Embedded OS Implementation, Fall 2022
Project #2 (due Nov 23, 2022 (Wednesday) 12:00)

[PART I] EDF Scheduler Implementation

Objective:

To implement the Earliest-Deadline-First (EDF) scheduler for periodic tasks and to observe the scheduling behaviors.

Problem Definition:

uC/OS-II supports priority-driven scheduling. However, it lacks deadline-driven scheduling. In this assignment, you are going to implement the EDF scheduler in uC/OS-II. To accomplish this assignment, you must know about the scheduler of uC/OS-II. It can be implemented based on the existing data structures of uC/OS-II. The objectives of this assignment are the following:

(1) To add some functional data structures for your EDF scheduler.

(2) To cooperate with existing data structures/mechanisms in uC/OS-II.

Implement the following examples. Add necessary code to the μC/OS-II scheduler **in the kernel level** to observe how the task suffers the schedule delay.

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

Example Task Set 1 = {τ_1 (1, 0, 4, 11), τ_2 (2, 0, 3, 9)}

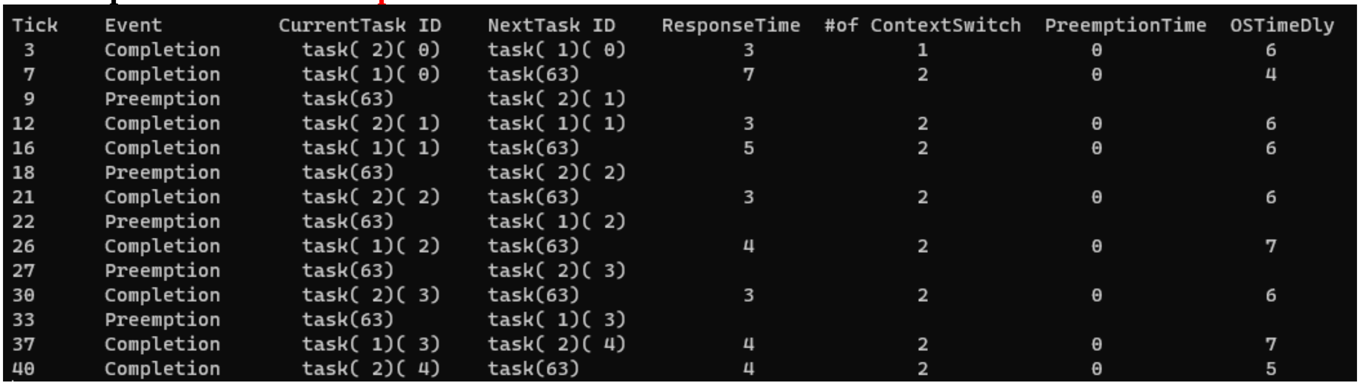
Example Task Set 2 = {τ_1 (1, 0, 2, 6), τ_2 (2, 0, 3, 8)}

Example Task Set 3 = {τ_1 (1, 0, 2, 5), τ_2 (2, 0, 4, 8), τ_3 (3, 1, 2, 6)}

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

※ If there are tasks with the same deadlines, the task with a lower task ID will be executed first.

The output results of Example 1:

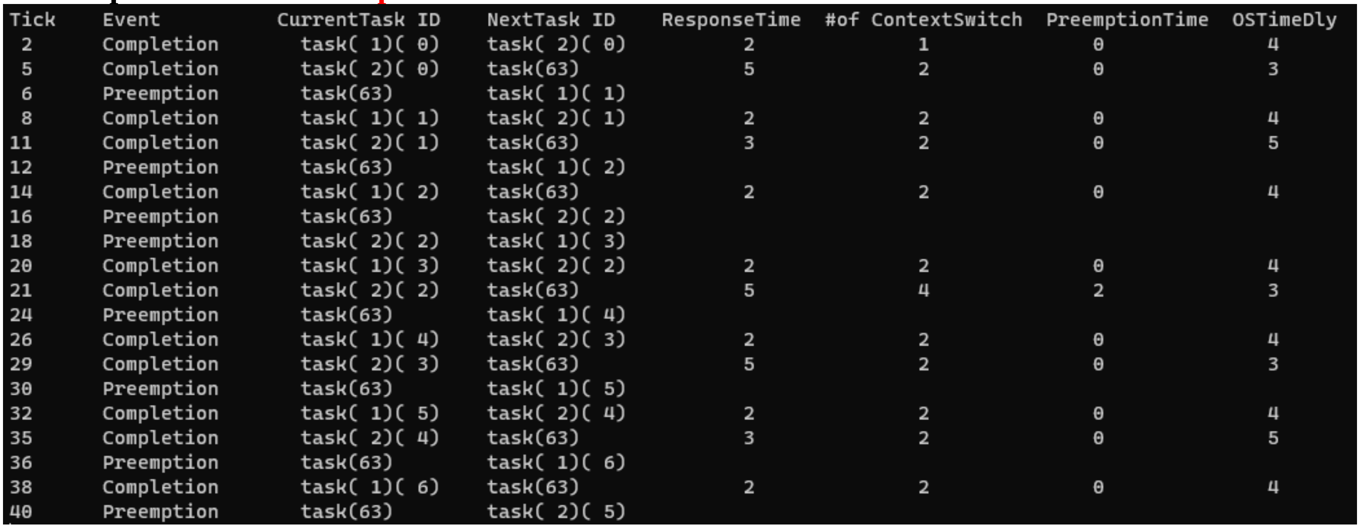


```
Tick    Event         CurrentTask ID   NextTask ID     ResponseTime  #of ContextSwitch  PreemptionTime  OSTimeDly
 3      Completion      task( 2)( 0)   task( 1)( 0)        3               1                0              6
 7      Completion      task( 1)( 0)   task(63)            7               2                0              4
 9      Preemption      task(63)       task( 2)( 1)
12      Completion      task( 2)( 1)   task( 1)( 1)        3               2                0              6
16      Completion      task( 1)( 1)   task(63)            5               2                0              6
18      Preemption      task(63)       task( 2)( 2)
21      Completion      task( 2)( 2)   task(63)            3               2                0              6
22      Preemption      task(63)       task( 1)( 2)
26      Completion      task( 1)( 2)   task(63)            4               2                0              7
27      Preemption      task(63)       task( 2)( 3)
30      Completion      task( 2)( 3)   task(63)            3               2                0              6
33      Preemption      task(63)       task( 1)( 3)
37      Completion      task( 1)( 3)   task( 2)( 4)        4               2                0              7
40      Completion      task( 2)( 4)   task(63)            4               2                0              5
```

Txt Output：

```
 3   Completion   task( 2)( 0)   task( 1)( 0)      3        1        0        6
 7   Completion   task( 1)( 0)   task(63)          7        2        0        4
 9   Preemption   task(63)       task( 2)( 1)
12   Completion   task( 2)( 1)   task( 1)( 1)      3        2        0        6
16   Completion   task( 1)( 1)   task(63)          5        2        0        6
18   Preemption   task(63)       task( 2)( 2)
21   Completion   task( 2)( 2)   task(63)          3        2        0        6
22   Preemption   task(63)       task( 1)( 2)
26   Completion   task( 1)( 2)   task(63)          4        2        0        7
27   Preemption   task(63)       task( 2)( 3)
30   Completion   task( 2)( 3)   task(63)          3        2        0        6
33   Preemption   task(63)       task( 1)( 3)
37   Completion   task( 1)( 3)   task( 2)( 4)      4        2        0        7
40   Completion   task( 2)( 4)   task(63)          4        2        0        5
```

The output results of Example 2:

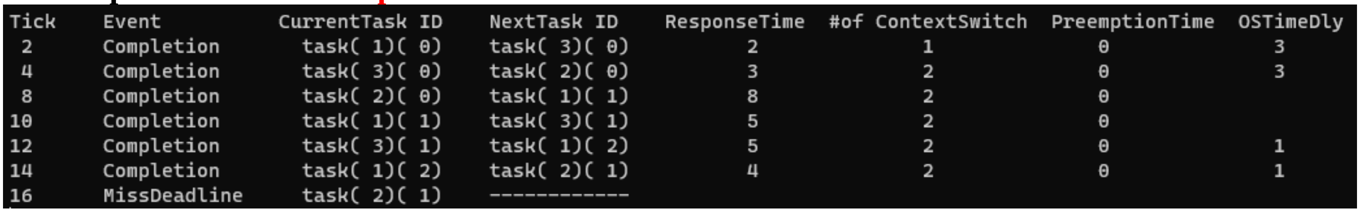


```
Tick    Event         CurrentTask ID    NextTask ID    ResponseTime  #of ContextSwitch  PreemptionTime  OSTimeDly
 2      Completion      task( 1)( 0)    task( 2)( 0)        2              1                 0              4
 5      Completion      task( 2)( 0)    task(63)            5              2                 0              3
 6      Preemption      task(63)        task( 1)( 1)
 8      Completion      task( 1)( 1)    task( 2)( 1)        2              2                 0              4
11      Completion      task( 2)( 1)    task(63)            3              2                 0              5
12      Preemption      task(63)        task( 1)( 2)
14      Completion      task( 1)( 2)    task(63)            2              2                 0              4
16      Preemption      task(63)        task( 2)( 2)
18      Preemption      task( 2)( 2)    task( 1)( 3)
20      Completion      task( 1)( 3)    task( 2)( 2)        2              2                 0              4
21      Completion      task( 2)( 2)    task(63)            5              4                 2              3
24      Preemption      task(63)        task( 1)( 4)
26      Completion      task( 1)( 4)    task( 2)( 3)        2              2                 0              4
29      Completion      task( 2)( 3)    task(63)            5              2                 0              3
30      Preemption      task(63)        task( 1)( 5)
32      Completion      task( 1)( 5)    task( 2)( 4)        2              2                 0              4
35      Completion      task( 2)( 4)    task(63)            3              2                 0              5
36      Preemption      task(63)        task( 1)( 6)
38      Completion      task( 1)( 6)    task(63)            2              2                 0              4
40      Preemption      task(63)        task( 2)( 5)
```

Txt Output：

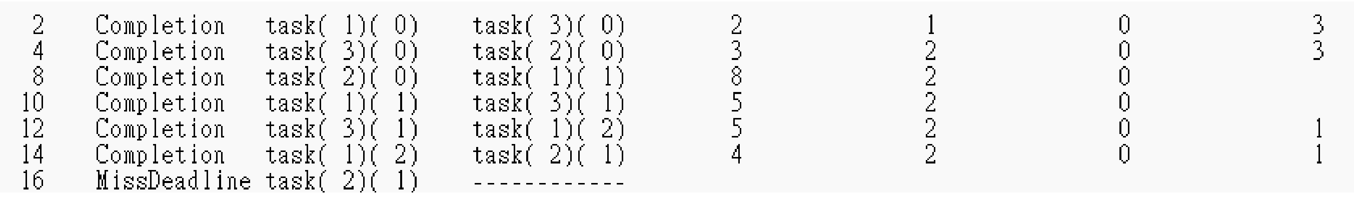
```
 2   Completion   task( 1)( 0)   task( 2)( 0)      2        1        0        4
 5   Completion   task( 2)( 0)   task(63)          5        2        0        3
 6   Preemption   task(63)       task( 1)( 1)
 8   Completion   task( 1)( 1)   task( 2)( 1)      2        2        0        4
11   Completion   task( 2)( 1)   task(63)          3        2        0        5
12   Preemption   task(63)       task( 1)( 2)
14   Completion   task( 1)( 2)   task(63)          2        2        0        4
16   Preemption   task(63)       task( 2)( 2)
18   Preemption   task( 2)( 2)   task( 1)( 3)
20   Completion   task( 1)( 3)   task( 2)( 2)      2        2        0        4
21   Completion   task( 2)( 2)   task(63)          5        4        2        3
24   Preemption   task(63)       task( 1)( 4)
26   Completion   task( 1)( 4)   task( 2)( 3)      2        2        0        4
29   Completion   task( 2)( 3)   task(63)          5        2        0        3
30   Preemption   task(63)       task( 1)( 5)
32   Completion   task( 1)( 5)   task( 2)( 4)      2        2        0        4
35   Completion   task( 2)( 4)   task(63)          3        2        0        5
36   Preemption   task(63)       task( 1)( 6)
38   Completion   task( 1)( 6)   task(63)          2        2        0        4
40   Preemption   task(63)       task( 2)( 5)
```

The output results of Example 3:



```
Tick    Event             CurrentTask ID     NextTask ID     ResponseTime   #of ContextSwitch   PreemptionTime   OSTimeDly
 2      Completion          task( 1)( 0)     task( 3)( 0)        2                1                  0               3
 4      Completion          task( 3)( 0)     task( 2)( 0)        3                2                  0               3
 8      Completion          task( 2)( 0)     task( 1)( 1)        8                2                  0
10      Completion          task( 1)( 1)     task( 3)( 1)        5                2                  0
12      Completion          task( 3)( 1)     task( 1)( 2)        5                2                  0               1
14      Completion          task( 1)( 2)     task( 2)( 1)        4                2                  0               1
16      MissDeadline        task( 2)( 1)     ------------
```

Txt Output：



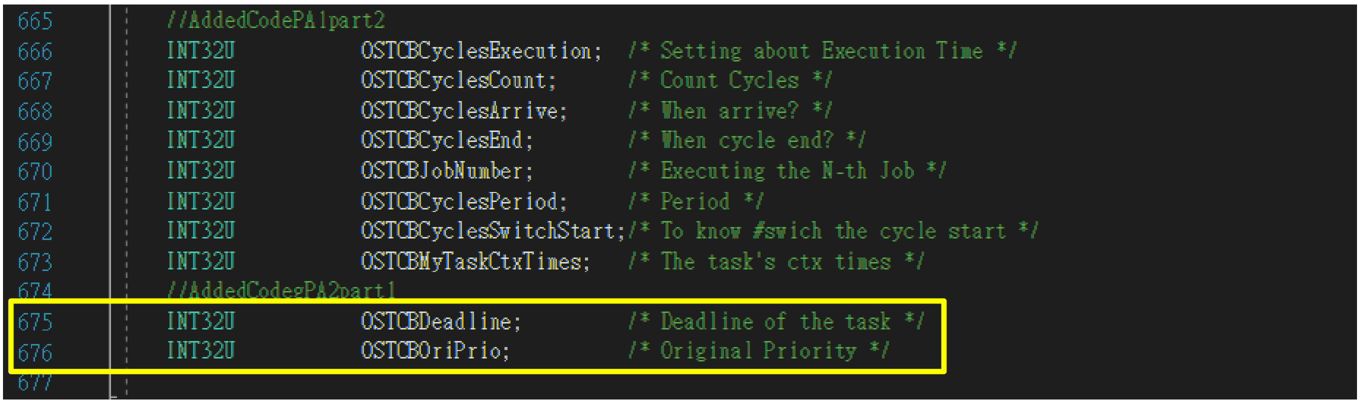
```
 2    Completion   task( 1)( 0)   task( 3)( 0)      2          1          0          3
 4    Completion   task( 3)( 0)   task( 2)( 0)      3          2          0          3
 8    Completion   task( 2)( 0)   task( 1)( 1)      8          2          0
10    Completion   task( 1)( 1)   task( 3)( 1)      5          2          0
12    Completion   task( 3)( 1)   task( 1)( 2)      5          2          0          1
14    Completion   task( 1)( 2)   task( 2)( 1)      4          2          0          1
16    MissDeadline task( 2)( 1)   ------------
```

