STM32H562xx/563xx/573xx device errata

Applicability

This document applies to the part numbers of STM32H562xx/563xx/573xx 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 RM0481.

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
STM32H562xx	STM32H562AI, STM32H562II, STM32H562RI, STM32H562VI, STM32H562ZI, STM32H562RG, STM32H562VG, STM32H562ZG, STM32H562IG, STM32H562AG
STM32H563xx	STM32H563RI, STM32H563VI, STM32H563ZI, STM32H563MI, STM32H563AI, STM32H563II, STM32H563RG, STM32H563VG, STM32H563ZG, STM32H563IG, STM32H563AG
STM32H573xx	STM32H573RI, STM32H573VI, STM32H573ZI, STM32H573MI, STM32H573AI, STM32H573II

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32H562xx/STM32H763xx/STM32H573xx	A	0x1000
	Z	0x1001
	X	0x1007

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

1 Summary of device errata

The following table gives a quick reference to the STM32H562xx/563xx/573xx device limitations and their status:

A = limitation present, workaround available

N = limitation present, no workaround available

P = limitation present, partial workaround available

"-" = limitation absent

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

Table 3. Summary of device limitations

Function	Section	Limitation	Status		
			Rev. A	Rev. Z	Rev. X
Core	2.1.1	Access permission faults are prioritized over unaligned Device memory faults	N	N	N
System	2.2.1	LSE crystal oscillator may be disturbed by transitions on PC13	N	N	N
	2.2.2	Peripheral triggers connected to RTC wakeup timer interrupt instead of RTC wakeup timer trigger signal	A	A	A
	2.2.3	HSE not operational as oscillator	N	-	-
	2.2.4	Incorrect behavior of ICACHE refill from SRAM when an SRAM AHB error occurs	N	N	N
	2.2.5	Device cannot exit from Standby mode except by a power-on reset	N	-	-
	2.2.6	SRAMx_RST option bits have an immediate effect	A	A	A
	2.2.7	Additional consumption on PA11/PA12 in Stop and Standby modes	A	-	-
	2.2.8	Product state change is not supported on early samples	P	-	-
	2.2.9	Flash memory latency is increased during read-while-write (RWW) operation	A	A	-
	2.2.10	Product state regression is not possible when data flash memory is enabled	N	N	-
	2.2.11	Full JTAG configuration without NJTRST pin cannot be used	A	A	A
	2.2.12	BOOT0 pin is sensitive to electrostatic discharge	N	N	-
	2.2.13	Flash memory endurance is limited to 1 Kcycle	N	N	-
	2.2.14	SRAM2 is erased when the backup domain is reset	A	A	A
	2.2.15	Clearing IWDG_SW might result in debug authentication or ST-iRoT failure	A	A	-
	2.2.16	Clearing WWDG_SW might result in debug authentication or ST-iRoT failure	A	A	A
	2.2.17	LSE low drive mode is not functional	N	N	N
	2.2.18	Read from flash memory may fail in VOS2 and VOS1 range	A	A	-
	2.2.19	Partial regression may lead to incorrect device configuration	A	A	-
	2.2.20	Incorrect backup domain reset	A	A	A
	2.2.21	Debug not available when TrustZone® security is disabled and the PRODUCT_STATE is iROT-Provisioned	A	A	A
	2.2.22	Incorrect flash memory configuration on early STM32H562xG and STM32H563xG devices	N	N	N

Function	Section	Limitation	Status		
			Rev. A	Rev. Z	Rev. X
System	2.2.23	Reset vector catch feature cannot be used when debugging is disabled for secure code	A	A	A
GPIO	2.3.1	Input hysteresis not properly enabled on PA1	N	-	-
FMC	2.4.1	Dummy read cycles inserted when reading synchronous memories	N	N	N
	2.4.3	Wrong data read from a busy NAND memory	A	A	A
OCTOSPI	2.5.1	Memory-mapped write error response when DQS output is disabled	P	P	P
	2.5.2	TCF set twice on an abort	A	A	A
	2.5.3	Deadlock in clock mode 3 when prefetching the 64 upper memory bytes	A	A	A
	2.5.4	Memory wrap instruction not enabled when DQS is disabled	N	N	N
	2.5.5	Deadlock or write-data corruption after spurious write to a misaligned address in OCTOSPI_ AR register	N	N	N
	2.5.6	Deadlock on consecutive out-of-range memory-mapped write operations	P	P	P
	2.5.7	Indirect write mode limited to 256 Mbytes	N	N	N
	2.5.8	Read-modify-write operation does not clear the MSEL bit	A	A	A
ADC	2.6.1	New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0	A	A	A
	2.6.2	Two consecutive context conversions fail when writing new context in ADC_JSQR just after previous context completion with JQDIS = 0 and JQM = 0	A	A	A
	2.6.3	Unexpected regular conversion when two consecutive injected conversions are performed in Dual interleaved mode	A	A	A
	2.6.4	ADC_AWDy_OUT reset by non-guarded channels	A	A	A
	2.6.5	Injected data stored in the wrong ADC_JDRx registers	A	A	A
	2.6.6	ADC slave data may be shifted in Dual regular simultaneous mode	A	A	A
VREFBUF	2.7.1	Slow discharge of the capacitor connected to VREF+ pin	A	-	-
TIM	2.8.1	Consecutive compare event missed in specific conditions	N	N	N
	2.8.2	Output compare clear not working with external counter reset	P	P	P
	2.8.3	Timer connection to USB SOF might not work properly	A	A	A
LPTIM	2.9.1	Device may remain stuck in LPTIM interrupt when entering Stop mode	A	A	A
	2.9.2	ARRM and CMPM flags are not set when APB clock is slower than kernel clock	P	P	P
	2.9.3	Device may remain stuck in LPTIM interrupt when clearing event flag	P	P	P
	2.9.4	Interrupt status flag is cleared by hardware upon writing its corresponding bit in LPTIM_DIER register	N	N	N
	2.9.5	Overcapture stops working when CCxOF flag is cleared in some specific conditions	P	-	-
	2.9.6	PLL2 output cannot be used as LPTIM clock source	A	A	A
IWDG	2.10.1	Independent watchdog does not wake up the system from Stop mode	N	N	N
RTC and TAMP	2.11.1	Alarm flag may be repeatedly set when the core is stopped in debug	N	N	N
	2.11.2	Tamper event 15 raised when a system reset occurs	N	-	-
	2.11.3	Timestamp flag unexpectedly raised when disabling timestamp	A	A	A

