STM32L471xx Errata sheet

STM32L471xx device limitations

Silicon identification

This errata sheet applies to revision ‘4’ of the STMicroelectronics STM32L471xx products. The STM32L471xx family features an ARM® 32-bit Cortex®-M4 core.

Section 2 gives a detailed description of the product silicon limitations.

The full list of part numbers is shown in *Table 2*. The products are identifiable by the revision code marked below the order code on the device package, as shown in *Table 1*.

Table 1. Device identification(1)

Sales type	Revision code marked on the device(2)
STM32L471xx	‘4’

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see the reference manual RM0392 for details on how to find the revision code).
2. Refer to datasheet for details on how to identify the revision code according to the packages.

Table 2. Device summary

Reference	Part numbers
STM32L471xx	STM32L471QE, STM32L471QG, STM32L471RE, STM32L471RG STM32L471VE, STM32L471VG, STM32L471ZE, STM32L471ZG

Contents

List of tables

List of figures

1 ARM® 32-bit Cortex®-M4 FPU core limitations

An errata notice of the STM32L471xx core is available from the following web address: *http://infocenter.arm.com.*

All the described limitations are minor and related to the revision r0p1-v1 of the Cortex®-M4 FPU core. *Table 3* summarizes these limitations and their implications on the behavior of STM32L471xx devices.

Table 3. Cortex®-M4 FPU core limitations and impact on microcontroller behavior

ARM ID	ARM category	ARM summary of errata	Impact
752419	Cat 2	Interrupted loads to SP can cause erroneous behavior	Minor

1.1 Cortex®-M4 FPU core interrupted loads to stack pointer can cause erroneous behavior

Description

An interrupt occurring during the data-phase of a single word load to the stack pointer (SP/R13) can cause an erroneous behavior of the device. In addition, returning from the interrupt results in the load instruction being executed with an additional time.

For all the instructions performing an update of the base register, the base register is erroneously updated on each execution, resulting in the stack pointer being loaded from an incorrect memory location.

The instructions affected by this limitation are the following:

- LDR SP, [Rn],#imm
- LDR SP, [Rn,#imm]!
- LDR SP, [Rn,#imm]
- LDR SP, [Rn]
- LDR SP, [Rn,Rm]

Workaround

As of today, no compiler generates these particular instructions. This limitation can only occur with hand-written assembly code.

Both issues can be solved by replacing the direct load to the stack pointer by an intermediate load to a general-purpose register followed by a move to the stack pointer.

Example:

Replace LDR SP, [R0] by

LDR R2,[R0]

MOV SP,R2

2 STM32L471xx silicon limitations

Table 4 gives quick references to all documented limitations.

Legend for *Table 4*: A = workaround available; N = no workaround available; P = partial workaround available, '-' and grayed = fixed.

Table 4. Summary of silicon limitation

Section	Limitation	Rev 4
Section 2.1: System limitations	*Section 2.1.1: Write operation in the Flash memory while it is not ready (Flash memory in power-down) is not correctly handled*	A
	Section 2.1.2: The configuration of the I/Os not available in WLCSP package can be modified by software	A
	Section 2.1.3: SRAM2 read access while the SRAM2 hardware erase is ongoing is not correctly supported	N
	Section 2.1.4: PH0/PH1 is controlled by the GPIOH registers when HSE is enabled	P
	Section 2.1.5: PWR_CR4 register write access may not be completed if a Low-power mode is entered just after the write operation	A
	Section 2.1.6: HSI user trim is limited on some samples	N(1)
	Section 2.1.7: Option byte loading can fail if MSI frequency is greater than 8 MHz	A
	Section 2.1.8: PLL may not lock if VCO is below 96 MHz and temperature is below 0 °C	A
	Section 2.1.9: Full JTAG configuration without NJTRST pin cannot be used	A
	Section 2.1.10: MSIRDY flag issue preventing entry in low power mode	A
	Section 2.1.11: PCPROP area within a single Flash memory page becomes unprotected at RDP change from level 1 to level 0	A
	Section 2.1.12: Data Cache might be corrupted during Flash Read While Write operation	A
	Section 2.1.13: MSI frequency overshoot upon Stop mode exit	A
	Section 2.1.14: Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled	A
	Section 2.1.15: OPAMP output: VDDA overconsumption	N
	Section 2.1.16: Spurious BOR when entering Stop mode after short Run sequence	A
Section 2.2: FMC peripheral limitations	*Section 2.2.1: Dummy read cycles inserted when reading synchronous memories*	N
	Section 2.2.2: Data corruption during burst read from FMC synchronous memory	A
	Section 2.2.3: FMC bank switching to asynchronous bank for write	A
	Section 2.2.4: Read burst access of nine words or more is not supported by FMC	P

Table 4. Summary of silicon limitation (continued)

Section	Limitation	Rev 4
Section 2.3: QUADSPI peripheral limitations	*Section 2.3.1: QUADSPI_BK1_IO1 is always an input when the command is sent in dual or quad SPI mode*	P
	Section 2.3.2: Hard fault is not generated in case of out-of-range memory-mapped access	N
	Section 2.3.3: Extra data written in the FIFO at the end of a read transfer	A
	Section 2.3.4: First nibble of data is not written after dummy phase	A
	Section 2.3.5: Wrong data can be read in memory-mapped after an indirect mode operation	A
Section 2.4: ADC peripheral limitations	*Section 2.4.1: Injected queue of context is not available in case of JQM=0*	N
	Section 2.4.2: DMA2 channels 2 and 3 (respectively) cannot be used when the ADC2 and ADC3 requests are selected on DMA2 channels 4 and 5	P
	Section 2.4.3: Wrong ADC conversion results when delay between calibration and first conversion or between two consecutive conversions is too long	N
	Section 2.4.4: Burst read or write accesses are not supported by the ADC	A
	Section 2.4.5: Unexpected end of transfer on DMA1 when using DMA1 channel 2 for ADC2 while DMA2 channel 4 is also used	A
	Section 2.4.6: Unexpected end of transfer on DMA1 when using DMA1 channel 3 for ADC3 while DMA2 channel 5 is also used	A
Section 2.5: DFSDM peripheral limitations	*Section 2.5.1: RDATACH[2:0] status bits are not implemented*	N
	Section 2.5.2: New regular channel selection taken into account at the end of an injected conversion when a regular conversion is pending	N
	Section 2.5.3: DFSDM triggers from timers can be missed in specific conditions	A
Section 2.6: I2C peripheral limitations	*Section 2.6.1: I2C Fast-mode Plus drive is not available on all SDA/SCL I/Os*	N
	Section 2.6.2: Wrong behavior related with MCU Stop mode when wakeup from Stop mode by I2C peripheral is disabled	A
	Section 2.6.3: Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period	P
	Section 2.6.4: Spurious Bus Error detection in master mode	A
	Section 2.6.5: I2C3 analog filters requires that both PC0/PC1 or both PG7/PG8 are configured as I2C3 alternate function	A
	Section 2.6.6: 10-bit master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave	A
Section 2.7: SDMMC peripheral limitations	*Section 2.7.1: Wrong CCRCFAIL status after a response without CRC is received*	A
	Section 2.7.2: Clock division per 255 is not possible	N
	Section 2.7.3: Wait for response bits “10” configuration does not work correctly	A
	Section 2.7.4: MMC stream write of less than 8 bytes does not work correctly	A
Section 2.8: bxCAN peripheral limitations	*Section 2.8.1: bxCAN time-triggered mode is not supported*	N

