Project 2

Advanced Real Time Systems - ECE 5550G

Sriamoghavarsha Beerangi Srinivasa

Email : sriamoghavarsha@vt.edu

Student ID : 906556267

I have neither given nor received unauthorized assistance on this assignment.

Sriamoghavarsha Beerangi Srinivasa

Question 1

FlowCharts attached at the end. (Letters have been added next to any process block which has function call and those function have been implemented as a separate flowchart)

Brief description of the functions in scheduler.cpp

- **prvGetTCBIndexFromHandle -** This function returns the index of the given task handle in the xTCBArray.
- **prvInitTCBArray -** This function initializes the xTCBArray with default values.
- **prvFindEmptyElementIndexTCB -** This function finds the first available index which is empty in the xTCBArray.
- **prvDeleteTCBFromArray -** This function deletes a TCB entry pointed by the index from the xTCBArray.
- **prvPeriodicTaskCode** - Updates the xlastWakeTime of each periodic task and runs the periodic task.
- **vSchedulerPeriodicTaskCreate** - This function is used by the user to create periodic task instances and store them in xTCBArray.
- **vSchedulerPeriodicTaskDelete** - This function is used by the user to delete the task instances in the xTCBArray and the kernel.
- **prvCreateAllTasks** - This function creates tasks stored in the xTCBArray and adds them to the list of tasks that are ready to run.
- **prvSetFixedPriorities** - This function assigns fixed priorities to all the periodic tasks stored in the xTCBArray using the Rate-Monotonic policy.
- **prvPeriodicTaskRecreate** - This function recreates a task that was deleted because it missed a deadline using the information of the task stored in the xTCBArray.
- **prvDeadlineMissedHook** - This function deletes a periodic task that missed its deadline and will recreate the task and release it during the next period.
- **prvCheckDeadline** - This function checks if the task specified has missed its deadline or not.
- **prvExecTimeExceedHook** - This function is used to block a periodic task that has exceeded its worst-case execution time until the next period.
- **prvSchedulerCheckTimingError** - This function checks if any of the tasks have timing errors and is used by the scheduler task.
- **prvSchedulerFunction** - Checks if there are any timing errors and tasks that have missed deadlines.
- **prvCreateSchedulerTas**k - This function is used to create the scheduler task instance.
- **prvWakeScheduler** - Runs at an interval of schedSCHEDULER_TASK_PERIOD. Performs a context switch to Scheduler function.
- **vApplicationTickHook** - This function is called at every software tick and is used to increment the execution time of the current task and check if a timing error has occurred.
- **vSchedulerInit** - This function is used to initialize the xTCBArray and should be called before any other function in scheduler.cpp

- **vSchedulerStart** - This function starts scheduling the tasks

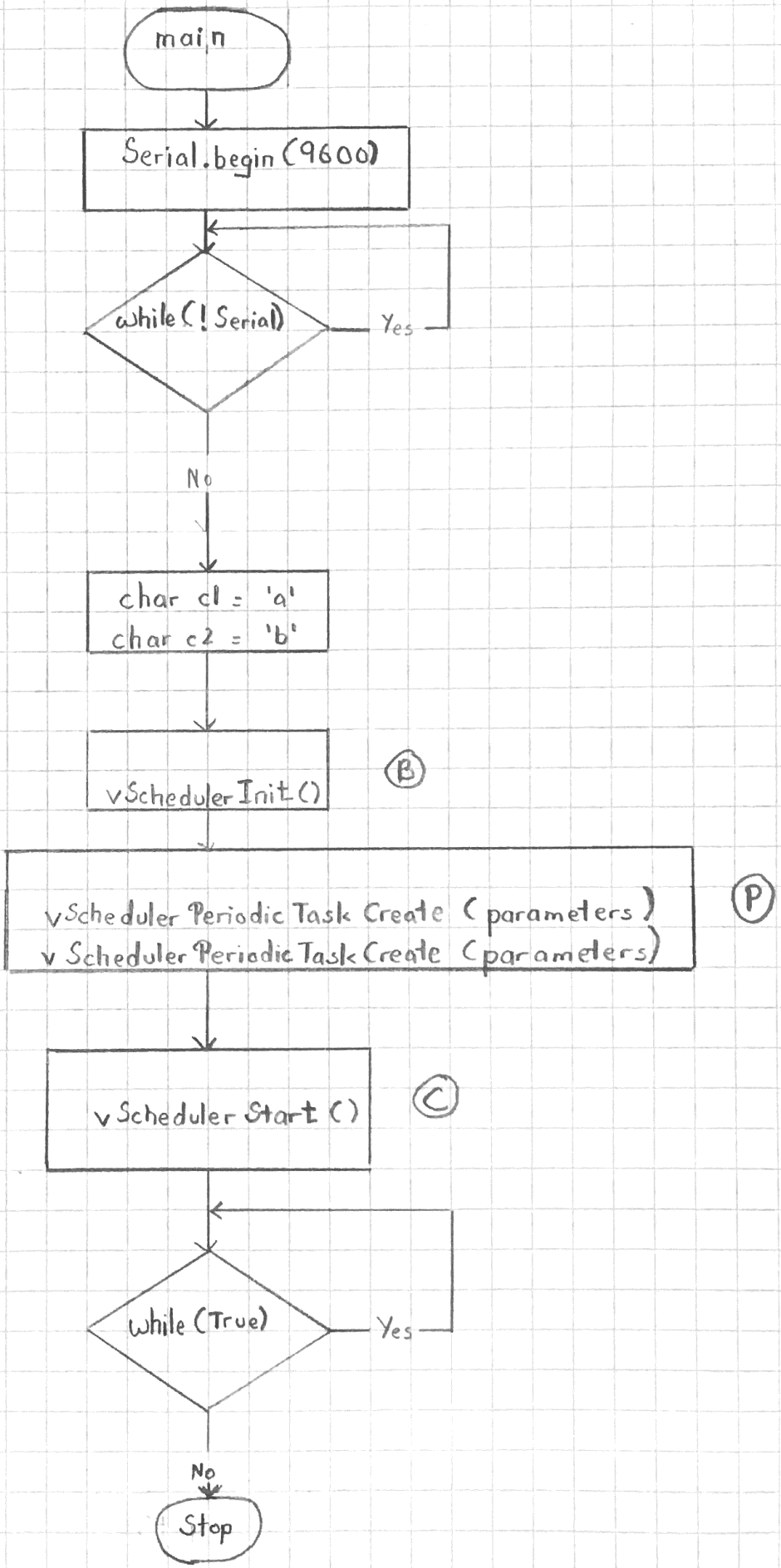
Question 2

- **Line 137 :** Search for the first available index in xTCBArray where the variable xInUse of SchedTCB_t data structure is set to pdFalse and return the index. If no such index is found, return -1.
- **Line 143 :** For the given xIndex, check if xIndex is less than schedMAX_NUMBER_OF_PERIODIC_TASKS. If true, go to the index in xTCBArray and set the variable xInUse of SchedTCB_t data structure to pdFALSE and decrement the xTaskCounter by 1.
- **Line 216 :**
    - pxNewTCB->xRelativeDeadline = xPhasetick + xDeadlineTick
    - pxNewTCB->xAbsoluteDeadline = xDeadlineTick
    - pxNewTCB->xExecTime = 0
    - pxNewTCB->xMaxExecTime = xMaxExecTimeTick
    - pxNewTCB-> xWorkIsDone = pdFALSE
- **Line 224 :**
- **Line 229 :** pxNewTCB->xExecutedOnce = pdFALSE
- **Line 247 :** Check if the xTaskHandle is not NULL. If True, get the xIndex of the xTaskHandle using the prvGetTCBIndexFromHandle(xTaskHandle) method. If the xIndex is not -1, delete the TCB from the xTCBArray using the prvDeleteTCBFromArray(xIndex) method.
- **Line 264 :** xTaskCreate(pxTCB->pvTaskCode, pxTCB->pcName, pxTCB->uxStackDepth, pxTCB->pvParameters, pxTCB->uxPriority, pxTCB->pxTaskHandle)
- **Line 291** : Check if variable xInUse of SchedTCB_t data structure is set to pdTRUE. If so, Store the address of value pointed by the index in TCBArray in pxTCB pointer
- **Line 293 :** Check if xPriorityIsSet of pxTCB data structure is set to pdFALSE. If True, Check if xPeriod of pxTCB data structure is less than xShortest. If True, Assign the value of xShortest to xPreviousShortest and Assign the value of xPeriod to xShortest and store the pointer to pxTCB in pxShortestTaskPointer.
- **Line 299 :** Check if the value of xShortest is not equal to the value of xPreviousShortest. If True, decrement xHighestPriority by 1. Using the pxShortestTaskPointer, assign the value of xHighestPriority to uxPriority of pxShortestTaskPointer and set xPriorityIsSet of pxShortestTaskPointer to pdTRUE.
- **Line 310 :** xTaskCreate(pxTCB->pvTaskCode, pxTCB->pcName, pxTCB->uxStackDepth, pxTCB->pvParameters, pxTCB->uxPriority, pxTCB->pxTaskHandle)
- **Line 328 :** vTaskDelete(pxTCB)
- **Line 333 :** After recreating the task, assign it with new WakeTime based on the current

