STM32F328C8 Errata sheet

STM32F328C8 Rev Z device limitations

Silicon identification

This errata sheet applies to revision Z of the STMicroelectronics STM32F328C8 product. These families feature an Arm® 32-bit Cortex®-M4 FPU core, for which an errata notice is also available (see *Section 1* for details).

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

The products are identifiable as shown in *Table 1*:

- By the revision code marked below the order code on the device package
- By the last three digits of the Internal order code printed on the box label

Table 1. Device identification(1)(2)

Order code	Revision code marked on device
STM32F328C8	"Z"

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see the STM32F328C8 reference manual for details on how to find the revision code).
2. Refer to datasheet for the device marking.

Contents

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

An errata notice of the STM32F328C8 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 2* summarizes these limitations and their implications on the behavior of STM32F3xxxx devices.

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

1.2 VDIV or VSQRT instructions might not complete correctly when very short ISRs are used

Description

On Cortex-M4 with FPU core, 14 cycles are required to execute a VDIV or VSQRT instruction.

This limitation is present when the following conditions are met:

- A VDIV or VSQRT is executed.
- The destination register for VDIV or VSQRT is one of s0 - s15.
- An interrupt occurs and is taken.
- The ISR being executed does not contain a floating point instruction.
- 14 cycles after the VDIV or VSQRT is executed, an interrupt return is executed.

In this case, if there are only one or two instructions inside the interrupt service routine, then the VDIV or VQSRT instruction does not complete correctly and the register bank and FPSCR are not updated, meaning that these registers hold incorrect out-of-date data.

Workaround

Two workarounds are applicable:

- Disable lazy context save of floating point state by clearing LSPEN to 0 (bit 30 of the FPCCR at address 0xE000EF34).
- Ensure that every ISR contains more than two instructions in addition to the exception return instruction.

2 STM32F328C8 silicon limitations

Table 2 gives quick references to all documented limitations.

Legend for *Table 2* is as follows:

A = workaround available;

N = no workaround available;

P = partial workaround available,

‘-’ and grayed = fixed.

Table 2. Summary of silicon limitations

Links to silicon limitations		Revision Z
Section 2.1: System limitations	*Section 2.1.1: Wakeup sequence from Standby mode when using more than one wakeup source*	A
	Section 2.1.2: Full JTAG configuration without NJTRST pin cannot be used	A
	Section 2.1.3: CCM RAM write protection register SYSCFG_RCR not reset by system reset.	A
Section 2.2: ADC peripheral limitations	*Section 2.2.1: DMA overrun in dual interleaved mode with single DMA channel*	A
	Section 2.2.2: Sampling time shortened in JAUTO autodelayed mode	A
	Section 2.2.3: Injected queue of context not available in case of JQM = 0	N
	Section 2.2.4: Load multiple not supported by ADC interface	A
	Section 2.2.5: Possible voltage drop caused by a transitory phase when the ADC switches from a regular channel to an injected channel Rank 1	A
	Section 2.2.6: Overrun flag may not be set if converted data are not read before writing new data	A
	Section 2.2.7: ADC differential mode: common mode input range	N
	Section 2.2.8: Imprecise VREFINT calibration values	N
Section 2.3: SPI peripheral limitations	*Section 2.3.1: SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI is under DMA transaction near the end of transfer or end of transfer ‘-1’*	P
	Section 2.3.2: BSY bit may stay high at the end of a SPI data transfer in slave mode	A

Table 2. Summary of silicon limitations (continued)

Links to silicon limitations		Revision Z
Section 2.4: I2C peripheral limitations	*Section 2.4.1: 10-bit slave mode: wrong direction bit value after read header reception*	A
	Section 2.4.2: 10-bit combined with 7-bit slave mode: ADDCODE may indicate wrong slave address detection	N
	Section 2.4.3: Wakeup frames may not wake up the MCU mode when Stop mode entry follows I2C enabling	A
	Section 2.4.4: Wrong behaviors related with MCU Stop mode when wakeup from Stop mode by I2C peripheral disabled	A
	Section 2.4.5: Wakeup frame may not wake up from Stop if $t_{HD(STA)}$ is close to $t_{su(HSI)}$ in Fast-mode and Fast-mode Plus.	P
	Section 2.4.6: Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period	P
	Section 2.4.7: Spurious bus error detection in master mode	A
Section 2.5: USART peripheral limitation	*Section 2.5.1: When PCLK is selected as clock source for USART1, PCLK1 is used instead of PCLK2*	N
	Section 2.5.2: Start bit detected too soon when sampling for NACK signal from the smartcard	N
	Section 2.5.3: Break request can prevent the transmission complete flag (TC) from being set	A
	Section 2.5.4: nRTS is active while RE or UE = 0	A
	Section 2.5.5: Receiver timeout counter starting in case of two stop bit configuration	A
	Section 2.5.6: Data corruption due to noisy receive line	N
Section 2.6: GPIO peripheral limitation	*Section 2.6.1: GPIOx locking mechanism is not working properly for GPIOx_OTYPE register*	A

