# STM32L4Rxxx and STM32L4Sxxx device errata

## Applicability

This document applies to the part numbers of STM32L4Rxxx and STM32L4Sxxx 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 RM0432.

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 |
|---|---|
| STM32L4Rxxx | STM32L4R5AG, STM32L4R5AI, STM32L4R5QG, STM32L4R5QI, STM32L4R5VG, STM32L4R5VI, STM32L4R5ZG, STM32L4R5ZI, STM32L4R7AI, STM32L4R7VI, STM32L4R7ZI, STM32L4R9AG, STM32L4R9AI, STM32L4R9VG, STM32L4R9VI, STM32L4R9ZG, STM32L4R9ZI |
| STM32L4Sxxx | STM32L4S5AI, STM32L4S5QI, STM32L4S5VI, STM32L4S5ZI, STM32L4S7AI, STM32L4S7VI, STM32L4S7ZI, STM32L4S9AI, STM32L4S9VI, STM32L4S9ZI |

**Table 2. Device variants**

| Reference | Silicon revision codes | |
|---|---|---|
| | Device marking(1) | REV_ID(2) |
| STM32L4Rxxx STM32L4Sxxx | Y | 0x1003 |
| | W | 0x100F |

1. *Refer to the device data sheet 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 STM32L4Rxxx and STM32L4Sxxx device limitations and their status:

A = workaround available

N = no workaround available

P = partial workaround available

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

**Table 3. Summary of device limitations**

| Function | Section | Limitation | Status | |
|---|---|---|---|---|
| | | | Rev. Y | Rev. W |
| Core | 2.1.1 | Interrupted loads to SP can cause erroneous behavior | A | A |
| System | 2.2.1 | Full JTAG configuration without NJTRST pin cannot be used | A | A |
| | 2.2.2 | Data cache might be corrupted during Flash memory read-while-write operation | A | A |
| | 2.2.3 | Flash memory might not be accessible when AHB prescaler is greater than eight | N | N |
| | 2.2.4 | Flash OPTVERR flag is always set after system reset | N | N |
| | 2.2.5 | HSE oscillator long startup at low voltage | P | P |
| | 2.2.6 | SDMMC1SMEN bit of RCC_AHB2SMENR register only modifiable with word access | A | A |
| | 2.2.7 | First double-word of Flash memory corrupted upon reset or power down while programming | A | A |
| | 2.2.8 | Unstable LSI when it clocks RTC or CSS on LSE | P | P |
| | 2.2.9 | LTDC and DSI not functional with Stop 2 | P | - |
| | 2.2.10 | Regulator startup failure at low VDD | N | - |
| | 2.2.11 | 1-Mbyte devices wrongly configured as 2-Mbyte | N | - |
| FW | 2.3.1 | Firewall protection size limitation | N | N |
| | 2.3.2 | Spurious Firewall reset is generated when accessing FMC or OCTOSPI | P | P |
| DMA | 2.4.1 | DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear | A | A |
| DMAMUX | 2.5.1 | SOFx not asserted when writing into DMAMUX_CFR register | N | N |
| | 2.5.2 | OFx not asserted for trigger event coinciding with last DMAMUX request | N | N |
| | 2.5.3 | OFx not asserted when writing into DMAMUX_RGCFR register | N | N |
| | 2.5.4 | Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event | A | A |
| FMC | 2.6.1 | Dummy read cycles inserted when reading synchronous memories | N | N |
| | 2.6.2 | Wrong data read from a busy NAND memory | A | A |

| Function | Section | Limitation | Status | |
|---|---|---|---|---|
| | | | Rev. Y | Rev. W |
| OCTOSPI | 2.7.1 | CSBOUND setting not fully respected | A | A |
| | 2.7.2 | Auto-polling mode not functional with new octal memories | A | A |
| | 2.7.3 | Command phase must be octal for octal transfers | A | A |
| | 2.7.4 | Write data lost with Clock mode 3 | A | A |
| | 2.7.5 | Deadlock upon disabling OCTOSPI during memory-mapped write | A | A |
| | 2.7.6 | Deadlock in dual-flash configuration with odd number of bytes | A | A |
| | 2.7.7 | No transfer error interrupt upon indirect write to address exceeding the limit | N | N |
| | 2.7.8 | DHQC not effective if DDTR not set | A | A |
| | 2.7.9 | Indirect read and auto-polling transfers without address phase not starting | A | A |
| | 2.7.10 | Unaligned AHB write requests not supported | N | N |
| | 2.7.11 | Data mask failing with odd start address writes to non-HyperBus memory | N | N |
| | 2.7.12 | Data masking for odd byte writes only working with D1/D0 ordering | A | A |
| | 2.7.13 | Memory-mapped read of the last memory space byte not possible in SDR octal mode | A | A |
| | 2.7.14 | Single-byte memory-mapped write to odd octal DDR address failing when a higher-priority event occurs | A | A |
| ADC | 2.8.1 | Injected queue of context is not available in case of JQM = 0 | N | N |
| | 2.8.2 | Writing ADCx_JSQR when JADCSTART and JQDIS are set might lead to incorrect behavior | N | N |
| | 2.8.3 | Wrong ADC result if conversion done late after calibration or previous conversion | A | A |
| | 2.8.4 | Spurious temperature measurement due to spike noise | A | - |
| COMP | 2.9.1 | Comparator outputs cannot be configured in open-drain | N | N |
| TIM | 2.11.1 | HSE/32 is not available for TIM16 input capture if RTC clock is disabled or other than HSE | A | A |
| LPTIM | 2.12.1 | MCU may remain stuck in LPTIM interrupt when entering Stop mode | A | A |
| | 2.12.2 | LPTIM1 outputs cannot be configured as open-drain | N | N |
| RTC and TAMP | 2.13.1 | RTC interrupt can be masked by another RTC interrupt | A | A |
| | 2.13.2 | Calendar initialization may fail in case of consecutive INIT mode entry | A | A |
| | 2.13.3 | RTC_REFIN and RTC_OUT on PB2 not operating in Stop 2 mode | P | P |
| I2C | 2.14.1 | 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave | A | A |
| | 2.14.3 | Wrong data sampling when data setup time (tSU;DAT) is shorter than one I2C kernel clock period | P | P |
| | 2.14.4 | Spurious bus error detection in master mode | A | A |
| | 2.14.5 | Last-received byte loss in reload mode | P | P |
| | 2.14.6 | Spurious master transfer upon own slave address match | P | P |
| | 2.14.7 | START bit is cleared upon setting ADDRCF, not upon address match | P | P |
| USART | 2.15.1 | nRTS is active while RE = 0 or UE = 0 | A | A |
| | 2.15.2 | UDR flag set while the SPI slave transmitter is disabled | A | A |
| LPUART | 2.16.1 | LPUART1 outputs cannot be configured as open-drain | N | N |

| Function | Section | Limitation | Status | |
|---|---|---|---|---|
| | | | Rev. Y | Rev. W |
| SPI | 2.17.1 | BSY bit may stay high when SPI is disabled | A | A |
| | 2.17.2 | BSY bit may stay high at the end of data transfer in slave mode | A | A |
| bxCAN | 2.18.1 | bxCAN time-triggered communication mode not supported | A | A |
| OTG_FS | 2.19.1 | Data FIFO gets corrupted if the write sequence to the transmit FIFO is interleaved with other OTGFS register access | N | N |

