STM32U375xx STM32U385xx

Errata sheet

STM32U375xx and STM32U385xx device errata

Applicability

This document applies to the part numbers of STM32U375xx and STM32U385xx 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 RM0487.

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
STM32U375xx	STM32U375VG, STM32U375VE, STM32U375RG, STM32U375RE, STM32U375KG, STM32U375KE, STM32U375CG, STM32U375CE
STM32U385xx	STM32U385VG, STM32U385RG, STM32U385KG, STM32U385CG

Table 2. **Device variants**

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32U375xx and STM32U385xx	Z	0x1001

1. *Refer to the device datasheet for how to identify this code on different types of package.*
2. *REV_ID[5:0] bitfield of DBGMCU_IDCODE fuse.*

1 Summary of device errata

The following table gives a quick reference to the STM32U375xx and STM32U385xx 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. Z
Core	2.1.1	Access permission faults are prioritized over unaligned Device memory faults	N
System	2.2.1	LSE crystal oscillator may be disturbed by transitions on PC13	N
	2.2.2	Device-specific authentication ID is not accessible in RDP Level 0	A
	2.2.3	Incorrect backup domain reset with V_{BAT} and V_{DD} supplied by different power source	A
	2.2.4	FLASH_OEMKEYSR not updated after FLASH_OEMyKEYRx register write	N
	2.2.5	Flash programming can remain stuck in case of programming sequence error	A
	2.2.6	HSI remains enabled after wake-up from Stop mode when debug in Stop mode is enabled	A
	2.2.7	Switching off then on the SRAM1 in Run and Sleep modes may lead to data corruption of other SRAMs	A
	2.2.8	Switching off then on the SRAM2 in Run and Sleep modes may lead to data corruption of other SRAMs	P
	2.2.9	Retained SRAMs may be corrupted after Stop 0 mode or after reset	N
	2.2.10	Retained SRAMs may be corrupted after Stop 1 mode or after reset	N
	2.2.11	SRAM1PD and SRAM2PD bits of PWR_CR1 register reset and access type not as expected	N
	2.2.12	Overconsumption from V_{DD} when V_{DDA} is lower than V_{DD}	N
	2.2.13	LSE in bypass mode may not be functional	N
OCTOSPI	2.3.1	Memory-mapped write error response when DQS output is disabled	P
	2.3.2	Byte possibly dropped during an SDR read in clock mode 3 when a transfer gets automatically split	A
	2.3.3	Deadlock or write-data corruption after spurious write to a misaligned address in OCTOSPI_AR register	N
	2.3.4	Read data corruption after a few bytes are skipped when crossing a four-byte boundary	A
	2.3.5	At least six cycles memory latency must be set when DQS is used for HyperBus™ memories	A
	2.3.6	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
	2.3.7	Setting the ABORT bit does not generate an error on the AHB bus for undefined-length incremental burst transfers	P

