STM32L051x6/8 Errata sheet

STM32L051x6/8 device limitations

Silicon identification

This errata sheet applies to the revision A, Z, Y and X of STMicroelectronics STM32L051x6/8 microcontrollers.

The STM32L051x6/8 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(1)

Order code	Revision code marked on device(2)
STM32L051x6/8	“A”, “Z”, “Y”, “X”

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see the STM32L0x1 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
STM32L051x6	STM32L051C6, STM32L051K6, STM32L051R6, STM32L051T6
STM32L051x8	STM32L051C8, STM32L051K8, STM32L051R8, STM32L051T8

Contents

1 ARM 32-bit Cortex-M0+ limitations 5

2 STM32L051x6/8 silicon limitations 6

2.1 System limitations 8

2.1.1 Writing in byte mode to the GPIOx_OTYPER register does not work . . . 8

2.1.2 Exiting Stop mode on a reset event is not possible when HSI16 is the clock system and it is selected as wakeup clock 8

2.1.3 Protection level1 does not work 9

2.1.4 LSE bypass feature cannot be used in Standby mode 9

2.1.5 PA4 and PA5 cannot be redirected to comparator 2 minus input 9

2.1.6 PB14 output speed configuration interferes with PB13 10

2.1.7 ADC transfer curve issue at VREF+/2 when VREF+ < VDDA 11

2.1.8 Delay after an RCC peripheral clock enabling 11

2.1.9 Flash memory wakeup issue when waking up from Stop or Sleep with Flash in power-down mode 12

2.1.10 Unexpected system reset when waking up from Stop mode with regulator in low-power mode 12

2.1.11 I2C and USART cannot wake up the device from Stop mode 13

2.1.12 LDM, STM, PUSH and POP not allowed in IOPORT bus 13

2.1.13 BOOT_MODE bits do not reflect the selected boot mode 13

2.2 ADC peripheral limitation 14

2.2.1 Incorrect first ADC conversion result when delay between two consecutive conversions is too long 14

2.2.2 Overrun flag might not be set when converted data have not been read before new data are written 14

2.3 Comparator limitations 14

2.3.1 COMP1_CSR and COMP2_CSR lock bit reset by SYSCFGRST bit in RCC_APB2RSTR register 14

2.3.2 Output of comparator 2 cannot be internally connected to input 1 of low-power timer 15

2.4 RTC limitations 15

2.4.1 Spurious tamper detection when disabling the tamper channel 15

2.4.2 Detection of a tamper event occurring before enabling the tamper detection is not supported in edge detection mode 15

2.5 I^2C peripheral limitations 16

2.5.1 Wrong behaviors in Stop mode when waking up from Stop mode is disabled in I^2C peripheral 16
2.5.2 Wrong data sampling when data set-up time ($t_{SU;DAT}$) is smaller than one I2CCLK period 16
2.6 SPI/I2S peripheral limitations 17
2.6.1 In I2S slave mode, WS level must to be set by the external master when enabling the I2S peripheral 17
2.6.2 BSY bit may stay high at the end of a SPI data transfer in slave mode . 17
2.6.3 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) 18
2.6.4 CRC may be corrupted when a peripheral connected to the same DMA channel than the SPI completes its DMA transaction 18
2.7 USART limitations 19
2.7.1 Start bit detected too soon when sampling for NACK signal from the smartcard 19
2.7.2 Break request can prevent the Transmission Complete flag (TC) from being set 19
2.7.3 nRTS is active while RE or UE = 0 19

3 Revision history 20

List of tables

Table 1. Device identification 1
Table 2. Device summary 1
Table 3. Summary of silicon limitations 6
Table 4. Port B output speed configuration 10
Table 5. PB13/PB14 truth table 10
Table 6. Document revision history 20

