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！

第七章：信号量

前面介绍的队列(queue)可以用于传输数据：在任务之间、任务和中断之间，消息队列用于传输多个数据，占用时间按也相对较长，但是有时候我们只需要传递状态，这个状态值需要用一个数值表示，就比如说：

使用我们的LED显示屏，去显示我们读取到的温湿度，如果我们的屏幕按固定的时间去刷新，就会非常的耗资源，但是如果使用我们的信号量，在我们读取到数据之后给屏幕发送一个信号，再让屏幕刷新，这样的话可以达到同样的效果，并且可以减少资源的占用，而且可以达到同步的效果

壹：信号量的特征

信号量这个名字，我们可以把它拆分来看，信号可以起到通知信号的作用，然后我们的量还可以用来表示资源的数量，当我们的量只有0和1的时候，它就可以被称作二值的信号量，只有两个状态，当我们的那个量没有限制的时候，它就可以被称作为计数型信号量

信号量也是队列的一种

我们前面讲过队列创建的时候，需要传入队列的长度以及队列的大小，而我们的信号量其实就是一种特殊的队列，只不过它的大小是0，毕竟我们的信号量是不需要传递数据，只需要传递信号，然后长度是N，当N=1的时候就是二值信号量，他就只有0和1两个状态，这里的0和1两个状态是指被填入和被拿走这两个状态，当N>1时，就是我们的计数信号量，他也不传递数据，只是传递一个数量值，一般是记录我们设备的资源数量

二值信号量：

二值信号量其实就是一个长度为1，然后大小为零的队列，然后它的状态只有0和1两种状态，也就是被写入和被取走的两种状态，通常情况下，我们用它来进行数据同步，还是以前面的那个例子，我们在项目中添加一个二值信号量，当我们读取到了温湿度，我们就把我们的二值信号量填上也就是置1，然后我们就可以在屏幕显示的那个任务中进行判断，当他读到了新数据，我们再进行屏幕的刷新，这样一说，是不是感觉就很像我们自己平时在程序中设置的那个变量flag，当我们完成了某件事，然后我们就把它标志为给打开，等屏幕刷新我们再把它清除

计数信号量：

而我们的计数信号量其实就是二指信号量的升级版，我们那只是信号量只有0和1两种状态，而我们的这个就有很多很多种状态，通常情况下是用来记录系统的资源，就比如说我们记录一个车库里的辆车，每当有人来停车或者取车的时候他的数量都会进行对应的加减，我们就可以很轻松的看到我们车库的状态，然后也知道车库里面有几辆车

两种信号大同小异，我们接下来讲一下相关的用法

贰：二值信号量的基本操作

创建：

手动创建默认初始值为0，CubeMX创建默认为1

动态创建：

```
SemaphoreHandle_t   xSemaphoreCreateBinary( void );
//无需传入参数
//返回句柄，非 NULL 表示成功
//默认初始化内部是空
```

```
SemaphoreHandle_t   vSemaphoreCreateBinary( void );
//此函数已过时，和上面的区别是信号量创建默认就是1
```

静态创建：

```
SemaphoreHandle_t   xSemaphoreCreateBinaryStatic(
                                StaticSemaphore_t *pxSemaphoreBuffer);
//几乎不使用，了解即可
```

删除：

```
void    vSemaphoreDelete( SemaphoreHandle_t xSemaphore );
//传入信号量句柄，无返回值
```

Take/Give：

Give：释放（置**1**）

正常任务中使用

```
BaseType_t  xSemaphoreGive( SemaphoreHandle_t   xSemaphore );
//传入信号量句柄，信号量将会被置1
//返回：
//  pdTRUE 表示成功
//  如果二进制信号量的计数值已经是 1，再次调用此函数则返回失败
```

中断中使用

```
BaseType_t xSemaphoreGiveFromISR( SemaphoreHandle_t xSemaphore,
                                  BaseType_t
*pxHigherPriorityTaskWoken);
//传入句柄，以及判断是否需要切换任务
```

Take：获取（清空）

正常任务中使用

```
BaseType_t  xSemaphoreTake(SemaphoreHandle_t
xSemaphore,TickType_t   xTicksToWait);
//传入句柄，以及等待等待时间
//0：不阻塞，马上返回
//portMAX_DELAY: 一直阻塞直到成功
//返回：pdTRUE 表示成功
```

中断中使用：

```
BaseType_t  xSemaphoreTakeFromISR(SemaphoreHandle_t xSemaphore,
                                  BaseType_t
*pxHigherPriorityTaskWoken);
//传入句柄，以及判断是否需要切换任务
```

叁：二值信号量的使用

实验一：

创建任务一：模拟温湿度采集（按键按下采集成功）

创建任务二：模拟LED屏幕刷新（使用串口发送信息）

现象：

我们的按键和我们的串口成功达成了同步，只有当我们按键释放了信号量，我们的串口才有机会发送信息

分析：

当我们的按键一按下时成释放信号量，若多次释放将会返回错误信息，串口也是，按下按键成功接收，多次按下，当等待时间为0时，返回失败，为最大值时，将无限等待

肆：计数信号量的基本操作

!!使用时configSUPPORT_DYNAMIC_ALLOCATION需要设置为1

创建：

如果使用的**Cube MX**的话其默认初始化是最大值

动态创建：

```
SemaphoreHandle_t xSemaphoreCreateCounting( UBaseType_t uxMaxCount,
                                            UBaseType_t
uxInitialCount);
//参数：最大计数值、初始化计数值
//返回：信号量句柄
```

静态创建：

```
SemaphoreHandle_t xSemaphoreCreateCountingStatic(
                                  UBaseType_t uxMaxCount,
                                  UBaseType_t uxInitialCount
                                  StaticSemaphore_t
*pxSemaphoreBuffer );
//几乎不会使用
```

删除：

同二值信号量，自行转跳

Take/Give：

同二值信号量，自行转跳

读取数值：

```
UBaseType_t uxSemaphoreGetCount( SemaphoreHandle_t xSemaphore );
//传入信号量句柄（二值信号量通用），返回数值
```

伍：计数信号量的使用

实验二：

模拟车库计数器

创建计数信号量大小为10

任务一：（按键一）可以将车停入车库（按键二）可以将车开出车库

任务二：串口每2秒显示一下车库内车辆数据

现象：

我们可以通过按键一和按键二去停车还有取出车辆，并且间隔两秒打印出车库内的状况

分析：

我们成功使用对应的函数，将我们的车停入到车库以及开出车库并且使用，还能获取到车库内的状况