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

<table>
<thead>
<tr>
<th>Reference</th>
<th>Part numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32L4Rxxx</td>
<td>STM32L4R5AG, STM32L4R5AI, STM32L4R5QG, STM32L4R5QI, STM32L4R5VG, STM32L4R5VI, STM32L4R5ZG, STM32L4R5ZI, STM32L4R7AI, STM32L4R7VI, STM32L4R7ZI, STM32L4R9AG, STM32L4R9AI, STM32L4R9VG, STM32L4R9VI, STM32L4R9ZG, STM32L4R9ZI</td>
</tr>
<tr>
<td>STM32L4Sxxx</td>
<td>STM32L4S5AI, STM32L4S5QI, STM32L4S5VI, STM32L4S5ZI, STM32L4S7AI, STM32L4S7VI, STM32L4S7ZI, STM32L4S9AI, STM32L4S9VI, STM32L4S9ZI</td>
</tr>
</tbody>
</table>

Table 2. Device variants

<table>
<thead>
<tr>
<th>Reference</th>
<th>Silicon revision codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32L4Rxxx STM32L4Sxxx</td>
<td>Device marking(1)</td>
</tr>
<tr>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>W</td>
</tr>
</tbody>
</table>

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** = 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>
<thead>
<tr>
<th>Function</th>
<th>Section</th>
<th>Limitation</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>2.1.1</td>
<td>Interrupted loads to SP can cause erroneous behavior</td>
<td>A A</td>
</tr>
<tr>
<td>System</td>
<td>2.2.1</td>
<td>Full JTAG configuration without NJTRST pin cannot be used</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.2.2</td>
<td>Data cache might be corrupted during Flash memory read-while-write operation</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.2.3</td>
<td>Flash memory might not be accessible when AHB prescaler is greater than two</td>
<td>N N</td>
</tr>
<tr>
<td></td>
<td>2.2.4</td>
<td>Flash OPTVERR flag is always set after system reset</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.2.5</td>
<td>HSE oscillator long startup at low voltage</td>
<td>P P</td>
</tr>
<tr>
<td></td>
<td>2.2.6</td>
<td>SDMMC1SMEN bit of RCC_AHB2SMENR register only modifiable with word access</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.2.7</td>
<td>Unstable LSI when it clocks RTC or CSS on LSE</td>
<td>P P</td>
</tr>
<tr>
<td></td>
<td>2.2.8</td>
<td>LTDC and DSI not functional with Stop 2</td>
<td>P -</td>
</tr>
<tr>
<td></td>
<td>2.2.9</td>
<td>Regulator startup failure at low VDD</td>
<td>N -</td>
</tr>
<tr>
<td></td>
<td>2.2.10</td>
<td>1-Mbyte devices wrongly configured as 2-Mbyte</td>
<td>N -</td>
</tr>
<tr>
<td></td>
<td>2.2.11</td>
<td>OBL_LAUNCH operation may corrupt SRAM content</td>
<td>P P</td>
</tr>
<tr>
<td></td>
<td>2.2.13</td>
<td>Bootloader not functional on 1-Mbyte devices</td>
<td>N -</td>
</tr>
<tr>
<td>FW</td>
<td>2.3.1</td>
<td>Firewall protection size limitation</td>
<td>N N</td>
</tr>
<tr>
<td></td>
<td>2.3.2</td>
<td>Spurious Firewall reset</td>
<td>P P</td>
</tr>
<tr>
<td>DMA</td>
<td>2.4.1</td>
<td>DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear</td>
<td>A A</td>
</tr>
<tr>
<td>DMAMUX</td>
<td>2.5.1</td>
<td>SOFx not asserted when writing into DMAMUX_CFR register</td>
<td>N N</td>
</tr>
<tr>
<td></td>
<td>2.5.2</td>
<td>OFx not asserted for trigger event coinciding with last DMAMUX request</td>
<td>N N</td>
</tr>
<tr>
<td></td>
<td>2.5.3</td>
<td>OFx not asserted when writing into DMAMUX_RGCFR register</td>
<td>N N</td>
</tr>
<tr>
<td></td>
<td>2.5.4</td>
<td>Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event</td>
<td>A A</td>
</tr>
<tr>
<td>FMC</td>
<td>2.6.1</td>
<td>Dummy read cycles inserted when reading synchronous memories</td>
<td>N N</td>
</tr>
<tr>
<td></td>
<td>2.6.2</td>
<td>Wrong data read from a busy NAND memory</td>
<td>A A</td>
</tr>
<tr>
<td>Function</td>
<td>Section</td>
<td>Limitation</td>
<td>Status</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>------------</td>
<td>--------</td>
</tr>
<tr>
<td>OCTOSPI</td>
<td>2.7.1</td>
<td>CSBOUND shall not be used</td>
<td>N N</td>
</tr>
<tr>
<td></td>
<td>2.7.2</td>
<td>Auto-polling mode not functional with new octal memories</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.7.3</td>
<td>Command phase must be octal for octal transfers</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.7.4</td>
<td>Write data lost with Clock mode 3</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.7.5</td>
<td>Deadlock upon disabling OCTOSPI during memory-mapped write</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.7.6</td>
<td>Deadlock in dual-flash configuration with odd number of bytes</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.7.7</td>
<td>No transfer error interrupt upon indirect write to address exceeding the limit</td>
<td>N N</td>
</tr>
<tr>
<td></td>
<td>2.7.8</td>
<td>DHQC not effective if DDTR not set</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.7.9</td>
<td>Indirect read and auto-polling transfers without address phase not starting</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.7.10</td>
<td>Unaligned AHB write requests not supported</td>
<td>N N</td>
</tr>
<tr>
<td></td>
<td>2.7.11</td>
<td>Write operation in octal-DTR mode starting at odd address is not supported</td>
