STM32L4Rxxx and STM32L4Sxxx device errata

Applicability

This document applies to the part numbers of STM32L4Rxxx and STM32L4Sxxx 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 RM0432.

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
STM32L4Rxxx	STM32L4R5AG, STM32L4R5AI, STM32L4R5QG, STM32L4R5QI, STM32L4R5VG, STM32L4R5VI, STM32L4R5ZG, STM32L4R5ZI, STM32L4R7AI, STM32L4R7VI, STM32L4R7ZI, STM32L4R9AG, STM32L4R9AI, STM32L4R9VG, STM32L4R9VI, STM32L4R9ZG, STM32L4R9ZI
STM32L4Sxxx	STM32L4S5AI, STM32L4S5QI, STM32L4S5VI, STM32L4S5ZI, STM32L4S7AI, STM32L4S7VI, STM32L4S7ZI, STM32L4S9AI, STM32L4S9VI, STM32L4S9ZI

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32L4Rxxx STM32L4Sxxx	Y	0x1003
	W	0x100F
	V	0x101F

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 STM32L4Rxxx and STM32L4Sxxx device limitations and their status:

A = limitation present, workaround available

N = limitation present, no workaround available

P = limitation present, partial workaround available

"-" = limitation absent

Applicability of a workaround may depend on specific conditions of target application. Adoption of a workaround may cause restrictions to target application. Workaround for a limitation is deemed partial if it only reduces the rate of occurrence and/or consequences of the limitation, or if it is fully effective for only a subset of instances on the device or in only a subset of operating modes, of the function concerned.

Table 3. Summary of device limitations

Function	Section	Limitation	Status		
			Rev. Y	Rev. W	Rev. V
Core	2.1.1	Interrupted loads to SP can cause erroneous behavior	A	A	A
System	2.2.1	Full JTAG configuration without NJTRST pin cannot be used	A	A	A
	2.2.2	Data cache might be corrupted during Flash memory read-while-write operation	A	A	A
	2.2.3	Flash memory might not be accessible when AHB prescaler is greater than two	N	N	N
	2.2.4	Flash OPTVERR flag is always set after system reset	A	A	A
	2.2.5	HSE oscillator long startup at low voltage	P	P	P
	2.2.6	SDMMC1SMEN bit of RCC_AHB2SMENR register only modifiable with word access	A	A	A
	2.2.7	Unstable LSI when it clocks RTC or CSS on LSE	P	P	P
	2.2.8	LTDC and DSI not functional with Stop 2	P	-	-
	2.2.9	Regulator startup failure at low VDD	N	-	-
	2.2.10	1-Mbyte devices wrongly configured as 2-Mbyte	N	-	-
	2.2.11	OBL_LAUNCH operation may corrupt SRAM content	P	P	P
	2.2.13	Bootloader not functional on 1-Mbyte devices	N	-	-
	2.2.14	FLASH_ECCR corrupted upon reset or power-down occurring during Flash memory program or erase operation	A	A	A
	2.2.16	Wrong instruction fetches from flash memory upon wakeup from Sleep or Stop mode when debug in low-power mode is enabled	A	A	A
GPIO	2.3.1	Some I/O functions on PC13 pin do not work with RTC_OUT mapped on PB2	A	A	A
FW	2.4.1	Firewall protection 127,93 Kbytes size limitation	N	N	N
	2.4.2	Spurious Firewall reset	P	P	P
DMA	2.5.1	DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear	A	A	A
DMAMUX	2.6.1	SOFx not asserted when writing into DMAMUX_CFR register	N	N	N
	2.6.2	OFx not asserted for trigger event coinciding with last DMAMUX request	N	N	N
	2.6.3	OFx not asserted when writing into DMAMUX_RGCFR register	N	N	N

Function	Section	Limitation	Status		
			Rev. Y	Rev. W	Rev. V
DMAMUX	2.6.4	Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event	A	A	A
FMC	2.7.1	Dummy read cycles inserted when reading synchronous memories	N	N	N
	2.7.2	Wrong data read from a busy NAND memory	A	A	A
	2.7.3	Data corruption upon a specific FIFO write sequence to synchronous PSRAM	A	A	-
	2.7.4	Unsupported AHB burst byte write to PSRAM	-	-	A
OCTOSPI	2.8.1	CSBOUND must not be used	N	N	N
	2.8.2	Automatic status-polling mode not functional with octal memory devices	A	A	A
	2.8.3	Command phase must be octal for octal transfers	A	A	A
	2.8.4	Write data lost with clock mode 3	A	A	A
	2.8.5	Deadlock upon disabling OCTOSPI during memory-mapped write	A	A	A
	2.8.6	Deadlock in dual-quad configuration with Flash memories and odd number of bytes	A	A	A
	2.8.7	No transfer error interrupt upon indirect write to address exceeding the limit	N	N	N
	2.8.8	DHQC not effective if DDTR not set	A	A	A
	2.8.9	Indirect read and automatic status-polling transfers without address phase not starting	A	A	A
	2.8.10	Unaligned AHB write requests not supported	N	N	N
	2.8.11	Write operation in DTR octal-SPI mode starting at odd address is not supported	N	N	N
	2.8.12	Data masking for odd number of byte writes only working with D1/D0 ordering	A	A	A
	2.8.13	Memory-mapped read of the last memory space byte not possible in SDR octal-SPI mode	A	A	A
	2.8.14	Single-byte memory-mapped write to odd DTR octal-SPI address failing when a higher-priority event occurs	A	A	A
	2.8.15	Memory-mapped write data to odd start address corrupted when BUSY = 0 and not in DTR octal-SPI mode	A	A	A
	2.8.16	Data not sampled correctly on reads without DQS and with less than two cycles before the data phase	A	A	A
	2.8.17	Memory-mapped write error response when DQS output is disabled	P	P	P
	2.8.18	Byte possibly dropped during an SDR read in clock mode 3 when a transfer gets automatically split	A	A	A
	2.8.19	Single-, dual- and quad-SPI modes not functional with DQS input enabled	N	N	N
	2.8.20	Additional bytes read in Indirect mode with DQS input enabled when data length is too short	A	A	A
	2.8.21	DQS output enabled too late on write	A	A	A
	2.8.23	Deadlock on memory-mapped write with timeout enabled	A	A	A
OCTOSPIM	2.9.1	Unaligned write access to OCTOSPIM configuration registers failing	A	A	A
ADC	2.10.3	New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0	A	A	A

Function	Section	Limitation	Status		
			Rev. Y	Rev. W	Rev. V
ADC	2.10.4	Two consecutive context conversions fail when writing new context in ADC_JSQR just after previous context completion with JQDIS = 0 and JQM = 0	A	A	A
	2.10.5	ADC_AWDy_OUT reset by non-guarded channels	A	A	A
	2.10.6	Wrong ADC result if conversion done late after calibration or previous conversion	A	A	A
	2.10.7	Spurious temperature measurement due to spike noise	A	-	-
	2.10.8	Wrong ADC differential conversion result for channel 5	A	A	A
	2.10.9	Selected external ADC inputs unduly clamped to VDD when all analog peripherals are disabled	A	A	A
DAC	2.11.1	Invalid DAC channel analog output if the DAC channel MODE bitfield is programmed before DAC initialization	A	A	A
	2.11.2	DMA underrun flag not set when an internal trigger is detected on the clock cycle of the DMA request acknowledge	N	N	N
COMP	2.12.1	Comparator outputs cannot be configured in open-drain	N	N	N
DSI	2.13.1	Tearing effect parasitic detection	P	P	P
	2.13.2	Incorrect calculation of the time to activate the clock between HS transmissions	P	P	P
	2.13.3	The immediate update procedure may fail	A	A	A
	2.13.4	Failing DSI read operation	A	A	A
	2.13.5	DSI-PHY may not start properly when the DSI PLL is restarted	A	A	A
TSC	2.14.2	TSC signal-to-noise concern under specific conditions	A	A	A
TIM	2.16.1	One-pulse mode trigger not detected in master-slave reset + trigger configuration	P	P	P
	2.16.2	Consecutive compare event missed in specific conditions	N	N	N
	2.16.3	Output compare clear not working with external counter reset	P	P	P
	2.16.4	HSE/32 is not available for TIM16 input capture if RTC clock is disabled or other than HSE	A	A	A
LPTIM	2.17.1	Device may remain stuck in LPTIM interrupt when entering Stop mode	A	A	A
	2.17.2	Device may remain stuck in LPTIM interrupt when clearing event flag	P	P	P
	2.17.3	LPTIM events and PWM output are delayed by 1 kernel clock cycle	P	P	P
	2.17.4	LPTIM1 outputs cannot be configured as open-drain	N	N	N
RTC and TAMP	2.18.1	RTC interrupt can be masked by another RTC interrupt	A	A	A
	2.18.2	Calendar initialization may fail in case of consecutive INIT mode entry	A	A	A
	2.18.3	Alarm flag may be repeatedly set when the core is stopped in debug	N	N	N
	2.18.4	RTC_REFIN and RTC_OUT on PB2 not operating in Stop 2 mode	P	P	P
I2C	2.19.1	10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave	A	A	A
	2.19.3	Wrong data sampling when data setup time (tSU;DAT) is shorter than one I2C kernel clock period	P	P	P
	2.19.4	Spurious bus error detection in master mode	A	A	A
	2.19.5	Last-received byte loss in reload mode	P	P	P
	2.19.6	Spurious master transfer upon own slave address match	P	P	P