Function	Section	Limitation	Status		
			Rev. A	Rev. Z	Rev. X
I2C	2.12.1	Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period	P	P	P
	2.12.2	Spurious bus error detection in master mode	A	A	A
	2.12.3	SDA held low upon SMBus timeout expiry in slave mode	A	A	A
I3C	2.13.1	I3C controller: unexpected read data bytes during a legacy I^2C read	A	A	A
	2.13.2	I3C controller: SCL clock is not stalled during address ACK/NACK phase following a frame start, when enabled through I3C_TIMINGR2 register	A	A	A
	2.13.3	I3C controller: unexpected first frame with a 0x7F address when the I3C peripheral is enabled	A	A	A
	2.13.4	I3C controller: no timestamp on IBI acknowledge when timing control is used in Asynchronous mode 0	A	A	A
USART	2.14.1	USART does not generate DMA requests after setting/clearing DMAT bit	A	-	-
	2.14.2	Wrong data received in smartcard mode and 0.5 stop bit configuration	A	-	-
LPUART	2.15.1	LPUART does not generate DMA requests after setting/clearing DMAT bit	A	-	-
	2.15.2	Possible LPUART transmitter issue when using low BRR[15:0] value	P	P	P
SPI	2.16.1	Truncation of SPI output signals after EOT event	A	A	A
FDCAN	2.17.1	Desynchronization under specific condition with edge filtering enabled	A	A	A
	2.17.2	Tx FIFO messages inverted under specific buffer usage and priority setting	A	A	A
USB	2.18.1	COUNTn_RX[9:0] bitfield reporting one byte less if APB frequency is below 12.6 MHz	A	A	A
	2.18.2	False wakeup detection for last K-state not terminated by EOP or reset before suspend	A	A	A
	2.18.3	ESOF interrupt timing desynchronized after resume signaling	A	A	A
	2.18.4	Incorrect CRC16 in the memory buffer	N	N	N
ETH	2.19.1	The MAC does not provide bus access to a higher priority request after a low priority request is serviced	N	N	N
	2.19.2	Rx DMA engine may fail to recover upon a restart following a bus error, with Rx timestamping enabled	A	A	A
	2.19.3	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	N	N
	2.19.4	Incorrectly weighted round robin arbitration between Tx and Rx DMA channels to access the common host bus	A	A	A
	2.19.5	Incorrect L4 inverse filtering results for corrupted packets	N	N	N
	2.19.6	IEEE 1588 Timestamp interrupt status bits are incorrectly cleared on write access to the CSR register with similar offset address	A	A	A
	2.19.7	Bus error along with Start-of-Packet can corrupt the ongoing transmission of MAC generated packets	N	N	N
	2.19.8	Spurious receive watchdog timeout interrupt	A	A	A
	2.19.9	Incorrect flexible PPS output interval under specific conditions	A	A	A
	2.19.10	Packets dropped in RMII 10Mbps mode due to fake dribble and CRC error	A	A	A
	2.19.11	ARP offload function not effective	A	A	A

Function	Section	Limitation	Status		
			Rev. A	Rev. Z	Rev. X
CEC	2.20.1	Missed CEC messages in normal receiving mode	A	A	A
	2.20.2	Unexpected TXERR flag during a message transmission	A	A	A

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

Table 4. Summary of device documentation errata

Function	Section	Documentation erratum
System	2.2.24	Ethernet alternate functions not available on PB13 and PB14
FMC	2.4.2	Missing information on prohibited 0xFF value of NAND transaction wait timing

2 Description of device errata

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

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

2.1 Core

Reference manual and errata notice for the Arm® Cortex®-M33 core revision r0p4 is available from http://infocenter.arm.com. Only applicable information from the Arm errata notice is replicated in this document.

2.1.1 Access permission faults are prioritized over unaligned Device memory faults

Description

A load or store which causes an unaligned access to Device memory will result in an UNALIGNED UsageFault exception. However, if the region is not accessible because of the MPU access permissions (as specified in MPU_RBAR.AP), then the resulting MemManage fault will be prioritized over the UsageFault.

The failure occurs when the MPU is enabled and:

- A load/store access occurs to an address which is not aligned to the data type specified in the instruction.
- The memory access hits one region only.
- The region attributes (specified in the MAIR register) mark the location as Device memory.
- The region access permissions prevent the access (that is, unprivileged or write not allowed).

The MemManage fault caused by the access permission violation will be prioritized over the UNALIGNED UsageFault exception because of the memory attributes.

Workaround

None. However, it is expected that no existing software is relying on this behavior since it was permitted in Armv7-M.

2.2 System

2.2.1 LSE crystal oscillator may be disturbed by transitions on PC13

Description

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.

2.2.2 Peripheral triggers connected to RTC wakeup timer interrupt instead of RTC wakeup timer trigger signal

Description

GPDMA1, GPDMA2 and TIM16 tim_ti1_in3 triggers are connected to RTC wakeup timer interrupt instead of RTC wakeup timer trigger signal (rtc_wut_trg).

Workaround

Enable the RTC wakeup timer interrupt. The wakeup timer flag must be cleared in the interrupt subroutine before getting a new trigger.

To avoid serving an interrupt, use triggers from RTC alarm or from LPTIM instead of RTC wakeup timer.

2.2.3 HSE not operational as oscillator

Description

HSE oscillator with an external crystal or a ceramic resonator does not operate.

Note: HSE bypass allowing the use of an external clock on the PH0 input operates as expected. HSI or CSI internal oscillators operate as expected, too.

Workaround

None.

2.2.4 Incorrect behavior of ICACHE refill from SRAM when an SRAM AHB error occurs

Description

If an AHB error is returned by the SRAM memory protection controller (MPCBB) during an ICACHE refill operation from SRAM, the next access may not be correctly handled and thus corrupted.

The SRAM (MPCBB) returns an AHB error during a CPU fetch operation if the one of the following conditions is met:

- a secure access to a non-secure area is ongoing (only when TrustZone® is enabled),
- a non-secure access to a secure area is ongoing (only when TrustZone® is enabled),
- a non-privileged access to a privilege area is ongoing, or
- a non-mapped address is accessed.

Workaround

None.

