STM32L433xx STM32L443xx Errata sheet

STM32L433xx and STM32L443xx device limitations

Silicon identification

This document applies to the part numbers of STM32L433xx and STM32L443xx devices listed in *Table 1* and their silicon revisions 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 RM0394.

Table 1. Device summary

Reference	Part numbers
STM32L433xx	STM32L433CC, STM32L433RC, STM32L433VC, STM32L433CB, STM32L433RB
STM32L443xx	STM32L443CC, STM32L443RC, STM32L443VC

Table 2. Device identification(1)

Reference	Revision code marked on the device(2)
STM32L433xx	‘A’, ‘Z’
STM32L443xx	‘A’, ‘Z’

1. The REV_ID bits in the DBGMCU_IDCODE register indicate the revision code of the device (see the reference manual for details on the revision code).
2. Refer to datasheet for details on how to identify the silicon revision code on different types of package.

Contents

1 Summary of device limitations

The following table gives a quick references to all documented limitations of the STM32L433xx and STM32L443xx devices and their status:

A = workaround available

N = no workaround available

P = partial workaround available

Table 3. Summary of device limitations

Function	Section	Limitation	Status
			Rev. 'A', 'Z'
Core	*2.1.1*	*Interrupted loads to stack pointer can cause erroneous behavior*	A
System	*2.2.1*	*MSIRDY flag issue preventing entry in low power mode*	A
	2.2.2	*PCPROP area within a single Flash memory page becomes unprotected at RDP change from level1 to level0*	A
	2.2.3	*MSI frequency overshoot upon Stop mode exit*	A
	2.2.4	*Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled*	A
RCC	*2.3.1*	*HSI48 ready interrupt capability is not supported*	A
QUADSPI	*2.4.1*	*Wrong data can be read in memory-mapped after an indirect mode operation*	A
	2.4.2	*First nibble of data is not written after dummy phase*	A
ADC	*2.5.1*	*Wrong ADC conversion results when delay between calibration and first conversion or between 2 consecutive conversions is too long*	N
COMP	*2.6.1*	*Comparator outputs cannot be configured as open-drain*	N
	2.6.2	*COMP1 and COMP2 configuration lost with software reset*	N
TSC	*2.7.1*	*Inhibited acquisition in short transfer phase configuration*	A
AES(1)	*2.8.1*	*Burst read or write accesses are not supported by AES*	A
	2.8.2	*Wrong TAG generation in GCM mode with encryption, for payloads smaller than 128 bits*	A
TIM16	*2.9.1*	*HSE/32 is not available as TIM16 input capture if RTC is disabled or RTC clock source is not HSE*	A
LPTIM	*2.10.1*	*Low-power timer 1 (LPTIM1) outputs cannot be configured as open-drain*	N

Table 3. Summary of device limitations (continued)

Function	Section	Limitation	Status
			Rev. 'A', 'Z'
I2C	*2.11.1*	*Wrong behavior related with MCU Stop mode when wakeup from Stop mode by I2C peripheral is disabled*	A
	2.11.2	*Wrong data sampling when data set-up time (tSU;DAT) is shorter than one I2CCLK period*	P
	2.11.3	*Spurious bus error detection in master mode*	A
	2.11.4	*10-bit master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave*	A
	2.11.5	*I2C3 analog filter activation requires that both PC0/PC1 are configured as I2C3 alternate function*	A
LPUART	*2.12.1*	*Low-power UART1 (LPUART1) outputs cannot be configured as open-drain*	N
SPI	*2.13.1*	*BSY bit may stay high at the end of a data transfer in slave mode*	A
SDMMC	*2.14.1*	*Wrong CCRCFAIL status after a response without CRC is received*	A
	2.14.2	*MMC stream write of less than 7 bytes does not work correctly*	A
bxCAN	*2.15.1*	*bxCAN time-triggered mode is not supported*	N
DBG	*2.16.1*	*Full JTAG configuration without NJTRST pin cannot be used*	A

1. Only applicable to STM32L443xx

2 Description of device limitations

2.1 Core

Errata notices for the ARM® Cortex® cores are available from http://infocenter.arm.com.

The limitations described in this section are related to the revision r0p1-v1 of the Cortex®-M4 FPU core.

2.1.1 Interrupted loads to stack pointer can cause erroneous behavior

This limitation is registered under ARM ID number 752419 as Cat 2, with minor impact to the silicon devices using this ARM core.

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

2.2 System

2.2.1 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).

2.2.2 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.

2.2.3 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. Select HSI as system clock, shut MSI clock off, and select MSI as clock source upon wakeup from Stop mode.

   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.

2.2.4 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.

2.3 RCC

2.3.1 HSI48 ready interrupt capability is not supported

Description

HSI48 ready interrupt feature described in the reference manual is not supported. The bit HSI48RDYIE in the register RCC_CIER is stuck at 0. It is not possible to set it to interrupt the CPU when the oscillator is ready.

Workaround

After switching the HSI48 internal oscillator on by setting the bit HSI48ON in the RCC_CRRCR register, the software has to poll the HSI48RDY bit in the RCC_CRRCR register to know when the oscillator is ready to provide the clock to RNG and/or SDMMC and/or USB.

2.4 QUADSPI

2.4.1 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 the Quad-SPI peripheral entered in memory-mapped mode without reset of both LSB bits in the QUADSPI_AR[1:0] address register.

