

## Applicability

This document applies to the part numbers of STM32L475xx devices listed in [Table 1](#) and their variants shown in [Table 2](#).

[Section 1](#) gives a summary and [Section 2](#) a description of workarounds for device limitations, with respect to the device datasheet and reference manual RM0351.

**Table 1. Device summary**

Reference	Part numbers
STM32L475xx	STM32L475RC, STM32L475RE, STM32L475RG STM32L475VC, STM32L475VE, STM32L475VG

**Table 2. Device variants**

Reference	Silicon revision codes	
	Device marking <sup>(1)</sup>	REV_ID <sup>(2)</sup>
STM32L475xx	4	0x1007

1. Refer to the device datasheet for details on how to identify this code on different types of package.
2. The REV\_ID[15:0] bit field of DBGMCU\_IDC register (refer to the reference manual RM0351).

# Contents

<b>1</b>	<b>Summary of device limitations</b>	<b>8</b>
<b>2</b>	<b>Description of device limitations</b>	<b>13</b>
2.1	Core	13
2.1.1	Interrupted loads to stack pointer can cause erroneous behavior	13
2.1.2	VDIV or VSQRT instructions might not complete correctly when very short ISRs are used	14
2.1.3	Store immediate overlapping exception return operation might vector to incorrect interrupt	15
2.2	FW	16
2.2.1	Code segment unprotected if non-volatile data segment length is 0	16
2.2.2	Code and non-volatile data unprotected upon bank swap	16
2.3	System	17
2.3.1	Write operation in the Flash memory while it is not ready (Flash memory in power-down) is not correctly handled	17
2.3.2	The configuration of the I/Os not available in WLCSP package can be modified by software	17
2.3.3	SRAM2 read access while the SRAM2 hardware erase is ongoing is not correctly supported	17
2.3.4	PH0/PH1 is controlled by the GPIOH registers when HSE is enabled	17
2.3.5	PWR_CR4 register write access may not be completed if a Low-power mode is entered just after the write operation	18
2.3.6	HSI user trim is limited on some samples	18
2.3.7	Option byte loading can fail if MSI frequency is greater than 8 MHz	19
2.3.8	PLL may not lock if VCO is below 96 MHz and temperature is below 0 °C	19
2.3.9	Full JTAG configuration without NJTRST pin cannot be used	20
2.3.10	MSIRDY flag issue preventing entry in low power mode	20
2.3.11	PCPROP area within a single Flash memory page becomes unprotected at RDP change from level 1 to level 0	21
2.3.12	Data Cache might be corrupted during Flash Read While Write operation	21
2.3.13	MSI frequency overshoot upon Stop mode exit	22
2.3.14	Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled	22
2.3.15	OPAMP output: VDDA overconsumption	23
2.3.16	Spurious BOR when entering Stop mode after short Run sequence	23

2.3.17	HSE long start-up at low voltage . . . . .	24
2.3.18	Current injection from VDD to VDDA through analog switch voltage booster . . . . .	24
2.3.19	Unstable LSI when it clocks RTC or CSS on LSE . . . . .	25
2.3.20	FLASH_ECCR corrupted upon reset or power-down occurring during Flash memory program or erase operation . . . . .	25
2.4	FMC . . . . .	26
2.4.1	Dummy read cycles inserted when reading synchronous memories . . .	26
2.4.2	Data corruption during burst read from FMC synchronous memory . . .	26
2.4.3	FMC NOR/PSRAM controller bank switch with different BUSTURN durations . . . . .	26
2.4.4	Read burst access of nine words or more is not supported by FMC . . .	27
2.4.5	Spurious clock stoppage with continuous clock feature enabled . . . . .	27
2.5	DMA . . . . .	28
2.5.1	DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear . . . . .	28
2.6	QUADSPI . . . . .	28
2.6.1	QUADSPI_BK1_IO1 is always an input when the command is sent in Dual- or Quad-SPI mode . . . . .	28
2.6.2	Hard fault is not generated in case of out-of-range memory-mapped access . . . . .	28
2.6.3	Extra data written in the FIFO at the end of a read transfer . . . . .	29
2.6.4	First nibble of data is not written after dummy phase . . . . .	29
2.6.5	Wrong data can be read in memory-mapped after an indirect mode operation . . . . .	30
2.6.6	Memory-mapped read operations may fail when timeout counter is enabled . . . . .	30
2.7	ADC . . . . .	30
2.7.1	Injected queue of context is not available in case of JQM = 0 . . . . .	30
2.7.2	DMA2 channels 2 and 3 (respectively) cannot be used when the ADC2 and ADC3 requests are selected on DMA2 channels 4 and 5 . . .	31
2.7.3	Wrong ADC conversion results when delay between calibration and first conversion or between two consecutive conversions is too long . .	31
2.7.4	Burst read or write accesses are not supported by the ADC . . . . .	31
2.7.5	Unexpected end of transfer on DMA1 when using DMA1 channel 2 for ADC2 while DMA2 channel 4 is also used . . . . .	32
2.7.6	Unexpected end of transfer on DMA1 when using DMA1 channel 3 for ADC3 while DMA2 channel 5 is also used . . . . .	32
2.7.7	Spurious temperature measurement due to spike noise . . . . .	32

2.7.8	Writing ADCx_JSQR when JADCSTART and JQDIS are set might lead to incorrect behavior .....	33
2.8	COMP .....	33
2.8.1	Comparators propagation delay is longer than expected for input steps higher than 200 mV .....	33
2.8.2	Comparators output cannot be configured in open-drain .....	34
2.8.3	COMP1 and COMP2 configuration lost with software reset .....	34
2.9	DFSDM .....	35
2.9.1	RDATACH[2:0] status bits are not implemented .....	35
2.9.2	New regular channel selection taken into account at the end of an injected conversion when a regular conversion is pending .....	35
2.9.3	DFSDM triggers from timers can be missed in specific conditions .....	35
2.10	TSC .....	35
2.10.1	Inhibited acquisition in short transfer phase configuration .....	35
2.11	LPTIM .....	36
2.11.1	Low power timer 1 (LPTIM1) outputs cannot be configured in open-drain .....	36
2.11.2	MCU may remain stuck in LPTIM interrupt when entering Stop mode ..	36
2.12	RTC .....	36
2.12.1	Spurious tamper detection when disabling the tamper channel .....	36
2.12.2	RTC calendar registers are not locked properly .....	37
2.12.3	RTC interrupt can be masked by another RTC interrupt .....	37
2.12.4	RTC_OUT on PB2 and RTC_REFIN on PB15 are not functional in Stop 2 mode .....	39
2.13	I2C .....	39
2.13.1	I2C Fast-mode Plus drive is not available on all SDA/SCL I/Os .....	39
2.13.2	Wrong behavior related with MCU Stop mode when wakeup from Stop mode by I2C peripheral is disabled .....	39
2.13.3	Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period .....	40
2.13.4	Spurious Bus Error detection in master mode .....	40
2.13.5	I2C3 analog filters requires that both PC0/PC1 or both PG7/PG8 are configured as I2C3 alternate function .....	40
2.13.6	10-bit master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave .....	41
2.13.7	Last received byte can be lost when using Reload mode with NBYTES > 1 .....	41
2.13.8	Spurious master transfer upon own slave address match .....	42
2.14	USART .....	43

2.14.1	Start bit detected too soon when sampling for NACK signal from the smartcard	43
2.14.2	Break request can prevent the Transmission Complete flag (TC) from being set	43
2.14.3	nRTS is active while RE or UE = 0	43
2.15	LPUART	44
2.15.1	Low power UART1 (LPUART1) outputs cannot be configured in open-drain	44
2.16	SPI	44
2.16.1	BSY bit may stay high at the end of a data transfer at slave mode	44
2.17	SWPMI	44
2.17.1	SUSPENDED mode entry delayed	44
2.17.2	SUSPENDED mode never entered	45
2.17.3	SRF flag not set	45
2.17.4	SWP SUSPENDED mode not supported when STM32L4 is in Stop 0 or Stop 1 mode	45
2.17.5	SWPMI_IO transceiver bypass mode is not functional	45
2.18	SDMMC	46
2.18.1	Wrong CCRCFAIL status after a response without CRC is received	46
2.18.2	Clock division per 255 is not possible	46
2.18.3	Wait for response bits “10” configuration does not work correctly	46
2.18.4	MMC stream write of less than 8 bytes does not work correctly	46
2.19	bxCAN	47
2.19.1	bxCAN time-triggered mode is not supported	47
2.20	OTG_FS	47
2.20.1	Suspend mode robustness marginality after receiving corrupted packet from Host	47
2.20.2	Bits OTG_FS_GLPMPCFG[6:2] are not write-protected in Device mode	48
2.20.3	L1 exit with simultaneous Device initiated and Host initiated Resume results in minimum Device wake-up time of 3 ms	48
2.20.4	Data FIFO gets corrupted if the write sequence to the Transmit FIFO is interleaved with other OTGFS register access	48
<b>3</b>	<b>Revision history</b>	<b>49</b>

List of tables

Table 1. Device summary . . . . . 1

Table 2. Device variants . . . . . 1

Table 3. Summary of device limitations . . . . . 8

Table 4. Minimum Run time to avoid spurious BORs . . . . . 23

Table 5. Example of IQM algorithm . . . . . 33

Table 6. COMP characteristics . . . . . 34

Table 7. Document revision history . . . . . 49



List of figures

Figure 1. HSI oscillator trimming characteristics ..... 19

Figure 2. Minimum Run time definition ..... 24

Figure 3. Masked RTC interrupt. .... 38

# 1 Summary of device limitations

[Table 3](#) gives quick references to all documented device limitations of STM32L475xx 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 the target application. Adoption of a workaround may cause restrictions to the target application. Workaround for a limitation is deemed partial if it only reduces the rate of occurrence and/or the 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	Rev 4
Core	2.1.1	<i>Interrupted loads to stack pointer can cause erroneous behavior</i>	A
	2.1.2	<i>VDIV or VSQRT instructions might not complete correctly when very short ISRs are used</i>	A
	2.1.3	<i>Store immediate overlapping exception return operation might vector to incorrect interrupt</i>	A
FW	2.2.1	<i>Code segment unprotected if non-volatile data segment length is 0</i>	A
	2.2.2	<i>Code and non-volatile data unprotected upon bank swap</i>	A
System	2.3.1	<i>Write operation in the Flash memory while it is not ready (Flash memory in power-down) is not correctly handled</i>	A
	2.3.2	<i>The configuration of the I/Os not available in WLCSP package can be modified by software</i>	A
	2.3.3	<i>SRAM2 read access while the SRAM2 hardware erase is ongoing is not correctly supported</i>	N
	2.3.4	<i>PH0/PH1 is controlled by the GPIOH registers when HSE is enabled</i>	P
	2.3.5	<i>PWR_CR4 register write access may not be completed if a Low-power mode is entered just after the write operation</i>	A
	2.3.6	<i>HSI user trim is limited on some samples</i>	N <sup>(1)</sup>
	2.3.7	<i>Option byte loading can fail if MSI frequency is greater than 8 MHz</i>	A
	2.3.8	<i>PLL may not lock if VCO is below 96 MHz and temperature is below 0 °C</i>	A
	2.3.9	<i>Full JTAG configuration without NJTRST pin cannot be used</i>	A
	2.3.10	<i>MSIRDY flag issue preventing entry in low power mode</i>	A
	2.3.11	<i>PCPROP area within a single Flash memory page becomes unprotected at RDP change from level 1 to level 0</i>	A



