Errata sheet

STM32F105xx, STM32F107xx device errata

Applicability

This document applies to the part numbers of STM32F105xx, STM32F107xx 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 RM0008.

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
STM32F105xx	STM32F105R8, STM32F105V8, STM32F105RB, STM32F105VB, STM32F105RC, STM32F105VC
STM32F107xx	STM32F107RB, STM32F107VB, STM32F107RC, STM32F107VC

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32F105xx	Z	0x1001
	2	0x1001
STM32F107xx	Z	0x1001
	2	0x1001

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 STM32F105xx, STM32F107xx device limitations and their status:

A = limitation present, workaround available

N = limitation present, no workaround available

P = limitation present, partial workaround available

"-" = limitation absent

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. 2	Rev. Z
Core	2.1.1	Cortex-M3 LDRD with base in list may result in incorrect base register when interrupted or faulted	A	A
	2.1.2	Cortex-M3 event register is not set by interrupts and debug	A	A
	2.1.3	Interrupted loads to SP can cause erroneous behavior	A	A
	2.1.4	SVC and BusFault/MemManage may occur out of order	A	A
	2.1.5	Arm Cortex-M3 BKPT in debug monitor mode can cause DFSR mismatch	A	A
	2.1.6	Arm Cortex-M3 may freeze for SLEEPONEXIT single instruction ISR	A	A
System	2.2.1	Flash memory read after WFI/WFE instruction	A	A
	2.2.2	Debugging Stop mode and SysTick timer	A	A
	2.2.3	Debugging Stop mode with WFE entry	A	A
	2.2.4	Wakeup sequence from Standby mode when using more than one wakeup source	A	A
	2.2.5	LSE startup in harsh environments	A	A
	2.2.6	RDP protection	N	N
	2.2.7	Boundary scan TAP: wrong pattern sent out after the "capture IR" state	A	A
	2.2.8	Flash memory BSY bit delay versus STRT bit setting	A	A
	2.2.9	LSI clock stabilization time	A	A
	2.2.10	PLL not locking when sourced by HSI/2 after reset if it was previously sourced by HSE with predivider > 1 or PLL2	A	A
GPIO	2.3.1	SPI1 in slave mode and USART2 in synchronous mode	N	N
	2.3.2	SPI1 in master mode and USART2 in synchronous mode	A	A
	2.3.3	SPI2 in slave mode and USART3 in synchronous mode	N	N
	2.3.4	SPI2 in master mode and USART3 in synchronous mode	A	A
	2.3.5	I2S2 in master/slave mode and USART3 in synchronous mode	A	A
	2.3.6	USARTx_TX pin usage	A	A
DMA	2.4.1	DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear	A	A
ADC	2.5.1	Voltage glitch on ADC input 0	N	N
TIM	2.6.1	PWM re-enabled in automatic output enable mode despite of system break	P	P

Function	Section	Limitation	Status	
			Rev. 2	Rev. Z
TIM	2.6.3	Consecutive compare event missed in specific conditions	N	N
	2.6.4	Output compare clear not working with external counter reset	P	P
	2.6.5	Missing capture flag	A	A
	2.6.6	Overcapture detected too early	N	N
	2.6.7	General-purpose timer regulation for 100% PWM	N	N
IWDG	2.7.1	RVU flag not cleared at low APB clock frequency	A	A
	2.7.2	PVU flag not cleared at low APB clock frequency	A	A
	2.7.5	RVU and PVU flags are not cleared in Stop mode	A	A
I2C	2.8.1	Some software events must be managed before the current byte is being transferred	A	A
	2.8.2	Wrong data read into data register	A	A
	2.8.3	SMBus standard not fully supported	A	A
	2.8.4	Wrong behavior of I2C peripheral in master mode after a misplaced Stop	A	A
	2.8.5	Mismatch on the “Setup time for a repeated Start condition” timing parameter	A	A
	2.8.6	Data valid time (tVD;DAT) violated without the OVR flag being set	A	A
