STM32L451xx/452xx/462xx device errata

Applicability

This document applies to the part numbers of STM32L451xx/452xx/462xx 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 RM0394.

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
STM32L451xx	STM32L451CC, STM32L451CE, STM32L451RC, STM32L451RE, STM32L451VC, STM32L451VE
STM32L452xx	STM32L452CC, STM32L452CE, STM32L452RC, STM32L452RE, STM32L452VC, STM32L452VE
STM32L462xx	STM32L462CE, STM32L462RE, STM32L462VE

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32L451xx, STM32L452xx, STM32L462xx	Y	0x2001

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 STM32L451xx/452xx/462xx 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. Y
Core	2.1.1	Interrupted loads to SP can cause erroneous behavior	A
	2.1.2	VDIV or VSQRT instructions might not complete correctly when very short ISRs are used	A
	2.1.3	Store immediate overlapping exception return operation might vector to incorrect interrupt	A
System	2.2.1	Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled	A
	2.2.2	Spurious brown-out reset after short run sequence	A
	2.2.3	Full JTAG configuration without NJTRST pin cannot be used	A
	2.2.4	Current injection from V_{DD} to V_{DDA} through analog switch voltage booster	A
	2.2.5	Unstable LSI when it clocks RTC or CSS on LSE	P
	2.2.6	HSI48 ready interrupt capability is not supported	A
	2.2.7	FLASH_ECCR corrupted upon reset or power-down occurring during flash memory program or erase operation	A
	2.2.8	HSE oscillator long startup at low voltage	P
	2.2.9	LSE crystal oscillator may be disturbed by transitions on PC13	N
	2.2.10	SRAM write error	A
	2.2.11	Corrupted content of the backup domain due to a missed power-on reset after this domain supply voltage drop	A
	2.2.12	Debugging Sleep/Stop mode with WFE/WFI entry	A
	2.2.13	Possible incorrect code execution when transiting from SRAM2 to Flash memory	A
	2.2.14	Option validity error set after reset	A
FW	2.3.1	Code segment unprotected if non-volatile data segment length is zero	A
	2.3.2	Code and non-volatile data unprotected upon bank swap	A
DMA	2.4.1	DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear	A
QUADSPI	2.5.1	First nibble of data not written after dummy phase	A
	2.5.2	Wrong data from memory-mapped read after an indirect mode operation	A
	2.5.3	Memory-mapped read operations may fail when timeout counter is enabled	P
	2.5.4	Memory-mapped access in indirect mode clearing QUADSPI_AR register	P
	2.5.5	Transmission error flag (TEF) in memory space access with ADMODE = 0	A

