STM32L552xx STM32L562xx

Errata sheet

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 VDD	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
FMC	2.3.1	Dummy read cycles inserted when reading synchronous memories	N	N	N
	2.3.2	Wrong data read from a busy NAND memory	A	A	A
OCTOSPI	2.4.1	Indirect read and auto-polling transfers without address phase not starting	A	A	A

Function	Section	Limitation	Status		
			Rev. A	Rev. B	Rev. Z
OCTOSPI	2.4.2	Maxtran period not respected in specific condition	P	P	P
	2.4.3	Octal DDR indirect read data corrupted if last two bytes are read at a specific condition	A	A	A
	2.4.4	Spurious interrupt in AND-match polling mode with full data masking	A	A	A
	2.4.5	Hybrid wrap data transfer corruption upon an internal event	A	A	A
	2.4.6	Hybrid wrap registers not functional	A	A	A
	2.4.7	Odd address alignment and odd byte number not supported at specific conditions	A	A	A
	2.4.8	Read data can be corrupted at precise frequencies	N	N	N
	2.4.9	Memory-mapped write error response when DQS output is disabled	P	P	P
	2.4.10	Byte possibly dropped during an SDR read in clock mode 3 when a transfer gets automatically split	A	A	A
	2.4.11	Single, dual and quad modes not functional with DQS input enabled	N	N	N
	2.4.12	Additional bytes read in indirect mode with DQS input enabled when data length is too short	A	A	A
ADC	2.5.1	New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0	A	A	A
	2.5.2	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
	2.5.3	Unexpected regular conversion when two consecutive injected conversions are performed in Dual interleaved mode	A	A	A
	2.5.4	Wrong ADC result if conversion done late after calibration or previous conversion	A	A	A
	2.5.5	End of ADC conversion disturbing other ADCs	A	-	-
	2.5.6	Wrong ADC differential conversion result for channel 5	A	A	A
COMP	2.6.1	Comparator outputs cannot be configured in open-drain	N	N	N
TIM	2.7.1	One-pulse mode trigger not detected in master-slave reset + trigger configuration	P	P	P
	2.7.4	HSE/32 is not available for TIM16 input capture if RTC clock is disabled or other than HSE	A	A	A
	2.7.2	Consecutive compare event missed in specific conditions	N	N	N
	2.7.3	Output compare clear not working with external counter reset	P	P	P
LPTIM	2.8.1	MCU may remain stuck in LPTIM interrupt when entering Stop mode	A	A	A
	2.8.2	ARRM and CMPM flags are not set when APB clock is slower than kernel clock	P	P	P
	2.8.3	MCU may remain stuck in LPTIM interrupt when clearing event flag	P	P	P
	2.8.4	LPTIM1/3 outputs cannot be configured as open-drain	N	N	N
RTC and TAMP	2.9.1	Internal tamper flags not output on RTC_OUT1 and RTC_OUT2	N	N	N
	2.9.2	Notification of illegal access to secured registers is not reliable	N	N	N
	2.9.3	RTC_MISR and TAMP_MISR can be read by non-privileged accesses when privilege-protected	N	N	N
	2.9.4	RTC configuration changes ignored at specific conditions	A	A	A
	2.9.5	Calibration formula changes when LPCAL is set	A	A	A

Function	Section	Limitation	Status		
			Rev. A	Rev. B	Rev. Z
RTC and TAMP	2.9.6	Calendar initialization may fail in case of consecutive INIT mode entry	A	A	A
	2.9.7	Alarm flag may be repeatedly set when the core is stopped in debug	N	N	N
I2C	2.10.1	Wrong data sampling when data setup time (tSU;DAT) is shorter than one I2C kernel clock period	P	P	P
	2.10.2	Spurious bus error detection in master mode	A	A	A
	2.10.3	Spurious master transfer upon own slave address match	P	P	P
	2.10.5	OVR flag not set in underrun condition	N	N	N
	2.10.6	Transmission stalled after first byte transfer	A	A	A
USART	2.11.1	Anticipated end-of-transmission signaling in SPI slave mode	A	A	A
	2.11.2	Data corruption due to noisy receive line	N	N	N
LPUART	2.12.1	LPUART1 outputs cannot be configured as open-drain	N	N	N
	2.12.2	Secure LPUART1 transmission on non-secure PA2 spuriously allowed	A	-	-
SPI	2.13.1	BSY bit may stay high when SPI is disabled	A	A	A
	2.13.2	BSY bit may stay high at the end of data transfer in slave mode	A	A	A
FDCAN	2.14.1	Desynchronization under specific condition with edge filtering enabled	A	A	A
	2.14.2	Tx FIFO messages inverted under specific buffer usage and priority setting	A	A	A
USB	2.15.1	USB may not operate correctly in Range 1	A	-	-
UCPD	2.16.1	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
I2C	2.10.4	START bit is cleared upon setting ADDRCF, not upon address match