Implement and describe how to handle the missing deadline situation under EDF：

實施並描述如何處理EDF下的錯過截止日期的情況，在此因為每一個 Tick 都會進入 OSIntExit 執行，等於每次都會進入自己撰寫的 OS_EDF_Int ，在此呼叫副程式 OS_Check_MissDeadline 去檢查是否有Miss Deadline的情況發生，而判斷依據是現在的時間若等於 Deadline，會去檢查這個 Task 從開始執行到現在過了多久，若小於它應有的 Execution Time 就代表該 Task 會來不及執行完，也就是發生 Miss Deadline，在此時會讓程式 print 出作業所要求的格式，並採用 exit(0) 讓程式立刻中止執行。（程式）會在下部分Experience Report 作呈現

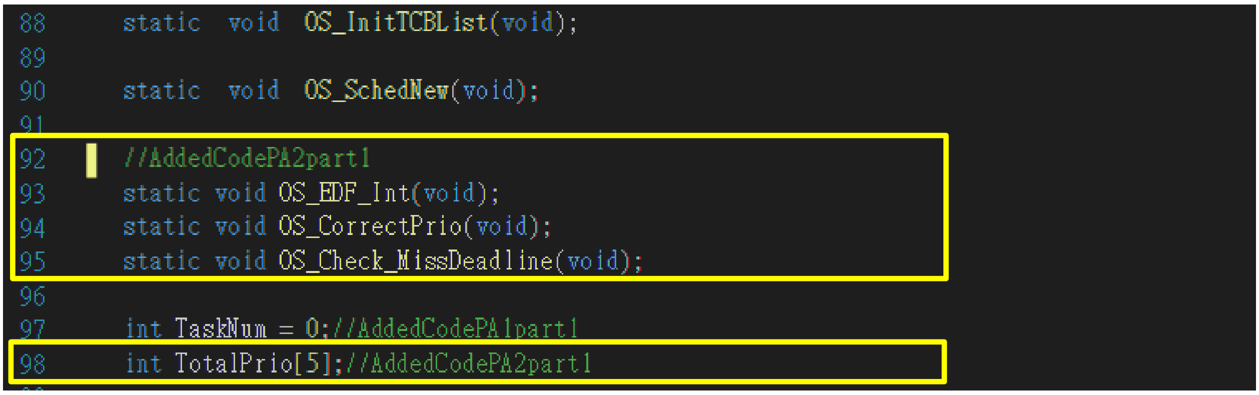
Experiment Report：

由於 EDF 是由最近的 Deadline 去決定當下該執行哪個 Task，因此會主要會需要增加幾個部分：1. Task priority 變更、2. 隨時紀錄該 Task 的Deadline，首先會於 ucos_ii.h 新增變數 OSTCBDeadline、OSTCBOriPrio 分別去紀錄當下該任務的 Deadline 以及最初該 Task 的優先權為多少，方便在日後修改回原本的優先權，此外，預設最多有5個任務(含idle)，也可以再更改 array大小去擴增：



```c
    //AddedCodePA1part2
    INT32U          OSTCBCyclesExecution;  /* Setting about Execution Time */
    INT32U          OSTCBCyclesCount;      /* Count Cycles */
    INT32U          OSTCBCyclesArrive;     /* When arrive? */
    INT32U          OSTCBCyclesEnd;        /* When cycle end? */
    INT32U          OSTCBJobNumber;        /* Executing the N-th Job */
    INT32U          OSTCBCyclesPeriod;     /* Period */
    INT32U          OSTCBCyclesSwitchStart;/* To know #swich the cycle start */
    INT32U          OSTCBMyTaskCtxTimes;   /* The task's ctx times */
    //AddedCodesPA2part1
    INT32U          OSTCBDeadline;         /* Deadline of the task */
    INT32U          OSTCBOriPrio;          /* Original Priority */
```

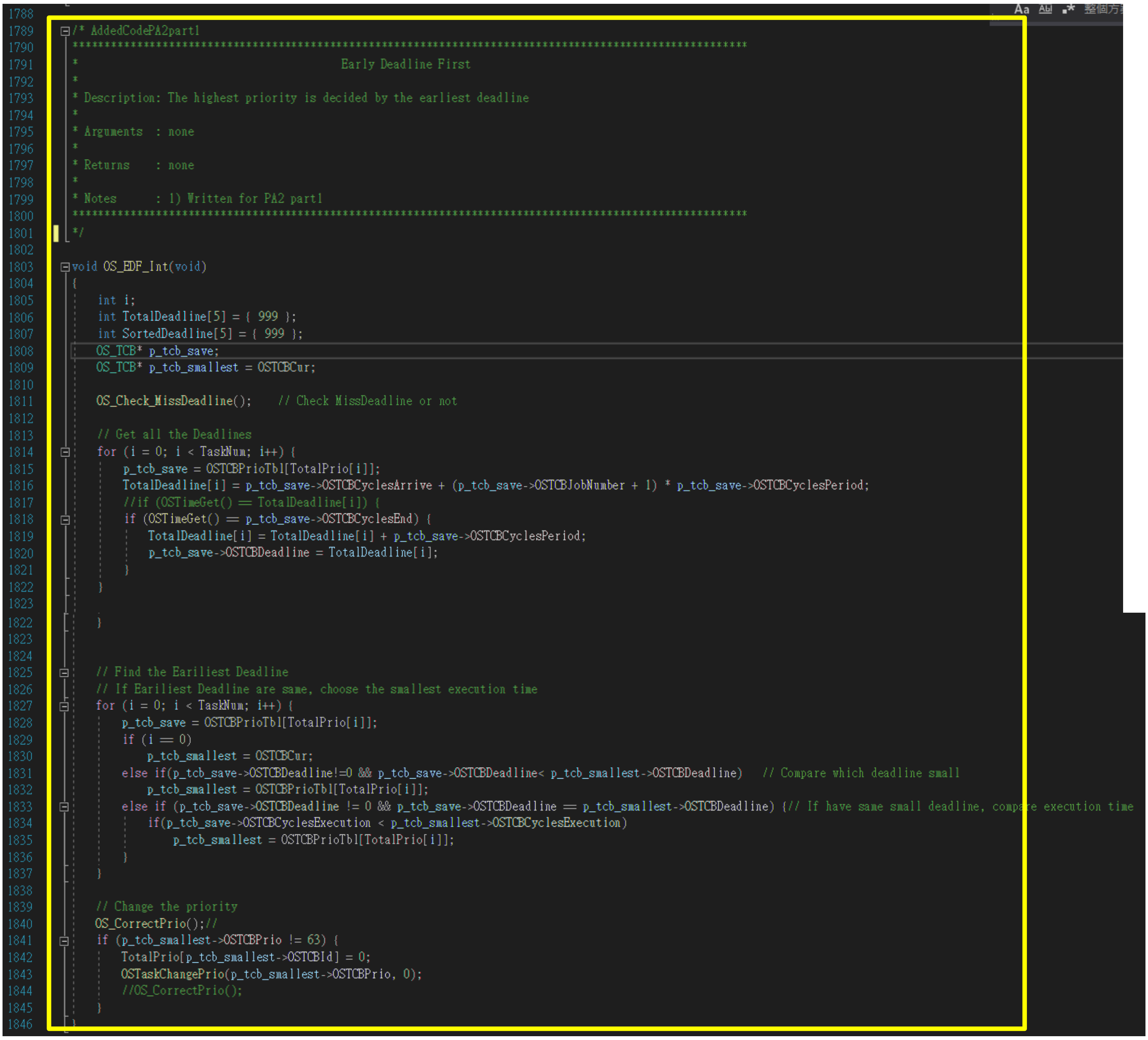
其餘剩下的程式則都在 os_core.c 檔中編寫即可，這次的想法主要為發現 Deadline 最早的 Task 時，將它的 priority 變更為 0，也就是最高優先權去執行，而原本程式內的優先權設定是”優先權=週期” ，為了方便抓取現在所有任務，因此額外多宣告了一個全域變數 TotalPrio 去記錄當下有使用那些優先權數值：



```
static  void  OS_InitTCBList(void);

static  void  OS_SchedNew(void);

//AddedCodePA2part1
static void OS_EDF_Int(void);
static void OS_CorrectPrio(void);
static void OS_Check_MissDeadline(void);

int TaskNum = 0;//AddedCodePA1part1
int TotalPrio[5];//AddedCodePA2part1
```

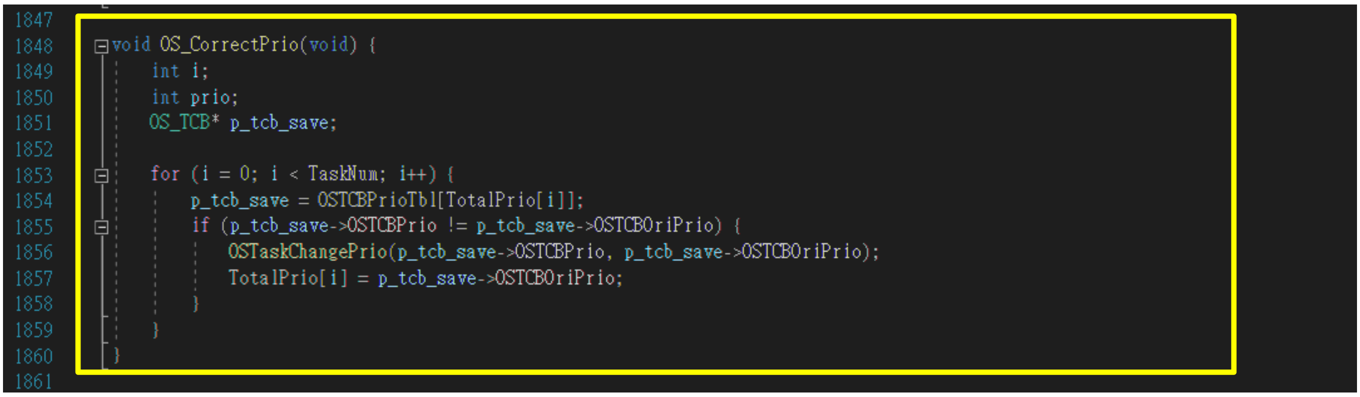
副程式 OS_EDF_Int 最主要是用來找最近的 Dealine 是哪一個 Task，在開始之前會先用自己寫的 OS_Check_MissDeadline 檢查是否有任務 Miss Deadline，而之後副程式中的第一個迴圈是去按照 Task ID 的順序紀錄每一個 Task 的 Deadline 並儲存於 array 中，方便我們在第二個迴圈去尋找所有Deadline中的最小值，除此之外，萬一發現有相同的 Deadline 最小值時，要再去比較何者的執行時間 Execution Time 最小，選最小的 Execution Time 去執行，最後在找到確切下一個要執行的任務時，變更 Priority 前先確認是否所有任務優先權皆為原本最初的優先權，再去將下一個該執行的任務其優先權變更為最高優先0：

由於有時自己撰寫的程式可能會發生 Deadline 沒有在任務一結束就更新，反而在Deadline時間到才更新，因此會在1818行做判斷，若發現現在剛好任務執行完了，就立刻刷新 Deadline，避免造成 Deadline 尚未更新，選擇錯誤的 Task 去執行的情況發生：



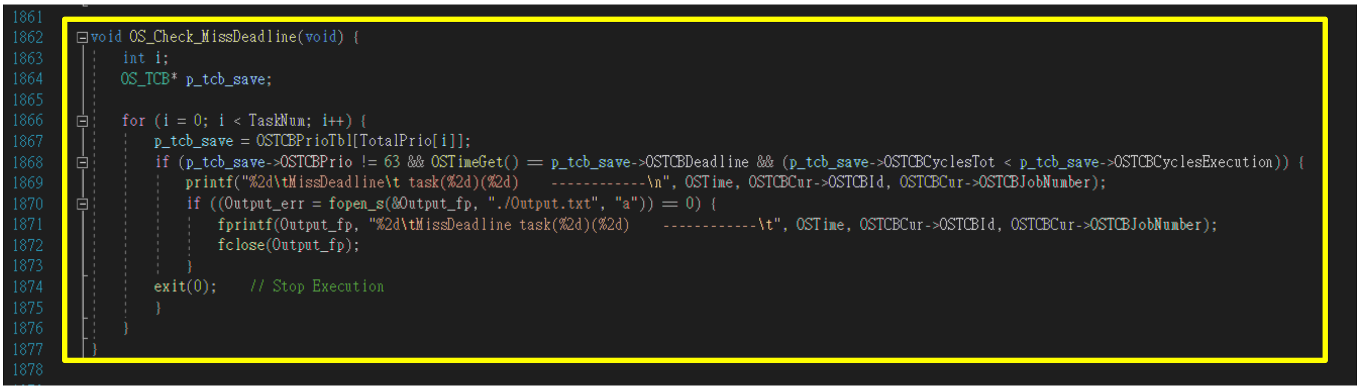
```c
/* AddedCodePA2part1
*********************************************************************************************************
*                                              Early Deadline First
*
* Description: The highest priority is decided by the earliest deadline
*
* Arguments  : none
*
* Returns    : none
*
* Notes      : 1) Written for PA2 part1
*********************************************************************************************************
*/

void OS_EDF_Int(void)
{
    int i;
    int TotalDeadline[5] = { 999 };
    int SortedDeadline[5] = { 999 };
    OS_TCB* p_tcb_save;
    OS_TCB* p_tcb_smallest = OSTCBCur;

    OS_Check_MissDeadline();    // Check MissDeadline or not

    // Get all the Deadlines
    for (i = 0; i < TaskNum; i++) {
        p_tcb_save = OSTCBPrioTbl[TotalPrio[i]];
        TotalDeadline[i] = p_tcb_save->OSTCBCyclesArrive + (p_tcb_save->OSTCBJobNumber + 1) * p_tcb_save->OSTCBCyclesPeriod;
        //if (OSTimeGet() == TotalDeadline[i]) {
        if (OSTimeGet() == p_tcb_save->OSTCBCyclesEnd) {
            TotalDeadline[i] = TotalDeadline[i] + p_tcb_save->OSTCBCyclesPeriod;
            p_tcb_save->OSTCBDeadline = TotalDeadline[i];
        }
    }

    }


    // Find the Eariliest Deadline
    // If Eariliest Deadline are same, choose the smallest execution time
    for (i = 0; i < TaskNum; i++) {
        p_tcb_save = OSTCBPrioTbl[TotalPrio[i]];
        if (i == 0)
            p_tcb_smallest = OSTCBCur;
        else if(p_tcb_save->OSTCBDeadline!=0 && p_tcb_save->OSTCBDeadline< p_tcb_smallest->OSTCBDeadline)   // Compare which deadline small
            p_tcb_smallest = OSTCBPrioTbl[TotalPrio[i]];
        else if (p_tcb_save->OSTCBDeadline != 0 && p_tcb_save->OSTCBDeadline == p_tcb_smallest->OSTCBDeadline) {// If have same small deadline, compare execution time
            if(p_tcb_save->OSTCBCyclesExecution < p_tcb_smallest->OSTCBCyclesExecution)
                p_tcb_smallest = OSTCBPrioTbl[TotalPrio[i]];
        }
    }

    // Change the priority
    OS_CorrectPrio();//
    if (p_tcb_smallest->OSTCBPrio != 63) {
        TotalPrio[p_tcb_smallest->OSTCBId] = 0;
        OSTaskChangePrio(p_tcb_smallest->OSTCBPrio, 0);
        //OS_CorrectPrio();
    }
}
```