2.2.5 Device cannot exit from Standby mode except by a power-on reset

Description

When the device is in Standby mode, it can only be woken up by a power-on reset.

Workaround

None.

2.2.6 SRAMx_RST option bits have an immediate effect

Description

Clearing the SRAMx_RST (x = 1, 2) option bit immediately triggers an SRAMx erase operation. This might lead to a CPU exception or unpredictable behavior if the erased SRAMx is being used by the application.

Workaround

The application must be reset immediately after SRAMx_RST clear.

2.2.7 Additional consumption on PA11/PA12 in Stop and Standby modes

Description

An additional current consumption of around 40 µA can be observed when PA11 or PA12 is left floating during Stop or Standby mode.

Workaround

Apply one of the following measures:

- Software measure:
  - In Stop mode, configure PA11 or PA12 in pull-down or in pull-up mode.
  - In Standby mode, enable the I/O retention feature.
- Hardware measure: fix externally the level of PA11 and PA12.

2.2.8 Product state change is not supported on early samples

Description

Early sample devices are locked if all the following conditions are met:

- The secure native services (SNS) identifier is equal to 0xA7CA 2047 or 0xFFFF FFFF (the SNS identifier can be read at address 0x0BF9 6000), and
- The PRODUCT_STATE option bits are not configured to Open.

Workaround

For early samples, the PRODUCT_STATE must remain set to Open.

2.2.9 Flash memory latency is increased during read-while-write (RWW) operation

Description

When the VOS0 and VOS1 voltage scaling is selected, the flash memory zero-wait state maximum frequency is limited to 32 MHz during read-while-write (RWW) operations, that is when reading (instruction fetching or data reading) from one of the flash memory banks (bank1 or bank2) while modifying (writing or erasing) the other bank.

Note: This also applies to option byte and OTP writing (bank1) while reading the bank2.

Workaround

Configure the adequate number of wait states (through the LATENCY[3:0] bitfield of the FLASH_ACR register) before performing an RWW operation.

Note: The initial wait state configuration can be restored after the RWW operation.

2.2.10 Product state regression is not possible when data flash memory is enabled

Description

When flash high-cycle data is enabled, either by setting the EDATA1_EN bit of the FLASH_EDATA1R_PRG register or the EDATA2_EN bit of the FLASH_EDATA1R_PRG register, it is not possible to perform product state regression and change product state to Open.

Workaround

None.

The flash high-cycle data feature must be disabled (EDATA1_EN = 0 and EDATA2_EN = 0) before changing the product state to a state different from Open.

2.2.11 Full JTAG configuration without NJTRST pin cannot be used

Description

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

Workaround

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

2.2.12 BOOT0 pin is sensitive to electrostatic discharge

Description

BOOT0 pin is sensitive to electrostatic discharge (human body model). For all packages, all pins pass the 2000 V except for BOOT0 for which the test fails below 2000 V

Workaround

None.

2.2.13 Flash memory endurance is limited to 1 Kcycle

Description

The flash memory endurance is limited to 1 Kcycle for ambient temperatures (T_A) up to 85 °C, while it should be 100 Kcycle up to 125 °C.

Workaround

None.

2.2.14 SRAM2 is erased when the backup domain is reset

Description

When the VSWRST bit of the RCC_BDCR register (entire VSW domain reset) and the DBP bit of the PWR_DBPCR register (write access to backup domain enabled) are both set, a backup domain reset also erases the SRAM2 content.

Workaround

Apply the following sequence to reset the backup domain and safely erase SRAM2:

1. After the device exits the reset state, check that no SRAM erase is ongoing in *SystemInit()*.
2. Reset the backup domain. This also erases the SRAM2 content.
3. Wait until SRAM2 erase is complete.
4. Proceed with normal application execution, and store data into the SRAM2.

2.2.15 Clearing IWDG_SW might result in debug authentication or ST-iRoT failure

Description

If the IWDG_SW configuration option bit is cleared (independent watchdog controlled by hardware), the IWDG is always enabled after a reset, and cannot be disabled. As a result:

- The ST-DA debug authentication sequence might fail, preventing any possibility to securely reopen the debug or launch regressions on secure products.
- The ST-iROT (immutable root of trust) execution might fail during the boot sequence.

This issue is present only on device revisions lower than ‘X’ (REV_ID < 0x1007).

Workaround

Do not enable the *"IWDG controlled by hardware"* configuration on devices revision lower than 'X'. Instead, use the *"IWDG controlled by software"* configuration (IWDG_SW configuration option bit set).

2.2.16 Clearing WWDG_SW might result in debug authentication or ST-iRoT failure

Description

If the WWDG_SW configuration option bit is cleared (window watchdog controlled by hardware), the WWDG is always enabled after a reset, and cannot be disabled. As a result:

- The ST-DA debug authentication sequence might fail, preventing any possibility to securely reopen the debug or launch regressions on secure products.
- The ST-iROT (immutable root of trust) execution might fail during the boot sequence.

Workaround

Do not enable the *"WWDG controlled by hardware"* configuration. Instead, use the *"WWDG controlled by software"* configuration (WWDG_SW configuration option bit set).

2.2.17 LSE low drive mode is not functional

Description

The LSE oscillator may not start or may stop in low drive mode (LSEDRV = 00). Using this mode is forbidden.

Workaround

None.

2.2.18 Read from flash memory may fail in VOS2 and VOS1 range

Description

When the VOS2 or VOS1 range is selected (VOS[1:0] bits of the PWR_VOSCR register set to 0x01 or 0x02), and the code performs a read from flash memory or is executed from flash memory, the read operation may fail, an ECC double error detected, and an NMI exception generated.

Workaround

Do not use the VOS2 and VOS1 ranges when reading/fetching code from flash memory.

Use only VOS3 or VOS0 when accessing the flash memory. Switching sequence from VOS3 to VOS0 must be performed with the code executed from SRAM to avoid reading the flash memory during the VOS transition.

2.2.19 Partial regression may lead to incorrect device configuration

Description

When the TrustZone® security is enabled (TZEN configuration bits equal to 0xB4), a partial regression (that is changing the product life cycle state from Closed state to TZ-Closed state) may lead to an incorrect device configuration. This issue is present on the system flash security package (SFSP) version 2.4.0 (address 0x0BF9 60CC contains 0x0204 0000).

