STM32F302x6/x8 device errata

Applicability

This document applies to the part numbers of STM32F302x6/x8 devices and the device variants as stated in this page.

It gives a summary and a description of the device errata, with respect to the device datasheet and reference manual RM0366.

Deviation of the real device behavior from the intended device behavior is considered to be a device limitation. Deviation of the description in the reference manual or the datasheet from the intended device behavior is considered to be a documentation erratum. The term *"errata"* applies both to limitations and documentation errata.

Table 1. Device summary

Reference	Part numbers
STM32F302x6	STM32F302C6, STM32F302K6, STM32F302R6
STM32F302x8	STM32F302C8, STM32F302K8, STM32F302R8

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32F302x6/x8	Z	0x1001

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

1 Summary of device errata

The following table gives a quick reference to the STM32F302x6/x8 device limitations and their status:

A = workaround available

N = no workaround available

P = partial workaround available

Applicability of a workaround may depend on specific conditions of target application. Adoption of a workaround may cause restrictions to target application. Workaround for a limitation is deemed partial if it only reduces the rate of occurrence and/or consequences of the limitation, or if it is fully effective for only a subset of instances on the device or in only a subset of operating modes, of the function concerned.

Table 3. Summary of device limitations

Function	Section	Limitation	Status
			Rev. Z
Core	2.1.1	Interrupted loads to SP can cause erroneous behavior	A
	2.1.2	VDIV or VSQRT instructions might not complete correctly when very short ISRs are used	A
	2.1.3	Store immediate overlapping exception return operation might vector to incorrect interrupt	A
System	2.2.1	Wakeup sequence from Standby mode when using more than one wakeup source	A
	2.2.2	Full JTAG configuration without NJTRST pin cannot be used	A
GPIO	2.3.1	GPIOx locking mechanism is not working properly for GPIOx_OTYPE register	A
ADC	2.4.1	Injected queue of context is not available in case of JQM = 0	N
	2.4.2	Sampling time shortened in JAUTO auto delayed mode	A
	2.4.3	Load multiple not supported by ADC interface	A
	2.4.4	ADEN bit cannot be set immediately after the ADC calibration is done	A
	2.4.5	Overrun flag might not be set when converted data have not been read before new data are written	A
	2.4.6	ADC differential mode Common mode input range	N
	2.4.7	Inaccurate VREFINT calibration values	N
COMP	2.5.1	VREFINT scaler startup time from power down parameter degradation	N
OPAMP	2.6.1	OPAMP timer-controlled multiplexer mode not working when OPAMP is used in PGA or follower mode	N
TIM	2.8.1	PWM re-enabled in automatic output enable mode despite of system break	P
	2.8.2	One-pulse mode trigger not detected in master-slave reset + trigger configuration	P
RTC and TAMP	2.9.1	Spurious tamper detection when disabling the tamper channel	N
	2.9.2	RTC calendar registers are not locked properly	A
	2.9.3	RTC interrupt can be masked by another RTC interrupt	A
	2.9.4	Calendar initialization may fail in case of consecutive INIT mode entry	A
	2.9.5	Alarm flag may be repeatedly set when the core is stopped in debug	N

Function	Section	Limitation	Status Rev. Z
I2C	2.10.1	10-bit slave mode: wrong direction bit value upon Read header receipt	A
	2.10.2	10-bit combined with 7-bit slave mode: ADDCODE may indicate wrong slave address detection	N
	2.10.3	Wakeup frames may not wake up the MCU when Stop mode entry follows I2C enabling	A
	2.10.4	Wakeup frame may not wake up the MCU from Stop mode if t close to I2C kernel clock startup time	P
	2.10.5	10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave	A
	2.10.7	Wrong data sampling when data setup time (tSU;DAT) is shorter than one I2C kernel clock period	P
	2.10.8	Spurious bus error detection in master mode	A
	2.10.9	Last-received byte loss in reload mode	P
	2.10.10	Spurious master transfer upon own slave address match	P
USART	2.11.1	Non-compliant sampling for NACK signal from smartcard	N
	2.11.2	Break request preventing TC flag from being set	A
	2.11.3	nRTS is active while RE = 0 or UE = 0	A
	2.11.5	When PCLK is selected as clock source for USART1, PCLK1 is used instead of PCLK2	A
SPI	2.12.1	BSY bit may stay high when SPI is disabled	A
	2.12.2	BSY bit may stay high at the end of data transfer in slave mode	A
	2.12.3	SPI CRC corruption upon DMA transaction completion by another peripheral	P
	2.12.4	In I2S slave mode, enabling I2S while WS is active causes desynchronization	A

The following table gives a quick reference to the documentation errata.

Table 4. Summary of device documentation errata

Function	Section	Documentation erratum
TSC	2.7.1	Inhibited acquisition in short transfer phase configuration
I2C	2.10.6	Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C
USART	2.11.4	Receiver timeout counter wrong start in two-stop-bit configuration
SPI	2.12.5	CRC error in SPI slave mode if internal NSS changes before CRC transfer

2 Description of device errata

The following sections describe limitations of the applicable devices with Arm® core and provide workarounds if available. They are grouped by device functions.

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

2.1 Core

Errata notice for the Arm® Cortex®-M4F core revision r0p1 is available from http://infocenter.arm.com.

2.1.1 Interrupted loads to SP can cause erroneous behavior

This limitation is registered under Arm ID number 752770 and classified into “Category B”. Its impact to the device is minor.

Description

If an interrupt occurs during the data-phase of a single word load to the stack-pointer (SP/R13), erroneous behavior can occur. In all cases, returning from the interrupt will result in the load instruction being executed an additional time. For all instructions performing an update to the base register, the base register will be erroneously updated on each execution, resulting in the stack-pointer being loaded from an incorrect memory location.