1 ARM 32-bit Cortex-M0+ limitations

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

2 STM32L051x6/8 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)	Revision Z	Revision Y	Revision X
Section 2.1: System limitations	*Section 2.1.1: Writing in byte mode to the GPIOx_OTYPER register does not work*	A	A	A	A
	Section 2.1.2: Exiting Stop mode on a reset event is not possible when HSI16 is the clock system and it is selected as wakeup clock	A	-	-	-
	Section 2.1.3: Protection level1 does not work	N	-	-	-
	Section 2.1.4: LSE bypass feature cannot be used in Standby mode	N	-	-	-
	Section 2.1.5: PA4 and PA5 cannot be redirected to comparator 2 minus input	N	-	-	-
	Section 2.1.6: PB14 output speed configuration interferes with PB13	N	-	-	-
	Section 2.1.7: ADC transfer curve issue at VREF+/2 when VREF+ < VDDA	N	-	-	-
	Section 2.1.8: Delay after an RCC peripheral clock enabling	A	A	A	A
	Section 2.1.9: Flash memory wakeup issue when waking up from Stop or Sleep with Flash in power-down mode	A	A	-	-
	Section 2.1.10: Unexpected system reset when waking up from Stop mode with regulator in low-power mode	A	A	-	-
	Section 2.1.11: I2C and USART cannot wake up the device from Stop mode	N	N	N	-
	Section 2.1.12: LDM, STM, PUSH and POP not allowed in IOPORT bus	N	N	N	N
	Section 2.1.13: BOOT_MODE bits do not reflect the selected boot mode	N	N	N	-
Section 2.2: ADC peripheral limitation	*Section 2.2.1: Incorrect first ADC conversion result when delay between two consecutive conversions is too long*	A	A	-	-
	Section 2.2.2: Overrun flag might not be set when converted data have not been read before new data are written	A	A	A	A

Table 3. Summary of silicon limitations (continued)

Links to silicon limitations		Revision A (samples)	Revision Z	Revision Y	Revision X
Section 2.3: Comparator limitations	*Section 2.3.1: COMP1_CSR and COMP2_CSR lock bit reset by SYSCFGRST bit in RCC_APB2RSTR register*	N	N	N	N
	Section 2.3.2: Output of comparator 2 cannot be internally connected to input 1 of low-power timer	A	A	-	-
Section 2.4: RTC limitations	*Section 2.4.1: Spurious tamper detection when disabling the tamper channel*	N	N	N	N
	Section 2.4.2: Detection of a tamper event occurring before enabling the tamper detection is not supported in edge detection mode	A	A	A	A
Section 2.5: I^2C peripheral limitations	*Section 2.5.1: Wrong behaviors in Stop mode when waking up from Stop mode is disabled in I^2C peripheral*	A	A	A	A
	Section 2.5.2: Wrong data sampling when data set-up time ($t_{SU;DAT}$) is smaller than one I2CCLK period	A	A	A	A
Section 2.6: SPI/I2S peripheral limitations	*Section 2.6.1: In I2S slave mode, WS level must to be set by the external master when enabling the I2S peripheral*	A	A	A	A
	Section 2.6.2: BSY bit may stay high at the end of a SPI data transfer in slave mode	A	A	A	A
	Section 2.6.3: 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	A
	Section 2.6.4: CRC may be corrupted when a peripheral connected to the same DMA channel than the SPI completes its DMA transaction	A	A	A	A
Section 2.7: USART limitations	*Section 2.7.1: Start bit detected too soon when sampling for NACK signal from the smartcard*	N	N	N	N
	Section 2.7.2: Break request can prevent the Transmission Complete flag (TC) from being set	A	A	A	A
	Section 2.7.3: nRTS is active while RE or UE = 0	A	A	A	A

2.1 System limitations

2.1.1 Writing in byte mode to the GPIOx_OTYPER register does not work

Description

The OTYPER[15:8] bits in GPIOx_OTYPER register cannot be written in byte mode. This is valid for A, B, C, D and H ports.

However, the following operations are possible:

- OTYPER[15:8] bits can be written in half-word and word mode
- OTYPER[7:0] bits can be written in byte, half-word or word mode

Workaround

Program GPIOx_OTYPER bits in half-word or word mode.

