STM32F76xxx STM32F77xxx Errata sheet

STM32F76xxx and STM32F77xxx device limitations

Silicon identification

This errata sheet applies to the revisions A and Z of STMicroelectronics STM32F76xxx and STM32F77xxx 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
STM32F76xxx	"A" and "Z"
STM32F77xxx	

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see the STM32F76xxx and STM32F77xxx reference manual (RM0410) 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
STM32F76xxx	STM32F765BG, STM32F765BI, STM32F765IG, STM32F765II, STM32F765NG, STM32F765NI, STM32F765VG, STM32F765VI, STM32F765ZG, STM32F765ZI, STM32F767BG, STM32F767BI, STM32F767IG, STM32F767II, STM32F767NG, STM32F767NI, STM32F767VG, STM32F767VI, STM32F767ZG, STM32F767ZI, STM32F768AI, STM32F769AG, STM32F769AI, STM32F769BG, STM32F769BI, STM32F769IG, STM32F769II, STM32F769NG, STM32F769NI
STM32F77xxx	STM32F777BI, STM32F777II, STM32F777NI, STM32F777VI, STM32F777ZI, STM32F778AI, STM32F779AI, STM32F779BI, STM32F779II, STM32F779NI

Contents

List of tables

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

An errata notice of the STM32F76xxx and STM32F77xxx 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 3 summarizes these limitations and their implications on the behavior of the STM32F76xxx and STM32F77xxx devices.

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

ARM ID	ARM category	Impact on STM32F76xxx and STM32F77xxx devices
851031	Cat C	Minor
850725	Cat C	Minor
850724	Cat C	Minor

2 STM32F76xxx and STM32F77xxx 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 A	Revision Z
Section 2.1: System limitations	*Section 2.1.1: Internal noise impacting the ADC accuracy*	A	A
	Section 2.1.2: Wakeup from Standby mode when the back-up SRAM regulator is enabled	A	A
	Section 2.1.3: LSE high driving and low driving capability is not usable for TFBGA216 package under certain conditions	A	-
	Section 2.1.4: DTCM-RAM not accessible in read when the MCU is in Sleep mode (WFI/WFE)	A	-
Section 2.2: I2C peripheral limitations	*Section 2.2.1: Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period*	A	A
	Section 2.2.2: BSY bit may stay high at the end of a data transfer in slave mode	A	A
	Section 2.2.3: Spurious bus error detection in Master mode	A	A
	Section 2.2.4: 10-bit Master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave	A	A
Section 2.3: USART peripheral limitations	*Section 2.3.1: nRTS is active while RE or UE = 0*	A	A
Section 2.4: FMC peripheral limitation	*Section 2.4.1: Dummy read cycles inserted when reading synchronous memories*	N	N
	Section 2.4.2: Wrong data read from a busy NAND memory	A	A
	Section 2.4.3: Missed clocks with continuous clock feature enabled	A	A
Section 2.5: SDMMC peripheral limitations	*Section 2.5.1: Wrong CCRCFAIL status after a response without CRC is received*	A	A
	Section 2.5.2: MMC stream write of less than 8 bytes does not work correctly	A	A
Section 2.6: BxCAN peripheral limitations	*Section 2.6.1: BxCAN time triggered mode not supported*	N	N

Table 4. Summary of silicon limitations (continued)

