

Applicability

This document applies to the part numbers of STM32F412xx devices listed in [Table 1](#) and their variants shown in [Table 2](#).

[Section 1](#) gives a summary and [Section 2](#) a description of / workaround for device limitations, with respect to the device datasheet and reference manual [RM0402].

Table 1. Device summary

Reference	Part numbers
STM32F412xx	STM32F412CE, STM32F412RE, STM32F412VE, STM32F412ZE, STM32F412CG, STM32F412RG, STM32F412VG, STM32F412ZG

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking ⁽¹⁾	REV_ID ⁽²⁾
STM32F412xx	Z	0x1001
STM32F412xx	C	0x3000

1. Refer to the device data sheet for how to identify this code on different types of package.
2. REV_ID[15:0] bit field of DBGMCU_IDCODE register. Refer to the reference manual.

Contents

1	Arm® 32-bit Cortex®-M4 with FPU limitations	5
1.1	Cortex-M4 interrupted loads to stack pointer can cause erroneous behavior	5
1.2	VDIV or VSQRT instructions might not complete correctly when very short ISRs are used	6
2	STM32F412xx silicon limitations	7
2.1	System limitations	9
2.1.1	Debugging Sleep/Stop mode with WFE/WFI entry	9
2.1.2	Wakeup sequence from Standby mode when using more than one wakeup source	9
2.1.3	Full JTAG configuration without NJTRST pin cannot be used	10
2.1.4	MPU attribute to RTC and IWDG registers could be managed incorrectly	10
2.1.5	Delay after an RCC peripheral clock enabling	10
2.1.6	Internal noise impacting the ADC accuracy	10
2.1.7	Flash sector erase issue for sectors 5 to 11	11
2.1.8	In some specific cases, DMA2 data corruption occurs when managing AHB and APB2 peripherals in a concurrent way	11
2.2	IWDG peripheral limitation	12
2.2.1	RVU and PVU flags are not reset in STOP mode	12
2.3	I2C peripheral limitations	12
2.3.1	SMBus standard not fully supported	12
2.3.2	Start cannot be generated after a misplaced Stop	12
2.3.3	Mismatch on the “Setup time for a repeated Start condition” timing parameter	13
2.3.4	Data valid time ($t_{VD;DAT}$) violated without the OVR flag being set	13
2.3.5	Both SDA and SCL maximum rise time (t_r) violated when VDD_I2C bus higher than $((VDD+0.3) / 0.7)$ V	14
2.3.6	Last received byte can be lost when using Reload mode with NBYTES > 1	14
2.4	FMPI2C peripheral limitation	15
2.4.1	Wrong data sampling when data set-up time ($t_{SU;DAT}$) is smaller than one FMPI2CCLK period	15
2.5	SPI/I2S peripheral limitation	15
2.5.1	Wrong CRC calculation when the polynomial is even.	15

2.5.2	BSY bit may stay high at the end of a data transfer in slave mode	15
2.5.3	Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback	16
2.6	USART peripheral limitations	17
2.6.1	Idle frame is not detected if receiver clock speed is deviated	17
2.6.2	In full duplex mode, the Parity Error (PE) flag can be cleared by writing to the data register	17
2.6.3	Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection	18
2.6.4	Break frame is transmitted regardless of nCTS input line status	18
2.6.5	nRTS signal abnormally driven low after a protocol violation	18
2.6.6	Start bit detected too soon when sampling for NACK signal from the smartcard	19
2.6.7	Break request can prevent the Transmission Complete flag (TC) from being set	19
2.6.8	Guard time is not respected when data are sent on TXE events	19
2.6.9	nRTS is active while RE or UE = 0	19
2.7	bxCAN limitation	20
2.7.1	bxCAN time triggered communication mode not supported	20
2.8	FSMC peripheral limitation	20
2.8.1	Dummy read cycles inserted when reading synchronous memories	20
2.9	SDIO peripheral limitations	20
2.9.1	Wrong CCRCFAIL status after a response without CRC is received	20
2.9.2	No underrun detection with wrong data transmission	21
2.10	ADC peripheral limitations	21
2.10.1	ADC sequencer modification during conversion	21
2.11	QuadSPI limitations	21
2.11.1	First nibble of data is not written after dummy phase	21
2.11.2	Wrong data can be read in memory-mapped after an indirect mode operation	22
3	Revision history	23

List of tables

Table 1. Device summary 1

Table 2. Device variants 1

Table 3. Cortex-M4 core limitations and impact on microcontroller behavior 5

Table 4. Summary of silicon limitations 7

Table 5. Maximum allowable APB frequency at 30 pF load 17

Table 6. Document revision history 23



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

An errata notice of the STM32F412xx core is available from <http://infocenter.arm.com>.

