STM32F101xC/D/E STM32F103xC/D/E

Errata sheet

STM32F101xC/D/E, STM32F103xC/D/E device errata

Applicability

This document applies to the part numbers of STM32F101xC/D/E, STM32F103xC/D/E 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 RMxxxx.

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
STM32F101xC/D/E	STM32F101RC, STM32F101RD, STM32F101RE, STM32F101VC, STM32F101VD, STM32F101VE, STM32F101ZC, STM32F101ZD, STM32F101ZE
STM32F103xC/D/E	STM32F103RC, STM32F103RD, STM32F103RE, STM32F103VC, STM32F103VD, STM32F103VE, STM32F103ZC, STM32F103ZD, STM32F103ZE

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32F101xC/D/E and STM32F103xC/D/E	Z	0x1001
	1	0x1003
	Y	0x1003
	2	0x1003
	3	0x1003
	X	0x1003

1. *Refer to the device datasheet 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 STM32F101xC/D/E, STM32F103xC/D/E 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	Rev. 1	Rev. Y	Rev. 2	Rev. 3	Rev. X
Core	2.1.1	Cortex-M3 LDRD with base in list may result in incorrect base register when interrupted or faulted	A	A	A	A	A	A
	2.1.2	Cortex-M3 event register is not set by interrupts and debug	A	A	A	A	A	A
	2.1.3	Interrupted loads to SP can cause erroneous behavior	A	A	A	A	A	A
	2.1.4	SVC and BusFault/MemManage may occur out of order	A	A	A	A	A	A
	2.1.5	Arm Cortex-M3 BKPT in debug monitor mode can cause DFSR mismatch	A	A	A	A	A	A
	2.1.6	Arm Cortex-M3 may freeze for SLEEPONEXIT single instruction ISR	A	A	A	A	A	A
System	2.2.1	Flash memory read after WFI/WFE instruction	A	A	A	A	A	A
	2.2.2	Debug registers cannot be read by user software	N	N	N	N	N	N
	2.2.3	Debugging Stop mode and SysTick timer	A	A	A	A	A	A
	2.2.4	Debugging Stop mode with WFE entry	A	A	A	A	A	A
	2.2.5	Wakeup sequence from Standby mode when using more than one wakeup source	A	A	A	A	A	A
	2.2.6	LSE startup in harsh environments	A	A	A	A	A	A
	2.2.7	RDP protection	N	N	N	N	N	N
	2.2.8	PVD and USB wakeup events	A	A	A	A	A	A
	2.2.9	Boundary scan TAP: wrong pattern sent out after the "capture IR" state	A	A	A	A	A	A
	2.2.10	Flash memory BSY bit delay versus STRT bit setting	A	A	A	A	A	A
	2.2.11	PC3 I/O pin not bonded in WCLSP64 package	A	A	A	A	A	A
	2.2.12	LSI clock stabilization time	A	A	A	A	A	A
GPIO	2.3.1	USART1_RTS and CAN_TX	A	A	A	A	A	A
	2.3.2	SPI1 in slave mode and USART2 in synchronous mode	N	N	N	N	N	N
	2.3.3	SPI1 in master mode and USART2 in synchronous mode	A	A	A	A	A	A
	2.3.4	SPI2 in slave mode and USART3 in synchronous mode	N	N	N	N	N	N
	2.3.5	SPI2 in master mode and USART3 in synchronous mode	A	A	A	A	A	A
	2.3.6	SDIO with TIM8	A	A	A	A	A	A
	2.3.7	SDIO and TIM3_REMAP	A	A	A	A	A	A
	2.3.8	SDIO with USART3 remapped and UART4	A	A	A	A	A	A

Function	Section	Limitation	Status					
			Rev. Z	Rev. 1	Rev. Y	Rev. 2	Rev. 3	Rev. X
GPIO	2.3.9	FSMC with I2C1 and TIM4_CH2	A	A	A	A	A	A
	2.3.10	FSMC with USART2 remapped	A	A	A	A	A	A
	2.3.11	FSMC with USART3 and TIM1 remapped	A	A	A	A	A	A
	2.3.12	I2S2 in master/slave mode and USART3 in synchronous mode	A	A	A	A	A	A
	2.3.13	USARTx_TX pin usage	A	A	A	A	A	A
FSMC	2.5.1	Multimaster access on the FSMC memory map	A	A	A	A	A	A
	2.5.2	Dummy read cycles inserted when reading synchronous memories	N	N	N	N	N	N
	2.5.3	1 dummy clock cycle inserted when writing to synchronous memories when CLKDIV=1	N	N	N	N	N	N
DMA	2.4.1	DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear	A	A	A	A	A	A
ADC	2.6.1	Voltage glitch on ADC input 0	N	N	N	N	N	N
TIM	2.7.1	PWM re-enabled in automatic output enable mode despite of system break	P	P	P	P	P	P
	2.7.3	Consecutive compare event missed in specific conditions	N	N	N	N	N	N
	2.7.4	Output compare clear not working with external counter reset	P	P	P	P	P	P
	2.7.5	Missing capture flag	A	A	A	A	A	A
	2.7.6	Overcapture detected too early	N	N	N	N	N	N
	2.7.7	General-purpose timer regulation for 100% PWM	N	N	N	N	N	N
IWDG	2.8.1	RVU flag not cleared at low APB clock frequency	A	A	A	A	A	A
	2.8.2	PVU flag not cleared at low APB clock frequency	A	A	A	A	A	A
	2.8.3	RVU and PVU flags are not cleared in Stop mode	A	A	A	A	A	A
I2C	2.9.1	Some software events must be managed before the current byte is being transferred	A	A	A	A	A	A
	2.9.2	Wrong data read into data register	A	A	A	A	A	A
	2.9.3	SMBus standard not fully supported	A	A	A	A	A	A
	2.9.4	Wrong behavior of I2C peripheral in master mode after a misplaced Stop	A	A	A	A	A	A
	2.9.5	Mismatch on the “Setup time for a repeated Start condition” timing parameter	A	A	A	A	A	A
	2.9.6	Data valid time (tVD;DAT) violated without the OVR flag being set	A	A	A	A	A	A
	2.9.7	I2C analog filter may provide wrong value, locking BUSY flag and preventing master mode entry	A	A	A	A	A	A
USART	2.10.1	Parity Error flag (PE) is set again after having been cleared by software	A	A	A	A	A	A
	2.10.2	Idle frame is not detected if receiver clock speed is deviated	N	N	N	N	N	N
	2.10.3	In full-duplex mode, the Parity Error (PE) flag can be cleared by writing the data register	A	A	A	A	A	A
	2.10.4	Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection	N	N	N	N	N	N

Function	Section	Limitation	Status					
			Rev. Z	Rev. 1	Rev. Y	Rev. 2	Rev. 3	Rev. X
USART	2.10.5	Break frame is transmitted regardless of nCTS input line status	N	N	N	N	N	N
	2.10.6	nRTS signal abnormally driven low after a protocol violation	A	A	A	A	A	A
SPI	2.11.1	SPI3 in I2S slave mode: timing sensitivity between I2S3_WS and I2S3_CK	N	N	N	N	N	N
	2.11.2	CRC still sensitive to communication clock when SPI is in slave mode even with NSS high	A	A	A	A	A	A
	2.11.3	SPI2/I2S2 slave mode wrong behavior in transmission and reception	A	-	-	-	-	-
	2.11.4	SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI is under DMA transaction close to the end of transfer or end of transfer -1	A	A	A	A	A	A
I2S	2.12.1	Wrong WS signal generation in 16-bit extended to 32-bit PCM long synchronisation mode	A	A	A	A	A	A
	2.12.2	In I2S slave mode, WS level must be set by the external master when enabling the I2S	A	A	A	A	A	A
	2.12.3	I2S slave mode desynchronisation with the master during communication	A	A	A	A	A	A
SDIO	2.13.1	Limited multibyte support with SDIO cards	A	A	A	A	A	A
	2.13.2	Data errors in SDIO hardware flow control mode	N	N	N	N	N	N
	2.13.3	Wrong CCRCFAIL status after a response without CRC is received	A	A	A	A	A	A
	2.13.4	Data corruption in SDIO clock dephasing (NEGEDGE) mode	N	N	N	N	N	N
	2.13.5	CE-ATA multiple write command and card busy signal management	A	A	A	A	A	A
	2.13.6	No underrun detection with wrong data transmission	A	A	A	A	A	A
bxCAN	2.14.1	bxCAN time-triggered communication mode not supported	N	N	N	N	N	N
USB	2.15.1	False wakeup detection for last K-state not terminated by EOP or reset before suspend	A	A	A	A	A	A
	2.15.2	ESOF interrupt timing desynchronized after resume signaling	A	A	A	A	A	A
	2.15.3	Incorrect CRC16 in the memory buffer	N	N	N	N	N	N
	2.15.4	USB packet buffer memory: over/underrun or COUNTn_RX[9:0] field reporting incorrect number if APB1 frequency is below 13 MHz	A	A	A	A	A	A

