STM32G071xx device errata

Applicability

This document applies to the part numbers of STM32G071xx devices and the device variants as stated in this page.

It gives a summary and a description of the device errata, with respect to the device datasheet and reference manual RM0444.

Deviation of the real device behavior from the intended device behavior is considered to be a device limitation. Deviation of the description in the reference manual or the datasheet from the intended device behavior is considered to be a documentation erratum. The term *"errata"* applies both to limitations and documentation errata.

Table 1. Device summary

Reference	Part numbers
STM32G071x8	STM32G071C8, STM32G071G8, STM32G071K8, STM32G071R8
STM32G071xB	STM32G071CB, STM32G071EB, STM32G071GB, STM32G071KB, STM32G071RB

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32G071xx	B	0x2000

1. *Refer to the device datasheet for how to identify this code on different types of package.*
2. *REV_ID[15:0] bitfield of DBGMCU_IDCODE register.*

1 Summary of device errata

The following table gives a quick reference to the STM32G071xx device limitations and their status:

A = workaround available

N = no workaround available

P = partial workaround available

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

Table 3. Summary of device limitations

Function	Section	Limitation	Status
			Rev. B
System	2.2.1	Unstable LSI when it clocks RTC or CSS on LSE	P
	2.2.2	WUFx wakeup flag wrongly set during configuration	A
	2.2.3	Under Level 1 read protection, booting from Main Flash memory selected through PA14-BOOT0 pin is not functional	N
	2.2.4	DMAMUX cannot be synchronized or triggered by EXTI	N
	2.2.5	Overwriting with all zeros a Flash memory location previously programmed with all ones fails	N
GPIO	2.3.1	GPIO assigned to DAC cannot be used in output mode when the DAC output is connected to on-chip peripheral	N
DMA	2.4.1	DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear	A
DMAMUX	2.5.1	SOFx not asserted when writing into DMAMUX_CFR register	N
	2.5.2	OFx not asserted for trigger event coinciding with last DMAMUX request	N
	2.5.3	OFx not asserted when writing into DMAMUX_RGCFR register	N
	2.5.4	Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event	A
ADC	2.6.1	Overrun flag is not set if EOC reset coincides with new conversion end	P
	2.6.2	Writing ADC_CFGR1 register while ADEN bit is set resets RES[1:0] bitfield	A
	2.6.3	Out-of-threshold value is not detected in AWD1 Single mode	A
TIM	2.7.1	One-pulse mode trigger not detected in master-slave reset + trigger configuration	P
	2.7.2	TIM1and TIM15 synchronization trigger might be missed	N
	2.7.3	TIM16 and TIM17 are unduly clocked by SYSCLK	N
LPTIM	2.8.1	MCU may remain stuck in LPTIM interrupt when entering Stop mode	A
	2.8.2	MCU may remain stuck in LPTIM interrupt when clearing event flag	P
RTC and TAMP	2.9.1	Calendar initialization may fail in case of consecutive INIT mode entry	A
I2C	2.10.1	Wrong data sampling when data setup time (tSU;DAT) is shorter than one I2C kernel clock period	P
	2.10.2	Spurious bus error detection in master mode	A
	2.10.3	Spurious master transfer upon own slave address match	P
SPI	2.12.1	BSY bit may stay high when SPI is disabled	A

Function	Section	Limitation	Status Rev. B
SPI	2.12.2	BSY bit may stay high at the end of data transfer in slave mode	A
USART	2.11.1	Data corruption due to noisy receive line	N
UCPD	2.13.1	UCPD wrong Rpdb default trimming value after power-on	N

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

Table 4. Summary of device documentation errata

Function	Section	Documentation erratum
USART	2.11.2	USART prescaler feature missing in USART implementation section

2 Description of device errata

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

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

2.1 Core

Reference manual and errata notice for the Arm® Cortex®-M0+ core revision r0p1 is available from http://infocenter.arm.com.

