STM32L011x3/4 Errata sheet

STM32L011x3/4 device limitations

Silicon identification

This errata sheet applies to the revision ‘A’ and ‘Z’ of STMicroelectronics STM32L011x3/4 microcontrollers.

The STM32L011x3/4 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)
STM32L011x3/4	‘A’ and ‘Z’

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
STM32L011x3/4	STM32L011G3, STM32L011K3, STM32L011E3, STM32L011F3, STM32L011D3, STM32L011G4, STM32L011K4, STM32L011E4, STM32L011F4, STM32L011D4

Contents

List of tables

1 ARM 32-bit Cortex-M0+ limitations

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

2 STM32L011x3/4 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
Section 2.1: System limitations	*Section 2.1.1: Delay after an RCC peripheral clock enabling*	A	A
	Section 2.1.2: I2C and USART cannot wake up the device from Stop mode	N	-
	Section 2.1.3: LDM, STM, PUSH and POP not allowed in IOPORT bus	N	-
	Section 2.1.4: BOOT_MODE bits do not reflect the selected boot mode	N	N
	Section 2.1.5: NSS pin synchronization required when using bootloader with SPI1 interface on TSSOP14 package	A	A
Section 2.2: ADC limitation	*Section 2.2.1: Overrun flag might not be set when converted data have not been read before new data are written*	A	A
Section 2.3: Comparator limitation	*Section 2.3.1: COMP1_CSR and COMP2_CSR lock bit reset by SYSCFGRST bit in RCC_APB2RSTR register*	N	N
Section 2.4: RTC limitation	*Section 2.4.1: Detection of a tamper event occurring before enabling the tamper detection is not supported in edge detection mode*	A	A
Section 2.5: I2C peripheral limitations	*Section 2.5.1: Wrong behaviors in Stop mode when waking up from Stop mode is disabled in I2C peripheral*	A	A
	Section 2.5.2: Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period	A	A
	Section 2.5.3: 10-bit master Master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave	A	A
Section 2.6: SPI peripheral limitations	*Section 2.6.1: BSY bit may stay high at the end of a SPI data transfer in Slave mode*	A	A
	Section 2.6.2: Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback	A	A
	Section 2.6.3: Wrong CRC transmitted in Master mode with delayed SCK feedback	A	A
Section 2.7: USART limitations	*Section 2.7.1: nRTS is active while RE or UE = 0*	A	A
	Section 2.7.2: DMA channel 3 (CH3) not functional when USART2_RX used for data reception	A	A

Table 3. Summary of silicon limitations (continued)

Links to silicon limitations		Revision A (samples)	Revision Z
Section 2.8: LPUART limitation	*Section 2.8.1: DMA channel 5 (CH5) not functional when LPUART1_RX used for data reception*	A	A
Section 2.9: IWDG limitation	*Section 2.9.1: IWDG does not always reset the device*	N	-

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 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’.

2.1.3 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.

2.1.4 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.

2.1.5 NSS pin synchronization required when using bootloader with SPI1 interface on TSSOP14 package

Description

When using the embedded bootloader with SPI1 interface on devices in TSSOP14 package, if the NSS pin is grounded (default status after device reset), SPI communications are not synchronized and the bootloader does not work.

To properly synchronize the SPI interface, an NSS pin falling edge is required before initiating communications.

Workaround

On devices in TSSOP14, toggle with NSS pin (PA14) after device reset.

2.2 ADC limitation

2.2.1 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 limitation

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.

2.4 RTC limitation

2.4.1 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.

2.5.3 10-bit master Master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave

Description

In Master mode, the master automatically sends a STOP bit when the slave has not acknowledged a byte during the address transmission.

In 10-bit addressing mode, if the first part of the 10-bit address (corresponding to 10-bit header + 2 MSB) has not been acknowledged by the slave, the STOP bit is sent but the START bit is not cleared and the master cannot launch a new transfer.

