STM32F303xD STM32F303xE Errata sheet

STM32F303xD STM32F303xE Rev Y device limitations

Silicon identification

This errata sheet applies to revision Y of STMicroelectronics STM32F303xD/xE products. These products feature an ARM® 32-bit Cortex®-M4F 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 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.

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

Table 1. Device identification(1)

Sales type	Revision code(2) marked on device
STM32F303xD/xE	"Y"

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see the STM32F303xx and STM32F3x8xx reference manual for details on how to find the revision code).
2. Refer to STM32F303xD/xE datasheet for the device marking.

Table 2. Device summary

Reference	Part number
STM32F303xD	STM32F303RD, STM32F303VD, STM32F303ZD
STM32F303xE	STM32F303RE, STM32F303VE, STM32F303ZE

Contents

List of tables

1 ARM® 32-bit Cortex®-M4F limitations

An errata notice of the 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®-M4F core. *Table 3* summarizes these limitations and their implications on the behavior of the STM32F30xxx devices.

Table 3. Cortex®-M4F core 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®-M4F 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

2 STM32F303xD/xE 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 Y
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: Minimum CPU frequency and Prefetch buffer state	N
Section 2.2: ADC limitations	*Section 2.2.1: DMA Overrun in dual interleaved mode with single DMA channel*	A
	Section 2.2.2: Sampling time shortened in JAUTO autodelayed mode	A
	Section 2.2.3: Injected queue of context is not available in case of JQM = 0	N
	Section 2.2.4: Load multiple not supported by ADC interface	A
Section 2.3: SPI peripheral limitations	*Section 2.3.1: 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'*	P
Section 2.4: I2C peripheral limitations	*Section 2.4.1: 10-bit slave mode: wrong direction bit value after Read header reception*	A
	Section 2.4.2: 10-bit combined with 7-bit slave mode: ADDCODE may indicate wrong slave address detection	N
	Section 2.4.3: Wakeup frames may not wakeup the MCU mode when STOP mode entry follows I2C enabling	A
	Section 2.4.4: Wrong behaviors in Stop mode when wakeup from Stop mode is disabled in I2C	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
Section 2.5: I2S limitations	*Section 2.5.1: In I2S slave mode, WS level must be set by the external master when enabling the I2S*	A
Section 2.6: Timer limitations	*Section 2.6.1: TIM20 Brk2 acts to COMPx_OUT even if COMPx_OUT is configured to be connected internally to TIM1 and TIM8 Brk2 only*	A

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 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.

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 Minimum CPU frequency and Prefetch buffer state

Description

The minimum frequency that can be used as CPU clock is 100 kHz.

The prefetch buffer must be kept always ON whatever the CPU clock.

Workaround

None.

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 only supports non-sequential read accesses.

Read accesses on AHB3 port (ADC interface) using Load multiple instruction are not supported.

The following sequence (read of JDR1, JDR2 and JDR3 injected data registers) will only cause the R1 register to be loaded with JDR1 value, and registers R2 and R3 will be zeroed.

```
LDR     R0 = 0x50000080
LDMIA   R0, {R1, R2, R3}
```

Workaround

Load multiple instruction LDMxx must be replaced by multiple single load (LD) instructions.

2.3 SPI peripheral limitations

2.3.1 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.

2.4 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 I^{2}C 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:

- I^{2}C has to be configured in 10-bit addressing mode (OA1MODE is set in the I2C_OAR1 register).
- The high LSBs of the I^{2}C 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 I^{2}C 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 I^{2}C 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 I^{2}C 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 & 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 I^2C is enabled (PE = 1) and wakeup from STOP 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 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.

2.4.4 Wrong behaviors in Stop mode when wakeup from Stop mode is disabled in I2C

Description

When wakeup from Stop mode is disabled in I2C (WUPEN = 0) and the MCU enters Stop mode while a transfer is on going on the bus, some wrong behaviors may happen:

1. BUSY flag can 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.
2. If clock stretching is enabled (NOSTRETCH = 0), the I2C clock SCL may be stretched low by the I2C as long as the MCU is in Stop mode. This limitation may occur when the Stop mode is entered during the address phase of a transfer on the I2C bus while SCL = 0. Therefore the transfer may be stalled as long as the MCU is in Stop mode. The probability of the occurrence depends also on the timings configuration, the peripheral clock frequency and the I2C bus frequency.

These behaviors can occur in Slave mode and in Master mode in a multi-master topology.

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.

2.5 I2S limitations

2.5.1 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.

2.6 Timer limitations

2.6.1 TIM20 Brk2 acts to COMPx_OUT even if COMPx_OUT is configured to be connected internally to TIM1 and TIM8 Brk2 only

Description

When TIM20 Brk2 is enabled and the COMPx_OUT (x = 1..7) is configured to be connected internally to TIM1 and TIM8 Brk2 input, that is COMPxOUTSEL = 0101 in COMPx_CSR register, TIM20 Brk2 input reacts to COMPx_OUT. This means that both configurations "COMPxOUTSEL = 0101" and "COMPxOUTSEL = 1110" are equivalent.

Workaround

There is no workaround.

3 Revision history

Table 5. Document revision history

Date	Revision	Changes
20-Jan-2015	1	Initial release.