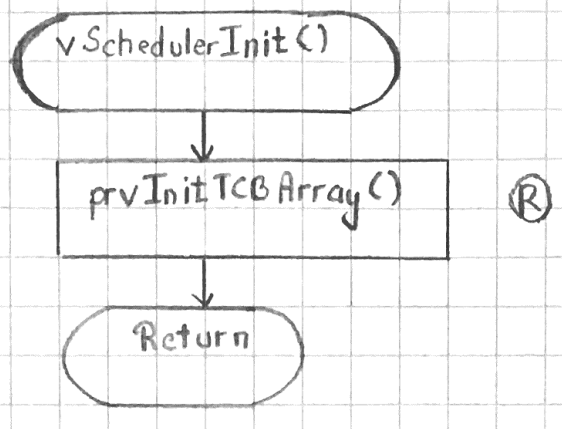
tickCount.

- **Line 341 :** Check if the task has missed a deadline by comparing it to the tickCount to xRelativeDeadline. If True, call the prvDeadlineMissedHook function.
- **Line 378 :** Check if a deadline has been missed by a periodic task.
- **Line 415 :** Check if any timing errors have occurred or if deadlines have been missed

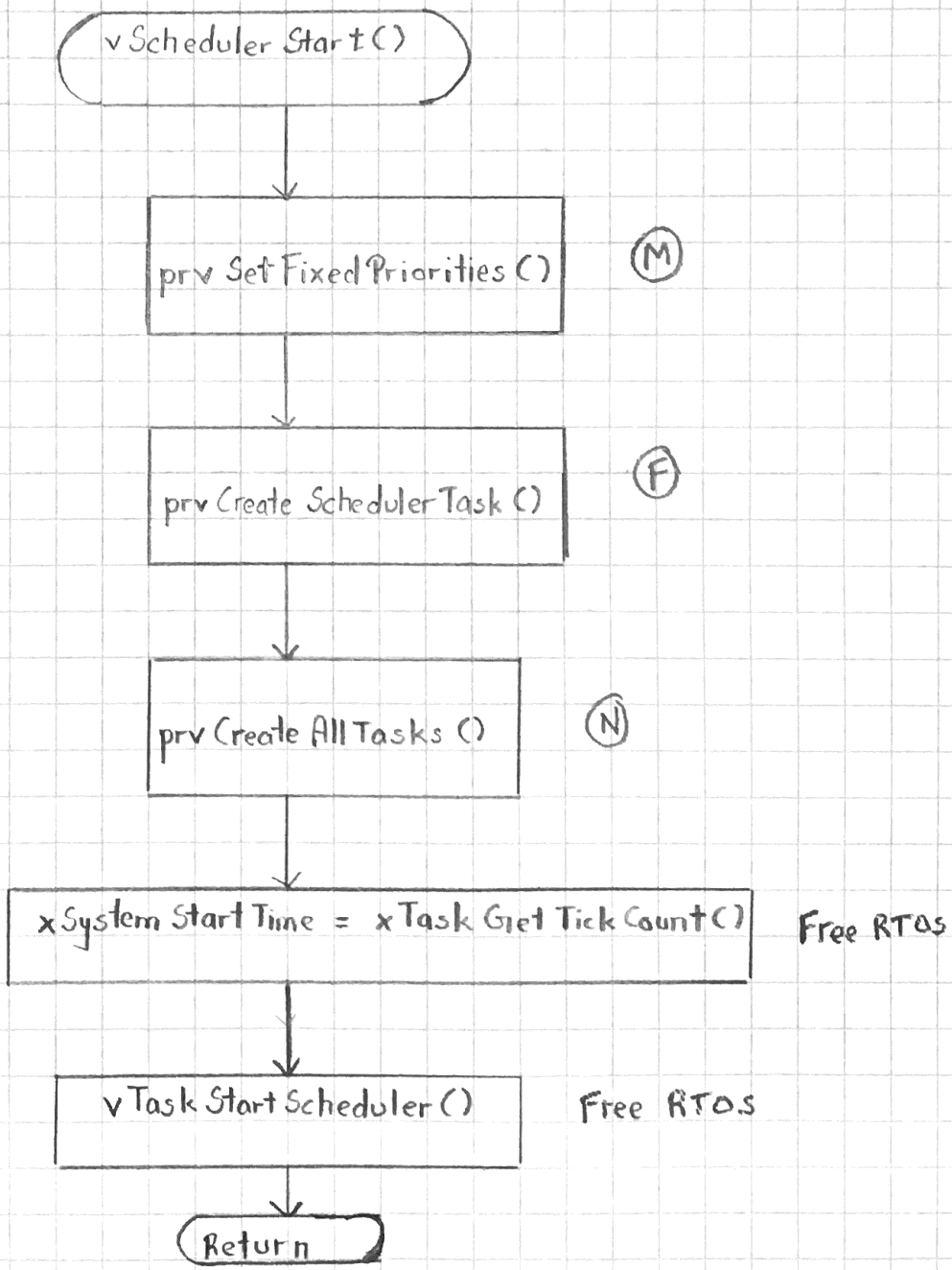
1) Flowchart of main()



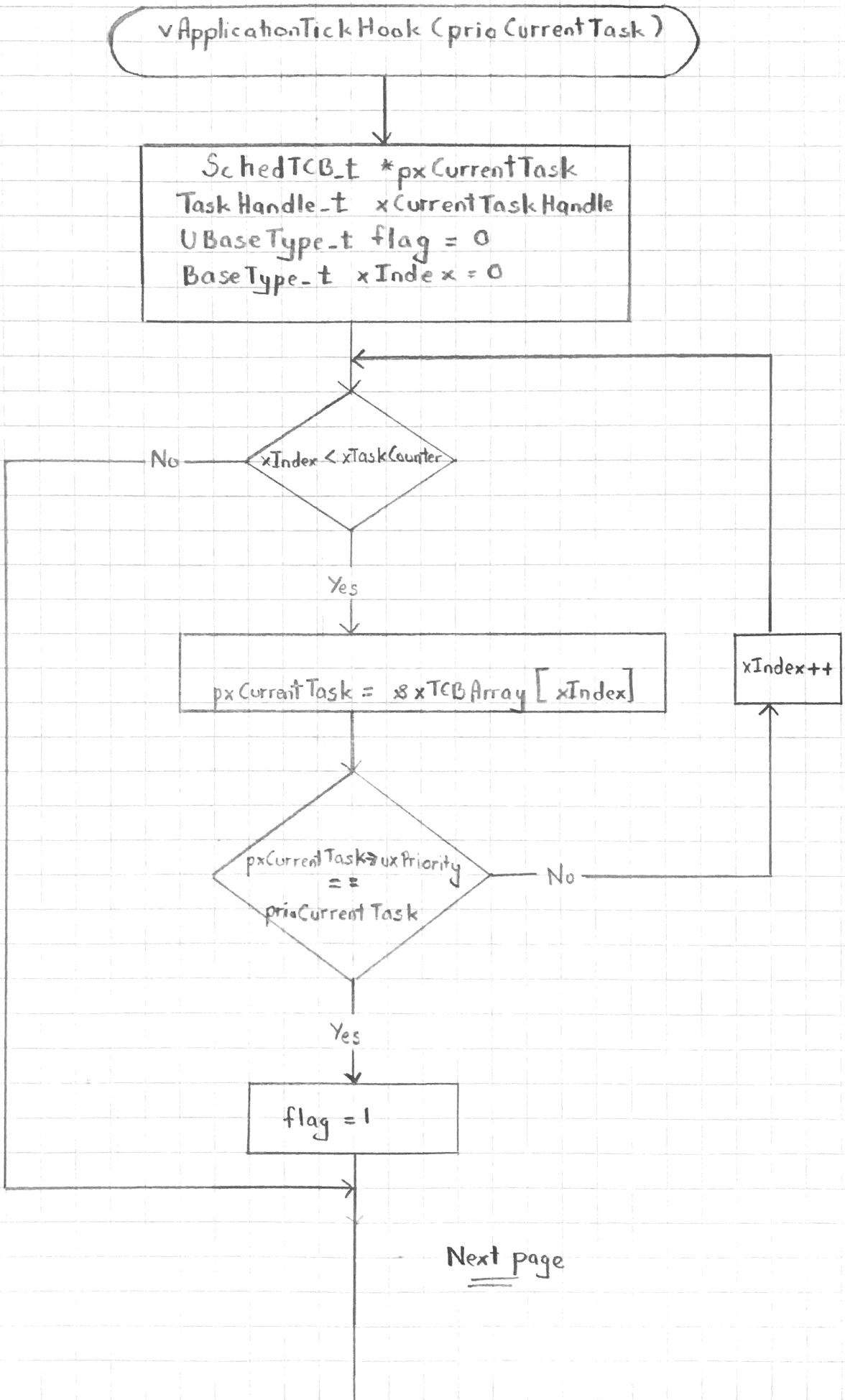
B Flowchart of Scheduler Init()