Function	Section	Limitation	Status		
			Rev. Y	Rev. W	Rev. V
I2C	2.19.8	OVR flag not set in underrun condition	N	N	N
	2.19.9	Transmission stalled after first byte transfer	A	A	A
USART	2.20.1	RTS is active while RE = 0 or UE = 0	A	A	A
	2.20.3	UDR flag set while the SPI slave transmitter is disabled	A	A	A
	2.20.4	Anticipated end-of-transmission signaling in SPI slave mode	A	A	A
	2.20.5	Data corruption due to noisy receive line	N	N	N
LPUART	2.21.1	Data corruption due to noisy receive line	N	N	N
	2.21.2	LPUART1 outputs cannot be configured as open-drain	N	N	N
SPI	2.22.1	BSY bit may stay high when SPI is disabled	A	A	A
	2.22.2	BSY bit may stay high at the end of data transfer in slave mode	A	A	A
SAI	2.23.1	Last SAI_SCK clock pulse truncated upon disabling SAI master with NODIV set and MCKDIV greater than one	P	P	P
	2.23.2	Last SAI_MCLK clock pulse truncated upon disabling SAI master	A	A	A
bxCAN	2.24.1	bxCAN time-triggered communication mode not supported	N	N	N
OTG_FS	2.25.1	Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_FS registers	A	A	A

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

Table 4. Summary of device documentation errata

Function	Section	Documentation erratum
System	2.2.12	Irrelevant BOOT0 electrical characteristics
	2.2.15	Flash memory ECC single-error correction prevents double-error detection from triggering NMI
DMAMUX	2.6.5	DMAMUX_RGCFR register is write-only, not read-write
	2.6.6	DMA request counter not kept at GNBREQ bitfield value as long as the corresponding request channel is disabled
	2.6.7	Synchronization event discarded if selected input DMA request is not active
OCTOSPI	2.8.22	PSRAM not supported
ADC	2.10.1	Writing ADC_JSQR when JADCSTART and JQDIS are set may lead to incorrect behavior
	2.10.2	ADEN bit cannot be set immediately after the ADC calibration is done
TSC	2.14.1	Inhibited acquisition in short transfer phase configuration
HASH	2.15.1	Superseded suspend sequence for data loaded by DMA
	2.15.2	Superseded suspend sequence for data loaded by the CPU
I2C	2.19.2	Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C
	2.19.7	START bit is cleared upon setting ADDRCF, not upon address match
USART	2.20.2	Receiver timeout counter wrong start in two-stop-bit configuration
SPI	2.22.3	CRC error in SPI slave mode if internal NSS changes before CRC transfer

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®-M4F core revision r0p1 is available from http://infocenter.arm.com.

2.1.1 Interrupted loads to SP can cause erroneous behavior

This limitation is registered under Arm ID number 752770 and classified into “Category B”. Its impact to the device is minor.

Description

If an interrupt occurs during the data-phase of a single word load to the stack-pointer (SP/R13), erroneous behavior can occur. In all cases, returning from the interrupt will result in the load instruction being executed an additional time. For all instructions performing an update to the base register, the base register will be erroneously updated on each execution, resulting in the stack-pointer being loaded from an incorrect memory location.

The affected instructions that can result in the load transaction being repeated are:

- LDR SP, [Rn],#imm
- LDR SP, [Rn,#imm]!
- LDR SP, [Rn,#imm]
- LDR SP, [Rn]
- LDR SP, [Rn,Rm]

The affected instructions that can result in the stack-pointer being loaded from an incorrect memory address are:

- LDR SP,[Rn],#imm
- LDR SP,[Rn,#imm]!

As compilers do not generate these particular instructions, the limitation is only likely to occur with hand-written assembly code.

Workaround

Both issues may be worked around by replacing the direct load to the stack-pointer, with an intermediate load to a general-purpose register followed by a move to the stack-pointer.

2.2 System

2.2.1 Full JTAG configuration without NJTRST pin cannot be used

Description

When using the JTAG debug port in Debug mode, the connection with the debugger is lost if the NJTRST pin (PB4) is used as a GPIO or for an alternate function other than NJTRST. Only the 4-wire JTAG port configuration is impacted.

Workaround

Use the SWD debug port instead of the full 4-wire JTAG port.

2.2.2 Data cache might be corrupted during Flash memory read-while-write operation

Description

When a write to the internal Flash memory is done, the data cache is normally updated to reflect the data value update. During this data cache update, a read to the other Flash memory bank may occur; this read can corrupt the data cache content and subsequent read operations at the same address (cache hits) will be corrupted.

This limitation only occurs in dual bank mode, when reading (data access or code execution) from one bank while writing to the other bank with data cache enabled.

Workaround

When the application is performing data accesses in both Flash memory banks, the data cache must be disabled by resetting the DCEN bit before any write to the Flash memory. Before enabling the data cache again, it must be reset by setting and then resetting the DCRST bit.

Code example:

```
/* Disable data cache  */
__HAL_FLASH_DATA_CACHE_DISABLE();

/* Set PG bit */
SET_BIT(FLASH->CR, FLASH_CR_PG);

/* Program the Flash word */
WriteFlash(Address, Data);

/* Reset data cache */
__HAL_FLASH_DATA_CACHE_RESET();

/* Enable data cache */
__HAL_FLASH_DATA_CACHE_ENABLE();
```

2.2.3 Flash memory might not be accessible when AHB prescaler is greater than two

Description

When the AHB prescaler (the HPRE[3:0] bitfield of the RCC_CFGR register) is set to a value greater than eight, the system clock (HCLK) is divided by four or more. When a system reset occurs during a Flash memory access, the Flash memory might become inaccessible. When this issue occurs, a hard fault exception is generated when the Flash memory is accessed. The application running from Flash memory can restart only after a power-on reset.

Workaround

None. Do not use a system clock divider value greater than two.

2.2.4 Flash OPTVERR flag is always set after system reset

Description

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

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

Workaround

After reset, clear the OPTVERR flag in FLASH_SR register.

2.2.5 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 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.

2.2.7 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} (internally in the package for products not featuring the VBAT pin, or externally) and a short (< 1 ms) V_{DD} drop under V_{DD}(min) occurs

Workaround

Apply one of the following measures:

- Clock the RTC with LSE or HSE/32, without using the CSS on LSE
- If LSI clocks the RTC or when the LSECSSON bit is set, reset the backup domain upon each V_{DD} power up (when the BORRSTF flag is set). If V_{BAT} is separate from V_{DD}, also restore the RTC configuration, backup registers and anti-tampering configuration.

2.2.8 LTDC and DSI not functional with Stop 2

Description

If the device enters Stop 2 mode while LTDC and/or DSI are enabled, then upon exiting Stop 2 mode, LTDC and/or DSI fetch wrong data from their respective internal FIFOs. Power-on reset is necessary after exiting Stop 2 mode, for LTDC and/or DSI to display correct data.

Workaround

Choose Stop 0 or Stop 1 instead of Stop 2 when using LTDC and/or DSI.

2.2.9 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.

2.2.10 1-Mbyte devices wrongly configured as 2-Mbyte

Description

STM32L4RxxG/SxxG 1-Mbyte part number devices are wrongly configured as 2-Mbyte devices. However, the Flash memory size FLASH_SIZE[15:0] in Flash size data register is configured correctly to 1 Mbyte.

Consequently, the bank2 erase and page erase in bank2 on those devices are not performed correctly.

In order to check the device configuration:

- Read address 0x1FFF 7500. If bit 24 is set, the device is 2-Mbyte. If the bit 24 is cleared , it is a 1-Mbyte device.

The problem is present on STM32L4RxxG/SxxG devices with date code week 48 2018 or earlier.

The STM32L4RxxG/SxxG 1-Mbyte part number devices with date code week 49 2018 or later have the correct 1-Mbyte configuration.

Workaround

None.

2.2.11 OBL_LAUNCH operation may corrupt SRAM content

Description

OBL_LAUNCH operation may corrupt SRAM1 and SRAM3 content. SRAM2 is safe.

Workaround

Use SRAM2 to store data that must be preserved after OBL_LAUNCH operation.

2.2.12 Irrelevant BOOT0 electrical characteristics

Description

Some data sheet revisions may unduly specify electrical characteristics of BOOT0 pin. However, as BOOT0 signal is shared with PH3 GPIO on the PH3-BOOT0 pin, the GPIO electrical characteristics apply and the BOOT0-specific electrical characteristics are irrelevant.

Workaround

No application workaround is required.

2.2.13 Bootloader not functional on 1-Mbyte devices

Description

The bootloader might not work on STM32L4RxxG/SxxG 1-Mbyte part number devices with date code week 04 2020 or earlier. This depends on the device bootloader version (V9.2 is not functional). In order to check whether the device has the fixed bootloader version or not, perform a read at address 0x1FFF 6FFE:

- If the read value is 0x92 then the version is V9.2 and the device has the non-functional bootloader.
- If the read value is 0x95 then the version is V9.5 and the device has the fixed bootloader.