The affected instructions that can result in the load transaction being repeated are:

- LDR SP, [Rn],#imm
- LDR SP, [Rn,#imm]!
- LDR SP, [Rn,#imm]
- LDR SP, [Rn]
- LDR SP, [Rn,Rm]

The affected instructions that can result in the stack-pointer being loaded from an incorrect memory address are:

- LDR SP,[Rn],#imm
- LDR SP,[Rn,#imm]!

As compilers do not generate these particular instructions, the limitation is only likely to occur with hand-written assembly code.

Workaround

Both issues may be worked around by replacing the direct load to the stack-pointer, with an intermediate load to a general-purpose register followed by a move to the stack-pointer.

2.1.2 VDIV or VSQRT instructions might not complete correctly when very short ISRs are used

This limitation is registered under Arm ID number 776924 and classified into “Category B”. Its impact to the device is limited.

Description

The VDIV and VSQRT instructions take 14 cycles to execute. When an interrupt is taken a VDIV or VSQRT instruction is not terminated, and completes its execution while the interrupt stacking occurs. If lazy context save of floating point state is enabled then the automatic stacking of the floating point context does not occur until a floating point instruction is executed inside the interrupt service routine.

Lazy context save is enabled by default. When it is enabled, the minimum time for the first instruction in the interrupt service routine to start executing is 12 cycles. In certain timing conditions, and if there is only one or two instructions inside the interrupt service routine, then the VDIV or VSQRT instruction might not write its result to the register bank or to the FPSCR.

The failure occurs when the following condition is met:

1. The floating point unit is enabled
2. Lazy context saving is not disabled
3. A VDIV or VSQRT is executed
4. The destination register for the VDIV or VSQRT is one of s0 - s15
5. An interrupt occurs and is taken
6. The interrupt service routine being executed does not contain a floating point instruction
7. Within 14 cycles after the VDIV or VSQRT is executed, an interrupt return is executed

A minimum of 12 of these 14 cycles are utilized for the context state stacking, which leaves 2 cycles for instructions inside the interrupt service routine, or 2 wait states applied to the entire stacking sequence (which means that it is not a constant wait state for every access).

In general, this means that if the memory system inserts wait states for stack transactions (that is, external memory is used for stack data), then this erratum cannot be observed.

The effect of this erratum is that the VDIV or VQSRT instruction does not complete correctly and the register bank and FPSCR are not updated, which means that these registers hold incorrect, out of date, data.

Workaround

A workaround is only required if the floating point unit is enabled. A workaround is not required if the stack is in external memory.

There are two possible workarounds:

- Disable lazy context save of floating point state by clearing LSPEN to 0 (bit 30 of the FPCCR at address 0xE000EF34).
- Ensure that every interrupt service routine contains more than 2 instructions in addition to the exception return instruction.

2.1.3 Store immediate overlapping exception return operation might vector to incorrect interrupt

This limitation is registered under Arm ID number 838869 and classified into “Category B (rare)”. Its impact to the device is minor.

Description

The core includes a write buffer that permits execution to continue while a store is waiting on the bus. Under specific timing conditions, during an exception return while this buffer is still in use by a store instruction, a late change in selection of the next interrupt to be taken might result in there being a mismatch between the interrupt acknowledged by the interrupt controller and the vector fetched by the processor.

The failure occurs when the following condition is met:

1. The handler for interrupt A is being executed.
2. Interrupt B, of the same or lower priority than interrupt A, is pending.
3. A store with immediate offset instruction is executed to a bufferable location.
   - STR/STRH/STRB <Rt>, [<Rn>,#imm]
   - STR/STRH/STRB <Rt>, [<Rn>,#imm]!
   - STR/STRH/STRB <Rt>, [<Rn>],#imm
4. Any number of additional data-processing instructions can be executed.
5. A BX instruction is executed that causes an exception return.
6. The store data has wait states applied to it such that the data is accepted at least two cycles after the BX is executed.
   - Minimally, this is two cycles if the store and the BX instruction have no additional instructions between them.
   - The number of wait states required to observe this erratum needs to be increased by the number of cycles between the store and the interrupt service routine exit instruction.
7. Before the bus accepts the buffered store data, another interrupt C is asserted which has the same or lower priority as A, but a greater priority than B.

Example:

The processor should execute interrupt handler C, and on completion of handler C should execute the handler for B. If the conditions above are met, then this erratum results in the processor erroneously clearing the pending state of interrupt C, and then executing the handler for B twice. The first time the handler for B is executed it will be at interrupt C's priority level. If interrupt C is pended by a level-based interrupt which is cleared by C's handler then interrupt C will be pended again once the handler for B has completed and the handler for C will be executed.

As the STM32 interrupt C is level based, it eventually becomes pending again and is subsequently handled.

Workaround

For software not using the memory protection unit, this erratum can be worked around by setting DISDEFWBUF in the Auxiliary Control Register.

In all other cases, the erratum can be avoided by ensuring a DSB occurs between the store and the BX instruction. For exception handlers written in C, this can be achieved by inserting the appropriate set of intrinsics or inline assembly just before the end of the interrupt function, for example:

ARMCC:

```
...
__schedule_barrier();
__asm{DSB};
__schedule_barrier();
}
```

GCC:

```
...
__asm volatile ("dsb 0xf":::"memory");
}
```

2.2 System

2.2.1 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.

2.2.2 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.

2.3 GPIO

2.3.1 GPIOx locking mechanism is not working properly for GPIOx_OTYPE register

Description

Locking of GPIOx_OTYPER[i] with i = 15..8 depends on the setting of GPIOx_LCKR[i-8] and not from the setting of GPIOx_LCKR[i]. GPIOx_LCKR[i-8] locks GPIOx_OTYPER[i] together with GPIOx_OTYPER[i-8]. It is not possible to lock GPIOx_OTYPER[i] with i = 15...8, without locking also GPIOx_OTYPER[i-8].

