STM32F427/437xx and STM32F429/439xx

Errata sheet

STM32F427/437 and STM32F429/439 device errata

Applicability

This document applies to the part numbers of STM32F427/437 and STM32F429/439 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 RM0090.

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
STM32F427xx	STM32F427VG, STM32F427ZG, STM32F427IG, STM32F427AG, STM32F427VI, STM32F427ZI, STM32F427II, STM32F427AI
STM32F437xx	STM32F437VG, STM32F437ZG, STM32F437IG, STM32F437VI, STM32F437ZI, STM32F437II, STM32F437AI
STM32F429xx	STM32F429VG, STM32F429ZG, STM32F429IG, STM32F429VI, STM32F429ZI, STM32F429II, STM32F429AI, STM32F429AG, STM32F429BG, STM32F429BI, STM32F429NI, STM32F429NG, STM32F429VE, STM32F429ZE, STM32F429IE, STM32F429BE, STM32F429NE
STM32F439xx	STM32F439VI, STM32F439VG, STM32F439ZG, STM32F439ZI, STM32F439IG, STM32F439II, STM32F439AI, STM32F439BG, STM32F439BI, STM32F439NI, STM32F439NG

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32F425xx, STM32F437xx, STM32F429xx, STM32F439xx	A	0x1000
	Y	0x1003
	1	0x1007
	3	0x2001
	4, 5, B	0x2003

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 STM32F427/437 and STM32F429/439 device limitations and their status:

A = limitation present, workaround available

N = limitation present, no workaround available

P = limitation present, partial workaround available

"-" = limitation absent

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 . A	Rev . Y	Rev . 1	Rev . 3	Rev . 4	Rev . 5	Rev . B
Core	2.1.1	Interrupted loads to SP can cause erroneous behavior	A	A	A	A	A	A	A
	2.1.2	VDIV or VSQRT instructions might not complete correctly when very short ISRs are used	A	A	A	A	A	A	A
	2.1.3	Store immediate overlapping exception return operation might vector to incorrect interrupt	A	A	A	A	A	A	A
System	2.2.1	Debugging Stop mode and SysTick timer	A	A	A	A	A	A	A
	2.2.2	Debugging Stop mode with WFE entry	A	A	A	A	A	A	A
	2.2.3	Debugging Sleep/Stop mode with WFE/WFI entry	A	A	A	A	A	A	A
	2.2.4	Wake-up sequence from Standby mode when using more than one wake-up source	A	A	A	A	A	A	A
	2.2.5	Full JTAG configuration without NJTRST pin cannot be used	A	A	A	A	A	A	A
	2.2.6	MPU attribute to RTC and IWDG registers incorrectly managed	A	A	A	A	A	A	A
	2.2.7	Delay after an RCC peripheral clock enabling	A	A	A	A	A	A	A
	2.2.8	Internal noise impacting the ADC accuracy	A	A	A	A	A	A	A
	2.2.9	Possible delay in backup domain protection disabling/ enabling after programming the DBP bit	A	A	A	A	A	A	A
	2.2.10	PC13 signal transitions disturb LSE	N	N	N	N	N	N	N
	2.2.11	Corrupted content of the backup domain due to a missed power-on reset after this domain supply voltage drop	A	A	A	A	A	A	A
	2.2.12	Over-drive and Under-drive modes unavailability	N	N	-	-	-	-	-
	2.2.13	Operating voltage extension down to 1.7 V in the whole temperature range	N	N	N	-	-	-	-
	2.2.14	Data read from flash memory corrupted when PA12 used as GPIO or alternate function	A	A	A	-	-	-	-
	2.2.15	Data cache might be corrupted during flash memory read-while-write operation	A	A	A	A	A	A	A
FMC	2.3.1	Dummy read cycles inserted when reading synchronous memories	N	N	N	N	N	N	N
	2.3.2	FMC synchronous mode and NWAIT signal disabled	A	A	A	-	-	-	-

Function	Section	Limitation	Status						
			Rev . A	Rev . Y	Rev . 1	Rev . 3	Rev . 4	Rev . 5	Rev . B
FMC	2.3.3	Read access to a non-initialized SDRAM bank	P	P	P	-	-	-	-
	2.3.4	Interruption of CPU burst read access to the end of an SDRAM row	A	A	A	-	-	-	-
	2.3.5	Corruption of data read by CPU from the FMC used as stack or heap	A	-	-	-	-	-	-
	2.3.6	FMC NOR/PSRAM controller: asynchronous read access on bank 2 to 4 returns wrong data when bank 1 is in synchronous mode (BURSTEN bit is set)	A	A	-	-	-	-	-
	2.3.7	FMC dynamic and static bank switching	A	A	A	-	-	-	-
	2.3.8	FMC NOR/PSRAM controller write protocol violation	A	A	A	-	-	-	-
	2.3.9	Data corruption during burst read from FMC synchronous memory	A	A	A	A	A	A	A
	2.3.11	Missed burst write transaction on multiplexed PSRAM	-	-	A	A	A	A	A
	2.3.12	FMC NOR/PSRAM controller bank switch with different BUSTURN durations	A	A	A	A	A	A	A
	2.3.13	Wrong data read from a busy NAND memory	A	A	A	A	A	A	A
	2.3.14	Spurious clock stoppage with continuous clock feature enabled	A	A	A	A	A	A	A
	2.3.15	SDRAM bank address corruption upon an interruption of CPU burst read access	A	A	A	A	A	A	A
SDIO	2.4.1	Wrong data written during SDIO hardware flow control	N	N	N	N	N	N	N
	2.4.2	Wrong CCRCFAIL status after a response without CRC is received	A	A	A	A	A	A	A
	2.4.3	Data corruption in SDIO clock dephasing (NEGEDGE) mode	N	N	N	N	N	N	N
	2.4.4	CE-ATA multiple write command and card busy signal management	A	A	A	A	A	A	A
	2.4.5	No underrun detection with wrong data transmission	A	A	A	A	A	A	A
ADC	2.5.1	ADC sequencer modification during conversion	A	A	A	A	A	A	A
DAC	2.6.1	DMA request not automatically cleared by clearing DMAEN	A	A	A	A	A	A	A
	2.6.2	DMA underrun flag not set when an internal trigger is detected on the clock cycle of the DMA request acknowledge	N	N	N	N	N	N	N
TIM	2.7.1	PWM re-enabled in automatic output enable mode despite of system break	P	P	P	P	P	P	P
	2.7.3	Consecutive compare event missed in specific conditions	N	N	N	N	N	N	N
	2.7.4	Output compare clear not working with external counter reset	P	P	P	P	P	P	P
IWDG	2.8.1	RVU flag not reset in Stop	A	A	A	A	A	A	A
	2.8.2	PVU flag not reset in Stop	A	A	A	A	A	A	A
	2.8.3	RVU flag not cleared at low APB clock frequency	A	A	A	A	A	A	A
	2.8.4	PVU flag not cleared at low APB clock frequency	A	A	A	A	A	A	A
RTC	2.9.1	Spurious tamper detection when disabling the tamper channel	N	N	N	N	N	N	N
	2.9.2	RTC calendar registers are not locked properly	A	A	A	A	A	A	A

Function	Section	Limitation	Status						
			Rev . A	Rev . Y	Rev . 1	Rev . 3	Rev . 4	Rev . 5	Rev . B
RTC	2.9.3	RTC interrupt can be masked by another RTC interrupt	A	A	A	A	A	A	A
	2.9.4	Calendar initialization may fail in case of consecutive INIT mode entry	A	A	A	A	A	A	A
	2.9.5	Alarm flag may be repeatedly set when the core is stopped in debug	N	N	N	N	N	N	N
	2.9.6	Detection of a tamper event occurring before enabling the tamper detection is not supported in edge detection mode	A	A	A	A	A	A	A
I2C	2.10.1	Spurious bus error detection in controller mode	A	A	A	A	A	A	A
	2.10.2	SMBus standard not fully supported	A	A	A	A	A	A	A
	2.10.3	Start cannot be generated after a misplaced Stop	A	A	A	A	A	A	A
	2.10.4	Mismatch on the “Setup time for a repeated Start condition” timing parameter	A	A	A	A	A	A	A
	2.10.5	Data valid time ($t_{VD;DAT}$) violated without the OVR flag being set	A	A	A	A	A	A	A
	2.10.6	Both SDA and SCL maximum rise times (t_r) violated when the VDD_I2C bus voltage is higher than ((V_{DD} + 0.3) / 0.7) V	A	A	A	A	A	A	A
USART	2.11.1	Idle frame is not detected if the receiver clock speed is deviated	N	N	N	N	N	N	N
	2.11.2	In full-duplex mode, the Parity Error (PE) flag can be cleared by writing to the data register	A	A	A	A	A	A	A
	2.11.3	Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection	N	N	N	N	N	N	N
	2.11.4	Break frame is transmitted regardless of CTS input line status	N	N	N	N	N	N	N
	2.11.5	RTS signal abnormally driven low after a protocol violation	A	A	A	A	A	A	A
	2.11.6	Start bit detected too soon when sampling for NACK signal from the smartcard	N	N	N	N	N	N	N
	2.11.7	Break request can prevent the transmission complete flag (TC) from being set	A	A	A	A	A	A	A
	2.11.8	Guard time not respected when data are sent on TXE events	A	A	A	A	A	A	A
	2.11.9	RTS is active while RE or UE = 0	A	A	A	A	A	A	A
SPI/I2S	2.12.1	BSY bit may stay high when SPI is disabled	A	A	A	A	A	A	A
	2.12.2	Anticipated communication upon SPI transit from slave receiver to master	A	A	A	A	A	A	A
	2.12.3	Wrong CRC calculation when the polynomial is even	A	A	A	A	A	A	A
	2.12.4	Corrupted last bit of data and/or CRC received in Master mode with delayed SCK feedback	A	A	A	A	A	A	A
	2.12.5	BSY flag may stay high at the end of a data transfer in Slave mode	A	A	A	A	A	A	A
	2.12.6	In I2S Slave mode, the WS level must be set by the external master when enabling the I2S	A	A	A	A	A	A	A
bxCAN	2.13.1	bxCAN time-triggered communication mode not supported	N	N	N	N	N	N	N

