STM32G544xx and STM32G534xx device errata

Applicability

This document applies to the part numbers of STM32G544xx and STM32G534xx 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 RM0514.

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
STM32G544xx	STM32G544KB, STM32G544HB, STM32G544CB, STM32G544RB, STM32G544MB, STM32G544C8, STM32G544H8, STM32G544K8, STM32G544M8, STM32G544R8, STM32G544CC, STM32G544HC, STM32G544KC, STM32G544MC, STM32G544RC
STM32G534xx	STM32G534KC, STM32G534HC, STM32G534CC, STM32G534RC, STM32G534MC

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(2)	REV_ID(1)
STM32G544xx, STM32G534xx	A	0x1000
	Z	0x1001

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

1 Summary of device errata

The following table gives a quick reference to the STM32G544xx and STM32G534xx device limitations and their status:

- **A** Limitation applicable. Workaround available
- **N** Limitation applicable. No workaround available
- **P** Limitation applicable. Partial workaround available
- **-** 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	Rev. Z
Core	2.1.1	Access permission faults are prioritized over unaligned device memory faults	N	N
System	2.2.1	Access to the last byte of the flash memory generates Hardfault	A	A
	2.2.2	CPU execution freeze upon first erase or program operation after power-on or wake-up from Standby	A	A
	2.2.3	EXTI line 58 is a configurable event	N	N
	2.2.4	Shift in temperature monitoring thresholds	N	-
	2.2.5	Accessing specific TPIU registers causes CPU lockup	A	A
	2.2.6	ECC error when reading EEPROM after flash memory read	A	A
	2.2.7	SRAMx_RST option bits have an immediate effect	A	A
	2.2.8	Standby mode exit failure when NRST reset occurs during Standby entry sequence	A	A
	2.2.9	Incorrect behavior of the JTAGIORETEN bit in the PWR_IORETR register	P	P
ADC	2.3.1	In certain dual modes, the fixed trigger latency for the injected conversions may not be respected	A	A
	2.3.2	In simultaneous regular mode, stopping an injected conversion may shift the next regular conversion master and slave timing by one clock cycle	A	A
	2.3.3	Injected conversions do not work when SMPTRIG bit is set, regular conversions start then stop, and SMPTRIG bit is cleared	A	A
	2.3.4	Extra ADC consumption from V_{DDA}	N	N
DAC	2.4.1	Extra error for DAC1/DAC2 in unbuffered mode	N	N
	2.4.2	DAC trigger latency is not constant under given conditions	A	A
RNG	2.5.1	RNG may report continuous seed errors in specific environmental conditions	N	N
HRTIM	2.6.1	Delayed idle protection malfunction when IDLESx = 1	N	N
	2.6.2	Deadtime insertion mode not functional	N	N
	2.6.3	Negative deadtime not supported in push-pull mode	N	N
	2.6.4	Deadtime slightly shortened in push-pull mode	N	N

Function	Section	Limitation	Status	
			Rev. A	Rev. Z
HRTIM	2.6.5	Generated pulse shorter than $t_{HRTIMER}$ period or missing for close set-reset or reset-set events	N	N
LPTIM	2.7.1	Device may remain stuck in LPTIM interrupt when entering Stop mode	A	A
	2.7.2	ARRM and CMPM flags are not set when APB clock is slower than kernel clock	A	A
	2.7.3	Interrupt status flag is cleared by hardware upon writing its corresponding bit in LPTIM_DIER register	N	N
	2.7.4	LPTIM cannot use HSI clock in Stop mode	A	A
RTC	2.8.1	Alarm flag may be repeatedly set when the core is stopped in debug	N	N
	2.8.2	RTC calendar read may return transient incorrect value when BYPSHAD = 0	A	A
	2.8.3	System stuck if some RTC or TAMP registers are written when write protection is enabled	N	N
I2C	2.9.1	Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period	P	P
	2.9.2	Spurious bus error detection in controller mode	A	A
USART	2.10.1	Received data may be corrupted upon clearing the ABREN bit	A	A
	2.10.2	Noise error flag set while ONEBIT is set	N	N
LPUART	2.11.1	Possible LPUART transmitter issue when using low BRR[15:0] value	P	P
SPI	2.12.1	RDY output failure at high serial clock frequency	N	N
FDCAN	2.13.1	Desynchronization under specific condition with edge filtering enabled	A	A
	2.13.2	Tx FIFO messages inverted under specific buffer usage and priority setting	A	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 r0p1 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 Access to the last byte of the flash memory generates Hardfault

