The Delete Task function

Function : void vTaskDelete (TaskHandle_t xTaskToDelete)

- **771** : The function takes a parameter **xTaskToDelete.** It is a null pointer (defined in line 103 osTask.h).
- **773 :** Now, locally a pointer of TCB_t (pxTCB) is created for storing the information of the task to be deleted.
- **775 :** taskENTER_CRITICAL() starts the critical section execution. it disables the interrupts to ensure exclusive access to shared resources.
- **779 :** Now, pxTCB is assigned to the return value of the function prvGetTCBFromHandle(xTaskToDelete), which returns the pointer of TCB of the task to be deleted.
- **785 :** The uxListRemove(&(pxTCB → xGenericListItem)) == 0 removes the task from the ready list with member address of xGenericListItem.
  - If the return value is 0, then the task was the only task at that priority level, so priority is reset using taskRESET_READY_PRIORITY.
- **795 :** listLIST_ITEM_CONTAINER(&(pxTCB->xEventListItem)) != NULL checks if the task is waiting on an event by verifying if the xEventListItem member of the pxTCB is associated with list item container. If not NULL, then it is waiting for an event.
  - If the task is waiting on an event , then the uxListRemove (&(pxTCB->xEventListItem)) removes the task from the event list with the address of xEventListItem.
- IF THE IF CONDITIONS IN LINE 785 and 795 doesn't satisfy , it calls the **mtCOVERAGE_TEST_MARKER() ,** which is a place for code coverage testing, allowing code coverage tools and much more to trace and track the execution of the code.
- **804 : vListInsertEnd(&xTasksWaitingTermination, &(pxTCB->xGenericListItem))**

  inserts the task into the termination list by calling the function with address of xGenericListItem, which is a member of TCB.

Finally, uxTasksDeleted is also incremented to signal that a task has been deleted and uxTaskNumber is also updated to include awareness for kernel debuggers that the task list needs regeneration as a the list has been modified.

- **817 :** taskEXIT_CRITICAL() is called to exit the critical section and to signal that interrupts can function normal again.

The task is not yet completed successful. What if the task deleted is the current task ? So ,

- **821 :** If , xSchedulerRunning is not pdFalse , (if the scheduler is running)
  - **823 :** If pxTCB == pxCurrentTCB (if the task is current task)
    - The function **portPRE_TASK_DELETE_HOOK(pxTCB, &xYieldPending)** is called as it is primarily used for performing specific clean up operations before a task is deleted.
    - Then , **portYIELD_WITHIN_API()** is called as in some situations after pre-delete hook functions, it may not be able to yield away from the task, so a pre-context switch is scheduled.
  - **835 :** If, the running task is not the current task, then :
    - We enter back to critical section using taskENTER_CRITICAL() , we call **prvResetNextTaskUnblockTime() (LINE 3602 osTasks.c)** to reset the next unblock time in case it referred to the task that has just been deleted.
    - Then, it exits the critical section using taskEXIT_CRITICAL().

uxTopReadyPriority:

- It is a variable or data structure that represents the priority of the highest ready task or thread in the system.
- In an RTOS, tasks or threads are typically assigned priorities based on their importance or urgency. The priority level determines the order in which tasks are scheduled for execution. The higher the priority, the more urgent or important the task is considered, and it will be scheduled to run before tasks with lower priorities.

- The "uxTopReadyPriority" variable or data structure is commonly used by the RTOS scheduler to keep track of the highest priority among the ready tasks or threads. It allows the scheduler to efficiently determine which task should be scheduled for execution next.
- By maintaining the highest priority, the scheduler can quickly identify the task with the most urgency and ensure that it gets CPU time as soon as possible.
- The “ux” prefix refers to unsigned x (unsigned integer with x bits) indicating that the variable stores an unsigned integer value representing the priority.

Where is it used ?

- It is first defined in line 268, which tracks the highest priority tasks.
- Line 331, a function to select highest priority task - **taskSELECT_HIGHEST_PRIORITY_TASK()**
- Line 351, a macro to reset priority to the task with high priority- **portRESET_READY_PRIORITY ().**