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

Table 4. Summary of device documentation errata

Function	Section	Documentation erratum
DMA	2.4.2	Byte and half-word accesses not supported
TIM	2.7.2	TRGO and TRGO2 trigger output failure

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

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

2.1.1 Cortex®-M3 LDRD with base in list may result in incorrect base register when interrupted or faulted

Description

This limitation is registered under Arm® ID number 602117 and classified into “Category 2”.

The Cortex®-M3 core has a limitation when executing an LDRD instruction from the system-bus area, with the base register in a list of the form LDRD Ra, Rb, [Ra, #imm]. The execution may not complete after loading the first destination register due to an interrupt before the second loading completes or due to the second loading getting a bus fault.

Workaround

1. This limitation does not impact the code execution when executing from the embedded flash memory, which is the standard use of the microcontroller.
2. Use the latest compiler releases. As of today, they no longer generate this particular sequence. Moreover, a scanning tool is provided to detect this sequence on previous releases (refer to your preferred compiler provider).

2.1.2 Cortex®-M3 event register is not set by interrupts and debug

Description

This limitation is registered under Arm® ID number 563915 and classified into “Category 2”.

When interrupts related to a WFE occur before the WFE is executed, the event register used for WFE wakeup events is not set and the event is missed. Therefore, when the WFE is executed, the core does not wake up from WFE if no other event or interrupt occur

Workaround

Use external events instead of interrupts to wake up the core from WFE by configuring an external or internal EXTI line in event mode.

2.1.3 Interrupted loads to SP can cause erroneous behavior

Description

This limitation is registered under Arm® ID number 752419 and classified into “Category 2”.

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 results in the load instruction being executed an additional time. For all instructions performing an update to 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 affected instructions are:

1. LDR SP,[Rn],#imm
2. LDR SP,[Rn,#imm]!

3. LDR SP,[Rn,#imm]
4. LDR SP,[Rn]
5. LDR SP,[Rn,Rm]

Workaround

As of today, there is no compiler generating these particular instructions. This limitation can only occur with hand-written assembly code.

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.

Example: the following instruction "LDR SP, [R0]" can be replaced by

“LDR R2,[R0]

MOV SP,R2 "

2.1.4 SVC and BusFault/MemManage may occur out of order

Description

This limitation is registered under Arm® ID number 740455 and classified into “Category 2”.

If an SVC exception is generated by executing the SVC instruction while the following instruction fetch is faulted, then the MemManage or BusFault handler may be entered even though the faulted instruction which followed the SVC should not have been executed.

Workaround

A workaround is only required if the SVC handler does not return to the return address that has been stacked for the SVC exception and the instruction access after the SVC faults. If this is the case then padding can be inserted between the SVC and the faulting area of code, for example, by inserting NOP instructions.

2.1.5 Arm® Cortex®-M3 BKPT in debug monitor mode can cause DFSR mismatch

Description

This limitation is registered under Arm® ID number 463763 and classified into “Category 3”.

A BKPT may be executed in debug monitor mode. This causes the debug monitor handler to be run. However, the bit 1 in the Debug fault status register (DFSR) at address 0xE000 ED30 is not set to indicate that it was originated by a BKPT instruction. This only occurs if an interrupt other than the debug monitor is already being processed just before the BKPT is executed.

Workaround

If the DFSR register does not have any bit set when the debug monitor is entered, this means that we must be in this “corner case” and so, that a BKPT instruction was executed in debug monitor mode.

2.1.6 Arm® Cortex®-M3 may freeze for SLEEPONEXIT single instruction ISR

Description

This limitation is registered under Arm® ID number 463764 and classified into “Category 3”.

If the Cortex®-M3 SLEEPONEXIT functionality is used and the concerned interrupt service routine (ISR) contains only a single instruction, the core becomes frozen. This freezing may occur if only one interrupt is active and it is preempted by an interrupt whose handler only contains a single instruction.

However, any new interrupt that causes a preemption would cause the core to become unfrozen and behave correctly again.

Workaround

This scenario does not happen in real application systems since all enabled ISRs should at least contain one instruction. Therefore, if an empty ISR is used, then insert an NOP or any other instruction before the exit instruction (BX or BLX).

2.2 System

2.2.1 Flash memory read after WFI/WFE instruction

Description

If a WFI/WFE instruction is executed during a flash memory access and the Sleep duration is very short (less than 2 clock cycles), the instruction fetch from the flash memory may be corrupted on the next wakeup event.

This issue occurs when the following conditions are met:

- Flash prefetch on
- Flash memory timing set to 2 wait states
- FLITF clock stopped in Sleep mode

Workaround

When using the flash memory with two wait states and prefetch on, the FLITF clock must not be stopped during the Sleep mode – the FLITFEN bit in the RCC_AHBENR register must be set (keep the reset value).

2.2.2 Debug registers cannot be read by user software

Description

The DBGMCU_IDCODE and DBGMCU_CR debug registers are accessible only in debug mode (not accessible by the user software). When these registers are read in user mode, the returned value is 0x00.

Workaround

None.

2.2.3 Debugging Stop mode and SysTick timer

Description

If the SysTick timer interrupt is enabled during the Stop mode debug (DBG_STOP bit set in the DBGMCU_CR register), it wakes up the system from Stop mode.

Workaround

To debug the Stop mode, disable the system tick timer interrupt.

2.2.4 Debugging Stop mode with WFE entry

Description

When the Stop debug mode is enabled (DBG_STOP bit set in the DBGMCU_CR register) this allows software debugging during Stop mode.

However, if the application software uses the WFE instruction to enter Stop mode, after wakeup some instructions could be missed if the WFE is followed by sequential instructions. This affects only Stop debug mode with WFE entry.

Workaround

To debug Stop mode with WFE entry, the WFE instruction must be inside a dedicated function with 1 instruction (NOP) between the execution of the WFE and the Bx LR.

Example 1 :

```
__asm void _WFE(void) {
WFE
NOP
BX lr }
```

2.2.5 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 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 could not be able to wake up from Standby mode.

Workaround

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

1. Disable all used wakeup sources.
2. Clear all related wakeup flags.
3. Re-enable all used wakeup sources.
4. Enter Standby mode.

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

2.2.6 LSE startup in harsh environments

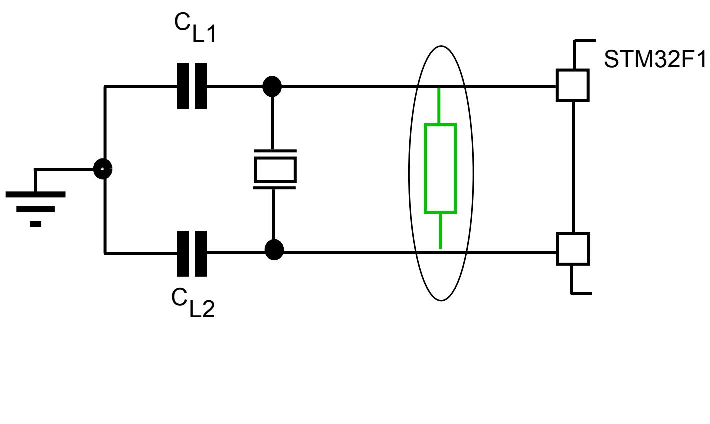
Description

The LSE (low-speed external) oscillator system has been designed to minimize the overall power consumption of the microcontroller. It is extremely important to take specific care in the design of the PCB to ensure this low power oscillator starts in harsh conditions. In some PCB designs without coating, an induced low leakage may prevent the LSE to startup, regardless of the 32.768 KHz crystal used. This phenomenon is amplified in humid environments that create frost on the OSC32_IN/OSC32_OUT tracks. This unwanted behavior may happen only at the first back-up domain power-on of the device.

