STM32L552xx/562xx device errata

Applicability

This document applies to the part numbers of STM32L552xx/562xx 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 RM438.

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
STM32L552xx	STM32L552CE, STM32L552CC, STM32L552ME, STM32L552QC, STM32L552QE, STM32L552RC, STM32L552RE, STM32L552VC, STM32L552VE, STM32L552ZC, STM32L552ZE
STM32L562xx	STM32L562CE, STM32L562ME, STM32L562QE, STM32L562RE, STM32L562VE, STM32L562ZE

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32L552xx/562xx	A	0x1000
STM32L552xx/562xx	B	0x2000
STM32L552xx/562xx	Z	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 STM32L552xx/562xx 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. A	Rev. B	Rev. Z
Core	2.1.1	Floating-point state can be incorrectly cleared on some exception return faults	N	N	N
	2.1.2	Access permission faults are prioritized over unaligned Device memory faults	N	N	N
System	2.2.1	Full JTAG configuration without NJTRST pin cannot be used	A	A	A
	2.2.2	Overconsumption in Stop 2 mode	A	-	-
	2.2.3	PWR_SRR register is not secure	N	-	-
	2.2.4	SDMMC1SMEN bit of RCC_AHB2SMENR register only modifiable with word access	A	-	-
	2.2.5	HSE oscillator long startup at low voltage	P	P	P
	2.2.6	SMPS step down converter low-power mode	N	-	-
	2.2.7	Unstable LSI when it clocks RTC or CSS on LSE	P	-	-
	2.2.8	Regulator startup failure at low V_{DD}	N	-	-
	2.2.9	Voltage scaling range not selectable in SMPS bypass mode	P	-	-
	2.2.10	Read of Bank 2 while writing may give unpredictable results	N	-	-
	2.2.11	USB, CRS and UCPD may not wake properly from Stop 2	N	-	-
	2.2.12	Low-power run mode not transiting to *"Standby with"* modes	A	-	-
	2.2.13	PA15_PUPEN option bit setting inhibits the UCPD dead battery pull-down resistor on PB15	A	-	-
	2.2.14	Spurious setting of PC1 as secure	N	-	-
	2.2.15	Missing GPIOs on UFBGA132 and WLCSP81 packages	N	-	-
	2.2.16	SMPS regulation loss upon transiting into SMPS LP mode	P	P	-
	2.2.17	Unpredictable SMPS state at power-on	-	N	-
	2.2.18	FLASH_ECCR corrupted upon reset or power-down occurring during flash memory program or erase operation	A	A	A
	2.2.19	SRAM write error	A	A	A
	2.2.20	Corrupted content of the backup domain due to a missed power-on reset after this domain supply voltage drop	A	A	A
	2.2.21	LSE crystal oscillator may be disturbed by transitions on PC13	N	N	N
GPIO	2.3.1	GPIO assigned to DAC cannot be used in output mode when the DAC output is connected to on-chip peripheral	N	N	N

Function	Section	Limitation	Status		
			Rev. A	Rev. B	Rev. Z
FMC	2.4.1	Dummy read cycles inserted when reading synchronous memories	N	N	N
	2.4.2	Wrong data read from a busy NAND memory	A	A	A
	2.4.3	Data corruption upon a specific FIFO write sequence to synchronous PSRAM	A	A	A
OCTOSPI	2.5.1	Indirect read and automatic status-polling transfers without address phase not starting	A	A	A
	2.5.2	DTR octal-SPI indirect read data corrupted if last two bytes are read at a specific condition	A	A	A
	2.5.3	Spurious interrupt in AND-match polling mode with full data masking	A	A	A
	2.5.4	Hybrid wrap data transfer corruption upon an internal event	A	A	A
	2.5.5	Hybrid wrap registers not functional	A	A	A
	2.5.6	Odd address alignment and odd byte number not supported at specific conditions	A	A	A
	2.5.7	Read data can be corrupted at precise frequencies	N	-	-
	2.5.8	Memory-mapped write error response when DQS output is disabled	P	P	P
	2.5.9	Byte possibly dropped during an SDR read in clock mode 3 when a transfer gets automatically split	A	A	A
	2.5.10	Single-, dual- and quad-SPI modes not functional with DQS input enabled	N	N	N
	2.5.11	Additional bytes read in Indirect mode with DQS input enabled when data length is too short	A	A	A
	2.5.12	Data not sampled correctly on reads without DQS and with less than two cycles before the data phase	A	A	A
	2.5.13	Deadlock or write-data corruption after spurious write to a misaligned address in OCTOSPI_ AR register	N	N	N
	2.5.14	At least six cycles memory latency must be set when DQS is used for HyperBus™ memories	A	A	A
	2.5.16	Data write discarded in memory-mapped mode if a write to a misaligned address is directly followed by a request to the same address	A	A	A
	2.5.17	Setting the ABORT bit does not generate an error on the AHB bus for undefined-length incremental burst transfers	P	P	P
ADC	2.6.1	New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0	A	A	A
	2.6.2	Two consecutive context conversions fail when writing new context in ADC_JSQR just after previous context completion with JQDIS = 0 and JQM = 0	A	A	A
	2.6.3	Unexpected regular conversion when two consecutive injected conversions are performed in Dual interleaved mode	A	A	A
	2.6.4	ADC_AWDy_OUT reset by non-guarded channels	A	A	A
	2.6.5	Injected data stored in the wrong ADC_JDRx registers	A	A	A
	2.6.6	ADC slave data may be shifted in Dual regular simultaneous mode	A	A	A
	2.6.7	Wrong ADC result if conversion done late after calibration or previous conversion	A	A	A
	2.6.8	End of ADC conversion disturbing other ADCs	A	-	-
	2.6.9	Wrong ADC differential conversion result for channel 5	A	A	A

