STM32WL55xx, STM32WL54xx device errata

Applicability

This document applies to the part numbers of STM32WL55xx, STM32WL54xx 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 RM0453.

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
STM32WL55xx	STM32WL55CC, STM32WL55JC, STM32WL55UC
STM32WL54xx	STM32WL54CC, STM32WL54JC, STM32WL54UC

Table 2. Device variants

Reference	Silicon revision codes	
	Device marking(1)	REV_ID(2)
STM32WL55xx, STM32WL54xx	Z	0x1001
	Y	0x1003

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_IDCODER register.*

1 Summary of device errata

The following table gives a quick reference to the STM32WL55xx, STM32WL54xx 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. Z	Rev. Y
M4 FPU core	2.1.1	Interrupted loads to SP can cause erroneous behavior	A	A
	2.1.2	Store immediate overlapping exception return operation might vector to incorrect interrupt	A	A
System	2.2.1	Wrong DMAMUX synchronization and trigger input connections to EXTI	A	-
	2.2.2	Perpetual CPU2 boot upon illegal access	A	-
	2.2.3	Overwriting with all zeros a flash memory location previously programmed with all ones fails	N	N
	2.2.4	Option byte loading failure at high MSI system clock frequency	A	-
	2.2.5	A system reset occurs when nRST_SHDW is set and nRST_STDBY is cleared and Shutdown mode is entered	A	-
	2.2.6	FLASH_ECCR corrupted upon reset or power-down occurring during flash memory program or erase operation	A	A
	2.2.7	Voltage drop on the 1.2 V regulated supply when switching MSI to 48 MHz	A	A
	2.2.8	Sensitivity affected by HSE activation in high bandwidth channel	A	-
	2.2.9	Sensitivity degradation in LNA boosted mode	A	A
	2.2.10	SysTick trigger in debug emulation generates HardFault	A	A
	2.2.11	Debug HALT command in debug emulation generates HardFault	A	-
	2.2.12	Potential deadlock condition on wakeup from some low-power modes	A	-
	2.2.13	JTAG cannot be used without the JTAG NRST pin	N	-
	2.2.14	Flash PCROP is not operating properly	N	-
	2.2.15	Unexpected C2DS flag starting from 2nd Standby wakeup	N	-
	2.2.16	CPU2 boot on system reset after illegal access detection	A	-
	2.2.17	Peripherals freeze on debug HALT command while CPU1 is in Stop mode and CPU2 is in Run mode	A	-
	2.2.18	TX spurs around carrier at a multiple of HSE clock frequency	A	A
	2.2.19	Sensitivity degradation of modulation with 500 kHz LoRa® bandwidth	A	A
	2.2.20	Over-protective resistance of Tx-to-antenna matching	A	A
	2.2.21	Unexpected timeout behavior in implicit header mode	A	A
	2.2.22	Packet loss with inverted IQ operation	A	A
	2.2.23	RX duty-cycle issue (set_RxDutyCycle command)	A	A

Function	Section	Limitation	Status	
			Rev. Z	Rev. Y
System	2.2.24	LSE crystal oscillator may be disturbed by transitions on PC13	N	N
	2.2.25	Potential isolation issue between CPU1 and CPU2	P	P
ADC	2.3.1	Overrun flag is not set if EOC reset coincides with new conversion end	P	P
	2.3.2	Writing ADC_CFGR1 register while ADEN bit is set resets RES[1:0] bitfield	A	A
	2.3.3	Out-of-threshold value is not detected in AWD1 Single mode	A	A
	2.3.4	Writing ADC_CFGR2 register while ADEN bit is set resets CKMODE[1:0] bitfield	A	A
TIM	2.5.1	One-pulse mode trigger not detected in master-slave reset + trigger configuration	P	P
	2.5.2	Consecutive compare event missed in specific conditions	N	N
	2.5.3	Output compare clear not working with external counter reset	P	P
LPTIM	2.6.1	Device may remain stuck in LPTIM interrupt when entering Stop mode	A	A
	2.6.2	ARRM and CMPM flags are not set when APB clock is slower than kernel clock	A	A
	2.6.3	Device may remain stuck in LPTIM interrupt when clearing event flag	A	A
RTC and TAMP	2.7.1	Alarm flag may be repeatedly set when the core is stopped in debug	N	N
	2.7.2	A tamper event fails to trigger timestamp or timestamp overflow events during a few cycles after clearing TSF	N	N
	2.7.3	REFCKON write protection associated to INIT KEY instead of CAL KEY	A	A
	2.7.4	Tamper flag not set on LSE failure detection	N	N
	2.7.5	Binary mode: SSR is not reloaded with 0xFFFF FFFF when SSCLR = 1	A	A
I2C	2.8.1	Wrong data sampling when data setup time ($t_{SU;DAT}$) is shorter than one I2C kernel clock period	P	P
	2.8.2	Spurious bus error detection in controller mode	A	A
	2.8.3	OVR flag not set in underrun condition	N	N
	2.8.4	Transmission stalled after first byte transfer	A	A
	2.8.5	SDA held low upon SMBus timeout expiry in target mode	A	A
USART	2.9.1	Anticipated end-of-transmission signaling in SPI slave mode	A	A
	2.9.2	Data corruption due to noisy receive line	A	A
LPUART	2.10.1	Possible LPUART transmitter issue when using low BRR[15:0] value	P	P