Function	Section	Limitation	Status
			Rev. Z
OCTOSPI	2.3.8	Read data corruption when a wrap transaction is followed by a linear read to the same MSB address	N
	2.3.9	Transactions are limited to 8 Mbytes in OctaRAM™ memories	N
	2.3.10	Variable latency is not supported when a refresh collision occurs during a write access to some OctaRAM™ memories	P
	2.3.11	OCTOSPI external memory read issue after exiting Stop 2 or Stop 3 mode when using DQS and delay block	A
SDMMC	2.4.1	Command response and receive data end bits not checked	N
ADC	2.5.1	In combined regular simultaneous plus alternate trigger mode, stopping injected conversion may delay regular conversion	A
	2.5.2	In certain dual ADC modes with regular oversampling continued mode enabled, the JEOC flag is not set at the end of the new injected conversion	A
	2.5.3	In certain dual ADC modes with DMA one-shot mode enabled, JADSTART and ADSTART may not be cleared if JADSTP and ADSTP are set at the same time	A
	2.5.4	Wrong injected conversion data in bulb sampling mode	A
	2.5.5	Shifted master and slave sequence in interleaved discontinuous mode	A
	2.5.6	When the ADC clock is derived from the AHB clock, the injected conversion latency is not respected if the injected trigger coincides with the stopping of the regular conversion	A
	2.5.7	When injected conversions are enabled and the ADC clock is slower than the ADC kernel clock, a wrong context can be applied in certain conditions	A
	2.5.8	In certain dual modes, the fixed trigger latency for the injected conversions may not be respected	A
	2.5.9	In simultaneous regular mode, stopping an injected conversion may shift the next regular conversion master and slave timing by one clock cycle	A
VREFBUF	2.6.1	V_{REFBUF_OUT} voltage overshoots in Range 2 or Stop 1 mode	A
TSC	2.7.1	TSC is not functional on PA15 and PB15	N
LPTIM	2.8.1	Device may remain stuck in LPTIM interrupt when entering Stop mode	A
	2.8.2	ARRM and CMPM flags are not set when APB clock is slower than kernel clock	A
	2.8.3	Interrupt status flag is cleared by hardware upon writing its corresponding bit in LPTIM_DIER register	N
RTC and TAMP	2.9.1	Alarm flag may be repeatedly set when the core is stopped in debug	N
	2.9.2	Parasitic tamper detection when debugger is used in RDP Level 0	A
I2C	2.10.1	Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period	P
	2.10.2	Spurious bus error detection in controller mode	A
USART	2.11.1	Wrong data received by SPI slave receiver in autonomous mode with CPOL = 1	A
	2.11.2	Received data may be corrupted upon clearing the ABREN bit	A
	2.11.3	Noise error flag set while ONEBIT is set	N
LPUART	2.12.1	Possible LPUART transmitter issue when using low BRR[15:0] value	P
SPI	2.13.1	Possible corruption of last-received data depending on CRCSIZE setting	A
	2.13.2	MODF flag cannot generate interrupt	A
	2.13.3	RDY output failure at high serial clock frequency	N
	2.13.4	Master communication suspension fails in Autonomous mode	N
	2.13.5	SPE may not be cleared upon MODF event	A

Function	Section	Limitation	Status
			Rev. Z
SPI	2.13.6	SPI slave stalls with masters not providing extra SCK periods upon *Not ready* signalling	A
	2.13.7	Truncation of SPI output signals after EOT event	A
FDCAN	2.14.1	Desynchronization under specific condition with edge filtering enabled	A
	2.14.2	Tx FIFO messages inverted under specific buffer usage and priority setting	A
USB	2.15.1	Buffer description table update completes after CTR interrupt triggers	A

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 r0p4 is available from http://infocenter.arm.com.

2.1.1 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 LSE crystal oscillator may be disturbed by transitions on PC13

Description

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

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

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

Workaround

None.
Avoid toggling PC13 when LSE is used on LQFP packages.

2.2.2 Device-specific authentication ID is not accessible in RDP Level 0

Description

The AUTH_ID bitfield of the DBGMCU_DBG_AUTH_DEVICE register is not accessible in RDP Level 0. The read value is always 0. Therefore, this bitfield cannot be used to discriminate between different devices.

Workaround

Increase the RDP to Level 1 before reading the device-specific authentication ID. Then, decrease the RDP back to Level 0.

2.2.3 Incorrect backup domain reset with V_{BAT} and V_{DD} supplied by different power source

Description

The backup domain reset may be missed upon backup domain power-on subsequent to a V_{BAT} power-off, in V_{BAT} mode, if the V_{BAT} voltage during the power-off phase drops to a few mV window before starting 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, which may cause a spurious behavior such as driving the LSCO output pin on PA2, raising an unexpected tamper event preventing the SRAM2 or PKA access, or influencing any of the backup domain functions.

Workaround

Apply the following measure to avoid the incorrect backup domain reset:

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

If this workaround is not applicable and the boot follows a backup domain power-on reset:

- Erase the backup domain by software.

In order to discriminate the backup domain power-on reset from a power-on reset or exit from Shutdown mode, at least one backup register (called, for example, BackupTestRegister) must be 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 in calculating the CRC of the backup domain registers: RCC_BDCR and RTC/TAMP registers, excluding bits modified by hardware.