	2.8.7	I2C analog filter may provide wrong value, locking BUSY flag and preventing master mode entry	A	A
USART	2.9.1	Parity Error flag (PE) is set again after having been cleared by software	A	A
	2.9.2	Idle frame is not detected if receiver clock speed is deviated	N	N
	2.9.3	In full-duplex mode, the Parity Error (PE) flag can be cleared by writing the data register	A	A
	2.9.4	Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection	N	N
	2.9.5	Break frame is transmitted regardless of nCTS input line status	N	N
	2.9.6	nRTS signal abnormally driven low after a protocol violation	A	A
SPI	2.10.1	CRC still sensitive to communication clock when SPI is in slave mode even with NSS high	A	A
	2.10.2	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
I2S	2.11.1	Wrong WS signal generation in 16-bit extended to 32-bit PCM long synchronisation mode	A	A
	2.11.2	In I2S slave mode, WS level must be set by the external master when enabling the I2S	A	A
	2.11.3	I2S slave mode desynchronisation with the master during communication	A	A
bxCAN	2.12.1	bxCAN time-triggered communication mode not supported	N	N
OTG_FS	2.13.1	Data in RxFIFO are overwritten when all channels are disabled simultaneously	A	A
	2.13.2	OTG host blocks the receive channel when receiving IN packets and no Tx FIFO is configured	A	A
	2.13.3	Host channel-halted interrupt not generated when the channel is disabled	A	A
	2.13.4	Error in software-read OTG_FS_DCFG register values	A	A
	2.13.5	Minimum AHB frequency to guarantee correct operation of USB OTG FS peripheral	N	N

Function	Section	Limitation	Status	
			Rev. 2	Rev. Z
ETH	2.14.1	Possible underflow when TxFIFO is configured in Store-and-Forward mode and a relatively large frame is aborted in Half-duplex mode	A	A
	2.14.2	Posible CRC error when TxFIFO is configured in Store-and-Forward mode and a relatively large frame is aborted in Half-duplex mode with transmit checksum offload enabled	A	A
	2.14.3	Erroneous automatic checksum insertion after TxFIFO is dynamically switched from Threshold to Store-and-Forward mode	A	A
	2.14.4	Erroneous automatic checksum insertion after a large frame (longer than the TxFIFO) transmission	A	A
	2.14.5	In half-duplex mode, the MAC transmitter ignores collisions after it is disabled during a frame transmission	A	A
	2.14.6	Interrupt due to an RMON (MMC) counter may be set again after it is cleared	N	N
	2.14.7	Incorrect layer 3 (L3) checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads	A	A
	2.14.8	The Ethernet MAC processes invalid extension headers in the received IPv6 frames	N	N
	2.14.9	MAC stuck in the Idle state on receiving the TxFIFO flush command exactly one clock cycle after a transmission completes	A	A
	2.14.10	Transmit frame data corruption	A	A
	2.14.11	Ethernet DMA not working after WFI/WFE instruction	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	2.6.2	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 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.

2.2.3 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 }
```

2.2.4 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.

2.2.5 LSE startup in harsh environments

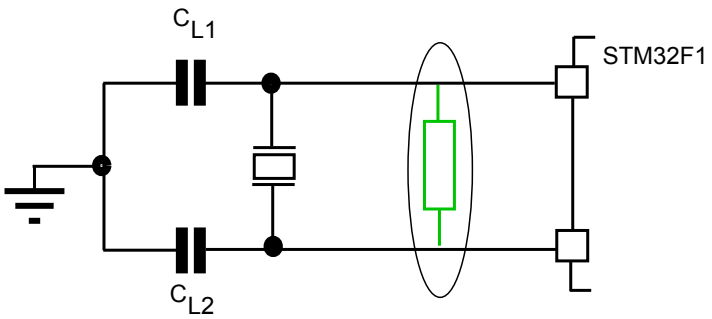
Description

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



MS32554V1

2.2.6 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.

2.2.7 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).

2.2.8 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.

2.2.9 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.