<td>N N</td>
</tr>
<tr>
<td></td>
<td>2.7.12</td>
<td>Data masking for odd number of byte writes only working with D1/D0 ordering</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.7.13</td>
<td>Memory-mapped read of the last memory space byte not possible in SDR octal mode</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.7.14</td>
<td>Single-byte memory-mapped write to odd octal DDR address failing when a higher-priority event occurs</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.7.15</td>
<td>Memory-mapped write data to odd start address corrupted when BUSY=0 and not in octal-DTR mode</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.7.16</td>
<td>Data not sampled correctly on reads without DQS and with less than two cycles before the data phase</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.7.17</td>
<td>Memory-mapped write error response when DQS output is disabled</td>
<td>P P</td>
</tr>
<tr>
<td></td>
<td>2.7.18</td>
<td>Byte possibly dropped during an SDR read in clock mode 3 when a transfer gets automatically split</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.7.19</td>
<td>Single, dual and quad modes not functional with DQS input enabled</td>
<td>N N</td>
</tr>
<tr>
<td></td>
<td>2.7.20</td>
<td>Additional bytes read in indirect mode with DQS input enabled when data length is too short</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.7.21</td>
<td>DQS output enabled too late on write</td>
<td>A A</td>
</tr>
<tr>
<td>ADC</td>
<td>2.8.1</td>
<td>Injected queue of context is not available if JQM = 0</td>
<td>N N</td>
</tr>
<tr>
<td></td>
<td>2.8.4</td>
<td>Wrong ADC result if conversion done late after calibration or previous conversion</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.8.5</td>
<td>Spurious temperature measurement due to spike noise</td>
<td>A -</td>
</tr>
<tr>
<td>DAC</td>
<td>2.9.1</td>
<td>Invalid DAC channel analog output if the DAC channel MODE bitfield is programmed before DAC initialization</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>2.9.2</td>
<td>DMA underrun flag not set when an internal trigger is detected on the clock cycle of the DMA request acknowledge</td>
<td>N N</td>
</tr>
<tr>
<td>COMP</td>
<td>2.10.1</td>
<td>Comparator outputs cannot be configured in open-drain</td>
<td>N N</td>
</tr>
<tr>
<td>DSI</td>
<td>2.11.1</td>
<td>Failing DSI read operation</td>
<td>A A</td>
</tr>
<tr>
<td>TIM</td>
<td>2.13.1</td>
<td>HSE/32 is not available for TIM16 input capture if RTC clock is disabled or other than HSE</td>
<td>A A</td>
</tr>
</tbody>
</table>
The following table gives a quick reference to the documentation errata.

### Table 4. Summary of device documentation errata

<table>
<thead>
<tr>
<th>Function</th>
<th>Section</th>
<th>Documentation erratum</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>2.2.12</td>
<td>Irrelevant BOOT0 electrical characteristics</td>
</tr>
<tr>
<td>DMAMUX</td>
<td>2.5.5</td>
<td>DMAMUX_RGCFR register is write-only, not read-write</td>
</tr>
<tr>
<td></td>
<td>2.5.6</td>
<td>DMA request counter not kept at GNBREQ bitfield value as long as the corresponding request channel is disabled</td>
</tr>
<tr>
<td></td>
<td>2.5.7</td>
<td>Synchronization event discarded if selected input DMA request is not active</td>
</tr>
<tr>
<td>OCTOSPI</td>
<td>2.7.22</td>
<td>PSRAM not supported</td>
</tr>
<tr>
<td>ADC</td>
<td>2.8.2</td>
<td>Writing ADC_JSQR when JADCSTART and JQDIS are set may lead to incorrect behavior</td>
</tr>
<tr>
<td></td>
<td>2.8.3</td>
<td>ADEN bit cannot be set immediately after the ADC calibration is done</td>
</tr>
<tr>
<td>TSC</td>
<td>2.12.1</td>
<td>Inhibited acquisition in short transfer phase configuration</td>
</tr>
<tr>
<td>Function</td>
<td>Section</td>
<td>Documentation erratum</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>I2C</td>
<td>2.16.2</td>
<td>Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C</td>
</tr>
<tr>
<td></td>
<td>2.16.7</td>
<td>START bit is cleared upon setting ADDRCF, not upon address match</td>
</tr>
<tr>
<td>SPI</td>
<td>2.19.3</td>
<td>CRC error in SPI slave mode if internal NSS changes before CRC transfer</td>
</tr>
</tbody>
</table>
2 Description of device errata

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

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

2.1 Core

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:
```c
/* 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 two

Description
When the AHB prescaler (the HPRE[3:0] bitfield of the RCC_CFGR register) 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 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 \( V_{DD}(\text{min}) \) 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.

### 2.2.8 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.
2.2.9 Regulator startup failure at low V\textsubscript{DD}

**Description**
Depending on V\textsubscript{DD} rising speed, the internal regulator might not start correctly with V\textsubscript{DD} below 2.9 V.

