STM32H743xI Errata sheet

STM32H743xI rev Y device limitations

Applicability

This document applies to the part numbers of STM32H743xI devices listed in *Table 1* and their variants shown in *Table 2*.

Section 1 gives a summary and *Section 2* a description of workarounds for device limitations, with respect to the device datasheet and reference manual RM0433.

Table 1. Device summary

Reference	Part numbers
STM32H743xI	STM32H743VI, STM32H743ZI, STM32H743II, STM32H743BI, STM32H743XI, STM32H743AI

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32H743xI	Y	0x1003

1. Refer to the device data sheet for how to identify this code on different types of package.
2. REV_ID[15:0] bit field of DBGMCU_IDC register. Refer to the reference manual.

Contents

1 Summary of device limitations

The following table gives a quick references to all documented device limitations of STM32H743xI 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	Next Silicon revision
			Rev. Y	
System	2.2.1	Timer system breaks do not work	N	-
	2.2.2	Clock recovery system synchronization with USB SOF does not work	A	-
	2.2.3	SysTick external clock is not HCLK/8	A	-
	2.2.4	Option byte loading can be done with the user wait-state configuration	A	-
	2.2.5	Flash BusFault address register may not be valid when an ECC double error occurs	A	-
	2.2.6	Flash ECC address register may not be updated	N	-
	2.2.7	PCROP-protected areas in Flash memory may be unprotected	A	-
	2.2.8	Flash memory bank swapping might impact embedded Flash memory interface behavior	N	-
	2.2.9	Reading from AXI SRAM may lead to data read corruption	A	-
	2.2.10	Clock switching does not work when LSE failure is detected by CSS	A	-
	2.2.11	RTC stopped when a system reset occurs while the LSI is used as a clock source	A	-
	2.2.12	USB OTG_FS PHY drive limit on DP/DM pins	N	-
	2.2.13	LSE oscillator driving capability selection bits are swapped	A	-
	2.2.14	HRTIM internal synchronization does not work	N	-

Table 3. Summary of device limitations (continued)

Function	Section	Limitation	Status Rev. Y	Next Silicon revision
ADC	2.3.1	Conversion overlap may impact the ADC accuracy	A	-
	2.3.2	ADC resolution limited by LSE activity	A	-
	2.3.3	ADC maximum sampling rate when VDDA is lower than 2 V	A	-
	2.3.4	ADC maximum resolution when VDDA is higher than 3.3 V	A	-
	2.3.5	First ADC injected conversion in a sequence may be corrupted	A	-
	2.3.6	Writing the ADC_JSQR register when JADCSTART = 1 and JQDIS = 1 may lead to incorrect behavior	A	-
FMC	2.4.1	Dummy read cycles inserted when reading synchronous memories	N	N
	2.4.2	Wrong data read from a busy NAND Flash memory	A	A
	2.4.3	Missed clocks with continuous clock feature enabled	A	-
QUADSPI	2.5.1	First nibble of data is not written after a dummy phase	A	-
	2.5.2	QUADSPI_CCR hangs when QUADSPI_CR is configured to 0x0000 0000	A	A
	2.5.3	QUADSPI cannot be used in Indirect read mode when only data phase is activated	A	A
LPTIM	2.6.1	MCU may remain stuck in LPTIM interrupt when entering Stop mode	A	-
RTC	2.7.1	RTC calendar registers are not locked properly	A	-
I2C	2.8.1	10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave	A	A
	2.8.2	Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C	A	A
	2.8.3	Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period	P	-
	2.8.4	Spurious bus error detection in Master mode	A	A
	2.8.5	Last-received byte loss in Reload mode	P	-
	2.8.6	Spurious master transfer upon own slave address match	P	P
	2.8.7	START bit is cleared upon setting ADDRCF, not upon address match	P	P
USART	2.9.1	Underrun flag is set when the USART is used in SPI Slave receive mode	A	A
SPI	2.10.1	Spurious DMA Rx transaction after simplex Tx traffic	A	-
	2.10.2	Master data transfer stall at system clock much faster than SCK	A	A
	2.10.3	Corrupted CRC return at non-zero UDRDET setting	P	P
	2.10.4	TXP interrupt occurring while SPI/I2S disabled	A	A

Table 3. Summary of device limitations (continued)