Table 3. Summary of device limitations (continued)

Function	Section	Limitation	Rev 4
System	2.3.12	<i>Data Cache might be corrupted during Flash Read While Write operation</i>	A
	2.3.13	<i>MSI frequency overshoot upon Stop mode exit</i>	A
	2.3.14	<i>Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled</i>	A
	2.3.15	<i>OPAMP output: VDDA overconsumption</i>	N
	2.3.16	<i>Spurious BOR when entering Stop mode after short Run sequence</i>	A
	2.3.17	<i>HSE long start-up at low voltage</i>	A
	2.3.18	<i>Current injection from VDD to VDDA through analog switch voltage booster</i>	A
	2.3.19	<i>Unstable LSI when it clocks RTC or CSS on LSE</i>	P
	2.3.20	<i>FLASH_ECCOR corrupted upon reset or power-down occurring during Flash memory program or erase operation</i>	A
FMC	2.4.1	<i>Dummy read cycles inserted when reading synchronous memories</i>	N
	2.4.2	<i>Data corruption during burst read from FMC synchronous memory</i>	A
	2.4.3	<i>FMC NOR/PSRAM controller bank switch with different BUSTURN durations</i>	A
	2.4.4	<i>Read burst access of nine words or more is not supported by FMC</i>	P
	2.4.5	<i>Spurious clock stoppage with continuous clock feature enabled</i>	A
DMA	2.5.1	<i>DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear</i>	A
QUADSPI	2.6.1	<i>QUADSPI_BK1_IO1 is always an input when the command is sent in Dual- or Quad-SPI mode</i>	P
	2.6.2	<i>Hard fault is not generated in case of out-of-range memory-mapped access</i>	N
	2.6.3	<i>Extra data written in the FIFO at the end of a read transfer</i>	A
	2.6.4	<i>First nibble of data is not written after dummy phase</i>	A
	2.6.5	<i>Wrong data can be read in memory-mapped after an indirect mode operation</i>	A
	2.6.6	<i>Memory-mapped read operations may fail when timeout counter is enabled</i>	A

Table 3. Summary of device limitations (continued)

Function	Section	Limitation	Rev 4
ADC	2.7.1	Injected queue of context is not available in case of JQM = 0	N
	2.7.2	DMA2 channels 2 and 3 (respectively) cannot be used when the ADC2 and ADC3 requests are selected on DMA2 channels 4 and 5	P
	2.7.3	Wrong ADC conversion results when delay between calibration and first conversion or between two consecutive conversions is too long	N
	2.7.4	Burst read or write accesses are not supported by the ADC	A
	2.7.5	Unexpected end of transfer on DMA1 when using DMA1 channel 2 for ADC2 while DMA2 channel 4 is also used	A
	2.7.6	Unexpected end of transfer on DMA1 when using DMA1 channel 3 for ADC3 while DMA2 channel 5 is also used	A
	2.7.7	Spurious temperature measurement due to spike noise	A
	2.7.8	Writing ADCx_JSQR when JADCSTART and JQDIS are set might lead to incorrect behavior	N
COMP	2.8.1	Comparators propagation delay is longer than expected for input steps higher than 200 mV	N
	2.8.2	Comparators output cannot be configured in open-drain	N
	2.8.3	COMP1 and COMP2 configuration lost with software reset	N
DFSDM	2.9.1	RDATACH[2:0] status bits are not implemented	N
	2.9.2	New regular channel selection taken into account at the end of an injected conversion when a regular conversion is pending	N
	2.9.3	DFSDM triggers from timers can be missed in specific conditions	A
TSC	2.10.1	Inhibited acquisition in short transfer phase configuration	A
LPTIM	2.11.1	Low power timer 1 (LPTIM1) outputs cannot be configured in open-drain	N
	2.11.2	MCU may remain stuck in LPTIM interrupt when entering Stop mode	P
RTC	2.12.1	Spurious tamper detection when disabling the tamper channel	P
	2.12.2	RTC calendar registers are not locked properly	A
	2.12.3	RTC interrupt can be masked by another RTC interrupt	A
	2.12.4	RTC_OUT on PB2 and RTC_REFIN on PB15 are not functional in Stop 2 mode	P

Table 3. Summary of device limitations (continued)

Function	Section	Limitation	Rev 4
I2C	2.13.1	<i>I2C Fast-mode Plus drive is not available on all SDA/SCL I/Os</i>	N
	2.13.2	<i>Wrong behavior related with MCU Stop mode when wakeup from Stop mode by I2C peripheral is disabled</i>	A
	2.13.3	<i>Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period</i>	P
	2.13.4	<i>Spurious Bus Error detection in master mode</i>	A
	2.13.5	<i>I2C3 analog filters requires that both PC0/PC1 or both PG7/PG8 are configured as I2C3 alternate function</i>	A
	2.13.6	<i>10-bit master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave</i>	A
	2.13.7	<i>Last received byte can be lost when using Reload mode with NBYTES &gt; 1</i>	P
	2.13.8	<i>Spurious master transfer upon own slave address match</i>	P
USART	2.14.1	<i>Start bit detected too soon when sampling for NACK signal from the smartcard</i>	N
	2.14.2	<i>Break request can prevent the Transmission Complete flag (TC) from being set</i>	A
	2.14.3	<i>nRTS is active while RE or UE = 0</i>	A
LPUART	2.15.1	<i>Low power UART1 (LPUART1) outputs cannot be configured in open-drain</i>	N
SPI	2.16.1	<i>BSY bit may stay high at the end of a data transfer at slave mode</i>	A
SWPMI	2.17.1	<i>SUSPENDED mode entry delayed</i>	N
	2.17.2	<i>SUSPENDED mode never entered</i>	P
	2.17.3	<i>SRF flag not set</i>	N
	2.17.4	<i>SWP SUSPENDED mode not supported when STM32L4 is in Stop 0 or Stop 1 mode</i>	N
	2.17.5	<i>SWPMI_IO transceiver bypass mode is not functional</i>	N
SDMMC	2.18.1	<i>Wrong CCRCFAIL status after a response without CRC is received</i>	A
	2.18.2	<i>Clock division per 255 is not possible</i>	N
	2.18.3	<i>Wait for response bits "10" configuration does not work correctly</i>	A
	2.18.4	<i>MMC stream write of less than 8 bytes does not work correctly</i>	A
bxCAN	2.19.1	<i>bxCAN time-triggered mode is not supported</i>	N

Table 3. Summary of device limitations (continued)

Function	Section	Limitation	Rev 4
OTG_FS	2.20.1	<i>Suspend mode robustness marginality after receiving corrupted packet from Host</i>	A
	2.20.2	<i>Bits OTG_FS_GLPMCFG[6:2] are not write-protected in Device mode</i>	P
	2.20.3	<i>L1 exit with simultaneous Device initiated and Host initiated Resume results in minimum Device wake-up time of 3 ms</i>	P
	2.20.4	<i>Data FIFO gets corrupted if the write sequence to the Transmit FIFO is interleaved with other OTGFS register access</i>	A

1. Fixed on products with date code newer than 609.

## 2 Description of device limitations

The following sections describe device limitations and provide workarounds if available. They are grouped by device function.

### 2.1 Core

Errata notice for the Arm<sup>®(a)</sup> Cortex<sup>®</sup>-M4 FPU core revisions r0 is available from <http://infocenter.arm.com>.

Only applicable information from the Arm<sup>®</sup> errata notice is replicated in this document. Extra information may be added for more clarity.



#### 2.1.1 Interrupted loads to stack pointer can cause erroneous behavior

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

##### Description

An interrupt occurring during the data-phase of a single word load to the stack pointer (SP/R13) can cause an erroneous behavior of the device. In addition, returning from the interrupt results in the load instruction being executed with an additional time.

For all the instructions performing an update of the base register, the base register is erroneously updated on each execution, resulting in the stack pointer being loaded from an incorrect memory location.

The instructions affected by this limitation are the following:

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

##### Workaround

As of today, no compiler generates these particular instructions. This limitation can only occur with hand-written assembly code.

Both issues can be solved by replacing the direct load to the stack pointer by an intermediate load to a general-purpose register followed by a move to the stack pointer.

---

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

Example:

```
Replace LDR SP, [R0] with  
LDR R2,[R0]  
MOV SP,R2
```

### 2.1.2 VDIV or VSQRT instructions might not complete correctly when very short ISRs are used

This limitation is registered under Arm® ID number 776924 and classified into “Category B”. Its impact to the device is limited.

#### Description

The VDIV and VSQRT instructions take 14 cycles to execute. When an interrupt is taken a VDIV or VSQRT instruction is not terminated, and completes its execution while the interrupt stacking occurs. If lazy context save of floating point state is enabled then the automatic stacking of the floating point context does not occur until a floating point instruction is executed inside the interrupt service routine.

Lazy context save is enabled by default. When it is enabled, the minimum time for the first instruction in the interrupt service routine to start executing is 12 cycles. In certain timing conditions, and if there is only one or two instructions inside the interrupt service routine, then the VDIV or VSQRT instruction might not write its result to the register bank or to the FPSCR.

The failure occurs when the following condition is met:

1. The floating point unit is enabled
2. Lazy context saving is not disabled
3. A VDIV or VSQRT is executed
4. The destination register for the VDIV or VSQRT is one of s0 - s15
5. An interrupt occurs and is taken
6. The interrupt service routine being executed does not contain a floating point instruction
7. Within 14 cycles after the VDIV or VSQRT is executed, an interrupt return is executed

A minimum of 12 of these 14 cycles are utilized for the context state stacking, which leaves two cycles for instructions inside the interrupt service routine, or two wait states applied to the entire stacking sequence (which means that it is not a constant wait state for every access).