**Workaround**
None.

2.2.10 1-Mbyte devices wrongly configured as 2-Mbyte

**Description**
STM32L4RxG/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.

The problem is present on STM32L4RxG/SxxG devices with date code week 48 2018 or earlier.
The STM32L4RxG/SxxG 1-Mbyte part number devices with date code week 49 2018 or later have the correct 1-Mbyte configuration.

**Workaround**
None.

2.2.11 OBL\_LAUNCH operation may corrupt SRAM content

**Description**
OBL\_LAUNCH operation may corrupt SRAM1 and SRAM3 content. SRAM2 is safe.

**Workaround**
Use SRAM2 to store data that must be preserved after OBL\_LAUNCH operation.

2.2.12 Irrelevant BOOT0 electrical characteristics

**Description**
Some data sheet revisions may unduly specify electrical characteristics of BOOT0 pin. However, as BOOT0 signal is shared with PH3 GPIO on the PH3-BOOT0 pin, the GPIO electrical characteristics apply and the BOOT0-specific electrical characteristics are irrelevant.

**Workaround**
No application workaround is required.

2.2.13 Bootloader not functional on 1-Mbyte devices

**Description**
The bootloader might not work on STM32L4RxG/SxxG 1-Mbyte part number devices with date code week 04 2020 or earlier. This depends on the device bootloader version (V9.2 is not functional). In order to check whether the device has the fixed bootloader version or not, perform a read at address 0x1FFF 6FFE:
- If the read value is 0x92 then the version is V9.2 and the device has the non-functional bootloader.
- If the read value is 0x95 then the version is V9.5 and the device has the fixed bootloader.

**Note:**
All STM32L4RxG/SxxG 1-Mbyte and 2-Mbytes part number devices with date code week W05 2020 or later have the new bootloader V9.5.
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

Description
After configuring Firewall to protect an area within SRAM1, followed by an SRAM1 access, any access to an address outside the protected SRAM1 area, of which the 18 LSBs correspond to an address within the SRAM1 protected area, unduly triggers Firewall reset.

Workaround
Use SRAM2 or SRAM3 as system RAM and avoid accessing SRAM1 before any access to an address outside the protected SRAM1 area, of which the 18 LSBs correspond to an address within the protected SRAM1 area.

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 shall not be used

Description
The CS boundary function is expected to split transfers just before the byte with the address \(2^{\text{CSBOUND}}\), where CSBOUND is the value of CSBOUND[4:0] bitfield of the OCTOSPI_DCR3 register. This feature is not working properly and shall not be used. The CSBOUND[4:0] bitfield of the OCTOSPI_DCR3 register shall be left at its reset default value (0xb00000).

Workaround
None.

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 Write operation in octal-DTR mode starting at odd address is not supported

Description
When the memory type is 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).

The same failure occurs for all types of memories, including HyperBus™, if a write starting with odd address is requested and the OCTOSPI is busy.

Workaround
None.

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

Description
The data masking for odd number of 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.

2.7.15 Memory-mapped write data to odd start address corrupted when BUSY=0 and not in octal-DTR mode

Description
In all modes except for octal-DTR mode, if the first memory-mapped write operation that it is requested while BUSY=0 is a write to an odd address, the first byte is written correctly, but then this same byte is written to the next address and all subsequent bytes are shifted up one address.

Workaround
Keep the BUSY flag set by using the following method:
1. Disable the timeout counter.
2. Avoid to abort the application.
3. As soon as the OCTOSPI is configured and ready to be used in Memory-mapped mode, perform a dummy read.
2.7.16 Data not sampled correctly on reads without DQS and with less than two cycles before the data phase

Description
A command is composed of five phases:

- Command
- Address
- Alternate byte
- Dummy (latency) cycles
- Data

Data are not sampled correctly if all the following conditions are true:

- Fewer than two cycles are required by the first four phases (command, address, alternate or dummy)
- DQS is disabled (DQSE=0)
- Data phase is enabled
- Data are read in Indirect or Memory-mapped mode

Workaround
Ensure that there are at least two cycles before the data phase using one of the following methods:

- Send one byte of address in quad-SDR mode (ADMODE=011, ADSIZE=00, ADDDTR=0)
- Send two bytes of address in octal-SDR mode (ADMODE=100, ADSIZE=01, ADDDTR=0)
- Send four bytes of address in octal-DTR mode (ADMODE=100, ADSIZE=11, ADDDTR=1)
- Send two bytes of instruction in quad-DTR mode (IMODE=011, ISIZE=01, IDDTR=1)
- Send one instruction byte in octal followed by one dummy cycle.
- Send one instruction byte in octal followed by one alternate byte in octal.

2.7.17 Memory-mapped write error response when DQS output is disabled

Description
If the DQSE control bit in OCTOSPI_WCCR is set to 0 for memories without DQS pin, it results in an error response for every memory-mapped write request.

Workaround
When doing memory-mapped writes, the DQSE bit in OCTOSPI_WCCR must be set to 1 even for memories which have no DQS pin.

Limitation of this workaround: if the DQS output is asserted on memory-mapped writes while the AXI bus transfer has some byte-enable bits deasserted, the bytes which should be masked get written to the memory.

