STM32L496xx STM32L4A6xx Errata sheet

STM32L496xx and STM32L4A6xx device limitations

Applicability

This errata sheet applies to silicon revisions of the STMicroelectronics STM32L496xx and STM32L4A6xx, as shown in *Table 2*. The STM32L496xx and STM32L4A6xx families feature 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 1*. The products are identifiable by the revision code marked below the order code on the device package, as shown in *Table 2*.

Table 1. Device summary

Reference	Part number
STM32L496xx	STM32L496AE, STM32L496AG, STM32L496QE, STM32L496QG, STM32L496RE, STM32L496RG, STM32L496VE, STM32L496VG, STM32L496ZE, STM32L496ZG
STM32L4A6xx	STM32L4A6AG, STM32L4A6QG, STM32L4A6RG, STM32L4A6VG, STM32L4A6ZG

Table 2. Device variants(1)

Reference	Revision code marked on the device(2)
STM32L496xx	A
STM32L4A6xx	

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see *STM32L4x6 advanced ARM®-based 32-bit MCUs* reference manual (RM0411) for details on how to find the revision code).
2. Refer to the device datasheet for details on how to identify the revision code according to the packages.

Contents

List of tables

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

An errata notice of the STM32L496xx and STM32L4A6xx core is available from the following web address: http://infocenter.arm.com.

All the described limitations are minor and are related to the revision r0p1-v1 of the Cortex®-M4 FPU core. *Table 3* summarizes these limitations and their implications on the behavior of STM32L496xx and STM32L4A6xx 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 STM32L496xx STM32L4A6xx 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

Function	Section	Limitation	Rev A
System	*2.1.1*	*Dual bank boot not working in RDP level 1 when the boot in flash is selected by BOOT0 pin*	A
	2.1.2	*PCPROP area within a single Flash memory page becomes unprotected at RDP change from level1 to level0*	A
	2.1.3	*Data Cache might be corrupted during Flash Read While Write operation*	A
	2.1.4	*MSI frequency overshoot upon Stop mode exit*	A
	2.1.5	*Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled*	A
	2.1.6	*OPAMP output: VDDA overconsumption*	N
	2.1.7	*PCPROP area within a single Flash memory page becomes unprotected at RDP change from level1 to level0*	A
	2.1.8	*Spurious Brown Out Reset after short Run sequence*	A
FMC	*2.2.1*	*Dummy read cycles inserted when reading synchronous memories*	N
QUADSPI	*2.3.1*	*First nibble of data is not written after dummy phase*	A
	2.3.2	*Wrong data can be read in memory-mapped after an indirect mode operation*	A
ADC	*2.4.1*	*Injected queue of context is not available in case of JQM=0*	N
	2.4.2	*Wrong ADC conversion results when delay between calibration and first conversion or between 2 consecutive conversions is too long*	N
LPTIM	*2.5.1*	*Low-power timer 1 (LPTIM1) outputs cannot be configured as open-drain*	N
	2.5.2	*MCU may remain stuck in LPTIM interrupt when entering Stop mode*	N
TIM16	*2.6.1*	*HSE/32 is not available as TIM16 input capture if RTC is disabled or RTC clock source is not HSE*	A
LPUART	*2.7.1*	*Low-power UART1 (LPUART1) outputs cannot be configured as open-drain*	N
JTAG	*2.8.1*	*Full JTAG configuration without NJTRST pin cannot be used*	A

Table 4. Summary of silicon limitation (continued)

Function	Section	Limitation	Rev A
I2C	2.9.1	Wrong behavior related with MCU Stop mode when wakeup from Stop mode by I2C peripheral is disabled	A
	2.9.2	Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period	P
	2.9.3	Spurious bus error detection in Master mode	A
	2.9.4	Master new transfer cannot be launched if first part of the 10-bit address is NOT Acknowledged by the slave.	-
	2.9.5	START bit is not cleared when the address is not acknowledged by the slave device	-
	2.9.6	Last received byte can be lost when using Reload mode with NBYTES > 1	P
TSC	2.10.1	Inhibited acquisition in short transfer phase configuration	-
AES	2.11.1	Wrong TAG generation in GCM mode with encryption, for payloads smaller than 128 bits	A
SDMMC	2.12.1	MMC stream write of less than 7 bytes does not work correctly	A
bxCAN	2.13.1	bxCAN Time-triggered mode not supported	N
OTG_FS	2.14.1	Data FIFO gets corrupted if the write sequence to the Transmit FIFO is interleaved with other OTGFS register access	A
USART	2.15.1	nRTS is active while RE or UE = 0	A
COMP	2.16.1	Comparators output cannot be configured in open-drain	N

2.1 System

2.1.1 Dual bank boot not working in RDP level 1 when the boot in flash is selected by BOOT0 pin

Description

When the Read Protection (RDP) = Level 1, nSWBoot0 option byte = 1 and BOOT0 pin = 0 the dual bank boot is not working (BFB2 option byte = 1).