Note: Note: All STM32L4Rxxx/Sxxx 1-Mbyte and 2-Mbytes part number devices with date code week W05 2020 or later have the new bootloader V9.5.

Workaround

None.

2.2.14 FLASH_ECCR corrupted upon reset or power-down occurring during Flash memory program or erase operation

Description

Reset or power-down occurring during a Flash memory location program or erase operation, followed by a read of the same memory location, may lead to a corruption of the FLASH_ECCR register content.

Workaround

Under such condition, erase the page(s) corresponding to the Flash memory location.

2.2.15 Flash memory ECC single-error correction prevents double-error detection from triggering NMI

Description

Some reference manual revisions may not state that:

- A double-error detection event occurring while the ECCC (or ECCC2) flag is set (after a single-error correction event) does not generate NMI.
- It is recommended to clear the ECCC (or ECCC2) flag as soon as possible, to preserve the ECC error detection capability.

This is a documentation issue rather than a device limitation.

Workaround

No application workaround is applicable provided that the recommendation is respected.

2.2.16 Wrong instruction fetches from flash memory upon wakeup from Sleep or Stop mode when debug in low-power mode is enabled

Description

When debug in low-power mode is enabled, wrong instructions can be fetched from flash memory and executed after waking up from Sleep or Stop mode, causing an unpredictable behavior of the device or a CPU exception.

This issue occurs when the device exits Sleep or Stop mode in the following conditions:

- At least one of DBG_SLEEP or DBG_STOP bit of DBGMCU_CR register is set.
- Interrupts with wakeup capability are disabled at Sleep or Stop entry.
- The flash memory interface gating is enabled by clearing FLASHSMEN bit of RCC_AHB1SMENR (only for Sleep mode).

Note: The issue affects only debug mode and has no effect on real applications.

Workaround

Apply one of the following measures:

- Add an ISB just after WFI (or WFE) instruction.
- Disable the flash memory interface gating, by setting FLASHSMEN bit (valid only for Sleep mode).

2.3 GPIO

2.3.1 Some I/O functions on PC13 pin do not work with RTC_OUT mapped on PB2

Description

With RTC_OUT (RTC_CALIB output) function enabled and mapped on the PB2 pin, the Output_OD, GPIO_Input, EVENTOUT, and RTC_TimeStamp I/O configurations of the PC13 pin do not operate.

Note: The other PC13 I/O configurations operate normally.

Workaround

Do not map enabled RTC_OUT on PB2.

2.4 FW

2.4.1 Firewall protection 127,93 Kbytes size limitation

Description

Only 127,93 Kbytes (131008 bytes) of SRAM1 can be protected by the firewall instead of the full memory space (192 Kbytes of SRAM1). This is because in FW_VDSL register, LENG field is limited to 0x1FFC0 and can not reach 0x20000 (bit 17 is not take into account).

Workaround

None.

2.4.2 Spurious Firewall reset

Description

After configuring Firewall to protect an area within SRAM1, followed by an SRAM1 access, any access to an address outside the protected SRAM1 area, of which the 18 LSBs correspond to an address within the SRAM1 protected area, unduly triggers Firewall reset.

Workaround

Use SRAM2 or SRAM3 as system RAM and avoid accessing SRAM1 before any access to an address outside the protected SRAM1 area, of which the 18 LSBs correspond to an address within the protected SRAM1 area.

2.5 DMA

2.5.1 DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear

Description

Upon a data transfer error in a DMA channel x, both the specific TEIFx and the global GIFx flags are raised and the channel x is normally automatically disabled. However, if in the same clock cycle the software clears the GIFx flag (by setting the CGIFx bit of the DMA_IFCR register), the automatic channel disable fails and the TEIFx flag is not raised.

This issue does not occur with ST's HAL software that does not use and clear the GIFx flag when the channel is active.

Workaround

Do not clear GIFx flags when the channel is active. Instead, use HTIFx, TCIFx, and TEIFx specific event flags and their corresponding clear bits.

2.6 DMAMUX

2.6.1 SOFx not asserted when writing into DMAMUX_CFR register

Description

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

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

Workaround

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

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

Description

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

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

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

Workaround

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

2.6.3 OFx not asserted when writing into DMAMUX_RGCFR register

Description

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

Workaround

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

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

Description

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

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

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

Workaround

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

2.6.5 DMAMUX_RGCFR register is write-only, not read-write

Description

Some reference manual revisions may wrongly state that the DMAMUX_RGCFR register is read-write, while it is write-only.

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

Workaround

No application workaround is required.

2.6.6 DMA request counter not kept at GNBREQ bitfield value as long as the corresponding request channel is disabled

Description

Some reference manual revisions may wrongly state that the DMA request counter is kept at GNBREQ bitfield value as long as the corresponding request channel is disabled.

Instead, at the DMA request counter underrun, the corresponding request generator channel stops generating DMA requests. Then upon the next trigger event, the DMA request counter is automatically reloaded with the GNBREQ bitfield value, regardless whether the corresponding request channel is enabled or disabled.

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

Workaround

No application workaround is required.

2.6.7 Synchronization event discarded if selected input DMA request is not active

Description

Some reference manual revisions may state that upon the detected edge of the synchronization input, the selected input DMA request line is connected to the DMAMUX multiplexer channel x output.

However, if the synchronization event occurs when the selected input DMA request line is not active (not asserted), the synchronization event is discarded. Connecting of a selected input DMA request line becoming active afterward requires a new synchronization event.

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

Workaround

No application workaround is required.

2.7 FMC

2.7.1 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.

2.7.2 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.

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

Description

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

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

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

Workaround

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

2.7.4 Unsupported AHB burst byte write to PSRAM

Description

Burst byte write operations to PSRAM do not work.

Workaround

Do not use burst byte write to program PSRAM. Instead, use half-word, word or double-word burst write.

2.8 OCTOSPI

2.8.1 CSBOUND must not be used

Description

The CS boundary function is expected to split transfers just before the byte with the address $2^{CSBOUND}$, where CSBOUND is the value of CSBOUND[4:0] bitfield of the OCTOSPI_DCR3 register.

This feature does not work properly and must not be used. The CSBOUND[4:0] bitfield of the OCTOSPI_DCR3 register must be left at its reset default value (0b00000).

Workaround

None.

2.8.2 Automatic status-polling mode not functional with octal memory devices

Description

Automatic status-polling mode with octal memories is not functional.

Workaround

Use memory-mapped access to poll the status registers.

2.8.3 Command phase must be octal for octal transfers

Description

Commands whose command phase use one, two, or four signal lines are not supported if one or more of the subsequent phases (address, alternate bytes, data) use eight signal lines.

Workaround

Configure the memory to accept eight-line commands (IMODE[2:0] bitfield of the OCTOSPI_CCR register) if eight-line address/data operation modes are used by the application.

Note: Some memories allow four-line command and eight-line address/data, but all such memories known to date allow an alternative mode where eight-line commands are accepted.

2.8.4 Write data lost with clock mode 3

Description

In eight-line SDR configuration with clock mode 3 (the clock remaining high between transfers), the last one or two bytes of an indirect write transfer may not be sent to the memory after the following sequence of events:

1. All except one or two data bytes of an indirect transfer are written to the data register (OCTOSPI_DR) and the OCTOSPI FIFO buffer.
2. All data bytes from the OCTOSPI FIFO buffer are sent to the memory. The FIFO buffer gets empty and the clock to the memory stops.
3. The last one or two bytes are written to the data register.

Workaround

Use clock mode 0, by setting the bit CKMODE of the OCTOSPI_DCR1 register.

Note: All memories known to date support clock mode 0.

2.8.5 Deadlock upon disabling OCTOSPI during memory-mapped write

Description

Disabling the OCTOSPI peripheral while a memory-mapped write is ongoing (while the BUSY flag of the OCTOSPI_SR register is high) causes the OCTOSPI AHB interface to stall upon the following memory-mapped write request, which may also lead to a general system deadlock.

Workaround

Apply one of the following measures:

- Let memory-mapped writes end (the BUSY flag of the OCTOSPI_SR register gets cleared) before disabling the peripheral.
- Reset the OCTOSPI peripheral when stalled.

2.8.6 Deadlock in dual-quad configuration with Flash memories and odd number of bytes

Description

In dual-quad configuration with Flash memories, bytes at even addresses are written to one memory and bytes at odd addresses to the other. As in this mode the OCTOSPI sends two bytes at a time (one to either memory), it always writes an even total number of bytes.

The peripheral is expected to reject the last byte of any write request with odd number of bytes to contiguous addresses. However, when there is another write request not contiguous with the former, the peripheral hangs when arriving at the last (odd) byte of the first write request, and the last byte remains in the write FIFO buffer.

Workaround

Apply one of the following measures:

- Ensure that all write requests in dual-quad configuration with Flash memories contain even number of bytes.
- Reset the OCTOSPI peripheral when stalled.