Function	Section	Limitation	Status Rev. Y	Next Silicon revision
SDMMC	2.11.1	Busy not detected when a write operation suspended during busy phase resumes	A	-
	2.11.2	Wrong data line 2 generation between two blocks during DDR transfer with Read wait mode enabled	A	-
	2.11.3	Unwanted overrun detection when an AHB error is reported whereas all bytes have been received	A	-
	2.11.4	Consecutive multiple block transfers can induce incorrect data length	A	-
	2.11.5	Clock stop reported during Read wait mode sequence	A	-
FDCAN	2.12.1	Writing FDCAN_TTTS during initialization corrupts FDCAN_TTTMC	A	A
	2.12.2	Wrong data may be read from Message RAM by the CPU when using two FDCANs	A	-
HDMI-CEC	2.13.1	Unexpected switch to Receive mode without automatic transmission retry and notification	A	A
	2.13.2	CEC header not received due to unjustified Rx-Overrun detection	A	A

2 Description of device limitations

The following sections describe device limitations and provide workarounds if available. They are grouped by device functions.

2.1 Arm® 32-bit Cortex®-M7

There are not limitations related to the Arm®(a) Cortex®-M7 (r1p1) core.

Refer to the following Arm® documents:

- Arm® processor Cortex®-M7 (AT610) and Cortex®-M7 with FPU (AT611) software developer errata notice.
- Arm® embedded trace macrocell CoreSight ETM-M7 (TM975) software developer errata notice.

2.2 System

2.2.1 Timer system breaks do not work

Description

System break sources (processor LOCKUP output, PVD detection, RAM ECC error, Flash ECC error or clock security system detection) do not generate a break event on TIM1, TIM8 and HRTIM.

Workaround

None

2.2.2 Clock recovery system synchronization with USB SOF does not work

Description

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

Workaround

When available, use the LSE oscillator as synchronization source.

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

2.2.3 SysTick external clock is not HCLK/8

Description

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

Workaround

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

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

Description

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

Workaround

When performing option byte loading (modification), configure the correct number of wait-states or use the default value (7 wait states).

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

Description

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

Workaround

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

2.2.6 Flash ECC address register may not be updated

Description

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

Workaround

None.

2.2.7 PCROP-protected areas in Flash memory may be unprotected

Description

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

Workaround

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

2.2.8 Flash memory bank swapping might impact embedded Flash memory interface behavior

Description

When Flash memory bank swapping feature is enabled, the embedded Flash memory interface behavior might become unpredictable.

Workaround

Do not enable the Flash memory bank swapping feature on devices revision Y.

2.2.9 Reading from AXI SRAM may lead to data read corruption

Description

Read data may be corrupted when the following conditions are met:

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

Workaround

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

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

Description

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

Workaround

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

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

Description

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

Workaround

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

2.2.12 USB OTG_FS PHY drive limit on DP/DM pins

Description

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

Workaround

None

2.2.13 LSE oscillator driving capability selection bits are swapped

Description

The LSEDRV[1:0] bits in the RCC_BDCR register, which are used to select LSE oscillator driving capability, are swapped (see *Table 4*).

Table 4. Expected vs effective LSE driving mode

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

Workaround

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

2.2.14 HRTIM internal synchronization does not work

Description

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

Workaround

None.

2.3 ADC

2.3.1 Conversion overlap may impact the ADC accuracy

Description

The following conditions may impact the ADC accuracy

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

Workaround

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

2.3.2 ADC resolution limited by LSE activity

Description

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

- ADC3 channels on pins PF3 to PF10

Workaround

16-bit and 14-bit data resolutions are not recommended on these pins. This limits data resolution configuration to 8 bits, 10 bits or 12 bits.

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

Description

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

Workaround

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

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

Description

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

Workaround

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

2.3.5 First ADC injected conversion in a sequence may be corrupted

Description

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

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

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

Workaround

Apply one of the following measures:

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

2.3.6 Writing the ADC_JSQR register when JADCSTART = 1 and JQDIS = 1 may lead to incorrect behavior

Description

Writing the ADCx_JSQR register when an injected conversion is ongoing (JADCSTART=1) may lead to unpredictable ADC behavior if the queue of context is not enabled (JQDIS=1).

Workaround

Apply one of the following measures:

- Use the context queue (JQDIS=0) to allow on-the-fly ADCx_JSQR modification
- Ensure that no injected conversion is ongoing (JADSTART=0) before modifying ADC_JSQR register.

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 AXI burst access. However, the extra data values that are read are not used by the FSMC and there is no functional failure.

Workaround

None.

2.4.2 Wrong data read from a busy NAND Flash 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 may 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.