Workaround

When the I2C is configured in 10-bit addressing Master mode and the NACKF status flag is set in the I2C_ISR register while the START bit is still set in I2C_CR2 register, then proceed as follows:

1. Wait for the STOP condition detection (STOPF = 1 in I2C_ISR register).
2. Disable the I2C peripheral.
3. Wait for a minimum of 3 APB cycles.
4. Enable the I2C peripheral again.

2.6 SPI peripheral limitations

2.6.1 BSY bit may stay high at the end of a SPI data transfer in Slave mode

Description

BSY flag may sporadically remain high at the end of a data transfer in Slave mode. The issue appears when an accidental synchronization happens between internal CPU clock and external SCK clock provided by master.

This is related to the end of data transfer detection while the SPI is enabled in Slave mode.

As a consequence, the end of data transaction may be not recognized when software needs to monitor it (e.g. at the end of session before entering the low-power mode or before direction of data line has to be changed at half duplex bidirectional mode). The BSY flag is unreliable to detect the end of any data sequence transaction.

Workaround

When NSS hardware management is applied and NSS signal is provided by master, the end of a transaction can be detected by the NSS polling by slave.

- If SPI receiving mode is enabled, the end of a transaction with master can be detected by the corresponding RXNE event signalizing the last data transfer completion.
- In SPI transmit mode, user can check the BSY under timeout corresponding to the time necessary to complete the last data frame transaction. The timeout should be measured from TXE event signalizing the last data frame transaction start (it is raised once the second bit transaction is ongoing). Either BSY becomes low normally or the timeout expires when the synchronization issue happens.

When upper workarounds are not applicable, the following sequence can be used to prevent the synchronization issue at SPI transmit mode.

1. Write last data to data register
2. Poll TXE until it becomes high to ensure the data transfer has started
3. Disable SPI by clearing SPE while the last data transfer is still ongoing
4. Poll the BSY bit until it becomes low
5. The BSY flag works correctly and can be used to recognize the end of the transaction.

Note: This workaround can be used only when CPU has enough performance to disable SPI after TXE event is detected while the data frame transfer is still ongoing. It is impossible to achieve it when ratio between CPU and SPI clock is low and data frame is short especially. In this specific case timeout can be measured from TXE, while calculating fixed number of CPU clock periods corresponding to the time necessary to complete the data frame transaction.

2.6.2 Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback

Description

In receive transaction, in both I^2S and SPI Master modes, the last bit of the transacted frame is not captured when the signal provided by internal feedback loop from the SCK pin exceeds a critical delay. The lastly transacted bit of the stored data then keeps the value from the pattern received previously. As a consequence, the last receive data bit may be wrong and/or the CRCERR flag can be unduly asserted in the SPI mode if any data under check sum and/or just the CRC pattern is wrongly captured.

In SPI mode, data are synchronous with the APB clock. A delay of up to two APB clock periods can thus be tolerated for the internal feedback delay. The I^2S mode is more sensitive than the SPI mode, especially when an odd I^2S prescaler factor is set and the APB clock is the system clock divided by two. In this case, the margin of the internal feedback delay is lower than 1.5 APB clock period.

The main factors contributing to the delay increase are low V_{DD} level, high temperature, high SCK pin capacitive load and low SCK I/O output speed. The SPI communication speed has no impact.

Workarounds

The following workaround can be adopted, jointly or individually:

- Decrease the APB clock speed.
- Configure the I/O pad of the SCK pin to be faster.

The following table gives the maximum allowable APB frequency versus GPIOx_OSPEEDR output speed control field setting for the SCK pin, at 30 pF of capacitive load.

