任务通知

任务通知基本概念

- FreeRTOS 从 V8.2.0 版本开始提供任务通知这个功能，每个任务都有一个 32 位的通知值，在大多数情况下，任务通知可以替代二值信号量、计数信号量、事件组， 也可以替代长度为 1 的队列（可以保存一个 32 位整数或指针值）
- 相对于以前使用 FreeRTOS 内核通信的资源，必须创建队列、二进制信号量、计数信号量或事件组的情况，使用任务通知显然更灵活。 按照 FreeRTOS 官方的说法，使用任务通知比通过信号量等 ICP 通信方式解除阻塞的任务要快 45%，并且更加省 RAM 内存空间（使用 GCC 编译器， -o2 优化级别）
- 任务通知的使用无需创建队列。 想要使用任务通知，必须将 FreeRTOSConfig.h 中的宏定义 configUSE_TASK_NOTIFICATIONS设置为 1

任务通知的优劣势

- 优势
  - 发送通知给任务，如果有通知未读，不覆盖通知值
  - 发送通知给任务，直接覆盖通知值
  - 发送通知给任务，设置通知值的一个或者多个位 ，可以当做事件组来使用
  - 发送通知给任务，递增通知值，可以当做计数信号量使用
- 劣势
  - 只能有一个指定接收通知的任务接收通知消息
  - 只有等待通知的任务可以被阻塞，发送通知的任务，在任何情况下都不会因为发送失败而进入阻塞态(消息完整正确性无法保障)

任务通知的运作机制

- 任务通知是属于任务中附带的资源， 所以在任务被创建的时候，任务通知也被初始化的
- 任务通知的数据结构包含在任务控制块中，只要任务存在，任务通知数据结构就已经创建完毕，可以直接使用
- 任务通知可以在任务中向指定任务发送通知，也可以在中断中向指定任务发送通知。 FreeRTOS 的每个任务都有一个 32 位的通知值，任务控制块中的成员变量 uINotifiedValue就是这个通知值
- 只有在任务中可以等待通知，而不允许在中断中等待通知
- 如果任务在等待的通知暂时无效，任务会根据用户指定的阻塞超时时间进入阻塞状态，我们可以将等待通知的任务看作是消费者；其它任务和中断可以向等待通知的任务发送通知，发送通知的任务和中断服务函数可以看作是生产者，当其他任务或者中断向这个任务发送任务通知，任务获得通知以后，该任务就会从阻塞态中解除

任务通知的数据结构

```
#if( configUSE_TASK_NOTIFICATIONS == 1 )
    volatile uint32_t ulNotifiedValue;
    volatile uint8_t ucNotifyState;
#endif
```