2.8.7 No transfer error interrupt upon indirect write to address exceeding the limit

Description

The transfer error flag TEF of the OCTOSPI_SR register is not set and interrupt not generated with the octal memories upon an indirect write to an address exceeding the limit.

Workaround

None.

2.8.8 DHQC not effective if DDTR not set

Description

With the DHQC bit of the OCTOSPI_TCR set (to insert a quarter-cycle delay) and the DDTR bit of the OCTOSPI_CCR register cleared (no DTR for the data phase), the delay is not inserted.

Workaround

Always set the DDTR bit, even for transactions with no data phase.

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

Description

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

Workaround

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

2.8.10 Unaligned AHB write requests not supported

Description

Upon a non-aligned write access request from AHB master, the OCTOSPI peripheral does not automatically align the data. With byte and half-word write accesses, the lowest-significant byte and the two lowest-significant input data bytes, respectively, are written to the memory, regardless of the non-alignment attribute.

Workaround

None.

2.8.11 Write operation in DTR octal-SPI mode starting at odd address is not supported

Description

When the memory type is different from HyperBus™ (the MTYP[2:0] bitfield of the OCTOSPI_DCR1 register set to a value other than 100 or 101), the DQS write mask for writes starting with odd address is not asserted even though enabled (DQSE bit of the OCTOSPI_WCCR register set).

The same failure occurs for all types of memories, including HyperBus™, if a write starting with odd address is requested and the OCTOSPI is busy.

Workaround

None.

2.8.12 Data masking for odd number of byte writes only working with D1/D0 ordering

Description

The data masking for odd number of byte writes operates with only the memory types using D1/D0 ordering.

Workaround

Set the MTYP[2:0] bitfield of the OCTOSPI_DCR1 register to a memory type with D1/D0 ordering, for example Macronix (MTYP[2:0] = 001).

2.8.13 Memory-mapped read of the last memory space byte not possible in SDR octal-SPI mode

Description

Memory-mapped read of the last byte of the memory space defined through the DEVSIZE[4:0] bitfield of the OCTOSPI_DCR1 register spuriously always returns zero. A subsequent memory-mapped read not separated from the previous memory-mapped read with a command causes the AHB interface to hang, with the HREADY flag never set.

Note: This failure does not occur in DTR octal-SPI mode.

Workaround

Apply one of the following measures:

- Avoid reading the last byte of the memory space through memory-mapped access. Use indirect read instead.
- Set DEVSIZE value so that the memory space it defines exceeds the memory size, then handle the memory boundary by software.

2.8.14 Single-byte memory-mapped write to odd DTR octal-SPI address failing when a higher-priority event occurs

Description

In DTR octal-SPI mode, single-byte memory-mapped write request to odd memory address, followed by a higher-priority event such as a memory-mapped read, results in the OCTOSPI peripheral to spuriously DQS-mask both bytes transferred in the same data clock period, which causes the loss of the byte sent to memory.

Workaround

Apply one of the following measures:

- Avoid using single-byte memory-mapped writes.
- Use indirect write to send a single byte to a memory.

2.8.15 Memory-mapped write data to odd start address corrupted when BUSY = 0 and not in DTR octal-SPI mode

Description

In all modes except for DTR octal-SPI mode, if the first memory-mapped write operation that is requested while BUSY = 0 is a write to an odd address, the first byte is written correctly, but then this same byte is written to the next address and all subsequent bytes are shifted up one address.

Workaround

Keep the BUSY flag set by using the following method:

1. Disable the timeout counter.
2. Avoid to abort the application.
3. As soon as the OCTOSPI is configured and ready to be used in Memory-mapped mode, perform a dummy read.

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

Description

A command is composed of five phases:

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

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

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

Workaround

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

- Send one byte of address in SDR quad-SPI mode (ADMODE = 011, ADSIZE = 00, ADDDTR = 0)
- Send two bytes of address in SDR octal-SPI mode (ADMODE = 100, ADSIZE = 01, ADDDTR = 0)
- Send four bytes of address in DTR octal-SPI mode (ADMODE = 100, ADSIZE = 11, ADDDTR = 1)
- Send two bytes of instruction in DTR quad-SPI mode (IMODE = 011, ISIZE = 01, IDDTR = 1)
- Send one instruction byte in octal followed by one dummy cycle.
- Send one instruction byte in octal followed by one alternate byte in octal.

2.8.17 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.

2.8.18 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.

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

Description

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

Workaround

None.

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

Description

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

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

Workaround

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

2.8.21 DQS output enabled too late on write

Description

At high kernel clock frequency, a hold violation on the DQS signal may occur upon sending a new write command to memory device, which possibly causes data transfer failure in two types of situations:

- DQS signal, high before starting the data phase, falls at the start of transfer. Because of the hold violation, DQS output is sampled high with the first output clock edge, which results in the memory device ignoring the first byte sent.
- DQS signal, low before starting the data phase, rises at the start of transfer. Because of the hold violation, DQS output is sampled low with the first output clock edge, which results in its wrong interpretation by the memory device.

Workaround

Reduce the kernel clock frequency. Adapt the output clock division factor accordingly so as to keep the output clock frequency unchanged.

2.8.22 PSRAM not supported

Description

Some reference manual and datasheet revisions unduly mention PSRAM while this type of memory is not supported by the device.

This is a documentation issue rather than a product limitation.

Workaround

Not relevant.

2.8.23 Deadlock on memory-mapped write with timeout enabled

Description

When OCTOSPI is configured in Memory-mapped mode, if the software is performing consecutive write operations (at sequential addresses) and if the number of bytes to write is not a multiple of four bytes, the next write operation to a non-consecutive address may be stalled indefinitely if a timeout event occurs at this time.

Workaround

Apply one of the following measures to avoid this issue when performing memory-mapped writes:

- Disable the timeout feature.
- Always use four-byte AHB write.

2.9 OCTOSPIM

2.9.1 Unaligned write access to OCTOSPIM configuration registers failing

Description

Unaligned write access to OCTOSPIM configuration registers is discarded, with hard fault. The de-assertion of *hready* AHB signal then takes three cycles, which is not compliant with the AHB standard that defines two cycles.

Workaround

Avoid unaligned write accesses to OCTOSPIM configuration registers.

2.10 ADC

2.10.1 Writing ADC_JSQR when JADCSTART and JQDIS are set may lead to incorrect behavior

Description

Some reference manual revisions specifiy that the ADC_JSQR register can be written when an injected conversion is ongoing (JADCSTART = 1). This may lead to unpredictable ADC behavior if the queues of context are not enabled (JQDIS = 1).

Workaround

No application workaround is required for this description inaccuracy issue.

2.10.2 ADEN bit cannot be set immediately after the ADC calibration is done

Description

Some reference manual revisions may not indicate that the ADEN bit cannot be set while the ADCAL bit is set and during a four ADC clock cycles after the ADCAL bit is cleared by hardware (end of the calibration).

Otherwise, if the ADEN bit is set during this four ADC clock cycle period, it will be reset by the calibration logic and the ADC will stay disabled. This is due to the fact that there is an internal reset of the ADEN bit four ADC clock cycles after the ADCAL bit is cleared by hardware.

Workaround

No application workaround is required for this description inaccuracy issue.

2.10.3 New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0

Description

Once an injected conversion sequence is complete, the queue is consumed and the context changes according to the new ADC_JSQR parameters stored in the queue. This new context is applied for the next injected sequence of conversions.

However, the programming of the new context in ADC_JSQR (change of injected trigger selection and/or trigger polarity) may launch the execution of this context without waiting for the trigger if:

- the queue of context is enabled (JQDIS cleared to 0 in ADC_CFGR), and
- the queue is never empty (JQM cleared to 0 in ADC_CFGR), and
- the injected conversion sequence is complete and no conversion from previous context is ongoing

Workaround

Apply one of the following measures:

- Ignore the first conversion.
- Use a queue of context with JQM = 1.
- Use a queue of context with JQM = 0, only change the conversion sequence but never the trigger selection and the polarity.

2.10.4 Two consecutive context conversions fail when writing new context in ADC_JSQR just after previous context completion with JQDIS = 0 and JQM = 0

Description

When an injected conversion sequence is complete and the queue is consumed, writing a new context in ADC_JSQR just after the completion of the previous context and with a length longer that the previous context, may cause both contexts to fail. The two contexts are considered as one single context. As an example, if the first context contains element 1 and the second context elements 2 and 3, the first context is consumed followed by elements 2 and 3 and element 1 is not executed.

This issue may happen if:

- the queue of context is enabled (JQDIS cleared to 0 in ADC_CFGR), and
- the queue is never empty (JQM cleared to 0 in ADC_CFGR), and
- the length of the new context is longer than the previous one

Workaround

If possible, synchronize the writing of the new context with the reception of the new trigger.

2.10.5 ADC_AWDy_OUT reset by non-guarded channels

Description

ADC_AWDy_OUT is set when a guarded conversion of a regular or injected channel is outside the programmed thresholds. It is reset after the end of the next guarded conversion that is inside the programmed thresholds.