Function	Section	Limitation	Status		
			Rev. A	Rev. B	Rev. Z
DAC	2.7.1	Invalid DAC channel analog output if the DAC channel MODE bitfield is programmed before DAC initialization	A	A	A
	2.7.2	DMA underrun flag not set when an internal trigger is detected on the clock cycle of the DMA request acknowledge	N	N	N
COMP	2.8.1	Comparator outputs cannot be configured in open-drain	N	N	N
TSC	2.9.1	TSC signal-to-noise concern under specific conditions	A	A	A
TIM	2.10.1	One-pulse mode trigger not detected in master-slave reset + trigger configuration	P	P	P
	2.10.2	Consecutive compare event missed in specific conditions	N	N	N
	2.10.3	Output compare clear not working with external counter reset	P	P	P
	2.10.4	Bidirectional break mode not working with short pulses	N	N	N
	2.10.5	HSE/32 is not available for TIM16 input capture if RTC clock is disabled or other than HSE	A	A	A
LPTIM	2.11.1	Device may remain stuck in LPTIM interrupt when entering Stop mode	A	A	A
	2.11.2	ARRM and CMPM flags are not set when APB clock is slower than kernel clock	P	P	P
	2.11.3	Device may remain stuck in LPTIM interrupt when clearing event flag	P	P	P
	2.11.4	LPTIM events and PWM output are delayed by one kernel clock cycle	P	P	P
	2.11.5	LPTIM1/3 outputs cannot be configured as open-drain	N	N	N
RTC and TAMP	2.12.1	Notification of illegal access to secured registers is not reliable	N	N	N
	2.12.2	RTC_MISR and TAMP_MISR can be read by non-privileged accesses when privilege-protected	N	N	N
	2.12.3	RTC configuration changes ignored at specific conditions	A	-	-
	2.12.4	Calibration formula changes when LPCAL is set	A	A	A
	2.12.5	Calendar initialization may fail in case of consecutive INIT mode entry	A	A	A
	2.12.6	Alarm flag may be repeatedly set when the core is stopped in debug	N	N	N
	2.12.7	A tamper event fails to trigger timestamp or timestamp overflow events during a few cycles after clearing TSF	N	N	N
	2.12.8	REFCKON write protection associated to INIT KEY instead of CAL KEY	A	A	A
	2.12.9	Tamper flag not set on LSE failure detection	N	N	N
	2.12.10	Binary mode: SSR is not reloaded with 0xFFFF FFFF when SSCLR = 1	A	A	A
I2C	2.13.1	Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period	P	P	P
	2.13.2	Spurious bus error detection in master mode	A	A	A
	2.13.3	Spurious master transfer upon own slave address match	P	P	P
	2.13.5	OVR flag not set in underrun condition	N	N	N
	2.13.6	Transmission stalled after first byte transfer	A	A	A
	2.13.7	SDA held low upon SMBus timeout expiry in slave mode	A	A	A
USART	2.14.1	Anticipated end-of-transmission signaling in SPI slave mode	A	A	A
	2.14.2	Data corruption due to noisy receive line	A	A	A
	2.14.3	Received data may be corrupted upon clearing the ABREN bit	A	A	A

Function	Section	Limitation	Status		
			Rev. A	Rev. B	Rev. Z
USART	2.14.4	Noise error flag set while ONEBIT is set	N	N	N
LPUART	2.15.1	Possible LPUART transmitter issue when using low BRR[15:0] value	P	P	P
	2.15.2	LPUART1 outputs cannot be configured as open-drain	N	N	N
	2.15.3	Secure LPUART1 transmission on non-secure PA2 spuriously allowed	A	-	-
SPI	2.16.1	BSY bit may stay high when SPI is disabled	A	A	A
	2.16.2	BSY bit may stay high at the end of data transfer in slave mode	A	A	A
FDCAN	2.17.1	Desynchronization under specific condition with edge filtering enabled	A	A	A
	2.17.2	Tx FIFO messages inverted under specific buffer usage and priority setting	A	A	A
USB	2.18.1	ESOF interrupt timing desynchronized after resume signaling	A	A	A
	2.18.2	Incorrect CRC16 in the memory buffer	N	N	N
	2.18.3	Buffer description table update completes after CTR interrupt triggers	A	A	A
	2.18.4	USB may not operate correctly in Range 1	A	-	-
UCPD	2.19.1	UCPD BMC Tx eye diagram test failure	N	-	-

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

Table 4. Summary of device documentation errata

Function	Section	Documentation erratum
OCTOSPI	2.5.15	Automatic status-polling mode cannot be used with HyperFlash™ memories
I2C	2.13.4	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®-M33 core revision r0p2 is available from http://infocenter.arm.com.

2.1.1 Floating-point state can be incorrectly cleared on some exception return faults

Description

The Armv8-M architecture defines integrity checks which are performed before the exception return unstacking occurs. These check the validity of the EXC_RETURN value and raise a fault if they fail. Because of this erratum it is possible for the floating-point state to be incorrectly cleared when one of these faults occurs.

The floating-point state will be incorrectly cleared when all the following conditions are met:

- One of the following exception return integrity checks fails:
  - SFSR.INVER
  - UFSR.INVPC (exiting a handler that is not active)
  - UFSR.INVPC (EXC_RETURN[1] != 0)
  - SFSR.LSERR (when attempting to clear because of FPCCR.CLRONRET)
- The floating-point state would have been unstacked if there had been no fault (that is, EXC_RETURN[4] = 0, FPCCR.LSPACT = 0 and access is permitted to the FPU).

The floating-point state can be incorrectly cleared if software causes one of the faults mentioned above. The scenario that could be problematic is when a Secure exception calls a Non-secure function, which in turn attempts to return from the exception. This erratum allows the Non-secure function to clear the Secure floating-point context. Note that doing so will always cause a Secure fault to be raised and no Secure state is ever leaked to Non-secure.

Workaround

None.

2.1.2 Access permission faults are prioritized over unaligned Device memory faults

Description

A load or store which causes an unaligned access to Device memory will result in an UNALIGNED UsageFault exception. However, if the region is not accessible because of the MPU access permissions (as specified in MPU_RBAR.AP), then the resulting MemManage fault will be prioritized over the UsageFault.

The failure occurs when the MPU is enabled and:

- A load/store access occurs to an address which is not aligned to the data type specified in the instruction.
- The memory access hits one region only.
- The region attributes (specified in the MAIR register) mark the location as Device memory.
- The region access permissions prevent the access (that is, unprivileged or write not allowed).

The MemManage fault caused by the access permission violation will be prioritized over the UNALIGNED UsageFault exception because of the memory attributes.

Workaround

None. However, it is expected that no existing software is relying on this behavior since it was permitted in Armv7-M.

2.2 System

2.2.1 Full JTAG configuration without NJTRST pin cannot be used

Description

When using the JTAG debug port in Debug mode, the connection with the debugger is lost if the NJTRST pin (PB4) is used as a GPIO 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.

2.2.2 Overconsumption in Stop 2 mode

Description

In Stop 2 mode, the PA15 pull-up is enabled. This conflicts with UCPD dead battery functionality, which causes overconsumption.

Workaround

Set the UCPD_DBDIS bit of the PWR_CR3 register before entering Stop 2 mode.

2.2.3 PWR_SRR register is not secure

Description

When the system is secure (TZEN=1) and the LPMSEC bit is set, non-secure read/write access to PWR_SCR register is still possible.

Workaround

None. Clearing of status flags can be managed by secure boot firmware.

2.2.4 SDMMC1SMEN bit of RCC_AHB2SMENR register only modifiable with word access

Description

The SDMMC1SMEN bit of the RCC_AHB2SMENR register cannot be modified with byte and half-word accesses to the register. Only word access is effective.

Workaround

Only use word access to the RCC_AHB2SMENR register to modify the SDMMC1SMEN bit.