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 if the BORRSTF flag of RCC_CSR is set (the reset is caused by a power-on).
2. If true, check that the BackupTestRegister content is different from BKP_REG_VAL, or the CRC recalculation is different from stored results, accordingly the chosen workaround implementation.
3. If true and if no tamper flag is set (when tamper detection is enabled), the reset is caused by a backup domain power-on. Apply the following sequence:
   a. Enable the PWR clock in RCC, by setting the PWREN bit of RCC_AHB3ENR.
   b. Enable backup domain access, by setting the DBP bit of PWR_DBPR.
   c. Reset the backup domain by:
      i. Writing 0x0001 0000 to RCC_BDCR, which sets the BDRST bit and clears other register bits that might not be reset
      ii. Reading RCC_BDCR, to make the reset time long enough
      iii. Writing 0x0000 0000 to RCC_BDCR, to clear the BDRST bit
   d. Clear BORRSTF, by setting the RMVF bit of RCC_CSR.

2.2.4 FLASH_OEMKEYSR not updated after FLASH_OEMyKEYRx register write

Description

The FLASH_OEMKEYSR register is not updated after any write operation to the FLASH_OEMyKEYRx registers, and thus it is not possible to check if the data were entered correctly by reading the associated CRC. The FLASH_OEMKEYSR is only updated after a new OBL (option byte loading).

Workaround

None.

2.2.5 Flash programming can remain stuck in case of programming sequence error

Description

If an operation is started while the write buffer is waiting for the next data (STRT or OPTSTRT is set while WDW is already set), the PGSERR flag is set and the programming sequence is expected to be aborted. However, the WDW bit remains set and the flash programming operation remains stuck, waiting for the second word to be written.

Workaround

Write the second word to the write buffer to either start the programming (if the programming sequence is correct) or raise a new error, which resets the WDW flag.

2.2.6 HSI remains enabled after wake-up from Stop mode when debug in Stop mode is enabled

Description

HSI clock remains enabled after wake-up from Stop mode where:

- HSI was used as system clock before entering Stop mode
- MSIS is selected as the wake-up clock
- A debugger is connected
- DBG_STOP bit is set in the DBGMCU_CR register

Workaround

If not needed, force HSI OFF by software after wake-up from Stop in above condition.

2.2.7 Switching off then on the SRAM1 in Run and Sleep modes may lead to data corruption of other SRAMs

Description

In Run and Sleep modes, switching on SRAM1 power by resetting SRAM1PD bit of the PWR_CR1 register after being switched off may lead to data corruption in other SRAM (ICACHE, SRAM2, USB RAM, PKA RAM, and FDCAN RAM). Any other SRAM access on a time window of 2 µs after SRAM1PD bit of the PWR_CR1 register may be corrupted.

Workaround

Do not switch off then on SRAM1 in Run and Sleep modes.

2.2.8 Switching off then on the SRAM2 in Run and Sleep modes may lead to data corruption of other SRAMs

Description

In Run and Sleep modes, switching on SRAM2 power by resetting the SRAM2PD bit of the PWR_CR1 register after being switched off may lead to data corruption in other SRAMs (ICACHE, SRAM2, USB RAM, PKA RAM, and FDCAN RAM). Any other SRAM access in a time window of 2 µs after the reset of the SRAM1PD bit of the PWR_CR1 register may be corrupted.

Workaround

Do not switch off then on SRAM2 in Run and Sleep modes.

2.2.9 Retained SRAMs may be corrupted after Stop 0 mode or after reset

Description

Some SRAM bit values (from any SRAM that was retained) may be lost on retained SRAMs when waking up from Stop 0 mode. This occurs when one or more SRAMs pages or instances are powered down during Stop 0 mode, through SRAM power-down bits of the PWR_CR1 or PWR_CR2 registers.

SRAM1 (or respectively SRAM2) content may also be corrupted after system reset if SRAM2PD (or respectively SRAM1PD) is set in the PWR_CR1 register.

Note: Do not power down any SRAM page or instance.

Workaround

None.

2.2.10 Retained SRAMs may be corrupted after Stop 1 mode or after reset

Description

Some SRAM bit values (from any SRAM that was retained) may be lost on retained SRAMs when waking up from Stop 1 mode. This occurs when one or more SRAMs pages or instances are powered down during Stop 1 mode, through SRAM power-down bits of the PWR_CR1 or PWR_CR2 registers.