Table 4. Summary of silicon limitation (continued)

Section	Limitation	Rev 4
Section 2.9: SPI limitations	*Section 2.9.1: BSY bit may stay high at the end of a data transfer at slave mode*	A
Section 2.10: SWPMI peripheral limitations	*Section 2.10.1: SUSPENDED mode entry delayed*	N
	Section 2.10.2: SUSPENDED mode never entered	P
	Section 2.10.3: SRF flag not set	N
	Section 2.10.4: SWP SUSPENDED mode not supported when STM32L4 is in Stop 0 or Stop 1 mode	N
	Section 2.10.5: SWPMI_IO transceiver bypass mode is not functional	N
Section 2.11: RTC peripheral limitations	*Section 2.11.1: Spurious tamper detection when disabling the tamper channel*	P
Section 2.12: USART limitations	*Section 2.12.1: Start bit detected too soon when sampling for NACK signal from the smartcard*	N
	Section 2.12.2: Break request can prevent the Transmission Complete flag (TC) from being set	A
	Section 2.12.3: nRTS is active while RE or UE = 0	A
Section 2.13: COMP peripheral limitations	*Section 2.13.1: Comparators propagation delay is longer than expected for input steps higher than 200 mV*	N
	Section 2.13.2: Comparators output cannot be configured in open-drain	N
	Section 2.13.3: COMP1 and COMP2 configuration lost with software reset	N
Section 2.14: LPTIM peripheral limitations	*Section 2.14.1: Low power timer 1 (LPTIM1) outputs cannot be configured in open-drain*	N
Section 2.15: LPUART peripheral limitations	*Section 2.15.1: Low power UART1 (LPUART1) outputs cannot be configured in open-drain*	N
Section 2.16: TSC peripheral limitations	*Section 2.16.1: Inhibited acquisition in short transfer phase configuration*	A

1. Fixed on products with date code newer than 609.

2.1 System limitations

2.1.1 Write operation in the Flash memory while it is not ready (Flash memory in power-down) is not correctly handled

Description

If the Flash memory is read while it is not ready after a power-down state (i.e after Stop mode, or when it was put in power-down by means of SLEEP_PD or RUN_PD bits in the FLASH_ACR register), the AHB bus is normally stalled until the Flash memory is ready and can provide the correct data.

However, this mechanism is not supported in case of a write operation in normal programming mode: the Flash memory read which is automatically performed before the programming in order to check that the address is virgin is incorrect. Consequently the write operation is skipped.

Workaround

The user must wait until the Flash memory is ready before performing a write operation: this is done by reading first an address of the Flash memory not present in the cache.

2.1.2 The configuration of the I/Os not available in WLCSP package can be modified by software

Description

Unbonded I/Os in WLCSP package can be configured by software. This can lead to an extra consumption if they are floating with Schmitt trigger ON.

Workaround

Configure all unbonded I/Os as analog input or as output push-pull 0 state.

2.1.3 SRAM2 read access while the SRAM2 hardware erase is ongoing is not correctly supported

Description

If a read access is done in the SRAM2, while a SRAM2 hardware erase operation is on going: the read access is stalled as long as the erase operation is not completed, and the read value is not 0x0, but the latest value previously read from SRAM2.

Workaround

None.

2.1.4 PH0/PH1 is controlled by the GPIOH registers when HSE is enabled

Description

The PH0 and PH1 GPIO configuration is not bypassed when the HSE is enabled. The oscillator can stop if the I/Os are not configured in analog mode.

Workaround

PH0 and PH1 must be configured in analog mode (reset value) when HSE is enabled.

2.1.5 PWR_CR4 register write access may not be completed if a Low-power mode is entered just after the write operation

Description

PWR_CR4 write access should insert 3 APB1 wait states, but it does not. Consequently, if a low-power mode is entered just after writing into this register, the PWR_CR4 register may not be updated before the low-power mode is entered. This can occur in particular when the APB1 clock is prescaled.

Workaround

PWR_CR4 register must be read after the write operation to ensure that the write is done before entering the low-power mode.

2.1.6 HSI user trim is limited on some samples

Description

The HSI user trimming step is typically:

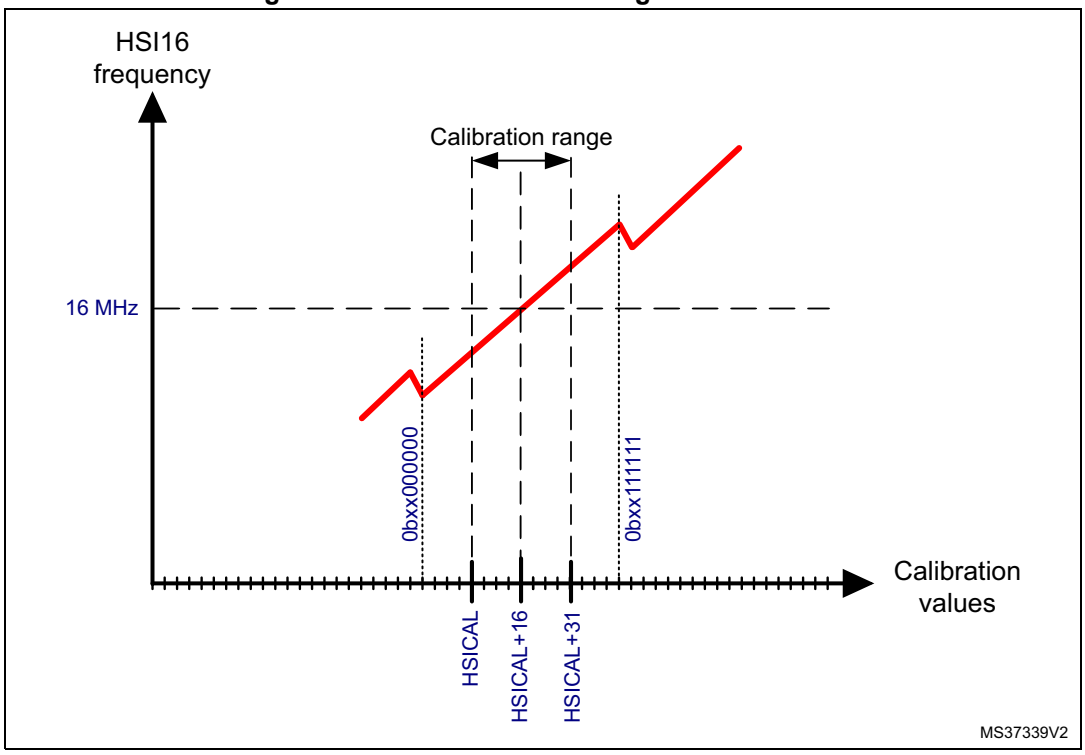
- +0.3% of frequency when the trimming code is not a multiple of 64;
- -7% of frequency when the trimming code is a multiple of 64.

