STM32H745xI/G, STM32H755xI, STM32H747xI/G, STM32H757xI device errata

Applicability

This document applies to the part numbers of STM32H745xI/G, STM32H755xI, STM32H747xI/G, STM32H757xI 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 RM0399.

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
STM32H745xI/G	STM32H745ZI, STM32H745II, STM32H745BI, STM32H745XI, STM32H745ZG, STM32H745IG, STM32H745BG, STM32H745XG
STM32H755xI	STM32H755ZI, STM32H755II, STM32H755BI, STM32H755XI
STM32H747xI/G	STM32H747AI, STM32H747BI, STM32H747II, STM32H747XI, STM32H747ZI, STM32H747AG, STM32H747BG, STM32H747IG, STM32H747XG
STM32H757xI	STM32H757AI, STM32H757BI, STM32H757II, STM32H757XI, STM32H757ZI

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32H745xI/G, STM32H755xI, STM32H747xI/G, STM32H757xI	V, U	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 STM32H745xI/G, STM32H755xI, STM32H747xI/G, STM32H757xI 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. V,U
Arm 32-bit Cortex-M7 core	2.1.1	Cortex-M7 data corruption when using Data cache configured in write-through	N
	2.1.2	Cortex®-M7 FPU interrupt not present on NVIC line 81	N
Arm 32-bit Cortex-M4 core	2.2.1	Interrupted loads to SP can cause erroneous behavior	A
	2.2.2	VDIV or VSQRT instructions may not complete correctly when very short ISRs are used	A
System	2.3.1	AXI domain locked when watchdog reset limited to CPU1 or CPU2	A
	2.3.2	Device stalled when two consecutive level regressions occur without accessing from/to backup SRAM	A
	2.3.3	GPIO assigned to DAC cannot be used in output mode when the DAC output is connected to on-chip peripheral	N
	2.3.4	Unstable LSI when it clocks RTC or CSS on LSE	P
	2.3.5	VDDLDO is not available on TFBGA100 package	N
	2.3.6	WWDG not functional when V_{DD} is lower than 2.7 V and VOS0 or VOS1 voltage level is selected	N
	2.3.7	A tamper event does not erase the backup RAM when the backup RAM clock is disabled	A
	2.3.8	LSE CSS detection occurs even when the LSE CSS is disabled	P
	2.3.10	Ethernet MII mode is not available on packages with PC2_C/PC3_C pins	A
	2.3.11	HASH input data may be corrupted when DMA is used	A
	2.3.12	LSE crystal oscillator may be disturbed by transitions on PC13	N
MDMA	2.4.1	Non-flagged MDMA write attempts to reserved area	A
BDMA	2.5.1	BDMA disable failure and error flag omission upon simultaneous transfer error and global flag clear	A
DMAMUX	2.7.1	SOFx not asserted when writing into DMAMUX_CFR register	N
	2.7.2	OFx not asserted for trigger event coinciding with last DMAMUX request	N
	2.7.3	OFx not asserted when writing into DMAMUX_RGCFR register	N
	2.7.4	Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event	A
FMC	2.8.1	Dummy read cycles inserted when reading synchronous memories	N
	2.8.3	Wrong data read from a busy NAND memory	A

