

## Silicon identification

This errata sheet applies to the revision A, B and Z of STMicroelectronics STM32L071x8/B/Z microcontrollers.

The STM32L071x8/B/Z devices feature an ARM® 32-bit Cortex®-M0+ core.

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

- by the revision code marked below the order code on the device package
- by the last three digits of the Internal order code printed on the box label

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

Order code	Revision code marked on device <sup>(2)</sup>
STM32L071x8/B/Z	"A" (engineering samples), "B" and "Z"

1. The REV\_ID bits in the DBGMCU\_IDCODE register show the revision code of the device (see the STM32L0x2 reference manual for details on how to find the revision code).
2. Refer to the device datasheet for details on how to identify the revision code and the date code on the different packages.

**Table 2. Device summary**

Reference	Part number
STM32L071x8	STM32L071C8, STM32L071V8, STM32L071K8
STM32L071xB	STM32L071CB, STM32L071VB, STM32L071RB, STM32L071KB
STM32L071xZ	STM32L071CZ, STM32L071VZ, STM32L071RZ, STM32L071KZ

# Contents

<b>1</b>	<b>ARM 32-bit Cortex-M0+ limitations</b>	<b>5</b>
<b>2</b>	<b>STM32L071x8/B/Z silicon limitations</b>	<b>6</b>
2.1	System limitations	7
2.1.1	Delay after an RCC peripheral clock enabling	7
2.1.2	BFB2 bit does not select boot from Bank 2 system memory	8
2.1.3	Interrupts masked when using dual-bank boot mechanism	10
2.1.4	PE2 AF2 alternate function (TIM3_ETR) not available	10
2.1.5	PB4 not available on UFQFPN32 package	10
2.1.6	Flash memory wakeup issue when waking up from Stop or Sleep with Flash in power-down mode	11
2.1.7	Unexpected system reset when waking up from Stop mode with regulator in low-power mode	11
2.1.8	Electrical sensitivity characteristics	12
2.1.9	I2C and USART cannot wake up the device from Stop mode	12
2.1.10	LDM, STM, PUSH and POP not allowed in IOPORT bus	12
2.1.11	BOOT_MODE bits do not reflect the selected boot mode	13
2.2	Comparator limitations	13
2.2.1	COMP1_CSR and COMP2_CSR lock bit reset by SYSCFGRST bit in RCC_APB2RSTR register	13
2.2.2	Output of comparator 2 cannot be internally connected to input 1 of low-power timer	13
2.3	RTC limitations	13
2.3.1	Spurious tamper detection when disabling the tamper channel	13
2.3.2	Detection of a tamper event occurring before enabling the tamper detection is not supported in edge detection mode	14
2.4	I <sup>2</sup> C peripheral limitations	14
2.4.1	Wrong behaviors in Stop mode when waking up from Stop mode is disabled in I <sup>2</sup> C peripheral	14
2.4.2	Wrong data sampling when data set-up time ( $t_{SU;DAT}$ ) is smaller than one I2CCLK period	15
2.5	SPI/I2S peripheral limitations	15
2.5.1	In I2S slave mode, WS level must to be set by the external master when enabling the I2S peripheral	15
2.5.2	BSY bit may stay high at the end of a SPI data transfer in slave mode	16

2.5.3	Last data bit or CRC calculation may be corrupted for the data received in master mode depending on the feedback communication clock timing with respect to the APB clock (SPI or I2S) . . . . .	16
2.5.4	Limited SPI frequency when peripheral configured in Master reception or in Slave transmission mode and VDD is below 2.7 V . . . . .	17
2.6	USART limitations . . . . .	17
2.6.1	Start bit detected too soon when sampling for NACK signal from the smartcard . . . . .	17
2.6.2	Break request can prevent the Transmission Complete flag (TC) from being set . . . . .	18
2.6.3	nRTS is active while RE or UE = 0 . . . . .	18
3	Revision history . . . . .	19

List of tables

Table 1. Device identification ..... 1

Table 2. Device summary ..... 1

Table 3. Summary of silicon limitations ..... 6

Table 4. Document revision history ..... 19



## 1 **ARM 32-bit Cortex-M0+ limitations**

There are not limitations related to the ARM Cortex-M0+ core.