As shown in *Figure 1*, the HSI user trimming allows to add or to subtract up to 16 steps compared to the factory trim value.

If the HSI factory trim value is close to a multiple of 64, with only 16 steps for positive or for negative correction, it is not possible to compensate the 7% drop when the code is a multiple of 64.

Consequently, the user trim correction must not jump over the codes multiple of 64, which can limit the user correction either in positive or in negative direction.

Figure 1. HSI oscillator trimming characteristics



Workaround

None.

Fix

This limitation is fixed on Rev 4 products with date code newer than 609.

2.1.7 Option byte loading can fail if MSI frequency is greater than 8 MHz

Description

The option byte loading operation can fail if the MSI clock is ON with a frequency equal or greater than 8 MHz before performing an option byte loading by setting OBL_LAUNCH bit in the FLASH _CR register. Some options and engineering bytes values can be corrupted.

Workaround

If MSI is ON at a frequency equal or higher than 8MHz, It is mandatory to reduce MSI frequency to 4 MHz or less before launching the Option Byte loading operation by setting the OBL_LAUNCH bit in the FLASH_CR register.

2.1.8 PLL may not lock if VCO is below 96 MHz and temperature is below 0 °C

Description

The VCO minimum value should be 64 MHz. When the VCO is below 96 MHz and the temperature is below 0 °C the PLL may never lock.

Workaround

Program the PLL with VCO at 96 MHz or above.

2.1.9 Full JTAG configuration without NJTRST pin cannot be used

Description

When using the JTAG debug port in debug mode, the connection with the debugger is lost if the NJTRST pin (PB4) is used as a GPIO. Only the 4-wire JTAG port configuration is impacted.

Workaround

Use the SWD debug port instead of the full 4-wire JTAG port.

2.1.10 MSIRDY flag issue preventing entry in low power mode

Description

If the MSI clock is stopped when it is running at high frequency (24 MHz or above), the MSIRDY (MSI ready) flag can stay at value 1 instead of 0.

Once this flag remains set while the MSI clock is OFF, entry in Stop 0, Stop 1, Stop 2, Standby and Shutdown modes is no longer possible (the product is blocked in the low power mode entry phase).

The following factors increase the probability to have this issue:

- a high MSI frequency
- a low V_{DD} external supply
- a high temperature.

Workaround

- The MSI clock can run at any frequency if both conditions below are met:
  a) the system clock is MSI or PLL fed by MSI when requesting entry in low power mode
  b) the wakeup clock is MSI
- If the system clock is any other clock than MSI, lower the MSI frequency to 16 MHz (or less) and add a short delay loop (200 ns minimum) before stopping MSI by software. This will ensure that the MSIRDY flag be really 0 before requesting entry in low power mode.
- If the system clock is MSI and the wakeup clock is not MSI, lower the MSI frequency to 16 MHz (or less) and add a short delay loop (200 ns minimum) before requesting entry in low power mode.
- If the system clock is PLL fed by MSI and the wakeup clock is not MSI, the MSI frequency used as PLL input should not exceed 16 MHz. In other words, do not use the PLL fed by MSI as system clock with MSI at high frequency (24 MHz or above) divided by PLLM (ensuring the PLL input is between 4 and 16 MHz).

2.1.11 PCPROP area within a single Flash memory page becomes unprotected at RDP change from level 1 to level 0

Description

With PCROP_RDP option bit set to 0, the change of RDP from level 1 to level 0 normally results in erasure of Flash memory banks except the Flash memory pages containing PCROP area. The PCROP area remains read-protected.

This operates as expected if the PCROP area crosses the limits of at least one Flash memory page, which is always true if PCROP area size exceeds 2 Kbytes. The limitation occurs if the PCROP area is fully contained within one single Flash memory page. Upon the RDP change from level 1 to level 0, the Flash memory bank with PCROP area is not erased and the read protection of the PCROP area is removed.

Workaround

Always define PCROP area so that it crosses the limits of at least one Flash memory page.

2.1.12 Data Cache might be corrupted during Flash Read While Write operation

Description

When a write to the internal Flash memory is done, the Data Cache is normally updated to reflect the data value update. During this Data Cache update, a read to the other Flash memory bank may occur; this read can corrupt the Data Cache content and subsequent read operations at the same address (Cache hits) will be corrupted.

This limitation only occurs in dual bank mode, when reading (data access or code execution) from one bank while writing to the other bank with Data Cache enabled.

Workaround

When the application is performing data accesses in both Flash memory banks, the Data Cache must be disabled by resetting the DCEN bit before any write to the Flash memory. Before enabling the Data Cache again, it must be reset by setting and then resetting the DCRST bit.

Code Example

```
  /* Disable data cache  */
  __HAL_FLASH_DATA_CACHE_DISABLE();

  /* Set PG bit */
  SET_BIT(FLASH->CR, FLASH_CR_PG);

  /* Program the Flash word */
WriteFlash(Address, Data);

  /* Reset data cache */
  __HAL_FLASH_DATA_CACHE_RESET();
  /* Enable data cache */
  __HAL_FLASH_DATA_CACHE_ENABLE();
```

2.1.13 MSI frequency overshoot upon Stop mode exit

Description

When

- the system is clocked by the MSI clock, and
- MSI is selected as system clock source upon wakeup from Stop mode, and
- a wakeup event occurs only a few system clock cycles before entering Stop mode,

then upon the exit from Stop mode, the MSI frequency can overshoot above its selected range.

