STM32L471xx/475xx/476xx/486xx device errata

Applicability

This document applies to the part numbers of STM32L471xx/475xx/476xx/486xx devices and the device variants as stated in this page.

It gives a summary and a description of the device errata, with respect to the device datasheet and reference manual RM0351.

Deviation of the real device behavior from the intended device behavior is considered to be a device limitation. Deviation of the description in the reference manual or the datasheet from the intended device behavior is considered to be a documentation erratum. The term *"errata"* applies both to limitations and documentation errata.

Table 1. Device summary

Reference	Part numbers
STM32L471xx	STM32L471QE, STM32L471QG, STM32L471RE, STM32L471RG, STM32L471VE, STM32L471VG, STM32L471ZE, STM32L471ZG
STM32L475xx	STM32L475RC, STM32L475RE, STM32L475RG STM32L475VC, STM32L475VE, STM32L475VG
STM32L476xx	STM32L476RC, STM32L476VC STM32L476JE, STM32L476ME, STM32L476QE, STM32L476RE, STM32L476VE, STM32L476ZE, STM32L476JG, STM32L476MG, STM32L476QG, STM32L476RG, STM32L476VG, STM32L476ZG
STM32L486xx	STM32L486JG, STM32L486QG, STM32L486RG, STM32L486VG, STM32L486ZG

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32L471xx, STM32L475xx	4	0x1007
STM32L476xx, STM32L486xx	3	0x1003
	4	0x1007

1. *Refer to the device datasheet for how to identify this code on different types of package.*
2. *REV_ID[15:0] bitfield of DBGMCU_IDCODE register.*

1 Summary of device errata

The following table gives a quick reference to the STM32L471xx/475xx/476xx/486xx device limitations and their status:

A = limitation present, workaround available

N = limitation present, no workaround available

P = limitation present, partial workaround available

"-" = limitation absent

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

Table 3. Summary of device limitations

Function	Section	Limitation	Status	
			Rev. 3	Rev. 4
Core	2.1.1	Interrupted loads to SP can cause erroneous behavior	A	A
	2.1.2	VDIV or VSQRT instructions might not complete correctly when very short ISRs are used	A	A
	2.1.3	Store immediate overlapping exception return operation might vector to incorrect interrupt	A	A
System	2.2.1	MSIRDY flag issue preventing entry in low-power mode	A	A
	2.2.2	PCPROP area within a single flash memory page becomes unprotected at RDP change from Level 1 to Level 0	A	A
	2.2.3	MSI frequency overshoot upon Stop mode exit	A	A
	2.2.4	Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled	A	A
	2.2.5	Spurious brown-out reset after short run sequence	A	A
	2.2.6	Full JTAG configuration without NJTRST pin cannot be used	A	A
	2.2.7	Current injection from V_{DD} to V_{DDA} through analog switch voltage booster	A	A
	2.2.8	Unstable LSI when it clocks RTC or CSS on LSE	P	P
	2.2.9	FLASH_ECCR corrupted upon reset or power-down occurring during flash memory program or erase operation	A	A
	2.2.10	HSE oscillator long startup at low voltage	P	P
	2.2.11	SRAM write error	A	A
	2.2.12	Data cache might be corrupted during flash memory read-while-write operation	A	A
	2.2.13	Write operation in the flash memory while it is not ready (flash memory in power-down) is not correctly handled	A	A
	2.2.14	The configuration of the I/Os not available in the WLCSP package can be modified by software	A	A
	2.2.15	Some I/Os must not be used as output when $V_{DDA} > V_{DD} + 0.6$ V or $V_{DDUSB} > V_{DD} + 0.6$ V or $V_{LCD} > V_{DD} + 0.6$ V	N	-
	2.2.16	Bootloader: SPI and CAN interfaces are not supported	N	-
	2.2.17	SRAM2 read access while the SRAM2 hardware erase is ongoing is not correctly supported	N	N
	2.2.18	PH0/PH1 is controlled by the GPIOH registers when HSE is enabled	P	P

Function	Section	Limitation	Status	
			Rev. 3	Rev. 4
System	2.2.19	PWR_CR4 register write access may not be completed if a Low-power mode is entered just after the write operation	A	A
	2.2.20	HSI user trim is limited on some samples	-	N(1)
	2.2.21	Option byte loading can fail if the MSI frequency is greater than 8 MHz	A	A
	2.2.22	MSI at high frequency ranges cannot be used as a wake-up from Stop 0 clock source	P	-
	2.2.23	PLL may not lock if VCO is below 96 MHz and the temperature is below 0 °C	A	A
	2.2.24	OPAMP output: VDDA overconsumption	N	N
FW	2.3.1	Code segment unprotected if non-volatile data segment length is zero	A	A
	2.3.2	Code and non-volatile data unprotected upon bank swap	A	A
DMA	2.4.1	DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear	A	A
FMC	2.5.1	Dummy read cycles inserted when reading synchronous memories	N	N
	2.5.2	Interruption of CPU burst read access to the end of an SDRAM row	A	A
	2.5.3	FMC NOR/PSRAM controller: asynchronous read access on bank 2 to 4 returns wrong data when bank 1 is in synchronous mode (BURSTEN bit is set)	A	A
	2.5.4	FMC dynamic and static bank switching	A	A
	2.5.5	FMC NOR/PSRAM controller write protocol violation	A	A
	2.5.6	Data corruption during burst read from FMC synchronous memory	A	A
	2.5.8	FMC NOR/PSRAM controller bank switch with different BUSTURN durations	A	A
	2.5.9	Wrong data read from a busy NAND memory	A	A
	2.5.10	Spurious clock stoppage with continuous clock feature enabled	A	A
	2.5.11	Read burst accesses of nine words or more are not supported by the FMC	P	P
QUADSPI	2.6.1	QUADSPI_BK1_IO1 is always an input when the command is sent in dual or quad SPI mode	A	A
	2.6.2	Hard fault not generated upon out-of-range memory-mapped access	N	N
	2.6.3	Extra data written in the FIFO at the end of a read transfer	A	A
	2.6.4	First nibble of data not written after dummy phase	A	A
	2.6.5	Wrong data from memory-mapped read after an indirect mode operation	A	A
	2.6.6	Memory-mapped read operations may fail when timeout counter is enabled	P	P
	2.6.7	Memory-mapped access in indirect mode clearing QUADSPI_AR register	P	P
	2.6.8	Transmission error flag (TEF) in memory space access with ADMODE = 0	A	A
SDMMC	2.7.1	Wrong CCRCFAIL status after a response without CRC is received	A	A
	2.7.2	MMC stream write of less than 8 bytes does not work correctly	A	A
	2.7.3	Maximum clock division factor not working	A	A
	2.7.4	Wait for response bits "10" configuration does not work correctly	A	A
ADC	2.8.1	Injected queue of context is not available if JQM = 0	N	N
	2.8.3	Load multiple not supported by ADC interface	A	A

Function	Section	Limitation	Status	
			Rev. 3	Rev. 4
ADC	2.8.5	New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0	A	A
	2.8.6	Two consecutive context conversions fail when writing new context in ADC_JSQR just after previous context completion with JQDIS = 0 and JQM = 0	A	A
	2.8.7	Unexpected regular conversion when two consecutive injected conversions are performed in Dual interleaved mode	A	A
	2.8.8	ADC_AWDy_OUT reset by non-guarded channels	A	A
	2.8.9	Injected data stored in the wrong ADC_JDRx registers	A	A
	2.8.10	ADC slave data may be shifted in Dual regular simultaneous mode	A	A
	2.8.11	Wrong ADC result if conversion done late after calibration or previous conversion	A	A
	2.8.12	Spurious temperature measurement due to spike noise	A	A
	2.8.13	ADC triggers from EXTI require external interrupt management	P	-
	2.8.14	DMA2 channels 2 and 3 cannot be used when the ADC2 and ADC3 requests are selected on DMA2 channels 4 and 5, respectively	P	P
	2.8.15	Burst read or write accesses are not supported by the ADC	A	A
	2.8.16	Unexpected end of transfer on DMA1 when using DMA1 channel 2 for ADC2 while DMA2 channel 4 is also used	A	A
	2.8.17	Unexpected end of transfer on DMA1 when using DMA1 channel 3 for ADC3 while DMA2 channel 5 is also used	A	A
	2.8.18	Selected external ADC inputs unduly clamped to V_{DD} when all analog peripherals are disabled	A	A
DAC	2.9.1	DMA underrun flag not set when an internal trigger is detected on the clock cycle of the DMA request acknowledge	N	N
COMP	2.10.1	COMP1 and COMP2 configuration lost with software reset	N	N
	2.10.2	Comparators output cannot be configured in open-drain	N	N
	2.10.3	Comparators propagation delay is longer than expected for input steps higher than 200 mV	N	N
OPAMP	2.11.1	OPAMP characteristics can be degraded depending on the current sunk on OPAMP output	N	-
DFSDM	2.12.1	RDATACH[2:0] status bits are not implemented	N	N
	2.12.2	New regular channel selection taken into account at the end of an injected conversion when a regular conversion is pending	N	N
	2.12.3	DFSDM triggers from timers can be missed in specific conditions	A	A
	2.12.4	DFSDM triggers from EXTI require external interrupt management	P	-
LCD	2.13.1	LCD segments 23 and 24 alternate function cannot be configured independently	P	P
TSC	2.14.2	TSC signal-to-noise concern under specific conditions	A	A
AES	2.16.6	AES does not support Load multiple	A	A
TIM	2.17.1	One-pulse mode trigger not detected in master-slave reset + trigger configuration	P	P
	2.17.2	Consecutive compare event missed in specific conditions	N	N
	2.17.3	Output compare clear not working with external counter reset	P	P
LPTIM	2.18.1	Device may remain stuck in LPTIM interrupt when entering Stop mode	A	A