Function	Section	Limitation	Status Rev. Y
SDMMC	2.6.1	Wrong CCRCFAIL status after a response without CRC is received	A
	2.6.2	MMC stream write of less than 8 bytes does not work correctly	A
ADC	2.7.3	New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0	A
	2.7.4	Two consecutive context conversions fail when writing new context in ADC_JSQR just after previous context completion with JQDIS = 0 and JQM = 0	A
	2.7.5	Unexpected regular conversion when two consecutive injected conversions are performed in Dual interleaved mode	A
	2.7.6	ADC_AWDy_OUT reset by non-guarded channels	A
	2.7.7	Injected data stored in the wrong ADC_JDRx registers	A
	2.7.8	ADC slave data may be shifted in Dual regular simultaneous mode	A
	2.7.9	Wrong ADC result if conversion done late after calibration or previous conversion	A
	2.7.10	Spurious temperature measurement due to spike noise	A
	2.7.11	Selected external ADC inputs unduly clamped to V_{DD} when all analog peripherals are disabled	A
DAC	2.8.1	DMA underrun flag not set when an internal trigger is detected on the clock cycle of the DMA request acknowledge	N
COMP	2.9.1	Comparator outputs cannot be configured in open-drain	N
TSC	2.10.2	TSC signal-to-noise concern under specific conditions	A
TIM	2.13.1	One-pulse mode trigger not detected in master-slave reset + trigger configuration	P
	2.13.2	Consecutive compare event missed in specific conditions	N
	2.13.3	Output compare clear not working with external counter reset	P
	2.13.4	HSE/32 is not available for TIM16 input capture if RTC clock is disabled or other than HSE	A
LPTIM	2.14.1	Device may remain stuck in LPTIM interrupt when entering Stop mode	A
	2.14.2	Device may remain stuck in LPTIM interrupt when clearing event flag	A
	2.14.3	LPTIM events and PWM output are delayed by one kernel clock cycle	A
	2.14.4	LPTIM1 outputs cannot be configured as open-drain	N
RTC and TAMP	2.15.1	RTC calendar registers are not locked properly	A
	2.15.2	RTC interrupt can be masked by another RTC interrupt	A
	2.15.3	Calendar initialization may fail in case of consecutive INIT mode entry	A
	2.15.4	Alarm flag may be repeatedly set when the core is stopped in debug	N
	2.15.5	A tamper event fails to trigger timestamp or timestamp overflow events during a few cycles after clearing TSF	N
	2.15.6	RTC_REFIN and RTC_OUT on PB2 not operating in Stop 2 mode	P
	2.15.8	V_{BAT} tamper limitation	A
	2.15.9	RTC_OUT redirection to PB2 does not release PC13	N
I2C	2.16.1	10-bit controller mode: new transfer cannot be launched if first part of the address is not acknowledged by the target	A
	2.16.3	Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period	P
	2.16.4	Spurious bus error detection in controller mode	A

Function	Section	Limitation	Status Rev. Y
I2C	2.16.5	Last-received byte loss in reload mode	P
	2.16.6	Spurious controller transfer upon own target address match	P
	2.16.8	OVR flag not set in underrun condition	N
	2.16.9	Transmission stalled after first byte transfer	A
	2.16.10	SDA held low upon SMBus timeout expiry in target mode	A
USART	2.17.1	RTS is active while RE = 0 or UE = 0	A
	2.17.2	Receiver timeout counter wrong start in two-stop-bit configuration	A
	2.17.3	Data corruption due to noisy receive line	A
	2.17.4	Received data may be corrupted upon clearing the ABREN bit	A
	2.17.5	Noise error flag set while ONEBIT is set	N
LPUART	2.18.1	RTS is active while RE = 0 or UE = 0	A
	2.18.2	Possible LPUART transmitter issue when using low BRR[15:0] value	P
	2.18.3	LPUART1 outputs cannot be configured as open-drain	N
SPI	2.19.1	BSY bit may stay high when SPI is disabled	A
	2.19.2	BSY bit may stay high at the end of data transfer in slave mode	A
	2.19.3	SPI master communication failure at high SYSCLK frequency within the specified voltage range	A
SAI	2.20.2	Last SAI_SCK clock pulse truncated upon disabling SAI master	N
	2.20.3	Last SAI_MCLK clock pulse truncated upon disabling SAI master	A
	2.20.4	SAI_MCLK clock absent in a specific configuration	A
bxCAN	2.21.1	bxCAN time-triggered communication mode not supported	N
USB	2.22.2	ESOF interrupt timing desynchronized after resume signaling	A
	2.22.3	Incorrect CRC16 in the memory buffer	N
	2.22.4	Buffer description table update completes after CTR interrupt triggers	A

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
ADC	2.7.1	Writing ADC_JSQR when JADCSTART and JQDIS are set may lead to incorrect behavior
	2.7.2	ADEN bit cannot be set immediately after the ADC calibration is done
TSC	2.10.1	Inhibited acquisition in short transfer phase configuration
RNG	2.11.1	RNG clock error does not stop random numbers
AES	2.12.1	TAG computation in GCM encryption mode
	2.12.2	CCM authentication mode not compliant with NIST CMAC
	2.12.3	Datatype initial configuration in GCM mode
	2.12.4	Wait until BUSY is low when suspending GCM encryption
RTC and TAMP	2.15.7	Setting GPIO properties of PC13 used as RTC_ALARM open-drain output
I2C	2.16.2	Wrong behavior in Stop mode when wake-up from Stop mode is disabled in I2C

Function	Section	Documentation erratum
I2C	2.16.7	START bit is cleared upon setting ADDRCF, not upon address match
SAI	2.20.1	Automatic restart upon late or anticipated frame error in I^2S slave mode not supported
USB	2.22.1	Possible packet memory overrun/underrun at low APB frequency

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 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.

