Embedded OS Implementation, Fall 2022

Project #1 (due November 2nd, 2022 (Wednesday) at 12:00)

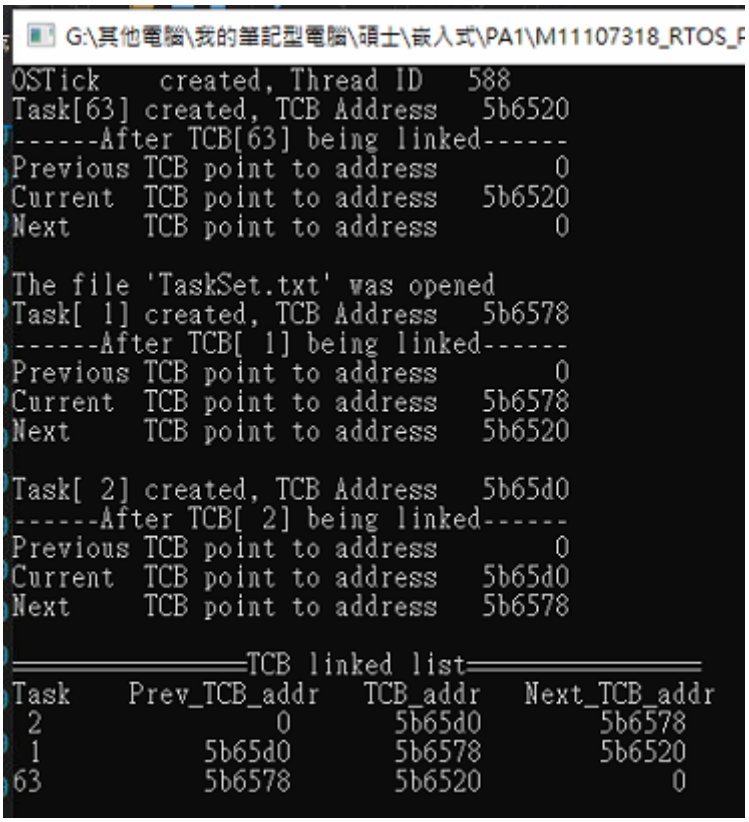
[PART I] Task Control Block Linked List

Objective:

Following the previous homework (HW1), please add some code to the μC/OS-II scheduler **in the kernel level** to observe the operations of the task control block (TCB) and TCB linked list.

※ The TCB address is dynamic.

- **The output results are shown below:**



A report that describes your implementation (please attach the screenshot of the code and **MARK** the modified part)：

在 Task 建置時，需要初始化 TCB，因此在 <os_core.c> 中的 OS_TCBInit 函式內，初始化後並 link 完的地方做 TCB 資訊的打印，%x 用來打印 16 進制的地址輸出，且輸出中的 ABCDEFG為小寫。(其中要注意指標變數 * 以及 -> 等的符號使用，否則會產生打印錯誤的狀況發生)：

```
2179  #if OS_TASK_CREATE_EXT_EN > 0u
2180  #if defined(OS_TLS_TBL_SIZE) && (OS_TLS_TBL_SIZE > 0u)
2181          for (j = 0u; j < OS_TLS_TBL_SIZE; j++) {
2182              ptcb->OSTCBTLSTbl[j] = (OS_TLS)0;
2183          }
2184          OS_TLS_TaskCreate(ptcb);                        /* Call TLS hook
2185  #endif
2186  #endif
2187
2188          OS_ENTER_CRITICAL();
2189          ptcb->OSTCBNext = OSTCBList;                    /* Link into TCB chain
2190          ptcb->OSTCBPrev = (OS_TCB *)0;
2191
2192
2193          if (OSTCBList != (OS_TCB *)0) {
2194              OSTCBList->OSTCBPrev = ptcb;
2195          }
2196          OSTCBList               = ptcb;
2197          OSRdyGrp               |= ptcb->OSTCBBitY;      /* Make task ready to run
2198          OSRdyTbl[ptcb->OSTCBY] |= ptcb->OSTCBBitX;
2199          OSTaskCtr++;                                    /* Increment the #tasks counter
2200          OS_TRACE_TASK_READY(ptcb);
2201          OS_EXIT_CRITICAL();
2202
2203          //AddedCodePA1part1
2204          TaskNum++;
2205          if (ptcb->OSTCBPrio == 63) {
2206              printf("Task[%2d] created, TCB Address\t%6x\n", ptcb->OSTCBPrio, ptcb);
2207              printf("------After TCB[%2d] being linked------\n", ptcb->OSTCBPrio);
2208          }
2209          else {
2210              printf("Task[%2d] created, TCB Address\t%6x\n", ptcb->OSTCBId, ptcb);
2211              printf("------After TCB[%2d] being linked------\n", ptcb->OSTCBId);
2212          }
2213          printf("Previous TCB point to address\t%6x\n",ptcb->OSTCBPrev);
2214          printf("Current  TCB point to address\t%6x\n", ptcb);
2215          printf("Next     TCB point to address\t%6x\n\n", ptcb->OSTCBNext);
```

為了確認 Linked task 的數量，因此在<os_core.c> 程式最上面宣告了靜態變數 TaskNum 去計算總共建置了多少 Task：

```
70  *********************************************************************
71  *                                          FUNCTION PROTOTYPES
72  *********************************************************************
73  */
74
75  static  void  OS_InitEventList(void);
76
77  static  void  OS_InitMisc(void);
78
79  static  void  OS_InitRdyList(void);
80
81  static  void  OS_InitTaskIdle(void);
82
83  #if OS_TASK_STAT_EN > 0u
84  static  void  OS_InitTaskStat(void);
85  #endif
86
87  static  void  OS_InitTCBList(void);
88
89  static  void  OS_SchedNew(void);
90
91  int TaskNum=0;//AddedCodePA1
```

最後在<os_core.c>的 OSStar t函式中，開始執行 Task 前 (OSStartHighRdy)，做 TCB linked list 的打印，而為了避免影響到原本的變數值，因此另外宣告了 OS_TCB *save 去做儲存並打印：

```
901     //AddedCodePA1part1
902     printf("================TCB linked list================\n");
903     printf("Task\tPrev_TCB_addr\tTCB_addr   Next_TCB_addr\n");
904     OS_TCB* save;
905     save = OSTCBList;
906
907     for (int i = 0; i < TaskNum; i++) {
908         if (save->OSTCBPrio == 63) {
909             printf("%2d           %6x\t  %6x\t%6x\t\n", save->OSTCBPrio, save->OSTCBPrev, save, save->OSTCBNext);
910         }
911         else {
912             printf("%2d           %6x\t  %6x\t%6x\t\n",save->OSTCBId, save->OSTCBPrev, save, save->OSTCBNext);
913         }
914         save = save->OSTCBNext;
915     }
916
917
918     OSStartHighRdy();                          /* Execute target specific code to start task     */
919
920 }
```

[PART II] RM Scheduler Implementation

Objective:

To implement the Rate Monotonic (RM) scheduler for periodic tasks and observe the scheduling behaviors.