2.2.5 HSE oscillator long startup at low voltage

Description

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

Workaround

The following sequence is recommended:

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

2.2.6 SMPS step down converter low-power mode

Description

The SMPS step down converter low-power mode can only be selected when power consumption does not exceed 6 mA.

Workaround

None.

2.2.7 Unstable LSI when it clocks RTC or CSS on LSE

Description

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

- LSI clocks the RTC, or it clocks the clock security system (CSS) on LSE (which holds when the LSECSSON bit set), and
- the V_{DD} power domain is reset while the backup domain is not reset, which happens:
  - upon exiting Shutdown mode
  - if V_{BAT} is separate from V_{DD} and V_{DD} goes off then on
  - if V_{BAT} is tied to V_{DD} 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) and restore the backup domain configuration.

2.2.8 Regulator startup failure at low V_{DD}

Description

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

Workaround

None.

2.2.9 Voltage scaling range not selectable in SMPS bypass mode

Description

In SMPS bypass mode, it is not possible to change the voltage scaling range.

Workaround

If the device enters SMPS bypass mode following a software action, disable the SMPS bypass mode, change the voltage scaling range and revert to the SMPS bypass mode by enabling it again.

There is no workaround if the SMPS bypass mode is entered as consequence of V_{DD} drop below V_{DD}(min).

2.2.10 Read of Bank 2 while writing may give unpredictable results

Description

While writing user option bytes, concurrent reading of Bank 2 is possible. However, if the write operation leads to erasing the Bank 2 (for example, as a consequence of decreasing RDP level), the read results are unpredictable.

Workaround

None.

2.2.11 USB, CRS and UCPD may not wake properly from Stop 2

Description

USB, CRS and UCPD peripherals state may not be properly restored upon wakeup from Stop 2 mode.

Workaround

None.

2.2.12 Low-power run mode not transiting to *"Standby with"* modes

Description

It is not possible to switch from *Low-power run* mode to *Standby with SRAM2_4KB* or to *Standby with SRAM2_Full* mode.

Workaround

Switch to *Run* mode before entering one of *"Standby with"* modes.

2.2.13 PA15_PUPEN option bit setting inhibits the UCPD dead battery pull-down resistor on PB15

Description

Setting the PA15_PUPEN option bit of the FLASH_OPTR register disconnects the UCPD dead battery pull-down resistor and connects the JTDI pull-up resistor on PA15, which enables its use for JTAG. However, this also spuriously inhibits the UCPD *dead battery* pull-down resistor on PB15 (UCPD1_CC2).

Workaround

Use serial wire for debug.

2.2.14 Spurious setting of PC1 as secure

Description

With TrustZone enabled, mapping LPTIM2_IN1 on PC0 while configuring both LPTIM2 and PC0 as secure spuriously sets PC1 as secure.

Workaround

None.

2.2.15 Missing GPIOs on UFBGA132 and WLCSP81 packages

Description

On UFBGA132 and WLCSP81 packages, the following GPIOs are not bonded and they cannot be used by application:

- PB12 GPIO on STM32L562QxIxQ/STM32L552QxIxQ devices
- PE13, PE14, and PE15 on STM32L562MxYxP/STM32L552MxYxP devices

Workaround

None.

2.2.16 SMPS regulation loss upon transiting into SMPS LP mode

Description

The SMPS regulation may stop upon transiting into LP mode, which results in the loss of V_{core} power domain supply. As a consequence, the SMPS LP mode must not be selected.

Workaround

Use the SMPS HP mode only.

2.2.17 Unpredictable SMPS state at power-on

Description

After power-down/power-up sequence applied while the device is in SMPS bypass mode (BYPASS_RDY = 1), the SMPS state is unpredictable. For example, it can be stuck in Bypass state, or the SMPS fast soft start enable bit (SMPSFSTEN) can be set.

The limitation occurrence probability is low.

Workaround

None.

2.2.18 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.