2.2.2 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.

2.2.3 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.

2.2.4 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 at least one of the analog peripherals (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.

2.2.5 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.

2.2.6 HSI48 ready interrupt capability is not supported

Description

HSI48 ready interrupt feature described in the reference manual is not supported. The bit HSI48RDYIE in the register RCC_CIER is stuck at 0. It is not possible to set it to interrupt the CPU when the oscillator is ready.

Workaround

After switching the HSI48 internal oscillator on by setting the bit HSI48ON of the RCC_CRRCR register, poll the HSI48RDY flag of the RCC_CRRCR register until it goes high, which indicates that the HSI48 oscillator is ready to provide the clock to peripherals.

2.2.7 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.

2.2.8 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.

2.2.9 LSE crystal oscillator may be disturbed by transitions on PC13

Description

On LQFP and UFQFPN packages, the LSE crystal oscillator clock frequency can be incorrect when PC13 is toggling in input or output (for example when used for RTC_OUT1).

The external clock input (LSE bypass) is not impacted by this limitation.

The WLCSP and UFBGA packages are not impacted by this limitation.

Workaround

None.

Avoid toggling PC13 when LSE is used on LQFP and UFQFPN packages.

2.2.10 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 //SRAM1 address
LDR     R1, [R0, #+0]   //read access
MOV32   R0, #0x20008000 //next SRAM1 instance
LDR     R1, [R0, #+0]   //dummy read, no consequence on R1 value
…
MOV32   R0, #0x20020000 //SRAM2 address
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.

2.2.11 Corrupted content of the backup domain due to a missed power-on reset after this domain supply voltage drop

Description

The backup domain reset may be missed upon a power-on following a power-off, if its supply voltage drops during the power-off phase hitting a window, which is few mV wide before it starts to rise again. In this critical window, the flip-flops are no longer able to safely retain the information and the backup domain reset has not yet been triggered. This window is located in the range between 100 mV and 700 mV, with the exact position depending mainly on the device and on the temperature.

This missed reset results in unpredictable values of the backup domain registers. This may cause a spurious behavior (such as driving the LSCO output pin on PA2 or influencing backup functions).

Workaround

Apply one of the following measures:

- In the application, let the V_{DD} and V_{BAT} supply voltages fall to a level below 100 mV for more than 200 ms before a new power-on.

- If the above workaround cannot be applied, and the boot follows a power-on reset, erase the backup domain by software.
  When the application is using shutdown mode, user needs to discriminate between the power-on reset or an exit from a shutdown mode.
  For this purpose, at least one backup register must have been previously programmed with a BKP_REG_VAL value with 16 bits set and 16 bits cleared.
  Robustness of this workaround can be significantly improved by using a CRC rather than registers. The registers are subject to backup domain reset.
  The workaround consists of calculating the CRC of the backup registers: RCC_BDCR and RTC registers, excluding bits modified by HW.
  The CRC result can be stored in the backup register instead of a fixed value. This value needs to be updated for each modification of values covered by CRC, such as by using CRC peripheral.
  At the very beginning of the boot code, insert the following software sequence:
  1. Check the BORRSTF flag of the RCC_CSR register. If set, the reset is caused by a power on, or is exiting from shutdown mode.
  2. If BORRSTF flag is true, and the shutdown mode is used in the application, check that the backup register value is different from BKP_REG_VAL. When tamper detection is enabled,check that no tamper flag is set. If both conditions are met then the reset is caused by a power-on.
  3. If the reset is caused by a power-on, apply the following sequence:
     a. Enable the PWR clock in the RCC, by setting the PWREN bit.
     b. Enable the backup domain access in the PWR, by setting the DBP bit.
     c. Reset the backup domain, by:
        i. Writing 0x0001 0000 in the RCC_BDCR register, which sets the BDRST bit and clears other register bits that might not be reset.
        ii. reading the RCC_BDCR register, to make the reset time long enough
        iii. writing 0x0000 0000 in the RCC_BDCR register, to clear the BDRST bit
     d. Clear the BORRSTF flag by setting the RMVF bit of the RCC_CSR register.