All the described limitations are minor and related to the revision r0p1-v1 of the Cortex-M4 core. [Table 3](#) summarizes these limitations and their implications on the behavior of STM32F412xx devices.

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

Arm ID	Arm category	Arm summary of errata	Impact on STM32F412xx
752770	Cat B	Interrupted loads to SP can cause erroneous behavior	Minor
776924	Cat B	VDIV or VSQRT instructions might not complete correctly when very short ISRs are used	Minor

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

Description

An interrupt occurring during the data-phase of a single word load to the stack pointer (SP/R13) can cause an erroneous behavior of the device. In addition, returning from the interrupt results in the load instruction being executed 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 limitations 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 VSQRT 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 STM32F412xx 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 C
Section 2.1: System limitations	Section 2.1.1: Debugging Sleep/Stop mode with WFE/WFI entry	A	A
	Section 2.1.2: Wakeup sequence from Standby mode when using more than one wakeup source	A	A
	Section 2.1.3: Full JTAG configuration without NJTRST pin cannot be used	A	A
	Section 2.1.4: MPU attribute to RTC and IWDG registers could be managed incorrectly	A	A
	Section 2.1.5: Delay after an RCC peripheral clock enabling	A	A
	Section 2.1.6: Internal noise impacting the ADC accuracy	A	A
	Section 2.1.7: Flash sector erase issue for sectors 5 to 11	A	-
	Section 2.1.8: In some specific cases, DMA2 data corruption occurs when managing AHB and APB2 peripherals in a concurrent way	A	A
Section 2.2: IWDG peripheral limitation	Section 2.2.1: RVU and PVU flags are not reset in STOP mode	A	A
Section 2.3: I2C peripheral limitations	Section 2.3.1: SMBus standard not fully supported	A	A
	Section 2.3.2: Start cannot be generated after a misplaced Stop	A	A
	Section 2.3.3: Mismatch on the "Setup time for a repeated Start condition" timing parameter	A	A
	Section 2.3.4: Data valid time ($t_{VD;DAT}$) violated without the OVR flag being set	A	A
	Section 2.3.5: Both SDA and SCL maximum rise time (t_r) violated when VDD_I2C bus higher than $((VDD+0.3) / 0.7)$ V	A	A
	Section 2.3.6: Last received byte can be lost when using Reload mode with NBYTES > 1	P	P
Section 2.4: FMPI2C peripheral limitation	Section 2.4.1: Wrong data sampling when data set-up time ($t_{SU;DAT}$) is smaller than one FMPI2CCLK period	A	A
Section 2.5: SPI/I2S peripheral limitation	Section 2.5.1: Wrong CRC calculation when the polynomial is even.	A	A
	Section 2.5.2: BSY bit may stay high at the end of a data transfer in slave mode	A	A
	Section 2.5.3: Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback	A	A

Table 4. Summary of silicon limitations (continued)

Links to silicon limitations		Revision Z	Revision C
Section 2.6: USART peripheral limitations	Section 2.6.1: Idle frame is not detected if receiver clock speed is deviated	N	N
	Section 2.6.2: In full duplex mode, the Parity Error (PE) flag can be cleared by writing to the data register	A	A
	Section 2.6.3: Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection	N	N
	Section 2.6.4: Break frame is transmitted regardless of nCTS input line status	N	N
	Section 2.6.5: nRTS signal abnormally driven low after a protocol violation	A	A
	Section 2.6.6: Start bit detected too soon when sampling for NACK signal from the smartcard	A	A
	Section 2.6.7: Break request can prevent the Transmission Complete flag (TC) from being set	A	A
	Section 2.6.8: Guard time is not respected when data are sent on TXE events	A	A
	Section 2.6.9: nRTS is active while RE or UE = 0	A	A
Section 2.7: bxCAN limitation	Section 2.7.1: bxCAN time triggered communication mode not supported	A	A
Section 2.8: FSMC peripheral limitation	Section 2.8.1: Dummy read cycles inserted when reading synchronous memories	N	N
Section 2.9: SDIO peripheral limitations	Section 2.9.1: Wrong CCRCFAIL status after a response without CRC is received	A	A
	Section 2.9.2: No underrun detection with wrong data transmission	A	A
Section 2.10: ADC peripheral limitations	Section 2.10.1: ADC sequencer modification during conversion	A	A
Section 2.11: QuadSPI limitations	Section 2.11.1: First nibble of data is not written after dummy phase	A	A
	Section 2.11.2: Wrong data can be read in memory-mapped after an indirect mode operation	A	A