## 2 STM32L071x8/B/Z silicon limitations

[Table 3](#) gives quick references to all documented limitations.

Legend for [Table 3](#): A = workaround available; N = no workaround available; P = partial workaround available, '-' and grayed = fixed.

**Table 3. Summary of silicon limitations**

Links to silicon limitations		Revision A (samples)	Rev B	Rev Z
Section 2.1: System limitations	<a href="#">Section 2.1.1: Delay after an RCC peripheral clock enabling</a>	A	A	A
	<a href="#">Section 2.1.2: BFB2 bit does not select boot from Bank 2 system memory</a>	A	A	-
	<a href="#">Section 2.1.3: Interrupts masked when using dual-bank boot mechanism</a>	A	A	A
	<a href="#">Section 2.1.4: PE2 AF2 alternate function (TIM3_ETR) not available</a>	A	-	-
	<a href="#">Section 2.1.5: PB4 not available on UFQFPN32 package</a>	N	-	-
	<a href="#">Section 2.1.6: Flash memory wakeup issue when waking up from Stop or Sleep with Flash in power-down mode</a>	A	-	-
	<a href="#">Section 2.1.7: Unexpected system reset when waking up from Stop mode with regulator in low-power mode</a>	A	-	-
	<a href="#">Section 2.1.8: Electrical sensitivity characteristics</a>	N	N	N
	<a href="#">Section 2.1.9: I2C and USART cannot wake up the device from Stop mode</a>	N	N	-
	<a href="#">Section 2.1.10: LDM, STM, PUSH and POP not allowed in IOPORT bus</a>	N	N	N
	<a href="#">Section 2.1.11: BOOT_MODE bits do not reflect the selected boot mode</a>	N	N	-
Section 2.2: Comparator limitations	<a href="#">Section 2.2.1: COMP1_CSR and COMP2_CSR lock bit reset by SYSCFGRST bit in RCC_APB2RSTR register</a>	N	N	N
	<a href="#">Section 2.2.2: Output of comparator 2 cannot be internally connected to input 1 of low-power timer</a>	A	-	-
Section 2.3: RTC limitations	<a href="#">Section 2.3.1: Spurious tamper detection when disabling the tamper channel</a>	N	N	N
	<a href="#">Section 2.3.2: Detection of a tamper event occurring before enabling the tamper detection is not supported in edge detection mode</a>	A	A	A

Table 3. Summary of silicon limitations (continued)

Links to silicon limitations		Revision A (samples)	Rev B	Rev Z
Section 2.4: I2C peripheral limitations	Section 2.4.1: Wrong behaviors in Stop mode when waking up from Stop mode is disabled in I2C peripheral	A	A	A
	Section 2.4.2: Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period	A	A	A
Section 2.5: SPI/I2S peripheral limitations	Section 2.5.1: In I2S slave mode, WS level must to be set by the external master when enabling the I2S peripheral	A	A	A
	Section 2.5.2: BSY bit may stay high at the end of a SPI data transfer in slave mode	A	A	A
	Section 2.5.3: Last data bit or CRC calculation may be corrupted for the data received in master mode depending on the feedback communication clock timing with respect to the APB clock (SPI or I2S)	A	A	A
	Section 2.5.4: Limited SPI frequency when peripheral configured in Master reception or in Slave transmission mode and VDD is below 2.7 V	N	-	-
Section 2.6: USART limitations	Section 2.6.1: Start bit detected too soon when sampling for NACK signal from the smartcard	N	N	N
	Section 2.6.2: Break request can prevent the Transmission Complete flag (TC) from being set	A	A	A
	Section 2.6.3: nRTS is active while RE or UE = 0	A	A	A

## 2.1 System limitations

### 2.1.1 Delay after an RCC peripheral clock enabling

#### Description

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

This delay depends on the peripheral mapping:

- If the peripheral is mapped on AHB: the delay should be equal to 1 AHB clock cycle after the clock enable bit is set in the hardware register.  
For I/O peripheral, the delay should be equal to 1 AHB clock cycle after the clock enable bit is set in the hardware register (only applicable to write accesses).
- If the peripheral is mapped on APB: No delay is necessary (no limitation).