Function	Section	Limitation	Status	
			Rev. 3	Rev. 4
LPTIM	2.18.2	Device may remain stuck in LPTIM interrupt when clearing event flag	A	A
	2.18.3	LPTIM events and PWM output are delayed by one kernel clock cycle	A	A
	2.18.4	LPTIM1 outputs cannot be configured as open-drain	N	N
RTC and TAMP	2.19.1	Spurious tamper detection when disabling the tamper channel	P	P
	2.19.2	RTC calendar registers are not locked properly	A	A
	2.19.3	RTC interrupt can be masked by another RTC interrupt	A	A
	2.19.4	Calendar initialization may fail in case of consecutive INIT mode entry	A	A
	2.19.5	Alarm flag may be repeatedly set when the core is stopped in debug	N	N
	2.19.6	A tamper event fails to trigger timestamp or timestamp overflow events during a few cycles after clearing TSF	N	N
	2.19.7	RTC_REFIN and RTC_OUT on PB2 not operating in Stop 2 mode	P	P
I2C	2.20.1	10-bit controller mode: new transfer cannot be launched if first part of the address is not acknowledged by the target	A	A
	2.20.3	Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period	P	P
	2.20.4	Spurious bus error detection in controller mode	A	A
	2.20.5	Last-received byte loss in reload mode	P	P
	2.20.6	Spurious controller transfer upon own target address match	P	P
	2.20.8	OVR flag not set in underrun condition	N	N
	2.20.9	Transmission stalled after first byte transfer	A	A
	2.20.10	SDA held low upon SMBus timeout expiry in target mode	A	A
	2.20.11	I2C Fast-mode Plus drive is not available on all SDA/SCL I/Os	N	N
	2.20.12	I2C3 analog filter activation requires that both PC0/PC1 are configured as I2C3 alternate function	A	A
	2.20.13	The last received byte can be lost when using reload mode with NBYTES > 1	A	A
USART	2.21.1	Non-compliant sampling for NACK signal from smartcard	N	N
	2.21.2	Break request preventing TC flag from being set	A	A
	2.21.3	RTS is active while RE = 0 or UE = 0	A	A
	2.21.4	Receiver timeout counter wrong start in two-stop-bit configuration	A	A
	2.21.5	Data corruption due to noisy receive line	A	A
	2.21.6	Received data may be corrupted upon clearing the ABREN bit	A	A
	2.21.7	Noise error flag set while ONEBIT is set	N	N
LPUART	2.22.1	Break request preventing TC flag from being set	A	A
	2.22.2	RTS is active while RE = 0 or UE = 0	A	A
	2.22.3	Possible LPUART transmitter issue when using low BRR[15:0] value	P	P
	2.22.4	LPUART1 outputs cannot be configured as open-drain	N	N
SPI	2.23.1	BSY bit may stay high when SPI is disabled	A	A
	2.23.2	BSY bit may stay high at the end of data transfer in slave mode	A	A
	2.23.3	SPI CRC corruption upon DMA transaction completion by another peripheral	P	P
SAI	2.24.2	Last SAI_SCK clock pulse truncated upon disabling SAI master	N	N

Function	Section	Limitation	Status	
			Rev. 3	Rev. 4
SAI	2.24.3	Last SAI_MCLK clock pulse truncated upon disabling SAI master	A	A
	2.24.4	SAI_MCLK clock absent in a specific configuration	A	A
SWPMI	2.25.1	SUSPENDED mode entry delayed	N	N
	2.25.2	SUSPENDED mode never entered	P	P
	2.25.3	SRF flag not set	N	N
	2.25.4	SWP SUSPENDED mode not supported when the MCU is in Stop 0 or Stop 1 mode	A	A
	2.25.5	SWPMI_IO transceiver bypass mode is not functional	N	N
bxCAN	2.26.1	bxCAN time-triggered communication mode not supported	N	N
OTG_FS	2.27.1	Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_FS registers	A	A
	2.27.2	Host packet transmission may hang when connecting through a hub to a low-speed device	N	N

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

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

Table 4. Summary of device documentation errata

Function	Section	Documentation erratum
DMA	2.4.2	Byte and half-word accesses not supported
FMC	2.5.7	Missing information on prohibited 0xFF value of NAND transaction wait timing
ADC	2.8.2	Writing ADC_JSQR when JADCSTART and JQDIS are set may lead to incorrect behavior
	2.8.4	ADEN bit cannot be set immediately after the ADC calibration is done
TSC	2.14.1	Inhibited acquisition in short transfer phase configuration
RNG	2.15.1	RNG clock error does not stop random numbers
AES	2.16.1	Burst read or write accesses not supported
	2.16.2	TAG computation in GCM encryption mode
	2.16.3	CCM authentication mode not compliant with NIST CMAC
	2.16.4	Datatype initial configuration in GCM mode
	2.16.5	Wait until BUSY is low when suspending GCM encryption
I2C	2.20.2	Wrong behavior in Stop mode when wake-up from Stop mode is disabled in I2C
	2.20.7	START bit is cleared upon setting ADDRCF, not upon address match
SAI	2.24.1	Automatic restart upon late or anticipated frame error in I^2S slave mode not supported

2 Description of device errata

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

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

2.1 Core

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

2.1.1 Interrupted loads to SP can cause erroneous behavior

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

Description

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

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

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

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

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

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

Workaround

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

2.1.2 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 2 cycles for instructions inside the interrupt service routine, or 2 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 VQSRT 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:

- Disable lazy context save of floating point state by clearing LSPEN to 0 (bit 30 of the FPCCR at address 0xE000EF34).
- Ensure that every interrupt service routine contains more than 2 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 pended by a level-based interrupt which is cleared by C's handler then interrupt C will be pended 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 System

2.2.1 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:

- high MSI frequency
- low VDD external supply
- high temperature

Workaround

- The MSI clock can run at any frequency if both conditions below are met:
  - The system clock is MSI or PLL fed by MSI when requesting entry in low-power mode.
  - 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 ensures 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).

2.2.2 PCPROP area within a single flash memory page becomes unprotected at RDP change from Level 1 to Level 0

Description

With PCROP_RDP option bit cleared, the change of RDP from Level 1 to Level 0 normally results in erasure of flash memory banks except the flash memory pages containing the 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 such that it crosses limits of at least one flash memory page.

2.2.3 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

Apply the following sequence:

1. Select HSI16 as system clock.
2. Shut MSI down.
3. Wait for MSIRDY to go low (after 6 MSI clock cycles).
4. Mask interrupts by setting PRIMASK (in the core).
5. Initiate Stop mode entry, with MSI selected as system clock source upon wakeup. At this point, the systems enters Stop mode (unless an early wakeup event occurs). The following steps are done upon Stop mode exit.
6. Enable MSI.
7. Wait for MSIRDY to go high.
8. Select MSI as system clock.
9. Unmask interrupts by clearing PRIMASK.

The steps 6 through 8 are required to cover the case of the MCU not entering Stop mode after the step 5 (due to an early wakeup event), thus maintaining HSI16 as system clock.

This workaround guarantees the best wakeup time.

2.2.4 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 the ADC(s) and the temperature sensor channel reduces the power consumption during Stop mode.

2.2.5 Spurious brown-out reset 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 may be generated.

This limitation depends on the supply voltage (see the following table).

Table 5. Minimum run time

V_{DD} supply voltage (V)	Minimum run time (µs)
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 the previous table corresponds to the firmware execution time on the core. There is no need to add a delay for the wakeup time. Note also that the MCO output length is longer than the firmware execution time.

This limitation applies to Stop 1 and Stop 2 modes.

Workaround

Ensure that the run time between exiting Stop mode and entering Stop mode again is long enough not to generate 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 in order to minimize power consumption.

2.2.6 Full JTAG configuration without NJTRST pin cannot be used

Description

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

Workaround

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

2.2.7 Current injection from V_{DD} to V_{DDA} through analog switch voltage booster

Description

With V_{DDA} below 2.4 V and V_{DD} above 3 V, a small current injected from VDD line to VDDA line may cause V_{DDA} to exceed its nominal value.

This current injection only occurs when the I/O analog switch voltage booster is disabled (the BOOSTEN bit of the SYSCFG_CFGR1 register is cleared) and any analog peripheral (ADC, COMP, DAC or OPAMP) is enabled.

Workaround

Enable the I/O analog switch voltage booster, by setting the BOOSTEN bit of the SYSCFG_CFGR1 register. when V_{DDA} is below 2.4 V and V_{DD} is above 3 V.

2.2.8 Unstable LSI when it clocks RTC or CSS on LSE

Description

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

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

Workaround

Apply one of the following measures:

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

2.2.9 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.

2.2.10 HSE oscillator long startup at low voltage

Description

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

Workaround

The following sequence is recommended:

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

2.2.11 SRAM write error

Description

In rare cases, system reset occurring in a critical instant may upset the SRAM state machine. The first SRAM read or write access after the system reset then restores the normal operation of the SRAM for any subsequent accesses. However, if it is a write access, it fails to write data.

Workaround

Upon system reset, make a dummy read access to each 32-Kbyte SRAM instance used by the application, through one of the following methods:

1. Place a dummy variable initialization, which allows the compiler to create the assembly code.
2. Use this initialization assembly code:

```
MOV32 R0, #0x20000000 //SRAM address
LDR R1, [R0, #+0] //read access
MOV32 R0, #0x20008000 //next SRAM instance
LDR R1, [R0, #+0] //dummy read, no consequence on R1 value
MOV32 R0, #0x20010000
LDR R1, [R0, #+0]
MOV32 R0, #0x10000000 //the SRAM2
LDR R1, [R0, #+0]
```

Follow the first method for SRAM instances with the parity check enabled. Follow the first or the second method for SRAM instances with the parity check disabled.

2.2.12 Data cache might be corrupted during flash memory read-while-write operation

Description

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

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

Workaround

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

Code example:

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

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

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

/* Reset data cache */
__HAL_FLASH_DATA_CACHE_RESET();

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

2.2.13 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 (that is, 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 to check that the address is virgin is incorrect. Consequently, the write operation is skipped

Workaround

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.

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

Description

Unbonded I/Os in the 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.