Workaround

For system flash security package (SFSP) version 2.4.0, do not use partial regression. Instead, use full regression.

This issue is fixed in SFSP version 2.5.0 (address 0x0BF9 60CC contains 0x0205 0000).

2.2.20 Incorrect backup domain reset

Description

The backup domain reset may be missed upon a backup domain power-on following a V_{BAT} power-off in VBAT mode, if the V_{BAT} voltage drops during the power-off phase hitting a window, which is a few mV wide before it starts to rise again. This window is located in the range between 100 mV and 700 mV, the exact position depending mainly on the device and on the temperature.

The missed reset results in unpredictable values of the backup domain registers, which may lead to a wrong device behavior, such as driving the LSCO output pin on PA2, raising an unexpected tamper event preventing the access to SRAM2 and PKA, or influencing any of the backup domain functions.

Workaround

Apply one of the following measures to avoid an incorrect backup domain reset:

- Before performing a new power-on, let the V_{BAT} supply voltage fall to a level below 100 mV for more than 200 ms.
- If none of the previous workarounds can be applied, and the boot follows a backup domain power-on reset, erase the backup domain by software. In order to discriminate the backup domain power-on reset from a power-on reset, at least one backup register (called, for example, BackupTestRegister) must be previously programmed with a BKP_REG_VAL value containing 16 bits set and 16 bits cleared. The robustness of this workaround can be significantly improved by using a CRC rather than registers, since the registers are subject to backup domain reset.
  The workaround consists in calculating the CRC of the backup domain registers, RCC_BDCR and RTC/TAMP registers, excluding the bits modified by hardware. The CRC result can be stored in the backup register, instead of a fixed BKP_REG_VAL value. The CRC result needs to be updated for each modification of values covered by the CRC, for example when the CRC peripheral is used.
  Insert the following software sequence at the very beginning of the boot code:
  1. Check if the BORRSTF flag of the RCC_RSR register is set (the reset is caused by a power-on).
  2. If it is set, check that the BackupTestRegister content is different from BKP_REG_VAL, or that the new CRC calculated value is different from stored results, depending on the chosen workaround implementation.
  3. If this is the case and if no tamper flag is set (when the tamper detection is enabled), the reset is caused by a backup domain power-on. Then apply the following sequence:
     a. Enable backup domain access by setting the DBP bit of the PWR_DBPCR register.
     b. Reset the backup domain by applying the following sequence:
        i. Write 0x0001 0000 to the RCC_BDCR register, which sets the VSWRST bit and clears the other register bits that may not be cleared.
        ii. Read the RCC_BDCR register to make the reset time long enough.
        iii. Write 0x0000 0000 to the RCC_BDCR register to clear the VSWRST bit.
     c. Clear the BORRSTF flag by setting the RMVF bit of the RCC_RSR register.

2.2.21 Debug not available when TrustZone® security is disabled and the PRODUCT_STATE is iROT-Provisioned

Description

When the TrustZone® security is disabled, and the PRODUCT_STATE is iROT-Provisioned, the debug must be available for a code executed from an HDPL 3 area. However, in this case, the code cannot be debugged.

Workaround

If PRODUCT_STATE = iROT-Provisioned, force the debug opening in the first stage of the boot sequence software.

Below an example of code:

```
/* This code ensures that in PRODUCT_STATE = IROT_PROVISIONED, the Debug is opened for HDPL3
when TZEN = disabled. */
 /* This code must be integrated in a HDPL1 code */
if (READ_BIT(FLASH->OPTSR_CUR, FLASH_OPTSR_PRODUCT_STATE_Msk) == OB_PROD_STATE_IROT_PROVISION
ED)
 {
 __HAL_RCC_SBS_CLK_ENABLE();
 ((SBS_TypeDef *)SBS_BASE)->DBGCR = 0x006FB4B4U; // 6F value to open debug when HDPL3 is reac
hed
 }
 /* To be able to attach the debugger, the code to debug must be executed in HDPL3. */
 /* The code must call the below function twice before reaching the code to debug: */
 /* HAL_SBS_IncrementHDPLValue(); */
```

2.2.22 Incorrect flash memory configuration on early STM32H562xG and STM32H563xG devices

Description

1-Mbyte part numbers (STM32H562xG and STM32H563xG with a date code earlier than week 35 2023) are not correctly configured to support the two 512-Kbyte flash memory banks. Instead, the flash memory configuration is set to 2 Mbytes (two 1-Mbyte banks). The incorrect configuration can be identified by reading the bootloader version (at address 0x0BF9 FAFE): it returns the value 0xE4 (2-Mbyte configuration), while it is 0xE5 for the correctly configured part numbers.

As a consequence, if the software size is 1 Mbyte, and no erase/protection operation is performed on Bank 2, there is no functional issue. However, if the software performs an erase or a protection operation on Bank 2, Bank 2 may not behave as expected when the following operations are executed:

- Erasing a Bank 2 sector
- Erasing Bank 2
- Swapping Bank 1 and Bank 2
- Enabling Bank2 protection:
  - Secure and nonsecure watermark-based protection
  - Secure and nonsecure block-based protection
  - Write protection (WRP)

Workaround

None.

The issue is fixed on part numbers with a date code later than week 35 2023.

2.2.23 Reset vector catch feature cannot be used when debugging is disabled for secure code

Description

If the product state (PRODUCT_STATE[7:0] bitfield of the FLASH_OPTSR_CUR register) is different from the open state, debugging is not allowed except by resetting the CPU by mean of a reset vector catch. This may lead to the CPU not halting before executing the first instruction of the nonsecure exception handler.

Consequently, nonsecure code execution may not stop after switching to the nonsecure state, and the CPU may proceed executing code in the nonsecure area.

Workaround

To halt the CPU in the reset handler, insert a software break point in the first nonsecure instruction of the user application. The address of the reset handler can be read from the NSBOOTADD[23:8] bitfield of the FLASH_NSBOOTR_CUR register.

2.2.24 Ethernet alternate functions not available on PB13 and PB14

Description

Early revisions of the datasheet (prior to revision 1) mention that the ETH_MII_TXD1/ETH_RMII_TXD1 and ETH_MII_TX_EN/ETH_RMII_TX_EN Ethernet alternate functions can be configured on PB13 and PB14, whereas these functions are not supported on these I/Os.

This is a documentation issue rather than a product limitation.