2.1.2 Exiting Stop mode on a reset event is not possible when HSI16 is the clock system and it is selected as wakeup clock

Description

The Stop mode can be entered whatever the system clock. The system clock after exiting from Stop mode is selected through the STOPWUCK control bit in RCC_CFGR register:

- when STOPWUCK = 0: the internal MSI oscillator (64 KHz to 4 MHz) is selected.
- when STOPWUCK = 1: the internal HSI16 (16 MHz) oscillator is selected.

The microcontroller is blocked (no system clock is selected) when the following events occur:

- Stop mode is entered when the system clock is HSI16 and STOPWUCK bit is set to 1 (wakeup clock is HSI16)
- and Stop mode is exited when a reset occurs

Only a power-on reset can restart the microcontroller.

Workaround

To avoid blocking the STM32L051x6/8 microcontroller if a reset occurs in Stop mode, select the MSI clock instead of HSI16 before entering Stop mode. Follow the sequence below if HSI16 is selected as system clock, STOPWUCK =1 and a Stop entry is requested by the application:

1. Switch to MSI.
2. Enter Stop mode.
3. When a reset occurs, the microcontroller is reset.
4. Code execution restarts normally when exiting from Stop mode.

2.1.3 Protection level1 does not work

Description

A mass erase is performed when the protection level is changed from level1 to level0. This mass erase consists in the following operations:

- Erasing the data EEPROM and Flash program memory area
- Erasing the protection option byte (RDPROT)
- Programming the targeted level (level0) in RDPROT.

Erase and write operations are not efficient when a mass erase is executed. As a result, the complete erasing of the data EEPROM and Flash program memory is not guaranteed as well as the programming of RDPROT to level0. After several mass erase operations, the RDPROT value is finally programmed with level0 but the total erasing of the data EEPROM and Flash program memory is still not guaranteed.

Note: The other erase and write operations are not impacted by this issue.

Workaround

No workaround available.

2.1.4 LSE bypass feature cannot be used in Standby mode

Description

The external clock selected in LSE bypass mode to drive the OSC32_IN pin is no more effective in Standby mode.

The LSE bypass mode is selected through LSEBYP and LSEON bits of RCC_CSR register.

This issue does not occur when the LSE clock is switched on and configured to be used with an crystal or ceramic resonator (LSEBYP bit = 0 and LSEON bit = 1 in RCC_CSR register).

Workaround

No workaround available.

2.1.5 PA4 and PA5 cannot be redirected to comparator 2 minus input

Description

PA4 and PA5 cannot be redirected to comparator 2 minus input. This is done by setting COMP2_INN_SEL bit of COMP2_CSR register: when COMP2_INN_SEL = 010 or 011 (respectively PA4 or PA5), the negative input of the comparator 2 is left floating.

Only PA2, PB3, VREFINT, 1/2VREFINT, 1/4VREFINT and 3/4VREFINT can be selected as comparator 2 minus input.

Note: The positive input of the comparator 2 is not impacted.

Workaround

No workaround available.

2.1.6 PB14 output speed configuration interferes with PB13

Description

Two GPIOB_OSPEEDR control bits (OSPEEDy[1:0], where y = 0 to 15) can be used to configure port B output speed.These bits are written by software (see *Table 4: Port B output speed configuration*).

Table 4. Port B output speed configuration

OSPEEDRy[1:0] y= 0 to 15	Port output speed
00	Very low speed
01	Low speed
10	Medium speed
11	High speed

When OSSPEED13[1] corresponding to PB13 is modified, OSSPEED14[1] of PB14 also changes. Refer to *Table 5* for the corresponding truth table.