2.2.15 Some I/Os must not be used as output when $V_{DDA} > V_{DD} + 0.6$ V or $V_{DDUSB} > V_{DD} + 0.6$ V or $V_{LCD} > V_{DD} + 0.6$ V

Description

When $V_{DDA} > V_{DD} + 0.6$ V, or when $V_{DDUSB} > V_{DD} + 0.6$ V, or when $V_{LCD} > V_{DD} + 0.6$ V, the I/Os listed below must not be used as:

- General-purpose output, push-pull, or open-drain
- Alternate function output, push-pull, or open-drain

An unexpected leakage occurs between the respective supply voltage (V_{DDA}, VDDUSB, or V_{LCD}) and V_{DD}, and the output state can be lost.

These I/Os can be used in:

- Input mode
- Analog mode
- Alternate function input or analog mode.

When $V_{DDA} > V_{DD} + 0.6 + V$, the concerned I/Os are:

- PA0, PA1, PA2, PA3, PA4, PA5, PA6, PA7
- PB0, PB1, PB2, PB3, PB4, PB5, PB6, PB7
- PC0, PC1, PC2, PC3, PC4, PC5
- PF3, PF4, PF5, PF6, PF7, PF8, PF9, PF10

When $V_{DDUSB} > V_{DD} + 0.6 + V$, the concerned I/Os are:

- PA9, PA10, PA11, PA12

When $V_{LCD} > V_{DD} + 0.6 + V$, the concerned I/Os are:

- PA1, PA2, PA3, PA6, PA7, PA8, PA9, PA10, PA15
- PB0, PB1, PB3, PB4, PB5, PB7, PB8, PB9, PB10, PB11, PB12, PB13, PB14, PB15
- PC0, PC1, PC2, PC4, PC5, PC6, PC7, PC8, PC9, PC10, PC11, PC12
- PD2, PD8, PD9, PD10, PD11, PD12, PD13, PD14, PD15
- PE0, PE1, PE2, PE3

Workaround

None.

2.2.16 Bootloader: SPI and CAN interfaces are not supported

Description

The bootloader does not support SPI and CAN interfaces.

Workaround

None.

2.2.17 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 ongoing: 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.

2.2.18 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

Configure PH0 and PH1 in analog mode (reset value) when HSE is enabled.

2.2.19 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.

2.2.20 HSI user trim is limited on some samples

Description

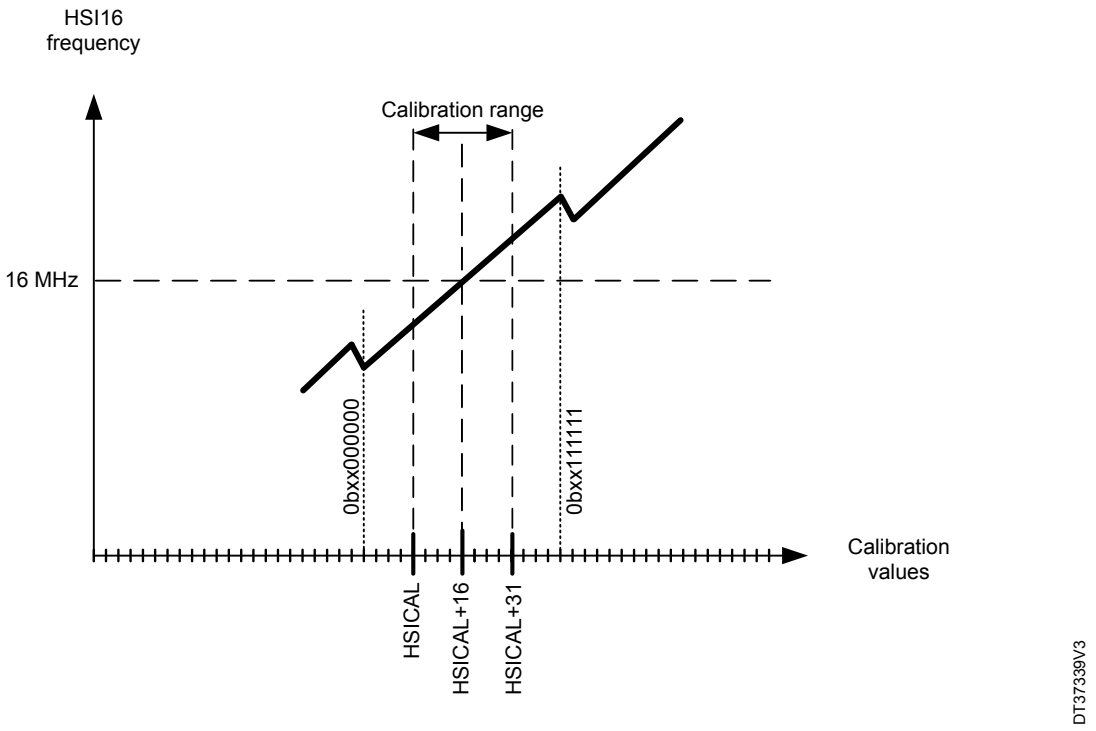
The HSI user trimming step is typically:

- +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 the negative direction.

Figure 1. HSI oscillator trimming characteristics



Workaround

None.

2.2.21 Option byte loading can fail if the 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 the OBL_LAUNCH bit of 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 8 MHz, reduce MSI frequency to 4 MHz or less before launching the option byte loading operation by setting the OBL_LAUNCH bit of the FLASH_CR register.

2.2.22 MSI at high frequency ranges cannot be used as a wake-up from Stop 0 clock source

Description

When V_{DD} > 3 V, the MSI startup frequency overshoot is higher than expected and can be higher than the maximum allowed product frequency. Therefore, it is not possible to select the MSI at 48 or at 32 MHz as wake-up clock source when exiting Stop 0 in Range 1, because the frequency reached during the overshoot is too high compared to the flash wait states configuration. For the same reason it is not possible to select the MSI at 24, 16 or 8 MHz as wake-up clock source when exiting Stop 0 in Range 2. The MSI in all ranges can be used to exit Stop 1 or Stop 2 mode, because the main regulator wake-up delay is used to avoid the clock propagation and the regulator wake-up time is greater than the MSI overshoot duration

Workaround

Wait for 6 µs after exiting Stop 0 before switching the MSI frequency to the higher frequency, or use Stop 1 or Stop 2 instead of Stop 0

2.2.23 PLL may not lock if VCO is below 96 MHz and the 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.

2.2.24 OPAMP output: VDDA overconsumption

Description

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

- A voltage VPAD is applied on PA3 or PB0 with V_{DDA} + 50 mV < VPAD < V_{DDA} + 600 mV.
- The OPAMP output is not used (OPAMPx_VOUT pins are not driven by the OPAMP).
- The 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 FW

2.3.1 Code segment unprotected if non-volatile data segment length is zero

Description

If during FW configuration the length of firewall-protected non-volatile data segment is set to zero 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.

2.3.2 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 flash memory bank in a symmetric way (same relative address and same length).

2.4 DMA

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

Description

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

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

Workaround

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

2.4.2 Byte and half-word accesses not supported

Description

Some reference manual revisions may wrongly state that the DMA registers are byte- and half-word-accessible. Instead, the DMA registers must always be accessed through aligned 32-bit words. Byte or half-word write accesses cause an erroneous behavior.

ST's low-level driver and HAL software only use aligned 32-bit accesses to the DMA registers.

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

Workaround

No application workaround is required.

2.5 FMC

2.5.1 Dummy read cycles inserted when reading synchronous memories

Description

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

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

Workaround

None.

2.5.2 Interruption of CPU burst read access to the end of an SDRAM row

Description

An interrupt occurring during a CPU AHB burst read access to the end of an SDRAM row may result in wrong data read from the next row if the following condition is met:

- the read access to SDRAM is 16- or 8-bit (32-bit accesses are not subject to the failure),
- the RBURST bit of the FMC_SDCR1 register is zero (read FIFO disabled),
- the length of the burst read access is undetermined, following LDM (load multiple) instruction, and
- the burst read spans over multiple SDRAM rows.

Workaround

Enable the read FIFO by setting the RBURST bit of the FMC_SDCR1 register.

2.5.3 FMC NOR/PSRAM controller: asynchronous read access on bank 2 to 4 returns wrong data when bank 1 is in synchronous mode (BURSTEN bit is set)

Description

If an interrupt occurs during a CPU AHB read access to one NOR/PSRAM bank (bank 2 to 4) which is enabled in asynchronous mode, while bank 1 of the NOR/PSRAM controller is configured in synchronous read mode (BURSTEN bit set), then the FMC NOR/PSRAM controller returns wrong data. This limitation does not occur when using the DMA or when only bank 1 is used in synchronous mode.

Workaround

If multiple banks are enabled in mixed asynchronous and synchronous modes, use any NOR/PSRAM bank for synchronous read accesses, except for bank 1. As a consequence the continuous clock feature is not available.

2.5.4 FMC dynamic and static bank switching

Description

The dynamic and static banks cannot be accessed concurrently.

Workaround

Do not use dynamic and static banks at the same time. The SDRAM device must be in self-refresh before switching to the static memory mapped on the NOR/PSRAM or NAND/PC-Card controller. Before switching from static memory to SDRAM, issue a Normal command to wake-up the device from self-refresh mode.

2.5.5 FMC NOR/PSRAM controller write protocol violation

Description

When an interrupted asynchronous or synchronous CPU read access to any NOR/PSRAM FMC bank is followed by an asynchronous write access to the same bank or to any other NOR/PSRAM FMC bank, this causes a write protocol violation. There is no functional issue but the FMC NOR/PSRAM controller write protocol is violated since the FMC_NWE signal is de-asserted at the same time as the Chip select (FMC_NEx).

Workaround

None.

2.5.6 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 of which the 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.

2.5.7 Missing information on prohibited 0xFF value of NAND transaction wait timing

Description

Some reference manual revisions may omit the information that the value 0xFF is prohibited for the wait timing of NAND transactions in their corresponding memory space (common or attribute).

Whatever the setting of the PWAITEN bit of the FMC_PCRx register, the wait timing set to 0xFF would cause a NAND transaction to stall the system with no fault generated.

