STM32H503CB/EB/KB/RB

Errata sheet

STM32H503CB/EB/KB/RB device errata

Applicability

This document applies to the part numbers of STM32H503CB/EB/KB/RB 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 RM0492.

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
STM32H503CB/EB/KB/RB	STM32H503CB, STM32H503EB, STM32H503KB, STM32H503RB

Table 2. **Device variants**

Reference	Silicon revision codes	
	Device marking[(1)]	REV_ID[(2)]
STM32H503CB/EB/KB/RB	A	0x1000
	Z	0x1001
	Y	0x1002

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 STM32H503CB/EB/KB/RB 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. Y
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	Additional consumption on PA11/PA12 in Stop and Standby modes	A	-	-
	2.2.5	LSE low drive mode is not functional	N	N	N
	2.2.6	Flash memory latency is increased during read-while-write (RWW) operation	A	A	-
	2.2.7	Full JTAG configuration without NJTRST pin cannot be used	A	A	A
	2.2.8	SRAM2 unduly erased upon a backup domain reset	A	A	A
	2.2.9	PLL2 output cannot be used as LPTIM clock source	A	A	A
	2.2.10	Clearing IWDG_SW might result in debug authentication failure	A	A	-
	2.2.11	Clearing WWDG_SW might result in debug authentication failure	A	A	A
	2.2.12	Corrupted content of the backup domain due to a missed power-on reset after this domain supply voltage drop	A	A	A
	2.2.13	Read from flash memory may fail in VOS2 and VOS1 range	A	A	-
	2.2.14	Debug not available when the PRODUCT_STATE is iROT-Provisioned	A	A	A
	2.2.16	Low-speed external clock in analog bypass mode might not work properly	A	A	A
	2.2.17	Incorrect backup domain reset	A	A	A
	2.2.19	PLL1P output can unduly be disabled by software when used as SYSCLK clock	A	A	A
	2.2.21	Invalid DAC output voltage for several DAC kernel clocks	A	A	A
	2.2.22	LPTIM2_CH1 might prevent the system from entering low-power modes	A	A	A
ADC	2.3.1	New context conversion initiated without waiting for trigger when writing new context in ADC_JSQR with JQDIS = 0 and JQM = 0	A	A	A
	2.3.2	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

Function	Section	Limitation	Status		
			Rev. A	Rev. Z	Rev. Y
ADC	2.3.3	ADC_AWDy_OUT reset by non-guarded channels	A	A	A
	2.3.4	Injected data stored in the wrong ADC_JDRx registers	A	A	A
COMP	2.4.1	The Comparator cannot wakeup the device from low power mode in some specific conditions	N	-	-
	2.4.2	COMP input PA0 interfering with PB0, PB2, or DAC	A	A	A
TIM	2.5.1	Bidirectional break mode not working with short pulses	N	N	N
LPTIM	2.6.1	Device may remain stuck in LPTIM interrupt when entering Stop mode	A	A	A
	2.6.2	ARRM and CMPM flags are not set when APB clock is slower than kernel clock	A	A	A
	2.6.3	Interrupt status flag is cleared by hardware upon writing its corresponding bit in LPTIM_DIER register	N	N	N
	2.6.4	Overcapture stops working when CCxOF flag is cleared in some specific conditions	P	-	-
IWDG	2.7.1	Independent watchdog does not wake up the system from Stop mode	N	N	N
RTC and TAMP	2.8.1	Alarm flag may be repeatedly set when the core is stopped in debug	N	N	N
	2.8.3	Timestamp flag unexpectedly raised when disabling timestamp	A	A	A
	2.8.2	Tamper event 15 raised when a system reset occurs	N	-	-
I2C	2.9.1	Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period	P	P	P
	2.9.2	Spurious bus error detection in master mode	A	A	A
	2.9.3	SDA held low upon SMBus timeout expiry in slave mode	A	A	A
I3C	2.10.1	I3C controller: unexpected read data bytes during a legacy I^2C read	A	A	A
	2.10.2	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
	2.10.3	I3C controller: unexpected first frame with a 0x7F address when the I3C peripheral is enabled	A	A	A
	2.10.4	I3C controller: no timestamp on IBI acknowledge when timing control is used in Asynchronous mode 0	A	A	A
USART	2.11.1	Data corruption due to noisy receive line	-	A	A
	2.11.2	USART does not generate DMA requests after setting/clearing DMAT bit	A	A	A
	2.11.3	Wrong data received in smartcard mode and 0.5 stop bit configuration	A	-	-
	2.11.4	Received data may be corrupted upon clearing the ABREN bit	A	A	A
	2.11.5	Noise error flag set while ONEBIT is set	N	N	N
LPUART	2.12.1	LPUART does not generate DMA requests after setting/clearing DMAT bit	A	A	A
	2.12.2	Possible LPUART transmitter issue when using low BRR[15:0] value	P	P	P
SPI	2.13.1	RDY output failure at high serial clock frequency	N	N	N
	2.13.2	Truncation of SPI output signals after EOT event	A	A	A
	2.13.3	TIFRE flag wrongly set in slave PCM long frame mode if FIXCH = 1	N	N	N
	2.13.4	TIFRE flag never set in slave PCM/I2S mode if FIXCH = 0	N	N	N
FDCAN	2.14.1	Desynchronization under specific condition with edge filtering enabled	A	A	A