2 Description of device errata

The following sections describe limitations 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. 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.

2.3 FMC

2.3.1 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.

2.3.2 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 OCTOSPI

2.4.1 Indirect read and auto-polling transfers without address phase not starting

Description

Indirect read and auto-polling transfers, configured through the CCR register to contain command and SDR or DDR 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 register.

2.4.2 Maxtran period not respected in specific condition

Description

Under the following condition:

- arbitration activated
- memory-mapped write initiated on one OCTOSPI instance, with a data byte number corresponding to less than two OCTOSPI clock cycles in the data phase
- another OCTOSPI instance requests the I/O port,

the I/O port is not granted even if the Maxtran period expired, unless another mechanism finishes the transaction of the first OCTOSPI instance, such as timeout, new memory request, or the arrival of additional bytes to write.

For example, in octal DDR, the minimum byte number for two clock cycles in the data phase is four. A memory-mapped write of less than four bytes by the OCTOSPI1 instance that holds the I/O port prevents the OCTOSPI2 instance to take it over when requested.

Workaround

Activate the timeout feature to trigger arbitration and select a medium timeout value. A too small value would lead to excessive chip select activity and increase power consumption, and a too big value would lead to excessive arbitration delay and inappropriate system latency.

2.4.3 Octal DDR indirect read data corrupted if last two bytes are read at a specific condition

Description

Indirect read from an octal DDR memory may lead to data corruption upon the following condition:

- 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.

2.4.4 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.

2.4.5 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.

2.4.6 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.

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

Description

Odd address alignment and odd transaction byte number is 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 Octo-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 Octo-SPI, RAM / DDR 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
Octo-SPI, RAM / DDR 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.

2.4.8 Read data can be corrupted at precise frequencies

Description

At certain precise frequencies, the OCTOSPI might overrun its RxFIFO and result in:
- In all modes except for octal-DTR mode: a byte of data corrupted when reading from the memory
- In octal-DTR mode: two bytes of data corrupted when reading from the memory

Workaround

None.
Both indirect and memory-mapped read operations can be affected in all configurations for any type of memory

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

Description

If the DQSE control bit in OCTOSPI_WCCR is set to 0 for memories without DQS pin, it results in an error response for every memory-mapped write request.

Workaround

When doing memory-mapped writes, the DQSE bit in OCTOSPI_WCCR must be set to 1 even for memories which have no DQS pin.

Limitation of this workaround: if the DQS output is asserted on memory-mapped writes while the AXI bus transfer has some byte-enable bits deasserted, the bytes which should be masked get written to the memory.

2.4.10 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, OCTOSPI can interrupt the transfer and automatically restart it at the next address when the CSBOUND, REFRESH, TIMEOUT or MAX-TRAN features are employed. Thus, a single continuous transfer can effectively be split into multiple smaller transfers.

When OCTOSPI is configured to use clock mode 3 (CKMODE bit in OCTOSPI_DCR1 is set to 1) and a continuous stream of data is read in SDR mode (CKMODE bit in OCTOSPI_DCR1 is set to 0), the last byte sent by the memory before an automatic split might get dropped, thus causing all the subsequent bytes to be seen one address earlier.

Workaround

Use clock mode 0 (CKMODE bit in OCTOSPI_DCR1 is set to 0) when in SDR mode.

2.4.11 Single, dual and quad modes not functional with DQS input enabled

Description

Data read from memory in single, dual or quad mode with the DQS input enabled (DQSE control bit in OCTOSPI_CCR is set to 1) can be corrupted. Only octal-data mode (DMODE bit in OCTOSPI_CCR is set to 100) is functional with the DQS input enabled.