Workaround

It is recommended to mount an additional parallel feedback resistor (from 16 MΩ to 22 MΩ) on board to help the oscillation startup in all cases (see Figure 1). For more details on compatible crystals and hardware techniques on PCB, refer to application note *Oscillator design guide for STM8AF/AL/S, STM32 MCUs and MPUs* (AN2867).

Figure 1. LSE startup using an additional resistor



2.2.7 RDP protection

Description

When the RDP protection is set, the debugger can still access the CPU program counter register and the NVIC registers as well. Remapping the table vector location at different address in the code memory map and triggering the interrupts may allow to retrieve a part of the flash memory code in CPU registers.

Workaround

None.

2.2.8 PVD and USB wakeup events

Description

PVD and USB Wakeup signals, which are internally linked to EXTI line16 and EXTI line18, respectively, cannot be used as event sources for the Cortex®-M3 core. As a consequence, these signals cannot be used to exit the Sleep or the Stop mode (exit WFE).

Workaround

Use interrupt sources and the WFI instruction if the application must be woken up from the Sleep or the Stop mode by PVD or USB Wakeup signals.

2.2.9 Boundary scan TAP: wrong pattern sent out after the "capture IR" state

Description

After the "capture IR" state of the boundary scan TAP, the two least significant bits in the instruction register should be loaded with 01 for them to be shifted out whenever a next instruction is shifted in. However, the boundary scan TAP shifts out the latest value loaded into the instruction register, which could be 00, 01, 10 or 11.

Workaround

The data shifted out, after the capture IR state, in the boundary scan flow should therefore be ignored and the software should check not only the two least significant bits (XXX01) but all register bits (XXXXX).

2.2.10 Flash memory BSY bit delay versus STRT bit setting

Description

When the STRT bit in the flash memory control register is set (to launch an erase operation), the BSY bit in the flash memory status register goes high one cycle later.

Therefore, if the FLASH_SR register is read immediately after the FLASH_CR register is written (STRT bit set), the BSY bit is read as 0.

Workaround

Read the BSY bit at least one cycle after setting the STRT bit.

2.2.11 PC3 I/O pin not bonded in WCLSP64 package

Description

In the WCLSP64 package, the PC3 I/O pin is not bonded. Since the default state of the I/O is input floating and if the PC3 is not fixed at low level by software, it results in an extra current consumption in low power mode.

Workaround

PC3 must be configured by software to output push-pull mode and writing 0 into the data register in order to avoid an extra consumption during low power modes.

2.2.12 LSI clock stabilization time

Description

When the LSIRDY flag is set, the clock may still be out of the specified frequency range (f_{LSI} parameter, see LSI oscillator characteristics in the product datasheet).

Workaround

To have a fully stabilized clock in the specified range, a software temporization of 100 µs should be added.

2.3 GPIO

2.3.1 USART1_RTS and CAN_TX

Description

When the following conditions are met:

- USART1 is clocked
- CAN is not clocked
- I/O port pin PA12 is configured as an alternate function output

even if CAN_TX is not used, this signal is set by default to 1 if I/O port pin PA12 is configured as an alternate function output

In this case USART1_RTS cannot be used.

Workaround

When USART1_RTS is used, the CAN must be remapped to either another IO configuration when the CAN is used, or to the unused configuration (CAN_REMAP[1:0] set to “01”) when the CAN is not used.

2.3.2 SPI1 in slave mode and USART2 in synchronous mode

Description

When the following conditions are met:

- SPI1 and USART2 are clocked
- I/O port pin PA4 is configured as an alternate function output

USART2 cannot be used in synchronous mode (USART2_CK signal) if SPI1 is used in slave mode.

Workaround

None.

2.3.3 SPI1 in master mode and USART2 in synchronous mode

Description

When the following conditions are met:

- SPI1 and USART2 are clocked
- I/O port pin PA4 is configured as an alternate function output

USART2 cannot be used in synchronous mode (USART2_CK signal) if SPI1 is used in master mode and SP1_NSS is configured in software mode. In this case USART2_CK is not output on the pin.

Workaround

In order to output USART2_CK, the SSOE bit in the SPI1_CR2 register must be set to configure the pin in output mode.

2.3.4 SPI2 in slave mode and USART3 in synchronous mode

Description

When the following conditions are met:

- SPI2 and USART3 are clocked
- I/O port pin PB12 is configured as an alternate function output

USART3 cannot be used in synchronous mode (USART3_CK signal) if SPI2 is used in slave mode.

Workaround

None.

2.3.5 SPI2 in master mode and USART3 in synchronous mode

Description

When the following conditions are met:

- SPI2 and USART3 are clocked
- I/O port pin PB12 is configured as an alternate function output

USART3 cannot be used in synchronous mode (USART3_CK signal) if SPI2 is used in master mode and SP2_NSS is configured in software mode. In this case USART3_CK is not output on the pin.

Workaround

In order to output USART3_CK, the SSOE bit in the SPI2_CR2 register must be set to configure the pin in output mode,

2.3.6 SDIO with TIM8

Description

Conflicts occur when:

- the SDIO is configured in 1- or 4-bit mode and TIM8_CH4 is configured as an output

The signals that conflict are the following:

- TIM8_CH4 and SDIO_D1

Workaround

Do not use TIM8_CH4 as an output when the SDIO is being used.

2.3.7 SDIO and TIM3_REMAP

Description

When SDIO is configured in 1- or 4-bit mode, and TIM3 channels are remapped to PC6 to PC9, and configured as outputs, a conflict occurs between:

- TIM3_CH4 and SDIO_D1

Workaround

Do not use TIM3_CH4 as an output when the SDIO is being used.

2.3.8 SDIO with USART3 remapped and UART4

Description

When SDIO is configured in 1-bit mode, there are conflicts with the USART3_TX pin remapped and with the UART4_TX pin. Conflicts are between the following signals:

- USART3_TX and SDIO_D2
- UART4_TX and SDIO_D2

Workaround

Use USART3_TX either in the default configuration (on the PB10 I/O) or remap USART3_TX to PD8 when the SDIO is being used.

Do not use UART4_TX when the SDIO is being used.

2.3.9 FSMC with I2C1 and TIM4_CH2

Description

When the FSMC is being used, the NADV signal is set to 1 by default when the alternate function output is selected for this pin. TIM4_CH2 and the I2C1 SDA signal are in conflict with the NADV signal.

Workaround

Do not use TIM4_CH2 as an output when the FSMC is being used.

Concerning I2C1, it is possible to use the remap functionality available on the PB8 and PB9 pins. Otherwise, disable the FSMC clock through the "RCC_AHBENR" register prior using the I2C1.

2.3.10 FSMC with USART2 remapped

Description

When the FSMC is being used and the alternate function output is selected on PD4, PD5 and PD7, the following signals are in conflict:

- USART2_RTS remapped to PD4 and FSMC_OEN
- USART2_TX remapped to PD5 and FSMC_WEN
- USART2_CK remapped to PD7 and FSMC_NE1/FSMC_NCE2.

Workaround

Use the USART2 default configuration (no remap).

2.3.11 FSMC with USART3 and TIM1 remapped

Description

When the FSMC is being used in 8-bit mode and the alternate function output is selected on the unused FSMC data lines 8 to 15, the following signals are in conflict:

- USART3_TX and USART3_CK remapped to PD8 and PD10, respectively, are in conflict with FSMC_D13 and FSMC_D15, respectively.
- TIM1_CH2, TIM1_CH3N, TIM1_CH3 and TIM1_CH4 used as outputs and remapped to PE11 to PE14, respectively, are in conflict with FSMC_D8 to FSMC_D11, respectively.

Workaround

- Use the USART3 partial remap functionality or default configuration (no remap).
- Do not use TIM1_CH2, TIM1_CH3N, TIM1_CH3 and TIM1_CH4 as outputs when they are remapped to PE11 to PE14, respectively.