2.4.3 Missed clocks with continuous clock feature enabled

Description

When the continuous clock feature is enabled, the FMC_CLK clock can be switched OFF in the following conditions:

- The FMC_CLK clock is divided by 2.
- An asynchronous byte transaction is performed on an FMC bank configured in 32-bit memory data width.

Note: When the FMC_CLK clock is switched OFF on static memories, it can be switched ON by issuing a synchronous transaction or any asynchronous transaction different from a byte access on 32-bit data bus width.

Workaround

When the continuous clock feature is enabled, do not use the FMC_CLK clock divider ratio of 2 when issuing a byte transaction to 32-bit asynchronous memories.

2.5 QUADSPI

2.5.1 First nibble of data is not written after a dummy phase

Description

The first nibble of data to be written to the external Flash memory is lost in the following conditions:

- The QUADSPI is used in the Indirect write mode,
- and at least one dummy cycle is used.

Workaround

Use an alternate-bytes phase instead of a dummy phase in order to add a latency period between the address phase and the data phase. This workaround works only if the number of dummy cycles corresponds to a multiple of 8 bits of data.

As an example:

- To generate 1 dummy cycle, send 1 alternate-byte in 4 data line DDR mode or Dual-Flash SDR mode.
- To generate 2 dummy cycles, send 1 alternate-byte in 4 data line SDR mode
- To generate 4 dummy cycles, send 2 alternate-bytes in 4 data line SDR mode or send 1 alternate-byte in 2 data line SDR mode
- To generate 8 dummy cycles, send 1 alternate-byte in 1 data line SDR mode.

2.5.2 QUADSPI_CCR hangs when QUADSPI_CR is configured to 0x0000 0000

Description

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

Workaround

Reset then set again the EN bit in the QUADSPI_CR register.

2.5.3 QUADSPI cannot be used in Indirect read mode when only data phase is activated

Description

When the QUADSPI is configured in Indirect read with only the data phase activated (in Single, Dual, Quad or Dual-quad I/O mode), the QUADSPI peripheral hangs and the BUSY flag remains high in the QUADSPI_SR register. An abort must be performed to reset the BUSY flag and exit from the hanging status.

Workaround

Use the dummy phase with at least two dummy cycles.

2.6 LPTIM

2.6.1 MCU 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 MCU from Stop mode may be frozen in active state. Consequently, when trying to enter Stop mode, this limitation prevents the MCU 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 LPTIMx_CR register. Instead, reset the whole LPTIMx peripheral via the RCC controller by setting and resetting its respective LPTIMxRST bit in RCC_APByRSTRz register.

2.7 RTC

2.7.1 RTC calendar registers are not locked properly

Description

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

Workaround

Apply one of the following measures:

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

2.8 I2C

2.8.1 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave

Description

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

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

Workaround

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

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

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

Description

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

Some reference manual revisions may omit this information.

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

1. BUSY flag is wrongly set when the MCU exits Stop mode. This prevents from initiating a transfer in Master mode, as the START condition cannot be sent when BUSY is set.
2. If clock stretching is enabled (NOSTRETCH = 0), the SCL line is pulled low by I2C and the transfer stalled as long as the MCU remains in Stop mode.
   The occurrence of such condition depends on the timing configuration, peripheral clock frequency, and I^2C-bus frequency.

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

Workaround

No application workaround is required.

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

Description

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

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

The MCU 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.

2.8.4 Spurious bus error detection in Master mode

Description

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

Workaround

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

2.8.5 Last-received byte loss in Reload mode

Description

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

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

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

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

Workaround

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

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

2.8.6 Spurious master transfer upon own slave address match

Description

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

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

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

Workaround

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

Normal mode (SBC = 0):

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

Slave byte control mode (SBC = 1):

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

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

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

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

Description

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

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

Workaround

No application workaround is required for this description inaccuracy issue.

2.9 USART

2.9.1 Underrun flag is set when the USART is used in SPI Slave receive mode

Description

When the USART is used in SPI Slave receive mode, the underrun flag (UDR bit in USART_ISR register) may be set even if the transmitter is disabled (TE bit set to 0 in USAR_CR1 register).

Workaround

Three workarounds are possible

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

2.10 SPI

2.10.1 Spurious DMA Rx transaction after simplex Tx traffic

Description

With empty RXFIFO, SPI/I2S can spuriously generate a DMA read request upon enabling DMA receive traffic (by setting RXDMAEN bit), provided that the preceding completed transaction is a simplex transmission.

Workaround

Before enabling DMA Rx transfer following a completed Tx simplex transfer, perform hardware reset of the SPI/I2S peripheral.