2.7.18 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, OCTOSPI can interrupt the transfer and automatically restart it at the next address when the CSBOUND, REFRESH, TIMEOUT or MAX-TRAN features are employed. Thus, a single continuous transfer can effectively be split into multiple smaller transfers.

When OCTOSPI is configured to use clock mode 3 (CKMODE bit in OCTOSPI_DCR1 is set to 1) and a continuous stream of data is read in SDR mode (CKMODE bit in OCTOSPI_DCR1 is set to 0), the last byte sent by the memory before an automatic split might get dropped, thus causing all the subsequent bytes to be seen one address earlier.

Workaround
Use clock mode 0 (CKMODE bit in OCTOSPI_DCR1 is set to 0) when in SDR mode.
2.7.19 Single, dual and quad modes not functional with DQS input enabled

Description
Data read from memory in single, dual or quad mode with the DQS input enabled (DQSE control bit in OCTOSPI_CCR is set to 1) can be corrupted. Only octal-data mode (DMODE bit in OCTOSPI_CCR is set to 100) is functional with the DQS input enabled.

Workaround
None.

2.7.20 Additional bytes read in indirect mode with DQS input enabled when data length is too short

Description
Extra bytes reception may appear when below two conditions are met at the same time:
• Data read in indirect-read mode with DQS enabled (DQSE bit in OCTOSPI_CCR set to 1)
• The number of cycles for data read phase is less than the sum of the number of cycles required for (command + address + alternate-byte + dummy) phases.

Workaround
• Avoid programming transfers with data phase shorter than (command + address + alternate-byte + dummy) phases
• Perform an abort just after reading all the data required bytes from OCTOSPI_DR register.

2.7.21 DQS output enabled too late on write

Description
At high kernel clock frequency, a hold violation on the DQS signal may occur upon sending a new write command to memory device, which possibly causes data transfer failure in two types of situations:
• DQS signal, high before starting the data phase, falls at the start of transfer. Because of the hold violation, DQS output is sampled high with the first output clock edge, which results in the memory device ignoring the first byte sent.
• DQS signal, low before starting the data phase, rises at the start of transfer. Because of the hold violation, DQS output is sampled low with the first output clock edge, which results in its wrong interpretation by the memory device.

Workaround
Reduce the kernel clock frequency. Adapt the output clock division factor accordingly so as to keep the output clock frequency unchanged.

2.7.22 PSRAM not supported

Description
Some reference manual and data sheet revisions unduly mention PSRAM while this type of memory is not supported by the device.
This is a documentation issue rather than a product limitation.

Workaround
Not relevant.

2.8 ADC
2.8.1 Injected queue of context is not available if JQM = 0

Description
The queue mechanism is not functional when JQM bit of ADC_CFGR is cleared to 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 ADC_JSQR register.

Workaround
None.

2.8.2 Writing ADC_JSQR when JADCSTART and JQDIS are set may lead to incorrect behavior

Description
Some reference manual revisions specify that the ADC_JSQR register can be written when an injected conversion is ongoing (JADCSTART = 1). This may lead to unpredictable ADC behavior if the queues of context are not enabled (JQDIS = 1).

Workaround
No application workaround is required for this description inaccuracy issue.

2.8.3 ADEN bit cannot be set immediately after the ADC calibration is done

Description
Some reference manual revisions may not indicate that the ADEN bit cannot be set while the ADCAL bit is set and during a four ADC clock cycles after the ADCAL bit is cleared by hardware (end of the calibration). Otherwise, if the ADEN bit is set during this four ADC clock cycle period, it will be reset by the calibration logic and the ADC will stay disabled. This is due to the fact that there is an internal reset of the ADEN bit four ADC clock cycles after the ADCAL bit is cleared by hardware.

Workaround
No application workaround is required for this description inaccuracy issue.

2.8.4 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.

2.8.5 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>
<thead>
<tr>
<th>Data</th>
<th>Sample</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquired</td>
<td>17.2</td>
<td>10.92</td>
</tr>
<tr>
<td>Sorted</td>
<td>1.1</td>
<td>2.12</td>
</tr>
<tr>
<td>Truncated</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

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 DAC

2.9.1 Invalid DAC channel analog output if the DAC channel MODE bitfield is programmed before DAC initialization

Description

When the DAC operates in Normal mode and the DAC enable bit is cleared, writing a value different from 000 to the DAC channel MODE bitfield of the DAC_MCR register before performing data initialization causes the corresponding DAC channel analog output to be invalid.

Workaround

Apply the following sequence:
1. Perform one write access to any data register.
2. Program the MODE bitfield of the DAC_MCR register.

2.9.2 DMA underrun flag not set when an internal trigger is detected on the clock cycle of the DMA request acknowledge

Description

When the DAC channel operates in DMA mode (DMAEN of DAC_CR register set), the DMA channel underrun flag (DMAUDR of DAC Sr register) fails to rise upon an internal trigger detection if that detection occurs during the same clock cycle as a DMA request acknowledge. As a result, the user application is not informed that an underrun error occurred.

This issue occurs when software and hardware triggers are used concurrently to trigger DMA transfers.

Workaround

None.

2.10 COMP
2.10.1 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.