However, the ADC_AWDy_OUT signal is also reset at the end of conversion of non-guarded channels, both regular and injected.

Workaround

When ADC_AWDy_OUT is enabled, it is recommended to use only the ADC channels that are guarded by a watchdog.

If ADC_AWDy_OUT is used with ADC channels that are not guarded by a watchdog, take only ADC_AWDy_OUT rising edge into account.

2.10.6 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.

2.10.7 Spurious temperature measurement due to spike noise

Description

Depending on the MCU activity, internal interference may cause temperature-dependent spike noise on the temperature sensor output to the ADC, resulting in occasional spurious (outlying) temperature measurement.

Workaround

Perform a series of measurements and process the acquired data samples such as to obtain a mean value not affected by the outlying samples.

For this, it is recommended to use interquartile mean (IQM) algorithm with at least 64 samples. IQM is based on rejecting the quarters (quartiles) of sample population with the lowest and highest values and on computing the mean value only using the remaining (interquartile) samples.

The acquired sample values are first sorted from lowest to highest, then the sample sequence is truncated by removing the lowest and highest sample quartiles.

Example:

Table 5. Measurement result after IQM post-processing

Data	Sample												Mean
	1	2	3	4	5	6	7	8	9	10	11	12	
Acquired	17.2	10.92	9.56	2.12	9.82	10.72	10.6	3.5	9.46	9.78	9.5	1.1	8.69
Sorted	1.1	2.12	3.5	9.46	9.5	9.56	9.78	9.82	10.6	10.72	10.92	17.2	8.69
Truncated	-	-	-	9.46	9.5	9.56	9.78	9.82	10.6	-	-	-	9.79

The measurement result after the IQM post-processing in the example is 9.79. For consistent results, use a minimum of 64 samples. It is recommended to optimize the code performing the sort task such as to minimize its processing power requirements.

2.10.8 Wrong ADC differential conversion result for channel 5

Description

The ADC 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.

2.10.9 Selected external ADC inputs unduly clamped to V_{DD} when all analog peripherals are disabled

Description

When all analog peripherals (other than VREFBUF) are disabled, the GPIO(s) selected as ADC input(s) are unduly clamped (through a parasitic diode) to V_{DD} instead to V_{DDA}. As a consequence, the input voltage is limited to V_{DD} + 0.3 V even if V_{DDA} is higher than V_{DD} + 0.3 V.

Note: The selection of GPIOs as ADC inputs is done with the SQy and JSQy bitfields of the ADC_SQRx and ADC_JSQR registers, respectively.

VREFBUF enable/disable has no impact to the issue described.

Workaround

Apply one of the following measures:

- Use V_{DDA} lower than V_{DD} + 0.3 V.
- Keep at least one analog peripheral (other than VREFBUF) enabled if GPIOs are selected as ADC inputs.
- Deselect GPIOs as ADC inputs (by clearing ADC_SQRx or/and ADC_JSQR registers) when no analog peripheral (other than VREFBUF) is enabled.

2.11 DAC

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

Description

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

Workaround

Apply the following sequence:

1. Perform one write access to any data register.
2. Program the MODE bitfield of the DAC_MCR register.

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

Description

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

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

Workaround

None.

2.12 COMP

2.12.1 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.

2.13 DSI

2.13.1 Tearing effect parasitic detection

Description

When using the tearing effect mechanism over the DSI link in the Adapted Command mode, the tearing effect interrupt flag (TEIF) of the DSI wrapper interrupt status register (DSI_WISR) is asserted when an acknowledge trigger is received from the display.

An acknowledge trigger can be received from the display:

- for each packet when the acknowledge request enable (ARE) bit of the DSI Host command mode configuration register (DSI_CMCR) is set
- when a display response is expected

Workaround

Do not use the tearing effect over the link but use the dedicated TE pin.

When using the tearing effect over the link, do not use the tearing effect interrupt nor the automatic refresh mode. Instead, launch the display refresh immediately after a set_tear_on or a set_scanline DCS command (as the display is driving the DSI link until the tearing effect occurs, the refresh is automatically stalled until the tearing effect occurs).

2.13.2 Incorrect calculation of the time to activate the clock between HS transmissions

Description

In the automatic clock lane control mode, the DSI Host can turn off the clock lane between two high-speed transmissions.

To do so, the DSI Host calculates the time required for the clock lane to change from either: high-speed to low-power, or from low-power to high-speed.

These timings are configured by the HS2LP_TIME[9:0] and LP2HS_TIME[9:0] bitfields of the DSI Host clock lane timer configuration register (DSI_CLTCR). The DSI Host does not calculate the value configured in LP2HS_TIME plus HS2LP_TIME but twice the value configured in HS2LP_TIME instead.

Workaround

Configure HS2LP_TIME and LP2HS_TIME with the same value as the maximum of either HS2LP_TIME and LP2HS_TIME.

As an example, if HS2LP_TIMER = 44 and LP2HS_TIME = 113 configure the register fields as follows:

- HS2LP_TIME = 113
- LP2HS_TIME =113

2.13.3 The immediate update procedure may fail

Description

The immediate update procedure implies that both the UR and the EN bits of the DSI Host video shadow control register (DSI_VSCR) are initially cleared, and are set by the same instruction.

In some cases, the immediate update procedure fails due to a race condition between the two signals. This leads the DSI Host to wait for the next frame end before updating the configuration.

Workaround

After an immediate update procedure, check the configuration is updated by reading the auto-cleared bit UR.

If the UR bit is not cleared, repeat the process by writing first 0x0000 then 0x0101 in DSI_VSCR.

2.13.4 Failing DSI read operation

Description

Following any type of read command sent by the device to a DSI panel display, the DSI peripheral on the device sporadically fails to capture valid data returned by the DSI panel display and signals packet size error (by setting the PSE flag of the DSI_ISR1 register), as if the display returned invalid data or aborted the transaction.

The defect occurs randomly, without any obvious dependency on a particular condition.

Workaround

Upon detecting a packet size error, resend the read command in a loop until the data is received without reporting packet size error.

The sequence is as follows:

1. Send a read command
2. Read data from the FIFO if it is not empty (if the PRDFE flag of the DSI_GPSR register is zero), pass otherwise
3. If the RCB flag of the DSI_GPSR register is set, signaling *read command busy*, revert to the point 3. Pass otherwise.
4. If the PSE flag of the DSI_ISR1 register is set, revert to the point 2 (read failure occurred).

2.13.5 DSI-PHY may not start properly when the DSI PLL is restarted

Description

When the DSI PLL is restarted (for example after the Stop mode), the DSI-PHY may not restart properly and, consequently, the display becomes frozen.

Workaround

Reset the DSI-PHY after the DSI PLL restarted and before initiating DSI transmissions, respecting the following sequence:

1. Disable the DSI PLL and wait for unlock flag.
2. Enter Stop mode.
3. Wake up from Stop mode.
4. Enable the HSE oscillator.
5. Enable the DSI PLL and wait for lock flag.
6. Reset the DSI-PHY.
7. Start DSI transmissions.

2.14 TSC

2.14.1 Inhibited acquisition in short transfer phase configuration

Description

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

- The PGPSC[2:0] bitfield set to 000 and the CTPL[3:0] bitfield to 0000 or 0001
- The PGPSC[2:0] bitfield set to 001 and the CTPL[3:0] bitfield to 0000

Failure to respect this restriction leads to an inhibition of the acquisition.

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

Workaround

No application workaround is required.

2.14.2 TSC signal-to-noise concern under specific conditions

Description

V_{DD} equal to or greater than V_{DDA} may lead (depending on part) to some degradation of the signal-to-noise ratio on the TSC analog I/O group 2.

The lower are the sampling capacitor (C_S) and the sensing electrode (C_X) capacitances, the worse is the signal-to-noise ratio degradation.

Workaround

Apply one of the following measures:

- Maximize C_S capacitance.
- Use the analog I/O group 2 as active shield.

2.15 HASH

2.15.1 Superseded suspend sequence for data loaded by DMA

Description

The section *HASH / Context swapping / Data loaded by DMA / Current context saving* of some reference manual revisions may suggest the following suspend sequence for using HASH with DMA:

1. Clear the DMAE bit to disable the DMA interface.
2. Wait until the current DMA transfer is complete (wait for DMAS = 0 in the HASH_SR register).

This recommendation is obsolete and superseded with the following sequence that suspends then resumes the secure digest computing in order to swap the context:

Suspend:

1. In Polling mode, wait for BUSY = 0. If the DCIS bit of the HASH_SR register is set, the hash result is available and the context swapping is useless. Otherwise, go to step 2.
2. In Polling mode, wait for BUSY = 1.
3. Disable the DMA channel. Then clear the DMAE bit of the HASH_CR register.
4. In Polling mode, wait for BUSY = 0. If the DCIS bit of the HASH_SR register is set, the hash result is available and the context swapping is useless. Otherwise, go to step 5.
5. Save the HASH_IMR, HASH_STR, HASH_CR, and HASH_CSR0 to HASH_CSR37 registers. The HASH_CSR38 to HASH_CSR53 registers must also be saved if an HMAC operation is ongoing.

