

## STM32F105xx and STM32F107xx revision Z connectivity line device limitations

### Silicon identification

This errata sheet applies to the revision Z of the STMicroelectronics STM32F105xx and STM32F107xx connectivity line products. This family features an ARM™ 32-bit Cortex®-M3 core, for which an errata notice is also available (see [Section 1](#) for details).

The full list of part numbers is shown in [Table 2](#). The products are identifiable as shown in [Table 1](#):

- by the revision code marked below the sales type on the device package
- by the last three digits of the internal sales type printed on the box label

**Table 1. Device Identification<sup>(1)</sup>**

Sales type	Revision code <sup>(2)</sup> marked on device
STM32F105xx	“Z”
STM32F107xx	“Z”

1. The REV\_ID bits in the DBGMCU\_IDCODE register show the revision code of the device (see the STM32F10xxx reference manual for details on how to find the revision code).
2. Refer to [Appendix A: Revision and date codes on device marking](#) for details on how to identify the revision code on the different packages.

**Table 2. Device summary**

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

# Contents

- 1 ARM™ 32-bit Cortex®-M3 limitations ..... 7**
  - 1.1 Cortex-M3 limitation description for the STM32F105xx / STM32F107xx connectivity line devices ..... 8
    - 1.1.1 Cortex-M3 LDRD with base in list may result in incorrect base register when interrupted or faulted ..... 8
    - 1.1.2 Cortex-M3 event register is not set by interrupts and debug ..... 8
    - 1.1.3 Cortex-M3 BKPT in debug monitor mode can cause DFSR mismatch .. 8
    - 1.1.4 Cortex-M3 may freeze for SLEEPONEXIT single instruction ISR ..... 9
    - 1.1.5 Interrupted loads to the SP can cause erroneous behaviour ..... 9
    - 1.1.6 SVC and BusFault/MemManage may occur out of order ..... 10
  
- 2 STM32F105xx and STM32F107xx silicon limitations ..... 11**
  - 2.1 Voltage glitch on ADC input 0 ..... 14
  - 2.2 Flash memory read after WFI/WFE instruction ..... 14
  - 2.3 Debugging Stop mode and system tick timer ..... 15
  - 2.4 Debugging Stop mode with WFE entry ..... 15
  - 2.5 Wakeup sequence from Standby mode when using more than one wakeup source ..... 15
  - 2.6 LSE start-up in harsh environments ..... 16
  - 2.7 Alternate function ..... 17
    - 2.7.1 SPI1 in slave mode and USART2 in synchronous mode ..... 17
    - 2.7.2 SPI1 in master mode and USART2 in synchronous mode ..... 17
    - 2.7.3 SPI2 in slave mode and USART3 in synchronous mode ..... 17
    - 2.7.4 SPI2 in master mode and USART3 in synchronous mode ..... 18
    - 2.7.5 I2S2 in master/slave mode and Ethernet/USART3 in synchronous mode ..... 18
    - 2.7.6 USARTx\_TX pin usage ..... 18
  - 2.8 Boundary scan TAP: wrong pattern sent out after the “capture IR” state . 19
  - 2.9 Flash memory BSY bit delay versus STRT bit setting ..... 19
  - 2.10 I<sup>2</sup>C peripheral ..... 20
    - 2.10.1 Some software events must be managed before the current byte is being transferred ..... 20
    - 2.10.2 Wrong data read into data register ..... 21
    - 2.10.3 SMBus standard not fully supported ..... 22



2.10.4	Wrong behavior of I2C peripheral in master mode after a misplaced Stop	22
2.10.5	Mismatch on the "Setup time for a repeated Start condition" timing parameter	23
2.10.6	Data valid time ( $t_{VD,DAT}$ ) violated without the OVR flag being set	23
2.10.7	I2C analog filter may provide wrong value, locking BUSY flag and preventing master mode entry	25
2.11	SPI peripheral	26
2.11.1	CRC still sensitive to communication clock when SPI is in slave mode even with NSS high	26
2.11.2	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	26
2.12	I2S peripheral	27
2.12.1	Wrong WS signal generation in 16-bit extended to 32-bit PCM long synchronisation mode	27
2.12.2	In I2S slave mode, WS level must be set by the external master when enabling the I2S	27
2.12.3	I2S slave mode desynchronisation with the master during communication	27
2.13	USART peripheral	28
2.13.1	Parity Error flag (PE) is set again after having been cleared by software	28
2.13.2	Idle frame is not detected if receiver clock speed is deviated	28
2.13.3	In full duplex mode, the Parity Error (PE) flag can be cleared by writing the data register	28
2.13.4	Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection	29
2.13.5	Break frame is transmitted regardless of nCTS input line status	29
2.13.6	nRTS signal abnormally driven low after a protocol violation	29
2.14	Timers	30
2.14.1	Missing capture flag	30
2.14.2	Overcapture detected too early	30
2.14.3	General-purpose timer: regulation for 100% PWM	30
2.15	IWDG peripheral	31
2.15.1	RVU and PVU flags are not reset in Stop mode	31
2.16	LSI clock stabilization time	31
2.17	PLL not locking when sourced by HSI/2 after reset if it was previously sourced by HSE with predivider >1 or PLL2	31
2.18	OTG_FS	32