2.1 System limitations

2.1.1 Wakeup sequence from Standby mode when using more than one wakeup source

Description

The various wakeup sources are logically OR-ed in front of the rising-edge detector that generates the wakeup flag (WUF). The WUF needs to be cleared prior to the Standby mode entry, otherwise the MCU wakes up immediately.

If one of the configured wakeup sources is kept high during the clearing of WUF (by setting the CWUF bit), it may mask further wakeup events on the input of the edge detector. As a consequence, the MCU could not be able to wake up from Standby mode.

Workaround

To avoid this limitation, the following sequence should be applied before entering the Standby mode:

- Disable all used wakeup sources.
- Clear all related wakeup flags.
- Re-enable all used wakeup sources.
- Enter Standby mode

Note: When applying this workaround, if one of the wakeup sources is still kept high, the MCU enters the Standby mode but then it wakes up immediately generating the power reset.

2.1.2 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.

2.1.3 CCM RAM write protection register SYSCFG_RCR not reset by system reset.

Description

The CCM RAM write protection register SYSCFG_RCR cannot be reset by system reset. It can be reset only by POR reset.

Workaround

None.

If the application needs to write protect the CCM RAM and to remove the CCM RAM write protection without applying a POR reset, other solutions can be adopted such as:

- Protecting the CCM RAM against unwanted write operation using the MPU or
- Simply using the parity check feature allowing the detection of CCM RAM content corruption.

2.2 ADC peripheral limitations

2.2.1 DMA overrun in dual interleaved mode with single DMA channel

Description

DMA overrun conditions can be encountered when two ADCs are working in dual interleaved mode with a single DMA channel for both (MDMA[1:0]bits equal to 0b10 or 0b11). This limitation applies in Single, Continuous and Discontinuous mode.

Workaround

The MDMA [1:0] bits must be kept cleared and each ADC must have its own DMA channel enabled (dual DMA configuration).

2.2.2 Sampling time shortened in JAUTO autodelayed mode

Description

When the ADC is configured in JAUTO single conversion mode (CONT = 0), with autodelayed mode enabled (AUTDLY = 1). If the last regular conversion is read and a new regular trigger arrives before the JEOS bit is cleared, the first regular conversion sampling time is shortened by one cycle.

This does not apply for configuration where SMP = 000 (1.5 cycle sampling time), or if the interval between triggers is always above the auto-injected sequence conversion period.

Workaround

The sampling time can be increased by one clock cycle if the situation is foreseen.

2.2.3 Injected queue of context 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 leads to a queue overflow and is ignored.

Consequently, the ADC must be stopped before programming the JSQR register.

Workaround

None.

2.2.4 Load multiple not supported by ADC interface

Description

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

Workaround

The workaround consists in preventing compilers from generating LDM, STM, LDRD and STRD instructions.

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.

2.2.5 Possible voltage drop caused by a transitory phase when the ADC switches from a regular channel to an injected channel Rank 1

Description

Consider the following conditions:

- ADCx channel A is a regular channel.
- ADCx channel B is an injected channel (Rank 1).
- ADCx converts ADCx channel A followed by ADCx channel B.

When ADCx switches from channel A to channel B, if a drop is observed on the analog signal, on an I/O used for DAC out, on comparator input or operational amplifier, and on which mapped an ADCx channel C (which is not configured to be converted), the root cause is the analog channel multiplexer selecting the ADCx channel C for a transitory time window.

For example, when ADC2 is switching from the regular channel 12 to injected channel 3, and the DAC1 channel 1 on PA4 is enabled and configured to output a signal, a drop is observed on the DAC1 output signal because there is a transitory phase passing by ADC2 channel 1 which is available on PA4.

Workaround

In the case DAC1 channel 1 output is enabled and a transitory phase is passing by ADC2 channel 1, a workaround is to reduce the output impedance by enabling the DAC1 channel 1 output buffer.

For the DAC1 channel 2, DAC2 channel 1, COMP and OPAMP, a workaround can be to change the ADC channels ranking.

2.2.6 Overrun flag may not be set if converted data are not read before writing new data

Description

When converted data are read from the ADC_DR register during the very same APB cycle used to write data from a new conversion, the previously written data or the new data are lost, but the overrun flag (OVR) may not be set to 1.