Function	Section	Limitation	Status						
			Rev . A	Rev . Y	Rev . 1	Rev . 3	Rev . 4	Rev . 5	Rev . B
OTG_FS	2.14.1	Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_FS registers	A	A	A	A	A	A	A
	2.14.2	Host packet transmission may hang when connecting through a hub to a low-speed device	N	N	N	N	N	N	N
	2.14.3	Data in RxFIFO is overwritten when all channels are disabled simultaneously	A	A	A	A	A	A	A
	2.14.4	OTG host blocks the receive channel when receiving IN packets and no TxFIFO is configured	A	A	A	A	A	A	A
	2.14.5	Host channel-halted interrupt not generated when the channel is disabled	A	A	A	A	A	A	A
	2.14.6	Wrong software-read OTG_FS_DCFG register values	A	A	A	A	A	A	A
OTG_HS	2.15.1	Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_HS registers	A	A	A	A	A	A	A
	2.15.2	Host packet transmission may hang when connecting the full speed interface through a hub to a low-speed device	N	N	N	N	N	N	N
ETH	2.16.1	Incorrect L3 checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads	A	A	A	A	A	A	A
	2.16.2	The Ethernet MAC processes invalid extension headers in the received IPv6 frames	N	N	N	N	N	N	N
	2.16.3	MAC stuck in the idle state on receiving the TxFIFO flush command exactly one clock cycle after a transmission completes	P	P	P	P	P	P	P
	2.16.4	Transmit frame data corruption	A	A	A	A	A	A	A
	2.16.5	Incorrect status and corrupted frames when RxFIFO overflow occurs on the penultimate word of Rx frames	A	A	A	A	A	A	A
	2.16.6	Successive write operations to the same register might not be fully taken into account	A	A	A	A	A	A	A
	2.16.7	Incorrect remote wakeup on global unicast packet	P	P	P	P	P	P	P
	2.16.8	Overflow status bits of missed frame and buffer overflow counters are cleared without a read operation	A	A	A	A	A	A	A
	2.16.9	MAC may provide incorrect Rx status for the MAC control frames when receive checksum offload is enabled	A	A	A	A	A	A	A
	2.16.10	MAC may provide an inaccurate Rx status when receive checksum offload is enabled in cut-through mode	P	P	P	P	P	P	P
	2.16.11	MAC may not drop received giant error frames	A	A	A	A	A	A	A

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

Table 4. Summary of device documentation errata

Function	Section	Documentation erratum
System	2.2.16	Setting GPIO properties of PC13 used as RTC_ALARM open-drain output
FMC	2.3.10	Missing information on prohibited 0xFF value of NAND transaction wait timing
TIM	2.7.2	TRGO and TRGO2 trigger output failure

2 Description of device errata

The following sections describe the errata 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®-M4F core revision r0p1 is available from http://infocenter.arm.com. Only applicable information from the Arm errata notice is replicated in this document.

2.1.1 Interrupted loads to SP can cause erroneous behavior

This limitation is registered under Arm ID number 752770 and classified into “Category B”. Its impact to the device is minor.

Description

If an interrupt occurs during the data-phase of a single word load to the stack-pointer (SP/R13), erroneous behavior can occur. In all cases, returning from the interrupt will result in the load instruction being executed an additional time. For all instructions performing an update to the base register, the base register will be erroneously updated on each execution, resulting in the stack-pointer being loaded from an incorrect memory location.

The affected instructions that can result in the load transaction being repeated are:

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

The affected instructions that can result in the stack-pointer being loaded from an incorrect memory address are:

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

As compilers do not generate these particular instructions, the limitation is only likely to occur with hand-written assembly code.

Workaround

Both issues may be worked around by replacing the direct load to the stack-pointer, with an intermediate load to a general-purpose register followed by a move to the stack-pointer.

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

This limitation is registered under Arm ID number 776924 and classified into “Category B”. Its impact to the device is limited.

Description

The VDIV and VSQRT instructions take 14 cycles to execute. When an interrupt is taken a VDIV or VSQRT instruction is not terminated, and completes its execution while the interrupt stacking occurs. If lazy context save of floating point state is enabled then the automatic stacking of the floating point context does not occur until a floating point instruction is executed inside the interrupt service routine.

Lazy context save is enabled by default. When it is enabled, the minimum time for the first instruction in the interrupt service routine to start executing is 12 cycles. In certain timing conditions, and if there is only one or two instructions inside the interrupt service routine, then the VDIV or VSQRT instruction might not write its result to the register bank or to the FPSCR.

The failure occurs when the following condition is met:

1. The floating point unit is enabled
2. Lazy context saving is not disabled
3. A VDIV or VSQRT is executed
4. The destination register for the VDIV or VSQRT is one of s0 - s15
5. An interrupt occurs and is taken
6. The interrupt service routine being executed does not contain a floating point instruction
7. Within 14 cycles after the VDIV or VSQRT is executed, an interrupt return is executed

A minimum of 12 of these 14 cycles are utilized for the context state stacking, which leaves 2 cycles for instructions inside the interrupt service routine, or 2 wait states applied to the entire stacking sequence (which means that it is not a constant wait state for every access).

In general, this means that if the memory system inserts wait states for stack transactions (that is, external memory is used for stack data), then this erratum cannot be observed.

The effect of this erratum is that the VDIV or VQSRT instruction does not complete correctly and the register bank and FPSCR are not updated, which means that these registers hold incorrect, out of date, data.

Workaround

A workaround is only required if the floating point unit is enabled. A workaround is not required if the stack is in external memory.

There are two possible workarounds:

- Disable lazy context save of floating point state by clearing LSPEN to 0 (bit 30 of the FPCCR at address 0xE000EF34).
- Ensure that every interrupt service routine contains more than 2 instructions in addition to the exception return instruction.

2.1.3 Store immediate overlapping exception return operation might vector to incorrect interrupt

This limitation is registered under Arm ID number 838869 and classified into “Category B (rare)”. Its impact to the device is minor.

Description

The core includes a write buffer that permits execution to continue while a store is waiting on the bus. Under specific timing conditions, during an exception return while this buffer is still in use by a store instruction, a late change in selection of the next interrupt to be taken might result in there being a mismatch between the interrupt acknowledged by the interrupt controller and the vector fetched by the processor.

The failure occurs when the following condition is met:

1. The handler for interrupt A is being executed.
2. Interrupt B, of the same or lower priority than interrupt A, is pending.
3. A store with immediate offset instruction is executed to a bufferable location.
   - STR/STRH/STRB <Rt>, [<Rn>,#imm]
   - STR/STRH/STRB <Rt>, [<Rn>,#imm]!
   - STR/STRH/STRB <Rt>, [<Rn>],#imm
4. Any number of additional data-processing instructions can be executed.
5. A BX instruction is executed that causes an exception return.
6. The store data has wait states applied to it such that the data is accepted at least two cycles after the BX is executed.
   - Minimally, this is two cycles if the store and the BX instruction have no additional instructions between them.
   - The number of wait states required to observe this erratum needs to be increased by the number of cycles between the store and the interrupt service routine exit instruction.
7. Before the bus accepts the buffered store data, another interrupt C is asserted which has the same or lower priority as A, but a greater priority than B.

Example:

The processor should execute interrupt handler C, and on completion of handler C should execute the handler for B. If the conditions above are met, then this erratum results in the processor erroneously clearing the pending state of interrupt C, and then executing the handler for B twice. The first time the handler for B is executed it will be at interrupt C's priority level. If interrupt C is pended by a level-based interrupt which is cleared by C's handler then interrupt C will be pended again once the handler for B has completed and the handler for C will be executed.

As the STM32 interrupt C is level based, it eventually becomes pending again and is subsequently handled.

Workaround

For software not using the memory protection unit, this erratum can be worked around by setting DISDEFWBUF in the Auxiliary Control Register.

In all other cases, the erratum can be avoided by ensuring a DSB occurs between the store and the BX instruction. For exception handlers written in C, this can be achieved by inserting the appropriate set of intrinsics or inline assembly just before the end of the interrupt function, for example:

ARMCC:

```
...
__schedule_barrier();
__asm{DSB};
__schedule_barrier();
}
```

GCC:

```
...
__asm volatile ("dsb 0xf":::"memory");
}
```

2.2 System

2.2.1 Debugging Stop mode and SysTick timer

Description

If the SysTick timer interrupt is enabled during the Stop mode debug (DBG_STOP bit set in the DBGMCU_CR register), it wakes up the system from Stop mode.

Workaround

To debug the Stop mode, disable the SysTick timer interrupt.

2.2.2 Debugging Stop mode with WFE entry

Description

When the Stop debug mode is enabled (DBG_STOP bit set in the DBGMCU_CR register), the software debugging is allowed during Stop mode. However, if the application software uses the WFE instruction to enter Stop mode, after wake-up, some instructions may be missed if the WFE is followed by sequential instructions. This affects only Stop debug mode with WFE entry.

Workaround

To debug Stop mode with WFE entry, the WFE instruction must be inside a dedicated function with one instruction (NOP) between the execution of the WFE and the Bx LR. For example:

```
__asm void _WFE(void)
WFE
NOP
BX lr }
```

2.2.3 Debugging Sleep/Stop mode with WFE/WFI entry

Description

When the Sleep debug or Stop debug mode is enabled (DBG_SLEEP bit or DBG_STOP bit are set in the DBGMCU_CR register), software debugging is allowed during Sleep or Stop mode. After wake-up, some unreachable instructions can be executed if the following conditions are met:

- The application software disables the Prefetch queue,
- the number of wait states configured for the flash memory interface is higher than zero, and
- the linker places the WFE or WFI instruction on a 4-byte aligned addresses (0x080xx xxx4).

Workaround

Apply one of the following measures:

- Add three NOPs after WFI/WFE instruction.
- Keep one AHB master active during Sleep (example keep DMA1 or DMA2 RCC clock enable bit set).
- Execute WFI/WFE instruction from routines inside the SRAM.

2.2.4 Wake-up sequence from Standby mode when using more than one wake-up source

Description

The various wake-up sources are logically OR-ed in front of the rising-edge detector that generates the wake-up flag (WUF). The WUF needs to be cleared before Standby mode entry, otherwise the MCU wakes up immediately. If one of the configured wake-up sources is kept high during the clearing of the WUF (by setting the CWUF bit), it may mask further wake-up events on the input of the edge detector. As a consequence, the MCU may not be able to wake up from Standby mode.

Workaround

To avoid this problem, apply the following sequence before entering Standby mode:

1. Disable all used wake-up sources.
2. Clear all related wake-up flags.
3. Reenable all used wake-up sources.
4. Enter Standby mode.

Note: Be aware that, when applying this workaround, if one of the wake-up sources is still kept high, the MCU enters Standby mode but then it wakes up immediately and generates a power reset.

2.2.5 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.

2.2.6 MPU attribute to RTC and IWDG registers incorrectly managed

Description

If the MPU is used and the nonbufferable attribute is set to the RTC or IWDG memory map region, the CPU access to the RTC or IWDG registers may be treated as bufferable, provided there is no APB prescaler configured (AHB/APB prescaler is equal to 1).

Workaround

If the nonbufferable attribute is required for these registers, perform by software a read after the write to guaranty the completion of the write access.

2.2.7 Delay after an RCC peripheral clock enabling

Description

A delay may be observed between an RCC peripheral clock enable and the effective peripheral enabling. It must be taken into account in order to manage the peripheral read/write from/to registers.

This delay depends on the peripheral mapping:

- If the peripheral is mapped on the AHB: the delay may be equal to two AHB cycles.
- If the peripheral is mapped on the APB: the delay may be equal to 1 + (AHB/APB prescaler) cycles.

Workaround

Apply one of the following measures:

- Use the DSB instruction to stall the Arm® Cortex®-M4 CPU pipeline until the instruction has completed.
- Insert “n” NOPs between the RCC enable bit write and the peripheral register writes (n = 2 for AHB peripherals, n = 1 + AHB/APB prescaler for APB peripherals).
- Simply insert a dummy read operation from the corresponding register just after enabling the peripheral clock.

2.2.8 Internal noise impacting the ADC accuracy

Description

An internal noise generated on V_{DD} supplies and propagated internally may impact the ADC accuracy. This noise is always present whatever the power mode of the MCU (Run or Sleep).

