STM32WB50CG STM32WB30CE

Errata sheet

STM32WB50CG/30CE device errata

Applicability

This document applies to STM32WB50CG/30CE 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 RM0471.

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 variants**

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32WB50CG	Y	0x2001
STM32WB30CE	A	0x2001
STM32WB50CG/30CE	X	0x2003

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 STM32WB50CG/30CE 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 2. Summary of device limitations

Function	Section	Limitation	Status	
			Rev. A, Y	Rev. X
Core	2.1.1	Interrupted loads to SP can cause erroneous behavior	A	A
	2.1.2	VDIV or VSQRT instructions might not complete correctly when very short ISRs are used	A	A
	2.1.3	Store immediate overlapping exception return operation might vector to incorrect interrupt	A	A
System	2.2.1	LSI2 not usable as RF low-power clock	A	A
	2.2.2	Unstable LSI1 when it clocks RTC or CSS on LSE	P	P
	2.2.3	Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled	A	A
	2.2.4	Full JTAG configuration without NJTRST pin cannot be used	A	A
	2.2.5	Auto-incrementing feature of the debug port cannot span more than 1 Kbyte	A	A
	2.2.6	Wrong DMAMUX synchronization and trigger input connections to EXTI	A	A
	2.2.7	Incomplete Stop 2 mode entry after a wakeup from debug upon EXTI line 48 event	A	A
	2.2.8	Flash OPTVERR flag is always set after system reset	A	A
	2.2.9	Overwriting with all zeros a flash memory location previously programmed with all ones fails	N	N
	2.2.10	FLASH_ECCR corrupted upon reset or power-down occurring during flash memory program or erase operation	A	A
	2.2.11	A system reset occurs when nRST_SHDW is set and nRST_STDBY is cleared and Shutdown mode is entered	A	A
	2.2.12	PH3 signal transitions disturb LSE	N	N
	2.2.13	Incorrect exit from Stop modes when the DBGMCU/DBG_STOP is enabled	A	A
	2.2.14	HSE is switched on immediately on Stop 2 exit, causing an undesired over-consumption	N	N
	2.2.15	When V_{DD} exceeds 2.5V in LDO configuration a glitch on HSE may create a hard fault on M4/M0	A	-
	2.2.16	System reset fails to initialize SRAM2 parity if SRAM2_RST is set	A	A
	2.2.17	SRAM2 content lost upon wakeup from Standby if SRAM2_RST is set	N	N
	2.2.18	With HSEGMC greater than 3, the HSE accuracy may not fulfill RF requirements	A	A

Function	Section	Limitation	Status	
			Rev. A, Y	Rev. X
System	2.2.19	Corrupted content of the backup domain due to a missed power-on reset after this domain supply voltage drop	A	A
	2.2.20	Overconsumption in Shutdown mode under specific conditions	A	A
	2.2.21	Excessive current on V_{DDA} or V_{DD} when V_{DD} level is different from V_{DDA} level	A	A
	2.2.22	CPU2 hard fault when changing C2HPRE clock divider in RCC_EXTCFGR register on CPU1	A	A
	2.2.23	Potential isolation issue between CPU1 and CPU2	P	P
	2.2.24	Debug HALT command in debug emulation generates HardFault	A	A
	2.2.25	LSCO on PH3/BOOT0 is interrupted if an alternate function is selected for PC3	A	A
	2.2.26	LSCO is erroneously output on PH3/BOOT instead of EVENT_OUT	A	A
Radio system	2.3.1	RF system wake-up clock set to HSE/1024 cannot be changed	A	A
DMA	2.4.1	DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear	A	A
DMAMUX	2.5.1	SOFx not asserted when writing into DMAMUX_CFR register	N	N
	2.5.2	OFx not asserted for trigger event coinciding with last DMAMUX request	N	N
	2.5.3	OFx not asserted when writing into DMAMUX_RGCFR register	N	N
	2.5.4	Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event	A	A
ADC	2.6.1	Wrong ADC result if conversion done late after calibration or previous conversion	A	A
	2.6.2	Selected external ADC inputs unduly clamped to V_{DD} when all analog peripherals are disabled	A	A
TIM	2.7.1	One-pulse mode trigger not detected in master-slave reset + trigger configuration	P	P
	2.7.2	Consecutive compare event missed in specific conditions	N	N
	2.7.3	Output compare clear not working with external counter reset	P	P
LPTIM	2.8.1	Device may remain stuck in LPTIM interrupt when entering Stop mode	A	A
	2.8.2	Device may remain stuck in LPTIM interrupt when clearing event flag	A	A
	2.8.3	LPTIM events and PWM output are delayed by one kernel clock cycle	A	A
	2.8.4	LPTIM clocked by LSI2 also requires LSI1	A	A
RTC and TAMP	2.9.1	RTC interrupt can be masked by another RTC interrupt	A	A
	2.9.2	Calendar initialization may fail in case of consecutive INIT mode entry	A	A
	2.9.3	Alarm flag may be repeatedly set when the core is stopped in debug	N	N
I2C	2.10.1	Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period	P	P
	2.10.2	Spurious bus error detection in controller mode	A	A
	2.10.3	Spurious controller transfer upon own target address match	P	P
	2.10.5	OVR flag not set in underrun condition	N	N
	2.10.6	Transmission stalled after first byte transfer	A	A
USART	2.11.1	Anticipated end-of-transmission signaling in SPI slave mode	A	A
	2.11.2	Data corruption due to noisy receive line	A	A