2.18.1	Data in RxFIFO are overwritten when all channels are disabled simultaneously	32
2.18.2	OTG host blocks the receive channel when receiving IN packets and no TxFIFO is configured	32
2.18.3	Host channel-halted interrupt not generated when the channel is disabled	32
2.18.4	Error in software-read OTG_FS_DCFG register values	33
2.18.5	Minimum AHB frequency to guarantee correct operation of USB OTG FS peripheral	33
2.19	Ethernet MAC	33
2.19.1	Possible underflow when TxFIFO is configured in Store-and-Forward mode and a relatively large frame is aborted in Half-duplex mode	33
2.19.2	Possible 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	34
2.19.3	Erroneous automatic checksum insertion after TxFIFO is dynamically switched from Threshold to Store-and-Forward mode	34
2.19.4	Erroneous automatic checksum insertion after a large frame (longer than the TxFIFO) transmission	35
2.19.5	In half-duplex mode, the MAC transmitter ignores collisions after it is disabled during a frame transmission	35
2.19.6	Interrupt due to an RMON (MMC) counter may be set again after it is cleared	36
2.19.7	Incorrect layer 3 (L3) checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads	36
2.19.8	The Ethernet MAC processes invalid extension headers in the received IPv6 frames	37
2.19.9	MAC stuck in the Idle state on receiving the TxFIFO flush command exactly 1 clock cycle after a transmission completes	37
2.19.10	Transmit frame data corruption	38
2.19.11	Ethernet DMA not working after WFI/WFE instruction	38
2.20	Bootloader limitations	38
<b>Appendix A Revision and date codes on device marking</b>		<b>39</b>
<b>Revision history</b>		<b>42</b>

## List of tables

Table 1.	Device Identification . . . . .	1
Table 2.	Device summary . . . . .	1
Table 3.	Cortex-M3 core limitations and impact on microcontroller behavior . . . . .	7
Table 4.	Summary of silicon limitations in revision Z devices . . . . .	11
Table 5.	Document revision history . . . . .	42

## List of figures

Figure 1.	LSE start-up using an additional resistor . . . . .	16
Figure 2.	LFPGA100 top package view . . . . .	39
Figure 3.	LQFP100 top package view . . . . .	40
Figure 4.	LQFP64 top package view . . . . .	41

# 1 ARM™ 32-bit Cortex®-M3 limitations

An errata notice of the STM32F10xxx core is available from the following web address:

<http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.eat0420a/>.

The direct link to the errata notice pdf is:

<http://infocenter.arm.com/help/topic/com.arm.doc.eat0420a/Cortex-M3-Errata-r1p1-v0.2.pdf>.

All the described limitations are minor and related to the revision r1p1-01rel0 of the Cortex-M3 core. [Table 3](#) summarizes these limitations and their implications on the behavior of the STM32F105xx / STM32F107xx connectivity line devices.

**Table 3. Cortex-M3 core limitations and impact on microcontroller behavior**

ARM ID	ARM category	ARM summary of errata	Impact on STM32F105xx / STM32F107xx connectivity line devices
752419	Cat 2	Interrupted loads to the stack pointer (SP) can cause erroneous behavior	Minor
740455	Cat 2	SVC and BusFault/MemManage may occur out of order	Minor
602117	Cat 2	LDRD with base in list may result in incorrect base register when interrupted or faulted	Minor
563915	Cat 2	Event register is not set by interrupts and debug	Minor
531064	impl	SWJ-DP missing POR reset sync	No
511864	Cat 3	Cortex-M3 may fetch instructions using incorrect privilege on return from an exception	No
532314	Cat 3	DWT CPI counter increments during sleep	No
538714	Cat 3	Cortex-M3 TPIU clock domain crossing	No
548721	Cat 3	Internal write buffer could be active whilst asleep	No
463763	Cat 3	BKPT in debug monitor mode can cause DFSR mismatch	Minor
463764	Cat 3	Core may freeze for SLEEPONEXIT single instruction ISR	Minor
463769	Cat 3	Unaligned MPU fault during a write may cause the wrong data to be written to a successful first access	No

## 1.1 Cortex-M3 limitation description for the STM32F105xx / STM32F107xx connectivity line devices

Only the limitations described below have an impact, even though minor, on the implementation of STM32F105xx / STM32F107xx connectivity line devices.

All the other limitations described in the ARM errata notice (and summarized in [Table 3](#) above) have no impact and are not related to the implementation of the STM32F105xx / STM32F107xx connectivity line devices (Cortex-M3 r1p1-01rel0).

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

#### Description

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.

#### Workarounds

1. This limitation does not impact the STM32F10xxx 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).

### 1.1.2 Cortex-M3 event register is not set by interrupts and debug

#### Description

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 STM32F10xxx external events instead of interrupts to wake up the core from WFE by configuring an external or internal EXTI line in event mode.

### 1.1.3 Cortex-M3 BKPT in debug monitor mode can cause DFSR mismatch

#### Description

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 0xE00ED30 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.

## 1.1.4 Cortex-M3 may freeze for SLEEPONEXIT single instruction ISR

### Description

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).

## 1.1.5 Interrupted loads to the SP can cause erroneous behaviour

### Description

If an interrupt occurs during the data-phase of a single word load to the stack pointer (SP/R13), erroneous behaviour 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, no compiler generates these particular instructions. This limitation can only occur with hand-written assembly code.