2.11 DSI

2.11.1 Failing DSI read operation

Description
Following any type of read command sent by the device to a DSI panel display, the DSI peripheral on the device sporadically fails to capture valid data returned by the DSI panel display and signals packet size error (by setting the PSE flag of the DSI_ISR1 register), as if the display returned invalid data or aborted the transaction.

The defect occurs randomly, without any obvious dependency on a particular condition.

Workaround
Upon detecting a packet size error, resend the read command in a loop until the data is received without reporting packet size error.

The sequence is as follows:
1. Send a read command
2. Read data from the FIFO if it is not empty (if the PRDFE flag of the DSI_GPSR register is zero), pass otherwise
3. If the RCB flag of the DSI_GPSR register is set, signaling read command busy, revert to the point 3. Pass otherwise.
4. If the PSE flag of the DSI_ISR1 register is set, revert to the point 2 (read failure occurred).

2.12 TSC

2.12.1 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.
2.13 TIM

2.13.1 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.

2.14 LPTIM

2.14.1 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_APBByRSTRz register.

2.14.2 MCU 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 by writing the LPTIM_ICR bit in the LPTIM_ISR 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 at ‘1’.
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 MCU cannot enter Stop mode.
Workaround

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

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

Note: The proper clear sequence is already implemented in the HAL_LPTIM_IRQHandler in the STM32Cube.

2.14.3 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.

2.15 RTC and TAMP

2.15.1 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:

```c
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 */
    }
}
```
Workaround

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

```c
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 */
    }
}
```

2.15.2 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 RTCCCLK cycles after being cleared, the INITF flag is set immediately instead of waiting for synchronization delay (which should be between one and two RTCCCLK cycles), and the initialization of registers may fail.

Depending on the INIT bit clearing and setting instants versus the RTCCCLK edges, it can happen that, after being immediately set, the INITF flag is cleared during one RTCCCLK 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.

2.15.3 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 though 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 tentative to clear the flag(s) ineffective.

Workaround

None.

2.15.4 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.

2.16 I2C

2.16.1 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave

Description

An I2C-bus master generates STOP condition upon non-acknowledge of I2C 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 I2C-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 I2C-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.
2.16.2 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 I2C-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 I2C-bus frequency.

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

Workaround

No application workaround is required.

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

Description

The I2C-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 I2C-bus SDA line when t_{SU;DAT} is smaller than one I2C kernel clock (I2C-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 I2C-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.

2.16.4 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 I2C-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.
2.16.5 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.

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.

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.

2.16.6 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.

**2.16.7 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.

**2.16.8 OVR flag not set in underrun condition**

**Description**

In slave transmission with clock stretching disabled (NOSTRETCH = 1 in the I2C.CR1 register), an underrun condition occurs if the current byte transmission is completed on the I2C bus, and the next data is not yet written in the TXDATA[7:0] bitfield. In this condition, the device is expected to set the OVR flag of the I2C.ISR register and send 0xFF on the bus.

However, if the I2C_TXDR is written within the interval between two I2C kernel clock cycles before and three APB clock cycles after the start of the next data transmission, the OVR flag is not set, although the transmitted value is 0xFF.

**Workaround**

None.

**2.16.9 Transmission stalled after first byte transfer**

**Description**

When the first byte to transmit is not prepared in the TXDATA register, two bytes are required successively, through TXIS status flag setting or through a DMA request. If the first of the two bytes is written in the I2C_TXDR register in less than two I2C kernel clock cycles after the TXIS/DMA request, and the ratio between APB clock and I2C kernel clock frequencies is between 1.5 and 3, the second byte written in the I2C_TXDR is not internally detected. This causes a state in which the I2C peripheral is stalled in master mode or in slave mode, with clock stretching enabled (NOSTRETCH = 0). This state can only be released by disabling the peripheral (PE = 0) or by resetting it.

**Workaround**

Apply one of the following measures:

- Write the first data in I2C_TXDR before the transmission starts.
- Set the APB clock frequency so that its ratio with respect to the I2C kernel clock frequency is lower than 1.5 or higher than 3.

**2.17 USART**
2.17.1 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.

2.17.2 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.

2.18 LPUART

2.18.1 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.

2.19 SPI

2.19.1 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.
2.19.2 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.

2.19.3 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 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.

2.20 bxCAN

2.20.1 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 bit of the CAN_MCR register must be kept cleared (time-triggered communication mode disabled).
Workaround
None.

2.21 OTG_FS

2.21.1 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.