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

Table 4. Summary of device documentation errata

Function	Section	Documentation erratum
DAC	2.4.1	CAL_FLAGx can remain cleared during a user trimming calibration

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 M4 FPU core

Reference manual and errata notice for the Arm® Cortex®-M4 FPU core revision r0p1 is available from http://infocenter.arm.com.

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 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 Wrong DMAMUX synchronization and trigger input connections to EXTI

Description

By error, synchronization and trigger inputs of the DMAMUX peripheral are connected to interrupt output lines of the EXTI block, instead of being connected to its SYSCFG multiplexer output lines.

The EXTI interrupt lines exhibit a rising-edge transition upon each active transition (rising, falling or both) of corresponding GPIOs, as defined in the EXTI_RTSRx and EXTI_FTSRx registers.

As a consequence, the falling active edge option of the DMAMUX synchronization and trigger inputs is unusable because falling edges on these inputs do not occur upon GPIO events but upon clearing the EXTI interrupt pending flags (by setting the corresponding PIF bits of the EXTI_PRx register).

Workaround

For the DMAMUX synchronization and trigger events to occur upon determined rising or/and falling edge of the corresponding GPIOs:

- Set the desired active edge polarities of the corresponding GPIOs through the EXTI_RTSRx and EXTI_FTSRx registers.
- Set the active edge polarity to rising for all corresponding DMAMUX input lines, through the SPOL bits of the DMAMUX_CxCR register (for synchronization inputs) and the GPOL bits of the DMAMUX_RGxCR register (for trigger inputs).
- Ensure that EXTI interrupt pending flags corresponding to the GPIOs used for DMAMUX inputs are cleared in the EXTI interrupt service routine.

Note: This can be ensured if using the `HAL_GPIO_IrqHandler` function provided by STMicroelectronics.

2.2.2 Perpetual CPU2 boot upon illegal access

Description

After handling an illegal access, the CPU2 re-boots upon each low-power mode entry as long as the C2BOOT bit remains set, whereas it should only re-boot upon the first low-power mode entry.

Workaround

In the ILAC handler of CPU2, execute the following sequence:

```
IWDG->KR = 0x0000CCCCu;
IWDG->RLR = 0x01u;
PWR->CR4 &= ~PWR_CR4_C2BOOT;
SET_BIT(SCB->SCR, ((uint32_t)SCB_SCR_SLEEPDEEP_Msk));
__WFI();
```

2.2.3 Overwriting with all zeros a flash memory location previously programmed with all ones fails

Description

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

Note: Flash memory locations in the erased state (that is, not programmed) are not affected by this failure. They can be programmed with any value.

Workaround

None.

2.2.4 Option byte loading failure at high MSI system clock frequency

Description

The option bytes are not loaded correctly upon setting the OBL_LAUNCH bit of the FLASH_CR register when the frequency of MSI oscillator used as system clock source is higher than 16 MHz.

Workaround

Before loading the option bytes by setting the OBL_LAUNCH bit of the FLASH_CR register, either change the system clock source to other than MSI oscillator, or set the MSI clock frequency to less than 16 MHz.