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

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

| Function | Section | Documentation erratum |
|---|---|---|
| DMAMUX | 2.5.5 | DMAMUX_RGCFR register is write-only, not read-write |
| | 2.5.6 | DMA request counter not kept at GNBREQ bitfield value as long as the corresponding request channel is disabled |
| | 2.5.7 | Synchronization event discarded if selected input DMA request is not active |
| TSC | 2.10.1 | Inhibited acquisition in short transfer phase configuration |
| I2C | 2.14.2 | Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C |
| SPI | 2.17.3 | CRC error in SPI slave mode if internal NSS changes before CRC transfer |

# 2 Description of device errata

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

arm

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

## 2.1 Core

Errata notice for the Arm® Cortex®-M4F core revision r0p1 is available from http://infocenter.arm.com.

### 2.1.1 Interrupted loads to SP can cause erroneous behavior

This limitation is registered under Arm ID number 752770 and classified into “Category B”. Its impact to the device is minor.

**Description**

If an interrupt occurs during the data-phase of a single word load to the stack-pointer (SP/R13), erroneous behavior can occur. In all cases, returning from the interrupt will result in the load instruction being executed an additional time. For all instructions performing an update to the base register, the base register will be erroneously updated on each execution, resulting in the stack-pointer being loaded from an incorrect memory location.

The affected instructions that can result in the load transaction being repeated are:

- LDR SP, [Rn],#imm
- LDR SP, [Rn,#imm]!
- LDR SP, [Rn,#imm]
- LDR SP, [Rn]
- LDR SP, [Rn,Rm]

The affected instructions that can result in the stack-pointer being loaded from an incorrect memory address are:

- LDR SP,[Rn],#imm
- LDR SP,[Rn,#imm]!

As compilers do not generate these particular instructions, the limitation is only likely to occur with hand-written assembly code.

**Workaround**

Both issues may be worked around by replacing the direct load to the stack-pointer, with an intermediate load to a general-purpose register followed by a move to the stack-pointer.

## 2.2 System

### 2.2.1 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. Only the 4-wire JTAG port configuration is impacted.

**Workaround**

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

### 2.2.2 Data cache might be corrupted during Flash memory read-while-write operation

#### Description

When a write to the internal Flash memory is done, the data cache is normally updated to reflect the data value update. During this data cache update, a read to the other Flash memory bank may occur; this read can corrupt the data cache content and subsequent read operations at the same address (cache hits) will be corrupted.

This limitation only occurs in dual bank mode, when reading (data access or code execution) from one bank while writing to the other bank with data cache enabled.

#### Workaround

When the application is performing data accesses in both Flash memory banks, the data cache must be disabled by resetting the DCEN bit before any write to the Flash memory. Before enabling the data cache again, it must be reset by setting and then resetting the DCRST bit.

Code example:

```
/* Disable data cache  */
__HAL_FLASH_DATA_CACHE_DISABLE();

/* Set PG bit */
SET_BIT(FLASH->CR, FLASH_CR_PG);

/* Program the Flash word */
WriteFlash(Address, Data);

/* Reset data cache */
__HAL_FLASH_DATA_CACHE_RESET();

/* Enable data cache */
__HAL_FLASH_DATA_CACHE_ENABLE();
```

### 2.2.3 Flash memory might not be accessible when AHB prescaler is greater than eight

#### Description

When the AHB prescaler is set to a value greater than eight, the system clock (HCLK) is divided by four or more. When a system reset occurs during a Flash memory access, the Flash memory might become inaccessible. When this issue occurs, a hard fault exception is generated when the Flash memory is accessed. The application running from Flash memory can restart only after a power-on reset.

#### Workaround

None. Do not use a system clock divider value greater than two.

### 2.2.4 Flash OPTVERR flag is always set after system reset

#### Description

During option byte loading, the options are read by double word with ECC. If the word and its complement are not matching, the OPTVERR flag is set.

However, the OPTVERR flag is always set after a system reset despite all option bytes being loaded and read correctly.

#### Workaround

After reset, clear the OPTVERR flag in FLASH_SR register.

### 2.2.5 HSE oscillator long startup at low voltage

#### Description

When $V_{DD}$ is below 2.7 V, the HSE oscillator may take longer than specified to start up. Several hundred milliseconds might elapse before the HSERDY flag in the RCC_CR register is set.

#### Workaround

The following sequence is recommended:

1. Configure PH0 and PH1 as standard GPIOs in output mode and low-level state.
2. Enable the HSE oscillator.

### 2.2.6 SDMMC1SMEN bit of RCC_AHB2SMENR register only modifiable with word access

#### Description

The SDMMC1SMEN bit of the RCC_AHB2SMENR register cannot be modified with byte and half-word accesses to the register. Only word access is effective.

#### Workaround

Only use word access to the RCC_AHB2SMENR register to modify the SDMMC1SMEN bit.

### 2.2.7 First double-word of Flash memory corrupted upon reset or power down while programming

#### Description

Power-down and external reset events occurring during Flash memory program operation may result in zeroing its first double-word (64 bits on the address 0x0800 0000). When this occurs, the boot sequence ends immediately after reset, generating Hard Fault.

In most cases, this corruption is temporary and the correct value is read again after a few milliseconds.

*Note: Corruption of Flash memory word on the address accessed in the instant of a reset event occurring during program or erase operation is a normal (specified) behavior.*

#### Workaround

Clone the first Flash memory page contents at another Flash memory location to use as a backup page. In the Hard Fault handler:

1. Compare the first Flash memory double-word content with the value backed up. If different:
   a. Erase the first Flash memory page.
   b. From the backup page, restore the first Flash memory page contents, excluding the first double-word.
   c. From the backup page, restore the first Flash memory double-word.
2. Execute a software reset (by setting the SYSRESETREQ bit), to resume execution.

*Note: In the process of firmware development, the first two 32-bit words of the Flash memory are used by the core as program counter (PC) and as stack pointer (SP), respectively, so this 64-bit value as well as the remaining content of the page can vary upon each build.*

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

#### Description

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

- LSI clocks the RTC, or it clocks the clock security system (CSS) on LSE (which holds when the LSECSSON bit set), and
- the $V_{DD}$ power domain is reset while the backup domain is not reset, which happens:
  - upon exiting Shutdown mode
  - if $V_{BAT}$ is separate from $V_{DD}$ and $V_{DD}$ goes off then on

– if $V_{BAT}$ is tied to $V_{DD}$ (internally in the package for products not featuring the VBAT pin, or externally) and a short (< 1 ms) $V_{DD}$ drop under $V_{DD}$(min) occurs

**Workaround**

Apply one of the following measures:

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

### 2.2.9 LTDC and DSI not functional with Stop 2

**Description**

If the device enters Stop 2 mode while LTDC and/or DSI are enabled, then upon exiting Stop 2 mode, LTDC and/or DSI fetch wrong data from their respective internal FIFOs. Power-on reset is necessary after exiting Stop 2 mode, for LTDC and/or DSI to display correct data.