Workaround

The only way to lock GPIOx_OTYPER[i] with i=15..8 is to lock also GPIOx_OTYPER[i-8].

2.4 ADC

2.4.1 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 leads to a queue overflow and is ignored. Consequently, the ADC must be stopped before programming the JSQR register.

Workaround

None.

2.4.2 Sampling time shortened in JAUTO auto delayed mode

Description

When the ADC is configured in JAUTO single conversion mode (CONT=0), with auto delayed 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.

2.4.3 Load multiple not supported by ADC interface

Description

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

Workaround

The workaround consists in preventing compilers from generating LDM, STM, LDRD and STRD instructions.

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

Description

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

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

Workaround

Adopt one of the following measures:

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

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

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) might not be set to ‘1’.

Workaround

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

2.4.6 ADC differential mode Common mode input range

Description

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

Workaround

None.

2.4.7 Inaccurate VREFINT calibration values

Description

Devices with a date code earlier than 648 may have inaccurate VREFINT calibration values.

Workaround

None.

2.5 COMP

2.5.1 VREFINT scaler startup time from power down parameter degradation

Description

The VREFINT scaler is an embedded voltage follower providing the VREFINT or its fractions (1/2, 1/4 or 3/4) to the comparator input. The maximum VREFINT scaler startup time, $t_{S\_SC(max)}$, is not as expected for the first activation of the VREFINT scaler after powering on the device and it can be up to 1s in worse case conditions. This maximum value depends mainly on the voltage and temperature, see the device datasheet for more details. If the VREFINT scaler is used, the comparator analog supply voltage V_{DDA} equal or greater than 2 V.

Workaround

None.

2.6 OPAMP

2.6.1 OPAMP timer-controlled multiplexer mode not working when OPAMP is used in PGA or follower mode

Description

When OPAMP operates in PGA or follower mode, the timer-controlled multiplexer function cannot be used.

Note: The timer-controlled multiplexer function can only be used with OPAMP in standalone mode.

Workaround

None.

2.7 TSC

2.7.1 Inhibited acquisition in short transfer phase configuration

Description

Some revisions of the reference manual may omit the information that the following configurations of the TSC_CR register are forbidden:

- The PGPSC[2:0] bitfield set to 000 and the CTPL[3:0] bitfield to 0000 or 0001
- The PGPSC[2:0] bitfield set to 111 and the CTPL[3:0] bitfield to 0000

Failure to respect this restriction leads to an inhibition of the acquisition.

This is a documentation inaccuracy issue rather than a product limitation.

Workaround

No application workaround is required.

2.8 TIM

2.8.1 PWM re-enabled in automatic output enable mode despite of system break

Description

In automatic output enable mode (AOE bit set in TIMx_BDTR register), the break input can be used to do a cycle-by-cycle PWM control for a current mode regulation. A break signal (typically a comparator with a current threshold) disables the PWM output(s) and the PWM is re-armed on the next counter period.

However, a system break (typically coming from the CSS Clock security System) is supposed to stop definitively the PWM to avoid abnormal operation (e.g. with PWM frequency deviation).

In the current implementation, the timer system break input is not latched. As a consequence, a system break will indeed disable the PWM output(s) when it occurs, but PWM output(s) will be re-armed on the next counter period.

Workaround

The control loops should preferably be implemented with the output clear enable function (OCxCE bit in the TIMx_CCMR1/CCMR2 register), leaving the use of break circuitry solely for internal and/or external fault protection (AOE bit reset).

2.8.2 One-pulse mode trigger not detected in master-slave reset + trigger configuration

Description

The failure occurs when several timers configured in one-pulse mode are cascaded, and the master timer is configured in combined reset + trigger mode with the MSM bit set:

OPM=1 in TIMx_CR1, SMS[3:0]=1000 and MSM=1 in TIMx_SMCR.

The MSM delays the reaction of the master timer to the trigger event, so as to have the slave timers cycle-accurately synchronized.

If the trigger arrives when the counter value is equal to the period value set in the TIMx_ARR register, the one-pulse mode of the master timer does not work and no pulse is generated on the output.

Workaround

None. However, unless a cycle-level synchronization is mandatory, it is advised to keep the MSM bit reset, in which case the problem is not present. The MSM=0 configuration also allows to decrease the timer latency to external trigger events.

2.9 RTC and TAMP

2.9.1 Spurious tamper detection when disabling the tamper channel

Description

If the tamper detection is configured for detecting on the falling edge event (TAMPFLT = 00 and TAMPxTRG = 1) and if the tamper event detection is disabled when the tamper pin is at high level, a false tamper event is detected.

Workaround

None.

2.9.2 RTC calendar registers are not locked properly

Description

When reading the calendar registers with BYPSHAD = 0, the RTC_TR and RTC_DR registers may not be locked after reading the RTC_SSR register. This happens if the read operation is initiated one APB clock period before the shadow registers are updated. This can result in a non-consistency of the three registers. Similarly, the RTC_DR register can be updated after reading the RTC_TR register instead of being locked.

Workaround

Apply one of the following measures:

- use BYPSHAD = 1 mode (bypass shadow registers), or
- if BYPSHAD = 0, read SSR again after reading SSR/TR/DR to confirm that SSR is still the same, otherwise read the values again.