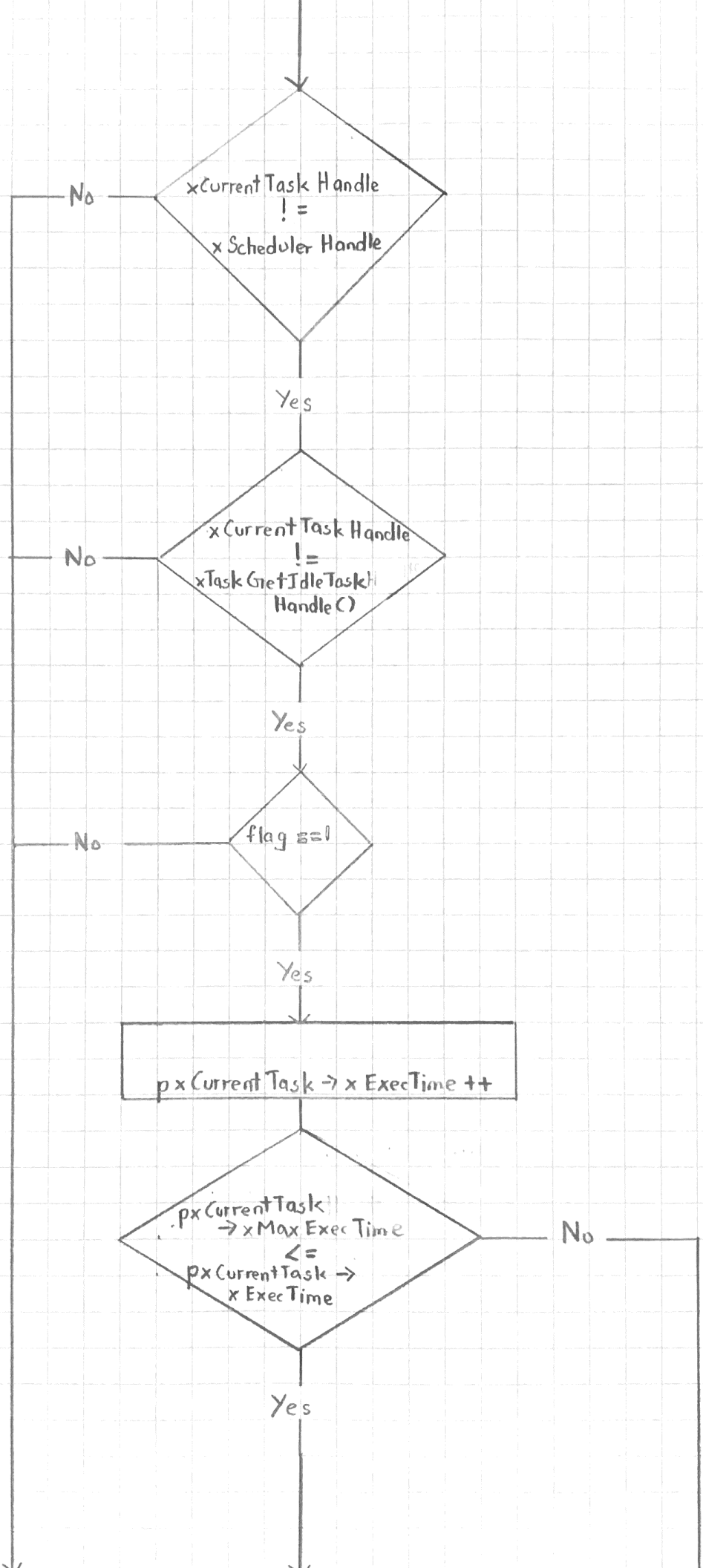


C Flowchart of vScheduler Start()

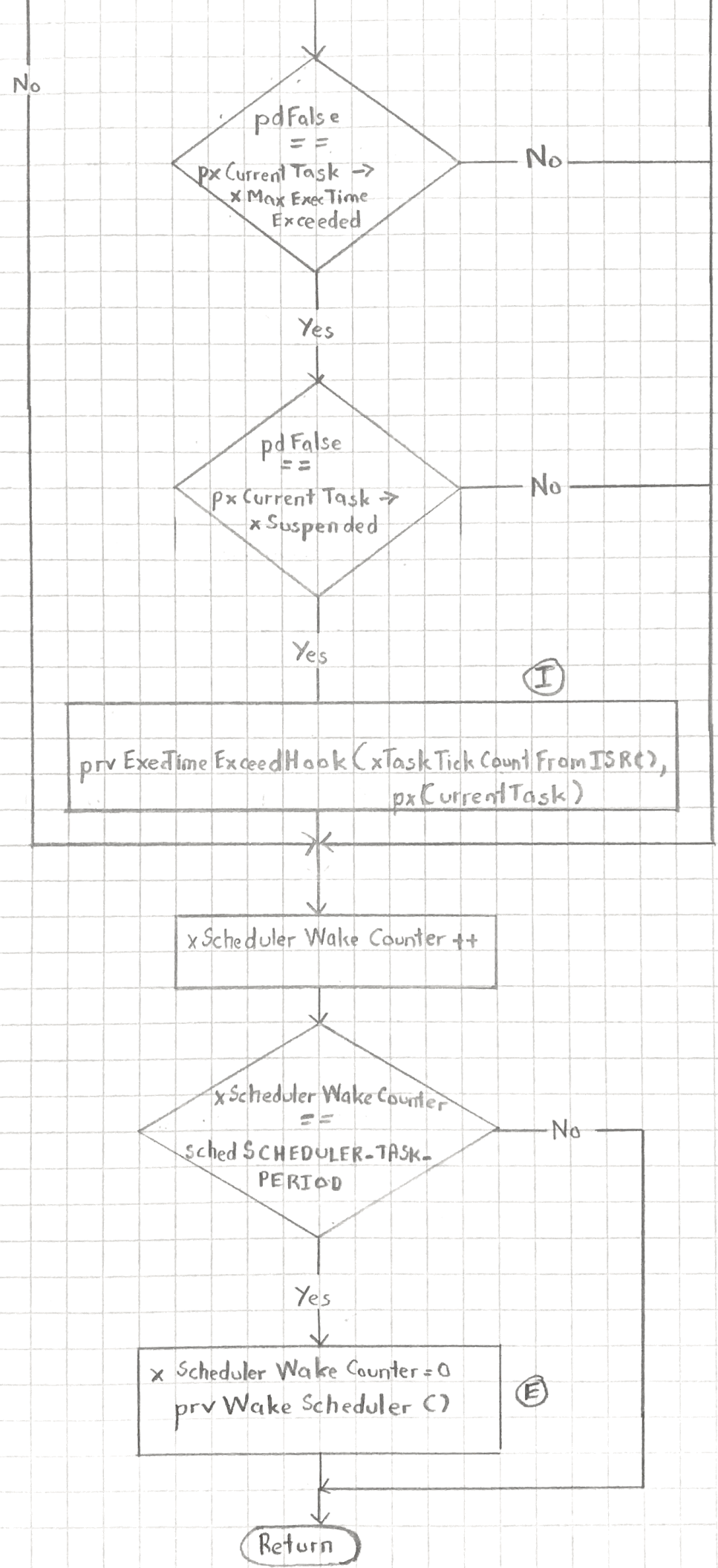


D Flowchart of vApplicationTickHook (prioCurrentTask)