**Workaround**

Choose Stop 0 or Stop 1 instead of Stop 2 when using LTDC and/or DSI.

### 2.2.10 Regulator startup failure at low $V_{DD}$

**Description**

Depending on $V_{DD}$ rising speed, the internal regulator might not start correctly with $V_{DD}$ below 2.9 V.

**Workaround**

None.

### 2.2.11 1-Mbyte devices wrongly configured as 2-Mbyte

**Description**

STM32L4RxxG/SxxG 1-Mbyte part number devices are wrongly configured as 2-Mbyte devices. However, the Flash memory size FLASH_SIZE[15:0] in Flash size data register is configured correctly to 1 Mbyte.

Consequently, the bank2 erase and page erase in bank2 on those devices are not performed correctly.

In order to check the device configuration:

- Read address 0x1FFF 7500. If bit 24 is set, the device is 2-Mbyte. If the bit 24 is cleared , it is a 1-Mbyte device.

**Workaround**

The problem is present on STM32L4RxxG/SxxG devices with date code week 48 2018 or earlier.

The STM32L4RxxG/SxxG 1-Mbyte part number devices with date code week 49 2018 or later have the correct 1-Mbyte configuration.

## 2.3 FW

### 2.3.1 Firewall protection size limitation

**Description**

Only 128 Kbytes of SRAM1 can be protected by the firewall instead of the full memory space (192 Kbytes of SRAM1).

#### Workaround

None.

### 2.3.2 Spurious Firewall reset is generated when accessing FMC or OCTOSPI

#### Description

If a Firewall is enabled on the SRAM1 area, a Firewall reset occurs even if there is no access to SRAM 1 when the two conditions below are met:

- An access is performed to the FMC or the OctoSPI address with the same 16 least- significant bits as those in the SRAM1 protected area.
- The FMC or OctoSPI access is preceded by any SRAM1 access.

#### Workaround

Use SRAM2 or SRAM3 as system RAM and avoid any access to SRAM1 before accessing FMC or OCTOSPI.

## 2.4 DMA

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

#### Description

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

This issue does not occur with ST's HAL software that does not use and clear the GIFx flag, but uses and clears the HTIFx, TCIFx, and TEIFx specific event flags instead.

#### Workaround

Do not clear GIFx flags. Instead, use HTIFx, TCIFx, and TEIFx specific event flags and their corresponding clear bits.

## 2.5 DMAMUX

### 2.5.1 SOFx not asserted when writing into DMAMUX_CFR register

#### Description

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

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

#### Workaround

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

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

#### Description

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

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

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

#### Workaround

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

### 2.5.3 OFx not asserted when writing into DMAMUX_RGCFR register

#### Description

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

#### Workaround

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

### 2.5.4 Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event

#### Description

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

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

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

#### Workaround

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

### 2.5.5 DMAMUX_RGCFR register is write-only, not read-write

#### Description

Some reference manual revisions may wrongly state that the DMAMUX_RGCFR register is read-write, while it is write-only.

This is a description inaccuracy issue rather than a product limitation.

#### Workaround

No application workaround is required.

### 2.5.6 DMA request counter not kept at GNBREQ bitfield value as long as the corresponding request channel is disabled

#### Description

Some reference manual revisions may wrongly state that the DMA request counter is kept at GNBREQ bitfield value as long as the corresponding request channel is disabled.

Instead, at the DMA request counter underrun, the corresponding request generator channel stops generating DMA requests. Then upon the next trigger event, the DMA request counter is automatically reloaded with the GNBREQ bitfield value, regardless whether the corresponding request channel is enabled or disabled.

This is a description inaccuracy issue rather than a product limitation.

#### Workaround

No application workaround is required.

### 2.5.7 Synchronization event discarded if selected input DMA request is not active

#### Description

Some reference manual revisions may state that upon the detected edge of the synchronization input, the selected input DMA request line is connected to the DMAMUX multiplexer channel x output.

However, if the synchronization event occurs when the selected input DMA request line is not active (not asserted), the synchronization event is discarded. Connecting of a selected input DMA request line becoming active afterward requires a new synchronization event.

This is a description inaccuracy issue rather than a product limitation.

#### Workaround

No application workaround is required.

## 2.6 FMC

### 2.6.1 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.

### 2.6.2 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.

## 2.7 OCTOSPI

### 2.7.1 CSBOUND setting not fully respected

**Description**

The CS boundary function is expected to split transfers just before the byte with the address $2^{CSBOUND}$, where CSBOUND is the value of CSBOUND[4:0] bitfield of the OCTOSPI_DCR3 register.
However, the split can spuriously occur up to 32 bytes after the expected byte.

**Workaround**

For external RAMs that limit the number of bytes in a single transfer in order to guarantee refresh, take into account the failure described when setting the CSBOUND value, by ensuring that $2^{CSBOUND}$ + 32 does not exceed the maximum number of bytes per single transfer allowed by the RAM.

### 2.7.2 Auto-polling mode not functional with new octal memories

**Description**

Auto-polling mode with octal memories is not functional.

**Workaround**

Use memory-mapped access to poll the status registers.

### 2.7.3 Command phase must be octal for octal transfers

**Description**

Commands whose command phase use one, two, or four signal lines are not supported if one or more of the subsequent phases (address, alternate bytes, data) use eight signal lines.

**Workaround**

Configure the memory to accept eight-line commands (IMODE[2:0] bitfield of the OCTOSPI_CCR register) if eight-line address/data operation modes are used by the application.

*Note: Some memories allow four-line command and eight-line address/data, but all such memories known to date allow an alternative mode where eight-line commands are accepted.*

### 2.7.4 Write data lost with Clock mode 3

**Description**

In eight-line SDR configuration with Clock mode 3 (the clock remaining high between transfers), the last one or two bytes of an indirect write transfer might not be sent to the memory after the following sequence of events:

1. All except one or two data bytes of an indirect transfer are written to the data register (OCTOSPI_DR) and the OCTOSPI FIFO buffer.
2. All data bytes from the OCTOSPI FIFO buffer are sent to the memory, the FIFO buffer gets empty and the clock to the memory stops.
3. The last one or two bytes are written to the data register.

**Workaround**

Use Clock mode 0, by setting the bit CKMODE of the OCTOSPI_DCR1 register.

*Note: All memories known to date support Clock mode 0.*

### 2.7.5 Deadlock upon disabling OCTOSPI during memory-mapped write

**Description**

Disabling the OCTOSPI peripheral while a memory-mapped write is ongoing (while the BUSY flag of the OCTOSPI_SR register is high) causes the OCTOSPI AHB interface to stall upon the following memory-mapped write request, which may also lead to a general system deadlock.

