STM32F378xx Errata sheet

STM32F378xx device limitations

Silicon identification

This errata sheet applies to revision B of the STMicroelectronics STM32F378xx products. These families feature an ARM® 32-bit Cortex®-M4 core, for which an errata notice is also available (see *Section 1* for details).

Section 2 gives a detailed description of the product silicon limitations.

The full list of part numbers is shown in *Table 2*. 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 device
STM32F378xx	“B”

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see the RM0313 reference manual for details on how to find the revision code).
2. Refer to the device datasheet for details on how to identify the revision code on the different packages.

Table 2. Device summary

Reference	Part number
STM32F378xx	STM32F378CC, STM32F378RC, STM32F378VC

Contents

List of tables

1 ARM® 32-bit Cortex®-M4 limitations

An errata notice of the STM32F378xx core is available from the following web address: http://infocenter.arm.com.

All the described limitations are minor and related to the revision r0p1-v1 of the Cortex®-M4 core. *Table 3* summarizes these limitations and their implications on the behavior of the STM32F378xx devices.

Table 3. Cortex®-M4 core limitations and impact on microcontroller behavior

ARM ID	ARM category	ARM summary of errata	Impact on STM32F378xx
752770	Cat B	Interrupted loads to SP can cause erroneous behavior	Minor
776924	Cat B	VDIV or VSQRT instructions might not complete correctly when very short ISRs are used	Minor

1.1 Cortex®-M4 interrupted loads to stack pointer can cause erroneous behavior

Description

An interrupt occurring during the data-phase of a single word load to the stack pointer (SP/R13) can cause an erroneous behavior of the device. In addition, returning from the interrupt results in the load instruction being executed with an additional time.

For all the instructions performing an update of the base register, the base register is erroneously updated on each execution, resulting in the stack pointer being loaded from an incorrect memory location.

The instructions affected by this limitation are the following:

- LDR SP, [Rn],#imm
- LDR SP, [Rn,#imm]!
- LDR SP, [Rn,#imm]
- LDR SP, [Rn]
- LDR SP, [Rn,Rm]

Workaround

As of today, no compiler generates these particular instructions. This limitation can only occur with hand-written assembly code.

Both issues can be solved by replacing the direct load to the stack pointer by an intermediate load to a general-purpose register followed by a move to the stack pointer.

Example:

Replace LDR SP, [R0] by

LDR R2,[R0]

MOV SP,R2

1.2 VDIV or VSQRT instructions might not complete correctly when very short ISRs are used

Description

On the Cortex®-M4 with FPU core, 14 cycles are required to execute a VDIV or VSQRT instruction.
This limitation is present when the following conditions are met:

- A VDIV or VSQRT is executed
- The destination register for VDIV or VSQRT is one of s0 - s15
- An interrupt occurs and is taken
- The ISR being executed does not contain a floating point instruction
- 14 cycles after the VDIV or VSQRT is executed, an interrupt return is executed.

In this case, if there are only one or two instructions inside the interrupt service routine, then the VDIV or VQSRT instruction does not complete correctly and the register bank and FPSCR are not updated, meaning that these registers hold incorrect out-of-date data.

Workarounds

Two workarounds are applicable:

- Disable the lazy context save of floating point state by clearing LSPEN to 0 (bit 30 of the FPCCR at address 0xE000EF34).
- Ensure that every ISR contains more than 2 instructions in addition to the exception return instruction.

