**STM32U575xx STM32U585xx**

Errata sheet

# STM32U575xx and STM32U585xx device errata

## Applicability

This document applies to the part numbers of STM32U575xx and STM32U585xx 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 RM0456.

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 |
|---|---|
| STM32U575xx | STM32U575AG, STM32U575AI, STM32U575CG, STM32U575CI, STM32U575OG, STM32U575OI, STM32U575QG, STM32U575QI, STM32U575RG, STM32U575RI, STM32U575VG, STM32U575VI, STM32U575ZG, STM32U575ZI |
| STM32U585xx | STM32U585AI, STM32U585CI, STM32U585OI, STM32U585QI, STM32U585RI, STM32U585VI, STM32U585ZI |

Table 2. **Device variants**

| Reference | Silicon revision codes | |
|---|---|---|
| | Device marking(1) | REV_ID(2) |
| STM32U575xx and STM32U585xx | X | 0x2001 |
| | W | 0x3001 |

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 STM32U575xx and STM32U585xx device limitations and their status:

A = limitation present, workaround available

N = limitation present, no workaround available

P = limitation present, partial workaround available

“-” = limitation absent

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

Table 3. **Summary of device limitations**

| Function | Section | Limitation | Status | |
|---|---|---|---|---|
| | | | Rev. X | Rev. W |
| Core | 2.1.1 | Access permission faults are prioritized over unaligned Device memory faults | N | N |
| System | 2.2.1 | LSE disturbed by RTC/TAMP OUT function on PC13 | N | N |
| | 2.2.2 | Too low MSI frequency upon exit from Standby or Stop 3 mode | A | A |
| | 2.2.3 | LSE low drive mode is not functional | N | -(1) |
| | 2.2.4 | Peripheral triggers connected to RTC wakeup timer interrupt instead of RTC wakeup timer trigger signal | A | A |
| | 2.2.5 | Device hang-up can occur when wakeup request received a few clock cycles before entering Stop 2 or Stop 3 mode | A | A |
| | 2.2.6 | SRAM2 page 1 content may be lost at high temperature when Standby mode is entered from Run mode with LDO enabled and Range 1, Range 2 or Range 3 selected | A | - |
| | 2.2.7 | Device authentication ID is not accessible in RDP Level 0 | A | A |
| | 2.2.8 | MSIK and HSI16 clocks cannot be stopped when used as kernel clock by ADCs, DAC1, MDF1, or ADF1 | A | A |
| | 2.2.9 | Incorrect backup domain reset with $V_{BAT}$ and $V_{DD}$ supplied by the same power source | A | - |
| | 2.2.10 | Incorrect backup domain reset with $V_{BAT}$ and $V_{DD}$ supplied by different power sources | A | A |
| | 2.2.11 | ICACHE and DCACHE access might be corrupted after exiting Stop 2 and Stop 3 modes | A | - |
| | 2.2.12 | LSI remains enabled even if not used in $V_{BAT}$ mode | N | - |
| | 2.2.13 | PWR_BDCR1 is not write-protected by DBP | N | N |
| | 2.2.14 | The PWR_S3WU interrupt is generated for internal wakeup sources (WUSELx = 11) | A | A |
| | 2.2.15 | OTG_FS is reset by OTGRST and DCMI_PSSIRST bits | A | - |
| | 2.2.16 | LSE medium-low drive mode is not available | N | -(1) |
| | 2.2.17 | STM32U575QIIxxx/STM32U585QIIxxx bootloader communication interfaces are not available if TZEN is set | N(2) | - |
| | 2.2.18 | Bootloader cannot increase RDP and modify the secure option bytes at the same time | A(3) | -(4) |
| | 2.2.19 | HardFault on wakeup from Stop mode may occur in debug mode | N | N |

| Function | Section | Limitation | Status | |
|---|---|---|---|---|
| | | | Rev. X | Rev. W |
| System | 2.2.20 | Full JTAG configuration without NJTRST pin cannot be used | A | A |
| | 2.2.21 | EXTI LOCK bit does not lock privilege configuration | N | N |
| | 2.2.22 | Device may be locked upon system reset under Stop 2 mode | - | P |
| GTZC | 2.3.1 | LPTIM3_IN1 and LPTIM3_IN2 is protected by LPTIM1SEC instead of LPTIM3SEC | A | - |
| | 2.3.2 | TIMICSEL[2:0] is not protected by LPTIM2SEC | A | A |
| GPIO | 2.4.1 | Unexpected current consumption with GPIOs configured as I2C alternate functions | A | - |
| FMC | 2.5.1 | Dummy read cycles inserted when reading synchronous memories | N | N |
| | 2.5.2 | Wrong data read from a busy NAND memory | A | A |
| | 2.5.3 | Data corruption upon a specific FIFO write sequence to synchronous PSRAM | A | - |
| | 2.5.4 | Unsupported AHB burst byte write to PSRAM | - | A |
| OCTOSPI | 2.6.1 | Memory-mapped write error response when DQS output is disabled | P | P |
| | 2.6.2 | Byte possibly dropped during an SDR read in clock mode 3 when a transfer gets automatically split | A | A |
| | 2.6.3 | Received data corrupted after arbitration ownership toggles when using clock mode 3 and no DQS for read direction | A | A |
| | 2.6.4 | Deadlock can occur under certain conditions | A | A |
| | 2.6.5 | Read data corruption after partial read when crossing CSBOUND boundary | P | - |
| | 2.6.6 | Deadlock or write-data corruption after spurious write to a misaligned address in OCTOSPI_ AR register | N | N |
| | 2.6.7 | Read data corruption after a few bytes are skipped when crossing a four-byte boundary | A | A |
| | 2.6.8 | At least six cycles memory latency must be set when DQS is used for HyperBus™ memories | A | A |
| | 2.6.10 | 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 | A |
| ADC1 | 2.8.1 | ADC_AWDy_OUT reset by non-guarded channels | A | A |
| | 2.8.2 | Injected data stored in the wrong ADC_JDRx registers | A | A |
| ADC4 | 2.9.1 | ADC4 conversion error when used simultaneously with ADC1 | P | P |
| | 2.9.2 | ADC clock request is active after reset | A | A |
| VREFBUF | 2.10.1 | $V_{REFBUF\_OUT}$ voltage overshoots in Range 4, Stop 1 or Stop 2 mode | A | A |
| MDF | 2.11.1 | In LFM mode MDF_CCK1 clock cannot be selected for SITFx interfaces | A | A |
| TIM | 2.13.1 | Bidirectional break mode not working with short pulses | N | N |
| LPTIM | 2.14.1 | Device may remain stuck in LPTIM interrupt when entering Stop mode | A | A |
| | 2.14.2 | ARRM and CMPM flags are not set when APB clock is slower than kernel clock | P | P |
| | 2.14.3 | Device may remain stuck in LPTIM interrupt when clearing event flag | P | P |
| | 2.14.4 | Interrupt status flag is cleared by hardware upon writing its corresponding bit in LPTIM_DIER register | N | N |
| | 2.14.5 | Overcapture stops working when CCxOF flag is cleared in some specific conditions | P | - |
| IWDG | 2.15.1 | IWDG interrupt does not wake up from Stop mode | N | N |

| Function | Section | Limitation | Status | |
|---|---|---|---|---|
| | | | Rev. X | Rev. W |
| IWDG | 2.15.2 | IWDG is stopped when BDRST is set | P | P |
| RTC and TAMP | 2.16.1 | Alarm flag may be repeatedly set when the core is stopped in debug | N | N |
| | 2.16.2 | Binary mode: SSR is not reloaded with 0xFFFF FFFF when SSCLR = 1 | A | A |
| | 2.16.3 | Parasitic tamper detection when debugger is used in RDP Level 0 | A | A |
| I2C | 2.17.1 | Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period | P | P |
| | 2.17.2 | Spurious bus error detection in master mode | A | A |
| | 2.17.3 | SDA held low upon SMBus timeout expiry in slave mode | A | A |
| USART | 2.18.1 | Data corruption due to noisy receive line | - | N |
| | 2.18.2 | USART does not generate DMA requests after setting/clearing DMAT bit | A | A |
| | 2.18.3 | Wakeup on a character match interrupt fails, in Autonomous mode with FIFO disabled | A | A |
| | 2.18.4 | Wrong data received in Smartcard mode and 0.5 stop bit configuration | A | - |
| | 2.18.5 | Wrong data received by SPI slave receiver in Autonomous mode with CPOL = 1 | A | A |
| | 2.18.6 | Received data may be corrupted upon clearing the ABREN bit | A | A |
| | 2.18.7 | Noise error flag set while ONEBIT is set | N | N |
| LPUART | 2.19.1 | LPUART does not generate DMA requests after setting/clearing DMAT bit | A | A |
| | 2.19.2 | Wakeup on a character match interrupt fails, in Autonomous mode with FIFO disabled | A | A |
| | 2.19.3 | Possible LPUART transmitter issue when using low BRR[15:0] value | P | P |
| SPI | 2.20.1 | Possible corruption of last-received data depending on CRCSIZE setting | A | A |
| | 2.20.2 | MODF flag cannot generate interrupt | A | A |
| | 2.20.3 | RDY output failure at high serial clock frequency | N | N |
| | 2.20.4 | Master communication suspension fails in Autonomous mode | N | N |
| | 2.20.5 | SPE may not be cleared upon MODF event | A | A |
| | 2.20.6 | SPI slave stalls with masters not providing extra SCK periods upon *Not ready* signalling | A | A |
| | 2.20.7 | Truncation of SPI output signals after EOT event | A | A |
| FDCAN | 2.21.1 | Desynchronization under specific condition with edge filtering enabled | A | A |
| | 2.21.2 | Tx FIFO messages inverted under specific buffer usage and priority setting | A | A |
| OTG_FS | 2.22.1 | Host packet transmission may hang when connecting through a hub to a low-speed device | N | N |
| UCPD | 2.23.1 | TXHRST upon write data underflow corrupting the CRC of the next packet | A | A |
| | 2.23.2 | Ordered set with multiple errors in a single K-code is reported as invalid | N | N |
| | 2.23.3 | Rp out of specification | N | - |
| | 2.23.4 | Rd out of specification | N | - |

