

## STM32G0B1xB/xC/xE device errata

## Applicability

This document applies to the part numbers of STM32G0B1xB/xC/xE 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 RM0444.

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
STM32G0B1xB	STM32G0B1CB, STM32G0B1KB, STM32G0B1MB, STM32G0B1RB, STM32G0B1VB
STM32G0B1xC	STM32G0B1CC, STM32G0B1KC, STM32G0B1MC, STM32G0B1RC, STM32G0B1VC
STM32G0B1xE	STM32G0B1CE, STM32G0B1KE, STM32G0B1ME, STM32G0B1NE, STM32G0B1RE, STM32G0B1VE

**Table 2. Device variants**

Reference	Silicon revision codes	
	Device marking <sup>(1)</sup>	REV_ID <sup>(2)</sup>
STM32G0B1xB/xC/xE	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 register.

# 1 Summary of device errata

The following table gives a quick reference to the STM32G0B1xB/xC/xE device limitations and their status:

A = workaround available

N = no workaround available

P = partial workaround available

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
System	2.2.1	Unstable LSI when it clocks RTC or CSS on LSE	P
	2.2.2	WUFx wakeup flag wrongly set during configuration	A
	2.2.3	Overwriting with all zeros a Flash memory location previously programmed with all ones fails	N
	2.2.4	Wakeup from Stop not effective under certain conditions	N
	2.2.5	Flash memory PCROP area weakness	N
	2.2.6	PC13 signal transitions disturb LSE	N
	2.2.7	SRAM contents corrupted upon entry in Standby	N
GPIO	2.3.1	Wakeup capability-enabled GPIOs not configurable after wakeup from Standby	P
DMA	2.4.1	DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear	A
DMAMUX	2.5.1	SOFx not asserted when writing into DMAMUX_CFR register	N
	2.5.2	OFx not asserted for trigger event coinciding with last DMAMUX request	N
	2.5.3	OFx not asserted when writing into DMAMUX_RGCFR register	N
	2.5.4	Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event	A
ADC	2.6.1	Overrun flag is not set if EOC reset coincides with new conversion end	P
	2.6.2	Writing ADC_CFGR1 register while ADEN bit is set resets RES[1:0] bitfield	A
	2.6.3	Out-of-threshold value is not detected in AWD1 Single mode	A
	2.6.4	ADC sampling time might be one cycle longer	N
TIM	2.7.1	One-pulse mode trigger not detected in master-slave reset + trigger configuration	P
	2.7.2	Consecutive compare event missed in specific conditions	N
	2.7.3	Output compare clear not working with external counter reset	P
LPTIM	2.8.1	Device may remain stuck in LPTIM interrupt when entering Stop mode	A
	2.8.2	Device may remain stuck in LPTIM interrupt when clearing event flag	P
RTC and TAMP	2.9.1	Calendar initialization may fail in case of consecutive INIT mode entry	A
I2C	2.10.1	Wrong data sampling when data setup time (t <sub>SU;DAT</sub> ) is shorter than one I2C kernel clock period	P
	2.10.2	Spurious bus error detection in master mode	A
SPI	2.12.1	BSY bit may stay high when SPI is disabled	A

Function	Section	Limitation	Status
			Rev. A
SPI	2.12.2	BSY bit may stay high at the end of data transfer in slave mode	A
USART	2.11.1	Data corruption due to noisy receive line	N
FDCAN	2.13.1	Desynchronization under specific condition with edge filtering enabled	A
	2.13.2	Tx FIFO messages inverted under specific buffer usage and priority setting	A
UCPD	2.14.1	UCPD transmitter output marginality at low temperature	N

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

**Table 4. Summary of device documentation errata**

Function	Section	Documentation erratum
System	2.2.8	Boot select after debug interface connection
USART	2.11.2	USART prescaler feature missing in USART implementation section

## 2 Description of device errata

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

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



### 2.1 Core

Reference manual and errata notice for the Arm® Cortex®-M0+ core revision r0p1 is available from <http://infocenter.arm.com>.