2.9.3 RTC interrupt can be masked by another RTC interrupt

Description

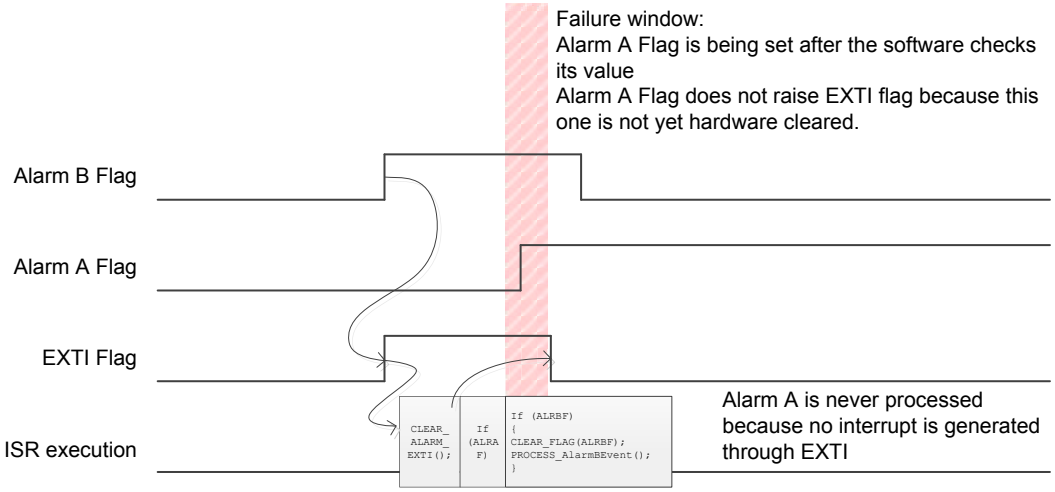
One RTC interrupt can mask another RTC interrupt if both share the same EXTI configurable line, such as the RTC Alarm A and Alarm B, of which the event flags are OR-de to the same EXTI line (refer to the **EXTI line connections** table in the **Extended interrupt and event controller (EXTI)** section of the reference manual).

The following code example and figure illustrate the failure mechanism: The Alarm A event is lost (fails to generate interrupt) as it occurs in the failure window, that is, after checking the Alarm A event flag but before the effective clear of the EXTI interrupt flag by hardware. The effective clear of the EXTI interrupt flag is delayed with respect to the software instruction to clear it.

Alarm interrupt service routine:

```
void RTC_Alarm_IRQHandler(void)
{
    CLEAR_ALARM_EXTI(); /* Clear the EXTI line flag for RTC alarms*/
    If(ALRAF) /* Check if Alarm A triggered ISR */
    {
        CLEAR_FLAG(ALRAF); /* Clear the Alarm A interrupt pending bit */
        PROCESS_AlarmAEvent(); /* Process Alarm A event */
    }
    If(ALRBF) /* Check if Alarm B triggered ISR */
    {
        CLEAR_FLAG(ALRBF); /* Clear the Alarm B interrupt pending bit */
        PROCESS_AlarmBEvent(); /* Process Alarm B event */
    }
}
```

Figure 1. Masked RTC interrupt



Workaround

In the interrupt service routine, apply three consecutive event flag ckecks - source one, source two, and source one again, as in the following code example:

```
void RTC_Alarm_IRQHandler(void)
{
    CLEAR_ALARM_EXTI(); /* Clear the EXTI's line Flag for RTC Alarm */
    If(ALRAF) /* Check if AlarmA triggered ISR */
    {
        CLEAR_FLAG(ALRAF); /* Clear the AlarmA interrupt pending bit */
        PROCESS_AlarmAEvent(); /* Process AlarmA Event */
    }
    If(ALRBF) /* Check if AlarmB triggered ISR */
    {
        CLEAR_FLAG(ALRBF); /* Clear the AlarmB interrupt pending bit */
        PROCESS_AlarmBEvent(); /* Process AlarmB Event */
    }
    If(ALRAF) /* Check if AlarmA triggered ISR */
    {
        CLEAR_FLAG(ALRAF); /* Clear the AlarmA interrupt pending bit */
        PROCESS_AlarmAEvent(); /* Process AlarmA Event */
    }
}
```

2.9.4 Calendar initialization may fail in case of consecutive INIT mode entry

Description

If the INIT bit of the RTC_ISR register is set between one and two RTCCLK cycles after being cleared, the INITF flag is set immediately instead of waiting for synchronization delay (which should be between one and two RTCCLK cycles), and the initialization of registers may fail.

Depending on the INIT bit clearing and setting instants versus the RTCCLK edges, it can happen that, after being immediately set, the INITF flag is cleared during one RTCCLK period then set again. As writes to calendar registers are ignored when INITF is low, a write during this critical period might result in the corruption of one or more calendar registers.

Workaround

After existing the initialization mode, clear the BYPSHAD bit (if set) then wait for RSF to rise, before entering the initialization mode again.

Note: It is recommended to write all registers in a single initialization session to avoid accumulating synchronization delays.

2.9.5 Alarm flag may be repeatedly set when the core is stopped in debug

Description

When the core is stopped in debug mode, the clock is supplied to subsecond RTC alarm downcounter even though the device is configured to stop the RTC in debug.

As a consequence, when the subsecond counter is used for alarm condition (the MASKSS[3:0] bitfield of the RTC_ALRMASSR and/or RTC_ALRMBSSR register set to a non-zero value) and the alarm condition is met just before entering a breakpoint or printf, the ALRAF and/or ALRBF flag of the RTC_SR register is repeatedly set by hardware during the breakpoint or printf, which makes any tentative to clear the flag(s) ineffective.

Workaround

None.

2.10 I2C

2.10.1 10-bit slave mode: wrong direction bit value upon Read header receipt

Description

Under specific conditions, the transfer direction bit DIR (bit 16 of status register I2C_ISR) remains low upon receipt of 10-bit addressing Read header, while normally it should be set high. Nevertheless, I2C operates correctly in slave transmission mode, and data can be sent using the TXIS flag.

