

**FreeRTOS on STM32 (simon.yuan)**

# 目录

**FreeRTOS在STM32上应用开发, 由于水平有限，难免有不当之处，欢迎交流讨论(QQ:343264122, QQ群：755127489)**

1. 移植FreeRTOS到STM32开发板准备工作[...................](#_bookmark0)2
2. 如何理解任务..............................................................3
3. 静态SRAM中创建任务...................................................9
4. 动态SRAM中创建任务.................................................10
5. 任务管理...................................................................10
6. 嵌入式操作系统........................................................10
7. Cortex-M处理器上的OS特性......................................10

# 移植FreeRTOS到STM32开发板的准备工作

## 下载官方的FreeRTOS源码

这里使用FreeRTOS V9.0.0源码([www.freertos.org](http://www.freertos.org)官方网站上下载)，下载完成后了解一下源码结构

├─FreeRTOS  
│  ├─Demo                 // 各种开发工具的完整Demo，开发者可以方便的以此搭建出自己的项目，甚至直接使用  
│  │  ├─Common        // 所有例程都可以使用的演示例程文件  
│  │  └─其他               // 对应平台和开发工具的项目例程（命名：平台\_开发工具，例如：CORTEX\_M4F\_M0\_LPC43xx\_Keil）  
│  ├─License           // 使用修改过的GPL  
│  └─Source               // FreeRTOS的源码  
│      ├─include         // 源码对应的头文件  
│      └─portable       // 每个支持的处理器架构需要一小段与处理器架构相关的RTOS代码。该目录下即为和开发平台相关的代码  
│            ├─MemMang    // FreeRTOS内存管理方案（一般要根据平台来选择以下5个之一）  
│            │     heap\_1.c  
│            │     heap\_2.c  
│            │     heap\_3.c  
│            │     heap\_4.c  
│            │     heap\_5.c  
│            └─其他         // 其他开发工具相关的代码，需要根据自己的开发工具进行选择  
│      croutine.c         // 协线程（协程）文件，和任务类似，在系统资源比较缺乏下使用  
│      event\_groups.c     // 事件标志组  
│      list.c             // 列表结构描述，在内核整体控制上都使用了列表格式数据处理,一切数据结构的基础  
│      queue.c             // 队列，任务和任务之间的通讯处理  
│      tasks.c             // 所有任务相关函数  
│      timers.c           // 软件定时器，以任务形式存在  
|      stream\_buffer.c     // 10.0.0 新增  
└─FreeRTOS-Plus          // FreeRTOS+组件和演示例程