### 2.2 System

#### 2.2.1 Unstable LSI when it clocks RTC or CSS on LSE

##### Description

The LSI clock can become unstable (duty cycle different from 50 %) and its maximum frequency can become significantly higher than 32 kHz, when:

- LSI clocks the RTC, or it clocks the clock security system (CSS) on LSE (which holds when the LSECSSON bit set), and
- the V<sub>DD</sub> power domain is reset while the backup domain is not reset, which happens:
  - upon exiting Shutdown mode
  - if V<sub>BAT</sub> is separate from V<sub>DD</sub> and V<sub>DD</sub> goes off then on
  - if V<sub>BAT</sub> is tied to V<sub>DD</sub> (internally in the package for products not featuring the VBAT pin, or externally) and a short (< 1 ms) V<sub>DD</sub> drop under V<sub>DD</sub>(min) occurs

##### Workaround

Apply one of the following measures:

- Clock the RTC with LSE or HSE/32, without using the CSS on LSE
- If LSI clocks the RTC or when the LSECSSON bit is set, reset the backup domain upon each V<sub>DD</sub> power up (when the BORRSTF flag is set). If V<sub>BAT</sub> is separate from V<sub>DD</sub>, also restore the RTC configuration, backup registers and anti-tampering configuration.

#### 2.2.2 WUFx wakeup flag wrongly set during configuration

##### Description

Upon configuring a wakeup pin (WKUPx), the corresponding wakeup flag (WUFx) might spuriously go high depending on the state and configuration of the wakeup pin.

##### Workaround

After configuring a wakeup pin, clear its corresponding WUFx flag.

#### 2.2.3 Overwriting with all zeros a Flash memory location previously programmed with all ones fails

##### Description

Any attempt to re-program with all zeros (0x0000 0000 0000 0000) a Flash memory location previously programmed with 0xFFFF FFFF FFFF FFFF fails and the PROGERR flag of the FLASH\_SR register is set.

**Workaround**

None.

**2.2.4 Wakeup from Stop not effective under certain conditions**

**Description**

With the HSI clock divider bitfield HSIDIV[2:0] set to a value different from 000, the device fails to enter Stop mode when SYSCLOCK is set to HSE clock.

With the HSI clock divider bitfield HSIDIV[2:0] set to a value different from 000, peripherals with clock request capability fail to wake the device up from Stop modes.

**Workaround**

None.

**2.2.5 Flash memory PCROP area weakness**

**Description**

When the CPU accesses PCROP-protected Flash memory areas:

- Fetch requests are allowed and are responded to normally.
- Read access are properly discarded. However, the bus holds and returns the value read during previous successful access.

**Workaround**

None.

*Note:* We recommend to use the PCROP protection in the following RDP and PCROP\_RDP configurations:

- RDP = Level 1 and PCROP\_RDP = 1
- RDP = Level 2

**2.2.6 PC13 signal transitions disturb LSE**

**Description**

The PC13 port toggling disturbs the LSE clock.

**Workaround**

None.

**2.2.7 SRAM contents corrupted upon entry in Standby**

**Description**

Entry of the device in Standby mode causes the SRAM content corruption.

**Workaround**

None.

**2.2.8 Boot select after debug interface connection**

**Description**

Some revisions of the reference manual may omit the following information.

After connecting the debug interface and until the device power-down, the boot source upon reset or wakeup from a low-power mode is determined by the PA14-BOOT0 pin level before connecting the debug interface (stored by the device), as opposed to the actual PA14-BOOT0 pin level. The device power-up restores the operation of the PA14-BOOT0 pin as direct boot source selector.

This is a documentation issue rather than a device limitation.

#### **Workaround**

No application workaround is required or applicable.

## **2.3 GPIO**

### **2.3.1 Wakeup capability-enabled GPIOs not configurable after wakeup from Standby**

#### **Description**

After the devices wakes up from Standby mode, GPIOs with the wakeup capability enabled cannot further be configured through the GPIO registers.