Function	Section	Limitation	Status
			Rev. V,U
FMC	2.8.4	Unsupported read access with unaligned address	P
QUADSPI	2.9.1	QUADSPI cannot be used in indirect read mode when only data phase is activated	P
	2.9.2	QUADSPI hangs when QUADSPI_CCR is cleared	P
	2.9.3	QUADSPI internal timing criticality	A
	2.9.4	Memory-mapped read of last memory byte fails	P
	2.9.5	QUADSPI memory failure when using HCLK quadspi_ker_ck clock as QUADSPI CLK	A
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
	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
	2.10.4	Unexpected regular conversion when two consecutive injected conversions are performed in Dual interleaved mode	A
	2.10.5	ADC_AWDy_OUT reset by non-guarded channels	A
	2.10.6	Injected data stored in the wrong ADC_JDRx registers	A
	2.10.7	ADC slave data may be shifted in Dual regular simultaneous mode	A
	2.10.8	Conversion triggered by context queue register update	A
	2.10.9	Updated conversion sequence triggered by context queue update	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
VREFBUF	2.12.1	Overshoot on VREFBUF output	A
	2.12.2	VREFBUF Hold mode cannot be used	N
LTDC	2.13.1	Device stalled when accessing LTDC registers while pixel clock is disabled	A
DSI	2.14.1	Tearing effect parasitic detection	P
	2.14.2	Incorrect calculation of the time to activate the clock between HS transmissions	P
	2.14.3	The immediate update procedure may fail	A
TIM	2.17.1	One-pulse mode trigger not detected in master-slave reset + trigger configuration	P
	2.17.2	Consecutive compare event missed in specific conditions	N
	2.17.3	Output compare clear not working with external counter reset	P
LPTIM	2.18.1	Device may remain stuck in LPTIM interrupt when entering Stop mode	A
	2.18.2	Device may remain stuck in LPTIM interrupt when clearing event flag	P
	2.18.3	LPTIM events and PWM output are delayed by one kernel clock cycle	P
RTC	2.19.1	RTC interrupt can be masked by another RTC interrupt	A
	2.19.2	Calendar initialization may fail in case of consecutive INIT mode entry	A
	2.19.3	Alarm flag may be repeatedly set when the core is stopped in debug	N
	2.19.4	A tamper event fails to trigger timestamp or timestamp overflow events during a few cycles after clearing TSF	N
I2C	2.20.1	Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period	P

Function	Section	Limitation	Status Rev. V,U
I2C	2.20.2	Spurious bus error detection in master mode	A
	2.20.3	Spurious master transfer upon own slave address match	P
	2.20.5	OVR flag not set in underrun condition	N
	2.20.6	Transmission stalled after first byte transfer	A
USART	2.21.2	Anticipated end-of-transmission signaling in SPI slave mode	A
	2.21.3	Data corruption due to noisy receive line	A
	2.21.5	DMA stream locked when transferring data to/from USART	A
LPUART	2.22.1	DMA stream locked when transferring data to/from LPUART	A
	2.22.2	Possible LPUART transmitter issue when using low BRR[15:0] value	P
SPI	2.23.1	Master data transfer stall at system clock much faster than SCK	A
	2.23.2	Corrupted CRC return at non-zero UDRDET setting	P
	2.23.3	TXP interrupt occurring while SPI disabled	A
	2.23.4	Possible corruption of last-received data depending on CRCSIZE setting	A
FDCAN	2.24.1	Desynchronization under specific condition with edge filtering enabled	A
	2.24.2	Tx FIFO messages inverted under specific buffer usage and priority setting	A
	2.24.3	DAR mode transmission failure due to lost arbitration	A
OTG_HS	2.25.1	Host packet transmission may hang when connecting the full speed interface through a hub to a low-speed device	N
ETH	2.26.1	The MAC does not provide bus access to a higher priority request after a low priority request is serviced	N
	2.26.2	Rx DMA engine may fail to recover upon a restart following a bus error, with Rx timestamping enabled	A
	2.26.3	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
	2.26.4	Incorrectly weighted round robin arbitration between Tx and Rx DMA channels to access the common host bus	A
	2.26.5	Incorrect L4 inverse filtering results for corrupted packets	N
	2.26.6	IEEE 1588 Timestamp interrupt status bits are incorrectly cleared on write access to the CSR register with similar offset address	A
	2.26.7	Bus error along with Start-of-Packet can corrupt the ongoing transmission of MAC generated packets	N
	2.26.8	Spurious receive watchdog timeout interrupt	A
	2.26.9	Incorrect flexible PPS output interval under specific conditions	A
	2.26.10	Packets dropped in RMII 10Mbps mode due to fake dribble and CRC error	A
	2.26.11	ARP offload function not effective	A
	2.26.12	Tx DMA may halt while fetching TSO header under specific conditions	A
CEC	2.27.1	Missed CEC messages in normal receiving mode	A
	2.27.2	Unexpected TXERR flag during a message transmission	A