2 STM32F378xx 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 B
Section 2.1: System limitations	*Section 2.1.1: Wakeup sequence from Standby mode when using more than one wakeup source*	A
	Section 2.1.2: Delay after an RCC peripheral clock enabling	A
	Section 2.1.3: Full JTAG configuration without NJTRST pin cannot be used	A
Section 2.2: USART peripheral limitations	*Section 2.2.1: Communication parameters reprogramming after ATR in Smartcard mode when SCLK is used to clock the card*	A
	Section 2.2.2: Last byte written in TDR might not be transmitted if TE is cleared just after writing in TDR	A
	Section 2.2.3: Start bit detected too soon when sampling for NACK signal from the smartcard	N
	Section 2.2.4: Break request can prevent the Transmission Complete flag (TC) from being set	A
	Section 2.2.5: nRTS is active while RE or UE = 0	A
	Section 2.2.6: Receiver timeout counter starting in case of a 2 stop bit configuration	A
Section 2.3: SDADC peripheral limitation	*Section 2.3.1: SDADC incorrect gain amplification in single-ended zero reference mode for 16x and 32x gains*	A
	Section 2.3.2: SDADC crosstalk into another SDADCs	A
Section 2.4: SPI/I2S peripheral limitations	*Section 2.4.1: Packing mode limitation at reception*	N
	Section 2.4.2: SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI finishes its DMA transaction	N
	Section 2.4.3: In I2S slave mode, WS level must be set by the external master when enabling the I2S	A
	Section 2.4.4: BSY bit may stay high at the end of a data transfer in Slave mode	A
	Section 2.4.5: Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback	A
	Section 2.4.6: Wrong CRC transmitted in Master mode with delayed SCK feedback	A

Table 4. Summary of silicon limitations (continued)

Links to silicon limitations		Revision B
Section 2.5: I2C peripheral limitations	*Section 2.5.1: 10-bit Slave mode: wrong direction bit value after read header reception*	A
	Section 2.5.2: 10-bit combined with 7-bit Slave mode: ADDCODE may indicate wrong slave address detection	N
	Section 2.5.3: Wakeup frames may not wake up the MCU mode when STOP mode entry follows I2C enabling	A
	Section 2.5.4: Wrong behavior related with MCU Stop mode when the wakeup from Stop mode by I2C peripheral is disabled	A
	Section 2.5.5: Wakeup frame may not wakeup from STOP if tHD(STA) is close to startup time	N
	Section 2.5.6: Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period	A
	Section 2.5.7: Spurious bus error detection in master mode	A
	Section 2.5.8: 10-bit Master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave	A
Section 2.6: GPIO peripheral limitation	*Section 2.6.1: GPIOx locking mechanism is not working properly for GPIOx_OTYPE register*	A
Section 2.7: Touch sensing peripheral limitation	*Section 2.7.1: Inhibited acquisition in case of short transfer phase configuration*	A
Section 2.8: RTC limitation	*Section 2.8.1: RTC calendar registers are not locked properly*	A
Section 2.9: BxCAN peripheral limitations	*Section 2.9.1: BxCAN time triggered mode not supported*	N
Section 2.10: Comparator peripheral limitation	*Section 2.10.1: VREFINT scaler startup time from power down parameter degradation*	N

2.1 System limitations

2.1.1 Wakeup sequence from Standby mode when using more than one wakeup source

Description

The various wakeup sources are logically OR-ed in front of the rising-edge detector that generates the wakeup flag (WUF). The WUF flag needs to be cleared prior to the Standby mode entry, otherwise the MCU wakes up immediately.

If one of the configured wakeup sources is kept high during the clearing of the WUF flag (by setting the CWUF bit), it may mask further wakeup events on the input of the edge detector. As a consequence, the MCU may not be able to wake up from the Standby mode.

Workaround

To avoid this limitation, the following sequence should be applied before entering the Standby mode:

- Disable all used wakeup sources,
- Clear all related wakeup flags,
- Re-enable all used wakeup sources,
- Enter Standby mode

Note: When applying this workaround, if one of the wakeup sources is still kept high, the MCU will enter the Standby mode but then it will wake up immediately and generate the power reset.

2.1.2 Delay after an RCC peripheral clock enabling

Description

A delay between an RCC peripheral clock enable and the effective peripheral enabling should be taken into account in order to manage the peripheral read/write from/to registers. This delay depends on the peripheral mapping.

If the peripheral is mapped on AHB: the delay is 2 AHB clock cycles after the clock enable bit is set on the hardware register.

If the peripheral is mapped on APB: the delay is 2 APB clock cycles after the clock enable bit is set on the hardware register.

Workarounds

1. Enable the peripheral clock sometimes before the peripheral read/write register is required.
2. For the AHB peripheral, insert two dummy read to the peripheral register.
3. For the APB peripheral, insert a dummy read to the peripheral register.

2.1.3 Full JTAG configuration without NJTRST pin cannot be used

Description

When using the JTAG debug port in debug mode, the connection with the debugger is lost if the NJTRST pin (PB4) is used as a GPIO. Only the 4-wire JTAG port configuration is impacted.