Description

When accessing the last byte of the flash memory, Hardfault is generated for the devices with less than 256 Kbytes of flash memory and with the following characteristics:

- 64 Kbytes of flash memory and access to the 0x0800 FFFF address
- 128 Kbytes of flash memory and access to the 0x0801 FFFF address

Workaround

Perform half-word or word access instead of byte access:

- For devices with 64 Kbytes of flash memory:
  - Half-word access to the 0x0800 FFFE address
  - Word access to the 0x0800 FFFC address
- For devices with 128 Kbytes of flash memory:
  - Half-word access to the 0x0801 FFFE address
  - Word access to the 0x0801_FFFC address

2.2.2 CPU execution freeze upon first erase or program operation after power-on or wake-up from Standby

Description

The first flash memory erase or program operation performed after power-on sequence or after wake-up from Standby makes the flash memory inaccessible for about 60 µs. Any CPU fetch or read access to the flash memory during this period freezes the CPU execution until the flash memory becomes accessible again.

This also concerns the read-while-write operation where the code executes in one bank, and it writes and reads another bank of the flash memory.

Note: Any subsequent flash memory erase or program operation is free of this issue.

Workaround

If the application is sensitive to this limitation, apply the following measure:

1. Relocate the interrupt vector table and critical interrupt handlers to the SRAM.
2. Execute the code for the first flash memory erase or program operation in the SRAM.

Ensure that the code executed in the SRAM does not access the flash memory during the period of its inaccessibility.

Once the first erase or program operation is completed, the software can resume execution in the flash memory.

2.2.3 EXTI line 58 is a configurable event

Description

EXTI line connections 58 (LSECSSD)is supposed to generate a direct event. Instead, it behaves as a configurable event line.

Workaround

None.

2.2.4 Shift in temperature monitoring thresholds

Description

The temperature monitoring thresholds are shifted versus the specification in the datasheet. The device exhibits the following values:

Table 4. Temperature monitoring characteristics

Symbol	Parameter	Min	Typ	Max	Unit
$Temp_{high}$	High temperature monitoring	120	126	132	°C
$Temp_{low}$	Low temperature monitoring	-41	-35	-29	

Workaround

None.

2.2.5 Accessing specific TPIU registers causes CPU lockup

Description

Accessing the following Trace port interface unit (TPIU) registers in the trace port clock domain causes the Cortex®-M33 core to lock up, which halts the entire system.

- TPIU_SSPSR
- TPIU_CSPSR
- TPIU_ACPR
- TPIU_SPPR
- TPIU_FFSR
- TPIU_FFCR
- TPIU_PSCR

Workaround

To avoid the CPU lockup, follow these steps:

1. Set Cortex®-M33 core TRCENA bit of the Debug exception and monitor control register (DEMCR).
2. Ensure that noninvasive debug remains enabled, as the TRCENA output is masked by the debug authentication setting.

Note: Refer to the "Cortex®-M33 Technical Reference Manual" available on the Arm website for information on registers of the Cortex®-M33 core.

2.2.6 ECC error when reading EEPROM after flash memory read

Description

An ECC error may occur during a read from the start of EEPROM data area immediately following a read from the final flash memory area located just before the EEPROM data area. This happens only when the flash memory is configured with prefetch enabled. The error is reported in the FLASH_ECCDR or FLASH_ECCDETR registers.