2.1 System limitations

2.1.1 Debugging Sleep/Stop mode with WFE/WFI entry

Description

When the Sleep debug or Stop debug mode is enabled (DBG_SLEEP bit or DBG_STOP bit are set in the DBGMCU_CR register), this allows software debugging during Sleep or Stop mode. After wakeup some unreachable instructions could be executed if the following condition are met:

- If the application software disables the Prefetch queue
- The number of wait state configured on Flash interface is higher than 0
- And Linker place WFE or WFI instructions on 4-bytes aligned addresses (0x080xx_xxx4)

Workaround

- Add three NOPs after WFI/WFE instruction
- Keep one AHB master active during sleep (example keep DMA1 or DMA2 RCC clock enable bit set)
- Execute WFI/WFE instruction from routines inside the SRAM

2.1.2 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 which generates the wakeup flag (WUF). The WUF needs to be cleared prior to Standby mode entry, otherwise the MCU wakes up immediately.

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

Workaround

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

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

Note: Be aware that, when applying this workaround, if one of the wakeup sources is still kept high, the MCU enters Standby mode but then it wakes up immediately generating a power reset.

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 MPU attribute to RTC and IWDG registers could be managed incorrectly

Description

If the MPU is used and the non bufferable attribute is set to the RTC or IWDG memory map region, the CPU access to the RTC or IWDG registers could be treated as bufferable, provided that there is no APB prescaler configured (AHB/APB prescaler is equal to 1).

Workaround

If the non bufferable attribute is required for these registers, the software could perform a read after the write to guaranty the completion of the write access.

2.1.5 Delay after an RCC peripheral clock enabling

Description

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

This delay depends on the peripheral's mapping:

- If the peripheral is mapped on AHB: the delay should be equal to 2 AHB cycles.
- If the peripheral is mapped on APB: the delay should be equal to 1 + (AHB/APB prescaler) cycles.

Workarounds

1. Use the DSB instruction to stall the Cortex-M4 CPU pipeline until the instruction is completed.
2. Insert "n" NOPs between the RCC enable bit write and the peripheral register writes (n = 2 for AHB peripherals, n = 1 + AHB/APB prescaler in case of APB peripherals).

2.1.6 Internal noise impacting the ADC accuracy

Description

An internal noise generated on V_{DD} supplies and propagated internally may impact the ADC accuracy.

This noise is always active whatever the power mode of the MCU (RUN or Sleep).

Workarounds

To adapt the accuracy level to the application requirements, set one of the following options:

- Option1
Set the ADCDC1 bit in the PWR_CR register.
- Option2
Set the corresponding ADCxDC2 bit in the SYSCFG_PMC register.

Only one option can be set at a time.

For more details on option 1 and option2 mechanisms, refer to AN4073.

2.1.7 Flash sector erase issue for sectors 5 to 11

Description

Under specific conditions, flash erase issues are observed.

The involved sectors are: 5 to 11

Workaround

Do not perform sector erase on sectors 5 to 11.

Use flash mass erase to erase sectors 5 to 11.

2.1.8 In some specific cases, DMA2 data corruption occurs when managing AHB and APB2 peripherals in a concurrent way

Description

When the DMA2 is managing concurrent requests of AHB and APB2 peripherals, the transfer on the AHB could be performed several times.

Impacted peripheral are:

- **Quad-SPI**: indirect mode read and write transfers
- **FSMC**: read and write operation with external device having FIFO
- **GPIO**: DMA2 transfers to GPIO registers (in memory-to-peripheral transfer mode). The transfers from GPIOs register are not impacted.

The data corruption is due to multiple DMA2 accesses over AHB peripheral port impacting peripherals embedding a FIFO.

For transfer to the internal SRAM through the DMA2 AHB peripheral port the accesses could be performed several times but without data corruptions in cases of concurrent requests.