The failure described occurs when the following conditions are all met:

- I2C is configured in 10-bit addressing mode (OA1MODE is set in the I2C_OAR1 register).
- High LSBs of the slave address are equal to the 10-bit addressing Read header value (that is, OA1[7:3] = 11110, OA1[2] = OA1[9], OA1[1] = OA1[8], and OA1[0] = 1, in the I2C_OAR1 register).
- I2C receives 10-bit addressing Read header (0X 1111 0XX1) after repeated START condition, to enter slave transmission mode.

Workaround

Avoid using the following 10-bit slave addresses:

- OA1[9:0] = 0011110001
- OA1[9:0] = 0111110011
- OA1[9:0] = 1011110101
- OA1[9:0] = 1111110111

If the use of one of these slave addresses cannot be avoided, do not use the DIR bit in the firmware.

2.10.2 10-bit combined with 7-bit slave mode: ADDCODE may indicate wrong slave address detection

Description

Under specific conditions, the ADDCODE (address match code) bitfield in the I2C_ISR register indicates a wrong slave address.

The failure occurs when the following conditions are all met:

- A 10-bit slave address OA1 is enabled (OA1EN = 1 and OA1MODE = 1).
- A 7-bit slave address OA2 is enabled (OA2EN = 1) and it matches the non-masked bits of OA1[7:1], that is, one of the following configurations 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] = 0000000
  - ALERTEN = 1 and OA1[7:1] = 0001100
  - SMBDEN = 1 and OA1[7:1] = 1100001
  - SMBHEN = 1 and OA1[7:1] = 0001000
- The MCU is addressed by a bus master with its 10-bit slave address OA1.

Upon the address receipt, 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 OA1 [7:1] part of the 10-bit slave address must be different than the 7-bit slave address.

2.10.3 Wakeup frames may not wake up the MCU when Stop mode entry follows I2C enabling

Description

If I2C is enabled (PE = 1) and wakeup from Stop mode is enabled in I2C (WUPEN = 1) while a transfer occurs on the I^2C-bus and Stop mode is entered during the same transfer while SCL = 0, I2C is not able to detect the following START condition. As a consequence, the MCU does not wake up from Stop mode when it is addressed on the I^2C-bus and it does not acknowledge the receipt of the address.

Workaround

After enabling I2C (by setting PE to 1), do not enter Stop mode until any I^2C-bus transaction in progress ends.

2.10.4 Wakeup frame may not wake up the MCU from Stop mode if t close to I2C kernel clock startup time

Description

Under specific conditions and if the START condition hold time $t_{HD;STA}$ is very close to the startup time of the internal oscillator selected for I2C kernel clock, I2C is not able to detect the address match and, as a consequence, to wake up the MCU from Stop mode.

The failure described occurs when one of the following condition is met:

1. Timeout detection is enabled (TIMOUTEN = 1 or TEXTEN = 1) and the frame before the wakeup frame is finished abnormally due to I2C timeout detection (TIMOUT = 1).
2. Slave arbitration is lost during the frame preceding the wakeup frame (ARLO = 1).

3. The MCU enters Stop mode while another slave is addressed, after the address phase and before STOP condition (BUSY = 1).
4. The MCU is in Stop mode and another slave is addressed before the MCU itself is addressed.

Note: The conditions 2, 3 and 4 can only occur in a multi-slave network.

In Stop mode, the internal oscillator selected for I2C kernel clock is switched on by I2C when START condition is detected. The I2C kernel clock is then used to receive the address. The internal oscillator is switched off upon the address receipt if the address received does not match the own slave address. If one of the conditions listed is met and if the SCL falling edge following the START condition occurs within the first cycle of the I2C kernel clock, the address is received incorrectly and the address match wakeup interrupt is not generated.

Workaround

None at MCU level.

Upon non-acknowledge by the MCU of a wakeup frame, the I^2C-bus master with programmable START condition hold time can set that hold time such that it exceeds one MCU internal oscillator period, then resend the wakeup frame.

2.10.5 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave

Description

An I^2C-bus master generates STOP condition upon non-acknowledge of I^2C address that it sends. This applies to 7-bit address as well as to each byte of 10-bit address.

When the MCU set as I^2C-bus master transmits a 10-bit address of which the first byte (5-bit header + 2 MSBs of the address + direction bit) is not acknowledged, the MCU duly generates STOP condition but it then cannot start any new I^2C-bus transfer. In this spurious state, the NACKF flag of the I2C_ISR register and the START bit of the I2C_CR2 register are both set, while the START bit should normally be cleared.

Workaround

In 10-bit-address master mode, if both NACKF flag and START bit get simultaneously set, proceed as follows:

1. Wait for the STOP condition detection (STOPF = 1 in I2C_ISR register).
2. Disable the I2C peripheral.
3. Wait for a minimum of three APB cycles.
4. Enable the I2C peripheral again.

2.10.6 Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C

Description

If the wakeup from Stop mode by I2C is disabled (WUPEN = 0), the correct use of the I2C peripheral is to disable it (PE = 0) before entering Stop mode, and re-enable it when back in Run mode.

Some reference manual revisions may omit this information.

Failure to respect the above while the MCU operating as slave or as master in multi-master topology enters Stop mode during a transfer ongoing on the I^2C-bus may lead to the following:

1. BUSY flag is 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 SCL line is pulled low by I2C and the transfer stalled as long as the MCU remains in Stop mode.

   The occurrence of such condition depends on the timing configuration, peripheral clock frequency, and I^2C-bus frequency.

This is a description inaccuracy issue rather than a product limitation.

Workaround

No application workaround is required.

2.10.7 Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period

Description

The I^2C-bus specification and user manual specify a minimum data setup time ($t_{SU;DAT}$) as:

- 250 ns in Standard mode
- 100 ns in Fast mode
- 50 ns in Fast mode Plus

The MCU does not correctly sample the I^2C-bus SDA line when $t_{SU;DAT}$ is smaller than one I2C kernel clock (I^2C-bus peripheral clock) period: the previous SDA value is sampled instead of the current one. This can result in a wrong receipt of slave address, data byte, or acknowledge bit.