2.2.12 Debugging Sleep/Stop mode with WFE/WFI entry

Description

When the Sleep debug or Stop debug mode is enabled (DBG_SLEEP bit or DBG_STOP bit is set in the DBGMCU_CR register), software debugging is permitted during Sleep or Stop mode. However, after wake-up, some unreachable instructions may be executed if the following conditions are met: • The application software has disabled the Prefetch queue. • The number of wait states configured for the flash memory interface is greater than zero. • The linker has placed the WFE or WFI instruction on a 4-byte aligned address (0x080xx xxx4).

Workaround

Choose one of the following measures to apply:

- Add three NOPs after WFI/WFE instruction.
- Keep one AHB master active during Sleep (For example: keep DMA1 or DMA2 RCC clock enable bit set).
- Execute WFI/WFE instruction from routines inside the SRAM.

2.2.13 Possible incorrect code execution when transiting from SRAM2 to Flash memory

Description

When transiting from the SRAM2 to the Flash memory, a failure of loading null operands to prefetch buffer may happen, provided that the SRAM2 is accessed through its native address range from 0x1000 0000 to 0x1000 8000. Depending on the instructions placed at the entry point to the Flash memory and on memory alignment, the possible consequence is an incorrect execution or a hard fault.

The instruction in SRAM2 that precedes the potential failure described is:

```
BX  Rm        ;Jump or return from function call
```

Other types of code execution transiting from SRAM2 to Flash memory, such as other branching instructions or interrupt servicing, do not cause the issue.

Note that inserting wait states or enabling/disabling prefetch in the Flash memory does not prevent this failure from occurring.

Workaround

Insert two NOP instructions in the Flash memory code at any entry point for code transiting from SRAM2 to the Flash memory.

Example 1:

```
BL  funC      ;Call of funC located in SRAM2
NOP           ;Entry point - return from call of funC function
NOP
```

Example 2:

```
Any instruction
NOP           ;Entry point - target of a jump from SRAM2
NOP
```

2.2.14 Option validity error set after reset

Description

On first production lot, the complement of one word inside the device configuration area is not correctly programmed. Although this word is not used for the device configuration, it is included in the test of configuration consistency after reset. As a consequence, the OPTVERR flag of the FLASH_SR register is set.

When using HAL drivers included in STM32CubeL4, this bit is tested prior to executing any Flash memory command and as a consequence, the FLASH_WaitForLastOperation() function fails.

Note: This limitation is fixed on parts with the production date code 721 (inclusive) and later.

Workaround

As this limitation has no impact to the device functionality, the OPTVERR flag may be ignored or cleared.

When using HAL, calling the __HAL_FLASH_CLEAR_FLAG(FLASH_FLAG_OPTVERR) function after reset allows clearing the OPTVERR flag and then using HAL Flash memory access commands normally.

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 QUADSPI

2.5.1 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.

2.5.2 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.*

2.5.3 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.

2.5.4 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

2.5.5 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.

2.6 SDMMC

2.6.1 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.

2.6.2 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 ADC

2.7.1 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.

2.7.2 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.

2.7.3 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.

2.7.4 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.

2.7.5 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.

2.7.6 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.

2.7.7 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).

2.7.8 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.

2.7.9 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.

2.7.10 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.

2.7.11 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.

2.8 DAC

2.8.1 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.

2.9 COMP

2.9.1 Comparator outputs cannot be configured in open-drain

Description

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

Workaround

None.

2.10 TSC

2.10.1 Inhibited acquisition in short transfer phase configuration

Description

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

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

2.10.2 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.

2.11 RNG

2.11.1 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.

2.12 AES

2.12.1 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.