This is a documentation error rather than a device limitation.

Workaround

No application workaround required provided that the 0xFF wait timing value is duly avoided.

2.5.8 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.

2.5.9 Wrong data read from a busy NAND memory

Description

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

Workaround

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

2.5.10 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.

division ratio set to 2, the FMC_CLK clock may spuriously stop upon an

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.

2.5.11 Read burst accesses of nine words or more are not supported by the FMC

Description

CPU read burst accesses equal to or more than nine registers to FMC return corrupted data starting from the ninth read word. These bursts can only be generated by Cortex®-M4 CPU and not by the other masters (that is, not by DMA).

This issue occurs when the stack is remapped to the external memory on the FMC, and POP operations are performed with nine or more registers. This also occurs when LDM/VLDM operations are used with nine or more registers.

Workaround

The stack must be located in the internal SRAM.

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

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

TASKING: the TASKING compiler starting from version v5.2r1 does not generate LDM instructions with more than eight 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 written assembly code, LDM/VLDM operations must be limited to eight registers.

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

If in dual or quad mode the command sequence does not use the IO1 line for sending the address, alternate byte or data, the line unduly remains configured as input, instead of becoming an output.

As a consequence, the command sequence does not operate in dual and in quad SPI modes.

Workaround

When using two or four data lines, do not limit the frame to the command only. As an example of Micron serial Flash memories, in the frame containing *"Write enable"* command, write the status register, too. Another way is to reset the serial memory through a reset pin (if available), for it to default to single-line operation, before sending a frame that only contains a command sequence.

2.6.2 Hard fault not generated upon out-of-range memory-mapped access

Description

When the CPU encounters an instruction to read data on an out-of-range memory-mapped address, it ignores the instruction and, instead of generating a hard fault, it continues executing.

Workaround

Avoid reading data on out-of-range memory-mapped addresses.

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

Description

When the following condition is met:

- 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),
- a data is read in the instant when the FIFO buffer gets full at the end of a read transfer,

an extra data is unduly written in the FIFO buffer.

Workaround

Apply one of the following measures:

- Read out the extra data until the BUSY flag goes low, then discard the extra data.
- After reading out all the expected received data, set the ABORT bit of the QUADSPI_CR register to flush FIFO and keep the BUSY flag low. The last register configuration is kept.

2.6.4 First nibble of data not written after dummy phase

Description

The first nibble of data to be written to the external flash memory is lost when the following condition is met:

- QUADSPI is used in indirect write mode.
- At least one dummy cycle is used.

Workaround

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

Example:

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

2.6.5 Wrong data from memory-mapped read after an indirect mode operation

Description

The first memory-mapped read in indirect mode can yield wrong data if the QUADSPI peripheral enters memory-mapped mode with bits ADDRESS[1:0] of the QUADSPI_AR register both set.

Workaround

Before entering memory-mapped mode, apply the following measure, depending on access mode:

- Indirect read mode: clear the QUADSPI_AR register then issue an abort request to stop reading and to clear the BUSY bit.
- Indirect write mode: clear the QUADSPI_AR register.

Caution: *The QUADSPI_DR register must not be written after clearing the QUADSPI_AR register.*

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

Description

In memory-mapped mode with the timeout counter enabled (by setting the TCEN bit of the QUADSPI_CR register), the QUADSPI peripheral may hang and memory-mapped read operation fail. This occurs if the timeout flag TOF is set at the same clock edge as a new memory-mapped read request.

Workaround

Disable the timeout counter. To raise the chip select, perform an abort at the end of each memory-mapped read operation.

2.6.7 Memory-mapped access in indirect mode clearing QUADSPI_AR register

Description

Memory-mapped accesses to the QUADSPI peripheral operating in indirect mode unduly clear the QUADSPI_AR register to 0x00.

Workaround

Adopt one of the following measures:

- Avoid memory-mapped accesses to the QUADSPI peripheral operating in indirect mode.
- After each memory-mapped access to the QUADSPI operating in indirect mode, write the QUADSPI_AR register with a desired value

2.6.8 Transmission error flag (TEF) in memory space access with ADMODE = 0

Description

The TEF (Transmission Error Flag) is designed to signal when memory is accessed outside of the memory address range. The controller calculates the address from the address register and the number of bytes to transfer to determine if the transaction hits the memory outside of the memory space.

However, if the previous transfer is closed to the end of the memory space and the number of bytes for the new transfer exceeds the memory size, even if no address is requested (ADMODE = 0), the TEF flag may be raised incorrectly. This is because the controller performs the check even if there are no address phases inside the transaction.

Workaround

To avoid the incorrect TEF assertion, the software application can clear the address register QUADSPI_AR when starting the transaction with ADMODE = 0.

Additionally, the HAL library driver has been updated to prevent this incorrect TEF assertion in this context.

2.7 SDMMC

2.7.1 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.

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

Description

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 writes 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 then 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.

2.7.3 Maximum clock division factor not working

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.

2.7.4 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, the 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.

2.8 ADC

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

Description

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

Workaround

None.

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

Description

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

Workaround

No application workaround is required for this description inaccuracy issue.

2.8.3 Load multiple not supported by ADC interface

Description

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

Workaround

The workaround consists in preventing compilers from generating LDM, STM, LDRD and STRD instructions.

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

Description

Some reference manual revisions may not indicate that the ADEN bit cannot be set while the ADCAL bit is set and during a four ADC clock cycles after the ADCAL bit is cleared by hardware (end of the calibration).

Otherwise, if the ADEN bit is set during this four ADC clock cycle period, it will be reset by the calibration logic and the ADC will stay disabled. This is due to the fact that there is an internal reset of the ADEN bit four ADC clock cycles after the ADCAL bit is cleared by hardware.

Workaround

No application workaround is required for this description inaccuracy issue.

2.8.5 New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0

Description

Once an injected conversion sequence is complete, the queue is consumed and the context changes according to the new ADC_JSQR parameters stored in the queue. This new context is applied for the next injected sequence of conversions.

However, the programming of the new context in ADC_JSQR (change of injected trigger selection and/or trigger polarity) may launch the execution of this context without waiting for the trigger if:

- the queue of context is enabled (JQDIS cleared to 0 in ADC_CFGR), and
- the queue is never empty (JQM cleared to 0 in ADC_CFGR), and
- the injected conversion sequence is complete and no conversion from previous context is ongoing

Workaround

Apply one of the following measures:

- Ignore the first conversion.
- Use a queue of context with JQM = 1.
- Use a queue of context with JQM = 0, only change the conversion sequence but never the trigger selection and the polarity.

2.8.6 Two consecutive context conversions fail when writing new context in ADC_JSQR just after previous context completion with JQDIS = 0 and JQM = 0

Description

When an injected conversion sequence is complete and the queue is consumed, writing a new context in ADC_JSQR just after the completion of the previous context and with a length longer that the previous context, may cause both contexts to fail. The two contexts are considered as one single context. As an example, if the first context contains element 1 and the second context elements 2 and 3, the first context is consumed followed by elements 2 and 3 and element 1 is not executed.

This issue may happen if:

- the queue of context is enabled (JQDIS cleared to 0 in ADC_CFGR), and
- the queue is never empty (JQM cleared to 0 in ADC_CFGR), and
- the length of the new context is longer than the previous one

Workaround

If possible, synchronize the writing of the new context with the reception of the new trigger.

2.8.7 Unexpected regular conversion when two consecutive injected conversions are performed in Dual interleaved mode

Description

In Dual ADC mode, an unexpected regular conversion may start at the end of the second injected conversion without a regular trigger being received, if the second injected conversion starts exactly at the same time than the end of the first injected conversion. This issue may happen in the following conditions:

- two consecutive injected conversions performed in Interleaved simultaneous mode (DUAL[4:0] of ADC_CCR = 0b00011), or
- two consecutive injected conversions from master or slave ADC performed in Interleaved mode (DUAL[4:0]of ADC_CCR = 0b00111)

Workaround

- In Interleaved simultaneous injected mode: make sure the time between two injected conversion triggers is longer than the injected conversion time.
- In Interleaved only mode: perform injected conversions from one single ADC (master or slave), making sure the time between two injected triggers is longer than the injected conversion time.

2.8.8 ADC_AWDy_OUT reset by non-guarded channels

Description

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

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

Workaround

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

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

2.8.9 Injected data stored in the wrong ADC_JDRx registers

Description

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

Workaround

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

2.8.10 ADC slave data may be shifted in Dual regular simultaneous mode

Description

In Dual regular simultaneous mode, ADC slave data may be shifted when all the following conditions are met:

- A read operation is performed by one DMA channel,
- OVRMOD = 0 in ADC_CFGR register (Overrrun mode enabled).

Workaround

Apply one of the following measures:

- Set OVRMOD = 1 in ADC_CFGR. This disables ADC_DR register FIFO.
- Use two DMA channels to read data: one for slave and one for master.

2.8.11 Wrong ADC result if conversion done late after calibration or previous conversion

Description

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

Workaround

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

2.8.12 Spurious temperature measurement due to spike noise

Description

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

Workaround

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

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

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

Example:

Table 6. Measurement result after IQM post-processing

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

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

2.8.13 ADC triggers from EXTI require external interrupt management

Description

When the ADC external trigger for regular or injected channel selects the EXTI line 11 or EXTI line 15, it is required to configure and enable the line as an interrupt source in the EXTI and NVIC peripherals.

Workaround

Configure the EXTI line as an interrupt source in the EXTI and enable the interrupt line in the NVIC. The EXTI interrupt flag must be cleared in the interrupt subroutine in order to enable new trigger event detection.

2.8.14 DMA2 channels 2 and 3 cannot be used when the ADC2 and ADC3 requests are selected on DMA2 channels 4 and 5, respectively

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.

2.8.15 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

2.8.16 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 for 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 is 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

2.8.17 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 for 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 is 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.