The user code will only be executed when BANK 0 is valid.

Workaround

The boot in flash selection must be configured thanks to the option bytes setting instead of BOOT0 pin.

Set nSWBoot0 option byte = 0, nBOOT0 option byte = 1 and BFB2 option byte = 1.

This setting allows the correct check of the address 0 of the 2 memory banks, in order to execute the correct one.

2.1.2 PCPROP area within a single Flash memory page becomes unprotected at RDP change from level1 to level0

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 level1 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 such that it crosses limits of at least one Flash memory page.

2.1.3 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();
```

2.1.4 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

1. Switch to HSI
2. Shutdown MSI
3. Wait for MSIRDY to go low (after 6 MSI clock cycles)
4. Mask_Interrupts (Set PRIMASK)
5. Enter in STOP mode with request to wakeup on MSI
6. Enable MSI
7. Wait for MSIRDY to go high
8. Switch to MSI (required as the system clock remains HSI in case the MCU did not enter Stop due to an early wakeup event)
9. Unmask_Interrupts (Clear PRIMASK)

This workaround guarantees the best wakeup time.

2.1.5 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.

2.1.6 OPAMP output: VDDA overconsumption

Description

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

- A voltage V_{PAD} is applied on PA3 or PB0 with VDDA+ 50 mV < VPAD < VDDA + 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 previous conditions.

2.1.7 PCPROP area within a single Flash memory page becomes unprotected at RDP change from level1 to level0

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 level1 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 such that it crosses limits of at least one Flash memory page.

2.1.8 Spurious Brown Out Reset 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 on the supply voltage,

Table 5. minimum Run time

VDD 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 the previous table corresponds to the firmware execution time on the Core. There is no need to add a delay for the wakeup time. Note also that the MCO output length is longer than the firmware execution time.

This limitation applies to Stop 1 and Stop 2 modes.

Workaround

Ensure that the run time between two Stop sequences is long enough not to generate 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 in order to minimize power consumption.

2.2 FMC

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.

2.3 QUADSPI

2.3.1 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. Instead, use alternate bytes to substitute the dummy cycles. The same latency can be achieved if the number of dummy cycles to substitute with alternate-byte cycles is an integer multiple of the number of cycles required for transferring one alternate byte, as shown in the table:

QUADSPI mode	Number of cycles per alternate byte
4-data-line DDR	1
4-data-line SDR	2
2-data-line SDR	3
2-data-line SDR	4

For example, the latency corresponding to eight dummy cycles can be exactly substituted with one single alternate byte in 1-data-line SDR mode, but two alternate bytes are required in 2-data-line SDR mode. One single dummy cycle can only exactly be substituted in 4-data-line DDR mode, using one alternate byte.

Note: This is also applicable to dual-flash memory mode.

2.3.2 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

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 1 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 Wrong ADC conversion results when delay between calibration and first conversion or between 2 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.

2.5 LPTIM

2.5.1 Low-power timer 1 (LPTIM1) outputs cannot be configured as open-drain

Description

LPTIM1 outputs are set in push-pull mode regardless of the configuration of corresponding GPIO outputs.

Workaround

None.

2.5.2 MCU may remain stuck in LPTIM interrupt when entering Stop mode

Description

This limitation occurs when disabling the low power timer (LPTIM).

When the firmware clears the LPTIM_CR.ENABLE bit within a small time window around one LPTIM interrupt occurrence, then the LPTIM interrupt signal used to wake up the MCU from Stop mode may be frozen in active state. Consequently, when trying to enter Stop mode, this limitation prevents the MCU from entering low power mode and the firmware remains stuck in the LPTIM interrupt routine.

This limitation applies to all Stop modes and to all instances of the LPTIM. Note that the occurrence of this issue is very low.

Workaround

In order to disable a low power timer (LPTIMx) peripheral, do not clear its ENABLE bit in its respective LPTIMx_CR register. Instead, reset the whole LPTIMx peripheral via the RCC controller by setting and resetting its respective LPTIMxRST bit in RCC_APByRSTRz register.

2.6 TIM16

2.6.1 HSE/32 is not available as TIM16 input capture if RTC is disabled or RTC clock source is not HSE

Description