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

Description

With the system clock (spi_pclk) substantially faster than SCK (spi_ker_ck divided by a prescaler), SPI/I2S master data transfer can stall upon setting the CSTART bit within one SCK cycle after the EOT event (EOT flag raise) signaling the end of the previous transfer.

Workaround

Apply one of the following measures:

- Disable then enable SPI/I2S after each EOT event.
- Upon EOT event, wait for at least one SCK cycle before setting CSTART.
- Prevent EOT events from occurring, by setting transfer size to undefined (TSIZE = 0) and by triggering transmission exclusively by TXFIFO writes.

2.10.3 Corrupted CRC return at non-zero UDRDET setting

Description

With non-zero setting of UDRDET[1:0] bitfield, the SPI/I2S slave can transmit the first bit of CRC pattern corrupted, coming wrongly from the UDRCFG register instead of SPI_TXCRC. All other CRC bits come from the SPI_TXCRC register, as expected.

Workaround

Keep TXFIFO non-empty at the end of transfer.

2.10.4 TXP interrupt occurring while SPI/I2S disabled

Description

SPI/I2S peripheral is set to its default state when disabled (SPE = 0). This flushes the FIFO buffers and resets their occupancy flags. TXP and TXC flags become set (the latter if the TSIZE field contains zero value), triggering interrupt if enabled with TXPIE or EOTIE bit, respectively. The resulting interrupt service can be spurious if it tries to write data into TXFIFO to clear the TXP and TXC flags, while both FIFO buffers are inaccessible (as the peripheral is disabled).

Workaround

Keep TXP and TXC (the latter if the TSIZE field contains zero value) interrupt disabled whenever the SPI/I2S peripheral is disabled.

2.11 SDMMC

2.11.1 Busy not detected when a write operation suspended during busy phase resumes

Description

When a card accepts a suspend command during a block write operation busy phase, the card may drive the data line 0 (SDMMC_D0) when the write transfer is resumed. The SDMMC does not detect that the data line 0 is Low when the write transfer resumes.

Workaround

To suspend a write transfer:

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

2.11.2 Wrong data line 2 generation between two blocks during DDR transfer with Read wait mode enabled

Description

The Read wait mode allows suspending an SDIO multiple block read operation when the host is not ready to receive the next bytes. The host can request the card to suspend temporarily the transfer by driving data line 2 (SDMMC_D2) low between two blocks.

When a double data rate (DDR) read operation is ongoing, data line 2 is not driven low but toggles constantly. Consequently, some bytes are not received and a CRC error failure is reported.

Workaround

Use the clock stretching method (RWMOD = 1) instead of data line 2 to suspend temporarily the transfer between two blocks.

2.11.3 Unwanted overrun detection when an AHB error is reported whereas all bytes have been received

Description

When the internal DMA is used and a write transfer initiated by the SDMMC on the AHB fails, the IDMATE flag is set in SDMMC_STAR and the transfer is aborted by flushing the FIFO.

When an AHB error occurs on the three last bursts of a successful read transfer, the FIFO is considered as empty (DATAEND flag set in SDMMC_STAR) but some bytes, not yet transferred to the FIFO, may still be present in the internal receive buffer. As a result, the following read operation will fail and report an overrun error.

Workaround

1. When DATAEND = 1, check IDMATE flag.
2. If IDMATE = 1 and DTDIR = 1 in SDMMC_DCTRL, reset SDMMC.

2.11.4 Consecutive multiple block transfers can induce incorrect data length

Description

When a new transfer is started by setting the DTEN bit in SDMMC_DCTRL control register while less than eight SDMMC clock cycles elapsed since the end of the previous transfer, the second transfer is performed with the number of blocks configured for the previous transfer. This is due to the fact that the new number of data to be transferred has not been reloaded in the internal data block counter.

Workaround

The user application must ensure that at least 8 SDMMC clock cycles elapsed between the successful completion of a transfer and the moment DTEN bit is set.

2.11.5 Clock stop reported during Read wait mode sequence

Description

When the SDMMC clock is stopped at low level, CKSTOP flag may be wrongly set in the SDMMC_STAR register.

Workaround

When the multiple block transfer completes (DATAEND = 1 in SDMMC_STAR), simultaneously set CKSTOPC and DATAENDC to 1 in SDMMC_ICR register.

2.12 FDCAN

2.12.1 Writing FDCAN_TTTS during initialization corrupts FDCAN_TTTMC

Description

During TTCAN initialization, writing to FDCAN TT Trigger Select Register (FDCAN_TTTS) also affects FDCAN TT Trigger Memory Configuration Register (FDCAN_TTTMC).