1. *LSE, PC14, and PC15 I/Os electrical characteristics are modified on the rev. W, due to this limitation removal. The new characteristics are to be included in revision 7 of STM32U585 and STM32U575 datasheet releases (DS13086 and DS13737).*
2. *There is a limitation on the embedded RSS:*
   - *Part numbers with RSS version 1.3 contain this limitation.*
   - *Part numbers with RSS version 1.4 do not contain this limitation.*

*3. This limitation is present in bootloader version 9.2.*

*4. This limitation is fixed in bootloader version 9.3.*

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

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

| Function | Section | Documentation erratum |
|---|---|---|
| System | 2.2.23 | SRAM ECC error flags and addresses are updated only if interrupt is enabled |
| OCTOSPI | 2.6.9 | Automatic status-polling mode cannot be used with HyperFlash™ memories |
| VREFINT | 2.7.1 | $t_{S\_vrefint}$ minimum value is wrong in datasheet |
| ADC1 | 2.8.3 | 14-bit ADC offset error and integral linearity error values are increased in datasheets |
| TSC | 2.12.1 | TSC_G3_IO1/TSC_G1_IO4 are removed from the datasheet |

# 2 Description of device errata

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

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

## 2.1 Core

Reference manual and errata notice for the Arm® Cortex®-M33 core revision r0p4 is available from http://infocenter.arm.com.

### 2.1.1 Access permission faults are prioritized over unaligned Device memory faults

#### Description

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

The failure occurs when the MPU is enabled and:

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

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

#### Workaround

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

## 2.2 System

### 2.2.1 LSE disturbed by RTC/TAMP OUT function on PC13

#### Description

LSE frequency can be incorrect when PC13 is used as output for RTC_OUT1 or TAMP_OUT2 function.

#### Workaround

None.
The RTC_OUT2 function must be used on PB2 and TAMP_OUT functions must be used on other I/Os.

### 2.2.2 Too low MSI frequency upon exit from Standby or Stop 3 mode

#### Description

The MSI clock frequency can be up to 25% lower than expected upon exit from Standby or Stop 3 mode, for a maximum of 200 µs.

#### Workaround

In case accuracy is needed, wait for 200 µs after exiting Standby or Stop 3 mode before using the MSI.

### 2.2.3 LSE low drive mode is not functional

#### Description

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

#### Workaround

None.

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

#### Description

GPDMA1, LPDMA1, SPIx (x = 1 to 3), I2Cx (x = 1 to 4), U(S)ARTx (x = 1 to 5) and LPUART1 triggers are connected to RTC wakeup timer interrupt instead of RTC wakeup timer trigger signal (rtc_wut_trg).

#### Workaround

Enable the RTC wakeup timer interrupt. The wakeup timer flag must be cleared in the interrupt subroutine before getting a new trigger.
To avoid serving an interrupt, use triggers from RTC alarm or from LPTIM instead of RTC wakeup timer.

### 2.2.5 Device hang-up can occur when wakeup request received a few clock cycles before entering Stop 2 or Stop 3 mode

#### Description

If PLL2, PLL3, HSI48, or SHSI is activated when entering Stop 2 or Stop 3 mode, there are a few clock cycles during the low-power entry sequence where a wakeup request can lead to a device hang-up.

#### Workaround

Switch off PLL2, PLL3, HSI48, and SHSI, then wait for the corresponding ready flags (respectively PLL2RDY, PLL3RDY, HSI48RDY, and SHSIRDY) to be cleared before entering Stop 2 or Stop 3 mode.

### 2.2.6 SRAM2 page 1 content may be lost at high temperature when Standby mode is entered from Run mode with LDO enabled and Range 1, Range 2 or Range 3 selected

#### Description

The content of SRAM2 page 1 (8 Kbytes) may be lost if the following conditions are met:
- The temperature is above 110 °C.
- The Standby mode with only SRAM2 page 1 retention (8-Kbytes) is entered from Run mode.
- The LDO regulator is selected.
- The voltage scaling is configured to Range 1, Range 2 or Range 3.

This limitation is not present when entering Standby mode from Range 4, or when the full SRAM2 is retained.

#### Workaround

When the temperature is above 110 °C, switch to Range 4 and wait for 10 ms (to let $V_{CORE}$ reach Range 4 voltage) before entering Standby mode with 8-Kbyte SRAM2 retention.
For SMPS packages, enter Standby mode with 8-Kbyte SRAM retention when running on SMPS.

### 2.2.7 Device authentication ID is not accessible in RDP Level 0

#### Description

The AUTH_ID bitfield of the DBGMCU_DBG_AUTH_DEVICE register is not accessible in RDP Level 0. The read value is always 0 so this bitfield cannot be used to discriminate between different devices.

#### Workaround

Increase the RDP to Level 1 before reading the device authentication ID. Then, decrease the RDP back to Level 0.

### 2.2.8 MSIK and HSI16 clocks cannot be stopped when used as kernel clock by ADCs, DAC1, MDF1, or ADF1

#### Description

MSIK can be used as kernel clock by the ADCs, DAC1, MDF1, and ADF1, by configuring the ADCDACSEL, MDF1SEL, and ADF1SEL bitfields of the RCC_CCIPRx registers.

HSI16 can also be used as kernel clock by the ADCs and DAC1.

*Note: The ADCDACSEL bitfield is common to all ADCs and DAC1 so these peripherals share the same kernel clock.*

Once selected as kernel clock source for the ADCs, DAC1, MDF1, or ADF1 peripherals, MSIK (resp. HSI16) can no longer be disabled using the MSIKON (resp. HSION) bits of the RCC_CR register because MDF1, ADF1, and ADC4 request their kernel clock by default after reset.

If the application requests entry into Stop 0, Stop 1, or Stop 2 mode, the selected kernel clock (MSIK or HSI16) remains enabled, which leads to an overconsumption in this mode. Stop 3 is not concerned by this limitation. MDF1 is not concerned for Stop 2 mode.

#### Workaround

Before disabling the MSIK or HSI16 clocks, configure ADCDACSEL, MDF1SEL, and ADF1SEL to HCLK.

In case the application wants to enter Stop 0 or Stop 1 and the MDF1, DAC1, ADC4, and ADF1 are not used during Stop mode, configure ADCDACSEL, MDF1SEL, and ADF1SEL bitfields to HCLK before entering Stop mode. Upon Stop mode exit, restore the required kernel clock (MSIK or HSI16) with the ADCDACSEL, MDF1SEL, and ADF1SEL bitfields. In case the application wants to enter Stop 2 mode and the DAC1, ADC4, and ADF1 are not used during Stop mode, the same procedure is only required for ADCDACSEL and ADF1SEL.

In case the application wants to enter Stop 0, Stop 1, or Stop 2 mode and the DAC1 is used during Stop mode, then the application needs to initialize the ADC4 peripheral, even if it is not used by application. The ADC4 stops requesting its kernel clock and MSIK or HSI16 will be disabled during Stop mode.

In case the application wants to enter Stop 0, Stop 1, or Stop 2 mode and the MDF1, ADC4, or ADF1 peripheral is used during Stop mode (Stop 0 and Stop 1 only for MDF1), then simply configure the ADCDACSEL, MDF1SEL, and ADF1SEL bitfields to required kernel clock (MSIK or HSI16). These peripherals request their kernel clock when they need it.

