STM32L010x3/4/6/8/B Errata sheet

STM32L010x3/4/6/8/B device limitations

Applicability

This document applies to the part numbers of STM32L010xx devices listed in *Table 1* and their variants shown in *Table 2*.

Section 1 gives a summary and *Section 2* a description of device limitations, with respect to the device datasheet and reference manual RM0451.

Table 1. Device summary

Reference	Part numbers
STM32L010x3	STM32L010D3
STM32L010x4	STM32L010F4, STM32L010F4
STM32L010x6	STM32L010C6
STM32L010x8	STM32L010K8, STM32L010R8
STM32L010xB	STM32L010RB

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32L010x3	Z	0x457
STM32L010x4	Z	0x457
STM32L010x6	X	0x425
STM32L010x8	X	0x417
STM32L010xB	Z	0x447

1. Refer to the device data sheet for how to identify this code on different types of package.
2. REV_ID[15:0] bit field of DBGMCU_IDCODE register. Refer to the reference manual.

Contents

1 Summary of device limitations

The following table gives a quick references to all documented device limitations of STM32L010xx and their status:

A = limitation present, workaround available

N = limitation present, no workaround available

P = limitation present, partial workaround available

"-" = limitation absent

Applicability of a workaround may depend on specific conditions of target application. Adoption of a workaround may cause restrictions to target application. Workaround for a limitation is deemed partial if it only reduces the rate of occurrence and/or consequences of the limitation, or if it is fully effective for only a subset of instances on the device or in only a subset of operating modes, of the function concerned.

Table 3. Summary of device limitations

Function	Section	Limitation	Status				
			STM32L010x3	STM32L010x4	STM32L010x6	STM32L010x8	STM32L010xB
			Rev.Z		Rev.X		Rev.Z
System	*2.1.1*	*Electrical sensitivity characteristics*	-	-	N	-	N
	2.1.2	*LDM, STM, PUSH and POP not allowed in IOPORT bus*	-	-	N	N	N
	2.1.3	*BOOT_MODE bits do not reflect the selected boot mode*	N	N	-	-	-
	2.1.4	*NSS pin synchronization required when using bootloader with SPI1 interface on TSSOP14 package*	A	A	-	-	-
RCC	*2.2.1*	*Delay after an RCC peripheral clock enabling*	A	A	A	A	A
GPIO	*2.3.1*	*Writing in byte mode to the GPIOx_OTYPER register does not work*	-	-	-	A	-
ADC	*2.4.1*	*Overrun flag might not be set when converted data have not been read before new data are written*	A	A	A	A	-
RTC	*2.5.1*	*Spurious tamper detection when disabling the tamper channel*	-	-	N	N	N
	2.5.2	*Detection of a tamper event occurring before enabling the tamper detection is not supported in edge detection mode*	A	A	A	A	A
I2C	*2.6.2*	*Wrong data sampling when data setup time (tSU;DAT) is shorter than one I2C kernel clock period*	A	A	A	A	A
	2.6.3	*10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave*	A	A	-	-	-

Table 3. Summary of device limitations

Function	Section	Limitation	Status				
			STM32L010x3	STM32L010x4	STM32L010x6	STM32L010x8	STM32L010xB
			Rev.Z		Rev.X		Rev.Z
USART	2.7.1	Start bit detected too soon when sampling for NACK signal from the smartcard	-	-	N	N	N
	2.7.2	Break request preventing the TC flag from being set	-	-	A	A	A
	2.7.3	nRTS is active while RE or UE = 0	A	A	A	A	A
	2.7.4	DMA channel 3 (CH3) not functional when USART2_RX used for data reception	A	A	-	-	-
LPUART	2.8.1	DMA channel 5 (CH5) not functional when LPUART1_RX used for data reception	A	A	-	-	-
SPI	2.9.1	BSY bit may stay high at the end of a SPI data transfer in Slave mode	A	A	A	A	A
	2.9.2	Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback	A	A	A	A	A
	2.9.3	SPI CRC corruption upon DMA transaction completion by another peripheral	-	-	-	A	-
	2.9.4	Wrong CRC transmitted in Master mode with delayed SCK feedback	A	A	-	-	-

The following table gives a quick reference to the documentation errata.

Table 4. Summary of device documentation errata

Function	Section	Documentation erratum
I2C	2.6.1	Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C

2 Description of device limitations

The following sections describe device limitations of the applicable Arm® [a] core devices and provide workarounds if available. They are grouped by device functions.

2.1 System

2.1.1 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.

2.1.2 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.

2.1.3 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.

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

2.1.4 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 RCC

2.2.1 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.

2.3 GPIO

2.3.1 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.

2.4 ADC

2.4.1 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.

2.5 RTC

2.5.1 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

2.5.2 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.

2.6 I2C

2.6.1 Wrong behavior in Stop mode when wakeup from Stop mode is disabled in I2C

Description

If the wakeup from Stop mode by I2C is disabled (WUPEN = 0), the correct use of the I2C peripheral is to disable it (PE = 0) before entering Stop mode, and re-enable it when back in Run mode.

Some reference manual revisions may omit this information.

Failure to respect the above while the MCU operating as slave or as master in multi-master topology enters Stop mode during a transfer ongoing on the I^2C-bus may lead to the following:

1. BUSY flag is wrongly set when the MCU 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 SCL line is pulled low by I2C and the transfer stalled as long as the MCU remains in Stop mode.

The occurrence of such condition depends on the timing configuration, peripheral clock frequency, and I2C-bus frequency.