SRAM1 (or respectively SRAM2) content may also be corrupted after system reset if SRAM2PD (or respectively SRAM1PD) is set in the PWR_CR1 register.

Note: Do not power down any SRAM page or instance.

Workaround

None.

2.2.11 SRAM1PD and SRAM2PD bits of PWR_CR1 register reset and access type not as expected

Description

The reference manual specifies that the SRAM1PD and SRAM2PD bits of the PWR_CR1 register are reset by power on reset, and are set only bits.
Actually, these bits are reset by the system reset, and support read and write access.

Workaround

None.

2.2.12 Overconsumption from V_{DD} when V_{DDA} is lower than V_{DD}

Description

An overconsumption from the V_{DD} supply occurs when the V_{DDA} voltage is lower than V_{DD} voltage in Run and all low-power modes.
The maximum added consumption is expected to be around 100 µA in worst case (V_{DD} = 3.6 V; V_{DDA} = 1.8 V).

Workaround

None.

2.2.13 LSE in bypass mode may not be functional

Description

LSE in bypass mode fails on around 1% of produced parts. The failure rate increases with colder temperature.

Workaround

None.

2.3 OCTOSPI

2.3.1 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.

2.3.2 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.

2.3.3 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.

2.3.4 Read data corruption after a few bytes are skipped when crossing a four-byte boundary

Description

A memory-mapped read is corrupted when the following sequence occurs:

1. An 8- or 16-bit read is performed in a four-byte window, and the last byte of this window is not read.
2. Any read in the next four-byte window is done before the completion of the previous read memory prefetch.

OCTOSPI immediately responds to this read request, even before the data are read from memory, thus resulting in incorrect data read.

If the next read operations are issued to consecutive addresses, then the read data are also corrupted.

Workaround

Apply one of the following measures:

- Perform only 32-bit memory-mapped read accesses.
  For system DMA, use only 32-bit data size for read accesses.
- If this is not possible and an 8-bit or 16-bit read is done at the beginning of a four-byte window, ensure that the next access is not a read from the next four-byte window or that the second access occurs after the data at the skipped addresses are prefetched from memory.

2.3.5 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.

2.3.6 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.

2.3.7 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.

2.3.8 Read data corruption when a wrap transaction is followed by a linear read to the same MSB address

Description

If a wrap transaction is followed by a linear read having the same MSB start address as the wrap (), then the linear read is wrongly considered as a sequential transaction to the previous one, taking back the prefetched data and causing data corruption.

Notice that for a wrap transaction, the prefetch starts after the last address of the wrap window.

Workaround

As prefetch cannot be disabled, there is no workaround. However, the issue is seldom encountered since wrap operations are mostly initiated by the internal cache to refresh its cacheline. All the other masters must avoid retrieving data by using a linear read access to the same MSB address as the wrap, which has been just completed.

2.3.9 Transactions are limited to 8 Mbytes in OctaRAM™ memories

Description

When the controller is configured in Macronix OctaRAM™ mode, by setting the MTYP[2:0] bitfield of the OCTOSPI_DCR1 register to 011, only 13 bits of row address are decoded and sent to the memory, meaning that only 8 K of 1-Kbyte blocks can be accessed (8 Mbytes).

Workaround

None.

This limitation is not present for PSRAMs or HyperRAM™ memories.

2.3.10 Variable latency is not supported when a refresh collision occurs during a write access to some OctaRAM™ memories

Description

When the memory type (MTYP[2:0] bitfield of the OCTOSPI_CR register) is configured to 0b011 to target an OctaRAM™ memory, the host controller does not support the variable latency requested by the external memory if a refresh collision occurs during the write access. For example, some OctaRAM™ memories, such as ISSI memories, request extra latency cycles for write accesses during refresh collision. In this case, the controller does not sample the DQS input signal during the instruction phase, and cannot detect the extra latency requested by the external memory for the refresh operation.This results in data corruption.

Some OctaRAM™ memories do not request any additional latency for write access during refresh cycles. It is required only when the refresh occurs during a read access. In this case, no issue can be observed.