**Workaround**

Apply one of the following measures:

- Let memory-mapped writes end (the BUSY flag of the OCTOSPI_SR register get cleared) before disabling the peripheral.
- Reset the OCTOSPI peripheral when stalled.

### 2.7.6 Deadlock in dual-flash configuration with odd number of bytes

**Description**

In dual-flash configuration, bytes at even addresses are written to one memory and bytes at odd address to the other. As in this mode the OCTOSPI sends two bytes at a time (one to either memory), it always writes an even total number of bytes.

The peripheral is expected to reject the last byte of any write request with odd number of bytes to contiguous addresses. However, when there is another write request not contiguous with the former, the peripheral hangs when arriving at the last (odd) byte of the first write request, and the last byte remains in the write FIFO buffer.

**Workaround**

Apply one of the following measures:

- Ensure that all write requests in dual-flash mode contain even number of bytes.
- Reset the OCTOSPI peripheral when stalled.

### 2.7.7 No transfer error interrupt upon indirect write to address exceeding the limit

**Description**

The transfer error flag TEF of the OCTOSPI_SR register is not set and interrupt not generated with the octal memories upon an indirect write to an address exceeding the limit.

**Workaround**

None.

### 2.7.8 DHQC not effective if DDTR not set

**Description**

With the DHQC bit of the OCTOSPI_TCR set (to insert a quarter-cycle delay) and the DDTR bit of the OCTOSPI_CCR register cleared (no DTR for the data phase), the delay is not inserted.

**Workaround**

Always set the DDTR bit, even for transactions with no data phase.

### 2.7.9 Indirect read and auto-polling transfers without address phase not starting

**Description**

Indirect read and auto-polling transfers, configured through the CCR register to contain command and SDR or DDR octal data phases but no address phase, do not start.

**Workaround**

Configure the transfer to contain address phase and no command phase, then send the command through the address register.

### 2.7.10 Unaligned AHB write requests not supported

**Description**

Upon a non-aligned write access request from AHB master, the OCTOSPI peripheral does not automatically align the data. With byte and half-word write accesses, the lowest-significant byte and the two lowest-significant input data bytes, respectively, are written to the memory, regardless of the non-alignment attribute.

**Workaround**

None.

### 2.7.11 Data mask failing with odd start address writes to non-HyperBus memory

**Description**

With memory type set to different from HyperBus™ (the MTYP[2:0] bitfield of the OCTOSPI_DCR1 register set to a value other than 100 or 101), the DQS write mask for writes starting with odd address is not asserted even though enabled (the DQSE bit of the OCTOSPI_WCCR register set).

**Workaround**

None.

*Tip: For memory types different from HyperBus, always start writes with even address.*

### 2.7.12 Data masking for odd byte writes only working with D1/D0 ordering

**Description**

The data masking for odd byte writes operates with only the memory types using D1/D0 ordering.

**Workaround**

Set the MTYP[2:0] bitfield of the OCTOSPI_DCR1 register to a memory type with D1/D0 ordering, for example Macronix (MTYP[2:0] = 001).

### 2.7.13 Memory-mapped read of the last memory space byte not possible in SDR octal mode

**Description**

Memory-mapped read of the last byte of the memory space defined through the DEVSIZE[4:0] bitfield of the OCTOSPI_DCR1 register spuriously always returns zero. A subsequent memory-mapped read not separated from the previous memory-mapped read with a command causes the AHB interface to hang, with the HREADY flag never set.

*Note: This failure does not occur in DDR octal mode.*

**Workaround**

Apply one of the following measures:

- Avoid reading the last byte of the memory space through memory-mapped access. Use indirect read instead.
- Set DEVSIZE value so that the memory space it defines exceeds the memory size, then handle the memory boundary by software.

### 2.7.14 Single-byte memory-mapped write to odd octal DDR address failing when a higher-priority event occurs

**Description**

In DDR octal mode, single-byte memory-mapped write request to odd memory address, followed by a higher-priority event such as a memory-mapped read, results in the OCTOSPI peripheral to spuriously DQS-mask both bytes transferred in the same data clock period, which causes the loss of the byte sent to memory.

**Workaround**

Apply one of the following measures:

- Avoid using single-byte memory-mapped writes.
- Use indirect write to send a single byte to a memory.

## 2.8 ADC

### 2.8.1 Injected queue of context is not available in case of JQM = 0

**Description**

The queue mechanism is not functional when JQM = 0. The effective queue length is equal to 1 stage: a new context written before the previous context's consumption leads to a queue overflow and is ignored. Consequently, the ADC must be stopped before programming the JSQR register.

**Workaround**

None.

### 2.8.2 Writing ADCx_JSQR when JADCSTART and JQDIS are set might lead to incorrect behavior

**Description**

Writing the ADCx_JSQR register when there is an on-going injected conversion (JADCSTART = 1) might lead to unpredictable ADC behavior if the queues of context are not enabled (JQDIS = 1).

**Workaround**

None.

### 2.8.3 Wrong ADC result if conversion done late after calibration or previous conversion

**Description**

The result of an ADC conversion done more than 1 ms later than the previous ADC conversion or ADC calibration might be incorrect.

**Workaround**

Perform two consecutive ADC conversions in single, scan or continuous mode. Reject the result of the first conversion and only keep the result of the second.

### 2.8.4 Spurious temperature measurement due to spike noise

**Description**

Depending on the MCU activity, internal interference may cause temperature-dependent spike noise on the temperature sensor output to the ADC, resulting in occasional spurious (outlying) temperature measurement.

**Workaround**

Perform a series of measurements and process the acquired data samples such as to obtain a mean value not affected by the outlying samples.

For this, it is recommended to use interquartile mean (IQM) algorithm with at least 64 samples. IQM is based on rejecting the quarters (quartiles) of sample population with the lowest and highest values and on computing the mean value only using the remaining (interquartile) samples.

The acquired sample values are first sorted from lowest to highest, then the sample sequence is truncated by removing the lowest and highest sample quartiles.

Example:

**Table 5. Measurement result after IQM post-processing**

| Data | Sample | | | | | | | | | | | | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | |
| Acquired | 17.2 | 10.92 | 9.56 | 2.12 | 9.82 | 10.72 | 10.6 | 3.5 | 9.46 | 9.78 | 9.5 | 1.1 | 8.69 |
| Sorted | 1.1 | 2.12 | 3.5 | 9.46 | 9.5 | 9.56 | 9.78 | 9.82 | 10.6 | 10.72 | 10.92 | 17.2 | 8.69 |
| Truncated | - | - | - | 9.46 | 9.5 | 9.56 | 9.78 | 9.82 | 10.6 | - | - | - | 9.79 |

The measurement result after the IQM post-processing in the example is 9.79. For consistent results, use a minimum of 64 samples. It is recommended to optimize the code performing the sort task such as to minimize its processing power requirements.

## 2.9 COMP

### 2.9.1 Comparator outputs cannot be configured in open-drain

#### Description

Comparator outputs are always forced in push-pull mode whatever the GPIO output type configuration bit value.