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

Table 4. Summary of device documentation errata

Function	Section	Documentation erratum
System	2.3.9	Output current sunk or sourced by Pxy_C pins
BDMA	2.5.2	Byte and half-word accesses not supported
DMA	2.6.1	USART/UART/LPUART DMA transfer abort
DMAMUX	2.7.5	DMAMUX_RGCFR register is write-only, not read-write
	2.7.6	DMA request counter not kept at GNBREQ bitfield value as long as the corresponding request channel is disabled
	2.7.7	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
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	2.20.4	START bit is cleared upon setting ADDRCF, not upon address match
USART	2.21.1	Receiver timeout counter wrong start in two-stop-bit configuration
	2.21.4	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 Arm 32-bit Cortex-M4 core

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

2.2.1 Interrupted loads to SP can cause erroneous behavior

Description

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

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

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

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

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

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

Workaround

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

Example:

Replace LDR SP,[R0] with:

```
LDR R2,[R0]
MOV SP,R2
```

2.2.2 VDIV or VSQRT instructions may not complete correctly when very short ISRs are used

Description

The VDIV and VSQRT instructions take 14 cycles to execute. When an interrupt is taken, a VDIV or VSQRT instruction is not terminated, and completes its execution while the interrupt stacking occurs. If the lazy context save of the floating-point state is enabled, then the automatic stacking of the floating-point context does not occur until a floating-point instruction is executed inside the interrupt service routine.

The lazy context save is enabled by default. When it is enabled, the minimum time for the first instruction in the interrupt service routine to start executing is 12 cycles. In certain timing conditions, and if there are only one or two instructions inside the interrupt service routine, then the VDIV or VSQRT instruction might not write its result to the register bank or to the FPSCR.

The failure occurs when all the following conditions are met:

- The floating-point unit is enabled.
- Lazy context saving is not disabled.
- A VDIV or VSQRT is executed.
- The destination register for the VDIV or VSQRT is one of s0 - s15.
- An interrupt occurs and is taken.
- The interrupt service routine being executed does not contain a floating-point instruction.
- Within 14 cycles after the VDIV or VSQRT is executed, an interrupt return is executed.

A minimum of 12 of these 14 cycles are used for the context state stacking, which leaves two cycles for instructions inside the interrupt service routine, or 2 wait states applied to the entire stacking sequence (which means that it is not a constant wait state for every access).

In general, this means that if the memory system inserts wait states for stack transactions (that is, external memory is used for stack data), then this erratum cannot be observed.

The effect of this limmitation is that the VDIV or VQSRT instruction does not complete correctly, the register bank and FPSCR are not updated, which means that these registers hold incorrect, out-of-date, data.

Workaround

A workaround is only required if the floating-point unit is enabled. A workaround is not required if the stack is in external memory.

There are two possible workarounds:

- Disable lazy context save of floating-point state by clearing LSPEN to 0 (bit 30 of the FPCCR at address 0xE000EF34).
- Ensure that every interrupt service routine contains more than two instructions in addition to the exception return instruction.

2.3 System

2.3.1 AXI domain locked when watchdog reset limited to CPU1 or CPU2

Description

The AXI domain may get locked when the watchdog reset is limited to CPU1 or CPU2 (WW1RSC or WW2RSC cleared in the RCC_GCR register), and a watchdog reset is generated.

Workaround

Configure the watchdog reset scope to reset the full system, by setting the WW1RSC or WW2RSC bit of the RCC_GCR register.

2.3.2 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).

2.3.3 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.

2.3.4 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.

2.3.5 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.

2.3.6 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.

2.3.7 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.

2.3.8 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.

2.3.9 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.

2.3.10 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.

2.3.11 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.

2.3.12 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.

2.4 MDMA

2.4.1 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.

2.5 BDMA

2.5.1 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.

2.5.2 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.

2.6 DMA

2.6.1 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.

2.7 DMAMUX

2.7.1 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.