The limitation applies to all Stop modes: Stop 0, Stop 1 and Stop 2.

Workaround

There are two possible ways to work the limitation around:

1. Go through the following sequence:
   a) Switch to HSI
   b) Shutdown MSI
   c) Wait for MSIRDY to go low (after six MSI clock cycles)
   d) Mask_Interrupts (Set PRIMASK)
   e) Enter in STOP mode with request to wakeup on MSI
   f) Enable MSI
   g) Wait for MSIRDY to go high
   h) Switch to MSI (required as the system clock remains HSI in case the MCU did not enter Stop due to an early wakeup event)
   i) Unmask_Interrupts (Clear PRIMASK)

   This workaround offers a shorter wakeup time.
2. Select HSI as clock source upon wakeup from Stop mode. After the wakeup, switch the system clock to MSI.

2.1.14 Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled

Description

When entering Stop mode with the temperature sensor channel and the associated ADC(s) enabled, the internal voltage reference may be corrupted.

The occurrence of the corruption depends on the supply voltage and the temperature.

The corruption of the internal voltage reference may cause:

- an overvoltage in V_{CORE} domain
- an overshoot / undershoot of internal clock (LSI, HSI, MSI) frequencies
- a spurious brown-out reset

The limitation applies to Stop 1 and Stop 2 modes.

Workaround

Before entering Stop mode

- disable the ADC(s) using the temperature sensor signal as input, and/or
- disable the temperature sensor channel, by clearing the CH17SEL bit of the ADCx_CCR register.

Disabling both allows consuming less power during Stop mode.

2.1.15 OPAMP output: V_{DDA} overconsumption

Description

An overconsumption can appear on V_{DDA} in the following conditions:

- a voltage V_{PAD} is applied on PA3 or PB0 with V_{DDA}+ 50 mV < V_{PAD} < V_{DDA} + 600 mV
- the OPAMP output is not used (OPAMPx_VOUT pins are not driven by the OPAMP)
- temperature is below 0 °C

This extra consumption is constant and can reach up to 1 mA.

Workaround

None, except avoiding the conditions indicated above.

2.1.16 Spurious BOR when entering Stop mode after short Run sequence

Description

When the MCU wakes up from Stop mode and enters the Stop mode again within a short period of time, a spurious Brown Out Reset may be generated.

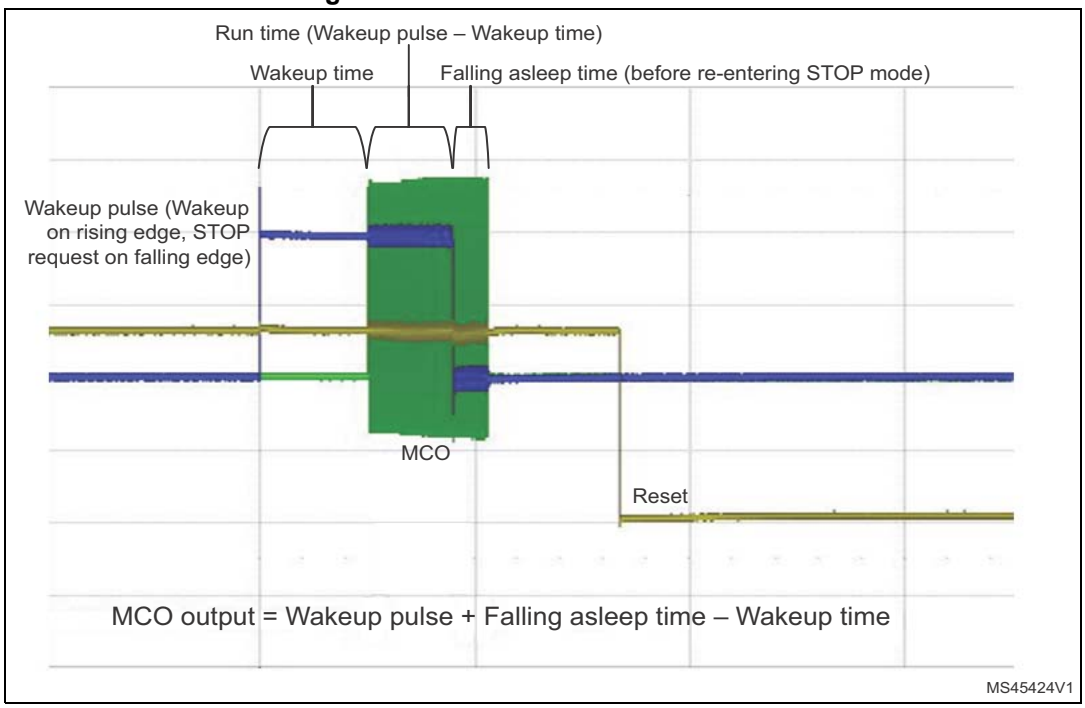
This limitation depends upon the supply voltage, as indicated in *Table 5*, and applies to Stop 1 and Stop 2 modes.

Table 5. Minimum Run time to avoid spurious BORs

V_{DD} supply voltage (V)	Minimum Run time (µs)
1.71	15
1.8	13
2.0	11
2.2	9
2.4	8
2.6	6
2.8	5
3.0	3
3.2 and above	2

The minimum Run time defined in *Table 5* corresponds to the firmware execution time on the core. Note also that, as shown in *Figure 2*, the length of the MCO output (green trace) is longer than the firmware execution time.

Figure 2. Minimum Run time definition



Workaround

Ensure that the Run time between two Stop sequences is long enough to avoid the generation of a Brown Out Reset.

This can be done by adding a software loop or using a timer to add a delay. The system clock frequency can be reduced during this waiting loop to minimize power consumption.

2.2 FMC peripheral limitations

2.2.1 Dummy read cycles inserted when reading synchronous memories

Description

When performing a burst read access to a synchronous memory, two dummy read accesses are performed at the end of the burst cycle whatever the type of AHB burst access.

However, the extra data values which are read are not used by the FMC and there is no functional failure.

Workaround

None.

2.2.2 Data corruption during burst read from FMC synchronous memory

Description

A burst read from static memory can be corrupted if all the following conditions are met:

- One FMC bank is configured in synchronous mode with WAITEN bit enabled while another FMC bank is used with WAITEN bit disabled.
- A read burst transaction is ongoing from static synchronous memory with wait feature enabled.
- The synchronous memory asserts the wait signal during the ongoing burst read.
- The read burst transaction is followed by an access to an FMC banks for which the WAITEN bit is disabled in the FMC_BCRx register.