Workaround

Ensure that the linker does not place any code or data in the last word of the flash address space, adjacent to the start of the EEPROM data area. Additionally, the EEPROM data read driver must not be located at the very end of the flash memory.

2.2.7 SRAMx_RST option bits have an immediate effect

Description

Clearing the SRAMx_RST 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.

Note: This reported SRAM erase is performed only once after system reset, and only if the SRAMxRST option bit is set to 1 during system reset.

Workaround

The application must be reset immediately after SRAMx_RST clear.

2.2.8 Standby mode exit failure when NRST reset occurs during Standby entry sequence

Description

The device may fail to exit Standby mode when an NRST reset occurs within a specific time window when entering Standby.

Workaround

Use the external pull-up (PU) or pull-down (PD) resistor on the PA15 pin to keep the voltage stable during wake-up from Standby mode by the NRST pin.

2.2.9 Incorrect behavior of the JTAGIORETEN bit in the PWR_IORETR register

Description

When the JTAGIORETEN bit in the PWR_IORETR register is set, the previously latched pull-up / pull-down configuration is applied immediately to the JTAG pins.

The applied configuration corresponds to the pull-up / pull-down state latched during the previous entry into Standby mode when the JTAGIORETEN bit was enabled.

Workaround

Set the JTAGIORETEN bit immediately before entering into Standby mode to minimize the time while previously latched pull-up/pull-down configuration is applied to the JTAG pins.

2.3 ADC

2.3.1 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.

2.3.2 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.

2.3.3 Injected conversions do not work when SMPTRIG bit is set, regular conversions start then stop, and SMPTRIG bit is cleared

Description

Injected conversions do not work after the following sequence of events:

1. The SMPTRIG bit of the ADC_CFGR2 register is set (sampling time control trigger mode enabled).
2. Regular conversions are started then stopped.
3. The SMPTRIG is cleared.

Workaround

If SMPTRIG is set, disable the ADC, clear the SMPTRIG, and enable again the ADC before starting injected conversions.

2.3.4 Extra ADC consumption from V_{DDA}

Description

Each ADC draws typically 0.55 µA (with a maximum of 12 µA) from the V_{DDA} supply at 25°C, even when disabled, causing extra current consumption.

Workaround

None.

2.4 DAC

2.4.1 Extra error for DAC1/DAC2 in unbuffered mode

Description

When the DAC1/DAC2 operates in unbuffered mode with output on the pin, an extra error up to 10 LSB may occur if the output voltage exceeds the reference DAC voltage minus 100 millivolts.

Workaround

None.

2.4.2 DAC trigger latency is not constant under given conditions

Description

Selecting the rcc_hrtim_ker_ck kernel clock for clocking the DAC (DACxSEL[2:0] = 001 in the RCC_CCIPR2 register) causes a trigger event latency of two clock pulses even if the AHB clock is undivided.

Workaround

To ensure DAC trigger event latency without jitter, only use the DAC kernel clock (DACxSEL[2:0] = 000).

2.5 RNG

2.5.1 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.

2.6 HRTIM

2.6.1 Delayed idle protection malfunction when IDLESx = 1

Description

Upon delayed idle or balanced idle mode entry, the outputs are forced in their IDLESx state.

However, due to the defect, the IDLESx state may not be applied if the delayed protection is entered during the deadtime duration.

This defect is present only when both:

- The deadtime logic is enabled.
- One of the IDLESx bit is set to 1 (IDLES1 = 0/IDLES2 = 1, or IDLES1 = 1/ IDLES2 = 0), to have one output active during the delayed protection.

The consequence differs with respect to the deadtime polarity:

- With positive deadtime, both outputs may become inactive (as if IDLES1 = IDLES2 = 0).
- With negative deadtime, both outputs may become active potentially creating cross-conduction.

