阿熊的FreeRTOS教程系列！

哈喽大家好！我是你们的老朋友阿熊！STM32教程系列更新完结已经有一段时间了，视频反馈还是不错的，从今天开始我们将会更新我们的FreeRTOS的教程

由于东西真的太多了，也纠结了很久要不要讲这个系列，毕竟难度真的很大，怕在难以做到那么通俗易懂，经过一段时间的考虑，还是决定好了给大家做一个入门级的讲解使用，由于FreeRTOS的内容真的很多，作为还是学生的我使用的也相对较少，操作系统层面的东西，我会用最大的能力去让大家理解，主要讲述主要功能，学完以后保证大伙可以理解80%以上的FreeRTOS的使用场景，好了废话不多说，开始我们的课程吧！

第一章：**FreeRTOS**的简述

说到FreeRTOS，就不得不先带着大家去了解一下我们的RTOS

壹：**RTOS**的简介

实时操作系统（Real Time Operating System，简称RTOS）是指当外界事件或数据产生时，能够接受并以足够快的速度予以处理，其处理的结果又能在规定的时间之内来控制生产过程或对处理系统做出快速响应，调度一切可利用的资源完成实时任务，并控制所有实时任务协调一致运行的操作系统。提供及时响应和高可靠性是其主要特点

https://baike.baidu.com/item/%E5%AE%9E%E6%97%B6%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F/357530?fromtitle=RTOS&fromid=987080&fr=aladdin

这里有转跳链接，小伙伴们可以转跳过去仔细看看，这里就不过多的赘述，简单的总结如下：

大体上，实时操作系统（RTOS）要求：

- 多任务
- 处理能被区分优先次序的进程线
- 一个中断水平的充分数量

被装入作为微型设备一部分的内含小操作系统经常要求实时操作系统。一些核心问题能被考虑以符合实时操作系统的需求。然而，因为像设备驱动程序这样的其他成份，也通常被特别的方法需要，一个实时操作系统通常比核心更大

RTOS只是一个统称，他可以分为各种各样的版本以及平台，由于RTOS需占用一定的系统资源(尤其是RAM资源)，只有μC/OS-II、embOS、salvo、FreeRTOS等少数能在小RAM单片机上运行。相对μC/OS-II、embOS等商业操作系统，FreeRTOS操作系统是完全免费的操作系统，具有源码公开、可移植、可裁减、调度策略灵活的特点，可以方便地移植到各种单片机上运行

然后我们现在来看我们的FreeRTOS

贰：**FreeRTOS**的简介

FreeRTOS是一个迷你的实时操作系统内核。作为一个轻量级的操作系统，功能包括：任务管理、时间管理、信号量、消息队列、内存管理、记录功能、软件定时器、协程等，可基本满足较小系统的需要

其功能特点如下：

- 用户可配置内核功能**(**可裁剪**)**
- 多平台的支持
- 提供一个高层次的信任代码的完整性
- 目标代码小，简单易用
- 遵循**MISRA-C**标准的编程规范
- 强大的执行跟踪功能
- 堆栈溢出检测
- 没有限制的任务数量
- 没有限制的任务优先级
- 多个任务可以分配相同的优先权

- 队列，二进制信号量，计数信号灯和递归通信和同步的任务
- 优先级继承
- 免费开源的源代码

并且最重要的一点是支持免费的商用，而且社区环境很好，使用的人很多

这里有转跳链接，小伙伴们可以转跳过去仔细看看

https://baike.baidu.com/item/FreeRTOS/9786143?fr=aladdin

叁：多任务操作系统的引入

说到多任务操作系统，我们就不得不提到我们的裸机开发，裸机开发的话，它并不是相对于多任务操作系统就很弱，他们俩在各自的领域有各自的使用场景，也有各自的优缺点，就比如说我们直接进行裸机开发，它主要是在我们的一个while循环中进行所有的项目操作，有些情况下，会有一个或者多个中断来处理一些突然发生的事或者已经设计好的事情，如果我们仅仅是开发一些小项目或者功能比较单一的项目，裸机是最可靠最有效而且很方便开发的一种模式，但是当我们的项目功能越来越多，我们就要引入多任务的操作系统，多任务操作系统顾名思义就是可以处理很多个任务，就像我们的手机一样，我们的手机后台也是可以挂起多个应用的

