STM32F74xxx STM32F75xxx Errata sheet

STM32F74xxx and STM32F75xxx device limitations

Silicon identification

This errata sheet applies to the revision Z of STMicroelectronics STM32F74xxx and STM32F75xxx products. This family features an ARM® 32-bit Cortex®-M7 with FPU core, for which an errata notice is also available (see *Section 1* for details).

The products are identifiable as shown in *Table 1*:

- By the revision code marked below the order code on the device package
- By the last three digits of the internal order code printed on the box label

Table 1. Device identification(1)

Order code	Revision code(2) marked on the device
STM32F74xxx	"Z"
STM32F75xxx	

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see the STM32F74xxx and STM32F75xxx reference manual (RM0385) for details on how to find the revision code).
2. Refer to the device datasheets for details on how identify the revision code on the different packages.

The full list of part numbers is shown in *Table 2*.

Table 2. Device summary

Reference	Part number
STM32F74xxx	STM32F745ZG, STM32F745IG, STM32F745ZE, STM32F745IE, STM32F745VG, STM32F745VE, STM32F746VG, STM32F746ZG, STM32F746IG, STM32F746BG, STM32F746NG, STM32F746IE, STM32F746VE, STM32F746ZE, STM32F746BE, STM32F746NE
STM32F75xxx	STM32F756VG, STM32F756ZG, STM32F756IG, STM32F756BG, STM32F756NG

Contents

List of tables

1 ARM® 32-bit Cortex®-M7 with FPU limitations

An errata notice of the STM32F74xx and STM32F75xx core is available from http://infocenter.arm.com.

All the described limitations are minor and related to the revision r0p1 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 3 summarizes these limitations and their implications on the behavior of STM32F74xx and STM32F75xx devices.

Table 3. Cortex®-M7 core limitations and impact on microcontroller behavior

ARM ID	ARM category	Impact on STM32F74xxx and STM32F75xxx devices
837070	Cat B	Minor
834922	Cat B	Minor
838169	Cat B (rare)	Minor
839269	Cat C	Minor
839169	Cat C	Minor
837069	Cat C	Minor
834971	Cat C	Minor
834924	Cat C	Minor
834923	Cat C	Minor
833872	Cat C	Minor
830969	Cat C	Minor

2 STM32F74xxx and STM32F75xxx silicon limitations

Table 4 gives quick references to all documented limitations.

The legend for *Table 4* is as follows:

A = workaround available,

N = no workaround available,

P = partial workaround available,

'-' and grayed = fixed.

Table 4. Summary of silicon limitations

Links to silicon limitations		Revision Z
Section 2.1: System limitations	*Section 2.1.1: Missed ADC triggers from TIM1/TIM8, TIM2/TIM5/TIM4/TIM6/TRGO or TGRO2 event*	A
	Section 2.1.2: Internal noise impacting the ADC accuracy	A
	Section 2.1.3: Wakeup from Standby mode when the Back-up SRAM regulator is enabled	A
Section 2.2: RTC limitation	*Section 2.2.1: Spurious tamper detection when disabling the tamper channel*	P
Section 2.3: I2C peripheral limitations	*Section 2.3.1: Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period*	A
	Section 2.3.2: BSY bit may stay high at the end of a data transfer in slave mode	A
	Section 2.3.3: Spurious bus error detection in Master mode	A
Section 2.4: USART peripheral limitations	*Section 2.4.1: Start bit detected too soon when sampling for NACK signal from the smartcard*	N
	Section 2.4.2: Break request can prevent the Transmission Complete flag (TC) from being set	A
	Section 2.4.3: nRTS is active while RE or UE = 0	A
Section 2.5: FMC peripheral limitation	*Section 2.5.1: Dummy read cycles inserted when reading synchronous memories*	N
	Section 2.5.2: Wrong data read from a busy NAND memory	A
	Section 2.5.3: Missed clocks with continuous clock feature enabled	A
Section 2.6: QUADSPI peripheral limitations	*Section 2.6.1: Extra data written in the FIFO at the end of a read transfer*	A
Section 2.7: SDMMC1 peripheral limitations	*Section 2.7.1: Wrong CCRCFAIL status after a response without CRC is received*	A
	Section 2.7.2: MMC stream write of less than 8 bytes does not work correctly	A

Table 4. Summary of silicon limitations (continued)