Workaround

None

2.3 GPIO

2.3.1 Input hysteresis not properly enabled on PA1

Description

PA1 hysteresis is not properly controlled since it is enabled when and only when PA0 is configured as an input.

Workaround

None.

2.4 FMC

2.4.1 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.

2.4.2 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.

2.4.3 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.

2.5 OCTOSPI

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

Description

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

Workaround

When doing memory-mapped writes, set the DQSE bit of the OCTOSPI_WCCR register, even for memories that have no DQS pin.

2.5.2 TCF set twice on an abort

Description

The TCF (total-count flag) of the OCTOSPI_SR register is set twice when a memory-mapped write is aborted (when the software sets the ABORT bit of the OCTOSPI_CR register), and a memory-mapped burst request is received on the same cycle that the BUSY falls. TCF is set once when BUSY falls, and again when the burst transfer finishes. The burst request gets an error response (HardFault) as expected.

The software does not notice that TCF gets set twice unless the software clears TCF before the burst transfer finishes.

Workaround

Apply one of the following measures:

- Ensure that no memory-mapped write requests are being sent after requesting an abort.
- When a HardFault is generated after a requested abort, ensure TCF is cleared prior to restart transfers.

2.5.3 Deadlock in clock mode 3 when prefetching the 64 upper memory bytes

Description

When CKMOD of the OCTOSPI_DCR1 register is set (clock mode 3 is enabled), a deadlock may occur if a memory-mapped request to a non-sequential address is received one memory clock-cycle after an access (normal or prefetch) to any of the 64 upper memory bytes.

Workaround

Apply one of the following measures:

- Recommended: Use clock mode 0, by setting the bit CKMODE of the OCTOSPI_DCR1 register (all memories known to date support clock mode 0).
- In Memory-mapped mode, do not access the 64 upper memory bytes.
- Set DEVSIZE bitfield of the OCTOSPI_DCR1 register to allocate more space than the actual size of the memory.

2.5.4 Memory wrap instruction not enabled when DQS is disabled

Description

Memory wrap instruction (as configured in the OCTOSPI_WPxxx registers) is not generated when DQS is disabled. The memory wrap instruction is replaced by two regular successive read instructions to ensure the correct data ordering: this split has very limited impact on performance.

Workaround

None.

2.5.5 Deadlock or write-data corruption after spurious write to a misaligned address in OCTOSPI_AR register

Description

Upon writing a misaligned address to OCTOSPI_AR just before switching to Memory-mapped mode (without first triggering the indirect write operation), with the OCTOSPI configured as follows:

- FMODE = 00 in OCTOSPI_CR (Indirect write mode)
- DQSE = 1 in OCTOSPI_CCR (DQS active)

then, the OCTOSPI may be deadlocked on the first memory-mapped request or the first memory-mapped write to memory (and any sequential writes after it) may be corrupted.

An address is misaligned if:

- the address is odd and the OCTOSPI is configured to send two bytes of data to the memory every cycle (octal-DTR mode or dual-quad-DTR mode), or
- the address is not a multiple of four when the OCTOSPI is configured to send four bytes of data to the memory (16-bit DTR mode or dual-octal DTR mode).

If the OCTOSPI_AR register is reprogrammed with an aligned address (without triggering the indirect write between the two writes to OCTOSPI register), the data sent to the memory during the indirect write operation are also corrupted.

Workaround

None.

2.5.6 Deadlock on consecutive out-of-range memory-mapped write operations

Description

The DEVSIZE[4:0] bitfield of the OCTOSPI_DCR1 register indicates that the size of the memory is 2 ^ [DEVSIZE + 1] bytes, and thus any memory-mapped access to address 2 ^ [DEVSIZE + 1] or above should get an error response.

However, no error response may be returned and the OCTOSPI may become deadlocked after the following sequence of events:

1. A memory-mapped write operation is ongoing on the AHB bus.
2. A second memory-mapped write is requested to an address close to the end of the memory but not consecutive to the address targeted by the first write operation.
3. A third memory-mapped write operation is requested, this time to an address consecutive to the address targeted by the second write, and the address of this third write is 2 ^ [DEVSIZE + 1] or an address consecutive to 2 ^ [DEVSIZE + 1].
   If the first write command has not completed writing data, then the write to 2 ^ [DEVSIZE + 1] does not return any error response and the next memory-mapped request gets stalled indefinitely.

Workaround

Ensure that no sequences of consecutive memory-mapped write operations pass the memory boundary.

2.5.7 Indirect write mode limited to 256 Mbytes

Description

In indirect write mode, if the address is greater than 256 Mbytes, the indirect write is not performed at the targeted address, even if it is located inside the allowed memory space configured through the device size (DEVSIZE[4:0] of OCTOSPI_DCR1). Actually, this write operation takes place within the 256-Mbyte memory space, thus corrupting the memory content.

Indirect read operations are not impacted.

Workaround

Indirect write operations have to be performed inside the first 256 Mbytes of the memory space.

2.5.8 Read-modify-write operation does not clear the MSEL bit

Description

When the MSEL bit of the OCTOSPI_CR register is set, it remains set even if the software attempts to clear it by performing a read-modify-write operation.

Workaround

To clear the MSEL bit, clear in a single write access bit 7 and bit 30 of the OCTOSPI_CR register, otherwise, the MSEL bit remains set.

2.6 ADC

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

Description

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

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

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

Workaround

Apply one of the following measures:

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

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

Description

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

This issue may happen if:

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

Workaround

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

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

Description

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

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

Workaround

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

2.6.4 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.

2.6.5 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).

2.6.6 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.

2.7 VREFBUF

2.7.1 Slow discharge of the capacitor connected to VREF+ pin

Description

The pull-down resistor on VREFBUF output is missing, thus causing a slow discharge of the capacitor on the VREF+ pin.

Workaround

When switching down the VREFBUF output, wait for a sufficient time to discharge of the capacitor connected to VREF+ pin.

2.8 TIM

2.8.1 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.

2.8.2 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).

2.8.3 Timer connection to USB SOF might not work properly

Description

USB SOF signal might not be properly synchronized with TIM2 and TIM5 ITR12.

Workaround

Connect externally USB SOF (PA8) and the timer ETR input pin.

2.9 LPTIM

2.9.1 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.