#### **Workaround**

Disable the wakeup capability of the GPIOs that must remain configurable through the GPIO registers after the wakeup from Standby.

## **2.4 DMA**

### **2.4.1 DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear**

#### **Description**

Upon a data transfer error in a DMA channel x, both the specific TEIFx and the global GIFx flags are raised and the channel x is normally automatically disabled. However, if in the same clock cycle the software clears the GIFx flag (by setting the CGIFx bit of the DMA\_IFCR register), the automatic channel disable fails and the TEIFx flag is not raised.

This issue does not occur with ST's HAL software that does not use and clear the GIFx flag when the channel is active.

#### **Workaround**

Do not clear GIFx flags when the channel is active. Instead, use HTIFx, TCIFx, and TEIFx specific event flags and their corresponding clear bits.

## **2.5 DMAMUX**

### **2.5.1 SOFx not asserted when writing into DMAMUX\_CFR register**

#### **Description**

The SOFx flag of the DMAMUX\_CSR status register is not asserted if overrun from another DMAMUX channel occurs when the software writes into the DMAMUX\_CFR register.

This can happen when multiple DMA channels operate in synchronization mode, and when overrun can occur from more than one channel. As the SOFx flag clear requires a write into the DMAMUX\_CFR register (to set the corresponding CSOFx bit), overrun occurring from another DMAMUX channel operating during that write operation fails to raise its corresponding SOFx flag.

### Workaround

None. Avoid the use of synchronization mode for concurrent DMAMUX channels, if at least two of them potentially generate synchronization overrun.

## 2.5.2 OFx not asserted for trigger event coinciding with last DMAMUX request

### Description

In the DMAMUX request generator, a trigger event detected in a critical instant of the last-generated DMAMUX request being served by the DMA controller does not assert the corresponding trigger overrun flag OFx. The critical instant is the clock cycle at the very end of the trigger overrun condition.

Additionally, upon the following trigger event, one single DMA request is issued by the DMAMUX request generator, regardless of the programmed number of DMA requests to generate.

The failure only occurs if the number of requests to generate is set to more than two ( $GNBREQ[4:0] > 00001$ ).

### Workaround

Make the trigger period longer than the duration required for serving the programmed number of DMA requests, so as to avoid the trigger overrun condition from occurring on the very last DMA data transfer.

## 2.5.3 OFx not asserted when writing into DMAMUX\_RGCFR register

### Description

The OFx flag of the DMAMUX\_RGSR status register is not asserted if an overrun from another DMAMUX request generator channel occurs when the software writes into the DMAMUX\_RGCFR register. This can happen when multiple DMA channels operate with the DMAMUX request generator, and when an overrun can occur from more than one request generator channel. As the OFx flag clear requires a write into the DMAMUX\_RGCFR register (to set the corresponding COFx bit), an overrun occurring in another DMAMUX channel operating with another request generator channel during that write operation fails to raise the corresponding OFx flag.

### Workaround

None. Avoid the use of request generator mode for concurrent DMAMUX channels, if at least two channels are potentially generating a request generator overrun.

## 2.5.4 Wrong input DMA request routed upon specific DMAMUX\_CxCR register write coinciding with synchronization event

### Description

If a write access into the DMAMUX\_CxCR register having the SE bit at zero and SPOL[1:0] bitfield at a value other than 00:

- sets the SE bit (enables synchronization),
- modifies the values of the DMAREQ\_ID[5:0] and SYNC\_ID[4:0] bitfields, and
- does not modify the SPOL[1:0] bitfield,

and if a synchronization event occurs on the previously selected synchronization input exactly two AHB clock cycles before this DMAMUX\_CxCR write, then the input DMA request selected by the DMAREQ\_ID[5:0] value before that write is routed.

### Workaround

Ensure that the SPOL[1:0] bitfield is at 00 whenever the SE bit is 0. When enabling synchronization by setting the SE bit, always set the SPOL[1:0] bitfield to a value other than 00 with the same write operation into the DMAMUX\_CxCR register.