Workaround

- The DMA2 AHB memory port must be used when reading/writing from/to Quad-SPI and FSMC instead of DMA2 AHB default peripheral port.
- The DMA2 AHB memory port must be used when writing to GPIOs instead of DMA2 AHB default peripheral port.

Refer to application note AN4031 section “Take benefits of DMA2 controller and system architecture flexibility” for more details about DMA controller feature.

2.2 IWDG peripheral limitation

2.2.1 RVU and PVU flags are not reset in STOP mode

Description

The RVU and PVU flags of the IWDG_SR register are set by hardware after a write access to the IWDG_RLR and the IWDG_PR registers, respectively. If the Stop mode is entered immediately after the write access, the RVU and PVU flags are not reset by hardware.

Before performing a second write operation to the IWDG_RLR or the IWDG_PR register, the application software must wait for the RVU or PVU flag to be reset. However, since the RVU/PVU bit is not reset after exiting the Stop mode, the software goes into an infinite loop and the independent watchdog (IWDG) generates a reset after the programmed timeout period.

Workaround

Wait until the RVU or PVU flag of the IWDG_SR register is reset before entering the Stop mode.

2.3 I2C peripheral limitations

2.3.1 SMBus standard not fully supported

Description

The I²C peripheral is not fully compliant with the SMBus v2.0 standard since It does not support the capability to NACK an invalid byte/command.

Workarounds

A higher-level mechanism should be used to verify that a write operation is being performed correctly at the target device, such as:

1. Using the SMBAL pin if supported by the host
2. the alert response address (ARA) protocol
3. the Host notify protocol

2.3.2 Start cannot be generated after a misplaced Stop

Description

If a master generates a misplaced Stop on the bus (bus error), the peripheral cannot generate a Start anymore.

Workaround

In the I²C standard, it is allowed to send a Stop only at the end of the full byte (8 bits + acknowledge), so this scenario is not allowed. Other derived protocols like CBUS allow it, but they are not supported by the I²C peripheral.

A software workaround consists in asserting the software reset using the SWRST bit in the I2C_CR1 control register.

2.3.3 Mismatch on the “Setup time for a repeated Start condition” timing parameter

Description

In case of a repeated Start, the “Setup time for a repeated Start condition” (named $T_{su;sta}$ in the I²C specification) can be slightly violated when the I²C operates in Master Standard mode at a frequency between 88 kHz and 100 kHz.

The limitation can occur only in the following configuration:

- in Master mode
- in Standard mode at a frequency between 88 kHz and 100 kHz (no limitation in Fast-mode)
- SCL rise time:
 - If the slave does not stretch the clock and the SCL rise time is more than 300 ns (if the SCL rise time is less than 300 ns, the limitation cannot occur)
 - If the slave stretches the clock

The setup time can be violated independently of the APB peripheral frequency.

Workaround

Reduce the frequency down to 88 kHz or use the I²C Fast-mode, if supported by the slave.

2.3.4 Data valid time ($t_{VD;DAT}$) violated without the OVR flag being set

Description

The data valid time ($t_{VD;DAT}$, $t_{VD;ACK}$) described by the I²C standard can be violated (as well as the maximum data hold time of the current data ($t_{HD;DAT}$)) under the conditions described below. This violation cannot be detected because the OVR flag is not set (no transmit buffer underrun is detected).

This limitation can occur only under the following conditions:

- in Slave transmit mode
- with clock stretching disabled (NOSTRETCH=1)
- if the software is late to write the DR data register, but not late enough to set the OVR flag (the data register is written before)

Workaround

If the master device allows it, use the clock stretching mechanism by programming the bit NOSTRETCH=0 in the I2C_CR1 register.

If the master device does not allow it, ensure that the software is fast enough when polling the TXE or ADDR flag to immediately write to the DR data register. For instance, use an interrupt on the TXE or ADDR flag and boost its priority to the higher level.

2.3.5 Both SDA and SCL maximum rise time (t_r) violated when VDD_I2C bus higher than $((V_{DD}+0.3) / 0.7)$ V

Description

When an external legacy I²C bus voltage (VDD_I2C) is set to 5 V while the MCU is powered from V_{DD}, the internal 5-Volt tolerant circuitry is activated as soon the input voltage (V_{IN}) reaches the V_{DD} + diode threshold level. An additional internal large capacitance then prevents the external pull-up resistor (R_P) from rising the SDA and SCL signals within the maximum timing (t_r) which is 300 ns in fast mode and 1000 ns in Standard mode.

