STM32F72xxx STM32F73xxx
Errata sheet

STM32F72xxx and STM32F73xxx device errata

Applicability

This document applies to the part numbers of STM32F72xxx and STM32F73xxx devices listed in *Table 1* and their variants shown in *Table 2*.

Section 1 gives a summary and *Section 2* a description of / workaround for device limitations, with respect to the device datasheet and reference manual RM0431.

Deviation of the device behavior from the description in its corresponding data sheet and/or reference manual can be considered as a device limitation or as a documentation error. The term "errata" applies both to limitations and documentation errata.

Table 1. Device summary

Reference	Part numbers
STM32F72xxx	STM32F722IE, STM32F722ZE, STM32F722VE, STM32F722RE, STM32F722IC, STM32F722ZC, STM32F722VC, STM32F722RC, STM32F723IE, STM32F723ZE, STM32F723VE, STM32F723IC, STM32F723ZC, STM32F723VC
STM32F73xxx	STM32F732IE, STM32F732ZE, STM32F732VE, STM32F732RE, STM32F733IE, STM32F733ZE, STM32F733VE STM32F730R8, STM32F730V8, STM32F730Z8, STM32F730I8

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32F72xxx	A and 1	0x1000
STM32F73xxx		

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_IDCODE register. Refer to the reference manual.

Contents

1 Summary of device errata

The following table gives a quick references to all documented device limitations of STM32F72xxx and STM32F73xxx 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 and 1
System	2.2.1	Internal noise impacting the ADC accuracy	A
	2.2.2	Wakeup from Standby mode when the back-up SRAM regulator is enabled	A
FMC	2.3.1	Dummy read cycles inserted when reading synchronous memories	N
	2.3.2	Wrong data read from a busy NAND memory	A
	2.3.3	Spurious clock stoppage with continuous clock feature enabled	A
	2.3.4	Data read might be corrupted when the write FIFO is disabled	A
QUADSPI	2.4.1	First nibble of data is not written after a dummy phase	A
	2.4.2	Wrong data can be read in memory-mapped after an indirect mode operation	A
	2.4.3	Memory-mapped read operations may fail when timeout counter is enabled.	A
ADC	2.5.1	ADC sequencer modification during conversion	A
DAC	2.6.1	DMA underrun flag management	A
	2.6.2	DMA request not automatically cleared by DMAEN=0	A
LPTIM	2.7.1	MCU may remain stuck in LPTIM interrupt when entering Stop mode	A
RTC	2.8.1	RTC calendar registers are not locked properly	P
I2C	2.9.1	Wrong data sampling when data setup time (tSU;DAT) is shorter than one I2C kernel clock period	A
	2.9.2	Spurious bus error detection in master mode	A
	2.9.3	10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave	A
	2.9.4	Last-received byte loss in reload mode	A

Table 3. Summary of device limitations (continued)

Function	Section	Limitation	Status
			Rev. A and 1
USART	*2.10.1*	*nRTS is active while RE or UE = 0*	A
SPI/I2S	*2.11.1*	*BSY bit may stay high at the end of a data transfer in Slave mode*	A
SDMMC	*2.12.1*	*Wrong CCRCFAIL status after a response without CRC is received*	A
	2.12.2	*MMC stream write of less than 8 bytes does not work correctly*	A
BxCAN	*2.13.1*	*BxCAN time triggered mode not supported*	N

2 Description of device errata

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

2.1 Core

An errata notice of the STM32F72xxx and STM32F73xxx core is available from http://infocenter.arm.com.

All the described limitations are minor and related to the revision r1p0 of the Cortex®-M7 core. Refer to:

- 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

Table 4 summarizes these limitations and their implications on the behavior of STM32F72xxx and STM32F73xxx devices.

Table 4. Cortex®-M7 core limitations and impact on microcontroller behavior

Arm ID	Arm category	Impact on STM32F72xxx and STM32F73xxx devices
851031	Cat C	Minor
850725	Cat C	Minor
850724	Cat C	Minor

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

2.2 System

2.2.1 Internal noise impacting the ADC accuracy

Description

An internal noise generated on V_{DD} supplies and propagated internally may impact the ADC accuracy.

This noise is always active whatever the power mode of the MCU (Run or Sleep).

Workaround

To adapt the accuracy level to the application requirements, set one of the following options:

- Option1
  Set the ADCDC1 bit in the PWR_CR register.
- Option2
  Set the corresponding ADCxDC2 bit in the SYSCFG_PMC register.

Only one option can be set at a time.