2.3.12 I2S2 in master/slave mode and USART3 in synchronous mode

Description

If I2S2 was used prior to operating USART3 in synchronous mode, a conflict occurs between the I2S2_WS and USART3_CK signal even though the I2S2 clock was disabled.

Conditions:

- USART3 in synchronous mode is clocked
- I2S2 is not clocked
- I/O port pin PB12 is configured as an alternate function output

Workaround

To use USART3 in synchronous mode, first disable the I2S2 clock, then perform a software reset of SPI2(I2S2).

2.3.13 USARTx_TX pin usage

Description

In USART receive-mode-only communication (TE = 0 in the USARTx_CR1 register), even when the USARTx_TX pin is not being used, the corresponding I/O port pin cannot be used to output another alternate function (in this mode the USARTx_TX output is set to 1 and thus no other alternate function output can be used).

This limitation applies to all USARTx_TX pins that share another alternate function output.

Workaround

Do not use the corresponding I/O port of the USARTx_TX pin in alternate function output mode. Only the input mode can be used (TE bit in the USARTx_CR1 has to be cleared).

2.4 DMA

2.4.1 DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear

Description

Upon a data transfer error in a DMA channel x, both the specific TEIFx and the global GIFx flags are raised and the channel x is normally automatically disabled. However, if in the same clock cycle the software clears the GIFx flag (by setting the CGIFx bit of the DMA_IFCR register), the automatic channel disable fails and the TEIFx flag is not raised.

This issue does not occur with ST's HAL software that does not use and clear the GIFx flag when the channel is active.

Workaround

Do not clear GIFx flags when the channel is active. Instead, use HTIFx, TCIFx, and TEIFx specific event flags and their corresponding clear bits.

2.4.2 Byte and half-word accesses not supported

Description

Some reference manual revisions may wrongly state that the DMA registers are byte- and half-word-accessible. Instead, the DMA registers must always be accessed through aligned 32-bit words. Byte or half-word write accesses cause an erroneous behavior.

ST's low-level driver and HAL software only use aligned 32-bit accesses to the DMA registers.

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

Workaround

No application workaround is required.

2.5 FSMC

2.5.1 Multimaster access on the FSMC memory map

Description

When multimaster accesses are performed on the FSMC memory map (address space from 0x6000 0000 to 0xA000 0FFF), an error could be generated, leading to either a bus fault or a DMA transfer error. A multimaster can be:
- DMA1 and DMA2
- DMA1 and CPU
- DMA2 and CPU

Workaround

If multimaster accesses are required on the FSMC address space, the software must ensure that accesses are performed one at a time, and not at the same time.

2.5.2 Dummy read cycles inserted when reading synchronous memories

Description

When performing a burst read access to a synchronous memory, some 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. The number of dummy reads corresponds to the AHB data size.

Example: if AHB data size = 32 bit and MEMSIZE= 16 bit, two extra 16-bit reads is performed.

Workaround

None

2.5.3 1 dummy clock cycle inserted when writing to synchronous memories when CLKDIV=1

Description

When performing a write access to a synchronous memory and CLKDIV=1 (in FSMC_BTRx register), one dummy clock cycle is generated after nWE is de-assserted whatever the type of write burst access. However, there is no dummy write to the memory since the extra clock is generated while nWE is de-asserted.

Workaround

None

2.6 ADC

2.6.1 Voltage glitch on ADC input 0

Description

A low-amplitude voltage glitch may be generated (on ADC input 0) on the PA0 pin, when the ADC is converting with injection trigger. It is generated by internal coupling and synchronized to the beginning and the end of the injection sequence, whatever the channel(s) to be converted.

The glitch amplitude is less than 150 mV with a typical duration of 10 ns (measured with the I/O configured as high-impedance input and left unconnected). If PA0 is used as a digital output, this has no influence on the signal. If PA0 is used has a digital input, it is not detected as a spurious transition, providing that PA0 is driven with an impedance lower than 5 kΩ. This glitch does not have any influence on the remaining port A pin or on the ADC conversion injection results, in single ADC configuration.

When using the ADC in dual mode with injection trigger, and in order to avoid any side effect, it is advised to distribute the analog channels so that Channel 0 is configured as an injected channel.

Workaround

None.

2.7 TIM

2.7.1 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 (for example with PWM frequency deviation).

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

Workaround

Preferably, implement control loops 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).

2.7.2 TRGO and TRGO2 trigger output failure

Description

Some reference manual revisions may omit the following information.

The timers can be linked using ITRx inputs and TRGOx outputs. Additionally, the TRGOx outputs can be used as triggers for other peripherals (for example ADC). Since this circuitry is based on pulse generation, care must be taken when initializing master and slave peripherals or when using different master/slave clock frequencies:

- If the master timer generates a trigger output pulse on TRGOx prior to have the destination peripheral clock enabled, the triggering system may fail.
- If the frequency of the destination peripheral is modified on-the-fly (clock prescaler modification), the triggering system may fail.

As a conclusion, the clock of the slave timer or slave peripheral must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are being received from the master timer.

This is a documentation issue rather than a product limitation.

Workaround

No application workaround is required or applicable as long as the application handles the clock as indicated.

2.7.3 Consecutive compare event missed in specific conditions

Description

Every match of the counter (CNT) value with the compare register (CCR) value is expected to trigger a compare event. However, if such matches occur in two consecutive counter clock cycles (as consequence of the CCR value change between the two cycles), the second compare event is missed for the following CCR value changes:

- in edge-aligned mode, from ARR to 0:
  - first compare event: CNT = CCR = ARR
  - second (missed) compare event: CNT = CCR = 0
- in center-aligned mode while up-counting, from ARR-1 to ARR (possibly a new ARR value if the period is also changed) at the crest (that is, when TIMx_RCR = 0):
  - first compare event: CNT = CCR = (ARR-1)
  - second (missed) compare event: CNT = CCR = ARR
- in center-aligned mode while down-counting, from 1 to 0 at the valley (that is, when TIMx_RCR = 0):
  - first compare event: CNT = CCR = 1
  - second (missed) compare event: CNT = CCR = 0

This typically corresponds to an abrupt change of compare value aiming at creating a timer clock single-cycle-wide pulse in toggle mode.

As a consequence:

- In toggle mode, the output only toggles once per counter period (squared waveform), whereas it is expected to toggle twice within two consecutive counter cycles (and so exhibit a short pulse per counter period).
- In center mode, the compare interrupt flag does note rise and the interrupt is not generated.

Note: The timer output operates as expected in modes other than the toggle mode.

Workaround

None.

2.7.4 Output compare clear not working with external counter reset

Description

The output compare clear event (ocref_clr) is not correctly generated when the timer is configured in the following slave modes: Reset mode, Combined reset + trigger mode, and Combined gated + reset mode.

The PWM output remains inactive during one extra PWM cycle if the following sequence occurs:

1. The output is cleared by the ocref_clr event.
2. The timer reset occurs before the programmed compare event.

Workaround

Apply one of the following measures:

- Use BKIN (or BKIN2 if available) input for clearing the output, selecting the Automatic output enable mode (AOE = 1).
- Mask the timer reset during the PWM ON time to prevent it from occurring before the compare event (for example with a spare timer compare channel open-drain output connected with the reset signal, pulling the timer reset line down).

2.7.5 Missing capture flag

Description

In capture mode, when a capture occurs while the CCRx register is being read, the capture flag (CCxIF) may be cleared without the overcapture flag (CCxOF) being set. The new data are actually captured in the capture register.

Workaround

An external interrupt can be enabled on the capture I/O just before reading the capture register (in the capture interrupt), and disabled just after reading the captured data. Possibly, a missed capture is detected by the EXTI peripheral.

2.7.6 Overcapture detected too early

Description

In capture mode, the overcapture flag (CCxOF) can be set even though no data have been lost.

This issue occurs when the following conditions are met:

- The capture occurs while the capture register is being read, an overcapture is detected even though the previously captured data are correctly read and the new data are correctly stored into the capture register.
- The system is at the limit of an overcapture but no data are lost.

Workaround

None.

