- FreeRTOS任务创建与删除
  - 任务创建和删除的API函数解析
    - 1、动态创建任务其内部实现
      - 1、 申请堆栈内存（返回首地址）
      - 2、 申请任务控制块内存（返回首地址）
      - 3、 把前面申请的堆栈地址，赋值给控制块的堆栈成员
      - 4、 调用prvInitialiseNewTask 初始化任务控制块中的成员
        - 1、初始化堆栈为0xa5（可选）
        - 2、记录栈顶，保存在pxTopOfStack
        - 3、保存任务名字到pxNewTCB->pcTaskName[x]中
        - 4、保存任务优先级到pxNewTCB->uxPriority
        - 5、设置状态列表项的所属控制块，设置事件列表项的值
        - 6、列表项的插入是从小到大插入，所以这里将越高优先级的任务他的事件列表项值设置越小，这样就可以拍到前面
        - 7、调用pxPortInitialiseStack初始化任务堆栈，用于保存当前任务上下文寄存器信息，已备后续任务切换使用
        - 8、将任务句柄等于任务控制块
      - 5、 调用prvAddNewTaskToReadyList 添加新创建任务到就绪列表中
        - 1、记录任务数量uxCurrentNumberOfTasks++
        - 2、判断新创建的任务是否为第一个任务
          - 如果创建的是第一个任务，初始化任务列表prvInitialiseTaskLists
          - 如果创建的不是第一个任务，并且调度器还未开始启动，比较新任务与正在执行的任务优先级大小，新任务优先级大的话，将当前控制块重新指向新的控制块
        - 3、将新的任务控制块添加到就绪列表中，使用函数prvAddTaskToReadyList
          - 将uxTopReadyPriority相应bit置一，表示相应优先级有就绪任务，比如任务优先级为5，就将该变量的位5置一，方便后续任务切换判断，对应的就绪列表是否有任务存在
          - 将新创建的任务插入对应的就绪列表末尾
        - 4、如果调度器已经开始运行，并且新任务的优先级更大的话，进行一次任务切换
    - 2、删除任务的内部实现
      - 1、获取所要删除任务的控制块　　通过传入的任务句柄，判断所需要删除哪个任务，NULL代表删除自身
      - 2、将被删除任务，移除所在列表　　将该任务在所在列表中移除，包括：就绪、阻塞、挂起、事件等列表
      - 3、判断所需要删除的任务
        - 删除任务自身，需先添加到等待删除列表，内存释放将在空闲任务执行
        - 删除其他任务，当前任务数量--　　更新下一个任务的阻塞超时时间，以防被删除的任务就是下一个阻塞超时的任务
      - 4、删除的任务为其他任务则直接释放内存prvDeleteTCB( )
      - 5、调度器正在运行且删除任务自身，则需要进行一次任务切换
    - 3、静态创建任务其内部实现
      - 1、获取控制块内存（首地址）
      - 2、获取堆栈内存（首地址）
      - 3、标记使用的静态的方式申请的TCB和堆栈内存
      - 4、调用prvInitialiseNewTask 初始化任务块，并将控制块信息返回给任务句柄，以便后续返回句柄信息
      - 5、调用prvAddNewTaskToReadyList 添加新创建任务到就绪列表中
  - 1，任务创建和删除的API函数（熟悉）
    - 任务的创建和删除本质就是调用FreeRTOS的API函数
    - 动态方式创建任务：xTaskCreate()
    - 静态方式创建任务：xTaskCreateStatic()
    - 删除任务：vTaskDelete()
    - 动态静态创建区别
      - 动态创建任务：任务的任务控制块以及任务的栈空间所需的内存，均由FreeRTOS 从 FreeRTOS 管理的堆中分配
      - 静态创建任务：任务的任务控制块以及任务的栈空间所需的内存，需用户分配提供
    - 实现动态创建任务流程
      - 使用动态创建任务，需将宏configSUPPORT_DYNAMIC_ALLOCATION 配置为 1
      - 1、定义函数入口参数
      - 2、编写任务函数
    - 静态创建任务流程
      - 使用静态创建任务，需将宏configSUPPORT_STATIC_ALLOCATION 配置为 1
      - 1、定义空闲任务&定时器任务的任务堆栈及TCB
      - 2、实现两个接口函数
        - vApplicationGetIdleTaskMemory( )
        - vApplicationGetTimerTaskMemory ( )
      - 3、编写任务函数
    - 删除任务流程
      - 使用删除任务函数，需将宏INCLUDE_vTaskDelete 配置为 1
      - 1、当传入的参数为NULL，则代表删除任务自身（当前正在运行的任务）
      - 2、空闲任务会负责释放被删除任务中由系统分配的内存，但是由用户在任务删除前申请的内存， 则需要由用户在任务被删除前提前释放，否则将导致内存泄露
  - 2，任务创建和删除（动态方法）（掌握）
    - 1、实验目的：学会 xTaskCreate( ) 和 vTaskDelete( ) 的使用
    - 2、将设计四个任务：start_task、task1、task2、task3
      - start_task：用来创建其他的两个任务
      - task1：实现LED0每500ms闪烁一次
      - task2：实现LED1每500ms闪烁一次
      - task3：判断按键KEY0是否按下，按下则删掉task1
  - 3，任务创建和删除（静态方法）（掌握）
    - 1、实验目的：学会 xTaskCreateStatic( ) 和 vTaskDelete( ) 的使用
    - 2、将设计四个任务：start_task、task1、task2、task3
      - start_task：用来创建其他的两个任务
      - task1：实现LED0每500ms闪烁一次
      - task2：实现LED1每500ms闪烁一次
      - task3：判断按键KEY0是否按下，按下则删掉task1
  - 4、需注意
    - 1、在实际的应用中，动态方式创建任务是比较常用的，除非有特殊的需求，一般都会使用动态方式创建任务
    - 2、动态创建相对简单，更为常用
    - 3、静态创建：可将任务堆栈放置在特定的内存位置，并且无需关心对内存分配失败的处理
    - 4、临界区保护，保护那些不想被打断的程序段，关闭freertos所管理的中断，中断无法打断，滴答中断和PendSV中断无法进行不能实现任务调度