STM32H742x/743xI/G, ST32H750xB and STM32H753xI device errata

Applicability

This document applies to the part numbers of STM32H742x/743xI/G, ST32H750xB and STM32H753xI 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 RM0433.

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
STM32H742xI/G	STM32H742VI, STM32H742ZI, STM32H742II, STM32H742BI, STM32H742XI, STM32H742AI, STM32H742VG, STM32H742ZG, STM32H742IG, STM32H742BG, STM32H742XG, STM32H742AG
STM32H743xI/G	STM32H743VI, STM32H743ZI, STM32H743II, STM32H743BI, STM32H743XI, STM32H743AI, STM32H743VG, STM32H743ZG, STM32H743IG, STM32H743BG, STM32H743XG, STM32H743AG
STM32H750xB	STM32H750VB, STM32H750IB, STM32H750XB, STM32H750ZB
STM32H753xI	STM32H753VI, STM32H753ZI, STM32H753II, STM32H753BI, STM32H753XI, STM32H753AI

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32H742xI/G, STM32H743xI/G, STM32H750xB, STM32H753xI	Y, W	0x1003
	X	0x2001
	V	0x2003

1. *Refer to the device datasheet for how to identify this code on different types of package.*
2. *REV_ID[15:0] bitfield of DBGMCU_IDC register.*

1 Summary of device errata

The following table gives a quick reference to the STM32H742x/743xI/G, ST32H750xB and STM32H753xI 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, W	Rev. X	Rev. V
Arm 32-bit Cortex-M7 core	2.1.1	Cortex-M7 data corruption when using Data cache configured in write-through	N	N	N
	2.1.2	Cortex®-M7 FPU interrupt not present on NVIC line 81	N	N	N
System	2.2.1	Timer system breaks do not work	N	-	-
	2.2.2	Clock recovery system synchronization with USB SOF does not work	A	-	-
	2.2.3	SysTick external clock is not HCLK/8	A	-	-
	2.2.4	Option byte loading can be done with the user wait-state configuration	A	-	-
	2.2.5	Flash memory BusFault address register may not be valid when an ECC double error occurs	A	-	-
	2.2.6	Flash ECC address register may not be updated	N	-	-
	2.2.7	PCROP-protected areas in flash memory may be unprotected	A	-	-
	2.2.8	Flash memory bank swapping may impact embedded flash memory interface behavior	N	-	-
	2.2.9	Reading from AXI SRAM may lead to data read corruption	A	-	-
	2.2.10	Clock switching does not work when an LSE failure is detected by CSS	A	-	-
	2.2.11	RTC stopped when a system reset occurs while the LSI is used as clock source	A	-	-
	2.2.12	USB OTG_FS PHY drive limited on DP/DM pins	N	-	-
	2.2.13	Unexpected leakage current on I/Os when V_{IN} higher that V_{DD}	A	-	-
	2.2.14	LSE oscillator driving capability selection bits are swapped	A	-	-
	2.2.15	HRTIM internal synchronization does not work	N	-	-
	2.2.16	Device stalled when two consecutive level regressions occur without accessing from/to backup SRAM	-	A	A
	2.2.17	Invalid flash memory CRC	P	-	-
	2.2.18	GPIO assigned to DAC cannot be used in output mode when the DAC output is connected to on-chip peripheral	N	N	N
	2.2.19	Unstable LSI when it clocks RTC or CSS on LSE	P	P	P
	2.2.20	480 MHz maximum CPU frequency not available	P	-	-
	2.2.21	VDDLDO is not available on TFBGA100 package	N	N	N

Function	Section	Limitation	Status		
			Rev. Y, W	Rev. X	Rev. V
System	2.2.22	WWDG not functional when V_{DD} is lower than 2.7 V and VOS0 or VOS1 voltage level is selected	N	N	N
	2.2.23	A tamper event does not erase the backup RAM when the backup RAM clock is disabled	A	A	A
	2.2.24	LSE CSS detection occurs even when the LSE CSS is disabled	P	P	P
	2.2.26	Ethernet MII mode is not available on packages with PC2_C/PC3_C pins	A	A	A
	2.2.27	HASH input data may be corrupted when DMA is used	A	A	A
	2.2.28	LSE crystal oscillator may be disturbed by transitions on PC13	N	N	N
MDMA	2.3.1	Non-flagged MDMA write attempts to reserved area	A	A	A
BDMA	2.4.1	BDMA 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
	2.6.4	Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event	A	A	A
DMA2D	2.7.1	DMA2D swap byte feature is not available	N	-	-
FMC	2.8.1	Dummy read cycles inserted when reading synchronous memories	N	N	N
	2.8.3	Wrong data read from a busy NAND memory	A	A	A
	2.8.4	Spurious clock stoppage with continuous clock feature enabled	A	-	-
	2.8.5	Unsupported read access with unaligned address	P	P	P
QUADSPI	2.9.1	First nibble of data not written after dummy phase	A	-	-
	2.9.2	QUADSPI cannot be used in indirect read mode when only data phase is activated	P	P	P
	2.9.3	QUADSPI hangs when QUADSPI_CCR is cleared	P	P	P
	2.9.4	QUADSPI internal timing criticality	A	A	A
	2.9.5	Memory-mapped read of last memory byte fails	P	P	P
ADC	2.10.2	New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0	A	A	A
	2.10.3	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
	2.10.4	Unexpected regular conversion when two consecutive injected conversions are performed in Dual interleaved mode	A	A	A
	2.10.5	ADC_AWDy_OUT reset by non-guarded channels	A	A	A
	2.10.6	Injected data stored in the wrong ADC_JDRx registers	A	A	A
	2.10.7	ADC slave data may be shifted in Dual regular simultaneous mode	A	A	A
	2.10.8	Conversion overlap might impact the ADC accuracy	A	-	-
	2.10.9	ADC3 resolution limited by LSE activity	A	-	-
	2.10.10	ADC maximum sampling rate when V_{DDA} is lower than 2 V	A	-	-
	2.10.11	ADC maximum resolution when V_{DDA} is higher than 3.3 V	A	-	-
	2.10.12	First ADC injected conversion in a sequence may be corrupted	A	-	-

Function	Section	Limitation	Status		
			Rev. Y, W	Rev. X	Rev. V
ADC	2.10.13	Conversion triggered by context queue register update	A	A	A
	2.10.14	Updated conversion sequence triggered by context queue update	A	A	A
DAC	2.11.1	Invalid DAC channel analog output if the DAC channel MODE bitfield is programmed before DAC initialization	-	-	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
VREFBUF	2.12.1	Overshoot on VREFBUF output	A	A	A
	2.12.2	VREFBUF Hold mode cannot be used	N	N	N
	2.12.3	VREFBUF trimming code not automatically initialized after reset	A	-	-
OPAMP	2.13.1	OPAMP high-speed mode must not be used	N	-	-
LTDC	2.14.1	Device stalled when accessing LTDC registers while pixel clock is disabled	A	A	A
TIM	2.17.1	One-pulse mode trigger not detected in master-slave reset + trigger configuration	P	P	P
	2.17.2	Consecutive compare event missed in specific conditions	N	N	N
	2.17.3	Output compare clear not working with external counter reset	P	P	P
LPTIM	2.18.1	Device may remain stuck in LPTIM interrupt when entering Stop mode	A	A	A
	2.18.2	Device may remain stuck in LPTIM interrupt when clearing event flag	P	P	P
	2.18.3	LPTIM events and PWM output are delayed by one kernel clock cycle	P	P	P
RTC	2.19.1	RTC calendar registers are not locked properly	A	-	-
	2.19.2	RTC interrupt can be masked by another RTC interrupt	A	A	A
	2.19.3	Calendar initialization may fail in case of consecutive INIT mode entry	A	A	A
	2.19.4	Alarm flag may be repeatedly set when the core is stopped in debug	N	N	N
	2.19.5	A tamper event fails to trigger timestamp or timestamp overflow events during a few cycles after clearing TSF	N	N	N
I2C	2.20.1	10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave	A	-	-
	2.20.3	Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period	P	P	P
	2.20.4	Spurious bus error detection in master mode	A	A	A
	2.20.5	Last-received byte loss in reload mode	P	-	-
	2.20.6	Spurious master transfer upon own slave address match	P	P	P
	2.20.8	OVR flag not set in underrun condition	N	N	N
	2.20.9	Transmission stalled after first byte transfer	A	A	A
USART	2.21.2	UDR flag set while the SPI slave transmitter is disabled	A	-	-
	2.21.3	Anticipated end-of-transmission signaling in SPI slave mode	A	A	A
	2.21.4	Data corruption due to noisy receive line	N	N	N
	2.21.6	DMA stream locked when transferring data to/from USART	-	A	A
LPUART	2.22.1	DMA stream locked when transferring data to/from LPUART	-	A	A
	2.22.2	Possible LPUART transmitter issue when using low BRR[15:0] value	P	P	P
SPI	2.23.1	Spurious DMA Rx transaction after simplex Tx traffic	A	-	-
	2.23.2	Master data transfer stall at system clock much faster than SCK	A	A	A

Function	Section	Limitation	Status		
			Rev. Y, W	Rev. X	Rev. V
SPI	2.23.3	Corrupted CRC return at non-zero UDRDET setting	P	P	P
	2.23.4	TXP interrupt occurring while SPI disabled	A	A	A
	2.23.5	Possible corruption of last-received data depending on CRCSIZE setting	A	A	A
SDMMC	2.24.1	Busy signal not detected at resume point when suspend command accepted by the card during busy phase	A	-	-
	2.24.2	Clock stop reported during read wait sequence	A	-	-
	2.24.3	Unwanted overrun detection when an AHB error is reported while all bytes have been received	A	-	-
	2.24.4	Consecutive multiple block transfers have to be separated by eight SDMMC bus clocks when started by setting the DTEN bit	A	-	-
	2.24.5	Data 2 line cannot be used to suspend double-data-rate transfers with read wait mode enabled	P	-	-
	2.24.6	End-of-buffer status flag not cleared when the last burst data is delayed by the slave	A	-	-
FDCAN	2.25.1	Writing FDCAN_TTTS during initialization corrupts FDCAN_TTTMC	A	-	-
	2.25.2	Wrong data may be read from message RAM by the CPU when using two FDCANs	N	-	-
	2.25.3	Desynchronization under specific condition with edge filtering enabled	A	A	A
	2.25.4	Tx FIFO messages inverted under specific buffer usage and priority setting	A	A	A
	2.25.5	DAR mode transmission failure due to lost arbitration	A	A	A
OTG_HS	2.26.2	Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_HS registers	A	-	-
	2.26.3	Host packet transmission may hang when connecting the full speed interface through a hub to a low-speed device	N	N	N
	2.26.1	Possible drift of USB PHY pull-up resistor	P	-	-
ETH	2.27.1	Tx DMA engine may halt when a Tx queue flush command is issued in OSP mode	A	-	-
	2.27.2	Rx DMA engine may not recover and restart an operation after getting a bus error response	A	-	-
	2.27.3	In OSP mode, Tx DMA engine may not correctly stop when instructed while the Ethernet is actively transferring data	A	-	-
	2.27.4	Giant packet error is incorrectly declared when a dribble error occurs for the Rx packet whose length is exactly equal to the giant packet limit	A	-	-
	2.27.5	Subsequent packets are not transmitted when the transmit queue is flushed during packet transmission	A	-	-
	2.27.6	Read-on-clear interrupt status not updated when an event occurs on the same clock cycle as a status register read	N	-	-
	2.27.7	Context descriptor contains incorrect receive timestamp status in Threshold mode when application clock is very fast	A	-	-
	2.27.8	Context descriptor is incorrectly written to descriptor memory when the Rx timestamp status is dropped due to unavailability of space in Rx queue	A	-	-
	2.27.9	MAC may indicate a power-down state even when disabled by software	A	-	-

Function	Section	Limitation	Status		
			Rev. Y, W	Rev. X	Rev. V
ETH	2.27.10	Incorrect Tx FIFO fill-level generation when MAC switches from Full-duplex to Half-duplex mode	A	-	-
	2.27.11	The MAC does not provide bus access to a higher priority request after a low priority request is serviced	N	N	N
	2.27.12	Rx DMA engine may fail to recover upon a restart following a bus error, with Rx timestamping enabled	A	A	A
	2.27.13	The Tx DMA engine fails to recover correctly or corrupts TSO/USO header data on receiving a bus error response from the AHB DMA slave	N	N	N
	2.27.14	Incorrectly weighted round robin arbitration between Tx and Rx DMA channels to access the common host bus	A	A	A
	2.27.15	Incorrect L4 inverse filtering results for corrupted packets	N	N	N
	2.27.16	IEEE 1588 Timestamp interrupt status bits are incorrectly cleared on write access to the CSR register with similar offset address	-	A	A
	2.27.17	Bus error along with Start-of-Packet can corrupt the ongoing transmission of MAC generated packets	N	N	N
	2.27.18	Spurious receive watchdog timeout interrupt	A	A	A
	2.27.19	Incorrect flexible PPS output interval under specific conditions	A	A	A
	2.27.20	Packets dropped in RMII 10Mbps mode due to fake dribble and CRC error	A	A	A
	2.27.21	ARP offload function not effective	A	A	A
	2.27.22	Tx DMA may halt while fetching TSO header under specific conditions	A	A	A
CEC	2.28.1	Transmission blocked when transmitted start bit is corrupted	P	-	-
	2.28.2	Missed CEC messages in normal receiving mode	A	-	-

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

Table 4. Summary of device documentation errata

Function	Section	Documentation erratum
System	2.2.25	Output current sunk or sourced by Pxy_C pins
BDMA	2.4.2	Byte and half-word accesses not supported
DMA	2.5.1	USART/UART/LPUART DMA transfer abort
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
FMC	2.8.2	Missing information on prohibited 0xFF value of NAND transaction wait timing
ADC	2.10.1	Writing ADC_JSQR when JADCSTART and JQDIS are set may lead to incorrect behavior
CRYP	2.15.1	Wrong endianness description
	2.15.2	CRYP_CSGCMCCMxR registers are missing
HASH	2.16.1	Superseded suspend sequence for data loaded by DMA
	2.16.2	Superseded suspend sequence for data loaded by the CPU
I2C	2.20.2	Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C
	2.20.7	START bit is cleared upon setting ADDRCF, not upon address match

Function	Section	Documentation erratum
USART	2.21.1	Receiver timeout counter wrong start in two-stop-bit configuration
	2.21.5	USART prescaler feature missing in USART implementation section

2 Description of device errata

The following sections describe the errata of the applicable devices with Arm® core and provide workarounds if available. They are grouped by device functions.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

2.1 Arm 32-bit Cortex-M7 core

Reference manual and errata notice for the Arm® Cortex®-M7 core revision r1p1 is available from http://infocenter.arm.com.