Workaround

When the application targets an OctaRAM™ memory that requests extra latency cycles for write access during refresh collision, force the fixed latency mode in the configuration register of the external memory. There is no constraint about read access, since both variable and fixed latency modes are supported.

2.3.11 OCTOSPI external memory read issue after exiting Stop 2 or Stop 3 mode when using DQS and delay block

Description

After exiting Stop 2 or Stop 3 mode, the first read in the OCTOSPI external memory can return erroneous data when the following condition is met:

- OCTOSPI DQS is enabled
- Delay block is enabled

Workaround

After exiting Stop 2 or Stop 3 mode, make a dummy read from the external memory before reading the required data.

2.4 SDMMC

2.4.1 Command response and receive data end bits not checked

Description

The command response and receive data end bits are not checked by the SDMMC. A reception with only a wrong end bit value is not detected. This does not cause a communication failure since the received command response or data is correct.

Workaround

None.

2.5 ADC

2.5.1 In combined regular simultaneous plus alternate trigger mode, stopping injected conversion may delay regular conversion

Description

In dual ADC combined regular simultaneous plus alternate trigger mode, the resumption of an active regular conversion may be delayed by a few ADC clock cycles when an injected trigger event coincides with stopping, by application, an ongoing injected conversion.

Note: *The dual ADC combined regular simultaneous plus alternate trigger mode is selected by setting the DUAL[4:0] bitfield of the ADCC_CCR register to 0x02. To stop an ongoing injected conversion, the software sets the JADSTP bit of the ADC_CR register.*

Workaround

- Design the application so as to avoid injected trigger events from coinciding with stopping, by software or DMA, an ongoing regular conversion.
- Stop the regular conversion before stopping the injected conversion.

2.5.2 In certain dual ADC modes with regular oversampling continued mode enabled, the JEOC flag is not set at the end of the new injected conversion

Description

In dual ADC regular simultaneous or interleaved mode, when the regular oversampling is enabled in continued mode, and injected conversions are ongoing on both ADCs, if an injected conversion is stopped and a new conversion is started for the same ADC, the JEOC flag is not set at the end of the new injected conversion.

Note: *The regular simultaneous and interleaved modes are selected by setting the DUAL[4:0] bitfield of the ADCC_CCR register to 0x06 and 0x07, respectively. The regular oversampling continued mode is enabled by setting the ROVSE bit and clearing the ROVSM bit of the ADC_CFGR2 register. To start a new injected conversion and stop the ongoing one, the software must set the JADSTART and JADSTP bits of the ADC_CR register, respectively. When set, the JEOC flag of the ADC_ISR register indicates the end of the injected channel conversion.*

Workaround

Apply one of the following measures:

- Use one of the following modes instead of the regular simultaneous or the interleaved mode:
  - Combined regular simultaneous + injected simultaneous mode (DUAL[4:0] = 0x01)
  - Combined regular simultaneous + alternate trigger mode (DUAL[4:0] = 0x02)
  - Combined interleaved mode + injected simultaneous mode (DUAL[4:0] = 0x03)
- If an injected conversion is already ongoing on one ADC, avoid triggering a new injected conversion on the other ADC after stopping the ongoing conversion.
- Instead of the regular oversampling continued mode, use the resumed mode (ROVSM = 1).

2.5.3 In certain dual ADC modes with DMA one-shot mode enabled, JADSTART and ADSTART may not be cleared if JADSTP and ADSTP are set at the same time

Description

In certain dual ADC modes (DUAL[4:0] = 0x05, 0x06, 0x07, and 0x09 in the ADCC_CCR register), ADSTART and JADSTART bits may not be cleared, if the DMA is configured in one-shot mode, and the stopping of the ongoing injected conversion by software coincides with the stopping of the ongoing regular conversion by DMA.