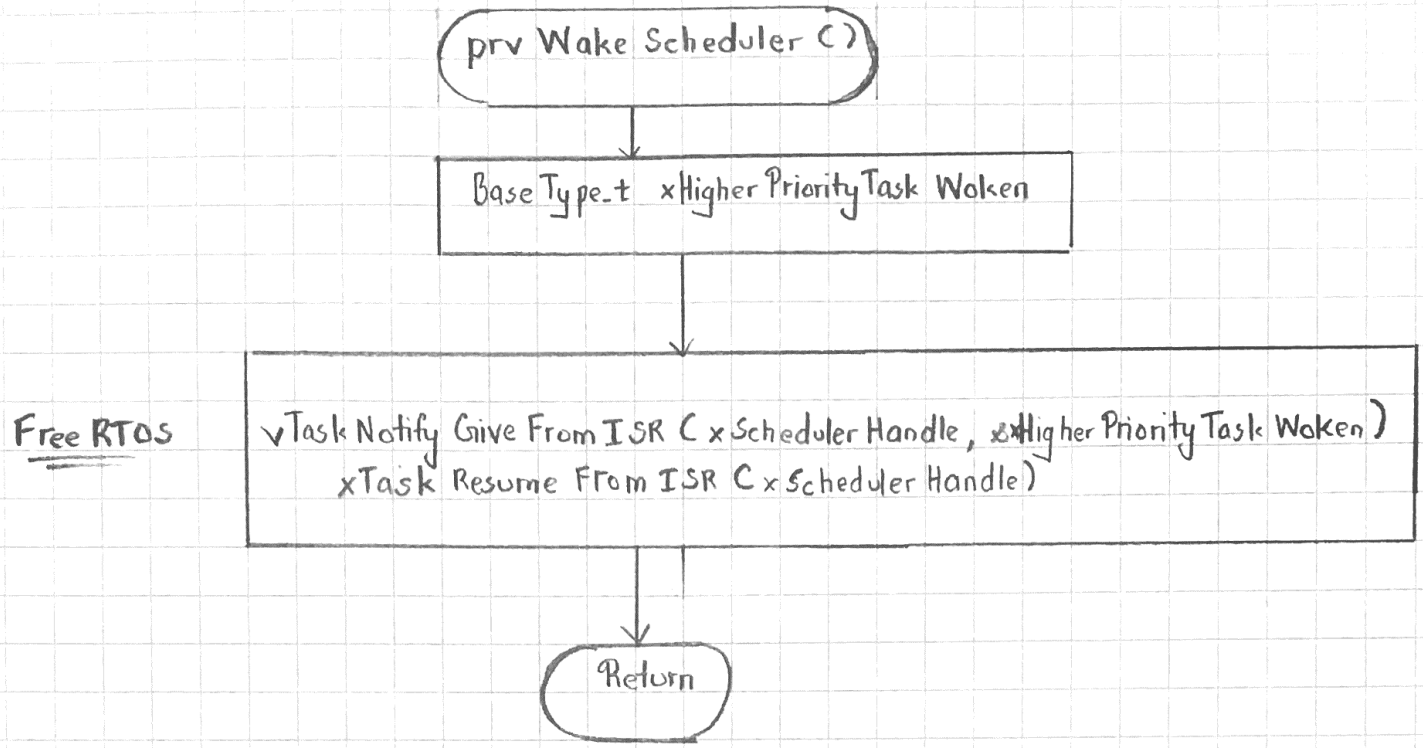



xCurrentTask Handle
!=
xScheduler Handle
No
Yes
xCurrentTask Handle
!=
xTaskGetIdleTaskH
Handle()
No
Yes
flag==1
No
Yes
pxCurrentTask->xExecTime ++
pxCurrentTask
->xMaxExecTime
<=
pxCurrentTask->
xExecTime
No
Yes

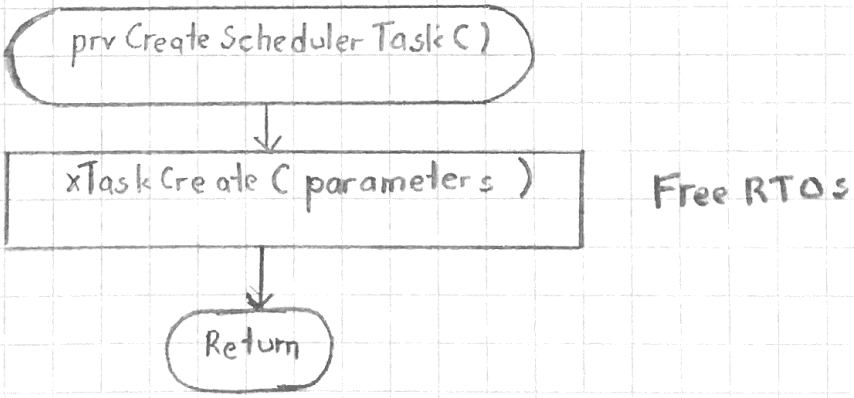
Next page


No
pdFalse
==
pxCurrentTask ->
xMaxExecTime
Exceeded
No
Yes
pdFalse
==
pxCurrentTask ->
xSuspended
No
Yes
I
prvExeTimeExceedHook(xTaskTickCountFromISR(),
pxCurrentTask)
xSchedulerWakeCounter ++
xSchedulerWakeCounter
==
schedSCHEDULER_TASK_
PERIOD
No
Yes
xSchedulerWakeCounter = 0
prvWakeScheduler()
E
Return

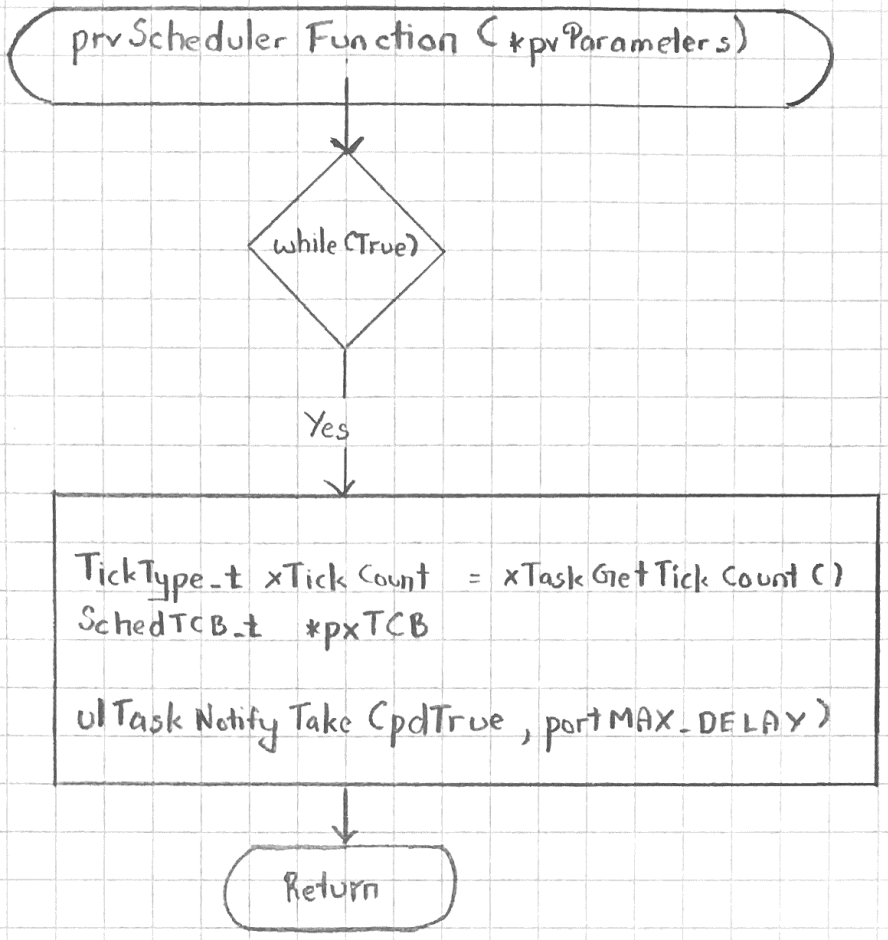
E) Flowchart of prv Wake Scheduler ()



F) Flowchart of prv Create Scheduler Task ()

