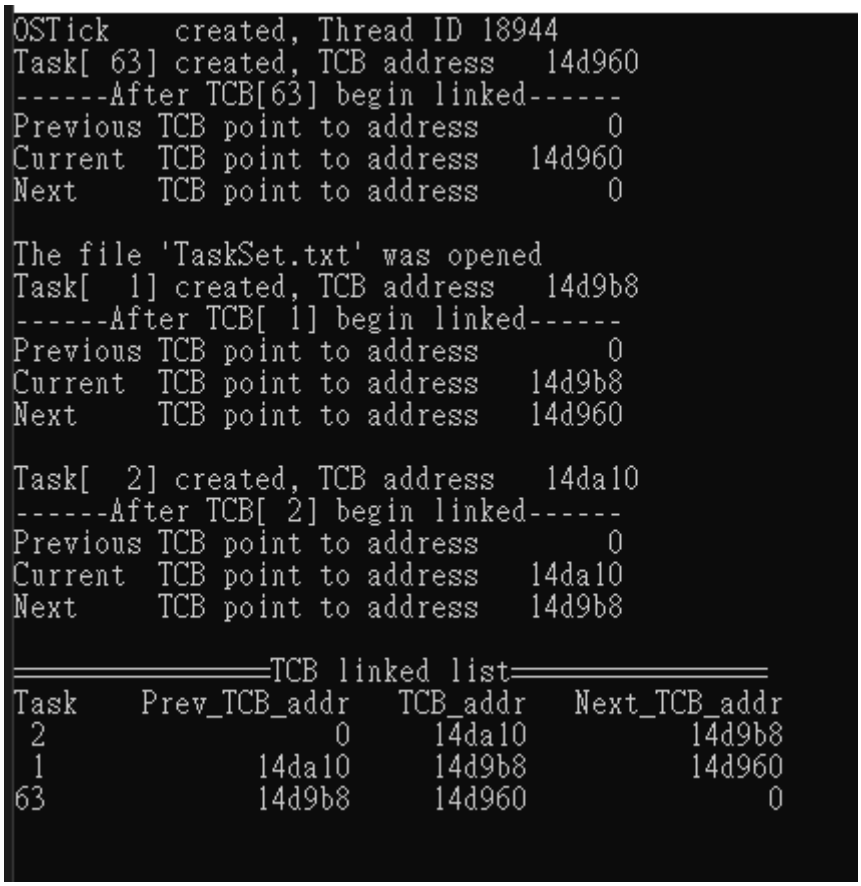
Embedded OS Implementation PA1

M11202117 呂長恩

1 PART I

1. Screenshot



2. Implementation

[os_cpu_c.c] -> OSTCBInitHook()

```
580  void  OSTCBInitHook (OS_TCB  *p_tcb)
581  {
582      OS_TASK_STK  *p_stk;
583
584
585  #if (OS_APP_HOOKS_EN > 0u) Active Preprocessor Block
587  #else Inactive Preprocessor Block
589  #endif
590
591      p_stk = (OS_TASK_STK *)p_tcb->OSTCBStkPtr;
592
593      p_stk->SignalPtr = CreateEvent(NULL, FALSE, FALSE, NULL);          /* See Note #2.
594      if (p_stk->SignalPtr == NULL) { ... }
600
601      p_stk->InitSignalPtr = CreateEvent(NULL, TRUE, FALSE, NULL);       /* See Note #2.
602      if (p_stk->InitSignalPtr == NULL) { ... }
611
612      p_stk->ThreadHandle = CreateThread(NULL, 0, OSTaskW32, p_tcb, CREATE_SUSPENDED, &p_stk->ThreadID);
613      if (p_stk->ThreadHandle == NULL) { ... }
624
625  #if (OS_MSG_TRACE > 0u)
626      /*ansel*/
627      // OS_Printf("Task[%3.1d] created, Thread ID %5.0d\n", p_tcb->OSTCBPrio, p_stk->ThreadID);
628      OS_Printf("Task[%3.1d] created, TCB address %8x\n", p_tcb->OSTCBPrio, p_tcb);
629      /*ansel*/
630  #endif
631
632      p_stk->TaskState = STATE_CREATED;
633      p_stk->OSTCBPtr  = p_tcb;
634  }
635  #endif
```

1. 將原本的顯示 Thread ID 的部分註解掉(不須顯示)。

2. 顯示 Task 的 TCB address 在每個 Task 被創立時。因為 OSTCBInitHook() 是每個 Task 被創立時需要呼叫的函式，並且在此階段 address 已被劃分給 TCB，所以可以在此時顯示出該資訊。

[os_core.c] -> OS_TCBInit()

```
2175 #if OS_TASK_REG_TBL_SIZE > 0u Active Preprocessor Block
2179 #endif
2180
2181          OSTCBInitHook(ptcb);
2182
2183          OS_ENTER_CRITICAL();
2184          OSTCBPrioTbl[prio] = ptcb;
2185          OS_EXIT_CRITICAL();
2186
2187          OSTaskCreateHook(ptcb);                          /* Call user defined hook                   */
2188
2189 #if OS_TASK_CREATE_EXT_EN > 0u Active Preprocessor Block
2196 #endif
2197
2198          OS_ENTER_CRITICAL();
2199          ptcb->OSTCBNext = OSTCBList;                     /* Link into TCB chain                      */
2200          ptcb->OSTCBPrev = (OS_TCB *)0;
2201          if (OSTCBList != (OS_TCB *)0) {
2202              OSTCBList->OSTCBPrev = ptcb;
2203          }
2204          /*ansel*/
2205          printf("------After TCB[%2d] begin linked------\n", ptcb->OSTCBPrio);
2206          printf("Previous TCB point to address %8x\n", ptcb->OSTCBPrev);
2207          printf("Current  TCB point to address %8x\n", ptcb);
2208          printf("Next     TCB point to address %8x\n\n", ptcb->OSTCBNext);
2209          /*ansel*/
2210          OSTCBList               = ptcb;                  /* Point to the newest task created         */
2211          OSRdyGrp               |= ptcb->OSTCBBitY;       /* Make task ready to run                   */
2212          OSRdyTbl[ptcb->OSTCBY] |= ptcb->OSTCBBitX;
2213          OSTaskCtr++;                                     /* Increment the #tasks counter             */
2214          OS_TRACE_TASK_READY(ptcb);
2215          OS_EXIT_CRITICAL();
2216          return (OS_ERR_NONE);
2217      }
2218      OS_EXIT_CRITICAL();
2219      return (OS_ERR_TASK_NO_MORE_TCB);
2220  }
```