### 2.2.9 Incorrect backup domain reset with $V_{BAT}$ and $V_{DD}$ supplied by the same power source

#### Description

The backup domain reset may be missed upon a power-on subsequent to a power-off, if the supply voltage during the power-off phase drops to a few mV window level 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.

The issue only occurs in applications using the same power source for $V_{DD}$ and $V_{BAT}$.

This missed reset results in unpredictable values of the backup domain registers, which can 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 measures:

- In the application, let the $V_{DD}/V_{BAT}$ supply voltage fall to a level below 100 mV for more than 200 ms before a new power-on.
- If the previous workaround is not applicable and the boot follows a power-on reset, erase the backup domain by software. In order to discriminate the power-on reset from a Shutdown mode exit when Shutdown mode is used, 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 the RCC_CSR register is set (the reset is caused by the power-on or by Shutdown mode exit).
2. If true and if Shutdown mode is used in the application, 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 power-on. Apply the following sequence:
   a. Enable PWR clock in RCC, by setting the PWREN bit of RCC_AHB3ENR.
   b. Enable backup domain access, by setting the DBP bit of PWR_DBPR.
   c. Reset backup domain, by:
      i. writing 0x0001 0000 in 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 in RCC_BDCR, to clear the BDRST bit
   d. Clear BORRSTF, by setting the RMVF bit of RCC_CSR.
4. Enable the continuous monitoring of the backup domain, by setting the MONEN bit of PWR_BDCR1, after enabling the PWR clock in RCC.

If it is not possible to insert the sequence at the very beginning of the boot code, perform it after enabling the RAMCFG clock in RCC, and after the SRAMBUSY bit of RAMCFG_M2ISR falls low (SRAM2 memory erase completed).

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

#### 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. The issue only occurs when MONEN = 0 in the PWR_BDCR1 register (backup domain supply and temperature monitoring are disabled).

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

- If the extra consumption is acceptable, enable backup domain supply and temperature monitoring (MONEN = 1 in PWR_BDCR1). The maximum extra consumption is 1.2 µA from $V_{BAT}$ when $V_{DD}$ is above $V_{BOR0}$, and around 3.6 µA from $V_{BAT}$ in $V_{BAT}$ mode.

- 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 the two previous workarounds are 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 PWR clock in RCC, by setting the PWREN bit of RCC_AHB3ENR.
    b. Enable backup domain access, by setting the DBP bit of PWR_DBPR.
    c. Reset backup domain, by:
        i. writing 0x0001 0000 in 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 in RCC_BDCR, to clear the BDRST bit
    d. Clear BORRSTF, by setting the RMVF bit of RCC_CSR.

### 2.2.11 ICACHE and DCACHE access might be corrupted after exiting Stop 2 and Stop 3 modes

#### Description

After exiting Stop 2 or Stop 3 mode, the first instruction fetch or data read from a 128-bit ICACHE or DCACHE line is corrupted if that same ICACHE or DCACHE line was the last to access before entering Stop 2 or Stop 3 mode.

The cache content itself is not corrupted. Fetching or reading data from any cache line other than the last accessed prior to Stop 2 or Stop 3 mode entry operates correctly.

#### Workaround

For ICACHE, apply one of the following measures:

- Ensure that the first-accessed cache line after exiting Stop2 or Stop 3 mode is different from the one accessed as last before entering Stop 2 or Stop 3 mode:
    – *Instruction fetch:* Configure the memory protection unit (MPU) such as to keep the code corresponding to the low-power mode entry non-cacheable.
    – *Data read:* After exiting Stop 2 or Stop 3 mode, read an ICACHE line other than the one accessed as last before entering Stop 2 or Stop 3 mode.
- Disable ICACHE before entering then enable it back upon exiting Stop 2 or Stop 3 mode. This reduces the performance as the ICACHE invalidation is done in the background and the application continues executing with a cold ICACHE.

For DCACHE1, apply one of the following measures:

- After exiting Stop 2 or Stop 3 mode, read any two locations in different DCACHE1 lines and ignore the results. Any following read access to the cache now yields correct data.
- Disable DCACHE1 before entering then enable it back upon exiting Stop 2 or Stop 3 mode. This delays Stop 2 or Stop 3 mode entry by 128 cycles and reduces the performance as the application continues executing with a cold DCACHE1.

## 2.2.12 LSI remains enabled even if not used in $V_{BAT}$ mode

**Description**

If the LSI is enabled with LSION = 1 in RCC_BDCR register, it remains enabled in $V_{BAT}$ mode, even if it is not used by the RTC or TAMP peripheral. This generates an extra consumption in $V_{BAT}$ mode.

*Note: If the IWDG is used with LSION = 0 (in this case the LSI is forced enabled by IWDG), the LSI is automatically switched off in $V_{BAT}$ mode, so there is no extra consumption.*

**Workaround**

None.

## 2.2.13 PWR_BDCR1 is not write-protected by DBP

**Description**

When the DBP bit of PWR_DBPR register is cleared, the write access to backup domain content is expected to be disabled. However, PWR_BDCR1 register is not write-protected when DBP = 0.

**Workaround**

None.

## 2.2.14 The PWR_S3WU interrupt is generated for internal wakeup sources (WUSELx = 11)

**Description**

According to some reference manual versions, the PWR_S3WU interrupt is not generated for internal wakeup sources (when WUSELx = 11 in PWR_WUCR3 register). However, upon wakeup from Stop 3 mode with WUSELx = 11, two interrupts are generated: the PWR_S3WU and the internal wakeup source (RTC or TAMP) interrupts.

*Note: The PWR_S3WU flag (WUFx in PWR_WUSR) is automatically cleared when clearing the internal wakeup source flag.*

**Workaround**

Set the PWR_S3WU interrupt with a lower priority than the internal source interrupt (RTC or TAMP). Consequently, upon wakeup from Stop 3 mode, the RTC or TAMP interrupt is serviced first. Then, the PWR_S3WU interrupt is entered without the corresponding flag (WUFx in PWR_WUSR) being set. This interrupt service routine can be returned without clearing any flag.

## 2.2.15 OTG_FS is reset by OTGRST and DCMI_PSSIRST bits

**Description**

The OTG_FS peripheral is partially reset when the DCMI_PSSIRST bit is set in the RCC_AHB2RSTR1 register, which can lead to OTG_FS peripheral unexpected state and operation failure.

In addition, setting the OTGRST bit only partially resets the OTG_FS peripheral. OTG_FS is fully reset when both OTGRST and DCMI_PSSIRST bits are set.

**Workaround**

The DCMI_PSSIRST bit must not be set when OTG_FS is used simultaneously.

If DCMI or PSSI needs to be reset by software:

- In case OTG_FS is used simultaneously, this procedure must be applied:
  - For DCMI:
    1. Disable DCMI by writing twice DCM_CR = 0x0. The second write ensures that the ENABLE bit clear is completed.
    2. Clear interrupt flags by writing DCMI_ICR = 0x0000 001F.
    3. Write other registers to their reset value.
  - For PSSI:
    1. Disable PSSI by writing twice PSSI_CR = 0x4000 0000. The second write ensures that the ENABLE bit clear is completed.
    2. Clear interrupt flags by writing PSSI_ICR = 0x0000 0002.
    3. Write PSSI_IER = 0x0.
- In case OTG_FS is not used simultaneously, this procedure must be applied:
  1. Set DCMI_PSSIRST in the RCC_AHB2RSTR1 register.
  2. Reinitialize OTG_FS if needed in the application.

If OTG_FS needs to be reset by software:

- In case DCMI or PSSI is used simultaneously, this procedure must be applied:
  - Set OTGRST in RCC_AHB2RSTR1 and set CRST in OTG_GRSTCTL.
- In case DCMI or PSSI is not used simultaneously, this procedure must be applied:
  1. Set both DCMI_PSSIRST and OTGRST in the RCC_AHB2RSTR1 register
  2. Reinitialize DCMI/PSSI if needed in the application.

### 2.2.16 LSE medium-low drive mode is not available

#### Description

The LSE oscillator may not start or may stop in medium-low drive mode (LSEDRV = 01). Using this mode is forbidden.

#### Workaround

None

### 2.2.17 STM32U575QIIxxx/STM32U585QIIxxx bootloader communication interfaces are not available if TZEN is set

#### Description

The limitation only impacts STM32U575QII/STM32U585QII part numbers (UFBGA132 package without SMPS with 2 Mbytes of Flash memory).

When Arm® TrustZone® is enabled and boot from RSS is selected (BOOT0 = 1) in order to use the bootloader, the ST embedded bootloader is blocked, making all the bootloader interfaces unusable (USB-DFU, USART, SPI, $I^2C$, and FDCAN).

Consequently, the SFI (secure firmware install) using bootloader communication interfaces does not work, and the option bytes cannot be changed using the bootloader interface.