Workaround

1. Set the WAITEN bit on all FMC static banks even if it is not used by the memory.
2. Set the same WAIT polarity on all static banks.
3. Enable the internal pull-up on PD6 in order to set to ready the FMC_NWAIT input when the synchronous memory is de-selected and the other FMC bank without wait feature is selected.

2.2.3 FMC bank switching to asynchronous bank for write

Description

When switching from one of the FMC banks in read transaction to another asynchronous bank for write, the FMC could hang in one of the following conditions:

- one FMC bank is enabled with BUSTURN timing > 0;
- a second FMC bank is enabled in asynchronous (multiplexed or not multiplexed) mode with BUSTURN = 0;
- a read from the first bank followed by a write transaction on the second bank.

Workaround

Use BUSTURN = 0, or different from 0 for used banks.

2.2.4 Read burst access of nine words or more is not supported by FMC

Description

CPU read burst access equal to or more than 9 registers to FMC returns corrupted data starting from the 9th read word. These bursts can only be generated by Cortex®-M4 CPU and not by the other masters (i.e not by DMA).

This issue occurs when the stack is remapped on the external memory on the FMC and POP operations are performed with 9 or more registers.

This also occurs when LDM/VLDM operations are used with 9 or more registers.

Workaround

Stack must be located in the internal SRAM.

IAR™-EWARM: EWARM compiler does not generate LDM/VLDM operations with more than 8 registers except when the *setjmp* or *longjmp* functions of C Library are explicitly used in the source code.

Keil® MDK-ARM™: Starting from version 5.06, ARM® Compiler implements a patch which limits the number of registers with LDM/VLDM operations. Starting from version 5.16, MDK-ARM™ includes ARM® compiler with this patch, addressing the hardware limitations of the STM32L4 FMC burst reads.

Tasking: TASKING compiler starting from version v5.2r1 does not generate LDM instructions with more than 8 registers except when the *setjmp* or *longjmp* functions of C Library are explicitly used in the source code. It generates POP instructions with nine registers.

In assembly written code, LDM/VLDM operations must be limited to eight registers.

2.3 QUADSPI peripheral limitations

2.3.1 QUADSPI_BK1_IO1 is always an input when the command is sent in dual or quad SPI mode

Description

The IO1 is staying as an input instead of changing to output to drive the command when this last is on two or four lines and there is no address, no data and no alternate byte in the frame, or they exist on one line.

Therefore it is not possible to use dual-/quad-mode for the command phase.

This bug affects Numonyx's dual-/quad-instruction mode.

Workaround

The dual-/quad-instruction mode of Micron or Numonyx flashes is working fine when user does not send only a command in the frame. For example, the «write enable» command can be avoided by writing directly the status register of the flash instead. Otherwise the application can reset the flash through the reset pin (if it exists) to disable the duad-/quad instruction mode before each time it needs to send a frame which contains only the command phase.

2.3.2 Hard fault is not generated in case of out-of-range memory-mapped access

Description

A hard fault is not generated when the Cortex CPU executes an instruction which reads from an out-of-range memory-mapped address. Consequently the CPU effectively skips the instruction (the destination register is not modified) and continues executing without giving a hard fault.

Workaround

None.

2.3.3 Extra data written in the FIFO at the end of a read transfer

Description

When all the conditions listed below are gathered:

- QUADSPI is used in indirect mode
- QUADSPI clock is AHB/2 (PRESCALER = 0x01 in the QUADSPI_CR)
- QUADSPI is in quad mode (DMODE = 0b11 in the QUADSPI_CCR)
- QUADSPI is in DDR mode (DDRM = 0b1 in the QUADSPI_CCR)

An extra data is incorrectly written in the FIFO when a data is read at the same time that the FIFO gets full at the end of a read transfer.

Workaround

One of the two workarounds listed below can be done:

- Read out the extra data until the BUSY flag goes low and discard it.
- Request an abort after reading out all the correct received data from FIFO in order to flush FIFO and have the busy low. Abort will keep the last register configuration (set the ABORT bit in the QUADSPI_CR).

2.3.4 First nibble of data is not written after dummy phase

Description

The first nibble of data to be written to the external Flash memory is lost in the following conditions:

- QUADSPI is used in indirect write mode
- And at least one dummy cycle is used

Workaround

Use alternate bytes instead of dummy phase to add latency between address phase and data phase. This works only if the number of dummy cycles corresponds to a multiple of eight bits of data.

Example: to generate

- one dummy cycle: send one alternate byte, possible only in four data lines DDR mode
- two dummy cycles: send one alternate byte in four data lines SDR mode
- four dummy cycles: send two alternate bytes in four data lines SDR mode, or send one alternate byte in two data lines SDR mode
- eight dummy cycles: send one alternate byte in one data line SDR mode

2.3.5 Wrong data can be read in memory-mapped after an indirect mode operation

Description

Wrong data can be read with the first memory-mapped read request when in the following condition:

- Quad-SPI peripheral entered memory-mapped mode with both LSB bits in the address register QUADSPI_AR[1:0] not reset.

Workaround

QUADSPI_AR register must be reset just before entering memory-mapped mode.

Depending on the current Quad-SPI operating mode, one of the two workarounds listed below can be used:

1. Indirect read mode: reset address register then do an abort request to stop reading and clear busy bit.
   Then enter to memory-mapped mode.
2. Indirect write mode: reset the address register then enter to memory-mapped mode

Note: User should take care to not write to QUADSPI_DR register after resetting address register.

2.4 ADC peripheral limitations

2.4.1 Injected queue of context is not available in case of JQM=0

Description

The queue mechanism is not functional when JQM = 0. The effective queue length is equal to one stage: a new context written before the previous context's consumption will lead to a queue overflow and will be ignored. Consequently, the ADC must be stopped before programming the JSQR register.

Workaround

None.

2.4.2 DMA2 channels 2 and 3 (respectively) cannot be used when the ADC2 and ADC3 requests are selected on DMA2 channels 4 and 5

Description

When the DMA2 channel 4 is used for the ADC2 requests (C4S=0000 in the DMA2_CSELR register) it is also needed to select request 0 for DMA2 channel 2 (C2S=0000 in the DMA2_CSELR register). The consequence is that the DMA2 channel 2 cannot be used by another peripheral.

