



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


CONNECT MIDDLEWARE
 File systems USB Networking



[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
 uxQueueMessagesWaiting()
 uxQueueMessagesWaitingFromISR()
 uxQueueSpacesAvailable()
 xQueueCreate()
 vQueueDelete()
 xQueueReset()
 xQueueSend()
 xQueueSendToBack()
 xQueueSendToFront()
 xQueueReceive()
 xQueueOverwrite()
 xQueueOverwriteFromISR()
 xQueuePeek()
 xQueuePeekFromISR()
 xQueueSendFromISR()
 xQueueSendToBackFromISR()
 xQueueSendToFrontFromISR()
 xQueueReceiveFromISR()
 vQueueAddToRegistry()
 vQueueUnregisterQueue()
 xQueuesQueueFullFromISR()
 xQueuesQueueEmptyFromISR()
 Queue Sets
 Semaphore / Mutexes
 Software Timers
 Event Groups (or 'Flags')
 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

xQueueSend

[Queue Management]

```

queue.h

BaseType_t xQueueSend(
    QueueHandle_t xQueue,
    const void * pvItemToQueue,
    TickType_t xTicksToWait
);

```

This is a macro that calls `xQueueGenericSend()`. It is included for backward compatibility with versions of FreeRTOS that did not include the `xQueueSendToFront()` and `xQueueSendToBack()` macros. It is equivalent to `xQueueSendToBack()`.

Post an item on a queue. The item is queued by copy, not by reference. This function must not be called from an interrupt service routine. See `xQueueSendFromISR()` for an alternative which may be used in an ISR.

Parameters:

| | |
|----------------------------|--|
| <code>xQueue</code> | The handle to the queue on which the item is to be posted. |
| <code>pvItemToQueue</code> | A pointer to the item that is to be placed on the queue. The size of the items the queue will hold was defined when the queue was created, so this many bytes will be copied from <code>pvItemToQueue</code> into the queue storage area. |
| <code>xTicksToWait</code> | The maximum amount of time the task should block waiting for space to become available on the queue, should it already be full. The call will return immediately if the queue is full and <code>xTicksToWait</code> is set to 0. The time is defined in tick periods so the constant <code>portTICK_PERIOD_MS</code> should be used to convert to real time if this is required. |

If `INCLUDE_vTaskSuspend` is set to '1' then specifying the block time as `portMAX_DELAY` will cause the task to block indefinitely (without a timeout).

Returns: `pdTRUE` if the item was successfully posted, otherwise `errQUEUE_FULL`. **Example usage:**

```

struct AMessage
{
    char ucMessageID;
    char ucData[ 20 ];
} xMessage;

unsigned long ulVar = 10UL;

void vATask( void *pvParameters )
{
    QueueHandle_t xQueue1, xQueue2;
    struct AMessage *pxMessage;

    /* Create a queue capable of containing 10 unsigned long values. */
    xQueue1 = xQueueCreate( 10, sizeof( unsigned long ) );

    /* Create a queue capable of containing 10 pointers to AMessage structures.
    These should be passed by pointer as they contain a lot of data. */
    xQueue2 = xQueueCreate( 10, sizeof( struct AMessage * ) );

    /* ... */

    if( xQueue1 != 0 )
    {
        /* Send an unsigned long. Wait for 10 ticks for space to become
        available if necessary. */
        if( xQueueSend( xQueue1,
            ( void * ) &ulVar,
            ( TickType_t ) 10 ) != pdPASS )
        {
            /* Failed to post the message, even after 10 ticks. */
        }
    }

    if( xQueue2 != 0 )
    {
        /* Send a pointer to a struct AMessage object. Don't block if the
        queue is already full. */
        pxMessage = &xMessage;
        xQueueSend( xQueue2, ( void * ) &pxMessage, ( TickType_t ) 0 );

        /* ... Rest of task code. */
    }
}

```

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!




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

DOWNLOAD NOW WITHOUT REGISTRATION

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**



Download FreeRTOS Project

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

Now Reliable Things

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

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.