Problem Definition:

Implement the following three task sets of periodic tasks. Add necessary code to the μC/OS-II scheduler **in the kernel level** to observe how the task suffers from the scheduler. We give the files for the parameter of the task.

Periodic Task Set = {τ_{ID} (ID, arrival time, execution time, period)}

Example Task Set 1 = {τ_1 (1, 1, 2, 4), τ_2 (2, 0, 4, 10)}

Example Task Set 2 = {τ_1 (1, 3, 4, 14), τ_2 (2, 0, 2, 8), τ_3 (3, 0, 4, 10), τ_4(4, 24, 2, 12)}

Example Task Set 3 = {τ_1 (1, 2, 2, 10), τ_2 (2, 1, 1, 5), τ_3 (3, 0, 8, 15)}

※ The priority of the task is set according to the RM scheduling rules.

The output results of Task Set 1:

Tick	Event	CurrentTask ID	NextTask ID	ResponseTime	#of ContextSwitch	PreemptionTime	OSTimeDly
1	Preemption	task(2)(0)	task(1)(0)				
3	Completion	task(1)(0)	task(2)(0)	2	2	0	2
5	Preemption	task(2)(0)	task(1)(1)				
7	Completion	task(1)(1)	task(2)(0)	2	2	0	2
8	Completion	task(2)(0)	task(63)	8	5	4	2
9	Preemption	task(63)	task(1)(2)				
11	Completion	task(1)(2)	task(2)(1)	2	2	0	2
13	Preemption	task(2)(1)	task(1)(3)				
15	Completion	task(1)(3)	task(2)(1)	2	2	0	2
17	Completion	task(2)(1)	task(1)(4)	7	4	3	3
19	Completion	task(1)(4)	task(63)	2	2	0	2
20	Preemption	task(63)	task(2)(2)				
21	Preemption	task(2)(2)	task(1)(5)				
23	Completion	task(1)(5)	task(2)(2)	2	2	0	2
25	Preemption	task(2)(2)	task(1)(6)				
27	Completion	task(1)(6)	task(2)(2)	2	2	0	2
28	Completion	task(2)(2)	task(63)	8	6	4	2
29	Preemption	task(63)	task(1)(7)				

1	Preemption	task(2)(0)	task(1)(0)				
3	Completion	task(1)(0)	task(2)(0)	2	2	0	2
5	Preemption	task(2)(0)	task(1)(1)				
7	Completion	task(1)(1)	task(2)(0)	2	2	0	2
8	Completion	task(2)(0)	task(63)	8	5	4	2
9	Preemption	task(63)	task(1)(2)				
11	Completion	task(1)(2)	task(2)(1)	2	2	0	2
13	Preemption	task(2)(1)	task(1)(3)				
15	Completion	task(1)(3)	task(2)(1)	2	2	0	2
17	Completion	task(2)(1)	task(1)(4)	7	4	3	3
19	Completion	task(1)(4)	task(63)	2	2	0	2
20	Preemption	task(63)	task(2)(2)				
21	Preemption	task(2)(2)	task(1)(5)				
23	Completion	task(1)(5)	task(2)(2)	2	2	0	2
25	Preemption	task(2)(2)	task(1)(6)				
27	Completion	task(1)(6)	task(2)(2)	2	2	0	2
28	Completion	task(2)(2)	task(63)	8	6	4	2
29	Preemption	task(63)	task(1)(7)				

The output results of Task Set 2:

Tick	Event	CurrentTask ID	NextTask ID	ResponseTime	#of ContextSwitch	PreemptionTime	OSTimeDly
2	Completion	task(2)(0)	task(3)(0)	2	1	0	6
6	Completion	task(3)(0)	task(1)(0)	6	2	2	4
8	Preemption	task(1)(0)	task(2)(1)				
10	Completion	task(2)(1)	task(3)(1)	2	2	0	6
14	Completion	task(3)(1)	task(1)(0)	4	2	0	6
16	Completion	task(1)(0)	task(2)(2)	13	5	9	1
18	Completion	task(2)(2)	task(1)(1)	2	2	0	6
20	Preemption	task(1)(1)	task(3)(2)				
24	Completion	task(3)(2)	task(2)(3)	4	2	0	6
26	Completion	task(2)(3)	task(4)(0)	2	2	0	6
28	Completion	task(4)(0)	task(1)(1)	4	2	2	8
30	Completion	task(1)(1)	task(3)(3)	13	6	9	1