#### Workaround

None.

## 2.10 TSC

### 2.10.1 Inhibited acquisition in short transfer phase configuration

#### Description

Some revisions of the reference manual may omit the information that the following configurations of the TSC_CR register are forbidden:

- The PGPSC[2:0] bitfield set to 000 and the CTPL[3:0] bitfield to 0000 or 0001
- The PGPSC[2:0] bitfield set to 111 and the CTPL[3:0] bitfield to 0000

Failure to respect this restriction leads to an inhibition of the acquisition.

This is a documentation inaccuracy issue rather than a product limitation.

#### Workaround

No application workaround is required.

## 2.11 TIM

### 2.11.1 HSE/32 is not available for TIM16 input capture if RTC clock is disabled or other than HSE

#### Description

If the RTC clock is either disabled or other than HSE, the HSE/32 clock is not available for TIM16 input capture even if selected (bitfield TI1_RMP[2:0] = 101 in the TIM16_OR1 register).

#### Workaround

Apply the following procedure:

1. Enable the power controller clock (bit PWREN = 1 in the RCC_APB1ENR1 register).
2. Disable the backup domain write protection (bit DBP = 0 in the PWR_CR1 register).
3. Enable RTC clock and select HSE as clock source for RTC (bits RTCSEL[1:0] = 11 and bit RTCEN = 1 in the RCC_BDCR register).
4. Select the HSE/32 as input capture source for TIM16 (bitfield TI1_RMP[2:0] = 101 in the TIM16_OR1 register).

Alternatively, use TIM17 that implements the same features as TIM16, and is not affected by the limitation described.

## 2.12 LPTIM

### 2.12.1 MCU 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 MCU from Stop mode may be frozen in active state. Consequently, when trying to enter Stop mode, this limitation prevents the MCU from entering low-power mode and the firmware remains stuck in the LPTIM interrupt routine.

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

#### Workaround

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

### 2.12.2 LPTIM1 outputs cannot be configured as open-drain

#### Description

LPTIM1 outputs are set in push-pull mode regardless of the configuration of corresponding GPIO outputs.

#### Workaround

None.

## 2.13 RTC and TAMP

### 2.13.1 RTC interrupt can be masked by another RTC interrupt

#### Description

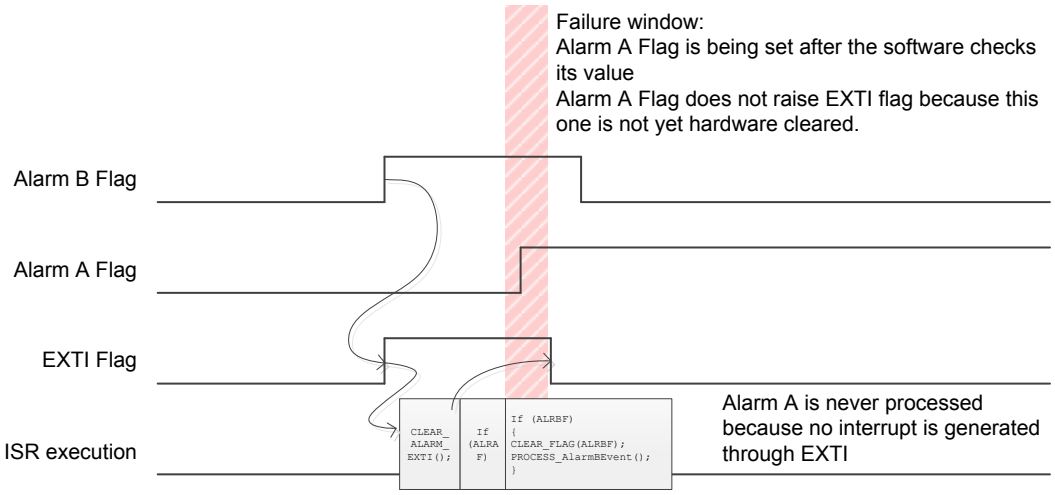
One RTC interrupt can mask another RTC interrupt if both share the same EXTI configurable line, such as the RTC Alarm A and Alarm B, of which the event flags are OR-de to the same EXTI line (refer to the **EXTI line connections** table in the **Extended interrupt and event controller (EXTI)** section of the reference manual).

The following code example and figure illustrate the failure mechanism: The Alarm A event is lost (fails to generate interrupt) as it occurs in the failure window, that is, after checking the Alarm A event flag but before the effective clear of the EXTI interrupt flag by hardware. The effective clear of the EXTI interrupt flag is delayed with respect to the software instruction to clear it.

Alarm interrupt service routine:

```
void RTC_Alarm_IRQHandler(void)
{
    CLEAR_ALARM_EXTI(); /* Clear the EXTI line flag for RTC alarms*/
    If(ALRAF) /* Check if Alarm A triggered ISR */
    {
        CLEAR_FLAG(ALRAF); /* Clear the Alarm A interrupt pending bit */
        PROCESS_AlarmAEvent(); /* Process Alarm A event */
    }
    If(ALRBF) /* Check if Alarm B triggered ISR */
    {
        CLEAR_FLAG(ALRBF); /* Clear the Alarm B interrupt pending bit */
        PROCESS_AlarmBEvent(); /* Process Alarm B event */
    }
}
```

**Figure 1. Masked RTC interrupt**




### Workaround

In the interrupt service routine, apply three consecutive event flag ckecks - source one, source two, and source one again, as in the following code example:

```
void RTC_Alarm_IRQHandler(void)
{
    CLEAR_ALARM_EXTI(); /* Clear the EXTI's line Flag for RTC Alarm */
    If(ALRAF) /* Check if AlarmA triggered ISR */
    {
        CLEAR_FLAG(ALRAF); /* Clear the AlarmA interrupt pending bit */
        PROCESS_AlarmAEvent(); /* Process AlarmA Event */
    }
    If(ALRBF) /* Check if AlarmB triggered ISR */
    {
        CLEAR_FLAG(ALRBF); /* Clear the AlarmB interrupt pending bit */
        PROCESS_AlarmBEvent(); /* Process AlarmB Event */
    }
    If(ALRAF) /* Check if AlarmA triggered ISR */
    {
```

```
        CLEAR_FLAG(ALRAF); /* Clear the AlarmA interrupt pending bit */
        PROCESS_AlarmAEvent(); /* Process AlarmA Event */
    }
}
```

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

**Description**

If the INIT bit of the RTC_ISR register is set between one and two RTCCLK cycles after being cleared, the INITF flag is set immediately instead of waiting for synchronization delay (which should be between one and two RTCCLK cycles), and the initialization of registers may fail.

Depending on the INIT bit clearing and setting instants versus the RTCCLK edges, it can happen that, after being immediately set, the INITF flag is cleared during one RTCCLK period then set again. As writes to calendar registers are ignored when INITF is low, a write during this critical period might result in the corruption of one or more calendar registers.

**Workaround**

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

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

### 2.13.3 RTC_REFIN and RTC_OUT on PB2 not operating in Stop 2 mode

**Description**

In Stop 2 low-power mode, the RTC_REFIN function does not operate and the RTC_OUT function does not operate if mapped on the PB2 pin.