2.2.5 A system reset occurs when nRST_SHDW is set and nRST_STDBY is cleared and Shutdown mode is entered

Description

When the following configuration is selected:

- nRST_SHDW is set
- nRST_STDBY is cleared,

a system reset is generated when Shutdown mode is entered.

Workaround

The only valid configuration to avoid a reset after entering into Shutdown mode is the following:

- nRST_SHDW is set
- nRST_STDBY is set.

2.2.6 FLASH_ECCR corrupted upon reset or power-down occurring during flash memory program or erase operation

Description

Reset or power-down occurring during a flash memory location program or erase operation, followed by a read of the same memory location, may lead to a corruption of the FLASH_ECCR register content.

Workaround

Under such condition, erase the page(s) corresponding to the flash memory location.

2.2.7 Voltage drop on the 1.2 V regulated supply when switching MSI to 48 MHz

Description

A voltage drop to 1.08 V may occur on the 1.2 V regulated supply when the MSI frequency is changed as follows:

- from MSI at 400 kHz to MSI at 24 MHz and above
- from MSI at 1 MHZ to MSI at 48 MHz

As a result, the voltage drop may cause CPU HardFault.

Workaround

To ensure there is no impact on the 1.2 V supply, introduce an intermediate MSI frequency change step as follows:

- Change the MSI frequency range from 400 kHz to 12 MHz, then to 24 MHz and above, as illustrated in these examples:
  - MSI@400 kHz → MSI@12 MHz → MSI@24 MHz
  - MSI@400 kHz → MSI@12 MHz → MSI@48 MHz.
- Change the MSI frequency range from 1 MHz to 12 MHz, then to 48 MHz, as illustrated in this example: MSI@400 kHz → MSI@12 MHz → MSI@48 MHz.

2.2.8 Sensitivity affected by HSE activation in high bandwidth channel

Description

The sensitivity of the high bandwidth channels (HB) EU864 and US928 is affected when HSE is used as system clock. HSE cannot be used as system clock when channels EU864 or US928 are used. Except if the LoRa® modulation is needed, the user can also replace 32 MHz HSE clock by 31.25 MHz HSE clock.

Workaround

Use a different clock source for the system, for example MSI or HSI with or without PLL.

2.2.9 Sensitivity degradation in LNA boosted mode

Description

Reduced performance is observed in some specific LNA boost mode use cases. There is a potential loss of sensitivity of approximately up to 1.5 dB on rev Z and 0.75 dB in rev Y, versus specification for LoRa® and GFSK modulation at 915 MHz.

Workaround

None.

2.2.10 SysTick trigger in debug emulation generates HardFault

Description

When the CPU enters its deepsleep state and debug emulation is active, the CPU SysTick is still running and able to trigger interrupts. The CPU is woken up and fetches code from the system Flash memory. As the Flash memory is not active, the CPU starts fetching code from a resource that is not available, and receives random data from the bus, causing HardFault.

When the debug emulation is active and the CPU enters its deepsleep state, the system exhibits unpredictable behavior due to the SysTick activity.

Workaround

Always disable SysTick before the CPU enters its deepsleep state. This consistently prevents any interrupt triggering and any possibility of HardFault.

2.2.11 Debug HALT command in debug emulation generates HardFault

Description

When the CPU is in deepsleep state with active debug emulation and a debug HALT/RUN command sequence is submitted, the CPU wakes up and starts fetching data from the flash memory. As the flash memory is unavailable, the CPU receives random data from the bus which causes HardFault.

Debug cannot be performed on the system when the CPU is in deep-sleep state and debug emulation is active.

Workaround

Change the debug sequence to perform a detach/attach procedure before any "HALT/RUN". Also enable the CDBGPWRUPREQ wake-up on the EXTI before the CPU goes in deep-sleep state. Set a breakpoint only when in Run or Sleep mode.

2.2.12 Potential deadlock condition on wakeup from some low-power modes

Description

Following multiple wakeups from Stop 1, Stop 2 or Standby, the power supply management block may not restart properly. This occurs on a specific timing of the wakeup event and Run duration.

The issue may occur in the event of an asynchronous wakeup when the MCU remains in Run for less than 60 µs after a wakeup.

When this issue happens, the power consumption in Run mode does not correspond to the datasheet. The global power consumption is not impacted.

