STM32F302xB/C Errata sheet

STM32F302xB/C Rev Z and Y device limitations

Silicon identification

This errata sheet applies to revisions Z and Y of the STMicroelectronics STM32F302xB/C products. This family features an ARM® 32-bit Cortex®-M4 with FPU, 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 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
STM32F302xB/C	“Z” and “Y”

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see the STM32F3xx 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.

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

Table 2. Device summary

Reference	Part number
STM32F302xB/C	STM32F302CB, STM32F302RB, STM32F302VB, STM32F302CC, STM32F302RC, STM32F302VC

Contents

List of tables

1 ARM® 32-bit Cortex®-M4 with FPU limitations

An errata notice of the STM32F3xx 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 with FPU. *Table 3* summarizes these limitations and their implications on the behavior of STM32F30xxx devices.

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

ARM ID	ARM category	ARM summary of errata	Impact on STM32F30xxx
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 with FPU 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 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.

Workaround

Two workarounds are applicable:

- Disable 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 STM32F302xB/C silicon limitations

Table 4 gives quick references to all documented limitations.

Legend for *Table 4*: 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	Revision Y
Section 2.1: System limitations	*Section 2.1.1: SYSCFG_CFGR2, comparators and operational amplifiers control registers reset by APB2 reset*	A	A
	Section 2.1.2: Data Read when the CPU accesses successively SRAM address "A" and SRAM address "A + offset of 16 KBytes (0x4000)"	A	-
	Section 2.1.3: Wakeup sequence from Standby mode when using more than one wakeup source	A	A
	Section 2.1.4: Full JTAG configuration without NJTRST pin cannot be used	A	A
Section 2.2: ADC limitations	*Section 2.2.1: DMA Overrun in dual interleaved mode with single DMA channel*	A	A
	Section 2.2.2: Sampling time shortened in JAUTO autodelayed mode	A	A
	Section 2.2.3: Injected queue of context is not available in case of JQM = 0	N	N
	Section 2.2.4: Load multiple not supported by ADC interface	A	A
	Section 2.2.5: ADEN bit cannot be set immediately after the ADC calibration is done	A	A
	Section 2.2.6: Overrun flag might not be set when the converted data have not been read before new data are written	A	A
	Section 2.2.7: ADC differential mode common mode input range	N	N
Section 2.3: SPI peripheral limitations	*Section 2.3.1: Packing mode limitation at reception*	N	N
	Section 2.3.2: SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI is under DMA transaction near the end of transfer or end of transfer '-1'	A	A
	Section 2.3.3: BSY bit may stay high at the end of a SPI data transfer in slave mode	A	A
	Section 2.3.4: Last data bit or CRC calculation may be corrupted for the data received in SPI/I2S master mode depending on the feedback communication clock timing with respect to the APB clock	A	A

Table 4. Summary of silicon limitations (continued)

Links to silicon limitations		Revision Z	Revision Y
Section 2.4: I2C peripheral limitations	*Section 2.4.1: 10-bit slave mode: wrong direction bit value after Read header reception*	A	A
	Section 2.4.2: 10-bit combined with 7-bit slave mode: ADDCODE may indicate wrong slave address detection	N	N
	Section 2.4.3: Wakeup frames may not wakeup the MCU mode when STOP mode entry follows I2C enabling	A	A
	Section 2.4.4: Wrong behavior related with MCU Stop mode when wakeup from Stop mode by I2C peripheral is disabled	A	A
	Section 2.4.5: Wakeup frame may not wakeup from STOP if tHD(STA) is close to tsu(HSI) in Fast-mode and Fast-mode Plus.	P	P
	Section 2.4.6: Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period	A	A
	Section 2.4.7: Spurious bus error detection in master mode	A	A
Section 2.5: USART limitations	*Section 2.5.1: Communication parameters reprogramming after ATR in Smartcard mode when SCLK is used to clock the card*	A	A
	Section 2.5.2: Last byte written in TDR might not be transmitted if TE is cleared just after writing in TDR.	A	A
	Section 2.5.3: Start bit detected too soon when sampling for NACK signal from the smartcard	N	N
	Section 2.5.4: Break request can prevent the Transmission Complete flag (TC) from being set	A	A
	Section 2.5.5: nRTS is active while RE or UE = 0	A	A
	Section 2.5.6: Receiver timeout counter starting in case of a 2 stop bit configuration	A	A
Section 2.6: I2S peripheral limitations	*Section 2.6.1: In I2S slave mode, WS level must be set by the external master when enabling the I2S*	A	A
Section 2.7: TIM peripheral limitations	*Section 2.7.1: Spurious break generation during TIM1/TIM8 BRK2 initialization*	A	A
Section 2.8: GPIO peripheral limitation	*Section 2.8.1: GPIOx locking mechanism is not working properly for GPIOx_OTYPE register*	A	A
Section 2.9: Comparator peripheral limitation	*Section 2.9.1: VREFINT scaler startup time from power down parameter degradation*	N	N

2.1 System limitations

2.1.1 SYSCFG_CFGR2, comparators and operational amplifiers control registers reset by APB2 reset

Description

The SYSCFG_CFGR2, COMPx_CSR (x = 1..7) and OPAMPx_CSR (x = 1..4) registers should be reset only by system reset, for functional safety purposes.