1. 在此階段，每個 Task 在初始化它的 TCB 時，都會將它前一個創立的 TCB 連接在 OSTCBNext。而 OSTCBPrev 是連接下一個創立的 TCB address，所以可以看到由於下一個 TCB 還沒創立，因此設成 0。而等到下一個 TCB 被創立時，它會透過 OSTCBList 得到上一個 TCB 的 address，並且將上一個 TCB 的 OSTCBPrev 設成現在新建立的 TCB address。這些過程可以在 Link into TCB chain 看到。
2. 在框起來的部分可以看到，我們分別輸出了現在建立的 TCB 的前一個和後一個 TCB 的 address，還有自己的 address。
3. OSTCBList 是存最近建立的 TCB address，所以可以看到在最後它會被設成最新建立的 TCB 的 address。

[os_core.c] -> OSStart()

```
886  void  OSStart (void)
887  {
888      OS_TCB *iter;
889      if (OSRunning == OS_FALSE) {
890          OS_SchedNew();                                /* Find highest priority's task priority number   */
891          OSPrioCur     = OSPrioHighRdy;
892          OSTCBHighRdy  = OSTCBPrioTbl[OSPrioHighRdy]; /* Point to highest priority task ready to run    */
893          OSTCBCur      = OSTCBHighRdy;
894          /*ansel*/
895          OSTimeSet(0);                                 /*Set OS Start Time is 0*/
896          printf("================TCB linked list================\n");
897          printf("Task\tPrev_TCB_addr\tTCB_addr   Next_TCB_addr\n");
898          for (iter = OSTCBList; iter != 0; iter = iter->OSTCBNext)
899          {
900              printf("%2d\t     %8x\t%8x\t%8x\n", iter->OSTCBPrio, iter->OSTCBPrev, iter, iter->OSTCBNext);
901          }
902          /*ansel*/
903          OSStartHighRdy();                             /* Execute target specific code to start task     */
904      }
905  }
```

1. 宣告了一個 OS_TCB 的指標 iter 去指向 OSTCBList，也就是指向最新建立的 TCB address。
2. 由於這些 TCB 是用 doubly linked-list 的結構下去儲存，所以透過這個方式

可以用 OSTCBPrev 和 OSTCBNext 分別顯示出每一個 TCB 前後連接的 TCB address。並且不斷迭代下去，直到所有 TCB 被顯示出。

2 PART II (RM)

1 Implementation

[ucos_ii.h] -> os_tcb

```
606  typedef struct os_tcb {
607      OS_STK          *OSTCBStkPtr;          /* Pointer to current top of stack                          */
608
609  #if OS_TASK_CREATE_EXT_EN > 0u
610      void            *OSTCBExtPtr;          /* Pointer to user definable data for TCB extension         */
611      OS_STK          *OSTCBStkBottom;       /* Pointer to bottom of stack                               */
612      INT32U           OSTCBStkSize;         /* Size of task stack (in number of stack elements)         */
613      INT16U           OSTCBOpt;             /* Task options as passed by OSTaskCreateExt()              */
614      INT16U           OSTCBId;              /* Task ID (0..65535)                                       */
615      INT8U            OSTCBExecuTime;       /* Task execution time                                      */
616      INT8U            OSTCBExecuTimeCtr;    /* Task execution time for counting                         */
617      INT8U            OSTCBArriTime;        /* Task arrive time                                         */
618      INT8U            OSTCBPeriod;          /* The period of task                                       */
619  #endif
620
621      struct os_tcb   *OSTCBNext;            /* Pointer to next     TCB in the TCB list                  */
622      struct os_tcb   *OSTCBPrev;            /* Pointer to previous TCB in the TCB list                  */
623
624  #if OS_TASK_CREATE_EXT_EN > 0u
625  #if defined(OS_TLS_TBL_SIZE) && (OS_TLS_TBL_SIZE > 0u)
626      OS_TLS           OSTCBTLSTbl[OS_TLS_TBL_SIZE];
627  #endif
628  #endif
629
630  #if (OS_EVENT_EN)
631      OS_EVENT        *OSTCBEventPtr;         /* Pointer to          event control block                  */
632  #endif
633
634  #if (OS_EVENT_EN) && (OS_EVENT_MULTI_EN > 0u)
635      OS_EVENT       **OSTCBEventMultiPtr;    /* Pointer to multiple event control blocks                 */
636  #endif
637
638  #if ((OS_Q_EN > 0u) && (OS_MAX_QS > 0u)) || (OS_MBOX_EN > 0u)
639      void            *OSTCBMsg;             /* Message received from OSMboxPost() or OSQPost()          */
640  #endif
641
```