### Table 6. Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-Oct-2017</td>
<td>1</td>
<td>Initial release. Added:</td>
</tr>
<tr>
<td>30-Mar-2018</td>
<td>2</td>
<td>• section 2.2.3: Flash memory might not be accessible when AHB prescaler is greater than eight</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.2.4: Flash OPTVERR flag is always set after system reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.2.5: HSE oscillator long startup at low voltage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.7.9: Octal indirect read operation is not started when OctoSPI is configured with address phase disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.9.1: Spurious temperature measurement due to spike noise</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.14.1: RTC_REFIN and RTC_OUT on PB2 not operating in Stop 2 mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.18.3: BSY bit may stay high when SPI is disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.1: Core</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.12.1: HSE/32 is not available for TIM16 input capture if RTC clock is disabled or other than HSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.13.1: LPTIM1 outputs cannot be configured as open-drain</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.15.6: Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.15.2: Wrong data sampling when data setup time (ISU;DAT) is shorter than one I2C kernel clock period</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.15.3: Spurious bus error detection in master mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.15.4: Last-received byte loss in reload mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.17.1: LPUART1 outputs cannot be configured as open-drain</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.18.1: BSY bit may stay high at the end of a data transfer in slave mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.18.2: CRC error in SPI slave mode if internal NSS changes before CRC transfer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Removed:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• START bit is not cleared when the address is not acknowledged by the slave device limitation from section 2.15: I2C</td>
</tr>
<tr>
<td>Date</td>
<td>Version</td>
<td>Changes</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>21-Jun-2018</td>
<td>3</td>
<td>Added:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Documentation errata to the document, summarized in Table 4: Summary of documentation errata and introduced in the cover page.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.3.2: Spurious Firewall reset is generated when accessing FMC or OctoSPI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.4.1: DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.4.2: Byte and half-word accesses not supported</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.5.1: DMAMUX_RGCFR register is write-only, not readwrite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.5.2: Request trigger overrun misdetection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.5.3: SOFx not asserted when writing into DMAMUX_CFR register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.5.4: DMA request counter not kept at GNBRQ bitfield value as long as the corresponding request channel is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.5.5: Synchronization event discarded if selected input DMA request is not active</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.7.10: Memory-mapped read of the last byte of the memory is not possible when in octal SDR mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.7.11: A memory-mapped write request with only one-byte length at odd-start address finished by any event is masked</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.8.3: Writing the register ADCx_JSQR when JADCSTART=1 and JQDIS=1 might lead to incorrect behavior</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.15.5: Spurious master transfer upon own slave address match</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• section 2.15.7: START bit is cleared upon setting ADDRCF, not upon address match</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 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</td>
</tr>
<tr>
<td>Date</td>
<td>Version</td>
<td>Changes</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 19-Dec-2018| 4       | **Added sections:**  

- section 2.2.6 SDMMC1SMEN bit of RCC_AHB2SMENR register only modifiable with word access  
- section 2.2.7 First double-word of Flash memory corrupted upon reset or power-down while programming  
- section 2.2.8 Unstable LSI when it clocks RTC or CSS on LSE  
- section 2.2.9 LTDC and DSI not functional with Stop 2  
- section 2.2.10 Regulator startup failure at low VDD  
- section 2.2.11 1-Mbyte devices wrongly configured as 2-Mbyte  
- section 2.5.3 OFx not asserted when writing into DMAMUX_RGCFR register  
- section 2.5.4 Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event  
- section 2.7.10 Unaligned AHB write requests not supported  
- section 2.7.12 Data masking for odd number of byte writes only working with D1/D0 ordering  
- section 2.7.14 Single-byte memory-mapped write to odd octal DDR address failing when a higher-priority event occurs  
- section 2.13.2 Calendar initialization may fail in case of consecutive INIT mode entry  

**Modified sections:**  

- section 2.5.5 DMAMUX_RGCFR register is write-only, not read-write  
- section 2.5.1 SOFx not asserted when writing into DMAMUX_CFR register moved to device limitations section  
- section 2.7.1 CSBOUND shall not be used updated workaround qualifier  
- section 2.7.2 Auto-polling mode not functional with new octal memories updated workaround qualifier  
- section 2.7.7 No transfer error interrupt upon indirect write to address exceeding the limit updated workaround qualifier  
- section 2.8.2 Writing ADCx_JSQR when JADCSTART and JQDIS are set might lead to incorrect behavior updated workaround  
- section 2.8.4 Spurious temperature measurement due to spike noise updated function section from TEMP to ADC  
- section 2.10.1 Inhibited acquisition in short transfer phase configuration moved to documentation errata section  
- section 2.14.2 Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C moved to documentation errata  
- section 2.14.3 Wrong data sampling when data setup time (tSU;DAT) is shorter than one I2C kernel clock period updated workaround qualifier  
- section 2.17.3 CRC error in SPI slave mode if internal NSS changes before CRC transfer moved to documentation errata
<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 18-Mar-2019| 5       | **Added:**  