Table 5. PB13/PB14 truth table(1)

OSPEEDR13[1] / OSPEEDR14[0]	PB14 I/O SPEED
00	Very low speed
01	Low speed
10	Medium speed
11	High speed

1. The values in pink shows OSPEED13[1] dependency versus OSPEED14[1].

Workaround

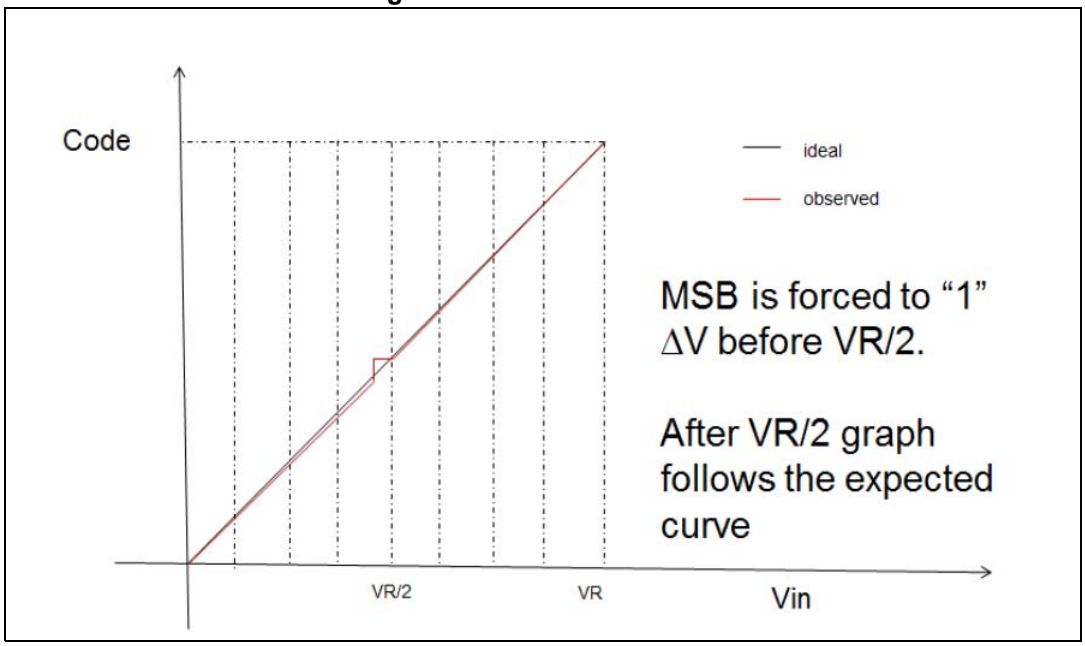
No workaround available.

2.1.7 ADC transfer curve issue at $V_{REF+}/2$ when $V_{REF+} < V_{DDA}$

Description

When V_{REF+} is lower than V_{DDA}, and only in this case, the data MSB is forced to '1' just before V_{IN} reaches $V_{REF+}/2$. As a result, the data transfer curve shows a discrepancy around $V_{REF+}/2$ (see *Figure 1*).

Figure 1. Data transfer curve



Note: The transfer curve behaves as expected when V_{IN} is strictly lower or higher that $V_{REF+}/2$.

2.1.8 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.

2.1.9 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.

2.1.10 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 (LPSDSR=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 (LPSDSR=0 in PWR_CR).
- Set the internal voltage regulator to Range1 before entering Stop mode.

2.1.11 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 LPSDSR bit of PWR_CR set to ‘0’.

2.1.12 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.

2.1.13 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 ADC peripheral limitation

2.2.1 Incorrect first ADC conversion result when delay between two consecutive conversions is too long

Description

When the ADC performs the first conversion or when the delay between two consecutive ADC conversions is longer than 0.5 ms, the result of the conversion may be incorrect. The same issue occurs when the delay between the calibration and the first conversion is longer than 0.5 ms. This issue is independent from the status of ADEN bit in the control register (ADC_CR).

Workaround

For the first conversion, or when the delay between two ADC conversions is longer than the limit specified above, perform two ADC consecutive conversions in single, scan or continuous mode:

1. Dummy conversion of any ADC channel. This conversion should not be taken into account by the application.
2. Conversion of the targeted channel that will be kept as the ADC result. Subsequent conversions do not need particular management unless the ADC conversion stops for more than 0.5 ms.