Note: *The DMA one-shot mode is enabled by setting the DMNGT[1:0] bitfield of the ADC_CFGR2 register to 0b01. To stop an injected conversion, the software must set the JADSTP bit of the ADC_CR register.*

Workaround

Apply one of the following measures:

- Use one of the following modes instead of the injected simultaneous, regular simultaneous or the interleaved mode:
  - Combined regular simultaneous + injected simultaneous mode (DUAL[4:0] = 0x01)
  - Combined regular simultaneous + alternate trigger mode (DUAL[4:0] = 0x02)
  - Combined interleaved mode + injected simultaneous mode (DUAL[4:0] = 0x03)
- Avoid using DMA one-shot mode to manage regular conversions. Instead, use interrupt or polling.

2.5.4 Wrong injected conversion data in bulb sampling mode

Description

In bulb sampling mode, if regular conversions are enabled but not running, and an injected conversion is requested just after the ADSTP bit of the ADC_CR register is set, the injected conversion data obtained on the selected channel may be wrong.

Note: *The bulb sampling mode is selected by setting the BULB bit of the ADC_CR register.*

Workaround

Avoid triggering injected conversions when the ADSTP bit is set.

2.5.5 Shifted master and slave sequence in interleaved discontinuous mode

Description

In dual ADC interleaved mode, when the discontinuous mode is enabled on regular channels, and the next trigger arrives before the end of the sequence, the master and slave interleave sequence may be shifted.

Note: The dual ADC interleaved modes are selected by setting the DUAL[4:0] bitfield of the ADCC_CCR register to 0x03 or 0x07. The discontinuous mode is enabled through the DISCEN bit of the ADC_CFGR1 register.

Workaround

Avoid triggering a new conversion when the ongoing conversion is not complete.

2.5.6 When the ADC clock is derived from the AHB clock, the injected conversion latency is not respected if the injected trigger coincides with the stopping of the regular conversion

Description

When the ADC clock is derived from the AHB clock, and the analog-to-digital conversion is triggered by a timer, the latency between the trigger and the start of the ADC sampling is fixed. When both injected and regular conversions are enabled, if the injected trigger coincides with the stopping of regular conversion, the latency of injected conversions becomes one clock cycle shorter than the expected latency.

Note: To stop an ongoing regular conversion, the software sets the ADSTP bit of the ADC_CR register.

Workaround

Apply one of the following measures:

- Avoid triggering injected conversions when the ADSTP bit is set.
- When an injected trigger is expected, keep the ADSTP bit cleared.

2.5.7 When injected conversions are enabled and the ADC clock is slower than the ADC kernel clock, a wrong context can be applied in certain conditions

Description

When the injected queue (JSQR) is enabled (by clearing the JQM bit of the ADC_CFGR1 register), if the JSQR queue becomes empty, the old context is maintained when a new trigger occurs, and the new sequence starts with the old context. When this sequence completes, the JEOS flag is set in the ADC_ISR register.

The JSQR queue can be updated by software, and started by a hardware trigger at any time. If these two actions occur simultaneously with the setting of JEOS, and the adc_hclk clock is slower than the ADC kernel clock, the JSQR queue is not updated at the right moment, thus resulting in an unpredictable sequence execution:

- The previous sequence is executed instead of the programmed sequence, or
- The programmed sequence is executed but it is triggered by the previous external event, or
- The injected conversion sequence has started with the previous context, but during the sequence execution, the old context is overridden by the new one.

Workaround

When the ADC clock is slower than the ADC kernel clock and the injected queue is empty, avoid a hardware trigger and a JSQR update from occurring simultaneously with the setting of JEOS. This can be done by disabling the injected trigger when a JSQR update is performed.

2.5.8 In certain dual modes, the fixed trigger latency for the injected conversions may not be respected

Description

If the application operates in regular simultaneous or interleaved mode (DUAL[4:0] = 0x06 or 0x07 in the ADCC_CCR register), the injected conversions can be activated either on the master or on the slave ADC. When a fixed trigger latency is configured for injected conversions, the regular conversions on both ADCs are interrupted by the injected trigger. In this case, the previous conversion trigger latency and the current conversion trigger latency may differ by one clock cycle.

Workaround

If the regular simultaneous mode or the interleaved mode is used (DUAL[4:0]=0x06 or 0x07), and the application needs injected conversions with a fixed trigger latency for both ADCs, apply one of the following measures:

- Select another dual mode, either DUAL[4:0] = 0x01 or 0x03.
- Make sure that no injected trigger event occurs when regular conversions are stopped, by setting the ADSTP bit in the ADC_CR register.

