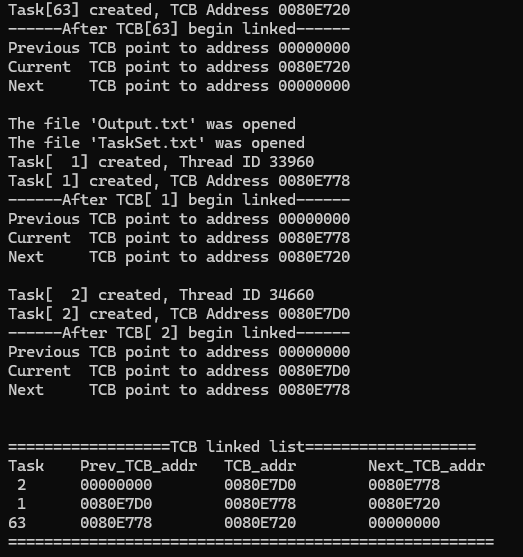
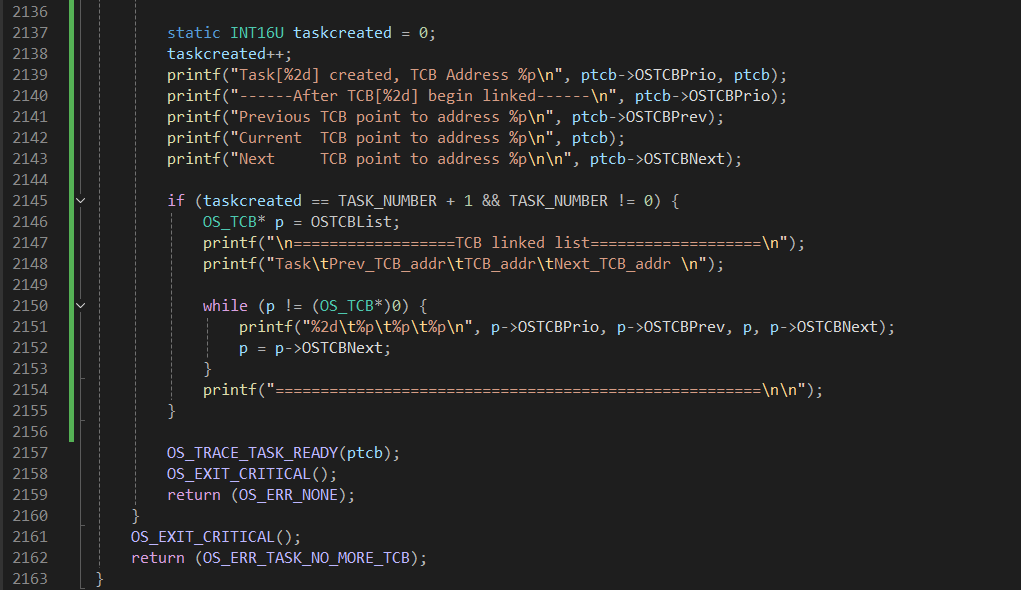
# PART I

TCB創建時記憶體資訊



修改部分

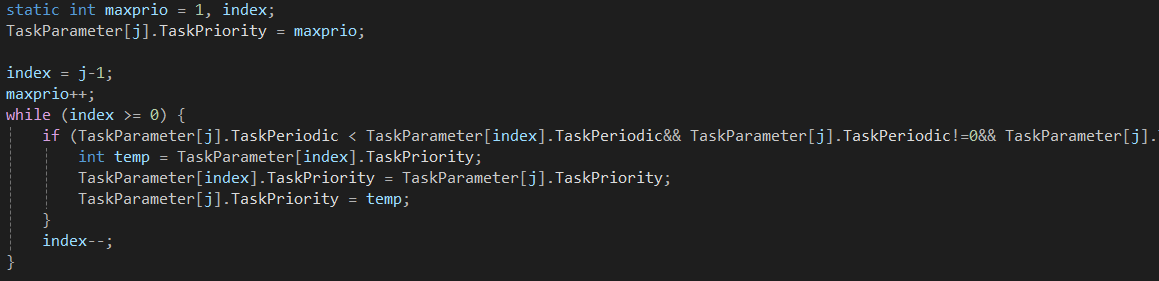
函式OS\_TCBInit()的末端:



這裡我使用task創建函式OS\_TCBInit()中所使用的ptcb來擷取資訊，因為它包含了所需的TCB資料，能指出前後TCB與自身位址，再用taskcreated變數來計算是否創立完所有TCB，最後輸出所有TCB的Linked list

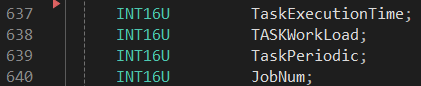
# PART II

App\_hooks.c:



在上次Lab的InputFile()中新增此段程式來重新依據Periodic排列Priority

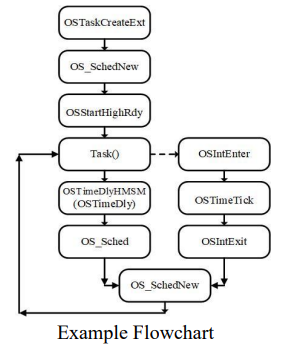
Ucos\_ii.h:

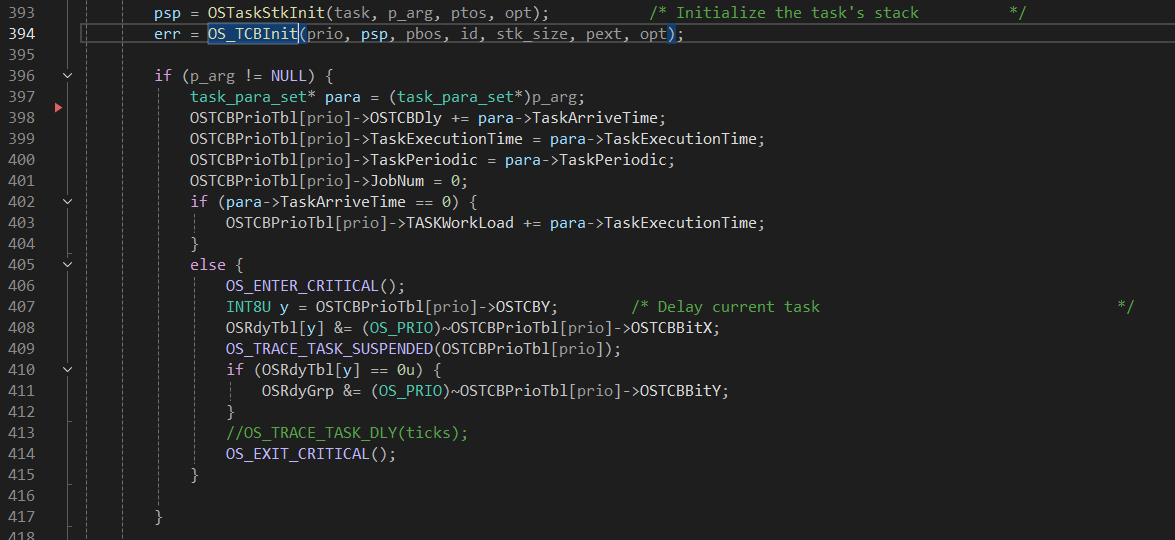


在OS\_TCB的STRUCT宣告中新增參數，TASKWorkLoad和JobNum分別是該Job剩餘工作量和第幾個Job