2.2.19 SRAM write error

Description

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

Workaround

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

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

```
MOV32   R0, #0x20000000 //SRAM address
LDR     R1, [R0, #+0]   //read access
MOV32   R0, #0x20008000 //next SRAM instance, +32KBytes
LDR     R1, [R0, #+0]   //dummy read, no consequence on R1 value
MOV32   R0, #0x20010000
LDR     R1, [R0, #+0]
…
MOV32 R0, #0x2003F000 //the SRAM2 final cut starts at this 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.

2.2.20 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.

2.2.21 LSE crystal oscillator may be disturbed by transitions on PC13

Description

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.

Workaround

None.

Avoid toggling PC13 when LSE is used.

2.3 GPIO

2.3.1 GPIO assigned to DAC cannot be used in output mode when the DAC output is connected to on-chip peripheral

Description

When a DAC output is connected only to an on-chip peripheral, the corresponding GPIO is expected to be available as an output for any other functions.

However, when the DAC output is configured for on-chip peripheral connection only, the GPIO output buffer remains disabled and cannot be used in output mode (GPIO or alternate function). It can still be used in input or analog mode.

This limitation applies to DAC1_OUT1 and DAC1_OUT2 connected to PA4 and PA5, respectively.

Workaround

None.

2.4 FMC

2.4.1 Dummy read cycles inserted when reading synchronous memories

Description

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

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

Workaround

None.

2.4.2 Wrong data read from a busy NAND memory

Description

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

Workaround

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

2.4.3 Data corruption upon a specific FIFO write sequence to synchronous PSRAM

Description

The following specific succession of events may cause the FMC, with the write FIFO buffer enabled, to send corrupted data to a synchronous PSRAM:

1. The application software sends a data write burst to the FMC that places the data onto consecutive write FIFO buffer locations.
2. A write FIFO buffer address roll-over occurs (upon the write burst in point 1 or upon some subsequent writes to the FMC).
3. The application software writes a data byte to the FMC. The data byte reaches the same write FIFO buffer location as the first byte of the data write burst under point 1.
4. The application software writes data of any size to the FMC while the FMC is sending the data byte from point 3 to a synchronous PSRAM.

Under these circumstances, the data byte from point 3 (being sent out to PSRAM in point 4) is corrupted, or, alternatively, the data byte immediately following that data byte is corrupted.

Workaround

Use the FMC peripheral in Asynchronous write mode or disable the write FIFO buffer.

2.5 OCTOSPI

2.5.1 Indirect read and automatic status-polling transfers without address phase not starting

Description

Indirect read and automatic status-polling transfers, configured through the OCTOSPI_CCR register to contain command and SDR or DTR octal data phases but no address phase, do not start.

Workaround

Configure the transfer to contain address phase and no command phase, then send the command through the address OCTOSPI_AR register.

2.5.2 DTR octal-SPI indirect read data corrupted if last two bytes are read at a specific condition

Description

Indirect read from an DTR octal-SPI memory may lead to data corruption when all the following conditions are met:

- Number of bytes to read, defined in OCTOSPI_DLR register, is a multiple of 32 plus two, for example 34, 66, 98, and so on.
- The last two bytes are read with different requests.
- The second-last request read size is different from one byte.

Workaround

Apply one of the following measures:

- Read the last two bytes of a transfer with the same request.
- Read the last two bytes each with transfer size of one byte.

2.5.3 Spurious interrupt in AND-match polling mode with full data masking

Description

In AND-match polling mode with the MASK[31:0] bitfield set to 0x0000 0000 (all bits masked), a spurious interrupt may occur.

Workaround

Avoid setting the MASK[31:0] bitfield to 0x0000 0000.

2.5.4 Hybrid wrap data transfer corruption upon an internal event

Description

An internal event pertaining to TIMEOUT[15:0], CSBOUND[4:0], MAXTRAN[7:0], or REFRESH[31:0] bitfields may disturb any ongoing hybrid wrap transaction and result in corruption of the remaining data to transfer.

Workaround

Manage the TIMEOUT[15:0], CSBOUND[4:0], MAXTRAN[7:0], and REFRESH[31:0] bitfields such as to avoid any related internal event during hybrid wrap transactions.

2.5.5 Hybrid wrap registers not functional

Description

OCTOSPI_WPABR and OCTOSPI_WPTCR registers are not functional. As a consequence, external memory devices that require the setting of OCTOSPI_WPABR and OCTOSPI_WPTCR registers for the hybrid wrap because it is different from the settings of OCTOSPI_ABR and OCTOSPI_TCR registers used for the read, are not supported.

Note: Most memory devices allow the same settings for the hybrid wrap and the read.

Workaround

Only use memory devices allowing the same settings for the hybrid wrap and the read.

2.5.6 Odd address alignment and odd byte number not supported at specific conditions

Description

Odd address alignment and odd transaction byte number are not supported for some combinations of memory access mode, access type, and other settings. The following table summarizes the supported combinations, and provides information on consequences of accessing an illegal address and/or of setting an illegal number of bytes in a transaction.

Table 5. Summary of supported combinations

Memory access mode / other settings(1)	Access type(2)	Address allowed	Consequence of illegal address access(3)	Byte number allowed	Consequence of illegal byte number(3)
Single-SPI, dual-SPI, quad-SPI, RAM / DQM = 0 or octal-SPI/SDR mode	ind read	any	N/A	any	N/A
	mm read	any	N/A	any	N/A
	ind write	any	N/A	any	N/A
	mm write	any	N/A	any	N/A
Single-SPI, dual-SPI, quad-SPI, RAM / DQM = 1 or octal-SPI, RAM / DTR mode, no RDS, no WDM	ind read	even	ADDR[0] cleared	even	DLR[0] cleared
	mm read	any	N/A	any	N/A
	ind write	even	ADDR[0] cleared	even	DLR[0] cleared
	mm write	even	slave error	even	last byte lost
Octal-SPI, RAM / DTR mode, with RDS or WDM or HyperBus™	ind read	even	ADDR[0] cleared	even	DLR[0] cleared
	mm read	any	N/A	any	N/A
	ind write	any	N/A	any	N/A
	mm write	any	N/A	any	N/A

1. *"RDS" = read data strobe, "WDM" = write data mask*
2. *"ind read" = indirect read, "mm read" = memory-mapped read, "ind write" = indirect write, "mm write" = memory-mapped write*
3. *"N/A" = not applicable*

Workaround

Avoid illegal address accesses and illegal byte numbers in transactions.

2.5.7 Read data can be corrupted at precise frequencies

Description

At certain precise frequencies, the OCTOSPI may overrun its RxFIFO and result in the following:

- in all modes except for DTR octal-SPI mode: a byte of data corrupted when reading from the memory
- in DTR octal-SPI mode: two bytes of data corrupted when reading from the memory

Both indirect and memory-mapped read operations can be affected in all configurations for any type of memory.

Workaround

None.

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

Description

If the DQSE control bit of the OCTOSPI_WCCR register is cleared for memories without DQS pin, it results in an error response for every memory-mapped write request.

Workaround

When doing memory-mapped writes, set the DQSE bit of the OCTOSPI_WCCR register, even for memories that have no DQS pin.

2.5.9 Byte possibly dropped during an SDR read in clock mode 3 when a transfer gets automatically split

Description

When reading a continuous stream of data from sequential addresses in a serial memory, the OCTOSPI can interrupt the transfer and automatically restart it at the next address when features generating transfer splits (CSBOUND, REFRESH, TIMEOUT or MAXTRAN) are active. Thus, a single continuous transfer can effectively be split into multiple smaller transfers.