1. 在定義 TCB 結構的地方，新增了框起來內的變數

   1.1 OSTCBExecuTime: 存放 Task 的 computation time。

   1.2 OSTCBExecuTimeCtr: 一個計數器。計算 Task 的 computation time 還剩下多久結束。

   1.3 OSTCBArriTime: 紀錄 Task 是甚麼時候抵達的。

   1.4 OSTCBPeriod: 紀錄 Task 的週期多長。

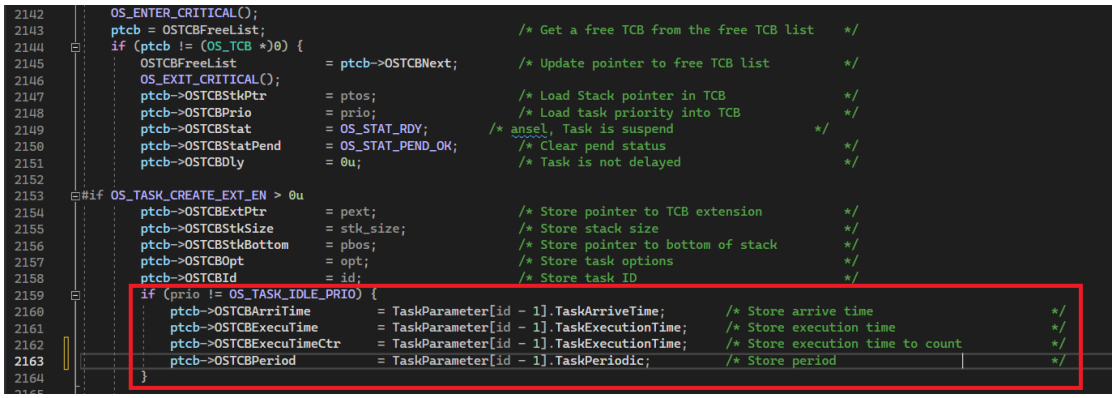
[app_hooks.c] -> InputFile()

```
105     errno_t err;
106     if ((err = fopen_s(&fp, INPUT_FILE_NAME, "r")) == 0) { ... }
110     else { ... }
114
115     char str[MAX];
116     char* ptr;
117     char* pTmp = NULL;
118     int TaskInfo[INFO], i, j = 0;
119     TASK_NUMBER = 0;
120     // int start_priority = 1;
121     while (!feof(fp))
122     {
123         i = 0;
124         memset(str, 0, sizeof(str));
125         fgets(str, sizeof(str) - 1, fp);
126         ptr = strtok_s(str, " ", &pTmp); // partition string by " "
127         while (ptr != NULL)
128         {
129             TaskInfo[i] = atoi(ptr);
130             ptr = strtok_s(NULL, " ", &pTmp);
131
132             if (i == 0) {
133                 TASK_NUMBER++;
134                 TaskParameter[j].TaskID = TaskInfo[i];
135             }
136             else if (i == 1)
137                 TaskParameter[j].TaskArriveTime = TaskInfo[i];
138             else if (i == 2)
139                 TaskParameter[j].TaskExecutionTime = TaskInfo[i];
140             else if (i == 3) {
141                 TaskParameter[j].TaskPeriodic = TaskInfo[i];
142                 TaskParameter[j].TaskPriority = TaskInfo[i];
143             }
144             i++;
145         }
```

1. 在定義參數的地方，將每一個輸入進入的值分配給對應的參數。

2. 將 Priority 設定成 Task 的週期，因為週期越短、Priority 越高，符合 RM 的規則。

[os_core.c] -> OS_TCBInit()



```
2142    OS_ENTER_CRITICAL();
2143    ptcb = OSTCBFreeList;                          /* Get a free TCB from the free TCB list    */
2144    if (ptcb != (OS_TCB *)0) {
2145        OSTCBFreeList           = ptcb->OSTCBNext;     /* Update pointer to free TCB list          */
2146        OS_EXIT_CRITICAL();
2147        ptcb->OSTCBStkPtr       = ptos;                /* Load Stack pointer in TCB                */
2148        ptcb->OSTCBPrio         = prio;                /* Load task priority into TCB              */
2149        ptcb->OSTCBStat         = OS_STAT_RDY;      /* ansel, Task is suspend                   */
2150        ptcb->OSTCBStatPend     = OS_STAT_PEND_OK;     /* Clear pend status                        */
2151        ptcb->OSTCBDly          = 0u;                  /* Task is not delayed                      */
2152
2153    #if OS_TASK_CREATE_EXT_EN > 0u
2154        ptcb->OSTCBExtPtr       = pext;                /* Store pointer to TCB extension           */
2155        ptcb->OSTCBStkSize      = stk_size;            /* Store stack size                         */
2156        ptcb->OSTCBStkBottom    = pbos;                /* Store pointer to bottom of stack         */
2157        ptcb->OSTCBOpt          = opt;                 /* Store task options                       */
2158        ptcb->OSTCBId           = id;                  /* Store task ID                            */
2159        if (prio != OS_TASK_IDLE_PRIO) {
2160            ptcb->OSTCBArriTime     = TaskParameter[id - 1].TaskArriveTime;     /* Store arrive time          */
2161            ptcb->OSTCBExecuTime    = TaskParameter[id - 1].TaskExecutionTime;  /* Store execution time       */
2162            ptcb->OSTCBExecuTimeCtr = TaskParameter[id - 1].TaskExecutionTime;  /* Store execution time to count */
2163            ptcb->OSTCBPeriod       = TaskParameter[id - 1].TaskPeriodic;       /* Store period               */
2164        }
```

1. 由於不能修改 User Space 的 code，所以在初始化 TCB 的地方，設定每個 Task 的參數。這樣一來就不需要再 user space 新增任何 code 了。

[os_core.c] -> OS_TCBInit()

```
2239            OS_ENTER_CRITICAL();
2240            ptcb->OSTCBNext = OSTCBList;                    /* Link into TCB chain                      */
2241            ptcb->OSTCBPrev = (OS_TCB *)0;
2242            if (OSTCBList != (OS_TCB *)0) {
2243                OSTCBList->OSTCBPrev = ptcb;
2244            }
2245
2246            OSTCBList               = ptcb;                 /* Point to the newest task created         */
2247            /*ansel*/
2248            if (ptcb->OSTCBPrio == OS_TASK_IDLE_PRIO) {
2249            OSRdyGrp               |= ptcb->OSTCBBitY;       /* Make task ready to run                   */
2250            OSRdyTbl[ptcb->OSTCBY]  |= ptcb->OSTCBBitX;
2251            }
2252            /*ansel*/
2253            OSTaskCtr++;                                    /* Increment the #tasks counter             */
2254            OS_TRACE_TASK_READY(ptcb);
2255            OS_EXIT_CRITICAL();
2256            return (OS_ERR_NONE);
2257        }
2258        OS_EXIT_CRITICAL();
2259        return (OS_ERR_TASK_NO_MORE_TCB);
2260    }
```