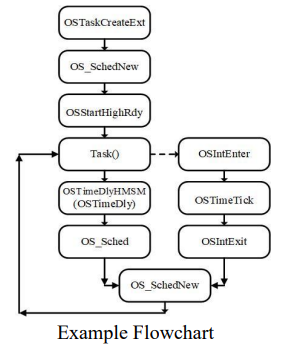


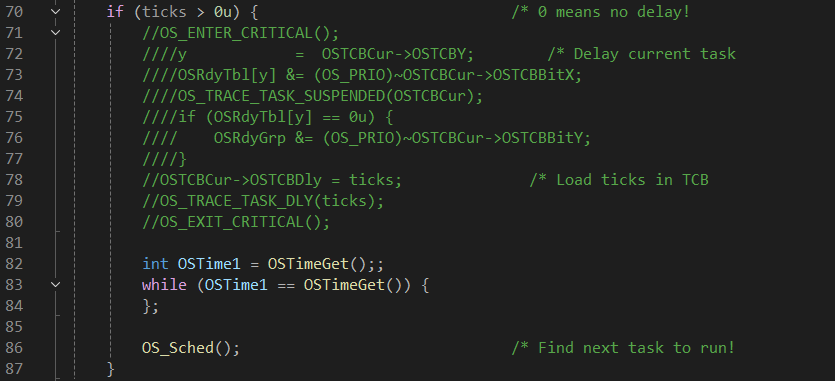
新增GLOBAL VARIABLES，是否MissDeadLine和Miss的TCB



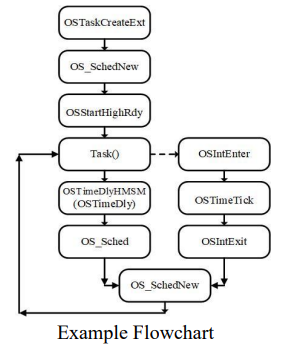


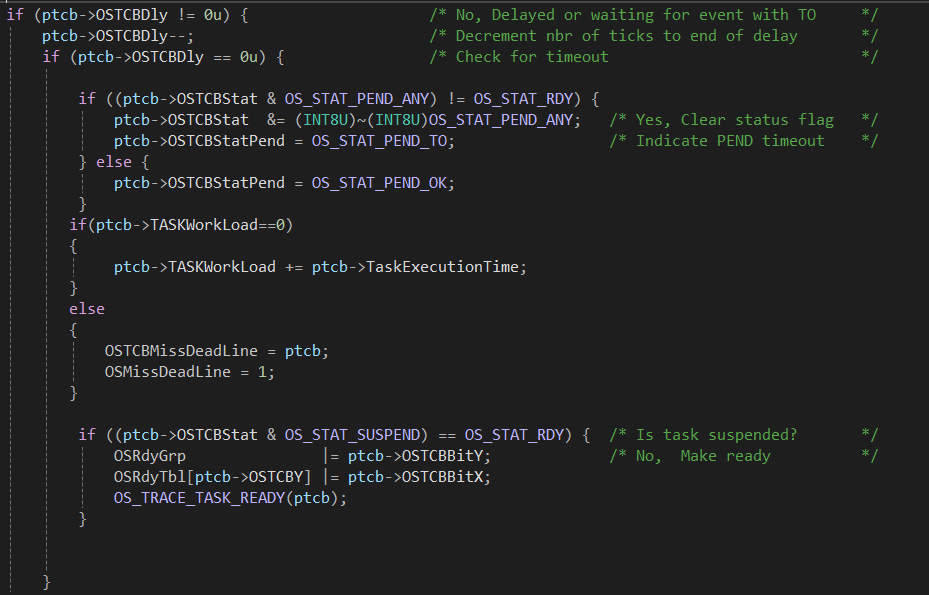
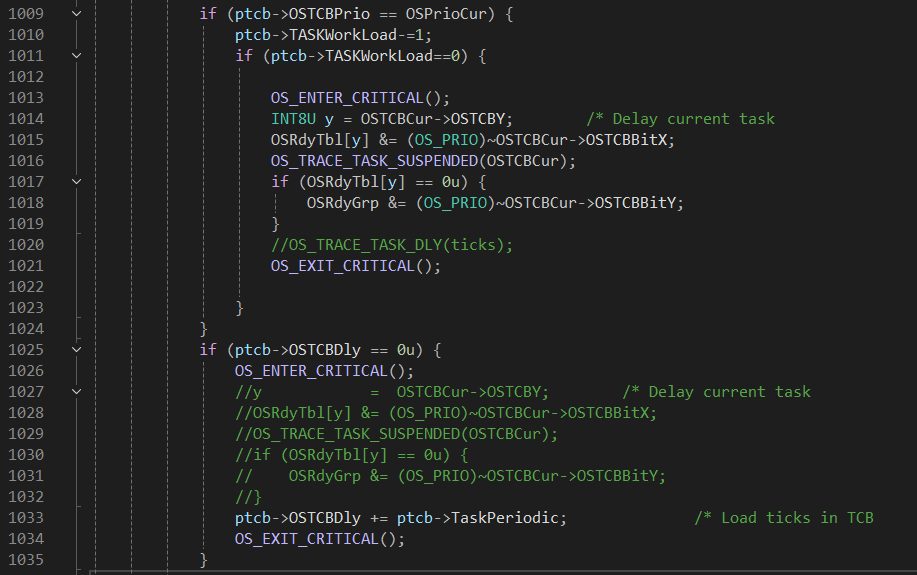
OSTaskCreateExt中，為TASK初始化一些多的參數，主要是運用TASKSET.TXT的資料，OSTCBDly會加上抵達時間的延遲，若有延遲，會將RdyTbl該TASK設為未就緒，沒有的話會直些賦予工作量。



