



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



[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
 [TaskHandle_t \(type\)](#)
 [xTaskCreate\(\)](#)
 [vTaskDelete\(\)](#)
 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')
 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

xTaskCreate

[Task Creation]

task. h

```
BaseType_t xTaskCreate(
    TaskFunction_t pvTaskCode,
    const char * const pcName,
    unsigned short usStackDepth,
    void *pvParameters,
    BaseType_t uxPriority,
    TaskHandle_t *pvCreatedTask
);
```

Create a new task and add it to the list of tasks that are ready to run.

If you are using [FreeRTOS-MPU](#) then it is recommended to use [xTaskCreateRestricted\(\)](#) in place of [xTaskCreate\(\)](#). Using [xTaskCreate\(\)](#) with FreeRTOS-MPU allows tasks to be created to run in either Privileged or User modes (see the description of [uxPriority](#) below). When Privileged mode it used the task will have access to the entire memory map, when User mode is used the task will have access to only its stack. In both cases the MPU will not automatically catch stack overflows, although the standard FreeRTOS stack overflow detection schemes can still be used. [xTaskCreateRestricted\(\)](#) permits much greater flexibility.

Parameters:

<i>pvTaskCode</i>	Pointer to the task entry function. Tasks must be implemented to never return (i.e. continuous loop).
<i>pcName</i>	A descriptive name for the task. This is mainly used to facilitate debugging. Max length defined by <code>configMAX_TASK_NAME_LEN</code> .
<i>usStackDepth</i>	The size of the task stack specified as the number of variables the stack can hold - not the number of bytes. For example, if the stack is 16 bits wide and <code>usStackDepth</code> is defined as 100, 200 bytes will be allocated for stack storage. The stack depth multiplied by the stack width must not exceed the maximum value that can be contained in a variable of type <code>size_t</code> .
	See the FAQ How big should the stack be?
<i>pvParameters</i>	Pointer that will be used as the parameter for the task being created.
<i>uxPriority</i>	The priority at which the task should run. Systems that include MPU support can optionally create tasks in a privileged (system) mode by setting bit <code>portPRIVILEGE_BIT</code> of the priority parameter. For example, to create a privileged task at priority 2 the <code>uxPriority</code> parameter should be set to <code>(2 portPRIVILEGE_BIT)</code> .
<i>pvCreatedTask</i>	Used to pass back a handle by which the created task can be referenced.

Returns:

`pdPASS` if the task was successfully created and added to a ready list, otherwise an error code defined in the file `projdefs.h`

Example usage:

```
/* Task to be created. */
void vTaskCode( void * pvParameters )
{
    for( ;; )
    {
        /* Task code goes here. */
    }
}

/* Function that creates a task. */
void vOtherFunction( void )
{
    static unsigned char ucParameterToPass;
    TaskHandle_t xHandle = NULL;

    /* Create the task, storing the handle. Note that the passed parameter
    ucParameterToPass must exist for the lifetime of the task, so in this
    case is declared static. If it was just an automatic stack variable
    it might no longer exist, or at least have been corrupted, by the time
    the new task attempts to access it. */
    xTaskCreate( vTaskCode, "NAME", STACK_SIZE, &ucParameterToPass, tskIDLE_PRIORITY,
                &xHandle );
    configASSERT( xHandle );

    /* Use the handle to delete the task. */
    if( xHandle != NULL )
    {
        vTaskDelete( xHandle );
    }
}
```

Google™ Custom Search

Search

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

Buildable Examples

FreeRTOS+TCPFreeRTOS+FAT

Try Them Now

Sponsored Links

↓ Now With No Code Size Limit! ↓

TrueSTUDIO


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

DOWNLOAD NOW WITHOUT REGISTRATION

NO CODE-SIZE LIMITATION

atollic

↑ 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

```
}
```

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

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.