**Workaround**

Apply one of the following measures:

- Use Stop 1 mode instead of Stop 2. This ensures the operation of both functions.
- Map RTC_OUT to the PC13 pin. This ensures the operation of the RTC_OUT function in either low-power mode. However, it has no effect to the RTC_REFIN function.

## 2.14 I2C

### 2.14.1 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave

**Description**

An $I^2C$-bus master generates STOP condition upon non-acknowledge of $I^2C$ address that it sends. This applies to 7-bit address as well as to each byte of 10-bit address.

When the MCU set as $I^2C$-bus master transmits a 10-bit address of which the first byte (5-bit header + 2 MSBs of the address + direction bit) is not acknowledged, the MCU duly generates STOP condition but it then cannot start any new $I^2C$-bus transfer. In this spurious state, the NACKF flag of the I2C_ISR register and the START bit of the I2C_CR2 register are both set, while the START bit should normally be cleared.

**Workaround**

In 10-bit-address master mode, if both NACKF flag and START bit get simultaneously set, proceed as follows:

1. Wait for the STOP condition detection (STOPF = 1 in I2C_ISR register).
2. Disable the I2C peripheral.
3. Wait for a minimum of three APB cycles.
4. Enable the I2C peripheral again.

### 2.14.2 Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C

#### Description

If the wakeup from Stop mode by I2C is disabled (WUPEN = 0), the correct use of the I2C peripheral is to disable it (PE = 0) before entering Stop mode, and re-enable it when back in Run mode.

Some reference manual revisions may omit this information.

Failure to respect the above while the MCU operating as slave or as master in multi-master topology enters Stop mode during a transfer ongoing on the $I^2C$-bus may lead to the following:

1. BUSY flag is wrongly set when the MCU exits Stop mode. This prevents from initiating a transfer in master mode, as the START condition cannot be sent when BUSY is set.
2. If clock stretching is enabled (NOSTRETCH = 0), the SCL line is pulled low by I2C and the transfer stalled as long as the MCU remains in Stop mode.

   The occurrence of such condition depends on the timing configuration, peripheral clock frequency, and $I^2C$-bus frequency.

This is a description inaccuracy issue rather than a product limitation.

#### Workaround

No application workaround is required.

### 2.14.3 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 MCU does not correctly sample the $I^2C$-bus SDA line when t 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.

### 2.14.4 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.

### 2.14.5 Last-received byte loss in reload mode

#### Description

If in master receiver mode or slave receive mode with SBC = 1 the following conditions are all met:

- I²C-bus stretching is enabled (NOSTRETCH = 0)
- RELOAD bit of the I2C_CR2 register is set
- NBYTES bitfield of the I2C_CR2 register is set to N greater than 1
- byte N is received on the I²C-bus, raising the TCR flag
- N - 1 byte is not yet read out from the data register at the instant TCR is raised,

then the SCL line is pulled low (I²C-bus clock stretching) and the transfer of the byte N from the shift register to the data register inhibited until the byte N-1 is read and NBYTES bitfield reloaded with a new value, the latter of which also clears the TCR flag. As a consequence, the software cannot get the byte N and use its content before setting the new value into the NBYTES field.

For I2C instances with independent clock, the last-received data is definitively lost (never transferred from the shift register to the data register) if the data N - 1 is read within four APB clock cycles preceding the receipt of the last data bit of byte N and thus the TCR flag raising. Refer to the product reference manual or datasheet for the I2C implementation table.

#### Workaround

- In master mode or in slave mode with SBC = 1, use the reload mode with NBYTES = 1.
- In master receiver mode, if the number of bytes to transfer is greater than 255, do not use the reload mode. Instead, split the transfer into sections not exceeding 255 bytes and separate them with repeated START conditions.
- Make sure, for example through the use of DMA, that the byte N - 1 is always read before the TCR flag is raised. Specifically for I2C instances with independent clock, make sure that it is always read earlier than four APB clock cycles before the receipt of the last data bit of byte N and thus the TCR flag raising.

The last workaround in the list must be evaluated carefully for each application as the timing depends on factors such as the bus speed, interrupt management, software processing latencies, and DMA channel priority.

### 2.14.6 Spurious master transfer upon own slave address match

#### Description

When the device is configured to operate at the same time as master and slave (in a multi- master I²C-bus application), a spurious master transfer may occur under the following condition:

- Another master on the bus is in process of sending the slave address of the device (the bus is busy).
- The device initiates a master transfer by bit set before the slave address match event (the ADDR flag set in the I2C_ISR register) occurs.
- After the ADDR flag is set:
  - the device does not write I2C_CR2 before clearing the ADDR flag, or
  - the device writes I2C_CR2 earlier than three I2C kernel clock cycles before clearing the ADDR flag

In these circumstances, even though the START bit is automatically cleared by the circuitry handling the ADDR flag, the device spuriously proceeds to the master transfer as soon as the bus becomes free. The transfer configuration depends on the content of the I2C_CR2 register when the master transfer starts. Moreover, if the I2C_CR2 is written less than three kernel clocks before the ADDR flag is cleared, the I2C peripheral may fall into an unpredictable state.

#### Workaround

Upon the address match event (ADDR flag set), apply the following sequence.

Normal mode (SBC = 0):

1. Set the ADDRCF bit.
2. Before Stop condition occurs on the bus, write I2C_CR2 with the START bit low.

Slave byte control mode (SBC = 1):

1. Write I2C_CR2 with the slave transfer configuration and the START bit low.
2. Wait for longer than three I2C kernel clock cycles.
3. Set the ADDRCF bit.
4. Before Stop condition occurs on the bus, write I2C_CR2 again with its current value.

The time for the software application to write the I2C_CR2 register before the Stop condition is limited, as the clock stretching (if enabled), is aborted when clearing the ADDR flag.

Polling the BUSY flag before requesting the master transfer is not a reliable workaround as the bus may become busy between the BUSY flag check and the write into the I2C_CR2 register with the START bit set.

### 2.14.7 START bit is cleared upon setting ADDRCF, not upon address match

#### Description

Some reference manual revisions may state that the START bit of the I2C_CR2 register is cleared upon slave address match event.

Instead, the START bit is cleared upon setting, by software, the ADDRCF bit of the I2C_ICR register, which does not guarantee the abort of master transfer request when the device is being addressed as slave. This product limitation and its workaround are the subject of a separate erratum.

#### Workaround

No application workaround is required for this description inaccuracy issue.

## 2.15 USART

### 2.15.1 nRTS is active while RE = 0 or UE = 0

#### Description

The nRTS line is driven low as soon as RTSE bit is set, even if the USART is disabled (UE = 0) or the receiver is disabled (RE = 0), that is, not ready to receive data.

#### Workaround

Upon setting the UE and RE bits, configure the I/O used for nRTS into alternate function.