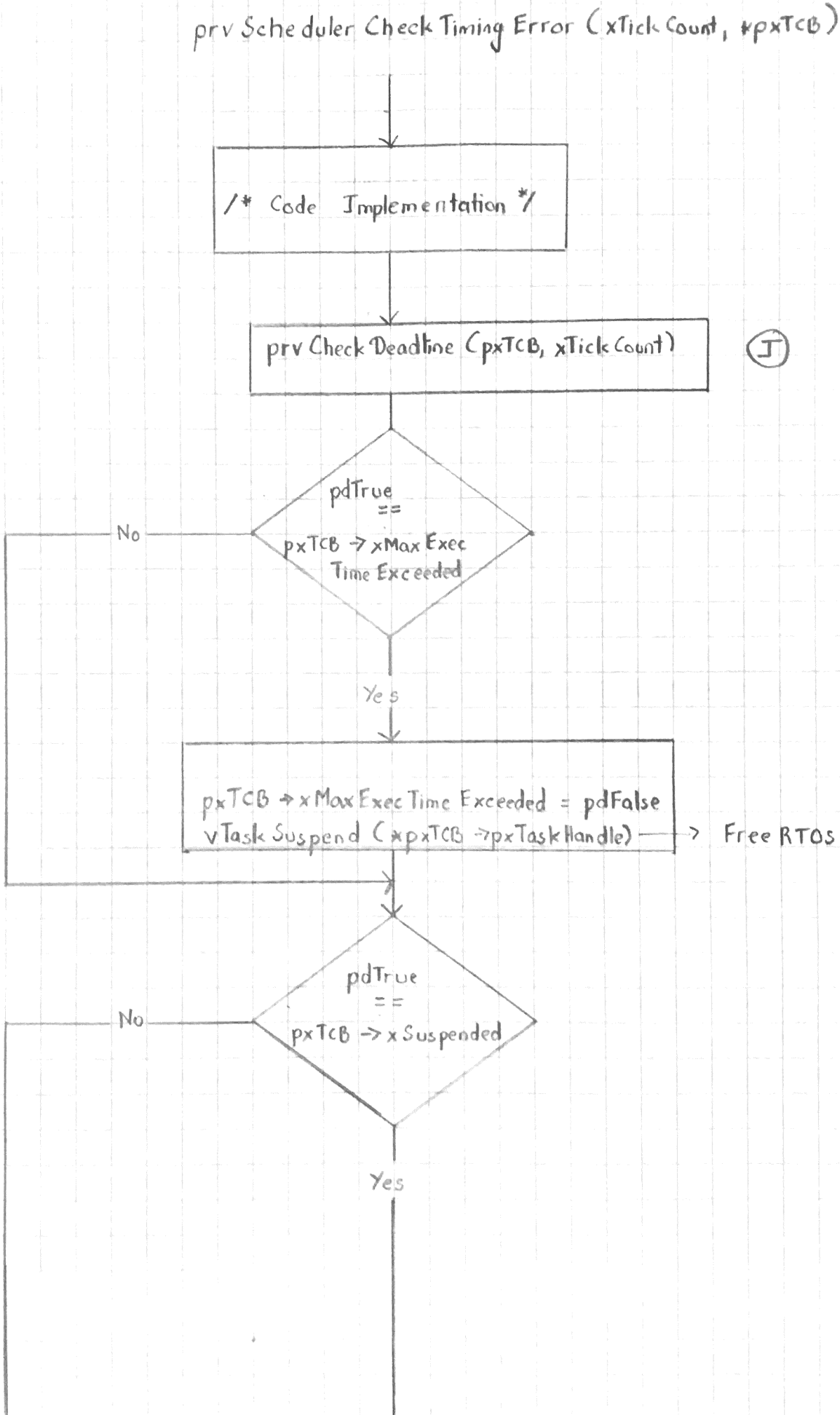


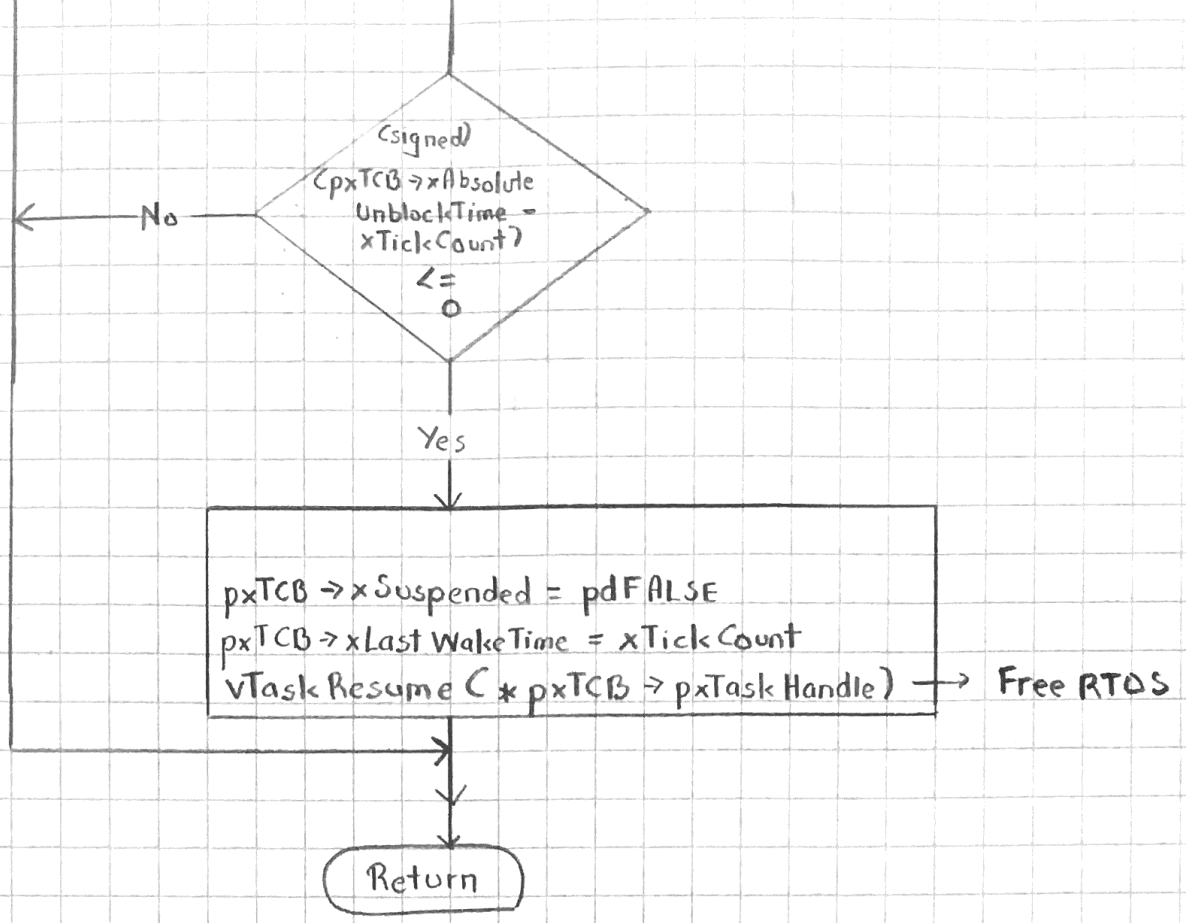
6) Flowchart of prvSchedulerFunction (*pvParameters)



