FreeRTOS学习笔记（三）

FreeRTOS任务的相关基础知识

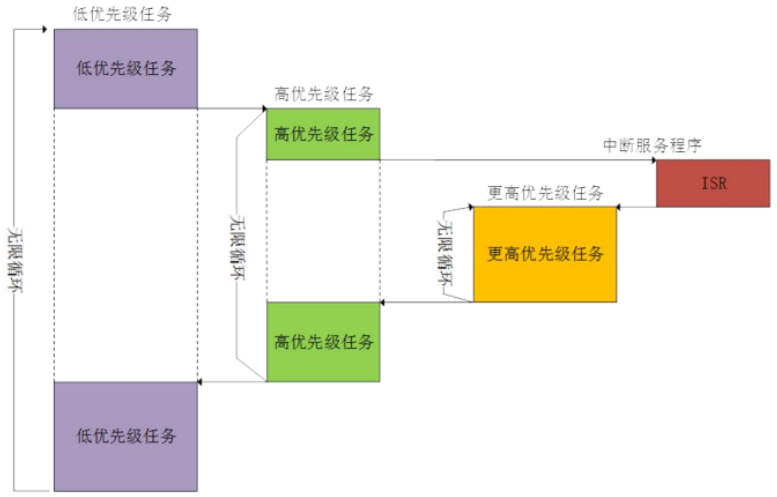

图 5.1.2 抢占式多任务系统

1. 任务的状态
   - 运行态：正在使用处理器的任务。
   - 就绪态：（这些任务没有被阻塞或者挂起）可以运行的任务。
   - 阻塞态：正在等待某些信号量，队列，事件组，通知等，或者是调用了vTaskDelay（）。
   - 挂起态：不能进入运行态。vTaskSuspend（），vTaskResume（）两个函数将任务挂起和解挂。

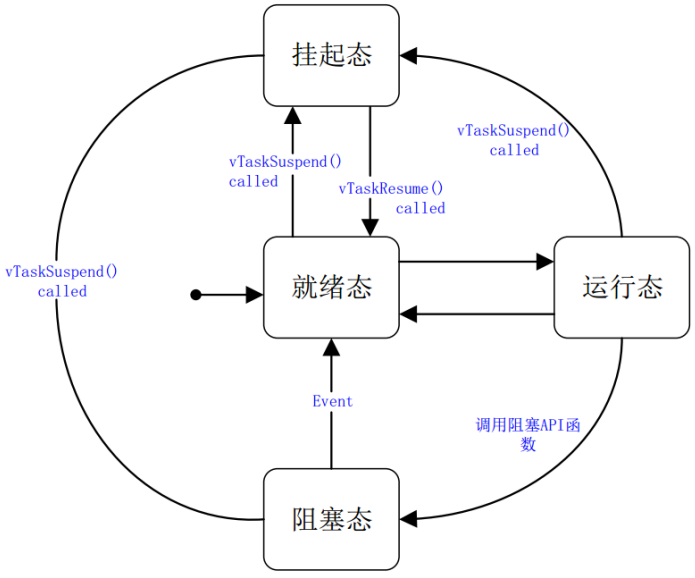

图 5.4.1 任务状态之间的转换

2. 任务的优先级
   - 任务优先级数字越小，优先级越低（和UCOS相反）
   - 只要宏configUSE_TIME_SLICING 被设置成1，则启用时间片轮转调度（同一优先级的任务，平均每个任务运行一段相同的时间）

3. 任务堆栈
   - 用于在任务调度的时候保护现场。xTaskCreate（）自动创建堆栈，xTaskCreateStatic（）需要自行定义任务堆栈。

FreeRTOS任务任务相关的API

```
/**
* 描述：动态创建任务
* 返回：pdPASS创建成功
*       errCOULD_NOT_ALLOCATE_REQUIRED_MEMORY创建失败
*/
BaseType_t xTaskCreate(TaskFunction_t       pxTaskCode,       /* 任务函数的函数名 */
                       const char * const   pcName,           /* 任务的名称（自己定义） */
                       const uint16_t       usStackDepth,     /* 任务堆栈的大小 */
                       void * const         pvParameters,     /* 传递给任务函数的参数 */
                       UBaseType_t          uxPriority,       /* 任务优先级 */
                       TaskHandle_t * const pxCreatedTask);   /* 任务句柄 */

/**
* 描述：静态创建任务
* 返回：非空就是创建成功
*       NULL创建失败
*/
TaskHandle_t xTaskCreateStatic( TaskFunction_t       pxTaskCode,    /* 任务函数的函数名 */
                                const char * const   pcName,        /* 任务的名称（自己定义） */
                                const uint32_t       ulStackDepth,  /* 任务堆栈的大小 */
                                void * const         pvParameters,  /* 传递给任务函数的参数 */
                                UBaseType_t          uxPriority,    /* 任务优先级 */
                                StackType_t * const  puxStackBuffer,/* 任务堆栈 */
                                StaticTask_t * const pxTaskBuffer)  /* 任务控制块 */

/**
* 描述：动态创建任务（要求MCU有MPU，也就是内存保护单元）
* 返回：pdPASS创建成功
*       其他值就是创建失败
*/
BaseType_t xTaskCreateRestricted(const TaskParameters_t * const pxTaskDefinition, /* 拥有任务属性的结构体 *
                                 TaskHandle_t*                 pxCreatedTask)    /* 任务句柄 */

/**
* 描述：删除任务
*/
vTaskDelete(TaskHandle_t xTaskToDelete)/* 要删除的任务句柄 */

/**
* 描述：挂起任务
* 参数：任务句柄，动态创建任务，那就是pxCreatedTask，静态创建就是创建函数的返回值是任务句柄
*       如果是NULL的话就是挂起任务自己
*/
void vTaskSuspend(TaskHandle_t xTaskToSuspend)

/**
* 描述：回复任务
*/
void vTaskResume(TaskHandle_t xTaskToResume)  /* 要恢复任务的任务句柄 */
```

```
/**
* 描述：中断服务函数中恢复一个任务
* 返回：pdTRUE  恢复的任务的优先级高于或等于当前任务，退出中断服务函数必须调度一次
*       pdFALSE 恢复的任务的优先级低于当前任务，退出中断服务函数不用调度
*/
BaseType_t xTaskResumeFromISR(TaskHandle_t xTaskToResume)  /* 要恢复任务的任务句柄 */
```