1. 由於 Task 必須等到 Arrive 後才能被 Ready。因此在初始化時，只允許 Idle task 被設定成 ready 的狀態。

[os_time.c] -> OSTimeDly()

```
55  void  OSTimeDly (INT32U ticks)
56  {
57      INT8U      y;
58  #if OS_CRITICAL_METHOD == 3u                    /* Allocate storage for CPU status register          */
59      OS_CPU_SR  cpu_sr = 0u;
60  #endif
61
62
63
64      if (OSIntNesting > 0u) {                    /* See if trying to call from an ISR                */
65          return;
66      }
67      if (OSLockNesting > 0u) {                   /* See if called with scheduler locked              */
68          return;
69      }
70      if (ticks > 0u) {                           /* 0 means no delay!                                */
71          //OS_ENTER_CRITICAL();
72          //y            =  OSTCBCur->OSTCBY;        /* Delay current task (Make its status in TCB Ready Table unready)   */
73          //OSRdyTbl[y] &= (OS_PRIO)~OSTCBCur->OSTCBBitX;
74          //OS_TRACE_TASK_SUSPENDED(OSTCBCur);
75          //if (OSRdyTbl[y] == 0u) {
76          //    OSRdyGrp &= (OS_PRIO)~OSTCBCur->OSTCBBitY;
77          //}
78          //OSTCBCur->OSTCBDly = ticks;             /* Load ticks in TCB                                 */
79          //OS_TRACE_TASK_DLY(ticks);
80          //OS_EXIT_CRITICAL();
81          //OS_Sched();                             /* Find next task to run!                            */
82          OSTaskIdleHook();                       /* Call user definable HOOK                         */
83      }
84  }
```

1. 將 OSTimeDly()的 code 註解掉，因為我們在使用 RM 時，不希望有其他 function 設定 Task 是否 ready，所以在這必須把所有關於設定 ready 的功能全註解掉。

2. 由於註解掉後，OSTimeDly()，就會像是 empty while 空轉，CPU 使用率拉高。因此加上 OSTaskIdleHook()來使 CPU 進入睡眠，降低空轉的使用率。

[os_core.c] -> OSStart()

```
void  OSStart (void)
{
    OS_TCB *ptcb;
    if (OSRunning == OS_FALSE) {
        OSTimeSet(0);                                /*Set OS Start Time is 0*/
        ptcb = OSTCBList;
        while (ptcb->OSTCBPrio != OS_TASK_IDLE_PRIO) {
            if (ptcb->OSTCBArriTime == OSTimeGet()) {
                ptcb->OSTCBArriTime = OSTimeGet();
                OSRdyGrp               |= ptcb->OSTCBBitY; /* Make ready                          */
                OSRdyTbl[ptcb->OSTCBY]  |= ptcb->OSTCBBitX;
            }
            ptcb = ptcb->OSTCBNext;
        }
        OS_SchedNew();                               /* Find highest priority's task priority number   */
        OSPrioCur     = OSPrioHighRdy;
        OSTCBHighRdy  = OSTCBPrioTbl[OSPrioHighRdy]; /* Point to highest priority task ready to run    */
        OSTCBCur      = OSTCBHighRdy;
        printf("Tick\t Event\t\t CurrentTask ID\t\tNextTaskID\t ResponseTime\t PreemptionTime\t OSTimeDly\n");
        if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0) {
            fprintf(Output_fp, "Tick\t Event\t\t CurrentTask ID\t\tNextTaskID\t ResponseTime\t PreemptionTime\t OSTimeDly\n");
            fclose(Output_fp);
        }
        OSStartHighRdy();                            /* Execute target specific code to start task     */
    }
}
```

1. 由於某些 Task 可能在第 0 秒時抵達，所以在 OSStart()時就必須檢查所有 Task 是否在此時 arrive 了。如果有，把 Task 設為 ready。