2.2 System

2.2.1 Unstable LSI when it clocks RTC or CSS on LSE

Description

The LSI clock can become unstable (duty cycle different from 50 %) and its maximum frequency can become significantly higher than 32 kHz, when:

- LSI clocks the RTC, or it clocks the clock security system (CSS) on LSE (which holds when the LSECSSON bit set), and
- the V_{DD} power domain is reset while the backup domain is not reset, which happens:
  - upon exiting Shutdown mode
  - if V_{BAT} is separate from V_{DD} and V_{DD} goes off then on
  - if V_{BAT} is tied to V_{DD} (internally in the package for products not featuring the VBAT pin, or externally) and a short (< 1 ms) V_{DD} drop under V_{DD}(min) occurs

Workaround

Apply one of the following measures:

- Clock the RTC with LSE or HSE/32, without using the CSS on LSE
- If LSI clocks the RTC or when the LSECSSON bit is set, reset the backup domain upon each V_{DD} power up (when the BORRSTF flag is set). If V_{BAT} is separate from V_{DD}, also restore the RTC configuration, backup registers and anti-tampering configuration.

2.2.2 WUFx wakeup flag wrongly set during configuration

Description

Upon configuring a wakeup pin (WKUPx), the corresponding wakeup flag (WUFx) might spuriously go high depending on the state and configuration of the wakeup pin.

Workaround

After configuring a wakeup pin, clear its corresponding WUFx flag.

2.2.3 Under Level 1 read protection, booting from Main Flash memory selected through PA14-BOOT0 pin is not functional

Description

With the Flash memory read protection set to Level 1 and the boot mode selected through the PA14-BOOT0 pin (BOOT0 function of the pin), an attempt to boot from Main Flash memory can wrongly be interpreted by the read protection mechanism as an unauthorized access, preventing the user code execution. Booting from Main Flash memory operates correctly if selected through option bits (nBOOT_SEL and nBOOT0 both set).

Workaround

None.

2.2.4 DMAMUX cannot be synchronized or triggered by EXTI

Description

The EXTI-related DMAMUX synchronization and trigger inputs are wrongly routed to the *it_exti_per(y)* output instead of being routed to the *exti[15:0]* output lines.

The *it_exti_per(y)* signals are not usable for synchronizing and triggering DMAMUX.

Workaround

None.

2.2.5 Overwriting with all zeros a Flash memory location previously programmed with all ones fails

Description

Any attempt to re-program with all zeros (0x0000 0000 0000 0000) a Flash memory location previously programmed with 0xFFFF FFFF FFFF FFFF fails and the PROGERR flag of the FLASH_SR register is set.

Workaround

None.

2.3 GPIO

2.3.1 GPIO assigned to DAC cannot be used in output mode when the DAC output is connected to on-chip peripheral

Description

When a DAC output is connected only to an on-chip peripheral, the corresponding GPIO is expected to be available as an output for any other functions.

However, when the DAC output is configured for on-chip peripheral connection only, the GPIO output buffer remains disabled and cannot be used in output mode (GPIO or alternate function). It can still be used in input or analog mode.

Workaround

None.

2.4 DMA

2.4.1 DMA disable failure and error flag omission upon simultaneous transfer error and global flag clear

Description

Upon a data transfer error in a DMA channel x, both the specific TEIFx and the global GIFx flags are raised and the channel x is normally automatically disabled. However, if in the same clock cycle the software clears the GIFx flag (by setting the CGIFx bit of the DMA_IFCR register), the automatic channel disable fails and the TEIFx flag is not raised.

This issue does not occur with ST's HAL software that does not use and clear the GIFx flag, but uses and clears the HTIFx, TCIFx, and TEIFx specific event flags instead.

Workaround

Do not clear GIFx flags. Instead, use HTIFx, TCIFx, and TEIFx specific event flags and their corresponding clear bits.

2.5 DMAMUX

2.5.1 SOFx not asserted when writing into DMAMUX_CFR register

Description

The SOFx flag of the DMAMUX_CSR status register is not asserted if overrun from another DMAMUX channel occurs when the software writes into the DMAMUX_CFR register.

This can happen when multiple DMA channels operate in synchronization mode, and when overrun can occur from more than one channel. As the SOFx flag clear requires a write into the DMAMUX_CFR register (to set the corresponding CSOFx bit), overrun occurring from another DMAMUX channel operating during that write operation fails to raise its corresponding SOFx flag.

Workaround

None. Avoid the use of synchronization mode for concurrent DMAMUX channels, if at least two of them potentially generate synchronization overrun.

2.5.2 OFx not asserted for trigger event coinciding with last DMAMUX request

Description

In the DMAMUX request generator, a trigger event detected in a critical instant of the last-generated DMAMUX request being served by the DMA controller does not assert the corresponding trigger overrun flag OFx. The critical instant is the clock cycle at the very end of the trigger overrun condition.