When the DMA2 channel 5 is used for the ADC3 requests (C5S=0000 in the DMA2_CSELR register) it is also needed to select request 0 for DMA2 channel 3 (C3S=0000 in the DMA2_CSELR register). The consequence is that the DMA2 channel 3 cannot be used by another peripheral.

Workaround

Select the DMA1 channel 2 for the ADC2 DMA requests or use other available mapping for the peripherals mapped on DMA2 channel 2.

Select the DMA1 channel 3 for the ADC3 DMA requests or use other available mapping for the peripherals mapped on DMA2 channel 3.

2.4.3 Wrong ADC conversion results when delay between calibration and first conversion or between two consecutive conversions is too long

Description

When the delay between two consecutive ADC conversions is higher than 1 ms the result of the second conversion might be incorrect. The same issue occurs when the delay between the calibration and the first conversion is higher than 1 ms.

Workaround

When the delay between two ADC conversions is higher than the above limit, perform two ADC consecutive conversions in single, scan or continuous mode: the first is a dummy conversion of any ADC channel. This conversion should not be taken into account by the application.

2.4.4 Burst read or write accesses are not supported by the ADC

Description

The ADC does not support LDM, STM, LDRD and STRD instructions for successive multiple-data read and write accesses to a contiguous address block.

Workaround

Prevent compilers from generating LDM, STM, LDRD and STRD instructions. In general, this can be achieved organizing the source code to avoid consecutive read or write accesses to neighboring addresses in lower-to-higher order. In cases where consecutive read or write accesses to neighboring addresses cannot be avoided, order the source code so that it accesses higher address first.

2.4.5 Unexpected end of transfer on DMA1 when using DMA1 channel 2 for ADC2 while DMA2 channel 4 is also used

Description

When the DMA1 channel 2 is used for the ADC2 requests and the DMA2 channel 4 is used by another peripheral (for instance the SPI1_TX requests), the end of transfer and acknowledge signals from the DMA2 can be received by the ADC2 while it should only be received by the peripheral using this DMA2 channel 4 (for instance the SPI1_TX). The consequence is that the DMA1 transfer will be interrupted earlier than expected. This issue only occurs when the DMA2 channel 2 is configured for the request 0 (reset configuration).

Workaround

When this limitation is observed, the DMA2 channel 2 should not stay configured for request 0 (reset value), but should be configured for any other request (1 to 7) even if it is useless for the application. This is done by configuring DMA2_CSELR register bits [7:4] to a value different from 0000. Note that there is no need to enable the DMA2 Channel 2.

2.4.6 Unexpected end of transfer on DMA1 when using DMA1 channel 3 for ADC3 while DMA2 channel 5 is also used

Description

When the DMA1 channel 3 is used for the ADC3 requests and the DMA2 channel 5 is used by another peripheral (for instance the UART4_RX requests), the end of transfer and acknowledge signals from the DMA2 can be received by the ADC3 while it should only be received by the peripheral using this DMA2 channel 5 (for instance the UART4). The consequence is that the DMA1 transfer will be interrupted earlier than expected. This issue only occurs when the DMA2 channel 3 is configured for the request 0 (reset configuration).

Workaround

When this limitation is observed, the DMA2 channel 3 should not stay configured for request 0 (reset value), but should be configured for any other request (1 to 7) even if it is useless for the application. This is done by configuring DMA2_CSELR register bits [11:8] to a value different from 0000. Note that there is no need to enable the DMA2 Channel 3. If the ADC1 was using the DMA2 Channel 3, it is necessary to remap the ADC1 DMA transfers to the DMA1 Channel 1.

2.5 DFSDM peripheral limitations

2.5.1 RDATACH[2:0] status bits are not implemented

Description

RDATACH[2:0] “Regular channel most recently converted” status bits are not implemented in the DFSDM data register for the regular channel (DFSDMx_RDATAR).

Workaround

None. These bits will be implemented in next silicon release.

2.5.2 New regular channel selection taken into account at the end of an injected conversion when a regular conversion is pending

Description

When a regular channel conversion is on going and is interrupted by an injected channel conversion: if the regular channel is changed on the fly during the injected channel conversion, the new regular channel selection is taken into account at the end of the injected conversion.

Workaround

None: do not change the regular channel selection on the fly when regular continuous conversions are requested.

2.5.3 DFSDM triggers from timers can be missed in specific conditions

Description

Triggers from timers to DFSDM can be missed when all the conditions listed below occur:

- the DFSDM clock is the APB2 clock PCLK2 (DFSDMSEL=0 in the RCC_CCIPR register)
- the DFSDM is triggered by TIM3_TRGO, TIM4_TRGO, TIM6_TRGO or TIM7_TRGO
- the APB2 frequency is smaller than the APB1 frequency

Workaround

Select the System clock as DFSDM clock (DFSDMSEL=1 in the RCC_CCIPR register).

2.6 I2C peripheral limitations

2.6.1 I2C Fast-mode Plus drive is not available on all SDA/SCL I/Os

Description

Only PB6,7,8,9,13,14, PC0,1 and PG7,8 a can effectively be configured in I2C Fm+ driving mode. Setting I2C1_FMP, I2C2_FMP and I2C3_FMP in the SYSCFG_CFGR1 register has no effect on PB10, PB11, PF0, PF1, PG13, PG14.

Note: When using the I/O without 20 mA Fm+ drive, it is still possible to reach 1 MHz Fm+ speed, with limited bus load.

Workaround

None.

2.6.2 Wrong behavior related with MCU Stop mode when wakeup from Stop mode by I2C peripheral is disabled

Description

When wakeup from Stop mode by I2C peripheral is disabled (WUPEN = 0) and the MCU enters Stop mode while a transaction is on-going on the I²C bus, the following wrong operations may occur:

1. BUSY flag may be 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. This failure may occur in master mode of the I2C peripheral used in multi-master I²C-bus environment.
2. If I²C-bus clock stretching is enabled in I2C peripheral (NOSTRETCH = 0), the I2C peripheral may pull SCL low as long as the MCU remains in Stop mode, suspending all I²C-bus activity during that time. This may occur when the MCU enters Stop mode during the address phase of an I²C-bus transaction, in low period of SCL. This failure may occur in slave mode of the I2C peripheral or, in master mode of the I2C peripheral used in multi-master I²C-bus environment. Its probability depends on the timing.

Workaround

Disable the I2C peripheral (PE=0) before entering Stop mode and re-enable it in Run mode.

2.6.3 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 specify 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.