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

Workaround

Use clock mode 0 (CKMODE bit of the OCTOSPI_DCR1 register cleared) when in SDR mode.

2.5.10 Single-, dual- and quad-SPI modes not functional with DQS input enabled

Description

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

Workaround

None.

2.5.11 Additional bytes read in Indirect mode with DQS input enabled when data length is too short

Description

Extra byte reception may appear when the two conditions below are met at the same time:

- Data read in Indirect-read mode with DQS enabled (DQSE bit of the OCTOSPI_CCR register set)
- The number of cycles for data read phase is less than the sum of the number of cycles required for (command + address + alternate-byte + dummy) phases.

Workaround

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

2.5.12 Data not sampled correctly on reads without DQS and with less than two cycles before the data phase

Description

A command is composed of five phases:

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

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

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

Workaround

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

- Send one byte of address in SDR quad-SPI mode (ADMODE = 011, ADSIZE = 00, ADDDTR = 0).
- Send two bytes of address in SDR octal-SPI mode (ADMODE = 100, ADSIZE = 01, ADDDTR = 0).
- Send four bytes of address in DTR octal-SPI mode (ADMODE = 100, ADSIZE = 11, ADDDTR = 1).

2.5.13 Deadlock or write-data corruption after spurious write to a misaligned address in OCTOSPI_AR register

Description

Upon writing a misaligned address to OCTOSPI_AR just before switching to Memory-mapped mode (without first triggering the indirect write operation), with the OCTOSPI configured as follows:

- FMODE = 00 in OCTOSPI_CR (Indirect write mode)
- DQSE = 1 in OCTOSPI_CCR (DQS active)

then, the OCTOSPI may be deadlocked on the first memory-mapped request or the first memory-mapped write to memory (and any sequential writes after it) may be corrupted.

An address is misaligned if:

- the address is odd and the OCTOSPI is configured to send two bytes of data to the memory every cycle (octal-DTR mode or dual-quad-DTR mode), or
- the address is not a multiple of four when the OCTOSPI is configured to send four bytes of data to the memory (16-bit DTR mode or dual-octal DTR mode).

If the OCTOSPI_AR register is reprogrammed with an aligned address (without triggering the indirect write between the two writes to OCTOSPI register), the data sent to the memory during the indirect write operation are also corrupted.

Workaround

None.

2.5.14 At least six cycles memory latency must be set when DQS is used for HyperBus™ memories

Description

For HyperBus™ memories, the TACC[7:0] bitfield of the OCTOSPI_HLCR register enables the setting of the memory latency in number of clock cycles. These dummy cycles are inserted between the address and the data phases during read operations.

When the DQS signal is used for HyperBus™ memories, and the number of latency clock cycles programmed in TACC[7:0] is lower than six, a deadlock occurs during read operations.

Workaround

Configure the memory and the octo-SPI controller to have at least six clock cycles of latency.

2.5.15 Automatic status-polling mode cannot be used with HyperFlash™ memories

Description

Some reference manuals mention that the automatic status-polling mode can be used with the HyperBus™ protocol. This is not possible since HyperFlash™ memories require two steps to read the status register (a write operation followed by a read command), while the automatic status-polling mode, already implemented in the regular-command protocol, requires a single read instruction to read back the status register.

This is a documentation issue rather than a product limitation.

Workaround

None.

2.5.16 Data write discarded in memory-mapped mode if a write to a misaligned address is directly followed by a request to the same address

Description

In memory-mapped mode with DQS enabled, a data write is discarded if the write targets a misaligned address and is directly followed by a request (cycle by cycle on AHB) to the same address.

An address is misaligned if the address is odd and the OCTOSPI is configured to send two bytes of data to the memory on every cycle that is targeted by one of the following modes:

- Octal DTR
- Dual-quad DTR

Workaround

Use one of the following measures:

- Configure the OCTOSPI to issue commands to aligned addresses, that is to an even address when two bytes are transferred during each clock cycle.
- Avoid consecutive back-to-back (AHB cycle by cycle) accesses to the memory after a write to a memory mapped at the same address. Instead, insert a NOP (no operation) or a software delay between the two accesses.

2.5.17 Setting the ABORT bit does not generate an error on the AHB bus for undefined-length incremental burst transfers

Description

An AHB error is expected to be generated when the ABORT bit of the OCTOSPI_CR register is set while a request is ongoing.

Instead, the controller does not trigger any AHB error if the ongoing request is an undefined-length incremental burst AHB transfer.

An AHB error is generated for all other transfer types.

Workaround

When possible, wait for the end of the transfer before setting the ABORT bit.

2.6 ADC

2.6.1 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.

2.6.2 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.

2.6.3 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.

2.6.4 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.

2.6.5 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).

2.6.6 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.

2.6.7 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.

2.6.8 End of ADC conversion disturbing other ADCs

Description

The end-of-conversion event of an ADC instance disturbs the reference voltage, causing a conversion error to any other ADC instances with conversion in progress.

Workaround

With concurrent operation of multiple ADCs, avoid the end of conversion of one ADC to occur during the conversion phase of another ADC. For example, set them all to the same resolution and the same sampling duration, and start their sampling phase at the same time.

2.6.9 Wrong ADC differential conversion result for channel 5

Description

The ADC (ADC1 or ADC2) configured in differential mode with the channel 5 selected provides wrong conversion result.

Workaround

Set an unused SQxx[4:0] bitfield of the corresponding ADC_SQRx register, or an unused JSQxx[4:0] bitfield of the ADC_JSQR register, to channel 6.

For example, write the SQ16[4:0] bitfield of the ADC_SQR4 register with 00110 when the L[3:0] bitfield of the ADC_SQR1 register is set to 0001.

2.7 DAC

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

Description

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

Workaround

Apply the following sequence:

1. Perform one write access to any data register.

2. Program the MODE bitfield of the DAC_MCR register.

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

Description

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

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

Workaround

None.

2.8 COMP

2.8.1 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.

2.9 TSC

2.9.1 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.

2.10 TIM

2.10.1 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.

2.10.2 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.

2.10.3 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).

2.10.4 Bidirectional break mode not working with short pulses

Description

The TIM_BKIN and TIM_BKIN2 I/Os can be configured in bidirectional mode using the BKBID and BK2BID bits in the TIMx_BDTR register, to be forced to 0 when a break/break2 event occurs. The bidirectional break/break2 mode is not functional when the pulse width on break/break2 input is lower than two tim_ker_clk periods.

This limitation is also valid when software break events are generated (the break event is correctly generated internally but not reflected on break inputs).