Workaround

Increase the I2C kernel clock frequency to get I2C kernel clock period within the transmitter minimum data setup time. Alternatively, increase transmitter's minimum data setup time. If the transmitter setup time minimum value corresponds to the minimum value provided in the I^2C-bus standard, the minimum I2CCLK frequencies are as follows:

- In Standard mode, if the transmitter minimum setup time is 250 ns, the I2CCLK frequency must be at least 4 MHz.
- In Fast mode, if the transmitter minimum setup time is 100 ns, the I2CCLK frequency must be at least 10 MHz.
- In Fast-mode Plus, if the transmitter minimum setup time is 50 ns, the I2CCLK frequency must be at least 20 MHz.

2.10.8 Spurious bus error detection in master mode

Description

In master mode, a bus error can be detected spuriously, with the consequence of setting the BERR flag of the I2C_SR register and generating bus error interrupt if such interrupt is enabled. Detection of bus error has no effect on the I^2C-bus transfer in master mode and any such transfer continues 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.

2.10.9 Last-received byte loss in reload mode

Description

If in master receiver mode or slave receive mode with SBC = 1 the following conditions are all met:

- I^2C-bus stretching is enabled (NOSTRETCH = 0)
- RELOAD bit of the I2C_CR2 register is set
- NBYTES bitfield of the I2C_CR2 register is set to N greater than 1
- byte N is received on the I^2C-bus, raising the TCR flag
- N - 1 byte is not yet read out from the data register at the instant TCR is raised,

then the SCL line is pulled low (I^2C-bus clock stretching) and the transfer of the byte N from the shift register to the data register inhibited until the byte N-1 is read and NBYTES bitfield reloaded with a new value, the latter of which also clears the TCR flag. As a consequence, the software cannot get the byte N and use its content before setting the new value into the NBYTES field.

For I2C instances with independent clock, the last-received data is definitively lost (never transferred from the shift register to the data register) if the data N - 1 is read within four APB clock cycles preceding the receipt of the last data bit of byte N and thus the TCR flag raising. Refer to the product reference manual or datasheet for the I2C implementation table.

Workaround

- In master mode or in slave mode with SBC = 1, use the reload mode with NBYTES = 1.
- In master receiver mode, if the number of bytes to transfer is greater than 255, do not use the reload mode. Instead, split the transfer into sections not exceeding 255 bytes and separate them with repeated START conditions.
- Make sure, for example through the use of DMA, that the byte N - 1 is always read before the TCR flag is raised. Specifically for I2C instances with independent clock, make sure that it is always read earlier than four APB clock cycles before the receipt of the last data bit of byte N and thus the TCR flag raising.

The last workaround in the list must be evaluated carefully for each application as the timing depends on factors such as the bus speed, interrupt management, software processing latencies, and DMA channel priority.

2.10.10 Spurious master transfer upon own slave address match

Description

When the device is configured to operate at the same time as master and slave (in a multi- master I^2C-bus application), a spurious master transfer may occur under the following condition:

- Another master on the bus is in process of sending the slave address of the device (the bus is busy).
- The device initiates a master transfer by bit set before the slave address match event (the ADDR flag set in the I2C_ISR register) occurs.
- After the ADDR flag is set:
  - the device does not write I2C_CR2 before clearing the ADDR flag, or
  - the device writes I2C_CR2 earlier than three I2C kernel clock cycles before clearing the ADDR flag

In these circumstances, even though the START bit is automatically cleared by the circuitry handling the ADDR flag, the device spuriously proceeds to the master transfer as soon as the bus becomes free. The transfer configuration depends on the content of the I2C_CR2 register when the master transfer starts. Moreover, if the I2C_CR2 is written less than three kernel clocks before the ADDR flag is cleared, the I2C peripheral may fall into an unpredictable state.

Workaround

Upon the address match event (ADDR flag set), apply the following sequence.

Normal mode (SBC = 0):

1. Set the ADDRCF bit.
2. Before Stop condition occurs on the bus, write I2C_CR2 with the START bit low.

Slave byte control mode (SBC = 1):

1. Write I2C_CR2 with the slave transfer configuration and the START bit low.
2. Wait for longer than three I2C kernel clock cycles.
3. Set the ADDRCF bit.
4. Before Stop condition occurs on the bus, write I2C_CR2 again with its current value.

The time for the software application to write the I2C_CR2 register before the Stop condition is limited, as the clock stretching (if enabled), is aborted when clearing the ADDR flag.

Polling the BUSY flag before requesting the master transfer is not a reliable workaround as the bus may become busy between the BUSY flag check and the write into the I2C_CR2 register with the START bit set.

2.11 USART

2.11.1 Non-compliant sampling for NACK signal from smartcard

Description

According to ISO/IEC 7816-3 standard, when a character parity error is detected, the receiver must assert a NACK signal, by pulling the transmit line low for one ETU period, at 10.3 to 10.7 ETU after the character START

bit falling edge. The transmitter is expected to sample the line for NACK (for low level) from 10.8 to 11.2 ETU after the character START bit falling edge.

Instead, the USART peripheral in Smartcard mode samples the transmit line for NACK from 10.3 to 10.7 ETU after the character START bit falling edge. This is unlikely to cause issues with receivers (smartcards) that respect the ISO/IEC 7816-3 standard. However, it may cause issues with respect to certification.

Workaround

None.