These registers (except SRAM_PEF flag in SYSCFG_CFGR2) can be reset also by APB2 reset using the SYSCFGRST bit in the RCC_APB2ENR register. Consequently, a spurious access into the SYSCFGRST bit of the RCC_APB2ENR register can cause the lock bits to be cleared, together with the configuration of the comparator and operational amplifiers.

Workaround

In order to protect the SYSCFG_CFGR2, COMPx_CSR (x = 1..7) and OPAMPx_CSR (x = 1..4) registers against unwanted write operation, a workaround is to protect the RCC using MPU, so that any spurious access will be discarded and will trigger a memory management fault exception. This RCC protection prevents spurious software accesses. If higher safety level is required, spurious DMA accesses can be prevented by assigning another MPU region to the DMA controller.

2.1.2 Data Read when the CPU accesses successively SRAM address “A” and SRAM address “A + offset of 16 KBytes (0x4000)”

Description

If the CPU writes to an address A in the SRAM memory and immediately (the cycle after) reads an address B in the SRAM memory, while B = A+0x4000, the read operation will return the content at address A instead of the content of address B.

Workaround

- Revision Z devices:
  The likelihood of such condition to occur is rare and will happen with a code having more than 16 Kbytes of RAM usage. In case of hand-written assembly code, the work-around is to insert a NOP between consecutive write and read from addresses A and A+0x4000 respectively. With a C compiler, it is advised to limit the SRAM usage to 16 Kb instead of 40 Kb. The 8 Kb of CCM memory can also be used for data to have a 24 Kb total RAM space available.
- Revision Y devices: fixed.

2.1.3 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 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 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

2.1.4 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 ADC limitations

2.2.1 DMA Overrun in dual interleaved mode with single DMA channel

Description

DMA overrun conditions can be encountered when two ADCs are working in dual interleaved mode with a single DMA channel for both (MDMA[1:0]bits equal to 0b10 or 0b11). This limitation applies in Single, Continuous and Discontinuous mode.

Workaround

The MDMA [1:0] bits must be kept cleared and each ADC must have its own DMA channel enabled (dual DMA configuration).

2.2.2 Sampling time shortened in JAUTO autodelayed mode

Description

When the ADC is configured in JAUTO single conversion mode (CONT=0), with autodelayed mode enabled (AUTDLY = 1), if the last regular conversion is read and a new regular trigger arrives before the JEOS bit is cleared, the first regular conversion sampling time is shortened by 1 cycle.

This does not apply for configuration where SMP = 000 (1.5 cycle sampling time), or if the interval between triggers is always above the auto-injected sequence conversion period.

Workaround

The sampling time can be increased by 1 clock cycle if the situation is foreseen.

2.2.3 Injected queue of context is not available in case of JQM = 0

Description

The queue mechanism is not functional when JQM = 0. The effective queue length is equal to 1 stage: a new context written before the previous context's consumption will lead to a queue overflow and will be ignored.

Consequently, the ADC must be stopped before programming the JSQR register.

Workaround

None.

2.2.4 Load multiple not supported by ADC interface

Description

The ADC interface does not support LDM, STM, LDRD and STRD instructions for successive multiple-data read and write accesses to a contiguous address block.

Workaround

The workaround consists in preventing compilers from generating LDM, STM, LDRD and STRD instructions. In general, this can be achieved through organizing the source code such as to avoid consecutive read and write accesses to neighboring addresses in lower-to-higher order. In case where consecutive read and write accesses to neighboring addresses cannot be avoided, order the source code such as to access higher address first.

2.2.5 ADEN bit cannot be set immediately after the ADC calibration is done

Description

At the end of the ADC calibration, there is an internal reset of ADEN bit 4 ADC clock cycle after the ADCAL bit cleared by hardware.

Due to that, if ADEN bit is set within those four ADC clock cycles, it will be reset by the calibration logic and the ADC will stay disabled.

Workarounds

- Continue to set the ADEN bit, until ADRDY bit become '1'.
- After ADCAL is cleared, wait for a minimum of four ADC clock cycles before setting the ADEN bit.

2.2.6 Overrun flag might not be set when the converted data have not been read before new data are written

Description

When the converted data are read from the ADC_DR register during the same APB cycle used to write data from a new conversion, the previously written data or the new data are lost but the overrun flag (OVR) might not set to ‘1’.