2.7.7 General-purpose timer regulation for 100% PWM

Description

When the OCREF_CLR functionality is activated, the OCxREF signal becomes de-asserted (and consequently OCx is deasserted / OCxN is asserted) when a high level is applied on the OCREF_CLR signal. The PWM then restarts (output re-enabled) at the next counter overflow.

But if the PWM is configured at 100% (CCxR > ARR), then it does not restart and OCxREF remains de-asserted.

Workaround

None.

2.7.8 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 (for example with PWM frequency deviation).

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

Workaround

Preferably, implement control loops 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).

2.7.9 TRGO and TRGO2 trigger output failure

Description

Some reference manual revisions may omit the following information.

The timers can be linked using ITRx inputs and TRGOx outputs. Additionally, the TRGOx outputs can be used as triggers for other peripherals (for example ADC). Since this circuitry is based on pulse generation, care must be taken when initializing master and slave peripherals or when using different master/slave clock frequencies:

- If the master timer generates a trigger output pulse on TRGOx prior to have the destination peripheral clock enabled, the triggering system may fail.
- If the frequency of the destination peripheral is modified on-the-fly (clock prescaler modification), the triggering system may fail.

As a conclusion, the clock of the slave timer or slave peripheral must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are being received from the master timer.

This is a documentation issue rather than a product limitation.

Workaround

No application workaround is required or applicable as long as the application handles the clock as indicated.

2.7.10 Consecutive compare event missed in specific conditions

Description

Every match of the counter (CNT) value with the compare register (CCR) value is expected to trigger a compare event. However, if such matches occur in two consecutive counter clock cycles (as consequence of the CCR value change between the two cycles), the second compare event is missed for the following CCR value changes:

- in edge-aligned mode, from ARR to 0:
  - first compare event: CNT = CCR = ARR
  - second (missed) compare event: CNT = CCR = 0
- in center-aligned mode while up-counting, from ARR-1 to ARR (possibly a new ARR value if the period is also changed) at the crest (that is, when TIMx_RCR = 0):
  - first compare event: CNT = CCR = (ARR-1)
  - second (missed) compare event: CNT = CCR = ARR
- in center-aligned mode while down-counting, from 1 to 0 at the valley (that is, when TIMx_RCR = 0):
  - first compare event: CNT = CCR = 1
  - second (missed) compare event: CNT = CCR = 0

This typically corresponds to an abrupt change of compare value aiming at creating a timer clock single-cycle-wide pulse in toggle mode.

As a consequence:

- In toggle mode, the output only toggles once per counter period (squared waveform), whereas it is expected to toggle twice within two consecutive counter cycles (and so exhibit a short pulse per counter period).
- In center mode, the compare interrupt flag does note rise and the interrupt is not generated.

Note: The timer output operates as expected in modes other than the toggle mode.

Workaround

None.

2.7.11 Output compare clear not working with external counter reset

Description

The output compare clear event (ocref_clr) is not correctly generated when the timer is configured in the following slave modes: Reset mode, Combined reset + trigger mode, and Combined gated + reset mode.

The PWM output remains inactive during one extra PWM cycle if the following sequence occurs:

1. The output is cleared by the ocref_clr event.
2. The timer reset occurs before the programmed compare event.

Workaround

Apply one of the following measures:

- Use BKIN (or BKIN2 if available) input for clearing the output, selecting the Automatic output enable mode (AOE = 1).
- Mask the timer reset during the PWM ON time to prevent it from occurring before the compare event (for example with a spare timer compare channel open-drain output connected with the reset signal, pulling the timer reset line down).

2.8 IWDG

2.8.1 RVU flag not cleared at low APB clock frequency

Description

Successful write to the IWDG_RLR register raises the RVU flag and prevents further write accesses to the register until the RVU flag is automatically cleared by hardware. However, at APB clock frequency lower than twice the IWDG clock frequency, the hardware never clears that flag, and writing to the IWDG_RLR register is no longer possible.

Workaround

Set the APB clock frequency higher than twice the IWDG clock frequency.

2.8.2 PVU flag not cleared at low APB clock frequency

Description

Successful write to the IWDG_PR register raises the PVU flag and prevents further write accesses to the register until the PVU flag is automatically cleared by hardware. However, at APB clock frequency lower than twice the IWDG clock frequency, the hardware never clears that flag, and writing to the IWDG_PR register is no longer possible.

Workaround

Set the APB clock frequency higher than twice the IWDG clock frequency.

2.8.3 RVU and PVU flags are not cleared in Stop mode

Description

The RVU and PVU flags in the IWDG_SR register are set by hardware after a write access to the IWDG_RLR or the IWDG_PR registers, respectively. If MCU enters Stop mode immediately after the write access, the RVU and PVU flags are not cleared by hardware. Consequently the next time the application attempts to write to the IWDG_RLR or the IWDG_PR registers, it waits in an infinite loop for the RVU and PVU flags to be cleared and the IWDG generates a reset after the programmed time-out period.

Workaround

The application has to wait until the RVU and PVU flags in the IWDG_SR register are cleared before entering Stop mode.

2.8.4 RVU flag not cleared at low APB clock frequency

Description

Successful write to the IWDG_RLR register raises the RVU flag and prevents further write accesses to the register until the RVU flag is automatically cleared by hardware. However, at APB clock frequency lower than twice the IWDG clock frequency, the hardware never clears that flag, and writing to the IWDG_RLR register is no longer possible.

Workaround

Set the APB clock frequency higher than twice the IWDG clock frequency.

2.8.5 PVU flag not cleared at low APB clock frequency

Description

Successful write to the IWDG_PR register raises the PVU flag and prevents further write accesses to the register until the PVU flag is automatically cleared by hardware. However, at APB clock frequency lower than twice the IWDG clock frequency, the hardware never clears that flag, and writing to the IWDG_PR register is no longer possible.

Workaround

Set the APB clock frequency higher than twice the IWDG clock frequency.

2.9 I2C

2.9.1 Some software events must be managed before the current byte is being transferred

Description

When the EV7, EV7_1, EV6_1, EV6_3, EV2, EV8, and EV3 events are not managed before
the current byte is being transferred, problems may be encountered such as receiving an extra byte, reading the same data twice or missing data.

Workaround

When it is not possible to manage the EV7, EV7_1, EV6_1, EV6_3, EV2, EV8, and EV3 events before the current byte transfer and before the acknowledge pulse when changing the ACK control bit, it is recommended to:

- **Workaround 1**
  Use the I2C with DMA in general, except when the Master is receiving a single byte.
- **Workaround 2**
  Use I2C interrupts and boost their priorities to the highest one in the application to make them uninterruptible
- **Workaround 3** (only for EV6_1 and EV6_3 events used in method 2)
  EV6_1 event (used in master receiver 2 bytes):
  Stretch SCL line between ADDR bit is cleared and ACK is cleared:
  1. ADDR=1
  2. Configure SCL I/O as GPIO open-drain output low
  3. Clear ADDR by reading SR1 register followed by reading SR3
  4. Program ACK=0
  5. Configure SCL I/O as Alternate Function open drain

  EV6_3 event (used in master receiver 1 byte):
  Stretch SCL line between ADDR bit is cleared and STOP bit programming:
  1. ADDR=1
  2. Program ACK=0
  3. Configure SCL I/O as GPIO open-drain output low
  4. Clear ADDR by reading SR1 register followed by reading SR3
  5. Program STOP=1
  6. Configure SCL I/O as Alternate Function open drain

2.9.2 Wrong data read into data register

Description

In Master Receiver mode, when closing the communication using method 2, the content of the last read data can be corrupted. The following two sequences are concerned by the limitation:

- **Sequence 1:** Transfer sequence for master receiver when N = 2:
  1. BTF = (Data N-1 in DR and Data N in shift register)
  2. Program STOP = 1,
  3. Read DR twice (Read Data N-1 and Data N) just after programming the STOP.
- **Sequence 2:** Transfer sequence for master receiver when N > 2:
  1. BTF = 1 (Data N-2 in DR and Data N-1 in shift register)
  2. Program ACK = 0,
  3. Read DataN-2 in DR.
  4. Program STOP = 1,
  5. Read DataN-1.

If the user software is not able to read the data N-1 before the STOP condition is generated on the bus, the content of the shift register (data N) is corrupted (data N is shifted 1-bit to the left).