Workaround

Use the following sequence to adapt the accuracy level to the application requirements:

1. Configure the flash memory ART with prefetch OFF and data + instruction cache ON.
2. Use averaging and filtering algorithms on ADC output codes.

For more detailed workarounds, refer to the application note "*How to improve ADC accuracy when using STM32F2xx and STM32F4xx microcontrollers*" (AN4073).

2.2.9 Possible delay in backup domain protection disabling/enabling after programming the DBP bit

Description

Depending on the AHB/APB1 prescaler, a delay between DBP bit programming and the effective disabling/enabling of the backup domain protection can be observed and must be taken into account.

The higher the APB1 prescaler value, the higher the delay.

Workaround

Apply one of the following measures:

- Insert a dummy read operation to the PWR_CR register just after programming the DBP bit.
- Wait for the end of the operation (reset through the BDRST bit or write to the backup domain) via a polling loop on targeted registers.

2.2.10 PC13 signal transitions disturb LSE

Description

PC13 toggling in input or output (for example when used for RTC_AF1) may cause an incorrect LSE crystal oscillator clock frequency.

Note: The external clock input (LSE bypass) is not impacted by this limitation.

Avoid toggling PC13 when LSE is used.

Workaround

None.

2.2.11 Corrupted content of the backup domain due to a missed power-on reset after this domain supply voltage drop

Description

The backup domain reset may be missed upon a power-on following a power-off, if its supply voltage drops during the power-off phase hitting a window, which is few mV wide before it starts to rise again. In this critical window, the flip-flops are no longer able to safely retain the information and the backup domain reset has not yet been triggered. This window is located in the range between 100 mV and 700 mV, with the exact position depending mainly on the device and on the temperature.

This missed reset results in unpredictable values of the backup domain registers. This may cause a spurious behavior (such as driving the LSCO output pin on or influencing backup functions).

Workaround

Apply one of the following measures:

- In the application, let the V_{DD} and V_{BAT} supply voltages fall to a level below 100 mV for more than 200 ms before a new power-on.
- If the above workaround cannot be applied, and the boot follows a power-on reset, erase the backup domain by software.

2.2.12 Over-drive and Under-drive modes unavailability

Description

The Over-drive and Under-drive modes are not available on revision A devices.

Workaround

None.

2.2.13 Operating voltage extension down to 1.7 V in the whole temperature range

Description

Revision A and Y devices powered from a 1.7 V supply voltage, operate only in the 0 to 70 °C temperature range.

Starting from revision 1, the operating voltage down to 1.7 V is extended to the full temperature ranges:

- -40 °C to 105 °C (suffix 7)
- -40 °C to 85 °C (for suffix 6)

Workaround

None.

2.2.14 Data read from flash memory corrupted when PA12 used as GPIO or alternate function

Description

When PA12 is used as GPIO or alternate function in input or output mode, the data read from flash memory might be corrupted.

This issue occurs only when the following conditions are met:

- The device operates from a 2.7 to 3.6 V V_{DD} power supply, whatever the temperature range.
- Flash memory Bank2 is used or the dual bank feature is enabled.

Impacted part numbers:

- STM32F42xxI and STM32F43xxI
- STM32F42xxG and STM32F43xxG only when the dual bank feature is enabled

Nonimpacted part numbers:

- STM32F42xxG and STM32F43xxG when the dual bank feature is disabled
- STM32F42xxE and STM32F43xxE

Workaround

Leave PA12 unconnected on the PCB (configured as push-pull and held low).

All the other GPIOs and all alternate functions can be used except for the ones mapped on PA12. Use the OTG_HS peripheral in full-speed mode instead of the OTG_FS peripheral.

This limitation is fixed in silicon starting from revision 3.

2.2.15 Data cache might be corrupted during flash memory read-while-write operation

Description

When a write operation to the internal flash memory is done, the data cache is updated to reflect the data update. If a read operation to the other memory bank occurs during the data cache update, the data cache content may be corrupted. In this case, subsequent read operations from the same address (cache hits) is corrupted.

This issue only occurs in dual bank mode when reading (data access or code execution) from one flash memory bank while writing to the other flash 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 in FLASH_ACR register before performing any write operation to Flash memory. Before enabling the data cache again, the cache must be reset by setting and then resetting the DCRST bit in FLASH_ACR register.

Example of code:

```
/* 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();
```

2.2.16 Setting GPIO properties of PC13 used as RTC_ALARM open-drain output

Description

Some reference manual revisions may omit the information that the PC13 GPIO must be set as input when the RTC_OR register configures PC13 as an open-drain output of the RTC_ALARM signal.

Note: Enabling the internal pull-up function through the PC13 GPIO settings allows sparing an external pull-up resistor.

This is a documentation issue rather than a product limitation.

Workaround

No application workaround is required provided that the described GPIO setting is respected.

2.3 FMC

2.3.1 Dummy read cycles inserted when reading synchronous memories

Description

When performing a burst read access from a synchronous memory, two dummy read accesses are performed at the end of the burst cycle whatever the type of burst access.
The extra data values read are not used by the FMC and there is no functional failure.

Workaround

None.

2.3.2 FMC synchronous mode and NWAIT signal disabled

Description

When the FMC operates in synchronous mode with the NWAIT signal disabled, if the polarity (WAITPOL in the FMC_BCRx register) of the NWAIT signal is identical to that of the NWAIT input signal level, the system hangs and no fault is generated.

Workaround

Do not set the PD6 port to AF12 (NWAIT input). Configure the NWAIT polarity to active high by setting the WAITPOL bit of the FMC_BCRx register.

2.3.3 Read access to a non-initialized SDRAM bank

Description

When a read access is performed to an SDRAM bank while the SDRAM controller is not yet initialized, the system hangs and no fault is generated.

Workaround

Do not read SDRAM until the SDRAM controller initialization is completed.

2.3.4 Interruption of CPU burst read access to the end of an SDRAM row

Description

An interrupt occurring during a CPU AHB burst read access to the end of an SDRAM row may result in wrong data read from the next row if the following condition is met:

- the read access to SDRAM is 16- or 8-bit (32-bit accesses are not subject to the failure),
- the RBURST bit of the FMC_SDCR1 register is zero (read FIFO disabled),
- the length of the burst read access is undetermined, following LDM (load multiple) instruction, and
- the burst read spans over multiple SDRAM rows.

Workaround

Enable the read FIFO by setting the RBURST bit of the FMC_SDCR1 register.

2.3.5 Corruption of data read by CPU from the FMC used as stack or heap

Description

When the FMC is used as stack, heap or variable data, an interrupt occurring during a CPU read access to the FMC may result in read data corruption or hard fault exception. Read accesses by bus masters other than the CPU are not subject to this failure. The failure does not occur when the interrupts are disabled.

Workaround

Apply one of the following measures:

- Do not use the FMC as stack or heap, and disable interrupts before reading the FMC by the CPU.
- Read the FMC with DMA and not directly by the CPU.

2.3.6 FMC NOR/PSRAM controller: asynchronous read access on bank 2 to 4 returns wrong data when bank 1 is in synchronous mode (BURSTEN bit is set)

Description

If an interrupt occurs during a CPU AHB read access to one NOR/PSRAM bank (bank 2 to 4) which is enabled in asynchronous mode, while bank 1 of the NOR/PSRAM controller is configured in synchronous read mode (BURSTEN bit set), then the FMC NOR/PSRAM controller returns wrong data. This limitation does not occur when using the DMA or when only bank 1 is used in synchronous mode.

Workaround

If multiple banks are enabled in mixed asynchronous and synchronous modes, use any NOR/PSRAM bank for synchronous read accesses, except for bank 1. As a consequence the continuous clock feature is not available.

2.3.7 FMC dynamic and static bank switching

Description

The dynamic and static banks cannot be accessed concurrently.

Workaround

Do not use dynamic and static banks at the same time. The SDRAM device must be in self-refresh before switching to the static memory mapped on the NOR/PSRAM or NAND/PC-Card controller. Before switching from static memory to SDRAM, issue a Normal command to wake-up the device from self-refresh mode.

2.3.8 FMC NOR/PSRAM controller write protocol violation

Description

When an interrupted asynchronous or synchronous CPU read access to any NOR/PSRAM FMC bank is followed by an asynchronous write access to the same bank or to any other NOR/PSRAM FMC bank, this causes a write protocol violation. There is no functional issue but the FMC NOR/PSRAM controller write protocol is violated since the FMC_NWE signal is de-asserted at the same time as the Chip select (FMC_NEx).

Workaround

None.

2.3.9 Data corruption during burst read from FMC synchronous memory

Description

A burst read from static memory can be corrupted if the following condition is met:

- one FMC bank is configured in synchronous mode with WAITEN bit set while another FMC bank is used with WAITEN bit cleared,
- a burst read is ongoing from static synchronous memory with wait feature enabled,

- the wait signal (asserted by the synchronous memory) is active,
- the burst read transaction is followed by an access to an FMC dynamic or static bank of which the WAITEN bit is disabled in the FMC_BCRx register.

Workaround

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

2.3.10 Missing information on prohibited 0xFF value of NAND transaction wait timing

Description

Some reference manual revisions may omit the information that the value 0xFF is prohibited for the wait timing of NAND transactions in their corresponding memory space (common or attribute).

Whatever the setting of the PWAITEN bit of the FMC_PCRx register, the wait timing set to 0xFF would cause a NAND transaction to stall the system with no fault generated.

This is a documentation error rather than a device limitation.

Workaround

No application workaround required provided that the 0xFF wait timing value is duly avoided.

2.3.11 Missed burst write transaction on multiplexed PSRAM

Description

If an interrupt occurs during a CPU AHB read from an FMC bank of multiplexed PSRAM with the write burst feature disabled (CBURSTRW bit of the FMC_BCRx register cleared), the following burst write access to another FMC bank with the write feature enabled is missed.

Workaround

Set the CBURSTRW bit even for the FMC bank that is not targeted by burst write accesses. Burst write transactions have no effect on the asynchronous read protocol. However, to perform write accesses to asynchronous FMC bank while the CBURSTRW bit is set, the AHB size must be equal to memory data width.

2.3.12 FMC NOR/PSRAM controller bank switch with different BUSTURN durations

Description

The system hangs when the FMC NOR/PSRAM memory controller switches between two banks while:

- one NOR/PSRAM bank is configured in synchronous mode and the BUSTURN bitfields in FMC_BTRx/ FMC_BTWx registers are set to non-zero values,
- another NOR/PSRAM bank is configured in asynchronous multiplexed mode and BUSTURN bitfields are set to 0,
- FMC clock division ratio (CLKDIV) is higher than or equal to four HCLK periods, and
- a single read transaction from the bank operating in synchronous mode is followed by any transaction in another bank operating in asynchronous multiplexed mode.

Workaround

If several NOR/PSRAM banks are used, the BUSTURN bitfield must be set to a nonzero value for each bank.

2.3.13 Wrong data read from a busy NAND memory

Description

When a read command is issued to the NAND memory, the R/B signal gets activated upon the de-assertion of the chip select. If a read transaction is pending, the NAND controller might not detect the R/B signal (connected to NWAIT) previously asserted and sample a wrong data. This problem occurs only when the MEMSET timing is configured to 0x00 or when ATTHOLD timing is configured to 0x00 or 0x01.

