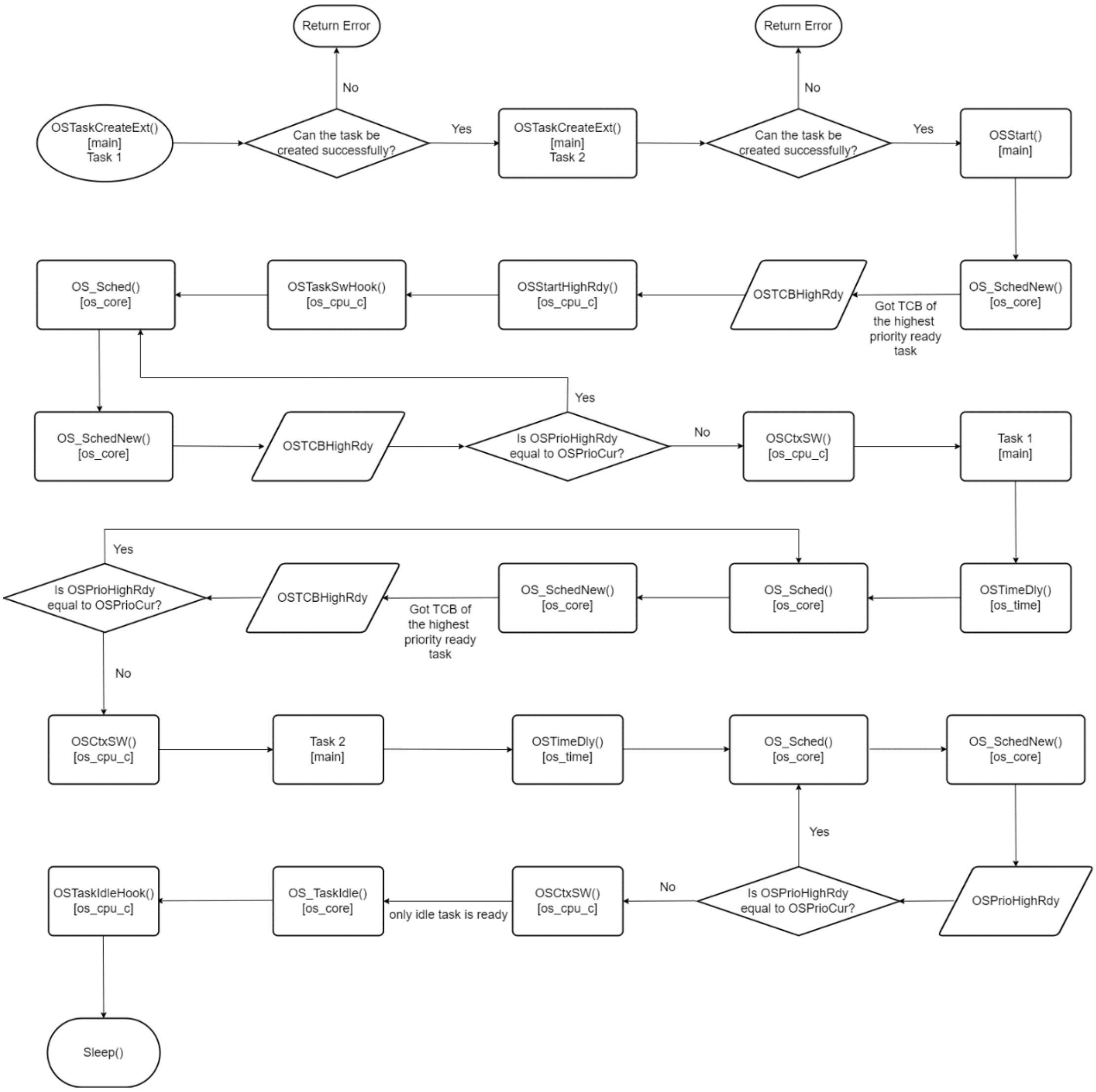
Embedded OS Implementation HW1

M11202117 呂長恩

1. System Flow

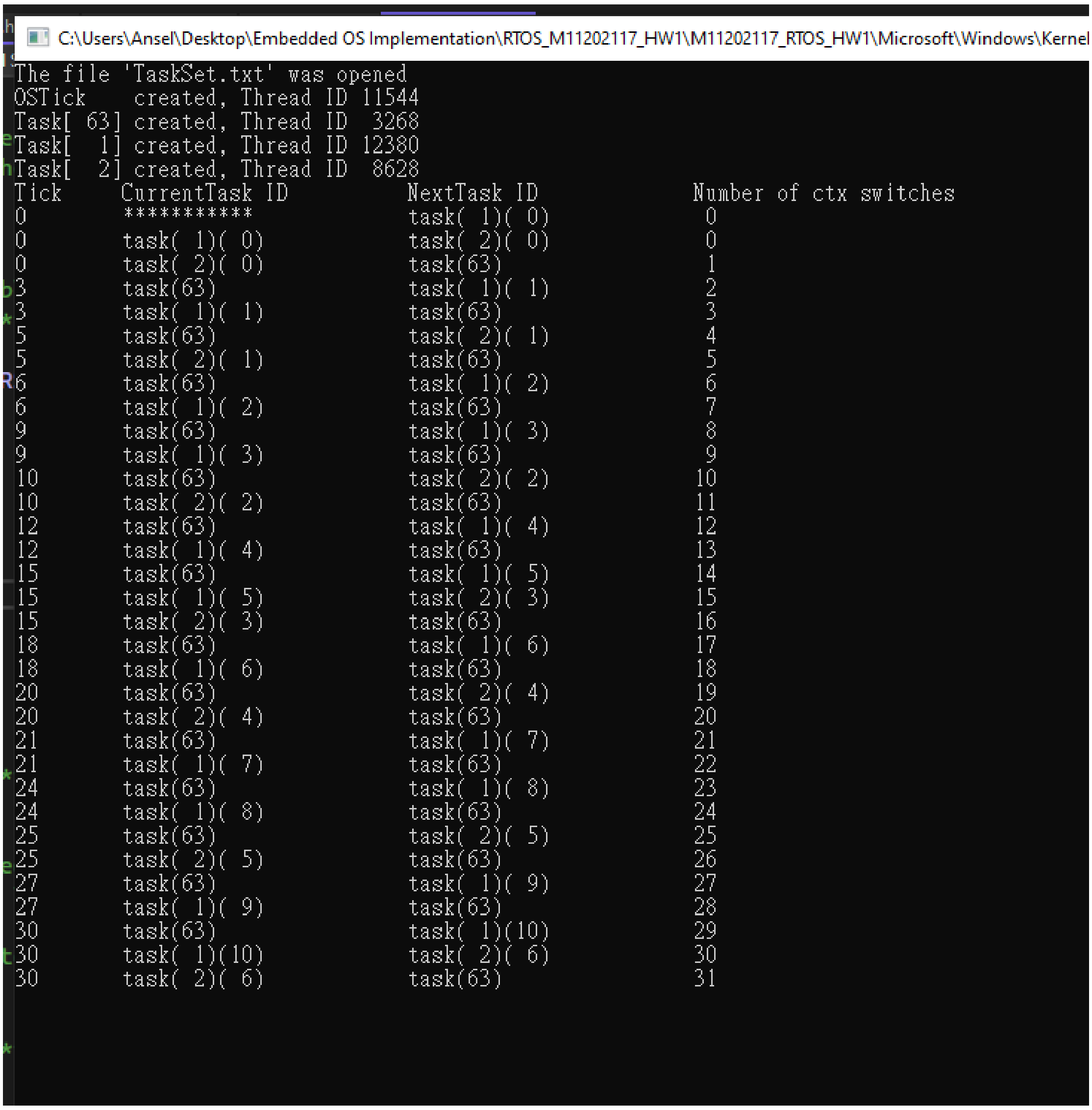


- **OSTaskCreateExt ():** This function is used to have user create tasks and have uC/OS-II manage the execution of a task. Tasks can either be created by user prior to the start of multitasking or by a running task.
- **OSStart ():** This function is used to start the multitasking process which let uC/OS-II manages the task that we just have created.
- **OS_SchedNew ():** The function is used to determine the highest priority task that is ready to run. **OSTCBHighRdy** is a pointer to the next highest-priority TCB that is ready to run(R-to-R). **OSPrioHighRdy** is a priority value of the next highest-priority task that is ready to

run.

- **OSStartHighRdy ():** This function is called by OSStart() to start the highest priority task that was created by my application before calling OSStart().
- **OSTaskSWHook ():** This allows users to perform other operations during a context switch.
- **OS_Sched ():** This function is called by other uC/OS-II services to determine whether a new, high priority task has been made ready to run.
- **OSCtxSW ():** This function is called when a task makes a higher priority task ready-to-run. It must save the current processor registers and current task's stack pointer into the current task's OS_TCB, then switch to the highest priority task.
- **OSTimeDly ():** This function is called to delay execution of the currently running task until the specified number of system ticks expires.
- **OS_TaskIdle ():** This task is internal to uC/OS-II and executes whenever no other higher priority tasks executes because they are ALL waiting for event(s) to occur. The priority of idle task is 63 (the lowest priority).
- **OSTaskIdleHook ():** This hook has been added to allow user to do such things as STOP the CPU to conserve power.
- **Sleep ():** Reduce CPU utilization.