Workaround

The user application must avoid writing to FDCAN_TTTS register during TTCAN initialization phase.

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

2.12.2 Wrong data may be read from Message RAM by the CPU when using two FDCANs

Description

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

The issue is not present if the CPU requests write access to Message RAM.

Workaround

To avoid concurrent read accesses between the CPU and FDCANs, use only one FDCAN at a time.

2.13 HDMI-CEC

2.13.1 Unexpected switch to Receive mode without automatic transmission retry and notification

Description

If the HDMI-CEC peripheral starts a transmit operation, and at the same time another CEC initiator attempts to perform the same operation but with a wrong start-bit timing and/or without following the signal free timing rules, the transmission is aborted and never restarts, and the HDMI switch to Receive mode. The user application is not informed of the bus error.

Workaround

Use transmission timeout: when a timeout occurs, to enable again the transmission of the pending message, disable and enable again the HDMI-CEC peripheral to clear TXSOM bit in CEC_CR register.

2.13.2 CEC header not received due to unjustified Rx-Overrun detection

Description

In a multinode CEC network, two messages are sent to two different followers. The CEC device which is the destination of the second message should ignore the first message and receive the second one. However, the header of the second message is not received and an Rx-Overrun is wrongly detected and signaled by setting the RXOVR flag in the CEC_ISR register and the RXOVRIE interrupt in the CEC_IER register.

The issue is not present:

- in Listen mode,
- when the first message is a broadcast message
- when the first message is a ping message (header only),
- or when the first message header is not acknowledged.

Workaround

Configure the HDMI-CEC to operate in Listen mode and apply message filtering based on destination address. In this case all the messages sent over the CEC bus will be received and it is up to the user application to discard the messages that are not sent to its address or that are broadcast.

3 Revision history

Table 5. Document revision history

Date	Revision	Changes
19-Jun-2017	1	Initial release.

Table 5. Document revision history (continued)

Date	Revision	Changes
02-Nov-2017	2	Added STM32H743AI part number. Removed JPEG limitation. Updated *Section 2.4.1: Dummy read cycles inserted when reading synchronous memories* and *Section 2.4.2: Wrong data read from a busy NAND Flash memory*. Added *Section 2.12.2: Wrong data may be read from Message RAM by the CPU when using two FDCANs*.

Table 5. Document revision history (continued)

Date	Revision	Changes
31-May-2018	3	**System limitations**: – Reorganized to group together Flash memory and RAM limitations – Removed limitations “48 MHz RC oscillator user calibration values lost after system reset” and “Flash memory wirte sequence error flag not set” – Replaced *“Accessing the system memory may stall the system when Flash memory banks are swapped”* and “*Write flags are not swapped when Flash memory banks are swapped”* limitations by *Section 2.2.8: Flash memory bank swapping might impact embedded Flash memory interface behavior*. **ADC limitations:** – Moved all ADC limitations under *Section 2.3: ADC*. – Added *Section 2.3.5: First ADC injected conversion in a sequence may be corrupted* and *Section 2.3.6: Writing the ADC_JSQR register when JADCSTART = 1 and JQDIS = 1 may lead to incorrect behavior*. Added *Section 2.7.1: RTC calendar registers are not locked properly*. Added *Section 2.5.2: QUADSPI_CCR hangs when QUADSPI_CR is configured to 0x0000 0000* and *Section 2.5.3: QUADSPI cannot be used in Indirect read mode when only data phase is activated*. **I2C limitations**: – Updated *Section 2.8.1: 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave* and *Section 2.8.3: Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period*. – Added *Section 2.8.2: Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C*, *Section 2.8.4: Spurious bus error detection in Master mode*, *Section 2.8.5: Last-received byte loss in Reload mode*, *Section 2.8.6: Spurious master transfer upon own slave address match* and *Section 2.8.7: START bit is cleared upon setting ADDRCF, not upon address match*. Removed limitation “NOSTRETCH setting may impact Master mode”. **SPI/I2S limitations**: – Removed limitation “Wrong DMA receive requests may be generated in Half-duplex mode” – Added *Section 2.10.1: Spurious DMA Rx transaction after simplex Tx traffic*, *Section 2.10.2: Master data transfer stall at system clock much faster than SCK*, *Section 2.10.3: Corrupted CRC return at non-zero UDRDET setting* and *Section 2.10.4: TXP interrupt occurring while SPI/I2S disabled*.