Function	Section	Limitation	Status	
			Rev. A, Y	Rev. X
SPI	2.12.1	BSY bit may stay high when SPI is disabled	A	A
	2.12.2	BSY bit may stay high at the end of data transfer in slave mode	A	A
	2.12.3	Wrong CRC in full-duplex mode handled by DMA with imbalanced setting of data counters	A	A

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

Table 3. **Summary of device documentation errata**

Function	Section	Documentation erratum
I2C	2.10.4	START bit is cleared upon setting ADDRCF, not upon address match

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®-M4 FPU core revision r0p1 is available from http://infocenter.arm.com. Only applicable information from the Arm errata notice is replicated in this document.

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 LSI2 not usable as RF low-power clock

Description

The calibration mechanism and jitter peaks cause the LSI2 clock to exceed the maximum 500 ppm frequency deviation specified in the Bluetooth® LE standard for low-speed clock.

Workaround

Clock the Bluetooth® peripheral using the LSE oscillator, instead of the LSI2.

2.2.2 Unstable LSI1 when it clocks RTC or CSS on LSE

Description

The LSI1 clock can become unstable (duty cycle different from 50 %) and its maximum frequency can become significantly higher than 32 kHz, when:

- LSI1 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 LSI1 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.

2.2.3 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.

2.2.4 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.

2.2.5 Auto-incrementing feature of the debug port cannot span more than 1 Kbyte

Description

The address auto-increment function of the AP2/AHB access ports is limited to 1 Kbyte of contiguous data.

Workaround

To write a contiguous chunk of data larger than 1 Kbyte to an AP2/AHB access port, split and write it in blocks not exceeding 1 Kbyte each.

2.2.6 Wrong DMAMUX synchronization and trigger input connections to EXTI

Description

By error, synchronization and trigger inputs of the DMAMUX peripheral are connected to interrupt output lines of the EXTI block, instead of being connected to its SYSCFG multiplexer output lines.

The EXTI interrupt lines exhibit a rising-edge transition upon each active transition (rising, falling or both) of corresponding GPIOs, as defined in the EXTI_RTSRx and EXTI_FTSRx registers.

As a consequence, the falling active edge option of the DMAMUX synchronization and trigger inputs is unusable because falling edges on these inputs do not occur upon GPIO events but upon clearing the EXTI interrupt pending flags (by setting the corresponding PIF bits of the EXTI_PRx register).

Workaround

For the DMAMUX synchronization and trigger events to occur upon determined rising or/and falling edge of the corresponding GPIOs:

- Set the desired active edge polarities of the corresponding GPIOs through the EXTI_RTSRx and EXTI_FTSRx registers.
- Set the active edge polarity to rising for all corresponding DMAMUX input lines, through the SPOL bits of the DMAMUX_CxCR register (for synchronization inputs) and the GPOL bits of the DMAMUX_RGxCR register (for trigger inputs).
- Ensure that EXTI interrupt pending flags corresponding to the GPIOs used for DMAMUX inputs are cleared in the EXTI interrupt service routine.

Note: This can be ensured if using the `HAL_GPIO_IrqHandler` function provided by STMicroelectronics.

