life.augmented

STM32F398VE Errata sheet

STM32F398VE Rev Y device limitations

Silicon identification

This errata sheet applies to revision Y of STMicroelectronics STM32F398VE products. These products feature an Arm® 32-bit Cortex®-M4 core 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)

Sales type	Revision code(2) marked on device
STM32F398VE	"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 STM32F398VE datasheet for the device marking.

Contents

List of tables

1 Arm® 32-bit Cortex®-M4 core with FPU limitations

For information on the Arm®[a] Cortex®-M4 core, please refer to the Cortex®-M4 technical reference manual, available from the www.arm.com website.

All the described limitations are minor and related to the revision r0p1-v1 of the Cortex®-M4 core with FPU. *Table 2* summarizes these limitations and their implications on the behavior of the STM32F398VE devices.

Table 2. Cortex®-M4 core with FPU limitations and impact on microcontroller behavior

Arm ID	Arm category	Arm summary of errata	Impact on STM32F398VE
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 core 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.

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Example:

Replace LDR SP,[R0] by

LDR R2,[R0]

MOV SP,R2

2 STM32F398VE silicon limitations

Table 3 gives quick references to all documented limitations.

Legend for *Table 3*: A = workaround available; N = no workaround available; P = partial workaround available, '-' and grayed = fixed.

Table 3. 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 2.1.3: Full JTAG configuration without NJTRST pin cannot be used	A
	Section 2.1.4: No reset of CCM RAM write protection register SYSCFG_RCR by system reset	N
Section 2.2: ADC peripheral limitations	*Section 2.2.1: DMA Overrun in dual interleaved mode with single DMA channel*	A
	Section 2.2.3: Sampling time shortened in JAUTO autodelayed mode	A
	Section 2.2.4: Injected queue of context is not available in case of JQM = 0	N
	Section 2.2.5: Multiple Loads not supported by ADC interface	A
	Section 2.2.6: ADC differential mode, common mode input range	N
	Section 2.2.2: Overrun flag may not be set if converted data are not read before writing new data	A
	Section 2.2.5: Multiple Loads not supported by ADC interface	A
Section 2.3: Comparator limitations	*Section 2.3: Comparator limitations*	N
Section 2.4: OPAMP limitations	*Section 2.4.1: OPAMP Timer controlled multiplexer mode not working when OPAMP is used in PGA or follower mode*	N
Section 2.5: SPI peripheral limitations	*Section 2.5.1: 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 2.5.2: BSY bit may stay high at the end of a SPI data transfer in slave mode	A

Table 3. Summary of silicon limitations (continued)

Links to silicon limitations		Revision Y
Section 2.6: I²C peripheral limitations	*Section 2.6.1: 10-bit slave mode: wrong direction bit value after Read header reception*	A
	Section 2.6.2: 10-bit combined with 7-bit slave mode: ADDCODE may indicate wrong slave address detection	N
	Section 2.6.3: Wakeup frames may not wakeup the MCU mode when STOP mode entry follows I²C enabling	A
	Section 2.6.4: Wrong behavior related with MCU Stop mode when wakeup from Stop mode by I²C peripheral is disabled	A
	Section 2.6.5: Wakeup frame may not wakeup from STOP if $t_{HD(STA)}$ is close to $t_{su(HSI)}$ in Fast-mode and Fast-mode Plus	P
	Section 2.6.6: Spurious Bus Error detection in master mode	A
Section 2.7: I²S limitations	*Section 2.7.1: In I²S slave mode, WS level must be set by the external master when enabling the I²S*	A
Section 2.8: Timer limitations	*Section 2.8.1: TIM20 Brk2 acts to COMPx_OUT even if COMPx_OUT is configured to be connected internally to TIM1 and TIM8 Brk2 only*	A
Section 2.9: USART peripheral limitations	*Section 2.9.1: Start bit detected too soon when sampling for NACK signal from the Smartcard*	N
	Section 2.9.2: A break request can prevent the Transmission Complete flag (TC) from being set	A
	Section 2.9.3: nRTS is active while RE = 0 or UE = 0	A
	Section 2.9.4: Receiver timeout counter starting in case of 2 stop bits configuration	A
Section 2.10: FMC limitations	*Section 2.10.1: Dummy read cycles inserted when reading synchronous memories*	N
	Section 2.10.2: Data corruption during burst read from FMC synchronous memory	A
	Section 2.10.3: FMC bank switching to asynchronous bank for write	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.

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 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.

2.1.3 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.

2.1.4 No reset of CCM RAM write protection register SYSCFG_RCR by system reset

Description

The CCM RAM write protection register SYSCFG_RCR cannot be reset by system reset. It can be reset only by POR reset.

Workaround

None.

If the application needs to write protect the CCM RAM and to remove the CCM RAM write protection without applying a POR reset, other solutions can be adopted such as:

- Protecting the CCM RAM against unwanted write operation using the MPU, or
- Simply using the parity check feature allowing the detection of CCM RAM content corruption.

2.2 ADC peripheral 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 Overrun flag may not be set if converted data are not read before writing new data

Description

When converted data are read from the ADC_DR register during the very 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) may not be set to ‘1’.

Workaround

To avoid overrun errors read the converted data before data from a new conversion are made available by the ADC.

2.2.3 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.

2.2.4 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.

2.2.5 Multiple Loads 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 or write accesses to neighboring addresses in lower-to-higher order. In case where consecutive read or write accesses to neighboring addresses cannot be avoided, order the source code such as to access higher address first.

2.2.6 ADC differential mode, common mode input range

Description

When the ADC is used in differential mode, the common mode input range is (VSSA + VREF+)/2 +/- 0.18V.