在我们引入操作系统之后，我们就不需要去精心的设计我们的时序流程，因为各个任务之间是不存在相互干扰的，我们只需要使用我们的命令去开启或者关闭某一些任务就可以了，当然这肯定是需要占用一些系统资源的，不过现在的单片机的内存都是足够大的，所以我们一般情况下是不需要担心我们的操作系统内核，而是直接移植进行使用，并且我们的多任务操作系统的思维，就很像我们人去思考的思维更加的方便，而我们的裸机开发更像是机械思维，在开发过程中以及开发流程中操作系统是非常方便使用以及开发的

我们可以再通俗一点的讲，把我们的单片机当做一台电脑的CPU，就像我们使用的电脑一样，我们在上面安装了windows系统，而我们需要执行的任务，比如说打开QQ或者其他的任务，只需要在操作系统上进行完成就可以了

说了这么多，相信小伙伴们还是对实际的多任务操作系统还是没什么概念，这里就简单给大家举一个例子

这里并不会出现源码，只是大概的体现一下RTOS与裸机开发的不同之处

裸机开发中LED闪烁：

```
//初始化
main()
{
```

```
    whlie(1)
    {
        //LED1置为高电平
        ......
        //延时1000MS
        ......
        //LED1置为低电平
        ......
        //延时1000MS
        ......
    }
}
```

这是只有一个LED的情况如果有两个LED，一个需要500MS闪烁，一个需要一秒闪烁一次，这样的话我们的代码是不是难度就加大了,好像逻辑就混乱了

```
//初始化
main()
{
    whlie(1)
    {
        //LED1置为高电平
        ......
        //延时1000MS
        ......
        //LED1置为低电平
        ......
        //延时1000MS
        ......
        //LED2置为高电平
        ......
        //延时500MS
        ......
        //LED置为低电平
        ......
        //延时500MS
        ......
    }

}
```

这是我们大脑最希望的添加代码方式，很显然他是错的，两个任务之间产生了相互的影响，使得两个任务都执行错误，很多同学就会说这是阿熊代码的问题，对没错，这种思想在裸机开发中肯定是错的，但是在我们的RTOS中他就可以是对的

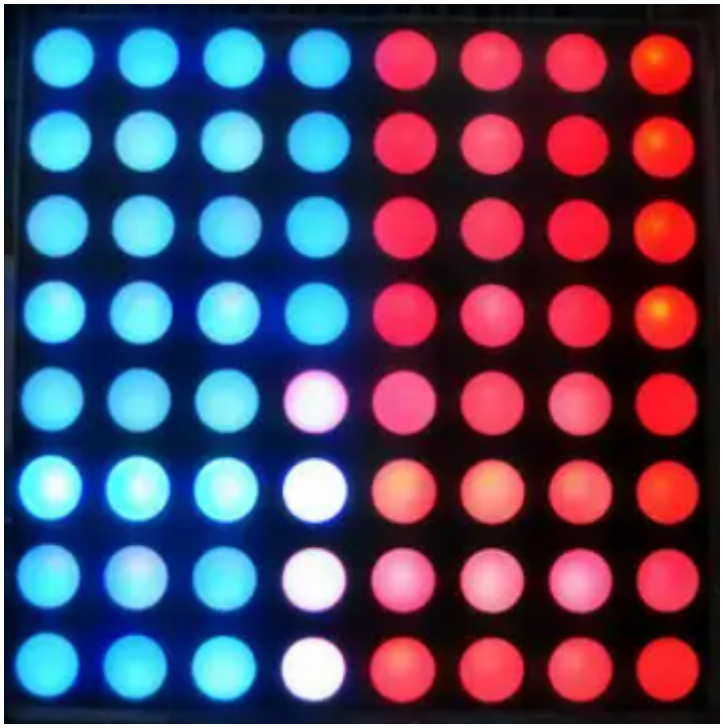
```
//创建LED2任务
LED2_Task(){
    //LED2置为高电平
    ......
    //延时500MS
    ......
    //LED置为低电平
    ......
    //延时500MS
    ......
}
```

```
//创建LED1任务
LED1_Task(){
    //LED1置为高电平
    ......
    //延时1000MS
    ......
    //LED1置为低电平
    ......
    //延时1000MS
    ......
}
```

这是我们独立的两个任务内容，我们只需要把他的扔到我们的任务执行器里，他就会“同时”运行了，很多小伙伴就会疑问了，单片机明明只有一个核，为什么可以同时执行多个任务呢？