2.2.7 Incomplete Stop 2 mode entry after a wakeup from debug upon EXTI line 48 event

Description

With the JTAG debugger enabled on GPIO pins and after a wakeup from debug triggered by an event on EXTI line 48 (CDBGPWRUPREQ), the device may enter in a state in which attempts to enter Stop 2 mode are not fully effective: The CPUs duly enter their respective CStop modes and are able to wake up so the software execution is not disturbed. However, the system does not transit to a low-power state and thus keeps a power consumption higher than expected.

Workaround

Before entering Stop 2 mode, mask *wakeup with interrupt request* from EXTI line 48 (CDBGPWRUPREQ) in both EXTI_IMR2 and EXTI_C2IMR2 registers.

2.2.8 Flash OPTVERR flag is always set after system reset

Description

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

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

Workaround

After reset, clear the OPTVERR flag in FLASH_SR register.

2.2.9 Overwriting with all zeros a flash memory location previously programmed with all ones fails

Description

Any attempt to re-program with all zeros (0x0000 0000 0000 0000) a flash memory location previously programmed with 0xFFFF FFFF FFFF FFFF fails and the PROGERR flag of the FLASH_SR register is set.

Note: Flash memory locations in the erased state (that is, not programmed) are not affected by this failure. They can be programmed with any value.

Workaround

None.

2.2.10 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.

2.2.11 A system reset occurs when nRST_SHDW is set and nRST_STDBY is cleared and Shutdown mode is entered

Description

When the following configuration is selected:

- nRST_SHDW is set
- nRST_STDBY is cleared,

a system reset is generated when Shutdown mode is entered.

Workaround

The only valid configuration to avoid a reset after entering into Shutdown mode is the following:

- nRST_SHDW is set
- nRST_STDBY is set.

2.2.12 PH3 signal transitions disturb LSE

Description

Toggling on the PH3 port disturbs the LSE clock. The PH3 port may not be usable at the same time as LSE is used.

Workaround

None

2.2.13 Incorrect exit from Stop modes when the DBGMCU/DBG_STOP is enabled

Description

When DBG_STOP is set in the microcontroller debug unit (DBGMCU), systick is still running and can trigger the CPU to exit Stop 0, Stop 1, or Stop 2, but the system remains in low power state. The CPU therefore fetches incorrect data from the inactive Flash which will generate a hard fault.

Workaround

Disable systick before entering in Stop mode when DBG_STOP is set in DBGMCU.

2.2.14 HSE is switched on immediately on Stop 2 exit, causing an undesired over-consumption

Description

An internal undefined value present on the RCC HSE input may trigger HSE to restart immediately when Stop 2 condition ends.

Workaround

None

2.2.15 When V_{DD} exceeds 2.5V in LDO configuration a glitch on HSE may create a hard fault on M4/M0

Description

When RF activity starts, the internal RF LDOs switches on at the same time and produces an extra HSE clock cycle pulse.

The CPUs process wrong data and Bluetooth® LE firmware becomes unstable.

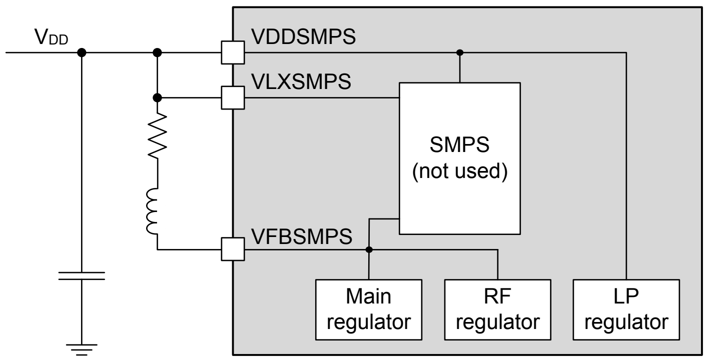
The Hard Fault can be observed in case SYSCLK=HSE. Otherwise the risk is radio malfunction since clocked from HSE directly too.

Workaround

Add a series inductance and resistor between VLXSMPS and VFBSMPS pins as illustrated in the figure.

The recommended inductor characteristics are the inductance 1.8 nH +/- 0.1 nH, self-resonance frequency of 6 Ghz +/-15 %, and rated current of 1000 mA. The resistor of 2.2 Ω recommended resistance must be dimensioned to dissipate 1 watt during 5 ns. Refer to Minimal BOM for STM32WB Series microcontrollers (AN5290) for more details.