2.7.2 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.

2.7.3 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.

2.7.4 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.

2.7.5 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.

2.7.6 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.

2.7.7 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.

2.8 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 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 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.

2.9.2 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.

2.9.3 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.

2.9.4 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.

2.9.5 QUADSPI memory failure when using HCLK quadspi_ker_ck clock as QUADSPI CLK

Description

When using HCLK as kernel clock for the QUADSPI peripheral and PRESCALER[7:0] bit = 0 in the QUADSPI_CR register, memories sensitive to clock duty cycle fail at high speed because the clock does not respect the 50% duty cycle.

Workaround

- When using HCLK as kernel clock for the QUADSPI peripheral, use a clock prescale that respects the 50% duty cycle.

Or

- Use an alternative kernel clock source, for example, pll1_q_ck or pll2_r_ck with QSPISEL[1:0] bits in the RCC_D1CCIPR register.

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 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.

2.10.9 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.

2.13 LTDC

2.13.1 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.

2.14 DSI

2.14.1 Tearing effect parasitic detection

Description

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

An acknowledge trigger can be received from the display:

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

Workaround

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

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

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

Description

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

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

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

Workaround

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

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

- HS2LP_TIME = 113
- LP2HS_TIME =113

2.14.3 The immediate update procedure may fail

Description

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

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

Workaround

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

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

2.15 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 5 and Table 6. 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 5. 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 6. 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 7 and Table 8. 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 7. 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 8. 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 9.

Table 9. 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 10.

Table 10. 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 11.

Table 11. 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 12.

Table 12. 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 13. Data is stored in system memory in **big-endian** format.

Table 13. 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 14. Data is stored in system memory in **big-endian** format.

Table 14. **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 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.

2.16 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

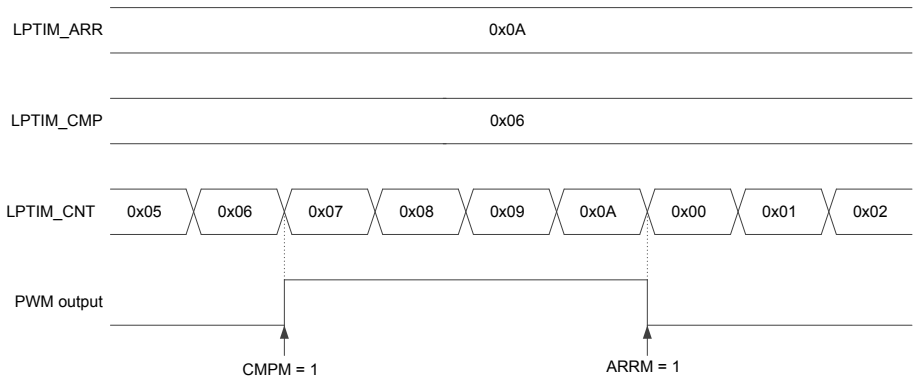
Description

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:

Figure 1. Example of PWM output mode



Workaround

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

2.19 RTC

2.19.1 RTC interrupt can be masked by another RTC interrupt

Description

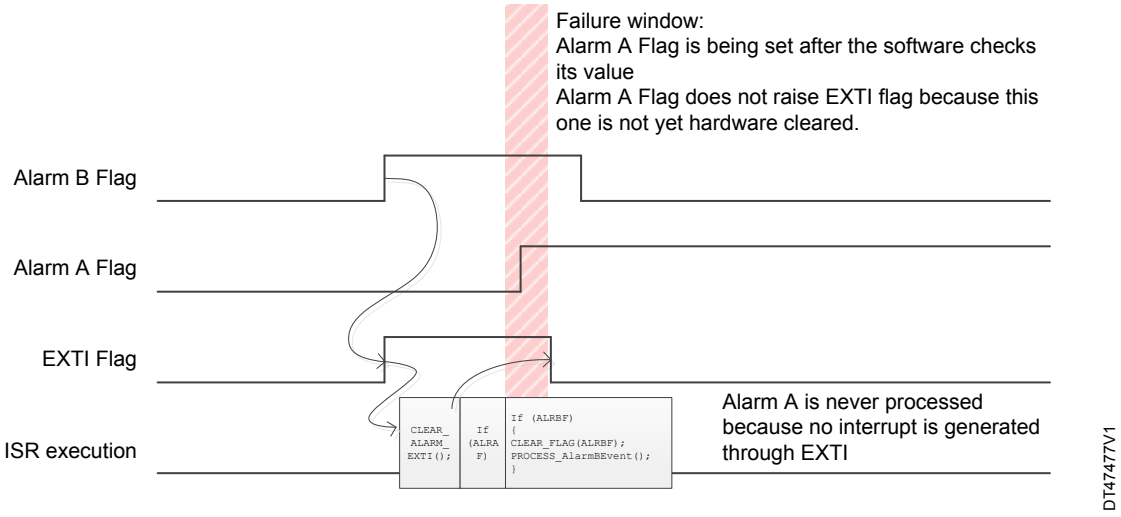
One RTC interrupt request can mask another RTC interrupt request if they share the same EXTI configurable line. For example, interrupt requests from Alarm A and Alarm B or those from tamper and timestamp events are OR-ed to the same EXTI line (refer to the *EXTI line connections* table in the *Extended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

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

Alarm interrupt service routine:

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

Figure 2. 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 */
    }
}
```

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

Description

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

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

Workaround

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

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

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

Description

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

2.19.4 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 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.

2.20.2 Spurious bus error detection in master mode

Description

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

Workaround

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

2.20.3 Spurious master transfer upon own slave address match

Description

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

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

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

Workaround

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

Normal mode (SBC = 0):

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

Slave byte control mode (SBC = 1):

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

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

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

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

Description

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

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

Workaround

No application workaround is required for this description inaccuracy issue.

2.20.5 OVR flag not set in underrun condition

Description

In slave transmission with clock stretching disabled (NOSTRETCH = 1 in the I2C_CR1 register), an underrun condition occurs if the current byte transmission is completed on the I^2C bus, and the next data is not yet written in the TXDATA[7:0] bitfield. In this condition, the device is expected to set the OVR flag of the I2C_ISR register and send 0xFF on the bus.

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

Workaround

None.

2.20.6 Transmission stalled after first byte transfer

Description

When the first byte to transmit is not prepared in the TXDATA register, two bytes are required successively, through TXIS status flag setting or through a DMA request. If the first of the two bytes is written in the I2C_TXDR register in less than two I2C kernel clock cycles after the TXIS/DMA request, and the ratio between APB clock and I2C kernel clock frequencies is between 1.5 and 3, the second byte written in the I2C_TXDR is not internally detected. This causes a state in which the I2C peripheral is stalled in master mode or in slave mode, with clock stretching enabled (NOSTRETCH = 0). This state can only be released by disabling the peripheral (PE = 0) or by resetting it.

Workaround

Apply one of the following measures:

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

2.21 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 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.

2.21.3 Data corruption due to noisy receive line

Description

In all modes, except synchronous slave mode, the received data may be corrupted if a glitch to zero shorter than the half-bit occurs on the receive line within the second half of the stop bit.

Workaround

Apply one of the following measures:

- Either use a noiseless receive line, or
- add a filter to remove the glitches if the receive line is noisy.

2.21.4 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.

2.21.5 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 a 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 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.

2.23.2 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.

2.23.3 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.

2.23.4 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 FDCAN

2.24.1 Desynchronization under specific condition with edge filtering enabled

Description

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

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

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

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

Workaround

Disable edge filtering or wait for frame retransmission.

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

Description

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

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

Workaround

Apply one of the following measures:

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

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

2.24.3 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.

2.25 OTG_HS

2.25.1 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.

2.26 ETH

2.26.1 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.

2.26.2 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.

2.26.3 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.

2.26.4 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.

2.26.5 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.

2.26.6 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.

2.26.7 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.

2.26.8 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.

2.26.9 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.

2.26.10 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.

2.26.11 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.

2.26.12 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.

2.27 CEC

2.27.1 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.

2.27.2 Unexpected TXERR flag during a message transmission

Description

During the transmission of a 0 or a 1, the HDMI-CEC drives the open-drain output to high-Z, so that the external pull-up implements a voltage rising ramp on the CEC line.

In some load conditions, with several powered-off devices connected to the HDMI-CEC line, the rising voltage may not drive the HDMI-CEC GPIO input buffer to V_{IH} within two HDMI-CEC clock cycles from the high-Z activation to TXERR flag assertion.

Workaround

Limit the maximum number of devices connected to the HDMI-CEC line to ensure the GPIO V_{IH} threshold is reached within a time of two HDMI-CEC clock cycles (~61 µs).

The maximum equivalent 10%-90% rise time for the HDMI-CEC line is 111.5 µs, considering a V_{IH} threshold equal to 0.7 x V_{DD}.

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 15. Document revision history

Date	Version	Changes
16-May-2019	1	Initial release.
07-Jun-2022	2	Changed Arm® processor Cortex®-M7 core revision to r1p1 in Section 2.2: Arm® 32-bit Cortex®-M7 core. Added Section 2.3.5: LSE CSS parasitic detection even when disabled and Section 2.3.6: Output current sunk or sourced by Pxy_C pins. Added Section 2.5.3: Memory-mapped read of last memory byte fails and updated Section 2.5.1: QUADSPI hangs when QUADSPI_CCR is cleared title. Added Section 3: Important security notice.
18-Jul-2023	3	Added silicon revision U. 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 VDDLDO is not available on TFBGA100 package, WWDG not functional when V_{DD} is lower than 2.7 V and VOS0 or VOS1 voltage level is selected, A tamper event does not erase the backup RAM when the backup RAM clock is disabled, Ethernet MII mode is not available on packages with PC2_C/PC3_C pins, and HASH input data may be corrupted when DMA is used errata. Added MDMA, BDMA, DMA and DMAMUX errata. 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. Added ADC, DAC and VREFBUF errata. DSI: • Removed *"Data corruption when accessing DSI Host registers while pixel clock is disabled"* erratum. • Added Tearing effect parasitic detection, Incorrect calculation of the time to activate the clock between HS transmissions and The immediate update procedure may fail errata. Added CRYP errata. Added HASH, TIM, LPTIM and RTC errata. I2C: • Removed "*10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave"* and *"Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C'* errata. • Added Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period, OVR flag not set in underrun condition and Transmission stalled after first byte transfer errata. USART: • Removed erratum *"Underrun flag is set when the USART is used in SPI Slave receive mode"*. • Added , Anticipated end-of-transmission signaling in SPI slave mode, Data corruption due to noisy receive line, DMA stream locked when transferring data to/from USART, 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.

Date	Version	Changes
		FDCAN: • Removed "*Writing FDCAN_TTTS during initialization corrupts FDCAN_TTTMC*" erratum. • Added DAR mode transmission failure due to lost arbitration erratum. Added OTG_HS errata. ETH: added errata The MAC does not provide bus access to a higher priority request after a low priority request is serviced, Incorrectly weighted round robin arbitration between Tx and Rx DMA channels to access the common host bus, IEEE 1588 Timestamp interrupt status bits are incorrectly cleared on write access to the CSR register with similar offset address, Bus error along with Start-of-Packet can corrupt the ongoing transmission of MAC generated packets, and Tx DMA may halt while fetching TSO header under specific conditions .
15-Dec-2023	4	CEC: added errata Section 2.27.2 Unexpected TXERR flag during a message transmission and Section 2.27.1 Missed CEC messages in normal receiving mode. Added Section 2.9.5 QUADSPI memory failure when using HCLK quadspi_ker_ck clock as QUADSPI CLK. Added Section 2.3.12 LSE crystal oscillator may be disturbed by transitions on PC13 erratum. Updated Section 2.21.3 Data corruption due to noisy receive line erratum.

Contents

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2023 STMicroelectronics – All rights reserved