Workaround

If it is key to control accurately the power consumption, use one of the following measures:

- General workaround (valid also if SMPS is used before entering wakeup): wait for the LDO to reach a stable condition before entering a low-power mode.
  A software implementation solution is to check that the LDORDY bit of the PWR_SR2 register is set before entering a low-power mode.

- LDO use case only (SMPS was not used before entering wakeup): the deadlock condition, which can potentially occur at wakeup from a low-power mode, can be solved by generating a pulse on the SMPSEN bit of the PWR_CR5 register controlling the SMPS activation.
  A software implementation solution is to toggle the SMPSEN bit of the PWR_CR5 register at each wakeup from low-power mode.

2.2.13 JTAG cannot be used without the JTAG NRST pin

Description

When CPU1 is in deepsleep state with active debug emulation and CPU2 is stopped as well, and a *debug HALT - debug RUN* command sequence is submitted, CPU1 wakes up and starts to fetch from Flash memory. In this condition, the Flash memory is off, CPU1 retrieves random information from the bus which causes a HardFault.

By default after reset, PB4 is configured as alternate function for JTAG NRESET function. If this pin is reused in another configuration (GPIO, analog or alternate function for another function), the product JTAG NRESET internal signal is forced to 0.

Workaround

None.

2.2.14 Flash PCROP is not operating properly

Description

The PCROP protection on the Flash memory area does not work as expected.

Workaround

None.

2.2.15 Unexpected C2DS flag starting from 2nd Standby wakeup

Description

If the system enters Standby while CPU2 is not booted (C2BOOT is cleared) and a wakeup request directed to CPU2 is received and then redirected to CPU1, the C2DS flag indicating the CPU2 deepsleep state behaves as follow:

- On the first wakeup from Standby, C2DS is set as expected.
- Starting from second wakeup from Standby, C2DS is cleared, while it should be set.

Workaround

None.

2.2.16 CPU2 boot on system reset after illegal access detection

Description

After an illegal access is detected with CPU2 active where C2BOOT is set and treated through ILAC handling procedure, on next system reset, CPU2 boots without any trigger generated by CPU1.

The process can be modelled as follows:

1. CPU1 runs through the boot sequence and boots CPU2 normally through C2BOOT.
2. An illegal access is generated from either: CPU1 or CPU2, is normally detected by CPU2.
3. At next system reset, CPU2 boots immediately.

Workaround

- Perform a power-on reset to restart from a reliable state.
- Handle the unexpected CPU2 boot by software.

2.2.17 Peripherals freeze on debug HALT command while CPU1 is in Stop mode and CPU2 is in Run mode

Description

When CPU1 is in Stop mode, CPU2 is in Run mode, debug emulation is active, and a *debug HALT - debug RUN* command sequence is submitted, CPU1 wakes up and starts, but peripherals exclusively associated to CPU1 are not clocked by RCC.

No debug can be performed on the system when CPU1 is in Stop mode, CPU2 is in Run mode and debug emulation is active.

Workaround

Change the debug sequence to perform a detach/attach operation before the HALT/RUN and set the CDBGPWRUPREQ wakeup on the AIEC before CPU1 goes in deepsleep state.

2.2.18 TX spurs around carrier at a multiple of HSE clock frequency

Description

Unwanted spurs may occur around the transmission carrier with frequency that is an integer multiple of the HSE clock frequency. For example, around 480 MHz or 864 MHz if the HSE clock frequency is 32 MHz.

Workaround

Apply one of the following measures:

- Choose the HSE clock frequency so that its integer multiples are at least 1.5 MHz away from the desired transmission carrier frequency.
- Do not use transmission channels within the range of ±1.5 MHz around carrier frequencies that are integer multiples of the HSE clock frequency.

Note: The first measure cannot be applied if the application requires LoRa® modulation that imposes the use of 32 MHz HSE clock.

2.2.19 Sensitivity degradation of modulation with 500 kHz LoRa® bandwidth

Description

A sensitivity degradation may be observed when receiving signals transmitted with a 500 kHz LoRa® bandwidth.

Workaround

Set bit #2 as follows at 0x0889 address before any packet transmission:

- Set it to 0 if the LoRa® bandwidth is 500 kHz.
- Set it to 1 for other LoRa® bandwidths and for any (G)FSK configuration.