Workaround

To avoid overrun errors, read the converted data before data from a new conversion are made available by the ADC.

2.2.7 ADC differential mode: common mode input range

Description

When the ADC is used in differential mode, the common mode input range is (VSSA + VREF+)/2 +/- 10%.

Workaround

None.

2.2.8 Imprecise VREFINT calibration values

Description

The devices with a data code less than 619 may have imprecise VREFINT calibration values.

Workaround

None.

2.3 SPI peripheral limitations

2.3.1 SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI is under DMA transaction near the end of transfer or end of transfer ‘-1’

Description

SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI is under DMA transaction near the end of transfer or end of transfer ‘-1’.

In the conditions listed below, the CRC may be frozen before the CRCNEXT bit is written, resulting in a CRC error:

- SPI is slave or master.
- Full duplex or simplex mode is used.
- CRC feature is enabled.
- SPI is configured to manage data transfers by software (interrupt or polling).
- A peripheral, mapped on the same DMA channel as the SPI, is doing DMA transfers.

Workaround

If the application allows it, use the DMA for SPI transfers.

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

Description

In slave mode, the 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 before entering low-power mode or modifying the SPI configuration (such as direction of the bidirectional mode).

Workaround

- When the SPI interface is in receive mode, the end of a transaction with the master can be detected by the corresponding RXNE event when this flag is set after the last bit of that transaction is sampled and the received data are stored.
- When the following sequence is used, the synchronization issue does not occur. The BSY bit works correctly and can be used to recognize the end of any transmission transaction (including when RXNE is not raised in bidirectional mode):
  a) Write the last data into data register.
  b) Poll TXE flag till it becomes high to make sure the data transfer has started.
  c) Disable the SPI interface by clearing SPE bit while the last data transfer is on going.
  d) Poll the BSY bit till it becomes low.

Note: The second workaround can be used only when the CPU is fast enough to disable the SPI interface after a TXE event is detected while the data frame transfer is ongoing. It cannot be implemented when the ratio between CPU and SPI clock is low and the data frame is particularly short. At this specific case, the timeout can be measured from the TXE event instead by calculating a fixed number of CPU clock cycles corresponding to the time necessary to complete the data frame transaction.

2.4 I2C peripheral limitations

2.4.1 10-bit slave mode: wrong direction bit value after read header reception

Description

Under specific conditions, the transfer direction bit DIR (bit 16 of status register I2C_ISR) is low instead of high after reception of the 10-bit addressing read header. Nevertheless, the I^2C operates correctly in slave transmission mode, and data can be sent using the TXIS flag.

To see the limitation, all the following conditions have to be fulfilled:

- I^2C has to be configured in 10-bit addressing mode (OA1MODE is set in the I2C_OAR1 register).
- The high LSBs of the I^2C slave address are equal to the 10-bit addressing read header value (i.e. OA1[7:3] = 11110, OA1[2] = OA1[9], OA1[1] = OA1[8] and OA1[0] = 1 in the I2C_OAR1 register).
- The I^2C receives the 10-bit addressing read header (0x 1111 0XX1) after the repeated start condition to enter slave transmission mode.

As a result, the DIR bit is incorrect in slave mode under specific conditions.

Workaround

If possible, do not use these four values as 10-bit addresses in slave mode:

- OA1[9:0] = 0011110001
- OA1[9:0] = 0111110011
- OA1[9:0] = 1011110101
- OA1[9:0] = 1111110111

If one of these addresses is the I2C slave address, the DIR bit must not be used in the firware.

2.4.2 10-bit combined with 7-bit slave mode: ADDCODE may indicate wrong slave address detection

Description

Under specific conditions, the ADDCODE (address match code) in the I2C_ISR register indicates a wrong slave address.

To see the limitation, all the following conditions have to be fulfilled:

- The I^2C slave address OA1 is enabled and configured in 10-bit mode (OA1EN = 1 and OA1MODE = 1)
- Another 7-bit slave address is enabled and the bits 1 to 7 of the 10-bit slave address OA1 are equal to the 7-bit slave address. One of the configurations below is set:
  - OA2EN = 1 and OA2MSK = 0 and OA1[7:1] = OA2[7:1]
  - OA2EN = 1 and OA2MSK = 1 and OA1[7:2] = OA2[7:2]
  - OA2EN = 1 and OA2MSK = 2 and OA1[7:3] = OA2[7:3]
  - OA2EN = 1 and OA2MSK = 3 and OA1[7:4] = OA2[7:4]
  - OA2EN = 1 and OA2MSK = 4 and OA1[7:5] = OA2[7:5]
  - OA2EN = 1 and OA2MSK = 5 and OA1[7:6] = OA2[7:6]
  - OA2EN = 1 and OA2MSK = 6 and OA1[7] = OA2[7]
  - OA2EN = 1 and OA2MSK = 7
  - GCEN = 1 and OA1[7:1] = 0000000
  - ALERTEN = 1 and OA1[7:1] = 0001100
  - SMBDEN = 1 and OA1[7:1] = 1100001
  - SMBHEN = 1 and OA1[7:1] = 0001000
- The master starts a transfer addressed to the 10-bit slave address OA1.

As a result, after the address reception, the ADDCODE value is OA1[7:1] equal to the 7-bit slave address, instead of 11110 and OA1[9:8].

Workaround

None. If several slave addresses are enabled, mixing 10-bit and 7-bit addresses, the 10-bit slave address OA1 [7:1] must not be equal to the 7-bit slave address.

2.4.3 Wakeup frames may not wake up the MCU mode when Stop mode entry follows I2C enabling

Description

If the I2C is enabled (PE = 1) and wakeup from Stop is enabled in I2C (WUPEN = 1) while a transfer occurs on the I^2C bus and Stop mode is entered during the same transfer while SCL=0, the I2C is not able to detect the following START condition. This means that, if the I2C is addressed, it does not wake up the MCU and this address is not acknowledged.

Workaround

After enabling the I2C (PE is set to 1), wait for a temporization before entering Stop mode, to ensure that the eventual on-going frame is finished.

2.4.4 Wrong behaviors related with MCU Stop mode when wakeup from Stop mode by I2C peripheral disabled

Description

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

1. BUSY flag can 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 I^2C-bus environment.
2. If I^2C-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 I^2C-bus activity during that time.
   This may occur when the MCU enters Stop mode during the address phase of an I^2C-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 I^2C-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.

2.4.5 Wakeup frame may not wake up from Stop if $t_{HD(STA)}$ is close to $t_{SU(HSI)}$ in Fast-mode and Fast-mode Plus.

Description

Under specific conditions and if the START condition hold time $t_{HD(STA)}$ duration is very close to the HSI start-up time duration $t_{su(HSI)}$, the I^2C is not able to detect the address match and to wake up the MCU from STOP. The $t_{su(HSI)}$ is between 1 µs and 2 µs (refer to product datasheet), therefore this issue cannot occur in Standard mode. To see the limitation, one of the conditions listed below has to be met:

- Timeout detection is enabled (TIMOUTEN = 1 or TEXTEN = 1) and the frame before the wakeup frame is abnormally finished due to a I^2C timeout detection (TIMOUT = 1).
- The slave arbitration is lost during the frame before the wakeup frame (ARLO = 1). According to standards, the slave arbitration is not applicable in I^2C and used only in SMBus, for which the transfer is done in Standard mode. Therefore when the standards are respected, this condition does not lead to the limitation.
- The MCU enters Stop mode while another slave is addressed, after the address phase and before the STOP condition (BUSY = 1).
- The MCU is in Stop mode and another slave is addressed before the I^2C is addressed.

Note: The last three conditions can occur only in a multi-slave network. In Stopmode, the HSI is powered on by the I²C when a START condition is detected (SDA falling edge while SCL is high). The HSI is used to receive the address and it is powered off after the address reception is case it is not the I²C slave address. If one of the conditions above is met and if the SCL falling edge following the START condition occurs on the first cycle of the I2CCLK clock (HSI), the address reception is not correctly done and the address match wakeup interrupt is not generated.

Workaround

None at MCU level. To ensure the correct behavior in a multi-slave network, the master should use a START condition hold time lower than 1 µs or greater than 2 µs.

If the wakeup frame is not acknowledged by the I²C:

- If the master can program the duration of the START hold time: the master should decrease or increase the START condition hold time for more than one HSI period and resend the wakeup frame.
- If the master can change the I²C transfer mode: the master should switch to Standard mode and resend the wakeup frame.

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

Description

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

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

The I²C SDA line is not correctly sampled when $t_{SU;DAT}$ is smaller than one I2CCLK (I²C clock) period: the previous SDA value is sampled instead of the current one. This can result in a wrong slave address reception, a wrong received data byte, or a wrong received acknowledge bit.

Workaround

Increase the I2CCLK frequency to get I2CCLK period smaller than the transmitter minimum data set-up time. Or, if it is possible, increase the transmitter minimum data set-up time.

2.4.7 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 generates 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.