Workaround

Either configure MEMSET timing to a value greater than 0x00 or ATTHOLD timing to a value greater than 0x01.

2.3.14 Spurious clock stoppage with continuous clock feature enabled

Description

With the continuous clock feature enabled, the FMC_CLK clock may spuriously stop when:

- the FMC_CLK clock is divided by 2, and
- an FMC bank set as 32-bit is accessed with a byte access.

division ratio set to 2, the FMC_CLK clock may spuriously stop upon an

Note: With static memories, a spuriously stopped clock can be restarted by issuing a synchronous transaction or any asynchronous transaction different from a byte access on 32-bit data bus width.

Workaround

With the continuous clock feature enabled, do not set the FMC_CLK clock division ratio to 2 when accessing 32-bit asynchronous memories with byte access.

2.3.15 SDRAM bank address corruption upon an interruption of CPU burst read access

Description

If an interrupt occurs during an CPU AHB burst read access to one SDRAM internal bank followed by a second read to another SDRAM internal bank, it may result in wrong data read if the following condition is met:

- SDRAM read FIFO enabled. RBURST bit of the FMC_SDCR1 register is set, and
- an interrupt occurs while CPU is performing an AHB incrementing burst read access of unspecified length (using LDM = Load Multiple instruction) to one SDRAM internal bank, followed by another CPU read access to another SDRAM internal bank.

The issue occurs only when the address of the second data read access in the following bank matches one of the data addresses in the read FIFO.

Workaround

Apply one of the following measures:

- Disable the SDRAM read FIFO.
- Send a “PRECHARGE ALL” command before the second read access (before switching to another SDRAM internal bank).
- Perform a dummy write before switching to another SDRAM internal bank.

2.4 SDIO

2.4.1 Wrong data written during SDIO hardware flow control

Description

When enabling the hardware flow control by setting bit 14 of the SDIO_CLKCR register, glitches may occur on the SDIOCLK output clock, resulting in wrong data to be written to the SD/MMC card or to the SDIO device. As a consequence, a CRC error is reported to the SD/SDIO MMC host interface (DCRCFAIL bit set in the SDIO_STA register).

Workaround

None.

Note: Do not use the hardware flow control. Overrun errors (Rx mode) and FIFO underrun (Tx mode) must be managed by the application software.

2.4.2 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 SDIO_STA register is set.

Workaround

The CCRCFAIL bit in the SDIO_STA register must be ignored by the software. CCRCFAIL must be cleared by setting the CCRCFAILC bit of the SDIO_ICR register after receiving the response to the CMD5 command.

2.4.3 Data corruption in SDIO clock dephasing (NEGEDGE) mode

Description

Setting the NEGEDGE bit may lead to invalid data and command response read.

Workaround

None.

Avoid configuring the NEGEDGE bit to 1.

2.4.4 CE-ATA multiple write command and card busy signal management

Description

The CE-ATA card may inform the host that it is busy by driving the SDIO_D0 line low, two cycles after the transfer of a write command (RW_MULTIPLE_REGISTER or RW_MULTIPLE_BLOCK). When the card is in a busy state, the host must not send any data until the BUSY signal is de-asserted (SDIO_D0 released by the card).

This condition is not respected if the data state machine leaves the IDLE state (write operation programmed and started, DTEN = 1, DTDIR = 0 in the SDIO_DCTRL register, and TXFIFOE = 0 in the SDIO_STA register).

As a consequence, the write transfer fails and the data lines are corrupted.

Workaround

After sending the write command (RW_MULTIPLE_REGISTER or RW_MULTIPLE_BLOCK), the application must check that the card is not busy by polling the BSY bit of the ATA status register using the FAST_IO (CMD39) command before enabling the data state machine.

2.4.5 No underrun detection with wrong data transmission

Description

In case there is an ongoing data transfer from the SDIO host to the SD card and the hardware flow control is disabled (bit 14 of the SDIO_CLKCR is not set), if an underrun condition occurs, the controller may transmit a corrupted data block (with a wrong data word) without detecting the underrun condition when the clock frequencies have the following relationship:

$[3 \times \text{period}(\text{PCLK2}) + 3 \times \text{period}(\text{SDIOCLK})] \geq (32/(\text{BusWidth})) \times \text{period}(\text{SDIO\_CK})$

Workaround

Avoid the above-mentioned clock frequency relationship, by applying one of the following measures:

- Increment the APB frequency, or
- decrease the transfer bandwidth, or
- reduce SDIO_CK frequency.

2.5 ADC

2.5.1 ADC sequencer modification during conversion

Description

When a software start-of-conversion is used as an ADC trigger, and if the ADC_SQRx or ADC_JSQRx register is modified during the conversion, the current conversion is reset and the ADC does not automatically restart the new conversion sequence. The hardware start-of-conversion trigger is not impacted and the ADC automatically restarts the new sequence when the next hardware trigger occurs.

Workaround

When a software start-of-conversion is used, apply the following sequence:

1. First set the SWSART bit in the ADC_CR2 register.
2. Then restart the new conversion sequence.

2.6 DAC

2.6.1 DMA request not automatically cleared by clearing DMAEN

Description

Upon an attempt to stop a DMA-to-DAC transfer, the DMA request is not automatically cleared by clearing the DAC channel bit of the DAC_CR register (DMAEN) or by disabling the DAC clock.

If the application stops the DAC operation while the DMA request is pending, the request remains pending while the DAC is reinitialized and restarted, with the risk that a spurious DMA request is serviced as soon as the DAC is enabled again.

Workaround

Apply the following sequence to stop the current DMA-to-DAC transfer and restart the DAC:

1. Check if DMAUDR bit is set in DAC_CR.
2. Clear the DAC channel DMAEN bit.
3. Disable the DAC clock.
4. Reconfigure the DAC, DMA and the triggers.
5. Restart the application.

2.6.2 DMA underrun flag not set when an internal trigger is detected on the clock cycle of the DMA request acknowledge

Description

When the DAC channel operates in DMA mode (DMAEN of DAC_CR register set), the DMA channel underrun flag (DMAUDR of DAC_SR register) fails to rise upon an internal trigger detection if that detection occurs during the same clock cycle as a DMA request acknowledge. As a result, the user application is not informed that an underrun error occurred.

This issue occurs when software and hardware triggers are used concurrently to trigger DMA transfers.

Workaround

None.

2.7 TIM

2.7.1 PWM re-enabled in automatic output enable mode despite of system break

Description

In automatic output enable mode (AOE bit set in TIMx_BDTR register), the break input can be used to do a cycle-by-cycle PWM control for a current mode regulation. A break signal (typically a comparator with a current threshold) disables the PWM output(s) and the PWM is re-armed on the next counter period.

However, a system break (typically coming from the CSS Clock security System) is supposed to stop definitively the PWM to avoid abnormal operation (for example with PWM frequency deviation).

In the current implementation, the timer system break input is not latched. As a consequence, a system break indeed disables the PWM output(s) when it occurs, but PWM output(s) is (are) re-armed on the following counter period.

Workaround

Preferably, implement control loops with the output clear enable function (OCxCE bit in the TIMx_CCMR1/CCMR2 register), leaving the use of break circuitry solely for internal and/or external fault protection (AOE bit reset).

2.7.2 TRGO and TRGO2 trigger output failure

Description

Some reference manual revisions may omit the following information.

The timers can be linked using ITRx inputs and TRGOx outputs. Additionally, the TRGOx outputs can be used as triggers for other peripherals (for example ADC). Since this circuitry is based on pulse generation, care must be taken when initializing master and slave peripherals or when using different master/slave clock frequencies:

- If the master timer generates a trigger output pulse on TRGOx prior to have the destination peripheral clock enabled, the triggering system may fail.
- If the frequency of the destination peripheral is modified on-the-fly (clock prescaler modification), the triggering system may fail.

As a conclusion, the clock of the slave timer or slave peripheral must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are being received from the master timer.

This is a documentation issue rather than a product limitation.

Workaround

No application workaround is required or applicable as long as the application handles the clock as indicated.

2.7.3 Consecutive compare event missed in specific conditions

Description

Every match of the counter (CNT) value with the compare register (CCR) value is expected to trigger a compare event. However, if such matches occur in two consecutive counter clock cycles (as consequence of the CCR value change between the two cycles), the second compare event is missed for the following CCR value changes:

- in edge-aligned mode, from ARR to 0:
  - first compare event: CNT = CCR = ARR
  - second (missed) compare event: CNT = CCR = 0
- in center-aligned mode while up-counting, from ARR-1 to ARR (possibly a new ARR value if the period is also changed) at the crest (that is, when TIMx_RCR = 0):
  - first compare event: CNT = CCR = (ARR-1)
  - second (missed) compare event: CNT = CCR = ARR
- in center-aligned mode while down-counting, from 1 to 0 at the valley (that is, when TIMx_RCR = 0):
  - first compare event: CNT = CCR = 1
  - second (missed) compare event: CNT = CCR = 0

This typically corresponds to an abrupt change of compare value aiming at creating a timer clock single-cycle-wide pulse in toggle mode.

As a consequence:

- In toggle mode, the output only toggles once per counter period (squared waveform), whereas it is expected to toggle twice within two consecutive counter cycles (and so exhibit a short pulse per counter period).
- In center mode, the compare interrupt flag does note rise and the interrupt is not generated.

Note: The timer output operates as expected in modes other than the toggle mode.

Workaround

None.

2.7.4 Output compare clear not working with external counter reset

Description

The output compare clear event (ocref_clr) is not correctly generated when the timer is configured in the following slave modes: Reset mode, Combined reset + trigger mode, and Combined gated + reset mode.

The PWM output remains inactive during one extra PWM cycle if the following sequence occurs:

1. The output is cleared by the ocref_clr event.
2. The timer reset occurs before the programmed compare event.

Workaround

Apply one of the following measures:

- Use BKIN (or BKIN2 if available) input for clearing the output, selecting the Automatic output enable mode (AOE = 1).
- Mask the timer reset during the PWM ON time to prevent it from occurring before the compare event (for example with a spare timer compare channel open-drain output connected with the reset signal, pulling the timer reset line down).

2.8 IWDG

2.8.1 RVU flag not reset in Stop

Description

Successful write to the IWDG_RLR register raises the RVU flag and prevents further write accesses to the register until the RVU flag is automatically cleared by hardware. However, if the device enters Stop mode while the RVU flag is set, the hardware never clears that flag, and writing to the IWDG_RLR register is no longer possible.

Workaround

Ensure that the RVU flag is cleared before entering Stop mode.

2.8.2 PVU flag not reset in Stop

Description

Successful write to the IWDG_PR register raises the PVU flag and prevents further write accesses to the register until the PVU flag is automatically cleared by hardware. However, if the device enters Stop mode while the PVU flag is set, the hardware never clears that flag, and writing to the IWDG_PR register is no longer possible.

Workaround

Ensure that the PVU flag is cleared before entering Stop mode.

2.8.3 RVU flag not cleared at low APB clock frequency

Description

Successful write to the IWDG_RLR register raises the RVU flag and prevents further write accesses to the register until the RVU flag is automatically cleared by hardware. However, at APB clock frequency lower than twice the IWDG clock frequency, the hardware never clears that flag, and writing to the IWDG_RLR register is no longer possible.