A workaround for both issues is possible 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 "
```

## 1.1.6 SVC and BusFault/MemManage may occur out of order

### Description

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 work around 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 STM32F105xx and STM32F107xx silicon limitations

Table 4 gives quick references to all documented limitations.

**Table 4. Summary of silicon limitations in revision Z devices**

Links to silicon limitations	
	<a href="#">Section 2.1: Voltage glitch on ADC input 0</a>
	<a href="#">Section 2.2: Flash memory read after WFI/WFE instruction</a>
	<a href="#">Section 2.3: Debugging Stop mode and system tick timer</a>
	<a href="#">Section 2.4: Debugging Stop mode with WFE entry</a>
	<a href="#">Section 2.5: Wakeup sequence from Standby mode when using more than one wakeup source</a>
	<a href="#">Section 2.6: LSE start-up in harsh environments</a>
<a href="#">Section 2.7: Alternate function</a>	<a href="#">Section 2.7.1: SPI1 in slave mode and USART2 in synchronous mode</a>
	<a href="#">Section 2.7.2: SPI1 in master mode and USART2 in synchronous mode</a>
	<a href="#">Section 2.7.3: SPI2 in slave mode and USART3 in synchronous mode</a>
	<a href="#">Section 2.7.4: SPI2 in master mode and USART3 in synchronous mode</a>
	<a href="#">Section 2.7.5: I2S2 in master/slave mode and Ethernet/USART3 in synchronous mode</a>
	<a href="#">Section 2.7.6: USARTx_TX pin usage</a>
	<a href="#">Section 2.8: Boundary scan TAP: wrong pattern sent out after the "capture IR" state</a>
	<a href="#">Section 2.9: Flash memory BSY bit delay versus STRT bit setting</a>
<a href="#">Section 2.10: I<sup>2</sup>C peripheral</a>	<a href="#">Section 2.10.1: Some software events must be managed before the current byte is being transferred</a>
	<a href="#">Section 2.10.2: Wrong data read into data register</a>
	<a href="#">Section 2.10.3: SMBus standard not fully supported</a>
	<a href="#">Section 2.10.4: Wrong behavior of I2C peripheral in master mode after a misplaced Stop</a>
	<a href="#">Section 2.10.5: Mismatch on the "Setup time for a repeated Start condition" timing parameter</a>
	<a href="#">Section 2.10.6: Data valid time (t<sub>VD;DAT</sub>) violated without the OVR flag being set</a>
	<a href="#">Section 2.10.7: I2C analog filter may provide wrong value, locking BUSY flag and preventing master mode entry</a>
<a href="#">Section 2.11: SPI peripheral</a>	<a href="#">Section 2.11.1: CRC still sensitive to communication clock when SPI is in slave mode even with NSS high</a>
	<a href="#">Section 2.11.2: 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>

**Table 4. Summary of silicon limitations in revision Z devices (continued)**

Links to silicon limitations	
<i>Section 2.12: I2S peripheral</i>	<i>Section 2.12.1: Wrong WS signal generation in 16-bit extended to 32-bit PCM long synchronisation mode</i>
	<i>Section 2.12.2: In I2S slave mode, WS level must be set by the external master when enabling the I2S</i>
	<i>Section 2.12.3: I2S slave mode desynchronisation with the master during communication</i>
<i>Section 2.13: USART peripheral</i>	<i>Section 2.13.1: Parity Error flag (PE) is set again after having been cleared by software</i>
	<i>Section 2.13.2: Idle frame is not detected if receiver clock speed is deviated</i>
	<i>Section 2.13.3: In full duplex mode, the Parity Error (PE) flag can be cleared by writing the data register</i>
	<i>Section 2.13.4: Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection</i>
	<i>Section 2.13.5: Break frame is transmitted regardless of nCTS input line status</i>
	<i>Section 2.13.6: nRTS signal abnormally driven low after a protocol violation</i>
<i>Section 2.14: Timers</i>	<i>Section 2.14.1: Missing capture flag</i>
	<i>Section 2.14.2: Overcapture detected too early</i>
	<i>Section 2.14.3: General-purpose timer: regulation for 100% PWM</i>
<i>Section 2.15: IWDG peripheral</i>	<i>Section 2.15.1: RVU and PVU flags are not reset in Stop mode</i>
<i>Section 2.16: LSI clock stabilization time</i>	