Links to silicon limitations		Revision A	Revision Z
Section 2.7: Ethernet peripheral limitations	*Section 2.7.1: Incorrect layer 3 (L3) checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads*	A	A
	Section 2.7.2: The Ethernet MAC processes invalid extension headers in the received IPv6 frames	N	N
	Section 2.7.3: MAC stuck in the idle state on receiving the TxFIFO flush command exactly 1 clock cycle after a transmission completes	A	A
	Section 2.7.4: Transmit frame data corruption	A	A
	Section 2.7.5: Successive write operations to the same register might not be fully taken into account	A	A
	Section 2.7.6: Ethernet erroneous data received in RMII configuration	N	-
Section 2.8: ADC peripheral limitations	*Section 2.8.1: ADC sequencer modification during conversion*	A	A
Section 2.9: DAC peripheral limitations	*Section 2.9.1: DMA underrun flag management*	A	A
	Section 2.9.2: DMA request not automatically cleared by DMAEN=0	A	A
Section 2.10: I2S limitations	*Section 2.10.1: Slave desynchronization in PCM short pulse mode*	A	A
Section 2.11: DSI Host peripheral limitations	*Section 2.11.1: When used over the DSI link, the tearing effect interrupt flag is set when an acknowledge trigger is received from the display*	A	A
	Section 2.11.2: The time to activate the clock between HS transmissions is not calculated correctly	A	A
	Section 2.11.3: The immediate update procedure may fail	A	A
Section 2.12: QUADSPI peripheral limitations	*Section 2.12.1: First nibble of data is not written after a dummy phase*	A	A
Section 2.13: JPEG peripheral limitations	*Section 2.13.1: False EOI marker is inserted after clearing the HDR bit*	A	A
	Section 2.13.2: No DMA transfer complete generated at the end of the encoding process after clearing the HDR bit	A	A

2.1 System limitations

2.1.1 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

2.1.2 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.

2.1.3 LSE high driving and low driving capability is not usable for TFBGA216 package under certain conditions

Description

On the TFBGA216 package when the LSE low driving capability or LSE High driving capability is selected (LSEDRV[1:0]=00 or LSEDRV[1:0]=11 in the RCC_BDCR register, respectively) for the LSE oscillator, the oscillation stability is impacted by toggling the MCU pins near the LSE input pin at relatively high-frequency.

The TFBGA216 pins impacting the LSE stability are: PF0, PF1, PI11 and PI12

Impact: under the above described conditions, intermittent LSE clock pulse losses (in low driving capability) or intermittent LSE clock pulse add-ons (in high driving capability) are possible.

Workaround

On the TFBGA216 package do not select the LSE high driving capability or the LSE low driving capability, and:

- Use the LSE medium high driving capability (LSEDRV[1:0]=01 in the RCC_BDCR register)
- Or the LSE medium low driving capability (LSEDRV[1:0]=10 in the RCC_BDCR register)

2.1.4 DTCM-RAM not accessible in read when the MCU is in Sleep mode (WFI/WFE)

Description

- The DTCM-RAM is not accessible in read during Sleep mode (when the CPU clock is gated). When a read access to the DTCM-RAM is performed by an AHB bus master (that are the DMAs) while the CPU is in sleep mode (CPU clock is gated), the data is not transmitted to the AHB bus and the AHB master reads 0x0000_0000.
- There is no issue when a write is performed to the DTCM-RAM while the CPU is in sleep mode, the data is correctly written in the DTCM-RAM.

Workaround

Use the AXI SRAM1 or SRAM2 for DMA data read transfers and use the AXI DTCM-RAM for DMA data write transfers in Sleep mode.

2.2 I2C peripheral limitations

2.2.1 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 the 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.

2.2.2 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. The issue appears when an accidental synchronization happens between the 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 the 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 the CPU has enough performance to disable the SPI after the TXE event is detected while the data frame transfer is still ongoing. It is impossible to achieve it when the ratio between the CPU and the SPI clock is low and the data frame is short especially. At this specific case, timeout can be measured from the TXE event calculating fixed number of CPU clock periods corresponding to the data frame transaction.

2.2.3 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.

2.2.4 10-bit Master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave

Description

In Master mode, the master automatically sends a STOP bit when the slave has not acknowledged a byte during the address transmission.

In 10-bit addressing mode, if the first part of the 10-bit address (corresponding to 10-bit header + 2 MSB) has not been acknowledged by the slave, the STOP bit is sent but the START bit is not cleared and the master cannot launch a new transfer.