SFI and option bytes change using JTAG/SWD interface remains functional.

#### Workaround

None. This limitation is fixed.

### 2.2.18 Bootloader cannot increase RDP and modify the secure option bytes at the same time

#### Description

When the TrustZone® option bit is set (TZEN = 1), if a RDP level change to Level 0.5 or Level 1 is requested with bootloader at the same time than secure option bytes change, only RDP level change is done. Secure option bytes remain unchanged.

#### Workaround

Request first the secure option bytes change with bootloader when RDP is Level 0. Once done, request the RDP level change to Level 0.5 or Level 1.

### 2.2.19 HardFault on wakeup from Stop mode may occur in debug mode

#### Description

A HardFault may occur at wakeup from Stop mode when the following conditions are met:

- Device is in debug mode.
- DBG_STOP bit is set in DBGMCU_CR.
- A wakeup event/interrupt from an SRD peripheral (except EXTI) occurs in a timing window of four clock cycles during Stop mode entry sequence. SRD peripherals are the ones connected to AHB3 and APB3.

#### Workaround

None.

### 2.2.20 Full JTAG configuration without NJTRST pin cannot be used

#### Description

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

#### Workaround

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

### 2.2.21 EXTI LOCK bit does not lock privilege configuration

#### Description

Both security and privilege configuration of the extended interrupts and event controller (EXTI) must be locked when the LOCK bit of the EXTI lock register (EXTI_LOCKR) is set. The EXTI security configuration register (EXTI_SECCFGR1) is locked as expected, but the EXTI privilege configuration register (EXTI_PRIVCFGR1) can still be modified when the LOCK bit is set.

#### Workaround

None.

### 2.2.22 Device may be locked upon system reset under Stop 2 mode

#### Description

A system reset occurs when the MCU is in Stop 2 mode on LDO regulator, with at least, one RAM in power-down (SRAMx, Caches, or peripheral SRAMs). This may lock the device in a high consumption state (dozens of mA). The IWDG, the external NRST pin, and the BOR thresholds 1 to 4 are the reset sources that create this limitation.

Only a power-on sequence recovers from this state: $V_{DD}$ must drop below $V_{BOR0}$ (brownout reset minimum threshold value), then raise to its expected value.

This limitation is not present:

- upon wake-up from Stop 2 mode with other wake-up sources (GPIO or peripheral interrupt).
- if the device is supplied by the SMPS regulator during Stop 2 mode.
- in Stop 0, Stop 1, and Stop 3 modes.

**Workaround**

Keep all SRAMs powered on during Stop 2 mode that is all xRAMxPD/PDS bits in PWR_CRx registers must be kept at their reset state.

### 2.2.23 SRAM ECC error flags and addresses are updated only if interrupt is enabled

**Description**

In the reference manual RM0456, up to version 4, it is stated that:

- If a single error is detected, SEDC and CSEDC flags are set in RAMCFG_MxISR and RAMCFG_MxICR respectively. The ECC single error address is updated in RAMCFG_MxSEAR.
- If a double error is detected, DED and CDED flags are set in RAMCFG_MxISR and RAMCFG_MxICR respectively. The ECC double error address is updated in RAMCFG_MxDEAR.

However, the real behavior is that:

- If a single error is detected, SEDC and CSEDC flags are set, and the associated ECC single error address is updated only if the single error interrupt is enabled (SEIE bit of RAMCFG_MxIER is set).
- If a double error is detected, DED and CDED flags are set, and the associated ECC double error address is updated only if double error interrupt or NMI is enabled (DEIE bit or ECCNMI bit of RAMCFG_MxIER is set).

This is a documentation error rather than a device limitation.

**Workaround**

When the application needs to get the ECC error flags and addresses status without servicing the associated interrupts, the RAMCFG SEIE/DEIE interrupts must be enabled with the associated NVIC RAMCFG vector 5 disabled.

## 2.3 GTZC

### 2.3.1 LPTIM3_IN1 and LPTIM3_IN2 is protected by LPTIM1SEC instead of LPTIM3SEC

**Description**

When LPTIM3SEC = 1 in GTZC2_TZSC_SECCFGR1, LPTIM3 peripheral is secure, and all LPTIM alternate functions are expected to be functional only when the allocated GPIO is also secure. However, when LPTIM3SEC = 1 and the LPTIM3_IN1 or LPTIM3_IN2 allocated GPIO is non secure, the input data from GPIO is not forced to 0, except if LPTIM1SEC = 1.

When LPTIM3SEC = 0 in GTZC2_TZSC_SECCFGR1, LPTIM3 peripheral is non-secure, and all LPTIM alternate functions are expected to be functional whatever the allocated GPIO security configuration. However, when LPTIM3SEC = 0 and the LPTIM3_IN1 or LPTIM3_IN2 allocated GPIO is secure, the input data from GPIO is functional only if LPTIM1SEC = 0.

**Workaround**

In case LPTIM3_IN1 or LPTIM3_IN2 is used, both LPTIM1SEC and LPTIM3SEC must be configured to the same value (both secure or both non-secure).

### 2.3.2 TIMICSEL[2:0] is not protected by LPTIM2SEC

#### Description

The TIMICSEL[2:0] bitfield in RCC_CCIPR1 register selects a clock source for TIM16, TIM17 and LPTIM2 internal input capture. This bitfield is expected to be secure when either TIM16 or TIM17 or LPTIM2 is secure. However, LPTIM2 being secure (LPTIM2SEC = 1 in GTZC1_TZSC_SECCFGR1) does not protect TIMICSEL against non-secure accesses.

#### Workaround

Configure TIM16 or TIM17 as secure (TIM16SEC = 1 or TIM17SEC = 1 in GTZC1_TZSC_SECCFGR2 register) in order to protect TIMICSEL against non-secure accesses.

## 2.4 GPIO

### 2.4.1 Unexpected current consumption with GPIOs configured as I2C alternate functions

#### Description

An unexpected current consumption is observed on $V_{DD}$ when all the following conditions are met:

- GPIO is configured as I2C alternate function.
- Analog noise filter remains enabled (the ANFOFF bit of the I2C_CR1 register is cleared).
- The pad is set high (due to external or internal pull-up resistors).
- Entry in Stop 2 or Stop 3 is requested.

This overconsumption is observed with all I2C instances except I2C3.

This overconsumption can also be observed with all I2C instances, the I2C3 instance inclusive, when the pad is at a high level and the following sequence is applied:

1. GPIO is configured as I2C alternate function.
2. Analog noise filter is disabled (the ANFOFF bit of the I2C_CR1 register is set).

The extra consumption is typically 5 µA per GPIO.

#### Workaround

Apply one of the following measures before entering Stop 2 or Stop 3 modes:

- Configure the GPIO as analog input.
- Disable any external or internal pull-up device.

In all power modes, this limitation can also be worked around by disabling the analog noise filter (by setting the ANFOFF bit of the I2C_CR1 register) before configuring the GPIO as I2C alternate function. When the analog noise filter is disabled, the I2C digital filter must be used, which prevents the use of the I2C peripherals in Stop mode.

## 2.5 FMC

### 2.5.1 Dummy read cycles inserted when reading synchronous memories

#### Description

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

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

#### Workaround

None.

### 2.5.2 Wrong data read from a busy NAND memory

#### Description

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

#### Workaround

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

### 2.5.3 Data corruption upon a specific FIFO write sequence to synchronous PSRAM

#### Description

The following specific succession of events may cause the FMC, with the write FIFO buffer enabled, to send corrupted data to a synchronous PSRAM:

1. The application software sends a data write burst to the FMC that places the data onto consecutive write FIFO buffer locations.
2. A write FIFO buffer address roll-over occurs (upon the write burst in point 1 or upon some subsequent writes to the FMC).
3. The application software writes a data byte to the FMC. The data byte reaches the same write FIFO buffer location as the first byte of the data write burst under point 1.
4. The application software writes data of any size to the FMC while the FMC is sending the data byte from point 3 to a synchronous PSRAM.

Under these circumstances, the data byte from point 3 (being sent out to PSRAM in point 4) is corrupted, or, alternatively, the data byte immediately following that data byte is corrupted.

#### Workaround

Enable the DCACHE1 and configure the PSRAM data with cacheable write-back attribute in the MPU (DCACHE only performs 32-bit access in write back transaction). In addition, configure data size in SDMMC, GPDMA or DMA2D so that they do not perform byte access to PSRAM.

Other workarounds can be considered: use FMC peripheral in Asynchronous write mode, or disable the write FIFO buffer.

### 2.5.4 Unsupported AHB burst byte write to PSRAM

#### Description

Burst byte write operations to PSRAM do not work.

#### Workaround

Do not use burst byte write to program PSRAM. Only GPDMA can generate bytes burst write. When using GPDMA, use half-word or word burst write to program PSRAM.