Workaround

None

2.3 Comparator limitations

Description

The comparator does not support the window mode.

Workaround

None.

2.4 OPAMP limitations

2.4.1 OPAMP Timer controlled multiplexer mode not working when OPAMP is used in PGA or follower mode

Description

When the OPAMP is operating in PGA or follower mode, the timer controlled multiplexer feature cannot be used.

Workaround

None. To use the timer controlled multiplexer feature, the OPAMP must be used in standalone mode.

2.5 SPI peripheral limitations

2.5.1 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.

2.5.2 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 ongoing.
  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.*

2.6 I^{2}C peripheral limitations

2.6.1 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.

2.6.2 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, 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.

2.6.3 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 ongoing frame is finished.

2.6.4 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 ongoing 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 I²C-bus environment.
2. If I²C-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 I²C-bus activity during that time. This may occur when the MCU enters Stop mode during the address phase of an I²C-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 I²C-bus environment. Its probability depends on the timing configuration, operating clock frequency of I²C peripheral and the I²C-bus timing.

Workaround

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

2.6.5 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 I²C 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 an I²C 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 I²C 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 I²C 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 I²C 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 I²C 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.

2.6.6 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 I^2C 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 ongoing transfer can be handled normally.

2.7 I^2S limitations

2.7.1 In I^2S slave mode, WS level must be set by the external master when enabling the I^2S

Description

In slave mode the WS signal level is used only to start the communication. If the I^2S (in slave mode) is enabled while the master is already sending the clock and the WS signal level is low (for I^2S 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 I^2S peripheral must be enabled when the external master sets the WS line at:

- High level when the I^2S protocol is selected.
- Low level when the LSB or MSB-justified mode is selected.

2.8 Timer limitations

2.8.1 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.

2.9 USART peripheral limitations

2.9.1 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 reaches 10.68 etu or more, the USART misinterprets this transition as a START bit even if the NACK is correctly detected.

Workaround

None.

2.9.2 A break request can prevent the Transmission Complete flag (TC) from being set

Description

After the end of transmission of 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 the data transfer, the break request should occur after making sure that the TC flag is set.

2.9.3 nRTS is active while RE = 0 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), 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.

2.9.4 Receiver timeout counter starting in case of 2 stop bits configuration

Description

In the case of 2 stop bits 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.

2.10 FMC limitations

2.10.1 Dummy read cycles inserted when reading synchronous memories

Description

When performing a burst read access to a synchronous memory, two dummy read accesses are performed at the end of the burst cycle whatever the type of AHB burst access. However, the extra data values which are read are not used by the FMC and there is no functional failure.

Workaround

None

2.10.2 Data corruption during burst read from FMC synchronous memory

Description

A burst read from static memory can be corrupted if all the following conditions are met:

- One FMC bank is configured in synchronous mode with WAITEN bit enabled while another FMC bank is used with WAITEN bit disabled.
- A read burst transaction is ongoing from static synchronous memory with wait feature enabled.
- The synchronous memory asserts the wait signal during the ongoing burst read.
- The read burst transaction is followed by an access to an FMC banks for which the WAITEN bit is disabled in the FMC_BCRx register.

Workaround

1. Set the WAITEN bit on all FMC static banks even if it is not used by the memory.
2. Set the same WAIT polarity on all static banks.
3. Enable the internal pull-up on PD6 in order to set to ready the FMC_NWAIT input when the synchronous memory is de-selected and the other FMC bank without wait feature is selected.

2.10.3 FMC bank switching to asynchronous bank for write

Description

When switching from one of the FMC banks in read transaction to another asynchronous bank for write, the FMC could hang in the following conditions:

- one FMC bank is enabled with BUSTURN timing> 0,
- a second FMC bank is enabled in asynchronous mode with BUSTURN = 0,
- a read from the first bank followed by a write transaction on the second bank.

Workaround

Use BUSTURN equal zero or different from zero for used banks.

3 Revision history

Table 4. Document revision history

Date	Revision	Changes
27-Jan-2015	1	Initial release.
23-Feb-2015	2	Added the following limitations: – *Section 2.1.3: Full JTAG configuration without NJTRST pin cannot be used* – *Section 2.9.1: Start bit detected too soon when sampling for NACK signal from the Smartcard* – *Section 2.9.2: A break request can prevent the Transmission Complete flag (TC) from being set* – *Section 2.9.3: nRTS is active while RE = 0 or UE = 0* – *Section 2.10.1: Dummy read cycles inserted when reading synchronous memories* – *Section 2.10.2: Data corruption during burst read from FMC synchronous memory* – *Section 2.10.3: FMC bank switching to asynchronous bank for write* Replaced all Cortex® -M4F occurrences with Cortex® -M4 core with FPU.
14-Sep-2015	3	Updated: – *Section 2.6.4: Wrong behavior related with MCU Stop mode when wakeup from Stop mode by I²C peripheral is disabled*. Added the following limitations: – *Section 2.1.4: No reset of CCM RAM write protection register SYSCFG_RCR by system reset*, – *Section 2.5.2: BSY bit may stay high at the end of a SPI data transfer in slave mode*, – *Section 2.6.6: Spurious Bus Error detection in master mode*, – *Section 2.9.4: Receiver timeout counter starting in case of 2 stop bits configuration*.
13-Dec-2016	4	Added the following limitations: – *Section 2.2.2: Overrun flag may not be set if converted data are not read before writing new data*, – *Section 2.2.5: Multiple Loads not supported by ADC interface*, – *Section 2.2.6: ADC differential mode, common mode input range*, – *Section 2.3: Comparator limitations*.
17-Dec-2018	5	Added *Section 2.4: OPAMP limitations*