Resume:

1. Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again. Do not forget to take into account the words already pushed into the FIFO if NBW[3:0] is higher than 0x0.
2. Program the values saved in memory to the HASH_IMR, HASH_STR, and HASH_CR registers.
3. Initialize the hash processor by setting the INIT bit of the HASH_CR register.
4. Program the values saved in memory to the HASH_CSRx registers.
5. Restart the processing from the point of interruption, by setting the DMAE bit.

Note: To optimize the resume process when NBW[3:0] = 0x0, HASH_CSR22 to HASH_CSR37 registers do not need to be saved then restored as the FIFO is empty.

This is a documentation issue rather than a product limitation.

Workaround

No application workaround is required as long as the new sequence is applied.

2.15.2 Superseded suspend sequence for data loaded by the CPU

Description

The section *HASH / Context swapping / Data loaded by software* of some reference manual revisions may instruct that *“the user application must wait until DINIS ≠ 1 (last block processed and input FIFO empty) or NBW 0 (FIFO not full and no processing ongoing)”*.

This instruction is obsolete and superseded with the following:

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing.

To suspend the processing of a message, proceed as follows after writing 16 words 32-bit (plus one if it is the first block):

1. In Polling mode, wait for BUSY = 0, then poll if the DINIS status bit is set to 1. In Interrupt mode, implement the next step in DINIS interrupt handler (recommended).
2. Store the contents of the following registers into memory:
   - HASH_IMR
   - HASH_STR
   - HASH_CR
   - HASH_CSR0 to HASH_CSR37 and, if an HMAC operation is ongoing, also HASH_CSR38 to HASH_CSR53

To resume the processing of a message, proceed as follows:

1. Write the HASH_IMR, HASH_STR, and HASH_CR registers with the values saved in memory.
2. Initialize the hash processor by setting the INIT bit of the HASH_CR register.
3. Write the HASH_CSRx registers with the values saved in memory.
4. Restart the processing from the point of interruption.

Note: To optimize the resume process when NBW[3:0]=0x0, HASH_CSR22 to HASH_CSR37 registers do not need to be saved then restored as the FIFO is empty.

This is a documentation issue rather than a product limitation.

Workaround

No application workaround is required as long as the new sequence is applied.

2.16 TIM

2.16.1 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.

2.16.2 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.

2.16.3 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).

2.16.4 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.

2.17 LPTIM

2.17.1 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 RCC_APByRSTRz register.

2.17.2 Device may remain stuck in LPTIM interrupt when clearing event flag

Description

This limitation occurs when the LPTIM is configured in interrupt mode (at least one interrupt is enabled) and the software clears any flag in LPTIM_ISR register by writing its corresponding bit in LPTIM_ICR register. If the interrupt status flag corresponding to a disabled interrupt is cleared simultaneously with a new event detection, the set and clear commands might reach the APB domain at the same time, leading to an asynchronous interrupt signal permanently stuck high.

This issue can occur either during an interrupt subroutine execution (where the flag clearing is usually done), or outside an interrupt subroutine.

Consequently, the firmware remains stuck in the LPTIM interrupt routine, and the device cannot enter Stop mode.

Workaround

To avoid this issue, it is strongly advised to follow the recommendations listed below:

- Clear the flag only when its corresponding interrupt is enabled in the interrupt enable register.
- If for specific reasons, it is required to clear some flags that have corresponding interrupt lines disabled in the interrupt enable register, it is recommended to clear them during the current subroutine prior to those which have corresponding interrupt line enabled in the interrupt enable register.
- Flags must not be cleared outside the interrupt subroutine.

Note: The standard clear sequence implemented in the HAL_LPTIM_IRQHandler in the STM32Cube is considered as the proper clear sequence.

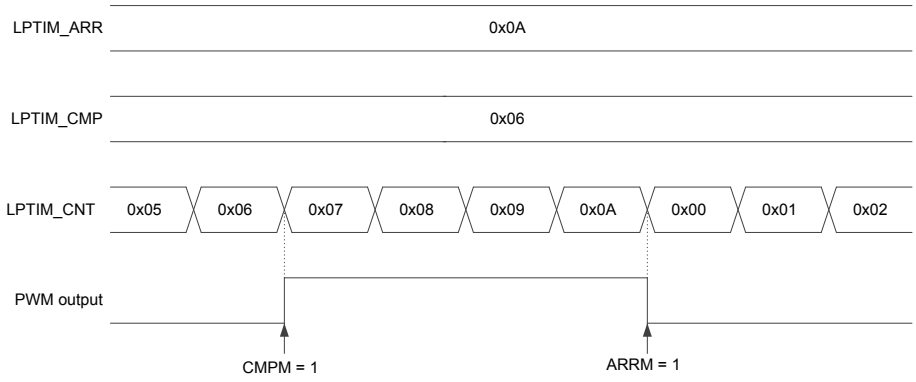
2.17.3 LPTIM events and PWM output are delayed by 1 kernel clock cycle

Description

The compare match event (CMPM), auto reload match event (ARRM), PWM output level and interrupts are updated with a delay of one kernel clock cycle.

Consequently, it is not possible to generate PWM with a duty cycle of 0% or 100%.

The following waveform gives the example of PWM output mode and the effect of the delay:



Workaround

Set the compare value to the desired value minus 1. For instance in order to generate a compare match when LPTM_CNT = 0x08, set the compare value to 0x07.

2.17.4 LPTIM1 outputs cannot be configured as open-drain

Description

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

Workaround

None.

2.18 RTC and TAMP

2.18.1 RTC interrupt can be masked by another RTC interrupt

Description

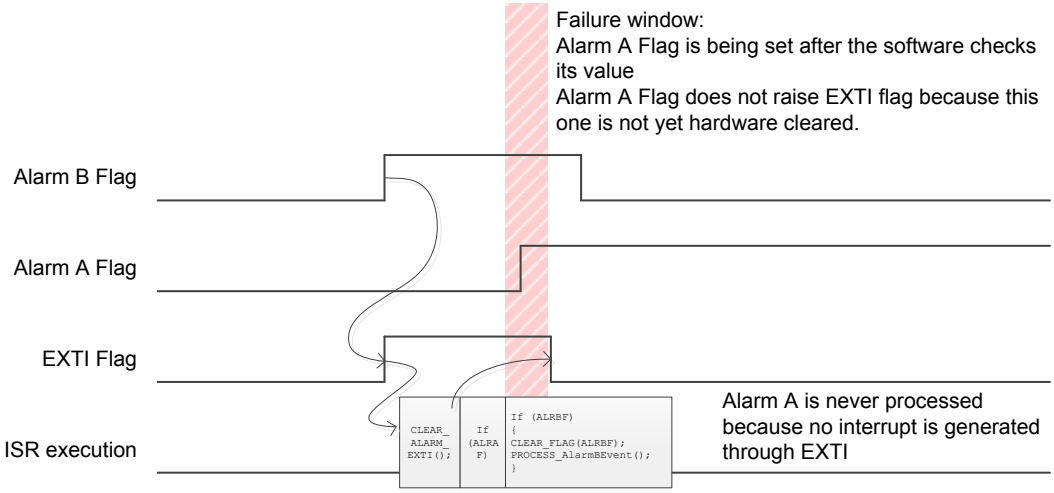
One RTC interrupt request can mask another RTC interrupt request if they share the same EXTI configurable line. For example, interrupt requests from Alarm A and Alarm B or those from tamper and timestamp events are OR-ed to the same EXTI line (refer to the *EXTI line connections* table in the *Extended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

The following code example and figure illustrate the failure mechanism: The Alarm A event is lost (fails to generate interrupt) as it occurs in the failure window, that is, after checking the Alarm A event flag but before the effective clear of the EXTI interrupt flag by hardware. The effective clear of the EXTI interrupt flag is delayed with respect to the software instruction to clear it.

Alarm interrupt service routine:

```
void RTC_Alarm_IRQHandler(void)
{
    CLEAR_ALARM_EXTI(); /* Clear the EXTI line flag for RTC alarms*/
    If(ALRAF) /* Check if Alarm A triggered ISR */
    {
        CLEAR_FLAG(ALRAF); /* Clear the Alarm A interrupt pending bit */
        PROCESS_AlarmAEvent(); /* Process Alarm A event */
    }
    If(ALRBF) /* Check if Alarm B triggered ISR */
    {
        CLEAR_FLAG(ALRBF); /* Clear the Alarm B interrupt pending bit */
        PROCESS_AlarmBEvent(); /* Process Alarm B event */
    }
}
```

Figure 1. Masked RTC interrupt



Workaround

In the interrupt service routine, apply three consecutive event flag ckecks - source one, source two, and source one again, as in the following code example:

```
void RTC_Alarm_IRQHandler(void)
{
    CLEAR_ALARM_EXTI(); /* Clear the EXTI's line Flag for RTC Alarm */
    If(ALRAF) /* Check if AlarmA triggered ISR */
    {
        CLEAR_FLAG(ALRAF); /* Clear the AlarmA interrupt pending bit */
        PROCESS_AlarmAEvent(); /* Process AlarmA Event */
    }
    If(ALRBF) /* Check if AlarmB triggered ISR */
    {
        CLEAR_FLAG(ALRBF); /* Clear the AlarmB interrupt pending bit */
        PROCESS_AlarmBEvent(); /* Process AlarmB Event */
    }
    If(ALRAF) /* Check if AlarmA triggered ISR */
    {
        CLEAR_FLAG(ALRAF); /* Clear the AlarmA interrupt pending bit */
        PROCESS_AlarmAEvent(); /* Process AlarmA Event */
    }
}
```

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

Description

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

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

Workaround

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

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

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

Description

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

2.18.4 RTC_REFIN and RTC_OUT on PB2 not operating in Stop 2 mode

Description

In Stop 2 low-power mode, the RTC_REFIN function does not operate and the RTC_OUT function does not operate if mapped on the PB2 pin.