When the HSE/32 clock source is selected as input capture for the timer16 by setting TI1_RMP[2:0] = 101 into TIM16_OR1 register, the clock is not present if the RTC clock is not enabled and if the HSE RTC clock source is not selected.

Workaround

To get HSE/32 as input capture source for TIM16, the procedure to respect is:

1. enable the power controller clock (bit PWREN = 1 in the RCC_APB1ENR1 register),
2. disable the VBAT backup domain protection (bit DBP = 1 in the PWR_CR1 register),
3. enable RTC and select HSE as clock source for RTC (bits RTCSEL[1:0] = 11 and bit RTCEN = 1 in the RCC_BDCR register),
4. select the HSE/32 as input capture source for the timer 16 (TI1_RMP[2:0] = 101 into the TIM16_OR1 register).

Alternatively, TIM17 that implements same features as TIM16 without limitation can be used instead of TIM16.

2.7 LPUART

2.7.1 Low-power UART1 (LPUART1) outputs cannot be configured as open-drain

Description

LPUART1 outputs are set in push-pull mode regardless of the configuration of corresponding GPIO outputs.

Workaround

None.

2.8 JTAG

2.8.1 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.

2.9 I2C

2.9.1 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.

2.9.2 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 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.

2.9.3 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.

2.9.4 Master new transfer cannot be launched if first part of the 10-bit address is NOT 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

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

Description

In these 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

Apply the following sequence:

- Wait until STOP condition detection (i.e. STOPF = 1)
- Disable the I2C peripheral
- Wait at least 3x APB clock cycles
- Re-enable the I2C peripheral

2.9.6 Last received byte can be lost when using Reload mode with NBYTES > 1

Description

The limitation can occur in master mode when reload mode is used (needed for transferring more than 255 bytes), or in slave byte control mode (SBC=1 in the I2C_CR1 register). The limitation occurs only when NBYTES > 1.

In Reload mode (RELOAD=1 in the I2C_CR2 register) with NBYTES programmed with a value N in the I2C_CR2, the Transfer Complete Reload flag (TCR) is set in the I2C_ISR register when the last byte is received in the shift register, even if not yet transferred in the Receive Data Register because the byte N-1 is not yet read.

The last received data is definitively lost (never transferred in the Data Register) if the data N-1 is read between 0 and 4 APB clock cycles before the TCR flag is being set is the I2C_ISR register.

Workaround

In slave byte control mode : Use the Reload mode with NBYTES=1.

In master mode: Do not use the Reload mode. If the number of bytes to be transferred is greater than 255 bytes, the total transfer should be split in several transfers not exceeding 255 bytes, separated by Repeated Start conditions.

Note that the use of DMA could manage that the data N-1 is always transferred before the 4 APB cycles preceding the TCR flag. However this must be evaluated carefully for each application depending on the bus bandwidth, maximum latency, and DMA channel priority.

2.10 TSC

2.10.1 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 0001
2. The PGPSC[2:0] field set to 111 and the CTPL[3:0] field to 0000

2.11 AES[1]

2.11.1 Wrong TAG generation in GCM mode with encryption, for payloads smaller than 128 bits

Description

When the AES is configured in GCM mode with encryption, the TAG generation is wrong during the Final phase if the size of the last plain text block of the payload is lower than 128 bits.

Workaround

During payload phase and before inserting a last payload block smaller than 128 bits, pursue the following steps:

- Switch the AES mode to CTR mode by writing the bitfield CHMOD[2:0] = 010b in the AES_CR register.
- Pad the last block smaller than 128 bits with zeros until reaching the size of 128 bits, then insert it as input to the AES.
- Upon completion, read the 128-bit generated data from the AES_DOUTR register and store it as intermediate data.
- Change the AES mode to GCM mode by writing the bitfield CHMOD[2:0] = 011b in the AES_CR register.
- Select Final phase by writing the bitfield GCMPH[1:0] = 11b in the AES_CR register.
- In the intermediate data, set to zero the bits corresponding to the padded bits of the last block of payload, then insert the resulting data as input to the AES.
- Upon completion, read the AES_DOUTR register. The data itself has no importance and can be ignored. This step is required to set up the internal state machine in a way for it to handle correctly the TAG generation during the GCM Final phase.
- Apply the normal Final phase as usual.

Although the reference manual indicates that mode changes should be avoided when the AES is enabled, the AES does not misbehave when this workaround is applied.

2.12 SDMMC

2.12.1 MMC stream write of less than 7 bytes does not work correctly

