
How to enter low-power mode through WFI instruction

Questions:

Why does application fail to enter low-power modes through `__WFI()` instruction?

It proceeds as below: When the `__WFI()` instruction is used to enter low-power modes, the system skips this instruction in the absence of a wakeup condition and causes the failure of low-power mode entry.

Answer:

Standby low-power mode is taken as an example in this document.

【Root cause】:

“`__WFI()`” instruction refers to “Wait For Interrupt”. Low-power modes are entered by executing this instruction. But if there is an interrupt pending in the NVIC, the application code will skip “`__WFI()`” instruction, causing the application to fail to enter low-power mode.

This is an inherent characteristics of ARM core. Thus this phenomenon happens in all ARM-based core MCUs.

【Solution】 :

Before executing “`__WFI()`” instruction, clear all NVIC pending bits in the NVIC interrupts.

Analysis:

Here is an application case showing why Standby low-power mode is not entered when USART1 receive interrupt is enabled.

【Erroneous code example】 (The irrelevant code is not posted below)

```
crm_periph_clock_enable(CRM_PWC_PERIPH_CLOCK, TRUE); ///① Enable PWR clock
nvic_irq_enable(USART1_IRQn, 0, 0);      ///② Enable USART1 NVIC interrupt
usart_interrupt_enable(USART1, USART_RDBF_INT, TRUE); ///③ Enable USART1 receive interrupt
__disable_irq();      ///④ Disable all NVIC interrupt requests
while(usart_flag_get(USART1, USART_RDBF_FLAG) == RESET); ///⑤ Wait RDBF flag to be set
pwc_standby_mode_enter(); ///⑥ Standby mode entry instruction
while(1);
```

【Analysis process】:

When a data is received,

- 1) RDBF flag of USART1 is set;
- 2) Because of ③, the NVIC pending bit corresponding to USART1 is also set with RDBF bit
- 3) Because of ②, the NVIC pending bit jumps to the corresponding interrupt function;
- 4) Because of ④, the application code does not jump to the interrupt function actually, causing that NVIC pending bit remain active;
- 5) Since the NVIC pending bit remains active, the system directly skips “`__WFI()`” instruction ⑥ and continues subsequent operation until the PC stops “`while(1)`” at the end of the code.

【How to solve this issue?】:

To enter Standby mode, the application has to clear all NVIC pending bits before executing “__WFI()” instruction, in other words, add the following codes between ⑤ and ⑥ instructions.

```
usart_flag_clear(USART1, USART_RDBF_FLAG); ///  
NVIC_ClearPendingIRQ(USART1_IRQn); ///  

```

【Cautions】 :

- A. This issue only happens in the case of entering low-power mode through “__WFI()” instruction. WFE instruction has no similar issue.
- B. Even if the NVIC interrupts of peripherals were not enabled, the NVIC pending bits would still be set. But the NVIC pending bits does not affect the application.
- C. In the above solution, it is important to clear interrupt flags of peripherals before clearing NVIC pending bits. As stated in above example, the NVIC pending bit of USART1 is set with the RDBF bit, and thus it is impossible to clear the NVIC pending bits if RDBF is not cleared in advance.
- D. For non-low-power applications, this issue should also be taken into account, as the NVIC pending bit (remains active) would cause the interrupt functions to be executed twice.
- E. The IAP with instruction jump or other related applications are susceptible to this issue. Thus special attention should be paid to the status of the NVIC pending bits of peripherals during code design.

Type: MCU applications

Applicable products: AT32 series

Main function: NVIC pending bit clear, low-power mode entry through __WFI() instruction

Minor function: None

Document revision history

Date	Revision	Changes
2022.2.28	2.0.0	Initial release

IMPORTANT NOTICE – PLEASE READ CAREFULLY

Purchasers are solely responsible for the selection and use of ARTERY's products and services, and ARTERY assumes no liability whatsoever relating to the choice, selection or use of the ARTERY products and services described herein.

No license, express or implied, to any intellectual property rights is granted under this document. If any part of this document deals with any third party products or services, it shall not be deemed a license grant by ARTERY for the use of such third party products or services, or any intellectual property contained therein, or considered as a warranty regarding the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

Unless otherwise specified in ARTERY's terms and conditions of sale, ARTERY provides no warranties, express or implied, regarding the use and/or sale of ARTERY products, including but not limited to any implied warranties of merchantability, fitness for a particular purpose (and their equivalents under the laws of any jurisdiction), or infringement of any patent, copyright or other intellectual property right

Purchasers hereby agrees that ARTERY's products are not designed or authorized for use in: (A) any application with special requirements of safety such as life support and active implantable device, or system with functional safety requirements; (B) any air craft application; (C) any automotive application or environment; (D) any space application or environment, and/or (E) any weapon application. Purchasers' unauthorized use of them in the aforementioned applications, even if with a written notice, is solely at purchasers' risk, and is solely responsible for meeting all legal and regulatory requirement in such use.

Resale of ARTERY products with provisions different from the statements and/or technical features stated in this document shall immediately void any warranty grant by ARTERY for ARTERY products or services described herein and shall not create or expand in any manner whatsoever, any liability of ARTERY

© 2022 Artery Technology -All rights reserved