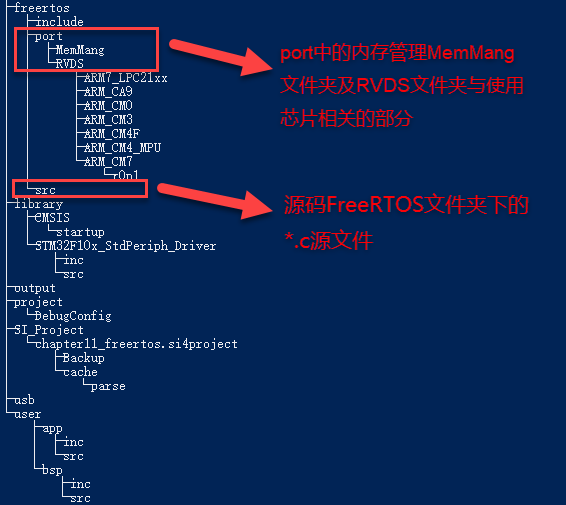
## 准备一个裸机项目固件工程

## 将freeRTOS的源码添加到裸机工程中.主要是添加以下几部分：

### 源码FreeRTOS文件夹下的\*.c文件

### 与MCU硬件对接的接口部分,包括MemMang的内存管理文件及RVDS中相应内核文件

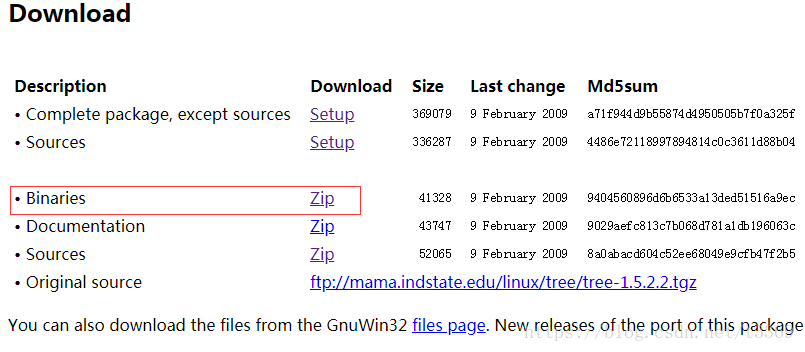
### 在源码的include文件夹下的所有的头文件



## 可以使用Git bash下的tree命令或者windows的cmd命令来查看结构.Git bash下没有原生的tree命令，通过以下方式获取

### <http://gnuwin32.sourceforge.net/packages/tree.htm下载Binaries>压缩包

### 将tree.exe放入git安装目录的git\usr\bin中



# 如何理解任务

## 什么是任务

## 任务实质就是一个函数，用来完成开发者设计设计的既定工作，例如点亮一个LED灯，那么电路LED灯的一个函数就是一个任务.

## 任务的必要素(即RTOS的调度器需要通过任务的必要素去管理各种任务)

## 任务具备的必要素为：

### 任务的主体函数：实现任务的具体的coding代码

### 任务栈：任务为函数，函数运行在栈上，所以任务运行需要栈空间；而且栈空间需要有特定地址范围的独占空间栈，以便于在多任务切换时对压栈和弹栈不产生混乱.

### 任务控制块：这个任务控制块优点类似于队列中的属性信息(也可以理解成FAT文件系统中的FAT表头)，用来记录存储空间(任务栈)的相关的属性信息，以便于RTOS在管理任务时方便，RTOS在管理任务时只需要通过操作任务控制块就可以简便的对任务及任务栈信息进行记录

## 任务在RTOS中存在的形态

任务在RTOS中为一个while(1)的死循环

## 基于RTOS中的任务形态，任务时如何被调度的

任务在RTOS中会有不同的状态存在：就绪态，挂起态，运行态等等，针对不同的态，会被RTOS添加到不同的状态链表中.

## 多任务在RTOS中如何被调度的

在任务系统中，不同的任务会因状态不同而被添加到不同的状态链表中，例如就绪态链表，其中可能有多个就绪态的任务存在，那么RTOS在进行任务调度时会先调度哪一个任务会根据开发者设计的思想来进行调度，例如按照优先级的话会根据开发者设定哪个优先级最高进行先调度，还有同优先级的时间片运行机制或者优先级反转机制等等.

# 静态SRAM中创建任务

## 创建任务

创建任务就需要创建任务的要素成员：任务栈，任务函数及任务控制块

## FreeRTOS的相关配置

## 启动及理解任务调度器

## 静态SRAM中创建任务相应知识点

### C运算优先级，下表列出 C 运算符的优先级和结合性。运算符从顶到底以降序列出。



### 基于STM32F103VCT6(32bit Cortex-M3内核),几种数据类型的占用内存大小



# 动态SRAM中创建任务

# 任务管理

任务管理的实质是通过FreeRTOS中提供的API函数实现对任务状态的迁移.用到的FreeRTOS中的API函数有：

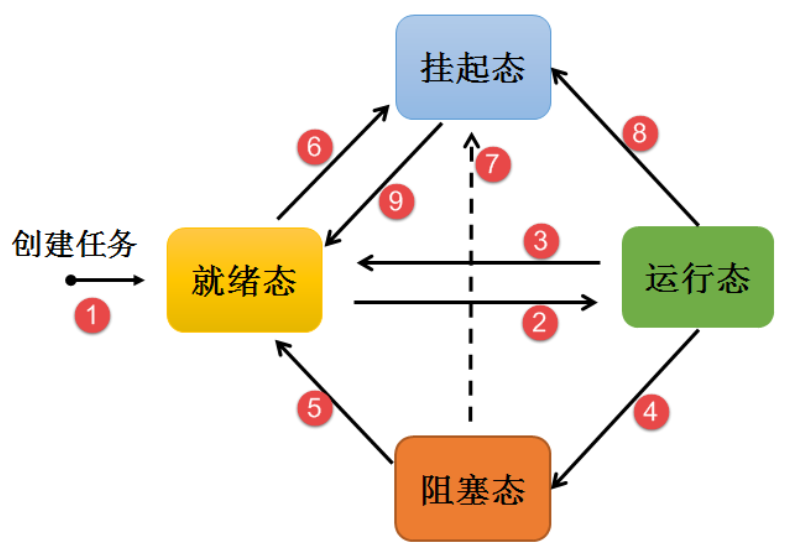
任务挂起函数(任务挂起，任务挂起所有)

任务恢复函数(任务恢复，任务恢复所有，任务从ISR中恢复)

任务删除函数

任务延时函数(相对延时函数)

任务绝对延时(vTaskDelayUntil)



# 嵌入式操作系统

按对外部事件的响应能力来分类，嵌入式操作系统有分时操作系统和实时操作系统。如果操作系统能使计算机系统及时的响应外部事件请求，并能控制所有实时设备和实时任务协调运行，且能在一个规定的时间内完成对事件的处理，那么这种系统就称为实时操作系统（ RTOS）

按时间的正确程度来分，实时操作系统又分为硬件的实时操作系统和软件的实时操作系统。系统必须在及其严格的时间内完成的任务叫做硬件的实时操作系统，要是不是很严格的话就是软件的实时操作系统

# Cortex-M处理器上的OS特性

## 调度器及合作式调度器

调度器就是使用相关的调度算法来决定当前需要执行的任务。所有的调度器有一个共同的特性：调度器可以区分就绪态任务和挂起任务（ 由于延迟，信号量等待，邮箱等待，事件组等待等原因而使得任务被挂起）。调度器可以选择就绪态中的一个任务，然后激活它（ 通过执行这个任务）。当前正在执行的任务是运行态的任务。不同调度器之间最大的区别就是如何分配就绪态任务间的完成时间。