### Workarounds

1. Enable the peripheral clock some time before the peripheral read/write register is required.
2. For AHB peripheral (including I/O), insert a dummy read operation to the corresponding register.

## 2.1.2 BFB2 bit does not select boot from Bank 2 system memory

### Description

The feature which allows booting either from Bank 2 or from Bank 1 depending on BFB2 bit status (bit 23 in FLASH\_OTPR register) does not work properly. When BFB2 user option bit is set, the code behavior after reset is incorrect.

### Workaround

BFB2 user option bit must be kept at '0'. The user code stored in Bank 1 should jump to System memory if it requires a boot from Bank 2 and intends to benefit from bank swapping mechanism. Otherwise the code in Bank 1 can jump directly to Bank 2 when required.

Below a proposal of code that uses a value stored in EEPROM to define if the code to be executed is located in Bank 1 or 2. This code can be used in both banks and the EEPROM data must be the same in both EEPROM bank. This code shall be placed at the beginning of *SystemInit()* function:

```
#define DATA_VALUE_SELECT          0x08080000 /* Location where to store
the value allowing to boot on Bank2 and Bank1 */

#define DATA_BANK2_SELECTED        0x37      /* This value indicates that
boot from Bank2 is required */

#define DATA_BANK1_SELECTED        0x73      /* This value indicates that
boot from Bank1 is required */

/* Non user defines: these values must not be modified by user */
#define BANK2_START_ADDRESS_192KB  0x08018000 /* Bank 2 Start Address
(192KB) */
#define BANK2_START_ADDRESS_128KB  0x08010000 /* Bank 2 Start Address
(128KB) */
#define FLASH_START_ADDRESS         0x08000000 /* Flash Start Address
(Bank1) */

#define ENGI_SALESTYPE_ADDRESS      0x1FF80034
#define ENGI_SALESTYPE_MASK         0x00000C00
#define ENGI_SALESTYPE_192KB        0x00000000
#define ENGI_SALESTYPE_128K         0x00000400
#define ENGI_SALESTYPE_64K          0x00000C00

#define SALESTYPE_192KB_DB           0x11001100
#define SALESTYPE_128K_DB            0x22002200
#define SALESTYPE_64K_SB             0x33003300

#define SYSCFG_MEMRMP_ADDRESS        0x40010000
```



```

#define SYSCFG_MEMMODE_MASK          0x00000003
#define SYSCFG_MEMMODE_SYSMEM        0x00000001
#define SYSCFG_FBMODE_MASK           0x00000008

void SystemInit (void)
{ void (*start_of_bootloader)(void);
  start_of_bootloader = (void(*) (void))*(unsigned int*)0x1FF00004; /*
0x1FF00004 is the reset vector of the system flash*/

  /* Check what is the device salestype (192KB, 128KB or 64KB) */
  if ((((__IO uint32_t *) (ENGI_SALESTYPE_ADDRESS)) &
ENGI_SALESTYPE_MASK) == ENGI_SALESTYPE_192KB)
  {
    /* If 192KB salestype */
    SalesType = SALESTYPE_192KB_DB;
  }
  else if (( (__IO uint32_t *) (ENGI_SALESTYPE_ADDRESS)) &
ENGI_SALESTYPE_MASK) == ENGI_SALESTYPE_128K)
  {
    /* If 128KB salestype */
    SalesType = SALESTYPE_128K_DB;
  }
  else
  {
    /* If 64KB salestype */
    SalesType = SALESTYPE_64K_SB;
  }
/* Enable the SYSCFG APB Clock */
*(__IO uint32_t *) 0x40021034 |= ((uint32_t)0x00000001);

/***** Get salestype *****/

/* Manage case when single bank salestypes are used */
if (SalesType != SALESTYPE_64K_SB)
{
  /* Get bank2 start address depending on the package */
  Bank2Addr = (SalesType == SALESTYPE_192KB_DB)?
BANK2_START_ADDRESS_192KB : BANK2_START_ADDRESS_128KB;

  if (((SYSCFG->CFGR1 & SYSCFG_FBMODE_MASK) ==0))
  { /* we are in bank1*/
/***** Check Data value *****/
    /* If data value selected to remap Bank2 */
    if ((__IO uint8_t *)DATA_VALUE_SELECT == DATA_BANK2_SELECTED)
    {