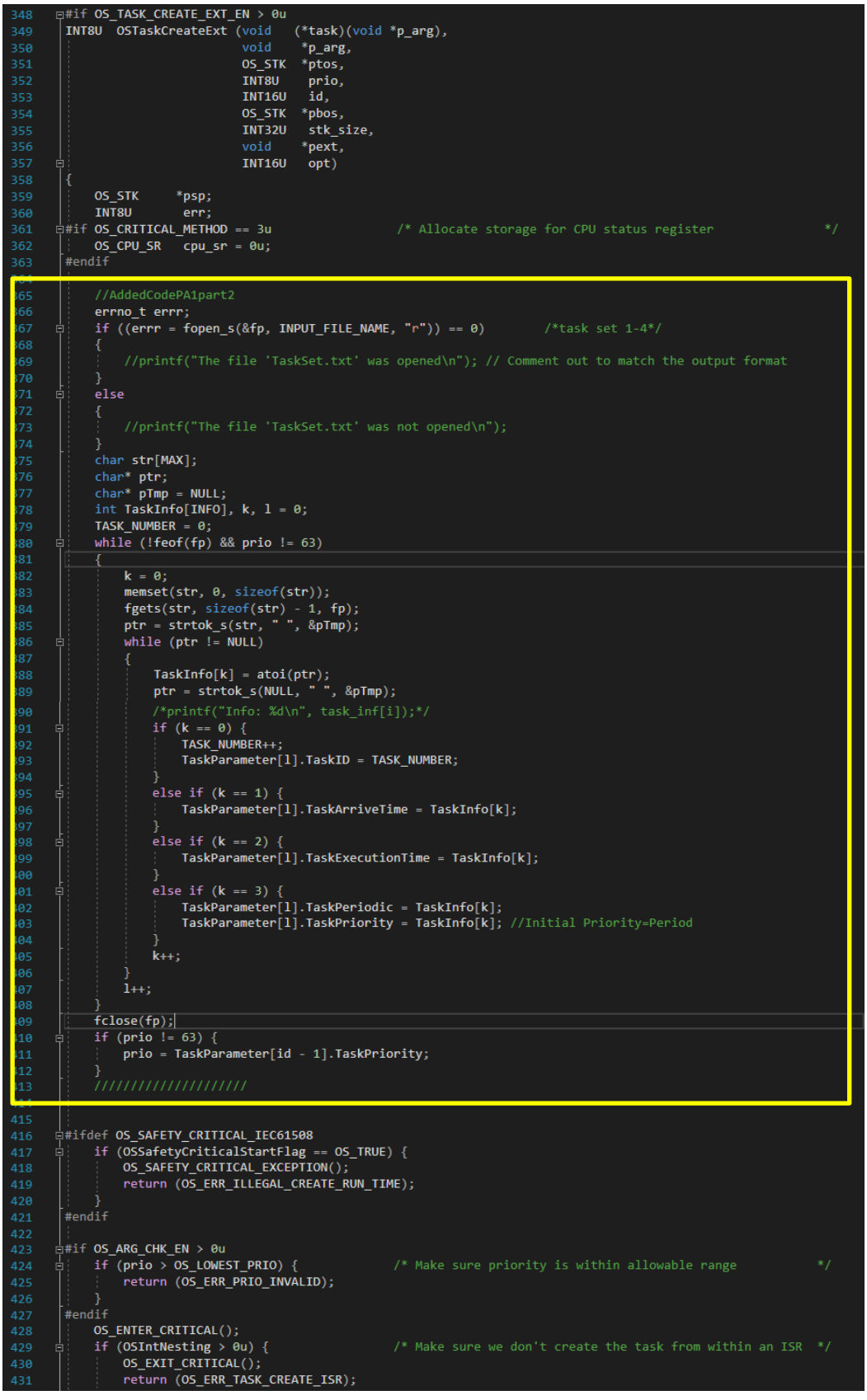
2	Completion	task(2)(0)	task(3)(0)	2	1	0	6
6	Completion	task(3)(0)	task(1)(0)	6	2	2	4
8	Preemption	task(1)(0)	task(2)(1)				
10	Completion	task(2)(1)	task(3)(1)	2	2	0	6
14	Completion	task(3)(1)	task(1)(0)	4	2	0	6
16	Completion	task(1)(0)	task(2)(2)	13	4	9	1
18	Completion	task(2)(2)	task(1)(1)	2	2	0	6
20	Preemption	task(1)(1)	task(3)(2)				
24	Completion	task(3)(2)	task(2)(3)	4	2	0	6
26	Completion	task(2)(3)	task(4)(0)	2	2	0	6
28	Completion	task(4)(0)	task(1)(1)	4	2	2	8
30	Completion	task(1)(1)	task(3)(3)	13	4	9	1

The output results of Task Set 3:

Tick	Event	CurrentTask ID	NextTask ID	ResponseTime	#of ContextSwitch	PreemptionTime	OSTimeDly
1	Preemption	task(3)(0)	task(2)(0)				
2	Completion	task(2)(0)	task(1)(0)	1	2	0	4
4	Completion	task(1)(0)	task(3)(0)	2	2	0	8
6	Preemption	task(3)(0)	task(2)(1)				
7	Completion	task(2)(1)	task(3)(0)	1	2	0	4
11	Preemption	task(3)(0)	task(2)(2)				
12	Completion	task(2)(2)	task(1)(1)	1	2	0	4
14	Completion	task(1)(1)	task(3)(0)	2	2	0	8
15	Completion	task(3)(0)	task(3)(0)	15	9	7	0
16	Preemption	task(3)(1)	task(2)(3)				
17	Completion	task(2)(3)	task(3)(1)	1	2	0	4
21	Preemption	task(3)(1)	task(2)(4)				
22	Completion	task(2)(4)	task(1)(2)	1	2	0	4
24	Completion	task(1)(2)	task(3)(1)	2	2	0	8
26	Preemption	task(3)(1)	task(2)(5)				
27	Completion	task(2)(5)	task(3)(1)	1	2	0	4
28	Completion	task(3)(1)	task(63)	13	9	5	2
30	Preemption	task(63)	task(3)(2)				

1	Preemption	task(3)(0)	task(2)(0)				
2	Completion	task(2)(0)	task(1)(0)	1	2	0	4
4	Completion	task(1)(0)	task(3)(0)	2	2	0	8
6	Preemption	task(3)(0)	task(2)(1)				
7	Completion	task(2)(1)	task(3)(0)	1	2	0	4
11	Preemption	task(3)(0)	task(2)(2)				
12	Completion	task(2)(2)	task(1)(1)	1	2	0	4
14	Completion	task(1)(1)	task(3)(0)	2	2	0	8
15	Completion	task(3)(0)	task(3)(0)	15	9	7	0
16	Preemption	task(3)(1)	task(2)(3)				
17	Completion	task(2)(3)	task(3)(1)	1	2	0	4
21	Preemption	task(3)(1)	task(2)(4)				
22	Completion	task(2)(4)	task(1)(2)	1	2	0	4
24	Completion	task(1)(2)	task(3)(1)	2	2	0	8
26	Preemption	task(3)(1)	task(2)(5)				
27	Completion	task(2)(5)	task(3)(1)	1	2	0	4
28	Completion	task(3)(1)	task(63)	13	9	5	2
30	Preemption	task(63)	task(3)(2)				