Workaround

- **Workaround 1**

  Stretch the SCL line by configuring SCL I/O as a general purpose I/O, open-drain output low level, before the SET STOP in sequence 1 and before the READ Data N-2 in séquence 2. Then configure back the SCL I/O as alternate function open-drain after the READ Data N-1. The sequences become:

  Sequence 1:
  1. BTF = 1 (Data N-1 in DR and Data N in shift register)
  2. Configure SCL I/O as GPIO open-drain output low
  3. Program STOP = 1
  4. Read Data N-1
  5. Configure SCL I/O as Alternate Function open drain
  6. Read Data N

  Sequence 2:
  1. BTF = (Data N-2 in DR and Data N-1 in shift register)
  2. Program ACK = 0
  3. Configure SCL I/O as GPIO open-drain output low
  4. Read Data N-2 in DR.
  5. Program STOP = ,
  6. Read Data N-1.
  7. Configure SCL I/O as Alternate Function open drain
- **Workaround 2**

  Mask all active interrupts between the SET STOP and the READ data N-1 for sequence 1; and between the READ data N-2, the SET STOP and the READ data N-1 for Sequence 2.
- **Workaround 3**

  Manage I2C RxNE events with DMA or interrupts with the highest priority level, so that the condition BTF = 1 never occurs.

2.9.3 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.

Workaround

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

2.9.4 Wrong behavior of I2C peripheral in master mode after a misplaced Stop

Description

If a misplaced Stop is generated on the bus, the peripheral cannot enter master mode properly:

- If a void message is received (START condition immediately followed by a STOP): the BERR (bus error) flag is not set, and the I2C peripheral is not able to send a start condition on the bus after the write to the START bit in the I2C_CR2 register.
- In the other cases of a misplaced STOP, the BERR flag is set. If the START bit is already set in I2C_CR2, the START condition is not correctly generated on the bus and can create bus errors.

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.

In case of a noisy environment in which unwanted bus errors can occur, it is recommended to implement a timeout to ensure that after the START control bit is set, the SB (start bit) flag is set. In case the timeout has elapsed, the peripheral must be reset by setting the SWRST bit in the I2C_CR2 control register. It should also be reset in the same way if a BERR is detected while the START bit is set in I2C_CR2.

2.9.5 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 Tsu;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 issue can occur only in the following configuration:

- in Master mode
- in Standard mode at a frequency between 88 kHz and 100 kHz (no issue 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 issue 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.

2.9.6 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. Moreover, if the data register is written too late and close to the SCL rising edge, an error can be generated on the bus (SDA toggles while SCL is high). These violations cannot be detected because the OVR flag is not set (no transmit buffer underrun is detected).

This issue can occur only under the following conditions:

- In Slave transmit mode
- With clock stretching disabled (NOSTRETCH=1)
- If the software is late in writing the DR data register, but not late enough to set the OVR flag (the data register is written before the SCL rising edge).

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 writes to the data register fast enough after TXE or ADDR events. For instance, use an interrupt on the TXE or ADDR flag and boost its priority to the higher level, or use DMA. Use this "NOSTRETCH" mode with a slow I2C bus speed.

Note: The first data byte to transmit must be written in the data register after the ADDR flag is cleared, and before the next SCL rising edge, so that the time window for writing the first data byte in the data register is less than t_{LOW}.

If this is not possible, a workaround can be used:

1. *Clear the ADDR flag.*
2. *Wait for the OVR flag to be set.*
3. *Clear OVR and write the first data byte.*
4. *Then the time window for writing the next data byte is the time to transfer one byte. In this case, the master must discard the first received data byte.*

2.9.7 I2C analog filter may provide wrong value, locking BUSY flag and preventing master mode entry

Description

The I2C analog filters embedded in the I2C I/Os may be tied to low level, whereas SCL and SDA lines are kept at high level. This can occur after an MCU power-on reset, or during ESD stress. Consequently, the I2C BUSY flag is set, and the I2C cannot enter master mode (START condition cannot be sent). The I2C BUSY flag cannot be cleared by the SWRST control bit, nor by a peripheral or a system reset. BUSY bit is cleared under reset, but it is set high again as soon as the reset is released, because the analog filter output is still at low level. This issue occurs randomly.

Note: Under the same conditions, the I2C analog filters may also provide a high level, whereas SCL and SDA lines are kept to low level. This should not create issues as the filters output is correct after next SCL and SDA transition.

Workaround

The SCL and SDA analog filter output is updated after a transition occurs on the SCL and SDA line respectively. The SCL and SDA transition can be forced by software configuring the I2C I/Os in output mode. Then, once the analog filters are unlocked and output the SCL and SDA lines level, the BUSY flag can be reset with a software reset, and the I2C can enter master mode. Therefore, the following sequence must be applied:

1. Disable the I2C peripheral by clearing the PE bit in I2Cx_CR1 register.
2. Configure the SCL and SDA I/Os as General Purpose Output Open-Drain, High level (Write 1 to GPIOx_ODR).
3. Check SCL and SDA High level in GPIOx_IDR.
4. Configure the SDA I/O as General Purpose Output Open-Drain, Low level (Write 0 to GPIOx_ODR).
5. Check SDA Low level in GPIOx_IDR.
6. Configure the SCL I/O as General Purpose Output Open-Drain, Low level (Write 0 to GPIOx_ODR).
7. Check SCL Low level in GPIOx_IDR.
8. Configure the SCL I/O as General Purpose Output Open-Drain, High level (Write 1 to GPIOx_ODR).
9. Check SCL High level in GPIOx_IDR.
10. Configure the SDA I/O as General Purpose Output Open-Drain , High level (Write 1 to GPIOx_ODR).
11. Check SDA High level in GPIOx_IDR.
12. Configure the SCL and SDA I/Os as Alternate function Open-Drain.
13. Set SWRST bit in I2Cx_CR1 register.
14. Clear SWRST bit in I2Cx_CR1 register.
15. Enable the I2C peripheral by setting the PE bit in I2Cx_CR1 register.

2.10 USART

2.10.1 Parity Error flag (PE) is set again after having been cleared by software

Description

The parity error flag (PE) is set at the end of the last data bit. It should be cleared by software by making a read access to the status register followed by reading the data in the data register.

Once the PE flag is set by hardware, if it is cleared by software before the middle of the stop bit, it is set again. Consequently, the software may jump several times to the same interrupt routine for the same parity error.

Workaround

Before clearing the Parity Error flag, the software must wait for the RXNE flag to be set.

2.10.2 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.

2.10.3 In full-duplex mode, the Parity Error (PE) flag can be cleared by writing 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 in 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.

2.10.4 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.

2.10.5 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 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.

2.10.6 nRTS signal abnormally driven low after a protocol violation

Description

When RTS hardware flow control is enabled, the nRTS signal goes high when a data is received. If this data was not read and a 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 some data has been lost.

Workaround

- **Workaround 1:**
  After data reception and before reading the data in the data register, the software takes control of the nRTS pin using the GPIO registers and keeps it high as long as needed. If the application knows the USART is not ready and that further data received reception from the other device may be discarded, it keeps the nRTS pin at high level. It then releases the nRTS pin when the USART is ready to continue reception.
- **Workaround 2:** Ensure that the received data is always read in a time window less than the duration of the 2nd data reception. One solution would be to handle all data reception by DMA.

Note: These workarounds are needed only if the other UART device has violated the protocol. In most systems (no limitation on the other device), the USART works fine and no workaround is needed.

2.11 SPI

2.11.1 SPI3 in I2S slave mode: timing sensitivity between I2S3_WS and I2S3_CK

Description

When SPI3 is configured in I2S slave audio mode in I2S Philips or PCM modes, If the I2S3_WS signal arrives too early with respect to the active edge of I2S3_CK, a wrong communication starting too soon may result: then, depending on the clock polarity and the Audio mode selected, it is either shifted by one bit from start to end or, the first left and right data items are lost and the others, shifted.

Workaround

None. Use I2S3 in slave mode in the MSB/LSB justified mode only.

2.11.2 CRC still sensitive to communication clock when SPI is in slave mode even with NSS high

Description

When the SPI is configured in slave mode with the CRC feature enabled, the CRC is calculated even if the NSS pin deselects the SPI (high level applied on the NSS pin).