```

```
        if (*(unsigned int*)Bank2Addr>0x20000000 && *(unsigned
int*)Bank2Addr<=0x20005004 )
        {
            start_of_bootloader();
        }
    }
}
....
}
```

### 2.1.3 Interrupts masked when using dual-bank boot mechanism

#### Description

When the dual-bank boot mechanism is used, either by setting BFB2 bit in FLASH\_OTPR register or by using the workaround described in [Section 2.1.2](#), the microcontroller starts executing the user code with bit 0 of the core Priority Mask register (PRIMASK) set. As a result, the interrupts are not serviced.

#### Workaround

Set bit 0 of PRIMASK register before activating any interrupt. See code below as an example:

```
__ASM ("CPSIE i");
```

### 2.1.4 PE2 AF2 alternate function (TIM3\_ETR) not available

#### Description

When PE2 is configured as an alternate function 2 (AF2) to receive TIM3\_ETR, incoming data can be corrupted due to the fact that the output logic of the same PAD is not properly disabled.

#### Workaround

Use another GPIO (PD2 AF2) for TIM3\_ETR

### 2.1.5 PB4 not available on UFQFPN32 package

#### Description

PB4 GPIO does not exist on UFQFPN32 package. As a result, PB4 cannot be used.

#### Workaround

None.

### 2.1.6 Flash memory wakeup issue when waking up from Stop or Sleep with Flash in power-down mode

#### Description

When an external wakeup event (EXTI) occurs in a narrow time window around low-power mode entry (Stop or Sleep mode with Flash memory in power-down state), the Flash wakeup time may be increased. As a result, the first data read or instruction fetch from Flash may be incorrect.

**The probability that this issue occurs is very low since it may happen only during a very narrow time window.**

#### Workaround

Three workarounds are available:

- Do not put the Flash memory module in power-down mode when entering Sleep or Low-power sleep modes.
- Before entering Stop mode by executing a WFI instruction from RAM, set the RUN\_PD bit in the FLASH\_ACR register. After exiting from Stop mode, the Flash memory is automatically powered ON and you can resume program execution from Flash memory. After wakeup, clear the RUN\_PD bit.
- Before entering Stop mode by executing WFI instruction from RAM, set the RUN\_PD bit in the FLASH\_ACR register and set the DS\_EE\_KOFF bit in PWR\_CR register. After resuming from STOP mode, the Flash memory stays in power-down mode. Wake-up the Flash memory by clearing FLASH\_ACR\_RUN\_PD bit and return to code execution.

### 2.1.7 Unexpected system reset when waking up from Stop mode with regulator in low-power mode

#### Description

When the device returns to Run mode after waking up from Stop mode while the internal voltage regulator is configured to switch to low-power mode in Stop mode (LPSSDR=1 in PWR\_CR register), an unexpected system reset may occur if the following conditions are met:

- The internal regulator is set to Range 2 or Range 3 before entering Stop mode.
- $V_{DD}$  power supply is below 2.7 V.

**The probability that this issue occurs is very low since it may happen only for very narrow supply voltage windows which vary from one device to another.**

This reset is internal only and does not affect the NRST pin state and the flags in the Control/status register (RCC\_CSR).

#### Workaround

Two workarounds are possible:

- Enter Stop mode with the internal voltage regulator set to main mode (LPSSDR=0 in PWR\_CR).
- Set the internal voltage regulator to Range1 before entering Stop mode.

### 2.1.8 Electrical sensitivity characteristics

#### Description

The ESD Absolute maximum ratings are lower compared to some other STM32 devices:

- Electrostatic discharge voltage human body model ( $V_{ESD(HBM)}$ ) class is 1C instead of 2 and the maximum value is 1000 V instead of 2000 V.
- Electrostatic discharge voltage charge device model ( $V_{ESD(CDM)}$ ) class is C3 and the maximum value is 250 V instead of 500 V.

