FAQ0133

Frequently Asked Questions

Peripheral event interrupt handing process

Questions:

The application usually uses peripheral interrupts to respond to callback tasks corresponding to event flags in the interrupt function, in order to achieve fast task handling. However, most peripherals feature multiple event flags that can trigger interrupts. It is therefore important for users to know how to accurately locate and respond to callback tasks. This document is written o present users with a solution to this issue.

Answer:

Here we use USART2 as an example to explain interrupt response procedures and relevant notes.

1) Event flag is set

When a condition to generate an event flag is met, hardware will set a corresponding event flag automatically.

Example: RDBF flag is set as soon as USART2 receive data buffer is full.

Note: An event flag is set once its corresponding event generation condition is satisfied, regardless of its corresponding interrupt enable status.

2) Interrupt service response

When an event flag is set and its corresponding peripheral interrupt is enabled, hardware requests an interrupt service application to the core via peripheral interrupt channels.

Example: After USART2's RDBF flag is set and RDBFIEN is enabled, a request of interrupt service will be sent to the core via USART2 peripheral interrupt channel. In the meantime, USART2's NVIC pending bit indicates that there is a pending interrupt request, informing the code to jump to the interrupt service function to handle it.

Note: If a peripheral interrupt is disabled, its corresponding event flag is set but would not request interrupt service from the core.

3) Interrupt function execution

When the interrupt response priority rule is met, code will automatically jump to the callback task in the interrupt function.

An interrupt channel features several event flags that can trigger interrupts. Such event flags share the same NVIC pending bit, and this pending bit will be automatically cleared once code jumps to interrupt functions.

Note: The setting of any event flag due to any interrupt enabled within interrupt channels would set the NVIC pending bit, without recording the count of NVIC pending bit. Such pending bit is automatically cleared by hardware after code jump to the corresponding interrupt function.

In order to prevent missing callback tasks of event flags, it is necessary to differentiate event flags through software as shown below:

Solution: How to avoid missing callback tasks?

When using multiple event interrupts from the same peripheral, software can be used to differentiate which interrupt an event flag belongs to.

In the interrupt function, check the status of every event flag to determine whether to respond to a callback task.

Example: when using RDBF and TDBE interrupts of USART2, its interrupt function should be designed like below:

```
void USART2_IRQHandler(void)
{
  if(usart_flag_get(USART2, USART_RDBF_FLAG) != RESET)
  {
    /* user code */
  }

  if(usart_flag_get(USART2, USART_TDBE_FLAG) != RESET)
  {
    /* user code */
  }
}
```

Users can also check our DEMO as a reference at the location path:

AT32Fxxx_Firmware_Library_V2.x.x\project\at_start_fxxx\examples\spi\halfduplex_interrupt

How to avoid improper execution of callback tasks?

When using several event interrupts from the same peripheral, software can be used to differentiate which interrupt an event flag belongs to.

In the interrupt function, check the interrupt status and flag status of every event to determine whether to respond to a callback task.

Example: when using USART2's RDBF interrupt, and TDBE interrupt on a time-sharing basis, its interrupt function should be designed like below:

```
void USART2_IRQHandler(void)
{
  if(usart_flag_get(USART2, USART_RDBF_FLAG) != RESET)
  {
    /* user code */
  }
  if(USART2->ctrl1_bit.tdbeien != RESET)
  {
    if(usart_flag_get(USART2, USART_TDBE_FLAG) != RESET)
    {
      /* user code */
    }
  }
}
```

Users can also check our DEMO as a reference at the location path:

AT32Fxxx_Firmware_Library_V2.x.x\project\at_start_fxxx\examples\spi\halfduplex_transceiver_switch

4) End of interrupt and return

When an interrupt callback task is executed, it is necessary to clear its event status flag by software as soon as possible. After that, code will automatically return to its main function.

Type: MCU application
Applicable products: AT32 Family
Main function: Interrupts
Other function: None

Document revision history

Date	Revision	Changes
2022.2.30	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.