**WCH TMOS详解**

**1.概述**

蓝牙为了实现同多个设备相连，实现多功能和多任务，产生了调度问题。虽然软件和协议栈可扩充，但终究最底层的执行部门只有一个。为了实现多事件和多任务切换，需要把事件和任务对应，针对这种应用起了一个TMOS名字操作系统抽象层。

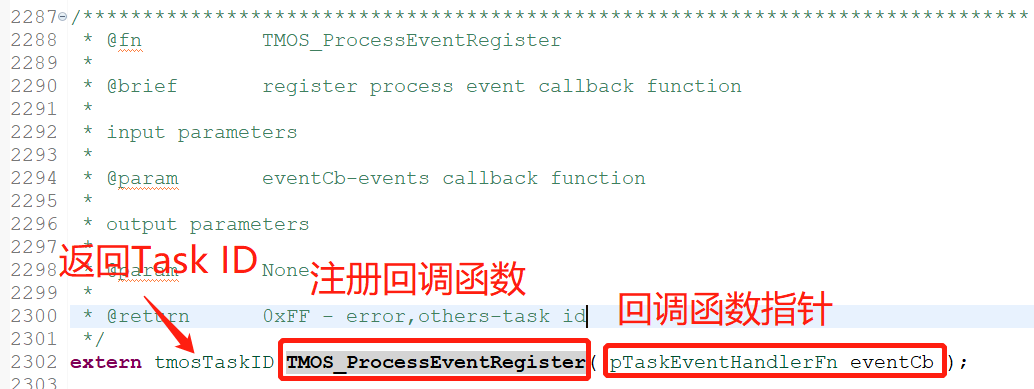
TMOS作为调度核心，BLE协议栈、profile定义、所有的应用都围绕它来实现。TMOS不是传统大家使用的操作系统，而是一个允许软件建立和执行事件的循环。

多任务管理方式实际上只有一个任务在运行，但是可以使用任务调度的策略将多个任务进行调度，每个任务占用一定的时间（**独占式，执行完当前任务退出**，继续查询其他可执行任务），所有的任务通过时间分片的方式处理。

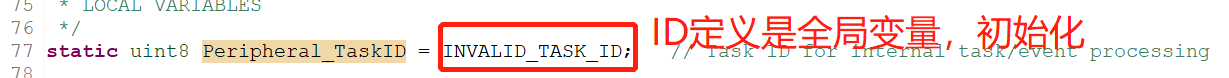
**TMOS 系统时钟单位为 625us**，以 RTC 为基准得到所有需要系统的时间。

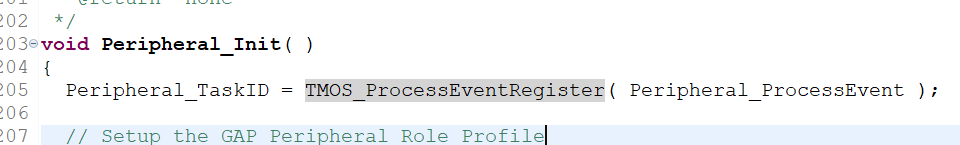
软件功能是由任务事件来实现的，创建一个任务事件需要以下工作：

1. 创建task identifier任务ID；

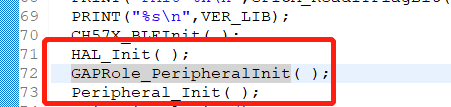


例如：

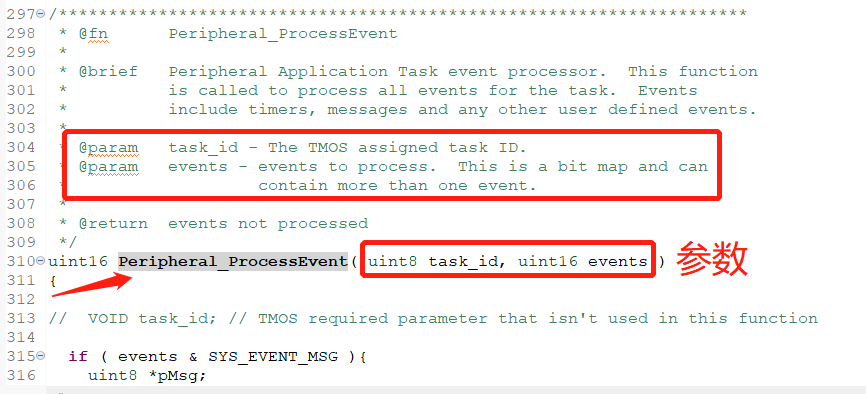




1. 编写任务初始化（task initialization routine）进程，并需要添加到TMOS初始化进程中，这就是说系统启动后不能动态添加功能(新的Task ID)；