2.11.2 Break request preventing TC flag from being set

Description

After the end of transmission of data (D1), the transmission complete (TC) flag is not set when the following condition is met:

- CTS hardware flow control is enabled
- D1 transmission is in progress
- D1 transmission is in progress
- D1 transmission is in progress

As a consequence, an application relying on the TC flag fails to detect the end of data transfer.

Workaround

In the application, only allow break request after the TC flag is set.

2.11.3 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

Upon setting the UE and RE bits, configure the I/O used for nRTS into alternate function.

2.11.4 Receiver timeout counter wrong start in two-stop-bit configuration

Description

In two-stop-bit configuration, the receiver timeout counter starts counting from the end of the second stop bit of the last character instead of starting from the end of the first stop bit.

Workaround

In two-stop-bit configuration, the receiver timeout counter starts counting from the end of the second stop bit of the last character instead of starting from the end of the first stop bit.

2.11.5 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.

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.

Workaround

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

2.12 SPI

2.12.1 BSY bit may stay high when SPI is disabled

Description

The BSY flag may remain high upon disabling the SPI while operating in:

- master transmit mode and the TXE flag is low (data register full).
- master receive-only mode (simplex receive or half-duplex bidirectional receive phase) and an SCK strobing edge has not occurred since the transition of the RXNE flag from low to high.
- slave mode and NSS signal is removed during the communication.

Workaround

When the SPI operates in:

- master transmit mode, disable the SPI when TXE = 1 and BSY = 0.
- master receive-only mode, ignore the BSY flag.
- slave mode, do not remove the NSS signal during the communication.

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

Description

BSY flag may sporadically remain high at the end of a data transfer in slave mode. This occurs upon coincidence of internal CPU clock and external SCK clock provided by master.

In such an event, if the software only relies on BSY flag to detect the end of SPI slave data transaction (for example to enter low-power mode or to change data line direction in half-duplex bidirectional mode), the detection fails.

As a conclusion, the BSY flag is unreliable for detecting the end of data transactions.

Workaround

Depending on SPI operating mode, use the following means for detecting the end of transaction:

- When NSS hardware management is applied and NSS signal is provided by master, use NSS flag.
- In SPI receiving mode, use the corresponding RXNE event flag.
- In SPI transmit-only mode, use the BSY flag in conjunction with a timeout expiry event. Set the timeout such as to exceed the expected duration of the last data frame and start it upon TXE event that occurs with the second bit of the last data frame. The end of the transaction corresponds to either the BSY flag becoming low or the timeout expiry, whichever happens first.

Prefer one of the first two measures to the third as they are simpler and less constraining.

Alternatively, apply the following sequence to ensure reliable operation of the BSY flag in SPI transmit mode:

1. Write last data to data register.
2. Poll the TXE flag until it becomes high, which occurs with the second bit of the data frame transfer.
3. Disable SPI by clearing the SPE bit mandatorily before the end of the frame transfer.
4. Poll the BSY bit until it becomes low, which signals the end of transfer.

Note: The alternative method can only be used with relatively fast CPU speeds versus relatively slow SPI clocks or/and long last data frames. The faster is the software execution, the shorter can be the duration of the last data frame.

2.12.3 SPI CRC corruption upon DMA transaction completion by another peripheral

Description

When the following conditions are all met:

- CRC function for the SPI is enabled
- SPI transaction managed by software (as opposed to DMA) is ongoing and CRCNEXT flag set

- another peripheral using the DMA channel on which the SPI is mapped completes a DMA transfer,

the CRCNEXT bit is unexpectedly cleared and the SPI CRC calculation may be corrupted, setting the CRC error flag.

Workaround

Ensure that the DMA channel on which the SPI is mapped is not concurrently in use by another peripheral.

2.12.4 In I^2S slave mode, enabling I2S while WS is active causes desynchronization

Description

In I^2S slave mode, the WS signal level is used to start the communication. If the I2S peripheral is enabled while the WS line is active (low for I^2S protocol, high for LSB- or MSB-justified mode), and if the master is already sending the clock, the I2S peripheral (slave) starts communicating data from the instant of its enable, which causes desynchronization between the master and the slave throughout the whole communication.

Workaround

Enable I2S peripheral while the WS line is at:

- high level, for I^2S protocol.
- low level, for LSB- or MSB-justified mode.

2.12.5 CRC error in SPI slave mode if internal NSS changes before CRC transfer

Description

Some reference manual revisions may omit the information that the device operating as SPI slave must be configured in software NSS control if the SPI master pulses the NSS (for (for example in NSS pulse mode).

Otherwise, the transition of the internal NSS signal after the CRCNEXT flag is set might result in wrong CRC value computed by the device and, as a consequence, in a CRC error. As a consequence, the NSS pulse mode cannot be used along with the CRC function.

This is a documentation error rather than a product limitation.

Workaround

No application workaround is required as long as the device operating as SPI slave is duly configured in software NSS control.

Revision history

Table 5. Document revision history

Date	Version	Changes
21-Mar-2014	1	Initial release.
09-Apr-2014	2	Removed part numbers with 16KByte Flash size
01-Oct-2014	3	Added: • note 2 in table *Device identification* • section *Wakeup sequence from Standby mode when using more than one wakeup source* • section *Load multiple not supported by ADC interface* • section *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'* Removed the package marking information.
21-Nov-2014	4	Added section *ADEN bit cannot be set immediately after the ADC calibration is done*
19-Feb-2015	5	Added: • section *Start bit detected too soon when sampling for NACK signal from the SmartCard* • section *A break request can prevent the Transmission Complete flag (TC) from being set* • section *nRTS is active while RE = 0 or UE = 0* • section *VREFINT scaler startup time from power down parameter degradation*
14-Sep-2015	6	Added: • section *Full JTAG configuration without NJTRST pin cannot be used* • section *BSY bit may stay high at the end of a SPI data transfer in slave mode* • section *Receiver timeout counter starting in case of a 2 stop bit configuration* • section *Spurious Bus Error detection in master mode* Modified section *Wrong behavior related with MCU Stop mode when wakeup from Stop mode by I2C peripheral is disabled*.
03-May-2016	7	Added: • section Overrun flag might not be set when converted data have not been read before new data are written • section ADC differential mode Common mode input range Modified: • section Load multiple not supported by ADC interface • table *Summary of silicon limitations*