Workaround

Set the APB clock frequency higher than twice the IWDG clock frequency.

2.8.4 PVU flag not cleared at low APB clock frequency

Description

Successful write to the IWDG_PR register raises the PVU flag and prevents further write accesses to the register until the PVU flag is automatically cleared by hardware. However, at APB clock frequency lower than twice the IWDG clock frequency, the hardware never clears that flag, and writing to the IWDG_PR register is no longer possible.

Workaround

Set the APB clock frequency higher than twice the IWDG clock frequency.

2.9 RTC

2.9.1 Spurious tamper detection when disabling the tamper channel

Description

If the tamper detection is configured for detecting on the falling edge event (TAMPFLT = 00 and TAMPxTRG = 1) and if the tamper event detection is disabled when the tamper pin is at high level, a false tamper event is detected.

Workaround

None.

2.9.2 RTC calendar registers are not locked properly

Description

When reading the calendar registers with BYPSHAD = 0, the RTC_TR and RTC_DR registers may not be locked after reading the RTC_SSR register. This happens if the read operation is initiated one APB clock period before the shadow registers are updated. This can result in a non-consistency of the three registers. Similarly, the RTC_DR register can be updated after reading the RTC_TR register instead of being locked.

Workaround

Apply one of the following measures:

- Use BYPSHAD = 1 mode (bypass shadow registers), or
- If BYPSHAD = 0, read SSR again after reading SSR/TR/DR to confirm that SSR is still the same, otherwise read the values again.

2.9.3 RTC interrupt can be masked by another RTC interrupt

Description

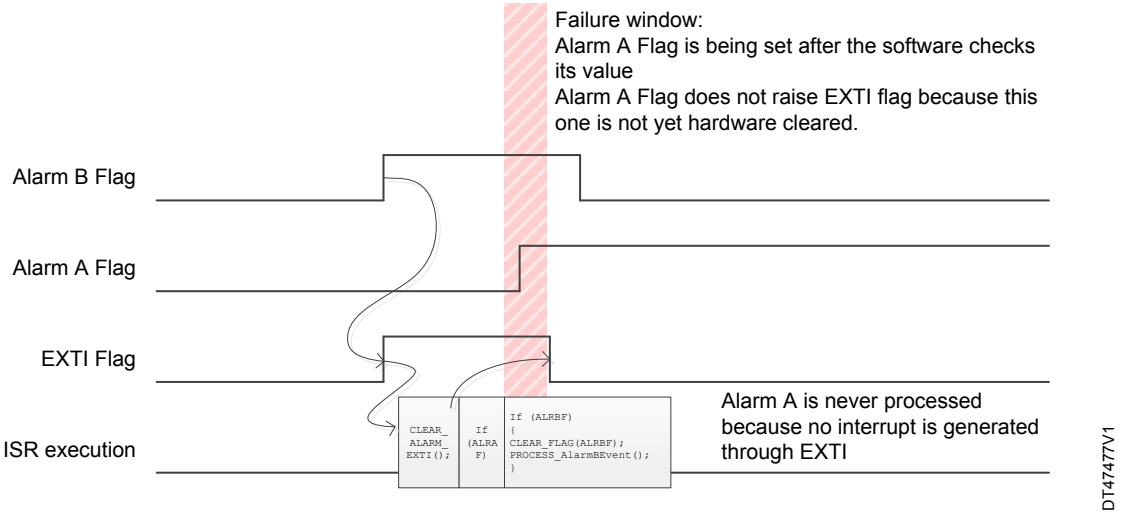
One RTC interrupt request can mask another RTC interrupt request if they share the same EXTI configurable line. For example, interrupt requests from Alarm A and Alarm B or those from tamper and timestamp events are OR-ed to the same EXTI line (refer to the *EXTI line connections* table in the *Extended interrupt and event controller (EXTI)* section of the reference manual).

The following code example and figure illustrate the failure mechanism: The Alarm A event is lost (fails to generate interrupt) as it occurs in the failure window, that is, after checking the Alarm A event flag but before the effective clear of the EXTI interrupt flag by hardware. The effective clear of the EXTI interrupt flag is delayed with respect to the software instruction to clear it.

Alarm interrupt service routine:

```
void RTC_Alarm_IRQHandler(void)
{
    CLEAR_ALARM_EXTI(); /* Clear the EXTI line flag for RTC alarms*/
    If(ALRAF) /* Check if Alarm A triggered ISR */
    {
        CLEAR_FLAG(ALRAF); /* Clear the Alarm A interrupt pending bit */
        PROCESS_AlarmAEvent(); /* Process Alarm A event */
    }
    If(ALRBF) /* Check if Alarm B triggered ISR */
    {
        CLEAR_FLAG(ALRBF); /* Clear the Alarm B interrupt pending bit */
        PROCESS_AlarmBEvent(); /* Process Alarm B event */
    }
}
```

Figure 1. Masked RTC interrupt



Workaround

In the interrupt service routine, apply three consecutive event flag ckecks - source one, source two, and source one again, as in the following code example:

```
void RTC_Alarm_IRQHandler(void)
{
    CLEAR_ALARM_EXTI(); /* Clear the EXTI's line Flag for RTC Alarm */
    If(ALRAF) /* Check if AlarmA triggered ISR */
    {
        CLEAR_FLAG(ALRAF); /* Clear the AlarmA interrupt pending bit */
        PROCESS_AlarmAEvent(); /* Process AlarmA Event */
    }
    If(ALRBF) /* Check if AlarmB triggered ISR */
    {
        CLEAR_FLAG(ALRBF); /* Clear the AlarmB interrupt pending bit */
        PROCESS_AlarmBEvent(); /* Process AlarmB Event */
    }
    If(ALRAF) /* Check if AlarmA triggered ISR */
    {
        CLEAR_FLAG(ALRAF); /* Clear the AlarmA interrupt pending bit */
        PROCESS_AlarmAEvent(); /* Process AlarmA Event */
    }
}
```

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

Description

If the INIT bit of the RTC_ISR 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 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.

2.9.5 Alarm flag may be repeatedly set when the core is stopped in debug

Description

When the core is stopped in debug mode, the clock is supplied to subsecond RTC alarm downcounter even when the device is configured to stop the RTC in debug.

As a consequence, when the subsecond counter is used for alarm condition (the MASKSS[3:0] bitfield of the RTC_ALRMASSR and/or RTC_ALRMBSSR register set to a non-zero value) and the alarm condition is met just before entering a breakpoint or printf, the ALRAF and/or ALRBF flag of the RTC_SR register is repeatedly set by hardware during the breakpoint or printf, which makes any attempt to clear the flag(s) ineffective.

Workaround

None.

2.9.6 Detection of a tamper event occurring before enabling the tamper detection is not supported in edge detection mode

Description

When the tamper detection is enabled in edge detection mode (TAMPFLT = 00):

- When TAMPxTRG = 0 (rising edge detection): if the tamper input is already high before enabling the tamper detection, the tamper event may or may not be detected when enabling the tamper detection. The probability to detect it increases with the APB frequency.
- When TAMPxTRG = 1 (falling edge detection): if the tamper input is already low before enabling the tamper detection, the tamper event is not detected when enabling the tamper detection.

Workaround

Check the I/O state by software in the GPIO registers, just after enabling the tamper detection and before writing sensitive values in the backup registers. This ensures that no active edge occurred before enabling the tamper event detection.

2.10 I2C

2.10.1 Spurious bus error detection in controller mode

Description

In controller 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 controller mode and any such transfer continues normally.

Workaround

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

2.10.2 SMBus standard not fully supported

Description

The I2C peripheral is not fully compliant with the SMBus v2.0 standard since it does not support the capability to NACK an invalid byte/command.

Workaround

A higher-level mechanism must be used to verify that a write operation is being performed correctly at the target device, such as:

- the SMBAL pin if it is supported by the host
- the alert response address (ARA) protocol
- the host-notify protocol

2.10.3 Start cannot be generated after a misplaced Stop

Description

If a controller generates a misplaced Stop on the bus (bus error) while the microcontroller I2C peripheral attempts to switch to Controller mode by setting the START bit, the Start condition is not properly generated.

Workaround

In the I^2C standard, it is allowed to send a Stop only at the end of the full byte (8 bits + acknowledge), so this scenario is not allowed. Other derived protocols such as CBUS allow it, but they are not supported by the I2C peripheral.

A software workaround consists in asserting the software reset using the SWRST bit of the I2C_CR1 control register.

2.10.4 Mismatch on the “Setup time for a repeated Start condition” timing parameter

Description

In case of repeated Start, the “Setup time for a repeated Start condition” (named Tsu;sta in the I^2C specification) can be slightly violated when the I2C operates in Controller standard mode at a frequency between 88 kHz and 100 kHz.

The issue can occur only in the following configuration:

- In Controller mode
- In Standard mode at a frequency between 88 kHz and 100 kHz (no limitation in Fast mode)
- SCL rise time:
  - If the target does not stretch the clock and the SCL rise time is more than 300 ns (if the SCL rise time is less than 300 ns, the issue does not occur).
  - If the target stretches the clock.

The setup time can be violated independently of the APB peripheral frequency.

Workaround

Reduce the frequency down to 88 kHz or use the I^2C Fast mode, if it is supported by the target.

2.10.5 Data valid time ($t_{VD;DAT}$) violated without the OVR flag being set

Description

The data valid time ($t_{VD;DAT}$, $t_{VD;ACK}$) described by the I^2C standard can be violated (as well as the maximum data hold time of the current data ($t_{HD;DAT}$)) under the conditions described below. This violation cannot be detected because the OVR flag is not set (no transmit buffer underrun is detected).

This limitation can occur only under the following conditions:

- in Target transmit mode
- with clock stretching disabled (NOSTRETCH = 1)
- if the software is late to write to the DR data register, but not late enough to set the OVR flag (the data register is written before)

Workaround

If the controller device allows it, use the clock stretching mechanism by clearing the bit NOSTRETCH of the I2C_CR1 register.

If the controller device does not allow it, ensure that the software is fast enough when polling the TXE or ADDR flag to immediately write to the DR data register. For instance, use an interrupt on the TXE or ADDR flag and boost its priority to the higher level.

2.10.6 Both SDA and SCL maximum rise times (t_r) violated when the VDD_I2C bus voltage is higher than ((V_{DD} + 0.3) / 0.7) V

Description

When an external legacy I^2C bus voltage (V_{DD_I2C}) is set to 5 V while the MCU is powered from V_{DD}, the internal 5-Volt tolerant circuitry is activated as soon the input voltage (V_{IN}) reaches the V_{DD} + diode threshold level. An additional internal large capacitance then prevents the external pull-up resistor (R_P) from rising the SDA and SCL signals within the maximum timing (t_r), which is 300 ns in Fast mode and 1000 ns in Standard mode.

The rise time (t_r) is measured from V_{IL} and V_{IH} with levels set at 0.3 V_{DD_I2C} and 0.7 V_{DD_I2C}.

Workaround

The external V_{DD_I2C} bus voltage must be limited to a maximum value of ((V_{DD} + 0.3) / 0.7) V. As a result, when the MCU is powered from V_{DD} = 3.3 V, V_{DD_I2C}must not exceed 5.14 V to be compliant with I^2C specifications.