副程式 OS_CorrectPrio 是用來校正優先權，因為每次都會把當下要執行的任務優先權變為 0，因此要額外撰寫一個副程式去校正所有任務的優先權為原本的優先權，避免發生所有任務優先權都變 0 的錯誤發生：



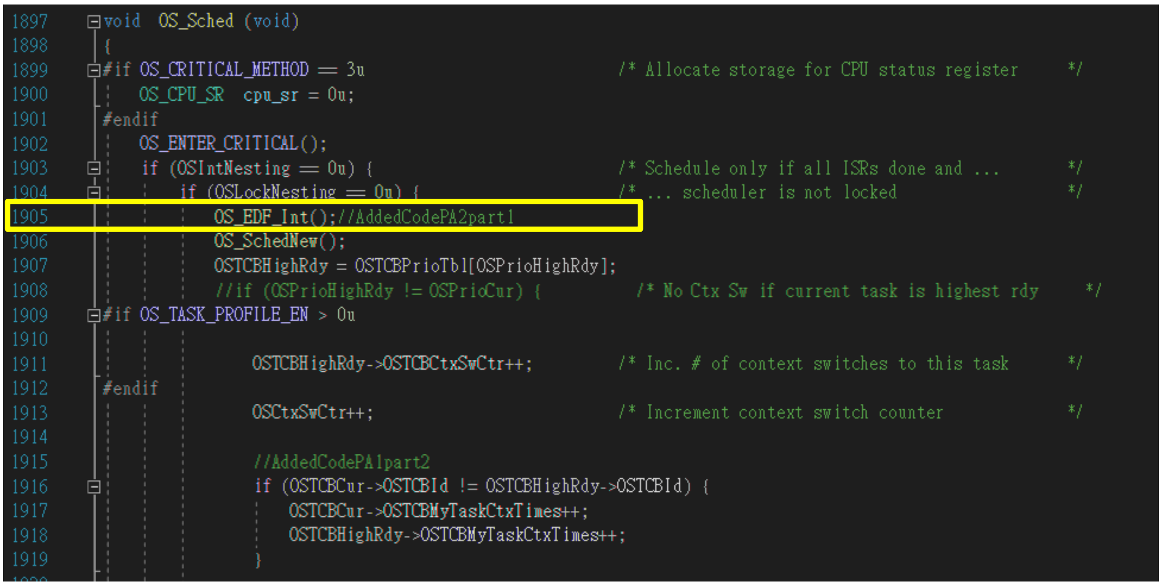
```c
void OS_CorrectPrio(void) {
    int i;
    int prio;
    OS_TCB* p_tcb_save;

    for (i = 0; i < TaskNum; i++) {
        p_tcb_save = OSTCBPrioTbl[TotalPrio[i]];
        if (p_tcb_save->OSTCBPrio != p_tcb_save->OSTCBOriPrio) {
            OSTaskChangePrio(p_tcb_save->OSTCBPrio, p_tcb_save->OSTCBOriPrio);
            TotalPrio[i] = p_tcb_save->OSTCBOriPrio;
        }
    }
}
```

副程式 OS_Check_MissDeadline 會在每一個Tick都被呼叫到一次，會在這裡檢查所有 Task 有無 Miss Deadline 的情況發生，而判斷依據是現在的時間若等於 Deadline，會去檢查這個 Task 從開始執行到現在過了多久，若小於它應有的 Execution Time 就代表該 Task 會來不及執行，也就是發生 Miss Deadline：



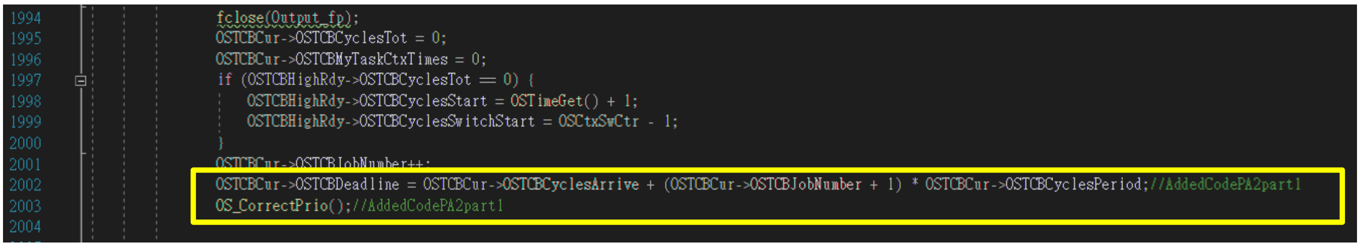
```
void OS_Check_MissDeadline(void) {
    int i;
    OS_TCB* p_tcb_save;

    for (i = 0; i < TaskNum; i++) {
        p_tcb_save = OSTCBPrioTbl[TotalPrio[i]];
        if (p_tcb_save->OSTCBPrio != 63 && OSTimeGet() == p_tcb_save->OSTCBDeadline && (p_tcb_save->OSTCBCyclesTot < p_tcb_save->OSTCBCyclesExecution)) {
            printf("%2d\tMissDeadline\t task(%2d)(%2d)    ------------\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBJobNumber);
            if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0) {
                fprintf(Output_fp, "%2d\tMissDeadline task(%2d)(%2d)    ------------\t", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBJobNumber);
                fclose(Output_fp);
            }
        exit(0);    // Stop Execution
        }
    }
}
```

呼叫自己撰寫的副程式的時機 - OS_Sched 中，會在原本尋找 High Priority 前先呼叫 OS_EDF_Int 去更新 OSTCBCur ：



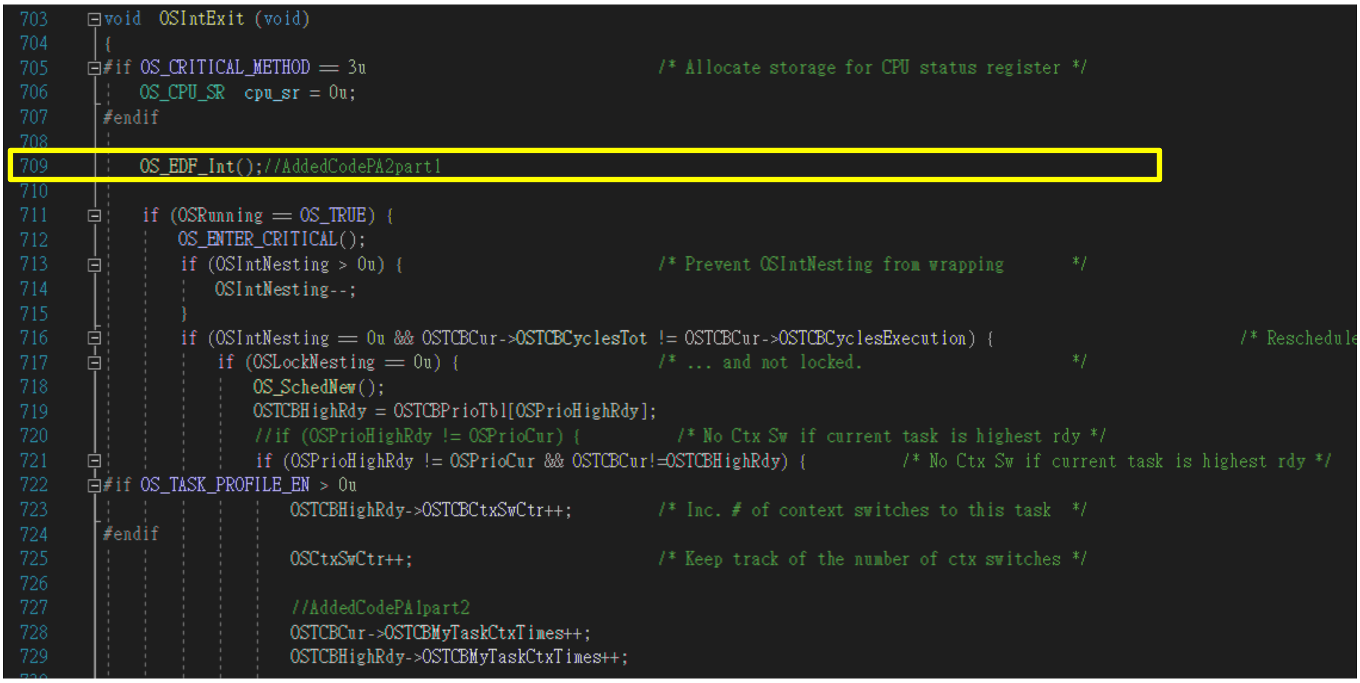
```
void  OS_Sched (void)
{
#if OS_CRITICAL_METHOD == 3u                           /* Allocate storage for CPU status register     */
    OS_CPU_SR  cpu_sr = 0u;
#endif
    OS_ENTER_CRITICAL();
    if (OSIntNesting == 0u) {                          /* Schedule only if all ISRs done and ...       */
        if (OSLockNesting == 0u) {                     /* ... scheduler is not locked                  */
            OS_EDF_Int();//AddedCodePA2part1
            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      */
#endif
                OSCtxSwCtr++;                          /* Increment context switch counter             */

                //AddedCodePA1part2
                if (OSTCBCur->OSTCBId != OSTCBHighRdy->OSTCBId) {
                    OSTCBCur->OSTCBMyTaskCtxTimes++;
                    OSTCBHighRdy->OSTCBMyTaskCtxTimes++;
                }
```

並且很重要的一點是要記得在OS_Sched 結束前記得把優先權校正回原本的樣子，還要記得更新 Deadline：



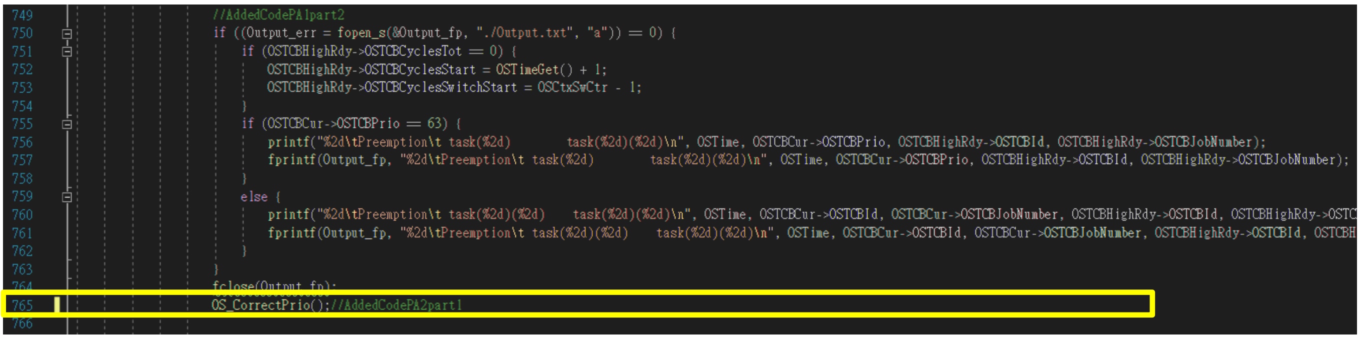
```
                fclose(Output_fp);
                OSTCBCur->OSTCBCyclesTot = 0;
                OSTCBCur->OSTCBMyTaskCtxTimes = 0;
                if (OSTCBHighRdy->OSTCBCyclesTot == 0) {
                    OSTCBHighRdy->OSTCBCyclesStart = OSTimeGet() + 1;
                    OSTCBHighRdy->OSTCBCyclesSwitchStart = OSCtxSwCtr - 1;
                }
                OSTCBCur->OSTCBJobNumber++;
                OSTCBCur->OSTCBDeadline = OSTCBCur->OSTCBCyclesArrive + (OSTCBCur->OSTCBJobNumber + 1) * OSTCBCur->OSTCBCyclesPeriod;//AddedCodePA2part1
                OS_CorrectPrio();//AddedCodePA2part1
```

呼叫自己撰寫的副程式的時機 - OSIntExit 中，在這裡我嘗試過在不同地方或是更晚呼叫，發現不能太晚呼叫，否則可能更改得太晚造成有點來不及，因此最後選擇在剛進 OSIntExit 不久就先呼叫 OS_EDF_Int 去尋找 OSTCBCur 應是誰：



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

    OS_EDF_Int();//AddedCodePA2part1

    if (OSRunning == OS_TRUE) {
        OS_ENTER_CRITICAL();
        if (OSIntNesting > 0u) {                           /* Prevent OSIntNesting from wrapping       */
            OSIntNesting--;
        }
        if (OSIntNesting == 0u && OSTCBCur->OSTCBCyclesTot != OSTCBCur->OSTCBCyclesExecution) {                   /* Reschedule
            if (OSLockNesting == 0u) {                     /* ... and not locked.                      */
                OS_SchedNew();
                OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy];
                //if (OSPrioHighRdy != OSPrioCur) {          /* No Ctx Sw if current task is highest rdy */
                if (OSPrioHighRdy != OSPrioCur && OSTCBCur!=OSTCBHighRdy) {          /* No Ctx Sw if current task is highest rdy */
#if OS_TASK_PROFILE_EN > 0u
                    OSTCBHighRdy->OSTCBCtxSwCtr++;         /* Inc. # of context switches to this task  */
#endif
                    OSCtxSwCtr++;                          /* Keep track of the number of ctx switches */