Workaround

None.

For applications that can afford some latency in bidirectional break mode, the break interrupt can eventually be enabled, for the CPU to verify the break input state and force it to zero when a break/break2 event occurred.

2.10.5 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.

2.11 LPTIM

2.11.1 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.

2.11.2 ARRM and CMPM flags are not set when APB clock is slower than kernel clock

Description

When LPTIM is configured in one shot mode and APB clock is lower than kernel clock, there is a chance that ARRM and CMPM flags are not set at the end of the counting cycle defined by the repetition value REP[7:0]. This issue can only occur when the repetition counter is configured with an odd repetition value.

Workaround

To avoid this issue the following formula must be respected:

{ARR, CMP} ≥ KER_CLK / (2* APB_CLK),

where APB_CLK is the LPTIM APB clock frequency, and KER_CLK is the LPTIM kernel clock frequency. ARR and CMP are expressed in decimal value.

Example: The following example illustrates a configuration where the issue can occur:

- APB clock source (MSI) = 1 MHz , Kernel clock source (HSI) = 16 MHz
- Repetition counter is set with REP[7:0] = 0x3 (odd value)

The above example is subject to issue, unless the user respects:

{CMP, ARR} ≥ 16 MHz / (2 * 1 MHz)

→ ARR must be ≥ 8 and CMP must be ≥ 8

Note: REP set to 0x3 means that effective repetition is REP+1 (= 4) but the user must consider the parity of the value loaded in LPTIM_RCR register (=3, odd) to assess the risk of issue.

2.11.3 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.

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

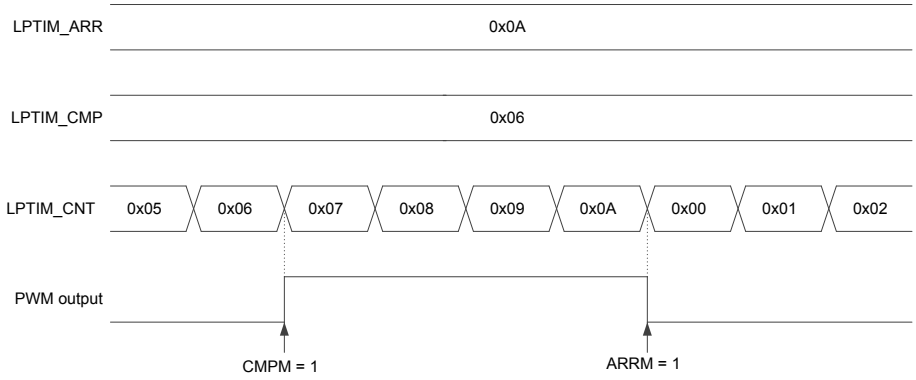
Description

The compare match event (CMPM), auto reload match event (ARRM), update event (UE), 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.

2.11.5 LPTIM1/3 outputs cannot be configured as open-drain

Description

LPTIM1/3 outputs are set in push-pull mode regardless of the configuration of corresponding GPIO outputs.

Workaround

None.

2.12 RTC and TAMP

2.12.1 Notification of illegal access to secured registers is not reliable

Description

When an RTC or a TAMP register is globally protected against non-secure accesses, the RTC and TAMP illegal access flag should be raised in the TrustZone illegal access controller upon non-secure accesses. However, the operation of this flag is not reliable. Consequently, it must not be used by the application.

Note: The register protection operates correctly: a write-secure-protected register ignores non-secure writes and a read-secure-protected register always returns zero upon non-secure reads.

Workaround

None.

2.12.2 RTC_MISR and TAMP_MISR can be read by non-privileged accesses when privilege-protected

Description

The RTC_MISR register bits can be read by non-privileged accesses even if their corresponding feature is configured with privilege protection.

The TAMP_MISR register bits can be read by non-privileged accesses even if the TAMPPRIV bit of the TAMP_PRIVCR register is set.

Workaround

None.

2.12.3 RTC configuration changes ignored at specific conditions

Description

Writes to some register bits may be ignored if done within a short period after exiting Stop or Standby mode and entering Stop or Standby mode again.

The register is correctly written, but the bit value is not propagated in the RTC kernel if the duration in Run or Sleep mode is too short. This concerns the WUTE (wakeup timer enable) bit, the ALRAE and ALRBE (Alarm A and Alarm B enable) bits, the TAMPxE (Tamper x enable) bits, all bits of RTC_CALR (the RTC calibration register), and the CWUTF (clear wakeup timer flag) bit.

The following paragraphs describe the failure mechanism for each function.

Enabling (or disabling) the wakeup timer:

1. The device is in Stop or Standby mode with the wakeup timer disabled (or enabled).
2. The device wakes up from low-power mode and enables (or disables) the wakeup timer.
3. The device enters Stop or Standby mode.

If the duration of the step 2 (device in Run or Sleep mode) is less than one RTCCLK period, the WUTE bit value change may not be taken into account.

Enabling (or disabling) alarm A or alarm B:

1. The device is in Stop or Standby mode with the alarm disabled (or enabled).
2. The device wakes up from low-power mode and enables (or disables) the alarm.
3. The device enters Stop or Standby mode.

If the duration of the step 2 (device in Run or Sleep mode) is less than two RTCCLK periods, the ALRAE or ALRBE bit value change may not be taken into account.

Enabling a tamper:

1. The device is in Stop or Standby mode with all tampers disabled.
2. The device wakes up from low-power mode and enables at least one tamper.
3. The device enters Stop or Standby mode.

If the duration of the step 2 (device in Run or Sleep mode) is less than two RTCCLK periods, the tamper may remain disabled.

Calibration register value change:

1. The device is in Stop or Standby mode with the RECALPF bit cleared.
2. The device wakes up from low-power mode and changes the RTC_CALR value.
3. The device enters Stop or Standby mode.

If the duration of the step 2 (device in Run or Sleep mode) is less than two RTCCLK periods, the RTC_CALR new value may not be taken into account.

Clearing wakeup timer flag:

1. The device is in Stop or Standby mode and WUTF is set.
2. The device wakes up from low-power mode and clears WUTF by setting the CWUTF bit of the RTC_SCR register.
3. The device enters Stop or Standby mode.

If the duration of the step 2 (device in Run or Sleep mode) is less than two RTCCLK periods, the WUTF bit may be stuck low and cannot be set when the wakeup timer reaches zero again.

Note: The same failures occur if the DBP (disable backup domain write protection) bit of the PWR register is set before changing the RTC configuration, and is cleared soon after.

Workaround

Always keep the DBP bit set. When the device wakes up (step 2): clear the RSF flag of the RTC_ICSR register and wait until it is set again before entering Stop or Standby mode. In case the BYPSHAD bit of the RTC_CR register is set, clear it before the RSF flag is set. The BYPSHAD bit can then be set again by software.

2.12.4 Calibration formula changes when LPCAL is set

Description

When the LPCAL bit is set, the frequency calibration formula unduly becomes:

$f_{CAL} = f_{RTCCLK} \times \left[\left(2^{20} - 1\right) / \left(2^{20} - 1 + CALM - CALP \times 512\right)\right]$

instead of:

$f_{CAL} = f_{RTCCLK} \times \left[2^{20} / \left(2^{20} + CALM - CALP \times 512\right)\right]$

As a consequence, the RTC frequency in the application that keeps the LPCAL bit set (to reduce power consumption) is slightly different from the frequency measured with the LPCAL bit cleared.

Workaround

In an application keeping the LPCAL bit set, apply a compensation reflecting the difference of the frequency formulas.

Note: LPCAL remains set when a new calibration value is applied. Checking the calibration result is only for validation or test purposes.

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

Description

If the INIT bit of the RTC_ICSR 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 occurring 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.

2.12.6 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.

2.12.7 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.

2.12.8 REFCKON write protection associated to INIT KEY instead of CAL KEY

Description

The write protection of the REFCKON bit is unlocked if the key sequence is written in RTC_WPR with the privilege and security rights set by the INITPRIV and INITSEC bits, instead of being set by the CALPRIV and CALSEC bits.

Workaround

Unlock the INIT KEY before writing REFCKON.

2.12.9 Tamper flag not set on LSE failure detection

Description

With the timestamp on tamper event enabled (the TAMPTS bit of the RTC_CR register set), the LSE failure detection (LSE clock stopped) event connected to the internal tamper 3 fails to raise the ITAMP3F and ITAMP3MF flags, although it duly erases or blocks (depending on the internal tamper 3 configuration) the backup registers and other device secrets, and the RTC and TAMP peripherals resume normally upon the LSE restart.

