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！

第十一章：软件定时器

顾名思义就是通过我们软件去实现我们的定时器功能，有时候我们的单片机它的内部硬件定时器资源并不是很多，然后我们FreeRTOS就给我们提供了现成的软件定时器，可以一定程度上去替代我们的硬件定时器，但是精度不高，毕竟是软件模拟的，而且最小的定时单位是我们的时间节拍，也就是1毫秒以上

壹：软件定时器的守护任务

软件定时器是通过守护任务去实现的，我们开启定时器的功能后也就是configUSE_TIMERS被设置为 1 时，我们的系统在初始化内核的时候就会创建一个守护任务，这个守护任务就是用于调度我们的软件定时器，其内部实现的原理是通过队列去完成的，就比如说我们的执行相关的定时器的函数，其实他就是向队列中发了一个数据，然后守护任务去做对应的处理，然后就实现了对应的功能

既然是任务，我们就可以设置他的优先级，可以通过configTIMER_TASK_PRIORITY去设置

一般建议优先级大一点，不然会被其他任务抢走了，使得定时不准确

既然有队列，我们也可以设置队列长度，configTIMER_QUEUE_LENGTH就是控制队列长度的宏定义，如果队列满了，就会导致我们的消息发送失败，然后定时器就无法完成特定功能

贰：定时器回调函数

我们在stm32的开发中就已经有碰到过很多的回调函数了，就比如说我们的中断回调函，其实它和普通的函数并没有什么区别，只是在我们触发中断之后，它会调用我们的另外一个函数进行转跳

这里只是带给大家回忆一下以前回调函数的概念，并且必须要告诉大家的是在回调函数中是不能使用相关的阻塞函数不然会出大问题

叁：软件定时器的基本函数

定时器的创建：

动态创建：

```
TimerHandle_t xTimerCreate( const char * const pcTimerName,
                            const TickType_t xTimerPeriodInTicks,
                            const UBaseType_t uxAutoReload,
                            void * const pvTimerID,
                            TimerCallbackFunction_t
pxCallbackFunction );

//pcTimerName:定时器名字，用处不大，尽在调试时用到
//xTimerPeriodInTicks: 周期，以 Tick 为单位
//uxAutoReload: 类型，pdTRUE 表示自动加载，pdFALSE 表示一次性
//pvTimerID: 回调函数可以使用此参数，比如分辨是哪个定时器
//pxCallbackFunction: 回调函数
//返回值: 成功则返回 TimerHandle_t，否则返回 NULL
```

静态创建：

```
TimerHandle_t xTimerCreateStatic(const char * const pcTimerName,
                                 TickType_t xTimerPeriodInTicks,
                                 UBaseType_t uxAutoReload,
                                 void * pvTimerID,
                                 TimerCallbackFunction_t
pxCallbackFunction,
                                 StaticTimer_t *pxTimerBuffer );
//pcTimerName:定时器名字，用处不大，尽在调试时用到
//xTimerPeriodInTicks: 周期，以 Tick 为单位
//uxAutoReload: 类型，pdTRUE 表示自动加载，pdFALSE 表示一次性
//pvTimerID: 回调函数可以使用此参数，比如分辨是哪个定时器
//pxCallbackFunction: 回调函数
//pxTimerBuffer: 传入一个 StaticTimer_t 结构体，将在上面构造定时器
//返回值: 成功则返回 TimerHandle_t，否则返回 NULL
```

回调函数类型：

```
typedef void (* TimerCallbackFunction_t)( TimerHandle_t xTimer );
//我们需要按照这种形式去写我们的回调函数命名,就像下面这种
void ATimerCallback( TimerHandle_t xTimer )
```

删除：

```
BaseType_t xTimerDelete( TimerHandle_t xTimer, TickType_t
xTicksToWait );
//传入句柄，以及超时时间（队列写入的等待时间）
```

启动定时器：

普通任务中使用：

```
BaseType_t xTimerStart( TimerHandle_t xTimer, TickType_t
xTicksToWait );
//传入句柄，以及超时时间（队列写入的等待时间）
```

中断中使用：

```
BaseType_t  xTimerStartFromISR( TimerHandle_t xTimer,
                                BaseType_t
*pxHigherPriorityTaskWoken );
//传入句柄，判断任务是否需要切换
```

停止定时器：

普通任务中使用：

```
BaseType_t xTimerStop( TimerHandle_t xTimer, TickType_t
xTicksToWait );
//传入句柄，以及超时时间（队列写入的等待时间）
```

中断中使用：

```
BaseType_t  xTimerStopFromISR(  TimerHandle_t xTimer,
                                BaseType_t
*pxHigherPriorityTaskWoken );
//传入句柄，判断任务是否需要切换
```

复位定时器：

普通任务中使用：

```
BaseType_t xTimerReset( TimerHandle_t xTimer, TickType_t
xTicksToWait );
//传入句柄，以及超时时间（队列写入的等待时间
```

中断中使用：

```
BaseType_t xTimerResetFromISR(  TimerHandle_t xTimer,
                                BaseType_t
*pxHigherPriorityTaskWoken );
//传入句柄，判断任务是否需要切换
```

其他：

修改周期：

```
BaseType_t xTimerChangePeriod(  TimerHandle_t xTimer,
                                TickType_t xNewPeriod,
                                TickType_t xTicksToWait );
//xTimer: 哪个定时器
//xNewPeriod: 新周期
//xTicksToWait: 超时时间，命令写入队列的超时时间
BaseType_t xTimerChangePeriodFromISR(   TimerHandle_t xTimer,
                                        TickType_t xNewPeriod,
                                        BaseType_t
*pxHigherPriorityTaskWoken );
```

获取**ID**：

```
void *pvTimerGetTimerID( TimerHandle_t xTimer );
//xTimer: 哪个定时器
```

修改**ID**：

```
void vTimerSetTimerID( TimerHandle_t xTimer, void *pvNewID );
//xTimer: 哪个定时器
//pvNewID: 新 ID
```

肆：软件定时器的使用

！！使用**CubeMX**创建默认周期为**1**，需要自行修改，手动则不需要

实验：软件定时器的基本使用

创建两个定时器，一个周期位500ms,一个周期位1000ms

按下KEY1，定时器启动

按下KEY2，定时器暂停