2.11 USART

2.11.1 Idle frame is not detected if the receiver clock speed is deviated

Description

If the USART receives an idle frame followed by a character, and the clock of the transmitter device is faster than the USART receiver clock, the USART receive signal falls too early when receiving the character start bit, with the result that the idle frame is not detected (the IDLE flag is not set).

Workaround

None.

2.11.2 In full-duplex mode, the Parity Error (PE) flag can be cleared by writing to the data register

Description

In full-duplex mode, when the Parity Error flag is set by the receiver at the end of a reception, it may be cleared while transmitting by reading the USART_SR register to check the TXE or TC flags and writing data to the data register. Consequently, the software receiver can read the PE flag as '0' even if a parity error occurred.

Workaround

The Parity Error flag should be checked after the end of reception and before transmission.

2.11.3 Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection

Description

If the USART receiver is in Mute mode, and is configured to exit from Mute mode using the address mark detection, when the USART receiver recognizes a valid address with a parity error, it exits from Mute mode without setting the Parity Error flag.

Workaround

None.

2.11.4 Break frame is transmitted regardless of CTS input line status

Description

When the CTS hardware flow control is enabled (CTSE = 1) and the send break bit (SBK) is set, the transmitter sends a break frame at the end of the current transmission regardless of CTS input line status. Consequently, if an external receiver device is not ready to accept a frame, the transmitted break frame is lost.

Workaround

None.

2.11.5 RTS signal abnormally driven low after a protocol violation

Description

When RTS hardware flow control is enabled, the RTS signal goes high when data is received. If this data was not read and new data is sent to the USART (protocol violation), the RTS signal goes back to low level at the end of this new data.

Consequently, the sender gets the wrong information that the USART is ready to receive further data.

On the USART side, an overrun is detected, which indicates that data has been lost.

Workaround

A workaround is required only if the other USART device violates the communication protocol, which is not the case in most applications.

Two workarounds can be used:

- After data reception and before reading the data in the data register, the software takes over the control of the RTS signal as a GPIO, and holds it high as long as needed. If the USART device is not ready, the software holds the RTS pin high, and releases it when the device is ready to receive new data.
- Make sure the time required by the software to read the received data is always lower than the duration of the second data reception. For example, this can be ensured by handling all the receptions in DMA mode.

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

Description

According to ISO/IEC 7816-3 standard, when a character parity error is detected, the receiver shall transmit a NACK error signal 10.5 ± 0.2 ETUs after the character START bit falling edge. In this case, the transmitter is able to detect correctly the NACK signal until 11 ± 0.2 ETUs after the character START bit falling edge. In Smartcard mode, the USART peripheral monitors the NACK signal during the receiver time frame (10.5 ± 0.2 ETUs), while it should wait for it during the transmitter one (11 ± 0.2 ETUs). In real cases, this would not be a problem as the card itself needs to respect a 10.7 ETU period when sending the NACK signal. However, this may be an issue to undertake a certification.

Workaround

None.

2.11.7 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 if the following conditions are met:

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

Workaround

If the application needs to detect the end of a data transfer, check that the TC flag is set, and issue a break request.

2.11.8 Guard time not respected when data are sent on TXE events

Description

In Smartcard mode, when sending a data on TXE event, the programmed guard time is not respected, that is the data written in the data register is transferred to the bus without waiting the completion of the guard-time duration corresponding to the previous transmitted data.

Workaround

Since in Smartcard mode the TC flag is set at the end of the guard time duration, wait until TC is set, then write the data.

2.11.9 RTS is active while RE or UE = 0

Description

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

Workaround

After setting the UE and RE bits, configure the I/O used for RTS as an alternate function.

2.12 SPI/I2S

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 Anticipated communication upon SPI transit from slave receiver to master

Description

Regardless of the master mode configured, the communication clock starts upon setting the MSTR bit even though the SPI is disabled, if transiting from receive-only (RXONLY = 1) or half-duplex receive (BIDIMODE = 1 and BIDIOE = 0) slave mode to master mode.

Workaround

Apply one of the following measures:

- Before transiting to master mode, hardware-reset the SPI via the reset controller.
- Set the MSTR and SPE bits of the SPI configuration register simultaneously, which forces the immediate start of the communication clock. In transmitter configuration, load the data register in advance with the data to send.

2.12.3 Wrong CRC calculation when the polynomial is even

Description

When the CRC is enabled, the CRC calculation is wrong if the polynomial is even.

Workaround

Use odd polynomial.

2.12.4 Corrupted last bit of data and/or CRC received in Master mode with delayed SCK feedback

Description

When performing a receive transaction in I2S or SPI Master mode, the last bit of the transacted frame is not captured when the signal provided by an internal feedback loop from the SCK pin exceeds a critical delay. The lastly transacted bit of the stored data then keeps the value from the pattern received previously. As a consequence, the last receive data bit may be wrong, and/or the CRCERR flag can be unduly asserted in the SPI mode if any data under checksum, and/or just the CRC pattern is wrongly captured.

In SPI mode, data are synchronous with the APB clock. A delay of up to two APB clock periods can thus be tolerated for the internal feedback delay.

The I2S mode is more sensitive than the SPI mode, especially in the case where an odd I2S prescaler factor is set and the APB clock is the system clock divided by two. In this case, the internal feedback delay is lower than 1.5 APB clock period.

The main factors contributing to the delay increase are low V_{DD} level, high temperature, high SCK pin capacitive load, and low SCK I/O output speed. The SPI communication speed has no impact.

Workaround

The following workarounds can be adopted, jointly or individually:

- Decrease the APB clock speed.
- Configure the I/O pad of the SCK pin to be faster.

The following table gives the maximum allowable APB frequency (that still prevents the issue from occurring) versus GPIOx_OSPEEDR output speed for the SCK pin, with a 30 pF capacitive load.

Table 5. Maximum allowable APB frequency at 30 pF load

OSPEEDR [1:0] for SCK pin	Max. APB frequency for SPI mode (MHz)	Max. APB frequency for I2S mode (MHz)
11 (very high), 10 (high)	84	42
01 (medium)	75	35
00 (low)	25	16

2.12.5 BSY flag 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 the 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 the data transaction may be not recognized when the software needs to monitor it (for example at the end of a session before entering the low-power mode or before the direction of the 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 signaling 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 must be measured from TXE event signaling 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 above workarounds are not applicable, the following sequence can be used to prevent the synchronization issue during SPI transmit mode:

1. Write the last data to the 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.
5. The BSY flag works correctly and can be used to recognize the end of the transaction.

Note: This workaround can be used only when the CPU has enough performance to disable the SPI after a 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 clock is low, and the data frame is short. In this specific case, the timeout can be measured from TXE, while calculating the fixed number of CPU clock periods corresponding to the time necessary to complete the data frame transaction.

2.12.6 In I2S Slave mode, the WS level must be set by the external master when enabling the I2S

Description

In Slave mode, the WS signal level is used only to start the communication. If the I2S (in Slave mode) is enabled while the master is already sending the clock and the WS signal level is low (for the I2S protocol) or is high (for the LSB or MSB-justified mode), the slave starts communicating data immediately. In this case, the master and slave are desynchronized throughout the whole communication.

Workaround

Make sure the I2S peripheral is enabled when the external master sets the WS line at:

- High level when the I2S protocol is selected.
- Low level when the LSB or MSB-justified mode is selected

2.13 bxCAN

2.13.1 bxCAN time-triggered communication mode not supported

Description

The time-triggered communication mode described in the reference manual is not supported. As a result, timestamp values are not available. The TTCM bit of the CAN_MCR register must be kept cleared (time-triggered communication mode disabled).

Workaround

None.

2.14 OTG_FS

2.14.1 Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_FS registers

Description

When the USB on-the-go full-speed peripheral is in Device mode, interrupting transmit FIFO write sequence with read or write accesses to OTG_FS endpoint-specific registers (those ending in *0* or *x*) leads to corruption of the next data written to the transmit FIFO.

Workaround

Ensure that the transmit FIFO write sequence is not interrupted with accesses to the OTG_FS registers.

2.14.2 Host packet transmission may hang when connecting through a hub to a low-speed device

Description

When the USB on-the-go full-speed peripheral connects to a low-speed device via a hub, the transmitter internal state machine may hang. This leads, after a timeout expiry, to a port disconnect interrupt.

Workaround

None. However, increasing the capacitance on the data lines may reduce the occurrence.

2.14.3 Data in RxFIFO is overwritten when all channels are disabled simultaneously

Description

If the available RxFIFO is just large enough to host one packet plus its data status, and it is currently occupied by the last received data plus its status, and, at the same time, the application requests that more IN channels are disabled, the OTG_FS peripheral does not first check for available space before inserting the disabled status of the IN channels. It just inserts them by overwriting the existing data payload.

Workaround

Use one of the following measures:

- Configure the RxFIFO to host a minimum of 2 × MPSIZ + 2 × data status entries.
- Check the RXFLVL bit (RxFIFO nonempty) of the OTG_FS_GINTSTS register before disabling each IN channel. If this bit is cleared, then disable an IN channel at a time. Each time the application disables an IN channel, it first has to check that the RXFLVL bit = 0 condition is true.

2.14.4 OTG host blocks the receive channel when receiving IN packets and no TxFIFO is configured

Description

When receiving data, the OTG_FS core erroneously checks for available TxFIFO space when it should only check for RxFIFO space. If the OTG_FS core cannot see any space allocated for data transmission, it blocks the reception channel, and no data is received.

Workaround

Set at least one TxFIFO equal to the maximum packet size. In this way, the host application, which intends to supports only IN traffic, also has to allocate some space for the TxFIFO.

Since a USB host is expected to support any kind of connected endpoint, it is good practice to always configure enough TxFIFO space for OUT endpoints.

2.14.5 Host channel-halted interrupt not generated when the channel is disabled

Description

When the application enables then immediately disables the host channel before the OTG_FS host has had time to begin the transfer sequence, the OTG_FS core, as a host, does not generate a channel-halted interrupt. The OTG_FS core continues to operate normally.

Workaround

Do not disable the host channel immediately after enabling it.

2.14.6 Wrong software-read OTG_FS_DCFG register values

Description

When the application writes to the DAD and PFIVL bitfields of the OTG_FS_DCFG register, and then reads the newly written bitfield values, the read values may not be correct.

However, the values written by the application are correctly retained by the core, and the normal operation of the device is not affected.

Workaround

Do not read the OTG_FS_DCFG register DAD and PFIVL bitfields just after programming them.

2.15 OTG_HS

2.15.1 Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_HS registers

Description

When the USB on-the-go high-speed peripheral is in Device mode, interrupting transmit FIFO write sequence with read or write accesses to OTG_HS endpoint-specific registers (those ending in *0* or *x*) leads to corruption of the next data written to the transmit FIFO.

Workaround

Ensure that the transmit FIFO write sequence is not interrupted with accesses to the OTG_HS registers. Note that enabling DMA mode guarantees this.

2.15.2 Host packet transmission may hang when connecting the full speed interface through a hub to a low-speed device

Description

When the USB on-the-go high-speed peripheral is used with the full speed interface (DM and DP pins, N.B. not available on all devices), and connects to a low-speed device via a hub, the transmitter internal state machine may hang. This leads, after a timeout expiry, to a port disconnect interrupt.