2.1.1 Cortex-M7 data corruption when using Data cache configured in write-through

Description

This limitation is registered under Arm® ID number 1259864 and classified into “Category A”.

If a particular sequence of stores and loads is performed to write-through memory, and some timing-based internal conditions are met, then a load might not get the last data stored to that address.

This erratum can only occur if the loads and stores are to write-through memory. This could be due to any of the following:

- The MPU has been programmed to set this address as write-through .
- The default memory map is being used and this address is write-through in that map.
- The memory is cacheable, and the CM7_CACR.FORCEWT bit is set.
- The memory is cacheable, shared, and the CM7_CACR.SIWT bit is set.

The following sequence is required for this erratum to occur:

1. The address of interest must be in the cache.
2. A write-through store to the same doubleword as the address of interest.
3. One of the following:
   - A linefill is started (to a different cacheline to the address of interest) that allocates to the same set as the address of interest.
   - An ECC error.
   - A cache maintenance operation without a following DSB.
4. A store to the address of interest.
5. A load from the address of interest.

If certain specific timing conditions are met, the load will get the data from the first store, or from what was in the cache at the start of the sequence instead of the data from the second store.

The effect of this erratum is that load operations can return incorrect data.

Workaround

There is no direct workaround for this erratum.

Where possible, Arm® recommends that you use the MPU to change the attributes on any write-through memory to write-back memory. If this is not possible, it might be necessary to disable the cache for sections of code that access write-through memory.

2.1.2 Cortex®-M7 FPU interrupt not present on NVIC line 81

Description

Arm® Cortex®-M7 FPU interrupt is not mapped on NVIC line 81.

Note: This limitation is due to an error of implementation of the Arm core on the die, as opposed to a limitation of the core itself.

Workaround

None.

2.2 System

2.2.1 Timer system breaks do not work

Description

The system break sources (processor LOCKUP output, PVD detection, RAMECC error, flash memory ECC error, and clock security system detection), do not generate a break event on TIM1, TIM8, and HRTIM.

Workaround

None.

2.2.2 Clock recovery system synchronization with USB SOF does not work

Description

The clock recovery system (CRS) synchronization by USB start-of-frame signal (SOF) does not work.

Workaround

When available, use the LSE oscillator as synchronization source.

2.2.3 SysTick external clock is not HCLK/8

Description

The SysTick external clock is the system clock, instead of the system clock divided by 8 (HCLK/8).

Workaround

Use the system clock (HCLK) as external clock, and multiply the reload value by 8 in the STK_LOAD register. Make sure that the maximum value is 2^{24} - 1.

2.2.4 Option byte loading can be done with the user wait-state configuration

Description

After an option byte change, the option byte loading is performed with the user wait-state configuration instead of the default configuration.

Workaround

When performing an option byte loading to modify the option bytes, configure the correct number of wait states, or use the default value (seven wait states).

2.2.5 Flash memory BusFault address register may not be valid when an ECC double error occurs

Description

When a first read operation is performed without ECC error, and a controller device accesses data with wait states, if a new access is done and contains an ECC double detection error, then the error message returns the address of the first data that has not generated the error.

Workaround

When a double ECC error flag is raised, check the failing address in the flash memory interface (FAIL_ECC_ADDR1/2 of the FLASH_ECC_FA1R/FA2R), and disregard the content of the BusFault address register.

2.2.6 Flash ECC address register may not be updated

Description

When two consecutive ECC errors occur, the content of the FLASH_ECC_FA1/2 register may not be updated if the error correction flag (SNECCERR1/2 or DBECCERR1/2 in FLASH_SR1/2 register) is cleared at the same time as a new ECC error occurs

Workaround

None.

2.2.7 PCROP-protected areas in flash memory may be unprotected

Description

In case of readout protection level regression from level 1 to level 0, the PCROP protected areas in flash memory may become unprotected.

Workaround

The user application must set the readout protection level to level 2 to avoid PCROP-protected areas from being unprotected.

2.2.8 Flash memory bank swapping may impact embedded flash memory interface behavior

Description

When the flash memory bank swapping feature is enabled, the embedded flash memory interface behavior may become unpredictable.

Workaround

Do not enable the flash memory bank swapping feature.

2.2.9 Reading from AXI SRAM may lead to data read corruption

Description

The read data may be corrupted when the following conditions are met:

- Several read transactions are performed to the AXI SRAM, and
- A controller device delays its data acceptance while a new transfer is requested.

Workaround

Set the READ_ISS_OVERRIDE bit in the AXI_TARG7_FN_MOD register. This reduces the read issuing capability to 1 at AXI interconnect level and avoids data corruption.

2.2.10 Clock switching does not work when an LSE failure is detected by CSS

Description

When a failure on the LSE oscillator is detected by the clock security system (CSS), the backup domain clock source cannot be changed.

Workaround

When the clock security system detects an LSE failure, reset the backup domain, and select a functional clock source.

2.2.11 RTC stopped when a system reset occurs while the LSI is used as clock source

Description

When the LSI clock is used as RTC clock source, the RTC is stopped (it does not receive the clock anymore) when a system reset occurs.

Workaround

Apply one of the following measures:

- Check the RTC clock source after each system reset.
- If the LSI clock is selected, enable it again.

2.2.12 USB OTG_FS PHY drive limited on DP/DM pins

Description

To avoid damaging parts, the user application must avoid to load more than 5 mA on OTG_FS_DP/DM pins.

Workaround

None.

2.2.13 Unexpected leakage current on I/Os when V_{IN} higher that V_{DD}

Description

When V_{IN} is higher than V_{DD}, and depending on the waveform applied to I/Os, an unexpected leakage current may be observed when V_{IN} decreases.

This leakage does not impact the product reliability.

Workaround

The application must maintain V_{IN} lower that V_{DD} to avoid current leakage on I/Os.

2.2.14 LSE oscillator driving capability selection bits are swapped

Description

The LSEDRV[1:0] bits of the RCC_BDCR register, which are used to select the LSE oscillator driving capability, are swapped (see Table 5).

Table 5. Expected vs effective LSE driving mode

LSEDRV[1:0]	LSE driving mode	
	Expected mode	Effective mode
01	Medium-low drive	Medium-high drive
10	Medium-high drive	Medium-low drive

Workaround

- Use LSEDRV[1:0] = 01 to select the LSE medium-high drive.
- Use LSEDRV[1:0] = 10 to select the LSE medium-low drive.

2.2.15 HRTIM internal synchronization does not work

Description

The HRTIM synchronization input source from an internal event (SYNCIN[1:0] = 10 of the HRTIM_MCR register) does not work. As a result, it is not possible to use the on-chip TIM1_TRGO output as synchronization event for HRTIM.

Workaround

None.

2.2.16 Device stalled when two consecutive level regressions occur without accessing from/to backup SRAM

Description

The device may be stalled when two consecutive level regressions (switching from RDP level 1 to RDP level 0) are performed without accessing (reading/writing) from/to the backup SRAM.

A power-on reset is required to recover from this failure.

Workaround

Perform a dummy access to backup SRAM before executing the level regression sequence (switching from RDP level 1 to RDP level 0).

2.2.17 Invalid flash memory CRC

Description

The CRC result may be corrupted when the flash memory CRC end-address register for bank 1 or 2 (FLASH_CRCEADD1/2R) targets the last address of sector 7.

Workaround

Do not use the flash memory CRC calculation feature.

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

Description

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

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

Workaround

None.

2.2.19 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.

2.2.20 480 MHz maximum CPU frequency not available

Description

VOS0 voltage scaling level is not available on silicon revision Y and W devices. This limits the maximum CPU frequency to 400 MHz.

Workaround

Use silicon revision V or X devices to reach a maximum frequency of 480 MHz.

2.2.21 VDDLDO is not available on TFBGA100 package

Description

The VDDLDO pin is not available on the TFBGA100 package. The F4 ball is internally connected to V_{DD}.

Workaround

None.

2.2.22 WWDG not functional when V_{DD} is lower than 2.7 V and VOS0 or VOS1 voltage level is selected

Description

The system window watchdog (WWDG) is not functional, that is, it does not generate a correct system reset and/or the WWDG reset flag is not asserted, when V_{DD} is lower than 2.7 V and VOS0 or VOS1 voltage level is selected. There is no dependency on V_{DDLDO}.

Workaround

None.

2.2.23 A tamper event does not erase the backup RAM when the backup RAM clock is disabled

Description

Upon a tamper event, the backup RAM is normally reset and its content erased. However, when the backup RAM clock is disabled (BKPRAMEN bit cleared in RCC_ AHB4ENR register), the backup RAM reset fails and the memory is not erased.

Workaround

Enable the backup RAM clock by setting BKPRAMEN bit in the RCC_AHB4 clock register (RCC_AHB4ENR). This can be done either during device initialization or during a tamper service routine.

2.2.24 LSE CSS detection occurs even when the LSE CSS is disabled

Description

The LSECSSD flag in RCC_BDCR register can be spuriously set in case of ESD stress when the device is in V_{BAT} mode, even if the CSS on LSE is disabled. The LSE clock is no longer propagated to the RTC nor the system as long as the LSECSSD flag is set. This is applicable even if the LSE oscillates. LSECSSD can be cleared only by a Backup domain reset.

During ST functional ESD tests, the failure was observed by stressing PC13, PC14, VBAT, PE5, and PE6. No failure is detected when both V_{DD} and V_{BAT} are present. The sensitivity observed on these five pins can be quantified through IEC1000-4-2 (ESD immunity) standard, with severity estimated between 1 (low immunity) and 2 (medium immunity), according to the same standard.

Workaround

To achieve good overall EMC robustness, follow the general EMC recommendations to increase equipment immunity (see *EMC design guide for STM8, STM32 and Legacy MCUs* application note (AN1709)). Robustness can be further improved for the impacted pins other than VBAT by inserting, where possible, serial resistors with the value as high as possible not exceeding 1 kΩ, as close as possible to the microcontroller.

2.2.25 Output current sunk or sourced by Pxy_C pins

Description

Some datasheets may not state that the current sunk or sourced by Pxy_C pins is limited to 1 mA when the analog switch between Pxy and Pxy_C pads is closed.

Workaround

No application workaround is required.

2.2.26 Ethernet MII mode is not available on packages with PC2_C/PC3_C pins

Description

The Ethernet MII mode is not available on packages where only the PC2_C/PC3_C pins are available.

Workaround

Instead, use the RMII Ethernet mode.

2.2.27 HASH input data may be corrupted when DMA is used

Description

When HASH uses DMA1 to transfer data, and DMA2 is used by the application to manage other peripherals, the HASH input data may get corrupted.

This issue also occurs when DMA2 is used for HASH data transfers, and DMA1 to manage other peripherals.

Workaround

Apply one of the following measures:

- During a HASH data transfer using DMA1, make sure that no DMA channels are enabled on DMA2 (and vice versa).
- Use the interrupt or polling mode to transfer data to the HASH peripheral when DMA1 or DMA2 channels cannot be disabled during the HASH transfer.

2.2.28 LSE crystal oscillator may be disturbed by transitions on PC13

Description

The LSE crystal oscillator clock frequency can be incorrect when PC13 is toggling in input or output (for example when used for RTC_OUT1).

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

Workaround

None.

Avoid toggling PC13 when LSE is used.

2.3 MDMA

2.3.1 Non-flagged MDMA write attempts to reserved area

Description

The 0x0000 0000 - 0x0003 FFFF address space is linked to ITCM. The TCM_AXI_SHARED[1:0] bitfield of the FLASH_OPTSR2_CUR option byte defines areas of this address space valid for access and reserved areas. MDMA write access (through the CPU AHBS) to an address in any reserved area is expected to signal bus error exception.

However, with TCM_AXI_SHARED[1:0] bitfield set to 10, although MDMA write attempts to addresses in the 0x0003 0000 - 0x0003 FFFF reserved area are duly ignored (no data write effected), they do not signal bus error exception (no flag is raised), which corresponds to MDMA wrongly reporting *write completed*.

Workaround

Avoid accessing reserved areas.

2.4 BDMA

2.4.1 BDMA disable failure and error flag omission upon simultaneous transfer error and global flag clear

Description

Upon a data transfer error in a BDMA 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 BDMA_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.