Workaround

The CRC has to be cleared on both Master and Slave sides between the slave deselection (high level on NSS) and the slave selection (low level on NSS), in order to resynchronize the Master and Slave for their respective CRC calculation.

To procedure to clear the CRC is the following:

1. disable the SPI (SPE = 0)
2. clear the CRCEN bit
3. set the CRCEN bit
4. enable the SPI (SPE = 1)

2.11.3 SPI2/I2S2 slave mode wrong behavior in transmission and reception

Description

When the SPI2/I2S2 slave mode is used:

- In transmit or in full-duplex mode, the TXE flag may go high two times before RXNE is set, that is, TXE may be asserted before the data already in the data register are transmitted. This may result in data loss during transmission.
- In receive or in full-duplex mode, with CRC calculation enabled, the CRC may be detected as wrong (the CRCERR flag is set) while the received data are correct.
- In transmit or in full-duplex mode, with CRC calculation enabled, the CRC may not be output on the data line after the CRCNEXT bit is set in CPU mode, or when the DMA in transmission reaches the end of transfer.

Workaround

- The transmit or full-duplex mode (CRC feature not used) may be used if the master stops the clock for every data transfer. The SPI2/I2S2 slave then has enough time to load the data register on the RXNE flag going high. DMA must not be used.
- In receive only mode, with CRC calculation enabled, at the end of the CRC reception, the software needs to check the CRCERR flag.
  – If it is found set, read back the SPI_RXCRC:
    ◦ If the value is 0, the complete data transfer is successful.
    ◦ Otherwise, one or more errors have been detected during the data transfer by CPU or DMA.
  – If it is found reset, the complete data transfer is successful.
- In full duplex or transmit only configurations, the CRC feature cannot be used.

2.11.4 SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI is under DMA transaction close to the end of transfer or end of transfer -1

Description

In the following conditions, the CRC may be frozen before the CRCNEXT bit is written, resulting in a CRC error:

- SPI is slave or master.
- Full duplex or simplex mode is used.
- CRC feature is enabled.
- SPI is configured to manage data transfers by software (interrupt or polling).
- A peripheral, mapped on the same DMA channel as the SPI, is executing DMA transfers.

Workaround

If the application allows it, you can use the DMA for SPI transfers.

2.12 I2S

2.12.1 Wrong WS signal generation in 16-bit extended to 32-bit PCM long synchronisation mode

Description

When I2S is master with PCM long synchronization is selected as16-bit data frame extended to 32-bit, the WS signal is generated every 16 bits rather than every 32 bits.

Workaround

Only the 16-bit mode with no data extension can be used when the I2S is master and when the selected mode has to be PCM long synchronization mode.

2.12.2 In I2S slave mode, WS level must be set by the external master when enabling the I2S

Description

In slave mode the WS signal level is used only to start the communication. If the I2S (in slave mode) is enabled while the master is already sending the clock and the WS signal level is low (for I2S protocol) or is high (for the LSB or MSB-justified mode), the slave starts communicating data immediately. In this case the master and slave is desynchronized throughout the whole communication.

Workaround

The I2S peripheral must be enabled when the external master sets the WS line at:
- High level when the I2S protocol is selected.
- Low level when the LSB or MSB-justified mode is selected.

2.12.3 I2S slave mode desynchronisation with the master during communication

Description

In I2S slave mode, if glitches on SCK or WS signals are generated at an unexpected time, a desynchronization of the master and the slave occurs. No error is reported to allow audio system to re-synchronize.

Workaround

The following workarounds can be applied in order to detect and react after a desynchronization by disabling and enabling I2S peripheral in order to resynchronize with the master.
1. Monitoring the I2S WS signal through an external interrupt to check the I2S WS signal status.
2. Monitoring the I2S clock signal through an input capture interrupt to check the I2S clock signal status.
3. Monitoring the I2S clock signal through an input capture interrupt and the I2S WS signal via an external interrupt to check the I2S clock and I2S WS signals status.

2.13 SDIO

2.13.1 Limited multibyte support with SDIO cards

Description

The SDIO standard allows multibyte transfers (to transfer any number of data bytes from 1, 2, 3 up to 512), which extends the block transfer of the SD standard.

In a multibyte transfer operation, CMD53 transfers only a number of bytes that is equal to a power of 2 (1, 2, 4, 8…512) between 1 and 512.

Workaround

In case the application needs to read a number of bytes N that is not a power of 2, and if the SDIO device supports the multiblock mode, it is then possible to read the N bytes using the CMD53 multiblock command:
- Configure the SDIO block size to 1 byte
- Read N bytes using the CMD53 multiblock command

2.13.2 Data errors in SDIO hardware flow control mode

Description

When HW Flow Control is enabled by setting bit 14 of SDIO_CLKCR register, some glitches can occur on the SDIO_CK output clock resulting in wrong data being written in the SD/MMC Card or the SDIO device. As consequence a data CRC error is reported to the SD/SDIO MMC host interface (DCRCFAIL bit in the SDIO_STA register is set).

Workaround

None. HW Flow control must not be used. Software has to manage Overrun Errors (Rx mode) and FIFO underrun (Tx mode).

2.13.3 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.

2.13.4 Data corruption in SDIO clock dephasing (NEGEDGE) mode

Description

When NEGEDGE bit is set to ‘1’, it may lead to invalid data and command response read.

Workaround

None. A configuration with the NEGEDGE bit equal to ‘1’ should not be used.

2.13.5 CE-ATA multiple write command and card busy signal management

Description

The CE-ATA card may inform the host that it is busy by driving the SDIO_D0 line low, two cycles after the transfer of a write command (RW_MULTIPLE_REGISTER or RW_MULTIPLE_BLOCK). When the card is in a busy state, the host must not send any data until the BUSY signal is de-asserted (SDIO_D0 released by the card).

This condition is not respected if the data state machine leaves the IDLE state (Write operation programmed and started, DTEN = 1, DTDIR = 0 in SDIO_DCTRL register and TXFIFOE = 0 in SDIO_STA register).

As a consequence, the write transfer fails and the data lines are corrupted.

Workaround

After sending the write command (RW_MULTIPLE_REGISTER or RW_MULTIPLE_BLOCK), the application must check that the card is not busy by polling the BSY bit of the ATA status register using the FAST_IO (CMD39) command before enabling the data state machine.

2.13.6 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 x period(PCLK2) + 3 x period(SDIOCLK)] ≥ (32 / (BusWidth)) x period(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.

2.14 bxCAN

2.14.1 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. The TTCM bit of the CAN_MCR register must be kept cleared (time-triggered communication mode disabled).

Workaround

None.

2.15 USB

2.15.1 False wakeup detection for last K-state not terminated by EOP or reset before suspend

Description

If a USB K-state is detected outside of a suspend state, the device generates *RESUME RECEIVED* signal and keeps it pending until:

- an EOP is detected on the USB, or
- a USB reset is detected on the USB, or
- a software reset is generated by setting the FRES bit.

It may happen that none of the above events occurs before the USB peripheral sets the suspend interrupt (after 3 ms idling), and the software decides to enter Suspend mode (by setting the FSUSP bit). The USB peripheral then immediately generates a wakeup interrupt (WKUP = 1) because it still detects the pending *RESUME RECEIVED* signal. This has a negative impact to the device power consumption.

A workaround is available for applications requiring to minimize the power consumption.

Workaround

Apply software reset prior to setting the FSUSP bit. This clears any spurious *RESUME RECEIVED* condition.

2.15.2 ESOF interrupt timing desynchronized after resume signaling

Description

Upon signaling resume, the device is expected to allow full 3 ms of time to the host or hub for sending the initial SOF (start of frame) packet, without triggering SUSP interrupt. However, the device only allows two full milliseconds and unduly triggers SUSP interrupt if it receives the initial packet within the third millisecond.

Workaround

When the device initiates resume (remote wakeup), mask the SUSP interrupt by setting the SUSPM bit for 3 ms, then unmask it by clearing SUSPM.