Function	Section	Limitation	Status		
			Rev. A	Rev. Z	Rev. Y
FDCAN	2.14.2	Tx FIFO messages inverted under specific buffer usage and priority setting	A	A	A
USB	2.15.1	Buffer description table update completes after CTR interrupt triggers	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	2.2.15	SRAM ECC error flags and addresses are updated only if interrupt is enabled
	2.2.18	Incorrect description of HDPx_STRT and HDPx_END default values
	2.2.20	Incorrect description of EXTI line 53

2 Description of device errata

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

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

2.1 Core

Reference manual and errata notice for the Arm® Cortex®-M33 core revision r0p4 is available from http://infocenter.arm.com.

2.1.1 Access permission faults are prioritized over unaligned Device memory faults

Description

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

The failure occurs when the MPU is enabled and:

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

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

Workaround

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

2.2 System

2.2.1 LSE crystal oscillator may be disturbed by transitions on PC13

Description

On LQFP and VFQFPN packages, the LSE crystal oscillator clock frequency can be incorrect when PC13 is toggling in input or output (for example when used for RTC_OUT1). The external clock input (LSE bypass) is not impacted by this limitation.

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

Workaround

None.

Avoid toggling PC13 when LSE is used on LQFP and VFQFPN.

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

Description

GPDMA1 and GPDMA2 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 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.

2.2.5 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.

2.2.6 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.*

2.2.7 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.

2.2.8 SRAM2 unduly erased upon a backup domain reset

Description

The software reset of the backup domain unduly erases SRAM2.

Note: To reset the backup domain, the software sets the VSWRST bit of the RCC_BDCR register and the DBP bit of the PWR_DBPCR register.

Workaround

Store SRAM2 data before executing the software reset of the backup domain.

2.2.9 PLL2 output cannot be used as LPTIM clock source

Description

When PLL2P (pll2_p_ck) is selected as LPTIM clock source, LPTIM does not receive its kernel clock and it does not operate properly.

Note: The value of the LPTIMxSEL[2:0] bitfield that selects PLL2P is 001.

Workaround

Set LPTIMxSEL[2:0] to a value other than 001, to avoid selecting PLL2P as LPTIM clock source.

2.2.10 Clearing IWDG_SW might result in debug authentication 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 launch regressions on secure products.

Workaround

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

2.2.11 Clearing WWDG_SW might result in debug authentication 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 launch regressions on secure products.

Workaround

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

2.2.12 Corrupted content of the backup domain due to a missed power-on reset after this domain supply voltage drop

Description

The backup domain reset may be missed upon a power-on following a power-off, if its supply voltage drops during the power-off phase hitting a window, which is few mV wide before it starts to rise again. In this critical window, the flip-flops are no longer able to safely retain the information and the backup domain reset has not yet been triggered. This window is located in the range between 100 mV and 700 mV, with the exact position depending mainly on the device and on the temperature.

This missed reset results in unpredictable values of the backup domain registers. This may cause a spurious behavior (such as driving the LSCO output pin on PB2 or influencing backup functions).

Workaround

Apply one of the following measures:

- In the application, let the V_{DD} and V_{BAT} supply voltages fall to a level below 100 mV for more than 200 ms before a new power-on.
- If the above workaround cannot be applied, and the boot follows a power-on reset, erase the backup domain by software.
  When the application is using shutdown mode, user needs to discriminate between the power-on reset or an exit from a shutdown mode.
  For this purpose, at least one backup register must have been previously programmed with a BKP_REG_VAL value with 16 bits set and 16 bits cleared.
  Robustness of this workaround can be significantly improved by using a CRC rather than registers. The registers are subject to backup domain reset.
  The workaround consists of calculating the CRC of the backup registers: RCC_BDCR and RTC registers, excluding bits modified by HW.
  The CRC result can be stored in the backup register instead of a fixed value. This value needs to be updated for each modification of values covered by CRC, such as by using CRC peripheral.
  At the very beginning of the boot code, insert the following software sequence:
  1. Check the BORRSTF flag of the RCC_CSR register. If set, the reset is caused by a power on, or is exiting from shutdown mode.
  2. If BORRSTF flag is true, and the shutdown mode is used in the application, check that the backup register value is different from BKP_REG_VAL. When tamper detection is enabled,check that no tamper flag is set. If both conditions are met then the reset is caused by a power-on.
  3. If the reset is caused by a power-on, apply the following sequence:
     a. Enable the PWR clock in the RCC, by setting the PWREN bit.
     b. Enable the backup domain access in the PWR, by setting the DBP bit.
     c. Reset the backup domain, by:
        i. Writing 0x0001 0000 in the RCC_BDCR register, which sets the BDRST bit and clears other register bits that might not be reset.
        ii. reading the RCC_BDCR register, to make the reset time long enough
        iii. writing 0x0000 0000 in the RCC_BDCR register, to clear the BDRST bit
     d. Clear the BORRSTF flag by setting the RMVF bit of the RCC_CSR register.

2.2.13 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.