Links to silicon limitations		Revision Z
Section 2.8: BxCAN peripheral limitations	*Section 2.8.1: BxCAN time triggered mode not supported*	N
Section 2.9: Ethernet peripheral limitations	*Section 2.9.1: Incorrect layer 3 (L3) checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads*	A
	Section 2.9.2: The Ethernet MAC processes invalid extension headers in the received IPv6 frames	N
	Section 2.9.3: MAC stuck in the Idle state on receiving the TxFIFO flush command exactly 1 clock cycle after a transmission completes	A
	Section 2.9.4: Transmit frame data corruption	A
	Section 2.9.5: Successive write operations to the same register might not be fully taken into account	A
Section 2.10: ADC peripheral limitations	*Section 2.10.1: ADC sequencer modification during conversion*	A
Section 2.11: DAC peripheral limitations	*Section 2.11.1: DMA underrun flag management*	A
	Section 2.11.2: DMA request not automatically cleared by DMAEN=0	A
Section 2.12: I2S limitations	*Section 2.12.1: Slave desynchronization in PCM short pulse mode*	A

2.1 System limitations

2.1.1 Missed ADC triggers from TIM1/TIM8, TIM2/TIM5/TIM4/TIM6/TRGO or TGRO2 event

Description

The ADC external trigger for regular and injected channels by the TIM1, TIM8, TIM2, TIM5, TIM4 and TIM6 TRGO or TRGO2 events are missed at the following conditions:

- Prescaler enabled on PCLK2 clock.
- TIMxCLK = 2xADCCLK and master mode selection (MMS or MMS2 bits in the TIMx_CR2 timer register) as reset, update, or compare pulse configuration.
- TIMxCLK = 4xADCCLK.

Workarounds

- For TIM1 and TIM8 TRGO or TRGO 2 events: select the trigger detection on both the rising and falling edges. The EXTEN[1:0] or JEXTEN[1:0] bits must be set to 0x11 in the in ADC_CR2 register.
- For TIM2/TIM4/TIM5/TIM6/ TRGO or TGRO2 events: Enable the DAC peripheral clock in the RCC_APB1ENR register.

2.1.2 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

2.1.3 Wakeup from Standby mode when the Back-up SRAM regulator is enabled

Description

When writing to 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 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.

2.2 RTC limitation

2.2.1 Spurious tamper detection when disabling the tamper channel

Description

If the tamper detection is configured for detection on the falling edge event (TAMPFLT=00 and TAMPxTRG=1) and if the tamper event detection is disabled when the tamper pin is at high level, a false tamper event is detected.

Workaround

The false tamper event detection cannot be avoided, but the backup registers erase can be avoided by setting TAMPxNOERASE bit before clearing TAMPxE bit. The two bits must be written in two separate RTC_TAMPCR write accesses.

2.3 I2C peripheral limitations

2.3.1 Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period

Description

The I2C bus specification and user manual specify a minimum data set-up time (tSU;DAT).

The I2C SDA line is not correctly sampled when tSU;DAT is smaller than one I2CCLK (I2C clock) period; the previous SDA value is sampled instead of the current one. This can result in a wrong slave address reception, a wrong received data byte, or a wrong received acknowledge bit.

Workaround

Increase the I2CCLK frequency to get I2CCLK period smaller than the transmitter minimum data set-up time. Or, if it is possible, increase the transmitter minimum data set-up time.

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

2.3.2 BSY bit may stay high at the end of a data transfer in slave mode

Description

BSY flag may sporadically remain high at the end of a data transfer in slave mode. The issue appears when an accidental synchronization happens between internal CPU clock and external SCK clock provided by master.

Conditions

It is related to the end of data transfer detection while the SPI is enabled at slave mode.

Implications

The end of data transaction is not recognized before an entry to low power mode or for a change of the SPI configuration (e.g. direction of the bidirectional mode). The BSY flag is not a reliable way to handle the end of a data frame transmission when it is used to signal the end of this transaction with master.

Workaround

1. When in a SPI receiving mode, the end of a transaction with master can be detected by the corresponding RXNE event, when this flag is set after the last bit of that transaction sampled and the received data stored.
2. When the following sequence is used, the condition of the synchronization issue is prevented and the BSY bit works correctly. The BSY flag then can be used to recognize end of any transmission transaction (including the case of bidirectional mode when RXNE is not raised):
   - Write last data to data register.
   - Poll TXE till it becomes high to ensure the data transfer has started.
   - Disable SPI by clearing SPE while the last data transfer is still on going.
   - Poll the BSY bit till it becomes low.