In general, this means that if the memory system inserts wait states for stack transactions (that is, external memory is used for stack data), then this erratum cannot be observed.

The effect of this erratum is that the VDIV or VSQRT instruction does not complete correctly and the register bank and FPSCR are not updated, which means that these registers hold incorrect (out of date) data.

#### Workaround

A workaround is only required if the floating point unit is enabled. A workaround is not required if the stack is in external memory.

There are two possible workarounds:

1. Disable lazy context save of floating point state by clearing LSPEN to 0 (bit 30 of the FPCCR at address 0xE000EF34).
2. Ensure that every interrupt service routine contains more than two instructions in addition to the exception return instruction.

### 2.1.3 Store immediate overlapping exception return operation might vector to incorrect interrupt

This limitation is registered under Arm® ID number 838869 and classified into “Category B (rare)”. Its impact to the device is minor.

#### Description

The core includes a write buffer that permits execution to continue while a store is waiting on the bus. Under specific timing conditions, during an exception return while this buffer is still in use by a store instruction, a late change in selection of the next interrupt to be taken might result in there being a mismatch between the interrupt acknowledged by the interrupt controller and the vector fetched by the processor.

The failure occurs when the following condition is met:

1. The handler for interrupt A is being executed.
2. Interrupt B, of the same or lower priority than interrupt A, is pending.
3. A store with immediate offset instruction is executed to a bufferable location.
  - STR/STRH/STRB <Rt>, [<Rn>,#imm]
  - STR/STRH/STRB <Rt>, [<Rn>,#imm]!
  - STR/STRH/STRB <Rt>, [<Rn>],#imm
4. Any number of additional data-processing instructions can be executed.
5. A BX instruction is executed that causes an exception return.
6. The store data has wait states applied to it such that the data is accepted at least two cycles after the BX is executed.
  - Minimally, this is two cycles if the store and the BX instruction have no additional instructions between them.
  - The number of wait states required to observe this erratum needs to be increased by the number of cycles between the store and the interrupt service routine exit instruction.
7. Before the bus accepts the buffered store data, another interrupt C is asserted which has the same or lower priority as A, but a greater priority than B.

Example:

The processor should execute interrupt handler C, and on completion of handler C should execute the handler for B. If the conditions above are met, then this erratum results in the processor erroneously clearing the pending state of interrupt C, and then executing the handler for B twice. The first time the handler for B is executed it will be at interrupt C's priority level. If interrupt C is pending by a level-based interrupt which is cleared by C's handler then interrupt C will be pending again once the handler for B has completed and the handler for C will be executed.

As the STM32 interrupt C is level based, it eventually becomes pending again and is subsequently handled.

### Workaround

For software not using the memory protection unit, this erratum can be worked around by setting DISDEFWBUF in the Auxiliary Control Register.

In all other cases, the erratum can be avoided by ensuring a DSB occurs between the store and the BX instruction. For exception handlers written in C, this can be achieved by inserting the appropriate set of intrinsics or inline assembly just before the end of the interrupt function, for example:

```
ARMCC:
...
__schedule_barrier();
__asm{DSB};
__schedule_barrier();
}
GCC:
...
__asm volatile ("dsb 0xf" ::: "memory");
}
```

## 2.2 FW

### 2.2.1 Code segment unprotected if non-volatile data segment length is 0

#### Description

If, during FW configuration, the length of firewall-protected non-volatile data segment is set to 0 through the LENG[21:8] bitfield of the FW\_NVDSL register, the firewall protection of code segment does not operate.

#### Workaround

Always set the LENG[21:8] bitfield of the FW\_NVDSL register to a non-zero value, even if no firewall protection of data in the non-volatile data segment is required.

### 2.2.2 Code and non-volatile data unprotected upon bank swap

#### Description

With firewall-protected code and non-volatile data segments located in the same Flash memory bank, both segments become unprotected (available to illegal access) upon Flash memory bank swap by software.

#### Workaround

Map the firewall-protected code segment in one Flash memory bank and the non-volatile data segment in another bank in a symmetric way (same relative address and same length).



## 2.3 System

### 2.3.1 Write operation in the Flash memory while it is not ready (Flash memory in power-down) is not correctly handled

#### Description

If the Flash memory is read while it is not ready after a power-down state (i.e after Stop mode, or when it was put in power-down by means of SLEEP\_PD or RUN\_PD bits in the FLASH\_ACR register), the AHB bus is normally stalled until the Flash memory is ready and can provide the correct data.

However, this mechanism is not supported in case of a write operation in normal programming mode: the Flash memory read that is automatically performed before the programming in order to check that the address is virgin is incorrect. Consequently the write operation is skipped.

#### Workaround

The user must wait until the Flash memory is ready before performing a write operation: this is done by reading first an address of the Flash memory not present in the cache.

### 2.3.2 The configuration of the I/Os not available in WLCSP package can be modified by software

#### Description

Unbonded I/Os in WLCSP package can be configured by software. This can lead to an extra consumption if they are floating with Schmitt trigger ON.

#### Workaround

Configure all unbonded I/Os as analog input or as output push-pull 0 state.

### 2.3.3 SRAM2 read access while the SRAM2 hardware erase is ongoing is not correctly supported

#### Description

If a read access is done in the SRAM2, while a SRAM2 hardware erase operation is on going: the read access is stalled as long as the erase operation is not completed, and the read value is not 0x0, but the latest value previously read from SRAM2.

#### Workaround

None.

### 2.3.4 PH0/PH1 is controlled by the GPIOH registers when HSE is enabled

#### Description

The PH0 and PH1 GPIO configuration is not bypassed when the HSE is enabled. The oscillator can stop if the I/Os are not configured in analog mode.

**Workaround**

PH0 and PH1 must be configured in analog mode (reset value) when HSE is enabled.

**2.3.5 PWR\_CR4 register write access may not be completed if a Low-power mode is entered just after the write operation****Description**

PWR\_CR4 write access should insert three APB1 wait states, but it does not. Consequently, if a low-power mode is entered just after writing into this register, the PWR\_CR4 register may not be updated before the low-power mode is entered. This can occur in particular when the APB1 clock is prescaled.

**Workaround**

PWR\_CR4 register must be read after the write operation to ensure that the write is done before entering the low-power mode.

**2.3.6 HSI user trim is limited on some samples****Description**

The HSI user trimming step is typically:

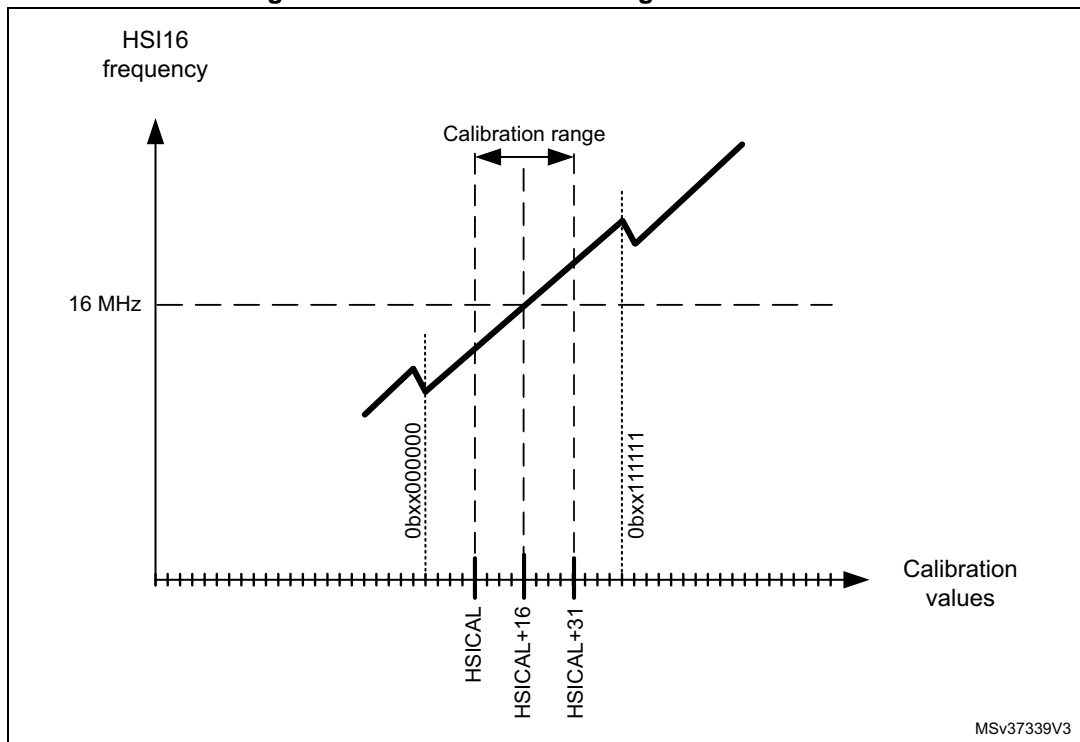
- +0.3% of frequency when the trimming code is not a multiple of 64
- -7% of frequency when the trimming code is a multiple of 64.

As shown in [Figure 1](#), the HSI user trimming makes it possible to add or to subtract up to 16 steps compared to the factory trim value.

If the HSI factory trim value is close to a multiple of 64, with only 16 steps for positive or for negative correction, it is not possible to compensate the 7% drop when the code is a multiple of 64.

Consequently, the user trim correction must not jump over the codes multiple of 64, which can limit the user correction either in positive or in negative direction.

Figure 1. HSI oscillator trimming characteristics



### Workaround

None.

### Fix

This limitation is fixed on Rev 4 products with date code newer than 609.

## 2.3.7 Option byte loading can fail if MSI frequency is greater than 8 MHz

### Description

The option byte loading operation can fail if the MSI clock is ON with a frequency equal or greater than 8 MHz before performing an option byte loading by setting OBL\_LAUNCH bit in the FLASH\_CR register. Some options and engineering bytes values can be corrupted.

### Workaround

If MSI is ON at a frequency equal or higher than 8MHz, It is mandatory to reduce MSI frequency to 4 MHz or less before launching the Option Byte loading operation by setting the OBL\_LAUNCH bit in the FLASH\_CR register.

## 2.3.8 PLL may not lock if VCO is below 96 MHz and temperature is below 0 °C

### Description

The VCO minimum value should be 64 MHz. When the VCO is below 96 MHz and the temperature is below 0 °C the PLL may never lock.