2.2.14 Debug not available when the PRODUCT_STATE is iROT-Provisioned

Description

When 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
*/
 /* 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(); */
```

2.2.15 SRAM ECC error flags and addresses are updated only if interrupt is enabled

Description

Some reference manual revisions wrongly state that:

- If a single error is detected, SEDC and CSEDC flags are set in RAMCFG_MxISR and RAMCFG_MxICR respectively. The ECC single error address is updated in RAMCFG_MxSEAR.
- If a double error is detected, DED and CDED flags are set in RAMCFG_MxISR and RAMCFG_MxICR respectively. The ECC double error address is updated in RAMCFG_MxDEAR.

However, the real behavior is that:

- If a single error is detected, SEDC and CSEDC flags are set, and the associated ECC single error address is updated only if the single error interrupt is enabled (SEIE bit of RAMCFG_MxIER is set).
- If a double error is detected, DED and CDED flags are set, and the associated ECC double error address is updated only if double error interrupt or NMI is enabled (DEIE bit or ECCNMI bit of RAMCFG_MxIER is set).

This is a documentation error rather than a device limitation.

Workaround

When the application needs to get the ECC error flags and addresses status without servicing the associated interrupts, the RAMCFG SEIE/DEIE interrupts must be enabled with the associated NVIC RAMCFG vector 5 disabled.

2.2.16 Low-speed external clock in analog bypass mode might not work properly

Description

While the LSE bypass in analog mode is selected (LSEBYP = 1 and LSEEXT = 0 in the RCC_BDCR register), the LSE clock might not work properly.

Workaround

Do not use the LSE bypass in analog mode. Instead, use the digital LSE bypass mode (LSEBYP = 1 and LSEEXT = 1 in the RCC_BDCR register).

2.2.17 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.

2.2.18 Incorrect description of HDPx_STRT and HDPx_END default values

Description

The reference manual incorrectly describes the default values for HDPx_STRT and HDPx_END:

- HDPx_STRT is listed as `0x01`, and HDPx_END is listed as `0x00`.

However, the correct default values are:

- HDPx_STRT is `0x00`, and HDPx_END is `0x01`.

This is a documentation error rather than a device limitation.

Workaround

None.

2.2.19 PLL1P output can unduly be disabled by software when used as SYSCLK clock

Description

The PLL1P output (pll1_p_ck clock signal) is expected to be protected against disabling by software while selected for SYSCLK (sys_ck clock signal). Unduly, it is on the PLL1Q output that this protection acts, instead on PLL1P. As a consequence, the PLL1P output selected as SYSCLK can be disabled by software, which leads to a system deadlock.

Note: The PLL1P output (pll1_p_ck clock signal) is selected as SYSCLK (sys_ck clock signal) by setting the SW[1:0] bitfield of the RCC_CFGR1 register to 0b11. The PLL1P and PLL1Q outputs of the PLL1 are enabled and disabled through, respectively, the bits PLL1PEN and PLL1QEN of the RCC_PLL1CFGR register.

Workaround

Do not clear the PLL1PEN bit while the PLL1P output is used as SYSCLK clock.

2.2.20 Incorrect description of EXTI line 53

Description

The reference manual incorrectly states that the V_{DDIO2} voltage monitor is mapped to EXTI line 53.

However, the correct description is that EXTI line 53 is Reserved.

This is a documentation error rather than a device limitation.

Workaround

None.

2.2.21 Invalid DAC output voltage for several DAC kernel clocks

Description

The DAC output voltage might be incorrect when the DAC kernel clock is different from HCLK (ADCDACSEL[2:0] bits equal to 000 in RCC_CCIPR5 register) or SYSCLK (ADCDACSEL[2:0] bits equal to 001 in RCC_CCIPR5 register).

Workaround

When DAC is used, use ADCDACSEL[2:0] = 000 or 001 in the RCC_CCIPR5 register.

2.2.22 LPTIM2_CH1 might prevent the system from entering low-power modes

Description

When LPTIM2_CH1 output is enabled, it might lead to a wake-up interrupt request from EXTI input event 49, as the default value for an IMR49 bit in the EXTI_IMR2 register is equal to 1 (EXTI interrupt input 49 is unmasked). Consequently, it might prevent the system from entering low-power modes.

Workaround

When LPTIM2_CH1 output is enabled, and the system needs to enter low-power mode, mask EXTI interrupt input 49 by setting the IMR49 bit in the EXTI_IMR2 register to 0.

2.3 ADC

2.3.1 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.

2.3.2 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.

2.3.3 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.

2.3.4 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).

2.4 COMP

2.4.1 The Comparator cannot wakeup the device from low power mode in some specific conditions

Description

The Comparator cannot wakeup the device from low power mode (sleep/stop) when the COMP_OUT state is changed from 1 to 0 .

Workaround

None.

2.4.2 COMP input PA0 interfering with PB0, PB2, or DAC

Description

PA0 selected as COMP input can interfere with PB0 or PB2 signal, or with the DAC output signal internally connected to an on-chip peripheral.

Note: *There is no such interference when PB0, PB2, or DAC output is selected as the COMP input.*

Workaround

When selecting PA0 as COMP input with the INPSEL[2:0] bitfield set to 001, configure the PB0 GPIO to a mode other than analog.

When selecting PA0 as COMP input with the INPSEL[2:0] bitfield set to 010, configure the PB2 GPIO to a mode other than analog.

When selecting PA0 as COMP input with the INPSEL[2:0] bitfield set to 101 or 111, either disable the DAC or do not connect its output to an on-chip peripheral.

Note: *The INPSEL[2:0] bitfield bit 0 is in the COMP_CFGR2 register and the bits 1 and 2 in the COMP_CFGR1 register.*

2.5 TIM

2.5.1 Bidirectional break mode not working with short pulses

Description

The TIM_BKIN and TIM_BKIN2 I/Os can be configured in bidirectional mode using the BKBID and BK2BID bits in the TIMx_BDTR register, to be forced to 0 when a break/break2 event occurs. The bidirectional break/break2 mode is not functional when the pulse width on break/break2 input is lower than two tim_ker_clk periods.

This limitation is also valid when software break events are generated (the break event is correctly generated internally but not reflected on break inputs).

Workaround

None.

For applications that can afford some latency in bidirectional break mode, the break interrupt can eventually be enabled, for the CPU to verify the break input state and force it to zero when a break/break2 event occurred.

2.6 LPTIM

2.6.1 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.

2.6.2 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.*

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

Description

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

Workaround

None.

2.6.4 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).

2.7 IWDG

2.7.1 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.

2.8 RTC and TAMP

2.8.1 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.