For more details on option1 and option2 mechanisms, refer to AN4073

2.2.2 Wakeup from Standby mode when the back-up SRAM regulator is enabled

Description

When writing to the PWR_CSR1 register to enable or disable the back-up SRAM regulator, if the EIWUP bit is overwritten 0, the RTC wakeup event (alarm, RTC Tamper, RTC TimeStamp or RTC wakeup time) does not wake up the system from Standby mode.

Workaround

For each write access on the PWR_CSR1 register to enable or disable the back-up SRAM regulator, the EIWKUP bit must be set to 1 in order to enable a wakeup from Standby mode using RTC events.

2.3 FMC

2.3.1 Dummy read cycles inserted when reading synchronous memories

Description

When performing a burst read access from a synchronous memory, two dummy read accesses are performed at the end of the burst cycle whatever the type of burst access.

The extra data values read are not used by the FMC and there is no functional failure.

Workaround

None.

2.3.2 Wrong data read from a busy NAND memory

Description

When a read command is issued to the NAND memory, the R/B signal gets activated upon the de-assertion of the chip select. If a read transaction is pending, the NAND controller might not detect the R/B signal (connected to NWAIT) previously asserted and sample a wrong data. This problem occurs only when the MEMSET timing is configured to 0x00 or when ATTHOLD timing is configured to 0x00 or 0x01.

Workaround

Either configure MEMSET timing to a value greater than 0x00 or ATTHOLD timing to a value greater than 0x01.

2.3.3 Spurious clock stoppage with continuous clock feature enabled

Description

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

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

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

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

Workaround

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

2.3.4 Data read might be corrupted when the write FIFO is disabled

Description

When the write FIFO is disabled, the FIFO empty event is generated for every write access. During a write access, if a new read access occurs, the FMC grants the read access and waits till the FIFO gets empty. If another read access occurs in a very short window (one cycle the FIFO empty event), the returned data are corrupted. This issue occurs only when the write FIFO is disabled (the WFDIS bit in the FMC_BCR1 register is set).

Workaround

Enable the write FIFO.

2.4 QUADSPI

2.4.1 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

Do not use dummy cycles for creating latency between the address phase and data phase, in indirect write mode. Instead, use alternate bytes to substitute the dummy cycles. The same latency can be achieved if the number of dummy cycles to substitute with alternate-byte cycles is an integer multiple of the number of cycles required for transferring one alternate byte, as shown in *Table 5*:

Table 5. Cycle number versus QUADSPI modes

QUADSPI mode	Number of cycles per alternate byte
4-data-line DDR	1
4-data-line SDR	2
2-data-line SDR	4
1-data-line SDR	8

For example, the latency corresponding to eight dummy cycles can be exactly substituted with one single alternate byte in 1-data-line SDR mode, but two alternate bytes are required in 2-data-line SDR mode. One single dummy cycle can only exactly be substituted in 4-data-line DDR mode, using one alternate byte.

Note: This is also applicable to dual-flash memory mode.

2.4.2 Wrong data can be read in memory-mapped after an indirect mode operation

Description

Wrong data can be read with the first memory-mapped read request when the Quad-SPI peripheral enters in memory-mapped mode without reset of both LSB bits in the QUADSPI_AR[1:0] address register.

Workaround

The QUADSPI_AR register must be reset just before entering in memory-mapped mode. This can be done in two different ways, depending on the current Quad-SPI operating mode:

1. Indirect read mode:
   a) Reset the address register
   b) Make an abort request to stop the reading and clear the busy bit
   c) Enter in memory-mapped mode.
2. Indirect write mode:
   a) Reset the address register
   b) Enter in memory-mapped mode

Note: The user must take care to not write to the QUADSPI_DR register after resetting the address register.

2.4.3 Memory-mapped read operations may fail when timeout counter is enabled.

Description

In the Memory-mapped mode when the TC is enabled, the Quad-SPI peripheral can hang and the memory-mapped read operations fail.

The Quad-SPI hang occurs if the timeout flag TOF is set at the same clock edge of a new memory mapped read request.

Workaround

The timeout counter must be disabled.

In order to rise the chip select high, the application can make an abort at the end of each memory-mapped read operation.

2.5 ADC

2.5.1 ADC sequencer modification during conversion

Description

If an ADC conversion is started by software (writing the SWSTART bit), and if the ADC_SQRx or ADC_JSQRx registers are modified during the conversion, the current conversion is reset and the ADC does not restart a new conversion sequence automatically. If an ADC conversion is started by hardware trigger, this limitation does not apply. The ADC restarts a new conversion sequence automatically.