Figure 1. **Workaround**



2.2.16 System reset fails to initialize SRAM2 parity if SRAM2_RST is set

Description

With the SRAM2_RST and SRAM2_PE bits set, the hardware does not correctly initialize the parity upon the system reset. As a consequence, read accesses to SRAM2 locations not previously written by the CPU trigger an NMI due to failing parity check.

Workaround

When enabling the SRAM2 parity by setting the SRAM2_PE bit, then before any reading of the unsecure SRAM2 memories (*a* and *b*), initialize them with the CPU1 then start the CPU2. The CPU2 takes care of initializing the SRAM2 secure memories (*a* and *b*).

Note: Unsecure SRAM2 zones that are never read do not require initialization by the CPU1.

2.2.17 SRAM2 content lost upon wakeup from Standby if SRAM2_RST is set

Description

With the SRAM2_RST option bit set, the retained SRAM2 content is unduly lost upon wakeup from Standby mode.

Note: With the SRAM2_RST option bit cleared, the retained SRAM2 content is duly kept upon wakeup from Standby mode.

Workaround

None.

2.2.18 With HSEGMC greater than 3, the HSE accuracy may not fulfill RF requirements

Description

HSEGMC[2:0] values above 3 may result in HSE instability that causes a breach of the RF requirements for HSE accuracy.

Workaround

Keep the value of the HSEGMC[2:0] bitfield lower than 4. That is, keep its bit 2 cleared.

2.2.19 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 backup domain access in the PWR, by setting the DBP bit.
     b. 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
     c. Clear the BORRSTF flag by setting the RMVF bit of the RCC_CSR register.

2.2.20 Overconsumption in Shutdown mode under specific conditions

Description

With the PWR_CR2 register PLS[2:0] bitfield value different from its reset value, the device exhibits an overconsumption of several hundreds of nanoamperes after entering Shutdown mode.

Workaround

Before entering Shutdown mode while PLS[2:0] is not at its reset value, set PLS[2:0] to its reset value and wait for at least the time specified in the following table.

T_J (°C)	Minimum waiting time (µs)			
	V_{DDA} = 2 V	V_{DDA} = 3 V	V_{DDA} = 3.3 V	V_{DDA} = 3.6 V
-40	26	38	42	45
25	20	29	32	35
50	18	27	29	32
105	15	23	25	27

2.2.21 Excessive current on V_{DDA} or V_{DD} when V_{DD} level is different from V_{DDA} level

Description

When V_{DDA} level is different from V_{DD} level, excessive current is present on V_{DDA} or V_{DD}.

Workaround

Maintain the V_{DDA} level equal to V_{DD} level.
Refer to the following electrical characteristics.

Table 4. **General operating conditions**

Symbol	Parameter	Conditions	Min	Max	Unit
V_{DDA}	Analog supply voltage	ADC or COMP used	V_{DD}-50 mV	min (V_{DD}+50 mV, 3.6)	V
		VREFBUF used	max (V_{DD}-50 mV, 2.35)	min (V_{DD}+50 mV, 3.6)	
		ADC, COMP, VREFBUF not used(1)	max (V_{DD} - 50 mV, 1.71)	min (V_{DD}+50 mV, 3.6)	

1. *When not used, V_{DDA} must be connected to V_{DD}.*

Table 5. **Absolute maximum ratings**

Symbol	Ratings	Voltage characteristics		
		Min	Max	Unit
$|\Delta (V_{DD}\ V_{DDA})|$	Difference between V_{DD} and V_{DDA} power pins	-	0. 3	V

2.2.22 CPU2 hard fault when changing C2HPRE clock divider in RCC_EXTCFGR register on CPU1

Description

When CPU1 switches the system clock frequency, the C2HPRE divider may need to be updated according to CPU2 frequency requirements. Changing the C2HPRE divider by CPU1 when CPU2 is running can cause a wrong instruction fetch on CPU2, which may lead to a hard fault.

Workaround

To avoid this situation, once CPU2 is started, the C2HPRE divider must not be updated by CPU1. CPU1 must call the SHCI_C2_SetSystemClock system command to request CPU2 to manage the switch of the system clock. The C2HPRE divider is then set in a specific procedure to avoid wrong instruction fetching by CPU2.

2.2.23 Potential isolation issue between CPU1 and CPU2

Description

Refer to security advisory SA0024 on www.st.com.