1. 编写任务处理程序；

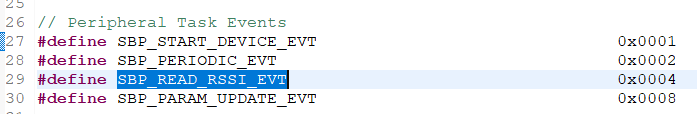


1. 定义任务事件，编写用户功能代码。

事件名按位定义，每一层taskID最多包含1个消息事件和15个任务事件（共16位）

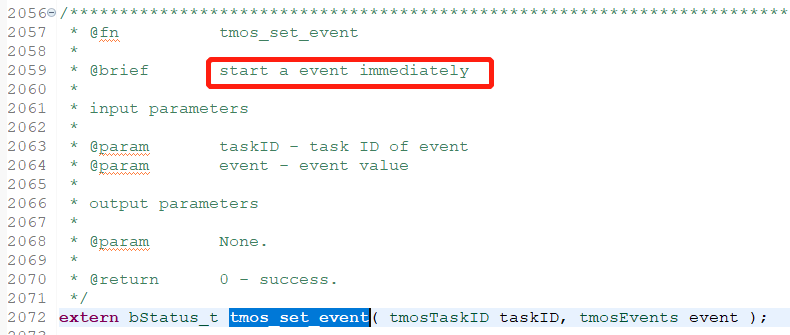
列如：

按位定义EVT任务事件，如下图：



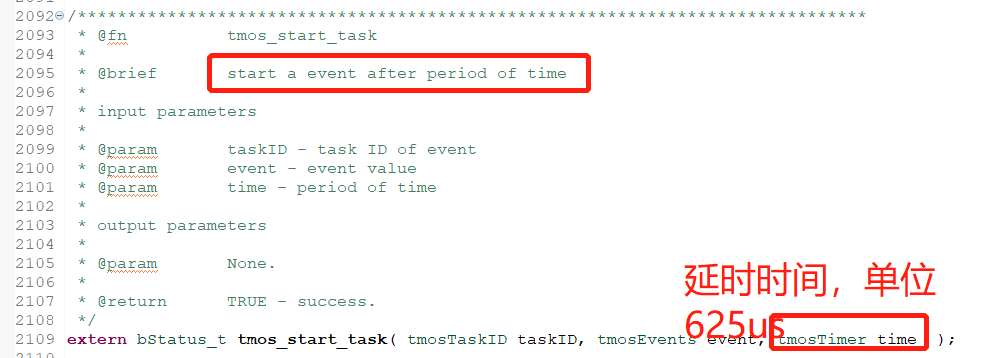
任务事件的启动有2种方式：（任务启动后只执行1次，如果重复执行，需要再重新开启任务）

1. 立即启动任务，调用后event时间立即执行

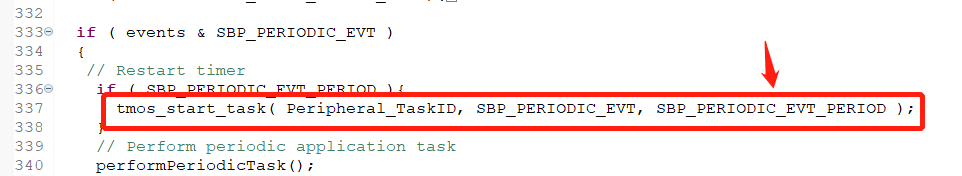


比如：  

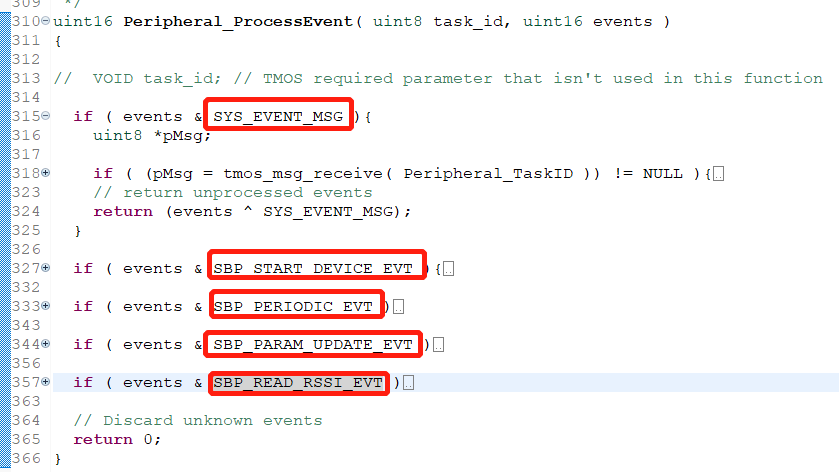
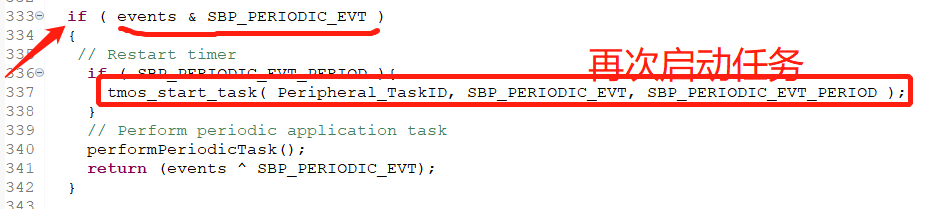