Workaround

None.

2.6.2 Deadtime insertion mode not functional

Description

When the deadtime insertion mode is enabled (DTM[1:0] = 01 or 10 in the HRTIM_DTxR register), deadtime durations are not respected in certain conditions, potentially creating deadtime violations.

The deadtime insertion modes must not be used, and it is mandatory to have DTM[1:0] = 00.

This defect does not affect the legacy deadtime mode (DTM[1:0] = 00 and DTEN = 1).

Workaround

None.

The deadtime insertion mode can be used for temporarily disabling an output. For example, when a half-bridge buck converter starts operating in discontinuous mode, the low side switch must no longer be switched. In this case, it is possible to dynamically update the DTEN bit to disable the complementary operation, since the DTEN bit can be preloaded.

For implementing a cycle-by-cycle protection without deadtime insertion, refer to the example application note AN4539 (HRTIM cookbook).

2.6.3 Negative deadtime not supported in push-pull mode

Description

In push-pull mode, the negative deadtime configuration (STRx = 1 and/or SDTFx = 1 in HRTIM_TIMxCR) provides in a null deadtime.

Workaround

None.

2.6.4 Deadtime slightly shortened in push-pull mode

Description

In push-pull mode, with positive deadtime configuration, the effective deadtime falling duration may be slightly reduced compared to the value programmed in DTFx[8:0] bitfield.

The maximum error is strictly below1 hrtim_ker_ck clock period (31/32 of hrtim_ker_ck).

Workaround

1. The deadtime falling value must be increased by 1 hrtim_ker_ck. This guarantees a safe minimum deadtime value and prevents any risk of cross-conduction. Still, the transition from inactive to active of output 2 may jitter (+0/-1 hrtim_ker_ck).
2. The push-pull guarantees nonoverlapping signals by design. Another option to insert deadtime is to use a compare event to set the output at the beginning of the push or pull phase, after a delay corresponding to the deadtime value.

2.6.5 Generated pulse shorter than $t_{HRTIMER}$ period or missing for close set-reset or reset-set events

Description

When a *set (reset)* event occurs within one $t_{HRTIMER}$ period and a *reset (set)* event within the next $t_{HRTIMER}$ period, the generated pulse is expected to last one full $t_{HRTIMER}$ period. However, if the interval between *set (reset)* and *reset (set)* events is shorter than one $t_{HRTIMER}$ period, the generated pulse can be shorter than one $t_{HRTIMER}$ period or missing.

Workaround

None.

2.7 LPTIM

2.7.1 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.

2.7.2 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
- 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} ≥ 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 the LPTIM_RCR register (=3, odd) to assess the risk of issue.

2.7.3 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.

2.7.4 LPTIM cannot use HSI clock in Stop mode

Description

LPTIM cannot be clocked by hsi_ker_ck in Stop mode.

hsi_ker_ck can be used as clock for LPTIM in Run or Sleep mode only.

Workaround

Use lsi_ck or lse_ck as clock for LPTIM in Stop mode.

2.8 RTC

2.8.1 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.

2.8.2 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.

2.8.3 System stuck if some RTC or TAMP registers are written when write protection is enabled

Description

The system becomes stuck if some RTC or TAMP registers bits are written to 1 whereas the write protection is enabled in the PWR controller (= 0). The bits creating the failure are:

- Any bit of the RTC status clear register (RTC_SCR)
- Any bit of the TAMP status clear register (TAMP_SCR)
- The BKERASE bit of the TAMP control register 2 (TAMP_CR2)

This does not occur if the software sequence described in reference manual is correctly executed.

Workaround

None. Only a system reset can recover from this stuck state.

2.9 I2C

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

Description

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

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

The device does not correctly sample the I^2C-bus SDA line when $t_{SU;DAT}$ is smaller than one I2C kernel clock (I^2C-bus peripheral clock) period: the previous SDA value is sampled instead of the current one. This can result in a wrong receipt of 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^2C-bus standard, the minimum I2CCLK frequencies are as follows:

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

2.9.2 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^2C-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.