2.12.2 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.

2.12.3 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.

2.12.4 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.

2.13 TIM

2.13.1 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.

2.13.2 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.

2.13.3 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).

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

Description

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

Workaround

Apply the following procedure:

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

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

2.14 LPTIM

2.14.1 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.

2.14.2 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.

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

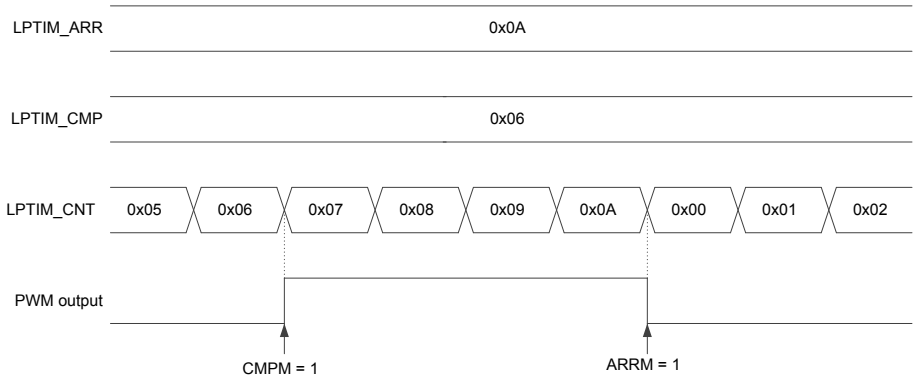
Description

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 1. 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.

2.14.4 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.

2.15 RTC and TAMP

2.15.1 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.