2.9.2 ARRM and CMPM flags are not set when APB clock is slower than kernel clock

Description

When LPTIM is configured in one shot mode and APB clock is lower than kernel clock, there is a chance that ARRM and CMPM flags are not set at the end of the counting cycle defined by the repetition value REP[7:0]. This issue can only occur when the repetition counter is configured with an odd repetition value.

Workaround

To avoid this issue the following formula must be respected:

{ARR, CMP} ≥ KER_CLK / (2* APB_CLK),

where APB_CLK is the LPTIM APB clock frequency, and KER_CLK is the LPTIM kernel clock frequency. ARR and CMP are expressed in decimal value.

Example: The following example illustrates a configuration where the issue can occur:

- APB clock source (MSI) = 1 MHz , Kernel clock source (HSI) = 16 MHz
- Repetition counter is set with REP[7:0] = 0x3 (odd value)

The above example is subject to issue, unless the user respects:

{CMP, ARR} ≥ 16 MHz / (2 * 1 MHz)

→ ARR must be ≥ 8 and CMP must be ≥ 8

Note: REP set to 0x3 means that effective repetition is REP+1 (= 4) but the user must consider the parity of the value loaded in LPTIM_RCR register (=3, odd) to assess the risk of issue.

2.9.3 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.

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

Description

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

Workaround

None.

2.9.5 Overcapture stops working when CCxOF flag is cleared in some specific conditions

Description

The overcapture stops working if the CCxOF flag in the LPTIMx_ISR register is cleared simultaneously with a new input capture event detection, that is, when an input capture pulse is active.

As a result, the LPTIM does not detect anymore overcapture events, no interrupt is generated, and the CCxOF flag is not set.

Workaround

Disable the corresponding input capture channel (the CCxE bit of the LPTIM_CCMRx register cleared) immediately after clearing the CCxOF flag, then enable the channel again after a delay that must be equal or greater than the value of (PRESC * 3) kernel clock cycles, PRESC[2:0] being the clock decimal division factor (1, 2, 4,..128).

2.9.6 PLL2 output cannot be used as LPTIM clock source

Description

When pll2_p_ck is selected as LPTIMx clock source by setting LPTIMxSEL[2:0] bitfield of the RCC_CCIPR2 register to 0b001, LPTIMx does not receive its kernel clock, and does not properly operate.

Workaround

Avoid selecting pll2_p_ck as clock source for LPTIMx.

Select any other clock source by programming LPTIMxSEL[2:0] to a value different from 0b001. Refer to the device reference manual for the list of possible values.

2.10 IWDG

2.10.1 Independent watchdog does not wake up the system from Stop mode

Description

The independent watchdog early wakeup interrupt does not wake up the system from Stop mode.

Workaround

None.

2.11 RTC and TAMP

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

Description

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

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

Workaround

None.

2.11.2 Tamper event 15 raised when a system reset occurs

Description

Tamper event 15 might be raised when a system reset occurs, resulting in a nonfunctional TAMP15.

Workaround

None.

2.11.3 Timestamp flag unexpectedly raised when disabling timestamp

Description

The TSF flag of RTC_SR is wrongly set when disabling the timestamp. This issue occurs when the following conditions are met:

- timestamp negative edge detection is requested, and
- no edge has occurred

The other detection configurations are not impacted.

Workaround

After clearing the TSE bit of the RTC_CR register:

- in polling mode, wait for 7 ms to allow the TSF flag to be set. Then clear it.
- in interrupt mode: clear the TSF flag as soon as it is set.

2.12 I2C

2.12.1 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.

2.12.2 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.

2.12.3 SDA held low upon SMBus timeout expiry in slave mode

Description

For the slave mode, the SMBus specification defines $t_{TIMEOUT}$ (detect clock low timeout) and $t_{LOW:SEXT}$ (cumulative clock low extend time) timeouts. When one of them expires while the I2C peripheral in slave mode drives SDA low to acknowledge either its address or a data transmitted by the master, the device is expected to report such an expiry and release the SDA line.

However, although the device duly reports the timeout expiry, it fails to release SDA. This stalls the I^2C bus and prevents the master from generating RESTART or STOP condition.

Workaround

When a timeout is reported in slave mode (TIMEOUT bit of the I2C_ISR register is set), apply this sequence:

1. Wait until the frame is expected to end.
2. Read the STOPF bit of the I2C_ISR register. If it is low, reset the I2C kernel by clearing the PE bit of the I2C_CR1 register.
3. Wait for at least three APB clock cycles before enabling again the I2C peripheral.

2.13 I3C

2.13.1 I3C controller: unexpected read data bytes during a legacy I^2C read

Description

Under specific conditions, unexpected data bytes are read during a legacy I^2C read transfer.

The issue occurs when all the following conditions are met:

- I3C acts as controller
- a legacy I^2C read message is generated
- the STALLT bit of I3C_TIMINGR2 register is set to request the SCL clock to be stalled at low level on the 9th T-bit phase of data bytes (also known as ACK/NACK phase)
- instead of releasing the SDA line, the I^2C target incorrectly drives SDA low on the 9th T-bit phase of the end of read from the I3C controller

To end a legacy I^2C read, the I3C controller is supposed not to drive SDA low on the 9th T-bit, and to emit a NACK. If the STALLT bit of I3C_TIMINGR2 is set, the controller does not NACK for the purpose of ending the data read transfer.

During the same clock cycle, if the I^2C target, instead of releasing the SDA line, incorrectly drives SDA low on this 9th T-bit phase of the end of read from the controller, then the controller detects an incorrect ACK on the I3C bus and keeps SCL clock running.

After 8 clock cycles, the I3C controller generates again an ACK instead of a NACK, and an unexpected dummy data byte is transferred to the RX-FIFO.

Then the target continues transferring data or releases the SDA line, thus causing additional dummy bytes to be received. The transfer can be stopped only when an overrun error occurs.