肆：任务“同时”执行的原理

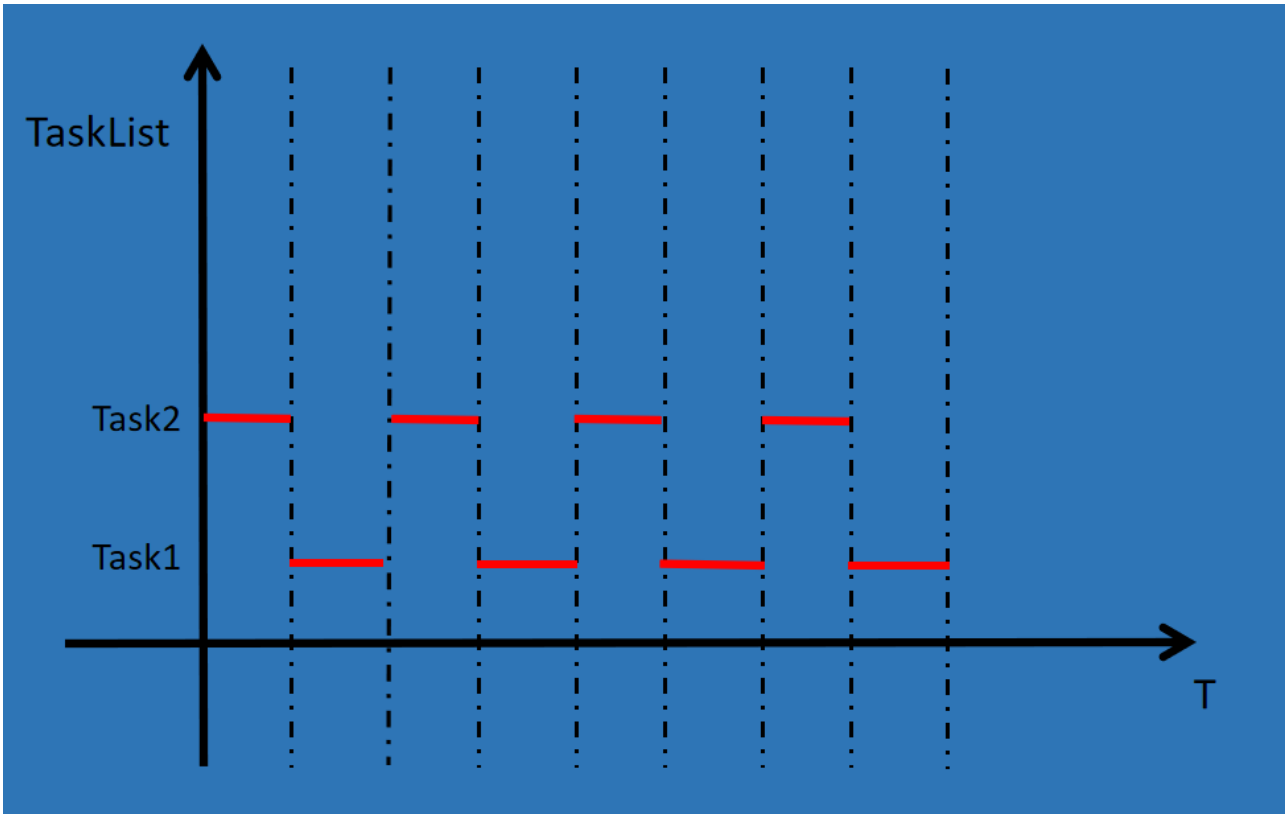
还记得我们51单片机里面讲的8*8LED的点阵吗？



就这家伙，还记得他是怎么做到使用16个引脚驱动64个灯的吗？

视觉暂停，也可以说是“视觉欺骗”，他并不是同时亮起来的，而是一排一排亮起来的，只不过切换的速度太快了，肉眼反应不过来，所以就同时亮起来了

我们RTOS也差不多是这样，我们可以让我们的每个任务执行一个时间单位，然后就切换到另外一个任务执行一个时间单位，再切换回去，两个任务都是独立运行的，互不影响，由于切换的频率很快，就感觉像是同时运行的一样



这是一个简单的示意图，小伙伴们试着理解一下，我们本章的内容就是这些，下节课开始我们将会教小伙伴们如何进行我们操作系统的移植并且让我们的LED可分别点亮