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

Description

When converted data are read from ADC_DR register during the same APB cycle as data from new conversion are written to this register, the previously written data or the new data are lost, but the overrun flag (OVR) might not set to ‘1’.

Workaround

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

2.3 Comparator limitations

2.3.1 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.

2.3.2 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.

2.4 RTC limitations

2.4.1 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

2.4.2 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.

2.5 I²C peripheral limitations

2.5.1 Wrong behaviors in Stop mode when waking up from Stop mode is disabled in I²C peripheral

Description

When wakeup from Stop mode is disabled in the I²C 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 I²C clock SCL may be kept low by the I²C as long as the microcontroller remains in Stop mode. This limitation may occur when Stop mode is entered during the address phase of a I²C 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 I²C bus frequency.

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

Workaround

Disable the I²C interface (PE=0) before entering Stop mode and enable it again in Run mode.

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

Description

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

2.6 SPI/I2S peripheral limitations

2.6.1 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.

2.6.2 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.

2.6.3 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.

2.6.4 CRC may be corrupted when a peripheral connected to the same DMA channel than the SPI completes its DMA transaction

Description

When the SPI interface is running in master or slave mode and the CRC feature is enabled, the CRC may be frozen and corrupted before the CRCNEXT bit is written. In this case the CRC error flag (CRCERR) is set. This issue occurs when a peripheral, mapped to the same DMA channel than the SPI, performs DMA transfers and reaches end of DMA transaction (n or n-1 event) while the SPI is configured to manage data transfers by software (interrupt or polling mode).

Workaround

When possible, use DMA for SPI transfers.

2.7 USART limitations

2.7.1 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 ± 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 ± 0.2 ETUs after the character START bit falling edge.

In Smartcard mode, the USART peripheral does not respect the 11 ± 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

2.7.2 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.

2.7.3 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 6. Document revision history

Date	Revision	Changes
29-Apr-2014	1	Initial release.
09-Oct-2014	2	Added – *Section 2.2.1: Incorrect first ADC conversion result when delay between two consecutive conversions is too long* – *Section 2.5.2: Wrong data sampling when data set-up time ($t_{SU;DAT}$) is smaller than one I2CCLK period* – *Section 2.6.2: BSY bit may stay high at the end of a SPI data transfer in slave mode*
17-Oct-2014	3	Updated *Section 2.1.8: Delay after an RCC peripheral clock enabling*.
30-Apr-2015	4	Added revision “Y”. Updated *Section 2.1.8: Delay after an RCC peripheral clock enabling*. Added *Section 2.1.9: Flash memory wakeup issue when waking up from Stop or Sleep with Flash in power-down mode*, *Section 2.1.10: Unexpected system reset when waking up from Stop mode with regulator in low-power mode* and *Section 2.3.2: Output of comparator 2 cannot be internally connected to input 1 of low-power timer*. Updated *Section 2.2.1: Incorrect first ADC conversion result when delay between two consecutive conversions is too long*. Added *Section 2.4: RTC limitations*. Updated *Section 2.6.2: BSY bit may stay high at the end of a SPI data transfer in slave mode*. Added *Section 2.6.3: 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)* and *Section 2.6.4: CRC may be corrupted when a peripheral connected to the same DMA channel than the SPI completes its DMA transaction*. Added *Section 2.7: USART limitations*.
11-Feb-2016	5	Added *Section 2.1.11: I2C and USART cannot wake up the device from Stop mode*, *Section 2.1.12: LDM, STM, PUSH and POP not allowed in IOPORT bus* and *Section 2.1.13: BOOT_MODE bits do not reflect the selected boot mode*. Added *Section 2.2.2: Overrun flag might not be set when converted data have not been read before new data are written*. Added *Section 2.3.1: COMP1_CSR and COMP2_CSR lock bit reset by SYSCFGRST bit in RCC_APB2RSTR register* . Updated *Section 2.6.3: 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)*.