2.8.18 Selected external ADC inputs unduly clamped to V_{DD} when all analog peripherals are disabled

Description

When all analog peripherals (other than VREFBUF) are disabled, the GPIO(s) selected as ADC input(s) are unduly clamped (through a parasitic diode) to V_{DD} instead to V_{DDA}. As a consequence, the input voltage is limited to V_{DD} + 0.3 V even if V_{DDA} is higher than V_{DD} + 0.3 V.

Note: The selection of GPIOs as ADC inputs is done with the SQy and JSQy bitfields of the ADC_SQRx and ADC_JSQR registers, respectively.

VREFBUF enable/disable has no impact to the issue described.

Workaround

Apply one of the following measures:

- Use V_{DDA} lower than V_{DD} + 0.3 V.
- Keep at least one analog peripheral (other than VREFBUF) enabled if GPIOs are selected as ADC inputs.
- Deselect GPIOs as ADC inputs (by clearing ADC_SQRx or/and ADC_JSQR registers) when no analog peripheral (other than VREFBUF) is enabled.

2.9 DAC

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

Description

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

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

Workaround

None.

2.10 COMP

2.10.1 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.

2.10.2 Comparators output cannot be configured in open-drain

Description

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

Workaround

None.

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

Description

Table 7 summarizes the comparator propagation delay values. The propagation delay for steps higher than 200 mV is out of the targeted specification.

Table 7. COMP characteristics

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

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
t_D	Propagation delay for 200 mV step with 100 mV overdrive	High-speed mode	$V_{DDA} \geq 2.7$ V	-	55	80	ns
			$V_{DDA} < 2.7$ V	-	65	100	
		Medium mode	$V_{DDA} \geq 2.7$ V	-	0.55	0.9	µs
			$V_{DDA} < 2.7$ V	-	0.65	1.0	

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
t_D	Propagation delay for 200 mV step with 100 mV overdrive	Ultralow-speed mode	$V_{DDA} \geq 2.7$ V	-	5	12	µs
			$V_{DDA} < 2.7$ 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-speed mode	-	-	8	24	

Workaround

None.

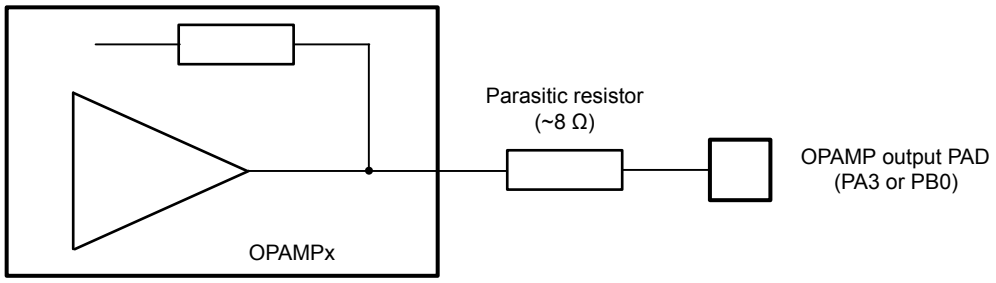
2.11 OPAMP

2.11.1 OPAMP characteristics can be degraded depending on the current sunk on OPAMP output

Description

The value of the resistor between OPAMP outputs and I/O (PA3 or PB0) is greater than expected: 8 Ω typical, as shown in Figure 2. As a consequence, the OPAMP characteristics can be impacted depending on the current sunk on the output.

Figure 2. Parasitic resistor on OPAMP output



Workaround

None.

2.12 DFSDM

2.12.1 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.

2.12.2 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 ongoing, and it 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.

2.12.3 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).

2.12.4 DFSDM triggers from EXTI require external interrupt management

Description

When the DFSDM external trigger for regular or injected channel selects the EXTI line 11 or EXTI line 15, it is required to configure and enable the line as an interrupt source in the EXTI and NVIC peripherals.

Workaround

Configure the EXTI line as an interrupt source in the EXTI and enable the interrupt line in the NVIC. The EXTI interrupt flag must be cleared in the interrupt subroutine in order to allow new trigger event detection.

2.13 LCD

2.13.1 LCD segments 23 and 24 alternate function cannot be configured independently

Description

When PC5 is configured as LCD alternate function to output LCD_SEG23, PC6 is also configured as LCD alternate function and outputs LCD_SEG24. As a consequence, when PC5 is used for LCD, PC6 cannot be used for another purpose than LCD. In addition, in order to use PC6 as LCD alternate function, it is required to configure PC5 as LCD alternate function.

Workaround

When the LCD is used, use both PC5 and PC6 as LCD segments, or use none of them.

2.14 TSC

2.14.1 Inhibited acquisition in short transfer phase configuration

Description

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

- The PGPSC[2:0] bitfield set to 000 and the CTPL[3:0] bitfield to 0000 or 0001
- The PGPSC[2:0] bitfield set to 001 and the CTPL[3:0] bitfield to 0000

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

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

Workaround

No application workaround is required.

2.14.2 TSC signal-to-noise concern under specific conditions

Description

V_{DD} equal to or greater than V_{DDA} may lead (depending on part) to some degradation of the signal-to-noise ratio on the TSC analog I/O group 2.

The lower are the sampling capacitor (C_S) and the sensing electrode (C_X) capacitances, the worse is the signal-to-noise ratio degradation.

Workaround

Apply one of the following measures:

- Maximize C_S capacitance.
- Use the analog I/O group 2 as active shield.

2.15 RNG

2.15.1 RNG clock error does not stop random numbers

Description

Some revisions of the reference manual may contain the following wrong statements to ignore:

- If the RNG clock frequency is too low, the RNG stops generating random numbers.
- The RNG operates only when the CECS flag is set to 0.

This is a documentation issue rather than a device limitation.

Workaround

No application workaround is required or applicable.

2.16 AES

2.16.1 Burst read or write accesses not supported

Description

Some revisions of the reference manual may omit the information that the AES peripheral does not support LDM, STM, LDRD and STRD instructions for successive multiple-data (burst) read and write accesses to a contiguous address block.

This is a documentation issue rather than a product limitation.

Workaround

No application workaround is required, provided that the multiple-data instructions are not used to access the AES peripheral.

Note: To prevent compilers from generating LDM, STM, LDRD and STRD instructions to access the AES peripheral, organize the source code such as to avoid consecutive read or write accesses to neighboring addresses in lower-to-higher order. In case where consecutive read or write accesses to neighboring addresses cannot be avoided, order the source code such as to access higher address first.

2.16.2 TAG computation in GCM encryption mode

Description

Some revisions of the reference manual may omit the following application guidelines for the device to correctly compute GCM encryption authentication tags when the input data in the last block is inferior to 128 bits.

During GCM encryption payload phase and before inserting a last plaintext block smaller than 128 bits, apply the following steps:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Change the mode to CTR by writing 010 to the CHMOD[2:0] bitfield of the AES_CR register.
3. Pad the last block (smaller than 128 bits) with zeros to have a complete block of 128 bits, then write it into AES_DINR register.
4. Upon encryption completion, read the 128-bit ciphertext from the AES_DOUTR register and store it as intermediate data.
5. Change again the mode to GCM by writing 011 to the CHMOD[2:0] bitfield of the AES_CR register.
6. Select Final phase by writing 11 to the GCMPH[1:0] bitfield of the AES_CR register.
7. In the intermediate data, set to zero the bits corresponding to the padded bits of the last block of payload, then insert the resulting data into AES_DINR register.
8. Wait for operation completion, and read data on AES_DOUTR. This data is to be discarded.
9. Apply the normal Final phase as described in the datasheet

This is a documentation issue rather than a product limitation.

Workaround

No further application workaround is required, provided that these guidelines are respected.

2.16.3 CCM authentication mode not compliant with NIST CMAC

Description

Some revisions of the reference manual may omit the information that setting the CHMOD[2:0] bitfield to 100 selects the CCM authentication mode. It is not compliant with NIST CMAC, as defined in Special Publication 800-38B. In order to fully implement the CCM chaining as specified in NIST Special Publication 800-38C, the payload must first be encrypted or decrypted with the AES peripheral set in CTR mode (CHMOD[2:0] = 010), then the message associated data and payload authenticated with the AES peripheral set in CCM mode (CHMOD[2:0] = 100).

This is a documentation issue rather than a product limitation.

Workaround

No further application workaround is required, provided that these guidelines are respected.

2.16.4 Datatype initial configuration in GCM mode

Description

Some revisions of the reference manual may omit the information that GCM generated TAG is not correct if data swapping is not disabled in GCM init phase (GCMPH[1:0] = 00) by setting the DATATYPE[1:0] bitfield of the AES_CR register to zero.

This is a documentation issue rather than a product limitation.

Workaround

No further application workaround is required, provided that these guidelines are respected.

2.16.5 Wait until BUSY is low when suspending GCM encryption

Description

Some revisions of the reference manual may omit the information that when suspending the GCM encryption of a message, in the payload phase the BUSY flag of the AES_SR register must be low before saving the AES_SUSPxR registers in the memory.

This is a documentation issue rather than a product limitation.

Workaround

No further application workaround is required, provided that these guidelines are respected.

2.16.6 AES does not support Load multiple

Description

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

Workaround

The workaround consists in preventing compilers from generating LDM, STM, LDRD, and STRD instructions.

In general, this can be achieved by organizing the source code in a way that avoids 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, the source code must be ordered to access the higher address first.

2.17 TIM

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

Description

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

OPM = 1 in TIMx_CR1, SMS[3:0] = 1000 and MSM = 1 in TIMx_SMCR.

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

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

Workaround

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

2.17.2 Consecutive compare event missed in specific conditions

Description

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

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

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

As a consequence:

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

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

Workaround

None.

2.17.3 Output compare clear not working with external counter reset

Description

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

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

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

Workaround

Apply one of the following measures:

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

2.18 LPTIM

2.18.1 Device may remain stuck in LPTIM interrupt when entering Stop mode

Description

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

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

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

Workaround

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

2.18.2 Device may remain stuck in LPTIM interrupt when clearing event flag

Description

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

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

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

Workaround

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

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

Note: The standard clear sequence implemented in the HAL_LPTIM_IRQHandler in the STM32Cube is considered as the proper clear sequence.