<i>Section 2.17: PLL not locking when sourced by HSI/2 after reset if it was previously sourced by HSE with predivider &gt;1 or PLL2</i>	
<i>Section 2.18: OTG_FS</i>	<i>Section 2.18.1: Data in RxFIFO are overwritten when all channels are disabled simultaneously</i>
	<i>Section 2.18.2: OTG host blocks the receive channel when receiving IN packets and no TxFIFO is configured</i>
	<i>Section 2.18.3: Host channel-halted interrupt not generated when the channel is disabled</i>
	<i>Section 2.18.4: Error in software-read OTG_FS_DCFG register values</i>
	<i>Section 2.18.5: Minimum AHB frequency to guarantee correct operation of USB OTG FS peripheral</i>

**Table 4. Summary of silicon limitations in revision Z devices (continued)**

Links to silicon limitations	
Section 2.19: Ethernet MAC	<a href="#">Section 2.19.1: Possible underflow when TxFIFO is configured in Store-and-Forward mode and a relatively large frame is aborted in Half-duplex mode</a>
	<a href="#">Section 2.19.2: Possible 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 href="#">Section 2.19.3: Erroneous automatic checksum insertion after TxFIFO is dynamically switched from Threshold to Store-and-Forward mode</a>
	<a href="#">Section 2.19.4: Erroneous automatic checksum insertion after a large frame (longer than the TxFIFO) transmission</a>
	<a href="#">Section 2.19.5: In half-duplex mode, the MAC transmitter ignores collisions after it is disabled during a frame transmission</a>
	<a href="#">Section 2.19.6: Interrupt due to an RMON (MMC) counter may be set again after it is cleared</a>
	<a href="#">Section 2.19.7: Incorrect layer 3 (L3) checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads</a>
	<a href="#">Section 2.19.8: The Ethernet MAC processes invalid extension headers in the received IPv6 frames</a>
	<a href="#">Section 2.19.9: MAC stuck in the Idle state on receiving the TxFIFO flush command exactly 1 clock cycle after a transmission completes</a>
	<a href="#">Section 2.19.10: Transmit frame data corruption</a>
	<a href="#">Section 2.19.11: Ethernet DMA not working after WFI/WFE instruction</a>
<a href="#">Section 2.20: Bootloader limitations</a>	

## 2.1 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 as a digital input, it will not be detected as a spurious transition, providing that PA0 is driven with an impedance lower than 5 k $\Omega$ . 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.

## 2.2 Flash memory read after WFI/WFE instruction

### Conditions

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

### 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.

### 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).

## 2.3 Debugging Stop mode and system tick timer

### Description

If the system tick timer interrupt is enabled during the Stop mode debug (DBG\_STOP bit set in the DBGMCU\_CR register ), it will wakeup the system from Stop mode.

### Workaround

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

## 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: `__asm void _WFE(void) {`

`WFE`

`NOP`

`BX lr }`

## 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 will enter the Standby mode, but then it will wake up immediately generating the power reset.

## 2.6 LSE start-up in harsh environments

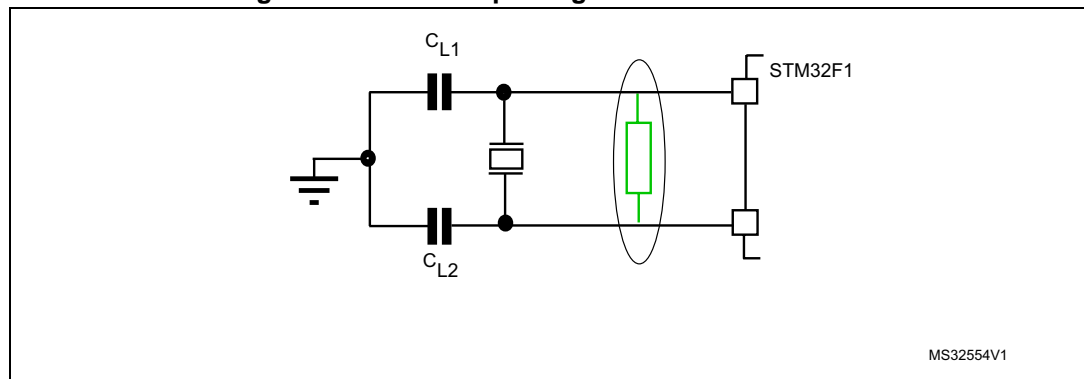
### Description

The LSE (Low Speed External) oscillator system has been designed to minimize the overall power consumption of the STM32F1 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 start-up, 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 M $\Omega$  to 22 M $\Omega$ ) on board to help the oscillation start-up in all cases (see [Figure 1](#)). For more details on compatible crystals and hardware techniques on PCB, refer to AN2867 application note.

Figure 1. LSE start-up using an additional resistor





## 2.7 Alternate function

In some specific cases, some potential weakness may exist between alternate functions mapped onto the same pin.

### 2.7.1 SPI1 in slave mode and USART2 in synchronous mode

#### Conditions

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