2.4.2 Byte and half-word accesses not supported

Description

Some reference manual revisions may wrongly state that the BDMA registers are byte- and half-word-accessible. Instead, the BDMA registers must always be accessed through aligned 32-bit words. Byte or half-word write accesses cause an erroneous behavior.

ST's low-level driver and HAL software only use aligned 32-bit accesses to the BDMA registers.

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

Workaround

No application workaround is required.

2.5 DMA

2.5.1 USART/UART/LPUART DMA transfer abort

Description

Some reference manual revisions may unduly present the bit 20 (TRBUFF in the corrected revisions) of the DMA_SxCR register as reserved, to be kept at reset value (low). This bit must be set to ensure the completion of USART/UART/LPUART DMA transfer when another DMA transfer is requested concurrently. Otherwise, it may occur that the other DMA transfer request is not served and that it leads to aborting the ongoing USART/UART/LPUART DMA transfer.

This is a documentation issue rather than a device limitation.

Workaround

No application workaround is required if the TRBUFF bit is used as indicated.

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 DMA2D

2.7.1 DMA2D swap byte feature is not available

Description

The SB bit of the register DMA2D_OPFCCR register is not implemented. As a result, the swap byte feature is not available.

Workaround

None.

2.8 FMC

2.8.1 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.

2.8.2 Missing information on prohibited 0xFF value of NAND transaction wait timing

Description

Some reference manual revisions may omit the information that the value 0xFF is prohibited for the wait timing of NAND transactions in their corresponding memory space (common or attribute).

Whatever the setting of the PWAITEN bit of the FMC_PCRx register, the wait timing set to 0xFF would cause a NAND transaction to stall the system with no fault generated.

This is a documentation error rather than a device limitation.

Workaround

No application workaround required provided that the 0xFF wait timing value is duly avoided.

2.8.3 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.

2.8.4 Spurious clock stoppage with continuous clock feature enabled

Description

With the continuous clock feature enabled, the FMC_CLK clock may spuriously stop when:

- the FMC_CLK clock is divided by 2, and
- an FMC bank set as 32-bit is accessed with a byte access.

division ratio set to 2, the FMC_CLK clock may spuriously stop upon an

Note: With static memories, a spuriously stopped clock can be restarted by issuing a synchronous transaction or any asynchronous transaction different from a byte access on 32-bit data bus width.

Workaround

With the continuous clock feature enabled, do not set the FMC_CLK clock division ratio to 2 when accessing 32-bit asynchronous memories with byte access.

2.8.5 Unsupported read access with unaligned address

Description

Read access with unaligned address, such as a half-word read access starting at odd address, is not supported.

Workaround

Compile the software that accesses the fmc region with a compiler option that ensures data alignment, such as –*no_unaligned_access*.

2.9 QUADSPI

2.9.1 First nibble of data not written after dummy phase

Description

The first nibble of data to be written to the external flash memory is lost when the following condition is met:

- QUADSPI is used in indirect write mode.
- At least one dummy cycle is used.

Workaround

Use alternate bytes instead of dummy phase to add latency between the address phase and the data phase. This works only if the number of dummy cycles to substitute corresponds to a multiple of eight bits of data.

Example:

- To substitute one dummy cycle, send one alternate byte (only possible in DDR mode with four data lines).
- To substitute two dummy cycles, send one alternate byte in SDR mode with four data lines.
- To substitute four dummy cycles, send two alternate bytes in SDR mode with four data lines, or one alternate byte in SDR mode with two data lines.
- To substitute eight dummy cycles, send one alternate byte in SDR mode with one data line.

2.9.2 QUADSPI cannot be used in indirect read mode when only data phase is activated

Description

When the QUADSPI peripheral is configured in indirect read with only the data phase activated (in single, dual, or quad I/O mode), the QUADSPI peripheral hangs and the BUSY flag of the QUADSPI_SR register remains high. An abort must be performed to reset the BUSY flag and exit the hanging state.

Workaround

Insert a dummy phase with at least two dummy cycles.

2.9.3 QUADSPI hangs when QUADSPI_CCR is cleared

Description

Writing 0x0000 0000 to the QUADSPI_CCR register causes the QUADSPI peripheral to hang while the BUSY flag of the QUADSPI_SR register remains set. Even an abort does not allow exiting this status.

Workaround

Clear then set the EN bit of the QUADSPI_CR register.

2.9.4 QUADSPI internal timing criticality

Description

The timing of some internal signals of the QUADSPI peripheral is critical. At certain conditions, this can lead to a general failure of the peripheral. As these conditions cannot be exactly determined, it is recommended to systematically apply the workaround as described.

Workaround

The code below have to be executed upon reset and upon switching from memory-mapped to any other mode:

```
// Save QSPI_CR and QSPI_CCR values if necessary
QSPI->QSPI_CR = 0; // ensure that prescaling factor is not at maximum, and  disable the perip
heral
while(QSPI->QSPI_SR & 0x20){}; // wait for BUSY flag to fall if not already low
QSPI->QSPI_CR = 0xFF000001; // set maximum prescaling factor, and  enable the peripheral
QSPI->QSPI_CCR = 0x20000000; // activate the free-running clock
QSPI->QSPI_CCR = 0x20000000; // repeat the previous instruction to prevent a back-to-back dis
able

// The following command must complete less than 127 kernel clocks after the first write to t
he QSPI_CCR register
QSPI->QSPI_CR = 0; // disable QSPI
while(QSPI->QSPI_SR & 0x20){}; // wait for busy to fall

// Restore CR and CCR values if necessary
```

For the worakround to be effective, it is important to complete the disable instruction less than 127 kernel clock pulses after the first write to the QSPI_CCR register.

2.9.5 Memory-mapped read of last memory byte fails

Description

Regardless of the number of I/O lines used (1, 2 or 4), a memory-mapped read of the last byte of the memory region defined through the FSIZE[4:0] bitfield of the QUADSPI_DCR register always yields 0x00, whatever the memory byte content is. A repeated attempt to read that last byte causes the AXI bus to stall.

Workaround

Apply one of the following measures:

- Avoid reading the last byte of the memory region defined through FSIZE, for example by taking margin in FSIZE bitfield setting.
- If the last byte is read, ignore its value and abort the ongoing process so as to prevent the AXI bus from stalling.
- For reading the last byte of the memory region defined through FSIZE, use indirect read.

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 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.

2.10.3 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.

2.10.4 Unexpected regular conversion when two consecutive injected conversions are performed in Dual interleaved mode

Description

In Dual ADC mode, an unexpected regular conversion may start at the end of the second injected conversion without a regular trigger being received, if the second injected conversion starts exactly at the same time than the end of the first injected conversion. This issue may happen in the following conditions:

- two consecutive injected conversions performed in Interleaved simultaneous mode (DUAL[4:0] of ADC_CCR = 0b00011), or
- two consecutive injected conversions from master or slave ADC performed in Interleaved mode (DUAL[4:0]of ADC_CCR = 0b00111)

Workaround

- In Interleaved simultaneous injected mode: make sure the time between two injected conversion triggers is longer than the injected conversion time.
- In Interleaved only mode: perform injected conversions from one single ADC (master or slave), making sure the time between two injected triggers is longer than the injected conversion time.

2.10.5 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 Injected data stored in the wrong ADC_JDRx registers

Description

When the AHB clock frequency is higher than the ADC clock frequency after the prescaler is applied (ratio > 10), if a JADSTP command is issued to stop the injected conversion (JADSTP bit set to 1 in ADC_CR register) at the end of an injected conversion, exactly when the data are available, then the injected data are stored in ADC_JDR1 register instead of ADC_JDR2/3/4 registers.

Workaround

Before setting JADSTP bit, check that the JEOS flag is set in ADC_ISR register (end of injected channel sequence).

2.10.7 ADC slave data may be shifted in Dual regular simultaneous mode

Description

In Dual regular simultaneous mode, ADC slave data may be shifted when all the following conditions are met:

- A read operation is performed by one DMA channel,
- OVRMOD = 0 in ADC_CFGR register (Overrrun mode enabled).

Workaround

Apply one of the following measures:

- Set OVRMOD = 1 in ADC_CFGR. This disables ADC_DR register FIFO.
- Use two DMA channels to read data: one for slave and one for master.

2.10.8 Conversion overlap might impact the ADC accuracy

Description

The following conditions might impact the ADC accuracy

- Several ADC conversions are running simultaneously
- ADC and DAC conversions are running simultaneously

Workaround

Avoid conversion overlapping. The application should ensure that conversions are performed sequentially.

2.10.9 ADC3 resolution limited by LSE activity

Description

The following ADC3 input pins may be impacted by adjacent LSE activity:

- ADC3 channels on pins PF3 to PF10.

Workaround

Avoid using 14-bit and 16-bit data resolutions on these pins. This limits data resolution configuration to 8 bits, 10 bits, or 12 bits.

2.10.10 ADC maximum sampling rate when V_{DDA} is lower than 2 V

Description

If V_{DDA} is lower than 2 V, the ADC conversion accuracy is not guaranteed over the full ADC sampling rate.

Workaround

The application must avoid a sampling rate higher than 1.5 MSPS when operating with V_{DDA} below 2 V.

2.10.11 ADC maximum resolution when V_{DDA} is higher than 3.3 V

Description

If V_{DDA} is higher than 3.3 V, the ADC conversion accuracy is not guaranteed for all data resolutions.

Workaround

12-bit, 14-bit, and 16-bit data resolutions are not useful in this configuration. This limits the available data resolution configurations to 8 bits and 10 bits.

2.10.12 First ADC injected conversion in a sequence may be corrupted

Description

The ADC injected conversion that follows a regular conversion may be corrupted if all the following conditions are met:

- A regular conversion successive approximation is ongoing (sampling phase finished), and
- an injected conversion sequence is triggered during the regular conversion successive approximation phase.

In this case, the first injected conversion returns an invalid result. Other conversions are not impacted.

Workaround

Apply one of the following measures:

- Use a sequence of at least two injected conversions, ignore the first injected value and consider the other ones.
- Synchronize regular and injected conversion to prevent regular channels and injected channels from overlapping.

2.10.13 Conversion triggered by context queue register update

Description

Modifying the trigger selection or the edge polarity detection may trigger the conversion of the new context without waiting for the trigger edge when all the following conditions are met:

- The injected queue conversion is enabled (JQDIS = 0 in the ADC_CGFR register), and
- the queue is never empty (JQM 0 in the ADC_CGFR register).

Workaround

Apply one of the following measures:

- Ignore the first converted sequence.
- Use the queue of context with JQM = 1 in ADC_CGFR.
- Use the queue of context with JQM = 0 in the ADC_CGFR, and change the sequence without modifying the trigger and the polarity.

2.10.14 Updated conversion sequence triggered by context queue update

Description

Modifying the context queue for injected conversions may trigger the conversion of the new context without waiting for the trigger edge when all the following conditions are met:

- The injected queue conversion is enabled (JQDIS = 0 in the ADC_CGFR register), and
- the queue is not empty (JQM = 0 in the ADC_CGFR register), and
- the context queue is programmed five cycles before the JEOS flag is set in the ADC_ISR register.

Workaround

Apply one of the following measures:

- Use the queue of context with JQM = 1 in the ADC_CGFR register.
- Synchronize the programming of the new context with the next trigger edge to make sure it is performed after the JEOS flag is set in the ADC_ISR register (for example at the trigger rising edge).

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 VREFBUF

2.12.1 Overshoot on VREFBUF output

Description

An overshoot might occur on VREFBUF output if VREF+ pin has residual voltage when VREFBUF is enabled (ENVR is set in VREFBUF_CSR register).

Workaround

Let the voltage on the VREF+ pin drop to 1 V under the target V_{REFBUF_OUT}. This can be achieved by switching VREFBUF buffer off (ENVR is cleared and HIZ is cleared in VREFBUF_CSR register) allowing sufficient time to discharge the capacitor on the VREF+ pin through the VREFBUF pull-down resistor.

2.12.2 VREFBUF Hold mode cannot be used

Description

VREFBUF can be configured to operate in Hold mode to reduce current consumption.

When VREFBUF enters Hold mode (by setting both HIZ and ENVR bits of the VREFBUF_CSR register), the VREF+ I/O transits to high impedance mode. If not discharged externally, the capacitor on the VREF+ pin keeps its charge and voltage. Exiting VREFBUF Hold mode (by clearing the HIZ bit) in this condition might lead to a voltage overshoot on the VREF+ output.

Workaround

None.

2.12.3 VREFBUF trimming code not automatically initialized after reset

Description

After reset, the VREFBUF trimming code defined by the TRIM[5:0] bitfield of the VREFBUF calibration control register (VREFBUF_CCR) is not automatically initialized with the trimming value stored in the device during production test.

Workaround

Follow the steps below to program the trimming code by software:

1. Enable the SYSCFG peripheral clock by setting the SYSCFGEN bit of the RCC_APB4ENR register.
2. Enable the VREFBUF peripheral clock by setting the VREFEN bit of the RCC_APB4ENR register.
3. Program the VREFBUF trimming code (TRIM[5:0] of VREFBUF_CCR) with the value stored in bits [5:0] at address 0x5800 0574.

2.13 OPAMP

2.13.1 OPAMP high-speed mode must not be used

Description

Signal limitations may be observed if the OPAMP high-speed mode is used (the OPAHSM bit is set in the OPAMPx_CSR register).