|            |         | • section 2.7.15 Memory-mapped write data to odd start address corrupted when BUSY=0 and not in octal-DTR mode  
|            |         | • section 2.7.16 Data not sampled correctly on reads without DQS and with less than two cycles before the data phase  
|            |         | • section 2.13.3 Alarm flag may be repeatedly set when the core is stopped in debug  
|            |         | **Modified:**  
|            |         | • section 2.7.1 CSBOUND shall not be used  
|            |         | • section 2.7.11 Write operation in octal-DTR mode starting at odd address is not supported  
|            |         | • section 2.7.12 Data masking for odd number of byte writes only working with D1/D0 ordering  
|            |         | • START bit is cleared upon setting ADDR CF, not upon address match erratum reference moved from Table 3 to Table 4 |
| 24-Oct-2019| 6       | **Added errata into** Table 3. Summary of device limitations or Table 4. Summary of device documentation errata, and in Section 2 Description of device errata, subsections:  
|            |         | • 2.2.13 Bootloader not functional on 1-Mbyte devices  
|            |         | • 2.2.11 OBL_LAUNCH operation may corrupt SRAM content  
|            |         | • 2.2.12 IrrelevantBOOTO electrical characteristics  
|            |         | • 2.7.17 Memory-mapped write error response when DQS output is disabled  
|            |         | • 2.7.18 Byte possibly dropped during an SDR read in clock mode 3 when a transfer gets automatically split  
|            |         | • 2.7.19 Single, dual and quad modes not functional with DQS input enabled  
|            |         | • 2.7.20 Additional bytes read in indirect mode with DQS input enabled when data length is too short  
|            |         | • 2.7.21 DQS output enabled too late on write  
|            |         | • 2.7.22 PSRAM not supported  
|            |         | • 2.16.8 OVR flag not set in underrun condition  
|            |         | • 2.16.9 Transmission stalled after first byte transfer  
|            |         | • 2.8.3 ADEN bit cannot be set immediately after the ADC calibration is done  
|            |         | • 2.9.1 Invalid DAC analog output if the DAC channel MODE bitfield is programmed before DAC initialization  
|            |         | • 2.9.2 DMA underrun flag not set when an internal trigger is detected on the clock cycle of the DMA request acknowledge  
|            |         | • 2.14.2 MCU may remain stuck in LPTIM interrupt when clearing event flag  
|            |         | **Modified errata in** Section 2 Description of device errata, subsections:  
|            |         | • 2.2.3 Flash memory might not be accessible when AHB prescaler is greater than two  
|            |         | • 2.2.10 1-Mbyte devices wrongly configured as 2-Mbyte  
|            |         | • 2.3.2 Spurious Firewall reset  
|            |         | • 2.8.2 Writing ADC_JSOR when JADCSTART and JQDIS are set may lead to incorrect behavior  
|            |         | **Removed erratum First double-word of Flash memory corrupted upon reset or power down while programming.**
Contents

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

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

  2.1 Core ..................................................................................... 6
    2.1.1 Interrupted loads to SP can cause erroneous behavior .......... 6

  2.2 System ............................................................................... 6
    2.2.1 Full JTAG configuration without NJTRST pin cannot be used .... 6
    2.2.2 Data cache might be corrupted during Flash memory read-while-write operation .... 6
    2.2.3 Flash memory might not be accessible when AHB prescaler is greater than two .... 7
    2.2.4 Flash OPTVERR flag is always set after system reset ............... 7
    2.2.5 HSE oscillator long startup at low voltage ......................... 7
    2.2.6 SDMMC1SMEN bit of RCC_AHB2SMENR register only modifiable with word access .... 8
    2.2.7 Unstable LSI when it clocks RTC or CSS on LSE ....................... 8
    2.2.8 LTDC and DSI not functional with Stop 2 .............................. 8
    2.2.9 Regulator startup failure at low V_DD ................................ 8
    2.2.10 1-Mbyte devices wrongly configured as 2-Mbyte .................. 9
    2.2.11 OBL_LAUNCH operation may corrupt SRAM content .............. 9
    2.2.12 Irrelevant BOOT0 electrical characteristics ............................ 9
    2.2.13 Bootloader not functional on 1-Mbyte devices ...................... 9

  2.3 FW ..................................................................................... 10
    2.3.1 Firewall protection size limitation ....................................... 10
    2.3.2 Spurious Firewall reset ....................................................... 10

  2.4 DMA .................................................................................. 10
    2.4.1 DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear .... 10

  2.5 DMAMUX .......................................................................... 10
    2.5.1 SOFx not asserted when writing into DMAMUX_CFR register .... 10
    2.5.2 OFx not asserted for trigger event coinciding with last DMAMUX request .......... 11
    2.5.3 OFx not asserted when writing into DMAMUX_RGCFR register .......... 11
    2.5.4 Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event .......... 11
    2.5.5 DMAMUX_RGCFR register is write-only, not read-write .......... 12
2.5.6 DMA request counter not kept at GNBREQ bitfield value as long as the corresponding request channel is disabled ................................................ 12
2.5.7 Synchronization event discarded if selected input DMA request is not active .... 12
2.6 FMC ........................................................................ 12
2.6.1 Dummy read cycles inserted when reading synchronous memories ............ 12
2.6.2 Wrong data read from a busy NAND memory ........................................ 13
2.7 OCTOSPI .................................................................. 13
2.7.1 CSBOUND shall not be used .......................................... 13
2.7.2 Auto-polling mode not functional with new octal memories ..................... 13
2.7.3 Command phase must be octal for octal transfers .................................... 13
2.7.4 Write data lost with Clock mode 3 ........................................ 13
2.7.5 Deadlock upon disabling OCTOSPI during memory-mapped write ............ 14
2.7.6 Deadlock in dual-flash configuration with odd number of bytes ............... 14
2.7.7 No transfer error interrupt upon indirect write to address exceeding the limit . 14