2.6.4 Spurious Bus Error detection in master mode

Description

In master mode, a bus error can be detected by mistake, so the BERR flag can be wrongly raised in the status register. This will generate a spurious Bus Error interrupt if the interrupt is enabled. A bus error detection has no effect on the transfer in master mode, therefore the I2C transfer can continue normally.

Workaround

If a bus error interrupt is generated in master mode, the BERR flag must be cleared by software. No other action is required and the on-going transfer can be handled normally.

2.6.5 I2C3 analog filters requires that both PC0/PC1 or both PG7/PG8 are configured as I2C3 alternate function

Description

I2C3 analog filters can be enabled for PC0 and/or PC1 only if both IOs are configured in I2C3 Alternate Function mode. For example if you use PC0 for clock and PB4 for data , PC0 filter can be enabled only if PC1 is configured in I2C3 Alternate Function mode too.

I2C3 filter can be enabled for PG7 and/or PG8 only if both IOs are configured in I2C3 Alternate Function mode. For example if you use PG7 for clock and PC1 for data , PG7 filter can be enabled only if PG8 is configured in I2C3 Alternate Function mode too.

Workaround

Use both PC0/PC1 as I2C3 alternate functions or use both PG7/PG8 as I2C3 alternate functions if the analog filter is needed.

2.6.6 10-bit 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 (c5-bit header + 2 MSBs of the address + direction bit) 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, 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

2.6.7 START bit is not cleared when the address is not acknowledged by the slave device

Description

In the following conditions

- The I2C is used as master
- 10-bit addressing mode is used
- The Slave device doesn't acknowledge:
  - Either the 10-bit address header in case of write.
  - Or the 8 LSBs of the address in case of read.

the START bit will never be cleared by hardware, and the I2C master will not be able to start a new transfer:

Workaround:

Go through the following sequence:

- Wait until STOP condition detection (i.e. STOPF = 1)
- Disable the I2C peripheral
- Wait at least three APB clock cycles
- Re-enable the I2C peripheral.

2.7 SDMMC peripheral limitations

2.7.1 Wrong CCRCFAIL status after a response without CRC is received

Description

The CRC is calculated even if the response to a command does not contain any CRC field. As a consequence, after the SDIO command IO_SEND_OP_COND (CMD5) is sent, the CCRCFAIL bit of the SDMMC_STA register is set.

Workaround

The CCRCFAIL bit in the SDMMC_STA register shall be ignored by the software. CCRCFAIL must be cleared by setting CCRCFAILC bit of the SDMMC_ICR register after reception of the response to the CMD5 command.

2.7.2 Clock division per 255 is not possible

Description

When CLKDIV divide factor in SDMMC_CLKCR register is equal to 255, the SDMMC_CK clock output is not provided.

Workaround

Do not use CLKDIV value equals to 255: use clock divider factors from 0 to 254.

2.7.3 Wait for response bits “10” configuration does not work correctly

Description

The Wait for response bits configuration “10” (WAITRESP in the SDMMC_CMD register) does not work correctly. When WAITRESP bits are programmed to ’10’ Command Path State Machine (CPSM) waits for a non-existing response.

Workaround

Do not use WAITRESP value equal to “10” when sending a command without a response.

Use WAITRESP value equal to “00” to indicate to SDMMC CPSM that no response is expected.

2.7.4 MMC stream write of less than 8 bytes does not work correctly

When SDMMC host starts a stream write (WRITE_DAT_UNTIL_STOP CMD20), the number of bytes to transfer is not known by the card.

The card will write data from the host until a STOP_TRANSMISSION (CMD12) command is received.

The WAITPEND bit 9 of SDMMC_CMD register is set to synchronize the sending of the STOP_TRANSMISSION (CMD12) command with the data flow.

When WAITPEND is set, the transmission of this command stays pending until 50 data bits including STOP bit remain to transmit.

For a stream write of less than 8 bytes, the STOP_TRANSMISSION (CMD12) command should be started before the data transfer starts. Instead of this, the data write and the command sending are started simultaneously.

It implies that when less than 8 bytes have to be transmitted, (8 - DATALENGTH) bytes are programmed to 0xFF in the card after the last byte programmed (where DATALENGTH is the number of data bytes to be transferred).

Workaround

Do not use stream write WRITE_DAT_UNTIL_STOP (CMD20) with a DATALENGTH less then 8 bytes.

Use set block length (SET_BLOCKLEN: CMD16) followed by single block write command (WRITE_BLOCK: CMD24) instead of stream write (CMD20) with desired block length.

2.8 bxCAN peripheral limitations

2.8.1 bxCAN time-triggered mode is not supported

Description

The time-triggered communication mode described in the reference manual is not supported, and so time-stamp values are not available. TTCM bit must be kept cleared in the CAN_MCR register (time-triggered communication mode disabled).

Workaround

None.

2.9 SPI limitations

2.9.1 BSY bit may stay high at the end of a data transfer at 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.

Workaround

To ensure BSY bit flag is always reset at the end of a frame communication, disable the SPI by software before the communication is finished by following sequence:

- write last data to data register to transmit the data,
- poll TXE till it becomes high to ensure the data transfer has started,
- disable SPI by clearing SPE while the last data transfer is still on going,
- poll the BSY bit till it becomes low.

2.10 SWPMI peripheral limitations

2.10.1 SUSPENDED mode entry delayed

Description

When activating the SWPMI by setting the SWPEN bit in SWPMI_CR register, the SWPMI will rise the SWPMI_IO to 1.8V, send a short transition sequence and 14 idle bits before switching to SUSPENDED mode. As a consequence, the SRF flag is set and the SUSP flag is cleared in SWPMI_ISR register.

Workaround

None.

2.10.2 SUSPENDED mode never entered

Description

When activating the SWPMI by setting the SWPEN bit in SWPMI_CR register, the SWPMI will generate the following sequence in loop: rise the SWPIO, send a short transition sequence and 14 idle bits. As a consequence, SWP stays in ACTIVATED state, and never switch to SUSPENDED state.

Workaround

Keep SWPMI1SEL bit cleared in RCC_CCIPR register to select PCLK1 as SWPMI clock source, and configure the PCLK1 prescaler to feed the SWPMI with a clock frequency below or equal to 8 MHz.

2.10.3 SRF flag not set

Description

If the SWPMI1SEL bit is set in RCC_CCIPR register to select HSI as SWPMI clock source, when receiving a resume by slave, the SRF flag may not be set. Nevertheless, the SWPMI is switching correctly from SUSPENDED to ACTIVATED when receiving a RESUME by slave. Therefore frame reception is not impacted.