2.18.3 LPTIM events and PWM output are delayed by one kernel clock cycle

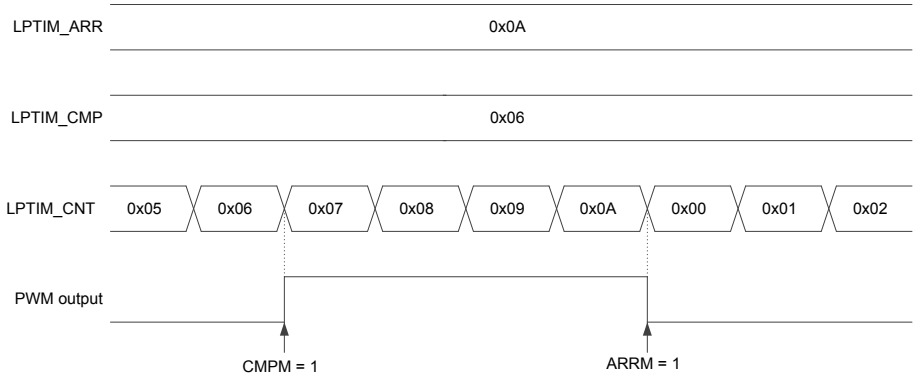
Description

The compare match event (CMPM), auto reload match event (ARRM), PWM output level and interrupts are updated with a delay of one kernel clock cycle.

Consequently, it is not possible to generate PWM with a duty cycle of 0% or 100%.

The following waveform gives the example of PWM output mode and the effect of the delay:

Figure 3. **Example of PWM output mode**



Workaround

Set the compare value to the desired value minus 1. For instance in order to generate a compare match when LPTM_CNT = 0x08, set the compare value to 0x07.

2.18.4 LPTIM1 outputs cannot be configured as open-drain

Description

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

Workaround

None.

2.19 RTC and TAMP

2.19.1 Spurious tamper detection when disabling the tamper channel

Description

If the tamper detection is configured for detecting on the 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, which may result in the erasure of backup registers.

Workaround

None for the false detection of tamper event. The erasure of the backup registers can be avoided by setting the TAMPxNOERASE bit before clearing the TAMPxE bit, in two separate RTC_TAMPCR write accesses.

2.19.2 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, the RTC_DR register can be updated after reading the RTC_TR register instead of being locked.

Workaround

Apply one of the following measures:

- 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.

2.19.3 RTC interrupt can be masked by another RTC interrupt

Description

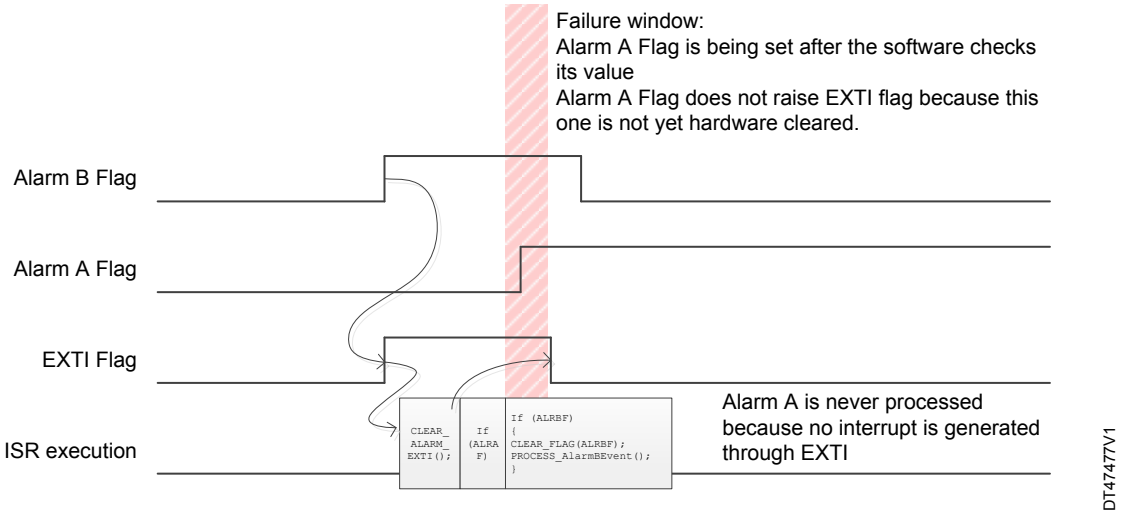
One RTC interrupt request can mask another RTC interrupt request if they share the same EXTI configurable line. For example, interrupt requests from Alarm A and Alarm B or those from tamper and timestamp events are OR-ed to the same EXTI line (refer to the *EXTI line connections* table in the *Extended 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 4. Masked RTC interrupt



Workaround

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

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

2.19.4 Calendar initialization may fail in case of consecutive INIT mode entry

Description

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

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

Workaround

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

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

2.19.5 Alarm flag may be repeatedly set when the core is stopped in debug

Description

When the core is stopped in debug mode, the clock is supplied to subsecond RTC alarm downcounter even when the device is configured to stop the RTC in debug.

As a consequence, when the subsecond counter is used for alarm condition (the MASKSS[3:0] bitfield of the RTC_ALRMASSR and/or RTC_ALRMBSSR register set to a non-zero value) and the alarm condition is met just before entering a breakpoint or printf, the ALRAF and/or ALRBF flag of the RTC_SR register is repeatedly set by hardware during the breakpoint or printf, which makes any attempt to clear the flag(s) ineffective.

Workaround

None.

2.19.6 A tamper event fails to trigger timestamp or timestamp overflow events during a few cycles after clearing TSF

Description

With the timestamp on tamper event enabled (TAMPTS bit of the RTC_CR register set), a tamper event is ignored if it occurs:

- within four APB clock cycles after setting the CTSF bit of the RTC_SCR register to clear the TSF flag, while the TSF flag is not yet effectively cleared (it fails to set the TSOVF flag)
- within two ck_apre cycles after setting the CTSF bit of the RTC_SCR register to clear the TSF flag, when the TSF flag is effectively cleared (it fails to set the TSF flag and timestamp the calendar registers)

Workaround

None.

2.19.7 RTC_REFIN and RTC_OUT on PB2 not operating in Stop 2 mode

Description

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

Workaround

Apply one of the following measures:

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

2.20 I2C

2.20.1 10-bit controller mode: new transfer cannot be launched if first part of the address is not acknowledged by the target

Description

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

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

Workaround

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

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

2.20.2 Wrong behavior in Stop mode when wake-up from Stop mode is disabled in I2C

Description

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

Some reference manual revisions may omit this information.

Failure to respect the above while the MCU operating as target or as controller in multi-controller topology enters Stop mode during a transfer ongoing on the I²C-bus may lead to the following:

1. BUSY flag is wrongly set when the MCU exits Stop mode. This prevents from initiating a transfer in controller mode, as the START condition cannot be sent when BUSY is set.
2. If clock stretching is enabled (NOSTRETCH = 0), the SCL line is pulled low by I2C and the transfer stalled as long as the MCU remains in Stop mode.
   The occurrence of such condition depends on the timing configuration, peripheral clock frequency, and I²C-bus frequency.

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

Workaround

No application workaround is required.

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

Description

The I²C-bus specification and user manual specify a minimum data setup time ($t_{SU;DAT}$) as:

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

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

Workaround

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

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

2.20.4 Spurious bus error detection in controller mode

Description

In controller mode, a bus error can be detected spuriously, with the consequence of setting the BERR flag of the I2C_SR register and generating bus error interrupt if such interrupt is enabled. Detection of bus error has no effect on the I²C-bus transfer in controller mode and any such transfer continues normally.

Workaround

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

2.20.5 Last-received byte loss in reload mode

Description

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

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

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

Workaround

- In controller mode or in target mode with SBC = 1, use the reload mode with NBYTES = 1.
- In controller receiver mode, if the number of bytes to transfer is greater than 255, do not use the reload mode. Instead, split the transfer into sections not exceeding 255 bytes and separate them with repeated START conditions.
- Make sure, for example through the use of DMA, that the byte N - 1 is always read before the TCR flag is raised.

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

2.20.6 Spurious controller transfer upon own target address match

Description

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

- Another controller on the bus is in process of sending the target address of the device (the bus is busy).
- The device initiates a controller transfer by bit set before the target 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 controller transfer as soon as the bus becomes free. The transfer configuration depends on the content of the I2C_CR2 register when the controller transfer starts. Moreover, if the I2C_CR2 is written less than three kernel clocks before the ADDR flag is cleared, the I2C peripheral may fall into an unpredictable state.

Workaround

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

Normal mode (SBC = 0):

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

Target byte control mode (SBC = 1):

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

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

Polling the BUSY flag before requesting the controller 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.

2.20.7 START bit is cleared upon setting ADDRCF, not upon address match

Description

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

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

Workaround

No application workaround is required for this description inaccuracy issue.

2.20.8 OVR flag not set in underrun condition

Description

In target transmission with clock stretching disabled (NOSTRETCH = 1 in the I2C_CR1 register), an underrun condition occurs if the current byte transmission is completed on the I^2C bus, and the next data is not yet written in the TXDATA[7:0] bitfield. In this condition, the device is expected to set the OVR flag of the I2C_ISR register and send 0xFF on the bus.

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

Workaround

None.

2.20.9 Transmission stalled after first byte transfer

Description

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

Workaround

Apply one of the following measures:

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

2.20.10 SDA held low upon SMBus timeout expiry in target mode

Description

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

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

Workaround

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

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

2.20.11 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.

2.20.12 I2C3 analog filter activation requires that both PC0/PC1 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, for using PC0 as clock and PB4 as data, PC0 filter can be enabled only if PC1 is configured in I2C3 alternate function mode, too.

Workaround

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

2.20.13 The last received byte can be lost when using reload mode with NBYTES > 1

Description

The issue can occur in controller mode when the reload mode is used (needed for transferring more than 255 bytes), or in target 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 4 APB clock cycles before the TCR flag is set in the I2C_ISR register.

