FreeRTOS学习笔记（八）

FreeRTOS软件定时器

- 软件定时器，定时执行指定的功能函数（回调函数）。注意：不要在回调函数中使用任何阻塞任务的API
- 定时器它是由定时器服务任务来提供的。大多数定时器API函数都是使用FreeRTOS的队列发送命令给定时服务任务。

    - 单次定时器和周期定时器
        - 单次定时器：启动之后运行一次对应的回调函数就关闭定时器
        - 周期定时器：按照制定好的时间周期，执行回调函数
    - 复位软件定时器
        - 简单来说，就是重新开始定时
    - 定时器相关的API函数

```
/**
* 描述：复位软件定时器
* 参数：xTimer       ：需要复位的定时器句柄
*       xTicksToWait：向定时器命令队列发送命令，阻塞时间
* 返回：pdFAIL：复位成功
*       pdPASS：复位失败
*/
BaseType_t xTimerReset(TimerHandle_t xTimer, TickType_t xTicksToWait);

/**
* 描述：创建软件定时器
* 参数：pcTimerName         ：软件定时器名字（字符串）
*       xTimerPeriodInTicks：定时周期（节拍数）
*       uxAutoReload        ：定时器模式（单次或者周期）
*       pvTimerID           ：定时器的ID（公用回调函数时用到）
*       pxCallbackFunction  ：回调函数
* 返回：NULL：创建失败
*       其他值：定时器句柄
*/
TimerHandle_t xTimerCreate(const char * const      pcTimerName,
                           const TickType_t        xTimerPeriodInTicks,
                           const UBaseType_t       uxAutoReload,
                           void * const            pvTimerID,
                           TimerCallbackFunction_t pxCallbackFunction);

/**
* 描述：开启定时器
* 参数：xTimer       ：定时器句柄
*       xTicksToWait：阻塞时间
* 返回：pdFAIL：开启失败
*       pdPASS：开启成功
```

```
*/
BaseType_t xTimerStart(TimerHandle_t xTimer, TickType_t xTicksToWait);

/**
* 描述：停止定时器
* 参数：xTimer       ：定时器句柄
*       xTicksToWait：阻塞时间
* 返回：pdFAIL：停止失败
*       pdPASS：停止成功
*/
BaseType_t xTimerStop(TimerHandle_t xTimer, TickType_t xTicksToWait);
```

FreeRTOS事件标志组

- 事件组：就是有很多个事件位（bit1表示队列是否有消息，bit2表示网络是否有数据等）
- configUSE_16_BIT_TICKS 为1的时候一个组就8个事件，为0的时候一个组就24个事件。

```
/**
* 描述：创建时间标志组
*       configUSE_16_BIT_TICKS == 1  （0~7）
*       configUSE_16_BIT_TICKS == 0  （0~23）
* 返回：NULL：创建失败
*       其他值：创建的时间标志组的句柄
*/
EventGroupHandle_t xEventGroupCreate(void);

/**
* 描述：指定事件位清零
* 参数：xEventGroup  ：事件标志组的句柄
*       uxBitsToClear：要清零的事件位
* 返回：将指定事件位清零之前的事件组值
*/
EventBits_t xEventGroupClearBits(EventGroupHandle_t xEventGroup,
                                 const EventBits_t  uxBitsToClear);

/**
* 描述：获取当前事件组的值（也就是清除所有事件位）
* 参数：xEventGroup：要获取事件标志组的句柄
* 返回：当前事件标志组的值
*/
EventBits_t xEventGroupGetBits(EventGroupHandle_t xEventGroup);

/**
* 描述：设置相应的事件位
* 参数：xEventGroup：事件标志组句柄
*       uxBitsToSet：要设置的事件位
* 返回：指定事件位置1后的事件组值
*/
```

```
EventBits_t xEventGroupSetBits(EventGroupHandle_t xEventGroup,
                               const EventBits_t  uxBitsToSet);

/**
* 描述：等待多个事件位
* 参数：xEventGroup     ：要等待的事件标志组
*       uxBitsToWaitFor：要等待的事件位
*       xClearOnExit    ：是否清零（pdTRUE，pdFALSE）
*       xWaitForAllBits：pdTRUE的话就要uxBitsToWaitFor设置的事件位都置1
*                        pdFALSE的话就随便一个置1
*       xTicksToWait    ：阻塞时间
* 返回：返回当所等待的事件位置1以后的事件标志组的值
*/
EventBits_t xEventGroupWaitBits(EventGroupHandle_t xEventGroup,
                                const EventBits_t  uxBitsToWaitFor,
                                const BaseType_t   xClearOnExit,
                                const BaseType_t   xWaitForAllBits,
                                TickType_t         xTicksToWait);
```