



Quality RTOS & Embedded Software
[About](#) [Contact](#) [Support](#) [FAQ](#)

FreeRTOS Tutorial Books and Reference Manuals
Become an expert, while supporting the FreeRTOS project

[Quick Start](#) | [Supported MCUs](#) | [Books & Kits](#) | [Trace Tools](#) | [Ecosystem](#) | [TCP & FAT](#) | [Training](#) | [Email List+](#)  

[Home](#)
[FreeRTOS Books and Manuals](#)
FreeRTOS
About FreeRTOS
Features / Getting Started...
More Advanced...
Demo Projects
Supported Devices & Demos
API Reference
PDF Reference Manual
Task Creation
Task Control
Task Utilities
RTOS Kernel Control
Direct To Task Notifications
FreeRTOS-MPU Specific
Queues
Queue Sets
Semaphore / Mutexes
Software Timers
Event Groups (or 'Flags')
xEventGroupCreate()
vEventGroupDelete()
xEventGroupWaitBits()
xEventGroupSetBits()
xEventGroupSetBitsFromISR()
xEventGroupClearBits()
xEventGroupClearBitsFromISR()
xEventGroupGetBits()
xEventGroupGetBitsFromISR()
xEventGroupSync()
Co-routines
Contact, Support, Advertising
FreeRTOS Interactive!

[Quick Start Guide](#)
[Download Source](#)
FreeRTOS+ Lab Projects
FreeRTOS+TCP:
Thread safe TCP/IP stack
FreeRTOS+FAT:
Thread aware file system
FreeRTOS+ Ecosystem
Internet of Things:
Innovative complete solution
Fail Safe File System:
Ensures data integrity
InterNiche TCP/IP:
Low cost pre-ported libraries
FreeRTOS BSPs:
3rd party driver packages
FAT SL File System:
Super lean FAT FS
UDP/IP:
Thread aware UDP stack
Trace & Visualisation:
Tracealyzer for FreeRTOS
CLI:
Command line interface
WolfSSL SSL / TLS:
Networking security protocols
Safety:
TUV certified RTOS
RTOS Training:
Delivered online or on-site
IO:
[read\(\)](#), [write\(\)](#), [ioctl\(\)](#) interface

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:

<i>xEventGroup</i>	The event group in which the bits are being tested. The event group must have previously been created using a call to xEventGroupCreate() .
<i>uxBitsToWaitFor</i>	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 <code>uxBitsToWaitFor</code> to <code>0x05</code> . To wait for bits 0 and/or bit 1 and/or bit 2 set <code>uxBitsToWaitFor</code> to <code>0x07</code> . Etc. <code>uxBitsToWaitFor</code> must not be set to 0.
<i>xClearOnExit</i>	If <code>xClearOnExit</code> is set to <code>pdTRUE</code> then any bits set in the value passed as the <code>uxBitsToWaitFor</code> parameter will be cleared in the event group before <code>xEventGroupWaitBits()</code> returns if <code>xEventGroupWaitBits()</code> returns for any reason other than a timeout. The timeout value is set by the <code>xTicksToWait</code> parameter. If <code>xClearOnExit</code> is set to <code>pdFALSE</code> then the bits set in the event group are not altered when the call to <code>xEventGroupWaitBits()</code> returns.
<i>xWaitForAllBits</i>	<code>xWaitForAllBits</code> 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 <code>xWaitForAllBits</code> is set to <code>pdTRUE</code> then <code>xEventGroupWaitBits()</code> will return when either all the bits set in the value passed as the <code>uxBitsToWaitFor</code> parameter are set in the event group or the specified block time expires. If <code>xWaitForAllBits</code> is set to <code>pdFALSE</code> then <code>xEventGroupWaitBits()</code> will return when any of the bits set in the value passed as the <code>uxBitsToWaitFor</code> parameter are set in the event group or the specified block time expires.
<i>xTicksToWait</i>	The maximum amount of time (specified in 'ticks') to wait for one/all (depending on the <code>xWaitForAllBits</code> value) of the bits specified by <code>uxBitsToWaitFor</code> 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;
```

Latest News:

FreeRTOS V9.0.0rc1 is now available for [download](#) and comment.

Buildable Examples

FreeRTOS+TCP



FreeRTOS+FAT

Try Them Now

Sponsored Links

↓ Now With No Code Size Limit! ↓

TrueSTUDIO



The best **UNLIMITED** **FREE** ARM[®] development on the planet.

 **NOW** WITHOUT RESTRICTIONS

NO CODE-SIZE LIMITATION



↑ Free Download Without Registering ↑

USB TCP/IP File Systems



Supplied as **integrated** and **functioning FreeRTOS projects**

from the **Official FreeRTOS Partner**

 **Reliance Edge™**

The Tiny, Open Source, Power Fail-safe File System for FreeRTOS

Now **Reliable Things**



```

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. */
}
}

```

[\[Back to the top \]](#)
[\[About FreeRTOS \]](#)
[\[Sitemap \]](#)
[\[Report an error on this page \]](#)

Copyright (C) 2004-2010 Richard Barry. Copyright (C) 2010-2016 Real Time Engineers Ltd.
 Any and all data, files, source code, html content and documentation included in the FreeRTOS™ distribution or available on this site are the exclusive
 property of Real Time Engineers Ltd.. See the files license.txt (included in the distribution) and this [copyright notice](#) for more information. FreeRTOS™
 and FreeRTOS.org™ are trade marks of Real Time Engineers Ltd.