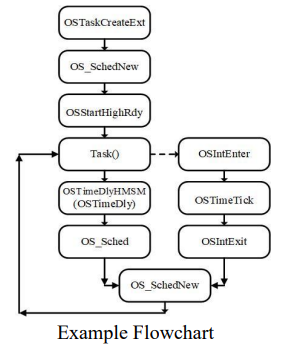


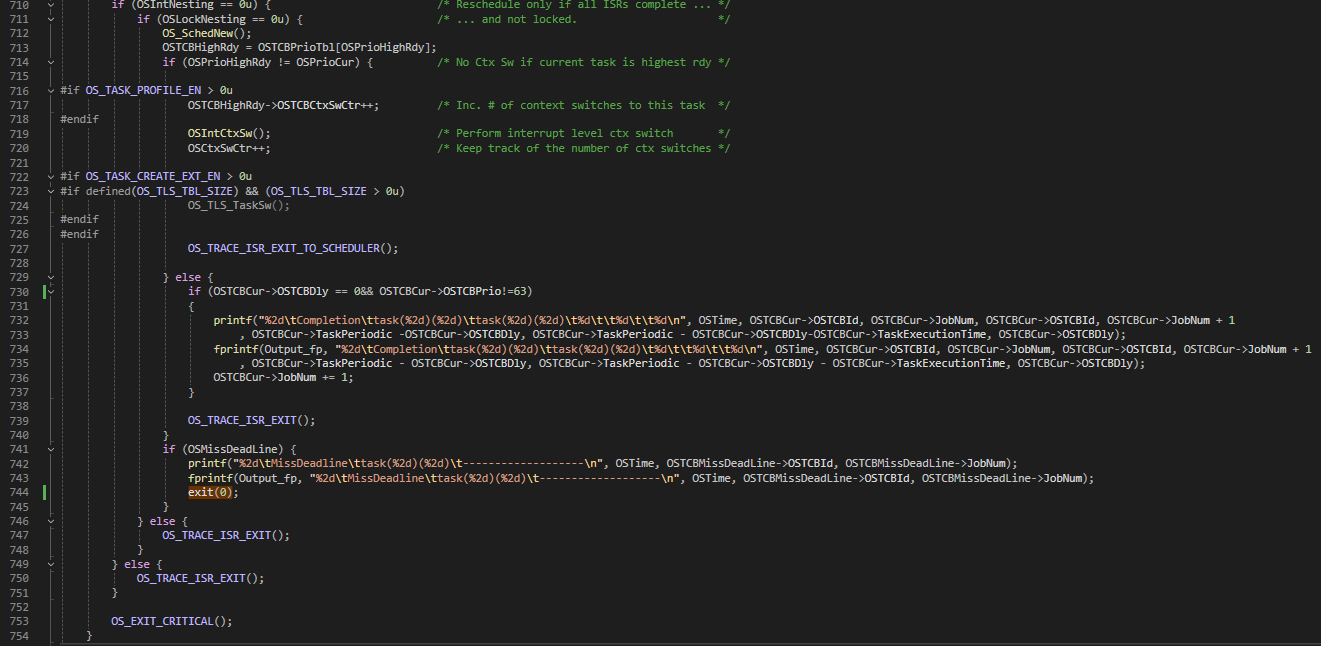
OSTimeDly本來被TASK使用後會讓其進入非預備狀態並OS\_sched()，這裡我用OSTimeGet()來讓他卡著直到下一Tick到來，並將相關排程事宜移到OSTimeTick()。