## 2.6 ADC

### 2.6.1 Overrun flag is not set if EOC reset coincides with new conversion end

#### Description

If the EOC flag is cleared by an ADC\_DR register read operation or by software during the same APB cycle in which the data from a new conversion are written in the ADC\_DR register, the overrun event duly occurs (which results in the loss of either current or new data) but the overrun flag (OVR) may stay low.

#### Workaround

Clear the EOC flag, by performing an ADC\_DR read operation or by software within less than one ADC conversion cycle period from the last conversion cycle end, in order to avoid the coincidence with the end of the new conversion cycle.

### 2.6.2 Writing ADC\_CFGR1 register while ADEN bit is set resets RES[1:0] bitfield

#### Description

Modifying the ADC\_CFGR1 register while ADC is enabled (ADEN set in ADC\_CR) and no conversion is ongoing (ADSTART cleared in ADC\_CR) resets RES[1:0] to 00 whatever the bitfield previous value.

#### Workaround

Apply the following sequence:

1. Set ADDIS to disable the ADC, and wait until ADEN is cleared.
2. Program the ADC\_CFGR1 register according to the application requirements.
3. Set ADEN bit.

### 2.6.3 Out-of-threshold value is not detected in AWD1 Single mode

#### Description

AWD1 analog watchdog does not detect that the result of a converted channel has reached the programmed threshold when the ADC operates in Single mode, performs a sequence of conversions, and one of the converted channels other than the first one is monitored by the AWD1 analog watchdog.

#### Workaround

Apply one of the following measures:

- Use a conversion sequence of one single channel.
- Configure the monitored channel as the first one of the sequence.

### 2.6.4 ADC sampling time might be one cycle longer

#### Description

For sampling time set to 1.5 or 3.5 cycles, the sampling in a single ADC conversion or in the first conversion of a sequence takes one extra cycle.

#### Workaround

None.



## 2.7 TIM

### 2.7.1 One-pulse mode trigger not detected in master-slave reset + trigger configuration

#### Description

The failure occurs when several timers configured in one-pulse mode are cascaded, and the master timer is configured in combined reset + trigger mode with the MSM bit set:

OPM = 1 in TIMx\_CR1, SMS[3:0] = 1000 and MSM = 1 in TIMx\_SMCR.

The MSM delays the reaction of the master timer to the trigger event, so as to have the slave timers cycle-accurately synchronized.

If the trigger arrives when the counter value is equal to the period value set in the TIMx\_ARR register, the one-pulse mode of the master timer does not work and no pulse is generated on the output.

#### Workaround

None. However, unless a cycle-level synchronization is mandatory, it is advised to keep the MSM bit reset, in which case the problem is not present. The MSM = 0 configuration also allows decreasing the timer latency to external trigger events.

### 2.7.2 Consecutive compare event missed in specific conditions

#### Description

Every match of the counter (CNT) value with the compare register (CCR) value is expected to trigger a compare event. However, if such matches occur in two consecutive counter clock cycles (as consequence of the CCR value change between the two cycles), the second compare event is missed for the following CCR value changes:

- in edge-aligned mode, from ARR to 0:
  - first compare event: CNT = CCR = ARR
  - second (missed) compare event: CNT = CCR = 0
- in center-aligned mode while up-counting, from ARR-1 to ARR (possibly a new ARR value if the period is also changed) at the crest (that is, when TIMx\_RCR = 0):
  - first compare event: CNT = CCR = (ARR-1)
  - second (missed) compare event: CNT = CCR = ARR
- in center-aligned mode while down-counting, from 1 to 0 at the valley (that is, when TIMx\_RCR = 0):
  - first compare event: CNT = CCR = 1
  - second (missed) compare event: CNT = CCR = 0

This typically corresponds to an abrupt change of compare value aiming at creating a timer clock single-cycle-wide pulse in toggle mode.