The rise time (t_r) is measured from V_{IL} and V_{IH} with levels set at 0.3VDD_I2C and 0.7VDD_I2C.

Workaround

The external VDD_I2C bus voltage should be limited to a maximum value of $((V_{DD}+0.3) / 0.7)$ V. As a result, when the MCU is powered from V_{DD}=3.3 V, VDD_I2C should not exceed 5.14 V to be compliant with I²C specifications.

2.3.6 Last received byte can be lost when using Reload mode with NBYTES > 1

Description

The limitation can occur in master mode when reload mode is used (needed for transferring more than 255 bytes), or in slave byte control mode (SBC=1 in the I2C_CR1 register). The limitation occurs only when NBYTES > 1.

In Reload mode (RELOAD=1 in the I2C_CR2 register) with NBYTES programmed with a value N in the I2C_CR2, the Transfer Complete Reload flag (TCR) is set in the I2C_ISR register when the last byte is received in the shift register, even if not yet transferred in the Receive Data Register because the byte N-1 is not yet read.

The last received data is definitively lost (never transferred in the Data Register) if the data N-1 is read between 0 and 4 APB clock cycles before the TCR flag is being set in the I2C_ISR register.

Workaround

In slave byte control mode : Use the Reload mode with NBYTES=1.

In master mode: Do not use the Reload mode. If the number of bytes to be transferred is greater than 255 bytes, the total transfer should be split in several transfers not exceeding 255 bytes, separated by Repeated Start conditions.

Note that the use of DMA could manage that the data N-1 is always transferred before the 4 APB cycles preceding the TCR flag. However this must be evaluated carefully for each application depending on the bus bandwidth, maximum latency, and DMA channel priority.

2.4 FMPI2C peripheral limitation

2.4.1 Wrong data sampling when data set-up time (tSU;DAT) is smaller than one FMPI2CCLK period

Description

The I2C bus specification and user manual specifies a minimum data set-up time (tSU;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 tSU;DAT is smaller than one FMPI2CCLK (FMPI2C clock) period: the previous SDA value is sampled instead of the current one. This can result in a wrong slave address reception, a wrong received data byte, or a wrong received acknowledge bit.

Workaround

Increase the I2CCLK frequency to get I2CCLK period smaller than the transmitter minimum data set-up time. Or, if it is possible, increase the transmitter minimum data set-up time

2.5 SPI/I2S peripheral limitation

2.5.1 Wrong CRC calculation when the polynomial is even.

Description:

When the CRC is enabled, the CRC calculation will be wrong if the polynomial is even.

Workaround:

Use odd polynomial.

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

Description

The BSY flag may sporadically remain high at the end of a data transfer in Slave mode. The issue appears when an accidental synchronization happens between the internal CPU clock and the external SCK clock provided by master.

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

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

Workaround

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

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

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

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

Note: This workaround can be used only when CPU has enough performance to disable SPI after TXE event is detected while the data frame transfer is still ongoing. It is impossible to achieve it when ratio between CPU and SPI clock is low and data frame is short especially. In this specific case timeout can be measured from TXE, while calculating fixed number of CPU clock periods corresponding to the time necessary to complete the data frame transaction.

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

Description

In receive transaction, in both I²S and SPI Master modes, the last bit of the transacted frame is not captured when the signal provided by internal feedback loop from the SCK pin exceeds a critical delay. The lastly transacted bit of the stored data then keeps the value from the pattern received previously. As a consequence, the last receive data bit may be wrong and/or the CRCERR flag can be unduly asserted in the SPI mode if any data under check sum and/or just the CRC pattern is wrongly captured.

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

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

Workarounds

The following workaround can be adopted, jointly or individually:

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

The following table gives the maximum allowable APB frequency versus GPIOx_OSPEEDR output speed control field setting for the SCK pin, at 30 pF of capacitive load.