Workaround

When the I2C is configured in 10-bit addressing Master mode and the NACKF status flag is set in the I2C_ISR register while the START bit is still set in the I2C_CR2 register, then proceed as follows:

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

2.3 USART peripheral limitations

2.3.1 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) 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.

2.4 FMC peripheral limitation

2.4.1 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.

2.4.2 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.

2.4.3 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 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.

2.5 SDMMC peripheral limitations

2.5.1 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.

2.5.2 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 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 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 single block

write command (WRITE_BLOCK_CMD24) instead of stream write (CMD20) with the desired block length.

2.6 BxCAN peripheral limitations

2.6.1 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.

2.7 Ethernet peripheral limitations

2.7.1 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 an 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).

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

Description

In the 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 the 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.

2.7.3 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.

2.7.4 Transmit frame data corruption

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

Description

The 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.

2.7.5 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		
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

Table 5. Impacted registers and bits (continued)

Register name	Bit number	Bit name
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
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		

Table 5. Impacted registers and bits (continued)

Register name	Bit number	Bit name
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.

2.7.6 Ethernet erroneous data received in RMII configuration

Description

- In the reduced media-independent interface (RMII) configuration, an erroneous data might be received on the RXD0 signal (PC4). The bit received might flip from 0 to 1 and lead to a received frame with a CRC error. The ETH_MMCRFCECR register increments each time a frame is received. This is related to internal timing constraints on the reference clock generated after the sync divider.
- Using the RMII reference clock of 50 MHz, the error is seen for both cases:
  - 100-Mbit/s operating rate (sync divider = div2)
  - 10-Mbit/s operating rate (sync divider = div20)
- The issue is not present in the MII mode with a direct reference clock from the pad (no division).

Workaround

None

2.8 ADC peripheral limitations

2.8.1 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.

2.9 DAC peripheral limitations

2.9.1 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.

2.9.2 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 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.

2.10 I2S limitations

2.10.1 Slave desynchronization in PCM short pulse mode

Description

When the I2S is configured in the 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 the ASTRTEN bit set to 0 in the case of the 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 the 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.

2.11 DSI Host peripheral limitations

2.11.1 When used over the DSI link, the tearing effect interrupt flag is set when an acknowledge trigger is received from the display

Description

In the adapted command mode, when the tearing effect mechanism is used over the DSI link, the Tearing Effect Interrupt Flag (TEIF) of the DSI Wrapper Interrupt Status Register (DSI_WISR) is asserted when an acknowledge trigger is received from the display.

An acknowledge trigger can be received from the display:

- For each packet when the Acknowledge Request Enable (ARE) bit of the DSI Host Command Mode Configuration Register (DSI_CMCR) is set,
- When a response is awaited from the display.

Workaround

Do not use the tearing effect over the link but use the dedicated TE pin.

When using the tearing effect over the link, do not use the tearing effect interrupt nor the automatic refresh mode, but launch the display refresh immediately after a set_tear_on or a set_scanline DCS command (as the display is driving the DSI link until the tearing effect occurs, the refresh will be automatically stalled until the tearing effect).

2.11.2 The time to activate the clock between HS transmissions is not calculated correctly

Description

In the automatic clock lane control mode, the DSI Host can turn off the clock lane between two high-speed transmissions.

To do so, the DSI Host calculates the time required for the clock lane to change from high-speed to low-power and from low-power to high-speed.

These timings are configured by the HS2LP_TIME and LP2HS_TIME in the DSI Host Clock Lane Timer Configuration Register (DSI_CLTCR). The DSI Host is not calculating LP2HS_TIME + HS2LP_TIME but 2 x HS2LP_TIME instead.

Workaround

Configure HS2LP_TIME and LP2HS_TIME with the same value as the max between HS2LP_TIME and LP2HS_TIME.

As an example, if HS2LP_TIMER = 44 and LP2HS_TIME = 113 configure the register fields as follows:

- HS2LP_TIME = 113,
- LP2HS_TIME =113.