#### Workaround

None.

### 2.1.9 I2C and USART cannot wake up the device from Stop mode

#### Description

When the microcontroller is in Stop mode with the regulator in low-power mode, an unexpected system reset may occur if the I2C or the USART attempts to wake up the device.

This limitation also impacts LPUART when the HSI16 is used as clock source instead of LSE.

This reset is internal only and does not affect the NRST pin state and the flags in the Control/status register (RCC\_CSR).

The lower the  $V_{DD}$  value, the more often this unpredictable behavior may occur.

#### Workaround

No workaround is available.

It is recommended to avoid using the USART and I2C wakeup from Stop mode features. To disable them, keep WUPEN bit in I2C\_CR1 and UESM bit in USARTx\_CR1 at '0'.

Two solutions are then possible to perform I2C or USART communications:

- Put the microcontroller in a mode different from Stop (or Standby mode) before initiating communications.
- Replace Stop mode with Stop mode plus regulator in main mode by keeping LPSSDR bit of PWR\_CR set to '0'.

### 2.1.10 LDM, STM, PUSH and POP not allowed in IOPORT bus

#### Description

The instructions Load Multiple (LM), Store Multiple (STM), PUSH and POP fail when the address points to the IOPORT bus memory area (address range = 0x5XXX XXXX).

#### Workaround

None.

### 2.1.11 **BOOT\_MODE bits do not reflect the selected boot mode**

#### **Description**

The BOOT\_MODE[1:0] bits of the SYSCFG\_CFGR1 register remain set to '0' while they should reflect the boot mode selected by the boot pins.

#### **Workaround**

None.

## 2.2 **Comparator limitations**

### 2.2.1 **COMP1\_CSR and COMP2\_CSR lock bit reset by SYSCFGRST bit in RCC\_APB2RSTR register**

#### **Description**

When the SYSCFGRST bit of RCC\_APB2RSTR register is set, the COMP1\_CSR and COMP2\_CSR register contents are reset even if COMP1LOCK and COMP2LOCK bits are set in COMP1\_CSR and the COMP2\_CSR register, respectively.

#### **Workaround**

No workaround is available.

For security reasons, it is recommended to avoid using SYSCFGRST bit of RCC\_APB2RSTR when COMP1LOCK and/or COMP2LOCK bits are set.

### 2.2.2 **Output of comparator 2 cannot be internally connected to input 1 of low-power timer**

#### **Description**

The COMP2LPTIMIN1 bit (bit 13 of COMP2\_CSR register) which internally connects COMP2VALUE to the low-power timer (LPTIM) input 1 has no effect.

#### **Workaround**

Connect COMP2\_OUT output to an external pin and configure LPTIM\_IN1 on an external pin, then connect both pins together externally.

## 2.3 **RTC limitations**

### 2.3.1 **Spurious tamper detection when disabling the tamper channel**

#### **Description**

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

**Workaround**

None

**2.3.2 Detection of a tamper event occurring before enabling the tamper detection is not supported in edge detection mode****Description**

When the tamper detection is enabled in edge detection mode (TAMPFLT=00):

- When TAMPxTRG=0 (rising edge detection): if the tamper input is already high before enabling the tamper detection, the tamper event may or may not be detected when enabling the tamper detection. The probability to detect it increases with the APB frequency.
- When TAMPxTRG=1 (falling edge detection): if the tamper input is already low before enabling the tamper detection, the tamper event is not detected when enabling the tamper detection.

**Workaround**

The I/O state should be checked by software in the GPIO registers, just after enabling the tamper detection and before writing sensitive values in the backup registers, in order to ensure that no active edge occurred before enabling the tamper event detection.