**Workaround**

Program the PLL with VCO at 96 MHz or above.

**2.3.9 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.

**2.3.10 MSIRDY flag issue preventing entry in low power mode****Description**

If the MSI clock is stopped when it is running at high frequency (24 MHz or above), the MSIRDY (MSI ready) flag can stay at value 1 instead of 0.

Once this flag remains set while the MSI clock is OFF, entry in Stop 0, Stop 1, Stop 2, Standby and Shutdown modes is no longer possible (the product is blocked in the low power mode entry phase).

The following factors increase the probability to have this issue:

- a high MSI frequency
- a low  $V_{DD}$  external supply
- a high temperature.

**Workaround**

- The MSI clock can run at any frequency if both conditions below are met:
  - a) the system clock is MSI or PLL fed by MSI when requesting entry in low power mode
  - b) the wakeup clock is MSI
- If the system clock is any other clock than MSI, lower the MSI frequency to 16 MHz (or less) and add a short delay loop (200 ns minimum) before stopping MSI by software. This will ensure that the MSIRDY flag be really 0 before requesting entry in low power mode.
- If the system clock is MSI and the wakeup clock is not MSI, lower the MSI frequency to 16 MHz (or less) and add a short delay loop (200 ns minimum) before requesting entry in low power mode.
- If the system clock is PLL fed by MSI and the wakeup clock is not MSI, the MSI frequency used as PLL input should not exceed 16 MHz. In other words, do not use the PLL fed by MSI as system clock with MSI at high frequency (24 MHz or above) divided by PLLM (ensuring the PLL input is between 4 and 16 MHz).

### 2.3.11 PCROP area within a single Flash memory page becomes unprotected at RDP change from level 1 to level 0

#### Description

With PCROP\_RDP option bit set to 0, the change of RDP from level 1 to level 0 normally results in erasure of Flash memory banks except the Flash memory pages containing PCROP area. The PCROP area remains read-protected.

This operates as expected if the PCROP area crosses the limits of at least one Flash memory page, which is always true if PCROP area size exceeds 2 Kbytes. The limitation occurs if the PCROP area is fully contained within one single Flash memory page. Upon the RDP change from level 1 to level 0, the Flash memory bank with PCROP area is not erased and the read protection of the PCROP area is removed.

#### Workaround

Always define PCROP area so that it crosses the limits of at least one Flash memory page.