Workaround

None.

2.14 LTDC

2.14.1 Device stalled when accessing LTDC registers while pixel clock is disabled

Description

Workaround

Enable the pixel clock before accessing the LTDC registers. Apply the following sequence to enable the LTDC clock:

1. Enable pll3_r_ck to feed the LTDC pixel clock (ltdc_ker_ck).
2. Enable the LTDC register interface clock by setting the LTDCEN bit of the RCC_APB3ENR register.

2.15 CRYP

2.15.1 Wrong endianness description

Description

In some reference manuals, the information that is provided in the *Cryptographic processor (CRYP)* chapter concerning key endianness, IV endianness, and data endianness, is wrong.

This is a documentation issue rather than a product limitation.

The following sections provide the correct descriptions of key endianness, IV endianness, and data endianness.

Key endianness

The eight CRYP_KxR/L write-only registers store the encryption or decryption key information, as shown in Table 6 and Table 7. In DES/TDES mode, the CRYP_K0R/L registers are never used.

Note: In memory and in CRYP key registers, the AES and DES/TDES keys are stored in big-endian format, with the most significant byte at the lowest address.

Table 6. Key endianness in CRYP_KxR/LR registers (AES 128/192/256-bit keys)

K0LR[31:0]	K0RR[31:0]	K1LR[31:0]	K1RR[31:0]	K2LR[31:0]	K2RR[31:0]	K3LR[31:0]	K3RR[31:0]
-	-	-	-	k[127:96]	k[95:64]	k[63:32]	k[31:0]
K0LR[31:0]	**K0RR[31:0]**	**K1LR[31:0]**	**K1RR[31:0]**	**K2LR[31:0]**	**K2RR[31:0]**	**K3LR[31:0]**	**K3RR[31:0]**
-	-	k[191:160]	k[159:128]	k[127:96]	k[95:64]	k[63:32]	k[31:0]
K0LR[31:0]	**K0RR[31:0]**	**K1LR[31:0]**	**K1RR[31:0]**	**K2LR[31:0]**	**K2RR[31:0]**	**K3LR[31:0]**	**K3RR[31:0]**
k[255:224]	k[223:192]	k[191:160]	k[159:128]	k[127:96]	k[95:64]	k[63:32]	k[31:0]

Table 7. Key endianness in CRYP_KxR/LR registers (DES K1 and TDES K1/2/3)

K0LR[31:0]	K0RR[31:0]	K1LR[31:0]	K1RR[31:0]	K2LR[31:0]	K2RR[31:0]	K3LR[31:0]	K3RR[31:0]
-	-	K1[64:33]	K1[32:1]	-	-	-	-
-	-	K1[64:33]	K1[32:1]	K2[64:33]	K2[32:1]	K3[64:33]	K3[32:1]

IV endianness

The four CRYP_IVxR/L registers store the initialization vector (IVI) information, as shown in Table 8 and Table 9. In DES/TDES mode, only CRYP_IV0x are used.

Note: In memory and in CRYP IV registers, the AES and DES/TDES initialization vectors are stored in big-endian format, with the most significant byte at the lowest address.

Table 8. Initialization vector endianness in CRYP_IVxR registers (AES)

CRYP_IV0L[31:0]	CRYP_IV0R[31:0]	CRYP_IV1L[31:0]	CRYP_IV1R[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Table 9. Initialization vector endianness in CRYP_IVxR registers (DES/TDES)

CRYP_IV0L[31:0]	CRYP_IV0R[31:0]	CRYP_IV1L[31:0]	CRYP_IV1R[31:0]
IVI[63:32]	IVI[31:0]	-	-

In CTR chaining mode, the CRYP initialization vectors must be initialized as shown in Table 10.

Table 10. Counter mode initialization vector definition

CRYP_IV0LR[31:0]	CRYP_IV0RR[31:0]	CRYP_IV1LR[31:0]	CRYP_IV1RR[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x1

During the GCM chaining mode initialization phase, the first counter block (CB1) must be initialized as shown in Table 11.

Table 11. GCM mode IV registers initialization

CRYP_IV0LR[31:0]	CRYP_IV0RR[31:0]	CRYP_IV1LR[31:0]	CRYP_IV1RR[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x2

The last block of a GCM message must be written in the CRYP_DINR register as defined in Table 12.

Table 12. GCM last block definition

Word order to CRYP_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

During the CCM chaining mode initialization phase, the first block of a message (B0) must be prepared as defined in Table 13.

Table 13. CCM mode IVI registers initialization

CRYP_IV0LR[31:0]	CRYP_IV0RR[31:0]	CRYP_IV1LR[31:0]	CRYP_IV1RR[31:0]
B0[127:96](1)	B0[95:64]	B0[63:32]	B0[31:0](2)

1. *The five most significant bits are cleared (flag bits).*
2. *Q length bits are cleared, except for bit 0 that is set.*

Data endianness

The DES/TDES data endianness and data swapping feature is summarized in Table 14. Data is stored in system memory in **big-endian** format.

Table 14. DES/TDES data swapping example

DATATYPE[1:0] in CRYP_CR	Type of swapping performed	First half data block (64 bits)
		System memory data (big-endian)
00	No swapping	Block[64..1]: 0xABCD 7720 6973 FE01(1)
		Address @, word[63..32]: 0xABCD 7720 Address @+0x4, word[31..0]: 0x6973 FE01(2)
01	Half-word (16 bits) swapping	Block[64..1]: 0x**ABCD** 7720 **6973** FE01
		Address @, word[63..32]: 0x7720 **ABCD** Address @+0x4, word[31..0]: 0xFE01 **6973**
10	Byte (8 bits) swapping	Block[64..1]: 0x**AB**CD **77**20 **69**73 **FE**01
		Address @, word[63..32]: 0x2077 CD**AB** Address @+0x4, word[31..0]: 0x01**FE** 73**69**

11	Bit swapping	Block[64..33]: 0xABCD 7720
		`1010 1011 1100 1101 0111 0111 0010 0000`
		Block[32..1]: 0x6973 FE01
		`0110 1001 0111 0011 1111 1110 0000 0001`

		Address @, word[63..32]: 0x04EE B3D5
		`0000 0100 1110 1110 1011 0011 1101 0101`
		Address @+4, word[31..0]: 0x807F CE96
		`1000 0000 0111 1111 1100 1110 1001 0110`

1. *The data block size is compliant with NIST standard recommendation.*
2. *The word size is compliant with NIST standard recommendation.*

The AES data endianness and data swapping feature is summarized in Table 15. Data is stored in system memory in **big-endian** format.

Table 15. AES data swapping example

DATATYPE[1:0] in CRYP_CR	Type of swapping performed	Data block
		System memory data (big-endian)
00	No swapping	Block[127..64]: 0x04EE F672 2E04 CE96 Block[63..0]: 0x4E6F 7720 6973 2074
		Address @, word[127..96]: 0x04EE F672 Address @ + 0x4, word[95..64]: 0x2E04 CE96 Address @ + 0x8, word[63..32]: 0x4E6F 7720 Address @ + 0xC, word[31..0]: 0x6973 2074
01	Half-word (16 bits) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
10	Byte (8 bits) swapping	Block[63..0]: 0x**4E**6F **77**20 **69**73 **20**74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**
11	Bit swapping	Block[63..32]: 0x4E6F 7720 `0100 1110 0110 1111 0111 0111 0010 0000`

DATATYPE[1:0] in CRYP_CR	Type of swapping performed	Data block
		System memory data (big-endian)
11	Bit swapping	Block[31..0]: 0x6973 2074 0110 1001 0111 0011 0010 0000 0111 0100
		Address @, word[63..32]: 0x04EE F672 0000 0100 1110 1110 1111 0110 0111 0010
		Address @ + 0x4, word[31..0]: 0x2E04 CE96 0010 1110 0000 0100 1100 1110 1001 0110

Workaround

No application workaround is required or applicable.

2.15.2 CRYP_CSGCMCCMxR registers are missing

Description

In some reference manuals, the *CRYP context swap GCM-CCM registers (CRYP_CSGCMCCMxR)* section is missing in the *cryptographic processor (CRYP)* chapter. These registers, described next, are required to run the suspend/resume operations for the GCM and CCM chaining modes.

CRYP context swap GCM-CCM registers (CRYP_CSGCMCCMxR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSGCMCCMx[31:0]																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Address offset 0x050 + x* 0x4 (x = 0 to 7)

Reset value 0x0000 0000

Description These registers contain the complete internal register states of the CRYP processor when the GCM/GMAC or CCM algorithm is selected. They are useful when a context swap has to be performed because a high-priority task needs the cryptographic processor while it is already in use by another task.

When such an event occurs, the CRYP_CSGCMCCM0..7R and CRYP_CSGCM0..7R (in GCM/GMAC mode) or CRYP_CSGCMCCM0..7R (in CCM mode) registers have to be read and the values retrieved have to be saved in the system memory space. The cryptographic processor can then be used by the preemptive task. Then when the cryptographic computation is complete, the saved context can be read from memory and written back into the corresponding context swap registers.

31:0	**CSGCMCCMx[31:0]:** CRYP processor internal register states for GCM, GMAC, and CCM modes. *Note: This register is not used in DES/TDES or other AES modes than the ones indicated.*

This is a documentation issue rather than a product limitation.

Workaround

No application workaround is required or applicable.

2.16 HASH

2.16.1 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.

2.16.2 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.

2.17 TIM

2.17.1 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.

2.17.2 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.

2.17.3 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).

2.18 LPTIM

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

Description

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

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

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

Workaround

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

2.18.2 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.

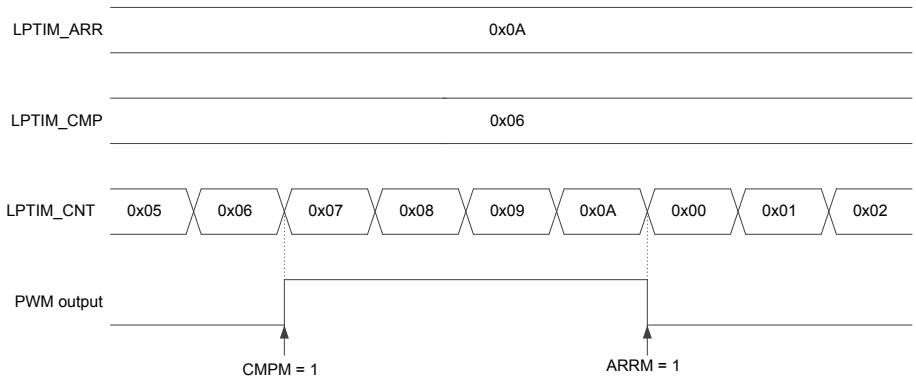
2.18.3 LPTIM events and PWM output are delayed by one kernel clock cycle

Description

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

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

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



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.

2.19 RTC

2.19.1 RTC calendar registers are not locked properly

Description

When reading the calendar registers with BYPSHAD = 0, the RTC_TR and RTC_DR registers may not be locked after reading the RTC_SSR register. This happens if the read operation is initiated one APB clock period before the shadow registers are updated. This can result in a non-consistency of the three registers. Similarly, the RTC_DR register can be updated after reading the RTC_TR register instead of being locked.

Workaround

Apply one of the following measures:

- Use BYPSHAD = 1 mode (bypass shadow registers), or
- If BYPSHAD = 0, read SSR again after reading SSR/TR/DR to confirm that SSR is still the same, otherwise read the values again.

2.19.2 RTC interrupt can be masked by another RTC interrupt

Description

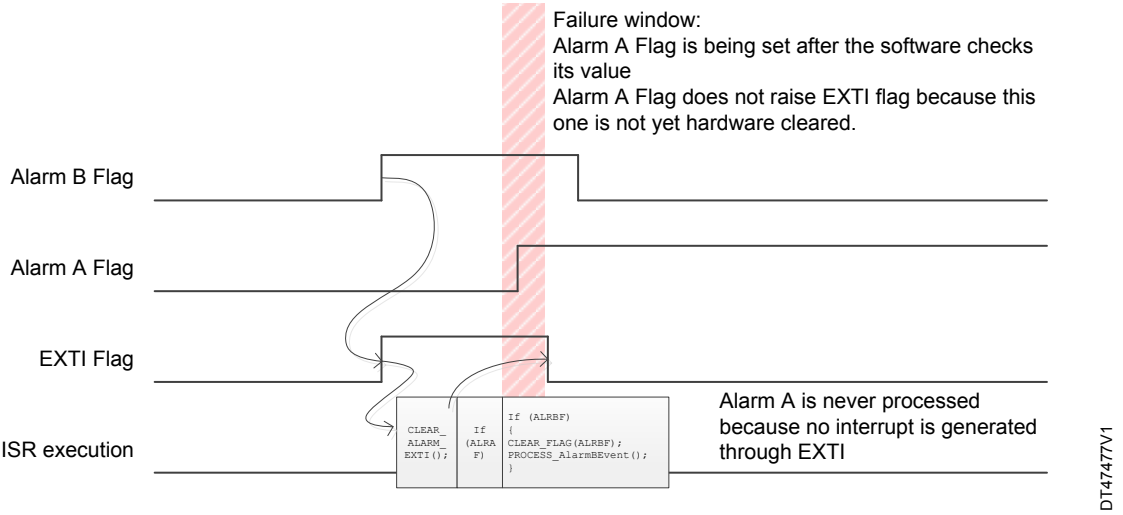
One RTC interrupt request can mask another RTC interrupt request if they share the same EXTI configurable line. For example, interrupt requests from Alarm A and Alarm B or those from tamper and timestamp events are OR-ed to the same EXTI line (refer to the *EXTI line connections* table in the *Extended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

2.19.3 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.