1. 设定延迟启动1个任务，从设定完成后开始计时



比如：Peripheral\_TaskID功能下的自定义的SBP\_PERIODIC\_EVT任务，延迟（SBP\_READ\_RSSI\_EVT\_PERIOD\*625）us后运行



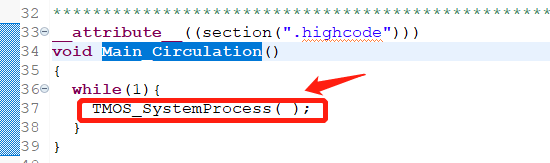
用户代码功能，前面有讲生成TaskID的时候需要向Tmos注册EVT处理函数指针，EVT执行条件满足后，Tmos就会自动调用该函数，如下图所示：



externbStatus\_ttmos\_stop\_task( tmosTaskIDtaskID, tmosEvents event );

此函数将停止一个会在taskID层生效的，名为event的任务，调用此函数后，该事件任务将不会生效。

5.主循环不停调用TMOS\_SystemProcess，查询可执行event事件；如果开始HAL\_SLEEP，芯片开启低功耗睡眠模式，Tmos会开启RTC唤醒功能，事件被执行前会自动唤醒，运行事件代码。

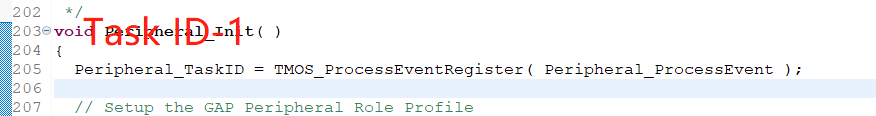
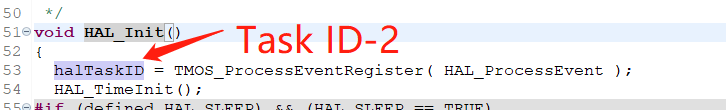


**任务调度函数使用注意事项：**

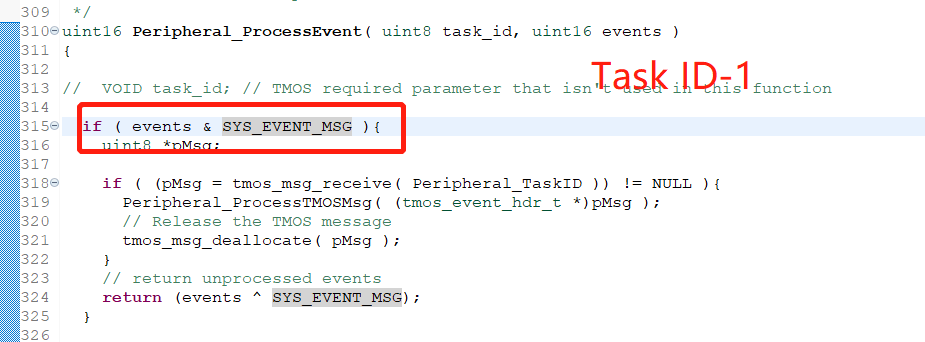
1. 禁止在中断中调用
2. 建议不要在单个任务中执行超过连接间隔一半时长的任务，否则将影响蓝牙通讯
3. 同理，在中断中建议不要执行超过连接间隔一半时长的任务，否则将影响蓝牙通讯
4. 在事件生效执行的代码中调用延时执行函数时，延时时间以当前事件生效时间点为基准偏移，所以对调用延时执行函数在生效执行的代码中摆放的位置没有要求。
5. 任务存在优先级，根据在xxx\_ProcessEvent函数中判断的先后顺序决定，同时生效的任务，先执行先判断，后执行后判断。注意，执行完先判断的事件任务后，要等到任务调度系统轮巡一遍后，才会执行后判断的事件任务。
6. 事件名按位定义，每一层taskID最多包含1个消息事件和15个任务事件（共16位）

前面讲了1个Task ID的应用情况，为了降低C文件或者功能之间的耦合，一般比较好的做法是把同功能或者相近的event放到同一个Task ID下，这样就产生了1个问题，不同的Task ID可能会有数据需要交互，Tmos提供了不同Task ID之间进行数据交互的函数。

比如：外设里面拿其中两个Task ID 举例子halTaskID和Peripheral\_TaskID，假设这两个Task之间要进行数据交互。

之前有讲每个Task ID都有1个消息event



  
上图演示的是接收消息，接收主要用到2个函数  