                    //AddedCodePA1part2
                    OSTCBCur->OSTCBMyTaskCtxTimes++;
                    OSTCBHighRdy->OSTCBMyTaskCtxTimes++;
```

很重要的一點是要記得在 OSIntExit 結束前記得把優先權校正回原本的樣子：



```
                    //AddedCodePA1part2
                    if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0) {
                        if (OSTCBHighRdy->OSTCBCyclesTot == 0) {
                            OSTCBHighRdy->OSTCBCyclesStart = OSTimeGet() + 1;
                            OSTCBHighRdy->OSTCBCyclesSwitchStart = OSCtxSwCtr - 1;
                        }
                        if (OSTCBCur->OSTCBPrio == 63) {
                            printf("%2d\tPreemption\t task(%2d)        task(%2d)(%2d)\n", OSTime, OSTCBCur->OSTCBPrio, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBJobNumber);
                            fprintf(Output_fp, "%2d\tPreemption\t task(%2d)        task(%2d)(%2d)\n", OSTime, OSTCBCur->OSTCBPrio, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBJobNumber);
                        }
                        else {
                            printf("%2d\tPreemption\t task(%2d)(%2d)    task(%2d)(%2d)\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBJobNumber, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTC
                            fprintf(Output_fp, "%2d\tPreemption\t task(%2d)(%2d)    task(%2d)(%2d)\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->OSTCBJobNumber, OSTCBHighRdy->OSTCBId, OSTCBH
                        }
                    }
                    fclose(Output_fp);
                    OS_CorrectPrio();//AddedCodePA2part1
```

[Part II] CUS Scheduler Implementation

Objective:

To implement Constant Utilization Servers (CUS) for serving aperiodic tasks and to observe the scheduling behaviors.

Problem Definition:

As you did in Part I, uC/OS-II supports the EDF scheduling algorithm. Based on your EDF scheduler, you are going to implement the Constant Utilization Servers (CUS) for serving aperiodic tasks.

Implement the following two task sets. Add necessary code to the µC/OS-II scheduler **in the kernel level** to observe how the task suffers the schedule delay.

Some periodic tasks and aperiodic jobs are included in the following two examples.

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

Aperiodic Job Set = {j_{num} (num, arrival time, execution time, absolute deadline)}

============================== **Example** ===================================

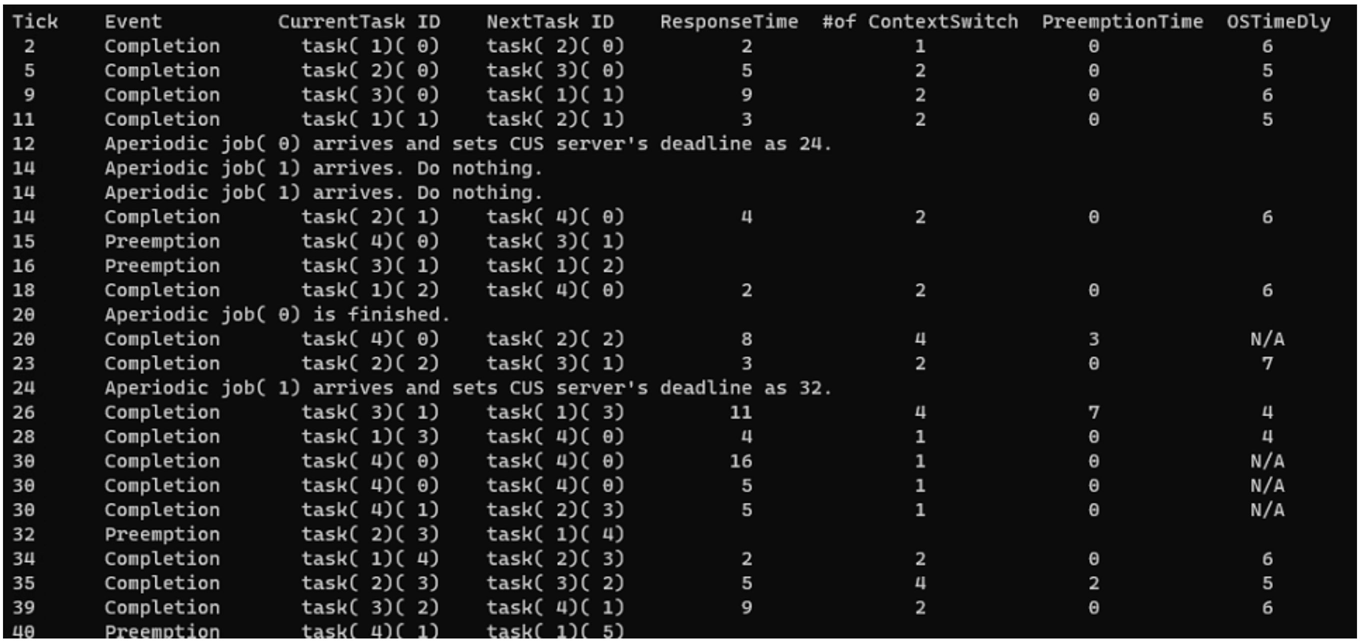
Periodic Task Set = {τ_1 (1, 0, 2, 8), τ_2 (2, 0, 3, 10), τ_3 (3, 0, 4, 15), $\tau_{4\_ServerSize}$ (4, 25%)}

Aperiodic Jobs Set = {j_0 (0, 12, 3, 26), j_1 (1, 14, 2, 34)}

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

※ If there are tasks with the same deadlines, the task with a lower task ID will be executed first.

The output results：

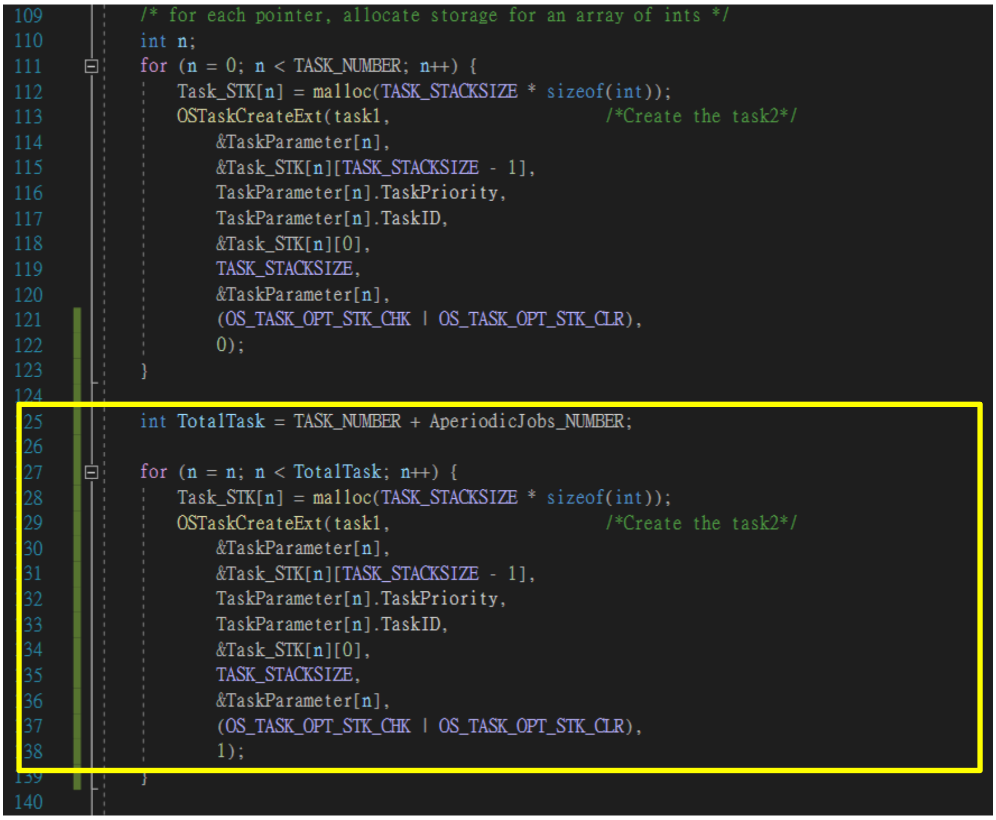


```
Tick    Event          CurrentTask ID    NextTask ID   ResponseTime  #of ContextSwitch   PreemptionTime  OSTimeDly
 2      Completion       task( 1)( 0)    task( 2)( 0)          2              1              0              6
 5      Completion       task( 2)( 0)    task( 3)( 0)          5              2              0              5
 9      Completion       task( 3)( 0)    task( 1)( 1)          9              2              0              6
11      Completion       task( 1)( 1)    task( 2)( 1)          3              2              0              5
12      Aperiodic job( 0) arrives and sets CUS server's deadline as 24.
14      Aperiodic job( 1) arrives. Do nothing.
14      Aperiodic job( 1) arrives. Do nothing.
14      Completion       task( 2)( 1)    task( 4)( 0)          4              2              0              6
15      Preemption       task( 4)( 0)    task( 3)( 1)
16      Preemption       task( 3)( 1)    task( 1)( 2)
18      Completion       task( 1)( 2)    task( 4)( 0)          2              2              0              6
20      Aperiodic job( 0) is finished.
20      Completion       task( 4)( 0)    task( 2)( 2)          8              4              3             N/A
23      Completion       task( 2)( 2)    task( 3)( 1)          3              2              0              7
24      Aperiodic job( 1) arrives and sets CUS server's deadline as 32.
26      Completion       task( 3)( 1)    task( 1)( 3)         11              4              7              4
28      Completion       task( 1)( 3)    task( 4)( 0)          4              1              0              4
30      Completion       task( 4)( 0)    task( 4)( 0)         16              1              0             N/A
30      Completion       task( 4)( 0)    task( 4)( 0)          5              1              0             N/A
30      Completion       task( 4)( 1)    task( 2)( 3)          5              1              0             N/A
32      Preemption       task( 2)( 3)    task( 1)( 4)
34      Completion       task( 1)( 4)    task( 2)( 3)          2              2              0              6
35      Completion       task( 2)( 3)    task( 3)( 2)          5              4              2              5
39      Completion       task( 3)( 2)    task( 4)( 1)          9              2              0              6
40      Preemption       task( 4)( 1)    task( 1)( 5)
```