2.7.8 DHQC not effective if DDTR not set ...................................... 14
2.7.9 Indirect read and auto-polling transfers without address phase not starting ... 15
2.7.10 Unaligned AHB write requests not supported ...................................... 15
2.7.11 Write operation in octal-DTR mode starting at odd address is not supported . 15
2.7.12 Data masking for odd number of byte writes only working with D1/D0 ordering. 15
2.7.13 Memory-mapped read of the last memory space byte not possible in SDR octal mode. 15
2.7.14 Single-byte memory-mapped write to odd octal DDR address failing when a higher-priority event occurs .............................................................. 16
2.7.15 Memory-mapped write data to odd start address corrupted when BUSY=0 and not in octal-DTR mode ................................................................. 16
2.7.16 Data not sampled correctly on reads without DQS and with less than two cycles before the data phase ................................................................. 16
2.7.17 Memory-mapped write error response when DQS output is disabled ............ 17
2.7.18 Byte possibly dropped during an SDR read in clock mode 3 when a transfer gets automatically split ................................................................. 17
2.7.19 Single, dual and quad modes not functional with DQS input enabled .......... 18
2.7.20 Additional bytes read in indirect mode with DQS input enabled when data length is too short ................................................................. 18
2.7.21 DQS output enabled too late on write ........................................... 18
2.7.22 PSRAM not supported ...................................................... 18
2.8 ADC ........................................................................ 18
2.8.1 Injected queue of context is not available if JQM = 0 ............................................. 18
2.8.2 Writing ADC_JSQR when JADCSTART and JQDIS are set may lead to incorrect behavior . 19
2.8.3 ADEN bit cannot be set immediately after the ADC calibration is done ..................... 19
2.8.4 Wrong ADC result if conversion done late after calibration or previous conversion ...... 19
2.8.5 Spurious temperature measurement due to spike noise ........................................ 19
2.9 DAC ......................................................................................................................... 20
2.9.1 Invalid DAC channel analog output if the DAC channel MODE bitfield is programmed before DAC initialization .......................................................... 20
2.9.2 DMA underrun flag not set when an internal trigger is detected on the clock cycle of the DMA request acknowledge ................................................. 20
2.10 COMP .................................................................................................... 20
2.10.1 Comparator outputs cannot be configured in open-drain ........................................ 20
2.11 DSI .......................................................................................................................... 21
2.11.1 Failing DSI read operation ................................................................................... 21
2.12 TSC ........................................................................................................................... 21
2.12.1 Inhibited acquisition in short transfer phase configuration ..................................... 21
2.13 TIM .......................................................................................................................... 21
2.13.1 HSE/32 is not available for TIM16 input capture if RTC clock is disabled or other than HSE. ................................................................. 22
2.14 LPTIM ....................................................................................................................... 22
2.14.1 MCU may remain stuck in LPTIM interrupt when entering Stop mode ..................... 22
2.14.2 MCU may remain stuck in LPTIM interrupt when clearing event flag ...................... 22
2.14.3 LPTIM1 outputs cannot be configured as open-drain ........................................... 23
2.15 RTC and TAMP ....................................................................................................... 23
2.15.1 RTC interrupt can be masked by another RTC interrupt ......................................... 23
2.15.2 Calendar initialization may fail in case of consecutive INIT mode entry ................... 24
2.15.3 Alarm flag may be repeatedly set when the core is stopped in debug ....................... 25
2.15.4 RTC_REFIN and RTC_OUT on PB2 not operating in Stop 2 mode ......................... 25
2.16 I2C ........................................................................................................................... 25
2.16.1 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave ........................................... 25
2.16.2 Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C ....... 25
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.16.3</td>
<td>Wrong data sampling when data setup time ((t_{SU;DAT})) is shorter than one I2C kernel clock period</td>
</tr>
<tr>
<td>2.16.4</td>
<td>Spurious bus error detection in master mode</td>
</tr>
<tr>
<td>2.16.5</td>
<td>Last-received byte loss in reload mode</td>
</tr>
<tr>
<td>2.16.6</td>
<td>Spurious master transfer upon own slave address match</td>
</tr>
<tr>
<td>2.16.7</td>
<td>START bit is cleared upon setting ADDRCF, not upon address match</td>
</tr>
<tr>
<td>2.16.8</td>
<td>OVR flag not set in underrun condition</td>
</tr>
<tr>
<td>2.16.9</td>
<td>Transmission stalled after first byte transfer</td>
</tr>
<tr>
<td>2.17</td>
<td>USART</td>
</tr>
<tr>
<td>2.17.1</td>
<td>nRTS is active while RE = 0 or UE = 0</td>
</tr>
<tr>
<td>2.17.2</td>
<td>UDR flag set while the SPI slave transmitter is disabled</td>
</tr>
<tr>
<td>2.18</td>
<td>LPUART</td>
</tr>
<tr>
<td>2.18.1</td>
<td>LPUART1 outputs cannot be configured as open-drain</td>
</tr>
<tr>
<td>2.19</td>
<td>SPI</td>
</tr>
<tr>
<td>2.19.1</td>
<td>BSY bit may stay high when SPI is disabled</td>
</tr>
<tr>
<td>2.19.2</td>
<td>BSY bit may stay high at the end of data transfer in slave mode</td>
</tr>
<tr>
<td>2.19.3</td>
<td>CRC error in SPI slave mode if internal NSS changes before CRC transfer</td>
</tr>
<tr>
<td>2.20</td>
<td>bxCAN</td>
</tr>
<tr>
<td>2.20.1</td>
<td>bxCAN time-triggered communication mode not supported</td>
</tr>
<tr>
<td>2.21</td>
<td>OTG_FS</td>
</tr>
<tr>
<td>2.21.1</td>
<td>Data FIFO gets corrupted if the write sequence to the transmit FIFO is interleaved with other OTGFS register access</td>
</tr>
</tbody>
</table>

**Revision history**