Additionally, upon the following trigger event, one single DMA request is issued by the DMAMUX request generator, regardless of the programmed number of DMA requests to generate.

The failure only occurs if the number of requests to generate is set to more than two (GNBREQ[4:0] > 00001).

Workaround

Make the trigger period longer than the duration required for serving the programmed number of DMA requests, so as to avoid the trigger overrun condition from occurring on the very last DMA data transfer.

2.5.3 OFx not asserted when writing into DMAMUX_RGCFR register

Description

The OFx flag of the DMAMUX_RGSR status register is not asserted if an overrun from another DMAMUX request generator channel occurs when the software writes into the DMAMUX_RGCFR register. This can happen when multiple DMA channels operate with the DMAMUX request generator, and when an overrun can occur from more than one request generator channel. As the OFx flag clear requires a write into the DMAMUX_RGCFR register (to set the corresponding COFx bit), an overrun occurring in another DMAMUX channel operating with another request generator channel during that write operation fails to raise the corresponding OFx flag.

Workaround

None. Avoid the use of request generator mode for concurrent DMAMUX channels, if at least two channels are potentially generating a request generator overrun.

2.5.4 Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event

Description

If a write access into the DMAMUX_CxCR register having the SE bit at zero and SPOL[1:0] bitfield at a value other than 00:

- sets the SE bit (enables synchronization),
- modifies the values of the DMAREQ_ID[5:0] and SYNC_ID[4:0] bitfields, and
- does not modify the SPOL[1:0] bitfield,

and if a synchronization event occurs on the previously selected synchronization input exactly two AHB clock cycles before this DMAMUX_CxCR write, then the input DMA request selected by the DMAREQ_ID[5:0] value before that write is routed.

Workaround

Ensure that the SPOL[1:0] bitfield is at 00 whenever the SE bit is 0. When enabling synchronization by setting the SE bit, always set the SPOL[1:0] bitfield to a value other than 00 with the same write operation into the DMAMUX_CxCR register.

2.6 ADC

2.6.1 Overrun flag is not set if EOC reset coincides with new conversion end

Description

If the EOC flag is cleared by an ADC_DR register read operation or by software during the same APB cycle in which the data from a new conversion are written in the ADC_DR register, the overrun event duly occurs (which results in the loss of either current or new data) but the overrun flag (OVR) may stay low.

Workaround

Clear the EOC flag, by performing an ADC_DR read operation or by software within less than one ADC conversion cycle period from the last conversion cycle end, in order to avoid the coincidence with the end of the new conversion cycle.

2.6.2 Writing ADC_CFGR1 register while ADEN bit is set resets RES[1:0] bitfield

Description

Modifying the ADC_CFGR1 register while ADC is enabled (ADEN set in ADC_CR) and no conversion is ongoing (ADSTART cleared in ADC_CR) resets RES[1:0] to 00 whatever the bitfield previous value.

Workaround

Apply the following sequence:

1. Set ADDIS to disable the ADC, and wait until ADEN is cleared.
2. Program the ADC_CFGR1 register according to the application requirements.
3. Set ADEN bit.

2.6.3 Out-of-threshold value is not detected in AWD1 Single mode

Description

AWD1 analog watchdog does not detect that the result of a converted channel has reached the programmed threshold when the ADC operates in Single mode, performs a sequence of conversions, and one of the converted channels other than the first one is monitored by the AWD1 analog watchdog.

Workaround

Apply one of the following measures:

- Use a conversion sequence of one single channel.
- Configure the monitored channel as the first one of the sequence.

2.7 TIM

2.7.1 One-pulse mode trigger not detected in master-slave reset + trigger configuration

Description

The failure occurs when several timers configured in one-pulse mode are cascaded, and the master timer is configured in combined reset + trigger mode with the MSM bit set:

OPM = 1 in TIMx_CR1, SMS[3:0] = 1000 and MSM = 1 in TIMx_SMCR.

The MSM delays the reaction of the master timer to the trigger event, so as to have the slave timers cycle-accurately synchronized.

If the trigger arrives when the counter value is equal to the period value set in the TIMx_ARR register, the one-pulse mode of the master timer does not work and no pulse is generated on the output.

Workaround

None. However, unless a cycle-level synchronization is mandatory, it is advised to keep the MSM bit reset, in which case the problem is not present. The MSM = 0 configuration also allows decreasing the timer latency to external trigger events.