2.8.2 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.

2.8.3 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.

2.9 I2C

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

Description

The I^{2}C-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^{2}C-bus SDA line when $t_{SU;DAT}$ is smaller than one I2C kernel clock (I^{2}C-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^{2}C-bus standard, the minimum I2CCLK frequencies are as follows:

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

2.9.2 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.

2.9.3 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.

2.10 I3C

2.10.1 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.

2.10.2 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^{2}C/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.

2.10.3 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.

2.10.4 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.

2.11 USART

2.11.1 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.

2.11.2 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.

2.11.3 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.

2.11.4 Received data may be corrupted upon clearing the ABREN bit

Description

The USART receiver may miss data or receive corrupted data when the auto baud rate feature is disabled by software (ABREN bit cleared in the USART_CR2 register) after an auto baud rate detection, while a reception is ongoing.

Workaround

Do not clear the ABREN bit.

2.11.5 Noise error flag set while ONEBIT is set

Description

When the ONEBIT bit is set in the USART_CR3 register (one sample bit method is used), the noise error (NE) flag must remain cleared. Instead, this flag is set upon noise detection on the START bit.

Workaround

None.

Note: *Having noise on the START bit is contradictory with the fact that the one sample bit method is used in a noise free environment.*

2.12 LPUART

2.12.1 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.

2.12.2 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.

2.13 SPI

2.13.1 RDY output failure at high serial clock frequency

Description

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

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

Workaround

None.

2.13.2 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.

2.13.3 TIFRE flag wrongly set in slave PCM long frame mode if FIXCH = 1

Description

When FIXCH = 1, the flag TIFRE indicates an error when channel length indicated by WS does not last as expected. In slave PCM long frame mode, TIFRE is wrongly set, indicating a frame error even if it did not occur.

This issue occurs when all the following conditions are met:

- I2SMOD[1:0] = 1 (I2S/PCM mode) in the SPI_I2SCFGR register
- I2SCFG[2:0] = 000 or 001 or 100 (slave modes) in the SPI_I2SCFGR register
- I2SSTD[1:0] = 11 (PCM) and PCMSYNC=1 (PCM long) in the SPI_I2SCFGR register
- FIXCH[1:0] = 1 (channel length given by CHLEN) in the SPI_I2SCFGR register

Workaround

None. Ignore the TIFRE flag.

2.13.4 TIFRE flag never set in slave PCM/I2S mode if FIXCH = 0

Description

When FIXCH = 0, the TIFRE flag in the SPI_SR register is set to indicate a frame error if a new frame synchronization is received while the shift-in or shift-out of the previous data is not complete (early frame error). Instead, this flag is not set, and no frame error is detected.

This issue occurs when all the following conditions are met:

- I2SMOD[1:0] = 1 (I2S/PCM mode) in the SPI_I2SCFGR register
- I2SCFG[2:0] = 000 or 001 or 100 (slave modes) in the SPI_I2SCFGR register
- FIXCH[1:0] = 0 (CHLEN different from 16 or 32) in the SPI_I2SCFGR register

Workaround

None. Ignore the TIFRE flag.

2.14 FDCAN

2.14.1 Desynchronization under specific condition with edge filtering enabled

Description

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

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

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

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

Workaround

Disable edge filtering or wait for frame retransmission.

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

Description

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

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

Workaround

Apply one of the following measures:

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

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

2.15 USB

2.15.1 Buffer description table update completes after CTR interrupt triggers

Description

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

Workaround

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

Important security notice

Revision history

Table 5. **Document revision history**

Date	Version	Changes
02-Mar-2023	1	Initial release
03-Jul-2023	2	Added errata: • **System:** Corrupted content of the backup domain due to a missed power-on reset after this domain supply voltage drop • Read from flash memory may fail in VOS2 and VOS1 range • Debug not available when the PRODUCT_STATE is iROT-Provisioned • SRAM ECC error flags and addresses are updated only if interrupt is enabled (documentation erratum) • **Timer:** Bidirectional break mode not working with short pulses • **USART:** Received data may be corrupted upon clearing the ABREN bit • **LPUART:** Possible LPUART transmitter issue when using low BRR[15:0] value • **USB:** Buffer description table update complets after CTR interrupt triggers Modified erratum: LSE crystal oscillator may be disturbed by transitions on PC13. Removed errata (unduly included in the previous revision): • **Timer:** *Consecutive compare event missed in specific conditions* • *Output compare clear not working with external counter reset*
08-Dec-2023	3	Added: • Section 2.11.1 Data corruption due to noisy receive line • Section 2.13.2 Truncation of SPI output signals after EOT event Removed: • Possible corruption of last-received data depending on CRCSIZE setting • COUNTn_RX[9:0] bitfield reporting one byte less if APB frequency is below 12.6 MHz • False wakeup detection for last K-state not terminated by EOP or reset before suspend • ESOF interrupt timing desynchronized after resume signaling • Incorrect CRC16 in the memory buffer
01-Oct-2024	4	Modified erratum: • LSE crystal oscillator may be disturbed by transitions on PC13 Added errata: • Low-speed external clock in analog bypass mode might not work properly • Incorrect backup domain reset • Incorrect description of HDPx_STRT and HDPx_END default values • PLL1P output can unduly be disabled by software when used as SYSCLK clock • Incorrect description of EXTI line 53 • Invalid DAC output voltage for several DAC kernel clocks • LPTIM2_CH1 might prevent the system from entering low-power modes • Noise error flag set while ONEBIT is set • TIFRE flag wrongly set in slave PCM long frame mode if FIXCH = 1 • TIFRE flag never set in slave PCM/I2S mode if FIXCH = 0 Removed errata not applicable for this product: • Unexpected regular conversion when two consecutive injected conversions are performed in Dual interleaved mode

Contents