Date	Version	Changes
28-Feb-2019	8	Added: • Section 2.1.3 Store immediate overlapping exception return operation might vector to incorrect interrupt • Section 2.4.7 Inaccurate VREFINT calibration values • Section 2.6.1 OPAMP timer-controlled multiplexer mode not working when OPAMP is used in PGA or follower mode • Section 2.7.1 Inhibited acquisition in short transfer phase configuration • Section 2.8.1 PWM re-enabled in automatic output enable mode despite of system break • Section 2.8.2 One-pulse mode trigger not detected in master-slave reset + trigger configuration • Section 2.9.1 Spurious tamper detection when disabling the tamper channel • Section 2.9.2 RTC calendar registers are not locked properly • Section 2.9.3 RTC interrupt can be masked by another RTC interrupt • Section 2.9.4 Calendar initialization may fail in case of consecutive INIT mode entry • Section 2.9.5 Alarm flag may be repeatedly set when the core is stopped in debug • Section 2.10.5 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave • Section 2.10.9 Last-received byte loss in reload mode • Section 2.10.10 Spurious master transfer upon own slave address match • Section 2.12.1 BSY bit may stay high when SPI is disabled • Section 2.12.5 CRC error in SPI slave mode if internal NSS changes before CRC transfer Modified: • Section 2.10.6 Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C and Section 2.11.4 Receiver timeout counter wrong start in two-stop-bit configuration references moved to the table of documentation errata • Text rewrite and/or title modification in some errata of I2C, USART, and SPI sections

Contents

1 **Summary of device errata** 2

2 **Description of device errata** 4

2.1 Core 4

2.1.1 Interrupted loads to SP can cause erroneous behavior 4

2.1.2 VDIV or VSQRT instructions might not complete correctly when very short ISRs are used . 4

2.1.3 Store immediate overlapping exception return operation might vector to incorrect interrupt 5

2.2 System 6

2.2.1 Wakeup sequence from Standby mode when using more than one wakeup source 6

2.2.2 Full JTAG configuration without NJTRST pin cannot be used 6

2.3 GPIO 7

2.3.1 GPIOx locking mechanism is not working properly for GPIOx_OTYPE register 7

2.4 ADC 7

2.4.1 Injected queue of context is not available in case of JQM = 0 7

2.4.2 Sampling time shortened in JAUTO auto delayed mode 7

2.4.3 Load multiple not supported by ADC interface 7

2.4.4 ADEN bit cannot be set immediately after the ADC calibration is done 7

2.4.5 Overrun flag might not be set when converted data have not been read before new data are written 8

2.4.6 ADC differential mode Common mode input range 8

2.4.7 Inaccurate VREFINT calibration values 8

2.5 COMP 8

2.5.1 VREFINT scaler startup time from power down parameter degradation 8

2.6 OPAMP 9

2.6.1 OPAMP timer-controlled multiplexer mode not working when OPAMP is used in PGA or follower mode 9

2.7 TSC 9

2.7.1 Inhibited acquisition in short transfer phase configuration 9

2.8 TIM 9

2.8.1 PWM re-enabled in automatic output enable mode despite of system break 9

2.8.2 One-pulse mode trigger not detected in master-slave reset + trigger configuration 9

2.9 RTC and TAMP 10

2.9.1 Spurious tamper detection when disabling the tamper channel . . . 10
2.9.2 RTC calendar registers are not locked properly . . . 10
2.9.3 RTC interrupt can be masked by another RTC interrupt . . . 10
2.9.4 Calendar initialization may fail in case of consecutive INIT mode entry . . . 11
2.9.5 Alarm flag may be repeatedly set when the core is stopped in debug . . . 12
2.10 I2C . . . 12
2.10.1 10-bit slave mode: wrong direction bit value upon Read header receipt . . . 12
2.10.2 10-bit combined with 7-bit slave mode: ADDCODE may indicate wrong slave address detection . . . 13
2.10.3 Wakeup frames may not wake up the MCU when Stop mode entry follows I2C enabling . 13
2.10.4 Wakeup frame may not wake up the MCU from Stop mode if t close to I2C kernel clock startup time . . . 13
2.10.5 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave . . . 14
2.10.6 Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C . . . 14
2.10.7 Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period . . . 14
2.10.8 Spurious bus error detection in master mode . . . 15
2.10.9 Last-received byte loss in reload mode . . . 15
2.10.10 Spurious master transfer upon own slave address match . . . 16
2.11 USART . . . 16
2.11.1 Non-compliant sampling for NACK signal from smartcard . . . 16
2.11.2 Break request preventing TC flag from being set . . . 17
2.11.3 nRTS is active while RE = 0 or UE = 0 . . . 17
2.11.4 Receiver timeout counter wrong start in two-stop-bit configuration . . . 17
2.11.5 When PCLK is selected as clock source for USART1, PCLK1 is used instead of PCLK2 . 17
2.12 SPI . . . 18
2.12.1 BSY bit may stay high when SPI is disabled . . . 18
2.12.2 BSY bit may stay high at the end of data transfer in slave mode . . . 18
2.12.3 SPI CRC corruption upon DMA transaction completion by another peripheral . . . 18
2.12.4 In I^2S slave mode, enabling I2S while WS is active causes desynchronization . . . 19
2.12.5 CRC error in SPI slave mode if internal NSS changes before CRC transfer . . . 19
Revision history . . . 20