Workaround

Apply one of the following measures:

- Use Stop 1 mode instead of Stop 2. This ensures the operation of both functions.
- Map RTC_OUT to the PC13 pin. This ensures the operation of the RTC_OUT function in either low-power mode. However, it has no effect to the RTC_REFIN function.

2.19 I2C

2.19.1 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave

Description

An I^2C-bus master generates STOP condition upon non-acknowledge of I^2C address that it sends. This applies to 7-bit address as well as to each byte of 10-bit address.

When the MCU set as I^2C-bus master transmits a 10-bit address of which the first byte (5-bit header + 2 MSBs of the address + direction bit) is not acknowledged, the MCU duly generates STOP condition but it then cannot start any new I^2C-bus transfer. In this spurious state, the NACKF flag of the I2C_ISR register and the START bit of the I2C_CR2 register are both set, while the START bit should normally be cleared.

Workaround

In 10-bit-address master mode, if both NACKF flag and START bit get simultaneously set, proceed as follows:

1. Wait for the STOP condition detection (STOPF = 1 in I2C_ISR register).
2. Disable the I2C peripheral.
3. Wait for a minimum of three APB cycles.
4. Enable the I2C peripheral again.

2.19.2 Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C

Description

The correct use of the I2C peripheral, if the wakeup from Stop mode by I2C is disabled (WUPEN = 0), is to disable it (PE = 0) before entering Stop mode, and re-enable it when back in Run mode.

Some reference manual revisions may omit this information.

Failure to respect the above while the MCU operating as slave or as master in multi-master topology enters Stop mode during a transfer ongoing on the I^2C-bus may lead to the following:

1. BUSY flag is wrongly set when the MCU exits Stop mode. This prevents from initiating a transfer in master mode, as the START condition cannot be sent when BUSY is set.
2. If clock stretching is enabled (NOSTRETCH = 0), the SCL line is pulled low by I2C and the transfer stalled as long as the MCU remains in Stop mode.

   The occurrence of such condition depends on the timing configuration, peripheral clock frequency, and I^2C-bus frequency.

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

Workaround

No application workaround is required.

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

Description

The I^2C-bus specification and user manual specify a minimum data setup time ($t_{SU;DAT}$) as:

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

The device does not correctly sample the I^2C-bus SDA line when $t_{SU;DAT}$ is smaller than one I2C kernel clock (I^2C-bus peripheral clock) period: the previous SDA value is sampled instead of the current one. This can result in a wrong receipt of slave address, data byte, or acknowledge bit.

Workaround

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

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

2.19.4 Spurious bus error detection in master mode

Description

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

Workaround

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

2.19.5 Last-received byte loss in reload mode

Description

If in master receiver mode or slave receive mode with SBC = 1 the following conditions are all met:

- I^2C-bus stretching is enabled (NOSTRETCH = 0)
- RELOAD bit of the I2C_CR2 register is set
- NBYTES bitfield of the I2C_CR2 register is set to N greater than 1
- byte N is received on the I^2C-bus, raising the TCR flag
- N - 1 byte is not yet read out from the data register at the instant TCR is raised,

then the SCL line is pulled low (I^2C-bus clock stretching) and the transfer of the byte N from the shift register to the data register inhibited until the byte N-1 is read and NBYTES bitfield reloaded with a new value, the latter of which also clears the TCR flag. As a consequence, the software cannot get the byte N and use its content before setting the new value into the NBYTES field.

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

Workaround

- In master mode or in slave mode with SBC = 1, use the reload mode with NBYTES = 1.
- In master receiver mode, if the number of bytes to transfer is greater than 255, do not use the reload mode. Instead, split the transfer into sections not exceeding 255 bytes and separate them with repeated START conditions.
- Make sure, for example through the use of DMA, that the byte N - 1 is always read before the TCR flag is raised. Specifically for I2C instances with independent clock, make sure that it is always read earlier than four APB clock cycles before the receipt of the last data bit of byte N and thus the TCR flag raising.

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

2.19.6 Spurious master transfer upon own slave address match

Description

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

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

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

Workaround

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

Normal mode (SBC = 0):

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

Slave byte control mode (SBC = 1):

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

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

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

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

Description

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

2.19.8 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.

2.19.9 Transmission stalled after first byte transfer

Description

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

2.20 USART

2.20.1 RTS is active while RE = 0 or UE = 0

Description

The RTS line is driven low as soon as RTSE bit is set, even if the USART is disabled (UE = 0) or the receiver is disabled (RE = 0), that is, not ready to receive data.

Workaround

Upon setting the UE and RE bits, configure the I/O used for RTS into alternate function.

2.20.2 Receiver timeout counter wrong start in two-stop-bit configuration

Description

Some reference manual revisions may omit the information that in two-stop-bit configuration, the receiver timeout counter starts counting from the end of the second stop bit of the last character instead of starting from the end of the first stop bit. The application must subtract one bit duration from the value in the RTO bitfield of the USARTx_RTOR register.

This is a documentation issue rather than a product limitation.

Workaround

No application workaround is required or applicable.

2.20.3 UDR flag set while the SPI slave transmitter is disabled

Description

When the USART is used in SPI slave receive mode, the underrun flag (UDR bit of USART_ISR register) might be set even if the SPI slave transmitter is disabled (TE bit cleared in USART_CR1 register).

Workaround

Apply one of the following measures:

- Ignore the UDR flag when the SPI slave transmitter is disabled.
- Clear the UDR flag every time it is set, even if the SPI slave transmitter is disabled.
- Write dummy data in the USART_TDR register to avoid setting the UDR flag.

2.20.4 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.

2.20.5 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.

2.21 LPUART

2.21.1 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.

2.21.2 LPUART1 outputs cannot be configured as open-drain

Description

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

Workaround

None.

2.22 SPI

2.22.1 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.

2.22.2 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.

2.22.3 CRC error in SPI slave mode if internal NSS changes before CRC transfer

Description

Some reference manual revisions may omit the information that the device operating as SPI slave must be configured in software NSS control if the SPI master pulses the NSS (for (for example in NSS pulse mode).

Otherwise, the transition of the internal NSS signal after the CRCNEXT flag is set might result in wrong CRC value computed by the device and, as a consequence, in a CRC error. As a consequence, the NSS pulse mode cannot be used along with the CRC function.

This is a documentation error rather than a product limitation.

Workaround

No application workaround is required as long as the device operating as SPI slave is duly configured in software NSS control.

2.23 SAI

2.23.1 Last SAI_SCK clock pulse truncated upon disabling SAI master with NODIV set and MCKDIV greater than one

Description

When disabling, during the communication, the SAI peripheral configured as master with the NODIV bit set (SAI_MCLK_x master clock disabled) and the MCKDIV[5:0] bitfield value greater than one (division ratio greater than one) in the corresponding SAI_xCR1 register , the device may truncate the last SAI_SCK_x clock pulse of the transaction, potentially causing a failure to the external codec logic.

Workaround

Either use SAI_MCLK_x master clock (clear NODIV) or, if SAI_MCLK_x is not used, keep MCKDIV[5:0] at zero or one.

2.23.2 Last SAI_MCLK clock pulse truncated upon disabling SAI master

Description

When disabling, during the communication, the SAI peripheral configured as master with the OUTDRIV bit of the corresponding SAI_xCR1 register cleared, the device may truncate the last SAI_MCLK_x bit clock pulse of the transaction, potentially causing a failure to the external codec logic.

Workaround

Set the OUTDRIV bit of the corresponding SAI_xCR1 register.

2.24 bxCAN

2.24.1 bxCAN time-triggered communication mode not supported

Description

The time-triggered communication mode described in the reference manual is not supported. As a result, timestamp values are not available. The TTCM bit of the CAN_MCR register must be kept cleared (time-triggered communication mode disabled).

Workaround

None.

2.25 OTG_FS

2.25.1 Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_FS registers

Description

When the USB on-the-go full-speed peripheral is in Device mode, interrupting transmit FIFO write sequence with read or write accesses to OTG_FS endpoint-specific registers (those ending in *0* or *x*) leads to corruption of the next data written to the transmit FIFO.

Workaround

Ensure that the transmit FIFO write sequence is not interrupted with accesses to the OTG_FS registers.

