xEventGroupWaitBits()

[Event Group API]

Available From FreeRTOS V8.0.0

event_groups.h

```
EventBits_t xEventGroupWaitBits(
                    const EventGroupHandle_t xEventGroup,
                    const EventBits_t uxBitsToWaitFor,
                    const BaseType_t xClearOnExit,
                    const BaseType_t xWaitForAllBits,
                    TickType_t xTicksToWait );
```

Read bits within an RTOS event group, optionally entering the Blocked state (with a timeout) to wait for a bit or group of bits to become set.

This function cannot be called from an interrupt.

The RTOS source file `FreeRTOS/source/event_groups.c` must be included in the build for the xEventGroupWaitBits() function to be available.

Parameters:

xEventGroup The event group in which the bits are being tested. The event group must have previously been created using a call to xEventGroupCreate().

uxBitsToWaitFor A bitwise value that indicates the bit or bits to test inside the event group. For example, to wait for bit 0 and/or bit 2 set uxBitsToWaitFor to 0x05. To wait for bits 0 and/or bit 1 and/or bit 2 set uxBitsToWaitFor to 0x07. Etc.

uxBitsToWaitFor must **not** be set to 0.

xClearOnExit If xClearOnExit is set to pdTRUE then any bits set in the value passed as the uxBitsToWaitFor parameter will be cleared in the event group before xEventGroupWaitBits() returns if xEventGroupWaitBits() returns for any reason other than a timeout. The timeout value is set by the xTicksToWait parameter.

If xClearOnExit is set to pdFALSE then the bits set in the event group are not altered when the call to xEventGroupWaitBits() returns.

xWaitForAllBits xWaitForAllBits is used to create either a logical AND test (where all bits must be set) or a logical OR test (where one or more bits must be set) as follows:

If xWaitForAllBits is set to pdTRUE then xEventGroupWaitBits() will return when either **all** the bits set in the value passed as the uxBitsToWaitFor parameter are set in the event group or the specified block time expires.

If xWaitForAllBits is set to pdFALSE then xEventGroupWaitBits() will return when **any** of the bits set in the value passed as the uxBitsToWaitFor parameter are set in the event group or the specified block time expires.

xTicksToWait The maximum amount of time (specified in 'ticks') to wait for one/all (depending on the xWaitForAllBits value) of the bits specified by uxBitsToWaitFor to become set.

Returns:

The value of the event group at the time either the event bits being waited for became set, or the block time expired. The current value of the event bits in an event group will be different to the returned value if a higher priority task or interrupt changed the value of an event bit between the calling task leaving the Blocked state and exiting the xEventGroupWaitBits() function.

Test the return value to know which bits were set. If xEventGroupWaitBits() returned because its timeout expired then not all the bits being waited for will be set. If xEventGroupWaitBits() returned because the bits it was waiting for were set then the returned value is the event group value before any bits were automatically cleared because the xClearOnExit parameter was set to pdTRUE.

Example usage:

```
#define BIT_0   ( 1 << 0 )
#define BIT_4   ( 1 << 4 )

void aFunction( EventGroupHandle_t xEventGroup )
{
EventBits_t uxBits;
```

```
const TickType_t xTicksToWait = 100 / portTICK_PERIOD_MS;

  /* Wait a maximum of 100ms for either bit 0 or bit 4 to be set within
  the event group.  Clear the bits before exiting. */
  uxBits = xEventGroupWaitBits(
            xEventGroup,   /* The event group being tested. */
            BIT_0 | BIT_4, /* The bits within the event group to wait for. */
            pdTRUE,        /* BIT_0 & BIT_4 should be cleared before returning. */
            pdFALSE,       /* Don't wait for both bits, either bit will do. */
            xTicksToWait );/* Wait a maximum of 100ms for either bit to be set. */

  if( ( uxBits & ( BIT_0 | BIT_4 ) ) == ( BIT_0 | BIT_4 ) )
  {
      /* xEventGroupWaitBits() returned because both bits were set. */
  }
  else if( ( uxBits & BIT_0 ) != 0 )
  {
      /* xEventGroupWaitBits() returned because just BIT_0 was set. */
  }
  else if( ( uxBits & BIT_4 ) != 0 )
  {
      /* xEventGroupWaitBits() returned because just BIT_4 was set. */
  }
  else
  {
      /* xEventGroupWaitBits() returned because xTicksToWait ticks passed
      without either BIT_0 or BIT_4 becoming set. */
  }
}
```

[Back to the top] [About FreeRTOS] [Sitemap] [Report an error on this page]