2.2.10 PLL not locking when sourced by HSI/2 after reset if it was previously sourced by HSE with predivider > 1 or PLL2

Description

The limitation occurs when the sequence below is followed:

- PLL source: HSI/2, SYSCLK source: PLL
- PLL source: HSE with predivider greater than 1 or PLL2, SYSCLK source: PLL
- System reset
- PLL source: HSI/2, SYSCLK source: PLL.

The PLL cannot be locked when sourced by HSI/2 after applying system reset if it was previously sourced by HSE with predivider greater than 1 or by PLL2.

Workaround

Enable the HSE oscillator and leave the PLL lock on it before switching the PLL source to HSI/2.

2.3 GPIO

2.3.1 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.

2.3.2 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.

2.3.3 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.

2.3.4 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,

2.3.5 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).

2.3.6 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 ADC

2.5.1 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.

2.6 TIM

2.6.1 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).

2.6.2 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.

2.6.3 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.

2.6.4 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).

2.6.5 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.

2.6.6 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.

2.6.7 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.

2.6.8 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).

2.6.9 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.

2.6.10 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.

2.6.11 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).

2.7 IWDG

2.7.1 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.

2.7.2 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.

2.7.3 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.

2.7.4 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.

2.7.5 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 I2C

2.8.1 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

2.8.2 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.

2.8.3 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

2.8.4 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.

2.8.5 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.

2.8.6 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.*

2.8.7 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.

2.9 USART

2.9.1 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.

2.9.2 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.

2.9.3 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.

2.9.4 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.

2.9.5 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.

2.9.6 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.

2.10 SPI

2.10.1 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)

2.10.2 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.

2.11 I2S

2.11.1 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.

2.11.2 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.

2.11.3 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.

2.12 bxCAN

2.12.1 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.

2.13 OTG_FS

2.13.1 Data in RxFIFO are overwritten when all channels are disabled simultaneously

Description

If the available RxFIFO is just large enough to host 1 packet + its data status, and is currently occupied by the last received data + its status and, at the same time, the application requests that more IN channels be disabled, the OTG_FS peripheral does not first check for available space before inserting the disabled status of the IN channels. It just inserts them by overwriting the existing data payload.

Workaround

Use one of the following recommendations:

1. Configure the RxFIFO to host a minimum of 2 × MPSIZ + 2 × data status entries.
2. The application has to check the RXFLVL bit (RxFIFO non-empty) in the OTG_FS_GINTSTS register before disabling each IN channel. If this bit is not set, then the application can disable an IN channel at a time. Each time the application disables an IN channel, however, it first has to check that the RXFLVL bit = 0 condition is true.

2.13.2 OTG host blocks the receive channel when receiving IN packets and no Tx FIFO is configured

Description

When receiving data, the OTG_FS core erroneously checks for available TxFIFO space when it should only check for RxFIFO space. If the OTG_FS core cannot see any space allocated for data transmission, it blocks the reception channel and no data are received.

Workaround

Set at least one TxFIFO equal to the maximum packet size. In this way, the host application, which intends to supports only IN traffic, also has to allocate some space for the TxFIFO.

Since a USB host is expected to support any kind of connected endpoint, it is good practice to configure enough TxFIFO space for OUT endpoints.

2.13.3 Host channel-halted interrupt not generated when the channel is disabled

Description

When the application enables, then immediately disables the host channel before the OTG_FS host has had time to begin the transfer sequence, the OTG_FS core, as a host, does not generate a channel-halted interrupt. The OTG_FS core continues to operate normally.

Workaround

Do not disable the host channel immediately after enabling it.

2.13.4 Error in software-read OTG_FS_DCFG register values

Description

When the application writes to the DAD and PFIVL bitfields in the OTG_FS_DCFG register, and then reads the newly written bitfield values, the read values may not be correct.

The values written by the application, however, are correctly retained by the core, and the normal operation of the device is not affected.