Use the code below before each packet transmission to properly configure the device:

```
if (packetType -- LORA) {
    if (LoraBandwidtch == BW500) {
        value = ReadRegister@0x0889;
        value = value & 0xFB;
        WriteRegister(value) @0x00889l;
    }
]
else (
    value = ReadRegister@0x00889;
    value = value | 0x04;
    WriteRegister(value) @0x0899;
}
```

2.2.20 Over-protective resistance of Tx-to-antenna matching

Description

The device embeds a PA (power amplifier) clamping mechanism, backing down the power when over-voltage conditions are detected internally. Considering a high-power operation of the device (supporting + 22 dBm on-chip), the clamping components are overly protective, and cause the device to back down its power when a reasonable mismatch is detected at PA output. The observation is typically 5 to 6 dB less output power than expected.

Workaround

In high-power configuration, after a power-on reset or a wakeup from cold start, set bits 1 to 4 of TxClampConfig register to 0b1111 to optimize the PA clamping threshold.

Use the code below:

```
value = ReadRegister@0x08D8;
value = value | 0x1E
WriteRegister(value)@0x08D8;
```

Note: This workaround improves the device functionality. The long-term reliability is guaranteed with or without it.

2.2.21 Unexpected timeout behavior in implicit header mode

Description

When receiving LoRa® packets in Rx mode with timeout active and no header (Implicit mode), the timer that generates the timeout is not stopped on a RxDone event. This may trigger an unexpected timeout in any subsequent mode in which the timer is not re-invoked, reset, and re-programmed.

Workaround

Add the code below after any Rx with timeout active sequence to stop the timer and clear the potential timeout event:

```
WriteRegister(0x00)@0x0920;
value = ReadRegister@0x0944;
value = value | 0x02;
WriteRegister(value)@0x0944;
```

The register at 0x0920 is used to stop the counter. The register at 0x0944 clears the potential event.

2.2.22 Packet loss with inverted IQ operation

Description

Some long packets may be lost when exchanging LoRa® packets with inverted IQ polarity.

Workaround

Set bit 2 at address 0x0736 to 0 when using the inverted IQ polarity (see `SetPacketParam` command). Set this bit to 1 when using the standard IQ polarity.

Use the code below:

```
value = ReadRegister@0x0736;
value = value | 0x04;
WriteRegister(value)@0x0736;
```

2.2.23 RX duty-cycle issue (set_RxDutyCycle command)

Description

When a preamble is detected, the timeout is stopped, and must be restarted with a value equal to $2 \times RxPeriod + SleepPeriod$.

In fact, this value is set to SleepPeriod.

Workaround

When a preamble is detected, reprogram the radio timeout timer to $2 \times RxPeriod + SleepPeriod$.

With RxPeriod and SleepPeriod expressed in 15.625 µs step, use the code below into the PreambleDetected interrupt handler:

```
/* Update Radio RTC period */
                Radio.Write(SUBGHZ_RTCPRDR2, ((2 × RxPeriod + SleepPeriod )>>16)&0xFF); /*Upd
ate Radio RTC Period MSB*/
                Radio.Write(SUBGHZ_RTCPRDR1, ((2 × RxPeriod + SleepPeriod )t>>8)&0xFF); /*Upd
ate Radio RTC Period MidByte*/
                Radio.Write(SUBGHZ_RTCPRDR0, ((2 × RxPeriod + SleepPeriod ))&0xFF);
/*Update Radio RTC Period lsb*/
                Radio.Write(SUBGHZ_RTCCTLR, Radio.Read(SUBGHZ_RTCCTLR)|0x1);
/*restart Radio RTC*/
                SubgRf.RxDcPreambleDetectTimeout = 0;
                /*Clear IRQ_PREAMBLE_DETECTED mask*/
```

2.2.24 LSE crystal oscillator may be disturbed by transitions on PC13

Description

On UFQFPN packages, the LSE crystal oscillator clock frequency can be incorrect when PC13 is toggling in input or output (for example when used for RTC_OUT1).

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

The UFBGA packages are not impacted by this limitation.

Workaround

None.

Avoid toggling PC13 when LSE is used on UFQFPN packages.