Workaround

In target byte control mode, use the reload mode with NBYTESN = 1.

In controller mode, do not use the reload mode. If the number of bytes to be transferred is larger than 255 bytes, the total transfer must be split in several transfers, each one not exceeding 255 bytes, separated by repeated start conditions.

Note that the use of DMA can ensure 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.

2.21 USART

2.21.1 Non-compliant sampling for NACK signal from smartcard

Description

According to ISO/IEC 7816-3 standard, when a character parity error is detected, the receiver must assert a NACK signal, by pulling the transmit line low for one ETU period, at 10.3 to 10.7 ETU after the character START bit falling edge. The transmitter is expected to sample the line for NACK (for low level) from 10.8 to 11.2 ETU after the character START bit falling edge.

Instead, the USART peripheral in smartcard mode samples the transmit line for NACK from 10.3 to 10.7 ETU after the character START bit falling edge. This is unlikely to cause issues with receivers (smartcards) that respect the ISO/IEC 7816-3 standard. However, it may cause issues with respect to certification.

Workaround

None.

2.21.2 Break request preventing TC flag from being set

Description

After the end of transmission of data (D1), the transmission complete (TC) flag is not set when the following condition is met:

- CTS hardware flow control is enabled
- D1 transmission is in progress
- a break transfer is requested before the end of D1 transfer
- CTS is de-asserted before the end of D1 transfer

As a consequence, an application relying on the TC flag fails to detect the end of data transfer.

Workaround

In the application, only allow break request after the TC flag is set.

2.21.3 RTS is active while RE = 0 or UE = 0

Description

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

Workaround

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

2.21.4 Receiver timeout counter wrong start in two-stop-bit configuration

Description

In two-stop-bit configuration, the receiver timeout counter starts counting from the end of the second stop bit of the last character instead of starting from the end of the first stop bit.

Workaround

Subtract one bit duration from the value in the RTO bitfield of the USARTx_RTOR register.

2.21.5 Data corruption due to noisy receive line

Description

In all modes, except synchronous slave mode, the received data may be corrupted if a glitch to zero shorter than the half-bit occurs on the receive line within the second half of the stop bit.

Workaround

Apply one of the following measures:

- Either use a noiseless receive line, or
- add a filter to remove the glitches if the receive line is noisy.

2.21.6 Received data may be corrupted upon clearing the ABREN bit

Description

The USART receiver may miss data or receive corrupted data when the auto baud rate feature is disabled by software (ABREN bit cleared in the USART_CR2 register) after an auto baud rate detection, while a reception is ongoing.

Workaround

Do not clear the ABREN bit.

2.21.7 Noise error flag set while ONEBIT is set

Description

When the ONEBIT bit is set in the USART_CR3 register (one sample bit method is used), the noise error (NE) flag must remain cleared. Instead, this flag is set upon noise detection on the START bit.

Workaround

None.

Note: Having noise on the START bit is contradictory with the fact that the one sample bit method is used in a noise free environment.

2.22 LPUART

2.22.1 Break request preventing TC flag from being set

Description

After the end of transmission of data (D1), the transmission complete (TC) flag is not set when the following condition is met:

- CTS hardware flow control is enabled
- D1 transmission is in progress
- a break transfer is requested before the end of D1 transfer

- CTS is de-asserted before the end of D1 transfer

As a consequence, an application relying on the TC flag fails to detect the end of data transfer.

Workaround

In the application, only allow break request after the TC flag is set.

2.22.2 RTS is active while RE = 0 or UE = 0

Description

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

Workaround

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

2.22.3 Possible LPUART transmitter issue when using low BRR[15:0] value

Description

The LPUART transmitter bit length sequence is not reset between consecutive bytes, which could result in a jitter that cannot be handled by the receiver device. As a result, depending on the receiver device bit sampling sequence, a desynchronization between the LPUART transmitter and the receiver device may occur resulting in data corruption on the receiver side.

This happens when the ratio between the LPUART kernel clock and the baud rate programmed in the LPUART_BRR register (BRR[15:0]) is not an integer, and is in the three to four range. A typical example is when the 32.768 kHz clock is used as kernel clock and the baud rate is equal to 9600 baud, resulting in a ratio of 3.41.

Workaround

Apply one of the following measures:

- On the transmitter side, increase the ratio between the LPUART kernel clock and the baud rate. To do so:
  - Increase the LPUART kernel clock frequency, or
  - Decrease the baud rate.
- On the receiver side, generate the baud rate by using a higher frequency and applying oversampling techniques if supported.

2.22.4 LPUART1 outputs cannot be configured as open-drain

Description

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

Workaround

None.

2.23 SPI

2.23.1 BSY bit may stay high when SPI is disabled

Description

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

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

Workaround

When the SPI operates in:

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

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

Description

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

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

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

Workaround

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

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

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

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

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

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

2.23.3 SPI CRC corruption upon DMA transaction completion by another peripheral

Description

When the following conditions are all met:

- CRC function for the SPI is enabled
- SPI transaction managed by software (as opposed to DMA) is ongoing and CRCNEXT flag set
- another peripheral using the DMA channel on which the SPI is mapped completes a DMA transfer,

the CRCNEXT bit is unexpectedly cleared and the SPI CRC calculation may be corrupted, setting the CRC error flag.

Workaround

Ensure that the DMA channel on which the SPI is mapped is not concurrently in use by another peripheral.

2.24 SAI

2.24.1 Automatic restart upon late or anticipated frame error in I²S slave mode not supported

Description

Some reference manual revisions may omit the following information.

In I²S (FSDEF = 1) slave mode, upon detecting a late or anticipated frame error in the midst of an audio frame, the corresponding flag is duly set and the transfer restarted upon the detection of the next start of frame, but the FIFO contents may get desynchronized with respect to data slots.

Therefore, upon late or anticipated frame error detection, the SAI peripheral must be resynchronized with the master through the following sequence:

1. Disable SAI by clearing the SAIXEN bit of the SAI_xCR1 register (wait until the bit returns zero upon read).
2. Flush the FIFO via the FFLUSH bit of the SAI_xCR2 register.
3. Enable SAI by setting the SAIXEN bit.

The resynchronization with the master starts upon the nearest FS line active state.

This is a documentation issue rather than a device limitation.

Workaround

No application workaround is applicable or required if the instruction, as described, is respected.

2.24.2 Last SAI_SCK clock pulse truncated upon disabling SAI master

Description

When disabling, during the communication, the SAI peripheral configured as master, it may truncate the last SAI_SCK bit clock pulse of the transaction, potentially causing a failure to the external codec logic.

Workaround

None.

2.24.3 Last SAI_MCLK clock pulse truncated upon disabling SAI master

Description

When disabling, during the communication, the SAI peripheral configured as master with the OUTDRIV bit of the corresponding SAI_xCR1 register cleared, the device may truncate the last SAI_MCLK_x bit clock pulse of the transaction, potentially causing a failure to the external codec logic.

Workaround

Set the OUTDRIV bit of the corresponding SAI_xCR1 register.

2.24.4 SAI_MCLK clock absent in a specific configuration

Description

When configured as master with PRTCFG[1:0] = 00, NODIV = 0, and MCKDIV = 0 in the SAI_xCR1 register of the audio sub-block *x*, and FRL = 0xFF in the corresponding SAI_xFRCR register, the SAI peripheral fails to generate the master clock on the corresponding SAI_MCLK_x output.

As in this configuration, the master and bit clocks are identical, the application can use the SAI_SCK_x output also as master clock line. The absence of clock signal on the SAI_MCLK_x output allows saving power.

Workaround

Apply one of the following measures:

- In the application, use SAI_SCK_x as master and bit clock output. Optionally, disable the SAI_MCLK_x master clock output by setting NODIV.

- Configure the RCC block to set SAI kernel clock frequency to an integer multiple of the desired SAI_MCLK_x master clock frequency. Set the MCKDIV[5:0] bitfield so as to obtain the desired SAI_MCLK_x master clock frequency.

2.25 SWPMI

2.25.1 SUSPENDED mode entry delayed

Description

When activating the SWPMI by setting the SWPEN bit in the SWPMI_CR register, the SWPMI rises the SWPMI_IO to 1.8 V, 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 the SWPMI_ISR register.

Workaround

None.

2.25.2 SUSPENDED mode never entered

Description

When activating the SWPMI by setting the SWPEN bit in the SWPMI_CR register, the SWPMI generates the following sequence in a 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 the 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.

2.25.3 SRF flag not set

Description

If the SWPMI1SEL bit is set in the RCC_CCIPR register to select HSI as the 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.

2.25.4 SWP SUSPENDED mode not supported when the MCU is in Stop 0 or Stop 1 mode

Description

The MCU 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.

2.25.5 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.
Use the internal transceiver.

2.26 bxCAN

2.26.1 bxCAN time-triggered communication mode not supported

Description

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

Workaround

None.

2.27 OTG_FS

2.27.1 Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_FS registers

Description

When the USB on-the-go full-speed peripheral is in Device mode, interrupting transmit FIFO write sequence with read or write accesses to OTG_FS endpoint-specific registers (those ending in *0* or *x*) leads to corruption of the next data written to the transmit FIFO.

Workaround

Ensure that the transmit FIFO write sequence is not interrupted with accesses to the OTG_FS registers.