Workaround

Refer to security advisory SA0024 on www.st.com.

2.2.24 Debug HALT command in debug emulation generates HardFault

Description

When the CPU is in low-power mode with active debug emulation, and a debug HALT/RUN command sequence is submitted, the CPU wakes up and starts fetching data from the flash memory. As the flash memory is unavailable, the CPU receives random data from the bus, which causes HardFault.

Debug cannot be performed on the system when the CPU is in low-power mode, and the debug emulation is active.

Workaround

Change the debug sequence to perform a detach/attach procedure before any HALT/RUN command. Also, enable the CDBGPWRUPREQ wake-up on EXTI before the CPU goes in deep-sleep state. Do not set a breakpoint while in low-power mode, or only set a breakpoint when in Run or Sleep mode.

2.2.25 LSCO on PH3/BOOT0 is interrupted if an alternate function is selected for PC3

Description

LSCO on the PH3/BOOT0 pin stops operating when an alternate function is set on the PC3 pin.

Note: *This issue does not impact the low-speed clock that is still properly running.*

Workaround

Use LSCO on the PA2 or PC12 pin.

2.2.26 LSCO is erroneously output on PH3/BOOT instead of EVENT_OUT

Description

If PH3 is configured for EVENT_OUT and LSCO is enabled on any pin, the LSCO output is also on PH3 instead of EVENT_OUT.

Note: *This issue does not impact the low-speed clock that is still properly running.*

Workaround

Use the PH3/BOOT0 pin as LSCO output and use another pin as EVENT_OUT.

2.3 Radio system

2.3.1 RF system wake-up clock set to HSE/1024 cannot be changed

Description

Once the RFWKPSEL[1:0] bitfield of the RCC_CSR register is set to 11, setting it to any other value or performing a system reset prevents the RF system from waking with another wakeup clock.

Note: The value 11 of the bitfield selects HSE divided by 1024 as the RF system wakeup clock.

Workaround

Apply a POR to change the RF system wakeup clock.

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.

2.5 DMAMUX

2.5.1 SOFx not asserted when writing into DMAMUX_CFR register

Description

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

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

Workaround

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

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

Description

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

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

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

Workaround

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

2.5.3 OFx not asserted when writing into DMAMUX_RGCFR register

Description

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

Workaround

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

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

Description

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

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

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

Workaround

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

2.6 ADC

2.6.1 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.

2.6.2 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.

2.7 TIM

2.7.1 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.

2.7.2 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.

2.7.3 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).

2.8 LPTIM

2.8.1 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.

2.8.2 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.

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

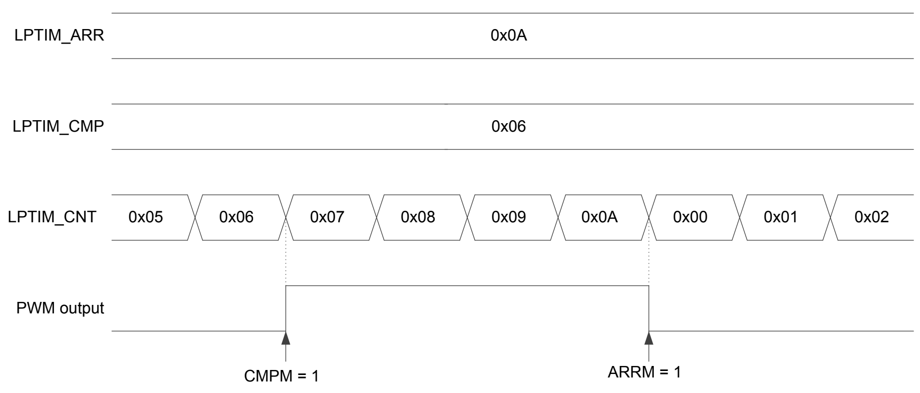
Description

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 2. 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.

2.8.4 LPTIM clocked by LSI2 also requires LSI1

Description

When

- LSI is selected as the LPTIMx clock, through the LPTIMxSEL[1:0] bitfield of the RCC_CCIPR register, and
- the LSI2 oscillator is enabled and automatically selected as the LSI clock source, and
- the LSI1 oscillator is idle,

the LPTIMx peripheral is expected to operate normally, clocked by LSI2.

However, as the LSI1 clock gates an internal signal required for LPTIMx, LPTIMx does not operate.