2.10 USART

2.10.1 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.

2.10.2 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.

2.11 LPUART

2.11.1 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.

2.12 SPI

2.12.1 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.

2.13 FDCAN

2.13.1 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.

2.13.2 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
```

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
28-Oct-2026	1	Initial release.

Contents

1 **Summary of device errata** 2
2 **Description of device errata** 4
2.1 Core 4
2.1.1 Access permission faults are prioritized over unaligned device memory faults 4
2.2 System 4
2.2.1 Access to the last byte of the flash memory generates Hardfault 4
2.2.2 CPU execution freeze upon first erase or program operation after power-on or wake-up from Standby 5
2.2.3 EXTI line 58 is a configurable event 5
2.2.4 Shift in temperature monitoring thresholds 5
2.2.5 Accessing specific TPIU registers causes CPU lockup 6
2.2.6 ECC error when reading EEPROM after flash memory read 6
2.2.7 SRAMx_RST option bits have an immediate effect 6
2.2.8 Standby mode exit failure when NRST reset occurs during Standby entry sequence 6
2.2.9 Incorrect behavior of the JTAGIORETEN bit in the PWR_IORETR register 7
2.3 ADC 7
2.3.1 In certain dual modes, the fixed trigger latency for the injected conversions may not be respected 7
2.3.2 In simultaneous regular mode, stopping an injected conversion may shift the next regular conversion master and slave timing by one clock cycle 7
2.3.3 Injected conversions do not work when SMPTRIG bit is set, regular conversions start then stop, and SMPTRIG bit is cleared 8
2.3.4 Extra ADC consumption from V_{DDA} 8
2.4 DAC 8
2.4.1 Extra error for DAC1/DAC2 in unbuffered mode 8
2.4.2 DAC trigger latency is not constant under given conditions 8
2.5 RNG 9
2.5.1 RNG may report continuous seed errors in specific environmental conditions 9
2.6 HRTIM 9
2.6.1 Delayed idle protection malfunction when IDLESx = 1 9
2.6.2 Deadtime insertion mode not functional 9
2.6.3 Negative deadtime not supported in push-pull mode 10
2.6.4 Deadtime slightly shortened in push-pull mode 10
2.6.5 Generated pulse shorter than $t_{HRTIMER}$ period or missing for close set-reset or reset-set events 10
2.7 LPTIM 10
2.7.1 Device may remain stuck in LPTIM interrupt when entering Stop mode 10

2.7.2 ARRM and CMPM flags are not set when APB clock is slower than kernel clock 11
2.7.3 Interrupt status flag is cleared by hardware upon writing its corresponding bit in LPTIM_DIER register 11
2.7.4 LPTIM cannot use HSI clock in Stop mode 11
2.8 RTC 12
2.8.1 Alarm flag may be repeatedly set when the core is stopped in debug 12
2.8.2 RTC calendar read may return transient incorrect value when BYPSHAD = 0 12
2.8.3 System stuck if some RTC or TAMP registers are written when write protection is enabled 12
2.9 I2C 13
2.9.1 Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period 13
2.9.2 Spurious bus error detection in controller mode 13
2.10 USART 13
2.10.1 Received data may be corrupted upon clearing the ABREN bit 13
2.10.2 Noise error flag set while ONEBIT is set 14
2.11 LPUART 14
2.11.1 Possible LPUART transmitter issue when using low BRR[15:0] value 14
2.12 SPI 14
2.12.1 RDY output failure at high serial clock frequency 14
2.13 FDCAN 15
2.13.1 Desynchronization under specific condition with edge filtering enabled 15
2.13.2 Tx FIFO messages inverted under specific buffer usage and priority setting 15
Important security notice 16
Revision history 17

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. 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