Workaround

The QUADSPI_AR register must be reset just before entering in memory-mapped mode. This can be done in two different ways, depending on the current Quad-SPI operating mode:

1. Indirect read mode:
   a) Reset the address register.
   b) Make an abort request to stop the reading and clear the busy bit.
   c) Enter in memory-mapped mode.
2. Indirect write mode:
   a) Reset the address register.
   b) Enter in memory-mapped mode.

Note: The QUADSPI_DR register should not be written after resetting the address register.

2.4.2 First nibble of data is not written after dummy phase

Description

The first nibble of data to be written to an external flash is lost if:

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

Workaround

Do not use dummy cycles for creating latency between address phase and data phase, in indirect write mode. 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	4
1-data-line SDR	8

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.

2.5 ADC

2.5.1 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.

2.6 COMP

2.6.1 Comparator outputs cannot be configured as open-drain

Description

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

Workaround

None.

2.6.2 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 RCC_APB2RSTR.

Workaround

None.

2.7 TSC

2.7.1 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

2.8 AES

2.8.1 Burst read or write accesses are not supported by AES

Description

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

Workaround

The workaround consists in preventing compilers from generating LDM, STM, LDRD and STRD instructions to access the AES peripheral.

In general, this can be achieved through organizing the source code such as to avoid consecutive read or write accesses to neighboring addresses in lower-to-higher order. In case where consecutive read or write accesses to neighboring addresses cannot be avoided, order the source code such as to access higher address first.

2.8.2 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.

2.9 TIM16

2.9.1 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).

2.10 LPTIM

2.10.1 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.

2.11 I2C

2.11.1 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 operation 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 configuration, operating clock frequency of I2C peripheral and the I²C-bus timing.

Workaround

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

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

Description

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

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

The I²C-bus SDA line is not correctly sampled when $t_{SU;DAT}$ is smaller than one I2CCLK (I²C-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 I2CCLK frequency to get I2CCLK period within the transmitter minimum data set-up time. Alternatively, increase transmitter's minimum data set-up time.

2.11.3 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 I^2C 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.

2.11.4 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 byte of the 10-bit address (5-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, then proceed as follows:

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

2.11.5 I2C3 analog filter activation requires that both PC0/PC1 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, for using PC0 as clock and PB4 as data, PC0 filter can be enabled only if PC1 is configured in I2C3 alternate function mode, too.

Workaround

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

2.12 LPUART

2.12.1 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.

2.13 SPI

2.13.1 BSY bit may stay high at the end of a data transfer in slave mode

Description

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

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

As a consequence, the end of data transaction may be not recognized when the software needs to monitor it (e.g. at the end of session before entering the low-power mode or before

direction of data line has to be changed at half duplex bidirectional mode). The BSY flag is unreliable to detect the end of any data sequence transaction.

Workaround

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

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

When the aforementioned workaround is not applicable, the following sequence can be used to prevent the synchronization issue at SPI transmit mode:

1. Write last data to data register
2. Poll TXE until it becomes high to ensure the data transfer has started
3. Disable SPI by clearing SPE while the last data transfer is still ongoing
4. Poll the BSY bit until it becomes low

The BSY flag now works correctly and can be used to recognize the end of the transaction.

Note: The latter workaround can be used only when the CPU has enough performance to disable the SPI after the TXE event is detected while the data frame transfer is still ongoing. It is impossible to achieve it when the ratio between CPU and SPI clocks is low and in particular for short data frames.

2.14 SDMMC

2.14.1 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.

2.14.2 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.

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.

STM32L4x3xx 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 STM32L4x3xx 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 STM32L4x3xx 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 command (CMD20) with DATALENGTH set to less than 7.

Instead, use SET_BLOCKLEN command (CMD16) followed with single-block write command WRITE_BLOCK (CMD24), with a desired block length.

2.15 bxCAN

2.15.1 bxCAN time-triggered mode is not supported

Description

The time-triggered communication mode described in the reference manual is not supported 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.

2.16 DBG

2.16.1 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.

3 Revision history

Table 4. Document revision history

Date	Revision	Changes
18-Apr-2016	1	Initial release.
05-Sep-2016	2	**Added:** – *Section 2.1.2: PCPROP area within a single Flash memory page becomes unprotected at RDP change from level1 to level0* – *Section 2.3.3: Spurious bus error detection in master mode* – *Section 2.13.1: Inhibited acquisition in short transfer phase configuration*
03-Dec-2016	3	**Added:** – Silicon revision Z – *Section 2.2.3: MSI frequency overshoot upon Stop mode exit* – *Section 2.2.4: Internal voltage reference corrupted upon Stop mode entry with temperature sensing enabled* – *Section 2.4.1: Wrong data can be read in memory-mapped after an indirect mode operation* – *Section 2.4.2: First nibble of data is not written after dummy phase* – *Section 2.11.4: 10-bit master mode: new transfer cannot be launched if first part of the address has not been acknowledged by the slave* **Modified:** – Document structure: cover page, titles, legend, removal of the list of tables, *Table 3: Summary of device limitations* – The order of functions aligned with the reference manual – JTAG function replaced with more general DBG – *Section 2.13.1: BSY bit may stay high at the end of a data transfer in slave mode*