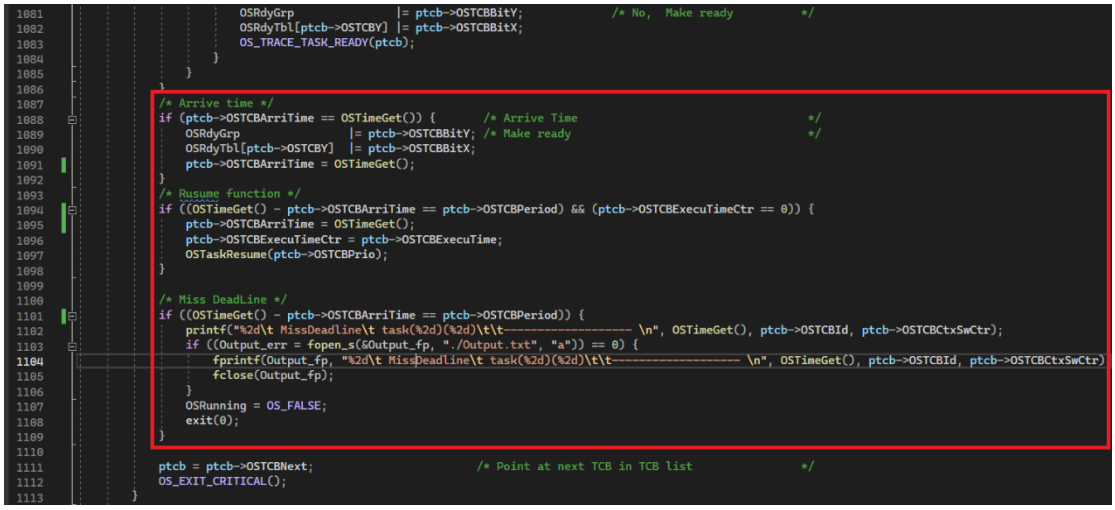
2. 顯示 Title。

[os_core.c] -> OSTimeTick()

```
1040    if (OSRunning == OS_TRUE) {
1041
1042        /*Exeuction Time*/
1043        if (--OSTCBCur->OSTCBExecuTimeCtr == 0u) {  /* Executing  */
1044            OSTaskSuspend(OS_PRIO_SELF);
1045        }
1046
1047    #if OS_TICK_STEP_EN > 0u
1048        switch (OSTickStepState) {                  /* Determine whether we need to process a tick  */
1049            case OS_TICK_STEP_DIS:                  /* Yes, stepping is disabled                    */
1050                 step = OS_TRUE;
1051                 break;
1052
1053            case OS_TICK_STEP_WAIT:                 /* No,  waiting for uC/OS-View to set ...       */
1054                 step = OS_FALSE;                   /*      .. OSTickStepState to OS_TICK_STEP_ONCE */
1055                 break;
1056
1057            case OS_TICK_STEP_ONCE:                 /* Yes, process tick once and wait for next ... */
1058                 step           = OS_TRUE;          /*      ... step command from uC/OS-View        */
1059                 OSTickStepState = OS_TICK_STEP_WAIT;
1060                 break;
1061
1062            default:                                /* Invalid case, correct situation              */
1063                 step           = OS_TRUE;
1064                 OSTickStepState = OS_TICK_STEP_DIS;
1065                 break;
1066        }
1067        if (step == OS_FALSE) {                     /* Return if waiting for step command           */
1068            return;
1069        }
1070    #endif
```

1. 當 OS 是 Running 時，把現在執行的 Task 的剩餘的 execution time 減 1。

2. 假如減到變成 0 時，代表這個 Task 的 execution 順利完成。並將這個 Task 用 OSTaskSuspend()將自己暫停，直到下一個 arrive time 抵達。



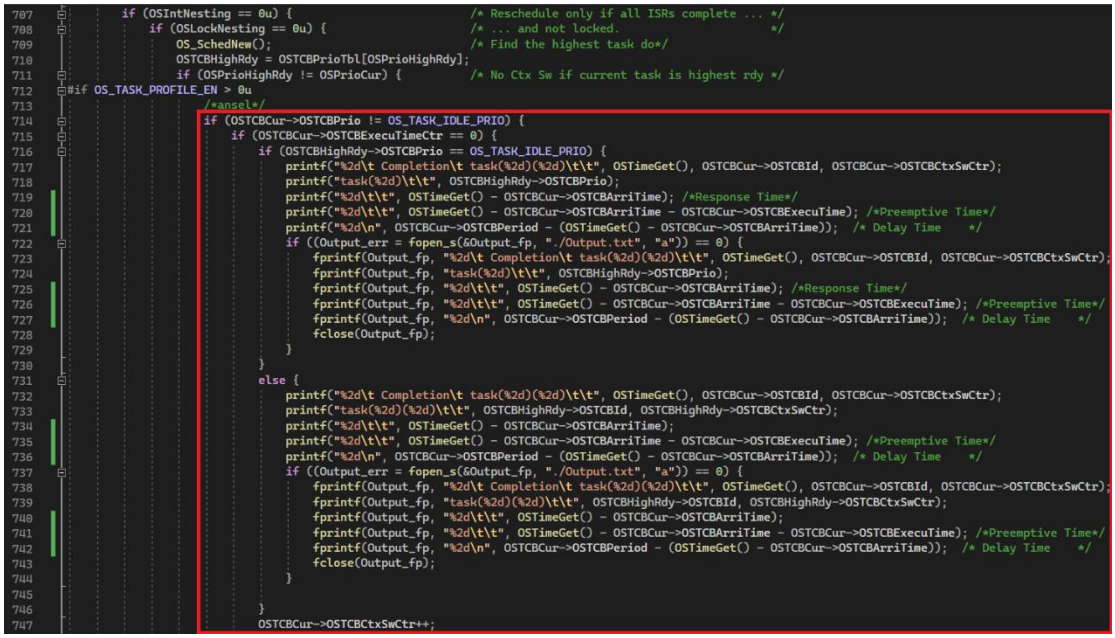
```
                    OSRdyGrp               |= ptcb->OSTCBBitY;             /* No,  Make ready          */
                    OSRdyTbl[ptcb->OSTCBY] |= ptcb->OSTCBBitX;
                    OS_TRACE_TASK_READY(ptcb);
                }
            }
        }
        /* Arrive time */
        if (ptcb->OSTCBArriTime == OSTimeGet()) {       /* Arrive Time                              */
            OSRdyGrp               |= ptcb->OSTCBBitY; /* Make ready                               */
            OSRdyTbl[ptcb->OSTCBY] |= ptcb->OSTCBBitX;
            ptcb->OSTCBArriTime = OSTimeGet();
        }
        /* Rusume function */
        if ((OSTimeGet() - ptcb->OSTCBArriTime == ptcb->OSTCBPeriod) && (ptcb->OSTCBExecuTimeCtr == 0)) {
            ptcb->OSTCBArriTime = OSTimeGet();
            ptcb->OSTCBExecuTimeCtr = ptcb->OSTCBExecuTime;
            OSTaskResume(ptcb->OSTCBPrio);
        }

        /* Miss DeadLine */
        if ((OSTimeGet() - ptcb->OSTCBArriTime == ptcb->OSTCBPeriod)) {
            printf("%2d\t MissDeadline\t task(%2d)(%2d)\t\t---------------------- \n", OSTimeGet(), ptcb->OSTCBId, ptcb->OSTCBCtxSwCtr);
            if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0) {
                fprintf(Output_fp, "%2d\t MissDeadline\t task(%2d)(%2d)\t\t---------------------- \n", OSTimeGet(), ptcb->OSTCBId, ptcb->OSTCBCtxSwCtr);
                fclose(Output_fp);
            }
            OSRunning = OS_FALSE;
            exit(0);
        }

        ptcb = ptcb->OSTCBNext;                     /* Point at next TCB in TCB list            */
        OS_EXIT_CRITICAL();
    }
```