Workaround

Read the converted data before the data from a new conversion are available, to avoid overrun errors.

2.2.7 ADC differential mode common mode input range

Description

When the ADC is used in differential mode, the common mode input range is $(V_{SSA} + V_{REF+})/2$ +/- 10%.

Workaround

None.

2.3 SPI peripheral limitations

2.3.1 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 bit. 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.

2.3.2 SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI is under DMA transaction near the end of transfer or end of transfer '-1'

Description

SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI is under DMA transaction near the end of transfer or end of transfer '*-1'.*

In the following conditions:

- SPI is slave or master,
- Full duplex or simplex mode is used,
- CRC feature is enabled,
- SPI is configured to manage data transfers by software (interrupt or polling),
- a peripheral, mapped on the same DMA channel as the SPI, is doing DMA transfers,

the CRC may be frozen before the CRCNEXT bit is written, resulting in a CRC error.

Workaround

If the application allows it, use the DMA for SPI transfers.

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

Description

In slave mode, BSY bit is not reliable to handle the end of data frame transaction due to some bad synchronization between the CPU clock and external SCK clock provided by master. Sporadically, the BSY bit is not cleared at the end of a data frame transfer. As a consequence, it is not recommended to rely on BSY bit before entering low-power mode or modifying the SPI configuration (e.g. direction of the bidirectional mode).

Workaround

- When the SPI interface is in receive mode, the end of a transaction with the master can be detected by the corresponding RXNE event when this flag is set after the last bit of that transaction is sampled and the received data are stored.
- When the following sequence is used, the synchronization issue does not occur. The BSY bit works correctly and can be used to recognize the end of any transmission transaction (including when RXNE is not raised in bidirectional mode):
  a) Write the last data into data register.
  b) Poll TXE flag till it becomes high to make sure the data transfer has started.
  c) Disable the SPI interface by clearing SPE bit while the last data transfer is on going.
  d) Poll the BSY bit till it becomes low.

Note: The second workaround can be used only when the CPU is fast enough to disable the SPI interface after a TXE event is detected while the data frame transfer is ongoing. It cannot be implemented when the ratio between CPU and SPI clock is low and the data frame is particularly short. At this specific case, the timeout can be measured from the TXE event instead by calculating a fixed number of CPU clock cycles corresponding to the time necessary to complete the data frame transaction.

2.3.4 Last data bit or CRC calculation may be corrupted for the data received in SPI/I2S master mode depending on the feedback communication clock timing with respect to the APB clock

Description

When the SPI or I2S interface is configured in master mode, the last transacted bit may be corrupted if the delay of internal feedback derived from SCK pin is comparable with the APB clock period. The last bit value is strobed too late into the shift register while its content has been already either copied into the data register or compared with the pattern calculated internally at RX CRC register. In case of data corruption, the bit position in the data register contains the value of the last bit received during the previous data transfer.

The main factors which can contribute negatively to the delay are: decreased V_{DD} level, extreme temperature, high SPI bus capacity load and low SCK I/O output speed. SPI communication speed has no impact.

Workarounds

Several workarounds are possible:

- Decrease the APB clock frequency
- Set the I/O pad configuration to achieve a faster I/O output speed at SCK pin
- Ignore the last transacted bit value.

2.4 I²C peripheral limitations

2.4.1 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 (i.e. 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.

2.4.2 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 I²C 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, ie. 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 & OA1[9:8].

Workaround

None. If several slave addresses 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.

2.4.3 Wakeup frames may not wakeup the MCU mode when STOP mode entry follows I²C enabling

Description

If the I²C is enabled (PE = 1) and wakeup from STOP enabled in I²C (WUPEN=1) while a transfer occurs on the I²C bus and STOP mode is entered during the same transfer while SCL=0, the I²C is not able to detect the following START condition. This means that if the I²C is addressed, it will not wake up the MCU and this address is not acknowledged.

Workaround

After enabling the I²C (PE is set to 1), wait for a temporization before entering STOP mode, to ensure that the eventual on-going frame is finished.

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

Description

When 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 I²C bus, the following wrong operation may occur:

1. 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.

2.4.5 Wakeup frame may not wakeup from STOP if $t_{HD(STA)}$ is close to $t_{SU(HSI)}$ in Fast-mode and Fast-mode Plus.

Description

Under specific conditions and if the START condition hold time $t_{HD(STA)}$ duration is very close to the HSI start-up time duration $t_{su(HSI)}$, the I^2C is not able to detect the address match and to wake up the MCU from STOP. The $t_{su(HSI)}$ is between 1 µs and 2 µs (refer to product datasheet), therefore this issue cannot occur in Standard mode. 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). According to standards, the slave arbitration is not applicable in I^2C and used only in SMBus, for which the transfer is done in Standard mode. Therefore when the standards are respected this condition does not lead to the limitation.
- 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 powered 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 and it is powered off after the address reception is case it 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. To ensure the correct behavior in a multi-slave network, the master should use a START condition hold time lower than 1 µs or greater than 2 µs.