Note: *This workaround can be used only when CPU has enough performance to disable SPI after TXE event is detected while the data frame transfer is still ongoing. It is impossible to achieve it when ratio between CPU and SPI clock is low and data frame is short especially. At this specific case timeout can be measured from TXE event calculating fixed number of CPU clock periods corresponding to the data frame transaction.*

2.3.3 Spurious bus error detection in Master mode

Description

In Master mode, a bus error can be detected by mistake, so the BERR flag can be wrongly raised in the status register. This will generate a spurious Bus Error interrupt if the interrupt is enabled. A bus error detection has no effect on the transfer in master mode, therefore the I2C transfer can continue 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 on-going transfer can be handled normally.

2.4 USART peripheral limitations

2.4.1 Start bit detected too soon when sampling for NACK signal from the smartcard

Description

In the ISO7816, when a character parity error is incorrect, the smartcard receiver shall transmit a NACK error signal at (10.5 +/- 0.2) etu after the character START bit falling edge. In this case, the USART transmitter should be able to detect correctly the NACK signal by sampling at (11.0 +/-0.2) etu after the character START bit falling edge.

The USART peripheral used in smartcard mode doesn't respect the (11 +/-0.2) etu timing, and when the NACK falling edge arrives at 10.68 etu or later, the USART might misinterpret this transition as a START bit even if the NACK is correctly detected.

Workaround

None.

2.4.2 Break request can prevent the Transmission Complete flag (TC) from being set

Description

After the end of transmission of a data (D1), the Transmission Complete (TC) flag will not be set in the following conditions:

- CTS hardware flow control is enabled.
- D1 is being transmitted.
- A break transfer is requested before the end of D1 transfer.
- nCTS is de-asserted before the end of transfer of D1.

Workaround

If the application needs to detect the end of transfer of the data, the break request should be done after making sure that the TC flag is set.

2.4.3 nRTS is active while RE or UE = 0

Description

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

Workaround

Configure the I/O used for nRTS as alternate function after setting the UE and RE bits.

2.5 FMC peripheral limitation

2.5.1 Dummy read cycles inserted when reading synchronous memories

Description

When performing a burst read access to a synchronous memory, two dummy read accesses are performed at the end of the burst cycle whatever the type of AHB burst access. However, the extra data values which are read are not used by the FMC and there is no functional failure.

Workaround

None.

2.5.2 Wrong data read from a busy NAND memory

Description

When the read command is issued to the NAND memory, the R/B signal gets activated upon the de-assertion of the chip select. In case a read transaction is pending, the NAND controller might not detect the R/B signal (connected to NWAIT) previously asserted, so that it will sample a wrong data. The problem occurs only when using MEMSET timing is configured to 0 or while ATTHOLD timing is configured to 0 or 1.

Workaround

Either configure MEMSET timing to a value greater than 0 or ATTHOLD timing to a value greater than 1.

2.5.3 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:

- FMC_CLK clock divided by 2
- An asynchronous byte transaction is performed on a FMC bank configured in 32-bit memory data width. When the FMC_CLK clock for static memories is switched OFF, it will be switched ON when issuing a synchronous transaction or any asynchronous transaction different from byte access on 32-bit memory width.

Workaround

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

2.6 QUADSPI peripheral limitations

2.6.1 Extra data written in the FIFO at the end of a read transfer

Description

When all the conditions listed below are gathered:

- QUADSPI is used in indirect mode.
- QUADSPI clock is AHB/2 (PRESCALER = 0x01 in the QUADSPI_CR).
- QUADSPI is in quad mode (DMODE = 0b11 in the QUADSPI_CCR).
- QUADSPI is in DDR mode (DDRM = 0b1 in the QUADSPI_CCR).

An extra data is incorrectly written in the FIFO when a data is read at the same time that the FIFO gets full at the end of a read transfer.

Workarounds

One of the two workarounds listed below can be done:

- Read out the extra data until the BUSY flag goes low and discard it.
- Request an abort after reading out all the correct received data from FIFO in order to flush FIFO and have the busy low. Abort will keep the last register configuration (set the ABORT bit in the QUADSPI_CR).

2.7 SDMMC1 peripheral limitations

2.7.1 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 CCRCFAILC bit in the SDIO_ICR register after reception of the response to the CMD5 command.

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

Description

When 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 will write data from the host until a STOP_TRANSMISSION (CMD12) command is received.

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

The WAITPEND bit 9 of 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 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 single block write command (WRITE_BLOCK_CMD24) instead of stream write (CMD20) with the desired block length.

2.8 BxCAN peripheral limitations

2.8.1 BxCAN time triggered mode not supported

Description

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

Workaround

None.