2.7.2 TIM1and TIM15 synchronization trigger might be missed

Description

A slave timer may miss synchronization trigger signal generated by its respective master timer TIM1 or TIM15 if the master timer clock is faster than the slave timer clock.

Workaround

None.

2.7.3 TIM16 and TIM17 are unduly clocked by SYSCLK

Description

The timers TIM16 and TIM17 are unduly clocked by SYSCLK instead of being clocked by the timer clock TIMPCLK. As a consequence, they do not reflect AHB and APB prescaler settings.

The TIM16 and TIM17 are fully functional as long as the SYSCLK-to-PCLK frequency ratio remains smaller than four.

Workaround

None.

2.8 LPTIM

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

Description

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

When the user application clears the ENABLE bit in the LPTIM_CR register 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 LPTIM_CR register. Instead, reset the whole LPTIMx peripheral via the RCC controller by setting and resetting its respective LPTIMxRST bit in RCC_APByRSTRz register.

2.8.2 MCU may remain stuck in LPTIM interrupt when clearing event flag

Description

This limitation occurs when the LPTIM is configured in interrupt mode (at least one interrupt is enabled) and the software clears any flag by writing the LPTIM_ICR bit in the LPTIM_ISR register. If the interrupt status flag corresponding to a disabled interrupt is cleared simultaneously with a new event detection, the set and clear commands might reach the APB domain at the same time, leading to an asynchronous interrupt signal permanently stuck high.

This issue can occur either during an interrupt subroutine execution (where the flag clearing is usually done), or outside an interrupt subroutine.

Consequently, the firmware remains stuck in the LPTIM interrupt routine, and the MCU cannot enter Stop mode.

Workaround

To avoid this issue, it is strongly advised to follow the recommendations listed below:

- Clear the flag only when its corresponding interrupt is enabled in the interrupt enable register.
- If for specific reasons, it is required to clear some flags that have corresponding interrupt lines disabled in the interrupt enable register, it is recommended to clear them during the current subroutine prior to those which have corresponding interrupt line enabled in the interrupt enable register.
- Flags must not be cleared outside the interrupt subroutine.

Note: The proper clear sequence is already implemented in the HAL_LPTIM_IRQHandler in the STM32Cube.

2.9 RTC and TAMP

2.9.1 Calendar initialization may fail in case of consecutive INIT mode entry

Description

If the INIT bit of the RTC_ICSR register is set between one and two RTCCLK cycles after being cleared, the INITF flag is set immediately instead of waiting for synchronization delay (which should be between one and two RTCCLK cycles), and the initialization of registers may fail. Depending on the INIT bit clearing and setting instants versus the RTCCLK edges, it can happen that, after being immediately set, the INITF flag is cleared during one RTCCLK period then set again. As writes to calendar registers are ignored when INITF is low, a write occurring during this critical period might result in the corruption of one or more calendar registers.

Workaround

After existing the initialization mode, clear the BYPSHAD bit (if set) then wait for RSF to rise, before entering the initialization mode again.

Note: It is recommended to write all registers in a single initialization session to avoid accumulating synchronization delays.

2.10 I2C

2.10.1 Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period

Description

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

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

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

Workaround

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

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

2.10.2 Spurious bus error detection in master mode

Description

In master mode, a bus error can be detected spuriously, with the consequence of setting the BERR flag of the I2C_SR register and generating bus error interrupt if such interrupt is enabled. Detection of bus error has no effect on the I^2C-bus transfer in master mode and any such transfer continues 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 ongoing transfer can be handled normally.

2.10.3 Spurious master transfer upon own slave address match

Description

When the device is configured to operate at the same time as master and slave (in a multi- master I^2C-bus application), a spurious master transfer may occur under the following condition:

- Another master on the bus is in process of sending the slave address of the device (the bus is busy).
- The device initiates a master transfer by bit set before the slave address match event (the ADDR flag set in the I2C_ISR register) occurs.
- After the ADDR flag is set:
  - the device does not write I2C_CR2 before clearing the ADDR flag, or
  - the device writes I2C_CR2 earlier than three I2C kernel clock cycles before clearing the ADDR flag

In these circumstances, even though the START bit is automatically cleared by the circuitry handling the ADDR flag, the device spuriously proceeds to the master transfer as soon as the bus becomes free. The transfer configuration depends on the content of the I2C_CR2 register when the master transfer starts. Moreover, if the I2C_CR2 is written less than three kernel clocks before the ADDR flag is cleared, the I2C peripheral may fall into an unpredictable state.