2. Screenshot of Result

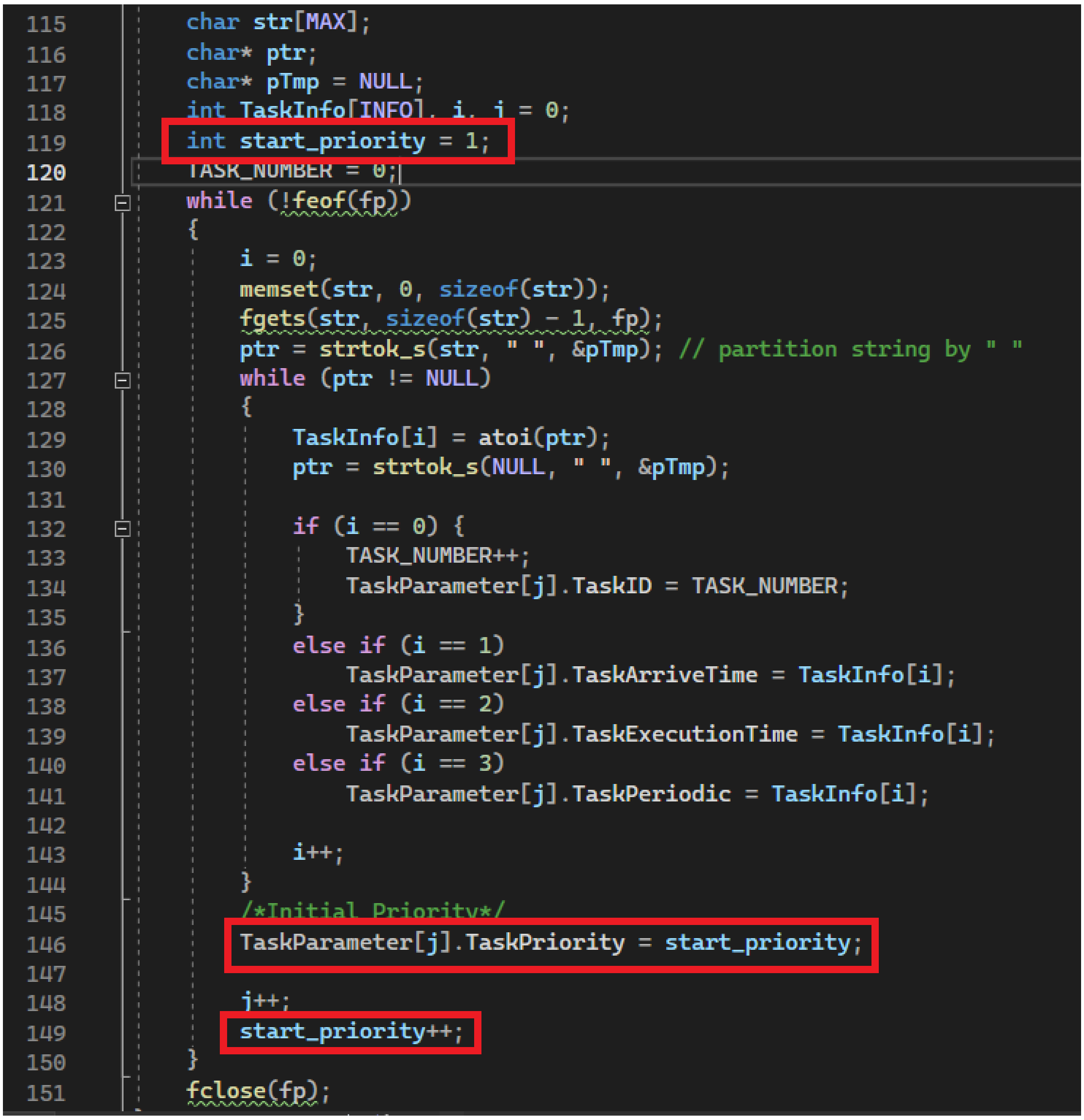


C:\Users\Ansel\Desktop\Embedded OS Implementation\RTOS_M11202117_HW1\M11202117_RTOS_HW1\Microsoft\Windows\Kernel

```
The file 'TaskSet.txt' was opened
OSTick    created, Thread ID 11544
Task[ 63] created, Thread ID  3268
Task[  1] created, Thread ID 12380
Task[  2] created, Thread ID  8628
Tick     CurrentTask ID          NextTask ID             Number of ctx switches
0        ***********             task( 1)( 0)             0
0        task( 1)( 0)            task( 2)( 0)             0
0        task( 2)( 0)            task(63)                 1
3        task(63)                task( 1)( 1)             2
3        task( 1)( 1)            task(63)                 3
5        task(63)                task( 2)( 1)             4
5        task( 2)( 1)            task(63)                 5
6        task(63)                task( 1)( 2)             6
6        task( 1)( 2)            task(63)                 7
9        task(63)                task( 1)( 3)             8
9        task( 1)( 3)            task(63)                 9
10       task(63)                task( 2)( 2)             10
10       task( 2)( 2)            task(63)                 11
12       task(63)                task( 1)( 4)             12
12       task( 1)( 4)            task(63)                 13
15       task(63)                task( 1)( 5)             14
15       task( 1)( 5)            task( 2)( 3)             15
15       task( 2)( 3)            task(63)                 16
18       task(63)                task( 1)( 6)             17