Workaround

Enable the LSI1 clock whenever LPTIMxSEL[1:0] selects LSI.

2.9 RTC and TAMP

2.9.1 RTC interrupt can be masked by another RTC interrupt

Description

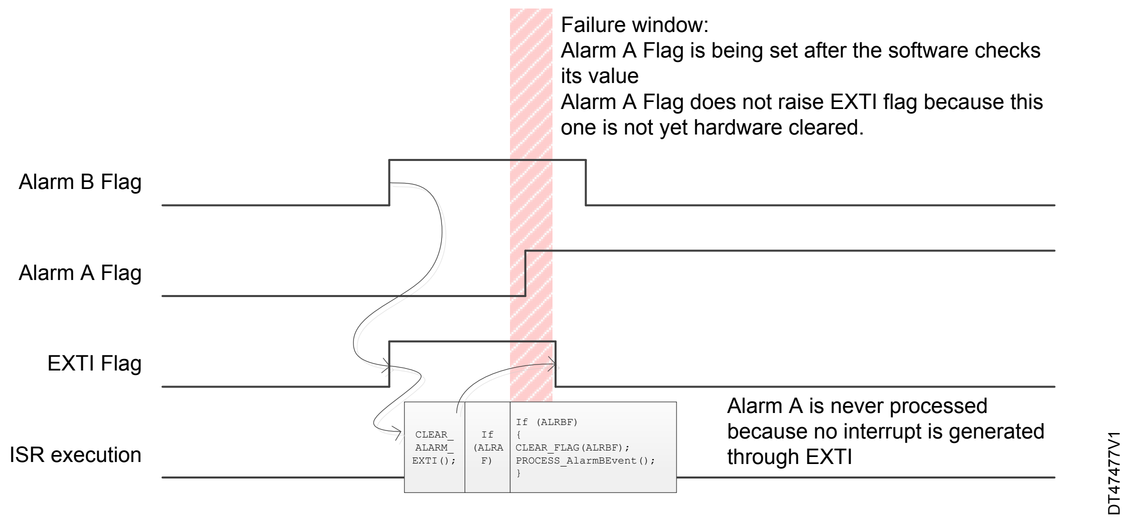
One RTC interrupt request can mask another RTC interrupt request if they share the same EXTI configurable line. For example, interrupt requests from Alarm A and Alarm B or those from tamper and timestamp events are OR-ed to the same EXTI line (refer to the *EXTI line connections* table in the *Extended 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 3. Masked RTC interrupt


Failure window:
Alarm A Flag is being set after the software checks its value
Alarm A Flag does not raise EXTI flag because this one is not yet hardware cleared.
Alarm B Flag
Alarm A Flag
EXTI Flag
ISR execution
CLEAR_ ALARM_ EXTI();
If (ALRAF)
If (ALRBF) { CLEAR_FLAG(ALRBF); PROCESS_AlarmBEvent(); }
Alarm A is never processed because no interrupt is generated through EXTI
DT47477V1