2.15.3 Incorrect CRC16 in the memory buffer

Description

Memory buffer locations are written starting from the address contained in the ADDRn_RX for a number of bytes corresponding to the received data packet length, CRC16 inclusive (that is, data payload length plus two bytes), or up to the last allocated memory location defined by BL_SIZE and NUM_BLOCK, whichever comes first. In the former case, the CRC16 checksum is written wrongly, with its least significant byte going to both memory buffer byte locations expected to receive the least and the most significant bytes of the checksum.

Although the checksum written in the memory buffer is wrong, the underlying CRC checking mechanism in the USB peripheral is fully functional.

Workaround

Ignore the CRC16 data in the memory buffer.

2.15.4 USB packet buffer memory: over/underrun or COUNTn_RX[9:0] field reporting incorrect number if APB1 frequency is below 13 MHz

Description

The USB peripheral's packet buffer memory is expected to operate at a minimum APB1 frequency of 8 MHz.

It may however happen that, when OUT transactions are sent by the Host with a data payload size exactly equal to the maximum packet size already programmed in the COUNTn_RX packet buffer memory (via the BLSIZE and NUM_BLOCK[4:0] fields), the packet and all bytes from the Host are correctly received and stored into the packet buffer memory, but, the COUNTn_RX[9:0] field indicates an incorrect number (one byte less).

Workaround

This limitation concerns applications that check the exact number of bytes received in the packet buffer memory. In order to avoid that these applications interpret a Host error and so, stall the OUT endpoint even if no data reception error actually occurred, it is recommended to:

1. increase the APB1 frequency to a minimum of 13 MHz, or
2. increase the APB1 frequency to a minimum of 10 MHz. Then program USB_COUNTn_RX (via the BLSIZE and NUM_BLOCK[4:0] fields) to have more than the number of bytes in the maximum packet size allocated for reception in the packet buffer memory

Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on www.st.com for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

Revision history

Table 5. **Document revision history**

Date	Revision	Changes
23-May-2008	1	Initial release.
21-Jul-2008	2	Section 2.10: PVD and USB wakeup events added.
29-Sep-2008	3	Figure 7: LQFP64 top package view on page 46 corrected.
11-Feb-2009	4	Section 1: Arm® 32-bit Cortex®-M3 limitations specified (Table 3: Cortex-M3 core limitations and impact on microcontroller behavior added limitations described). Limitation added: SPI3 in I2S slave mode: timing sensitivity between I2S3_WS and I2S3_CK on page 22. Added limitations: • Boundary scan TAP: wrong pattern sent out after the “capture IR” state • Flash memory BSY bit delay versus STRT bit setting • I2C peripheral • Timers • LSI clock stabilization time Section 2.21.1: Multimaster access on the FSMC memory map Section 2.22.1: Limited multibyte support with SDIO cards Table 4: Summary of silicon limitations on page 11 added.
22-Jun-2009	5	Introduction text updated in Section 2.9: Alternate functions. Section 2.9.10: FSMC with USART2 remapped updated. Section 2.9.11: FSMC with USART3 and TIM1 remapped added. Section 2.22.1: Limited multibyte support with SDIO cards clarified. Section 2.23.1: USB packet buffer memory: over/underrun or COUNTn_RX[9:0] field reporting incorrect number if APB1 frequency is below 13 MHz added. Figure 4: WLCSP64 top package view added.
21-Jul-2009	6	Section 2.9.12: I2S2 in master/slave mode and USART3 in synchronous mode added.
11-Jan-2010	7	Workaround modified in Section 2.11: SPI3 in I2S slave mode: timing sensitivity between I2S3_WS and I2S3_CK. Added limitations: • Section 2.9.13: USARTx_TX pin usage • Section 2.15.4: Wrong behavior of I2C peripheral in master mode after a misplaced Stop • Section 2.15.5: Mismatch on the “Setup time for a repeated Start condition” timing parameter • Section 2.15.6: Data valid time ($t_{VD;DAT}$) violated without the OVR flag being set • Section 2.16.1: CRC still sensitive to communication clock when SPI is in slave mode even with NSS high • Section 2.23.1: USB packet buffer memory: over/underrun or COUNTn_RX[9:0] field reporting incorrect number if APB1 frequency is below 13 MHz Date code added to Figure 2 to Figure 7.
17-Jun-2010	8	Updated for rev Y silicon Updated Table 4: Summary of silicon limitations with fix status. Added Section 2.4: Debugging Stop mode and system tick timer Added Section 2.5: Debugging Stop mode with WFE entry

Date	Revision	Changes
		Added Section 2.21: FSMC limitations Added Section 2.15.2: Wrong data read into data register Updated Section 2.15.4: Wrong behavior of I2C peripheral in master mode after a misplaced Stop Modified section Section 2.16.2: SPI2/I2S2 slave mode wrong behavior in transmission and reception Added Section 2.18: USART peripheral Updated Section 2.15.6: Data valid time ($t_{VD;DAT}$) violated without the OVR flag being set
21-Jan-2011	9	Updated Section 2.9.9: FSMC with I2C1 and TIM4_CH2 Added Section 2.17: I2S peripheral Added Section 2.18.6: nRTS signal abnormally driven low after a protocol violation
07-Feb-2011	10	Updated workarounds in Section 2.15.1: Some software events must be managed before the current byte is being transferred and Section 2.15.2: Wrong data read into data register
20-Jun-2011	11	Added reference to revision code 1 in Table 1 Added Section 1.1.5: Interrupted loads to SP can cause erroneous behavior Section 1.1.6: SVC and BusFault/MemManage may occur out of order Section 2.22.2: Data errors in SDIO hardware flow control mode
07-Oct-2013	12	Added: • Section 2.6: Wakeup sequence from Standby mode when using more than one wakeup source • Section 2.7: LSE start-up in harsh environments • Section 2.15.7: I2C analog filter may provide wrong value, locking BUSY flag and preventing master mode entry • Section 2.16.3: SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI is under DMA transaction close to the end of transfer or end of transfer -1 • Section 2.22.3: Wrong CCRCFAIL status after a response without CRC is received • Section 2.22.4: Data corruption in SDIO clock dephasing (NEGEDGE) mode • Section 2.22.5: CE-ATA multiple write command and card busy signal management • Section 2.22.6: No underrun detection with wrong data transmission Updated: • Section : Silicon identification • Table 1: Device Identification, “Revision code marked on device” column. • Table 2: Device summary • Table 4: Summary of silicon limitations.
10-Apr-2014	13	Added: • Section 2.14: PC3 I/O pin not bonded in WCLSP64 package • Added new limitation inside Table 4: Summary of silicon limitations
26-Nov-2015	14	Updated: • Silicon identification • Table 1: Device Identification
17-Dec-2018	15	Updated: • Table 4: Summary of silicon limitations
09-Apr-2020	16	Updated Section 1: Arm® 32-bit Cortex®-M3 limitations. Added Section 2.8: RDP protection.

Date	Revision	Changes
		Updated Table 4: Summary of silicon limitations. Updated Figure 7: LQFP64 top package view. Minor text edits across the whole document.
28-Jun-2022	17	Added: • Section 2.4.1 DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear • Section 2.4.1 DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear • Section 2.4.2 Byte and half-word accesses not supported • Section 2.7.1 PWM re-enabled in automatic output enable mode despite of system breakSection 2.7.2 TRGO and TRGO2 trigger output failure • Section 2.7.3 Consecutive compare event missed in specific conditions • Section 2.7.4 Output compare clear not working with external counter reset • Section 2.8.1 RVU flag not cleared at low APB clock frequency • Section 2.8.2 PVU flag not cleared at low APB clock frequency • Section 2.8.3 RVU and PVU flags are not cleared in Stop mode • Section 2.14.1 bxCAN time-triggered communication mode not supported • Section 2.15.1 False wakeup detection for last K-state not terminated by EOP or reset before suspend • Section 2.15.2 ESOF interrupt timing desynchronized after resume signaling • Section 2.15.3 Incorrect CRC16 in the memory bufferSection 2.15.4 USB packet buffer memory: over/underrun or COUNTn_RX[9:0] field reporting incorrect number if APB1 frequency is below 13 MHz • Table 4. Summary of device documentation errata Updated: • Section 2.2.3 Debugging Stop mode and SysTick timer

Contents