FreeRTOS 封装后的 API

内核控制函数

```
osStatus osKernelStart (void);                    //内核开始运行
int32_t osKernelRunning(void);                    //返回值为 1 表示正在运行  作用：系统是否正常工作
uint32_t osKernelSysTick (void);                  //系统当前节拍数    作用：毫秒级计时
osKernelSysTickMicroSec(microsec)                 //微妙（宏函数）    作用：微秒级计时
```

线程管理函数

```
osThreadDef(name, thread, priority, instances, stacksz)    //(宏函数) 作用：定义 osThreadDef_t 结构体
osThread(name)                                    //(宏函数)      作用：获取 os_thread_def_##name 结构体的指针
osThreadId osThreadCreate (const osThreadDef_t *thread_def, void *argument);  //创建线程，错误返回 NULL
osThreadId osThreadGetId (void);                  //返回当前线程
osStatus osThreadTerminate (osThreadId thread_id); //终结线程
osStatus osThreadYield (void);                    //调度一次
osStatus osThreadSetPriority (osThreadId thread_id, osPriority priority);      //设置线程优先级
osPriority osThreadGetPriority (osThreadId thread_id);     //获取线程优先级
```

等待函数

```
osStatus osDelay (uint32_t millisec);     //毫秒级延时
osEvent osWait (uint32_t millisec);       //未实现
```

信号量管理函数

```
osSemaphoreDef(name)              //（宏函数） 作用：定义 os_semaphore_def_##name 结构体
osSemaphore(name)                 //（宏函数） 作用：定义 os_semaphore_def_##name 结构体的指针
osSemaphoreId osSemaphoreCreate (const osSemaphoreDef_t *semaphore_def, int32_t count);    //创建信号量
int32_t osSemaphoreWait (osSemaphoreId semaphore_id, uint32_t millisec);      //信号量减一
osStatus osSemaphoreRelease (osSemaphoreId semaphore_id);   //信号量加一
osStatus osSemaphoreDelete (osSemaphoreId semaphore_id);    //删除信号量
```

互斥锁管理函数

```
osMutexDef(name)                  //（宏函数）      作用：定义 os_mutex_def_##name 结构体
osMutex(name)                     //宏函数       作用：定义 os_mutex_def_##name 结构体的指针
osMutexId osMutexCreate (const osMutexDef_t *mutex_def);   //创建互斥锁
osStatus osMutexWait (osMutexId mutex_id, uint32_t millisec); //锁定互斥锁
osStatus osMutexRelease (osMutexId mutex_id);               //解锁互斥锁
osStatus osMutexDelete (osMutexId mutex_id);                //删除互斥锁
```

队列管理函数

osMessageQDef(name, queue_sz, type) //参数二：队列 size 参数三：成员 size 作用：定义 os_messageQ_def_##name 结构体

osMessageQ(name) //宏函数 作用：定义 os_messageQ_def_##name 结构体的指针

osMessageQId osMessageCreate (const osMessageQDef_t *queue_def, osThreadId thread_id); //创建队列

osStatus osMessagePut (osMessageQId queue_id, uint32_t info, uint32_t millisec); //向队列中放入消息

osEvent osMessageGet (osMessageQId queue_id, uint32_t millisec); //从队列中获取消息

邮箱队列管理函数

osMailQDef(name, queue_sz, type) //参数二：邮箱 size 参数三：消息 size 作用：定义 os_mailQ_def_##name 结构体

osMailQ(name) //宏函数 作用：定义 os_mailQ_def_##name 结构体的指针

osMailQId osMailCreate (const osMailQDef_t *queue_def, osThreadId thread_id); //创建邮箱

void *osMailAlloc (osMailQId queue_id, uint32_t millisec); //分配一个消息的空间

void *osMailCAlloc (osMailQId queue_id, uint32_t millisec); //分配一个消息的空间并且清零

osStatus osMailPut (osMailQId queue_id, void *mail); //将成员丢到邮箱中

osEvent osMailGet (osMailQId queue_id, uint32_t millisec); //从邮箱中获取消息

osStatus osMailFree (osMailQId queue_id, void *mail); //释放消息空间

软件定时器管理函数

osTimerDef(name, function) //宏函数 作用：定义 os_timer_def_##name 结构体

osTimer(name) //宏函数 作用：定义 os_timer_def_##name 结构体的指针

osTimerId osTimerCreate (const osTimerDef_t *timer_def, os_timer_type type, void *argument); //参数二：是否重复回调 作用：创建软件定时器

osStatus osTimerStart (osTimerId timer_id, uint32_t millisec); //开始定时

osStatus osTimerStop (osTimerId timer_id); //停止定时

osStatus osTimerDelete (osTimerId timer_id); //删除定时器

信号管理函数

int32_t osSignalSet (osThreadId thread_id, int32_t signals); //带通知值的任务通知

int32_t osSignalClear (osThreadId thread_id, int32_t signals); //未实现

osEvent osSignalWait (int32_t signals, uint32_t millisec); //参数二：取出值后位置零 作用：等待通知到来

内存池管理函数

osPoolDef(name, no, type) //参数二：内存池 size 参数三：成员 size 作用：定义 os_pool_def_##name 结构体

osPool(name) //宏函数 作用：定义 os_pool_def_##name 结构体的指针

osPoolId osPoolCreate (const osPoolDef_t *pool_def); //创建内存池

void *osPoolAlloc (osPoolId pool_id); //分配内存

void *osPoolCAlloc (osPoolId pool_id); //分配内存并且清零

osStatus osPoolFree (osPoolId pool_id, void *block); //释放内存

特殊功能函数

```
void osSystickHandler(void);                                    //定时器中断处理函数
osThreadState osThreadGetState(osThreadId thread_id);           //获取线程状态
osStatus osThreadIsSuspended(osThreadId thread_id);             //线程是否被挂起
osStatus osThreadSuspend (osThreadId thread_id);                //挂起线程
osStatus osThreadResume (osThreadId thread_id);                 //恢复调度
osStatus osThreadSuspendAll (void);                             //挂起所有线程
osStatus osThreadResumeAll (void);                              //恢复调度所有线程
osStatus osDelayUntil (uint32_t *PreviousWakeTime, uint32_t millisec);   //绝对延时
osStatus osThreadList (uint8_t *buffer);                        //获取任务信息 任务名、优先级
osEvent osMessagePeek (osMessageQId queue_id, uint32_t millisec);        //瞥一眼队列中有没有消息
osMutexId osRecursiveMutexCreate (const osMutexDef_t *mutex_def);        //创建递归互斥锁
osStatus osRecursiveMutexRelease (osMutexId mutex_id);          //释放递归互斥锁
osStatus osRecursiveMutexWait (osMutexId mutex_id, uint32_t millisec);   //锁定递归互斥锁
```