2.9 Ethernet peripheral limitations

2.9.1 Incorrect layer 3 (L3) checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads

Description

The application provides the per-frame control to instruct the MAC to insert the L3 checksums for TCP, UDP and ICMP packets. When automatic checksum insertion is enabled and the input packet is an IPv6 packet without the TCP, UDP or ICMP payload, then the MAC may incorrectly insert a checksum into the packet. For IPv6 packets without a TCP, UDP or ICMP payload, the MAC core considers the next header (NH) field as the extension header and continues to parse the extension header. Sometimes, the payload data in such packets matches the NH field for TCP, UDP or ICMP and, as a result, the MAC core inserts a checksum.

Workaround

When the IPv6 packets have a TCP, UDP or ICMP payload, enable checksum insertion for transmit frames, or bypass checksum insertion by using the CIC (checksum insertion control) bits in TDES0 (bits 23:22).

2.9.2 The Ethernet MAC processes invalid extension headers in the received IPv6 frames

Description

In IPv6 frames, there can be zero or some extension headers preceding the actual IP payload. The Ethernet MAC processes the following extension headers defined in the IPv6 protocol: Hop-by-Hop Options header, Routing header and Destination Options header. All extension headers, except the Hop-by-Hop extension header, can be present multiple times and in any order before the actual IP payload. The Hop-by-Hop extension header, if present, has to come immediately after the IPv6’s main header.

The Ethernet MAC processes all (valid or invalid) extension headers including the Hop-by-Hop extension headers that are present after the first extension header. For this reason, the GMAC core will accept IPv6 frames with invalid Hop-by-Hop extension headers. As a consequence, it will accept any IP payload as valid IPv6 frames with TCP, UDP or ICMP payload, and then incorrectly update the receive status of the corresponding frame.

Workaround

None.

2.9.3 MAC stuck in the Idle state on receiving the TxFIFO flush command exactly 1 clock cycle after a transmission completes

Description

When the software issues a TxFIFO flush command, the transfer of frame data stops (even in the middle of a frame transfer). The TxFIFO read controller goes into the Idle state (TFRS=00 in ETH_MACDBGR) and then resumes its normal operation.

However, if the TxFIFO read controller receives the TxFIFO flush command exactly one clock cycle after receiving the status from the MAC, the controller remains stuck in the Idle state and stops transmitting frames from the TxFIFO. The system can recover from this state only with a reset (e.g. a soft reset).

Workaround

Do not use the TxFIFO flush feature.

If TXFIFO flush is really needed, wait until the TxFIFO is empty prior to using the TxFIFO flush command.

2.9.4 Transmit frame data corruption

Frame data corrupted when the TxFIFO is repeatedly transitioning from non empty to empty and then back to non empty.

Description

Frame data may get corrupted when the TxFIFO is repeatedly transitioning from non empty to empty for a very short period, and then from empty to non empty, without causing an underflow.

This transitioning from non empty to empty and back to non empty happens when the rate at which the data is being written to the TxFIFO is almost equal to or a little less than the rate at which the data is being read.

This corruption cannot be detected by the receiver when the CRC is inserted by the MAC, as the corrupted data is used for the CRC computation.

Workaround

Use the Store-and-Forward mode: TSF=1 (bit 21 in ETH_DMAOMR). In this mode, the data is transmitted only when the whole packet is available in the TxFIFO.

2.9.5 Successive write operations to the same register might not be fully taken into account

Description

A write to a register might not be fully taken into account if a previous write to the same register is performed within a time period of four TX_CLK/RX_CLK clock cycles. When this error occurs, reading the register returns the most recently written value, but the Ethernet MAC continues to operate as if the latest write operation never occurred.

See *Table 5: Impacted registers and bits* for the registers and bits impacted by this limitation.

Table 5. Impacted registers and bits

Register name	Bit number	Bit name
DMA registers		
ETH_DMABMR	7	EDFE
ETH_DMAOMR	26	DTCEFD
	25	RSF
	20	FTF
	7	FEF
	6	FUGF
	4:3	RTC
GMAC registers		

Table 5. Impacted registers and bits (continued)

Register name	Bit number	Bit name
ETH_MACCR	25	CSTF
	23	WD
	22	JD
	19:17	IFG
	16	CSD
	14	FES
	13	ROD
	12	LM
	11	DM
	10	IPCO
	9	RD
	7	APCS
	6:5	BL
	4	DC
	3	TE
	2	RE
ETH_MACFFR	-	MAC frame filter register
ETH_MACHTHR	31:0	Hash Table High Register
ETH_MACHTLR	31:0	Hash Table Low Register
ETH_MACFCR	31:16	PT
	7	ZQPD
	5:4	PLT
	3	UPFD
	2	RFCE
	1	TFCE
	0	FCB/BPA
ETH_MACVLANTR	16	VLANTC
	15:0	VLANTI
ETH_MACRWUFFR	-	all remote wakeup registers
ETH_MACPMTCSR	31	WFFRPR
	9	GU
	2	WFE
	1	MPE
	0	PD
ETH_MACA0HR	-	MAC address 0 high register