Workaround

When an ADC conversion sequence is started by software, a new conversion sequence can be restarted only by setting the SWSTART bit in the ADC_CR2 register.

2.6 DAC

2.6.1 DMA underrun flag management

Description

If the DMA is not fast enough to input the next digital data to the DAC, as a consequence, the same digital data is converted twice. In these conditions, the DMAUDR flag is set, which usually leads to disable the DMA data transfers. This is not the case: the DMA is not disabled by DMAUDR=1, and it keeps serving the DAC.

Workaround

To disable the DAC DMA stream, reset the EN bit (corresponding to the DAC DMA stream) in the DMA_SxCR register.

2.6.2 DMA request not automatically cleared by DMAEN=0

Description

If the application wants to stop the current DMA-to-DAC transfer, the DMA request is not automatically cleared by DMAEN=0, or by DACEN=0.

If the application stops the DAC operation while the DMA request is high, the DMA request is pending while the DAC is reinitialized and restarted; with the risk that a spurious unwanted DMA request is served as soon as the DAC is re-enabled.

Workaround

To stop the current DMA-to-DAC transfer and restart, the following sequence should be applied:

1. Check if DMAUDR is set.
2. Clear the DAC/DMAEN bit.
3. Clear the EN bit of the DAC DMA/Stream.
4. Reconfigure by software the DAC, DMA, triggers.
5. Restart the application.

2.7 LPTIM

2.7.1 MCU may remain stuck in LPTIM interrupt when entering Stop mode

Description

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

When the firmware clears the LPTIM_CR.ENABLE bit 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.

2.8 RTC

2.8.1 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

1. Use BYPSHAD = 1 mode (bypass shadow registers), or
2. If BYPSHAD = 0, read the RTC_SSR register again after reading the RTC_SSR, RTC_TR, RTC_DR registers to confirm that RTC_SSR is still the same, otherwise read the values again.

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 I²C-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 I²C-bus SDA line when $t_{SU;DAT}$ is smaller than one I2C kernel clock (I²C-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 I2C-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 I²C-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 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave

Description

An I²C-bus master generates STOP condition upon non-acknowledge of I²C 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.

2.9.4 Last-received byte loss in reload mode

Description

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

- I2C-bus stretching is enabled (NOSTRETCH = 0)
- RELOAD bit of the I2C_CR2 register is set
- NBYTES bitfield of the I2C_CR2 register is set to N greater than 1 • byte N is received on the I2C-bus, raising the TCR flag
- N - 1 byte is not yet read out from the data register at the instant TCR is raised,

then the SCL line is pulled low (I2C-bus clock stretching) and the transfer of the byte N from the shift register to the data register inhibited until the byte N-1 is read and NBYTES bitfield reloaded with a new value, the latter of which also clears the TCR flag. As a consequence, the software cannot get the byte N and use its content before setting the new value into the NBYTES field.

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

Workaround

- In 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 bytes, 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.

2.10 USART

2.10.1 nRTS is active while RE or UE = 0

Description

The nRTS line is driven low as soon as the RTSE bit is set, even if the USART is disabled (UE = 0) or the receiver is disabled (RE = 0), that is, not ready to receive data.

Workaround

Upon setting the UE and RE bits, configure the I/O used for nRTS into alternate function.

2.11 SPI/I2S

2.11.1 BSY bit may stay high at the end of a data transfer in Slave mode

Description

The BSY flag may sporadically remain high at the end of a data transfer in slave mode. This occurs upon coincidence of internal CPU clock and external SCK clock provided by master.

In such an event, if the software only relies on BSY flag to detect the end of SPI slave data transaction (for example to enter low-power mode or to change data line direction in half-duplex bidirectional mode), the detection fails.

As a conclusion, the BSY flag is unreliable for detecting the end of data transactions.

Workaround

Depending on SPI operating mode, use the following means for detecting the end of transaction:

- When NSS hardware management is applied and NSS signal is provided by master, use NSS flag.
- In SPI receiving mode, use the corresponding RXNE event flag.
- In SPI transmit-only mode, use the BSY flag in conjunction with a timeout expiry event. Set the timeout such as to exceed the expected duration of the last data frame and start it upon TXE event that occurs with the second bit of the last data frame. The end of the transaction corresponds to either the BSY flag becoming low or the timeout expiry, whichever happens first.