18       task( 1)( 6)            task(63)                 18
20       task(63)                task( 2)( 4)             19
20       task( 2)( 4)            task(63)                 20
21       task(63)                task( 1)( 7)             21
21       task( 1)( 7)            task(63)                 22
24       task(63)                task( 1)( 8)             23
24       task( 1)( 8)            task(63)                 24
25       task(63)                task( 2)( 5)             25
25       task( 2)( 5)            task(63)                 26
27       task(63)                task( 1)( 9)             27
27       task( 1)( 9)            task(63)                 28
30       task(63)                task( 1)(10)             29
30       task( 1)(10)            task( 2)( 6)             30
30       task( 2)( 6)            task(63)                 31
```

3. Implementation

1. InputFile ()_[app_hook.c]



```c
char str[MAX];
char* ptr;
char* pTmp = NULL;
int TaskInfo[INFO], i, j = 0;
int start_priority = 1;
TASK_NUMBER = 0;
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)
    {
        TaskInfo[i] = atoi(ptr);
        ptr = strtok_s(NULL, " ", &pTmp);

        if (i == 0) {
            TASK_NUMBER++;
            TaskParameter[j].TaskID = TASK_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];

        i++;
    }
    /*Initial Priority*/
    TaskParameter[j].TaskPriority = start_priority;

    j++;
    start_priority++;
}
fclose(fp);
```

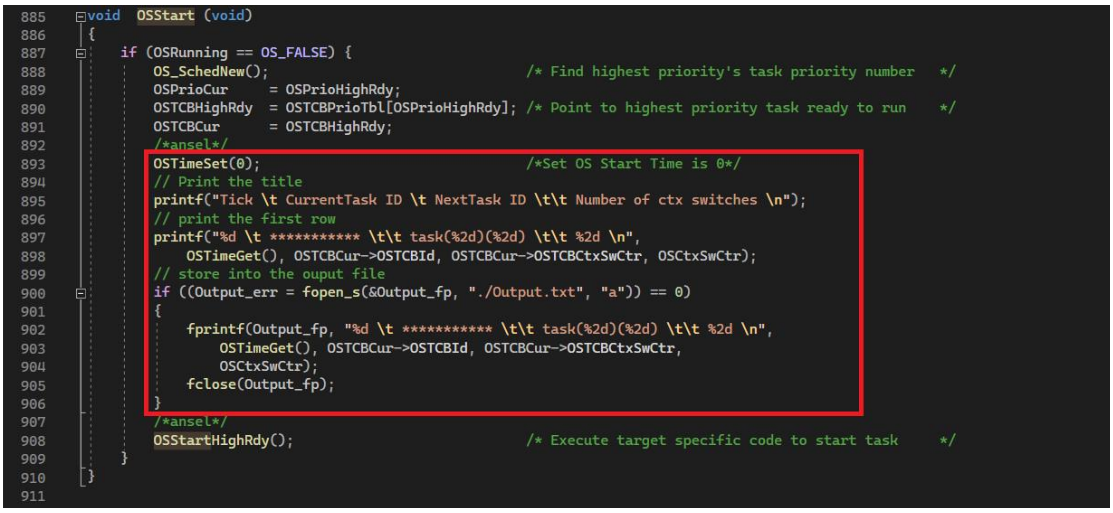
I. I have declared a local variable named “start_priority” and set it start from 1.