```
2      Completion          task( 1)( 0)   task( 2)( 0)      2     1     0     6
5      Completion          task( 2)( 0)   task( 3)( 0)      5     2     0     5
9      Completion          task( 3)( 0)   task( 1)( 1)      9     2     0     6
11     Completion          task( 1)( 1)   task( 2)( 1)      3     2     0     5
12     Aperiodic job( 0) arrives and sets CUS server's deadline as 24.
14     Aperiodic job( 1) arrives. Do nothing.
14     Aperiodic job( 1) arrives. Do nothing.
14     Completion          task( 2)( 1)   task( 4)( 0)      4     2     0     6
15     Preemption          task( 4)( 0)   task( 3)( 1)
16     Preemption          task( 3)( 1)   task( 1)( 2)
18     Completion          task( 1)( 2)   task( 4)( 0)      2     2     0     6
20     Aperiodic job( 0) is finished.
20     Completion          task( 4)( 0)   task( 2)( 2)      8     4     3     N/A
23     Completion          task( 2)( 2)   task( 3)( 1)      3     2     0     7
24     Aperiodic job( 1) arrives and sets CUS server's deadline as 32.
26     Completion          task( 3)( 1)   task( 1)( 3)      11     4     7     4
28     Completion          task( 1)( 3)   task( 4)( 0)      4     1     0     4
30     Completion          task( 4)( 0)   task( 4)( 0)      16     1     0     N/A
30     Completion          task( 4)( 0)   task( 4)( 0)      5     1     0     N/A
30     Completion          task( 4)( 1)   task( 2)( 3)      5     1     0     N/A
32     Preemption          task( 2)( 3)   task( 1)( 4)
34     Completion          task( 1)( 4)   task( 2)( 3)      2     2     0     6
35     Completion          task( 2)( 3)   task( 3)( 2)      5     4     2     5
39     Completion          task( 3)( 2)   task( 4)( 1)      9     2     0     6
40     Preemption          task( 4)( 1)   task( 1)( 5)
```

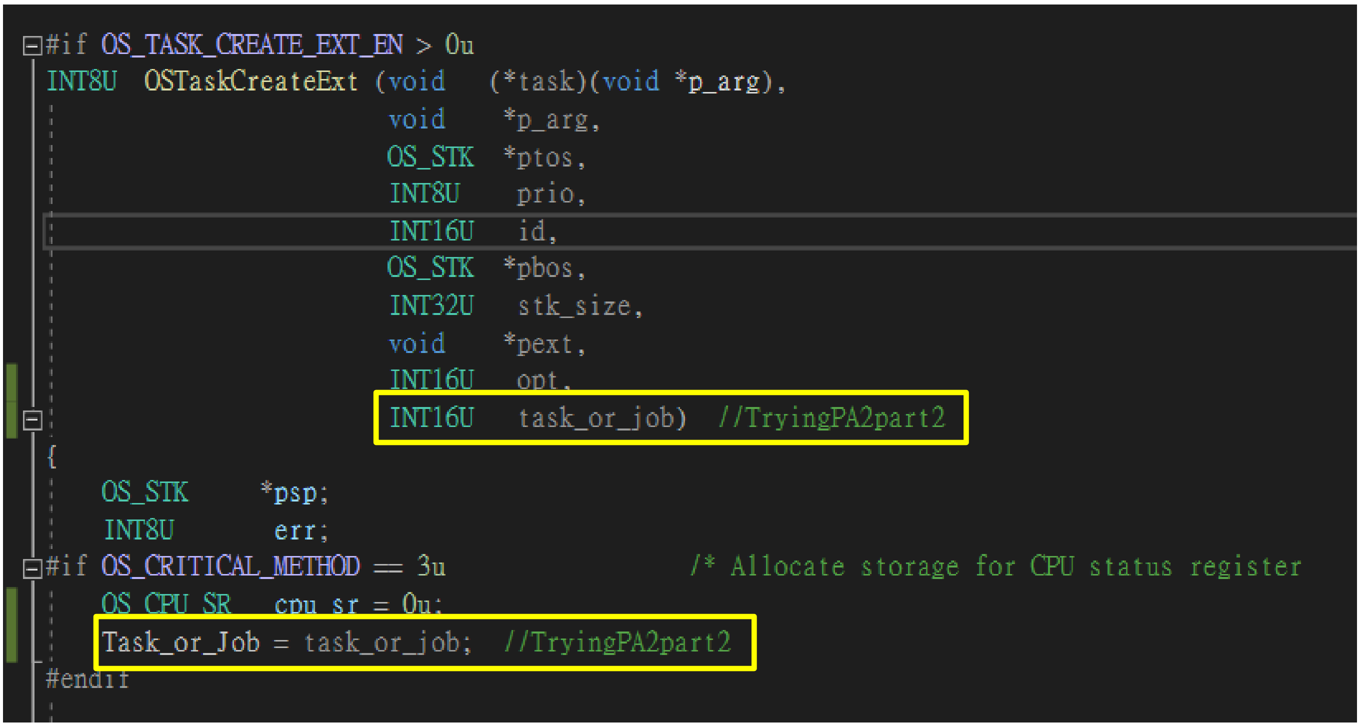
Experiment Report：

由於需要多讀一個檔案，因此在main.c的部分新增讀檔功能，方便獲取非週期任務的資訊：(其中另外宣告 AperiodJobs_NUMBER 去紀錄非週期任務的數量)



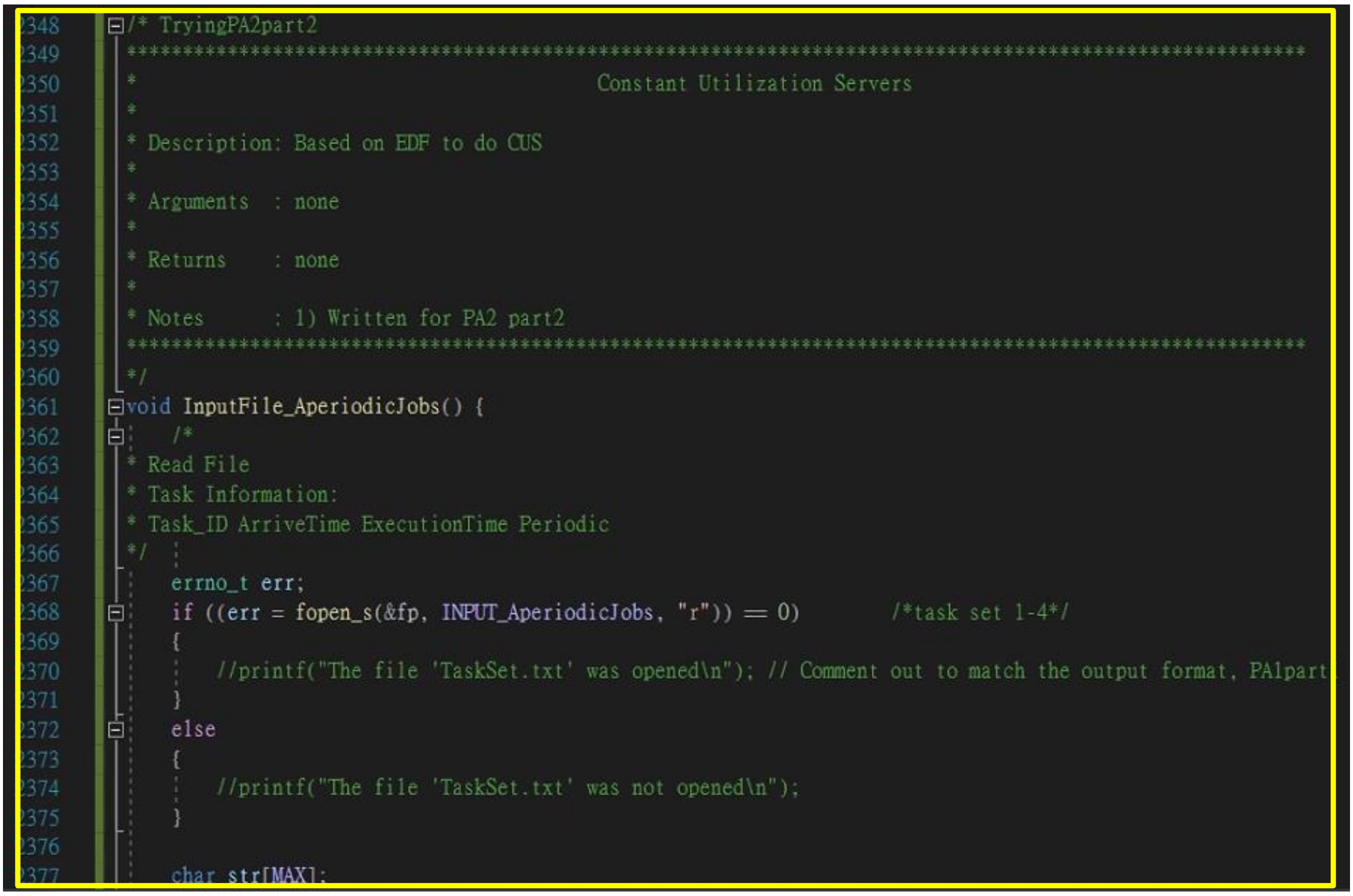
```c
	/* for each pointer, allocate storage for an array of ints */
	int n;
	for (n = 0; n < TASK_NUMBER; n++) {
		Task_STK[n] = malloc(TASK_STACKSIZE * sizeof(int));
		OSTaskCreateExt(task1,                          /*Create the task2*/
			&TaskParameter[n],
			&Task_STK[n][TASK_STACKSIZE - 1],
			TaskParameter[n].TaskPriority,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

	int TotalTask = TASK_NUMBER + AperiodicJobs_NUMBER;

	for (n = n; n < TotalTask; 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),
			1);
	}
```

而為了方便辨別誰是非週期任務以及誰是週期任務，因此在 os_task.c 的任務建立 OSTaskCreateExt 中去新增了變數 task_or_job ，若為0代表為週期任務，若為1則代表非週期任務：

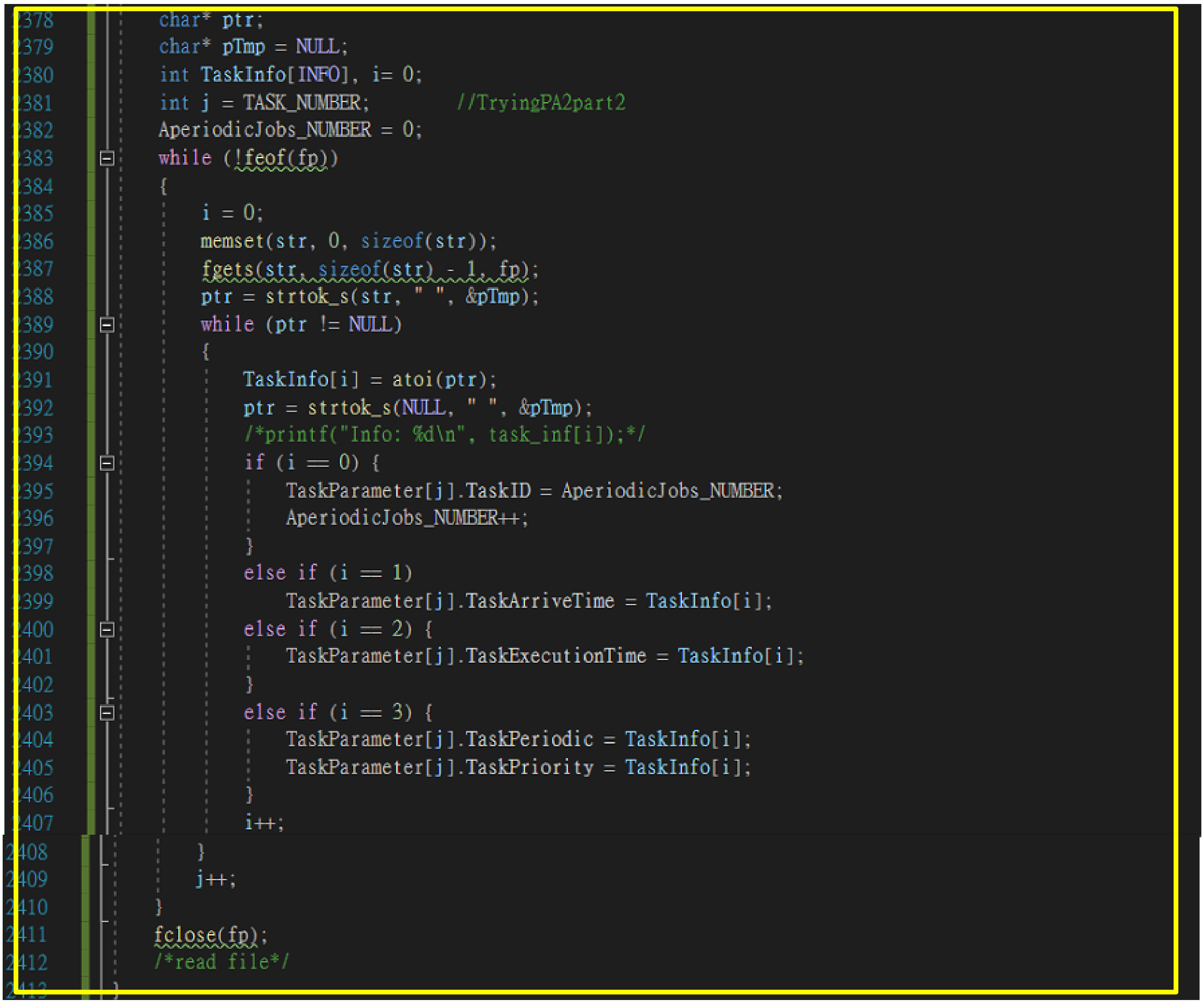


```
#if OS_TASK_CREATE_EXT_EN > 0u
INT8U  OSTaskCreateExt (void   (*task)(void *p_arg),
                        void    *p_arg,
                        OS_STK  *ptos,
                        INT8U    prio,
                        INT16U   id,
                        OS_STK  *pbos,
                        INT32U   stk_size,
                        void    *pext,
                        INT16U   opt,
                        INT16U   task_or_job)  //TryingPA2part2
{
    OS_STK     *psp;
    INT8U       err;
#if OS_CRITICAL_METHOD == 3u                          /* Allocate storage for CPU status register
    OS_CPU_SR   cpu_sr = 0u;
    Task_or_Job = task_or_job;  //TryingPA2part2
#endif
```

其餘剩下的撰寫部分都在 os_core.c 中進行，而因為非週期任務有需要重新讀檔的可能，因此撰寫了讀檔程式 InputFile_AperiodJobs 去獲取參數：

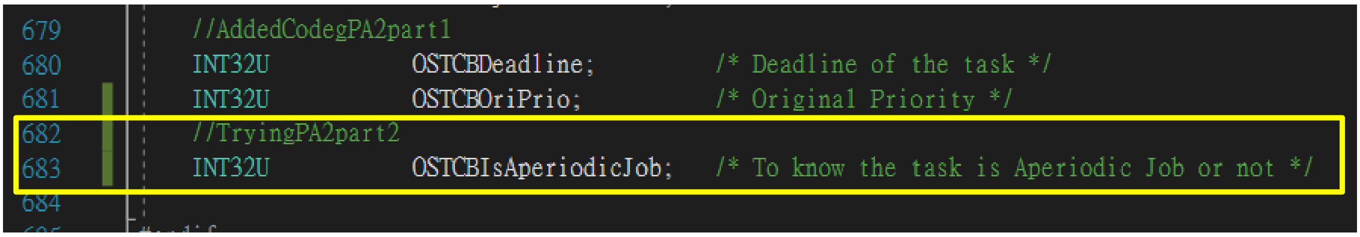


```
/* TryingPA2part2
*****************************************************************************************************
*                                      Constant Utilization Servers
*
* Description: Based on EDF to do CUS
*
* Arguments  : none
*
* Returns    : none
*
* Notes      : 1) Written for PA2 part2
*****************************************************************************************************
*/
void InputFile_AperiodicJobs() {
    /*
* Read File
* Task Information:
* Task_ID ArriveTime ExecutionTime Periodic
*/
    errno_t err;
    if ((err = fopen_s(&fp, INPUT_AperiodicJobs, "r")) == 0)        /*task set 1-4*/
    {
        //printf("The file 'TaskSet.txt' was opened\n"); // Comment out to match the output format, PA1part
    }
    else
    {
        //printf("The file 'TaskSet.txt' was not opened\n");
    }