2.15.2 RTC interrupt can be masked by another RTC interrupt

Description

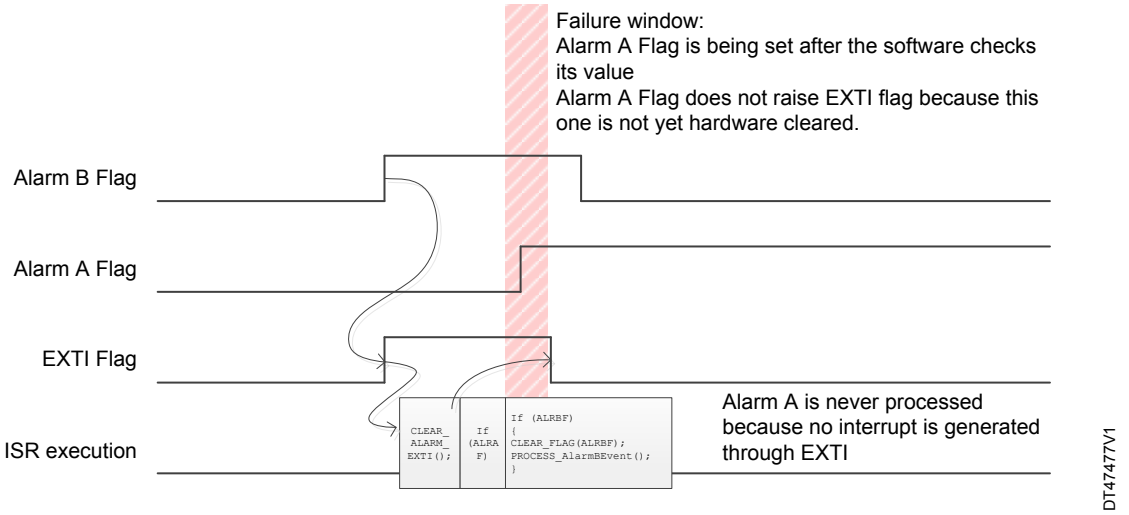
One RTC interrupt request can mask another RTC interrupt request if they share the same EXTI configurable line. For example, interrupt requests from Alarm A and Alarm B or those from tamper and timestamp events are OR-ed to the same EXTI line (refer to the *EXTI line connections* table in the *Extended 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 2. 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 */
    }
}
```

2.15.3 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.

2.15.4 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.

2.15.5 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.

2.15.6 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.

2.15.7 Setting GPIO properties of PC13 used as RTC_ALARM open-drain output

Description

Some reference manual revisions may omit the information that the PC13 GPIO must be set as input when the RTC_OR register configures PC13 as open-drain output of the RTC_ALARM signal.

Note: Enabling the internal pull-up function through the PC13 GPIO settings allows sparing an external pull-up resistor.

This is a documentation issue rather than a product limitation.

Workaround

No application workaround is required provided that the described GPIO setting is respected.

2.15.8 V_{BAT} tamper limitation

Description

Tamper functionality in V_{BAT} mode may be compromised with false tamper events when a pin that contains an alternate analog configuration is used. Only GPIOs with two power supplies (VDD and VDDA) are affected by this issue therefore pins with alternate analog configuration only are affected; pins with no analog functionality are not having this problem.

Workaround

Short V_{DD} fall time to zero causes the V_{BAT} to be engaged rapidly and correctly. Avoid lingering V_{DD} below 500 mV.

2.15.9 RTC_OUT redirection to PB2 does not release PC13

Description

When the RTC_OUT_RMP of RTC_CR is used to redirect RTC output to PB2, PC13 is not freed to be used as GPIO, or RTC input. However, it can still be used as RTC_ALARM output as described in the reference manual.

Workaround

None. Use different GPIO mapping.

2.16 I2C

2.16.1 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.

2.16.2 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 I^2C-bus may lead to the following:

1. BUSY flag is wrongly set when the MCU exits Stop mode. This prevents from initiating a transfer in 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 I^2C-bus frequency.

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

Workaround

No application workaround is required.

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

Description

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

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

The 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.

2.16.4 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.

2.16.5 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.

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

Workaround

- In 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. Specifically for I2C instances with independent clock, make sure that it is always read earlier than four APB clock cycles before the receipt of the last data bit of byte N and thus the TCR flag raising.

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

2.16.6 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 I^2C-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.

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

Description

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

2.16.8 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.

2.16.9 Transmission stalled after first byte transfer

Description

When the first byte to transmit is not prepared in the TXDATA register, two bytes are required successively, through TXIS status flag setting or through a DMA request. If the first of the two bytes is written in the I2C_TXDR register in less than two I2C kernel clock cycles after the TXIS/DMA request, and the ratio between APB clock and I2C kernel clock frequencies is between 1.5 and 3, the second byte written in the I2C_TXDR is not internally detected. This causes a state in which the I2C peripheral is stalled in 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.

2.16.10 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.

2.17 USART

2.17.1 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.

2.17.2 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.

2.17.3 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.

2.17.4 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.

2.17.5 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.

2.18 LPUART

2.18.1 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.

2.18.2 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.

2.18.3 LPUART1 outputs cannot be configured as open-drain

Description

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

Workaround

None.

2.19 SPI

2.19.1 BSY bit may stay high when SPI is disabled

Description

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

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

Workaround

When the SPI operates in:

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

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

Description

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