If the wakeup frame is not acknowledged by the I^2C:

- 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.
- If the master can change the I^2C transfer mode: the master should switch to Standard mode and resend the wakeup frame.

2.4.6 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.

2.4.7 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 USART limitations

2.5.1 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 doesn’t 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).

2.5.2 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.

Workaround

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

2.5.3 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 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.

2.5.4 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.

2.5.5 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.

2.5.6 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 with subtracting 1 bit duration.

2.6 I2S peripheral limitations

2.6.1 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.

2.7 TIM peripheral limitations

2.7.1 Spurious break generation during TIM1/TIM8 BRK2 initialization

Description

When the BRK2 polarity is configured (using BK2P bit in TIMx_BDTR register) and the BRK2 is enabled (setting BK2E bit in TIMx_BDTR register) in the same write access, the B2IF flag in the TIMx_SR register may be wrongly set resulting in a parasitic break detection.

Workarounds

If the LOCK feature is not used, the BRK2 must be first configured (setting or resetting BK2P) and then enabled (setting BK2E) in two separate write accesses.

If the LOCK feature is used, and since BK2E, BK2P and LOCK are in the same register, they must be all configured in the same write. Consequently, the BRK2 cannot be configured/enabled using two consecutive write accesses.

- If the BRK2 input is active low (BK2P = 0), no workaround is needed.
- If the BRK2 input is active high (BK2P = 1), the software must anticipate that a parasitic break event may happen upon BRK2 enable, so:
  - Set BK2P and BK2E in TIMx_BDTR register, in the same write access, with keeping MOE reset.
  - Wait for the duration during which the flag BK2IF may be set (4 Timer clock cycles is the minimum duration for B2IF flag to be set).
  - Check the B2IF flag status by reading the TIMx_SR register. If found set, clear it by writing it to '0'.
  - The outputs can be enabled by setting MOE bit and the break interrupt can be enabled.

2.8 GPIO peripheral limitation

2.8.1 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].

2.9 Comparator peripheral limitation

2.9.1 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 in worse case conditions. This maximum value depends mainly on the voltage and temperature, see the device datasheet for more details.

Workaround

None.

3 Revision history

Table 5. Document revision history

Date	Revision	Changes
25-Mar-2014	1	Initial release
04-Dec-2014	2	Added: - *Section 1.2: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used*. - *Section 2.1.3: Wakeup sequence from Standby mode when using more than one wakeup source*. - *Section 2.2.4: Load multiple not supported by ADC interface*. - *Section 2.2.5: ADEN bit cannot be set immediately after the ADC calibration is done*. - *Section 2.4.6: Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period*. - *Section 2.5.2: Last byte written in TDR might not be transmitted if TE is cleared just after writing in TDR.* Removed appendix A: LQFP100, LQFP64 and LQFP48 device markings.
19-Feb-2015	3	Added: - *Section 2.5.3: Start bit detected too soon when sampling for NACK signal from the smartcard*. - *Section 2.5.4: Break request can prevent the Transmission Complete flag (TC) from being set*. - *Section 2.5.5: nRTS is active while RE or UE = 0*. - *Section 2.9: Comparator peripheral limitation*.

Table 5. Document revision history (continued)

Date	Revision	Changes
19-Nov-2015	4	Updated – *Section 2.4.4: Wrong behavior related with MCU Stop mode when wakeup from Stop mode by I2C peripheral is disabled*. – *Section 2.2.4: Load multiple not supported by ADC interface*. – *Section 2.7.1: Spurious break generation during TIM1/TIM8 BRK2 initialization*. Added the following limitations: – *Section 2.1.4: Full JTAG configuration without NJTRST pin cannot be used*, – *Section 2.5.6: Receiver timeout counter starting in case of a 2 stop bit configuration*, – *Section 2.4.7: Spurious bus error detection in master mode*. – *Section 2.3.3: BSY bit may stay high at the end of a SPI data transfer in slave mode*. – *Section 2.3.4: Last data bit or CRC calculation may be corrupted for the data received in SPI/I2S master mode depending on the feedback communication clock timing with respect to the APB clock*. Updated *Section 2.1: System limitations* removing CCM RAM limitation.
02-May-2016	5	Added: – *Section 2.2.6: Overrun flag might not be set when the converted data have not been read before new data are written*. – *Section 2.2.7: ADC differential mode common mode input range*.