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

Description

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

As a consequence, when the subsecond counter is used for alarm condition (the MASKSS[3:0] bitfield of the RTC_ALRMASSR and/or RTC_ALRMBSSR register set to a non-zero value) and the alarm condition is met just before entering a breakpoint or printf, the ALRAF and/or ALRBF flag of the RTC_SR register is repeatedly set by hardware during the breakpoint or printf, which makes any attempt to clear the flag(s) ineffective.

Workaround

None.

2.19.5 A tamper event fails to trigger timestamp or timestamp overflow events during a few cycles after clearing TSF

Description

With the timestamp on tamper event enabled (TAMPTS bit of the RTC_CR register set), a tamper event is ignored if it occurs:

- within four APB clock cycles after setting the CTSF bit of the RTC_SCR register to clear the TSF flag, while the TSF flag is not yet effectively cleared (it fails to set the TSOVF flag)
- within two ck_apre cycles after setting the CTSF bit of the RTC_SCR register to clear the TSF flag, when the TSF flag is effectively cleared (it fails to set the TSF flag and timestamp the calendar registers)

Workaround

None.

2.20 I2C

2.20.1 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.

2.20.2 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.

2.20.3 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.

2.20.4 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.

2.20.5 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:

- I²C-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 I²C-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 (I²C-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.

2.20.6 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 I²C-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.

2.20.7 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.

2.20.8 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 I^2C bus, and the next data is not yet written in the TXDATA[7:0] bitfield. In this condition, the device is expected to set the OVR flag of the I2C_ISR register and send 0xFF on the bus.

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

Workaround

None.

2.20.9 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.

2.21 USART

2.21.1 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.

2.21.2 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.

2.21.3 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.

2.21.4 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.

2.21.5 USART prescaler feature missing in USART implementation section

Description

Some reference manual revisions may omit the information that the USART prescaler is not present in all USART instances. This information is provided in the USART implementation section of the corresponding reference manual.

This is a documentation issue rather than a product limitation.

Workaround

No application workaround is required or applicable.

2.21.6 DMA stream locked when transferring data to/from USART

Description

When a USART is issuing a DMA request to transfer data, if a concurrent transfer occurs, the requested transfer may not be served and the DMA stream may stay locked.

Workaround

Use the alternative peripheral DMA channel protocol by setting bit 20 of the DMA_SxCR register.

This bit is reserved in the documentation and must be used only on the stream that manages data transfers for USART peripherals.

2.22 LPUART

2.22.1 DMA stream locked when transferring data to/from LPUART

Description

When an LPUART is issuing a DMA request to transfer data, if a concurrent transfer occurs, the requested transfer may not be served and the DMA stream may stay locked.

Workaround

Use the alternative peripheral DMA channel protocol by setting bit 20 of the DMA_SxCR register.

This bit is reserved in the documentation and must be used only on the stream that manages data transfers for LPUART peripherals.

2.22.2 Possible LPUART transmitter issue when using low BRR[15:0] value

Description

The LPUART transmitter bit length sequence is not reset between consecutive bytes, which could result in a jitter that cannot be handled by the receiver device. As a result, depending on the receiver device bit sampling sequence, a desynchronization between the LPUART transmitter and the receiver device may occur resulting in data corruption on the receiver side.

This happens when the ratio between the LPUART kernel clock and the baud rate programmed in the LPUART_BRR register (BRR[15:0]) is not an integer, and is in the three to four range. A typical example is when the 32.768 kHz clock is used as kernel clock and the baud rate is equal to 9600 baud, resulting in a ratio of 3.41.

Workaround

Apply one of the following measures:

- On the transmitter side, increase the ratio between the LPUART kernel clock and the baud rate. To do so:
  - Increase the LPUART kernel clock frequency, or
  - Decrease the baud rate.
- On the receiver side, generate the baud rate by using a higher frequency and applying oversampling techniques if supported.

2.23 SPI

2.23.1 Spurious DMA Rx transaction after simplex Tx traffic

Description

With empty RXFIFO, SPI can spuriously generate a DMA read request upon enabling DMA receive traffic (by setting RXDMAEN bit), provided that the preceding completed transaction is a simplex transmission.

Workaround

Before enabling DMA Rx transfer following a completed Tx simplex transfer, perform hardware reset of the SPI peripheral.

2.23.2 Master data transfer stall at system clock much faster than SCK

Description

With the system clock (spi_pclk) substantially faster than SCK (spi_ker_ck divided by a prescaler), SPI master data transfer can stall upon setting the CSTART bit within one SCK cycle after the EOT event (EOT flag raise) signaling the end of the previous transfer.

Workaround

Apply one of the following measures:

- Disable then enable SPI after each EOT event.
- Upon EOT event, wait for at least one SCK cycle before setting CSTART.
- Prevent EOT events from occurring, by setting transfer size to undefined (TSIZE = 0) and by triggering transmission exclusively by TXFIFO writes.

2.23.3 Corrupted CRC return at non-zero UDRDET setting

Description

With non-zero setting of UDRDET[1:0] bitfield, the SPI slave can transmit the first bit of CRC pattern corrupted, coming wrongly from the UDRCFG register instead of SPI_TXCRC. All other CRC bits come from the SPI_TXCRC register, as expected.

Workaround

Keep TXFIFO non-empty at the end of transfer.

2.23.4 TXP interrupt occurring while SPI disabled

Description

SPI peripheral is set to its default state when disabled (SPE = 0). This flushes the FIFO buffers and resets their occupancy flags. TXP and TXC flags become set (the latter if the TSIZE field contains zero value), triggering interrupt if enabled with TXPIE or EOTIE bit, respectively. The resulting interrupt service can be spurious if it tries to write data into TXFIFO to clear the TXP and TXC flags, while both FIFO buffers are inaccessible (as the peripheral is disabled).

Workaround

Keep TXP and TXC (the latter if the TSIZE field contains zero value) interrupt disabled whenever the SPI peripheral is disabled.

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

Description

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

Workaround

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

2.24 SDMMC

2.24.1 Busy signal not detected at resume point when suspend command accepted by the card during busy phase

Description

When a card accepts a suspend command during the block busy phase of a write transfer, the card may drive the data line 0 when the write transfer is resumed. However, the SDMMC does not detect that the data line 0 is low when the write transfer resumes, and the resumed transfer may fail.

This issue can be observed when the following sequence is executed:

1. An SDIO block write transfer is ongoing (DTMODE[1:0] = 0b00 or 0b11, and DTDIR = 0 in the SDMMC_DCTRL register).
2. The card accepts the suspend request during the block busy phase (CMDREND = 1 in the SDMMC_STAR register, and bit 0 of the SDMMC_RESP1R = 1).
3. A resume is requested by the SDMMC (CMDSUSPEND = 1, CMDTRANS = 1, and CPSMEN = 1).
4. The card drives the data line 0 low.

Workaround

Apply the following sequence to suspend a write transfer:

1. Set the DTHOLD bit in the SDMMC_CMDR register.
2. Wait until the DHOLD status flag is set in the SDMMC_STAR register, to make sure the Busy line has been released.
3. Send a suspend command (CMDSUSPEND = 1, CMDTRANS = 0, CPSMEN = 1)

2.24.2 Clock stop reported during read wait sequence

Description

During a voltage switch sequence, the host has to stop the clock at low level, and inform the software that the clock is stopped by setting the CKSTOP status bit of the SDMMC_STAR register.

During a read wait sequence, the clock is stopped at high level for standard cards, and at low level for high-frequency cards. In the latter case, the CKSTOP status bit is set while it should not. If it is not cleared, the next voltage switch sequence may fail.

This issue can be observed when the following sequence is executed:

1. A multiple block read is ongoing (DTMODE[1:0] = 0b00 or 0b11, and DTDIR = 1 in the SDMMC_DCTRL register).
2. The read wait mode using clock stop is enabled (RWSTART = 1, RWMOD = 1, and RWSTOP = 0 in the SDMMC_DCTRLR register).
3. A high-frequency card is selected (BUSSPEED = 1 in the SDMMC_CLKCR register).

Workaround

When the multiple block transfer ends (DATAEND = 1 in the SDMMC_STAR register), set both the CKSTOPC and the DATAENDC bits in the SDMMC_ICR register.

2.24.3 Unwanted overrun detection when an AHB error is reported while all bytes have been received

Description

When the internal DMA is used, a write initiated by the SDMMC on the AHB master bus may fail because of software wrong access configuration. In this case, the IDMATE flag of the SDMMC_STAR register is set, and the transfer is aborted by flushing the FIFO.

When a read transfer ends successfully (that is, all bytes have been transferred on the external SDMMC bus), the DPSM waits until the FIFO is empty before setting the DATAEND flag in the SDMMC_STAR register. When an AHB error occurs, the FIFO is considered empty, but some bytes may still be present in the internal receive buffer (not yet transferred to the FIFO).

For this reason, the Rx buffer must be reset when the DPSM returns to idle state, which is always the case when the AHB transfers correctly end, but not when an AHB error occurs. In the latter case, the next read operation fails and an overrun is reported.

This issue can be observed when all the following conditions are met:

- A read transfer completes successfully (DTDIR = 1 in the SDMMC_DCTRLR register and DATAEND = 1 in the SDMMC_STAR register).
- An AHB error is reported by the slave on the three last bursts of the transfer (IDMATE = 1).

Workaround

When the DATAEND flag is set, check IDMATE. If both IDMATE and DTDIR bits are set, reset the SDMMC.

2.24.4 Consecutive multiple block transfers have to be separated by eight SDMMC bus clocks when started by setting the DTEN bit

Description

When a transfer ends, the MMC and SD standards request at least eight SDMMC bus clock cycles free before starting a new transfer. However, when enabling a new transfer by setting the DTEN bit of the SDMMC_DCTRLR register, a new transfer can start whereas less than eight SDMMC clock cycles have elapsed since the end of the previous transfer. For this reason, a hardware protection is implemented to guarantee eight SDMMC bus clock cycles free between the two transfers.

When the hardware delays the new transfer because DTEN was set less than eight cycles after the last transfer, the number of data to transfer is not reloaded in the internal data block counter whereas it should be. As a result, the second transfer is performed with the number of blocks configured for the previous transfer.

This issue can be observed when all the following conditions are met:

- Consecutive multiple-block transfers are done with different number of blocks to transfer.
- The second block transfer starts by setting the DTEN bit of the SDMMC_DCTRLR register.
- The second transfer is requested less than eight SDMMC bus clock cycles after the end of the previous one.

Workaround

Before starting a new transfer, make sure that at least eight SDMMC bus clock cycles have elapsed between the reported successful end-of-transfer and the setting of DTEN.

2.24.5 Data 2 line cannot be used to suspend double-data-rate transfers with read wait mode enabled

Description

The read wait mode allows an SDIO multiple block read to be held when the host is not ready to receive the next bytes. When the read wait mode is enabled, the host can request a card to suspend the transfer, either by setting the data line 2 low or by stretching the clock between two blocks.

However, when the device uses the data line 2 to suspend a double-data-rate multiple-block read transfer, the data line 2 oscillates instead of being steadily low. As a consequence, the card may keep sending data bytes, whereas the device is not ready to receive them. This then causes CRC errors.

This issue can be observed when all the following conditions are met:

- An SDIO double-data-rate multiple-block read is ongoing (DTMODE[1:0] = 0b00 or 0b11 in the SDMMC_DCTRL register, DTDIR = 1 in the SDMMC_DCTRL register, and DDR = 1 in the SDMMC_CLKCR register).
- A read wait mode using data line 2 is configured (RWSTART = 1 and RWMOD = 0 in the SDMMC_DCTRL register).

Workaround

Apply the following measures:

- Use the clock stretching method (RWMOD = 1) instead of the data line 2, and
- make sure no command is sent between two block reads.

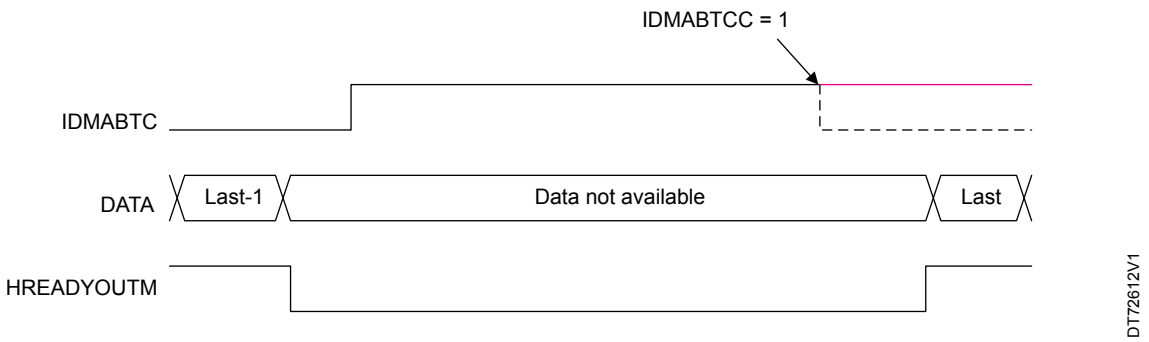
2.24.6 End-of-buffer status flag not cleared when the last burst data is delayed by the slave

Description

The end-of-buffer status flag IDMABTC of the SDMMC_STAR register is set when the last 32-bit word of the last AHB burst of the current buffer is transferred between the SDMMC controller AHB interface and an internal RAM or an external memory (through a Quad/Octo-SPI or an FMC interface). This flag is cleared by setting the IDMABTCC bit of the SDMMC_ICR register, by software or by using the MDMA.

