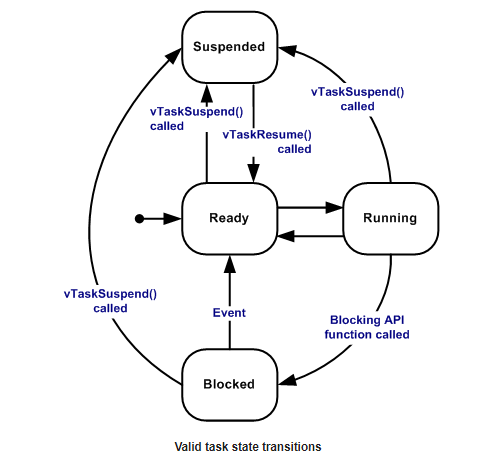
Bibliographie : [www.freeRTOS.org](http://www.freeRTOS.org) !!!!

Task :



Task priority :

FreeRTOSconfig.h -> configMAX\_PRIORITIES pour augmenter le nombre de priorité, 0 = idle donc moins prioritaire.

configUSE\_PORT\_OPTIMISED\_TASK\_SELECTION=1 => utilisation du port optimisation => limitation à 32 niveux de priorité.

configUSE\_TIME\_SLICING=0 donc pas de partage du processing time pour les tache de même priorité

Task ne doivent rien renvoyer sinon appel configASSERT(), si besoin de sortir dela tache utilisé : vTaskDelete( NULL );

Il existe des notifications de tache pour communiquer avec les tache, is 45% faster \* and uses less RAM than unblocking a task with a binary semaphore. Notifications are sent using the [xTaskNotify()](http://www.freertos.org/xTaskNotify.html) and [xTaskNotifyGive()](http://www.freertos.org/xTaskNotifyGive.html) and remain pending until the receiving RTOS task calls either of the [xTaskNotifyWait()](http://www.freertos.org/xTaskNotifyWait.html) or[ulTaskNotifyTake()](http://www.freertos.org/ulTaskNotifyTake.html).

Possibilité de sauver 8bit par task en desactivant les notifications: setting configUSE\_TASK\_NOTIFICATIONS to 0 in FreeRTOSConfig.h.

Queues = FIFO pour envoyer des messages entre les task et les interrupt. Messages copier directement dedans ou pointeur vers un buffer.