Workaround

Apply the following measures:

- If the I3C controller is configured with S-FIFO mode enabled (SMODE bit set in I3C_CFGR), the transfer goes on until RX-FIFO is full. Then ERRF = 1 in I3C_EVR (an error occurred), PERR = 1 in I3C_SER (protocol error), DOVR = 1 in I3C_SER (RX-FIFO overrun), and CODERR[3:0] = 001 in I3C_SER (CE1 error).
  It is recommended to enable the error interrupt by setting ERRIE in I3C_IER. When DOVR = 1 and CODERR[3:0] = 0001, flush the RX-FIFO inside the error interrupt service routine by setting RXFLUSH in I3C_CFGR, then clear the CERRF error flag.
- If the I3C controller is configured with S-FIFO mode disabled (SMODE bit cleared in I3C_CFGR), the I3C status register (I3C_SR) may be overwritten by the hardware if unread, thus failing to report any status overrun. An overrun can occur only as a data overrun if the DMA or the software stops reading the RX-FIFO during enough time for the RX-FIFO to be full with dummy bytes. Then both CE1 and DOVR flags are set and an error is reported (ERRF = 1, PERR = 1, CODERR[3:0] = 0001 and DOVR = 1).

Whatever S-FIFO configuration, implement a software timeout to inform that neither FCF nor ERRF error bit was raised during an acceptable time. Then, stop reading RX-FIFO to cause a data overrun to be reported. When an error is reported, if both CE1 and DOVR flags are set (ERRF = 1, PERR = 1, CODERR[3:0] = 0001 and DOVR = 1), flush the RX-FIFO.

2.13.2 I3C controller: SCL clock is not stalled during address ACK/NACK phase following a frame start, when enabled through I3C_TIMINGR2 register

Description

Under specific conditions, the I3C controller does not stall the SCL clock during the address ACK/NACK phase when this feature is configured through I3C_TIMINGR2 register.

The issue occurs when all the following conditions are met:

- I3C acts as controller
- I3C is programmed to stall the SCL clock low during the address ACK/NACK phase (STALLA bit of I3C_TIMINGR2 set to 1 and STALL[7:0] bitfield of I3C_TIMINGR2 set to a non-null value)
- the address emitted by the controller follows a frame start and not a repeated start

The purpose of this programmed SCL clock stall time is to add an additional duration for the I3C target(s) to respond on the address ACK/NACK phase. However, the SCL clock is not stalled on this address ACK/NACK phase.

Workaround

Set NOARBH = 0 in I3C_CFGR in order to insert the arbitrable header between the frame start and the emitted address.

If the I^2C/I3C target has still not enough time to respond to the emitted static/dynamic address, increase the SCL low duration for any open-drain phase by increasing SCLL_OD[7:0] value in I3C_TIMINGR0.

2.13.3 I3C controller: unexpected first frame with a 0x7F address when the I3C peripheral is enabled

Description

After I3C has been initialized as controller, an unexpected frame is generated when the I3C peripheral is enabled. The issue occurs after the following sequence:

1. I3C is initialized as I3C controller (CRINIT bit is set in I3C_CFGR whereas EN bit is kept cleared in I3C_CFGR).
2. I3C is enabled (EN bit set in I3C_CFGR).

As a result, the I3C controller can incorrectly detect that the SDA line has been driven low by a target, interpret it as a start request, activate the SCL clock, and generate a 0x7F address followed by RNW bit = 1 that is not acknowledged.

This first frame completes without any other impact than this unexpected I3C bus activity.

Workaround

Respect the sequence below during I3C controller initialization:

1. Instead of configuring the alternate GPIO of the SDA line without any pull-up, temporary enable the GPIO pull-up.
2. After a delay of 1 ms, disable GPIO pull-up.
3. Initialize I3C as I3C controller by setting CRINIT in I3C_CFGR whereas EN bit is kept cleared in I3C_CFGR.
4. Enable I3C by setting EN bit in I3C_CFGR.

As a result the I3C controller does not detect SDA low when it is enabled, and no unexpected frame is generated.

2.13.4 I3C controller: no timestamp on IBI acknowledge when timing control is used in Asynchronous mode 0

Description

When I3C acts as controller, it cannot provide a timestamp on an IBI acknowledge (named C_REF in MIPI I3C v1.1 specification).

As a result, when timing control is used in Asynchronous mode 0, the controller software cannot calculate the timestamp of the sampled data of the target(s) following a received and acknowledged IBI using payload data for timing control (T_C1 and T_C2) (see MIPI formula: $\mathrm{C\_TS} = \mathrm{C\_REF} - \mathrm{C\_C2} \times \mathrm{T\_C1}/\mathrm{T\_C2}$), despite the fact that the controller software can compute the duration C_C2 by using the formula:

$$\mathrm{C\_C2} = 9 \times (\mathrm{I3C\_TIMINGR0.SCLL\_PP}[7:0] + 1 + \mathrm{I3C\_TIMINGR0.SCLH\_I3C}[7:0] + 1) \times T_{I3CCLK}$$

When operating in Asynchronous mode 0, the sampled data received from the target(s) cannot be associated with a computed timestamp, and on controller side, they can not be time-correlated.

Workaround

Follow the sequence below:

1. Allocate an available product timer by software and approximate the IBI acknowledge moment by when the timer is notified by an interrupt of a received and complete IBI.
2. Program a broadcast/direct SETXTIME CCC with subcommand byte 0xDF to enter Asynchronous mode 0.
3. After being notified of the command completion by the flag and/or the related interrupt (FCF flag is set in I3C_EVR), reset and enable the timer to start the counter.
4. After being notified that an IBI is complete by the flag and/or the related interrupt (IBIF flag is set in I3C_EVR), read the value of the timer as C_TIM. The timestamp of the sampled data can then be approximated by using the formula:

   $$\mathrm{C\_TS} = \mathrm{C\_TIM} - \mathrm{C\_C2} \times (\mathrm{T\_C1}/\mathrm{T\_C2} + 4)$$

   knowing that

   $$\mathrm{C\_C2} = 9 \times (\mathrm{I3C\_TIMINGR0.SCLL\_PP}[7:0] + 1 + \mathrm{I3C\_TIMINGR0.SCLH\_I3C}[7:0] + 1) \times \mathrm{T}_{I3CCLK}$$

   and that the IBI is complete after a 4-byte payload.
5. Generate a broadcast/direct SETXTIME CCC with subcommand byte 0xFF to exit Asynchronous mode 0 to disable/deallocate the timer resource.

2.14 USART

2.14.1 USART does not generate DMA requests after setting/clearing DMAT bit

Description

If the DMA is used for data transmission (DMAT = 1 in USART_CR3 register), and the software clears DMAT bit and sets it again to prepare the next transmission, then the peripheral does not generate DMA requests anymore. As a result, data are not transmitted.