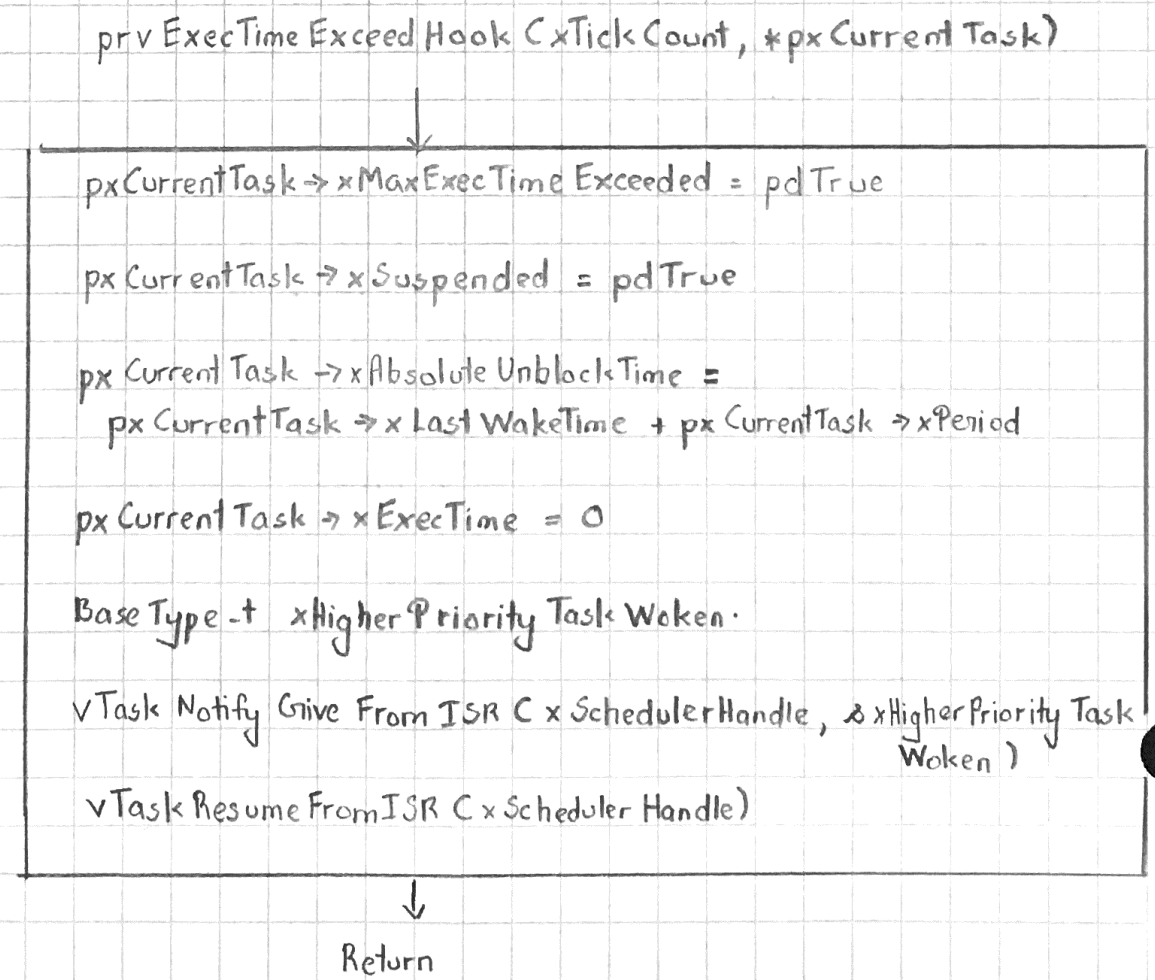
FreeRTOS

H) Flowchart of prv Scheduler Check Timing Error (xTick Count, *pxTCB)



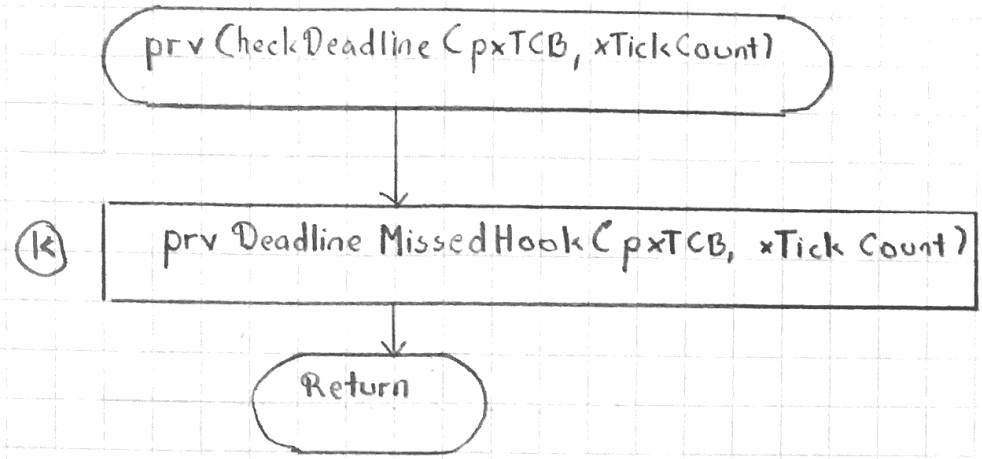


I) Flowchart of prvExecTimeExceedHook(xTickCount, *pxCurrentTask)

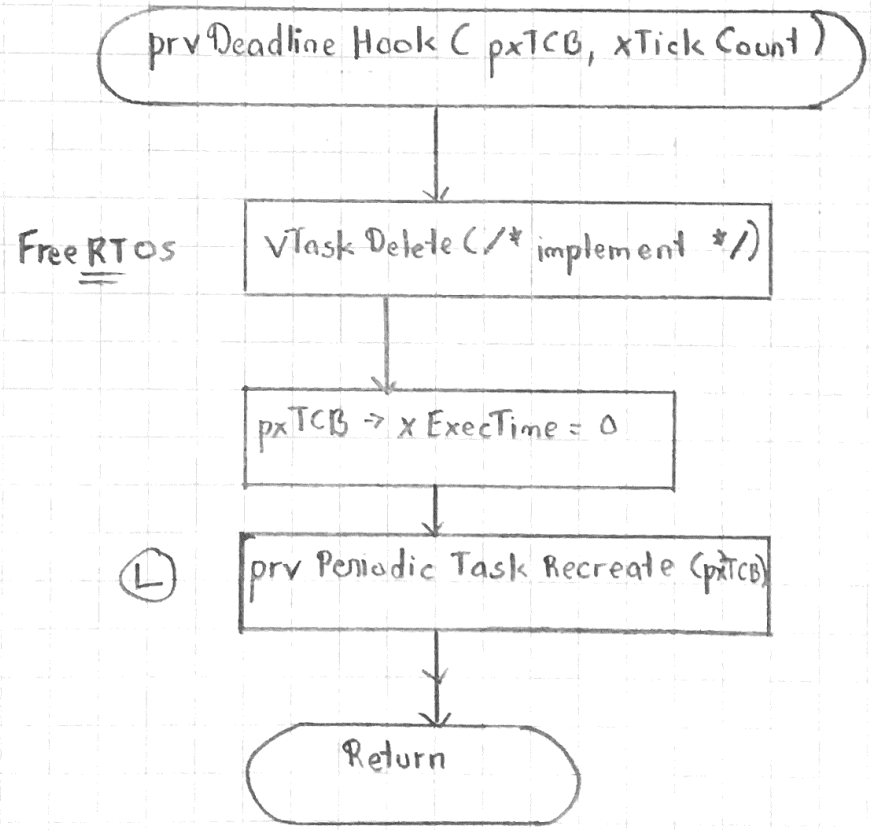


FreeRTOS

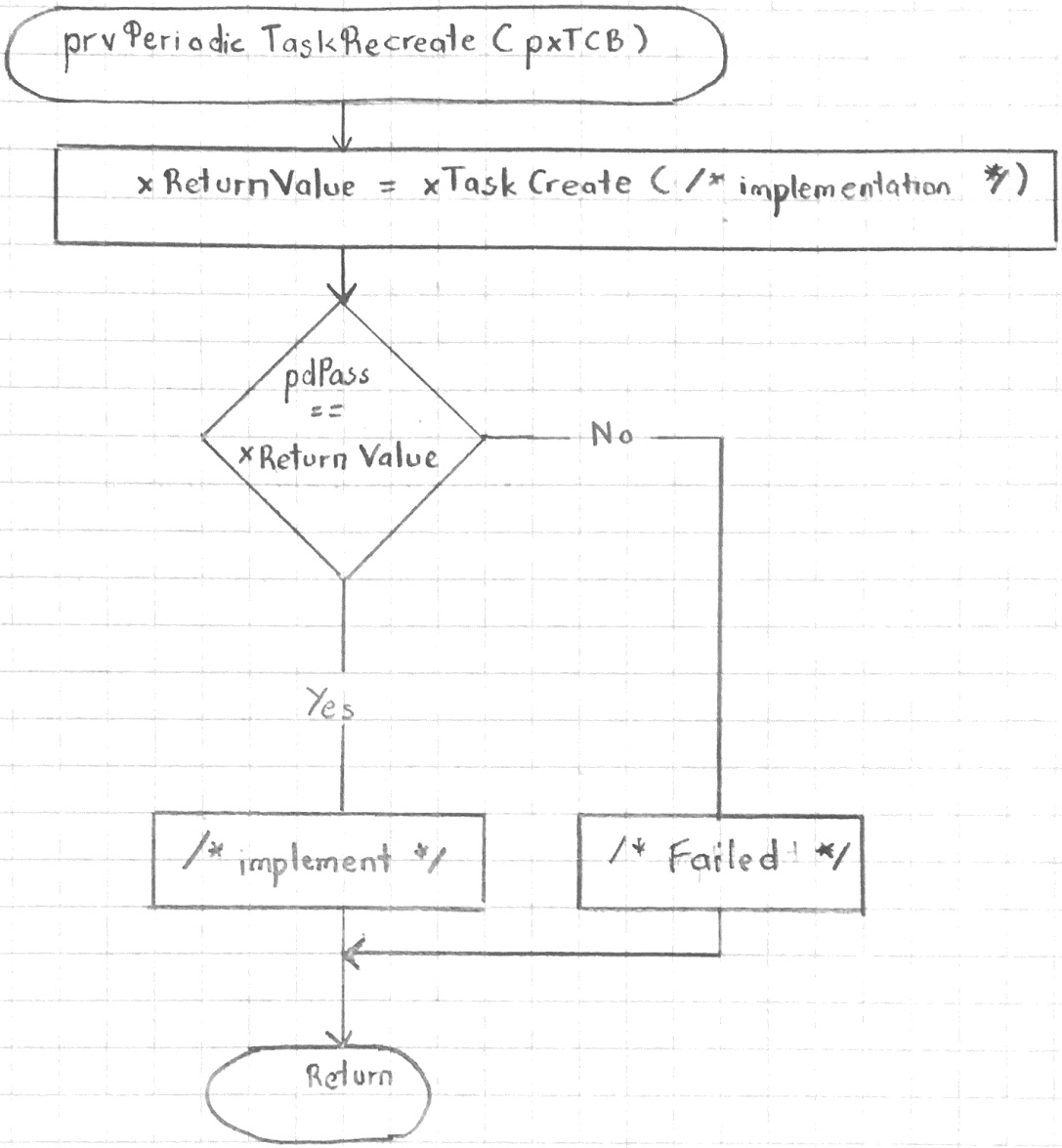
J) Flowchart of prvCheckDeadline(pxTCB , xTickCount)