Workaround

None. However, increasing the capacitance on the data lines may reduce the occurrence.

2.16 ETH

2.16.1 Incorrect L3 checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads

Description

The application provides a frame-by-frame control to instruct the MAC to insert the layer 3 (L3) checksums for TCP, UDP and ICMP packets. When automatic checksum insertion is enabled and the input packet is an IPv6 packet without the TCP, UDP or ICMP payload, then the MAC may incorrectly insert a checksum into the packet. For IPv6 packets without a TCP, UDP or ICMP payload, the MAC core considers the next header (NH) field as the extension header and continues to parse the extension header. Sometimes, the payload data in such packets matches the NH field for TCP, UDP or ICMP and, as a result, the MAC core inserts a checksum.

Workaround

When the IPv6 packets have a TCP, UDP or ICMP payload, enable checksum insertion for transmit frames, or bypass checksum insertion by using the CIC bits of the TDES0 transmit descriptor word0.

2.16.2 The Ethernet MAC processes invalid extension headers in the received IPv6 frames

Description

In IPv6 frames, the extension headers which precede the actual IP payload may or may not be present. The Ethernet MAC processes the following extension headers defined in the IPv6 protocol: hop-by-hop options header, routing header and destination options header.

All extension headers, except the hop-by-hop extension header, can be present multiple times and in any order before the actual IP payload. The hop-by-hop extension header, if present, has to come immediately after the IPv6 main header.

The Ethernet MAC processes all extension headers whether valid or invalid including the hop-by-hop extension headers that are present after the first extension header. For this reason, the GMAC core accepts IPv6 frames with invalid hop-by-hop extension headers. As a consequence, it accepts any IP payload as valid IPv6 frames with TCP, UDP or ICMP payload, and then incorrectly update the receive status of the corresponding frame.

Workaround

None.

2.16.3 MAC stuck in the idle state on receiving the TxFIFO flush command exactly one clock cycle after a transmission completes

Description

When the software issues a TxFIFO flush command, the transfer of frame data stops, even in the middle of a frame transfer. The TxFIFO read controller goes into the Idle state by clearing the TFRS [1:0] bit field of the ETH_MACDBGR register. It then resumes its normal operation.

However, if the TxFIFO read controller receives the TxFIFO flush command exactly one clock cycle after receiving the status from the MAC, the controller remains stuck in the Idle state and stops transmitting frames from the TxFIFO. The system only recovers from this state with a reset (for example a soft reset).

Workaround

Wait until the TxFIFO is empty before using the TxFIFO flush command.

2.16.4 Transmit frame data corruption

Description

Frame data may get corrupted when the TxFIFO repeatedly switches from non-empty to empty, and back to non-empty again for a very short period, without causing any underflow.

The issue occurs when switching back and forth between non-empty and empty happens when the rate the data is being written to the TxFIFO is almost equal to or a little slower than the rate at which the data is read.

This corruption cannot be detected by the receiver when the CRC is inserted by the MAC, as the corrupted data is used for the CRC computation.

Workaround

Use the transmit Store-and-Forward mode by setting the TSF bit of the ETH_DMAOMR register. In this mode, the data is transmitted only when the whole packet is available in the TxFIFO.

2.16.5 Incorrect status and corrupted frames when RxFIFO overflow occurs on the penultimate word of Rx frames

Description

When operating in Threshold mode, the RxFIFO may overflow when the received frame data is written faster than the speed at which the application reads it from the RxFIFO. The RxFIFO overflow is declared at the moment that a non-EOF word is received and the RxFIFO has only two locations available. The receiver descriptor overflow error (OE) bit of the RDES0 receive descriptor word0 is set to indicate that the receive frame is incomplete.

The problem occurs after the following events:

1. RxFIFO overflow is declared exactly on the penultimate word of the Rx Frame. The EOF word is received in the next clock cycle.
2. The EOF word has exactly one valid byte. This is possible only when the length of the packet, after CRC or PAD stripping (if enabled), is a multiple of 4 bytes plus 1 (for example, 5, 9, 13, 17).

After the above sequence, the frames status information is corrupted and the overflow error flag is not set. Furthermore, if the next frame arrives soon enough, the MAC might falsely interpret that there is space in the RxFIFO and overwrite unread data with the next frame, thus corrupting the existing frames.

The MAC recovers automatically after transferring a few corrupt or incorrect packets.

Workaround

Operate the RxFIFO in the Store-and-Forward mode.

2.16.6 Successive write operations to the same register might not be fully taken into account

Description

A write to a register might not be fully taken into account if a previous write to the same register is performed within a time period of four TX_CLK/RX_CLK clock cycles. When this error occurs, reading the register returns the most recently written value, but the Ethernet MAC continues to operate as if the latest write operation never occurred.

Refer to the following table for the registers and bits impacted by this limitation.

Impacted registers and bits

Register name	Bit number	Bit name
DMA registers		

Register name	Bit number	Bit name
ETH_DMABMR	7	EDFE
ETH_DMAOMR	26	DTCEFD
	25	RSF
	20	FTF
	7	FEF
	6	FUGF
	4:3	RTC
GMAC registers		
ETH_MACCR	25	CSTF
	23	WD
	22	JD
	19:17	IFG
	16	CSD
	14	FES
	13	ROD
	12	LM
	11	DM
	10	IPCO
	9	RD
	7	APCS
	6:5	BL
	4	DC
	3	TE
	2	RE
ETH_MACFFR MAC	-	frame filter register
ETH_MACHTHR	31:0	Hash Table High Register
ETH_MACHTLR	31:0	Hash Table Low Register
ETH_MACFCR	31:16	PT
	7	ZQPD
	5:4	PLT
	3	UPFD
	2	RFCE
	1	TFCE
	0	FCB/BPA
ETH_MACVLANTR	16	VLANTC
	15:0	VLANTI
ETH_MACRWUFFR	-	all remote wakeup registers
ETH_MACPMTCSR	31	WFFRPR
	9	GU
	2	WFE
	1	MPE

Register name	Bit number	Bit name
ETH_MACPMTCSR	0	PD
ETH_MACA0HR	-	MAC address 0 high register
ETH_MACA0LR	-	MAC address 0 low register
ETH_MACA1HR	-	MAC address 1 high register
ETH_MACA1LR	-	MAC address 1 low register
ETH_MACA2HR	-	MAC address 2 high register
ETH_MACA2LR	-	MAC address 2 low register
ETH_MACA3HR	-	MAC address 3 high register
ETH_MACA3LR	-	MAC address 3 low register
IEEE 1588 time stamp registers		
ETH_PTPTSCR	18	TSPFFMAE
	17:16	TSCNT
	15	TSSMRME
	14	TSSEME
	13	TSSIPV4FE
	12	TSSIPV6FE
	11	TSSPTPOEFE
	10	TSPTPPSV2E
	9	TSSSR
	8	TSSARFE
	5	TSARU
	3	TSSTU
	2	TSSTI
	1	TSFCU
	0	TSE

Workaround

Applty on of the following measures:

- Ensure a delay of four TX_CLK/RX_CLK clock cycles between the successive write operations to the same register.
- Make several successive write operations without delay, then read the register when all the operations are complete, and finally reprogram it after a delay of four TX_CLK/RX_CLK clock cycles.

2.16.7 Incorrect remote wakeup on global unicast packet

Description

The PMT remote wakeup block can be enabled to generate remote wakeup interrupt on receiving a global unicast packet, for example a unicast destination address (DA) that perfectly matches one of the MAC address registers enabled for DA.

However, the PMT remote wakeup block generates interrupts for any unicast packet that passes the DA filter (not necessarily for a unicast packet that passes DA perfect filter). For example, the address filter is set for inverse filtering or hash filtering, it gives a PASS for any packet whose DA is accepted by the inverse/hash filter. This results in power down exit on unintended received packets.

Workaround

Only enable DA perfect filter when global unicast based remote wakeup is enabled.

2.16.8 Overflow status bits of missed frame and buffer overflow counters are cleared without a read operation

Description

The direct memory access (DMA) controller maintains two counters to track the number of frames missed because of the following events:

- Rx descriptor not being available
- Rx FIFO buffer overflow during reception

The missed frame and buffer overflow counter of the ETH_DMAMFBOCR register indicate the current value of the missed frames and FIFO overflow frame counters. This register also has the overflow status bits: OFOC bit OMFC bit which indicate whether the rollover occurred for the respective counters and are set when the respective counters roll over. They should remain set until this register is read.

However, when the counter rollover occurs a second time after the status bit is set, the respective status bits are cleared. Therefore, the application may incorrectly detect that the rollover did not occur since the last read operation.

Workaround

The application should read the missed frame and buffer overflow counter register periodically (or after the overflow or rollover status bits are set) such that the counter rollover does not occur twice between read operations.

2.16.9 MAC may provide incorrect Rx status for the MAC control frames when receive checksum offload is enabled

Description

The MAC can be programmed to forward the MAC control frames (with length/type field 0x8808) to the application by setting the PCF bits of the ETH_MACFFR register. When the IPv4 checksum offload function is enabled by setting the IPCO bit of the ETH_MACCR register and clearing the EDFE bit of the ETH_DMABMR register, the MAC provides the encoded Rx status in the FT bit, IPHCE/TSV bit, and the PCE/ESA bit of the RDES0 receive descriptor word0. When a MAC control frame is received, the MAC should provide 0b011 in RDES0 receive descriptor word0 of Rx status. This indicates that an Ethernet type frame is received, which is neither IPv4 nor IPv6 (the checksum offload engine bypasses the checksum completely).

However, when a MAC control frame is received with IPv4 checksum offload function enabled without the extended status, the MAC provides incorrect Rx status (0b100) in RDES0 receive descriptor word0, indicating that an IPv4 or IPv6 type frame is received with no checksum error.

Workaround

Ignore these status bits, as reported in the RDES0 receive descriptor word0, about IP header error after it identifies the control packet from the received packet and processes it accordingly.

2.16.10 MAC may provide an inaccurate Rx status when receive checksum offload is enabled in cut-through mode

Description

The MAC can be programmed in cut-through mode by resetting the DTCEFD bit of the ETH_DMAOMR register. When IPv4 checksum offload function is enabled by setting the IPCO bit of ETH_MACCR register and extended status is not enabled, the MAC provides the bit 25 (error summary), bit 14 (descriptor error), bit 11 (overflow error), bit 7 (IPC checksum error), bit 6 (late collision), bit 4 (watchdog error), bit 3 (receive error) and bit 0 (payload checksum error) of the RDES0: Receive descriptor Word0.

However, when a frame with a payload checksum error is received with the IPv4 checksum offload function enabled without the extended status, the MAC provides a correct error summary status but an incorrect Rx status on bit 0, indicating the IPv4 or IPv6 type frame is received with no checksum error. The source of the error is not specified, none of the individual status bits are set.

Workaround

Enable the Store-and-Forward mode by setting the RSF bit of the ETH_DMAOMR register. In this case the faulty frames are silently discarded as expected.

2.16.11 MAC may not drop received giant error frames

Description

The MAC considers a received frame with a length of more than 1522 bytes, as a giant error frame. When RxFIFO is operating in the Store-and-Forward mode and is programmed to drop error frames, by resetting the FEF bit of the ETH_DMAOMR, all error frames should be dropped in the FIFO layer. Due to the error frames being dropped in the FIFO layer, the DMA controller does not send them to the host.