### 2.3.12 Data Cache might be corrupted during Flash 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();
```

### 2.3.13 MSI frequency overshoot upon Stop mode exit

#### Description

When

- the system is clocked by the MSI clock, and
  - MSI is selected as system clock source upon wakeup from Stop mode, and
  - a wakeup event occurs only a few system clock cycles before entering Stop mode,
- then upon the exit from Stop mode, the MSI frequency can overshoot above its selected range.

The limitation applies to all Stop modes: Stop 0, Stop 1 and Stop 2.

#### Workaround

There are two possible ways to work the limitation around:

1. Go through the following sequence:
  - a) Switch to HSI
  - b) Shutdown MSI
  - c) Wait for MSIRDY to go low (after six MSI clock cycles)
  - d) Mask\_Interrupts (Set PRIMASK)
  - e) Enter in STOP mode with request to wakeup on MSI
  - f) Enable MSI
  - g) Wait for MSIRDY to go high
  - h) Switch to MSI (required as the system clock remains HSI in case the MCU did not enter Stop due to an early wakeup event)
  - i) Unmask\_Interrupts (Clear PRIMASK)

This workaround offers a shorter wakeup time.
2. Select HSI as clock source upon wakeup from Stop mode. After the wakeup, switch the system clock to MSI.

### 2.3.14 Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled

#### Description

When entering Stop mode with the temperature sensor channel and the associated ADC(s) enabled, the internal voltage reference may be corrupted.

The occurrence of the corruption depends on the supply voltage and the temperature.

The corruption of the internal voltage reference may cause:

- an overvoltage in  $V_{CORE}$  domain
- an overshoot / undershoot of internal clock (LSI, HSI, MSI) frequencies
- a spurious brown-out reset

The limitation applies to Stop 1 and Stop 2 modes.

**Workaround**

Before entering Stop mode

- disable the ADC(s) using the temperature sensor signal as input, and/or
- disable the temperature sensor channel, by clearing the CH17SEL bit of the ADCx\_CCR register.

Disabling both allows consuming less power during Stop mode.

**2.3.15 OPAMP output:  $V_{DDA}$  overconsumption****Description**

An overconsumption can appear on  $V_{DDA}$  in the following conditions:

- a voltage  $V_{PAD}$  is applied on PA3 or PB0 with  $V_{DDA} + 50\text{ mV} < V_{PAD} < V_{DDA} + 600\text{ mV}$
- the OPAMP output is not used (OPAMPx\_VOUT pins are not driven by the OPAMP)
- temperature is below 0 °C

This extra consumption is constant and can reach up to 1 mA.

**Workaround**

None, except avoiding the conditions indicated above.

**2.3.16 Spurious BOR when entering Stop mode after short Run sequence****Description**

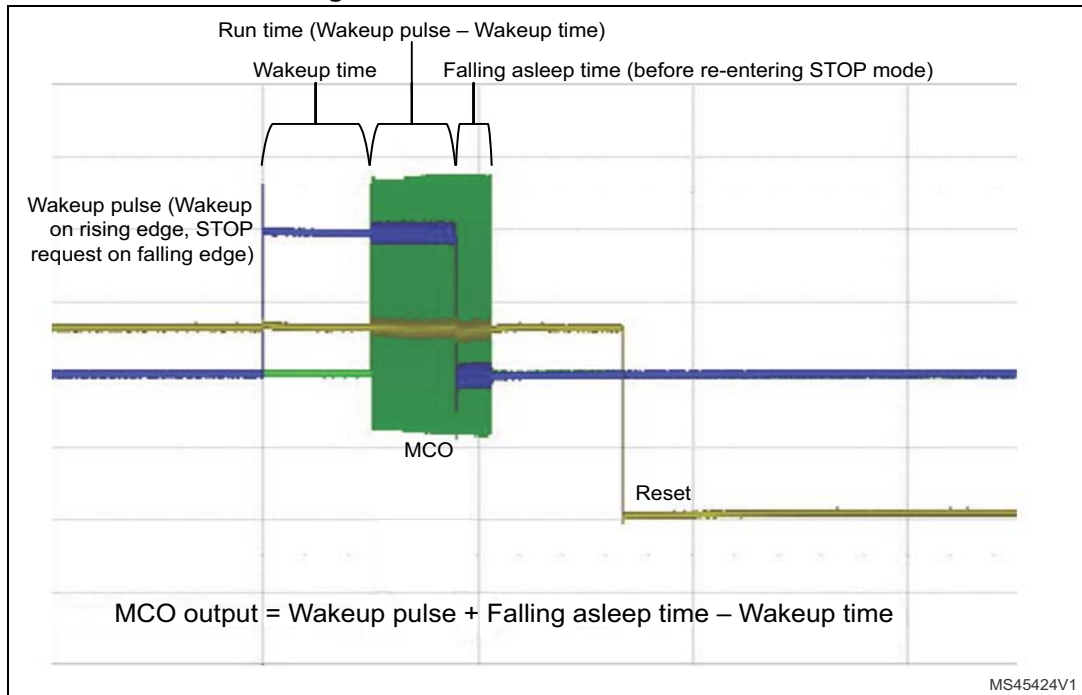
When the MCU wakes up from Stop mode and enters the Stop mode again within a short period of time, a spurious Brown Out Reset (BOR) may be generated.

This limitation depends upon the supply voltage, as indicated in [Table 4](#), and applies to Stop 1 and Stop 2 modes.

**Table 4. Minimum Run time to avoid spurious BORs**

$V_{DD}$ supply voltage (V)	Minimum Run time ( $\mu\text{s}$ )
1.71	15
1.8	13
2.0	11
2.2	9
2.4	8
2.6	6
2.8	5
3.0	3
3.2 and above	2

The minimum Run time defined in [Table 4](#) corresponds to the firmware execution time on the core. Note also that, as shown in [Figure 2](#), the length of the MCO output (green trace) is longer than the firmware execution time.

**Figure 2. Minimum Run time definition****Workaround**

Ensure that the Run time between two Stop sequences is long enough to avoid the generation of a Brown Out Reset.

This can be done by adding a software loop or using a timer to add a delay. The system clock frequency can be reduced during this waiting loop to minimize power consumption.

**2.3.17 HSE long start-up at low voltage****Description**

When  $V_{DD}$  is below 2.7 V, the HSE oscillator may take longer than specified to startup. 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.

**2.3.18 Current injection from  $V_{DD}$  to  $V_{DDA}$  through analog switch voltage booster****Description**

When using  $V_{DDA}$  below 2.4 V and  $V_{DD}$  above 3 V, a small injected current may appear between  $V_{DD}$  and  $V_{DDA}$ . Consequently,  $V_{DDA}$  may rise above its nominal voltage.



This injected current appears when the I/O analog switch voltage booster is disabled (BOOSTEN = 0 in SYSCFG\_CFGR1 register) and any analog peripheral (ADC, COMP, DAC or OPAMP) is enabled.

#### Workaround

To prevent this current injection, the I/O analog switch voltage booster should be enabled (BOOSTEN = 1 in SYSCFG\_CFGR1 register) when  $V_{DDA}$  is below 2.4 V and  $V_{DD}$  is above 3 V.

### 2.3.19 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.

### 2.3.20 FLASH\_ECCR corrupted upon reset or power-down occurring during Flash memory program or erase operation

#### Description

Reset or power-down occurring during a Flash memory location program or erase operation, followed by a read of the same memory location, may lead to a corruption of the FLASH\_ECCR register content.

#### Workaround

Under such condition, erase the page(s) corresponding to the Flash memory location.

## 2.4 FMC

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

#### Description

When performing a burst read access to 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.

### 2.4.2 Data corruption during burst read from FMC synchronous memory

#### Description

A burst read from static memory can be corrupted if the following condition is met:

- one FMC bank is configured in synchronous mode with WAITEN bit set while another FMC bank is used with WAITEN bit cleared,
- a burst read is ongoing from static synchronous memory with wait feature enabled,
- the wait signal (asserted by the synchronous memory) is active,
- the burst read transaction is followed by an access to an FMC dynamic or static bank whose WAITEN bit is disabled in the FMC\_BCRx register.

#### Workaround

1. Set the WAITEN bit (even if not used by the memory) on all FMC static banks.
2. Set the same WAITPOL bit configuration on all static banks.
3. Enable the internal pull-up on PD6 in order to make the FMC\_NWAIT input ready when the synchronous memory is de-selected and the other FMC bank without wait feature selected.

### 2.4.3 FMC NOR/PSRAM controller bank switch with different BUSTURN durations

#### Description

The system hangs when the FMC NOR/PSRAM memory controller switches between two banks while:

- one NOR/PSRAM bank is configured in synchronous mode and the BUSTURN bitfields in FMC\_BTRx/ FMC\_BTWx registers are set to non-zero values,
- another NOR/PSRAM bank is configured in asynchronous multiplexed mode and BUSTURN bitfields are set to 0,
- FMC clock division ratio (CLKDIV) is higher than or equal to four HCLK periods, and
- a single read transaction from the bank operating in synchronous mode is followed by any transaction in another bank operating in asynchronous multiplexed mode.

### Workaround

If several NOR/PSRAM banks are used, the BUSTURN bitfield must be set to a nonzero value for each bank.

## 2.4.4 Read burst access of nine words or more is not supported by FMC

### Description

CPU read burst access equal to or more than 9 registers to FMC returns corrupted data starting from the 9<sup>th</sup> read word. These bursts can only be generated by Cortex<sup>®</sup>-M4 CPU and not by the other masters (i.e not by DMA).

This issue occurs when the stack is remapped on the external memory on the FMC and POP operations are performed with 9 or more registers.

This also occurs when LDM/VLDM operations are used with 9 or more registers.

### Workaround

Stack must be located in the internal SRAM.

IAR<sup>™</sup>-EWARM: EWARM compiler does not generate LDM/VLDM operations with more than 8 registers except when the *setjmp* or *longjmp* functions of C Library are explicitly used in the source code.

Keil<sup>®</sup> MDK: Starting from version 5.06, Arm<sup>®</sup> Compiler implements a patch which limits the number of registers with LDM/VLDM operations. Starting from version 5.16, Keil<sup>®</sup> MDK includes Arm<sup>®</sup> compiler with this patch, addressing the hardware limitations of the STM32L4 FMC burst reads.

Tasking: TASKING compiler starting from version v5.2r1 does not generate LDM instructions with more than 8 registers except when the *setjmp* or *longjmp* functions of C Library are explicitly used in the source code. It generates POP instructions with nine registers.

In assembly written code, LDM/VLDM operations must be limited to eight registers.

## 2.4.5 Spurious clock stoppage with continuous clock feature enabled

### Description

With the continuous clock feature enabled, the FMC\_CLK clock may spuriously stop when:

- the FMC\_CLK clock is divided by 2, and
- an FMC bank set as 32-bit is accessed with a byte access.

*Note:* With static memories, a spuriously stopped clock can be restarted by issuing a synchronous transaction or any asynchronous transaction different from a byte access on 32-bit data bus width.

### Workaround

With the continuous clock feature enabled, do not set the FMC\_CLK clock division ratio to 2 when accessing 32-bit asynchronous memories with byte access.

## 2.5 DMA

### 2.5.1 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.

## 2.6 QUADSPI

### 2.6.1 QUADSPI\_BK1\_IO1 is always an input when the command is sent in Dual- or Quad-SPI mode

#### Description

The IO1 is staying as an input instead of changing to output to drive the command when this last is on two or four lines and there is no address, no data and no alternate byte in the frame, or they exist on one line.

Therefore it is not possible to use Dual-/Quad-mode for the command phase.

This bug affects Numonyx's Dual-/Quad-instruction mode.

#### Workaround

The Dual-/Quad-instruction mode of Micron or Numonyx Flash memories is working fine when user does not send only a command in the frame. For example, the Write enable command can be avoided by writing directly the status register of the Flash memory instead. Otherwise the application can reset the memory through the reset pin (if it exists) to disable the Dual-/Quad-instruction mode before each time it needs to send a frame that contains only the command phase.

### 2.6.2 Hard fault is not generated in case of out-of-range memory-mapped access

#### Description

A hard fault is not generated when the Cortex® CPU executes an instruction that reads from an out-of-range memory-mapped address. Consequently the CPU effectively skips the instruction (the destination register is not modified) and continues executing without giving a

hard fault.

#### Workaround

None.

### 2.6.3 Extra data written in the FIFO at the end of a read transfer

#### Description

When all the conditions listed below are gathered:

- QUADSPI is used in indirect mode
- QUADSPI clock is AHB/2 (PRESCALER = 0x01 in the QUADSPI\_CR)
- QUADSPI is in quad mode (DMODE = 0b11 in the QUADSPI\_CCR)
- QUADSPI is in DDR mode (DDRM = 0b1 in the QUADSPI\_CCR)

An extra data is incorrectly written in the FIFO when a data is read at the same time that the FIFO gets full at the end of a read transfer.

#### Workaround

One of the two workarounds listed below can be done:

- Read out the extra data until the BUSY flag goes low and discard it.
- Request an abort after reading out all the correct received data from FIFO in order to flush FIFO and have the busy low. Abort will keep the last register configuration (set the ABORT bit in the QUADSPI\_CR).

### 2.6.4 First nibble of data is not written after dummy phase

#### Description

The first nibble of data to be written to the external Flash memory is lost in the following conditions:

- QUADSPI is used in indirect write mode
- And at least one dummy cycle is used

#### Workaround

Use alternate bytes instead of dummy phase to add latency between address phase and data phase. This works only if the number of dummy cycles corresponds to a multiple of eight bits of data.

Example: to generate

- one dummy cycle: send one alternate byte, possible only in four data lines DDR mode
- two dummy cycles: send one alternate byte in four data lines SDR mode
- four dummy cycles: send two alternate bytes in four data lines SDR mode, or send one alternate byte in two data lines SDR mode
- eight dummy cycles: send one alternate byte in one data line SDR mode.

## 2.6.5 Wrong data can be read in memory-mapped after an indirect mode operation

### Description

Wrong data can be read with the first memory-mapped read request when in the following condition:

- Quad-SPI peripheral entered memory-mapped mode with both LSB bits in the address register QUADSPI\_AR[1:0] not reset.

### Workaround

QUADSPI\_AR register must be reset just before entering memory-mapped mode.

Depending on the current Quad-SPI operating mode, one of the two workarounds listed below can be used:

1. Indirect read mode: reset address register then do an abort request to stop reading and clear busy bit.  
Then enter to memory-mapped mode.
2. Indirect write mode: reset the address register then enter to memory-mapped mode

*Note:* User should take care to not write to QUADSPI\_DR register after resetting address register.

## 2.6.6 Memory-mapped read operations may fail when timeout counter is enabled

### Description

In Memory-mapped mode when the TC is enabled, the QuadSPI peripheral can hang and memory-mapped read operations fail. The hang occurs if the timeout flag TOF is set at the same clock edge of a new memory-mapped read request.

### Workaround

The timeout counter must be disabled.

To rise the chip select high, the application can do an abort at the end of each memory-mapped read operation.

## 2.7 ADC

### 2.7.1 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 one stage: a new context written before the previous context's consumption will lead to a queue overflow and will be ignored. Consequently, the ADC must be stopped before programming the JSQR register.

#### Workaround

None.

## 2.7.2 DMA2 channels 2 and 3 (respectively) cannot be used when the ADC2 and ADC3 requests are selected on DMA2 channels 4 and 5

### Description

When the DMA2 channel 4 is used for the ADC2 requests (C4S=0000 in the DMA2\_CSELR register) it is also needed to select request 0 for DMA2 channel 2 (C2S=0000 in the DMA2\_CSELR register). The consequence is that the DMA2 channel 2 cannot be used by another peripheral.

When the DMA2 channel 5 is used for the ADC3 requests (C5S=0000 in the DMA2\_CSELR register) it is also needed to select request 0 for DMA2 channel 3 (C3S=0000 in the DMA2\_CSELR register). The consequence is that the DMA2 channel 3 cannot be used by another peripheral.

### Workaround

Select the DMA1 channel 2 for the ADC2 DMA requests or use other available mapping for the peripherals mapped on DMA2 channel 2.

Select the DMA1 channel 3 for the ADC3 DMA requests or use other available mapping for the peripherals mapped on DMA2 channel 3.

## 2.7.3 Wrong ADC conversion results when delay between calibration and first conversion or between two consecutive conversions is too long

### Description

When the delay between two consecutive ADC conversions is higher than 1 ms the result of the second conversion might be incorrect. The same issue occurs when the delay between the calibration and the first conversion is higher than 1 ms.

### Workaround

When the delay between two ADC conversions is higher than the above limit, perform two ADC consecutive conversions in single, scan or continuous mode: the first is a dummy conversion of any ADC channel. This conversion should not be taken into account by the application.

## 2.7.4 Burst read or write accesses are not supported by the ADC

### Description

The ADC does not support LDM, STM, LDRD and STRD instructions for successive multiple-data read and write accesses to a contiguous address block.

### Workaround

Prevent compilers from generating LDM, STM, LDRD and STRD instructions. In general, this can be achieved organizing the source code to avoid consecutive read or write accesses to neighboring addresses in lower-to-higher order. In cases where consecutive read or write accesses to neighboring addresses cannot be avoided, order the source code so that it accesses higher address first.

### 2.7.5 Unexpected end of transfer on DMA1 when using DMA1 channel 2 for ADC2 while DMA2 channel 4 is also used

#### Description

When the DMA1 channel 2 is used for the ADC2 requests and the DMA2 channel 4 is used by another peripheral (for instance the SPI1\_TX requests), the end of transfer and acknowledge signals from the DMA2 can be received by the ADC2 while it should only be received by the peripheral using this DMA2 channel 4 (for instance the SPI1\_TX). The consequence is that the DMA1 transfer will be interrupted earlier than expected. This issue only occurs when the DMA2 channel 2 is configured for the request 0 (reset configuration).

#### Workaround

When this limitation is observed, the DMA2 channel 2 should not stay configured for request 0 (reset value), but should be configured for any other request (1 to 7) even if it is useless for the application. This is done by configuring DMA2\_CSELR register bits [7:4] to a value different from 0000.

*Note:* There is no need to enable the DMA2 Channel 2.

### 2.7.6 Unexpected end of transfer on DMA1 when using DMA1 channel 3 for ADC3 while DMA2 channel 5 is also used

#### Description

When the DMA1 channel 3 is used for the ADC3 requests and the DMA2 channel 5 is used by another peripheral (for instance the UART4\_RX requests), the end of transfer and acknowledge signals from the DMA2 can be received by the ADC3 while it should only be received by the peripheral using this DMA2 channel 5 (for instance the UART4). The consequence is that the DMA1 transfer will be interrupted earlier than expected. This issue only occurs when the DMA2 channel 3 is configured for the request 0 (reset configuration).

#### Workaround

When this limitation is observed, the DMA2 channel 3 should not stay configured for request 0 (reset value), but should be configured for any other request (1 to 7) even if it is useless for the application. This is done by configuring DMA2\_CSELR register bits [11:8] to a value different from 0000. Note that there is no need to enable the DMA2 Channel 3. If the ADC1 was using the DMA2 Channel 3, it is necessary to remap the ADC1 DMA transfers to the DMA1 Channel 1.

### 2.7.7 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 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.

**Table 5. Example of IQM algorithm**

Data	Sample												Mean
	1	2	3	4	5	6	7	8	9	10	11	12	
Acquired	17.20	10.92	9.56	2.12	9.82	10.72	10.60	3.50	9.46	9.78	9.50	1.10	8.69
Sorted	1.10	2.12	3.50	9.46	9.50	9.56	9.78	9.82	10.60	10.72	10.92	17.20	8.69
Truncated	-	-	-	9.46	9.50	9.56	9.78	9.82	10.60	-	-	-	9.79

The measurement result after the IQM post-processing in the example detailed in [Table 5](#) is 9.79. For consistent results, use a minimum of 64 samples.

It is recommended to optimize the code performing the sort task so that its processing power requirements are minimized.

### 2.7.8 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 COMP

### 2.8.1 Comparators propagation delay is longer than expected for input steps higher than 200 mV

#### Description

[Table 6](#) summarizes the comparators propagation delay values. The propagation delay for steps higher than 200 mV is out of targeted specification.

Table 6. COMP characteristics<sup>(1)</sup>

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
t <sub>D</sub>	Propagation delay for 200 mV step with 100 mV overdrive	High-speed mode	V <sub>DDA</sub> ≥ 2.7 V	-	55	80	ns
			V <sub>DDA</sub> < 2.7 V	-	65	100	
		Medium mode	V <sub>DDA</sub> ≥ 2.7 V	-	0.55	0.9	μs
			V <sub>DDA</sub> < 2.7 V	-	0.65	1.0	
		Ultralow-power mode	V <sub>DDA</sub> ≥ 2.7 V	-	5	12	
			V <sub>DDA</sub> < 2.7 V	-	5	12	
	Propagation delay for step > 200 mV with 100 mV overdrive on positive inputs	High-speed mode	-	-	0.1	1.0	μs
		Medium mode	-	-	2	3	
		Ultralow-power mode	-	-	8	24	

1. Data guaranteed by design, not tested in production, unless otherwise specified.

### Workaround

None.

## 2.8.2 Comparators output cannot be configured in open-drain

### Description

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

### Workaround

None.

## 2.8.3 COMP1 and COMP2 configuration lost with software reset

### Description

The Comparators 1 and 2 control and status registers (COMP1\_CSR and COMP2\_CSR) should be protected from software changes when the LOCK bit is set. Therefore these registers should be reset only by a system reset. However, it is possible to reset these registers by setting SYSCFGRST in the RCC\_APB2RSTR.

### Workaround

None.

## 2.9 DFSDM

### 2.9.1 RDATACH[2:0] status bits are not implemented

#### Description

RDATACH[2:0] "Regular channel most recently converted" status bits are not implemented in the DFSDM data register for the regular channel (DFSDMx\_RDATAR).

#### Workaround

None. These bits will be implemented in next silicon release.

### 2.9.2 New regular channel selection taken into account at the end of an injected conversion when a regular conversion is pending

#### Description

When a regular channel conversion is on going and is interrupted by an injected channel conversion: if the regular channel is changed on the fly during the injected channel conversion, the new regular channel selection is taken into account at the end of the injected conversion.

#### Workaround

None: do not change the regular channel selection on the fly when regular continuous conversions are requested.

### 2.9.3 DFSDM triggers from timers can be missed in specific conditions

#### Description

Triggers from timers to DFSDM can be missed when all the conditions listed below occur:

- the DFSDM clock is the APB2 clock PCLK2 (DFSDMSEL=0 in the RCC\_CCIPR register)
- the DFSDM is triggered by TIM3\_TRGO, TIM4\_TRGO, TIM6\_TRGO or TIM7\_TRGO
- the APB2 frequency is smaller than the APB1 frequency

#### Workaround

Select the System clock as DFSDM clock (DFSDMSEL=1 in the RCC\_CCIPR register).

## 2.10 TSC

### 2.10.1 Inhibited acquisition in short transfer phase configuration

#### Description

The input buffer of the I/O is normally masked outside the transfer window time, then sampled twice before being checked for acquisition. Such check is normally performed on the last TSC clock cycle of the transfer of charge phase. When the transfer of charge duration is less than three cycles the acquisition is inhibited.

**Workaround**

The following configurations are forbidden:

1. The PGPSC[2:0] field set to 000 and the CTPL[3:0] field to 0000 or 1111
2. The PGPSC[2:0] field set to 111 and the CTPL[3:0] field to 0000

**2.11 LPTIM****2.11.1 Low power timer 1 (LPTIM1) outputs cannot be configured in open-drain****Description**

LPTIM1 outputs are always forced in push-pull mode whatever the GPIO output type configuration bit value.

**Workaround**

None.

**2.11.2 MCU may remain stuck in LPTIM interrupt when entering Stop mode****Description**

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

When the firmware clears the LPTIM\_CR.ENABLE bit 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 LPTIMx\_CR register. Instead, reset the whole LPTIMx peripheral via the RCC controller by setting and resetting its respective LPTIMxRST bit in RCC\_APB1RSTRz register.

**2.12 RTC****2.12.1 Spurious tamper detection when disabling the tamper channel****Description**

If the tamper detection is configured for detection on falling edge event (TAMPFLT=00 and TAMPxTRG=1) and if the tamper event detection is disabled when the tamper pin is at high level, a false tamper event is detected.

### Workaround

The false tamper event detection cannot be avoided, but the backup registers erase can be avoided by setting TAMPxNOERASE bit before clearing TAMPxE bit. The two bits must be written in two separate RTC\_TAMPCR write accesses.

## 2.12.2 RTC calendar registers are not locked properly

### Description

When reading the calendar registers with BYPSHAD=0, the RTC\_TR and RTC\_DR registers may not be locked after reading the RTC\_SSR register. This happens if the read operation is initiated one APB clock period before the shadow registers are updated. This can result in a non-consistency of the three registers. Similarly, RTC\_DR register can be updated after reading the RTC\_TR register instead of being locked.

### Workaround

- Use BYPSHAD = 1 mode (Bypass shadow registers), or
- if BYPSHAD = 0, read SSR again after reading SSR/TR/DR to confirm that SSR is still the same, otherwise read the values again.

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

### Description

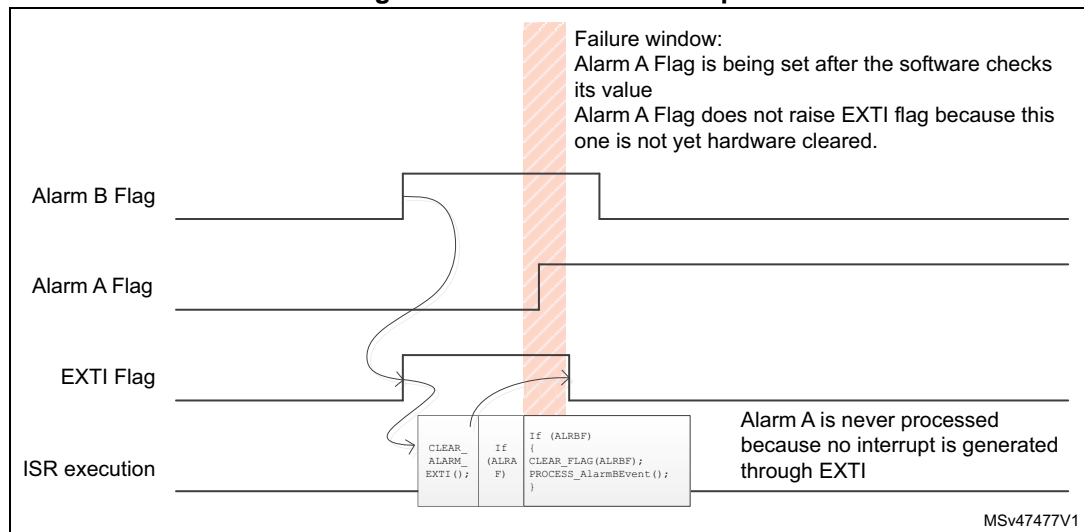
One RTC interrupt can mask another RTC interrupt if both share the same EXTI configurable line, such as the RTC Alarm A and Alarm B, of which the event flags are OR-ed to the same EXTI line (refer to the EXTI line connections table in the Extended interrupt and event controller (EXTI) section of the 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 3. Masked RTC interrupt



### 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:

```

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 */
    }
}

```

## 2.12.4 RTC\_OUT on PB2 and RTC\_REFIN on PB15 are not functional in Stop 2 mode

### Description

Contrary to the description provided in table “RTC functions over modes” of reference manual RM0351, both RTC\_OUT on PB2 and RTC\_REFIN on PB15 are not functional in Stop 2 mode.

### Workaround

Enter Stop 1 mode instead of Stop 2 mode to use RTC\_OUT on PB2 and RTC\_REFIN.

If Stop 2 mode usage is required for power consumption reasons:

- use PC13 standard output (rather than PB2) for RTC\_OUT
- there is no workaround for RTC\_REFIN.

## 2.13 I2C

### 2.13.1 I2C Fast-mode Plus drive is not available on all SDA/SCL I/Os

#### Description

Only PB6/7/8/9/13/14, PC0/1 and PG7/8 can effectively be configured in I2C Fm+ driving mode. Setting I2C1\_FMP, I2C2\_FMP and I2C3\_FMP in the SYSCFG\_CFGR1 register has no effect on PB10, PB11, PF0, PF1, PG13 and PG14.

*Note: When using the I/O without 20 mA Fm+ drive, it is still possible to reach 1 MHz Fm+ speed, with limited bus load.*

#### Workaround

None.

### 2.13.2 Wrong behavior related with MCU Stop mode when wakeup from Stop mode by I2C peripheral is disabled

#### Description

When wakeup from Stop mode by I2C peripheral is disabled (WUPEN = 0) and the MCU enters Stop mode while a transaction is on-going on the I2C bus, the following wrong operations may occur:

1. BUSY flag may be 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. This failure may occur in master mode of the I2C peripheral used in multi-master I2C-bus environment.
2. If I2C-bus clock stretching is enabled in I2C peripheral (NOSTRETCH = 0), the I2C peripheral may pull SCL low as long as the MCU remains in Stop mode, suspending all I2C-bus activity during that time. This may occur when the MCU enters Stop mode during the address phase of an I2C-bus transaction, in low period of SCL. This failure may occur in slave mode of the I2C peripheral or, in master mode of the I2C peripheral used in multi-master I2C-bus environment. Its probability depends on the timing.

**Workaround**

Disable the I2C peripheral (PE=0) before entering Stop mode and re-enable it in Run mode.

**2.13.3 Wrong data sampling when data set-up time ( $t_{\text{SU;DAT}}$ ) is smaller than one I2CCLK period****Description**

The I2C bus specification and user manual specify a minimum data set-up time ( $t_{\text{SU;DAT}}$ ) at:

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

The I2C SDA line is not correctly sampled when  $t_{\text{SU;DAT}}$  is smaller than one I2CCLK (I2C clock) period: the previous SDA value is sampled instead of the current one. This can result in a wrong slave address reception, a wrong received data byte, or a wrong received acknowledge bit.

**Workaround**

Increase the I2CCLK frequency to get I2CCLK period smaller than the transmitter minimum data set-up time. Or, if it is possible, increase the transmitter minimum data set-up time.

**2.13.4 Spurious Bus Error detection in master mode****Description**

In master mode, a bus error can be detected by mistake, so the BERR flag can be wrongly raised in the status register. This will generate a spurious Bus Error interrupt if the interrupt is enabled. A bus error detection has no effect on the transfer in master mode, therefore the I2C transfer can continue 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 on-going transfer can be handled normally.

**2.13.5 I2C3 analog filters requires that both PC0/PC1 or both PG7/PG8 are configured as I2C3 alternate function****Description**

I2C3 analog filters can be enabled for PC0 and/or PC1 only if both IOs are configured in I2C3 Alternate Function mode. For example if you use PC0 for clock and PB4 for data, PC0 filter can be enabled only if PC1 is configured in I2C3 Alternate Function mode too.

I2C3 filter can be enabled for PG7 and/or PG8 only if both IOs are configured in I2C3 Alternate Function mode. For example if you use PG7 for clock and PC1 for data, PG7 filter can be enabled only if PG8 is configured in I2C3 Alternate Function mode too.



**Workaround**

Use both PC0/PC1 as I2C3 alternate functions or use both PG7/PG8 as I2C3 alternate functions if the analog filter is needed.

**2.13.6 10-bit master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave****Description**

In master mode, the master automatically sends a STOP bit when the slave has not acknowledged a byte during the address transmission.

In 10-bit addressing mode, if the first part of the 10-bit address (c5-bit header + 2 MSBs of the address + direction bit) has not been acknowledged by the slave, the STOP bit is sent but the START bit is not cleared and the master cannot launch a new transfer.

**Workaround**

When the I2C is configured in 10-bit addressing master mode and the NACKF status flag is set in the I2C\_ISR register while the START bit is still set in I2C\_CR2 register, 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.

**2.13.7 Last received byte can be lost when using Reload mode with NBYTES > 1****Description**

The limitation can occur in master mode when reload mode is used (needed for transferring more than 255 bytes), or in slave byte control mode (SBC = 1 in the I2C\_CR1 register). The limitation occurs only when NBYTES > 1.

In Reload mode (RELOAD=1 in the I2C\_CR2 register) with NBYTES programmed with a value N in the I2C\_CR2, the Transfer Complete Reload flag (TCR) is set in the I2C\_ISR register when the last byte is received in the shift register, even if not yet transferred in the Receive Data Register because the byte N-1 is not yet read.

The last received data is definitively lost (never transferred in the Data Register) if the data N-1 is read between 0 and four APB clock cycles before the TCR flag is set in the I2C\_ISR register.

**Workaround**

In slave byte control mode: Use the Reload mode with NBYTES = 1

In master mode: Do not use the Reload mode. If the number of bytes to be transferred is larger than 255 bytes, the total transfer should be split in several transfers, each one not exceeding 255 bytes, separated by Repeated Start conditions.

Note that the use of DMA could manage that data N-1 is always transferred before the four APB cycles preceding the TCR flag. However this must be evaluated carefully for each application, depending on the bus bandwidth, maximum latency, and DMA channel priority.

### 2.13.8 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 writing the I2C\_CR2 register with its START 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):
  - a) Set the ADDRCONF bit.
  - b) Before Stop condition occurs on the bus, write I2C\_CR2 with the START bit low.
- Slave byte control mode (SBC = 1):
  - a) Write I2C\_CR2 with the slave transfer configuration and the START bit low.
  - b) Wait for longer than three I2C kernel clock cycles.
  - c) Set the ADDRCONF bit.
  - d) 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 USART

### 2.14.1 Start bit detected too soon when sampling for NACK signal from the smartcard

#### Description

In the ISO7816, when a character parity error is incorrect, the Smartcard receiver should transmit a NACK error signal at  $(10.5 \pm 0.2)$  etu after the character START bit falling edge. In this case, the USART transmitter should be able to detect correctly the NACK signal by sampling at  $(11.0 \pm 0.2)$  etu after the character START bit falling edge.

The USART peripheral used in Smartcard mode doesn't respect the  $(11 \pm 0.2)$  etu timing, and when the NACK falling edge arrives at 10.68 etu or later, the USART might misinterpret this transition as a START bit even if the NACK is correctly detected.

#### Workaround

None.

### 2.14.2 Break request can prevent the Transmission Complete flag (TC) from being set

#### Description

After the end of transmission of a data (D1), the Transmission Complete (TC) flag will not be set in the following conditions:

- CTS hardware flow control is enabled;
- D1 is being transmitted;
- abreak transfer is requested before the end of D1 transfer;
- nCTS is de-asserted before the end of transfer of D1.

#### Workaround

If the application needs to detect the end of transfer of the data, the break request should be done after making sure that the TC flag is set.

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

#### Description

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

#### Workaround

Configure the I/O used for nRTS as an alternate function after setting the UE and RE bits.

## 2.15 LPUART

### 2.15.1 Low power UART1 (LPUART1) outputs cannot be configured in open-drain

#### Description

LPUART1 outputs are always forced in push-pull mode whatever the GPIO output type configuration bit value.

#### Workaround

None.

## 2.16 SPI

### 2.16.1 BSY bit may stay high at the end of a data transfer at slave mode

#### Description

In slave mode, BSY bit is not reliable to handle the end of data frame transaction due to some bad synchronization between the CPU clock and external SCK clock provided by master. Sporadically, the BSY bit is not cleared at the end of a data frame transfer. As a consequence, it is not recommended to rely on BSY bit before entering low power mode or modifying the SPI configuration.

#### Workaround

To ensure BSY bit flag is always reset at the end of a frame communication, disable the SPI by software before the communication is finished by following sequence:

- write last data to data register to transmit the data,
- poll TXE till it becomes high to ensure the data transfer has started,
- disable SPI by clearing SPE while the last data transfer is still on going,
- poll the BSY bit till it becomes low.

## 2.17 SWPMI

### 2.17.1 SUSPENDED mode entry delayed

#### Description

When activating the SWPMI by setting the SWPEN bit in SWPMI\_CR register, the SWPMI will rise the SWPMI\_IO to 1.8V, send a short transition sequence and 14 idle bits before switching to SUSPENDED mode. As a consequence, the SRF flag is set and the SUSP flag is cleared in SWPMI\_ISR register.

#### Workaround

None.

### 2.17.2 SUSPENDED mode never entered

#### Description

When activating the SWPMI by setting the SWPEN bit in SWPMI\_CR register, the SWPMI will generate the following sequence in loop: rise the SWPIO, send a short transition sequence and 14 idle bits. As a consequence, SWP stays in ACTIVATED state, and never switch to SUSPENDED state.

#### Workaround

Keep SWPMI1SEL bit cleared in RCC\_CCIPR register to select PCLK1 as SWPMI clock source, and configure the PCLK1 prescaler to feed the SWPMI with a clock frequency below or equal to 8 MHz.

### 2.17.3 SRF flag not set

#### Description

If the SWPMI1SEL bit is set in RCC\_CCIPR register to select HSI as SWPMI clock source, when receiving a resume by slave, the SRF flag may not be set. Nevertheless, the SWPMI is switching correctly from SUSPENDED to ACTIVATED when receiving a RESUME by slave. Therefore frame reception is not impacted.

#### Workaround

None.

### 2.17.4 SWP SUSPENDED mode not supported when STM32L4 is in Stop 0 or Stop 1 mode

#### Description

STM32L4 cannot enter Stop 0 or Stop 1 mode if SWPMI is activated and is in SUSPENDED state.

#### Workaround

Deactivate the SWP bus before requesting entry in Stop 0 or Stop 1 mode. Refer to the SWPMI section in the product reference manual for the deactivation procedure.

### 2.17.5 SWPMI\_IO transceiver bypass mode is not functional

#### Description

When the internal SWPMI transceiver is bypassed by setting the SWP\_TBYP bit in the SWPMI\_OR register, SWPMI1\_RX mapped on PB14 is forced in output mode instead of input mode.