在最一開始先設定Priority=Period去更改優先權，而在<os_task.c>副程式 OSTaskCreateExt 中，去再次讀檔並更動 priority，由於優先權和週期成反比，優先度越低，週期越大，因此在這邊簡單設定 TaskParameter[l].TaskPriority 和 TaskParameter[l].TaskPeriod 相等。



```
#if OS_TASK_CREATE_EXT_EN > 0u
INT8U  OSTaskCreateExt (void   (*task)(void *p_arg),
                        void    *p_arg,
                        OS_STK  *ptos,
                        INT8U    prio,
                        INT16U   id,
                        OS_STK  *pbos,
                        INT32U   stk_size,
                        void    *pext,
                        INT16U   opt)
{
    OS_STK     *psp;
    INT8U       err;
#if OS_CRITICAL_METHOD == 3u                 /* Allocate storage for CPU status register              */
    OS_CPU_SR   cpu_sr = 0u;
#endif

    //AddedCodePA1part2
    errno_t errr;
    if ((errr = fopen_s(&fp, INPUT_FILE_NAME, "r")) == 0)        /*task set 1-4*/
    {
        //printf("The file 'TaskSet.txt' was opened\n"); // Comment out to match the output format
    }
    else
    {
        //printf("The file 'TaskSet.txt' was not opened\n");
    }
    char str[MAX];
    char* ptr;
    char* pTmp = NULL;
    int TaskInfo[INFO], k, l = 0;
    TASK_NUMBER = 0;
    while (!feof(fp) && prio != 63)
    {
        k = 0;
        memset(str, 0, sizeof(str));
        fgets(str, sizeof(str) - 1, fp);
        ptr = strtok_s(str, " ", &pTmp);
        while (ptr != NULL)
        {
            TaskInfo[k] = atoi(ptr);
            ptr = strtok_s(NULL, " ", &pTmp);
            /*printf("Info: %d\n", task_inf[i]);*/
            if (k == 0) {
                TASK_NUMBER++;
                TaskParameter[l].TaskID = TASK_NUMBER;
            }
            else if (k == 1) {
                TaskParameter[l].TaskArriveTime = TaskInfo[k];
            }
            else if (k == 2) {
                TaskParameter[l].TaskExecutionTime = TaskInfo[k];
            }
            else if (k == 3) {
                TaskParameter[l].TaskPeriodic = TaskInfo[k];
                TaskParameter[l].TaskPriority = TaskInfo[k]; //Initial Priority=Period
            }
            k++;
        }
        l++;
    }
    fclose(fp);
    if (prio != 63) {
        prio = TaskParameter[id - 1].TaskPriority;
    }
    ////////////////////


#ifdef OS_SAFETY_CRITICAL_IEC61508
    if (OSSafetyCriticalStartFlag == OS_TRUE) {
        OS_SAFETY_CRITICAL_EXCEPTION();
        return (OS_ERR_ILLEGAL_CREATE_RUN_TIME);
    }
#endif

#if OS_ARG_CHK_EN > 0u
    if (prio > OS_LOWEST_PRIO) {             /* Make sure priority is within allowable range          */
        return (OS_ERR_PRIO_INVALID);
    }
#endif
    OS_ENTER_CRITICAL();
    if (OSIntNesting > 0u) {                 /* Make sure we don't create the task from within an ISR  */
        OS_EXIT_CRITICAL();
        return (OS_ERR_TASK_CREATE_ISR);
```

在 Task 建立之後，需要初始化 TCB ，因此在<os_core.c>的 OS_TCBInit 再次讀檔，並在此更動 Priority Table，使抵達時間不是0秒的 Task能先不要 ready 並做 delay 之設定。

```
2157    INT8U  OS_TCBInit (INT8U    prio,
2158                       OS_STK  *ptos,
2159                       OS_STK  *pbos,
2160                       INT16U   id,
2161                       INT32U   stk_size,
2162                       void    *pext,
2163                       INT16U   opt)
2164    {
2165        OS_TCB    *ptcb;
2166    #if OS_CRITICAL_METHOD == 3u                          /* Allocate storage for CPU status register */
2167        OS_CPU_SR  cpu_sr = 0u;
2168    #endif
2169    #if OS_TASK_REG_TBL_SIZE > 0u
2170        INT8U      i;
2171    #endif
2172    #if OS_TASK_CREATE_EXT_EN > 0u
2173    #if defined(OS_TLS_TBL_SIZE) && (OS_TLS_TBL_SIZE > 0u)
2174        INT8U      j;
2175    #endif
2176    #endif
2177
2178
2179        OS_ENTER_CRITICAL();
2180        ptcb = OSTCBFreeList;                             /* Get a free TCB from the free TCB list    */
2181        if (ptcb != (OS_TCB *)0) {
2182            OSTCBFreeList            = ptcb->OSTCBNext;   /* Update pointer to free TCB list          */
2183            OS_EXIT_CRITICAL();
2184            ptcb->OSTCBStkPtr        = ptos;              /* Load Stack pointer in TCB                */
2185            ptcb->OSTCBPrio          = prio;              /* Load task priority into TCB              */
2186            ptcb->OSTCBStat          = OS_STAT_RDY;       /* Task is ready to run                     */
2187            ptcb->OSTCBStatPend      = OS_STAT_PEND_OK;   /* Clear pend status                        */
2188            ptcb->OSTCBDly           = 0u;                /* Task is not delayed                      */
2189
2190    #if OS_TASK_CREATE_EXT_EN > 0u
2191            ptcb->OSTCBExtPtr        = pext;              /* Store pointer to TCB extension           */
2192            ptcb->OSTCBStkSize       = stk_size;          /* Store stack size                         */
2193            ptcb->OSTCBStkBottom     = pbos;              /* Store pointer to bottom of stack         */
2194            ptcb->OSTCBOpt           = opt;               /* Store task options                       */
2195            ptcb->OSTCBId            = id;                /* Store task ID                            */
2196    #else
2197            pext                     = pext;              /* Prevent compiler warning if not used     */
2198            stk_size                 = stk_size;
2199            pbos                     = pbos;
2200            opt                      = opt;
2201            id                       = id;
2202    #endif
2203
2204            //AddedCodePA1part2
2205            errno_t errr;
2206            if ((errr = fopen_s(&fp, INPUT_FILE_NAME, "r")) == 0)        /*task set 1-4*/
2207            {
2208                //printf("The file 'TaskSet.txt' was opened\n"); // Comment out to match the output format
2209            }
2210            else
2211            {
2212                //printf("The file 'TaskSet.txt' was not opened\n");
2213            }
2214            char str[MAX];
2215            char* ptr;
2216            char* pTmp = NULL;
2217            int TaskInfo[INFO], k, l = 0;
2218            TASK_NUMBER = 0;
2219            while (!feof(fp) && prio != 63)
2220            {
2221                k = 0;
2222                memset(str, 0, sizeof(str));
2223                fgets(str, sizeof(str) - 1, fp);
2224                ptr = strtok_s(str, " ", &pTmp);
2225                while (ptr != NULL)
2226                {
2227                    TaskInfo[k] = atoi(ptr);
2228                    ptr = strtok_s(NULL, " ", &pTmp);
2229                    /*printf("Info: %d\n", task_inf[i]);*/
2230                    if (k == 0) {
```

```
2231                    TASK_NUMBER++;
2232                    TaskParameter[l].TaskID = TASK_NUMBER;
2233                }
2234                else if (k == 1) {
2235                    TaskParameter[l].TaskArriveTime = TaskInfo[k];
2236                }
2237                else if (k == 2) {
2238                    TaskParameter[l].TaskExecutionTime = TaskInfo[k];
2239                }
2240                else if (k == 3) {
2241                    TaskParameter[l].TaskPeriodic = TaskInfo[k];
2242                    TaskParameter[l].TaskPriority = TaskInfo[k]; //Initial Priority=Period
2243                }
2244                k++;
2245            }
2246            l++;
2247        }
2248        fclose(fp);
2249        if (prio != 63) {
2250            unsigned int delay = TaskParameter[id - 1].TaskArriveTime;
2251            unsigned int exetime = TaskParameter[id - 1].TaskExecutionTime;
2252            ptcb->OSTCBCyclesExecution = exetime;
2253            ptcb->OSTCBCyclesCount = 0u;
2254            ptcb->OSTCBJobNumber = 0u;
2255            ptcb->OSTCBCyclesEnd = 0u;
2256            ptcb->OSTCBCyclesSwitchStart = 0u;
2257            ptcb->OSTCBCyclesPeriod = TaskParameter[id - 1].TaskPeriodic;
2258            ptcb->OSTCBCyclesArrive = TaskParameter[id - 1].TaskArriveTime;
2259            while (ptcb->OSTCBDly != delay) {
2260                ptcb->OSTCBDly++;
2261            }
2262        }
2263        ////////////////////
2264
2265    #if OS_TASK_DEL_EN > 0u
2266        ptcb->OSTCBDelReq        = OS_ERR_NONE;
2267    #endif
2268
2269    #if OS_LOWEST_PRIO <= 63u                                        /* Pre-compute X, Y                     */
2270        ptcb->OSTCBY             = (INT8U)(prio >> 3u);
2271        ptcb->OSTCBX             = (INT8U)(prio & 0x07u);
2272    #else                                                            /* Pre-compute X, Y                     */
2273        ptcb->OSTCBY             = (INT8U)((INT8U)(prio >> 4u) & 0xFFu);
2274        ptcb->OSTCBX             = (INT8U) (prio & 0x0Fu);
2275    #endif
2276                                                                     /* Pre-compute BitX and BitY            */
2277        ptcb->OSTCBBitY          = (OS_PRIO)(1uL << ptcb->OSTCBY);
2278        ptcb->OSTCBBitX          = (OS_PRIO)(1uL << ptcb->OSTCBX);
```