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 */
    }
}
```

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

Description

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

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

Workaround

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

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

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

Description

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

2.10 I2C

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

Description

The I^{2}C-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 I^{2}C-bus SDA line when $t_{SU;DAT}$ is smaller than one I2C kernel clock (I^{2}C-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 I^{2}C-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.

2.10.2 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 I^{2}C-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.

2.10.3 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.

2.10.4 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.

2.10.5 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.

2.10.6 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.

2.11 USART

2.11.1 Anticipated end-of-transmission signaling in SPI slave mode

Description

In SPI slave mode, at low USART baud rate with respect to the USART kernel and APB clock frequencies, the *transmission complete* flag TC of the USARTx_ISR register may unduly be set before the last bit is shifted on the transmit line.
This leads to data corruption if, based on this anticipated end-of-transmission signaling, the application disables the peripheral before the last bit is transmitted.

Workaround

Upon the TC flag rise, wait until the clock line remains idle for more than the half of the communication clock cycle. Then only consider the transmission as ended.

2.11.2 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.

2.12 SPI

2.12.1 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.

2.12.2 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.

2.12.3 Wrong CRC in full-duplex mode handled by DMA with imbalanced setting of data counters

Description

When SPI is handled by DMA in full-duplex master or slave mode with CRC enabled, the CRC computation may temporarily freeze for the ongoing frame, which results in corrupted CRC.

This happens when the receive counter reaches zero upon the receipt of the CRC pattern (as the receive counter was set to a value greater, by CRC length, than the transmit counter). An internal signal dedicated to receive-only mode is left unduly pending. Consequently, the signal can cause the CRC computation to freeze during a next transaction in which DMA TXE event service is accidentally delayed (for example, due to DMA servicing a request from another channel).

Workaround

Apply one of the following measures prior to each full-duplex SPI transaction:

- Set the DMA transmission and reception data counters to equal values. Upon the transaction completion, read the CRC pattern out from RxFIFO separately by software.
- Reset the SPI peripheral via peripheral reset register.

Important security notice

Revision history

Table 6. **Document revision history**

Date	Version	Changes
01-Jul-2019	1	Initial release.
03-Apr-2020	2	Added Table 3. Summary of device documentation errata and errata in Section 2: Description of device errata and summary tables: • FLASH_ECCR corrupted upon reset or power-down occurring during flash memory program or erase operation (supersedes the former erratum *First double-word of Flash memory corrupted upon reset or power-down while programming*) • A system reset occurs when nRST_SHDW is set and nRST_STDBY is cleared and Shutdown mode is entered • PH3 signal transitions disturb LSE • One-pulse mode trigger not detected in master-slave reset + trigger configuration • Consecutive compare event missed in specific conditions • Output compare clear not working with external counter reset • OVR flag not set in underrun condition • Transmission stalled after first byte transfer Modified errata: • Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled • Wrong DMAMUX synchronization and trigger input connections to EXTI • START bit is cleared upon setting ADDRCF, not upon address match moved to Table 3. Summary of device documentation errata
30-Jul-2020	3	Added errata in Section 2: Description of device errata and Table 2. Summary of device limitations: • Anticipated end-of-transmission signaling in SPI slave mode • Data corruption due to noisy receive line • Incorrect exit from Stop modes when the DBGMCU/DBG_STOP is enabled • HSE is switched on immediately on Stop 2 exit, causing an undesired over-consumption Coverage of STM2WB30CE.
15-Jul-2021	4	Added erratum Selected external ADC inputs unduly clamped to V_{DD} when all analog peripherals are disabled.
22-Nov-2021	5	Added device version *X* for STM32WB50CG/30CE. Added errata: • System reset fails to initialize SRAM2 parity if SRAM2_RST is set • SRAM2 content lost upon wakeup from Standby if SRAM2_RST is set Modified erratum LSI2 not usable as RF low-power clock: workaround re-qualified as *A*.
11-Mar-2022	6	Added erratum With HSEGMC greater than 3, the HSE accuracy may not fulfill RF requirements. Modified erratum Device may remain stuck in LPTIM interrupt when clearing event flag.
09-Jun-2022	7	Added section Important security notice and the errata: • LPTIM clocked by LSI2 also requires LSI1 • RF system wake-up clock set to HSE/1024 cannot be changed Modified errata: • Overwriting with all zeros a flash memory location previously programmed with all ones fails - a note added. • Alarm flag may be repeatedly set when the core is stopped in debug

Date	Version	Changes
14-Feb-2023	8	Added errata: • Corrupted content of the backup domain due to a missed power-on reset after this domain supply voltage drop • Overconsumption in Shutdown mode under specific conditions
19-Apr-2023	9	Added erratum Section 2.2.21: Excessive current on V_{DDA} or V_{DD} when V_{DD} level is different from V_{DDA} level.
27-Mar-2025	10	Added errata: • **System:** CPU2 hard fault when changing C2HPRE clock divider in RCC_EXTCFGR register on CPU1 • Potential isolation issue between CPU1 and CPU2 • Debug HALT command in debug emulation generates HardFault • **LPTIM:** LPTIM events and PWM output are delayed by one kernel clock cycle Modified errata: • **LPTIM:** Device may remain stuck in LPTIM interrupt when entering Stop mode • Device may remain stuck in LPTIM interrupt when clearing event flag (workaround qualifier) • **I2C:** All errata, replacing *master* and *slave* terms with *controller* and *target*, respectively. • **USART:** Data corruption due to noisy receive line
05-Nov-2025	11	Added System errata: • LSCO on PH3/BOOT0 is interrupted if an alternate function is selected for PC3 • LSCO is erroneously output on PH3/BOOT instead of EVENT_OUT Modified System erratum Corrupted content of the backup domain due to a missed power-on reset after this domain supply voltage drop.

Contents