2.2.25 Potential isolation issue between CPU1 and CPU2

Description

Refer to security advisory SA0024 on www.st.com.

Workaround

Refer to security advisory SA0024 on www.st.com.

2.3 ADC

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

Description

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

Workaround

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

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

Description

Modifying the ADC_CFGR1 register while ADC is enabled (ADEN set in ADC_CR) resets RES[1:0] to 00 whatever the bitfield previous value.

Workaround

Apply the following sequence:

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

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

Description

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

Workaround

Apply one of the following measures:

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

2.3.4 Writing ADC_CFGR2 register while ADEN bit is set resets CKMODE[1:0] bitfield

Description

Modifying the ADC_CFGR2 register while ADC is enabled (ADEN set in ADC_CR) resets CKMODE[1:0] to 00 whatever the bitfield previous value.

Workaround

Apply the following sequence:

1. Set ADDIS to disable the ADC, and wait until ADEN is cleared.
2. Program the ADC_CFGR2 register according to the application requirements.
3. Set ADEN bit.

2.4 DAC

2.4.1 CAL_FLAGx can remain cleared during a user trimming calibration

Description

The reference manual describes the sequence to be followed to perform a user trimming calibration on a given channel. Some reference manual versions only specify the steps to be followed if the corresponding CAL_FLAGx bit of the DAC_SR register is set.

However, the CAL_FLAGx bit can remain cleared for some products.

Workaround

No application workaround is required, provided the OTRIMx bitfield of the DAC_CCR register is programmed to 0x1F when the CAL_FLAGx bit has remained cleared.

2.5 TIM

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

Description

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

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

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

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

Workaround

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

2.5.2 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.

2.5.3 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).

2.6 LPTIM

2.6.1 Device may remain stuck in LPTIM interrupt when entering Stop mode

Description

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

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

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

Workaround

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

2.6.2 ARRM and CMPM flags are not set when APB clock is slower than kernel clock

Description

When LPTIM is configured in one shot mode and APB clock is lower than kernel clock, there is a chance that ARRM and CMPM flags are not set at the end of the counting cycle defined by the repetition value REP[7:0]. This issue can only occur when the repetition counter is configured with an odd repetition value.

Workaround

To avoid this issue the following formula must be respected:

{ARR, CMP} ≥ KER_CLK / (2* APB_CLK),

where APB_CLK is the LPTIM APB clock frequency, and KER_CLK is the LPTIM kernel clock frequency. ARR and CMP are expressed in decimal value.

Example: The following example illustrates a configuration where the issue can occur:

- APB clock source (MSI) = 1 MHz , Kernel clock source (HSI) = 16 MHz
- Repetition counter is set with REP[7:0] = 0x3 (odd value)

The above example is subject to issue, unless the user respects:

{CMP, ARR} ≥ 16 MHz / (2 * 1 MHz)

→ ARR must be ≥ 8 and CMP must be ≥ 8

Note: REP set to 0x3 means that effective repetition is REP+1 (= 4) but the user must consider the parity of the value loaded in LPTIM_RCR register (=3, odd) to assess the risk of issue.

2.6.3 Device may remain stuck in LPTIM interrupt when clearing event flag

Description

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

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

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

Workaround

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

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

Note: The standard clear sequence implemented in the HAL_LPTIM_IRQHandler in the STM32Cube is considered as the proper clear sequence.

2.7 RTC and TAMP

2.7.1 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.

2.7.2 A tamper event fails to trigger timestamp or timestamp overflow events during a few cycles after clearing TSF

Description

With the timestamp on tamper event enabled (TAMPTS bit of the RTC_CR register set), a tamper event is ignored if it occurs:

- within four APB clock cycles after setting the CTSF bit of the RTC_SCR register to clear the TSF flag, while the TSF flag is not yet effectively cleared (it fails to set the TSOVF flag)
- within two ck_apre cycles after setting the CTSF bit of the RTC_SCR register to clear the TSF flag, when the TSF flag is effectively cleared (it fails to set the TSF flag and timestamp the calendar registers)

Workaround

None.

2.7.3 REFCKON write protection associated to INIT KEY instead of CAL KEY

Description

The write protection of the REFCKON bit is unlocked if the key sequence is written in RTC_WPR with the privilege and security rights set by the INITPRIV and INITSEC bits, instead of being set by the CALPRIV and CALSEC bits.