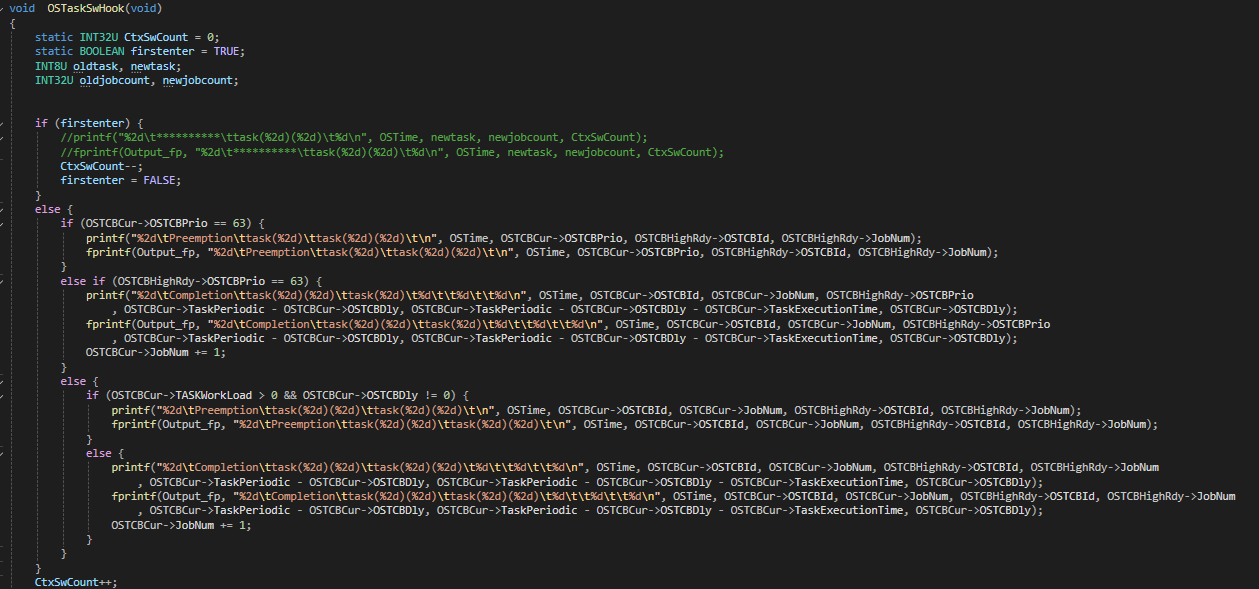




這裡我用TASKWorkLoad來判斷該JOB工作是否完成，如果變0則將其設為非預備狀態，OSTCBDLY則移到歸零時累加週期時間當DEADLINE並新增工作，若時間到但工作量未完成則視為MissDeadLine。

