freeRTOS在stm32cube上的参数介绍

1. config parameters
   1. kernel settings

**USE\_PREEMPITION**---------------preemption(抢占)是否使用方式

**TICK\_RATE\_HZ**------------------设置tick中断的频率，也是线程调度的频率。

**MAX\_PRIORITIES**----------------设置最大的优先级。可以设置的优先级0（最小）到（MAX\_PRIORITIES-1）（最大。）

**MINIMAL\_STACK\_SIZE**------------设置分配给空闲任务的堆栈大小.

max value = configTOTAL\_HEAP\_SIZE/4 (when allocation is dynamic).

max value = MCU ram size/4 (when allocation is static).

**MAX\_TASK\_NAME\_LEN**-------------设置最大的任务的名字，包括NULL终止符包含在字符数中。

**configUSE\_16\_BIT\_TICKS** -------16位的心跳数，更好的支持16位和8位MCU

**IDLE\_SHOULD\_YIELD**-------------只有在抢占式的系统中有效。设置为1，在空闲任务中花费的最短时间。

**USE\_MUTEXES** ------------------设置1，在代码构建中包含互斥的功能。

**USE\_RECURSIVE\_MUTEXES**---------设置1，在代码构建中包含递归互斥体功能。允许两个值。

**USE\_COUNTING\_SEMAPHORES** ------设置1，在代码构建中包含计数信号量功能。

**QUEUE\_REGISTRY\_SIZE** ----------队列注册表有两个目的，两者都与RTOS内核感知调试相关联。它允许将文本名称与队列相关联，以便在调试GUI中轻松地进行队列识别。它包含调试器查找每个注册队列和信号量所需的信息。 队列注册表没有目的，除非您使用RTOS内核感知调试器。

**USE\_APPLICATION\_TASK\_TAG** ----- USE\_APPLICATION\_TASK\_TAG must be defined as 1 for the vTaskSetApplicationTaskTag function to be available. 只对高级用户。

1，可以将标签分配给每个任务（此值仅用于应用程序 - RTOS内核本身不会以任何方式使用它）。

2. FreeRTOS跟踪宏文档页面提供了一个应用程序如何使用此功能的良好示例。

**ENABLE\_BACKWARD\_COMPATIBILITY** --这些宏允许应用程序代码将它们所构建的FreeRTOS版本从8.0.0版本更新到8.0.0版本，而无需修改。

**USE\_PORT\_OPTIMISED\_TASK\_SELECTION** --- Forced to 1 (Enabled) with Freertos v9 。fressrtos有两种方法来设置任务调度，设置1为特别的方法。0为普通的方法。

**USE\_TICKLESS\_IDLE** --------------设置1，空闲任务会抑制刻度，并且处理器尽可能长时间保持在低功耗模式。主要在低功耗中使用。

two macros, configPRE\_SLEEP\_PROCESSING and configPOST\_SLEEP\_PROCESSING are generated in FreeRTOSConfig.h, and two empty functions, PreSleepProcessing and PostSleepProcessing generated in freertos.c (to be completed with user code).

参考：

https://wenku.baidu.com/view/87b4bbb7b8d528ea81c758f5f61fb7360a4c2b63.html

**USE\_TASK\_NOTIFICATIONS** -----------每个RTOS任务都有一个32位通知值，可以直接发送，可以解除阻塞接收任务，并可选择更新接收任务通知值。把对应为取消，可以节约每个任务，saving 8 bytes

* 1. Memory management settings

**Memory Allocation** -----------内存分配方式，

two new parameters are available: configSUPPORT\_DYNAMIC\_ALLOCATION and configSUPPORT\_STATIC\_ALLOCATION.

**TOTAL\_HEAP\_SIZE** -------------总共的堆的尺寸。

**Memory Management scheme** ----为了更换和改进C标准库malloc（）和free（）函数的使用，FreeRTOS下载包括四个示例内存分配实现。这个参数就是实现哪一种方式。

The schemes are implemented in the heap\_1.c, heap\_2.c, heap\_3.c, heap\_4.c and heap5.c sources files respectively.

* 1. Hook function related definitions

**USE\_IDLE\_HOOK** ---------- The idle task hook function is a hook (or callback) function that, if defined and configured, will be called by the idle task on each iteration of its implementation.

**USE\_TICK\_HOOK** ---------- The tick hook function is a hook (or callback) function that, if defined and configured, will be called during each tick interrupt.

**USE\_MALLOC\_FAILED\_HOOK** -- The malloc() failed hook function is a hook (or callback) function that.

**USE\_DAEMON\_TASK\_STARTUP\_HOOK** ---- If USE\_TIMERS and USE\_DAEMON\_TASK\_STARTUP\_HOOK are both set to 1 (Enabled) then the application must define an hook function

**CHECK\_FOR\_STACK\_OVERFLOW** --------当内存溢出的时候，触发对应的回调函数

FreeRTOS provides two optional mechanisms that can be used to assist in stack overflow detection and debugging

----- when CHECK\_FOR\_STACK\_OVERFLOW is set to 1 (option1), one mechanism is used  
----- when CHECK\_FOR\_STACK\_OVERFLOW is set to 2 (option2), both mechanisms are used.