Workaround

- Avoid clearing DMAT.
- If clearing DMAT is needed after the end of DMA transfers, once DMAT is cleared, disable and reenable the peripheral through UE bit of USART_CR1 register. This workaround is acceptable only if the peripheral is not used in receiver mode.
- DMAT can be cleared if the next transmission is based on polling/interrupt.

2.14.2 Wrong data received in smartcard mode and 0.5 stop bit configuration

Description

The USART receiver reads wrong data in smartcard mode and 0.5 stop bit configuration.

Workaround

Use the 1.5 stop bit configuration.

2.15 LPUART

2.15.1 LPUART does not generate DMA requests after setting/clearing DMAT bit

Description

If the DMA is used for data transmission (DMAT = 1 in LPUART_CR3 register), and the software clears DMAT bit and sets it again to prepare the next transmission, then the peripheral does not generate DMA requests anymore. As a result, data are not transmitted.

Workaround

- Avoid clearing DMAT.
- If clearing DMAT is needed after the end of DMA transfers, once DMAT is cleared, disable and reenable the peripheral through UE bit of LPUART_CR1 register. This workaround is acceptable only if the peripheral is not used in receiver mode.
- DMAT can be cleared if the next transmission is based on polling/interrupt.

2.15.2 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.

2.16 SPI

2.16.1 Truncation of SPI output signals after EOT event

Description

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

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

Workaround

Apply one of the following measures or their combination:

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

2.17 FDCAN

2.17.1 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.

2.17.2 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
```

2.18 USB

2.18.1 COUNTn_RX[9:0] bitfield reporting one byte less if APB frequency is below 12.6 MHz

Description

The USB interface is expected to correctly operate down to a minimum APB frequency of 8 MHz. Under certain conditions, the packet from the host is correctly received and stored in the packet buffer memory, but the COUNTn_RX[9:0] bitfield indicating the number of received bytes incorrectly reports one byte less than the actual packet size. This may occur when:

- USB out transactions are operated at APB frequency less than 12.6 MHz, and
- Data payload size sent by the host exactly matches the maximum packet size (MPS) programmed into COUNTn_RX[9:0].

Workaround

Increase PCLK frequency to at least 12.6 MHz, or program USB_COUNTn_RX to allocate for reception at least one byte more than the value of MPS in the packet buffer memory.

2.18.2 False wakeup detection for last K-state not terminated by EOP or reset before suspend

Description

If a USB K-state is detected outside of a suspend state, the device generates *RESUME RECEIVED* signal and keeps it pending until:

- an EOP is detected on the USB, or
- a USB reset is detected on the USB, or
- a software reset is generated by setting the FRES bit.

It may happen that none of the above events occurs before the USB peripheral sets the suspend interrupt (after 3 ms idling), and the software decides to enter Suspend mode (by setting the FSUSP bit). The USB peripheral then immediately generates a wakeup interrupt (WKUP = 1) because it still detects the pending *RESUME RECEIVED* signal. This has a negative impact to the device power consumption.

A workaround is available for applications requiring to minimize the power consumption.

Workaround

Apply software reset prior to setting the FSUSP bit. This clears any spurious *RESUME RECEIVED* condition.

2.18.3 ESOF interrupt timing desynchronized after resume signaling

Description

Upon signaling resume, the device is expected to allow full 3 ms of time to the host or hub for sending the initial SOF (start of frame) packet, without triggering SUSP interrupt. However, the device only allows two full milliseconds and unduly triggers SUSP interrupt if it receives the initial packet within the third millisecond.

Workaround

When the device initiates resume (remote wakeup), mask the SUSP interrupt by setting the SUSPM bit for 3 ms, then unmask it by clearing SUSPM.

2.18.4 Incorrect CRC16 in the memory buffer

Description

Memory buffer locations are written starting from the address contained in the ADDRn_RX for a number of bytes corresponding to the received data packet length, CRC16 inclusive (that is, data payload length plus two bytes), or up to the last allocated memory location defined by BL_SIZE and NUM_BLOCK, whichever comes first. In the former case, the CRC16 checksum is written wrongly, with its least significant byte going to both memory buffer byte locations expected to receive the least and the most significant bytes of the checksum.

Although the checksum written in the memory buffer is wrong, the underlying CRC checking mechanism in the USB peripheral is fully functional.

Workaround

Ignore the CRC16 data in the memory buffer.

2.19 ETH

2.19.1 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.

2.19.2 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.

2.19.3 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.

2.19.4 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.

2.19.5 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.

2.19.6 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.

2.19.7 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.

2.19.8 Spurious receive watchdog timeout interrupt

Description

Setting the RWTU[1:0] bitfield of the ETH_DMACRXIWTR 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.

2.19.9 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.

2.19.10 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 ETH_MTLRXQOMR register. Accept packets of transactions flagging both dribble and CRC errors.

2.19.11 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.

2.20 CEC

2.20.1 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.

2.20.2 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 5. **Document revision history**

Date	Version	Changes
23-Feb-2023	1	Initial release.
01-Mar-2023	2	Added STM32H563AG part number. Added Clearing IWDG_SW might result in debug authentication or ST-iRoT failure and Clearing WWDG_SW might result in debug authentication or ST-iRoT failure errata.
22-Jun-2023	3	Added STM32H562xG part numbers in Table 1. Device summary. Updated LSE crystal oscillator may be disturbed by transitions on PC13 and changed status to 'N' for rev Z and X. Added LSE low drive mode is not functional, Read from flash memory may fail in VOS2 and VOS1 range, Partial regression may lead to incorrect device configuration, Incorrect backup domain reset, and Debug not available when TrustZone® security is disabled and the PRODUCT_STATE is iROT-Provisioned.
23-Nov-2023	4	Table 3. Summary of device limitations: • Updated Peripheral triggers connected to RTC wakeup timer interrupt instead of RTC wakeup timer trigger signal status for revisions Z and X • Updated Clearing IWDG_SW might result in debug authentication or ST-iRoT failure status for revision X SYSTEM errata: added Incorrect flash memory configuration on early STM32H562xG and STM32H563xG devices, Reset vector catch feature cannot be used when debugging is disabled for secure code and Ethernet alternate functions not available on PB13 and PB14. OCTOSPI errata: added Memory-mapped write error response when DQS output is disabled. CEC errata: added Missed CEC messages in normal receiving mode and Unexpected TXERR flag during a message transmission.

Contents