## 2.6 OCTOSPI

### 2.6.1 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.

### 2.6.2 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.

### 2.6.3 Received data corrupted after arbitration ownership toggles when using clock mode 3 and no DQS for read direction

#### Description

When two OCTOSPI peripherals compete the ownership for the same external bus through the I/O manager and the following conditions are met:

- at least one of the OCTOSPIs operating in read mode
- at least one of the OCTOSPIs set with clock mode 3

the received data is corrupted due to a spurious sampling event when the ownership of the external bus toggles.

#### Workaround

Use clock mode 0, by setting the bit CKMODE of the OCTOSPI_DCR1 register (all memories known to date support clock mode 0).

### 2.6.4 Deadlock can occur under certain conditions

#### Description

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

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

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

- The Octo-SPI interface that currently owns the external bus (for example OCTOSPI1) waits for a transfer to occur with the external memory, to complete its transfer on the internal interconnect matrix bus.
- A data transfer request on the internal interconnect matrix bus arrives to the other Octo-SPI interface (for example OCTOSPI2).

This leads to an ownership conflict where:

- OCTOSPI2 cannot get ownership of the external bus which is currently in use by OCTOSPI1.
- OCTOSPI1 cannot get ownership of the internal interconnect matrix bus which is currently in use by OCTOSPI2.

#### Workaround

Apply one of the following measures:

- If any of the features generating automatic transfer split (MAXTRAN, REFRESH, CSBOUND, TIMEOUT) is set, OCTOSPI1 splits its transfer at some point in time, releasing the bus. OCTOSPI2 can then process its data, and when OCTOSPI1 gets ownership back again, it resumes its transfer thanks to its embedded capability to restart at the address following the last address accessed. In this case, the deadlock is resolved.
  Limitation of the workaround: The automatic resume of the transfer does not work with certain Flash memories in write direction only. These memories require an extra "write enable" command before resuming a write transfer. This "write enable" command is not generated by the OCTOSPI.
- The application must ensure that it has sufficient room left in the OCTOSPI internal FIFO for each and every transfer before launching it. The internal interconnect matrix bus activity no longer depends on what happens on external bus side, and the deadlock condition is avoided.

### 2.6.5 Read data corruption after partial read when crossing CSBOUND boundary

#### Description

If a first read access of 8 or 16 bits is performed within the last 4-byte word located just before page boundary (defined in CSBOUND[4:0] of OCTOSPI_CR register), and another read operation addresses the first word of the next page while the first read operation is still ongoing on memory side, then the second read returns invalid data.

This limitation is not present when the SDMMC accesses the OCTOSPI since the SDMMC performs only 32-bit read accesses.

#### Workaround

To avoid data corruption, perform only 32-bit read accesses to the OCTOSPI memory.

For system DMA, use only 32-bit data size for read accesses.

For CPU accesses, enable ICACHE and/or DCACHE, and configure the OCTOSPI memory as cacheable (CACHE only performs 32-bit read accesses).

If the DMA2D uses 4-, 8-, 16-, or 24-bpp picture format, add dummy pixels at the end of each picture line to increase the number of bytes that are skipped when performing read operations. As an example, a 128 x 28 x 24 bpp picture can be extended to 132 x 128 x 24 bpp to ensure that any sequence skipping bytes skips a minimum of two words on each line.

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

#### Description

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

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

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

An address is misaligned if:

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

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

#### Workaround

None.

### 2.6.7 Read data corruption after a few bytes are skipped when crossing a four-byte boundary

#### Description

A memory-mapped read is corrupted when the following sequence occurs:

1. An 8- or 16-bit read is performed in a four-byte window, and the last byte of this window is not read.
2. Any read in the next four-byte window is done before the completion of the previous read memory prefetch.

OCTOSPI immediately responds to this read request, even before the data are read from memory, thus resulting in incorrect data read.

If the next read operations are issued to consecutive addresses, then the read data are also corrupted.

This limitation is not present when the SDMMC accesses the OCTOSPI since the SDMMC performs only 32-bit read accesses.

#### Workaround

Apply one of the following measures:

- Perform only 32-bit memory-mapped read accesses. For CPU read accesses, enable ICACHE and/or DCACHE and configure the OCTOSPI memory as cacheable (CACHE only performs 32-bit read accesses).
  For system DMA, use only 32-bit data size for read accesses.
  If the DMA2D uses 4-, 8-, 16- or 24-bpp picture format, add dummy pixels at the end of each picture line to increase the number of bytes that are skipped when performing read operations. As an example, a 128 x 28 x 24 bpp picture can be extended to 132 x 128 x 24 bpp to ensure that any sequence skipping bytes skips a minimum of two words on each line.
- If this is not possible and an 8-bit or 16-bit read is done at the beginning of a four-byte window, ensure that the next access is not a read from the next four-byte window or that the second access occurs after the data at the skipped addresses are prefetched from memory.

### 2.6.8 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.

### 2.6.9 Automatic status-polling mode cannot be used with HyperFlash™ memories

#### Description

Some reference manuals mention that the automatic status-polling mode can be used with the HyperBus™ protocol. This is not possible since HyperFlash™ memories require two steps to read the status register (a write operation followed by a read command), while the automatic status-polling mode, already implemented in the regular-command protocol, requires a single read instruction to read back the status register.

This is a documentation issue rather than a product limitation.

#### Workaround

None.

### 2.6.10 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.

## 2.7 VREFINT

### 2.7.1 $t_{S\_vrefint}$ minimum value is wrong in datasheet

**Description**

STM32U585 and STM32U575 datasheets up to revision 6 (DS13086 and DS13737) specify that the "ADC sampling time when reading the internal reference voltage" parameter minimum value = 4 µs. This value is wrong, the correct value = 12.65 µs.

This is a documentation error rather than a device limitation.

**Workaround**

This datasheet error has no impact if the ADC sampling time is already configured at the maximum value. Otherwise, the ADC sampling time configuration may need to be modified.

## 2.8 ADC1

### 2.8.1 ADC_AWDy_OUT reset by non-guarded channels

**Description**

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

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

**Workaround**

When ADC_AWDy_OUT is enabled, it is recommended to use only the ADC channels that are guarded by a watchdog.

If ADC_AWDy_OUT is used with ADC channels that are not guarded by a watchdog, take only ADC_AWDy_OUT rising edge into account.

### 2.8.2 Injected data stored in the wrong ADC_JDRx registers

**Description**

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

**Workaround**

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

### 2.8.3 14-bit ADC offset error and integral linearity error values are increased in datasheets

**Description**

STM32U585 and STM32U575 datasheets up to revision 6 (DS13086 and DS13737) specify that:

- The offset error maximum value is ±5 LSB in singled ended mode, while the correct value is ±12 LSB.
- The integral linearity error maximum value is ±5 LSB in singled ended mode, while the correct value is ±7 LSB.

This is a documentation error rather than a device limitation. This error has no impact on the ADC accuracy.

**Workaround**

None.

## 2.9 ADC4

### 2.9.1 ADC4 conversion error when used simultaneously with ADC1

**Description**

The ADC4 conversion is disturbed when carried out simultaneously with an ADC1 conversion and results in an ADC4 conversion error. The error is due to a $V_{REF+}$ disturbance during an ADC1 sampling phase, and occurs regardless of which $V_{REF+}$ is provided: either by external reference, or by VREFBUF.

**Workaround**

ADC1 and ADC4 analog converter clocks must be identical (same clock source, same frequency, and same phase) to avoid any disturbance. This is possible only when both ADC prescalers are programmed without any division factor. The bitfields of the both registers listed below must be set to 0b0000:

- ADC1 prescaler: PRESC[3:0] in the ADC12_CCR register
- ADC4 prescaler: PRESC[3:0] in the ADC4_CCR register

ADC1 and ADC4 analog converter clock frequencies must not exceed 55 MHz and the ADC4 clock duty cycle must be set to between 45% and 55%. Selecting AHB clock as the ADC kernel clock limits the whole AHB to 55 MHz. Other independent clock sources can be used to avoid this drawback. As a result, the triggers from timers, that are clocked by AHB clock, have an uncertainty due to trigger synchronization delay from timers AHB clock to ADC asynchronous kernel clock. In case PLL output pll2_r_ck is used as ADC kernel clock, the PLL division (controlled by the PLL2R[6:0] bitfield in the RCC_ RCC_PLL2_DIVR register) must be set to an even division (division by 2, 4, and so on) to guarantee a 50% ratio.

*Note: The use of ADC kernel clock division factor is possible if ADC1 and ADC4 do not convert simultaneously.*

### 2.9.2 ADC clock request is active after reset

#### Description

After reset (system startup), the ADC requests the kernel clock even if it is not configured nor activated. This creates additional current consumption.