Table 5. Maximum allowable APB frequency at 30 pF load

OSPEEDR [1:0] for SCK pin	Max. APB frequency for SPI mode [MHz]	Max. APB frequency for I ² S mode [MHz]
11 (very high)	100	84 (100 if $V_{DD} > 2.7$ V)
10 (high)	100	60
01 (medium)	80	30
00 (low)	28	14

2.6 USART peripheral limitations

2.6.1 Idle frame is not detected if receiver clock speed is deviated

Description

If the USART receives an idle frame followed by a character, and the clock of the transmitter device is faster than the USART receiver clock, the USART receive signal falls too early when receiving the character start bit, with the result that the idle frame is not detected (IDLE flag is not set).

Workaround

None.

2.6.2 In full duplex mode, the Parity Error (PE) flag can be cleared by writing to the data register

Description

In full duplex mode, when the Parity Error flag is set by the receiver at the end of a reception, it may be cleared while transmitting by reading the USART_SR register to check the TXE or TC flags and writing data to the data register.

Consequently, the software receiver can read the PE flag as '0' even if a parity error occurred.

Workaround

The Parity Error flag should be checked after the end of reception and before transmission.

2.6.3 Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection

Description

The USART receiver is in Mute mode and is configured to exit the Mute mode using the address mark detection. When the USART receiver recognizes a valid address with a parity error, it exits the Mute mode without setting the Parity Error flag.

Workaround

None.

2.6.4 Break frame is transmitted regardless of nCTS input line status

Description

When CTS hardware flow control is enabled (CTSE = 1) and the Send Break bit (SBK) is set, the transmitter sends a break frame at the end of the current transmission regardless of nCTS input line status.

Consequently, if an external receiver device is not ready to accept a frame, the transmitted break frame is lost.

Workaround

None.

2.6.5 nRTS signal abnormally driven low after a protocol violation

Description

When RTS hardware flow control is enabled, the nRTS signal goes high when data is received. If this data was not read and new data is sent to the USART (protocol violation), the nRTS signal goes back to low level at the end of this new data.

Consequently, the sender gets the wrong information that the USART is ready to receive further data.

On USART side, an overrun is detected, which indicates that data has been lost.

Workaround

Workarounds are required only if the other USART device violates the communication protocol, which is not the case in most applications.

Two workarounds can be used:

- After data reception and before reading the data in the data register, the software takes over the control of the nRTS signal as a GPIO and holds it high as long as needed. If the USART device is not ready, the software holds the nRTS pin high, and releases it when the device is ready to receive new data.
- The time required by the software to read the received data must always be lower than the duration of the second data reception. For example, this can be ensured by treating all the receptions by DMA mode.

2.6.6 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

2.6.7 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 if the following conditions are met:

- 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 D1 data transfer.

Workaround

If the application needs to detect the end of a data transfer, the break request should be issued after checking that the TC flag is set.

2.6.8 Guard time is not respected when data are sent on TXE events

Description

In smartcard mode, when sending a data on TXE event, the programmed guard time is not respected i.e. the data written in the data register is transferred on the bus without waiting the completion of the guardtime duration corresponding to the previous transmitted data.

Workaround

Write the data after TC is set because in smartcard mode, the TC flag is set at the end of the guard time duration.

2.6.9 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 if the receiver is disabled (RE=0) i.e. not ready to receive data.

Workaround

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

2.7 bxCAN limitation

2.7.1 bxCAN time triggered communication mode not supported

Description

The time triggered communication mode described in the reference manual is not supported. As a result timestamp values are not available. TTCM bit must be kept cleared in the CAN_MCR register (time triggered communication mode disabled).

Workaround

None

2.8 FSMC peripheral limitation

2.8.1 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 FSMC and there is no functional failure.

Workaround

None.

2.9 SDIO peripheral limitations

2.9.1 Wrong CCRCFAIL status after a response without CRC is received

Description

The CRC is calculated even if the response to a command does not contain any CRC field. As a consequence, after the SDIO command IO_SEND_OP_COND (CMD5) is sent, the CCRCFAIL bit of the SDIO_STA register is set.

Workaround

The CCRCFAIL bit in the SDIO_STA register shall be ignored by the software. CCRCFAIL must be cleared by setting CCRCFAILC bit of the SDIO_ICR register after reception of the response to the CMD5 command.