阻止一開始就將 Task 設定為Ready（藉由新增一個 if 判斷ArriveTime是否為0u，若為0u或是為 idle Task 就設定為Ready）：

```
2338
2339        //ModifyCodePA1part2
2340        if (prio == 63 || ptcb->OSTCBCyclesArrive == 0u) {
2341            OSRdyGrp |= ptcb->OSTCBBitY;                    /* Make task ready to run               */
2342            OSRdyTbl[ptcb->OSTCBY] |= ptcb->OSTCBBitX;
2343        }
2344
2345
2346        OSTaskCtr++;                                        /* Increment the #tasks counter         */
2347        OS_TRACE_TASK_READY(ptcb);
2348        OS_EXIT_CRITICAL();
```

由於會需要大量參數進行運算，因此在<ucos_ii.h>中先定義參數，並於上一部份的<os_core.c> 的 OS_TCBInit 中進行初始化：

```
659 #if OS_TASK_PROFILE_EN > 0u
660     INT32U           OSTCBCtxSwCtr;         /* Number of time the task was switched in                  */
661     INT32U           OSTCBCyclesTot;        /* Total number of clock cycles the task has been running   */
662     INT32U           OSTCBCyclesStart;      /* Snapshot of cycle counter at start of task resumption    */
663     OS_STK          *OSTCBStkBase;          /* Pointer to the beginning of the task stack               */
664     INT32U           OSTCBStkUsed;          /* Number of bytes used from the stack                      */
665     //AddedCodePA1part2
666     INT32U           OSTCBCyclesExecution;  /* Setting about Execution Time */
667     INT32U           OSTCBCyclesCount;      /* Count Cycles */
668     INT32U           OSTCBCyclesArrive;     /* When arrive? */
669     INT32U           OSTCBCyclesEnd;        /* When cycle end? */
670     INT32U           OSTCBJobNumber;        /* Executing the N-th Job */
671     INT32U           OSTCBCyclesPeriod;     /* Period */
672     INT32U           OSTCBCyclesSwitchStart;/* To know #swich the cycle start */
673 #endif
```

在 OSStart 中設定輸出格式的欄位：

```
905 void  OSStart (void)
906 {
907     if (OSRunning == OS_FALSE) {
908         OS_SchedNew();                             /* Find highest priority's task priority number   */
909         OSPrioCur     = OSPrioHighRdy;
910         OSTCBHighRdy  = OSTCBPrioTbl[OSPrioHighRdy]; /* Point to highest priority task ready to run    */
911         OSTCBCur      = OSTCBHighRdy;
912 
913         //AddedCodePA1part1
914         printf("================TCB linked list===============\n");
915         printf("Task\tPrev_TCB_addr\tTCB_addr   Next_TCB_addr\n");
916         OS_TCB* save;
917         save = OSTCBList;
918 
919         for (int i = 0; i < TaskNum; i++) {
920             if (save->OSTCBPrio == 63) {
921                 printf("%2d          %6x\t  %6x\t%6x\t\n", save->OSTCBPrio, save->OSTCBPrev, save, save->OSTCBNext);
922             }
923             else {
924                 printf("%2d          %6x\t  %6x\t%6x\t\n", save->OSTCBId, save->OSTCBPrev, save, save->OSTCBNext);
925             }
926             save = save->OSTCBNext;
927         }
928 
929         //AddedCodeHW1
930         /*if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0)
931         {
932             printf("\nTick\tCurrentTask ID\tNextTask ID\tCaller\n");
933             printf("%2d\t**********\ttask(%2d)(%2d)\tOSStart ()\n", OSTime, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBCtxSwCtr);
934             fprintf(Output_fp, "%2d\t**********\ttask(%2d)(%2d)\tOSStart ()\n", OSTime, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBCtxSwCtr);
935         }
936         fclose(Output_fp);*/
937 
938         //AddedCodePA1part2
939         printf("\nTick\tEvent         CurrentTask ID\t NextTask ID\tResponseTime  #of ContextSwitch\t PreemptionTime\t OSTimeDly\n");
940 
941 
942         OSStartHighRdy();                          /* Execute target specific code to start task     */
943 
944     }
945 }
```

在會進行Context Switch 的 OS_Sched中進行相關參數的打印，其中

Response Time = Arrive Time + Job Number * Period

Context Switch = Total Context Switch –Total Context Switch Number when the task start

Preemption Time = End Time – (Arrive Time + Job Number * Period + Setting Execution Time)