Workaround

None.

2.4.12 Additional bytes read in indirect mode with DQS input enabled when data length is too short

Description

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

- Data read in indirect-read mode with DQS enabled (DQSE bit in OCTOSPI_CCR set to 1)
- 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 OCTOSPI_DR register.

2.5 ADC

2.5.1 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.

2.5.2 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.

2.5.3 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.

2.5.4 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.

2.5.5 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.

2.5.6 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.

2.6 COMP

2.6.1 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.

2.7 TIM

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

Description

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

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

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

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

Workaround

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

2.7.2 Consecutive compare event missed in specific conditions

Description

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

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

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

As a consequence:

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

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

Workaround

None.

2.7.3 Output compare clear not working with external counter reset

Description

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

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

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

Workaround

Apply one of the following measures:

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

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

Description

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

Workaround

Apply the following procedure:

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

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

2.8 LPTIM

2.8.1 MCU 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 MCU from Stop mode may be frozen in active state. Consequently, when trying to enter Stop mode, this limitation prevents the MCU from entering low-power mode and the firmware remains stuck in the LPTIM interrupt routine.

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

Workaround

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

2.8.2 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.*

2.8.3 MCU 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 MCU 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 proper clear sequence is already implemented in the HAL_LPTIM_IRQHandler in the STM32Cube.*

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

Description

LPTIM1/3 outputs are set in pushpull mode regardless of the configuration of corresponding GPIO outputs.

Workaround

None.

2.9 RTC and TAMP

2.9.1 Internal tamper flags not output on RTC_OUT1 and RTC_OUT2

Description

RTC_OUT1 and RTC_OUT2 can output the TAMPALRM signal. The TAMPALRM signal should be a logical-OR product of all external and internal tamper flags. Instead, when the TAMPOE control bit of the RTC_CR register is set, the TAMPALRM signal is a logical-OR product of external tamper flags only, ignoring the internal tamper flags.

Workaround

None.

2.9.2 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.

2.9.3 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.

2.9.4 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.

2.9.5 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.

2.9.6 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.

2.9.7 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 though 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 tentative to clear the flag(s) ineffective.

Workaround

None.

2.10 I2C

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

Description

The I^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 MCU 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^{2}C-bus standard, the minimum I2CCLK frequencies are as follows:

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

2.10.2 Spurious bus error detection in 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^{2}C-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.

2.10.3 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^{2}C-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.

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

Description

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

2.10.5 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 I2C bus, and the next data is not yet written in the TXDATA[7:0] bitfield. In this condition, the device is expected to set the OVR flag of the I2C_ISR register and send 0xFF on the bus.

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

Workaround

None.

2.10.6 Transmission stalled after first byte transfer

Description

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

2.11 USART

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

Description

In SPI slave mode, at low USART baud rate with respect to the USART kernel and APB clock frequencies, the *transmission complete* flag TC of the USARTx_ISR register may unduly be set before the last bit is shifted on the transmit line.

This leads to data corruption if, based on this anticipated end-of-transmission signaling, the application disables the peripheral before the last bit is transmitted.

Workaround

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

2.11.2 Data corruption due to noisy receive line

Description

In UART mode with oversampling by 8 or 16 and with 1 or 2 stop bits, 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

None.

2.12 LPUART

2.12.1 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.

2.12.2 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.

2.13 SPI

2.13.1 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.

2.13.2 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.

2.14 FDCAN

2.14.1 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.

2.14.2 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
```

2.15 USB

2.15.1 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.

2.16 UCPD

2.16.1 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.

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 VDD • 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 descriptors:: • SMPS regulation loss upon transiting into SMPS LP mode • Unpredictable SMPS state at power-on • Maxtran period not respected in specific condition • Octal DDR indirect read data corrupted if last two bytes are read at a specific condition • Spurious interrupt in AND-match polling mode with full data maskingHybrid 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 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

Date	Version	Changes
		• 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 • 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 descriptors: • 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 Section 2.7.2 Consecutive compare event missed in specific conditions and Section 2.7.3 Output compare clear not working with external counter reset. Replaced LPTIM1 by LPTIM1/3 in Section 2.8.4 LPTIM1/3 outputs cannot be configured as open-drain.

Contents