Workaround

Unlock the INIT KEY before writing REFCKON.

2.7.4 Tamper flag not set on LSE failure detection

Description

With the timestamp on tamper event enabled (the TAMPTS bit of the RTC_CR register set), the LSE failure detection (LSE clock stopped) event connected to the internal tamper 3 fails to raise the ITAMP3F and ITAMP3MF flags, although it duly erases or blocks (depending on the internal tamper 3 configuration) the backup registers and other device secrets, and the RTC and TAMP peripherals resume normally upon the LSE restart.

Note: As expected in this particular case, the TSF and TSMF flags remain low as long as LSE is stopped as they require running RTCCLK clock to operate.

Workaround

None.

2.7.5 Binary mode: SSR is not reloaded with 0xFFFF FFFF when SSCLR = 1

Description

When SSCLR bit of the RTC_ALRMxSSR register is set when in binary mode, SSR is reloaded with 0xFFFF FFFF at the end of the ck_apre cycle when RTC_SSR is set to RTC_ALRxBINR (*x* stands for either A or B)

RTC_SSR is not reloaded with 0xFFFF FFFF if RTC_ALRxBINR is modified while RTC_SSR is set to RTC_ALRxBINR. Rather, SSR continues to decrement.

Workaround

The workarounds are described for alarm A, and can be applied in the same manner for alarm B. Two workarounds are proposed, the second one requires to use the second alarm.

- Wait for one ck_apre cycle after an alarm A event before changing the RTC_ALRABINR register value.
- Do not reprogram RTC_ALRABINR following the alarm A event itself. Instead, use alarm B configured with RTC_ALRBBINR set to 0xFFFF FFFF, and reprogram RTC_ALRABINR after the alarm B event. This ensures that one ck_apre cycle elapses following the alarm A event.

2.8 I2C

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

Description

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

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

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

Workaround

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

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

2.8.2 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.

2.8.3 OVR flag not set in underrun condition

Description

In target transmission with clock stretching disabled (NOSTRETCH = 1 in the I2C_CR1 register), an underrun condition occurs if the current byte transmission is completed on the I^2C bus, and the next data is not yet written in the TXDATA[7:0] bitfield. In this condition, the device is expected to set the OVR flag of the I2C_ISR register and send 0xFF on the bus.

However, if the I2C_TXDR is written within the interval between two I2C kernel clock cycles before and three APB clock cycles after the start of the next data transmission, the OVR flag is not set, although the transmitted value is 0xFF.

Workaround

None.

2.8.4 Transmission stalled after first byte transfer

Description

When the first byte to transmit is not prepared in the TXDATA register, two bytes are required successively, through TXIS status flag setting or through a DMA request. If the first of the two bytes is written in the I2C_TXDR register in less than two I2C kernel clock cycles after the TXIS/DMA request, and the ratio between APB clock and I2C kernel clock frequencies is between 1.5 and 3, the second byte written in the I2C_TXDR is not internally detected. This causes a state in which the I2C peripheral is stalled in controller mode or in target mode, with clock stretching enabled (NOSTRETCH = 0). This state can only be released by disabling the peripheral (PE = 0) or by resetting it.

Workaround

Apply one of the following measures:

- Write the first data in I2C_TXDR before the transmission starts.
- Set the APB clock frequency so that its ratio with respect to the I2C kernel clock frequency is lower than 1.5 or higher than 3.

2.8.5 SDA held low upon SMBus timeout expiry in target mode

Description

For the target mode, the SMBus specification defines $t_{TIMEOUT}$ (detect clock low timeout) and $t_{LOW:SEXT}$ (cumulative clock low extend time) timeouts. When one of them expires while the I2C peripheral in target mode drives SDA low to acknowledge either its address or a data transmitted by the controller, the device is expected to report such an expiry and release the SDA line.

However, although the device duly reports the timeout expiry, it fails to release SDA. This stalls the I^2C bus and prevents the controller from generating RESTART or STOP condition.

Workaround

When a timeout is reported in target mode (TIMEOUT bit of the I2C_ISR register is set), apply this sequence:

1. Wait until the frame is expected to end.
2. Read the STOPF bit of the I2C_ISR register. If it is low, reset the I2C kernel by clearing the PE bit of the I2C_CR1 register.
3. Wait for at least three APB clock cycles before enabling again the I2C peripheral.