Workaround

Do not read from the OTG_FS_DCFG register DAD and PFIVL bitfields just after programming them.

2.13.5 Minimum AHB frequency to guarantee correct operation of USB OTG FS peripheral

Description

In order to guarantee correct operation of the USB OTG FS peripheral, the AHB frequency should be configured to be not less than 14.2 MHz.

Workaround

None

2.14 ETH

2.14.1 Possible underflow when TxFIFO is configured in Store-and-Forward mode and a relatively large frame is aborted in Half-duplex mode

Description

In Store-and-Forward mode, the frame is transferred to the MAC only when the full frame is available in the TxFIFO. There is one exception to this rule, when the TxFIFO is almost full.

This problem may arise when a relatively large frame is loaded into the TxFIFO and a second frame is partially loaded (because of lack of space) into the free TxFIFO space, generating an almost full condition. In this case, if the first frame is aborted, due to a carrier loss, the absence of a carrier or a late collision, the second frame is sent to the MAC before it is completely loaded into the TxFIFO. If the DMA bandwidth does not allow the rest of the frame to be uploaded before the end of the frame transmission, an underflow may occur.

In this case, the software must detect the underflow condition and resend the frame.

Workaround

Resend the frame on underflow detection.

2.14.2 Posible CRC error when TxFIFO is configured in Store-and-Forward mode and a relatively large frame is aborted in Half-duplex mode with transmit checksum offload enabled

Description

In Store-and-Forward mode, the frame is transferred to the MAC only when the full frame is available in the TxFIFO. There is one exception to this rule, when the TxFIFO is almost full.

This problem may arise when a relatively large frame is loaded into the TxFIFO and a second frame is partially loaded (because of lack of space) into the free TxFIFO space, generating an almost full condition. In this case, if the first frame is aborted, due to a carrier loss, the absence of a carrier or a late collision, the second frame is sent to the MAC before it is completely loaded into the TxFIFO. Since for CRC computation, the entire frame must be available in the TxFIFO, the CRC of the subsequent frames is erroneous.

TCP/IP checksum errors may be detected at the remote end, causing data to be dropped.

Workaround

Wait for the TxFIFO to become empty and then send the txFIFO flush command.

2.14.3 Erroneous automatic checksum insertion after TxFIFO is dynamically switched from Threshold to Store-and-Forward mode

Description

Automatic checksum insertion in transmitted frames can be enabled (through the CIC bits in the TDES0 register) and used only when the TxFIFO is configured to operate in Store-and-Forward mode. When the TxFIFO is operated in Threshold mode, the CIC bits are ignored.

When the TxFIFO dynamically switches from Threshold to Store-and-Forward mode, the CRC computation may be erroneous on the subsequent frames.

This results in the detection of TCP/IP checksum errors at the remote end, causing data to be dropped.

Workaround

Use the Store-and-Forward mode only.

2.14.4 Erroneous automatic checksum insertion after a large frame (longer than the TxFIFO) transmission

Description

Automatic checksum insertion in transmitted frames can be enabled (through the CIC bits in TDES0) and used only for frames with a length lesser than the TxFIFO depth.

If a long frame is transmitted, with checksum insertion disabled (CIC=00), immediately followed by a short frame, with checksum insertion enabled, the computed checksum may be erroneous. This is due to the fact that the second frame may have be transmitted too early from the TxFIFO, due to a TxFIFO almost full condition.

This results in the detection of TCP/IP checksum errors at the remote end, causing data to be dropped.

Workaround

1. Avoid enabling automatic checksum insertion for any frame if your system transmits frames with a size larger than the depth of the TxFIFO.
2. Wait for the long frame transmission completion before re-enabling the automatic checksum insertion.

2.14.5 In half-duplex mode, the MAC transmitter ignores collisions after it is disabled during a frame transmission

Description

If the TE bit is cleared during a transmission, the transmission part of the MAC is effectively disabled after the complete transmission of the current frame. From the clearing of the TE bit until the end of the transmission, potential collisions are ignored.