2.9.2 No underrun detection with wrong data transmission

Description

In case there is an ongoing data transfer from the SDIO host to the SD card and the hardware flow control is disabled (bit 14 of the SDIO_CLKCR is not set), if an underrun condition occurs, the controller may transmit a corrupted data block (with wrong data word) without detecting the underrun condition when the clock frequencies have the following relationship:

$$[3 \times \text{period}(\text{PCLK2}) + 3 \times \text{period}(\text{SDIOCLK})] \geq (32 / (\text{BusWidth})) \times \text{period}(\text{SDIO_CK})$$

Workaround

Avoid the above-mentioned clock frequency relationship, by:

- Incrementing the APB frequency
- or decreasing the transfer bandwidth
- or reducing SDIO_CK frequency

2.10 ADC peripheral limitations

2.10.1 ADC sequencer modification during conversion

Description

If an ADC conversion is started by software (writing the SWSTART bit), and if the ADC_SQRx or ADC_JSQRx registers are modified during the conversion, the current conversion is reset and the ADC does not restart a new conversion sequence automatically.

If an ADC conversion is started by hardware trigger, this limitation does not apply. The ADC restarts a new conversion sequence automatically.

Workaround

When an ADC conversion sequence is started by software, a new conversion sequence can be restarted only by setting the SWSTART bit in the ADC_CR2 register.

2.11 QuadSPI limitations

2.11.1 First nibble of data is not written after dummy phase

Description:

- The first nibble of data to be written to an external flash is lost if:
- QUADSPI is used in indirect write mode, and
- at least one dummy cycle is used

Workaround

Do not use dummy cycles for creating latency between address phase and data phase, in indirect write mode. Instead, use alternate bytes to substitute the dummy cycles. The same

latency can be achieved if the number of dummy cycles to substitute with alternate-byte cycles is an integer multiple of the number of cycles required for transferring one alternate byte, as shown in the table:

QUADSPI mode	Number of cycles per alternative byte
4-data-line DDR	1
4-data-line SDR	2
2-data-line SDR	4
1-data-line SDR	8

For example, the latency corresponding to eight dummy cycles can be exactly substituted with one single alternate byte in 1-data-line SDR mode, but two alternate bytes are required in 2-data-line SDR mode. One single dummy cycle can only exactly be substituted in 4-data-line DDR mode, using one alternate byte.

Note: This is also applicable to dual-flash memory mode.

2.11.2 Wrong data can be read in memory-mapped after an indirect mode operation

Description

Wrong data can be read with the first memory-mapped read request in the following condition:

Quad-SPI peripheral entered memory-mapped mode with both LSB bits in the address register QUADSPI_AR[1:0] not reset.

Workaround

QUADSPI_AR register must be reset just before entering memory-mapped mode.

Depending on the current Quad-SPI operating mode, one of the two workarounds listed below can be used:

- Indirect read mode: reset address register then do an abort request to stop reading and clear busy bit. Then enter to memory-mapped mode.
- Indirect write mode: reset the address register then enter to memory-mapped mode.

Note: User should take care to not write to QUADSPI_DR register after resetting address register.

3 Revision history

Table 6. Document revision history

Date	Revision	Changes
21-Sep-2015	1	Initial release
24-Mar-2016	2	Updated: – Table 1: Device identification – Table 4: Summary of silicon limitations
15-Apr-2016	3	Updated: – Table 1: Device identification – Table 4: Summary of silicon limitations
23-May-2016	4	Updated: – Table 1: Device identification – Table 4: Summary of silicon limitations
17-Jun-2016	5	Updated: – Table 1: Device identification Added: – Section 2.1.8: In some specific cases, DMA2 data corruption occurs when managing AHB and APB2 peripherals in a concurrent way Removed: – Section 2.12: Quad-SPI limitation
30-Jun-2016	6	Updated: – Table 4: Summary of silicon limitations – Section 2.5: SPI/I2S peripheral limitation Removed: – Section 2.11: DAC peripheral limitations
25-Jan-2017	7	Updated: – Table 4: Summary of silicon limitations – Section 2.5: SPI/I2S peripheral limitation – Section 2.5.2: BSY bit may stay high at the end of a data transfer in slave mode – Section 2.5.3: Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback Added: – Table 5: Maximum allowable APB frequency at 30 pF load
16-Oct-2017	8	Updated: – Table 4: Summary of silicon limitations Added: – Section 2.3.6: Last received byte can be lost when using Reload mode with NBYTES > 1 – Section 2.11: QuadSPI limitations – Section 2.11.1: First nibble of data is not written after dummy phase – Section 2.11.2: Wrong data can be read in memory-mapped after an indirect mode operation

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2017 STMicroelectronics – All rights reserved