Workaround

Use the SWD debug port instead of the full 4-wire JTAG port.

2.2 USART peripheral limitations

2.2.1 Communication parameters reprogramming after ATR in Smartcard mode when SCLK is used to clock the card

Description

If the USART is used in Smartcard mode and the card cannot use the default communication parameters after Answer To Reset and does not support clock stop, it is not possible to use SCLK to clock the card. This is due to the fact that the USART and its clock output must be disabled while reprogramming some of the parameters.

Workaround

Use another clock source to clock the card (e.g. a timer output programmed to the desired clock frequency).

2.2.2 Last byte written in TDR might not be transmitted if TE is cleared just after writing in TDR

Description

If the USART clock source is slow (for example LSE) and TE bit is cleared immediately after the last write to TDR, the last byte will probably not be transmitted.

Workarounds

1. Wait until TXE flag is set before clearing TE bit.
2. Wait until TC flag is set before clearing TE bit.

2.2.3 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 does not 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.

2.2.4 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.

2.2.5 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

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

2.2.6 Receiver timeout counter starting in case of a 2 stop bit configuration

Description

In the case of a 2 stop bit configuration, the receiver timeout counter starts counting from the end of the second stop bit of the last character instead of the end of the first stop bit.

Workaround

Change the RTO value in the USARTx_RTOR register by subtracting 1 bit duration.

2.3 SDADC peripheral limitation

2.3.1 SDADC incorrect gain amplification in single-ended zero reference mode for 16x and 32x gains

Description

When using the Single-ended mode (zero reference) for 16x and 32x gains, the device subtracts incorrectly the analog offset when performing the digital multiplication (for 2x and 4x respectively while using analog gain 8x), and the results obtained in Single-ended mode (zero reference) for gains 16x and 32x will be incorrect.

Workaround

Use Single-ended offset mode instead of Single-ended mode (zero reference) for gains 16x and 32x. This mode also features better dynamic characteristics for these gains.

2.3.2 SDADC crosstalk into another SDADCs

Description

When the SDADC mode is changed as follows:

- On/off state change
- From/to the single ended zero reference mode change
- Gain setting change

This change causes a gain change of another SDADC. The SDADC gain change is constant for a given operation mode setting. A crosstalk due to this gain change can cause the following maximum errors (worst case errors got when the measure is done at the full range scale):

- The SDADC1 enable/disable causes an error of about 5 LSB into SDADC2 and SDADC3
- The SDADC2 enable/disable causes an error of about 5 LSB into SDADC1 and SDADC3
- The SDADC3 enable/disable causes an error of about 30 LSB into SDADC1 and SDADC2
- The SDADC1 mode change (from/to single ended zero reference mode) causes an error of about 5 LSB into SDADC2 and SDADC3
- The SDADC2 mode change (from/to single ended zero reference mode) causes an error of about 5 LSB into SDADC1 and SDADC3
- The SDADC3 mode change (from/to single ended zero reference mode) causes an error of about 12 LSB into SDADC1 and SDADC2

Workaround:

1. While an SDADC is performing a conversion then another SDADCs must be in a given constant on/off state.
2. While an SDADC is performing a conversion then another SDADCs must be in a given constant operation mode (constant gain and constant differential/single ended mode). The change between the differential mode and the single ended offset mode is allowed.

2.4 SPI/I2S peripheral limitations

2.4.1 Packing mode limitation at reception

Description

When the SPI is configured in the short data frame mode, the packing mode on the reception side may not be usable. Using this feature may generate a wrong RXNE event to an Interrupt or DMA request and so the software may read back inconsistent data with FIFO pointers misalignment on the reception FIFO.

The worst case is the Slave mode if the external master is running in continuous mode without clock interruption between two data transfers.

In full duplex Master mode, it runs correctly if the SPI is working in non-continuous mode, meaning that the SPI is transferring two data, then stopping the data transmission until the two data received are read back before sending the next two data.