If a collision event occurs after the MAC is disabled, the transmitter does not recognize the event and continues to transmit the complete frame without any JAM pattern. It reports a successful frame transmission status (without any collision) even though the frame is corrupted at the remote receivers due to collision.

Since a JAM signal might not be sent during a collision, frames may be lost when a remote transmitter does not detect that a collision occurred.

Workaround

Disable the MAC transmitter only after the completion of the transmission of all scheduled frames in Half-duplex mode.

2.14.6 Interrupt due to an RMON (MMC) counter may be set again after it is cleared

Description

When enabled, an interrupt asserted due to an RMON (remote monitoring) counter becoming full, is cleared when the corresponding counter is read. The counter is also cleared by the read operation if bit 1 (counter stop rollover) in ETH_MMCCR is set. If this clear signal coincides with the counter update signal generated by the presence of a new frame, then the corresponding interrupt bit is set again. This results in the software getting a spurious interrupt from the MMC even though the corresponding counter value is very low.

Workaround

None

2.14.7 Incorrect layer 3 (L3) checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads

Description

The application provides the per-frame control to instruct the MAC to insert the L3 checksums for TCP, UDP and ICMP packets. When automatic checksum insertion is enabled and the input packet is an IPv6 packet without the TCP, UDP or ICMP payload, then the MAC may incorrectly insert a checksum into the packet. For IPv6 packets without a TCP, UDP or ICMP payload, the MAC core considers the next header (NH) field as the extension header and continues to parse the extension header. Sometimes, the payload data in such packets matches the NH field for TCP, UDP or ICMP and, as a result, the MAC core inserts a checksum.

Workaround

When the IPv6 packets have a TCP, UDP or ICMP payload, enable checksum insertion for transmit frames, or bypass checksum insertion by using the CIC (checksum insertion control) bits in TDES0 (bits 23:22).

2.14.8 The Ethernet MAC processes invalid extension headers in the received IPv6 frames

Description

In IPv6 frames, there can be zero or some extension headers preceding the actual IP payload. The Ethernet MAC processes the following extension headers defined in the IPv6 protocol: Hop-by-Hop Options header, Routing header and Destination Options header.

All extension headers except the Hop-by-Hop extension header can be present multiple times and in any order before the actual IP payload. The Hop-by-Hop extension header, if present, has to come immediately after the IPv6's main header.

The Ethernet MAC processes all (valid or invalid) extension headers including the Hop-by-Hop extension headers that are present after the first extension header. For this reason, the GMAC core accepts IPv6 frames with invalid Hop-by-Hop extension headers. As a consequence, it accepts any IP payload as valid IPv6 frames with TCP, UDP or ICMP payload, and then incorrectly update the Receive status of the corresponding frame.

Workaround

None

2.14.9 MAC stuck in the Idle state on receiving the TxFIFO flush command exactly one clock cycle after a transmission completes

Description

When the software issues a TxFIFO flush command, the transfer of frame data stops (even in the middle of a frame transfer). The TxFIFO read controller goes into the Idle state (TFRS=00 in ETH_MACDBGR) and then resumes its normal operation.

However, if the TxFIFO read controller receives the TxFIFO flush command exactly one clock cycle after receiving the status from the MAC, the controller remains stuck in the Idle state and stops transmitting frames from the TxFIFO. The system can recover from this state only with a reset (e.g. a soft reset).

Workaround

Do not use the TxFIFO flush feature.

If TXFIFO flush is really needed, wait until the TxFIFO is empty prior to using the TxFIFO flush command.

2.14.10 Transmit frame data corruption

Frame data corrupted when the TxFIFO is repeatedly transitioning from non-empty to empty and then back to non-empty.

Description

Frame data may get corrupted when the TxFIFO is repeatedly transitioning from non-empty to empty for a very short period, and then from empty to non-empty, without causing an underflow.