#### Workaround

Apply the following sequence at system startup:
1. Set ADEN bit of ADC_CR to enable the ADC, and wait a minimum time of two ADC_CLK cycles.
2. Set the ADDIS bit of ADC_CR, then wait until ADEN is cleared (it takes six ADC_CLK cycles).
3. Clear ADVREGEN bit of ADC_CR to disable the ADC voltage regulator, previously set by setting ADEN bit.

## 2.10 VREFBUF

### 2.10.1 $V_{REFBUF\_OUT}$ voltage overshoots in Range 4, Stop 1 or Stop 2 mode

#### Description

The $V_{REFBUF\_OUT}$ output voltage overshoots when started:
- while the regulator is in Range 4.
- while the device is in Stop 1 or Stop 2 mode.

#### Workaround

Modify the regulator voltage range to Range 1, 2, or 3 before enabling the voltage reference buffer.

## 2.11 MDF

### 2.11.1 In LFM mode MDF_CCK1 clock cannot be selected for SITFx interfaces

#### Description

In low-frequency master (LFM) mode, the input clock selectors of the SITFx serial interfaces do not allow selecting the MDF_CCK1 clock. The STIFx clock selector is controlled through the SCKSRC[1:0] bitfield of the corresponding MDF_SITFxCR register. The following table shows the expected and the actual behavior of the device:

Table 5. **SITFx clock selector operation**

| SCKSRC[1:0] | Clock selected | |
|---|---|---|
| | Expected | Actual |
| 00 | MDF_CCK0 | MDF_CCK0 |
| 01 | MDF_CCK1 | MDF_CCK0 |
| 1x | MDF_CKIx(1) | MDF_CKIx(1) |

1. *Disabled in LFM mode.*

As in the LFM mode the MDF_CCK1 cannot be selected and the MDF_CKIx is disabled, MDF_CCK0 is the only applicable clock for the SITFx interfaces.

#### Workaround

Always enable the MDF_CCK0 clock (by setting the CCK0EN bit of the MDF_CKGCR register) and select it for the SITFx interfaces, even in applications that only use the MDF_CCK1 clock output on the I/Os.

*Note: As the MDF_CCK1 and MDF_CCK0 clocks originate from the same clock source, the use of MDF_CCK1 for clocking external microphones and MDF_CCK0 for clocking the SITFx interfaces does not compromise the performance.*

## 2.12 TSC

### 2.12.1 TSC_G3_IO1/TSC_G1_IO4 are removed from the datasheet

#### Description

TSC_G3_IO1 and TSC_G1_IO4 alternate functions are implemented on PC2 and PC3 respectively, in all STM32U585 and STM32U575 devices revisions including future revisions. Due to noise sensitivity, these I/Os cannot reach the expected touch sensing performance. The TSC_G3_IO1 and TSC_G1_IO4 are removed from PC2 and PC3 alternate functions in STM32U585 and STM32U575 datasheets revision 6 and later revisions (DS13086 and DS13737).

This is a documentation error rather than a device limitation.

#### Workaround

None.

## 2.13 TIM

### 2.13.1 Bidirectional break mode not working with short pulses

#### Description

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

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

#### Workaround

None.

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

## 2.14 LPTIM

### 2.14.1 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_APByRSTRz register.

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

#### Description

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

#### Workaround

To avoid this issue the following formula must be respected:

{ARR, CMP} ≥ KER_CLK / (2* APB_CLK),

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

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

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

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

{CMP, ARR} ≥ 16 MHz / (2 * 1 MHz)

→ ARR must be ≥ 8 and CMP must be ≥ 8

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

### 2.14.3 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 standard clear sequence implemented in the HAL_LPTIM_IRQHandler in the STM32Cube is considered as the proper clear sequence.*

### 2.14.4 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.

### 2.14.5 Overcapture stops working when CCxOF flag is cleared in some specific conditions

#### Description

The overcapture stops working if the CCxOF flag in the LPTIMx_ISR register is cleared simultaneously with a new input capture event detection, that is, when an input capture pulse is active.

As a result, the LPTIM does not detect anymore overcapture events, no interrupt is generated, and the CCxOF flag is not set.

#### Workaround

Disable the corresponding input capture channel (the CCxE bit of the LPTIM_CCMRx register cleared) immediately after clearing the CCxOF flag, then enable the channel again after a delay that must be equal or greater than the value of (PRESC * 3) kernel clock cycles, PRESC[2:0] being the clock decimal division factor (1, 2, 4,..128).

## 2.15 IWDG

### 2.15.1 IWDG interrupt does not wake up from Stop mode

#### Description

The independent watchdog (IWDG) is functional in Stop mode and the device exits Stop 0, Stop 1, Stop 2 and Stop 3 modes in case of IWDG reset.

However, the early wakeup interrupt does not wake up the device from Stop 0, Stop 1 and Stop 2 modes.

*Note: As specified, the IWDG early wakeup interrupt does not wake up the device from Stop 3 mode.*

#### Workaround

None.

### 2.15.2 IWDG is stopped when BDRST is set

#### Description

IWDG, once started, is expected to stop only in case of system reset. However, the LSI (IWDG clock) is stopped when BDRST is set in the RCC_BDCR register.

In addition, the BDRST bit is not protected against non-secure access when LSI or IWDG is secure.

#### Workaround

If a Backup domain reset must be done, set BDRST and clear it right after to minimize the duration LSI is stopped.

BDRST can be protected against non-secure access by configuring at least one function of RTC or TAMP as secure.

## 2.16 RTC and TAMP

### 2.16.1 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.

### 2.16.2 Binary mode: SSR is not reloaded with 0xFFFF FFFF when SSCLR = 1

#### Description

When SSCLR bit of the RTC_ALRMxSSR register is set when in binary mode, SSR is reloaded with 0xFFFF FFFF at the end of the ck_apre cycle when RTC_SSR is set to RTC_ALRxBINR (*x* stands for either A or B)

RTC_SSR is not reloaded with 0xFFFF FFFF if RTC_ALRxBINR is modified while RTC_SSR is set to RTC_ALRxBINR. Rather, SSR continues to decrement.

#### Workaround

The workarounds are described for alarm A, and can be applied in the same manner for alarm B. Two workarounds are proposed, the second one requires to use the second alarm.

- Wait for one ck_apre cycle after an alarm A event before changing the RTC_ALRABINR register value.
- Do not reprogram RTC_ALRABINR following the alarm A event itself. Instead, use alarm B configured with RTC_ALRBBINR set to 0xFFFF FFFF, and reprogram RTC_ALRABINR after the alarm B event. This ensures that one ck_apre cycle elapses following the alarm A event.

### 2.16.3 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.

## 2.17 I2C

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

#### Description

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

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

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

#### Workaround

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

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

### 2.17.2 Spurious bus error detection in master mode

#### Description

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

#### Workaround

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

### 2.17.3 SDA held low upon SMBus timeout expiry in slave mode

#### Description

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

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

#### Workaround

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

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

## 2.18 USART

### 2.18.1 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.

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

**Description**

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

**Workaround**

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

### 2.18.3 Wakeup on a character match interrupt fails, in Autonomous mode with FIFO disabled

**Description**

When the peripheral is used in Autonomous mode with FIFO mode disabled, it is not possible to wake up the device from Stop mode using the enabled character match interrupt (CMIE bit set in USART_CR1 register).

**Workaround**

Use the DMA for data reception.

### 2.18.4 Wrong data received in Smartcard mode and 0.5 stop bit configuration

**Description**

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

**Workaround**

Use the 1.5 stop bit configuration.

### 2.18.5 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.

### 2.18.6 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.

### 2.18.7 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.*

## 2.19 LPUART

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

#### Description

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

#### Workaround

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

### 2.19.2 Wakeup on a character match interrupt fails, in Autonomous mode with FIFO disabled

#### Description

When the peripheral is used in Autonomous mode with FIFO mode disabled, it is not possible to wake up the device from Stop mode using the enabled character match interrupt (CMIE bit set in LPUART_CR1 register).

#### Workaround

Use the DMA for data reception.

### 2.19.3 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 LPUART_BRR register (BRR[15:0]) is not an integer, and is in the three to four range. A typical example is when the LSE 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:

- Increase the ratio between the LPUART kernel clock and the baud rate. To do so:
  - Increase the LPUART kernel clock frequency by using a clock different from the LSE clock, or
  - Decrease the baud rate.
- Generate the baud rate on the receiver side by using a higher frequency and applying oversampling techniques if supported.

## 2.20 SPI

### 2.20.1 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.

### 2.20.2 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.

### 2.20.3 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.

### 2.20.4 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_AUTOCR 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.*

### 2.20.5 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.

### 2.20.6 SPI slave stalls with masters not providing extra SCK periods upon *Not ready* signalling

#### Description

In Stop mode, the device SPI operating as slave with the *Ready* signalling 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* signalling.