2.5.9 In simultaneous regular mode, stopping an injected conversion may shift the next regular conversion master and slave timing by one clock cycle

Description

In simultaneous regular mode (DUAL[4:0] = 0x06 in the ADCC_CCR register), injected conversions can be activated either on ADC1 or ADC2. When regular conversions are ongoing, if an injected conversion stop (JADSTP) occurs at the same time as an injected trigger event, then ADC1 and ADC2 timing may be shifted by one clock cycle.

Workaround

Apply one of the following measures:

- Avoid the following events from occurring simultaneously:
  - Triggering injected conversions
  - Stopping injected conversions
  - Enabling regular conversion in simultaneous regular mode (DUAL[4:0] = 0x06)
- Stop regular conversions before stopping injected conversions.
- Stop injected trigger source before stopping injected conversions.
- Configure DUAL[4:0] to 0x01 or 0x02.

2.6 VREFBUF

2.6.1 V_{REFBUF_OUT} voltage overshoots in Range 2 or Stop 1 mode

Description

The V_{REFBUF_OUT} output voltage overshoots when started:

- while the regulator is in Range 2 or
- while the device is in Stop 1 mode.

Workaround

Modify the regulator voltage range to Range 1 before enabling the voltage reference buffer, then go back to Range 2 or Stop 1 mode.

2.7 TSC

2.7.1 TSC is not functional on PA15 and PB15

Description

PA15/PB15 can be used as an alternate function as sensor, as a shield, or even as a sampling capacitor for an active shield TSC group However, it cannot be used as a sampling capacitor in a TSC group with sensors.

Workaround

None.

2.8 LPTIM

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

Description

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

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

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

Workaround

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

2.8.2 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 Interrupt status flag is cleared by hardware upon writing its corresponding bit in LPTIM_DIER register

Description

When any interrupt bit of the LPTIM_DIER register is modified, the corresponding flag of the LPTIM_ISR register is cleared by hardware.

Workaround

None.

2.9 RTC and TAMP

2.9.1 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.

2.9.2 Parasitic tamper detection when debugger is used in RDP Level 0

Description

The internal tamper 6 flag (ITAMP6F) can be unexpectedly set in the TAMP status register (TAMP_SR) when a debugger is connected in RDP Level 0, in case a switch to V_{BAT} occurs (V_{DD} is below the BOR0 threshold).

Workaround

Keep internal tamper 6 flag disabled as long as debug is needed, and enable it once development phase is complete. The tamper flag cannot be set if no debug access is done.

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 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 target address, data byte, or acknowledge bit.

Workaround

Increase the I2C kernel clock frequency to get I2C kernel clock period within the transmitter minimum data setup time. Alternatively, increase transmitter's minimum data setup time. If the transmitter setup time minimum value corresponds to the minimum value provided in the I^{2}C-bus standard, the minimum I2CCLK frequencies are as follows:

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

2.10.2 Spurious bus error detection in controller mode

Description

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

Workaround

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

2.11 USART

2.11.1 Wrong data received by SPI slave receiver in autonomous mode with CPOL = 1

Description

The SPI slave receiver device receives wrong data when all the following conditions are met:

- The USART is used in SPI master transmitter mode
- The autonomous mode is used
- The CPOL bit of the USART_CR2 register is set

Workaround

When the autonomous mode is used, do not set the CPOL bit in USART_CR2.

2.11.2 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.

2.11.3 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.

2.12 LPUART

2.12.1 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.

2.13 SPI

2.13.1 Possible corruption of last-received data depending on CRCSIZE setting

Description

With the CRC calculation disabled (CRCEN = 0), the transfer size bitfield set to a value greater than zero (TSIZE[15:0] > 0), and the length of CRC frame set to less than 8 bits (CRCSIZE[4:0] < 00111), the last data received in the RxFIFO may be corrupted.

Workaround

Keep the CRCSIZE[4:0] bitfield at its default setting (00111) during the data reception if CRCEN = 0 and TSIZE[15:0] > 0.