If the internal/external memory is not ready to transmit/receive the last word before the IDMABTCC is set, the IDMABTC flag is not cleared whereas it should be (see Figure 2). As a result, the software may consider it several times.

Figure 2. SDMMC AHB transfer vs IDMABTC flag status



This issue can be observed when all the following conditions are met:

- The IDMABTC status flag of the SDMMC_STAR register is used in the software (typically, to build the linked list descriptors in parallel to the current transfer).
- The last word of the buffer is not available, and causes AHB-not-ready signal activation (HREADYOUTM) for *n1* CPU cycles.
- The time between IDMABTC and IDMABTCC bit setting lasts *n2* CPU cycles, with *n2* < *n1*:
  - According to the system, the issue may not occur when an internal memory is addressed if the *n2* < *n1* condition can never be reached.
  - According to the external memory and the associated interface, the issue can certainly be reproduced under specific conditions.

Note: n2 is in the order of 30 to 40 cycles.

Workaround

Keep writing IDMABTCC until the IDMABTC flag is cleared.

2.25 FDCAN

2.25.1 Writing FDCAN_TTTS during initialization corrupts FDCAN_TTTMC

Description

During TTCAN initialization, writing the FDCAN TT trigger select register (FDCAN_TTTS) also affects the FDCAN TT trigger memory configuration register (FDCAN_TTTMC).

Workaround

Avoid writing the FDCAN_TTTS register during TTCAN initialization phase.

Note: Outside the TTCAN initialization phase, write operations to FDCAN_TTTS do not impact FDCAN_TTTMC since this register is write-protected.

2.25.2 Wrong data may be read from message RAM by the CPU when using two FDCANs

Description

When using two FDCAN controllers, and the CPU and FDCANs simultaneously request read accesses from message RAM, the CPU read may return erroneous data.

Note: CPU write accesses to message RAM operate correctly. CPU read accesses to message RAM operate correctly if a single FDCAN is used at a time.

Workaround

None

2.25.3 Desynchronization under specific condition with edge filtering enabled

Description

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

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

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

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

Workaround

Disable edge filtering or wait for frame retransmission.

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

Description

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

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

Workaround

Apply one of the following measures:

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

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

2.25.5 DAR mode transmission failure due to lost arbitration

Description

In DAR mode, the transmission may fail due to lost arbitration at the first two identifier bits.

Workaround

Upon failure, clear the corresponding Tx buffer transmission request bit TRPx of the FDCAN_TXBRP register and set the corresponding cancellation finished bit CFx of the FDCAN_TXBCF register, then restart the transmission.

2.26 OTG_HS

2.26.1 Possible drift of USB PHY pull-up resistor

Description

When $V_{DDUSB33}$ is present, and the USB idle resistor (pull-up 1 connected between VDDUSB and the ground) remains activated for a long period of time, the pull-up resistor value may drift, reaching the maximum value defined in the USB Specification.

This issue occurs only during device transmit and reset states in device mode. Other device states are not impacted.

The degradation can be observed after USB PHY pull-up continuous activation in the following conditions:

- 10 years of continuous transmit operations, or
- An 86 million reset operations (from the USB host) provided that the reset period is 2.5 ms (duration might vary depending on host reset duration).

The above results are based on design simulation.

Workaround

For USB OTG_HS1, use the external USB PHY instead of the internal PHY.

For USB OTG_HS2: no workaround is available.

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

Description

When the USB on-the-go high-speed peripheral is in Device mode, interrupting transmit FIFO write sequence with read or write accesses to OTG_HS 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_HS registers. Note that enabling DMA mode guarantees this.

2.26.3 Host packet transmission may hang when connecting the full speed interface through a hub to a low-speed device

Description

When the USB on-the-go high-speed peripheral is used with the full speed interface (DM and DP pins, N.B. not available on all devices), and connects to a low-speed device via a hub, the transmitter internal state machine may hang. This leads, after a timeout expiry, to a port disconnect interrupt.

Workaround

None. However, increasing the capacitance on the data lines may reduce the occurrence.

2.27 ETH

2.27.1 Tx DMA engine may halt when a Tx queue flush command is issued in OSP mode

Description

When operating in the Operate-on-second packet (OSP) mode, the Tx DMA engine fetches the next Tx packet from the host memory before writing the status of the packet currently transmitted from the MAC. The Tx DMA engine accepts the transmit status in the status FIFO when the MTL has completely accepted the next packet. When the application flushes the content of a Tx queue by setting bit 0 (FTQ) of the Tx queue operating mode register (ETH_MTLTXQOMR) while the Tx DMA engine is active, the MTL stops accepting any further packets until the flush command of the Tx queue is complete and all the available statuses in the status FIFO are read by the Tx DMA engine.

However, the flush command may not complete if the Tx queue flush command is given soon after the Tx DMA engine issues the request for fetching the descriptor of the next packet. This issue occurs only if the Tx queue flush command is issued before the Tx DMA engine completes the descriptor fetch and attempts to write the control word of the next packet to the Tx queue. In this scenario, the MTL does not accept the control word and the Tx DMA engine does not accept the transmit status of the previous packet. This results in a deadlock situation since the flush command is never complete as the status FIFO is not emptied, and no more transfers occur.

Workaround

Issue a Tx queue flush command only after checking that the Tx DMA engine is inactive (either in Stop or Suspend state) and is not fetching any descriptor.

This can be ensured either by stopping the Tx DMA engine (by clearing the ST bit of the channel transmit control register (ETH_DMACTXCR)) or by ensuring that Tx DMA engine has serviced all the available descriptors and is in Suspend state. The state of the DMA engine is reflected in the TPS0 bitfield of the debug status register (ETH_DMADSR).

2.27.2 Rx DMA engine may not recover and restart an operation after getting a bus error response

Description

When a bus error response is received while the Rx DMA engine is transferring a received packet or reading/ writing the descriptors, the Rx DMA engine enters the Stop state. In this state, if the RPF bit (Rx packet flush) in the channel receive control register (ETH_DMACRXCR) is set, the Rx DMA engine flushes and drops the subsequent packets intended to be serviced by this Rx DMA engine. When the application reconfigures and restarts the Rx DMA engine, the Rx DMA engine forwards the subsequent received packets to the application.

However, the internal transaction is not stopped correctly if a bus error response is observed when the Rx DMA engine is writing the last descriptor (final status) of a received packet.

Hence, when the application reconfigures and restarts the Rx DMA engine, it may not resume the normal packet transfer operation from the Rx queue to the host memory.

When the RPF bit is set, this issue is observed only if no subsequent packet is intended to be serviced by the same Rx DMA engine before the Rx DMA engine is restarted. If a subsequent packet is flushed before the Rx DMA engine restarts, the internal error transaction correctly completes and the Rx DMA engine resumes the packet transfer.

When the RPF bit is reset, this failure is always observed.

Workaround

When a fatal bus error interrupt is generated:

1. Read the REB bit of the channel status register (ETH_DMACSR) to check if the bus error occurred during the Rx descriptor write operation.
2. Issue a software reset by setting the SWR bit of the ETH_DMAMR DMA mode register.
3. Reconfigure and restart the Rx DMA engine.

Note: The software must reconfigure the whole Ethernet peripheral since setting the SWR bit resets the entire peripheral except for the standard bus interface modules.

2.27.3 In OSP mode, Tx DMA engine may not correctly stop when instructed while the Ethernet is actively transferring data

Description

In Operate-on-second packet (OSP) mode, the Tx DMA channel can fetch the next packet from host memory before it receives the status for the current packet transmission from the MAC. The software can stop the Tx DMA engine at any time by clearing the ST bit of the channel transmit control register (ETH_DMACTXCR). When this bit is cleared, the Tx DMA engine completes the transmission of the packet being transferred, closes the descriptors of all the transmitted packets, and enters the Stop state.

When the transmit interrupt enable (TIE) bit of the channel interrupt enable register (ETH_DMACIER) is set, the Tx DMA engine also generates the completion interrupt after the Tx status is written to the Tx descriptor in host memory.

However, when the stop command is issued in OSP mode while the Tx DMA engine is actively transferring data, the following issues may be observed:

- The Tx DMA engine may not generate the Tx packet completion interrupt. As a result, if the software relies on the completion interrupt to reuse the descriptors for subsequent packets, it may be delayed until the Tx DMA engine is restarted and the next packet is transferred.
- The Tx DMA engine may not even enter the Stop state and indefinitely wait for the Tx status from the MAC, even though all pending statutes are already written to the respective descriptors in host memory. To exit this state, the software must perform a soft reset, reconfigure, and restart the Tx DMA engine.
- The Tx DMA engine may unnecessarily transfer one additional Tx packet before entering the Stop state.

Workaround

Issue a stop command to the Tx DMA engine only after all Tx packets have been transferred, and the DMA engine is in Idle (or Suspend) state. To do this:

1. Stop updating the tail pointer so that no new packets are to be scheduled for transfer.
2. Wait for the Tx DMA engine to complete the transfer of all scheduled packets. To do this, monitor the respective TPS0 bit of the debug status register (ETH_DMADSR) until it is read as Suspend state.

2.27.4 Giant packet error is incorrectly declared when a dribble error occurs for the Rx packet whose length is exactly equal to the giant packet limit

Description

The MAC indicates a giant packet (GP) in the Rx status when the received packet length:

- exceeds 9018, 9022, or 9026 bytes for untagged, or single VLAN tagged, or double VLAN tagged packets, when the JE bit of the ETH_MACCR register is set
- exceeds 2000 bytes when the S2KP bit of the operating mode configuration register (ETH_MACCR) is set
- exceeds the number of bytes specified in the GPSL bitfield of the extended operating mode configuration register (ETH_MACECR), with GPSLCE bit set in the operating mode configuration register (ETH_MACCR)
- exceeds 1518, 1522, or 1526 bytes for untagged, or single VLAN tagged, or double VLAN tagged packets

However, if the received packet length has exactly the same number of bytes as the one specified above, and an additional dribble nibble is received on the MII interface, then the MAC incorrectly declares it as a giant packet (GP) in the Rx status, and incorrectly increments the alignment (dribble) error.

Workaround

When a dribble error is set, ignore the giant packet error if the PL field provided in the Rx status is equal to the programmed giant limit (not considering the CRC/tag stripping effects).

2.27.5 Subsequent packets are not transmitted when the transmit queue is flushed during packet transmission

Description

When the transmit queue is flushed, the MTL layer completes the ongoing packet transmission and waits for a status from the MAC layer. This status is provided to the application by updating the FF bit of the TDES0 transmit descriptor word0. The MTL layer provides a dummy status with the FF bit set for every packet flushed in the transmit queue. After the transmit queue flush operation is complete, the transmission restarts for packets received from the application.

However, when the flush occurs during the second word of the packet being provided to the MAC layer, the subsequent packets after the flush operation are not accepted for transmission by the MAC layer.

Workaround

Make sure the Tx path is inactive before issuing a Tx queue flush command. To do this, follow the following steps:

1. Make sure the Tx DMA engine is inactive (either in the Stop state or Suspend state), and not fetching any descriptor. To do this, stop the Tx DMA engine (by clearing the ST bit of the channel transmit control register (ETH_DMACTXCR)) or ensure that the Tx DMA engine has serviced all the available descriptors, and is in Suspend state. The state of the DMA engine is reflected in the TPS0 bitfield of the ETH_DMADSR register.
2. Make sure the Tx MTL is inactive. To do this, check that all the bits of the Tx queue debug register (ETH_MTLTXQDR) return 0.
3. Make sure the Tx MAC is inactive. To do this, check that all the bits of the MAC debug register (ETH_MACDR) return 0.

2.27.6 Read-on-clear interrupt status not updated when an event occurs on the same clock cycle as a status register read

Description

The read-on-clear interrupt status bits are cleared when the status register is read. The read operation returns the correct status value just before it is cleared due to the read operation.

However, the interrupt status bits are not updated for a new event when a new interrupt event occurs on the same cycle clock during which the clear pulse to the register is generated.

The following interrupt status bits of the interrupt status register (ETH_MACISR) are affected:

- RXSTSIS
- TXSTSIS
- TSIS
- LPIIS
- PMTIS
- PHYIS

Due to exceptions, RXSTIS and TXSTIS correspond to packet abort events in Rx path and Tx path, respectively. All the other bits correspond to optional features (EEE, 1588 timestamping, PMT low-power mode, PHY interface and GPIO). These interrupt events are not frequent, and are not related to normal operation.

If the software misses these exception interrupt events, it is not able to take corrective actions or responses.

Workaround

None.

2.27.7 Context descriptor contains incorrect receive timestamp status in Threshold mode when application clock is very fast

Description

When the Rx timestamp status is available on the ARI interface, the Rx DMA engine writes the context descriptor containing the receive timestamp status to the descriptor memory.

However, the Rx DMA engine may provide an incorrect receive timestamp status in the context descriptor written to descriptor memory, if all of the following conditions are met:

- The application clock is very high compared to receive clock,
- the MTL Rx queue operates in Threshold mode, and
- the Rx queue is empty on the read controller side after the receive status word is read out.