As a consequence:

- In toggle mode, the output only toggles once per counter period (squared waveform), whereas it is expected to toggle twice within two consecutive counter cycles (and so exhibit a short pulse per counter period).
- In center mode, the compare interrupt flag does not rise and the interrupt is not generated.

*Note:* The timer output operates as expected in modes other than the toggle mode.

#### Workaround

None.

### 2.7.3 Output compare clear not working with external counter reset

#### Description

The output compare clear event (ocref\_clr) is not correctly generated when the timer is configured in the following slave modes: Reset mode, Combined reset + trigger mode, and Combined gated + reset mode.

The PWM output remains inactive during one extra PWM cycle if the following sequence occurs:

1. The output is cleared by the ocref\_clr event.
2. The timer reset occurs before the programmed compare event.

#### Workaround

Apply one of the following measures:

- Use BKIN (or BKIN2 if available) input for clearing the output, selecting the Automatic output enable mode (AOE = 1).
- Mask the timer reset during the PWM ON time to prevent it from occurring before the compare event (for example with a spare timer compare channel open-drain output connected with the reset signal, pulling the timer reset line down).

## 2.8 LPTIM

### 2.8.1 Device may remain stuck in LPTIM interrupt when entering Stop mode

#### Description

This limitation occurs when disabling the low-power timer (LPTIM).

When the user application clears the ENABLE bit in the LPTIM\_CR register within a small time window around one LPTIM interrupt occurrence, then the LPTIM interrupt signal used to wake up the device from Stop mode may be frozen in active state. Consequently, when trying to enter Stop mode, this limitation prevents the device from entering low-power mode and the firmware remains stuck in the LPTIM interrupt routine.

This limitation applies to all Stop modes and to all instances of the LPTIM. Note that the occurrence of this issue is very low.

#### Workaround

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

### 2.8.2 Device may remain stuck in LPTIM interrupt when clearing event flag

#### Description

This limitation occurs when the LPTIM is configured in interrupt mode (at least one interrupt is enabled) and the software clears any flag in LPTIM\_ISR register by writing its corresponding bit in LPTIM\_ICR register. If the interrupt status flag corresponding to a disabled interrupt is cleared simultaneously with a new event detection, the set and clear commands might reach the APB domain at the same time, leading to an asynchronous interrupt signal permanently stuck high.

This issue can occur either during an interrupt subroutine execution (where the flag clearing is usually done), or outside an interrupt subroutine.

Consequently, the firmware remains stuck in the LPTIM interrupt routine, and the device cannot enter Stop mode.

#### Workaround

To avoid this issue, it is strongly advised to follow the recommendations listed below:

- Clear the flag only when its corresponding interrupt is enabled in the interrupt enable register.
- If for specific reasons, it is required to clear some flags that have corresponding interrupt lines disabled in the interrupt enable register, it is recommended to clear them during the current subroutine prior to those which have corresponding interrupt line enabled in the interrupt enable register.
- Flags must not be cleared outside the interrupt subroutine.

*Note:* The proper clear sequence is already implemented in the HAL\_LPTIM\_IRQHandler in the STM32Cube.

## 2.9 RTC and TAMP

### 2.9.1 Calendar initialization may fail in case of consecutive INIT mode entry

#### Description

If the INIT bit of the RTC\_ICSR register is set between one and two RTCCLK cycles after being cleared, the INITF flag is set immediately instead of waiting for synchronization delay (which should be between one and two RTCCLK cycles), and the initialization of registers may fail. Depending on the INIT bit clearing and setting instants versus the RTCCLK edges, it can happen that, after being immediately set, the INITF flag is cleared during one RTCCLK period then set again. As writes to calendar registers are ignored when INITF is low, a write occurring during this critical period might result in the corruption of one or more calendar registers.

#### Workaround

After exiting the initialization mode, clear the BYPSHAD bit (if set) then wait for RSF to rise, before entering the initialization mode again.

*Note:* It is recommended to write all registers in a single initialization session to avoid accumulating synchronization delays.