1. Arrive time: 每當 TimeTick 被觸發時，會檢查所有非 idle task 的 TCB，檢查有沒有任何 task 已經抵達了。如果有，將其設為 ready 狀態。
2. Resume function: 每當 TimeTick 被觸發時，會檢查所有非 idle task 的 TCB。檢查在 task 抵達後是否過了一個週期，並且此時的剩餘 execution time 必須要是 0，否則代表 miss deadline。而當過了一個週期並且沒有 miss deadline 的情況下，會重新設定 arrive time、重新設定剩餘的 execution time，並且利用 OSTaskResume()將 Task 復原成 ready 的狀態，等待被執行。
3. Miss Deadline: 當在過了一個週期時，因為剩餘的 execution time 不是 0，所以沒有進入上面的 Resume function，重設 arrive time。這表示 task 已經 miss deadline 了，所以將 OSRunning 設定為 OS_FALSE，並且 exit(0)、關掉 uC/OS-2。

[os_core.c] -> OSIntExit()



```
    if (OSIntNesting == 0u) {                          /* Reschedule only if all ISRs complete ... */
        if (OSLockNesting == 0u) {                     /* ... and not locked.                      */
            OS_SchedNew();                             /* Find the highest task do*/
            OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy];
            if (OSPrioHighRdy != OSPrioCur) {          /* No Ctx Sw if current task is highest rdy */
#if OS_TASK_PROFILE_EN > 0u
                /*ansel*/
                if (OSTCBCur->OSTCBPrio != OS_TASK_IDLE_PRIO) {
                    if (OSTCBCur->OSTCBExecuTimeCtr == 0) {
                        if (OSTCBHighRdy->OSTCBPrio == OS_TASK_IDLE_PRIO) {
                            printf("%2d\t Completion\t task(%2d)(%2d)\t\t", OSTimeGet(), OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr);
                            printf("task(%2d)\t\t", OSTCBHighRdy->OSTCBPrio);
                            printf("%2d\t\t", OSTimeGet() - OSTCBCur->OSTCBArriTime); /*Response Time*/
                            printf("%2d\t\t", OSTimeGet() - OSTCBCur->OSTCBArriTime - OSTCBCur->OSTCBExecuTime); /*Preemptive Time*/
                            printf("%2d\n", OSTCBCur->OSTCBPeriod - (OSTimeGet() - OSTCBCur->OSTCBArriTime));  /* Delay Time    */
                            if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0) {
                                fprintf(Output_fp, "%2d\t Completion\t task(%2d)(%2d)\t\t", OSTimeGet(), OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr);
                                fprintf(Output_fp, "task(%2d)\t\t", OSTCBHighRdy->OSTCBPrio);
                                fprintf(Output_fp, "%2d\t\t", OSTimeGet() - OSTCBCur->OSTCBArriTime); /*Response Time*/
                                fprintf(Output_fp, "%2d\t\t", OSTimeGet() - OSTCBCur->OSTCBArriTime - OSTCBCur->OSTCBExecuTime); /*Preemptive Time*/
                                fprintf(Output_fp, "%2d\n", OSTCBCur->OSTCBPeriod - (OSTimeGet() - OSTCBCur->OSTCBArriTime));  /* Delay Time    */
                                fclose(Output_fp);
                            }
                        }
                        else {
                            printf("%2d\t Completion\t task(%2d)(%2d)\t\t", OSTimeGet(), OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr);
                            printf("task(%2d)(%2d)\t\t", OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBCtxSwCtr);
                            printf("%2d\t\t", OSTimeGet() - OSTCBCur->OSTCBArriTime);
                            printf("%2d\t\t", OSTimeGet() - OSTCBCur->OSTCBArriTime - OSTCBCur->OSTCBExecuTime); /*Preemptive Time*/
                            printf("%2d\n", OSTCBCur->OSTCBPeriod - (OSTimeGet() - OSTCBCur->OSTCBArriTime));  /* Delay Time    */
                            if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0) {
                                fprintf(Output_fp, "%2d\t Completion\t task(%2d)(%2d)\t\t", OSTimeGet(), OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr);
                                fprintf(Output_fp, "task(%2d)(%2d)\t\t", OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBCtxSwCtr);
                                fprintf(Output_fp, "%2d\t\t", OSTimeGet() - OSTCBCur->OSTCBArriTime);
                                fprintf(Output_fp, "%2d\t\t", OSTimeGet() - OSTCBCur->OSTCBArriTime - OSTCBCur->OSTCBExecuTime); /*Preemptive Time*/
                                fprintf(Output_fp, "%2d\n", OSTCBCur->OSTCBPeriod - (OSTimeGet() - OSTCBCur->OSTCBArriTime));  /* Delay Time    */
                                fclose(Output_fp);
                            }

                        }
                        OSTCBCur->OSTCBCtxSwCtr++;
```

1. 當 Current Task 不是 idle task 時進入這個 block。
2. 判斷現在 Task 剩餘的時間是不是 0，假如是 0 的話，代表該 Task 被完成了，所以可以輸出 Completion 的資訊。
3. 當 Task 是 Completion 時，根據它下一個要執行的 Task 有不同樣的輸出。假如下一個要做的是 Idle task 則用上面的規格輸出；假如不是，就用下面的。最下面的 OSTCBCtxSwCtr++會去記錄現在是完成哪個 job 了。
4. 各種時間的計算方法：
   - 4.1 Response Time: 完成的時間 – 抵達的時間。
   - 4.2 Preemptive Time: 完成的時間 – 抵達的時間 – 執行的時間。也就是在這個 Response Time 中等待的時間。
   - 4.3 Delay Time: 週期 –(完成的時間 – 抵達的時間)。這代表在 delay 這段時間就可以進入下一個 job 的週期。