任务通知的函数接口

- 发送任务通知函数 xTaskGenericNotify()
  - xTaskNotifyGive()
    - xTaskNotifyGive()是一个宏，它实开是调用函数 xTaskNotify((xTaskToNotify), (0), eIncrement ）， 即向一个任务发送通知，并将对方的任务通知加加 1
    - 在这种情况下对象任务在等待任务通知的时候应该是使用函数ulTaskNotifyTake()
    - #define xTaskNotifyGive(xTaskToNotify)
      xTaskGenericNotify((xTaskToNotify), (0), eIncrement, NULL)
  - vTaskNotifyGiveFromISR()
    - vTaskNotifyGiveFromISR()是 vTaskNotifyGive()的中断保护版本。 用于在中断中向指定任务发送任务通知，并更新对方的任务通知值（加 1 操作）
    - void vTaskNotifyGiveFromISR(TaskHandle_t xTaskToNotify, BaseType_t* pxHigherPriorityTaskWoken);
    - vTaskNotifyGiveFromISR()函数作用， 每次调用该函数都会增加任务的通知值， 任务通过接收函数返回值是大于零，判断是否获取到了通知，任务通知值初始化为 0， （如果与信号量做对比）则对应为信号量无效。当中断调用vTaskNotifyGiveFromISR()通知函数给任务的时候，任务的通知值增加，使其大于零，使其表示的通知值变为有效，任务获取有效的通知值将会被恢复。
  - xTaskNotify()
    - xTaskNotify()用于在任务中直接向另外一个任务发送一个事件， 接收到该任务通知的任务有可能解锁
    - xTaskNotify()函数在发送任务通知的时候会指定一个通知值， 并且用户可以指定通知值发送的方式
    - BaseType_t xTaskNotify(
      TaskHandle_t xTaskToNotify,
      uint32_t uiValue,
      eNotifyAction eAction);
  - xTaskNotifyAndQuery()
    - xTaskNotifyAndQuery()与 xTaskNotify()很像，都是调用通用的任务通知发送函数 xTaskGenericNotify() 来实现通知的发送
    - 不 同 的 是 多 了 一 个 附 加 的 参 数pulPreviousNotifyValue 用于获得接收任务的上一个通知值
    - BaseType_t xTaskNotifyAndQuery(
      TaskHandle_t xTaskToNotify,
      uint32_t uiValue,
      eNotifyAction eAction,
      uint32_t *pulPreviousNotifyValue)
- 中断中发送任务通知通用函数 xTaskGenericNotifyFromISR()
  - xTaskNotifyFromISR()
    - xTaskNotifyFromISR()是 xTaskNotify()的中断保护版本。真正起作用的函数是中断发送任务通知通用函数 xTaskGenericNotifyFromISR()
    - 用于在中断中向指定的任务发送一个任务通知， 该任务通知是带有通知值并且用户可以指定通知的发送方式， 不返回上一个任务的通知值
    - BaseType_t xTaskNotifyFromISR(TaskHandle_t xTaskToNotify,
      uint32_t uiValue,
      eNotifyAction eAction,
      BaseType_t *pxHigherPriorityTaskWoken);
  - xTaskNotifyAndQueryFromISR()
    - xTaskNotifyAndQueryFromISR()是 xTaskNotifyAndQuery ()的中断版本，用于向指定的任务发送一个任务通知， 并返回对象任务的上一个通知值
    - 该函数也是一个宏定义， 真正实现发送通知的是 xTaskGenericNotifyFromISR()
    - BaseType_t xTaskNotifyAndQueryFromISR(
      TaskHandle_t xTaskToNotify,
      uint32_t uiValue,
      eNotifyAction eAction,
      uint32_t *pulPreviousNotifyValue,
      BaseType_t *pxHigherPriorityTaskWoken)
- 获取任务通知函数
  (获取任务通知函数只能用在任务中，没有带中断保护版本)
  - ulTaskNotifyTake()
    是为代替二值信号量和计数信号量而专门设计的，它和发送通知API 函数 xTaskNotifyGive()， vTaskNotifyGiveFromISR()配合使用
    - 当一个任务使用其自身的任务通知操作为二值信号量或者计数信号量时， 其他任务应该使用函数 xTaskNotifyGive()(或者 xTaskNotify((xTaskToNotify), (0), eIncrement)来向其发送信号量
    - xTaskNotifyTake()在退出的时候处理任务的通知值的时候有两种方法，一种是在函数退出时将通知值清零，这种方法适用于实现二值信号量；另外一种是在函数退出时将通知值减 1，这种方法适用于实现计数信号量
    - uint32_t ulTaskNotifyTake(BaseType_t xClearCountOnExit,
      TickType_t xTicksToWait)
      xClearCountOnExit设置为 pdFALSE 时，函数 xTaskNotifyTake()退出后，将任务的通知值减 1，可以用来实现计数信号量；设置为 pdTRUE 时，函数xTaskNotifyTake()退出后，将任务通知值清零，可以用来实现二值信号量。
      xTicksToWait最的时间， 单位为系统节拍周期。宏 pdMS_TO_TICKS 用于将毫秒转化为系统节拍数
      设定任务的当前通知值， 在其减 1 或者清 0 之前
    - BaseType_t xTaskNotifyWait(
      uint32_t uiBitsToClearOnEntry,
      uint32_t uiBitsToClearOnExit,
      uint32_t *pulNotificationValue,
      TickType_t xTicksToWait);
  - xTaskNotifyWait()
    是全功能版的等待通知，可以根据不同的参数实现轻量级二值信号量、计数信号量、事件组和长度为 1 的队列
    - uiBitsToClearOnEntry 表示在使用通知之前，将任务通知值的哪些位清0. 实现过程就是将任务的通知值与参数uiBitsToClearOnEntry的按位取反值按位与操作。如果 uiBitsToClearOnEntry设置为0x01,那么在函数进入前,任务通知值的位0会被清0. 其 他 位 保 持 不 变 .如 果uiBitsToClearOnEntry 设置为 0xFFFFFFFF (ULONG_MAX). 那么在进入函数前任务通知值的所有位都会被清 0， 表示清零任务通知值
    - uiBitsToClearOnExit 表示在函数 xTaskNotifyWait()退出前,决定任务接收到的通知值的哪些位会被清 0.实现过程就是将任务的通知值与参数 uiBitsToClearOnExit 的按位取反值按位与操作.在清 0 前,接收到的任务通知值会先被保存到形参*pulNotificationValue 中.如果 uiBitsToClearOnExit 设置为 0x03,那么在函数退出前,接收到的任务通知值的位 0 和位 1 会被清0.其他位保持不变.如果uiBitsToClearOnExit设置为0xFFFFFFFF(ULONG_MAX),那么在退出函数前接收到的任务通知值的所有位都会被清 0.表示退出时清零任务通知值
    - pulNotificationValue用于保存接收到的任务通知值.如果接收到的任务通知不需要使用 则设置为NULL即可.这个通知值形参数uiBitsToClearOnExit起作用前将通知值拷贝到 *pulNotificationValue 中
    - xTicksToWait等待超时时间,单位为系统节拍周期.宏pdMS_TO_TICKS用于将单位毫秒转化为系统节拍数