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

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

Workaround

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

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

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

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

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

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

2.19.3 SPI master communication failure at high SYSCLK frequency within the specified voltage range

Description

The SPI peripheral configured as master may spuriously generate an extra clock pulse when:

- the CPHA bit of the SPIx_CR1 register is set, and
- the BR[2:0] bitfield of the SPIx_CR1 register is at 001 (f_{PCLK} divided by four), and
- the SPI clock is about to be suspended or stopped (which occurs at the end of a data frame), and
- the SYSCLK frequency exceeds the values as per the following table

This leads to a de-synchronization of the SPI data flow.

Table 7. Maximum f_{SYSCLK} for safe SPI operation

SPI instance	f_{SYSCLK}(max) [MHz]	
	Range 1	Range 2
SPI1	>80 (OK)	>26 (OK)
SPI2	69.4	24.5
SPI3	>80 (OK)	>26 (OK)

Workaround

Depending on the application constraints, apply one of the following measures:

- Set the SYSCLK frequency to within the limits in the table.
- Choose a combination of AHB and APB prescaler settings that allows setting the BR[2:0] bitfield to a value other than 001 ($f_{PCLK}/4$).
- Use another SPI instance on another APB if this can avoid setting BR[2:0] to 001. If the other instance is available on the same pins through the GPIO alternate function, the same application hardware can be used.

2.20 SAI

2.20.1 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 I^2S (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.

2.20.2 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.

2.20.3 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.

2.20.4 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.

2.21 bxCAN

2.21.1 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.

2.22 USB

2.22.1 Possible packet memory overrun/underrun at low APB frequency

Description

Some data sheet and/or reference manual revisions may omit the information that 10 MHz minimum APB clock frequency is required to avoid USB data overrun/underrun issues.

Operating the USB peripheral with lower APB clock frequency may lead to:

- Overrun for *out* transactions - the USB peripheral fails to store the received data into the PBM before the next byte is received on the USB (PBM overrun). The USB cell detects an internal error condition, discards the last received byte, stops writing into the PBM, sends no acknowledge (forcing the host to retry the transaction), and informs the application by setting the PMAOVR flag/interrupt.
- Underrun for *in* transactions - the USB peripheral fails to read from the PBM the next byte to transmit before the transmission of the previous one is completed on the USB. The USB cell detects an internal error condition, stops reading from PBM, generates a bit stuffing error on the USB (forcing the host to retry the transaction), and informs the application by setting the PMAOVR flag/interrupt.

This is a documentation issue rather than a device limitation.

Workaround

No application workaround is required if the minimum APB clock frequency of 10 MHz is respected.

2.22.2 ESOF interrupt timing desynchronized after resume signaling

Description

Upon signaling resume, the device is expected to allow full 3 ms of time to the host or hub for sending the initial SOF (start of frame) packet, without triggering SUSP interrupt. However, the device only allows two full milliseconds and unduly triggers SUSP interrupt if it receives the initial packet within the third millisecond.

Workaround

When the device initiates resume (remote wake-up), mask the SUSP interrupt by setting the SUSPM bit for 3 ms, then unmask it by clearing SUSPM.

2.22.3 Incorrect CRC16 in the memory buffer

Description

Memory buffer locations are written starting from the address contained in the ADDRn_RX for a number of bytes corresponding to the received data packet length, CRC16 inclusive (that is, data payload length plus two bytes), or up to the last allocated memory location defined by BL_SIZE and NUM_BLOCK, whichever comes first. In the former case, the CRC16 checksum is written wrongly, with its least significant byte going to both memory buffer byte locations expected to receive the least and the most significant bytes of the checksum.

Although the checksum written in the memory buffer is wrong, the underlying CRC checking mechanism in the USB peripheral is fully functional.

Workaround

Ignore the CRC16 data in the memory buffer.

2.22.4 Buffer description table update completes after CTR interrupt triggers

Description

During OUT transfers, the correct transfer interrupt (CTR) is triggered a little before the last USB SRAM accesses have completed. If the software responds quickly to the interrupt, the full buffer contents may not be correct.

Workaround

Software should ensure that a small delay is included before accessing the SRAM contents. This delay should be 800 ns in Full Speed mode and 6.4 µs in Low Speed mode.

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
10-Feb-2017	1	Initial release.
30-Mar-2017	2	Change of document classification from ST Restricted to Public.
26-May-2017	3	Added section *Option validity error set after reset*