Workaround

Upon the address match event (ADDR flag set), apply the following sequence.

Normal mode (SBC = 0):

1. Set the ADDRCF bit.
2. Before Stop condition occurs on the bus, write I2C_CR2 with the START bit low.

Slave byte control mode (SBC = 1):

1. Write I2C_CR2 with the slave transfer configuration and the START bit low.
2. Wait for longer than three I2C kernel clock cycles.
3. Set the ADDRCF bit.
4. Before Stop condition occurs on the bus, write I2C_CR2 again with its current value.

The time for the software application to write the I2C_CR2 register before the Stop condition is limited, as the clock stretching (if enabled), is aborted when clearing the ADDR flag.

Polling the BUSY flag before requesting the master transfer is not a reliable workaround as the bus may become busy between the BUSY flag check and the write into the I2C_CR2 register with the START bit set.

2.11 USART

2.11.1 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.

2.11.2 USART prescaler feature missing in USART implementation section

Description

Some reference manual revisions may omit the information that the USART prescaler is not present in all USART instances. This information is provided in the USART implementation section of the corresponding reference manual.

This is a documentation issue rather than a product limitation.

Workaround

No application workaround is required or applicable.

2.12 SPI

2.12.1 BSY bit may stay high when SPI is disabled

Description

The BSY flag may remain high upon disabling the SPI while operating in:

- master transmit mode and the TXE flag is low (data register full).
- master receive-only mode (simplex receive or half-duplex bidirectional receive phase) and an SCK strobing edge has not occurred since the transition of the RXNE flag from low to high.

- slave mode and NSS signal is removed during the communication.

Workaround

When the SPI operates in:

- master transmit mode, disable the SPI when TXE = 1 and BSY = 0.
- master receive-only mode, ignore the BSY flag.
- slave mode, do not remove the NSS signal during the communication.

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

Description

BSY flag may sporadically remain high at the end of a data transfer in slave mode. This occurs upon coincidence of internal CPU clock and external SCK clock provided by master.

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

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

Workaround

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

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

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

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

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

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

2.13 UCPD

2.13.1 UCPD wrong R_{pdb} default trimming value after power-on

Description

When the device is used in application supporting dead battery mode of USB Type-C™, the R_{pdb} pull-down resistor is expected to be 5.1 kΩ ±20% for the device powered down or in the process of booting and 5.1 kΩ ±10% when it is powered and running, as per the USB Type-C™ specification.

However, upon powering the device up, the R_{pdb} values spuriously change from 5.1 kΩ ±20% to 3.9 kΩ ±20% in the instant when V_{DD} crosses the V_{POR} threshold and keep that value until the software validates the configuration through the UCPDx_STROBE bits of the SYSCFG_CFGR1 register.

This behavior remains compliant with the USB Type-C™ specification revision 1.3 as long as the duration of this state is shorter than 100 ms ($t_{CCDebounce}$ parameter).

Workaround

In applications supporting the dead battery mode of USB Type-C™, validate the configuration of the pull-down resistors, through UCPDx_STROBE bits, within the $t_{CCDebounce}$ time after power-on, to stay compliant with the USB Type-C™ specification.

Revision history

Table 5. Document revision history

Date	Version	Changes
14-Nov-2018	1	Initial release.
16-Jan-2020	2	Added: • Section 2.2.4 DMAMUX cannot be synchronized or triggered by EXTI • Section 2.2.5 Overwriting with all zeros a Flash memory location previously programmed with all ones fails • Section 2.3.1 GPIO assigned to DAC cannot be used in output mode when the DAC output is connected to on-chip peripheral • Section 2.6.1 Overrun flag is not set if EOC reset coincides with new conversion end • Section 2.6.2 Writing ADC_CFGR1 register while ADEN bit is set resets RES[1:0] bitfield • Section 2.6.3 Out-of-threshold value is not detected in AWD1 Single mode • Section 2.7.2 TIM1and TIM15 synchronization trigger might be missed • Section 2.7.3 TIM16 and TIM17 are unduly clocked by SYSCLK • Section 2.11.1 Data corruption due to noisy receive line • Section 2.11.2 USART prescaler feature missing in USART implementation section • Table 4. Summary of device documentation errata Updated Table 3. Summary of device limitations.

Contents