This transitioning from non-empty to empty and back to non-empty happens when the rate at which the data are being written to the TxFIFO is almost equal to or a little less than the rate at which the data are being read.

This corruption cannot be detected by the receiver when the CRC is inserted by the MAC, as the corrupted data are used for the CRC computation.

Workaround

Use the Store-and-Forward mode: TSF=1 (bit 21 in ETH_DMAOMR). In this mode the data are transmitted only when the whole packet is available in the TxFIFO.

2.14.11 Ethernet DMA not working after WFI/WFE instruction

Description

If a WFI/WFE instruction is executed to put the system in sleep mode while the Ethernet MAC master clock on the AHB bus matrix is ON and all remaining masters clocks are OFF, the Ethernet DMA is unable to perform any AHB master accesses during sleep mode.

Workaround

Enable DMA1 or DMA2 clocks in the RCC_AHBENR register before executing the WFI/WFE instruction.

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	Version	Changes
16-Jun-2009	1	Initial release.
16-Oct-2009	2	Section 2.8.5: I2S2 in master/slave mode and Ethernet/USART3 in synchronous mode added. Table 4: Summary of silicon limitations in revision Z devices modified. Section 2.19: OTG_FS added. Section 2.20: Ethernet MAC added. Section 2.21: Bootloader limitations added. Figure 3: LQFP100 top package view and Figure 4: LQFP64 top package view updated to show the Date code.
15-Dec-2009	3	Added limitations: • Section 2.8.6: USARTx_TX pin usage • Section 2.6.3: Start cannot be generated after a misplaced Stop • Section 2.11.5: Mismatch on the “Setup time for a repeated Start condition” timing parameter • Section 2.11.6: Data valid time (tVD;DAT) violated without the OVR flag being set • Section 2.12.1: CRC still sensitive to communication clock when SPI is in slave mode even with NSS high
29-Mar-2010	4	Section 2.18.2: USART bootloader: incorrect protocol version returned by the Get Version command added.
18-Jun-2010	5	Added Section 2.4: Debugging Stop mode with WFE entry Added Section 2.4: Debugging Stop mode with WFE entry Added Section 2.11.2: Wrong data read into data register Modified Section 2.11.4: Wrong behavior of I2C peripheral in master mode after a misplaced Stop Modified Section 2.11.6: Data valid time (tVD;DAT) violated without the OVR flag being set Added Section 2.14: USART peripheral Added Figure 2: LFBGA100 top package view
06-Jan-2011	6	Added Section 2.13.1: Wrong WS signal generation in 16-bit extended to 32-bit PCM long synchronization mode Added Section 2.13.2: In I2S slave mode, WS level must be set by the external master when enabling the I2S Added Section 2.13.3: I2S slave mode desynchronization with the master during communication Added Section 2.14.6: nRTS signal abnormally driven low after a protocol violation Added Section 2.19.5: Minimum AHB frequency to guarantee correct operation of USB OTG FS peripheral Added Section 2.20.11: Ethernet DMA not working after WFI/WFE instruction
07-Feb-2011	7	Updated workarounds in Section 2.11.1: Some software events must be managed before the current byte is being transferred and Section 2.11.2: Wrong data read into data register Updated Section 2.13.2: In I2S slave mode, WS level must be set by the external master when enabling the I2S
04-Jan-2012	8	Table 3: Cortex-M3 core limitations and impact on microcontroller behavior: added limitations 752419 and 740455.

Date	Version	Changes
		Added Section 1.1.5: Interrupted loads to the SP can cause erroneous behavior and Section 1.1.6: SVC and BusFault/MemManage may occur out of order. Updated “workaround” of Section 2.14.6: nRTS signal abnormally driven low after a protocol violation Added .Section 2.16.1: RVU and PVU flags are not reset in Stop mode. Created Section 2.21: Bootloader limitations and removed previous Section 2.18 and Section 2.19.
07-Oct-2013	9	Added: • Section 2.5: Wakeup sequence from Standby mode when using more than one wakeup source • Section 2.6: LSE start-up in harsh environments • Section 2.11.7: I2C analog filter may provide wrong value, locking BUSY flag and preventing master mode entry • Section 2.12.2: 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 Updated Table 4: Summary of silicon limitations in revision Z devices.
