1	Incorrect DAC output voltage due to DAC kernel clock setting . . . . .	14
2.8	VREFBUF . . . . .	15
2.8.1	V <sub>REFBUF_OUT</sub> voltage overshoots in Range 2 or Stop 1 mode. . . . .	15
2.9	RNG . . . . .	15
2.9.1	RNG may report continuous seed errors in specific environmental conditions . . . . .	15
2.10	TIM . . . . .	15
2.10.1	Unexpected PWM output when using ocref_clr. . . . .	15
2.11	LPTIM. . . . .	15
2.11.1	Device may remain stuck in LPTIM interrupt when entering Stop mode . . . . .	15
2.11.2	ARRM and CMPM flags are not set when APB clock is slower than kernel clock . . . . .	16
2.11.3	Interrupt status flag is cleared by hardware upon writing its corresponding bit in LPTIM_DIER register . . . . .	16
2.12	RTC and TAMP . . . . .	16
2.12.1	Alarm flag may be repeatedly set when the core is stopped in debug . . . . .	16
2.12.2	RTC calendar read may return transient incorrect value when BYPSHAD = 0 . . . . .	17
2.12.3	Parasitic tamper detection when debugger is used in RDP Level 0 . . . . .	17
2.13	I2C . . . . .	17
2.13.1	Wrong data sampling when data setup time (t <sub>SU;DAT</sub> ) is shorter than one I2C kernel clock period. . . . .	17
2.13.2	Spurious bus error detection in controller mode . . . . .	18
2.14	USART . . . . .	18
2.14.1	Wrong data received by SPI slave receiver in autonomous mode with CPOL = 1 . . . . .	18
2.14.2	Received data may be corrupted upon clearing the ABREN bit. . . . .	18

2.14.3	Noise error flag set while ONEBIT is set . . . . .	18
<b>2.15</b>	<b>LPUART</b> . . . . .	<b>19</b>
2.15.1	Possible LPUART transmitter issue when using low BRR[15:0] value . . . . .	19
<b>2.16</b>	<b>SPI</b> . . . . .	<b>19</b>
2.16.1	Possible corruption of last-received data depending on CRCSIZE setting . . . . .	19
2.16.2	MODF flag cannot generate interrupt . . . . .	19
2.16.3	RDY output failure at high serial clock frequency . . . . .	20
2.16.4	Master communication suspension fails in autonomous mode . . . . .	20
2.16.5	SPE may not be cleared upon MODF event . . . . .	20
2.16.6	SPI slave stalls with masters not providing extra SCK periods upon <i>Not ready</i> signaling . . . . .	20
2.16.7	Truncation of SPI output signals after EOT event . . . . .	21
<b>2.17</b>	<b>FDCAN</b> . . . . .	<b>21</b>
2.17.1	Desynchronization under specific condition with edge filtering enabled . . . . .	21
2.17.2	Tx FIFO messages inverted under specific buffer usage and priority setting . . . . .	21
<b>2.18</b>	<b>USB</b> . . . . .	<b>22</b>
2.18.1	Buffer description table update completes after CTR interrupt triggers . . . . .	22
<b>Important security notice</b> . . . . .		<b>23</b>
<b>Revision history</b> . . . . .		<b>24</b>



**IMPORTANT NOTICE – READ CAREFULLY**

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers' market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to [www.st.com/trademarks](http://www.st.com/trademarks). All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2026 STMicroelectronics – All rights reserved