Revision history

Table 6. **Document revision history**

Date	Version	Changes
10-Oct-2017	1	Initial release.
30-Mar-2018	2	Added errata: • Flash memory might not be accessible when AHB prescaler is greater than eight • Flash OPTVERR flag is always set after system reset • HSE oscillator long startup at low voltage • Octal indirect read operation is not started when OctoSPI is configured with address phase disabled • Spurious temperature measurement due to spike noise • RTC_REFIN and RTC_OUT on PB2 not operating in Stop 2 mode • BSY bit may stay high when SPI is disabled. Modified section Core and errata: • HSE/32 is not available for TIM16 input capture if RTC clock is disabled or other than HSE • LPTIM1 outputs cannot be configured as open-drain • 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave • Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C • Wrong data sampling when data setup time (tSU;DAT) is shorter than one I2C kernel clock period • Spurious bus error detection in master mode • section 2.15.4: Last-received byte loss in reload mode • LPUART1 outputs cannot be configured as opendrain • BSY bit may stay high at the end of a data transfer in slave mode • CRC error in SPI slave mode if internal NSS changes before CRC transfer Removed erratum *START bit is not cleared when the address is not acknowledged by the slave* from section I2C.
21-Jun-2018	3	Added documentation errata to the document, summarized in Table 4:*Summary of documentation errata* and introduced in the cover page. Added errata: • Spurious Firewall reset is generated when accessing FMC or OctoSPI • DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear • Byte and half-word accesses not supported • DMAMUX_RGCFR register is write-only, not readwrite • Request trigger overrun misdetection • SOFx not asserted when writing into DMAMUX_CFR register • DMA request counter not kept at GNBREQ bitfield value as long as the corresponding request channel is disabled • Synchronization event discarded if selected input DMA request is not active • Memory-mapped read of the last byte of the memory is not possible when in octal SDR mode • A memory-mapped write request with only one-byte length at odd-start address finished by any event is masked • Writing the register ADCx_JSQR when JADCSTART=1 and JQDIS=1 might lead to incorrect behavior • Spurious master transfer upon own slave address match • START bit is cleared upon setting ADDRCF, not upon address match

Date	Version	Changes
		Modified erratum *Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C* and reclassified as documentation erratum.
19-Dec-2018	4	Added errata: • SDMMC1SMEN bit of RCC_AHB2SMENR register only modifiable with word access • First double-word of Flash memory corrupted upon reset or power-down while programming • Unstable LSI when it clocks RTC or CSS on LSE • LTDC and DSI not functional with Stop 2 • Regulator startup failure at low VDD • 1-Mbyte devices wrongly configured as 2-Mbyte • OFx not asserted when writing into DMAMUX_RGCFR register • Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event • Unaligned AHB write requests not supported • Data masking for odd number of byte writes only working with D1/D0 ordering • Single-byte memory-mapped write to odd octal DDR address failing when a higher-priority event occurs • Calendar initialization may fail in case of consecutive INIT mode entry Modified errata: • DMAMUX_RGCFR register is write-only, not read-write moved to documentation errata section • SOFx not asserted when writing into DMAMUX_CFR register moved to device limitations section • CSBOUND shall not be used updated workaround qualifier • Auto-polling mode not functional with new octal memories updated workaround qualifier • No transfer error interrupt upon indirect write to address exceeding the limit updated workaround qualifier • Writing ADCx_JSQR when JADCSTART and JQDIS are set might lead to incorrect behavior updated workaround • Spurious temperature measurement due to spike noise • updated function section from TEMP to ADC • Inhibited acquisition in short transfer phase configuration moved to documentation errata section • Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C moved to documentation errata • Wrong data sampling when data setup time (tSU;DAT) is shorter than one I2C kernel clock period updated workaround qualifier • CRC error in SPI slave mode if internal NSS changes before CRC transfer moved to documentation errata
18-Mar-2019	5	Added errata: • Memory-mapped write data to odd start address corrupted when BUSY=0 and not in octal-DTR mode • Data not sampled correctly on reads without DQS and with less than two cycles before the data phase • Alarm flag may be repeatedly set when the core is stopped in debug Modified errata: • CSBOUND shall not be used • Write operation in octal-DTR mode starting at odd address is not supported • Data masking for odd number of byte writes only working with D1/D0 ordering • START bit is cleared upon setting ADDRCF, not upon address match erratum reference moved from Table 3 to Table 4

Date	Version	Changes
24-Oct-2019	6	Added errata into Table 3. Summary of device limitations or Table 4. Summary of device documentation errata, and in Section 2 Description of device errata: • Bootloader not functional on 1-Mbyte devices • OBL_LAUNCH operation may corrupt SRAM content • Irrelevant BOOT0 electrical characteristics • 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 • DQS output enabled too late on write • PSRAM not supported • OVR flag not set in underrun condition • Transmission stalled after first byte transfer STM32L4Rxxx STM32L4Sxxx • ADEN bit cannot be set immediately after the ADC calibration is done • Invalid DAC channel analog output if the DAC channel MODE bitfield is programmed before DAC initialization • DMA underrun flag not set when an internal trigger is detected on the clock cycle of the DMA request acknowledge • MCU may remain stuck in LPTIM interrupt when clearing event flag Modified errata: • Flash memory might not be accessible when AHB prescaler is greater than two • 1-Mbyte devices wrongly configured as 2-Mbyte • Spurious Firewall reset • Writing ADC_JSQR when JADCSTART and JQDIS are set may lead to incorrect behavior Removed erratum *First double-word of Flash memory corrupted upon reset or power down while programming.*
02-Sept-20	7	Added errata into Table 3. Summary of device limitations or Table 4. Summary of device documentation errata, and in Section 2 Description of device errata, subsections: • FLASH_ECCR corrupted upon reset or power-down occurring during Flash memory program or erase operation • Unaligned write access to OCTOSPIM configuration registers failing • 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 • Wrong ADC differential conversion result for channel 5 • Tearing effect parasitic detection • Incorrect calculation of the time to activate the clock between HS transmissions • The immediate update procedure may fail • One-pulse mode trigger not detected in master-slave reset + trigger configuration • Consecutive compare event missed in specific conditions • Output compare clear not working with external counter reset • LPTIM events and PWM output are delayed by 1 kernel clock cycle • Anticipated end-of-transmission signaling in SPI slave mode • Data corruption due to noisy receive line • Last SAI_SCK clock pulse truncated upon disabling SAI master with NODIV set and MCKDIV greater than one • Last SAI_MCLK clock pulse truncated upon disabling SAI master

Date	Version	Changes
		• Superseded suspend sequence for data loaded by DMA • Superseded suspend sequence for data loaded by the CPU • Receiver timeout counter wrong start in two-stop-bit configuration Modified erratum Firewall protection 127,93 Kbytes size limitation. Removed erratum ADC - Injected queue of context is not available in case of JQM=0
10-Dec-2020	8	Added die revision *V*. Added errata: • Some I/O functions on PC13 pin do not work with RTC_OUT mapped on PB2 • Data corruption upon a specific FIFO write sequence to synchronous PSRAM • DSI-PHY may not start properly when the DSI PLL is restarted
15-Jun-2021	9	Added errata: • Selected external ADC inputs unduly clamped to VDD when all analog peripherals are disabled • TSC signal-to-noise concern under specific conditions • Flash memory ECC single-error correction prevents double-error detection from triggering NMI Modified errata: • Full JTAG configuration without NJTRST pin cannot be used: added “or for an alternate function other than NJTRST” • Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C: added “when wakeup from Stop mode is disabled in I2C” • Last-received byte loss in reload mode: added information for instances with independent clock Removed duplication of the erratum FLASH_ECCR corrupted upon reset or power-down occurring during Flash memory program or erase operation from Section 2 Description of device errata.
11-Mar-2022	10	Added errata: • Wrong instruction fetches from flash memory upon wakeup from Sleep or Stop mode when debug in low-power mode is enabled • Unsupported AHB burst byte write to PSRAM • Deadlock on memory-mapped write with timeout enabled • ADC_AWDy_OUT reset by non-guarded channels • Data corruption due to noisy receive line

Date	Version	Changes
		Modified errata: • CSBOUND must not be usedAutomatic status-polling mode not functional with octal memory devices • Write data lost with clock mode 3 • Deadlock in dual-quad configuration with Flash memories and odd number of bytes • Indirect read and automatic status-polling transfers without address phase not starting • Write operation in DTR octal-SPI mode starting at odd address is not supported • Memory-mapped read of the last memory space byte not possible in SDR octal-SPI mode • Single-byte memory-mapped write to odd DTR octal-SPI address failing when a higher-priority event occurs • Memory-mapped write data to odd start address corrupted when BUSY = 0 and not in DTR octal-SPI mode • Data not sampled correctly on reads without DQS and with less than two cycles before the data phase • Memory-mapped write error response when DQS output is disabled • Byte possibly dropped during an SDR read in clock mode 3 when a transfer gets automatically split • Single-, dual- and quad-SPI modes not functional with DQS input enabled • Additional bytes read in Indirect mode with DQS input enabled when data length is too short • Deadlock on memory-mapped write with timeout enabled • Device may remain stuck in LPTIM interrupt when clearing event flag • RTS is active while RE = 0 or UE = 0

Contents