**2.4 I<sup>2</sup>C peripheral limitations****2.4.1 Wrong behaviors in Stop mode when waking up from Stop mode is disabled in I<sup>2</sup>C peripheral****Description**

When wakeup from Stop mode is disabled in the I<sup>2</sup>C interface (WUPEN = 0) and the microcontroller enters Stop mode while a transfer is ongoing on the bus, some wrong behavior may happen:

1. The BUSY flag can be wrongly set when the microcontroller exits Stop mode. This prevents from initiating a transfer in master mode, as the START condition cannot be sent when BUSY is set.
2. If clock stretching is enabled (NOSTRETCH = 0), the I<sup>2</sup>C clock SCL may be kept low by the I<sup>2</sup>C as long as the microcontroller remains in Stop mode. This limitation may occur when Stop mode is entered during the address phase of a I<sup>2</sup>C bus transfer while SCL = 0. Therefore the transfer may be stalled as long as the microcontroller is in Stop mode. The probability that this issue occurs depends also on the timings configuration, the peripheral clock frequency and the I<sup>2</sup>C bus frequency.

These behaviors can occur in Slave mode and in Master mode in a multi-master topology.

**Workaround**

Disable the I<sup>2</sup>C interface (PE=0) before entering Stop mode and enable it again in Run mode.

## 2.4.2 Wrong data sampling when data set-up time ( $t_{\text{SU;DAT}}$ ) is smaller than one I2CCLK period

### Description

The I2C bus specification and user manual specifies a minimum data set-up time ( $t_{\text{SU;DAT}}$ ) at:

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

The I2C SDA line is not correctly sampled when  $t_{\text{SU;DAT}}$  is smaller than one I2CCLK (I2C clock) period: the previous SDA value is sampled instead of the current one. This can result in a wrong slave address reception, a wrong received data byte, or a wrong received acknowledge bit.

### Workaround

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

## 2.5 SPI/I2S peripheral limitations

### 2.5.1 In I2S slave mode, WS level must to be set by the external master when enabling the I2S peripheral

#### Description

In slave mode, the WS signal level is used only to start communications. If the I2S (in slave mode) is enabled while the master is already sending the clock, and the WS signal level is either low (for I2S protocol) or 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

Enable the I2S peripheral 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.

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

### Description

In slave mode, BSY bit is not reliable to handle the end of data frame transaction due to some bad synchronization between the CPU clock and external SCK clock provided by master. Sporadically, the BSY bit is not cleared at the end of a data frame transfer. As a consequence, it is not recommended to rely on BSY bit before entering low-power mode or modifying the SPI configuration (e.g. direction of the bidirectional mode).

### Workaround

- When the SPI interface is in receive mode, the end of a transaction with the master can be detected by the corresponding RXNE event when this flag is set after the last bit of that transaction is sampled and the received data are stored.
- When the following sequence is used, the synchronization issue does not occur. The BSY bit works correctly and can be used to recognize the end of any transmission transaction (including when RXNE is not raised in bidirectional mode):
  - a) Write the last data into data register.
  - b) Poll TXE flag till it becomes high to make sure the data transfer has started.
  - c) Disable the SPI interface by clearing SPE bit while the last data transfer is on going.
  - d) Poll the BSY bit till it becomes low.

*Note: The second workaround can be used only when the CPU is fast enough to disable the SPI interface after a TXE event is detected while the data frame transfer is ongoing. It cannot be implemented when the ratio between CPU and SPI clock is low and the data frame is particularly short. At this specific case, the timeout can be measured from the TXE event instead by calculating a fixed number of CPU clock cycles corresponding to the time necessary to complete the data frame transaction.*

## 2.5.3 Last data bit or CRC calculation may be corrupted for the data received in master mode depending on the feedback communication clock timing with respect to the APB clock (SPI or I2S)

### Description

When the SPI or I2S interface is configured in master mode, the last transacted bit of the received data may be corrupted if the delay of the internal feedback clock, which is derived from SCK pin, is higher than the APB clock period. In this case, the last bit value is strobed too late into the shift register while its content has already been either copied to the data register or compared to the pattern calculated internally.

When data corruption occurs, the bit position in the data register contains the value of the last bit received during the previous data transfer or the CRC error flag (CRCERR) is asserted in spite of the fact that all data have been correctly received.