Table 4. Maximum allowable APB frequency at 30 pF load

OSPEEDR [1:0] for SCK pin	Max. APB frequency for SPI mode [MHz]	Max. APB frequency for I^2S mode [MHz]
11 (very high)	32	-
10 (high)	32	-
01 (medium)	18 (20 if V_{DD} > 1.8 V)	-
00 (low)	2 (3 if V_{DD} > 1.8 V)	-

2.6.3 Wrong CRC transmitted in Master mode with delayed SCK feedback

Description

In transmit transaction of the SPI/I^2S interface in SPI Master mode with CRC enabled, the CRC data transmission may be corrupted if the delay of an internal feedback signal derived from the SCK output (further feedback clock) is greater than two APB clock periods. While data and CRC bit shifting and transfer is based on an internal clock, the CRC progressive calculation uses the feedback clock. If the delay of the feedback clock is greater than two APB periods, the transmitted CRC value may get wrong.

The main factors contributing to the delay increase are low V_{DD} level, high temperature, high SCK pin capacitive load and low SCK I/O output speed. The SPI communication speed has no impact.

Workaround

The following workaround can be adopted, jointly or individually:

- Decrease the APB clock speed.
- Configure the I/O pad of the SCK pin to be faster.

The following table gives the maximum allowable APB frequency versus GPIOx_OSPEEDR output speed control field setting for the SCK pin, at 30 pF of capacitive load.

Table 5. Maximum allowable APB frequency at 30 pF load

OSPEEDR [1:0] for SCK pin	Max. APB frequency for SPI mode [MHz]	Max. APB frequency for I^2S mode [MHz]
11 (very high)	32	-
10 (high)	32	-
01 (medium)	18 (20 if V_{DD} > 1.8 V)	-
00 (low)	2 (3 if V_{DD} > 1.8 V)	-

2.7 USART limitations

2.7.1 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.

2.7.2 DMA channel 3 (CH3) not functional when USART2_RX used for data reception

Description

When USART2 uses DMA channel 3 for data reception, the data transfer is blocked after the first byte has been received. As a result, DMA channel 3 cannot be used for USART2 data reception.

Workaround

Use DMA channel 5 (CH5) for USART2 data reception.

2.8 LPUART limitation

2.8.1 DMA channel 5 (CH5) not functional when LPUART1_RX used for data reception

Description

When LPUART1 uses DMA channel 5 for data reception, the data transfer is blocked after the first byte has been received. As a result, DMA channel 5 cannot be used for LPUART1 data reception.

Workaround

Use DMA channel 3 (CH3) for LPUART1 data reception.

2.9 IWDG limitation

2.9.1 IWDG does not always reset the device

Description

The IWDG must be configured so that the counter starts counting down and generates a reset when the end of count value is reached. In some cases, the configuration does not work, the IWDG remains stopped and does not reset the device.

Workaround

None.

3 Revision history

Table 6. Document revision history

Date	Revision	Changes
20-Nov-2015	1	Initial release.
01-Dec-2015	2	Updated status of limitation “Bootloader not functional with SPI1 interface on TSSOP14 package” for revision Z, in *Table 3: Summary of silicon limitations*.
11-Feb-2016	3	Updated *Section 2.1.5: NSS pin synchronization required when using bootloader with SPI1 interface on TSSOP14 package* and extended to device revision Z. Added *Section 2.2.1: Overrun flag might not be set when converted data have not been read before new data are written*.
14-Nov-2016	4	*Section 2.5: I2C peripheral limitations*: – Added *Section 2.5.3: 10-bit master Master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave*. *Section 2.6: SPI peripheral limitations*: – Updated *Section 2.6.1: BSY bit may stay high at the end of a SPI data transfer in Slave mode*. – Updated *Section 2.6.2: Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback*. – Added *Section 2.6.3: Wrong CRC transmitted in Master mode with delayed SCK feedback* *Section 2.7: USART limitations*: – Added *Section 2.7.2: DMA channel 3 (CH3) not functional when USART2_RX used for data reception*. Added *Section 2.8: LPUART limitation* and *Section 2.8.1: DMA channel 5 (CH5) not functional when LPUART1_RX used for data reception*.