Note: As expected in this particular case, the TSF and TSMF flags remain low as long as LSE is stopped as they require running RTCCLK clock to operate.

Workaround

None.

2.12.10 Binary mode: SSR is not reloaded with 0xFFFF FFFF when SSCLR = 1

Description

When SSCLR bit of the RTC_ALRMxSSR register is set when in binary mode, SSR is reloaded with 0xFFFF FFFF at the end of the ck_apre cycle when RTC_SSR is set to RTC_ALRxBINR (*x* stands for either A or B)

RTC_SSR is not reloaded with 0xFFFF FFFF if RTC_ALRxBINR is modified while RTC_SSR is set to RTC_ALRxBINR. Rather, SSR continues to decrement.

Workaround

The workarounds are described for alarm A, and can be applied in the same manner for alarm B. Two workarounds are proposed, the second one requires to use the second alarm.

- Wait for one ck_apre cycle after an alarm A event before changing the RTC_ALRABINR register value.
- Do not reprogram RTC_ALRABINR following the alarm A event itself. Instead, use alarm B configured with RTC_ALRBBINR set to 0xFFFF FFFF, and reprogram RTC_ALRABINR after the alarm B event. This ensures that one ck_apre cycle elapses following the alarm A event.

2.13 I2C

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

Description

The I^2C-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^2C-bus SDA line when $t_{SU;DAT}$ is smaller than one I2C kernel clock (I^2C-bus peripheral clock) period: the previous SDA value is sampled instead of the current one. This can result in a wrong receipt of slave address, data byte, or acknowledge bit.

Workaround

Increase the I2C kernel clock frequency to get I2C kernel clock period within the transmitter minimum data setup time. Alternatively, increase transmitter's minimum data setup time. If the transmitter setup time minimum value corresponds to the minimum value provided in the I^2C-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.

2.13.2 Spurious bus error detection in master mode

Description

In master mode, a bus error can be detected spuriously, with the consequence of setting the BERR flag of the I2C_SR register and generating bus error interrupt if such interrupt is enabled. Detection of bus error has no effect on the I^2C-bus transfer in master mode and any such transfer continues normally.

Workaround

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

2.13.3 Spurious master transfer upon own slave address match

Description

When the device is configured to operate at the same time as master and slave (in a multi- master I^2C-bus application), a spurious master transfer may occur under the following condition:

- Another master on the bus is in process of sending the slave address of the device (the bus is busy).
- The device initiates a master transfer by bit set before the slave address match event (the ADDR flag set in the I2C_ISR register) occurs.
- After the ADDR flag is set:
  - the device does not write I2C_CR2 before clearing the ADDR flag, or
  - the device writes I2C_CR2 earlier than three I2C kernel clock cycles before clearing the ADDR flag

In these circumstances, even though the START bit is automatically cleared by the circuitry handling the ADDR flag, the device spuriously proceeds to the master transfer as soon as the bus becomes free. The transfer configuration depends on the content of the I2C_CR2 register when the master transfer starts. Moreover, if the I2C_CR2 is written less than three kernel clocks before the ADDR flag is cleared, the I2C peripheral may fall into an unpredictable state.

Workaround

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

Normal mode (SBC = 0):

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

Slave byte control mode (SBC = 1):

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

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

Polling the BUSY flag before requesting the master transfer is not a reliable workaround as the bus may become busy between the BUSY flag check and the write into the I2C_CR2 register with the START bit set.

2.13.4 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 slave address match event.

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

Workaround

No application workaround is required for this description inaccuracy issue.

2.13.5 OVR flag not set in underrun condition

Description

In slave transmission with clock stretching disabled (NOSTRETCH = 1 in the I2C_CR1 register), an underrun condition occurs if the current byte transmission is completed on the 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.

2.13.6 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 master mode or in slave mode, with clock stretching enabled (NOSTRETCH = 0). This state can only be released by disabling the peripheral (PE = 0) or by resetting it.

Workaround

Apply one of the following measures:

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

2.13.7 SDA held low upon SMBus timeout expiry in slave mode

Description

For the slave 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 slave mode drives SDA low to acknowledge either its address or a data transmitted by the master, 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 master from generating RESTART or STOP condition.

Workaround

When a timeout is reported in slave 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.

2.14 USART

2.14.1 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.

2.14.2 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.

2.14.3 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.

2.14.4 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.

2.15 LPUART

2.15.1 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.

2.15.2 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.

2.15.3 Secure LPUART1 transmission on non-secure PA2 spuriously allowed

Description

Selection of LPUART1_TX as alternate function of PA2 should normally be inhibited when LPUART1 is configured as secure and PA2 as non-secure. Instead, that selection is possible.

Workaround

Keep PA2 configured as secure as long as LPUART1 is configured as secure.

2.16 SPI

2.16.1 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.

2.16.2 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.

2.17 FDCAN

2.17.1 Desynchronization under specific condition with edge filtering enabled

Description

FDCAN may desynchronize and incorrectly receive the first bit of the frame if:

- the edge filtering is enabled (the EFBI bit of the FDCAN_CCCR register is set), and
- the end of the integration phase coincides with a falling edge detected on the FDCAN_Rx input pin

If this occurs, the CRC detects that the first bit of the received frame is incorrect, flags the received frame as faulty and responds with an error frame.

Note: This issue does not affect the reception of standard frames.

Workaround

Disable edge filtering or wait for frame retransmission.

2.17.2 Tx FIFO messages inverted under specific buffer usage and priority setting

Description

Two consecutive messages from the Tx FIFO may be inverted in the transmit sequence if:

- FDCAN uses both a dedicated Tx buffer and a Tx FIFO (the TFQM bit of the FDCAN_TXBC register is cleared), and
- the messages contained in the Tx buffer have a higher internal CAN priority than the messages in the Tx FIFO.

Workaround

Apply one of the following measures:

- Ensure that only one Tx FIFO element is pending for transmission at any time:
  The Tx FIFO elements may be filled at any time with messages to be transmitted, but their transmission requests are handled separately. Each time a Tx FIFO transmission has completed and the Tx FIFO gets empty (TFE bit of FDACN_IR set to 1) the next Tx FIFO element is requested.
- Use only a Tx FIFO:
  Send both messages from a Tx FIFO, including the message with the higher priority. This message has to wait until the preceding messages in the Tx FIFO have been sent.
- Use two dedicated Tx buffers (for example, use Tx buffer 4 and 5 instead of the Tx FIFO). The following pseudo-code replaces the function in charge of filling the Tx FIFO:

```
Write message to Tx Buffer 4
Transmit Loop:
    Request Tx Buffer 4 - write AR4 bit in FDCAN_TXBAR
    Write message to Tx Buffer 5
    Wait until transmission of Tx Buffer 4 complete (IR bit in FDCAN_IR),
    read TO4 bit in FDCAN_TXBTO
    Request Tx Buffer 5 - write AR5 bit of FDCAN_TXBAR
    Write message to Tx Buffer 4
    Wait until transmission of Tx Buffer 5 complete (IR bit in FDCAN_IR),
    read TO5 bit in FDCAN_TXBTO
```

2.18 USB

2.18.1 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 wakeup), mask the SUSP interrupt by setting the SUSPM bit for 3 ms, then unmask it by clearing SUSPM.

2.18.2 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.

2.18.3 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.

2.18.4 USB may not operate correctly in Range 1

Description

With the voltage scaling set to Range 1, the *USB device* peripheral may exhibit timing violations leading to its malfunction.

Workaround

When operating the *USB device* peripheral, always set the voltage scaling to Range 0.

2.19 UCPD

2.19.1 UCPD BMC Tx eye diagram test failure

Description

Duty cycle of the transmitter is outside of specification causing BMC Tx eye diagram tests to fail.

Workaround

None.

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 6. **Document revision history**

Date	Version	Changes
4-Oct-2018	1	Initial release.
8-Apr-2019	2	Added errata descriptors: • Regulator startup failure at low V_{DD} • Voltage scaling range not selectable in SMPS bypass mode • Read of Bank 2 while writing may give unpredictable results • USB, CRS and UCPD may not wake properly from Stop 2 • Low-power run mode not transiting to *"Standby with"* modes • PA15_PUPEN option bit setting inhibits the UCPD dead battery pull-down resistor on PB15 • Spurious setting of PC1 as secure • USB may not operate correctly in Range 1 Modified: • Unstable LSI when it clocks RTC or CSS on LSE • Missing GPIOs on UFBGA132 and WLCSP81 packages • START bit is cleared upon setting ADDRCF, not upon address match moved to the table of documentation errata • errata summary tables, reflecting changes in the errata description section Removed: • bxCAN, FDCAN, and USART sections • *Maxtran period not respected in specific condition* erratum
21-May-2019	3	Added information on silicon revision B.
29-Nov-2019	4	Added errata: • SMPS regulation loss upon transiting into SMPS LP mode • Unpredictable SMPS state at power-on • • DTR octal-SPI indirect read data corrupted if last two bytes are read at a specific condition • Spurious interrupt in AND-match polling mode with full data masking • Hybrid wrap data transfer corruption upon an internal event • Hybrid wrap registers not functional • Odd address alignment and odd byte number not supported at specific conditions • *Read data can be corrupted at precise frequencies* • Memory-mapped write error response when DQS output is disabled • Byte possibly dropped during an SDR read in clock mode 3 when a transfer gets automatically split • Single-, dual- and quad-SPI modes not functional with DQS input enabled • Additional bytes read in Indirect mode with DQS input enabled when data length is too short • 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 • START bit is cleared upon setting ADDRCF, not upon address match • OVR flag not set in underrun condition • Transmission stalled after first byte transfer • Anticipated end-of-transmission signaling in SPI slave mode • Data corruption due to noisy receive line • Desynchronization under specific condition with edge filtering enabled

Date	Version	Changes
		• Tx FIFO messages inverted under specific buffer usage and priority setting Modified errata summary tables, reflecting changes in the errata description section.
28-Apr-2020	5	Added errata: • FLASH_ECCR corrupted upon reset or power-down occurring during flash memory program or erase operation • Wrong ADC differential conversion result for channel 5 Modified errata summary tables, reflecting changes in the errata description section.
04-Aug-2020	6	Added errata Section 2.10.2 Consecutive compare event missed in specific conditions and Section 2.10.3 Output compare clear not working with external counter reset. Replaced LPTIM1 by LPTIM1/3 in Section 2.11.5 LPTIM1/3 outputs cannot be configured as open-drain.
19-Oct-2020	7	Added errata: • ADC_AWDy_OUT reset by non-guarded channels • Data not sampled correctly on reads without DQS and with less than two cycles before the data phase • Data corruption upon a specific FIFO write sequence to synchronous PSRAM • Invalid DAC channel analog output if the DAC channel MODE bitfield is programmed before DAC initialization • DMA underrun flag not set when an internal trigger is detected on the clock cycle of the DMA request acknowledge Removed errata: *Read data can be corrupted at precise frequencies*.
12-Jul-2021	8	Erratum TSC signal-to-noise concern under specific conditions added.
12-Sep-2023	9	Added errata: • **System:** SRAM write error • Corrupted content of the backup domain due to a missed power-on reset after this domain supply voltage drop • LSE crystal oscillator may be disturbed by transitions on PC13 • **GPIO:** GPIO assigned to DAC cannot be used in output mode when the DAC output is connected to on-chip peripheral • **OCTOSPI:** Read data can be corrupted at precise frequencies • Deadlock or write-data corruption after spurious write to a misaligned address in OCTOSPI_ AR register • At least six cycles memory latency must be set when DQS is used for HyperBus™ memories • Read data can be corrupted at precise frequencies (documentation erratum) • Data write discarded in memory-mapped mode if a write to a misaligned address is directly followed by a request to the same address • Setting the ABORT bit does not generate an error on the AHB bus for undefined-length incremental burst transfers • **ADC:** Injected data stored in the wrong ADC_JDRx registers • ADC slave data may be shifted in Dual regular simultaneous mode • **TIM:** Bidirectional break mode not working with short pulses • **LPTIM:** LPTIM events and PWM output are delayed by one kernel clock cycle • **RTC:** A tamper event fails to trigger timestamp or timestamp overflow events during a few cycles after clearing TSF • REFCKON write protection associated to INIT KEY instead of CAL KEY • REFCKON write protection associated to INIT KEY instead of CAL KEY • Tamper flag not set on LSE failure detection • Binary mode: SSR is not reloaded with 0xFFFF FFFF when SSCLR = 1 • **I2C:** SDA held low upon SMBus timeout expiry in slave mode • **USART:** Received data may be corrupted upon clearing the ABREN bit • Noise error flag set while ONEBIT is set

Date	Version	Changes
		• **LPUART:** Possible LPUART transmitter issue when using low BRR[15:0] value • **USB:** ESOF interrupt timing desynchronized after resume signaling • Incorrect CRC16 in the memory buffer • Buffer description table update completes after CTR interrupt triggers Modified errata: • **OCTOSPI:** Indirect read and automatic status-polling transfers without address phase not starting • DTR octal-SPI indirect read data corrupted if last two bytes are read at a specific condition • Odd address alignment and odd byte number not supported at specific conditions • Memory-mapped write error response when DQS output is disabled • Single-, dual- and quad-SPI modes not functional with DQS input enabled • Data not sampled correctly on reads without DQS and with less than two cycles before the data phase • **LPTIM:** Device may remain stuck in LPTIM interrupt when entering Stop mode • **RTC:** RTC configuration changes ignored at specific conditions (removed from device variants B and Z) • **USART:** Data corruption due to noisy receive line (workaround now available) Removed errata: • **OCTOSPI:** *Period defined in MAXTRAN bitfield not respected in specific condition* (not applicable to this product) • **RTC:** *Internal tamper flags not output on RTC_OUT1 and RTC_OUT2* (not applicable to this product) Other: • Introduction in Section 2.1 Core updated with Cortex-M33. • Section Important security notice added.

Contents