This limitation may be observed only when the device is configured in SPI or I2S is master (full-duplex or receiver mode).



The main factors which can increase the above delay, and the risk that this issue occurs, are:

- High external SPI clock capacitive load
- Low SCK I/O output speed
- Low  $V_{DD}$  level
- Extreme temperature

*Note:* SPI communication speed has no impact.

#### Workaround

Set the I/O pad configuration to achieve a faster I/O output speed on SCK pin, regardless the SPI speed. Max SCK line capacitance must be limited below 30pF.

### 2.5.4 Limited SPI frequency when peripheral configured in Master reception or in Slave transmission mode and $V_{DD}$ is below 2.7 V

#### Description

When the SPI is configured in Master reception or in Slave transmission mode, the maximum SPI frequency should be 16 MHz in Range1 ( $V_{DD}$  ranging from 1,71 to 3,6 V). However a timing issue limits the maximum transaction reachable frequency when  $V_{DD}$  is below 2.7 V.

#### Workaround

None.

## 2.6 USART limitations

### 2.6.1 Start bit detected too soon when sampling for NACK signal from the smartcard

#### Description

According to ISO/IEC 7816-3 standard, when a character parity error is incorrect, the smartcard receiver shall transmit a NACK error signal  $10.5 \pm 0.2$  ETUs after the character START bit falling edge. In this case, the USART transmitter should be able to detect correctly the NACK signal by sampling at  $11 \pm 0.2$  ETUs after the character START bit falling edge.

In Smartcard mode, the USART peripheral does not respect the  $11 \pm 0.2$  ETU timing. As a result, when the NACK falling edge occurs 10.68 ETUs or later, the USART may misinterpret this transition as a START bit even if the NACK is correctly detected.

#### Workaround

None

## 2.6.2 Break request can prevent the Transmission Complete flag (TC) from being set

### Description

After the end of transmission of a data (D1), the Transmission Complete (TC) flag will not be set if the following conditions are met:

- CTS hardware flow control is enabled.
- D1 is being transmitted.
- A break transfer is requested before the end of D1 transfer.
- nCTS is deasserted before the end of D1 data transfer.

### Workaround

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

## 2.6.3 nRTS is active while RE or UE = 0

### Description

The nRTS line is driven low as soon as the RTSE bit is set and even if the USART is disabled (UE = 0) or if the receiver is disabled (RE=0) i.e. not ready to receive data.

### Workaround

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

### 3 Revision history

**Table 4. Document revision history**

Date	Revision	Changes
02-Jul-2015	1	Initial release.
14-Oct-2014	2	<p>Changed confidentiality level to public.</p> <p>Added STM32L071C8 on cover page.</p> <p>Updated <a href="#">Section 2.1.2: BFB2 bit does not select boot from Bank 2 system memory</a> and added <a href="#">Section 2.1.3: Interrupts masked when using dual-bank boot mechanism</a>.</p> <p>Added <a href="#">Section 2.1.9: I2C and USART cannot wake up the device from Stop mode</a> and <a href="#">Section 2.1.10: LDM, STM, PUSH and POP not allowed in IOPORT bus</a></p>
05-Feb-2016	3	<p>The following limitations are fixed in device revision Z:</p> <ul style="list-style-type: none"> <li>– <a href="#">BFB2 bit does not select boot from Bank 2 system memory</a></li> <li>– <a href="#">I2C and USART cannot wake up the device from Stop mode</a></li> <li>– <a href="#">BOOT_MODE bits do not reflect the selected boot mode</a> (this limitation has been added in current errata sheet revision).</li> </ul> <p>Added <a href="#">Section 2.2.1: COMP1_CSR and COMP2_CSR lock bit reset by SYSCFGRST bit in RCC_APB2RSTR register</a>.</p> <p>Updated <a href="#">Section 2.5.3: Last data bit or CRC calculation may be corrupted for the data received in master mode depending on the feedback communication clock timing with respect to the APB clock (SPI or I2S)</a>.</p>

**IMPORTANT NOTICE – PLEASE READ CAREFULLY**

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

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

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

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

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

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

© 2016 STMicroelectronics – All rights reserved