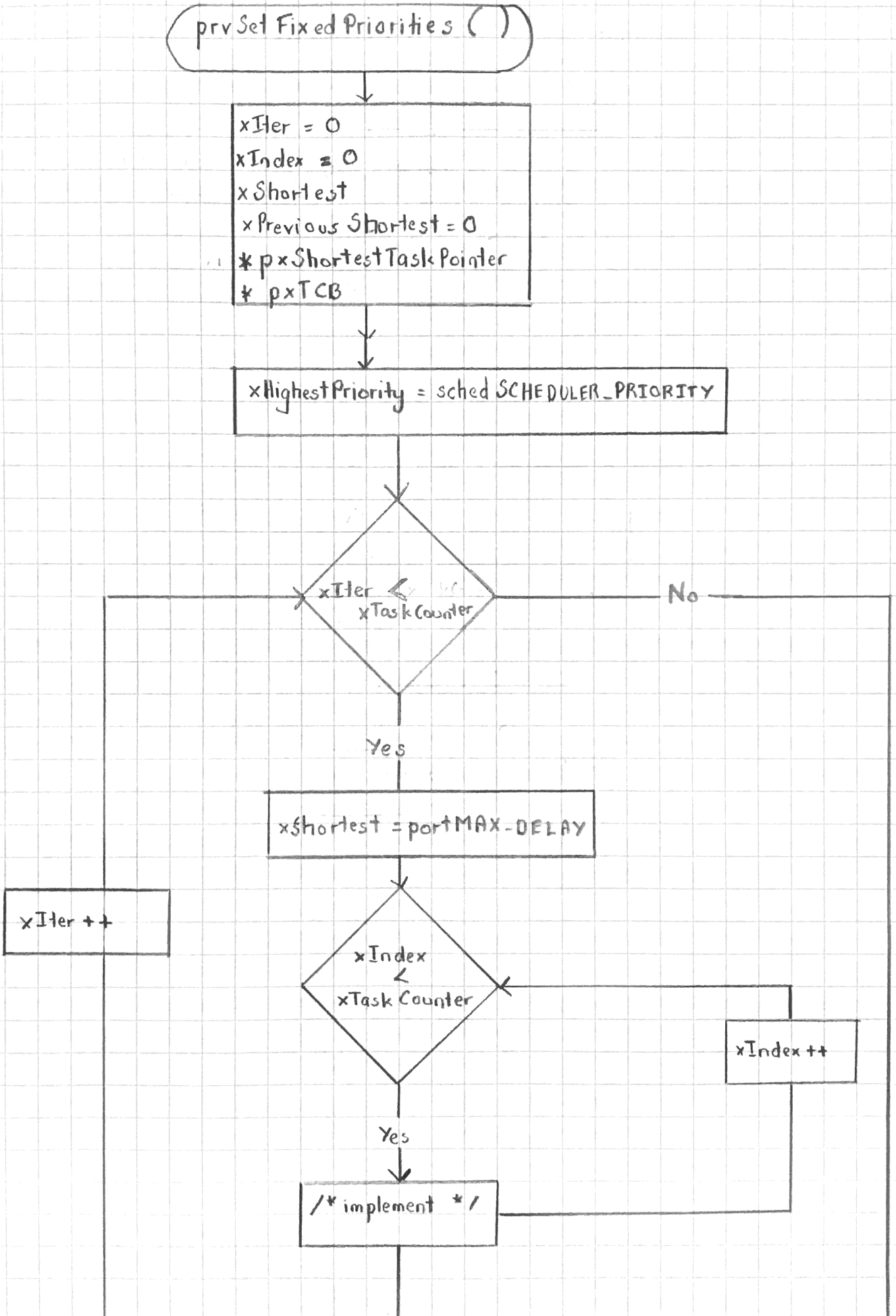
K) Flowchart of prvDeadlineMissedHook (pxTCB , xTickCount)

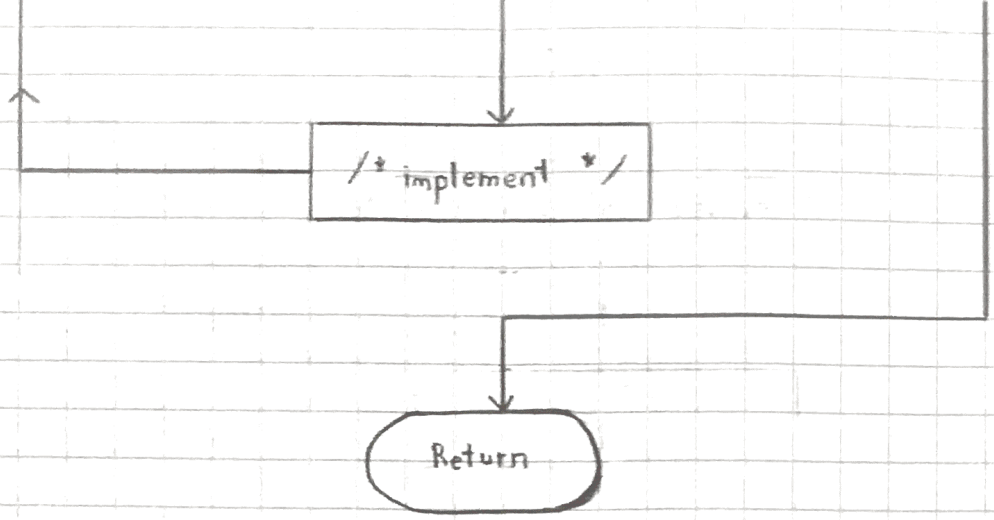


4) Flowchart of prvPeriadicTaskRecreate (pxTCB)



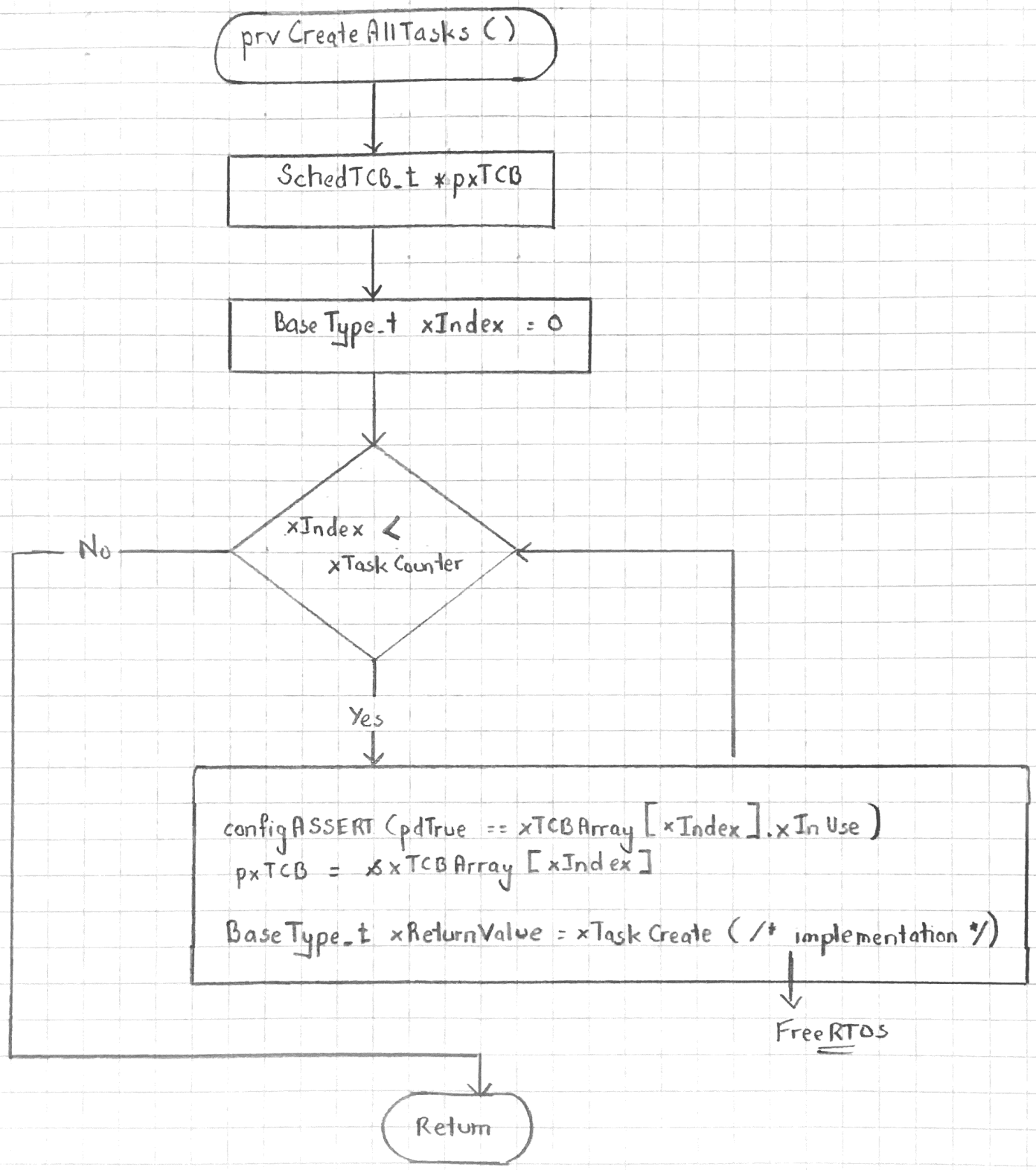
M) Flowchart of prvSetFixedPriorities ( )