07-Sep-2017	4	Added section *Possible incorrect code execution when transiting from SRAM2 to Flash memory*
07-Dec-2018	5	Added errata: • VDIV or VSQRT instructions might not complete correctly when very short ISRs are used • Store immediate overlapping exception return operation might vector to incorrect interrupt • Spurious brown-out reset after short run sequence • Full JTAG configuration without NJTRST pin cannot be used • Unstable LSI when it clocks RTC or CSS on LSE • Current injection from VDD to VDDA through analog switch voltage booster • First double-word of Flash memory corrupted upon reset or power down while programming • Code segment unprotected if non-volatile data segment length is zero • Writing ADCx_JSQR when JADCSTART and JQDIS are set might lead to incorrect behavior • Spurious temperature measurement due to spike noise • MCU may remain stuck in LPTIM interrupt when entering Stop mode • MCU may remain stuck in LPTIM interrupt when clearing event flag • Last-received byte loss in reload mode • Spurious master transfer upon own slave address match • RTS is active while RE = 0 or UE = 0 • Wrong CRC in full-duplex mode handled by DMA with imbalanced setting of data counters • CRC error in SPI slave mode if internal NSS changes before CRC transfer Modified errata: • Option validity error set after reset • HSI48 ready interrupt capability is not supported moved to section System • Wrong ADC result if conversion done late after calibration or previous conversion • Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C moved to the documentation errata
03-Jun-2021	6	Added errata: • FLASH_ECCR corrupted upon reset or power-down occurring during flash memory program or erase operation • HSE oscillator long startup at low voltage • Selected external ADC inputs unduly clamped to VDD when all analog peripherals are disabled • TSC signal-to-noise concern under specific conditions • SPI master communication failure at high SYSCLK frequency within the specified voltage range

Date	Version	Changes
		Modified errata: • VDIV or VSQRT instructions might not complete correctly when very short ISRs are used • Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled • Full JTAG configuration without NJTRST pin cannot be used • Inhibited acquisition in short transfer phase configuration • Device may remain stuck in LPTIM interrupt when entering Stop mode • Device may remain stuck in LPTIM interrupt when clearing event flag • RTC interrupt can be masked by another RTC interrupt • Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C • Wrong data sampling when data setup time (tSU;DAT) is shorter than one I2C kernel clock period • bxCAN time-triggered communication mode not supported (workaround qualifier) Removed erratum First double-word of Flash memory corrupted upon reset or power down while programming (superseded with the erratum FLASH_ECCR corrupted upon reset or power-down occurring during flash memory program or erase operation).
19-May-2022	7	Added erratum SRAM write error. Updated erratum Device may remain stuck in LPTIM interrupt when clearing event flag. Added Section Important security notice.
14-Jun-2022	8	Updated erratum SRAM write error - correction of the SRAM instance size
23-Feb-2023	9	Added erratum LSE crystal oscillator may be disturbed by transitions on PC13. Modified erratum: • SRAM write error • SPI master communication failure at high SYSCLK frequency within the specified voltage range - now applicable to high fSYSCLK condition not only to the specific case of high $f_{PCLK} = f_{SYSCLK}$. Re-qualified to A and a workaround described.
23-Oct-2023	10	Added erratum: • Corrupted content of the domain due to a missed power\u0002on reset after this domain supply voltage drop • VBAT tamper limitation Modified erratum: LSE crystal oscillator may be disturbed by transitions on PC13
06-Dec-2024	11	Added STM32L452xx and STM32L462xx part numbers. Updated errata • Added UFQFPN packages in LSE crystal oscillator may be disturbed by transitions on PC13 • Corrupted content of the backup domain due to a missed power-on reset after this domain supply voltage drop • Updated all I2C errata to change master and slave into controller and target, respectively

Date	Version	Changes
		Added errata: • SYSTEM: Debugging Sleep/Stop mode with WFE/WFI entry • FW: Code and non-volatile data unprotected upon bank swap • DMA, DAC, RNG, AES, SAI, and USB errata • QUADSPI: Memory-mapped read operations may fail when timeout counter is enabled, Memory-mapped access in indirect mode clearing QUADSPI_AR register, and Transmission error flag (TEF) in memory space access with ADMODE = 0 • ADC: 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, and ADC slave data may be shifted in Dual regular simultaneous mode • 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: LPTIM events and PWM output are delayed by one kernel clock cycle • RTC and TAMP: 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: START bit is cleared upon setting ADDRCF, not upon address match, OVR flag not set in underrun condition, Transmission stalled after first byte transfer, and SDA held low upon SMBus timeout expiry in target mode • 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: 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 • SPI: *Wrong CRC in full-duplex mode handled by DMA with imbalanced setting of data counters* and *CRC error in SPI slave mode if internal NSS changes before CRC transfer*

Contents