2.5 USART peripheral limitation

2.5.1 When PCLK is selected as clock source for USART1, PCLK1 is used instead of PCLK2

Description

USART1 is mapped on the fast APB (APB2) and its clock can be selected among four different sources using the USART1SW [1:0] bits in the RCC_CFGR3 register.

The default configuration selects PCLK1 (APB1 clock) as USART1 clock source instead of PCLK2 (APB2 clock).

Workaround

There is no workaround. To reach 9 Mbaud, the system clock (SYSCLK) should be selected as USART1 clock source.

2.5.2 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 must 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 does not 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.

2.5.3 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 is not set in the following conditions:

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

Workaround

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

2.5.4 nRTS is active while RE or UE = 0

Description

The nRTS line is driven low as soon as RTSE bit is set even if the USART is disabled (UE = 0) or the receiver is disabled (RE = 0, not ready to receive data).

Workaround

Configure the I/O used for nRTS as alternate function after setting the UE and RE bits.

2.5.5 Receiver timeout counter starting in case of two stop bit configuration

Description

In the case of two stop bit configuration, the receiver timeout counter starts counting from the end of the second stop bit of the last character instead of the end of the first stop bit.

Workaround

Change the RTO value in the USARTx_RTOR register with subtracting 1 bit duration.

2.5.6 Data corruption due to noisy receive line

Description

In UART mode with oversampling by 8 or 16 and with 1 or 2 stop bits, the received data may be corrupted if a glitch to zero shorter than the half-bit occurs on the receive line within the second half of the stop bit.

Workaround

None.

2.6 GPIO peripheral limitation

2.6.1 GPIOx locking mechanism is not working properly for GPIOx_OTYPE register

Description

Locking of GPIOx_OTYPER[i] with i = 15 ..8 depends on the setting of GPIOx_LCKR[i-8] and not from the setting of GPIOx_LCKR[i]. GPIOx_LCKR[i-8] locks GPIOx_OTYPER[i] together with GPIOx_OTYPER[i-8]. It is not possible to lock GPIOx_OTYPER[i] with i = 15...8, without locking also GPIOx_OTYPER[i-8].

Workaround

The only way to lock GPIOx_OTYPER[i] with i=15..8 is to lock also GPIOx_OTYPER[i-8].

3 Revision history

Table 3. Document revision history

Date	Revision	Changes
05-Jun-2014	1	Initial release
06-Oct-2014	2	Updated *Table 4: Summary of silicon limitations*. Added *Section 2.4.6: Wrong data sampling when data set-up time (tSU;DAT) is smaller than one I2CCLK period*. Added *Section 2.1: System limitations*. Added *Section 2.2.4: Load multiple not supported by ADC interface*. Added *Section 2.3.2: SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI is under DMA transaction near the end of transfer or end of transfer '-1'*.
20-Nov-2014	3	Updated *Section 2.3: SPI peripheral limitations*.
09-Apr-2015	4	Updated *Table 4: Summary of silicon limitations* Added *Section 2.1.2: Full JTAG configuration without NJTRST pin cannot be used* Added limitations in USART: *Section 2.5.2: Start bit detected too soon when sampling for NACK signal from the smartcard*, *Section 2.5.3: Break request can prevent the transmission complete flag (TC) from being set* and *Section 2.5.3: Break request can prevent the transmission complete flag (TC) from being set* and *Section 2.5.4: nRTS is active while RE or UE = 0*
23-Sep-2015	5	Updated – *Table 4: Summary of silicon limitations* – *Section 2.4.5: Wakeup frame may not wake up from Stop if $t_{HD(STA)}$ is close to $t_{su(HSI)}$ in Fast-mode and Fast-mode Plus.* Added – *Section 2.1.3: CCM RAM write protection register SYSCFG_RCR not reset by system reset.* – *Section 2.2.6: Possible voltage drop caused by a transitory phase when the ADC is switching from a regular channel to an injected channel Rank 1* – *Section 2.3.3: BSY bit may stay high at the end of a SPI data transfer in slave mode* – *Section 2.5.5: Receiver timeout counter starting in case of two stop bit configuration* – *Section 2.4.7: Spurious bus error detection in master mode*
12-May-2016	6	Updated: – *Table 4: Summary of silicon limitations* Added: – *Section 2.2.7: Overrun flag may not be set if converted data are not read before writing new data* – *Section 2.2.8: ADC differential mode Common mode input range.*
28-Sep-2020	7	Added: – *Section 2.2.8: Imprecise VREFINT calibration values* – *Section 2.5.6: Data corruption due to noisy receive line*