As a result, the Rx DMA engine writes an incorrect timestamp status value to the context descriptor, which may result in an incorrect time correction and/or delay in PTP time synchronization.

This failure is observed only when the application clock frequency is twenty times the MII clock frequency and there are minimal delays on the arbitration and on the read/write access times.

Workaround

Operate the Rx queue in Store-and-forward mode.

2.27.8 Context descriptor is incorrectly written to descriptor memory when the Rx timestamp status is dropped due to unavailability of space in Rx queue

Description

When the Rx timestamp status is captured for a given packet but dropped due to the fact that no more space is available in the MTL Rx queue, the Rx DMA engine sets both the timestamp available (TA) and timestamp dropped (TD) status bits in the last descriptor for this packet, and skips writing the context descriptor as timestamp information is not available.

However, due to a defect, the context descriptor is written to the descriptor memory with invalid Rx timestamp status, and the Rx DMA engine incorrectly makes context writing decision based only on TA status bit, ignoring the TD status bit.

There is no functional impact if the software ignores the next context descriptor based on the setting of the TD status bit. However, this redundant descriptor write operation wastes a descriptor location and unnecessarily consumes system bandwidth.

Workaround

When the TD status bit of RDES1 is set, ignore the next descriptor if it is a context descriptor.

2.27.9 MAC may indicate a power-down state even when disabled by software

Description

Upon writing 1 to the PWRDWN bit of the PMT control status register (ETH_MACPCSR), the MAC enters power-down state and the receiver drops all the received packets until it receives the expected magic packet or the remote wakeup packet. The PWRDWN bit is then self-cleared and the power-down mode is disabled. The PWRDWN bit can also be cleared by software to exit power-down state.

However, the PWRDWN bit of ETH_MACPCSR may not be effectively cleared when the software resets it.

Workaround

Clear the PWRDWN bit twice.

2.27.10 Incorrect Tx FIFO fill-level generation when MAC switches from Full-duplex to Half-duplex mode

Description

The MAC can switch between Half-duplex and Full-duplex modes based on the result of the autonegotiation operation. It seamlessly switches between these modes without affecting the subsequent traffic.

When a collision occurs in Half-duplex mode, the MAC fetches the packet from the Tx queue and transmits it again. To enable this function, the pointer to the start-of-packet is stored in the read controller logic. When the number of bytes transmitted exceeds the collision window or if the packet is transmitted completely without collision, this pointer is updated with the current read pointer and transferred to the write controller to indicate that these locations are now available for storing subsequent packets/data. In Full-duplex mode, this "retry" pointer is not required and it is cleared.

However, when the MAC switches from Full-duplex to Half-duplex mode, this pointer is not updated with the current active read pointer.

This results in an incorrect FIFO fill-level generation and a non-empty Tx queue status. This is reflected in TXQSTS bit of the Tx queue debug register (ETH_MTLTXQDR) even when the Tx queue is actually empty, falsely indicating that the Tx path is not Idle and the packet is still waiting to be transmitted.

Workaround

Flush the Tx queue by setting the FTQ bit in the Tx queue operating mode register (ETH_MTLTXQOMR) immediately after switching from Full-duplex to Half-duplex mode. This clears the write pointer and synchronizes again the read and write pointers.

2.27.11 The MAC does not provide bus access to a higher priority request after a low priority request is serviced

Description

The ETH_DMAMR DMA mode register in the MAC can be programmed to arbitrate between the DMA channels to access the system bus:

- Use a weighted round robin (WRR) algorithm for selecting between transmit or receive DMA channels by clearing DA bit
- Give higher priority to transmit or receive DMA channels by programming the TXPR bit of the ETH_DMAMR register
- Select the priority ratio of TX over RX or vice versa (as per TXPR) by programming the PR[2:0] field

For the WRR algorithm, the MAC provides bus access to a higher priority request provided it is within the priority ratio. It services a lower priority request only when higher priority requests have been serviced as per priority ratio or when there are no higher priority requests.

However, in the WRR algorithm operation, when there are requests pending from both Tx DMA engine and Rx DMA engine after a lower priority request gets serviced, the MAC incorrectly selects the lower priority request, thus violating the PR ratio. The MAC continues to service all the subsequent low priority requests until there are no low priority requests, before servicing any high priority request.

This results in a delay in servicing the higher priority requests. If the high priority request is programmed for receive DMA channels (TXPR is cleared), the receive queue can overflow with a resulting loss of packets. If the high priority request is programmed for transmit DMA (TXPR is set) channels, the transmit queue can get starved in store and forward mode resulting in low throughput. Otherwise when operating in threshold mode, the transmit queue can underflow, resulting in discarding of packet by remote end. In both cases the quality of service or throughput may be affected.

Also, when priority ratio of 8:1 is programmed, the serviced request count rolls over to 0 after reaching 7 and does not reach maximum value which is 8. So, if the higher priority request is being serviced, lower priority request does not get serviced until there is no higher priority request.

These issues do not affect the functionality but impacts the performance.

Workaround

None.

2.27.12 Rx DMA engine may fail to recover upon a restart following a bus error, with Rx timestamping enabled

Description

When the timestamping of the Rx packets is enabled, some or all of the received packets can have an Rx timestamp which is written into a descriptor upon the completion of the Rx packet/status transfer.

However, when a bus error occurs during the descriptor read (that is subsequently used as context descriptor to update the Rx timestamp), the context descriptor write is skipped by the DMA engine. Also, the Rx DMA engine does not flush the Rx timestamp stored in the intermediate buffers during the error recovery process and enters stop state. Due to this residual timestamp in the intermediate buffer remaining after the restart, the Rx DMA engine does not transfer any packets.

Workaround

Issue a soft reset to drop all Tx packets and Rx packets present inside the controller at the time of a bus error. After the soft reset, reconfigure the controller and re-create the descriptors.

Note: The workaround introduces additional latency.

2.27.13 The Tx DMA engine fails to recover correctly or corrupts TSO/USO header data on receiving a bus error response from the AHB DMA slave

Description

When a bus error is received from the AHB DMA slave, the controller generates an interrupt by setting the FBE bit of the ETH_DMACSR register. This stops the corresponding DMA channel by resetting the ST bit of the ETH_DMACTXCR register after recovering from the error. The software recreates the list of descriptors and restarts the DMA engine by setting the ST bit 0 of the ETH_DMACTXCR register without issuing the software reset to the controller.

However, the Tx DMA engine fails to recover or corrupts the TSO/USO header data when the TSO/USO segmentation is enabled in the Tx Descriptor and if either:

- a bus error is detected while transferring the header data from the system memory
- a bus error occurs for the intermediate beat transfer of the header data

In this case the first packet (with TSO/USO enabled after re-starts) gets corrupted after the DMA engine restarts.

Workaround

Issue a soft reset to recover from this scenario. Issuing a soft reset results in loss of all Tx packets and Rx packets present inside the controller at the time of bus-error. Also, the software must reconfigure the controller and re-create the descriptors. This is an overhead which introduces additional latency.

2.27.14 Incorrectly weighted round robin arbitration between Tx and Rx DMA channels to access the common host bus

Description

The ethernet peripheral has independent transmit (Tx) and receive (Rx) DMA engines. The transaction requests from the Tx and Rx DMA engines are arbitrated to allow access to the common DMA master interface. The following two types of arbitrations are supported by programming Bit DA of the ETH_DMAMR register:

- Weighted round-robin arbitration
- Fixed-priority arbitration

The PR[2:0] bit field controls the ratio of the weights between the Tx DMA and Rx DMA engines in the weighted round robin scheme.

However, the programmed polarity ratio PR[2:0] in the weighted round-robin scheme is not adhered to, when there is a priority difference between Rx and Tx. In other words when Rx DMA engine is given higher priority over Tx DMA engine or vice-versa.

The defect occurs in the following conditions:

- The weighted round robin arbitration scheme is selected by clearing the DA bit of the ETH_DMAMR
- Programming different weights in the TXPR and PR fields of ETH_DMAMR
- Both Tx and Rx DMA engines are simultaneously requesting for access.

As a consequence, the expected quality of service (QoS) requirement between Tx and Rx DMA channels for host bus bandwidth allocation might not get adhered to. This defect might have an impact only if the host bus bandwidth is limited and close to or above the total ethernet line rate traffic. The impact can be in terms of buffer underflow (for Tx in cut-through mode) or Buffer overflows (for Rx). If the host side bandwidth is much more than the ethernet line rate traffic, then this bandwidth allocation of WRR scheme is of no consequence.

Workaround

Operate in fixed priority arbitration mode where the DA bit of the ETH_DMAMR is set with Rx DMA engine having a higher priority over Tx clearing the TXPR bit. Operate the Tx buffers in Store-and-Forward mode to avoid any buffer underflows/overflows.

2.27.15 Incorrect L4 inverse filtering results for corrupted packets

Description

Received corrupted IP packets with payload (for IPv4) or total (IPv6) length of less than two bytes for L4 source port (SP) filtering or less than four bytes for L4 destination port (DP) filtering are expected to cause a mismatch. However, the inverse filtering unduly flags a match and the corrupted packets are forwarded to the software application. The L4 stack gets incomplete packet and drops it.

Note: The perfect filtering correctly reports a mismatch.

Workaround

None.

2.27.16 IEEE 1588 Timestamp interrupt status bits are incorrectly cleared on write access to the CSR register with similar offset address

Description

When RCWE bit of the ETH_MACCSRSWCR register is set, all interrupt status bits (events) are cleared only when the specific status bits are set.

However, the status bits[3:0] of the ETH_MACTSSR register at address 0x0B20 are unintentionally cleared when 1 is written to the corresponding bit positions in any CSR register with address offset [7:0] = 0x20. The Status bits[3:0] correspond to the following events:

- Timestamp seconds register overflow interrupt TSSOVF
- Auxiliary timestamp trigger snapshot AUXTSTRIG
- Target time interrupt TSTARGT0

- Target time programming error interrupt TSTRGTERR0

This defect occurs only when the software enables the write 1 to clear interrupt status bits, by setting RCWE of the ETH_MACCSRSWCR register.

As a consequence, when any of the target time interrupts or timestamp seconds overflow events occur, the software might inadvertently clear the corresponding status bits and as a concequence de-assert the interrupt, if it first writes to any CSR register at the shadow address (0x0_xx20 or 0x1_xx20). Consequently, the interrupt service routine might not identify the source of these interrupt events, as the corresponding status bits are already cleared.

Note: The timestamp seconds register overflow event is extremely rare (once in ~137 years) and the target time error interrupt can be avoided by appropriate programming. The frequency of target time reached interrupt events depends on the application usage.

Workaround

When RCWE is set and the timestamp event interrupts are enabled, process and clear the MAC timestamp interrupt events first in the interrupt service routine software, so that write operations to other shadow CSR registers are avoided.

2.27.17 Bus error along with Start-of-Packet can corrupt the ongoing transmission of MAC generated packets

Description

If a bus error is asserted along with the start of a new packet while the MAC is transmitting an internally generated packet such as: ARP, PTO or Pause, the error indication aborts the ongoing transmission prematurely and corrupts the MAC generated packet being transmitted.

As a consequence, the MAC generated packet is sent on the line as a runt frame with corrupted FCS. The aborted packet is not retransmitted and can cause:

- Failure of the intended flow control in case of a Pause/PFC packet corruption.
- Delay in ARP handshake from ARP offload engine; the ARP stack recovers because it sends ARP requests periodically
- Delay in PTP response/SYNC packets generated by PTP offload engine; the PTP stack recovers because it sends request packets periodically.

The probability of occurrence of an bus error on the first beat of data and coinciding with a MAC generated packet transmission is very low.

Workaround

None.

2.27.18 Spurious receive watchdog timeout interrupt

Description

Setting the RWTU[1:0] bitfield of the register to a non-zero value while the RWT[7:0] bitfield is at zero leads to a spurious receive watchdog timeout interrupt (if enabled) and, as a consequence, to executing an unnecessary interrupt service routine with no packets to process.

Workaround

Ensure that the RWTU[1:0] bitfield is not set to a non-zero value while the RWT[7:0] bitfield is at zero. For setting RWT[7:0] and RWTU[1:0] bitfields each to a non-zero value, perform two successive writes. The first is either a byte-wide write to the byte containing the RWT[7:0] bitfield, or a 32-bit write that only sets the RWT[7:0] bitfield and keeps the RWTU[1:0] bitfield at zero. The second is either a byte-wide write to the RWTU[1:0] bitfield or a 32-bit write that sets the RWTU[1:0] bitfield while keeping the RWT[7:0] bitfield unchanged.

2.27.19 Incorrect flexible PPS output interval under specific conditions

Description

The use of the fine correction method for correcting the IEEE 1588 internal time reference, combined with a large frequency drift of the driving clock from the grandmaster source clock, leads to an incorrect interval of the flexible PPS output used in Pulse train mode. As a consequence, external devices synchronized with the flexible PPS output of the device can go out of synchronization.

Workaround

Use the coarse method for correcting the IEEE 1588 internal time reference.

2.27.20 Packets dropped in RMII 10Mbps mode due to fake dribble and CRC error

Description

When operating with the RMII interface at 10 Mbps, the Ethernet peripheral may generate a fake extra nibble of data repeating the last packet (nibble) of the data received from the PHY interface. This results in an odd number of nibbles and is flagged as a dribble error. As the RMII only forwards to the system completed bytes of data, the fake nibble would be ignored and the issue would have no consequence. However, as the CRC error is also flagged when this occurs, the error-packet drop mechanism (if enabled) discards the packets.