### 2.20.7 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.

## 2.21 FDCAN

### 2.21.1 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.

### 2.21.2 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
```

## 2.22 OTG_FS

### 2.22.1 Host packet transmission may hang when connecting through a hub to a low-speed device

#### Description

When the USB on-the-go full-speed peripheral connects to a low-speed device via a hub, the transmitter internal state machine may hang. This leads, after a timeout expiry, to a port disconnect interrupt.

#### Workaround

None. However, increasing the capacitance on the data lines may reduce the occurrence.

## 2.23 UCPD

### 2.23.1 TXHRST upon write data underflow corrupting the CRC of the next packet

#### Description

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

#### Workaround

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

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

#### Description

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

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

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

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

#### Workaround

None.

### 2.23.3 Rp out of specification

#### Description

When UCPD is enabled as a source (SRC), the pull-up current source Rp is active. For 1.5 A mode, Rp may not respect the ±8% tolerance allowed in the USB Type C® standard. The measured tolerance is up to ±10% for $V_{DD}$ = 3.3 V ±5%, and up to ± 12% for 3.0 V ≤ $V_{DD}$ ≤ 3.6 V. It is in the ±8% tolerance for $V_{DD}$ = 3.3V ±5% and temperature between -10°C and 90°C.

#### Workaround

None

### 2.23.4 Rd out of specification

#### Description

When UCPD is enabled as a sink (SNK), the pull-down resistor Rd is active and may not respect the ±10% tolerance allowed in the USB Type C standard ([4590 Ω; 5610 Ω]), for discrimination of the different Rp values. It is in the expected range for 3.0 V ≤ $V_{DD}$ ≤ 3.6 V and temperature between -10°C and 90°C. For temperature between -40°C and -10°C the resistance may be down to 4200 Ω. For temperature between 90°C and 125°C the resistance may be up to 5800 Ω.

#### Workaround

None

## Important security notice

## Revision history

Table 6. **Document revision history**

| Date | Version | Changes |
|---|---|---|
| 24-Sep-2021 | 1 | Initial release |
| 21-Oct-2021 | 2 | Added erratum Incorrect Backup domain reset with VBAT and VDD supplied by the same power source.<br>Updated erratum Device authentication ID is not accessible in RDP Level 0. |
| 28-Feb-2022 | 3 | Added errata section MDF<br>Added errata:<br>• **System**: Incorrect Backup domain reset with VBAT and VDD supplied by different power source<br>• ICACHE and DCACHE access might be corrupted after exiting Stop 2 and Stop 3 modes<br>• LSI remains enabled even if not used in VBAT mode<br>• PWR_BDCR1 is not write-protected by DBP<br>• The PWR_S3WU interrupt is generated for internal wakeup sources (WUSELx = 11)<br>• OTG_FS is reset by OTGRST and DCMI_PSSIRST bits<br>• LSE medium-low drive mode is not available<br>• **GTZC**: LPTIM3_IN1 and LPTIM3_IN2 is protected by LPTIM1SEC instead of LPTIM3SEC<br>• TIMICSEL[2:0] is not protected by LPTIM2SEC<br>• **OCTOSPI**: Read data corruption after partial read when crossing CSBOUND boundary<br>• Deadlock or write-data corruption after spurious write to a misaligned address in OCTOSPI_ AR register<br>• Read data corruption after a few bytes are skipped when crossing a four-byte boundary<br>• **ADC4**: ADC clock request is active after reset<br>• **IWDG**: IWDG interrupt does not wake up from Stop mode<br>• IWDG is stopped when BDRST is set<br>• **UCPD**: Rp out of specification<br>• Rd out of specification<br>Removed erratum:<br>• **OCTOSPIM**: Unaligned write access to OCTOSPIM configuration registers failing |
| 13-Jul-2022 | 4 | Added errata section TSC<br>Added errata:<br>• **System:** HardFault on wakeup from Stop mode may occur in debug mode<br>• **USART:** Wrong data received in Smartcard mode and 0.5 stop bit configuration<br>• **LPUART:** Wrong data received when the communication nodes are two LPUART instances<br>• **TSC:** TSC_G3_IO1/TSC_G1_IO4 are removed from PC2/PC3<br>Modified errata:<br>• **System:** Incorrect backup domain reset with VBAT and VDD supplied by the same power source<br>• Incorrect backup domain reset with VBAT and VDD supplied by different power sources<br>• **OCTOSPI:** Read data corruption after partial read when crossing CSBOUND boundary<br>• Read data corruption after a few bytes are skipped when crossing a four-byte boundary<br>• **USART:** USART does not generate DMA requests after setting/clearing DMAT bit<br>• **LPUART:** LPUART does not generate DMA requests after setting/clearing DMAT bit |

| Date | Version | Changes |
|---|---|---|
| 24-Nov-2022 | 5 | Added device revision W<br><br>Added errata sections VREFINT and VREFBUF<br><br>Added errata:<br>• **System:** Section 2.2.18 Bootloader cannot increase RDP and modify the secure option bytes at the same time<br>• Section 2.2.20 Full JTAG configuration without NJTRST pin cannot be used<br>• **FMC:** Section 2.5.4 Unsupported AHB burst byte write to PSRAM<br>• **VREFINT:** Section 2.7.1 tS_vrefint minimum value is wrong in datasheet<br>• **ADC1:** Section 2.8.2 Injected data stored in the wrong ADC_JDRx registers<br>• **VREFBUF:** Section 2.10.1 VREFBUF_OUT voltage overshoots in Range 4, Stop 1 or Stop 2 mode<br>• **I2C:** Section 2.17.3 SDA held low upon SMBus timeout expiry in slave mode<br>• **USART:** Section 2.18.1 Data corruption due to noisy receive line<br><br>Modified errata:<br>• **System:** Section 2.2.2 Too low MSI frequency upon exit from Standby or Stop 3 mode<br>• Section 2.2.14 The PWR_S3WU interrupt is generated for internal wakeup sources (WUSELx = 11)<br>• Section 2.2.9 Incorrect backup domain reset with VBAT and VDD supplied by the same power source<br>• Section 2.2.10 Incorrect backup domain reset with VBAT and VDD supplied by different power sources<br><br>Removed errata not applicable for this product:<br>• **ADC1:** New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0<br>• Two consecutive context conversions fail when writing new context in ADC_JSQR just after previous context completion with JQDIS = 0 and JQM = 0 |
| 23-Mar-2023 | 6 | Added errata:<br>• **System:** EXTI LOCK bit does not lock privilege configuration<br>• Device may be locked upon system reset under Stop 2 mode<br>• SRAM ECC error flags and addresses are updated only if interrupt is enabled<br>• **OCTOSPI:** At least six cycles memory latency must be set when DQS is used for HyperBus™ memories<br>• Automatic status-polling mode cannot be used with HyperFlash™ memories<br>• Data write discarded in memory-mapped mode if a write to a misaligned address is directly followed by a request to the same address<br>• **ADC1:** 14-bit ADC offset error and integral linearity error values are increased in datasheets<br>• **TIM:** Bidirectional break mode not working with short pulses<br>• **USART:** Wrong data received by SPI slave receiver in Autonomous mode with CPOL = 1<br>• Received data may be corrupted upon clearing the ABREN bit<br>• Noise error flag set while ONEBIT is set<br>• **UCPD:** TXHRST upon write data underflow corrupting the CRC of the next packet<br>• Ordered set with multiple errors in a single K-code is reported as invalid<br><br>Modified errata:<br>• **System:** Incorrect backup domain reset with $V_{BAT}$ and $V_{DD}$ supplied by the same power source<br>• Incorrect backup domain reset with $V_{BAT}$ and $V_{DD}$ supplied by different power sources<br>• **TSC:** TSC_G3_IO1/TSC_G1_IO4 are removed from the datasheet<br><br>Removed errata not applicable for this product:<br>• **TIM:** Consecutive compare event missed in specific conditions<br>• Output compare clear not working with external counter reset |

# Contents

**1 Summary of device errata . . . . . 2**

**2 Description of device errata . . . . . 6**

2.1 Core . . . . . 6

2.1.1 Access permission faults are prioritized over unaligned Device memory faults . . . . . 6

2.2 System . . . . . 6

2.2.1 LSE disturbed by RTC/TAMP OUT function on PC13 . . . . . 6

2.2.2 Too low MSI frequency upon exit from Standby or Stop 3 mode . . . . . 6

2.2.3 LSE low drive mode is not functional. . . . . 7

2.2.4 Peripheral triggers connected to RTC wakeup timer interrupt instead of RTC wakeup timer trigger signal . . . . . 7

2.2.5 Device hang-up can occur when wakeup request received a few clock cycles before entering Stop 2 or Stop 3 mode . . . . . 7

2.2.6 SRAM2 page 1 content may be lost at high temperature when Standby mode is entered from Run mode with LDO enabled and Range 1, Range 2 or Range 3 selected . . . . . 7