Delay Time = Arrive Time + Job Number * Period – Execution End Time

並且要注意當 HighRdyTask 結束時 (也就是已完成執行時間OSTCBCyclesTot==0)，要將 OSTCBCyclesStart 和 OSTCBCyclesSwitchStart 重置。除此之外，OS_Sched 多發生在前一個任務完成時，要尋找下一個任務。

```
void  OS_Sched (void)
{
#if OS_CRITICAL_METHOD == 3u                       /* Allocate storage for CPU status register     */
    OS_CPU_SR  cpu_sr = 0u;
#endif

    OS_ENTER_CRITICAL();
    if (OSIntNesting == 0u) {                      /* Schedule only if all ISRs done and ...       */
        if (OSLockNesting == 0u) {                 /* ... scheduler is not locked                  */
            OS_SchedNew();
            OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy];
            if (OSPrioHighRdy != OSPrioCur) {          /* No Ctx Sw if current task is highest rdy     */
#if OS_TASK_PROFILE_EN > 0u

                OSTCBHighRdy->OSTCBCtxSwCtr++;         /* Inc. # of context switches to this task      */
            //if (OSPrioHighRdy != OSPrioCur) {          /* No Ctx Sw if current task is highest rdy     */
#if OS_TASK_PROFILE_EN > 0u

                // AddedCodeHW1
                //if (OSTCBHighRdy->OSTCBCtxSwCtr == 1 && OSCtxSwCtr == 1) {
                //    OSTCBHighRdy->OSTCBCtxSwCtr = OSTCBHighRdy->OSTCBCtxSwCtr - 1;
                //}
                //if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0)
                //{
                //    if (OSTCBHighRdy->OSTCBPrio == 63) {
                //        printf("%2d\ttask(% 2d)(%2d)\ttask(%2d)\tOS_Sched ()\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr, OSTCBHighRdy->OSTCBPrio);
                //        fprintf(Output_fp, "%2d\ttask(% 2d)(%2d)\ttask(%2d)     \tOS_Sched ()\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr, OSTCBHighRdy->OSTCBPrio);
                //    }
                //    else {
                //        printf("%2d\ttask(%2d)(%2d)\ttask(%2d)(%2d)\tOS_Sched ()\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBCtxSwCtr);
                //        fprintf(Output_fp, "%2d\ttask(%2d)(%2d)\ttask(%2d)(%2d)\tOS_Sched ()\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBCtxSwCtr);
                //    }
                //}
                //fclose(Output_fp);


                //AddedCodePA1part2
                //printf("%2d Task %2d SwitchStart%2d Total%2d OS_Sched\n", OSTimeGet(), OSTCBCur->OSTCBId, OSTCBCur->OSTCBCyclesSwitchStart, OSCtxSwCtr);
                OSTCBCur->OSTCBCyclesEnd = OSTimeGet();
                OSTCBCur->OSTCBDly = (OSTCBCur->OSTCBCyclesArrive);
                OSTCBCur->OSTCBDly = OSTCBCur->OSTCBDly + (((OSTCBCur->OSTCBJobNumber+1)) * (OSTCBCur->OSTCBCyclesPeriod));
                OSTCBCur->OSTCBDly = OSTCBCur->OSTCBDly - (OSTCBCur->OSTCBCyclesEnd);
                if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0){
                    if (OSTCBHighRdy->OSTCBPrio == 63) {
                        printf("%2d\tCompletion\t task(%2d)(%2d)    task(%2d)", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBJobNumber, OSTCBHighRdy->OSTCBPrio);
                        printf("            %2d", OSTimeGet() - (OSTCBCur->OSTCBCyclesArrive + OSTCBCur->OSTCBJobNumber * OSTCBCur->OSTCBCyclesPeriod));
                        printf("            %2d", OSCtxSwCtr - OSTCBCur->OSTCBCyclesSwitchStart);
                        printf("            %2d", OSTCBCur->OSTCBCyclesEnd - (OSTCBCur->OSTCBCyclesArrive + OSTCBCur->OSTCBJobNumber * OSTCBCur->OSTCBCyclesPeriod + OSTCBCur->OSTCBCyclesExecution));
                        printf("            %2d\n", OSTCBCur->OSTCBDly);
                        fprintf(Output_fp, "%2d\tCompletion\t task(%2d)(%2d)    task(%2d)", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBJobNumber, OSTCBHighRdy->OSTCBPrio);
                        fprintf(Output_fp, "            %2d", OSTimeGet() - (OSTCBCur->OSTCBCyclesArrive + OSTCBCur->OSTCBJobNumber * OSTCBCur->OSTCBCyclesPeriod));
                        fprintf(Output_fp, "            %2d", OSCtxSwCtr - OSTCBCur->OSTCBCyclesSwitchStart);
                        fprintf(Output_fp, "            %2d", OSTCBCur->OSTCBCyclesEnd - (OSTCBCur->OSTCBCyclesArrive + OSTCBCur->OSTCBJobNumber * OSTCBCur->OSTCBCyclesPeriod + OSTCBCur->OSTCBCyclesExecution));
                        fprintf(Output_fp, "            %2d\n", OSTCBCur->OSTCBDly);
                    }
                    else {
                        printf("%2d\tCompletion\t task(%2d)(%2d)    task(%2d)(%2d)\t", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBJobNumber, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBJobNumber);
                        printf("      %2d", OSTimeGet() - (OSTCBCur->OSTCBCyclesArrive + OSTCBCur->OSTCBJobNumber * OSTCBCur->OSTCBCyclesPeriod));
                        printf("            %2d", OSCtxSwCtr - OSTCBCur->OSTCBCyclesSwitchStart);
                        printf("            %2d", OSTCBCur->OSTCBCyclesEnd - (OSTCBCur->OSTCBCyclesArrive + OSTCBCur->OSTCBJobNumber * OSTCBCur->OSTCBCyclesPeriod + OSTCBCur->OSTCBCyclesExecution));
                        printf("            %2d\n", OSTCBCur->OSTCBDly);
                        fprintf(Output_fp, "%2d\tCompletion \t task(%2d)(%2d)    task(%2d)(%2d)\t", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBJobNumber, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBJobNumber);
                        fprintf(Output_fp, "      %2d", OSTimeGet() - (OSTCBCur->OSTCBCyclesArrive + OSTCBCur->OSTCBJobNumber * OSTCBCur->OSTCBCyclesPeriod));
                        fprintf(Output_fp, "            %2d", OSCtxSwCtr - OSTCBCur->OSTCBCyclesSwitchStart);
                        fprintf(Output_fp, "            %2d", OSTCBCur->OSTCBCyclesEnd - (OSTCBCur->OSTCBCyclesArrive + OSTCBCur->OSTCBJobNumber * OSTCBCur->OSTCBCyclesPeriod + OSTCBCur->OSTCBCyclesExecution));
                        fprintf(Output_fp, "            %2d\n", OSTCBCur->OSTCBDly);
                        //printf("%2d\t", OSTCBCur->OSTCBCyclesEnd - OSTCBCur->OSTCBJobNumber * OSTCBCur->OSTCBCyclesExecution);
                        //printf("%2d\t\n", OSTCBCur->OSTCBDly);
                    }
                    //printf("End%2d\n", OSTCBCur->OSTCBCyclesEnd);
                    //printf("Delay%2d\n", OSTCBCur->OSTCBDly);
                }
                fclose(Output_fp);
                OSTCBCur->OSTCBCyclesTot = 0;
                if (OSTCBHighRdy->OSTCBCyclesTot == 0) {
                    OSTCBHighRdy->OSTCBCyclesStart = OSTimeGet() + 1;
                    OSTCBHighRdy->OSTCBCyclesSwitchStart = OSCtxSwCtr - 1;
                }
                OSTCBCur->OSTCBJobNumber++;


#if OS_TASK_CREATE_EXT_EN > 0u
#if defined(OS_TLS_TBL_SIZE) && (OS_TLS_TBL_SIZE > 0u)
                OS_TLS_TaskSw();
#endif
#endif
```