#### Description

USART2 cannot be used in synchronous mode (USART2\_CK signal), if SPI1 is used in slave mode.

#### Workaround

None.

### 2.7.2 SPI1 in master mode and USART2 in synchronous mode

#### Conditions

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

#### Description

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.

### 2.7.3 SPI2 in slave mode and USART3 in synchronous mode

#### Conditions

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

#### Description

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

#### Workaround

None.

## 2.7.4 SPI2 in master mode and USART3 in synchronous mode

### Conditions

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

### Description

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.

## 2.7.5 I2S2 in master/slave mode and Ethernet/USART3 in synchronous mode

### Conditions

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

### Description

If I2S2 was used prior to operating USART3 in synchronous mode or the Ethernet, a conflict occurs between the I2S2\_WS and the ETH\_MII\_TXD0 / USART3\_CK signals even though the I2S2 clock was disabled.

### Workaround

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

To use the Ethernet, first disable the I2S2 clock, then either perform a software reset of SPI2(I2S2) or switch off the I2S mode of SPI2.

## 2.7.6 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).

## 2.8 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 lower 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 (XXXXXX).

## 2.9 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.

## 2.10 I<sup>2</sup>C peripheral

### 2.10.1 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.

#### Workarounds

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:
  - a) ADDR=1
  - b) Configure SCL I/O as GPIO open-drain output low
  - c) Clear ADDR by reading SR1 register followed by reading SR3
  - d) Program ACK=0
  - e) Configure SCL I/O as Alternate Function open drainEV6\_3 event (used in master receiver 1 byte):  
Stretch SCL line between ADDR bit is cleared and STOP bit programming:
  - a) ADDR=1
  - b) Program ACK=0
  - c) Configure SCL I/O as GPIO open-drain output low
  - d) Clear ADDR by reading SR1 register followed by reading SR3
  - e) Program STOP=1
  - f) Configure SCL I/O as Alternate Function open drain

## 2.10.2 Wrong data read into data register

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  $N = 2$ :
  - a) BTF = 1(Data N-1 in DR and Data N in shift register)
  - b) Program STOP = 1,
  - c) Read DR twice (Read Data N-1 and Data N) just after programming the STOP.
- **Sequence 2:** Transfer sequence for master receiver when  $N > 2$ :
  - a) BTF = 1 (Data N-2 in DR and Data N-1 in shift register)
  - b) Program ACK = 0,
  - c) Read DataN-2 in DR.
  - d) Program STOP = 1,
  - e) 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) will be corrupted (data N is shifted 1-bit to the left).

### Workarounds

- 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:

  - a) BTF = 1(Data N-1 in DR and Data N in shift register)
  - b) Configure SCL I/O as GPIO open-drain output low
  - c) Program STOP = 1
  - d) Read Data N-1
  - e) Configure SCL I/O as Alternate Function open drain
  - f) Read Data N

Sequence 2:

  - a) BTF = 1 (Data N-2 in DR and Data N-1 in shift register)
  - b) Program ACK = 0
  - c) Configure SCL I/O as GPIO open-drain output low
  - d) Read Data N-2 in DR.
  - e) Program STOP = 1,
  - f) Read Data N-1.
  - g) 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.

### 2.10.3 SMBus standard not fully supported

#### Description

The I<sup>2</sup>C peripheral is not fully compliant with the SMBus v2.0 standard since it does not support the capability to NACK an invalid byte/command.

#### Workarounds

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

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