2.2.7 Device authentication ID is not accessible in RDP Level 0 . . . . . 7

2.2.8 MSIK and HSI16 clocks cannot be stopped when used as kernel clock by ADCs, DAC1, MDF1, or ADF1 . . . . . 8

2.2.9 Incorrect backup domain reset with $V_{BAT}$ and $V_{DD}$ supplied by the same power source . . . 8

2.2.10 Incorrect backup domain reset with $V_{BAT}$ and $V_{DD}$ supplied by different power sources . . . 9

2.2.11 ICACHE and DCACHE access might be corrupted after exiting Stop 2 and Stop 3 modes 10

2.2.12 LSI remains enabled even if not used in $V_{BAT}$ mode . . . . . 11

2.2.13 PWR_BDCR1 is not write-protected by DBP. . . . . 11

2.2.14 The PWR_S3WU interrupt is generated for internal wakeup sources (WUSELx = 11) . . . 11

2.2.15 OTG_FS is reset by OTGRST and DCMI_PSSIRST bits . . . . . 11

2.2.16 LSE medium-low drive mode is not available . . . . . 12

2.2.17 STM32U575QIIxxx/STM32U585QIIxxx bootloader communication interfaces are not available if TZEN is set . . . . . 12

2.2.18 Bootloader cannot increase RDP and modify the secure option bytes at the same time . . 13

2.2.19 HardFault on wakeup from Stop mode may occur in debug mode . . . . . 13

2.2.20 Full JTAG configuration without NJTRST pin cannot be used . . . . . 13

2.2.21 EXTI LOCK bit does not lock privilege configuration . . . . . 13

2.2.22 Device may be locked upon system reset under Stop 2 mode . . . . . 13

2.2.23 SRAM ECC error flags and addresses are updated only if interrupt is enabled . . . . . 14

2.3 GTZC . . . . . 14

2.3.1 LPTIM3_IN1 and LPTIM3_IN2 is protected by LPTIM1SEC instead of LPTIM3SEC. . . . . 14

2.3.2 TIMICSEL[2:0] is not protected by LPTIM2SEC . . . . . 15

2.4 GPIO . . . . . 15

2.4.1 Unexpected current consumption with GPIOs configured as I2C alternate functions . . . . 15
2.5 FMC . . . . 15
2.5.1 Dummy read cycles inserted when reading synchronous memories . . . . 15
2.5.2 Wrong data read from a busy NAND memory . . . . 16
2.5.3 Data corruption upon a specific FIFO write sequence to synchronous PSRAM . . . . 16
2.5.4 Unsupported AHB burst byte write to PSRAM . . . . 16
2.6 OCTOSPI . . . . 16
2.6.1 Memory-mapped write error response when DQS output is disabled . . . . 16
2.6.2 Byte possibly dropped during an SDR read in clock mode 3 when a transfer gets automatically split . . . . 17
2.6.3 Received data corrupted after arbitration ownership toggles when using clock mode 3 and no DQS for read direction . . . . 17
2.6.4 Deadlock can occur under certain conditions . . . . 17
2.6.5 Read data corruption after partial read when crossing CSBOUND boundary . . . . 18
2.6.6 Deadlock or write-data corruption after spurious write to a misaligned address in OCTOSPI_ AR register . . . . 18
2.6.7 Read data corruption after a few bytes are skipped when crossing a four-byte boundary . 19
2.6.8 At least six cycles memory latency must be set when DQS is used for HyperBus™ memories . . . . 19
2.6.9 Automatic status-polling mode cannot be used with HyperFlash™ memories . . . . 19
2.6.10 Data write discarded in memory-mapped mode if a write to a misaligned address is directly followed by a request to the same address . . . . 20
2.7 VREFINT . . . . 20
2.7.1 $t_{S\_vrefint}$ minimum value is wrong in datasheet . . . . 20
2.8 ADC1 . . . . 20
2.8.1 ADC_AWDy_OUT reset by non-guarded channels . . . . 20
2.8.2 Injected data stored in the wrong ADC_JDRx registers . . . . 21
2.8.3 14-bit ADC offset error and integral linearity error values are increased in datasheets . . . 21
2.9 ADC4 . . . . 21
2.9.1 ADC4 conversion error when used simultaneously with ADC1 . . . . 21
2.9.2 ADC clock request is active after reset . . . . 22
2.10 VREFBUF . . . . 22
2.10.1 $V_{REFBUF\_OUT}$ voltage overshoots in Range 4, Stop 1 or Stop 2 mode . . . . 22
2.11 MDF . . . . 22
2.11.1 In LFM mode MDF_CCK1 clock cannot be selected for SITFx interfaces . . . . 22
2.12 TSC . . . . 23
2.12.1 TSC_G3_IO1/TSC_G1_IO4 are removed from the datasheet . . . . 23
2.13 TIM . . . . 23
2.13.1 Bidirectional break mode not working with short pulses . . . . 23

2.14 LPTIM . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
2.14.1 Device may remain stuck in LPTIM interrupt when entering Stop mode . . . . . . . . . . . . . . 23
2.14.2 ARRM and CMPM flags are not set when APB clock is slower than kernel clock . . . . . . . 24
2.14.3 Device may remain stuck in LPTIM interrupt when clearing event flag . . . . . . . . . . . . . . . 24
2.14.4 Interrupt status flag is cleared by hardware upon writing its corresponding bit in LPTIM_DIER register . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
2.14.5 Overcapture stops working when CCxOF flag is cleared in some specific conditions . . . . 25
2.15 IWDG . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
2.15.1 IWDG interrupt does not wake up from Stop mode . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
2.15.2 IWDG is stopped when BDRST is set . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
2.16 RTC and TAMP . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
2.16.1 Alarm flag may be repeatedly set when the core is stopped in debug . . . . . . . . . . . . . . . . 25
2.16.2 Binary mode: SSR is not reloaded with 0xFFFF FFFF when SSCLR = 1 . . . . . . . . . . . . . 26
2.16.3 Parasitic tamper detection when debugger is used in RDP Level 0 . . . . . . . . . . . . . . . . . 26
2.17 I2C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
2.17.1 Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
2.17.2 Spurious bus error detection in master mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
2.17.3 SDA held low upon SMBus timeout expiry in slave mode . . . . . . . . . . . . . . . . . . . . . . . 27
2.18 USART . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
2.18.1 Data corruption due to noisy receive line. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
2.18.2 USART does not generate DMA requests after setting/clearing DMAT bit. . . . . . . . . . . . . 28
2.18.3 Wakeup on a character match interrupt fails, in Autonomous mode with FIFO disabled . . 28
2.18.4 Wrong data received in Smartcard mode and 0.5 stop bit configuration . . . . . . . . . . . . . . 28
2.18.5 Wrong data received by SPI slave receiver in Autonomous mode with CPOL = 1. . . . . . . 28
2.18.6 Received data may be corrupted upon clearing the ABREN bit. . . . . . . . . . . . . . . . . . . . . 28
2.18.7 Noise error flag set while ONEBIT is set . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
2.19 LPUART . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
2.19.1 LPUART does not generate DMA requests after setting/clearing DMAT bit. . . . . . . . . . . . 29
2.19.2 Wakeup on a character match interrupt fails, in Autonomous mode with FIFO disabled . . 29
2.19.3 Possible LPUART transmitter issue when using low BRR[15:0] value. . . . . . . . . . . . . . . . 29
2.20 SPI . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
2.20.1 Possible corruption of last-received data depending on CRCSIZE setting. . . . . . . . . . . . . 30
2.20.2 MODF flag cannot generate interrupt . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
2.20.3 RDY output failure at high serial clock frequency . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
2.20.4 Master communication suspension fails in Autonomous mode . . . . . . . . . . . . . . . . . . . . 31
2.20.5 SPE may not be cleared upon MODF event . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
2.20.6 SPI slave stalls with masters not providing extra SCK periods upon *Not ready* signalling . 31

2.20.7 Truncation of SPI output signals after EOT event . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
2.21 FDCAN . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
2.21.1 Desynchronization under specific condition with edge filtering enabled. . . . . . . . . . . . . . . 32
2.21.2 Tx FIFO messages inverted under specific buffer usage and priority setting . . . . . . . . . . . 32
2.22 OTG_FS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
2.22.1 Host packet transmission may hang when connecting through a hub to a low-speed device . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
2.23 UCPD . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
2.23.1 TXHRST upon write data underflow corrupting the CRC of the next packet . . . . . . . . . . . 33
2.23.2 Ordered set with multiple errors in a single K-code is reported as invalid . . . . . . . . . . . . . 33
2.23.3 Rp out of specification. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
2.23.4 Rd out of specification. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
**Important security notice . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35**
**Revision history . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36**