    char str[MAX];
```



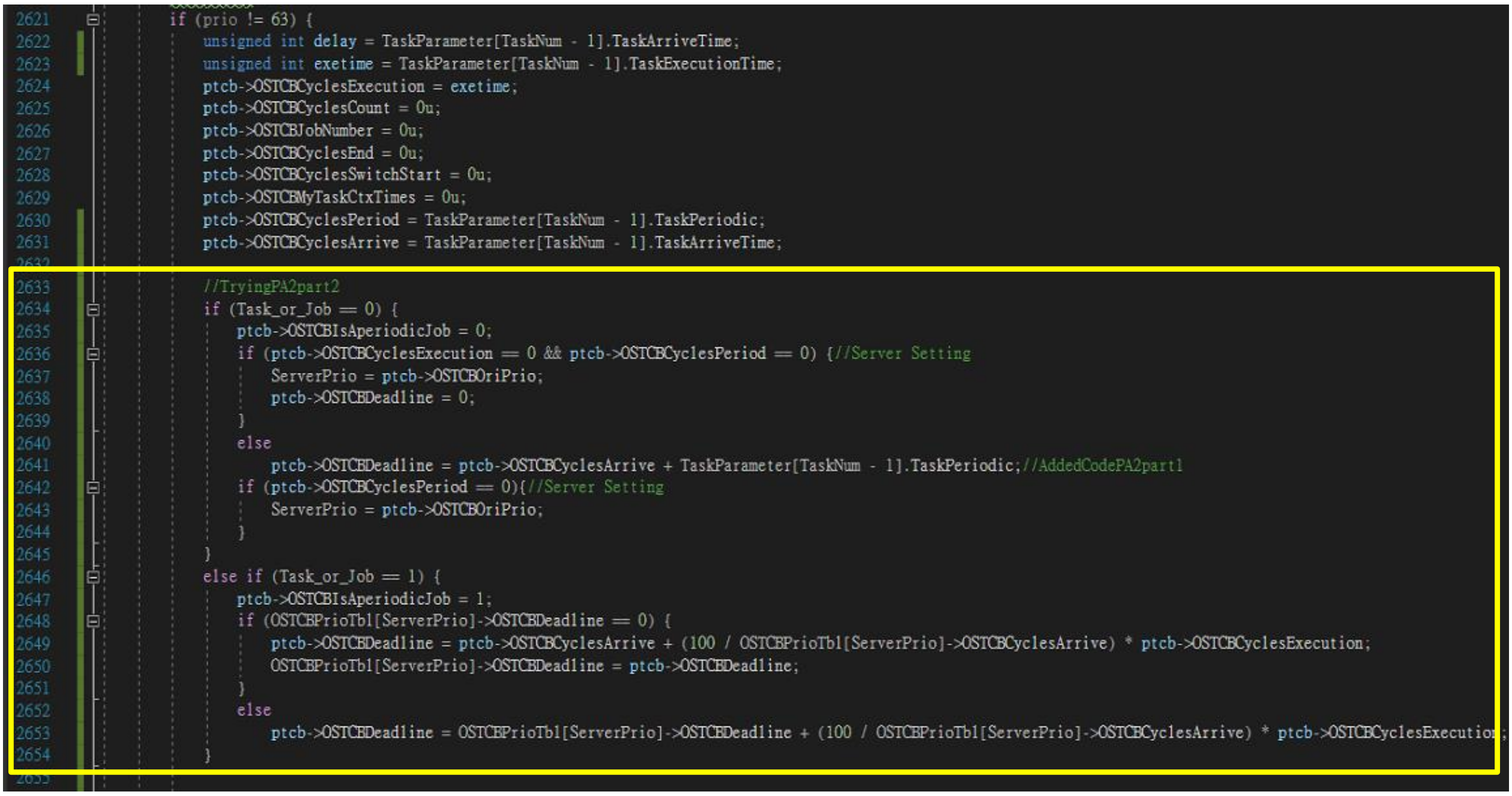
```
    char* ptr;
    char* pTmp = NULL;
    int TaskInfo[INFO], i= 0;
    int j = TASK_NUMBER;        //TryingPA2part2
    AperiodicJobs_NUMBER = 0;
    while (!feof(fp))
    {
        i = 0;
        memset(str, 0, sizeof(str));
        fgets(str, sizeof(str) - 1, fp);
        ptr = strtok_s(str, " ", &pTmp);
        while (ptr != NULL)
        {
            TaskInfo[i] = atoi(ptr);
            ptr = strtok_s(NULL, " ", &pTmp);
            /*printf("Info: %d\n", task_inf[i]);*/
            if (i == 0) {
                TaskParameter[j].TaskID = AperiodicJobs_NUMBER;
                AperiodicJobs_NUMBER++;
            }
            else if (i == 1)
                TaskParameter[j].TaskArriveTime = TaskInfo[i];
            else if (i == 2) {
                TaskParameter[j].TaskExecutionTime = TaskInfo[i];
            }
            else if (i == 3) {
                TaskParameter[j].TaskPeriodic = TaskInfo[i];
                TaskParameter[j].TaskPriority = TaskInfo[i];
            }
            i++;
        }
        j++;
    }
    fclose(fp);
    /*read file*/
}
```

由於有非週期任務與週期任務，因此在TCB初始化時新增一變數 OSTCBIsAperiodJob 去做紀錄該筆 Task 為何種任務：



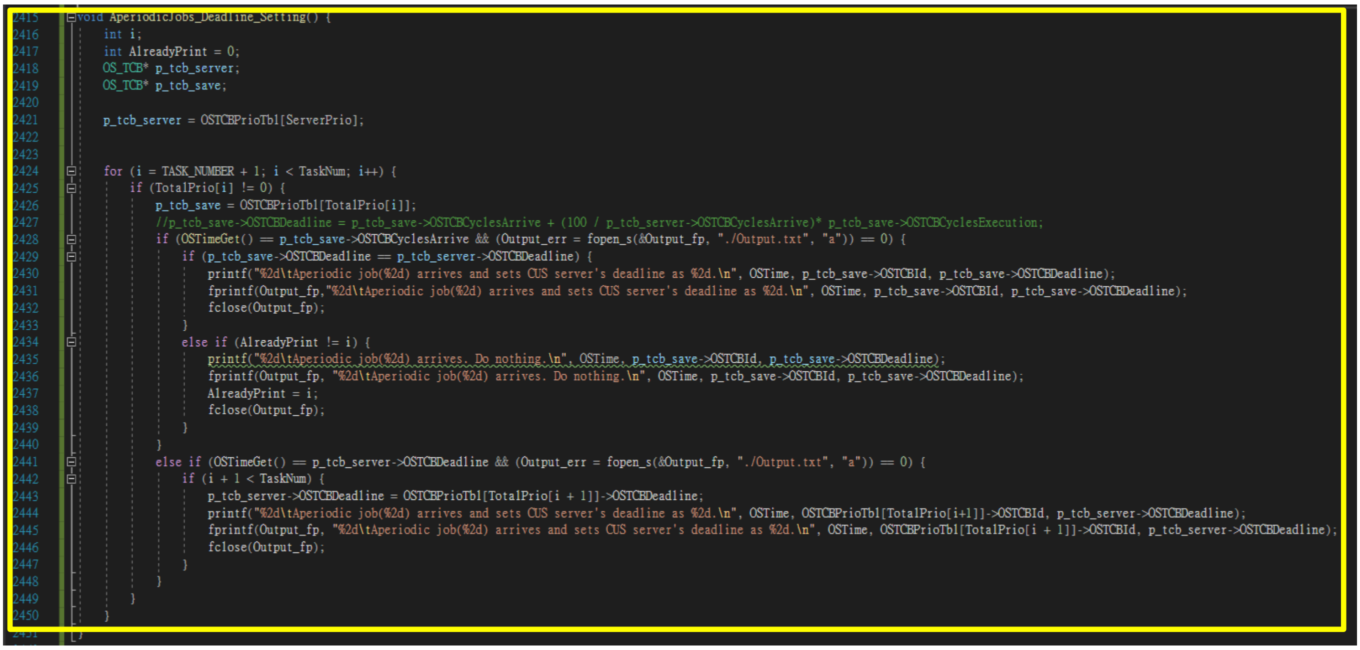
```
    //AddedCodegPA2part1
    INT32U           OSTCBDeadline;          /* Deadline of the task */
    INT32U           OSTCBOriPrio;           /* Original Priority */
    //TryingPA2part2
    INT32U           OSTCBIsAperiodicJob;    /* To know the task is Aperiodic Job or not */

#endif
```

在 TCB 初始化時，依據參數 OSTCBIsAperiodJob 去做判斷並為 Server 做設定(Server ID恰巧為週期任務數量)，並再針對週期任務與非週期任務去計算各自的 Deadline：



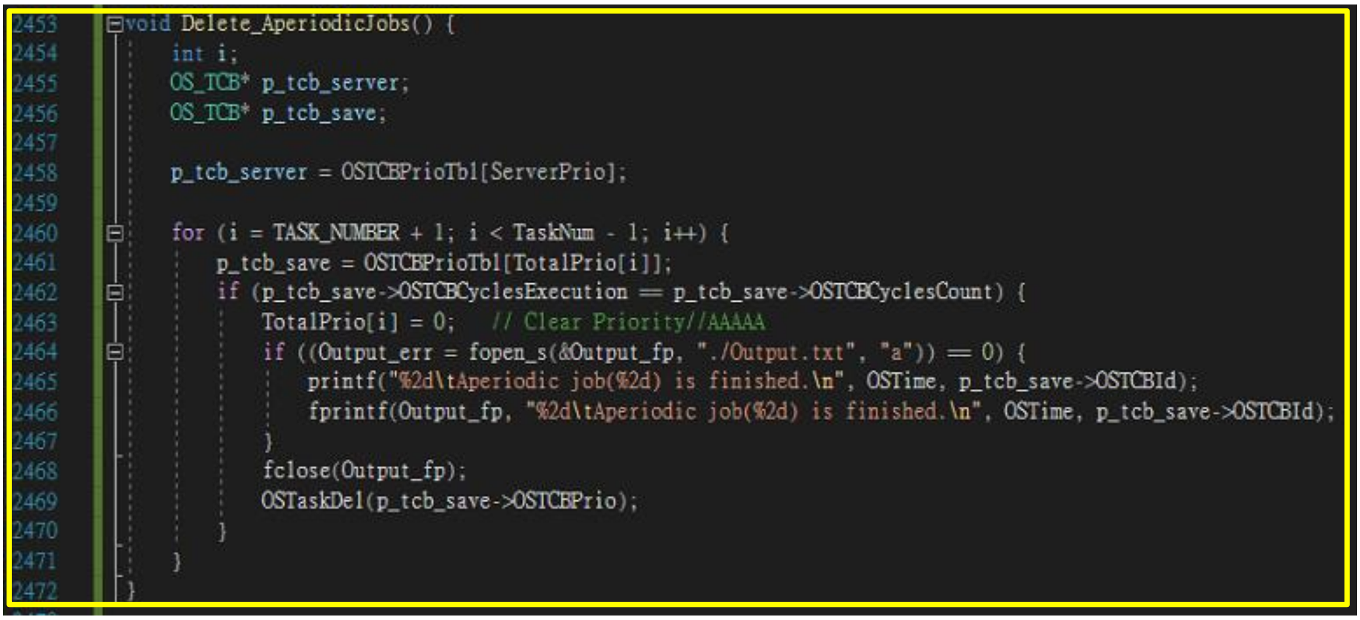
```
if (prio != 63) {
    unsigned int delay = TaskParameter[TaskNum - 1].TaskArriveTime;
    unsigned int exetime = TaskParameter[TaskNum - 1].TaskExecutionTime;
    ptcb->OSTCBCyclesExecution = exetime;
    ptcb->OSTCBCyclesCount = 0u;
    ptcb->OSTCBJobNumber = 0u;
    ptcb->OSTCBCyclesEnd = 0u;
    ptcb->OSTCBCyclesSwitchStart = 0u;
    ptcb->OSTCBMyTaskCtxTimes = 0u;
    ptcb->OSTCBCyclesPeriod = TaskParameter[TaskNum - 1].TaskPeriodic;
    ptcb->OSTCBCyclesArrive = TaskParameter[TaskNum - 1].TaskArriveTime;

    //TryingPA2part2
    if (Task_or_Job == 0) {
        ptcb->OSTCBIsAperiodicJob = 0;
        if (ptcb->OSTCBCyclesExecution == 0 && ptcb->OSTCBCyclesPeriod == 0) {//Server Setting
            ServerPrio = ptcb->OSTCBOriPrio;
            ptcb->OSTCBDeadline = 0;
        }
        else
            ptcb->OSTCBDeadline = ptcb->OSTCBCyclesArrive + TaskParameter[TaskNum - 1].TaskPeriodic;//AddedCodePA2part1
        if (ptcb->OSTCBCyclesPeriod == 0){//Server Setting
            ServerPrio = ptcb->OSTCBOriPrio;
        }
    }
    else if (Task_or_Job == 1) {
        ptcb->OSTCBIsAperiodicJob = 1;
        if (OSTCBPrioTbl[ServerPrio]->OSTCBDeadline == 0) {
            ptcb->OSTCBDeadline = ptcb->OSTCBCyclesArrive + (100 / OSTCBPrioTbl[ServerPrio]->OSTCBCyclesArrive) * ptcb->OSTCBCyclesExecution;
            OSTCBPrioTbl[ServerPrio]->OSTCBDeadline = ptcb->OSTCBDeadline;
        }
        else
            ptcb->OSTCBDeadline = OSTCBPrioTbl[ServerPrio]->OSTCBDeadline + (100 / OSTCBPrioTbl[ServerPrio]->OSTCBCyclesArrive) * ptcb->OSTCBCyclesExecution;
    }
```

程式的主架構為EDF，因此我是以新撰寫函數+修改part1的部分去做結合，新撰寫函數的部分 AperiodicJobs_Deadline_Setting 副程式中，每一個 tick 都會進入這個副程式，並每次為非週期性任務做 Deadline 的檢查與計算，Server 隨著非週期性任務的不同，其設定上的更改和打印都在這個副程式進行：



```
void AperiodicJobs_Deadline_Setting() {
    int i;
    int AlreadyPrint = 0;
    OS_TCB* p_tcb_server;
    OS_TCB* p_tcb_save;

    p_tcb_server = OSTCBPrioTbl[ServerPrio];