在 OSIntExit 中大多是因為打斷了當前正在執行的 task，而因此進來執行優先權較高的 task ，所以在這裡除了打印之外，在HighRdyTask 結束時也要進行OSTCBCyclesStart 和 OSTCBCyclesSwitchStart 重置。

```
696  void  OSIntExit (void)
697  {
698  #if OS_CRITICAL_METHOD == 3u                          /* Allocate storage for CPU status register */
699      OS_CPU_SR  cpu_sr = 0u;
700  #endif
701
702      if (OSRunning == OS_TRUE) {
703          OS_ENTER_CRITICAL();
704          if (OSIntNesting > 0u) {                      /* Prevent OSIntNesting from wrapping       */
705              OSIntNesting--;
706          }
707          if (OSIntNesting == 0u && OSTCBCur->OSTCBCyclesTot != OSTCBCur->OSTCBCyclesExecution) {                      /* Reschedule only if all ISRs complete ...,ModifiedCodePA1part2 */
708              if (OSLockNesting == 0u) {                /* ... and not locked.                      */
709                  OS_SchedNew();
710                  OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy];
711                  if (OSPrioHighRdy != OSPrioCur) {     /* No Ctx Sw if current task is highest rdy */
712  #if OS_TASK_PROFILE_EN > 0u
713                      OSTCBHighRdy->OSTCBCtxSwCtr++;    /* Inc. # of context switches to this task  */
714  #endif
715                      OSCtxSwCtr++;                     /* Keep track of the number of ctx switches */
716
717                      //AddedCodeHW1
718                      //if (OSTCBHighRdy->OSTCBCtxSwCtr == 1 && OSCtxSwCtr == 1) {
719                      //    OSTCBHighRdy->OSTCBCtxSwCtr = OSTCBHighRdy->OSTCBCtxSwCtr - 1;
720                      //}
721                      //if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0)
722                      //{
723                      //    if (OSTCBCur->OSTCBPrio == 63) {
724                      //        printf("%2d\ttask(%2d)\ttask(%2d)(%2d)\tOSIntExit ()\n", OSTime, OSTCBCur->OSTCBPrio, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBCtxSwCtr);
725                      //        //fprintf(Output_fp, "%2d\ttask(%2d)     \ttask(%2d)(%2d)\tOSIntExit ()\n", OSTime, OSTCBCur->OSTCBPrio, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBCtxSwCtr);
726                      //    }
727                      //    else {
728                      //        printf("%2d\ttask(%2d)(%2d)\ttask(%2d)(%2d)\tOSIntExit ()\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBCtxSwCtr);
729                      //        //fprintf(Output_fp, "%2d\ttask(%2d)(%2d)\ttask(%2d)(%2d)\tOSIntExit ()\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBCtxSwCtr);
730                      //    }
731                      //}
732                      //fclose(Output_fp);

34                       //AddedCodePA1part2
35                       if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0) {
36                           if (OSTCBHighRdy->OSTCBCyclesTot == 0) {
37                               OSTCBHighRdy->OSTCBCyclesStart = OSTimeGet() + 1;
38                               OSTCBHighRdy->OSTCBCyclesSwitchStart = OSCtxSwCtr - 1;
39                           }
40                           if (OSTCBCur->OSTCBPrio == 63) {
41                               printf("%2d\tPreemption\t task(%2d)        task(%2d)(%2d)\n", OSTime, OSTCBCur->OSTCBPrio, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBJobNumber);
42                               fprintf(Output_fp, "%2d\tPreemption\t task(%2d)        task(%2d)(%2d)\n", OSTime, OSTCBCur->OSTCBPrio, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBJobNumber);
43                           }
44                           else {
45                               printf("%2d\tPreemption\t task(%2d)(%2d)    task(%2d)(%2d)\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBJobNumber, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBJobNumber);
46                               fprintf(Output_fp, "%2d\tPreemption\t task(%2d)(%2d)    task(%2d)(%2d)\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBJobNumber, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBJobNumber);
47                           }
48                       }
49                       fclose(Output_fp);
50
751
```

每一秒都會進入OSTimeTick，因此在這裡會進行 Current Task 的執行時間計數(OSTCBCyclesTot 最多只會計數到=Execution Time)，而 OSTCBCyclesCount 是用來確認當前Task執行了幾秒，是否可進行delay，並幫delay值做更新。(OSTimeTick先再OSTimeDly再OS_Sched)

```
1004 void  OSTimeTick (void)
1005 {
1006     OS_TCB     *ptcb;
1007 #if OS_TICK_STEP_EN > 0u
1008     BOOLEAN    step;
1009 #endif
1010 #if OS_CRITICAL_METHOD == 3u                          /* Allocate storage for CPU status register     */
1011     OS_CPU_SR  cpu_sr = 0u;
1012 #endif
1013
1014
1015 #if OS_TIME_TICK_HOOK_EN > 0u
1016     OSTimeTickHook();                                 /* Call user definable hook                     */
1017 #endif
1018 #if OS_TIME_GET_SET_EN > 0u
1019     OS_ENTER_CRITICAL();                              /* Update the 32-bit tick counter               */
1020     OSTime++;
1021     OS_TRACE_TICK_INCREMENT(OSTime);
1022     OS_EXIT_CRITICAL();
1023 #endif

1025     //AddedCodePA1part2
1026     OSTCBCur->OSTCBCyclesCount++;
1027     OSTCBCur->OSTCBCyclesTot++;
1028     //printf("%2d Task %2d SwitchStart%2d Total%2d\n",OSTimeGet(),OSTCBCur->OSTCBId, OSTCBCur->OSTCBCyclesSwitchStart, OSCtxSwCtr);
1029     if (OSTCBCur->OSTCBCyclesTot == OSTCBCur->OSTCBCyclesExecution) {
1030         OSTCBCur->OSTCBCyclesEnd = OSTimeGet();
1031         //Trying
1032         OSTCBCur->OSTCBCyclesEnd = OSTimeGet();
1033         OSTCBCur->OSTCBDly = (OSTCBCur->OSTCBCyclesArrive);
1034         OSTCBCur->OSTCBDly = OSTCBCur->OSTCBDly + (((OSTCBCur->OSTCBJobNumber + 1)) * (OSTCBCur->OSTCBCyclesPeriod));
1035         OSTCBCur->OSTCBDly = OSTCBCur->OSTCBDly - (OSTCBCur->OSTCBCyclesEnd);
1036         //printf("%2d\t", OSTimeGet() - OSTCBCur->OSTCBCyclesStart);
1037         //printf("%2d\t", OSTCBCur->OSTCBCtxSwCtr, OSTCBCur->OSTCBCyclesEnd - OSTCBCur->OSTCBDly);
1038         //printf("%2d\t", OSTCBCur->OSTCBCyclesEnd - OSTCBCur->OSTCBJobNumber * OSTCBCur->OSTCBCyclesExecution);
1039         //printf("%2d\t\n", OSTCBCur->OSTCBDly);
1040     }
1041     //printf("\tTot %2d Start %2d End %2d \n", OSTCBCur->OSTCBCyclesTot, OSTCBCur->OSTCBCyclesStart, OSTCBCur->OSTCBCyclesEnd);
```