Conditions to see this limitation:

- Packing mode is used
- SPI master (in continuous mode) or Slave (worst case)
- Full duplex or receiver mode

If the packing mode is used in reception mode, the FIFO reception threshold has to be set to 16 bits. Under those setting and conditions, when a read operation (half-word to read two data in one APB access) takes place while the FIFO level is equal to 3/4 (new data came before the two first ones are read), the 16-bit read decreases the FIFO level to 1/4. The RXNE flag is not de-asserted (clear condition on FIFO empty event) and a new request is present to read back two data although the FIFO contains only one data. Read and write pointers in the FIFO become misaligned and the data is corrupted.

The packing mode in reception has to be discarded when the conditions described above are met. It means that the reception FIFO requests that the data is read back until the FIFO content is empty. It also means that for short data frame (the worst case being the 4-bit data size), if the software or the DMA is not able to manage the high data rate when the SPI is running full speed, an Overrun condition may occur at regular intervals.

Workaround

There is no workaround.

The only way to avoid this overrun condition would be to slow down the SPI communication clock frequency in order to let time to the DMA (best case) to read back data without any FIFO full condition.

2.4.2 SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI finishes its DMA transaction

Description

When the SPI is running with the CRC feature enabled at all the modes, the CRC calculation may be frozen and checked corrupted before the CRCNEXT bit is written. It can happen if a peripheral, mapped on the same DMA channel than the SPI, is doing a DMA transfer whereas the SPI is configured to manage data transfers by software (IT or polling). As a consequence the CRC error flag is unduly raised.

Workaround

There is no known workaround of this conflict. If possible use the DMA for SPI transfers to avoid this conflict.

2.4.3 In I2S slave mode, WS level must be set by the external master when enabling the I2S

Description

In slave mode the WS signal level is used only to start the communication. If the I2S (in slave mode) is enabled while the master is already sending the clock and the WS signal level is low (for I2S protocol) or high (for the LSB or MSB-justified mode), the slave starts communicating data immediately. In this case the master and slave will be desynchronized throughout the whole communication.

Workaround

The I2S peripheral must be enabled when the external master sets the WS line at:

- High level when the I2S protocol is selected.
- Low level when the LSB or MSB-justified mode is selected.

2.4.4 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 the external SCK clock provided by the master.

This is related to the end of data transfer detection while the SPI is enabled in Slave mode.

As a consequence, the end of data transaction may be not recognized when the software needs to monitor it (e.g. at the end of session before entering the low-power mode or before direction of data line has to be changed at half duplex bidirectional mode). The BSY flag is unreliable to detect the end of any data sequence transaction.

Workaround

When the NSS hardware management is applied and the NSS signal is provided by the master, the end of a transaction can be detected by the NSS polling by the slave.

- If the SPI receiving mode is enabled, the end of a transaction with master can be detected by the corresponding RXNE event signalizing the last data transfer completion.
- In SPI transmit mode, the user can check the BSY under timeout corresponding to the time necessary to complete the last data frame transaction. The timeout should be measured from TXE event signalizing the last data frame transaction start (it is raised once the second bit transaction is ongoing). Either BSY becomes low normally or the timeout expires when the synchronization issue happens.

When upper workarounds are not applicable, the following sequence can be used to prevent the synchronization issue at SPI transmit mode.

1. Write last data to data register
2. Poll TXE until it becomes high to ensure the data transfer has started
3. Disable SPI by clearing SPE while the last data transfer is still ongoing
4. Poll the BSY bit until it becomes low
5. The BSY flag works correctly and can be used to recognize the end of the transaction.

Note: This workaround can be used only when CPU has enough performance to disable 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 CPU and SPI clock is low and the data frame is short especially. In this specific case the timeout can be measured from TXE, while calculating a fixed number of CPU clock periods corresponding to the time necessary to complete the data frame transaction.

2.4.5 Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback

Description

In receive transaction, in both I^2S and SPI Master modes, the last bit of the transacted frame is not captured when the signal provided by internal feedback loop from the SCK pin exceeds a critical delay. The lastly transacted bit of the stored data then keeps the value from the pattern received previously. As a consequence, the last receive data bit may be wrong and/or the CRCERR flag can be unduly asserted in the SPI mode if any data under check sum and/or just the CRC pattern is wrongly captured.

In SPI mode, data are synchronous with the APB clock. A delay of up to two APB clock periods can thus be tolerated for the internal feedback delay. The I2S mode is more sensitive than the SPI mode since the SCK clock is not synchronized with the APB clock. In this case, the margin of the internal feedback delay is lower than one APB clock period.