2.13.2 MODF flag cannot generate interrupt

Description

Mode fault detection results in disabling SPI. With the MODFIE bit of the SPI_IER register set, the mode fault flag (MODF) going high is expected to trigger an interrupt. However, disabling SPI unduly blocks this interrupt request.

Workaround

To detect a mode fault event, poll the MODF flag by software.

2.13.3 RDY output failure at high serial clock frequency

Description

When acting as slave with RDY alternate function enabled through setting the RDIOM bit of the SPI_CFG2 register, the device may fail to indicate its *Not ready* status in time through the RDY output signal to suspend communication. This may then lead to data overrun and/or underrun on the device side. The failure occurs when the serial clock frequency exceeds:

- twice the APB clock frequency, with data sizes from 8 to 15 bits
- six times the APB clock frequency, with data sizes from 16 to 23 bits
- fourteen times the APB clock frequency, with data sizes from 24 to 32 bits

Workaround

None.

2.13.4 Master communication suspension fails in Autonomous mode

Description

The SPI peripheral is blocked regardless of the completion of the ongoing data frame transaction, and the SUPSF flag is never set, when:

- the master provides a communication triggered in Autonomous mode (TRIGEN=1 of the SPI_AUTOCR register), and
- the suspension of the ongoing transaction is applied by setting the CSUSP bit through the smart DMA in Stop mode.

Workaround

None.

Note: *The user software must avoid any master suspension in Stop mode while the master operates in Autonomous mode and waits for EOT if TSIZE is greater than 0. If an endless transaction is applied (TSIZE = 0), the suspension is the only way to stop the ongoing transaction. Then to unblock the peripheral, the software must disable SPI then apply the hardware reset. Otherwise, the system cannot proceed to the next transaction.*

2.13.5 SPE may not be cleared upon MODF event

Description

The failure described applies to multi-master topology when the device is configured to monitor the SS input signal by hardware (SSM = 0, SSOE = 0 of the SPI_CFG2 register).

If the software sets the SPE (SPI enable) bit of the SPI_CR1 register at the instant of the SS signal transiting to its active logical level, the resulting MODF event duly switches the SPI into slave mode, but it fails to clear the SPE bit and thus disable the SPI.

Note: *The SS active logical level is the one that matches the SSIOP bit of the SPI_CFG2 register.*

Workaround

Whenever MODF event fails to clear the SPE bit, do it by software.

2.13.6 SPI slave stalls with masters not providing extra SCK periods upon *Not ready* signalling

Description

In Stop mode, the device SPI operating as slave with the *Ready* signalling enabled (the RDIOM of the SPI_CFG2 register set) may stall and never retrieve the *Ready* state. This occurs when SCK stops immediately after *Not ready* status.

Note: *STM32 devices supporting the Ready signaling and operating as SPI master provide some extra SCK periods upon detecting Not ready signal, thus allowing the SPI slaves to operate correctly.*

Workaround

If in the application, there is an SPI master that stops SCK immediately upon *Not ready* signal, without providing some extra SCK periods, do not enable the *Ready* signalling.

2.13.7 Truncation of SPI output signals after EOT event

Description

After an EOT event signaling the end of a non-zero transfer size transaction (TSIZE > 0) upon sampling the last data bit, the software may disable the SPI peripheral. As expected, disabling SPI deactivates the SPI outputs (SCK, MOSI and SS when the SPI operates as a master, MISO when as a slave), by making them float or statically output their by-default levels, according to the AFCNTR bit of the SPI_CFG2 register.

With fast software execution (high PCLK frequency) and slow SPI (low SCK frequency), the SPI disable occurring too fast may result in truncating the SPI output signals. For example, the device operating as a master then generates an asymmetric last SCK pulse (with CPHA = 0), which may prevent the correct last data bit reception by the other node involved in the communication.

Workaround

Apply one of the following measures or their combination:

- Add a delay between the EOT event and SPI disable action.
- Decrease the ratio between PCLK and SCK frequencies.

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 Buffer description table update completes after CTR interrupt triggers

Description

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

Workaround

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

Important security notice

Revision history

Table 4. Document revision history

Date	Version	Changes
13-Feb-2025	1	Initial release

Contents