This is a description inaccuracy issue rather than a product limitation.

Workaround

No application workaround is required.

2.6.2 Wrong data sampling when data setup time (tSU;DAT) is shorter than one I2C kernel clock period

Description

The I^2C-bus specification and user manual specify a minimum data setup time ($t_{SU;DAT}$) as:

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

The MCU does not correctly sample the I^2C-bus SDA line when $t_{SU;DAT}$ is smaller than one I2C kernel clock (I^2C-bus peripheral clock) period: the previous SDA value is sampled instead of the current one. This can result in a wrong receipt of slave address, data byte, or acknowledge bit.

Workaround

Increase the I2C kernel clock frequency to get I2C kernel clock period within the transmitter minimum data setup time. Alternatively, increase transmitter's minimum data setup time. If the transmitter setup time minimum value corresponds to the minimum value provided in the I^2C-bus standard, the minimum I2CCLK frequencies are as follows:

- In Standard mode, if the transmitter minimum setup time is 250 ns, the I2CCLK frequency must be at least 4 MHz.
- In Fast mode, if the transmitter minimum setup time is 100 ns, the I2CCLK frequency must be at least 10 MHz.
- In Fast-mode Plus, if the transmitter minimum setup time is 50 ns, the I2CCLK frequency must be at least 20 MHz.

2.6.3 10-bit master mode: new transfer cannot be launched if first part of the address is not acknowledged by the slave

Description

An I^2C-bus master generates STOP condition upon non-acknowledge I^2C address that it sends. This applies to 7-bit address as well as to each byte of 10-bit address.

When the MCU set as I^2C-bus master transmits a 10-bit address of which the first byte (5-bit header + 2 MSBs of the address + direction bit) is not acknowledged, the MCU duly generates STOP condition but it then cannot start any new I^2C-bus transfer. In this spurious state, the NACKF flag of the I2C_ISR register and the START bit of the I2C_CR2 register are both set, while the START bit should normally be cleared.

Workaround

In 10-bit-address master mode, if both NACKF flag and START bit get simultaneously set, 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 three APB cycles.
4. Enable the I2C peripheral again.

2.7 USART

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 preventing the TC flag from being set

Description

After the end of transmission of a data (D1), the transmission complete (TC) flag is not set when the following conditions are met:

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

As a consequence, an application relying on the TC flag fails to detect the end of data transfer.

Workaround

In the application, only allow break request after 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

Upon setting the UE and RE bits, configure the I/O used for nRTS into an alternate function.

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

Description

When USART2 uses DMA channel 3 for data reception, data transfers are 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

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 SPI

2.9.1 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. This occurs upon coincidence of internal CPU clock and external SCK clock provided by master.

In such an event, if the software only relies on BSY flag to detect the end of SPI slave data transaction (for example to enter low-power mode or to change data line direction in half-duplex bidirectional mode), the detection fails.

As a conclusion, the BSY flag is unreliable for detecting the end of data transactions.

Workaround

Depending on SPI operating mode, use the following means for detecting the end of transaction:

- When NSS hardware management is applied and NSS signal is provided by master, use NSS flag.
- In SPI receiving mode, use the corresponding RXNE event flag.
- In SPI transmit-only mode, use the BSY flag in conjunction with a timeout expiry event. Set the timeout such as to exceed the expected duration of the last data frame and start it upon TXE event that occurs with the second bit of the last data frame. The end of the transaction corresponds to either the BSY flag becoming low or the timeout expiry, whichever happens first.

Prefer one of the first two measures to the third as they are simpler and less constraining.

Alternatively, apply the following sequence to ensure reliable operation of the BSY flag in SPI transmit mode:

1. Write last data to data register.
2. Poll the TXE flag until it becomes high, which occurs with the second bit of the data frame transfer.
3. Disable SPI by clearing the SPE bit mandatorily before the end of the frame transfer.
4. Poll the BSY bit until it becomes low, which signals the end of transfer.

Note: The alternative method can only be used with relatively fast CPU speeds versus relatively slow SPI clocks or/and long last data frames. The faster is the software execution, the shorter can be the duration of the last data frame.

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

Description

In receive transaction in SPI Master mode, 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.

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 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.

The following table gives the maximum allowable APB frequency versus GPIOx_OSPEEDR output speed bitfield setting for the SCK pin, at 30 pF of capacitive load. The operation is safe up to that frequency.

Table 5. Maximum allowable APB frequency at 30 pF load

OSPEEDR [1:0] for SCK pin	Max. APB frequency for SPI 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)

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.

2.9.3 SPI CRC corruption upon DMA transaction completion by another peripheral

Description

When the following conditions are all met:

- CRC function for the SPI is enabled
- SPI transaction managed by software (as opposed to DMA) is ongoing and CRCNEXT flag set
- Another peripheral using the DMA channel on which the SPI is mapped completes a DMA transfer,

the CRCNEXT bit is unexpectedly cleared and the SPI CRC calculation may be corrupted, setting the CRC error flag.

Workaround

Ensure that the DMA channel on which the SPI is mapped is not concurrently in use by another peripheral.

2.9.4 Wrong CRC transmitted in Master mode with delayed SCK feedback

Description

In transmit transaction 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.

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 6. Maximum allowable APB frequency at 30 pF load

OSPEEDR [1:0] for SCK pin	Max. APB frequency for SPI 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)

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.

3 Revision history

Table 7. Document revision history

Date	Revision	Changes
03-Jun-2019	1	Initial release.