The main factors contributing to the delay increase are low V_{DD} level, high temperature, high SCK pin capacitive load and low SCK I/O output speed. The SPI communication speed has no impact.

Workaround

The following workaround can be adopted, jointly or individually:

- Decrease the APB clock speed.
- Configure the IO pad of the SCK pin to be faster.

2.4.6 Wrong CRC transmitted in Master mode with delayed SCK feedback

Description

In transmit transaction of the SPI interface in Master mode with the CRC enabled, the CRC data transmission may be corrupted if the delay of an internal feedback signal derived from the SCK output (further feedback clock) is greater than two APB clock periods. While the data and the CRC bit shifting and the transfer is based on an internal clock, the CRC progressive calculation uses the feedback clock. If the delay of the feedback clock is greater than two APB periods, the transmitted CRC value may get wrong.

The main factors contributing to the delay increase are low V_{DD} level, high temperature, high SCK pin capacitive load and low SCK I/O output speed. The SPI communication speed has no impact.

Workaround

The following workaround can be adopted, jointly or individually:

- Decrease the APB clock speed.
- Configure the IO pad of the SCK pin to be faster.

2.5 I²C peripheral limitations

2.5.1 10-bit Slave mode: wrong direction bit value after read header reception

Description

Under specific conditions, the transfer direction bit DIR (bit 16 of status register I2C_ISR) is low instead of high after reception of the 10-bit addressing read header. Nevertheless, the I²C operates correctly in Slave transmission mode, and data can be sent using the TXIS flag.

To see the limitation, all the following conditions have to be fulfilled:

- I²C has to be configured in 10-bit addressing mode (OA1MODE is set in the I2C_OAR1 register).
- The high LSBs of the I²C slave address are equal to the 10-bit addressing read header value (that is OA1[7:3] = 11110, OA1[2] = OA1[9], OA1[1] = OA1[8] and OA1[0] = 1 in the I2C_OAR1 register).
- The I²C receives the 10-bit addressing read header (0x 1111 0XX1) after the repeated start condition to enter Slave transmission mode.

As a result, the DIR bit is incorrect in Slave mode under specific conditions.

Workaround

If possible, do not use these four values as 10-bit addresses in Slave mode:

- OA1[9:0] = 0011110001
- OA1[9:0] = 0111110011
- OA1[9:0] = 1011110101
- OA1[9:0] = 1111110111

If one of these addresses is the I²C slave address, the DIR bit must not be used in the FW.

2.5.2 10-bit combined with 7-bit Slave mode: ADDCODE may indicate wrong slave address detection

Description

Under specific conditions, the ADDCODE (Address match code) in the I2C_ISR register indicates a wrong slave address.

To see the limitation, all the following conditions have to be fulfilled:

- The I^2C slave address OA1 is enabled and configured in 10-bit mode (OA1EN=1 and OA1MODE=1)
- Another 7-bit slave address is enabled and the bits 1 to 7 of the 10-bit slave address OA1 are equal to the 7-bit slave address, that is one of the configurations below is set:
  - OA2EN=1 and OA2MSK = 0 and OA1[7:1] = OA2[7:1]
  - OA2EN=1 and OA2MSK = 1 and OA1[7:2] = OA2[7:2]
  - OA2EN=1 and OA2MSK = 2 and OA1[7:3] = OA2[7:3]
  - OA2EN=1 and OA2MSK = 3 and OA1[7:4] = OA2[7:4]
  - OA2EN=1 and OA2MSK = 4 and OA1[7:5] = OA2[7:5]
  - OA2EN=1 and OA2MSK = 5 and OA1[7:6] = OA2[7:6]
  - OA2EN=1 and OA2MSK = 6 and OA1[7] = OA2[7]
  - OA2EN=1 and OA2MSK = 7
  - GCEN=1 and OA1[7:1] = 0b0000000
  - ALERTEN=1 and OA1[7:1] = 0b0001100
  - SMBDEN=1 and OA1[7:1] = 0b1100001
  - SMBHEN=1 and OA1[7:1] = 0b0001000
- The master starts a transfer addressed to the 10-bit slave address OA1.