2.11.3 The immediate update procedure may fail

Description

The immediate update procedure implies that both the Update Register (UR) and the Enable (EN) bits of the DSI Host Video Shadow Control Register (DSI_VSCR) are initially cleared, and are set by the same instruction.
Because of a race condition between the two signals, this immediate update procedure may fail in few cases, leading the DSI Host to wait until the next frame end before updating the configuration.

Workaround

After an immediate update procedure, verify if the configuration is updated by reading the auto-cleared bit UR.

If the UR bit is not cleared, repeat the process by writing first 0x0000 then 0x0101 in DSIHOST_VSCR.

2.12 QUADSPI peripheral limitations

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

Example:

To generate:

- 1 dummy cycle: send 1 alternate-byte, possible only in 4 data line DDR mode or Dual-flash SDR mode
- 2 dummy cycles: send 1 alternate-byte in 4 data line SDR mode
- 4 dummy cycles: send 2 alternate-bytes in 4 data line SDR mode or send 1 alternate-byte in 2 data line SDR mode
- 8 dummy cycles: send 1 alternate-byte in 1 data line SDR mode

2.13 JPEG peripheral limitations

2.13.1 False EOI marker is inserted after clearing the HDR bit

Description

An extra end of image (EOI) marker (0xFFD9) is written automatically into the output FIFO at the end of an encoding process with header processing. If the HDR mode is enabled and when the software clears the HDR bit at the end of the encoding process before making a software reset, an extra data (EOI marker = 0xFFD9) is inserted into the output FIFO. it implies that the extra data might be outputted from the FIFO and stored in a RAM

Workaround

The software must clear the EOC flag and perform a software reset before changing the HDR bit configuration in the JPEG codec configuration register 1 (JPEG_CONFR1).

2.13.2 No DMA transfer complete generated at the end of the encoding process after clearing the HDR bit

If the JPEG is configured as the DMA flow controller, the DMA might enter an infinite wait due to the fact that the JPEG does not generate the correct last data request for it.

The JPEG might not generate the correct last request if the following conditions are met:

- The software clears the HDR bit at the end of the encoding process
- The encoding process has the header processing enabled (the EOC flag is asserted and no software reset is performed)
- And the FIFO level is equal to the threshold.

Workaround

- The software must clear the EOC flag and perform a software reset before changing the HDR bit configuration in the JPEG codec configuration register 1 (JPEG_CONFR1).
- Or use the DMA as the flow controller

3 Revision history

Table 6. Document revision history

Date	Revision	Changes
18-Feb-2016	1	Initial release.
21-Apr-2016	2	Added QUADSPI peripheral limitation: *Section 2.12.1: First nibble of data is not written after a dummy phase*. Added system limitation: *Section 2.1.3: LSE high driving and low driving capability is not usable for TFBGA216 package under certain conditions*.
29-Sep-2016	3	Added system limitation: *Section 2.1.4: DTCM-RAM not accessible in read when the MCU is in Sleep mode (WFI/WFE)*. Added ethernet limitation: *Section 2.7.6: Ethernet erroneous data received in RMII configuration*. Added JPEG limitations: – *Section 2.13.1: False EOI marker is inserted after clearing the HDR bit*. – *Section 2.13.2: No DMA transfer complete generated at the end of the encoding process after clearing the HDR bit*. Added I2C limitation: *Section 2.2.4: 10-bit Master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave*. Removed USART limitations: – Start bit detected too soon when sampling for NACK signal from the smartcard. – Break request can prevent the Transmission Complete flag (TC) from being set. Updated *Table 2: Device summary* adding STM32F765xx devices.
21-Oct-2016	4	Added revision Z: – Updated *Table 1: Device identification*. – Updated *Table 4: Summary of silicon limitations* with two system limitations and one ethernet limitation marked as ‘fixed’.