## 2.10 I2C

### 2.10.1 Wrong data sampling when data setup time ( $t_{\text{SU;DAT}}$ ) 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 ( $t_{\text{SU;DAT}}$ ) 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  $t_{\text{SU;DAT}}$  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 slave address, data byte, or acknowledge bit.

#### Workaround

Increase the I2C kernel clock frequency to get I2C kernel clock period within the transmitter minimum data setup time. Alternatively, increase transmitter's minimum data setup time. If the transmitter setup time minimum value corresponds to the minimum value provided in the I<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.

### 2.10.2 Spurious bus error detection in master mode

#### Description

In master mode, a bus error can be detected spuriously, with the consequence of setting the BERR flag of the I2C\_SR register and generating bus error interrupt if such interrupt is enabled. Detection of bus error has no effect on the I<sup>2</sup>C-bus transfer in master mode and any such transfer continues normally.

### Workaround

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

## 2.11 USART

### 2.11.1 Data corruption due to noisy receive line

#### Description

In UART mode with oversampling by 8 or 16 and with 1 or 2 stop bits, the received data may be corrupted if a glitch to zero shorter than the half-bit occurs on the receive line within the second half of the stop bit.

#### Workaround

None.

### 2.11.2 USART prescaler feature missing in USART implementation section

#### Description

Some reference manual revisions may omit the information that the USART prescaler is not present in all USART instances. This information is provided in the USART implementation section of the corresponding reference manual.

This is a documentation issue rather than a product limitation.

#### Workaround

No application workaround is required or applicable.

## 2.12 SPI

### 2.12.1 BSY bit may stay high when SPI is disabled

#### Description

The BSY flag may remain high upon disabling the SPI while operating in:

- master transmit mode and the TXE flag is low (data register full).
- master receive-only mode (simplex receive or half-duplex bidirectional receive phase) and an SCK strobing edge has not occurred since the transition of the RXNE flag from low to high.
- slave mode and NSS signal is removed during the communication.

#### Workaround

When the SPI operates in:

- master transmit mode, disable the SPI when TXE = 1 and BSY = 0.
- master receive-only mode, ignore the BSY flag.
- slave mode, do not remove the NSS signal during the communication.

### 2.12.2 BSY bit may stay high at the end of data transfer in slave mode

#### Description

BSY flag may sporadically remain high at the end of a data transfer in slave mode. This occurs upon coincidence of internal CPU clock and external SCK clock provided by master.

In such an event, if the software only relies on BSY flag to detect the end of SPI slave data transaction (for example to enter low-power mode or to change data line direction in half-duplex bidirectional mode), the detection fails.

As a conclusion, the BSY flag is unreliable for detecting the end of data transactions.

### Workaround

Depending on SPI operating mode, use the following means for detecting the end of transaction:

- When NSS hardware management is applied and NSS signal is provided by master, use NSS flag.
- In SPI receiving mode, use the corresponding RXNE event flag.
- In SPI transmit-only mode, use the BSY flag in conjunction with a timeout expiry event. Set the timeout such as to exceed the expected duration of the last data frame and start it upon TXE event that occurs with the second bit of the last data frame. The end of the transaction corresponds to either the BSY flag becoming low or the timeout expiry, whichever happens first.

Prefer one of the first two measures to the third as they are simpler and less constraining.

Alternatively, apply the following sequence to ensure reliable operation of the BSY flag in SPI transmit mode:

1. Write last data to data register.
2. Poll the TXE flag until it becomes high, which occurs with the second bit of the data frame transfer.
3. Disable SPI by clearing the SPE bit mandatorily before the end of the frame transfer.
4. Poll the BSY bit until it becomes low, which signals the end of transfer.

*Note:* The alternative method can only be used with relatively fast CPU speeds versus relatively slow SPI clocks or/and long last data frames. The faster is the software execution, the shorter can be the duration of the last data frame.