Prefer one of the first two measures to the third as they are simpler and less constraining. Alternatively, apply the following sequence to ensure reliable operation of the BSY flag in SPI transmit mode:

1.Write last data to data register

2. Poll the TXE flag until it becomes high, which occurs with the second bit of the data frame transfer

3. Disable SPI by clearing the SPE bit mandatorily before the end of the frame transfer
4. Poll the BSY bit until it becomes low, which signals the end of transfer

Note: The alternative method can only be used with relatively fast CPU speeds versus relatively slow SPI clocks or/and long last data frames. The faster is the software execution, the shorter can be the duration of the last data frame.

2.12 SDMMC

2.12.1 Wrong CCRCFAIL status after a response without CRC is received

Description

The CRC is calculated even if the response to a command does not contain any CRC field. As a consequence, after the SDIO command IO_SEND_OP_COND (CDM5) is sent, the CCRCFAIL bit of the SDIO_STA register is set.

Workaround

The CCRCFAIL bit in the SDIO_STA register shall be ignored by the software CCRCFAIL must be cleared by setting the CCRCFAILC bit in the SDIO_ICR register after reception of the response to the CMD5 command.

2.12.2 MMC stream write of less than 8 bytes does not work correctly

Description

When the SDMMC host starts a stream write (WRITE_DAT_UNTIL_STOP CMD20), the number of bytes to transfer is not known by the card.

The card writes data from the host until a STOP_TRANSMISSION (CMD12) command is received.

Use the WAITRESP value equal to “00” to indicate to SDMMC CPSM that no response is expected.

The WAITPEND bit 9 of the SDMMC_CMD register is set to synchronize the sending of the STOP_TRANSMISSION (CMD12) command with the data flow.

When WAITPEND is set, the transmission of this command stays pending until 50 data bits (including the Stop bit) remain to transmit.

For a stream write of less than 8 bytes, the STOP_TRANSMISSION (CMD12) command should be started before the data transfer starts. Instead of this, the data write and the command sending are started simultaneously.

It implies that when less than 8 bytes have to be transmitted, (8 - DATALENGTH) bytes are programmed to 0xFF in the card after the last byte programmed (where DATALENGTH is the number of data bytes to be transferred).

Workaround

Do not use stream write WRITE_DAT_UNTIL_STOP (CMD20) with a DATALENGTH less then 8 bytes. Use set block length (SET_BLOCKLEN: CMD16) followed by the single block write command (WRITE_BLOCK_CMD24) instead of the stream write (CMD20) with the desired block length.

2.13 BxCAN

2.13.1 BxCAN time triggered mode not supported

Description

The time triggered communication mode described in the reference manual is not supported. As a result the time stamp values are not available. The TTCM bit must be kept cleared in the CAN_MCR register (time triggered communication mode disabled).

Workaround

None.

3 Revision history

Table 6. Document revision history

Date	Revision	Changes
02-Feb-2017	1	Initial release.
22-Aug-2017	2	Updated whole document in line with the new errata sheet structure. FMC limitation: – Added *Section 2.3.4: Data read might be corrupted when the write FIFO is disabled*. QUADSPI limitation: – Updated *Section 2.4.1: First nibble of data is not written after a dummy phase*. SPI/I2S limitation: – Moved *Section 2.11.1: BSY bit may stay high at the end of a data transfer in Slave mode* from I2C to SPI/I2S limitation.
28-Jun-2018	3	Updated *Table 1: Device summary* adding STM32F730x8 part numbers. Added Revision 1: – Updated *Table 2: Device variants*. – Updated *Table 3: Summary of device limitations* QUADSPI limitation: – Added *Section 2.4.3: Memory-mapped read operations may fail when timeout counter is enabled.*. I2C limitations: – Added *Section 2.9.4: Last-received byte loss in reload mode*. – Updated limitation in line with I2C2 errata sheet. FMC limitations: – Updated *Section 2.3.1: Dummy read cycles inserted when reading synchronous memories*. – Updated *Section 2.3.2: Wrong data read from a busy NAND memory*. – Updated *Section 2.3.3: Spurious clock stoppage with continuous clock feature enabled*. – Updated *Section 2.3.4: Data read might be corrupted when the write FIFO is disabled*. RTC limitation: – Added *Section 2.8.1: RTC calendar registers are not locked properly*. LPTIM limitation: – Added *Section 2.7.1: MCU may remain stuck in LPTIM interrupt when entering Stop mode*. SPI/I2S limitation: – Updated *Section 2.11.1: BSY bit may stay high at the end of a data transfer in Slave mode*.