### 2.10.4 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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.

## 2.10.5 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  $T_{su;sta}$  in the I<sup>2</sup>C specification) can be slightly violated when the I<sup>2</sup>C 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 I<sup>2</sup>C Fast-mode if supported by the slave.

## 2.10.6 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 I<sup>2</sup>C 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:*

*Clear the ADDR flag*

*Wait for the OVR flag to be set*

*Clear OVR and write the first data byte.*

*Then the time window for writing the next data byte will be the time to transfer one byte. In this case, the master must discard the first received data byte.*



## 2.10.7 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 will be 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.

## 2.11 SPI peripheral

### 2.11.1 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)

### 2.11.2 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.

## 2.12 I2S peripheral

### 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 as 16-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 will be 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 USART peripheral

### 2.13.1 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 will be 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.

### 2.13.2 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.

### 2.13.3 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.

### 2.13.4 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.

### 2.13.5 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.

### 2.13.6 nRTS signal abnormally driven low after a protocol violation

#### Description

When RTS hardware flow control is enabled, the nRTS signal goes high when data is received. If this data was not read and new data is sent to the USART anyway (protocol violation), the nRTS signal goes back to low level at the end of this new data. Consequently, the sender gets wrong information that the USART is ready to receive further data which is not really the case. On the USART side, an overrun is detected which indicates that some data has been lost.

#### Workarounds

The workarounds below are needed only if another UART device violates the protocol. In most of the systems (no limitation on another device), the USART works fine and no workaround is needed.

1. After data reception and before reading the data in the data register, the software takes over the control of the nRTS as a GPIO and keeps it high as long as needed. If the USART knows that further reception from another device may be discarded, the software keeps the nRTS pin at high level. The software should then release the nRTS pin when the USART is ready to continue reception.
2. Ensure that the received data is always read in less than the duration of the 2nd data reception. One solution would be to treat all the receptions by DMA.

## 2.14 Timers

These limitations apply only to TIM1, TIM2, TIM3, TIM4 and TIM5

### 2.14.1 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 will be detected by the EXTI peripheral.

### 2.14.2 Overcapture detected too early

#### Description

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

#### Conditions

If a 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.

### 2.14.3 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.

## 2.15 IWDG peripheral

### 2.15.1 RVU and PVU flags are not reset in Stop mode

#### Description

The RVU and PVU flags within the IWDG\_SR register are set by hardware next to a write access to the IWDG\_RLR or the IWDG\_PR registers, respectively. If the write access is immediately followed by an entry into Stop mode, the RVU and PVU flags are not reset by hardware. When the user's application needs to write a second time to the IWDG\_RLR or the IWDG\_PR registers, respectively, it has to wait for the RVU and PVU flags to be reset. However, after exiting Stop mode the RVU and PVU bits are not reset. Therefore, the user's application stays in an infinite loop and the IWDG generates a reset after the programmed timeout period.