## 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
```

## 2.14 UCPD

### 2.14.1 UCPD transmitter output marginality at low temperature

#### Description

At low temperature, the UCPD transmitter high level may go as low as 1.04 V, not respecting the specified minimum of 1.05 V. The external load (Rp/Rd) can mitigate this slight marginality observed on a low percentage of parts.

#### Workaround

None.

## Revision history

**Table 5. Document revision history**

Date	Version	Changes
12-Jan-2021	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>4</b>
2.1	Core	4
2.2	System	4
2.2.1	Unstable LSI when it clocks RTC or CSS on LSE	4
2.2.2	WUFx wakeup flag wrongly set during configuration	4
2.2.3	Overwriting with all zeros a Flash memory location previously programmed with all ones fails	4
2.2.4	Wakeup from Stop not effective under certain conditions	5
2.2.5	Flash memory PCROP area weakness	5
2.2.6	PC13 signal transitions disturb LSE	5
2.2.7	SRAM contents corrupted upon entry in Standby	5
2.2.8	Boot select after debug interface connection	5
2.3	GPIO	6
2.3.1	Wakeup capability-enabled GPIOs not configurable after wakeup from Standby	6
2.4	DMA	6
2.4.1	DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear	6
2.5	DMAMUX	6
2.5.1	SOFx not asserted when writing into DMAMUX_CFR register	6
2.5.2	OFx not asserted for trigger event coinciding with last DMAMUX request	7
2.5.3	OFx not asserted when writing into DMAMUX_RGCFR register	7
2.5.4	Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event	7
2.6	ADC	8
2.6.1	Overrun flag is not set if EOC reset coincides with new conversion end	8
2.6.2	Writing ADC_CFGR1 register while ADEN bit is set resets RES[1:0] bitfield	8
2.6.3	Out-of-threshold value is not detected in AWD1 Single mode	8
2.6.4	ADC sampling time might be one cycle longer	8
2.7	TIM	9
2.7.1	One-pulse mode trigger not detected in master-slave reset + trigger configuration	9
2.7.2	Consecutive compare event missed in specific conditions	9



2.7.3	Output compare clear not working with external counter reset . . . . .	9
<b>2.8</b>	<b>LPTIM</b> . . . . .	<b>10</b>
2.8.1	Device may remain stuck in LPTIM interrupt when entering Stop mode . . . . .	10
2.8.2	Device may remain stuck in LPTIM interrupt when clearing event flag . . . . .	10
<b>2.9</b>	<b>RTC and TAMP</b> . . . . .	<b>11</b>
2.9.1	Calendar initialization may fail in case of consecutive INIT mode entry . . . . .	11
<b>2.10</b>	<b>I2C</b> . . . . .	<b>11</b>
2.10.1	Wrong data sampling when data setup time ( $t_{\text{SU;DAT}}$ ) is shorter than one I2C kernel clock period . . . . .	11
2.10.2	Spurious bus error detection in master mode . . . . .	11
<b>2.11</b>	<b>USART</b> . . . . .	<b>12</b>
2.11.1	Data corruption due to noisy receive line. . . . .	12
2.11.2	USART prescaler feature missing in USART implementation section . . . . .	12
<b>2.12</b>	<b>SPI</b> . . . . .	<b>12</b>
2.12.1	BSY bit may stay high when SPI is disabled . . . . .	12
2.12.2	BSY bit may stay high at the end of data transfer in slave mode . . . . .	12
<b>2.13</b>	<b>FDCAN</b> . . . . .	<b>13</b>
2.13.1	Desynchronization under specific condition with edge filtering enabled. . . . .	13
2.13.2	Tx FIFO messages inverted under specific buffer usage and priority setting . . . . .	13
<b>2.14</b>	<b>UCPD</b> . . . . .	<b>14</b>
2.14.1	UCPD transmitter output marginality at low temperature. . . . .	14
<b>Revision history</b> . . . . .		<b>15</b>

**IMPORTANT NOTICE – PLEASE 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. 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 acknowledgement.

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 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.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, please 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.

© 2021 STMicroelectronics – All rights reserved