### 2.15.2 UDR flag set while the SPI slave transmitter is disabled

#### Description

When the USART is used in SPI slave receive mode, the underrun flag (UDR bit of USART_ISR register) might be set even if the SPI slave transmitter is disabled (TE bit cleared in USART_CR1 register).

#### Workaround

Apply one of the following measures:

- Ignore the UDR flag when the SPI slave transmitter is disabled.
- Clear the UDR flag every time it is set, even if the SPI slave transmitter is disabled.
- Write dummy data in the USART_TDR register to avoid setting the UDR flag.

## 2.16 LPUART

### 2.16.1 LPUART1 outputs cannot be configured as open-drain

#### Description

LPUART1 outputs are set in push-pull mode regardless of the configuration of corresponding GPIO outputs.

**Workaround**

None.

## 2.17 SPI

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

**Description**

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

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

**Workaround**

When the SPI operates in:

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

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

**Description**

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

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

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

**Workaround**

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

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

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

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

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

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

### 2.17.3 CRC error in SPI slave mode if internal NSS changes before CRC transfer

#### Description

Some reference manual revisions may omit the information that the device operating as SPI slave must be configured in software NSS control if the SPI master pulses the NSS (for (for example in NSS pulse mode).

Otherwise, the transition of the internal NSS signal after the CRCNEXT flag is set might result in wrong CRC value computed by the device and, as a consequence, in a CRC error. As a consequence, the NSS pulse mode cannot be used along with the CRC function.

This is a documentation error rather than a product limitation.

#### Workaround

No application workaround is required as long as the device operating as SPI slave is duly configured in software NSS control.

## 2.18 bxCAN

### 2.18.1 bxCAN time-triggered communication mode not supported

#### Description

The time-triggered communication mode described in the reference manual is not supported. As a result, timestamp values are not available. The TTCM of the CAN_MCR register must be kept cleared (time-triggered communication mode disabled).

#### Workaround

None.

## 2.19 OTG_FS

### 2.19.1 Data FIFO gets corrupted if the write sequence to the transmit FIFO is interleaved with other OTGFS register access

#### Description

When the OTG full-speed cell is in Host or Device mode, interrupting the write sequence in the transmit FIFO by any access (read or write) to OTG registers leads to corruption of the next data written to the transmit FIFO.

#### Workaround

Ensure that the transmit FIFO write accesses cannot be interrupted by a procedure performing accesses to the USB cell registers.

# Revision history

Table 6. **Document revision history**

| Date | Version | Changes |
|---|---|---|
| 10-Oct-2017 | 1 | Initial release. |
| 30-Mar-2018 | 2 | Updated:<br>• Section 2.1: Core<br>• Section 2.12.1: HSE/32 is not available for TIM16 input capture if RTC clock is disabled or other than HSE<br>• Section 2.13.1: LPTIM1 outputs cannot be configured as open-drain<br>• Section 2.15.1: 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave<br>• Section 2.15.6: Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C<br>• Section 2.15.2: Wrong data sampling when data setup time (tSU;DAT) is shorter than one I2C kernel clock period<br>• Section 2.15.3: Spurious bus error detection in master mode<br>• Section 2.15.4: Last-received byte loss in reload mode<br>• Section 2.17.1: LPUART1 outputs cannot be configured as opendrain<br>• Section 2.18.1: BSY bit may stay high at the end of a data transfer in slave mode<br>• Section 2.18.2: CRC error in SPI slave mode if internal NSS changes before CRC transfer<br>Added:<br>• Section 2.2.3: Flash memory might not be accessible when AHB prescaler is greater than eight<br>• Section 2.2.4: Flash OPTVERR flag is always set after system reset<br>• Section 2.2.5: HSE oscillator long startup at low voltage<br>• Section 2.7.9: Octal indirect read operation is not started when OctoSPI is configured with address phase disabled<br>• Section 2.9.1: Spurious temperature measurement due to spike noise<br>• Section 2.14.1: RTC_REFIN and RTC_OUT on PB2 not operating in Stop 2 mode<br>• Section 2.18.3: BSY bit may stay high when SPI is disabled.<br>Deleted:<br>• START bit is not cleared when the address is not acknowledged by the slave device limitation from Section 2.15: I2C |

| Date | Version | Changes |
|---|---|---|
| 21-Jun-2018 | 3 | Updated:<br>• Section 2.15.6: Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C: section content was updated and this errata was reclassified as documentation errata<br>Added:<br>• Documentation errata to the document, summarized in Table 4: Summary of documentation errata and introduced in the cover page.<br>• Section 2.3.2: Spurious Firewall reset is generated when accessing FMC or OctoSPI<br>• Section 2.4.1: DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear<br>• Section 2.4.2: Byte and half-word accesses not supported<br>• Section 2.5.1: DMAMUX_RGCFR register is write-only, not readwrite<br>• Section 2.5.2: Request trigger overrun misdetection<br>• Section 2.5.3: SOFx not asserted when writing into DMAMUX_CFR register<br>• Section 2.5.4: DMA request counter not kept at GNBREQ bitfield value as long as the corresponding request channel is disabled<br>• Section 2.5.5: Synchronization event discarded if selected input DMA request is not active<br>• Section 2.7.10: Memory-mapped read of the last byte of the memory is not possible when in octal SDR mode<br>• Section 2.7.11: A memory-mapped write request with only one-byte length at odd-start address finished by any event is masked<br>• Section 2.8.3: Writing the register ADCx_JSQR when JADCSTART=1 and JQDIS=1 might lead to incorrect behavior<br>• Section 2.15.5: Spurious master transfer upon own slave address match<br>• Section 2.15.7: START bit is cleared upon setting ADDRCF, not upon address match |

| Date | Version | Changes |
|---|---|---|
| 19-Dec-2018 | 4 | Updated:<br>• Section 2.5.5 DMAMUX_RGCFR register is write-only, not read-write moved to documentation errata section<br>• Section 2.5.1 SOFx not asserted when writing into DMAMUX_CFR register moved to device limitations section<br>• Section 2.7.1 CSBOUND setting not fully respected updated workaround qualifier<br>• Section 2.7.2 Auto-polling mode not functional with new octal memories updated workaround qualifier<br>• Section 2.7.7 No transfer error interrupt upon indirect write to address exceeding the limit updated workaround qualifier<br>• Section 2.8.2 Writing ADCx_JSQR when JADCSTART and JQDIS are set might lead to incorrect behavior updated workaround<br>• Section 2.8.4 Spurious temperature measurement due to spike noise updated function section from TEMP to ADC<br>• Section 2.10.1 Inhibited acquisition in short transfer phase configuration moved to documentation errata section<br>• Section 2.14.2 Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C moved to documentation errata<br>• Section 2.14.3 Wrong data sampling when data setup time (tSU;DAT) is shorter than one I2C kernel clock period updated workaround qualifier<br>• Section 2.17.3 CRC error in SPI slave mode if internal NSS changes before CRC transfer moved to documentation errata<br>Added:<br>• Section 2.2.6 SDMMC1SMEN bit of RCC_AHB2SMENR register only modifiable with word access<br>• Section 2.2.7 First double-word of Flash memory corrupted upon reset or power down while programming<br>• Section 2.2.8 Unstable LSI when it clocks RTC or CSS on LSE<br>• Section 2.2.9 LTDC and DSI not functional with Stop 2<br>• Section 2.2.10 Regulator startup failure at low VDD<br>• Section 2.2.11 1-Mbyte devices wrongly configured as 2-Mbyte<br>• Section 2.5.3 OFx not asserted when writing into DMAMUX_RGCFR register<br>• Section 2.5.4 Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event<br>• Section 2.7.10 Unaligned AHB write requests not supported<br>• Section 2.7.12 Data masking for odd byte writes only working with D1/D0 ordering<br>• Section 2.7.14 Single-byte memory-mapped write to odd octal DDR address failing when a higher-priority event occurs<br>• Section 2.13.2 Calendar initialization may fail in case of consecutive INIT mode entry |