2.9 USART

2.9.1 Anticipated end-of-transmission signaling in SPI slave mode

Description

In SPI slave mode, at low USART baud rate with respect to the USART kernel and APB clock frequencies, the *transmission complete* flag TC of the USARTx_ISR register may unduly be set before the last bit is shifted on the transmit line.

This leads to data corruption if, based on this anticipated end-of-transmission signaling, the application disables the peripheral before the last bit is transmitted.

Workaround

Upon the TC flag rise, wait until the clock line remains idle for more than the half of the communication clock cycle. Then only consider the transmission as ended.

2.9.2 Data corruption due to noisy receive line

Description

In all modes, except synchronous slave mode, the received data may be corrupted if a glitch to zero shorter than the half-bit occurs on the receive line within the second half of the stop bit.

Workaround

Apply one of the following measures:

- Either use a noiseless receive line, or
- add a filter to remove the glitches if the receive line is noisy.

2.10 LPUART

2.10.1 Possible LPUART transmitter issue when using low BRR[15:0] value

Description

The LPUART transmitter bit length sequence is not reset between consecutive bytes, which could result in a jitter that cannot be handled by the receiver device. As a result, depending on the receiver device bit sampling sequence, a desynchronization between the LPUART transmitter and the receiver device may occur resulting in data corruption on the receiver side.

This happens when the ratio between the LPUART kernel clock and the baud rate programmed in the LPUART_BRR register (BRR[15:0]) is not an integer, and is in the three to four range. A typical example is when the 32.768 kHz clock is used as kernel clock and the baud rate is equal to 9600 baud, resulting in a ratio of 3.41.

Workaround

Apply one of the following measures:

- On the transmitter side, increase the ratio between the LPUART kernel clock and the baud rate. To do so:
  - Increase the LPUART kernel clock frequency, or
  - Decrease the baud rate.
- On the receiver side, generate the baud rate by using a higher frequency and applying oversampling techniques if supported.

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 5. **Document revision history**

Date	Version	Changes
04-Nov-2020	1	Initial release.
21-Apr-2021	2	Added: • TX spurs around carrier at a multiple of HSE clock frequency • Binary mode: SSR is not reloaded with 0xFFFF FFFF when SSCLR = 1 • die revision Y Updated: • Sensitivity affected by HSE activation in high bandwidth channel • Sensitivity degradation in LNA boosted mode • Potential deadlock condition on wakeup from some low-power modes Removed *VDIV or VSQRT instructions might not complete correctly when very short ISRs are used* in Core 1.
30-Jul-2021	3	Updated Potential deadlock condition on wakeup from some low-power modes
14-Mar-2022	4	Added: • Sensitivity degradation of modulation with 500 kHz LoRa® bandwidth • Over-protective resistance of Tx-to-antenna matching • Unexpected timeout behavior in implicit header mode • Packet loss with inverted IQ operation • RX duty-cycle issue (set_RxDutyCycle command) • Overrun flag is not set if EOC reset coincides with new conversion end • Writing ADC_CFGR1 register while ADEN bit is set resets RES[1:0] bitfield • Out-of-threshold value is not detected in AWD1 Single mode • Writing ADC_CFGR2 register while ADEN bit is set resets CKMODE[1:0] bitfield Updated: • REV_ID in Table 2. Device variants • Note in Device may remain stuck in LPTIM interrupt when entering Stop mode
6-Feb-2023	5	Updated Overwriting with all zeros a flash memory location previously programmed with all ones fails Added SDA held low upon SMBus timeout expiry in slave mode and Possible LPUART transmitter issue when using low BRR[15:0] value. Removed erratum *DMA stream locked when transferring data to/from USART*. Added Section Important security notice at the end of the document.
24-Jan-2025	6	Added: • LSE crystal oscillator may be disturbed by transitions on PC13 • Potential isolation issue between CPU1 and CPU2 • CAL_FLAGx can remain cleared during a user trimming calibration • Possible LPUART transmitter issue when using low BRR[15:0] value • Debug HALT command in debug emulation generates HardFault Removed: Wrong data received when the communication nodes are two LPUART instances

Contents