15-Apr-2020	10	Updated Section 1: Arm® 32-bit Cortex®-M3 limitations and Appendix A: Revision and date codes on device marking. Updated Table 4: Summary of silicon limitations in revision Z devices. Added Section 2.7: RDP protection. Updated Figure 2: LFBGA100 top package view. Minor text edits across the whole document.
30-Jun-2022	11	Added: • 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 2.6.1 PWM re-enabled in automatic output enable mode despite of system break • Section 2.6.2 TRGO and TRGO2 trigger output failure • Section 2.6.3 Consecutive compare event missed in specific conditions • Section 2.6.4 Output compare clear not working with external counter reset • Section 2.7.1 RVU flag not cleared at low APB clock frequency • Section 2.7.2 PVU flag not cleared at low APB clock frequency • Section 2.12.1 bxCAN time-triggered communication mode not supported • Table 4. Summary of device documentation errata Updated: • Section 2.2.3 Debugging Stop mode with WFE entry Removed Bootloader limitations section

Contents

1 **Summary of device errata** 2
2 **Description of device errata** 5
2.1 Core 5
2.1.1 Cortex®-M3 LDRD with base in list may result in incorrect base register when interrupted or faulted 5
2.1.2 Cortex®-M3 event register is not set by interrupts and debug 5
2.1.3 Interrupted loads to SP can cause erroneous behavior 5
2.1.4 SVC and BusFault/MemManage may occur out of order 6
2.1.5 Arm® Cortex®-M3 BKPT in debug monitor mode can cause DFSR mismatch 6
2.1.6 Arm® Cortex®-M3 may freeze for SLEEPONEXIT single instruction ISR 6
2.2 System 7
2.2.1 Flash memory read after WFI/WFE instruction 7
2.2.2 Debugging Stop mode and SysTick timer 7
2.2.3 Debugging Stop mode with WFE entry 7
2.2.4 Wakeup sequence from Standby mode when using more than one wakeup source 7
2.2.5 LSE startup in harsh environments 8
2.2.6 RDP protection 8
2.2.7 Boundary scan TAP: wrong pattern sent out after the "capture IR” state 9
2.2.8 Flash memory BSY bit delay versus STRT bit setting 9
2.2.9 LSI clock stabilization time 9
2.2.10 PLL not locking when sourced by HSI/2 after reset if it was previously sourced by HSE with predivider > 1 or PLL2 9
2.3 GPIO 10
2.3.1 SPI1 in slave mode and USART2 in synchronous mode 10
2.3.2 SPI1 in master mode and USART2 in synchronous mode 10
2.3.3 SPI2 in slave mode and USART3 in synchronous mode 10
2.3.4 SPI2 in master mode and USART3 in synchronous mode 10
2.3.5 I2S2 in master/slave mode and USART3 in synchronous mode 11
2.3.6 USARTx_TX pin usage 11
2.4 DMA 11
2.4.1 DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear 11
2.4.2 Byte and half-word accesses not supported 11
2.5 ADC 12
2.5.1 Voltage glitch on ADC input 0 12
2.6 TIM 12
2.6.1 PWM re-enabled in automatic output enable mode despite of system break 12

2.6.2 TRGO and TRGO2 trigger output failure . . . 12
2.6.3 Consecutive compare event missed in specific conditions . . . 13
2.6.4 Output compare clear not working with external counter reset . . . 13
2.6.5 Missing capture flag . . . 14
2.6.6 Overcapture detected too early . . . 14
2.6.7 General-purpose timer regulation for 100% PWM . . . 14
2.6.8 PWM re-enabled in automatic output enable mode despite of system break . . . 14
2.6.9 TRGO and TRGO2 trigger output failure . . . 15
2.6.10 Consecutive compare event missed in specific conditions . . . 15
2.6.11 Output compare clear not working with external counter reset . . . 15
2.7 IWDG . . . 16
2.7.1 RVU flag not cleared at low APB clock frequency . . . 16
2.7.2 PVU flag not cleared at low APB clock frequency . . . 16
2.7.3 RVU flag not cleared at low APB clock frequency . . . 16