#### Workaround

The user's application has to wait until RVU and PVU flags within the IWDG\_SR register are reset before entering Stop mode.

## 2.16 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  $\mu$ s should be added.

## 2.17 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 >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 >1 or by PLL2.

#### Workaround

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

## 2.18 OTG\_FS

### 2.18.1 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  $2 \times \text{MPSIZ} + 2 \times \text{data status entries}$ .
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.

### 2.18.2 OTG host blocks the receive channel when receiving IN packets and no TxFIFO 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 support 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 always configure enough TxFIFO space for OUT endpoints.

### 2.18.3 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.



## 2.18.4 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's DAD and PFIVL bitfields just after programming them.

## 2.18.5 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.

## 2.19 Ethernet MAC

### 2.19.1 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.

### **2.19.2 Possible 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 will be 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.

### **2.19.3 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.

## 2.19.4 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 been 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.

## 2.19.5 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.

## 2.19.6 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.

## 2.19.7 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).

## 2.19.8 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 will accept IPv6 frames with invalid Hop-by-Hop extension headers. As a consequence, it will accept 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.

## 2.19.9 MAC stuck in the Idle state on receiving the Tx FIFO flush command exactly 1 clock cycle after a transmission completes

### Description

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

However, if the Tx FIFO read controller receives the Tx FIFO 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 Tx FIFO. The system can recover from this state only with a reset (e.g. a soft reset).

### Workaround

Do not use the Tx FIFO flush feature.

If Tx FIFO flush is really needed, wait until the Tx FIFO is empty prior to using the Tx FIFO flush command.

### 2.19.10 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.

### 2.19.11 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 will be not able 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.

## 2.20 Bootloader limitations

A full description of the bootloader limitations and workarounds is available in the AN2606 in the subsection "Bootloader version" for STM32F105xx and STM32F107xx devices.

## Appendix A Revision and date codes on device marking

Figure 2, Figure 3 and Figure 4 show the marking compositions for the LQFP100 and LQFP64 packages, respectively. The only fields shown are the Additional field containing the revision code and the Year and Week fields making up the date code.

Figure 2. LFBGA100 top package view

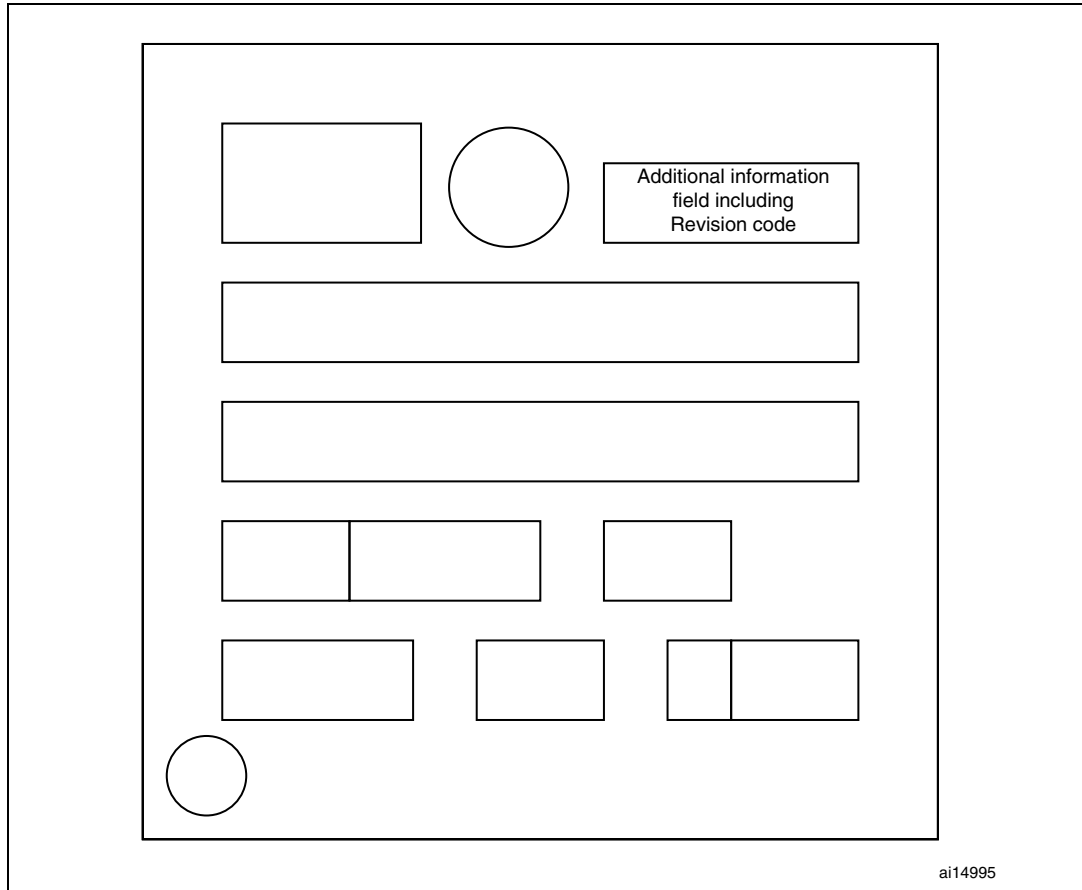


Figure 3. LQFP100 top package view

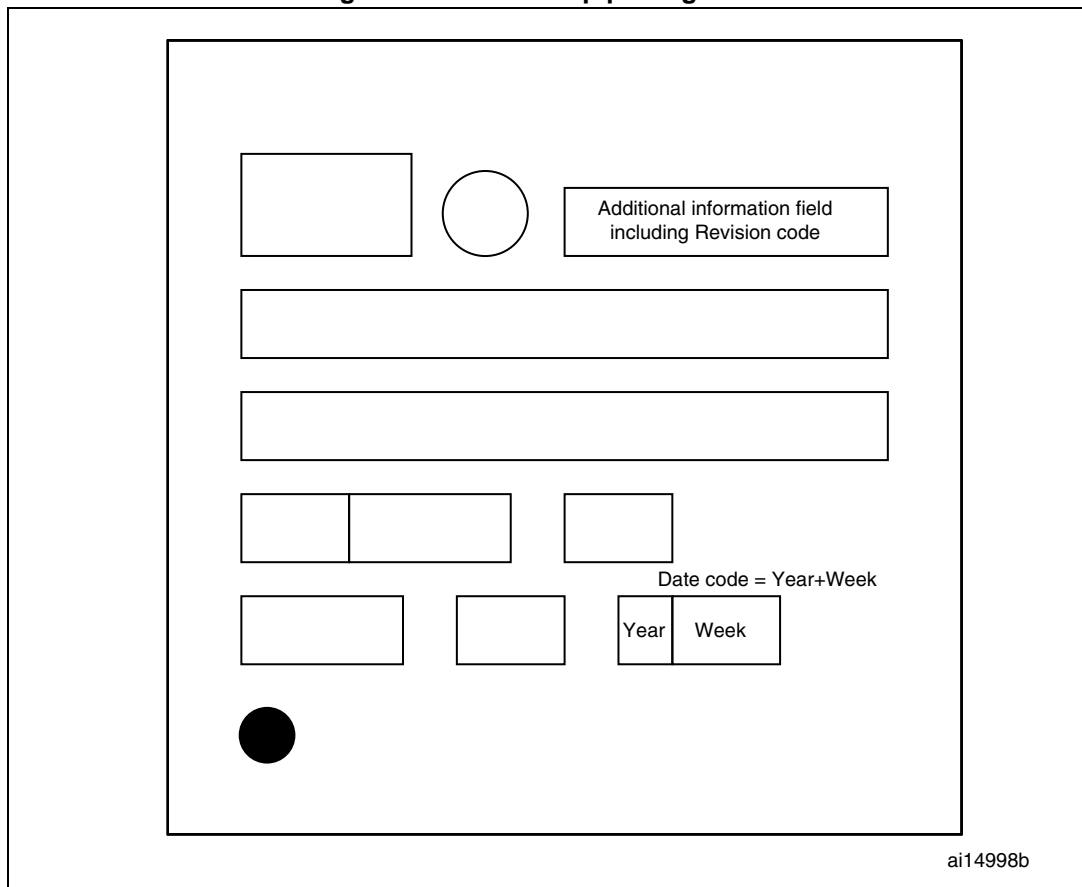
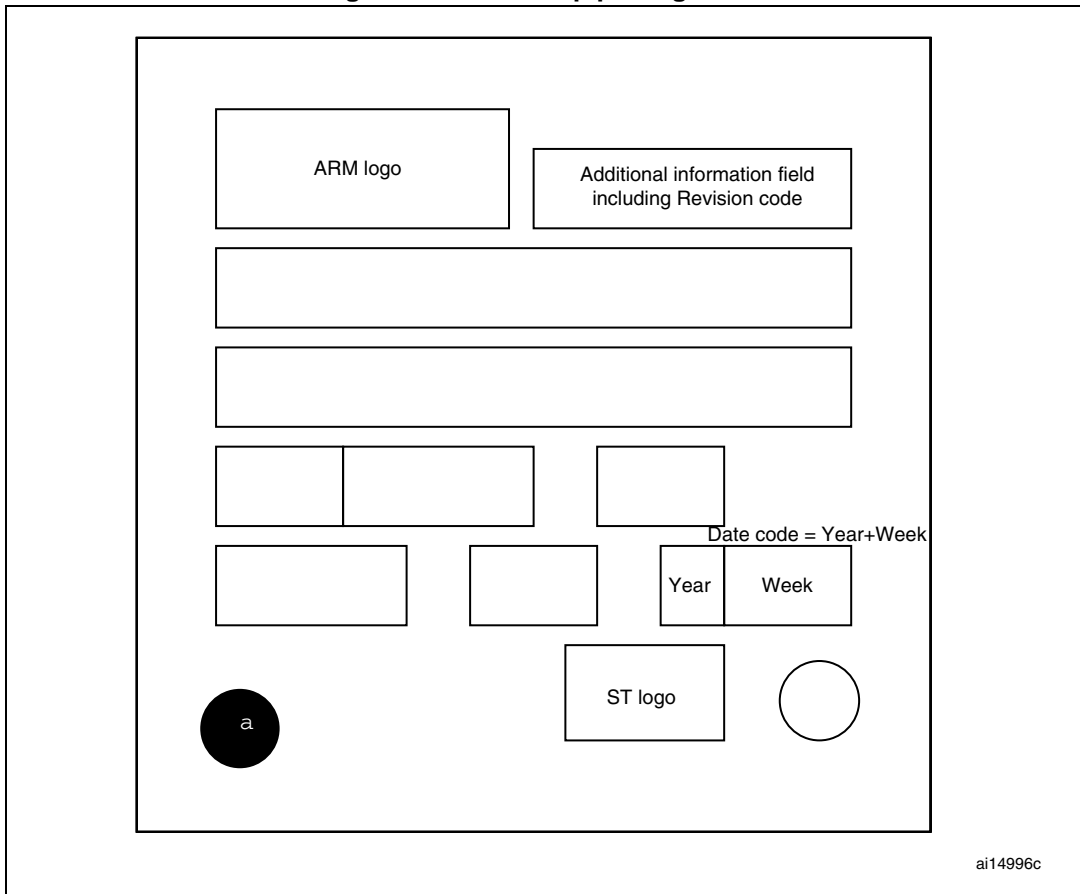




Figure 4. LQFP64 top package view



## Revision history

**Table 5. Document revision history**

Date	Revision	Changes
16-Jun-2009	1	Initial release.
16-Oct-2009	2	<p><i>Section 2.7.5: I2S2 in master/slave mode and Ethernet/USART3 in synchronous mode</i> added.</p> <p><i>Table 4: Summary of silicon limitations in revision Z devices</i> modified.</p> <p><i>Section 2.18: OTG_FS</i> added.</p> <p><i>Section 2.19: Ethernet MAC</i> added.</p> <p><i>Section 2.20: Bootloader limitations</i> added.</p> <p><i>Figure 3: LQFP100 top package view</i> and <i>Figure 4: LQFP64 top package view</i> updated to show the Date code.</p>