    for (i = TASK_NUMBER + 1; i < TaskNum; i++) {
        if (TotalPrio[i] != 0) {
            p_tcb_save = OSTCBPrioTbl[TotalPrio[i]];
            //p_tcb_save->OSTCBDeadline = p_tcb_save->OSTCBCyclesArrive + (100 / p_tcb_server->OSTCBCyclesArrive)* p_tcb_save->OSTCBCyclesExecution;
            if (OSTimeGet() == p_tcb_save->OSTCBCyclesArrive && (Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0) {
                if (p_tcb_save->OSTCBDeadline == p_tcb_server->OSTCBDeadline) {
                    printf("%2d\tAperiodic job(%2d) arrives and sets CUS server's deadline as %2d.\n", OSTime, p_tcb_save->OSTCBId, p_tcb_save->OSTCBDeadline);
                    fprintf(Output_fp,"%2d\tAperiodic job(%2d) arrives and sets CUS server's deadline as %2d.\n", OSTime, p_tcb_save->OSTCBId, p_tcb_save->OSTCBDeadline);
                    fclose(Output_fp);
                }
                else if (AlreadyPrint != i) {
                    printf("%2d\tAperiodic job(%2d) arrives. Do nothing.\n", OSTime, p_tcb_save->OSTCBId, p_tcb_save->OSTCBDeadline);
                    fprintf(Output_fp, "%2d\tAperiodic job(%2d) arrives. Do nothing.\n", OSTime, p_tcb_save->OSTCBId, p_tcb_save->OSTCBDeadline);
                    AlreadyPrint = i;
                    fclose(Output_fp);
                }
            }
            else if (OSTimeGet() == p_tcb_server->OSTCBDeadline && (Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0) {
                if (i + 1 < TaskNum) {
                    p_tcb_server->OSTCBDeadline = OSTCBPrioTbl[TotalPrio[i + 1]]->OSTCBDeadline;
                    printf("%2d\tAperiodic job(%2d) arrives and sets CUS server's deadline as %2d.\n", OSTime, OSTCBPrioTbl[TotalPrio[i+1]]->OSTCBId, p_tcb_server->OSTCBDeadline);
                    fprintf(Output_fp, "%2d\tAperiodic job(%2d) arrives and sets CUS server's deadline as %2d.\n", OSTime, OSTCBPrioTbl[TotalPrio[i + 1]]->OSTCBId, p_tcb_server->OSTCBDeadline);
                    fclose(Output_fp);
                }
            }
        }
    }
```

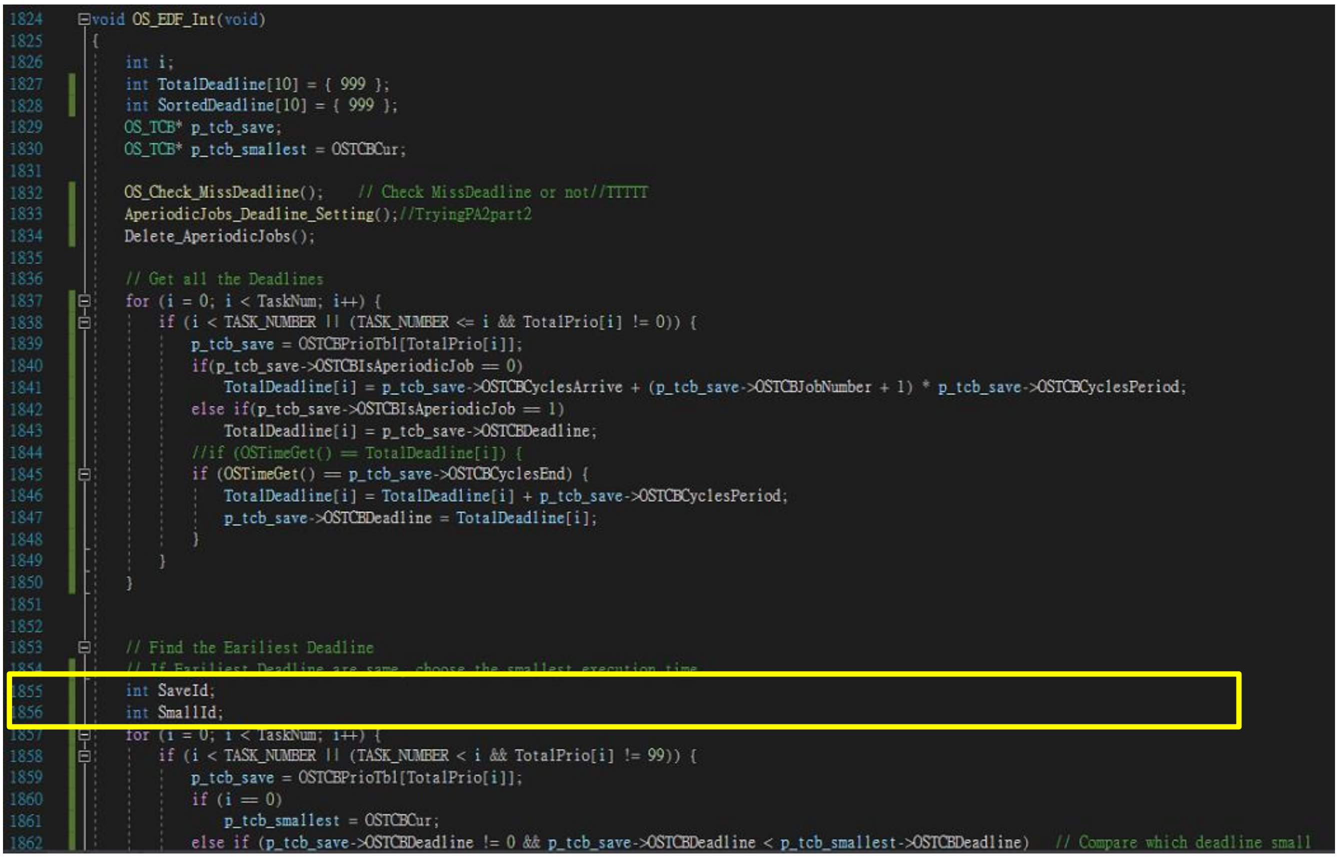
在非週期性任務執行完後即不存在，因此額外撰寫一副程式 Delete_AperiodicJob 也一樣在每一個 tick 時進入副程式做檢查當下是否有非週期任務執行完畢，並將其刪除：



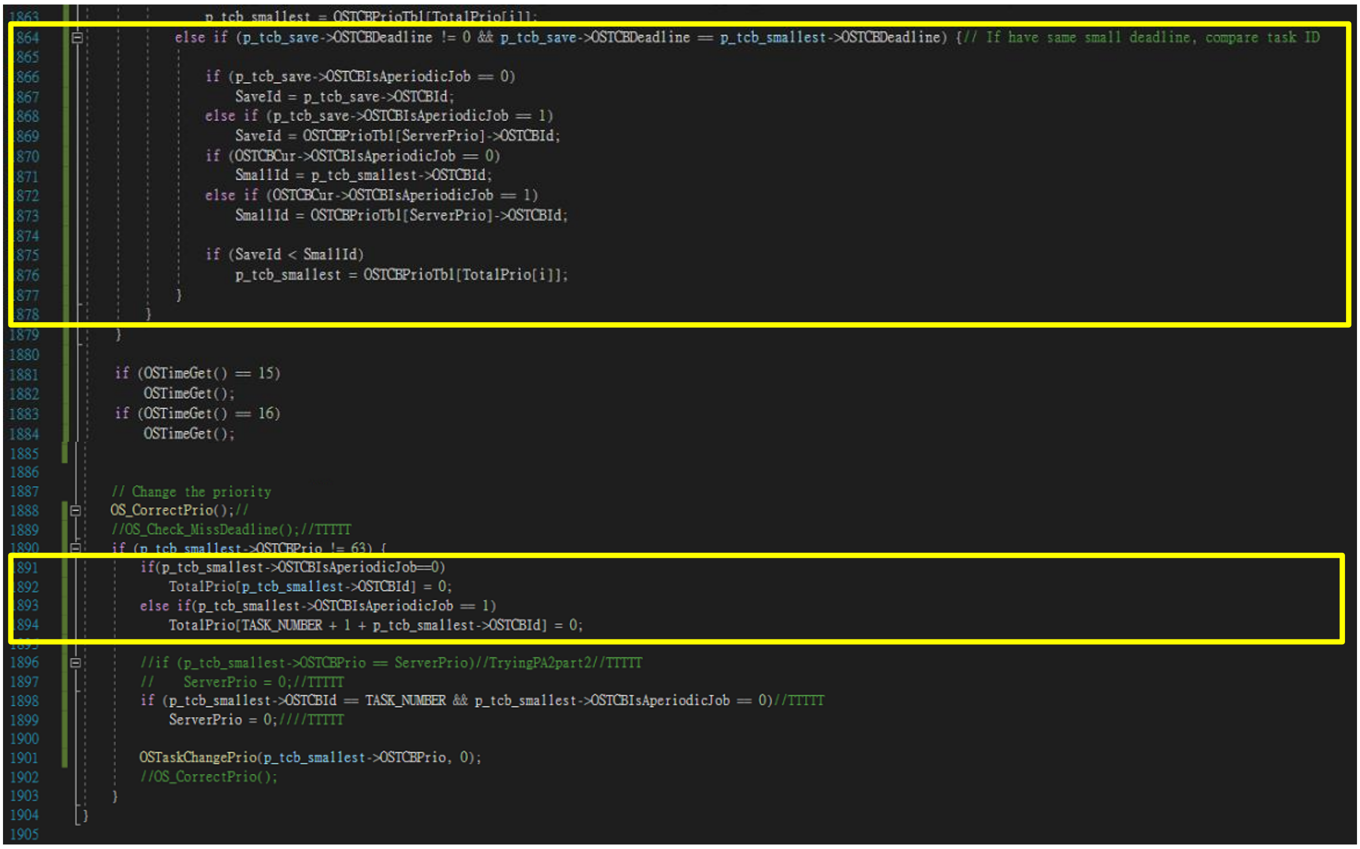
```
void Delete_AperiodicJobs() {
    int i;
    OS_TCB* p_tcb_server;
    OS_TCB* p_tcb_save;

    p_tcb_server = OSTCBPrioTbl[ServerPrio];

    for (i = TASK_NUMBER + 1; i < TaskNum - 1; i++) {
        p_tcb_save = OSTCBPrioTbl[TotalPrio[i]];
        if (p_tcb_save->OSTCBCyclesExecution == p_tcb_save->OSTCBCyclesCount) {
            TotalPrio[i] = 0;   // Clear Priority//AAAAA
            if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0) {
                printf("%2d\tAperiodic job(%2d) is finished.\n", OSTime, p_tcb_save->OSTCBId);
                fprintf(Output_fp, "%2d\tAperiodic job(%2d) is finished.\n", OSTime, p_tcb_save->OSTCBId);
            }
            fclose(Output_fp);
            OSTaskDel(p_tcb_save->OSTCBPrio);
        }
    }
}
```

比照自己在 part1 的模式，因為每一個 tick都會進入 part1 所撰寫的 OS_EDF_Int ，這次副程式的呼叫與修改都在此進行而不會在 OSSched 或是 OSIntExit ，因此需要將 part1 原先所撰寫的進行修改，並增加不少判斷是否為非週期性任務的步驟，避免在任務排程上出錯：

(我的task儲存方式為先儲存週期性任務+Server+非週期性任務)

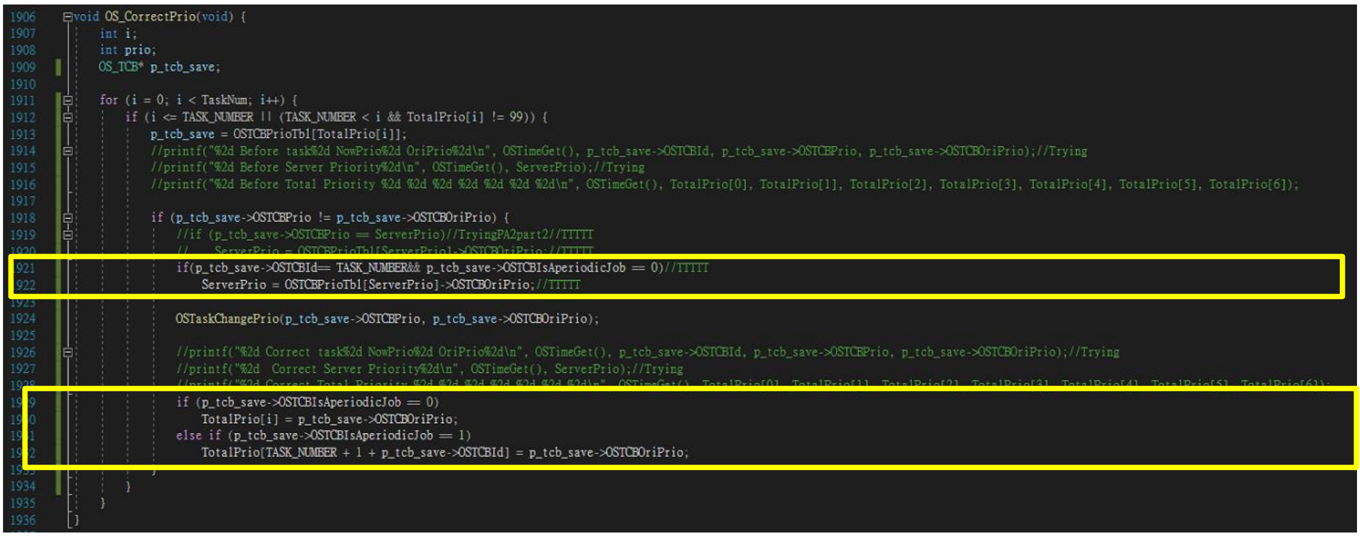


```
void OS_EDF_Int(void)
{
    int i;
    int TotalDeadline[10] = { 999 };
    int SortedDeadline[10] = { 999 };
    OS_TCB* p_tcb_save;
    OS_TCB* p_tcb_smallest = OSTCBCur;

    OS_Check_MissDeadline();    // Check MissDeadline or not//TTTTT
    AperiodicJobs_Deadline_Setting();//TryingPA2part2
    Delete_AperiodicJobs();

    // Get all the Deadlines
    for (i = 0; i < TaskNum; i++) {
        if (i < TASK_NUMBER || (TASK_NUMBER <= i && TotalPrio[i] != 0)) {
            p_tcb_save = OSTCBPrioTbl[TotalPrio[i]];
            if(p_tcb_save->OSTCBIsAperiodicJob == 0)
                TotalDeadline[i] = p_tcb_save->OSTCBCyclesArrive + (p_tcb_save->OSTCBJobNumber + 1) * p_tcb_save->OSTCBCyclesPeriod;
            else if(p_tcb_save->OSTCBIsAperiodicJob == 1)
                TotalDeadline[i] = p_tcb_save->OSTCBDeadline;
            //if (OSTimeGet() == TotalDeadline[i]) {
            if (OSTimeGet() == p_tcb_save->OSTCBCyclesEnd) {
                TotalDeadline[i] = TotalDeadline[i] + p_tcb_save->OSTCBCyclesPeriod;
                p_tcb_save->OSTCBDeadline = TotalDeadline[i];
            }
        }
    }


    // Find the Eariliest Deadline
    // If Eariliest Deadline are same, choose the smallest execution time
    int SaveId;
    int SmallId;
    for (i = 0; i < TaskNum; i++) {
        if (i < TASK_NUMBER || (TASK_NUMBER < i && TotalPrio[i] != 99)) {
            p_tcb_save = OSTCBPrioTbl[TotalPrio[i]];
            if (i == 0)
                p_tcb_smallest = OSTCBCur;
            else if (p_tcb_save->OSTCBDeadline != 0 && p_tcb_save->OSTCBDeadline < p_tcb_smallest->OSTCBDeadline)   // Compare which deadline small
```



```c
                p_tcb_smallest = OSTCBPrioTbl[TotalPrio[i]];
            else if (p_tcb_save->OSTCBDeadline != 0 && p_tcb_save->OSTCBDeadline == p_tcb_smallest->OSTCBDeadline) {// If have same small deadline, compare task ID

                if (p_tcb_save->OSTCBIsAperiodicJob == 0)
                    SaveId = p_tcb_save->OSTCBId;
                else if (p_tcb_save->OSTCBIsAperiodicJob == 1)
                    SaveId = OSTCBPrioTbl[ServerPrio]->OSTCBId;
                if (OSTCBCur->OSTCBIsAperiodicJob == 0)
                    SmallId = p_tcb_smallest->OSTCBId;
                else if (OSTCBCur->OSTCBIsAperiodicJob == 1)
                    SmallId = OSTCBPrioTbl[ServerPrio]->OSTCBId;

                if (SaveId < SmallId)
                    p_tcb_smallest = OSTCBPrioTbl[TotalPrio[i]];
            }
        }
    }

    if (OSTimeGet() == 15)
        OSTimeGet();
    if (OSTimeGet() == 16)
        OSTimeGet();


    // Change the priority
    OS_CorrectPrio();//
    //OS_Check_MissDeadline();//TTTTT
    if (p_tcb_smallest->OSTCBPrio != 63) {
        if(p_tcb_smallest->OSTCBIsAperiodicJob==0)
            TotalPrio[p_tcb_smallest->OSTCBId] = 0;
        else if(p_tcb_smallest->OSTCBIsAperiodicJob == 1)
            TotalPrio[TASK_NUMBER + 1 + p_tcb_smallest->OSTCBId] = 0;