Note: Real dribble errors are rare. They may result from synchronization issues due to faulty clock recovery.

Workaround

When using the RMII 10 MHz mode, disable the error-packet drop mechanism by setting the FEP bit of the register. Accept packets of transactions flagging both dribble and CRC errors.

2.27.21 ARP offload function not effective

Description

When the Target Protocol Address of a received ARP request packet matches the device IP address set in the ETH_MACARPAR register, the source MAC address in the SHA field of the ARP request packet is compared with the device MAC address in ETH_MACA0LR and ETH_MACA0HR registers (Address0), to filter out ARP packets that are looping back.

Instead, a byte-swapped comparison is performed by the device. As a consequence, the packet is forwarded to the application as a normal packet with no ARP indication in the packet status, and the device does not generate an ARP response.

For example, with the Address0 set to 0x665544332211:

- If the SHA field of the received ARP packet is 0x665544332211, the ARP response is generated while it should not.
- If the SHA field of the received ARP packet is 0x112233445566, the ARP response not is generated while it should.

Workaround

Parse the received frame by software and send the ARP response if the source MAC address matches the byte-swapped Address0.

2.27.22 Tx DMA may halt while fetching TSO header under specific conditions

Description

Workaround

Ensure that Tx DMA initiates a burst of at least two beats when fetching header bytes from the system memory, through one of the following measures:

- Disable address-aligned beats by clearing the AAL bit of the ETH_DMASBMR register.
- Align the buffer address pointing to the packet start to a data width boundary (set the bits[2:0] of the address to zero for 64-bit data width).
- Set the buffer address pointing to the packet start to a value that ensures a burst of minimum two beats when AAL = 1.

2.28 CEC

2.28.1 Transmission blocked when transmitted start bit is corrupted

Description

When the CEC communication start bit transmitted by the device is corrupted by another device on the CEC line, the CEC transmission is stalled.

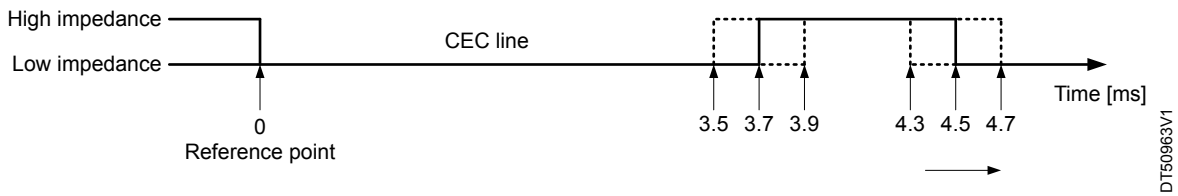
This failure is unlikely to happen as the CEC start bit corruption by another device can only occur if that device does not respect the CEC communication protocol.

The start bit timing standard tolerances are shown in Figure 3. The start bit is initiated by the device by driving the CEC line low (reference point). After 3.7 ms, the device releases the CEC line and starts checking its level. The following conditions must be met for the start bit to be valid:

- the CEC line goes high no later than 3.9 ms (4.05 ms with extended tolerance) from the reference point
- a falling edge on the CEC line does not occur earlier than 4.3 ms (4.15 ms with extended tolerance) from the reference point

If one of these conditions is not met, the transmission is aborted and never automatically retried. No error flag is set and the TXSOM (Tx Start Of Message) bit is not cleared.

Figure 3. CEC start bit format with tolerances



Workaround

The only way to detect this error is for the application software to start a timeout when setting the TXSOM bit, restart it upon ARBLST or any RX event (as the transmission can be delayed by interleaved reception), and stop it upon TXBR (proof that the start bit was transmitted successfully) or TXEND event, or upon any TX error (which clears TXSOM). If the timeout expires (because none of those events occurred), the application software must restart the CEC peripheral and retransmit the message.

2.28.2 Missed CEC messages in normal receiving mode

Description

In normal receiving mode, any CEC message with destination address different from the own address should normally be ignored and have no effect to the CEC peripheral. Instead, such a message is unduly written into the reception buffer and sets the CEC peripheral to a state in which any subsequent message with the destination address equal to the own address is rejected (NACK), although it sets RXOVR flag (because the reception buffer is considered full) and generates (if enabled) an interrupt. This failure can only occur in a multi-node CEC framework where messages with addresses other than own address can appear on the CEC line.

The listen mode operates correctly.

Workaround

Use listen mode (set LSTEN bit) instead of normal receiving mode. Discard messages to single listeners with destination address different from the own address of the CEC peripheral.

Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on www.st.com for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

Revision history

Table 16. **Document revision history**

Date	Version	Changes
19-Jun-2017	1	Initial release.
02-Nov-2017	2	Added STM32H743AI part number. Removed JPEG limitation. Updated Section 2.3.1: Dummy read cycles inserted when reading synchronous memories and Section 2.3.2: Wrong data read from a busy NAND Flash memory. Added Section 2.16.2: Wrong data may be read from Message RAM by the CPU when using two FDCANs.
31-May-2018	3	System limitations: • Reorganized to group together Flash memory and RAM limitations. • Removed limitations “48 MHz RC oscillator user calibration values lost after system reset” and “Flash memory wirte sequence error flag not set”. • Replaced “Accessing the system memory may stall the system when Flash memory banks are swapped” and “Write flags are not swapped when Flash memory banks are swapped” limitations by Section 2.2.8: Flash memory bank swapping might impact embedded Flash memory interface behavior. ADC limitations: • Moved all ADC limitations under Section 2.5: ADC. • Added Section 2.5.5: First ADC injected conversion in a sequence may be corrupted and Section 2.5.6: Writing the ADC_JSQR register when JADCSTART = 1 and JQDIS = 1 may lead to incorrect behavior. Added Section 2.11.1: RTC calendar registers are not locked properly. Added Section 2.4.2: QUADSPI hangs when QUADSPI_CCR is cleared and Section 2.4.3: QUADSPI cannot be used in Indirect read mode when only data phase is activated. I2C limitations: • Updated Section 2.12.1: 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave and Section 2.12.3: Wrong data sampling when data setup time (tSU;DAT) is shorter than one I2C kernel clock period. • Added Section 2.12.2: Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C, Section 2.12.4: Spurious bus error detection in Master mode, Section 2.12.5: Last-received byte loss in Reload mode, Section 2.12.6: Spurious master transfer upon own slave address match and Section 2.12.7: START bit is cleared upon setting ADDRCF, not upon address match. Removed limitation “NOSTRETCH setting may impact Master mode”. SPI/I2S limitations: • Removed limitation “Wrong DMA receive requests may be generated in Half\u0002duplex mode”. • Added Section 2.14.1: Spurious DMA Rx transaction after simplex Tx traffic, Section 2.14.2: Master data transfer stall at system clock much faster than SCK, Section 2.14.3: Corrupted CRC return at non-zero UDRDET setting and Section 2.14.4: TXP interrupt occurring while SPI/I2Sdisabled.
19-Jun-2018	4	Removed revision from document title. Added revision X in Table 2: Device variantsTable 2: Device variants.Replaced “new silicon revision” by revision X in Table 3: Summary of device limitations. Added Section 2.2.13: Unexpected leakage current on I/Os when VIN higher that VDD.
27-Mar-2019	5	Added STM32H742xI/G part numbers.

Date	Version	Changes
		Added revision V for all devices. Added Arm® 32-bit Cortex®-M7 core limitations. System: added Section 2.2.16: Device stalled when two consecutive level regressions occur without accessing from/to backup SRAM, Section 2.2.17: Invalid Flash memory CRC and Section 2.2.18: GPIO assigned to DAC cannot be used in output mode when the DAC output is connected to on-chip peripheral. Added OPAMP limitation. Updated Section 2.4.2: QUADSPI hangs when QUADSPI_CCR is cleared. Updated Section 2.12.1: 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave Added LCD-TFT limitation. Added USB OTG_HS limitation.
20-Jun-2019	6	Added Section 2.2.20: 480 MHz maximum CPU frequency not available on silicon revision Y. Added Section 2.5.7: Conversion may be triggered by context queue register update and Section 2.5.8: Updated conversion sequence may be trigged by context queue update. Added Section 2.18: ETH. Added Section 2.16.3: Mis-synchronization in Edge filtering mode when the falling edge at FDCAN_Rx input pin coincides with the end of the integration phase and Section 2.16.4: Tx FIFO messages inverted when both Tx buffer and FIFO are used and the messages in the Tx buffer have higher priority than in the Tx FIFO.
28-Feb-2020	7	Changed Arm Cortex core revision to r1p1. Added new system limitation Section 2.2.21: VDDLDO is not available on TFBGA100 package on devices revision Y and V. Added new DMA limitation Section 2.13.2: DMA stream locked when transferring data to/from USART/UART. Added new VREFBUF limitations: – Section 2.6.1: Overshoot on VREFBUF output – Section 2.6.2: VREFBUF Hold mode cannot be used
12-Feb-2021	8	SYSTEM limitations: • Added Section 2.2.22: WWDG not functional when VDD is lower than 2.7 V and VOS0 or VOS1 voltage level is selected and Section 2.2.23: A tamper event does not erase the backup RAM when the backup RAM clock is disabled and Section 2.2.24: LSE CSS parasitic detection even when disabled QUADSPI limitations: • Added Section 2.4.4: Memory-mapped read of last memory byte fails. • Updated Section 2.4.2: QUADSPI hangs when QUADSPI_CCR is cleared title. Added TIM limitations: Section 2.9.1: One-pulse mode trigger not detected in master-slave reset + trigger configuration, Section 2.9.2: Consecutive compare event missed in specific conditions and Section 2.9.3: Output compare clear not working with external counter reset. Moved Section 2.13.2: DMA stream locked when transferring data to/from USART/UART from DMA to USART section.
07-Jun-2022	9	Updated Section 2.2.24: LSE CSS parasitic detection even when disabled. Added documentation erratum Section 2.2.25: Output current sunk or sourced by Pxy_C pins. Added Section 3: Important security notice.
08-Aug-2023	10	Added STM32H750xB and STM32H753xI part numbers as well as the corresponding errata. Added silicon revision W.

Date	Version	Changes
		Arm 32-bit Cortex-M7 core: changed Cortex-M7 data corruption when using Data cache configured in write-through and Cortex®-M7 FPU interrupt not present on NVIC line 81 workaround status to 'N'. System: Added Ethernet MII mode is not available on packages with PC2_C/ PC3_C pins, HASH input data may be corrupted when DMA is used, and LSE crystal oscillator may be disturbed by transitions on PC13 errata. Added MDMA, BDMA, DMA and DMAMUX errata. Added DMA2D swap byte feature is not available erratum. FMC: added Unsupported read access with unaligned address and Missing information on prohibited 0xFF value of NAND transaction wait timingerrata. QUADSPI: added QUADSPI internal timing criticality erratum. ADC: added New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0, Two consecutive context conversions fail when writing new context in ADC_JSQR just after previous context completion with JQDIS = 0 and JQM = 0, Unexpected regular conversion when two consecutive injected conversions are performed in Dual interleaved mode, ADC_AWDy_OUT reset by non-guarded channels, Injected data stored in the wrong ADC_JDRx registers and ADC slave data may be shifted in Dual regular simultaneous mode errata. Added DAC errata. VREFBUF: added VREFBUF trimming code not automatically initialized after reset erratum. Added CRYP errata. Added HASH errata. LPTIM: added Device may remain stuck in LPTIM interrupt when clearing event flag and LPTIM events and PWM output are delayed by one kernel clock cycle errata. RTC: added RTC interrupt can be masked by another RTC interrupt, Calendar initialization may fail in case of consecutive INIT mode entry, Alarm flag may be repeatedly set when the core is stopped in debug and A tamper event fails to trigger timestamp or timestamp overflow events during a few cycles after clearing TSF errata. I2C: added OVR flag not set in underrun condition erratum. USART: • Removed erratum *"Underrun flag is set when the USART is used in SPI Slave receive mode"*. • Added UDR flag set while the SPI slave transmitter is disabled, Anticipated end-of-transmission signaling in SPI slave mode, Data corruption due to noisy receive line, Receiver timeout counter wrong start in two-stop-bit configuration, and USART prescaler feature missing in USART implementation section errata. Added LPUART errata. SPI: added Possible corruption of last-received data depending on CRCSIZE setting erratum. SDMMC: • Changed Data 2 line cannot be used to suspend double-data-rate transfers with read wait mode enabled erratum workaround to 'P'. • Added End-of-buffer status flag not cleared when the last burst data is delayed by the slave erratum. FDCAN: added DAR mode transmission failure due to lost arbitration erratum. OTG_HS: • Added Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_HS registers and Host packet transmission may hang when connecting the full speed interface through a hub to a low-speed device errata. • Updated Possible drift of USB PHY pull-up resistor.

Date	Version	Changes
		ETH: added errata Tx DMA engine may halt when a Tx queue flush command is issued in OSP mode to The MAC does not provide bus access to a higher priority request after a low priority request is serviced, The Tx DMA engine fails to recover correctly or corrupts TSO/USO header data on receiving a bus error response from the AHB DMA slave to Bus error along with Start-of-Packet can corrupt the ongoing transmission of MAC generated packets. CEC: • Removed "Unexpected switch to Receive mode without automatic transmission retry and notification" and "CEC header not received due to unjustified Rx-Overrun detection" errata. • Added Transmission blocked when transmitted start bit is corrupted and Missed CEC messages in normal receiving mode errata.

Contents