2.7.4 PVU flag not cleared at low APB clock frequency . . . 17
2.7.5 RVU and PVU flags are not cleared in Stop mode . . . 17
2.8 I2C . . . 17
2.8.1 Some software events must be managed before the current byte is being transferred . . . 17
2.8.2 Wrong data read into data register . . . 18
2.8.3 SMBus standard not fully supported . . . 19
2.8.4 Wrong behavior of I2C peripheral in master mode after a misplaced Stop . . . 19
2.8.5 Mismatch on the “Setup time for a repeated Start condition” timing parameter . . . 20
2.8.6 Data valid time ($t_{VD;DAT}$) violated without the OVR flag being set . . . 20
2.8.7 I2C analog filter may provide wrong value, locking BUSY flag and preventing master mode entry . . . 21
2.9 USART . . . 21
2.9.1 Parity Error flag (PE) is set again after having been cleared by software . . . 21
2.9.2 Idle frame is not detected if receiver clock speed is deviated . . . 22
2.9.3 In full-duplex mode, the Parity Error (PE) flag can be cleared by writing the data register . . . 22
2.9.4 Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection . . . 22
2.9.5 Break frame is transmitted regardless of nCTS input line status . . . 22
2.9.6 nRTS signal abnormally driven low after a protocol violation . . . 22
2.10 SPI . . . 23
2.10.1 CRC still sensitive to communication clock when SPI is in slave mode even with NSS high . . . 23
2.10.2 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 . . . 23
2.11 I2S . . . 23

2.11.1 Wrong WS signal generation in 16-bit extended to 32-bit PCM long synchronisation mode . . . 23
2.11.2 In I2S slave mode, WS level must be set by the external master when enabling the I2S . 24
2.11.3 I2S slave mode desynchronisation with the master during communication . . . 24
2.12 bxCAN . . . 24
2.12.1 bxCAN time-triggered communication mode not supported. . . . 24
2.13 OTG_FS . . . 25
2.13.1 Data in RxFIFO are overwritten when all channels are disabled simultaneously . . . 25
2.13.2 OTG host blocks the receive channel when receiving IN packets and no Tx FIFO is configured . . . 25
2.13.3 Host channel-halted interrupt not generated when the channel is disabled . . . 25
2.13.4 Error in software-read OTG_FS_DCFG register values . . . 25
2.13.5 Minimum AHB frequency to guarantee correct operation of USB OTG FS peripheral . . . 26
2.14 ETH. . . . 26
2.14.1 Possible underflow when TxFIFO is configured in Store-and-Forward mode and a relatively large frame is aborted in Half-duplex mode. . . . 26
2.14.2 Posible CRC error when TxFIFO is configured in Store-and-Forward mode and a relatively large frame is aborted in Half-duplex mode with transmit checksum offload enabled . . . 26
2.14.3 Erroneous automatic checksum insertion after TxFIFO is dynamically switched from Threshold to Store-and-Forward mode . . . 26
2.14.4 Erroneous automatic checksum insertion after a large frame (longer than the TxFIFO) transmission . . . 27
2.14.5 In half-duplex mode, the MAC transmitter ignores collisions after it is disabled during a frame transmission . . . 27
2.14.6 Interrupt due to an RMON (MMC) counter may be set again after it is cleared . . . 27
2.14.7 Incorrect layer 3 (L3) checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads . . . 28
2.14.8 The Ethernet MAC processes invalid extension headers in the received IPv6 frames. . . . 28
2.14.9 MAC stuck in the Idle state on receiving the TxFIFO flush command exactly one clock cycle after a transmission completes . . . 28
2.14.10 Transmit frame data corruption . . . 28
2.14.11 Ethernet DMA not working after WFI/WFE instruction . . . 29
Important security notice . . . 30
Revision history . . . 31