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

Description

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

Workaround

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

Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on www.st.com for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

Revision history

Table 8. **Document revision history**

Date	Version	Changes
29-May-2015	1	Initial release.
25-Sep-2015	2	Updated Table 1: Device summary and Table 3: Summary of device limitations. Updated Section 2.3.2: The configuration of the I/Os not available in WLCSP package can be modified by software, Section 2.4.4: Read burst access of nine words or more is not supported by FMC, Section 2.7.5: Burst read or write accesses are not supported by the ADC, Section 2.18.1: I2C Fast-mode Plus drive is not available on all SDA/SCL I/Os and Section 2.18.2: Wrong behavior related with MCU Stop mode when wakeup from Stop mode by I2C peripheral is disabled. Added Section 2.3.14: Full JTAG configuration without NJTRST pin cannot be used, Section 2.18.4: Spurious Bus Error detection in master mode, Section 2.18.5: I2C3 analog filters requires that both PC0/PC1 or both PG7/PG8 are configured as I2C3 alternate function and Section 2.15: AES.
03-Dec-2015	3	Updated Table 3: Summary of device limitations. Added Section 2.3.15: MSIRDY flag issue preventing entry in low power mode, Section 2.16.2: Calibration procedure does not work in PGA mode and Section 2.10.3: COMP1 and COMP2 configuration lost with software reset. Updated Section 2.3.1: Write operation in the Flash memory while it is not ready (Flash memory in power-down) is not correctly handled, Section 2.3.12: MSI at high frequency ranges cannot be used as wakeup from Stop 0 clock source, Section 2.4.4: Read burst access of nine words or more is not supported by FMC and Section 2.22.4: SWP SUSPENDED mode not supported when STM32L4 is in Stop 0 or Stop 1 mode.
06-Dec-2016	4	Updated Table 3: Summary of device limitations and added footnote 1. Added Section 2.3.16: PCPROP area within a single Flash memory page becomes unprotected at RDP change from level 1 to level 0, Section 2.3.17: Data Cache might be corrupted during Flash Read While Write operation, Section 2.3.18: MSI frequency overshoot upon Stop mode exit, Section 2.3.19: Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled, Section 2.6.4: First nibble of data is not written after dummy phase, Section 2.6.5: Wrong data can be read in memory-mapped after an indirect mode operation, Section 2.7.6: Unexpected end of transfer on DMA1 when using DMA1 channel 2 for ADC2 while DMA2 channel 4 is also used, Section 2.7.7: Unexpected end of transfer on DMA1 when using DMA1 channel 3 for ADC3 while DMA2 channel 5 is also used, Section 2.18.6: 10-bit master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave, Section 2.14: TSC and Section 2.15.3: Wrong TAG generation in GCM mode with encryption, for payloads smaller than 128 bits. Removed former Section 2.16.2: Calibration procedure does not work in PGA mode.
24-Apr-2017	5	Updated Table 3: Summary of device limitations. Updated Section 2.3.18: MSI frequency overshoot upon Stop mode exit. Added Section 2.3.20: OPAMP output: VDDA overconsumption, Section 2.3.21: Spurious BOR when entering Stop mode after short Run sequence, Section 2.10.8: START bit is not cleared when the address is not acknowledged by the slave device and Section 2.25.4: Data FIFO gets corrupted if the write sequence to the Transmit FIFO is interleaved with other OTGFS register access
31-May-2018	6	Replaced former Silicon identification with Applicability, and former Table 1: Device identification with Table 2: Device variants. Updated Section 1: Summary of device limitations, Section 2.1: Core and Section 2.1.1: Interrupted loads to stack pointer can cause erroneous behavior. Updated Table 3: Summary of device limitations.

Date	Version	Changes
		Rearranged description of limitations according to order of reference manual RM0351. Removed former Section 1: ARM® 32-bit Cortex®-M4 FPU core limitations, former Section 2.8.7: START bit is not cleared when the address is not acknowledged by the slave device and former Table 4: Cortex®-M4 FPU core limitations and impact on microcontroller behavior. Added Section 2.1.2: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used, Section 2.1.3: Store immediate overlapping exception return operation might vector to incorrect interrupt, Section 2.2: FW and its subsections, Section 2.3.22: HSE long start-up at low voltage, Section 2.3.23: Current injection from VDD to VDDA through analog switch voltage booster, Section 2.6.6: Memory-mapped read operations may fail when timeout counter is enabled, Section 2.7.8: Spurious temperature measurement due to spike noise, Section 2.16.2: MCU may remain stuck in LPTIM interrupt when entering Stop mode, Section 2.17.3: RTC interrupt can be masked by another RTC interrupt, Section 2.17.4: RTC_OUT on PB2 and RTC_REFIN on PB15 are not functional in Stop 2 mode, Section 2.18.7: Last received byte can be lost when using Reload mode with NBYTES > 1 and Section 2.18.8: Spurious master transfer upon own slave address match.
31-Oct-2018	7	Updated Section 2.4.1: Dummy read cycles inserted when reading synchronous memories, Section 2.4.2: Data corruption during burst read from FMC synchronous memory, and Section 2.4.3: FMC NOR/PSRAM controller bank switch with different BUSTURN durations. Updated Table 3: Summary of device limitations. Added Section 2.3.24: Unstable LSI when it clocks RTC or CSS on LSE, Section 2.3.25: FLASH_ECCR corrupted upon reset or power\u0002down occurring during Flash memory program or erase operation, Section 2.4.5: Spurious clock stoppage with continuous clock feature enabled, Section 2.4.6: FMC NOR/PSRAM: asynchronous read access on banks 2 to 4 returns wrong data when bank 1 is in synchronous mode (BURSTEN bit is set), Section 2.4.7: Wrong data read from a busy NAND memory, Section 2.5: DMA, Section 2.7.9: Writing ADCx_JSQR when JADCSTART and JQDIS are set might lead to incorrect behavior and Section 2.16.3: MCU may remain stuck in LPTIM interrupt when clearing event flag.
11-Feb-2020	8	Updated Section 2.3.25: FLASH_ECCR corrupted upon reset or power\u0002down occurring during Flash memory program or erase operation (formerly First double-word of Flash memory corrupted upon reset or power down while programming).
6-Jul-2021	9	Added Section 2.7.10: Selected external ADC inputs unduly clamped to VDD when all analog peripherals are disabled and Section 2.14.2: TSC signal-to-noise concern under specific conditions.
22-Sep-2021	10	Added Section 2.3.26: PC13 signal transitions disturb LSE, Section 2.9.1: Incorrect VREFBUF_OUT parameter values and Section 2.11.2: Excessive input current to OPAMP inverting input I/O ports. Updated Applicability section.
06-Dec-2024	11	Added STM32L471xx and STM32L475xx part numbers.

Date	Version	Changes
		Added errata: • SYSTEM: ADC_AWDy_OUT reset by non-guarded channels • FMC: Interruption of CPU burst read access to the end of an SDRAM row, FMC dynamic and static bank switching, and FMC NOR/PSRAM controller write protocol violation, and Missing information on prohibited 0xFF value of NAND transaction wait timing • QUADSPI: Memory-mapped access in indirect mode clearing QUADSPI_AR register, and Transmission error flag (TEF) in memory space access with ADMODE = 0 • ADC: Load multiple not supported by ADC interface, ADEN bit cannot be set immediately after the ADC calibration is done, New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0, Two consecutive context conversions fail when writing new context in ADC_JSQR just after previous context completion with JQDIS = 0 and JQM = 0, Unexpected regular conversion when two consecutive injected conversions are performed in Dual interleaved mode, ADC_AWDy_OUT reset by non-guarded channels, Injected data stored in the wrong ADC_JDRx registers, ADC slave data may be shifted in Dual regular simultaneous mode • DAC: DMA underrun flag not set when an internal trigger is detected on the clock cycle of the DMA request acknowledge • DFSDM: DFSDM triggers from EXTI require external interrupt management • LCD: LCD segments 23 and 24 alternate function cannot be configured independently • RNG: RNG clock error does not stop random numbers • AES: CCM authentication mode not compliant with NIST CMAC, Datatype initial configuration in GCM mode, and Wait until BUSY is low when suspending GCM encryption • TIM: One-pulse mode trigger not detected in master-slave reset + trigger configuration, Consecutive compare event missed in specific conditions, and Output compare clear not working with external counter reset • LPTIM: Device may remain stuck in LPTIM interrupt when clearing event flag and LPTIM events and PWM output are delayed by one kernel clock cycle • RTC: Calendar initialization may fail in case of consecutive INIT mode entry, Alarm flag may be repeatedly set when the core is stopped in debug, and A tamper event fails to trigger timestamp or timestamp overflow events during a few cycles after clearing TSF • I2C: Last-received byte loss in reload mode, OVR flag not set in underrun condition, Transmission stalled after first byte transfer, SDA held low upon SMBus timeout expiry in target mode, Wrong behavior in Stop mode when wake-up from Stop mode is disabled in I2C, and START bit is cleared upon setting ADDRCF, not upon address match • USART: Receiver timeout counter wrong start in two-stop-bit configuration, Data corruption due to noisy receive line, Received data may be corrupted upon clearing the ABREN bit, and Noise error flag set while ONEBIT is set • LPUART: Break request preventing TC flag from being set, RTS is active while RE = 0 or UE = 0, and Possible LPUART transmitter issue when using low BRR[15:0] value • SPI: BSY bit may stay high when SPI is disabled and SPI CRC corruption upon DMA transaction completion by another peripheral • SAI: , Last SAI_SCK clock pulse truncated upon disabling SAI master, Last SAI_MCLK clock pulse truncated upon disabling SAI master, and SAI_MCLK clock absent in a specific configuration Updated errata: • Updated all I2C errata to change master and slave into controller and target, respectively. • USART: Non-compliant sampling for NACK signal from smartcard • SWP SUSPENDED mode not supported when the MCU is in Stop 0 or Stop 1 mode: changed workaround from 'N' to 'A'.

Date	Version	Changes
		Removed errata: • System: *Extra consumption when VDDA is below 0.7 V*, *System clock is stopped during few µs when switching between Low-power run and Run modes*, and *PC13 signal transitions disturb LSE* • FMC: *Read burst access of nine words or more is not supported by FMC* • DAC: *PA4 and PA5 must be configured in analog mode when DAC1_OUT1 or DAC1_OUT2 respectively is connected to OPAMP input*, *PA4 and PA5 can't be used in output mode when DAC1_OUT1 or DAC1_OUT2 respectively is connected to on-chip peripherals*, *DAC cannot be used at low VDDA if APB1 frequency is higher than 40 MHz*, and *DAC characteristics can be degraded depending on the current sunk on DAC output* • OPAMP: *Excessive input current to OPAMP inverting input I/O ports* • VREFBUF: *Incorrect V_{REFBUF_OUT} parameter values*

Contents