15-Dec-2009	3	<p>Added limitations:</p> <ul style="list-style-type: none"> <li>– <i>Section 2.7.6: USARTx_TX pin usage</i></li> <li>– <i>Section 2.6.3: Start cannot be generated after a misplaced Stop</i></li> <li>– <i>Section 2.10.5: Mismatch on the “Setup time for a repeated Start condition” timing parameter</i></li> <li>– <i>Section 2.10.6: Data valid time (<math>t_{VD;DAT}</math>) violated without the OVR flag being set</i></li> <li>– <i>Section 2.11.1: CRC still sensitive to communication clock when SPI is in slave mode even with NSS high</i></li> </ul>
29-Mar-2010	4	<p><i>Section 2.18.2: USART bootloader: incorrect protocol version returned by the Get Version command</i> added.</p>
18-Jun-2010	5	<p>Added <i>Section 2.4: Debugging Stop mode with WFE entry</i></p> <p>Added <i>Section 2.4: Debugging Stop mode with WFE entry</i></p> <p>Added <i>Section 2.10.2: Wrong data read into data register</i></p> <p>Modified <i>Section 2.10.4: Wrong behavior of I2C peripheral in master mode after a misplaced Stop</i></p> <p>Modified <i>Section 2.10.6: Data valid time (<math>t_{VD;DAT}</math>) violated without the OVR flag being set</i></p> <p>Added <i>Section 2.13: USART peripheral</i></p> <p>Added <i>Figure 2: LFBGA100 top package view</i></p>
06-Jan-2011	6	<p>Added <i>Section 2.12.1: Wrong WS signal generation in 16-bit extended to 32-bit PCM long synchronisation mode</i></p> <p>Added <i>Section 2.12.2: In I2S slave mode, WS level must be set by the external master when enabling the I2S</i></p> <p>Added <i>Section 2.12.3: I2S slave mode desynchronisation with the master during communication</i></p> <p>Added <i>Section 2.13.6: nRTS signal abnormally driven low after a protocol violation</i></p> <p>Added <i>Section 2.18.5: Minimum AHB frequency to guarantee correct operation of USB OTG FS peripheral</i></p> <p>Added <i>Section 2.19.11: Ethernet DMA not working after WFI/WFE instruction</i></p>

**Table 5. Document revision history (continued)**

Date	Revision	Changes
07-Feb-2011	7	<p>Updated workarounds in <i>Section 2.10.1: Some software events must be managed before the current byte is being transferred</i> and <i>Section 2.10.2: Wrong data read into data register</i></p> <p>Updated <i>Section 2.12.2: In I2S slave mode, WS level must be set by the external master when enabling the I2S</i></p>
04-Jan-2012	8	<p><i>Table 3: Cortex-M3 core limitations and impact on microcontroller behavior</i>: added limitations 752419 and 740455.</p> <p>Added <i>Section 1.1.5: Interrupted loads to the SP can cause erroneous behaviour</i> and <i>Section 1.1.6: SVC and BusFault/MemManage may occur out of order</i>.</p> <p>Updated “workaround” of <i>Section 2.13.6: nRTS signal abnormally driven low after a protocol violation</i></p> <p>Added <i>.Section 2.15.1: RVU and PVU flags are not reset in Stop mode</i>.</p> <p>Created <i>Section 2.20: Bootloader limitations</i> and removed previous <i>Section 2.18</i> and <i>Section 2.19</i>.</p>
07-Oct-2013	9	<p>Added:</p> <ul style="list-style-type: none"> <li>– <i>Section 2.5: Wakeup sequence from Standby mode when using more than one wakeup source</i></li> <li>– <i>Section 2.6: LSE start-up in harsh environments</i></li> <li>– <i>Section 2.10.7: I2C analog filter may provide wrong value, locking BUSY flag and preventing master mode entry</i></li> <li>– <i>Section 2.11.2: 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</i></li> </ul> <p>Updated <i>Table 4: Summary of silicon limitations in revision Z devices</i>.</p>

**Please Read Carefully:**

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

**UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.**

**ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER'S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR "AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL" INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.**

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2013 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

[www.st.com](http://www.st.com)