STM32F334x4/x6/x8
Errata sheet

STM32F334x4/x6/x8 Rev Z device limitations

Silicon identification

This errata sheet applies to revision Z of the STMicroelectronics STM32F334x4/x6/x8 products. These families feature an ARM® 32-bit Cortex®-M4 FPU 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)(2)

Order code	Revision code marked on device
STM32F334x4/x6/x8	“Z”

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see the STM32F334x4/x6/x8 reference manual for details on how to find the revision code).
2. Refer to datasheet for the device marking.

Table 2. Device summary

Reference	Part number
STM32F334x4/x6/x8	STM32F334K4, STM32F334C4, STM32F334R4 STM32F334K6, STM32F334C6, STM32F334R6 STM32F334K8, STM32F334C8, STM32F334R8

Contents

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

An errata notice of the STM32F334x4/x6/x8 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 FPU core. *Table 3* summarizes these limitations and their implications on the behavior of STM32F3xxxx devices.

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

ARM ID	ARM category	ARM summary of errata	Impact on STM32F3xxxx
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 FPU core 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 STM32F334x4/x6/x8 silicon limitations

Table 4 gives quick references to all documented limitations.

Legend for *Table 4* is as follows:

A = workaround available;

N = no workaround available;

P = partial workaround available,

'-' and grayed = fixed.

Table 4. Summary of silicon limitations

Links to silicon limitations		Revision Z
Section 2.1: ADC limitations	*Section 2.1.1: DMA Overrun in dual interleaved mode with single DMA channel*	A
	Section 2.1.2: Sampling time shortened in JAUTO autodelayed mode	A
	Section 2.1.3: Injected queue of context is not available in case of JQM = 0	N
Section 2.2: SPI peripheral limitations	*Section 2.2.1: Packing mode limitation at reception*	N
Section 2.3: I2C peripheral limitations	*Section 2.3.1: 10-bit slave mode: wrong direction bit value after Read header reception*	A
	Section 2.3.2: 10-bit combined with 7-bit slave mode: ADDCODE may indicate wrong slave address detection	N
	Section 2.3.3: Wakeup frames may not wakeup the MCU mode when STOP mode entry follows I2C enabling	A
	Section 2.3.4: Wrong behaviors in Stop mode when wakeup from Stop mode is disabled in I2C	A
	Section 2.3.5: Wakeup frame may not wakeup from STOP if tHD(STA) is close to tsu(HSI) in Fast-mode and Fast-mode Plus.	P
	Section 2.3.6: Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period	P
Section 2.4: USART peripheral limitation	*Section 2.4.1: When PCLK is selected as clock source for USART1, PCLK1 is used instead of PCLK2*	A
Section 2.5: GPIO peripheral limitation	*Section 2.5.1: GPIOx locking mechanism is not working properly for GPIOx_OTYPE register*	A

2.1 ADC limitations

2.1.1 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).

2.1.2 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.

2.1.3 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 SPI peripheral limitations

2.2.1 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 I^2C peripheral limitations

2.3.1 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^2C 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^2C has to be configured in 10-bit addressing mode (OA1MODE is set in the I2C_OAR1 register).
- The high LSBs of the I^2C 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^2C 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^2C slave address, the DIR bit must not be used in the FW.

2.3.2 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, 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.

2.3.3 Wakeup frames may not wakeup the MCU mode when STOP mode entry follows I^2C 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.

2.3.4 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.

2.3.5 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.

2.3.6 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 USART peripheral limitation

2.4.1 When PCLK is selected as clock source for USART1, PCLK1 is used instead of PCLK2

Description

USART1 is mapped on the fast APB (APB2) and its clock can be selected among four different sources using the USART1SW [1:0] bits in the RCC_CFGR3 register.

The default configuration selects PCLK1 (APB1 clock) as USART1 clock source instead of PCLK2 (APB2 clock).

Workaround

There is no workaround. To reach 9 Mbaud, System Clock (SYSCLK) should be selected as USART1 clock source.

2.5 GPIO peripheral limitation

2.5.1 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].

3 Revision history

Table 5. Document revision history

Date	Revision	Changes
13-Jun-2014	1	Initial release