Description

Stream write initiated with WRITE_DAT_UNTIL_STOP command (CMD20) does not define the amount of data bytes to store. The card keeps storing data coming in from the SDMMC host until it gets a valid STOP_TRANSMISSION (CMD12) command. The commands are streamed on a line separate from data line, with common clock line.

1. Limitation valid only fo STM32LA6xx

As the STOP_TRANSMISSION command is 48-bit long and due to the bus protocol, the STOP_TRANSMISSION command start bit must be advanced by 50 clocks with respect to the stop bit of the data bitstream.

Therefore, for small data chunks of up to 6 bytes, SDMMC hosts should normally operate such that, the start of the STOP_TRANSMISSION (CMD12) command streaming precedes the start of the data streaming.

The microcontrollers duly anticipate the STOP_TRANSMISSION command streaming start, with respect to the data bitstream end. WAITPEND (bit 9 of SDMMC_CMD register) must be set for this mechanism to operate.

However, a failure occurs in case of small data chunks of up to 6 bytes. Instead of starting the STOP_TRANSMISSION command 50 clocks ahead of the data bitstream stop bit, the SDMMC peripheral on STM32L496xx and STM32L4A6xx MCUs starts the command along with the first bit of the data bitstream. As the command is longer than the data, it ends a number of clocks behind the data that the STM32L496xx and STM32L4A6xx software intended to store onto the card by setting the DATALENGTH register. During the clocks in excess, the SDMMC peripheral keeps the data line in logical-one level.

As a consequence, the card intercepts more data and updates more memory locations than the number set in DATALENGTH. The spuriously updated locations of memory receive 0xFF values.

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.

2.13 bxCAN

2.13.1 bxCAN Time-triggered mode 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.

2.14 OTG_FS

2.14.1 Data FIFO gets corrupted if the write sequence to the Transmit FIFO is interleaved with other OTGFS register access

Description

When the OTG Full Speed cell is in Host or Device mode, interrupting the write sequence in the transmit FIFO by any access (Read or Write) to OTG registers will corrupt the next data written to the transmit FIFO.

Workaround

Ensure that the transmit FIFO write accesses cannot be interrupted by a procedure performing accesses to the USB cell registers.

2.15 USART

2.15.1 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.

2.16 COMP

2.16.1 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.

3 Revision history

Table 6. Document revision history

Date	Revision	Changes
24-Feb-2016	1	Initial release.
05-Sep-2016	2	Updated: – *Table 4: Summary of silicon limitation* – *Section 2.12.1: MMC stream write of less than 7 bytes does not work correctly* Added: – *Section 2.1.2: PCPROP area within a single Flash memory page becomes unprotected at RDP change from level1 to level0*, *Section 2.4.2: Wrong ADC conversion results when delay between calibration and first conversion or between 2 consecutive conversions is too long*, *Section 2.5: LPTIM*, *Section 2.6: TIM16*, *Section 2.7: LPUART*, *Section 2.8: JTAG*, *Section 2.9.4: Master new transfer cannot be launched if first part of the 10-bit address is NOT Acknowledged by the slave.*, *Section 2.10: TSC*, *Section 2.11: AES*
09-Jan-2017	3	Updated: – *Table 4: Summary of silicon limitation* – *Applicability* Added: – *Section 2.1.1: Dual bank boot not working in RDP level 1 when the boot in flash is selected by BOOT0 pin*, *Section 2.1.3: Data Cache might be corrupted during Flash Read While Write operation*, *Section 2.1.4: MSI frequency overshoot upon Stop mode exit*, *Section 2.1.5: Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled*, *Section 2.3.1: First nibble of data is not written after dummy phase*, *Section 2.3.2: Wrong data can be read in memory-mapped after an indirect mode operation*

Table 6. Document revision history (continued)

Date	Revision	Changes
01-Mar-2017	4	Updated: – *Table 4: Summary of silicon limitation*, – *Section 2.1.4: MSI frequency overshoot upon Stop mode exit*, *Section 2.3.1: First nibble of data is not written after dummy phase* Added: – *Section 2.1.7: PCPROP area within a single Flash memory page becomes unprotected at RDP change from level1 to level0*, *Section 2.9.5: START bit is not cleared when the address is not acknowledged by the slave device*
08-Jun-2017	5	Replaced former *Silicon identification* with *Applicability*, and former *Table 2: Device identification* with *Table 2: Device variants*. Updated: – *Table 4: Summary of silicon limitation*, *Section 2: STM32L496xx STM32L4A6xx silicon limitations* Added: – *Section 2.1.8: Spurious Brown Out Reset after short Run sequence*, *Section 2.1.6: OPAMP output: VDDA overconsumption*, *Section 2.9.6: Last received byte can be lost when using Reload mode with NBYTES > 1*, *Section 2.14: OTG_FS*, *Section 2.5.2: MCU may remain stuck in LPTIM interrupt when entering Stop mode*