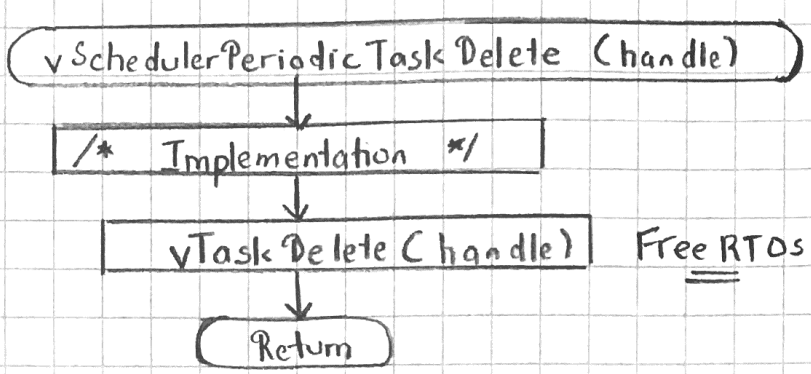

/* implement */

Return

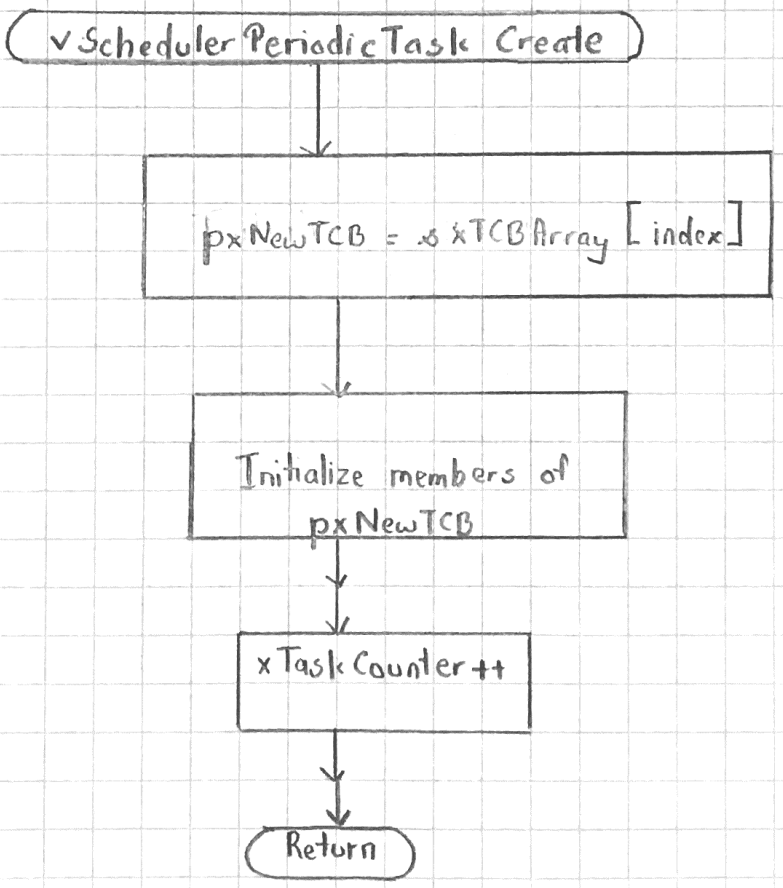
N) Flowchart of prvCreateAllTasks ()



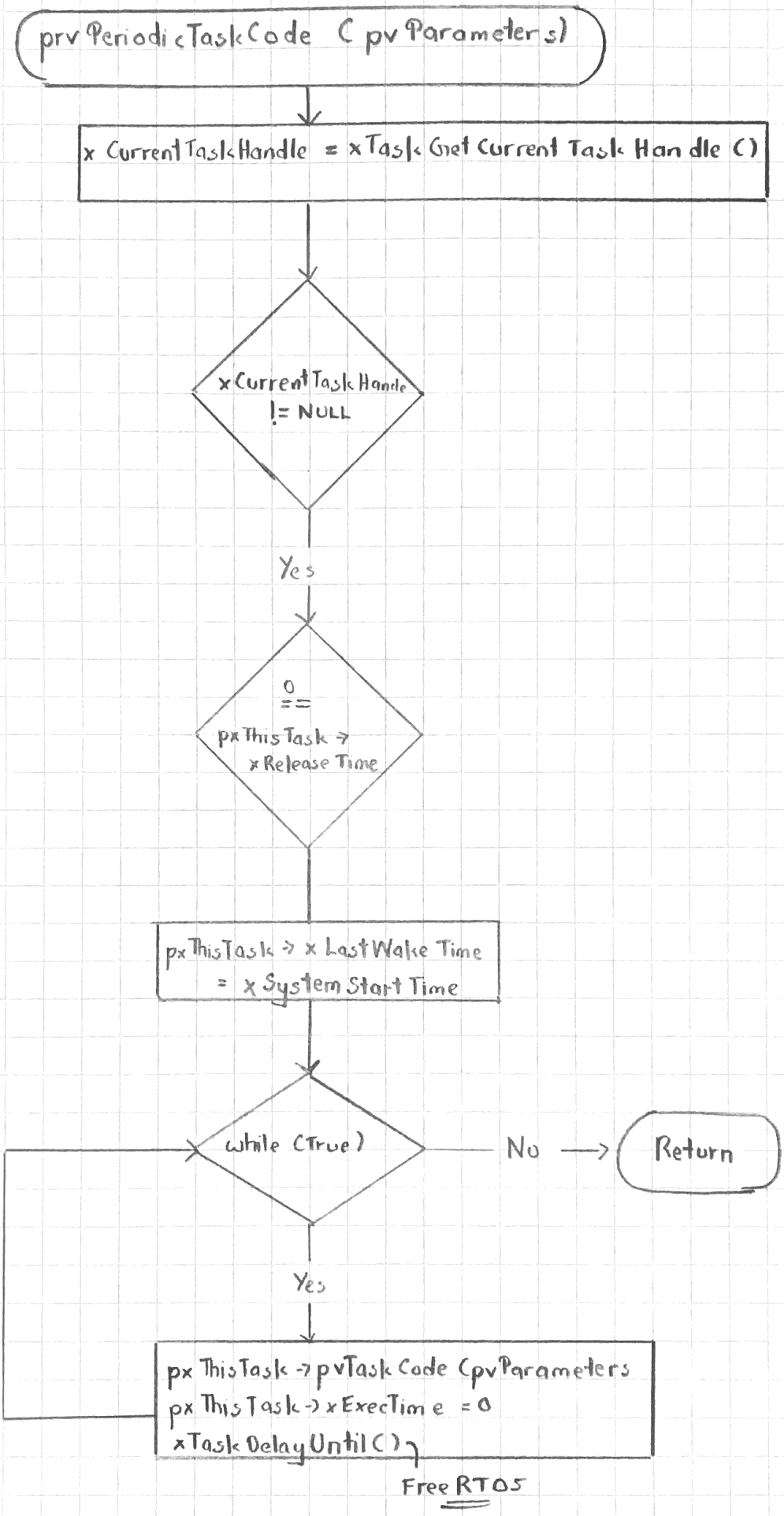
o) Flowchart of vSchedulerPeriodicTaskDelete



p) Flowchart of vSchedulerPeriodicTaskCreate



9) Flowchart of prvPeriodicTaskCode (pvParameters)



R) Flowchart of prvInitTCBArray()