However, the MAC does not drop the giant error frames and the DMA controller unnecessarily wastes system bandwidth transferring the giant frame to the host.

Workaround

The software driver must check the FL bits of the RDES0 receive descriptor word0, ignore or drop the frame and not forward it to the upper layer.

The frame length field is valid only when the LS bit of the RDES0 receive descriptor word0 register is set and the DE bit of the RDES0 receive descriptor word0 is reset.

Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on www.st.com for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

Revision history

Table 6. **Document revision history**

Date	Version	Changes
11-Feb-2013	1	Initial release.
25-Feb-2013	2	Document converted to new template. Added Section 2.11.4: Corruption of data read from the FMC
26-Apr-2013	3	Added Silicon revision Y. Removed the reference to 'Cortex-M4F' in the whole document. Updated Section 2.11.1: Dummy read cycles inserted when reading synchronous memories. Added Section 2.1.3: Wakeup sequence from Standby mode when using more than one wakeup source, Section 2.10.5: Successive write operations to the same register might not be fully taken into account and Section 2.8.3: FSMC NOR Flash/PSRAM controller asynchronous access on bank 2 to 4 when bank 1 is in synchronous mode (CBURSTRW bit is set). Removed limitation 2.10.3 SDIO clock divider BYPASS mode may not work properly. Updated Section 2.12.5: No underrun detection with wrong data transmission.
19-Sep-2013	4	Added Section 2.8.1: bxCAN time triggered communication mode not supported. Added STM32F429xx and STM32F439xx devices. Removed FSMC limitations. Added Section 2.4.5: Both SDA and SCL maximum rise time (tr) violated when VDD_I2C bus higher than ((VDD+0.3) / 0.7) V. Updated Section 2.11.5: Interruption of CPU read burst access to an end of SDRAM row. Added Section 2.11.1: Dummy read cycles inserted when reading synchronous memories, Section 2.11.2: FMC synchronous mode and NWAIT signal disabled, Section 2.11.3: Read access to a non\u0002initialized FMC_SDRAM bank, Section 2.11.4: Corruption of data read from the FMC, Section 2.11.5: Interruption of CPU read burst access to an end of SDRAM row, Section 2.11.6: FMC NOR/PSRAM controller: asynchronous read access on bank 2 to 4 returns wrong data when bank 1 is in synchronous mode (BURSTEN bit is set) and Section 2.11.7: FMC dynamic and static bank switching. Added Figure 1: TFBGA216 top package view, Figure 2: WLCSP143 top package view, and Figure 3: LQFP208 top package view
23-Sep-2013	5	Updated workaround in Section 2.11.6: FMC NOR/PSRAM controller: asynchronous read access on bank 2 to 4 returns wrong data when bank 1 is in synchronous mode (BURSTEN bit is set).
09-Jan-2014	6	Added silicon revision 1. Added STM32F429xE, STM32F427Ax, STM32F437Ax, STM32F429Ax, and STM32F439Ax part numbers. Removed mention of limitation fix in Section 2.1.8: Over-drive and Under-drive modes unavailability, Section 2.11.4: Corruption of data read from the FMC and Section 2.11.6: FMC NOR/PSRAM controller: asynchronous read access on bank 2 to 4 returns wrong data when bank 1 is in synchronous mode (BURSTEN bit is set). Updated Section 2.11.7: FMC dynamic and static bank switching to indicate the limitation will be fixed in next silicon revision.

Date	Version	Changes
05-May-2014	7	Added silicon revision 3. Added Section 1.2: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used. Added Section 2.1.9: Operating voltage extension down to 1.7 V in the whole temperature range, Section 2.11.8: NAND/PCCard transaction and Wait timing, Section 2.11.9: Data corruption during burst read from FMC synchronous memory, and Section 2.11.10: Missed burst write transaction on multiplexed PSRAM. Moved all device marking schematics to datasheets
20-Jun-2014	8	Added Section 2.1.10: PA12 GPIO limitations, Section 2.11.11: FMC NOR/PSRAM controller write protocol violation and Section 2.11.12: FMC NOR/PSRAM controller bank switch with different BUSTURN durations.
03-Oct-2014	9	Updated FMC NOR/PSRAM controller write protocol violation limitation and FMC synchronous mode and NWAIT signal disabled limitations in Table 4: Summary of silicon limitations. Updated Section 2.1.10: PA12 GPIO limitations. Added Section 2.8.1: bxCAN time triggered communication mode not supported.
02-Nov-2016	10	Added workaround in Section 2.1.6: Delay after an RCC peripheral clock enabling. Added Section 2.1.11: Data cache might be corrupted during Flash read-while-write operation. Added RTC limitations: • Section 2.3.1: Spurious tamper detection when disabling the tamper channel • Section 2.3.2: Detection of a tamper event occurring before enabling the tamper detection is not supported in edge detection mode • Section 2.3.3: RTC calendar registers are not locked properly. Updated limitation description in Section 2.4.2: Start cannot be generated after a misplaced Stop. Added Section 2.4.6: Spurious Bus Error detection in Master mode. Added SPI limitations: • Section 2.5.1: Wrong CRC calculation when the polynomial is even • Section 2.5.2: Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback • Section 2.5.3: Wrong CRC transmitted in Master mode with delayed SCK feedback • Section 2.5.4: BSY bit may stay high at the end of a data transfer in Slave mode Added Section 2.6.2: Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback in Section 2.6: I2S peripheral limitations. Added FMC limitations: Section 2.11.13: Wrong data read from a busy NAND memory and Section 2.11.14: Missed clocks with continuous clock feature enabled.
20-Apr-2017	11	Updated: • Table 2: Device variants • Table 3: Summary of device limitations • Section 2.2.11: PA12 GPIO limitations • Section 2.3.4: Corruption of data read from the FMC • Section 2.3.6: FMC NOR/PSRAM controller: asynchronous read access on bank 2 to 4 returns wrong data when bank 1 is in synchronous mode (BURSTEN bit is set) • Section 2.3.7: FMC dynamic and static bank switching Added Section 2.3.15: SDRAM bank address corruption upon an interruption of CPU read burst access.

Date	Version	Changes
31-Jul-2018	12	Updated: • Table 2: Device variants • Table 3: Summary of device limitations • Section 2.11.2: Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback
25-Feb-2019	13	Updated device marking and REV_ID Table 2: Device variants. Removed revision 4 in Table 3: Summary of device limitations. Change revision 4 to revision 5 and B in Section 2.3.4: Corruption of data read from the FMC, Section 2.3.6: FMC NOR/PSRAM controller: asynchronous read access on bank 2 to 4 returns wrong data when bank 1 is in synchronous mode (BURSTEN bit is set) and Section 2.3.7: FMC dynamic and static bank switching.
09-Jan-2020	14	Renamed Ethernet MAC peripheral into ETH and RTC into RTC and TAMP. Added Possible delay in backup domain protection disabling/enabling after programming the DBP bit, Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_FS registers and Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_HS registers.
09-Oct-2020	15	Added: • Section 2.2.3: Debugging Sleep/Stop mode with WFE/WFI entry • Section 2.2.3: Debugging Sleep/Stop mode with WFE/WFI entry • Section 2.6.1: PWM re-enabled in automatic output enable mode despite of system break • Section 2.6.2: Consecutive compare event missed in specific conditions • Section 2.6.3: Output compare clear not working with external counter resettle of the limitation here • Section 2.6.4: TRGO and TRGO2 trigger output failure • Section 2.17.6: Incorrect status and corrupted frames when RxFIFO overflow occurs on the penultimate word of Rx frames • Section 2.17.7: Incorrect remote wakeup on global unicast packet • Section 2.17.8: Overflow status bits of missed frame and buffer overflow counters are cleared without a read operation • Section 2.17.9: MAC may provide incorrect Rx status for the MAC control frames when receive checksum offload is enabled • Section 2.17.10: MAC may provide an inaccurate Rx status when receive checksum offload is enabled in cut-through mode • Section 2.17.11: MAC may not drop received giant error frames Updated Section 2.16.1: Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_HS registers.
02-Mar-2021	16	Updated Full JTAG configuration without NJTRST pin cannot be used description. Updated Table 3: Summary of device limitations: • Updated references to limitations Wrong CRC calculation when the polynomial is even and Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback. • PA12 GPIO limitations fixed for rev 3, B and 5. Replaced USB on-the-go full-speed peripheral by USB on-the\u0002go high-speed peripheral in . Section 2.16.1: Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_HS registers.
06-Oct-2021	17	Added silicon revision 4. Updated Section 2.15.5: Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_FS registers to refer to USB on-the-go full-speed and OTG_FS. Updated Section 2.16.1: Transmit data FIFO is corrupted when a write sequence to the FIFO is interrupted with accesses to certain OTG_HS registers to refer to USB on-the-go high-speed and OTG_HS.

Date	Version	Changes
18-Nov-2021	18	Updated Section 2.2.14: Setting GPIO properties of PC13 used as RTC_ALARM open-drain output and changed workaround to 'D' in Table 3: Summary of device limitations. Updated Section 2.6.3: Output compare clear not working with external counter resettle of the limitation here. Added new OTG-FS limitation, Section 2.15.6: Host packet transmission may hang when connecting through a hub to a low-speed device and new OTG-HS limitation Section 2.16.2: Host packet transmission may hang when connecting through a hub to a low-speed device.
22-Feb-2023	19	System: • Updated workaround of Debugging Sleep/Stop mode with WFE/WFI entry, and MPU attribute to RTC and IWDG registers incorrectly managed errata. • Updated Data read from flash memory corrupted when PA12 used as GPIO or alternate function erratum title. FMC: • Updated Missing information on prohibited 0xFF value of NAND transaction wait timing erratum. IWDG: added RVU flag not cleared at low APB clock frequency and PVU flag not cleared at low APB clock frequency errata. RTC: added RTC interrupt can be masked by another RTC interrupt, Calendar initialization may fail in case of consecutive INIT mode entry and Alarm flag may be repeatedly set when the core is stopped in debug errata. USART: • Changed nCTS and nRTS to CTS and RTS respectively. • Added Start bit detected too soon when sampling for NACK signal from the smartcard, Break request can prevent the transmission complete flag (TC) from being set , Guard time not respected when data are sent on TXE events, and RTS is active while RE or UE = 0 errata. SPI/I2S: • Remove duplicate erratum "Corrupted last bit of data and/or CRC, received in Master mode with delayed SCK feedback". • Added BSY bit may stay high when SPI is disabled, and Anticipated communication upon SPI transit from slave receiver to master errata. SDIO: updated Wrong data written during SDIO hardware flow control erratum title. Added Section Important security notice..
13-Feb-2024	20	Updated errata: PC13 signal transitions disturb LSE
04-Apr-2024	21	Errata added: Corrupted last bit of data and/or CRC received in Master mode with delayed SCK feedback
27-Jan-2025	22	Master and slave terms in I2C replaced with controller and target, respectively. Added errata: Corrupted content of the backup domain due to a missed power-on reset after this domain supply voltage drop
25-Feb-2025	23	Corrupted content of the backup domain due to a missed power-on reset after this domain supply voltage drop updated.

Contents