Table 5. Impacted registers and bits (continued)

Register name	Bit number	Bit name
ETH_MACA0LR	-	MAC address 0 low register
ETH_MACA1HR	-	MAC address 1 high register
ETH_MACA1LR	-	MAC address 1 low register
ETH_MACA2HR	-	MAC address 2 high register
ETH_MACA2LR	-	MAC address 2 low register
ETH_MACA3HR	-	MAC address 3 high register
ETH_MACA3LR	-	MAC address 3 low register
IEEE 1588 time stamp registers		
ETH_PTPTSCR	18	TSPFFMAE
	17:16	TSCNT
	15	TSSMRME
	14	TSSEME
	13	TSSIPV4FE
	12	TSSIPV6FE
	11	TSSPTPOEFE
	10	TSPTPPSV2E
	9	TSSSR
	8	TSSARFE
	5	TSARU
	3	TSSTU
	2	TSSTI
	1	TSFCU
	0	TSE

Workarounds

Two workarounds could be applicable:

- Ensure a delay of four TX_CLK/RX_CLK clock cycles between the successive write operations to the same register.
- Make several successive write operations without delay, then read the register when all the operations are complete, and finally reprogram it after a delay of four TX_CLK/RX_CLK clock cycles.

2.10 ADC peripheral limitations

2.10.1 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.

2.11 DAC peripheral limitations

2.11.1 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 servicing the DAC.

Workaround

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

2.11.2 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 will be pending while the DAC is reinitialized and restarted; with the risk that a spurious unwanted DMA request is serviced 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.

2.12 I2S limitations

2.12.1 Slave desynchronization in PCM short pulse mode

Description

When the I2S is configured in Slave PCM short frame synchronization mode and the asynchronous start is disabled (Bit ASTRTEN of the SPIx_I2SCFGR register set to 0), the data received or transmitted by the slave might be corrupted. Note that having ASTRTEN bit set to 0 in the case of PCM short frame synchronization mode is useless as the width of the frame synchronization pulse is one period of the bit clock.

Workaround

If the I2S is configured in Slave PCM short frame synchronization mode, the bit ASTRTEN must be set to 1. For all other I2S modes the bit ASTRTEN must be set 0.

3 Revision history

Table 6. Document revision history

Date	Revision	Changes
19-Jan-2015	1	Initial release.
26-May-2015	2	Updated *Section 2.7: SDMMC1 peripheral limitations* removing limitation: – Section: SDIO interrupt period not correctly handled when 4 data lines are selected. – Section: Wait for response bits “10” configuration does not work correctly. Added *Section 2.4: USART peripheral limitations*: – *Section 2.4.1: Start bit detected too soon when sampling for NACK signal from the smartcard*. – *Section 2.4.2: Break request can prevent the Transmission Complete flag (TC) from being set*. – *Section 2.4.3: nRTS is active while RE or UE = 0*. Updated *Section 2.1: System limitations* removing: – Section: Extra current consumption in Standby mode when PC13 or PI8 pins are left floating. – Section: Extra consumption in Standby mode when wakeup pins configured in falling edge. – Section: Standby mode entry. – Section LSE Oscillator not working at hot temperature. – Section: Bypass regulator mode forcing all wakeup pins in input mode. – Section: ADC external triggers unavailability – Section: User option byte update when RDP level 1 is active and BOOT pin is high Updated *Section 2.1.1: Missed ADC triggers from TIM1/TIM8, TIM2/TIM5/TIM4/TIM6/TRGO or TGRO2 event* titles and description. Updated the whole document: – Changing revision A into the revision Z. – Updating with STM32F75xxx and STM32F74xxx. Updated *Table 2: Device summary* adding STM32F745xx RPNs. Removed “NAND/PCCard transaction and wait timing” limitation. Added *Section 1: ARM® 32-bit Cortex®-M7 with FPU limitations*.
07-Jan-2016	3	Added limitations: – *Section 2.1.3: Wakeup from Standby mode when the Back-up SRAM regulator is enabled*. in System limitation section. – *Section 2.12.1: Slave desynchronization in PCM short pulse mode* in I2S limitation section. – *Section 2.3.3: Spurious bus error detection in Master mode* in I2C limitation section. – *Section 2.5.2: Wrong data read from a busy NAND memory* and *Section 2.5.3: Missed clocks with continuous clock feature enabled* in FMC limitation section.