As a result, after the address reception, the ADDCODE value is OA1[7:1] equal to the 7-bit slave address, instead of 0b11110 and OA1[9:8].

Workaround

None. If several slave address are enabled, mixing 10-bit and 7-bit addresses, the 10-bit slave address OA1 [7:1] must not be equal to the 7-bit slave address.

2.5.3 Wakeup frames may not wake up the MCU mode when STOP mode entry follows I^2C enabling

Description

If the I^2C is enabled (PE = 1) and wakeup from STOP is enabled in I^2C (WUPEN=1) while a transfer occurs on the I^2C bus and STOP mode is entered during the same transfer while SCL=0, the I^2C is not able to detect the first following START condition. This means that if the I^2C is addressed, it will not wake up the MCU and this address is not acknowledged.

Workaround

After enabling the I^2C (PE is set to 1), wait for a temporization before entering STOP mode, to ensure that the eventual on-going frame is finished.

2.5.4 Wrong behavior related with MCU Stop mode when the wakeup from Stop mode by I2C peripheral is disabled

Description

When the wakeup from Stop mode by I2C peripheral is disabled (WUPEN = 0) and the MCU enters Stop mode while a transaction is on-going on the I^2C bus, the following wrong operation may occur:

1. The BUSY flag may be 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. This failure may occur in master mode of the I2C peripheral used in multi-master I^2C-bus environment.
2. If I^2C-bus clock stretching is enabled in I2C peripheral (NOSTRETCH = 0), the I2C peripheral may pull SCL low as long as the MCU remains in Stop mode, suspending all I^2C-bus activity during that time. This may occur when the MCU enters Stop mode during the address phase of an I^2C-bus transaction, in low period of SCL. This failure may occur in slave mode of the I2C peripheral or, in master mode of the I2C peripheral used in multi-master I^2C-bus environment. Its probability depends on the timing configuration, operating clock frequency of I2C peripheral and the I^2C-bus timing.

Workaround

Disable the I2C (PE=0) before entering Stop mode and re-enable it in Run mode.

2.5.5 Wakeup frame may not wakeup from STOP if $t_{HD(STA)}$ is close to startup time

Description

Under specific conditions and if the START condition hold time $t_{HD(STA)}$ duration is very close to the HSI startup time duration, the I^2C is not able to detect the address match and to wake up the MCU from STOP. To see the limitation, one of the conditions listed below has to be met:

- Timeout detection is enabled (TIMOUTEN=1 or TEXTEN=1) and the frame before the wakeup frame is abnormally finished due to a I^2C Timeout detection (TIMOUT=1).
- The slave arbitration is lost during the frame before the wakeup frame (ARLO=1).
- The MCU enters STOP mode while another slave is addressed, after the address phase and before the STOP condition (BUSY=1).
- The MCU is in STOP mode and another slave is addressed before the I^2C is addressed.

Note: The last three conditions can occur only in a multi-slave network.

In STOP mode, the HSI is switched on by the I^2C when a START condition is detected (SDA falling edge while SCL is high). The HSI is used to receive the address. HSI is switched off after the address reception if received address is not the I^2C slave address. If one of the conditions above is met and if the SCL falling edge following the START condition occurs on the first cycle of the I2CCLK clock (HSI), the address reception is not correctly done and the address match wakeup interrupt is not generated.

Workaround

- None at MCU level. If the wakeup frame is not acknowledged by the I^2C and if the master can program the duration of the START hold time: the master should decrease or increase the START condition hold time for more than one HSI period and resend the wakeup frame.

2.5.6 Wrong data sampling when data set-up time ($t_{SU;DAT}$) is smaller than one I2CCLK period

Description

The I2C bus specification and user manual specifies a minimum data set-up time ($t_{SU;DAT}$) at:

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

The I2C SDA line is not correctly sampled when $t_{SU;DAT}$ 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.

2.5.7 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.

2.5.8 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 the 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 I2C_ISR register).
2. Disable the I2C peripheral.
3. Wait for a minimum of 3 APB cycles.
4. Enable the I2C peripheral again.