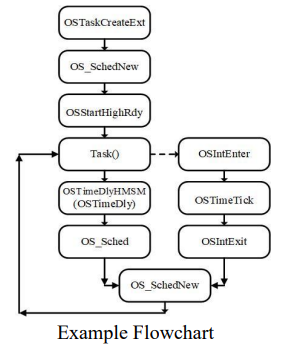


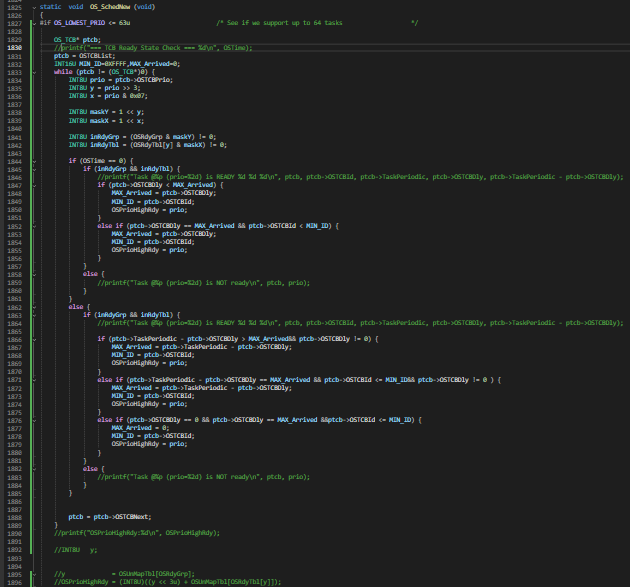




OSTimetick執行完後，查看新舊TASK是不是相同且剛好舊的週期歸零，是的話輸出TASK完成並執行新JOB，如果新舊不同則進入OSIntCtxSw()，裡面會Call OSTaskSwHook()，這裡會依據切換TASK時的狀況顯示是Completion還是被Preemption，最後查看有沒有MissDeadLine，有的話暫停程序並退出。

# PART III





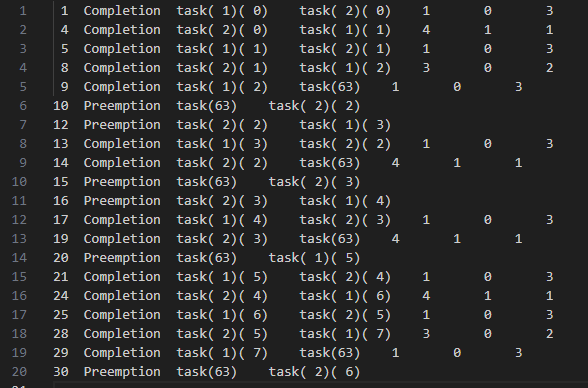
FIFO我主要用剛剛RM的專案稍微修改OS\_SchedNew的部分，我將選擇最高優先度TASK的方式改為檢查哪個TASK已抵達的時間最長，並會在TASK剛進入新周期和系統剛啟動時做特殊判斷

RM VS FIFO

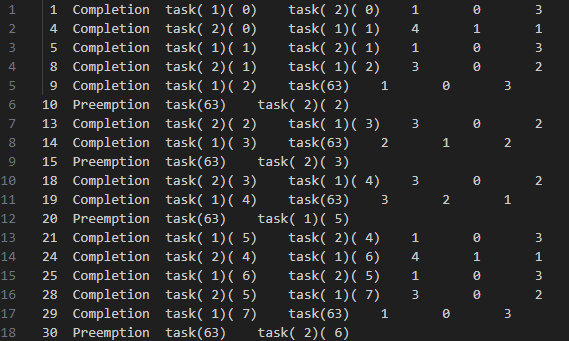
測資



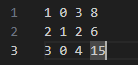
RM:



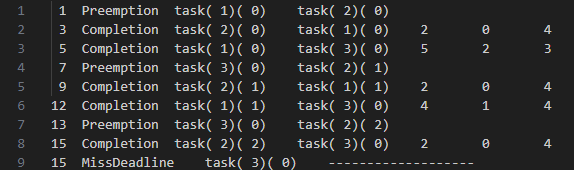
FIFO:

  
兩者都成功

測資



RM:



FIFO

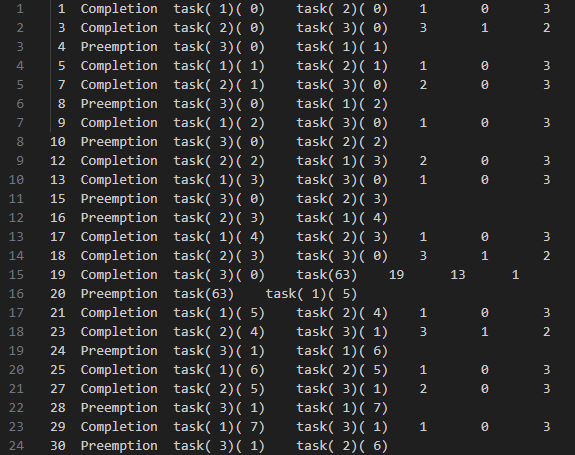


兩者都失敗

測資



RM:



FIFO



RM成功，FIFO失敗

從排程過程發現由於FIFO是JOB開始後就無法變更執行任務的，因此會出現過長執行時間的TASK佔領CPU導致MissDeadLine的問題