1. 上面被框起來的部分是當 Task 還沒做完時，被別的高優先權的 Task 搶斷，這時就會輸出這個資訊。

2. 下面框起來的部分是當 Idle Task 被搶斷時，會依照規定顯示。

3 PART III (FIFO)

1 Implementation (相較於 RM 不同的部分)

概念：利用一個 queue 去存放 FIFO 的順序，越早抵達的會在 queue 的頭部，越早被執行到。而後加入的 task 會放在 queue 的尾巴，越晚被執行到。

[app_hooks.c] -> InputFile()

```
void InputFile() {
    /*
    * Read File
    * Task Information
    * Task_ID ArriveTime ExecutionTime Periodic
    */
    errno_t err;
    if ((err = fopen_s(&fp, INPUT_FILE_NAME, "r")) == 0)        /*task set 1-4*/
    {
        printf("The file 'TaskSet.txt' was opened\n");
    }
    else
    {
        printf("The file 'TaskSet.txt' was not opened\n");
    }

    char str[MAX];
    char* ptr;
    char* pTmp = NULL;
    int TaskInfo[INFO], i, j = 0;
    TASK_NUMBER = 0;
    int start_priority = 1;
    while (!feof(fp))
    {
        i = 0;
        memset(str, 0, sizeof(str));
        fgets(str, sizeof(str) - 1, fp);
        ptr = strtok_s(str, " ", &pTmp); // partition string by " "
        while (ptr != NULL) { ... }
        /*Initial Priority*/
        TaskParameter[j].TaskPriority = start_priority;
        start_priority++;
        j++;
        // start_priority++;
    }
    fclose(fp);
}
```

1. 將每個 Task 的 Priority 設定都不同，因為在 FIFO 不會使用到該參數，所以只要設定不同即可。

[ucos_ii.h] -> typedef struct os_task_node{}

```
78   /*  Output file */
79   FILE* Output_fp;
80   errno_t Output_err;
81   /*  Output file */
82
83   /*  Task Structure  */
84   typedef struct task_para_set {
85       INT16U TaskID;
86       INT16U TaskArriveTime;
87       INT16U TaskExecutionTime;
88       INT16U TaskPeriodic;
89       INT16U TaskNumber;
90       INT16U TaskPriority;
91   } task_para_set;
92
93   int TASK_NUMBER;    // number of the input tasks
94   /*  Task Structure  */
95
96   /*Dynamic Create the Stack Size*/
97   OS_STK** Task_STK;
98
99   /*Create Task*/
100  task_para_set TaskParameter[OS_MAX_TASKS];
101
102  /* Create Task Node Structure */
103  typedef struct os_task_node {
104      struct os_tcb          *tcb;
105      struct os_task_node    *next;
106      struct os_task_node    *prev;
107
108  } OS_TASK_NODE;
109
110  #ifdef   OS_GLOBALS
111  #define  OS_EXT
112  #else
113  #define  OS_EXT  extern
114  #endif
115
```

1. 新增一個 OS_TASK_NODE 的結構，定義 FIFO 這個 queue 中的每一個 Node。

Tcb: 指向這個 node 對應的 TCB。

Next: 指向在 queue 中，後面的 Node(比他晚抵達的 task)。

Prev: 指向在 queue 中，前面的 Node(比它早抵達的 task)。

```
744  *********************************************************************************************************
745  *                                          GLOBAL VARIABLES
746  *********************************************************************************************************
747  */
748
749  OS_EXT  INT32U           OSCtxSwCtr;              /* Counter of number of context switches          */
750  OS_EXT  OS_TASK_NODE     *OSTaskNodeHead;         /* Pointer to head of linked list of Task Node Queue */
751  OS_EXT  OS_TASK_NODE     *OSTaskNodeRear;         /* Pointer to rear of linked list of Task Node Queue */
752
753  #if (OS_EVENT_EN) && (OS_MAX_EVENTS > 0u)
754  OS_EXT  OS_EVENT         *OSEventFreeList;        /* Pointer to list of free EVENT control blocks    */
755  OS_EXT  OS_EVENT          OSEventTbl[OS_MAX_EVENTS];/* Table of EVENT control blocks                  */
756  #endif
```

1. 新增兩個全域變數，分別指向 queue 的頭和尾的 Node。

[os_core.c] -> OS_SchedNew()

```
1923    static  void  OS_SchedNew (void)                /* Find the highest priority task */
1924    {
1925    #if OS_LOWEST_PRIO <= 63u                       /* See if we support up to 64 tasks                   */
1926        INT8U   y;
1927
1928
1929        /*y             = OSUnMapTbl[OSRdyGrp];
1930        OSPrioHighRdy = (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]);*/
1931        OSPrioHighRdy = OSTaskNodeHead->tcb->OSTCBPrio;
1932    #else                                           /* We support up to 256 tasks                         */
1933        INT8U     y;
1934        OS_PRIO  *ptbl;
1935
1936
1937        if ((OSRdyGrp & 0xFFu) != 0u) {
1938            y = OSUnMapTbl[OSRdyGrp & 0xFFu];
1939        } else {
1940            y = OSUnMapTbl[(OS_PRIO)(OSRdyGrp >> 8u) & 0xFFu] + 8u;
1941        }
1942        ptbl = &OSRdyTbl[y];
1943        if ((*ptbl & 0xFFu) != 0u) {
1944            OSPrioHighRdy = (INT8U)((y << 4u) + OSUnMapTbl[(*ptbl & 0xFFu)]);
1945        } else {
1946            OSPrioHighRdy = (INT8U)((y << 4u) + OSUnMapTbl[(OS_PRIO)(*ptbl >> 8u) & 0xFFu] + 8u);
1947        }
1948    #endif
1949    }
1950
```