        //if (p_tcb_smallest->OSTCBPrio == ServerPrio)//TryingPA2part2//TTTTT
        //    ServerPrio = 0;//TTTTT
        if (p_tcb_smallest->OSTCBId == TASK_NUMBER && p_tcb_smallest->OSTCBIsAperiodicJob == 0)//TTTTT
            ServerPrio = 0;////TTTTT

        OSTaskChangePrio(p_tcb_smallest->OSTCBPrio, 0);
        //OS_CorrectPrio();
    }
}
```

　　最後同樣的因為多了非週期性任務，因此在 Priority 校正前需要多一步判斷去識別該任務是否為週期性任務，隨著結果不同，Priority 的校正邏輯也不同：



```c
void OS_CorrectPrio(void) {
    int i;
    int prio;
    OS_TCB* p_tcb_save;

    for (i = 0; i < TaskNum; i++) {
        if (i <= TASK_NUMBER || (TASK_NUMBER < i && TotalPrio[i] != 99)) {
            p_tcb_save = OSTCBPrioTbl[TotalPrio[i]];
            //printf("%2d Before task%2d NowPrio%2d OriPrio%2d\n", OSTimeGet(), p_tcb_save->OSTCBId, p_tcb_save->OSTCBPrio, p_tcb_save->OSTCBOriPrio);//Trying
            //printf("%2d Before Server Priority%2d\n", OSTimeGet(), ServerPrio);//Trying
            //printf("%2d Before Total Priority %2d %2d %2d %2d %2d %2d %2d\n", OSTimeGet(), TotalPrio[0], TotalPrio[1], TotalPrio[2], TotalPrio[3], TotalPrio[4], TotalPrio[5], TotalPrio[6]);

            if (p_tcb_save->OSTCBPrio != p_tcb_save->OSTCBOriPrio) {
                //if (p_tcb_save->OSTCBPrio == ServerPrio)//TryingPA2part2//TTTTT
                //    ServerPrio = OSTCBPrioTbl[ServerPrio]->OSTCBOriPrio;//TTTTT
                if(p_tcb_save->OSTCBId== TASK_NUMBER&& p_tcb_save->OSTCBIsAperiodicJob == 0)//TTTTT
                    ServerPrio = OSTCBPrioTbl[ServerPrio]->OSTCBOriPrio;//TTTTT

                OSTaskChangePrio(p_tcb_save->OSTCBPrio, p_tcb_save->OSTCBOriPrio);

                //printf("%2d Correct task%2d NowPrio%2d OriPrio%2d\n", OSTimeGet(), p_tcb_save->OSTCBId, p_tcb_save->OSTCBPrio, p_tcb_save->OSTCBOriPrio);//Trying
                //printf("%2d  Correct Server Priority%2d\n", OSTimeGet(), ServerPrio);//Trying
                //printf("%2d Correct Total Priority %2d %2d %2d %2d %2d %2d %2d\n", OSTimeGet(), TotalPrio[0], TotalPrio[1], TotalPrio[2], TotalPrio[3], TotalPrio[4], TotalPrio[5], TotalPrio[6]);
                if (p_tcb_save->OSTCBIsAperiodicJob == 0)
                    TotalPrio[i] = p_tcb_save->OSTCBOriPrio;
                else if (p_tcb_save->OSTCBIsAperiodicJob == 1)
                    TotalPrio[TASK_NUMBER + 1 + p_tcb_save->OSTCBId] = p_tcb_save->OSTCBOriPrio;
            }
        }
    }
}
```

Credit:

[PART I] EDF Scheduler Implementation [70%]

- The correctness of schedule results of examples. Note the testing task set might not be the same as the given example task set. (20%)
- Implement and describe how to handle the missing deadline situation under EDF. (10%)
- A report that describes your implementation (please attach the screenshot of the code and **MARK** the modified part). (40%)

[PART II] CUS Scheduler Implementation [30%]

- The correctness of schedule results of examples. Note the testing task set might not be the same as the given example task set. (15%)
- A report that describes your implementation (please attach the screenshot of the code and **MARK** the modified part). (15%)

[Bonus I] CUS & Button-triggered Aperiodic Job [10%]

- Implement the CUS scheduling and set button-triggered events as aperiodic jobs. (10%)
    - You can select to implement example 1 or example 2 in Part II.

※ You must modify the source code!

※ Standard input and output filenames in the project are necessary for the checker. Please check the file names before submitting. You must print out the result on the Output.txt file.

```
#define INPUT_FILE_NAME "./TaskSet.txt"
#define OUTPUT_FILE_NAME "./Output.txt"
#define APERIODIC_FILE_NAME "./Aperiodicjobs.txt"
```

※ Please set the system end time as 40 seconds in this project.

```
#define SYSTEM_END_TIME 40
```

※ We will use different task sets to verify your code.

※ You will submit two μC/OS-II projects for PART I and PART II, respectively.

Project submit:

Submit to Moodle.

Submit deadline: Nov 23, 2022 (Wednesday) 12:00

File name format: RTOS_Myyyddxxx_PA2.zip

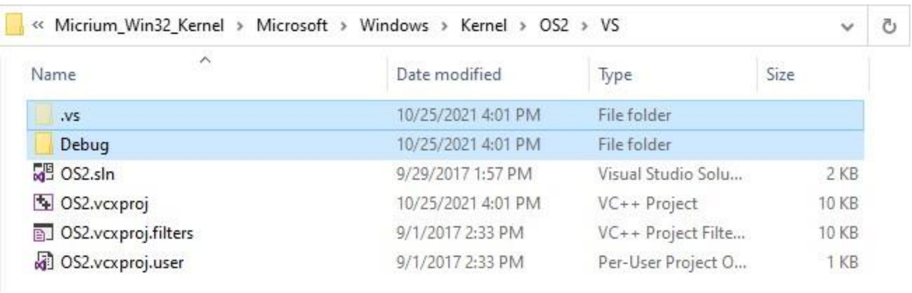
RTOS_Myyyddxxx_PA2.zip includes:

- The report (RTOS_Myyyddxxx_PA2.pdf).
- Folder with the executable µC/OS-II project (**RTOS_Myyyddxxx_PA2_EDF**).
- Folder with the executable µC/OS-II project (**RTOS_Myyyddxxx_PA2_CUS**).

※ Plagiarizing is strictly prohibited.

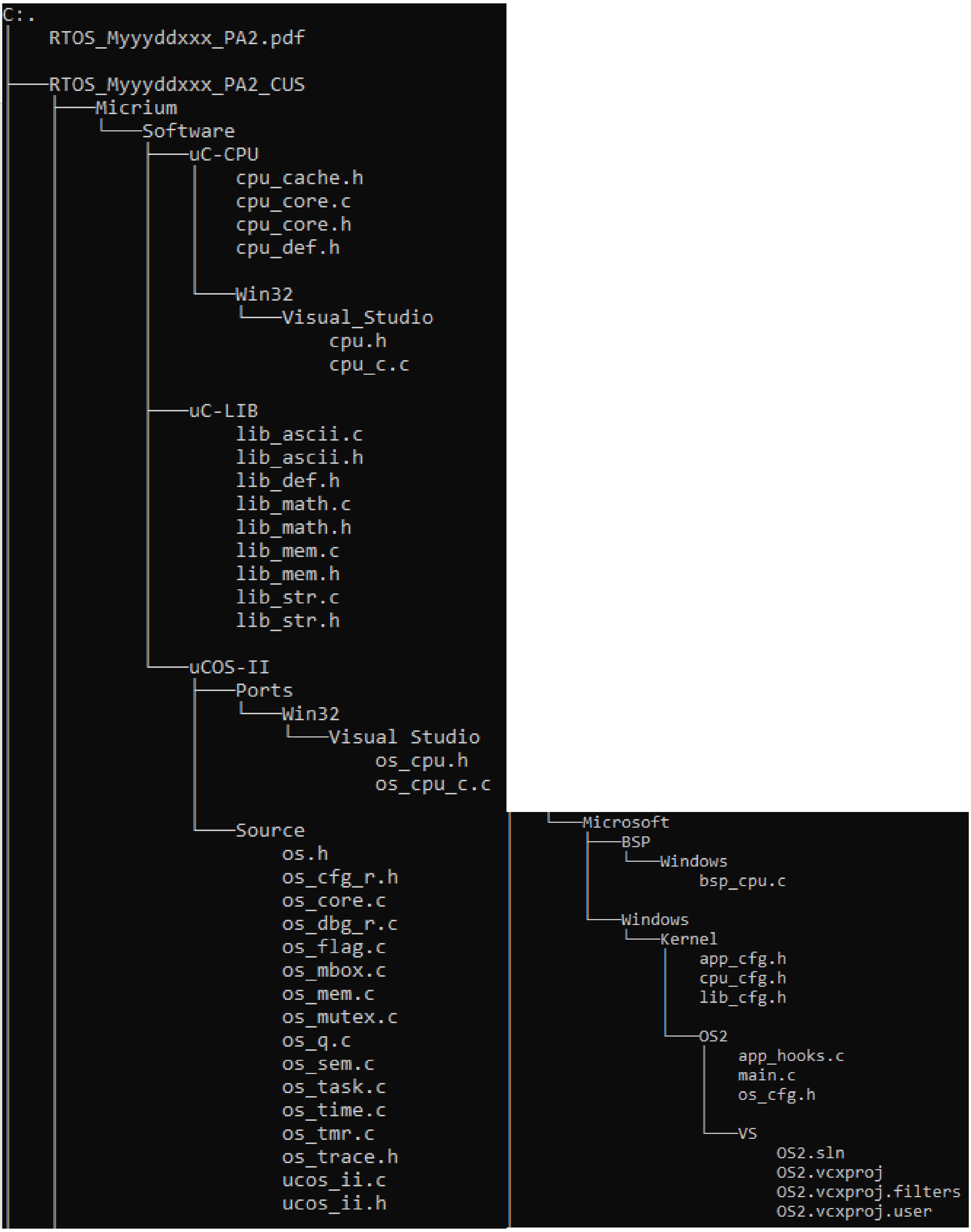
Hints:

1. Please delete the “.vs” and “Debug” folders.


« Micrium_Win32_Kernel › Microsoft › Windows › Kernel › OS2 › VS

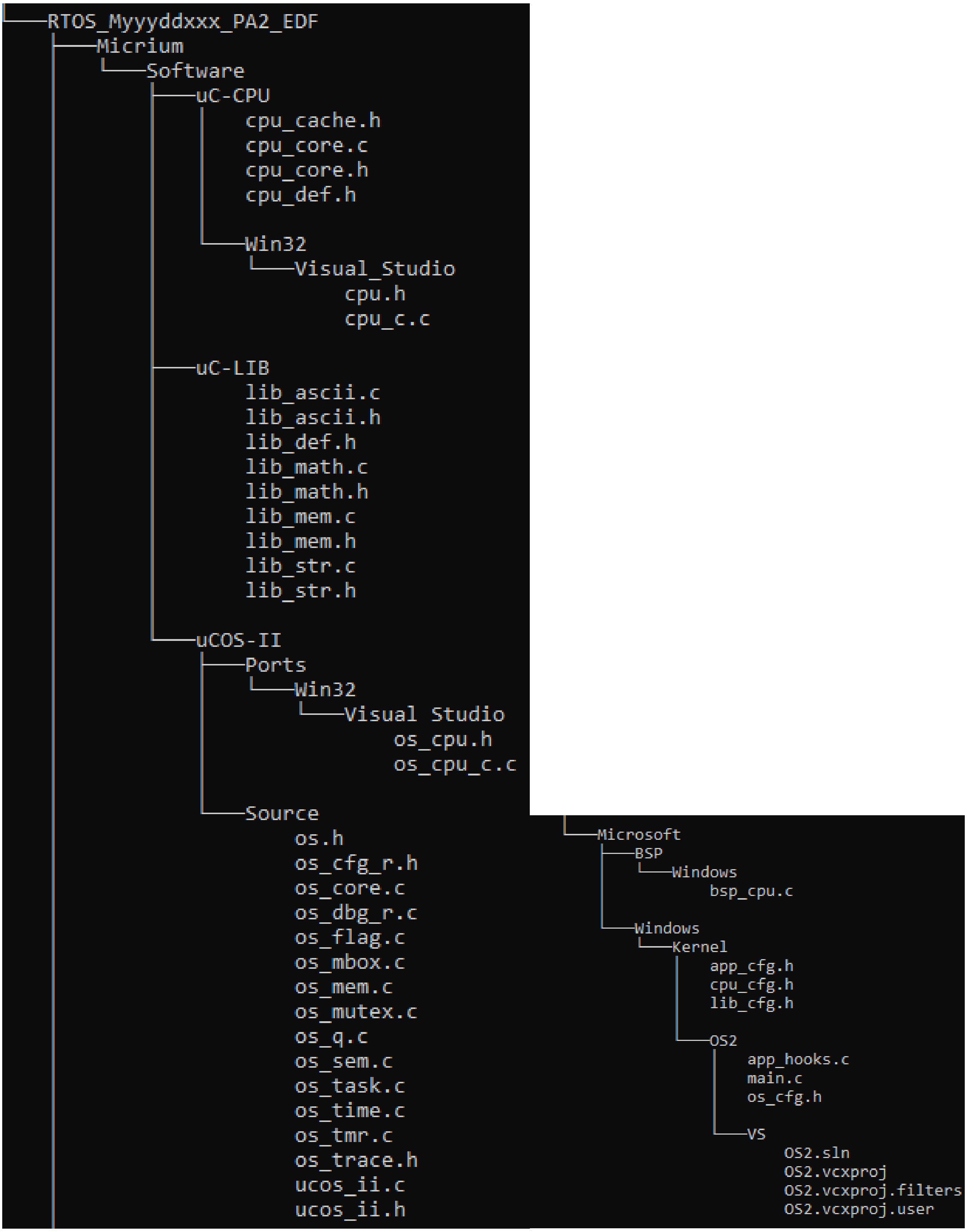
Name	Date modified	Type	Size
.vs	10/25/2021 4:01 PM	File folder	
Debug	10/25/2021 4:01 PM	File folder	
OS2.sln	9/29/2017 1:57 PM	Visual Studio Solu...	2 KB
OS2.vcxproj	10/25/2021 4:01 PM	VC++ Project	10 KB
OS2.vcxproj.filters	9/1/2017 2:33 PM	VC++ Project Filte...	10 KB
OS2.vcxproj.user	9/1/2017 2:33 PM	Per-User Project O...	1 KB

2. RTOS_Myyyddxxx_PA2.zip must be including files as follow:



```
C:.
│   RTOS_Myyyddxxx_PA2.pdf
│
├───RTOS_Myyyddxxx_PA2_CUS
│   ├───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
```



```
└───RTOS_Myyyddxxx_PA2_EDF
    ├───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
```