#### Workaround

None. The internal transceiver must be used.

## 2.18 SDMMC

### 2.18.1 Wrong CCRCFAIL status after a response without CRC is received

#### Description

The CRC is calculated even if the response to a command does not contain any CRC field. As a consequence, after the SDIO command IO\_SEND\_OP\_COND (CMD5) is sent, the CCRCFAIL bit of the SDMMC\_STA register is set.

#### Workaround

The CCRCFAIL bit in the SDMMC\_STA register shall be ignored by the software. CCRCFAIL must be cleared by setting CCRCFAILC bit of the SDMMC\_ICR register after reception of the response to the CMD5 command.

### 2.18.2 Clock division per 255 is not possible

#### Description

When CLKDIV divide factor in SDMMC\_CLKCR register is equal to 255, the SDMMC\_CK clock output is not provided.

#### Workaround

Do not use CLKDIV value equals to 255: use clock divider factors from 0 to 254.

### 2.18.3 Wait for response bits “10” configuration does not work correctly

#### Description

The Wait for response bits configuration “10” (WAITRESP in the SDMMC\_CMD register) does not work correctly. When WAITRESP bits are programmed to '10' Command Path State Machine (CPSM) waits for a non-existing response.

#### Workaround

Do not use WAITRESP value equal to “10” when sending a command without a response.

Use WAITRESP value equal to “00” to indicate to SDMMC CPSM that no response is expected.

### 2.18.4 MMC stream write of less than 8 bytes does not work correctly

When SDMMC host starts a stream write (WRITE\_DAT\_UNTIL\_STOP CMD20), the number of bytes to transfer is not known by the card.

The card will write data from the host until a STOP\_TRANSMISSION (CMD12) command is received.

The WAITPEND bit 9 of SDMMC\_CMD register is set to synchronize the sending of the STOP\_TRANSMISSION (CMD12) command with the data flow.

When WAITPEND is set, the transmission of this command stays pending until 50 data bits including STOP bit remain to transmit.

For a stream write of less than 8 bytes, the STOP\_TRANSMISSION (CMD12) command should be started before the data transfer starts. Instead of this, the data write and the command sending are started simultaneously.

It implies that when less than 8 bytes have to be transmitted, (8 - DATALENGTH) bytes are programmed to 0xFF in the card after the last byte programmed (where DATALENGTH is the number of data bytes to be transferred).

#### **Workaround**

Do not use stream write WRITE\_DAT\_UNTIL\_STOP (CMD20) with a DATALENGTH less than 8 bytes.

Use set block length (SET\_BLOCKLEN: CMD16) followed by single block write command (WRITE\_BLOCK: CMD24) instead of stream write (CMD20) with desired block length.

## **2.19 bxCAN**

### **2.19.1 bxCAN time-triggered mode is not supported**

#### **Description**

The time-triggered communication mode described in the reference manual is not supported, and so time-stamp values are not available. TTCM bit must be kept cleared in the CAN\_MCR register (time-triggered communication mode disabled).

#### **Workaround**

None.

## **2.20 OTG\_FS**

### **2.20.1 Suspend mode robustness marginality after receiving corrupted packet from Host**

#### **Description**

Once the early suspend interrupt is set after 3 ms of USB idleness, the corrupted packet with missing EOP unexpectedly sent by the Host stops the suspend timer and when the bus becomes idle again, suspend/early suspend interrupts are no more generated. No error flag is set and it is not possible to issue remote wake-up to the Host. Device restarts operate normally only upon reception of next EOP from Host.

#### **Workaround**

If such scenario is believed possible, implement a timeout check started once early suspend interrupt is detected. If within next 4 ms there is no suspend, resume, early suspend or SOF detected, it means that the suspend timer is blocked and need to be recovered by reinitializing the USB peripheral by a soft reset (CSRST bit in OTG\_FS\_GRSTCTL). After that, when the USB system is eventually woken up from the Suspend mode (either by Host or by Device generated remote wake-up), it is necessary to generate a soft disconnect & re-connect, as the enumeration status is lost by the soft reset.

**2.20.2 Bits OTG\_FS\_GLPMCFG[6:2] are not write-protected in Device mode****Description**

Bits OTG\_FS\_GLPMCFG[6:2] are not write-protected in Device mode as expected, but could be modified by application software. In Device mode, the core updates the fields with the values received in the LPM Token. When some read-modify-write access is performed to OTG\_FS\_GLPMCFG register during the time of LPM token exchange, the application software could over-write the values written by OTG\_FS core with the old value.

**Workaround**

Program the LPMACK bit in OTG\_FS\_GLPMCFG register to NYET response before updating OTG\_FS\_GLPMCFG register while USB transfers are in progress. Set the LPMACK bit back to ACK response when updating is finished.

**2.20.3 L1 exit with simultaneous Device initiated and Host initiated Resume results in minimum Device wake-up time of 3 ms****Description**

When the device application initiates the remote wakeup by setting the RWUSIG bit in OTG\_FS\_DCTL register while the host is initiating resume at the same time, further LPM tokens sent by the Host within next 3 ms are ignored (the device does not enter L1 on any LPM token acknowledgment). If the host has sent an LPM token during these 3 ms and is acknowledged by the device core, then host enters L1 whereas device does not enter L1. Instead, the device enters normal Suspend mode due to 3 ms of bus inactivity.

**Workaround**

Program the LPMACK bit in OTG\_FS\_GLPMCFG register to NYET response before initiating remote wakeup and set it back to ACK only after 3 ms.

**2.20.4 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 will corrupt 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.



### 3 Revision history

**Table 7. Document revision history**

Date	Revision	Changes
26-Jan-2016	1	Initial release.
07-Dec-2016	2	<p>Updated <a href="#">Table 3: Summary of device limitations</a> and added footnote 1.</p> <p>Added <a href="#">Section 2.3.11: PCPROP area within a single Flash memory page becomes unprotected at RDP change from level 1 to level 0</a>, <a href="#">Section 2.3.12: Data Cache might be corrupted during Flash Read While Write operation</a>, <a href="#">Section 2.3.13: MSI frequency overshoot upon Stop mode exit</a>, <a href="#">Section 2.3.14: Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled</a>, <a href="#">Section 2.6.4: First nibble of data is not written after dummy phase</a>, <a href="#">Section 2.6.5: Wrong data can be read in memory-mapped after an indirect mode operation</a>, <a href="#">Section 2.7.5: Unexpected end of transfer on DMA1 when using DMA1 channel 2 for ADC2 while DMA2 channel 4 is also used</a>, <a href="#">Section 2.7.6: Unexpected end of transfer on DMA1 when using DMA1 channel 3 for ADC3 while DMA2 channel 5 is also used</a>, <a href="#">Section 2.13.6: 10-bit master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave</a> and <a href="#">Section 2.10: TSC</a>.</p> <p>Updated <a href="#">Section 2.3.6: HSI user trim is limited on some samples</a>, <a href="#">Section 2.3.10: MSIRDY flag issue preventing entry in low power mode</a> and <a href="#">Section 2.19.1: bxCAN time-triggered mode is not supported</a>.</p> <p>Removed former <a href="#">Section 2.16.2: Calibration procedure does not work in PGA mode</a>.</p>
26-Apr-2017	3	<p>Updated <a href="#">Table 3: Summary of device limitations</a>.</p> <p>Updated <a href="#">Section 2.3.13: MSI frequency overshoot upon Stop mode exit</a>.</p> <p>Added <a href="#">Section 2.3.15: OPAMP output: VDDA overconsumption</a>, <a href="#">Section 2.3.16: Spurious BOR when entering Stop mode after short Run sequence</a>, <a href="#">Section 2.8.7: START bit is not cleared when the address is not acknowledged by the slave device</a> and <a href="#">Section 2.20.4: Data FIFO gets corrupted if the write sequence to the Transmit FIFO is interleaved with other OTGFS register access</a>.</p>
05-Sep-2017	4	<p>Replaced former <a href="#">Silicon identification</a> with <a href="#">Applicability</a>, and former <a href="#">Table 1: Device identification</a> with <a href="#">Table 2: Device variants</a>.</p> <p>Updated <a href="#">Section 1: Summary of device limitations</a>.</p> <p>Updated <a href="#">Table 3: Summary of device limitations</a>.</p> <p>Removed former <a href="#">Section 1: ARM® 32-bit Cortex®-M4 FPU core limitations</a> and former <a href="#">Section 2.8.7: START bit is not cleared when the address is not acknowledged by the slave device</a>.</p> <p>Added <a href="#">Section 2.13.7: Last received byte can be lost when using Reload mode with NBYTES &gt; 1</a> and <a href="#">Section 2.11.2: MCU may remain stuck in LPTIM interrupt when entering Stop mode</a>.</p>

Table 7. Document revision history (continued)

Date	Revision	Changes
19-Jun-2018	5	<p>Updated <a href="#">Section 2.1: Core</a>, <a href="#">Section 2.1.1: Interrupted loads to stack pointer can cause erroneous behavior</a>, <a href="#">Section 2.4.1: Dummy read cycles inserted when reading synchronous memories</a>, <a href="#">Section 2.4.2: Data corruption during burst read from FMC synchronous memory</a> and <a href="#">Section 2.4.3: FMC NOR/PSRAM controller bank switch with different BUSTURN durations</a>.</p> <p>Updated <a href="#">Table 3: Summary of device limitations</a>.</p> <p>Rearranged description of limitations according to order of reference manual RM0351.</p> <p>Removed former <a href="#">Table 4: Cortex®-M4 FPU core limitations and impact on microcontroller behavior</a>.</p> <p>Added <a href="#">Section 2.1.2: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used</a>, <a href="#">Section 2.1.3: Store immediate overlapping exception return operation might vector to incorrect interrupt</a>, <a href="#">Section 2.2: FW</a> and its subsections, <a href="#">Section 2.3.17: HSE long start-up at low voltage</a>, <a href="#">Section 2.3.18: Current injection from VDD to VDDA through analog switch voltage booster</a>, <a href="#">Section 2.4.5: Spurious clock stoppage with continuous clock feature enabled</a>, <a href="#">Section 2.5: DMA</a> and its subsection, <a href="#">Section 2.6.6: Memory-mapped read operations may fail when timeout counter is enabled</a>, <a href="#">Section 2.7.7: Spurious temperature measurement due to spike noise</a>, <a href="#">Section 2.7.8: Writing ADCx_JSQR when JADCSTART and JQDIS are set might lead to incorrect behavior</a>, <a href="#">Section 2.12.2: RTC calendar registers are not locked properly</a>, <a href="#">Section 2.12.3: RTC interrupt can be masked by another RTC interrupt</a>, <a href="#">Section 2.12.4: RTC_OUT on PB2 and RTC_REFIN on PB15 are not functional in Stop 2 mode</a> and <a href="#">Section 2.13.8: Spurious master transfer upon own slave address match</a>.</p>
09-Mar-2020	6	<p>Updated <a href="#">Table 3: Summary of device limitations</a>.</p> <p>Added <a href="#">Section 2.3.19: Unstable LSI when it clocks RTC or CSS on LSE</a> and <a href="#">Section 2.3.20: FLASH_ECCR corrupted upon reset or power-down occurring during Flash memory program or erase operation</a>.</p>

**IMPORTANT NOTICE – PLEASE READ CAREFULLY**

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, please refer to [www.st.com/trademarks](http://www.st.com/trademarks). All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2020 STMicroelectronics – All rights reserved