為了使task不要永遠處於delay狀態，必須擁有執行時間，因此在這裡新增了一個 if 的判斷式，確認是否已完成執行時間，才去做delay使其可被其他工作搶佔。

```
void task1(void* p_arg) {
    task_para_set* task_data;
    task_data = p_arg;
    while (1)
    {
        //printf("Tick: %d, Hello from task%d\n", OSTime, task_data->TaskID);
        /*if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0)
        {
            fprintf(Output_fp, "Tick: %d, Hello from task%d\n", OSTime, task_data->TaskID);
            fclose(Output_fp);
        }*/

        //AddedCodePA1part2
        if (OSTCBCur->OSTCBCyclesCount == OSTCBCur->OSTCBCyclesExecution) {
            OSTCBCur->OSTCBCyclesCount = 0;
            //OSTimeDly(task_data->TaskPeriodic);   //Orginal
            OSTimeDly(OSTCBCur->OSTCBDly);
        }
    }
}

void task2(void* p_arg) {
    task_para_set* task_data;
    task_data = p_arg;
    while (1)
    {
        //printf("Tick: %d, Hello from task%d\n", OSTime, task_data->TaskID);
        /*if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0)
        {
            fprintf(Output_fp, "Tick: %d, Hello from task%d\n", OSTime, task_data->TaskID);
            fclose(Output_fp);
        }*/

        //AddedCodePA1part2
        if (OSTCBCur->OSTCBCyclesCount == OSTCBCur->OSTCBCyclesExecution) {
            OSTCBCur->OSTCBCyclesCount = 0;
            //OSTimeDly(task_data->TaskPeriodic);   //Orginal
            OSTimeDly(OSTCBCur->OSTCBDly);
        }
    }
}
```

因應不定數量的 Task ，在這裡新增了一個迴圈可以將 txt 的每一行讀入，並擁有正確數量的task。

```
for (n = 0; n < TASK_NUMBER; n++) {
    Task_STK[n] = malloc(TASK_STACKSIZE * sizeof(int));
    OSTaskCreateExt(task1,                      /*Create the task2*/
        &TaskParameter[n],
        &Task_STK[n][TASK_STACKSIZE - 1],
        TaskParameter[n].TaskPriority,
        TaskParameter[n].TaskID,
        &Task_STK[n][0],
        TASK_STACKSIZE,
        &TaskParameter[n],
        (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));
}

/*Create Tsak Set*/
//OSTaskCreateExt(task1,                    /*Create the task1*/
//    &TaskParameter[0],
//    &Task_STK[0][TASK_STACKSIZE - 1],
//    TaskParameter[0].TaskPriority,
//    TaskParameter[0].TaskID,
//    &Task_STK[0][0],
//    TASK_STACKSIZE,
//    &TaskParameter[0],
//    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));

/*Create Tsak Set*/
//OSTaskCreateExt(task2,                    /*Create the task2*/
//    &TaskParameter[1],
//    &Task_STK[1][TASK_STACKSIZE - 1],
//    TaskParameter[1].TaskPriority,
//    TaskParameter[1].TaskID,
//    &Task_STK[1][0],
//    TASK_STACKSIZE,
//    &TaskParameter[1],
//    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));
```

Project submit:

Submit to Moodle2.

Submit deadline: November 2nd, 2022 (Wednesday) at 12:00

File name format: RTOS_Myyyddxxx_PA1.zip

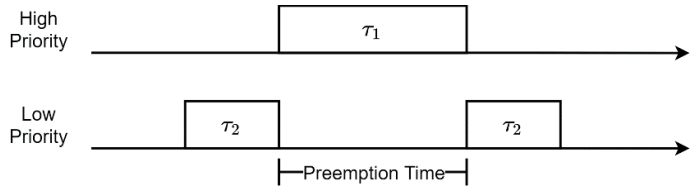
RTOS_Myyyddxxx_PA1.zip includes (The tree structure of files is shown as hints):

- The report (RTOS_Myyyddxxx_PA1.pdf).
- Folder with the executable μC/OS-II project (RTOS_Myyyddxxx_PA1).

※ Plagiarizing is strictly prohibited.

Hints:

1. Preemption time is introduced in multiple tasking.



2. RTOS_Myyyddxxx_PA1.zip include files as follows:

```
C:.
|   RTOS_Myyyddxxx_PA1.pdf
|
\---RTOS_Myyyddxxx_PA1
    |   ReadMe.txt
    |
    +---Micrium
    |   \---Software
    |       +---uC-CPU
    |       |   |   cpu_cache.h
    |       |   |   cpu_core.c
    |       |   |   cpu_core.h
    |       |   |   cpu_def.h
    |       |   |
    |       |   \---Win32
    |       |       \---Visual_Studio
    |       |               cpu.h
    |       |               cpu_c.c
    |       |
    |       +---uC-LIB
    |       |       lib_ascii.c
    |       |       lib_ascii.h
    |       |       lib_def.h
    |       |       lib_math.c
    |       |       lib_math.h
    |       |       lib_mem.c
    |       |       lib_mem.h
    |       |       lib_str.c
    |       |       lib_str.h
    |       |
    |       \---uCOS-II
    |           +---Ports
    |           |   \---Win32
    |           |       \---Visual Studio
    |           |               os_cpu.h
    |           |               os_cpu_c.c
    |           |
    |           \---Source
    |                   os.h
    |                   os_cfg_r.h
    |                   os_core.c
    |                   os_dbg_r.c
    |                   os_flag.c
    |                   os_mbox.c
    |                   os_mem.c
    |                   os_mutex.c
    |                   os_q.c
    |                   os_sem.c
    |                   os_task.c
    |                   os_time.c
    |                   os_tmr.c
    |                   os_trace.h
    |                   ucos_ii.c
    |                   ucos_ii.h
    |
```

```
\---Microsoft
    +---BSP
    |   \---Windows
    |           bsp_cpu.c
    |
    \---Windows
        \---Kernel
            |   app_cfg.h
            |   cpu_cfg.h
            |   lib_cfg.h
            |
            \---OS2
                |   app_hooks.c
                |   main.c
                |   os_cfg.h
                |
                \---VS
                        OS2.sln
                        OS2.vcxproj
                        OS2.vcxproj.filters
                        OS2.vcxproj.user
```