Workaround

None.

2.10.4 SWP SUSPENDED mode not supported when STM32L4 is in Stop 0 or Stop 1 mode

Description

STM32L4 cannot enter Stop 0 or Stop 1 mode if SWPMI is activated and is in SUSPENDED state.

Workaround

Deactivate the SWP bus before requesting entry in Stop 0 or Stop 1 mode. Refer to the SWPMI section in the product reference manual for the deactivation procedure.

2.10.5 SWPMI_IO transceiver bypass mode is not functional

Description

When the internal SWPMI transceiver is bypassed by setting the SWP_TBYP bit in the SWPMI_OR register, SWPMI1_RX mapped on PB14 is forced in output mode instead of input mode.

Workaround

None. The internal transceiver must be used.

2.11 RTC peripheral limitations

2.11.1 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

The false tamper event detection cannot be avoided, but the backup registers erase can be avoided by setting TAMPxNOERASE bit before clearing TAMPxE bit. The two bits must be written in two separate RTC_TAMPCR write accesses.

2.12 USART limitations

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

Description

In the ISO7816, when a character parity error is incorrect, the Smartcard receiver should transmit a NACK error signal at (10.5 +/- 0.2) etu 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 +/-0.2) etu after the character START bit falling edge.

The USART peripheral used in Smartcard mode doesn't respect the (11 +/-0.2) etu timing, and when the NACK falling edge arrives at 10.68 etu or later, the USART might misinterpret this transition as a START bit even if the NACK is correctly detected.

Workaround

None.

2.12.2 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 in the following conditions:

- CTS hardware flow control is enabled;
- D1 is being transmitted;
- abreak transfer is requested before the end of D1 transfer;
- nCTS is de-asserted before the end of transfer of D1.

Workaround

If the application needs to detect the end of transfer of the data, the break request should be done after making sure that the TC flag is set.

2.12.3 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.

2.13 COMP peripheral limitations

2.13.1 Comparators propagation delay is longer than expected for input steps higher than 200 mV

Description

Table 6 summarizes the comparators propagation delay values. The propagation delay for steps higher than 200 mV is out of targeted specification.

Table 6. COMP characteristics(1)

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
t_D	Propagation delay for 200 mV step with 100 mV overdrive	High-speed mode	$V_{DDA} \geq 2.7$ V	-	55	80	ns
			$V_{DDA} < 2.7$ V	-	65	100	
		Medium mode	$V_{DDA} \geq 2.7$ V	-	0.55	0.9	µs
			$V_{DDA} < 2.7$ V	-	0.65	1.0	
		Ultralow-power mode	$V_{DDA} \geq 2.7$ V	-	5	12	
			$V_{DDA} < 2.7$ V	-	5	12	
	Propagation delay for step > 200 mV with 100 mV overdrive on positive inputs	High-speed mode	-	-	0.1	1.0	µs
		Medium mode	-	-	2	3	
		Ultralow-power mode	-	-	8	24	

1. Data guaranteed by design, not tested in production, unless otherwise specified.

Workaround

None.

2.13.2 Comparators output cannot be configured in open-drain

Description

Comparators output are always forced in push-pull mode whatever the GPIO output type configuration bit value.

Workaround

None.

2.13.3 COMP1 and COMP2 configuration lost with software reset

Description

The Comparators 1 and 2 control and status registers (COMP1_CSR and COMP2_CSR) should be protected from software changes when the LOCK bit is set. Therefore these registers should be reset only by a system reset. However, it is possible to reset these registers by setting SYSCFGRST in the RCC_APB2RSTR.

Workaround

None.

2.14 LPTIM peripheral limitations

2.14.1 Low power timer 1 (LPTIM1) outputs cannot be configured in open-drain

Description

LPTIM1 outputs are always forced in push-pull mode whatever the GPIO output type configuration bit value.

Workaround

None.

2.15 LPUART peripheral limitations

2.15.1 Low power UART1 (LPUART1) outputs cannot be configured in open-drain

Description

LPUART1 outputs are always forced in push-pull mode whatever the GPIO output type configuration bit value.

Workaround

None.

2.16 TSC peripheral limitations

2.16.1 Inhibited acquisition in short transfer phase configuration

Description

The input buffer of the I/O is normally masked outside the transfer window time, then sampled twice before being checked for acquisition. Such check is normally performed on

the last TSC clock cycle of the transfer of charge phase. When the transfer of charge duration is less than three cycles the acquisition is inhibited.

Workaround

The following configurations are forbidden:

1. The PGPSC[2:0] field set to 000 and the CTPL[3:0] field to 0000 or 1111
2. The PGPSC[2:0] field set to 111 and the CTPL[3:0] field to 0000

3 Revision history

Table 7. Document revision history

Date	Revision	Changes
26-Jan-2016	1	Initial release.
07-Dec-2016	2	Updated *Table 4: Summary of silicon limitation* and added footnote *1*. Added *Section 2.1.11: PCPROP area within a single Flash memory page becomes unprotected at RDP change from level 1 to level 0*, *Section 2.1.12: Data Cache might be corrupted during Flash Read While Write operation*, *Section 2.1.13: MSI frequency overshoot upon Stop mode exit*, *Section 2.1.14: Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled*, *Section 2.3.4: First nibble of data is not written after dummy phase*, *Section 2.3.5: Wrong data can be read in memory-mapped after an indirect mode operation*, *Section 2.4.5: Unexpected end of transfer on DMA1 when using DMA1 channel 2 for ADC2 while DMA2 channel 4 is also used*, *Section 2.4.6: Unexpected end of transfer on DMA1 when using DMA1 channel 3 for ADC3 while DMA2 channel 5 is also used*, *Section 2.6.6: 10-bit master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave* and *Section 2.16: TSC peripheral limitations*. Updated *Section 2.1.6: HSI user trim is limited on some samples*, *Section 2.1.10: MSIRDY flag issue preventing entry in low power mode* and *Section 2.8.1: bxCAN time-triggered mode is not supported*. Removed former *Section 2.16.2: Calibration procedure does not work in PGA mode*.
24-Apr-2017	3	Updated *Table 4: Summary of silicon limitation*. Updated *Section 2.1.13: MSI frequency overshoot upon Stop mode exit*. Added *Section 2.1.15: OPAMP output: VDDA overconsumption*, *Section 2.1.16: Spurious BOR when entering Stop mode after short Run sequence* and *Section 2.6.7: START bit is not cleared when the address is not acknowledged by the slave device*.