1. OSPrioHighRdy 設定為 Queue 中的頭的 Node 的 Task 的 priority。

[os_core.c] -> OSStart ()

```
void  OSStart (void)
{
    OS_TCB *ptcb;
    OS_TASK_NODE* queue;
    queue = (OS_TASK_NODE *)0;

    if (OSRunning == OS_FALSE) {
        OSTimeSet(0);
        ptcb = OSTCBList;
        while (ptcb->OSTCBPrio != OS_TASK_IDLE_PRIO) {
            if (ptcb->OSTCBArriTime == OSTimeGet()) {
                queue = OSPushTaskNode(queue, ptcb);
            }
            ptcb = ptcb->OSTCBNext;
        }
        // printf("Time 0 %d\n", queue->tcb->OSTCBId);
        if (queue != 0) {
            OS_TASK_NODE* iter = queue;
            while (iter->next != 0) {
                iter = iter->next;
            }
            OSTaskNodeRear->prev = iter;
            iter->next = OSTaskNodeHead;
            OSTaskNodeHead = queue;
        }

        OS_SchedNew();                                  /* Find highest priority's task priority number   */
        OSPrioCur     = OSPrioHighRdy;
        OSTCBHighRdy  = OSTCBPrioTbl[OSPrioHighRdy];    /* Point to highest priority task ready to run    */
        OSTCBCur      = OSTCBHighRdy;

        printf("Tick\t Event\t\t CurrentTask ID\t\tNextTaskID\t ResponseTime\t PreemptionTime\t OSTimeDly\n");
        if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0) {
            fprintf(Output_fp, "Tick\t Event\t\t CurrentTask ID\t\tNextTaskID\t ResponseTime\t PreemptionTime\t OSTimeDly\n");
            fclose(Output_fp);
        }
        OSStartHighRdy();                               /* Execute target specific code to start task     */
    }
}
```

1. 如果 Task 在 time=0 時抵達，加入 queue 中。

2. 更新 OSTaskNodeRear 和 OSTaskNodeRear 的資料。

[os_core.c] -> OSPushTaskNode()

```
OS_TASK_NODE*  OSPushTaskNode (OS_TASK_NODE* q, OS_TCB* ptcb)
{
    OS_TASK_NODE *iter = q;
    OS_TASK_NODE *new_node = malloc(sizeof(OS_TASK_NODE));
    new_node->tcb = ptcb;
    new_node->next = (OS_TASK_NODE*)0;
    new_node->prev = (OS_TASK_NODE*)0;

    if (q == 0) { // if q is nullptr
        return new_node;
    }

    while (iter->tcb->OSTCBId < new_node->tcb->OSTCBId) {
        iter = iter->next;
    }

    if (iter == q) { // push in head
        new_node->next = iter;
        iter->prev = new_node;
        return new_node;
    }
    else {
        new_node->prev = iter->prev;
        iter->prev->next = new_node;
        new_node->next = iter;
        return q;
    }
}
```

1. 將 TCB 存入 Node 中，並且將 Node 插入 queue 中。

2. 根據 ID 的大小決定假如有相同抵達時間的 Task，誰要放前、誰要放後。

[os_core.c] -> OSTimeTick()

```
1083    if (OSRunning == OS_TRUE) {
1084        /*Exeuction Time*/
1085        if (--OSTCBCur->OSTCBExecuTimeCtr == 0u) {  /* Executing  */
1086            OSTaskNodeHead = OSTaskNodeHead->next;
1087        }
1088
```

1. 當 Task 執行完，換到 queue 中的下一個 node 的 Task。

```
1135            /* Arrive time */
1136            if (ptcb->OSTCBArriTime == OSTimeGet()) {       /* Arrive Time                          */
1137                queue = OSPushTaskNode(queue, ptcb);
1138            }
1139            /* Rusume function */
1140            if ((OSTimeGet() - ptcb->OSTCBArriTime == ptcb->OSTCBPeriod) && (ptcb->OSTCBExecuTimeCtr == 0)) {
1141                ptcb->OSTCBArriTime = OSTimeGet();
1142                ptcb->OSTCBExecuTimeCtr = ptcb->OSTCBExecuTime;
1143                queue = OSPushTaskNode(queue, ptcb);
1144            }
1145
1146            /* Miss DeadLine */
1147            if ((OSTimeGet() - ptcb->OSTCBArriTime == ptcb->OSTCBPeriod) && (ptcb->OSTCBArriTime < OSTimeGet())) {
1148                printf("%2d\t MissDeadline\t task(%2d)(%2d)\t\t------------------ \n", OSTimeGet(), ptcb->OSTCBId, ptcb->OSTCBCtxSwCtr);
1149                if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0) {
1150                    fprintf(Output_fp, "%2d\t MissDeadline\t task(%2d)(%2d)\t\t------------------ \n", OSTimeGet(), ptcb->OSTCBId, ptcb->OSTCBCtxSwCtr);
1151                    fclose(Output_fp);
1152                }
1153                OSRunning = OS_FALSE;
1154                exit(0);
1155            }
```

2. 當 Task 抵達後，放入 queue 中。

3. 當 Task 抵達下一個週期，放入 queue 中。

4. 當 missdeadline，顯示、並結束 ucos-2。

Tick	Event	CurrentTask ID	NextTaskID	ResponseTime	PreemptionTime	OSTimeDly
1	Completion	task(1)(0)	task(2)(0)	1	0	4
4	Completion	task(2)(0)	task(63)	4	1	4
5	Preemption	task(63)	task(1)(1)			
6	Completion	task(1)(1)	task(63)	1	0	4
8	Preemption	task(63)	task(2)(1)			
11	Completion	task(2)(1)	task(1)(2)	3	0	5
12	Completion	task(1)(2)	task(63)	2	1	3
15	Preemption	task(63)	task(1)(3)			
16	Completion	task(1)(3)	task(2)(2)	1	0	4
19	Completion	task(2)(2)	task(63)	3	0	5
20	Preemption	task(63)	task(1)(4)			
21	Completion	task(1)(4)	task(63)	1	0	4
24	Preemption	task(63)	task(2)(3)			
27	Completion	task(2)(3)	task(1)(5)	3	0	5
28	Completion	task(1)(5)	task(63)	3	2	2
30	Preemption	task(63)	task(1)(6)			

1. 可以看到由於 FIFO 會等到目前的 task 執行完後，才換下一個最早抵達的 task，所以不會有除了 idle task 外的任何 task 被中斷。