# Contents

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

2 **Description of device errata** . . . . . 5

2.1 Core . . . . . 5

2.1.1 Interrupted loads to SP can cause erroneous behavior . . . . . 5

2.2 System . . . . . 5

2.2.1 Full JTAG configuration without NJTRST pin cannot be used . . . . . 5

2.2.2 Data cache might be corrupted during Flash memory read-while-write operation . . . . . 5

2.2.3 Flash memory might not be accessible when AHB prescaler is greater than eight . . . . . 6

2.2.4 Flash OPTVERR flag is always set after system reset . . . . . 6

2.2.5 HSE oscillator long startup at low voltage . . . . . 6

2.2.6 SDMMC1SMEN bit of RCC_AHB2SMENR register only modifiable with word access . . . . 7

2.2.7 First double-word of Flash memory corrupted upon reset or power down while programming . . . . . 7

2.2.8 Unstable LSI when it clocks RTC or CSS on LSE . . . . . 7

2.2.9 LTDC and DSI not functional with Stop 2 . . . . . 8

2.2.10 Regulator startup failure at low $V_{DD}$ . . . . . 8

2.2.11 1-Mbyte devices wrongly configured as 2-Mbyte . . . . . 8

2.3 FW . . . . . 8

2.3.1 Firewall protection size limitation . . . . . 8

2.3.2 Spurious Firewall reset is generated when accessing FMC or OCTOSPI . . . . . 9

2.4 DMA . . . . . 9

2.4.1 DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear . . . . . 9

2.5 DMAMUX . . . . . 9

2.5.1 SOFx not asserted when writing into DMAMUX_CFR register . . . . . 9

2.5.2 OFx not asserted for trigger event coinciding with last DMAMUX request . . . . . 9

2.5.3 OFx not asserted when writing into DMAMUX_RGCFR register . . . . . 10

2.5.4 Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event . . . . . 10

2.5.5 DMAMUX_RGCFR register is write-only, not read-write . . . . . 10

2.5.6 DMA request counter not kept at GNBREQ bitfield value as long as the corresponding request channel is disabled . . . . . 11

2.5.7 Synchronization event discarded if selected input DMA request is not active . . . 11
2.6 FMC . . . 11
2.6.1 Dummy read cycles inserted when reading synchronous memories . . . 11
2.6.2 Wrong data read from a busy NAND memory . . . 11
2.7 OCTOSPI . . . 12
2.7.1 CSBOUND setting not fully respected . . . 12
2.7.2 Auto-polling mode not functional with new octal memories . . . 12
2.7.3 Command phase must be octal for octal transfers . . . 12
2.7.4 Write data lost with Clock mode 3 . . . 12
2.7.5 Deadlock upon disabling OCTOSPI during memory-mapped write . . . 12
2.7.6 Deadlock in dual-flash configuration with odd number of bytes . . . 13
2.7.7 No transfer error interrupt upon indirect write to address exceeding the limit . . . 13
2.7.8 DHQC not effective if DDTR not set . . . 13
2.7.9 Indirect read and auto-polling transfers without address phase not starting . . . 13
2.7.10 Unaligned AHB write requests not supported . . . 14
2.7.11 Data mask failing with odd start address writes to non-HyperBus memory . . . 14
2.7.12 Data masking for odd byte writes only working with D1/D0 ordering . . . 14
2.7.13 Memory-mapped read of the last memory space byte not possible in SDR octal mode . . . 14
2.7.14 Single-byte memory-mapped write to odd octal DDR address failing when a higher-priority event occurs . . . 14
2.8 ADC . . . 15
2.8.1 Injected queue of context is not available in case of JQM = 0 . . . 15
2.8.2 Writing ADCx_JSQR when JADCSTART and JQDIS are set might lead to incorrect behavior . . . 15
2.8.3 Wrong ADC result if conversion done late after calibration or previous conversion . . . 15
2.8.4 Spurious temperature measurement due to spike noise . . . 15
2.9 COMP . . . 16
2.9.1 Comparator outputs cannot be configured in open-drain . . . 16
2.10 TSC . . . 16
2.10.1 Inhibited acquisition in short transfer phase configuration . . . 16
2.11 TIM . . . 16
2.11.1 HSE/32 is not available for TIM16 input capture if RTC clock is disabled or other than HSE . . . 16

2.12 LPTIM . . . 17
2.12.1 MCU may remain stuck in LPTIM interrupt when entering Stop mode . . . 17
2.12.2 LPTIM1 outputs cannot be configured as open-drain . . . 17
2.13 RTC and TAMP . . . 17
2.13.1 RTC interrupt can be masked by another RTC interrupt . . . 17
2.13.2 Calendar initialization may fail in case of consecutive INIT mode entry . . . 19
2.13.3 RTC_REFIN and RTC_OUT on PB2 not operating in Stop 2 mode . . . 19
2.14 I2C . . . 19
2.14.1 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave . . . 19
2.14.2 Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C . . . 20
2.14.3 Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period . . . 20
2.14.4 Spurious bus error detection in master mode . . . 20
2.14.5 Last-received byte loss in reload mode . . . 20
2.14.6 Spurious master transfer upon own slave address match . . . 21
2.14.7 START bit is cleared upon setting ADDRCF, not upon address match . . . 22
2.15 USART . . . 22
2.15.1 nRTS is active while RE = 0 or UE = 0 . . . 22
2.15.2 UDR flag set while the SPI slave transmitter is disabled . . . 22
2.16 LPUART . . . 22
2.16.1 LPUART1 outputs cannot be configured as open-drain . . . 22
2.17 SPI . . . 23
2.17.1 BSY bit may stay high when SPI is disabled . . . 23
2.17.2 BSY bit may stay high at the end of data transfer in slave mode . . . 23
2.17.3 CRC error in SPI slave mode if internal NSS changes before CRC transfer . . . 23
2.18 bxCAN . . . 24
2.18.1 bxCAN time-triggered communication mode not supported . . . 24
2.19 OTG_FS . . . 24
2.19.1 Data FIFO gets corrupted if the write sequence to the transmit FIFO is interleaved with other OTGFS register access . . . 24
**Revision history . . . 25**