2.6 GPIO peripheral limitation

2.6.1 GPIOx locking mechanism is not working properly for GPIOx_OTYPE register

Description

Locking of GPIOx_OTYPER[i] with i = 15 ..8 depends on the setting of GPIOx_LCKR[i-8] and not from the setting of GPIOx_LCKR[i]. GPIOx_LCKR[i-8] locks GPIOx_OTYPER[i] together with GPIOx_OTYPER[i-8]. It is not possible to lock GPIOx_OTYPER[i] with i = 15...8, without locking also GPIOx_OTYPER[i-8].

Workaround

The only way to lock GPIOx_OTYPER[i] with i=15..8 is to lock also GPIOx_OTYPER[i-8].

2.7 Touch sensing peripheral limitation

2.7.1 Inhibited acquisition in case of short transfer phase configuration

Description

The GPIO input buffer is masked outside the transfer window time and then sampled twice before being checked for the acquisition. This check is performed on the last touch sensing clock cycle of the charge transfer phase. When the charge transfer duration is less than three clock cycles, the acquisition is inhibited.

Workaround

Do not use the following TSC control register configurations:

- PGPSC[2:0] bits set to 000 and CTPL[3:0] bits set to 0000 or 0001 in the TSC_CR register
- PGPSC[2:0] bits set to 001 and CTPL[3:0] bits are set to 0000 in the TSC_CR register

2.8 RTC limitation

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 SSR again after reading SSR/TR/DR to confirm that SSR is still the same, otherwise read the values again.

2.9 BxCAN peripheral limitations

2.9.1 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.

2.10 Comparator peripheral limitation

2.10.1 V_{REFINT} scaler startup time from power down parameter degradation

Description

The V_{REFINT} scaler is an embedded voltage follower providing the V_{REFINT} or its fractions (1/2, 1/4 or 3/4) to the comparator input.

The maximum V_{REFINT} scaler startup time, t_{S_SC}(max), is not as expected for the first activation of the V_{REFINT} scaler after powering on the device and it can be up to 1s (instead of 0.2ms) in worse case conditions. This maximum value depends mainly on the voltage and temperature, see the device datasheet for more details.

If the V_{REFINT} scaler is used, the comparator analog supply voltage, V_{DDA}, has to be min 2V.

Workaround

None.

3 Revision history

Table 5. Document revision history

Date	Revision	Changes
17-Feb-2014	1	Initial release.
02-Mar-2015	2	Added: - *Section 1.2: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used*. - *Section 2.1.3: Full JTAG configuration without NJTRST pin cannot be used*. - *Section 2.5.6: Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period*. - *Section 2.2.3: Start bit detected too soon when sampling for NACK signal from the smartcard*. - *Section 2.2.4: Break request can prevent the Transmission Complete flag (TC) from being set*. - *Section 2.2.5: nRTS is active while RE or UE = 0*. - *Section 2.1.2: Delay after an RCC peripheral clock enabling*. - *Section 2.10: Comparator peripheral limitation*.
06-Jun-2016	3	Updated *Section 2.5.4: Wrong behavior related with MCU Stop mode when the wakeup from Stop mode by I2C peripheral is disabled*. Added: – *Section 2.2.6: Receiver timeout counter starting in case of a 2 stop bit configuration*. – *Section 2.3.2: SDADC crosstalk into another SDADCs*. – *Section 2.5.7: Spurious bus error detection in master mode*. – *Section 2.4.4: BSY bit may stay high at the end of a data transfer in Slave mode*. – *Section 2.4.5: Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback*. Removed CRC limitation.

Table 5. Document revision history (continued)

Date	Revision	Changes
08-Dec-2016	4	I^2C limitation: – Added *Section 2.5.8: 10-bit Master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave*. Touch sensing limitation: – Added *Section 2.7.1: Inhibited acquisition in case of short transfer phase configuration* RTC limitation: – Added *Section 2.8.1: RTC calendar registers are not locked properly*. BxCAN limitation: – Added *Section 2.9.1: BxCAN time triggered mode not supported* . SPI/I2S limitations: – Updated *Section 2.4.4: BSY bit may stay high at the end of a data transfer in Slave mode*. – Updated *Section 2.4.2: SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI finishes its DMA transaction*. – Updated *Section 2.4.5: Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback*. – Added *Section 2.4.6: Wrong CRC transmitted in Master mode with delayed SCK feedback*.