设置1或2后，需要用户具体去完成。

* 1. Run time and task stats gathering related definitions

**GENERATE\_RUN\_TIME\_STATS** -------------- FreeRTOS可以配置为收集任务运行时统计信息。 vTaskGetRunTimeStats（）API函数将收集的运行时统计信息格式化为可读表。

**USE\_TRACE\_FACILITY** -------------------设置1，包含其他结构成员和函数来协助执行可视化和跟踪

**USE\_STATS\_****FORMATTING\_FUNCTIONS** -------将USE\_TRACE\_FACILITY和USE\_STATS\_FORMATTING\_FUNCTIONS设置为1（启用）以在生成中包含vTaskList（）和vTaskGetRunTimeStats（）函数。设置0，就不会包含vTaskList（）和vTaskGetRunTimeStats（）函数。

1.5 Co-routine related definition

**USE\_CO\_ROUTINES** --------------------设置为1以在构建中包括协同例行功能

**MAX\_CO\_ROUTINE\_PRIORITIES** ----------设置可以分配给协同程序的最大优先级。 可以从0分配优先级，这是优先级最低的（MAX\_CO\_ROUTINE\_PRIORITIES-1），这是最高优先级。

1.6 Software timer definitions

**USE\_TIMERS** -------------------------设置1，启动软件定时器功能。启用时，将为定时器任务分配内存：请求的大小（以字节为单位）= 4 \* configTIMER\_TASK\_STACK\_DEPTH。

1.7 Interrupt nesting behavior configuration

**LIBRARY\_LOWEST\_INTERRUPT\_PRIORITY** -----------可以在调用“设置优先级”功能时使用的最低中断优先级。

**LIBRARY\_MAX\_SYSCALL\_INTERRUPT\_PRIORITY** -------任何中断服务程序可以使用的最高中断优先级，使得调用中断安全的FreeRTOS API函数。

1. Include definitions

**vTaskPrioritySet** ------设置1，使能vTaskPrioritySet()

**uxTaskPriorityGet------**设置1， 使能uxTaskPriorityGet()

**vTaskDelete** -----------设置1，使能vTaskDelete()

**vTaskCleanUpResources** --设置1，使能vTaskCleanUpResources （）把任务挂起。

**vTaskSuspend** ---------设置1，使能vTaskSuspend() 把任务挂起

**vTaskDelayUntil** ------设置1，使能vTaskDelayUntil() 任务以一定周期运行。

**vTaskDelay** -----------设置1，使能vTaskDelay() 软延时。

**xTaskGetSchedulerState** –设置1，使能xTaskGetSchedulerState() 。获取调度器当前状态。taskSCHEDULER\_NOT\_STARTED（未启动）、taskSCHEDULER\_RUNNING（正常运行）、taskSCHEDULER\_SUSPENDED（挂起）。

**xTaskResumeFromISR** -----设置1，使能xTaskResumeFromISR()用于恢复一个挂起的任务，用在ISR（用于中断服务）中。尽量任务被挂起，中断内才可以执行该函数。

**xQueueGetMutexHolder** ----设置1，as the default value of xSemaphoreGetMutexHolder (if it is not explicitly set)

**xSemaphoreGetMutexHolder** –设置1，使能xSemaphoreGetMutexHolder()

任务持有互斥信号量的任务句柄

**pcTaskGetTaskName** ----------设置1，使能pcTaskGetTaskName()：Looks up the name of a task from the task's handle.

**uxTaskGetStackHighWaterMark** 设置1，使能uxTaskGetStackHighWaterMark() 返回任务开始执行后对任务可用的剩余堆栈空间的最小量

**xTaskGetCurrentTaskHandle** 设置1，使能xTaskGetCurrentTaskHandle() ：返回任务的句柄。

**eTaskGetState** ------------设置1，使能eTaskGetState() ：返回任务的状态。

**xEventGroupSetBitFromISR** 设置1，使能xEventGroupSetBitFromISR()

：设置事件组中的位将自动解除阻塞等待位的任务。

**xTimerPendFunctionCall** 设置1：使能xTimerPendFunctionCall() ：用于将功能执行到RTOS守护程序任务

**xTaskAbortDelay** --------设置1：使能xTaskAbortDelay() API ：强制任务离开“已阻止”状态，并进入“就绪”状态

**xTaskGetHandle** --------设置1：使能xTaskGetHandle() ：从任务的名称查找任务的句柄。

1. FREErtos Heap Usage

3.1 summary

**HEAP STILL AVAILABLE** ----堆的数量仍然可用(equals to config\_TOTAL\_HEAP\_SIZE - TOTAL HEAP USED)

**TOTAL HEAP USED** -------Total amount of the heap used by known objects

**Total amount for tasks** -- Total amount of RAM used by defined tasks.

**Total amount for queues** ----- Total amount of RAM used by defined queues

**Total amount for timers** ------ Total amount of RAM used by defined timers.

**Total amount for mutexes and semaphores** ---- Total amount of RAM used by defined mutexes and semaphores .

3.2 FreeRTOS tasks

**Idle task** ------------- The idle task

**defaultTask** ----------- User task