II. When a task is created, its “TaskPriority” should be set to “start_prioirty”.

III. After setting, “start_priority” will increment by 1.

IV. Therefore, it meets the requirement of having two tasks with priorities of 1 and 2, respectively.

2. OSStart ()_[os_core.c]



```c
void  OSStart (void)
{
    if (OSRunning == OS_FALSE) {
        OS_SchedNew();                                   /* Find highest priority's task priority number   */
        OSPrioCur     = OSPrioHighRdy;
        OSTCBHighRdy  = OSTCBPrioTbl[OSPrioHighRdy]; /* Point to highest priority task ready to run    */
        OSTCBCur      = OSTCBHighRdy;
        /*ansel*/
        OSTimeSet(0);                                    /*Set OS Start Time is 0*/
        // Print the title
        printf("Tick \t CurrentTask ID \t NextTask ID \t\t Number of ctx switches \n");
        // print the first row
        printf("%d \t *********** \t\t task(%2d)(%2d) \t\t %2d \n",
            OSTimeGet(), OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr, OSCtxSwCtr);
        // store into the ouput file
        if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0)
        {
            fprintf(Output_fp, "%d \t *********** \t\t task(%2d)(%2d) \t\t %2d \n",
                OSTimeGet(), OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr,
                OSCtxSwCtr);
            fclose(Output_fp);
        }
        /*ansel*/
        OSStartHighRdy();                                /* Execute target specific code to start task     */
    }
}
```

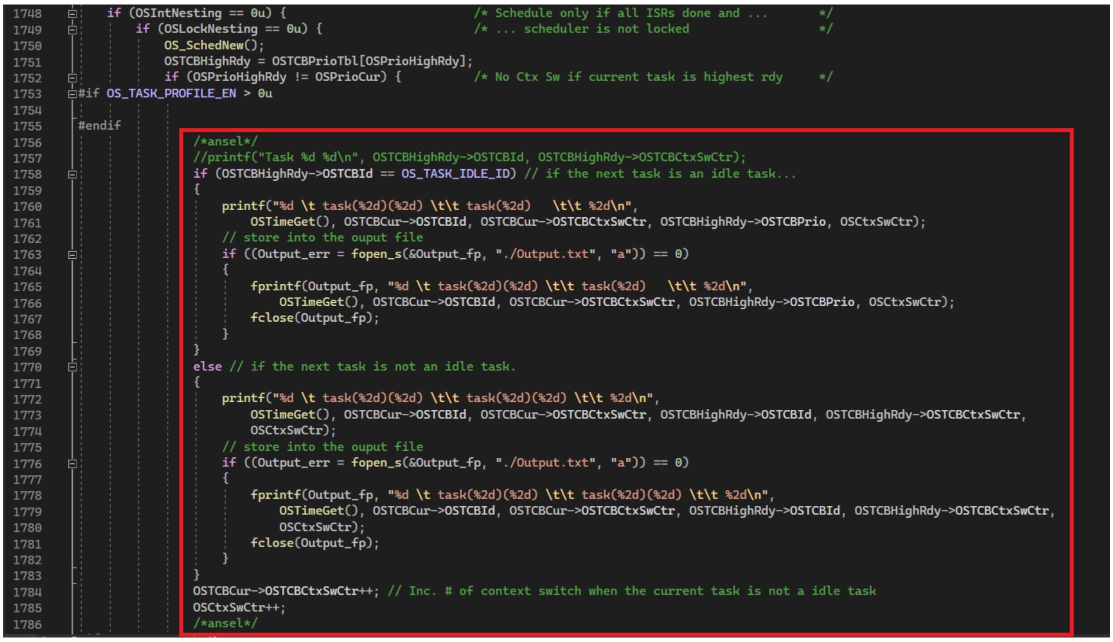
I. I set OS start time 0 by calling OSTimeSet(0) function.

II. I print the title of the data that I need to get.

III. I print the first data using “OSTCBCur” to get the current running task’s ID and context switch counter’s value in its TCB stack.

IV. Print out these data into output file by using fprintf() function.

3. OS_Sched ()_[os_core.c]



```
    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

#endif
                /*ansel*/
                //printf("Task %d %d\n", OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBCtxSwCtr);
                if (OSTCBHighRdy->OSTCBId == OS_TASK_IDLE_ID) // if the next task is an idle task...
                {
                    printf("%d \t task(%2d)(%2d) \t\t task(%2d)   \t\t %2d\n",
                        OSTimeGet(), OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr, OSTCBHighRdy->OSTCBPrio, OSCtxSwCtr);
                    // store into the ouput file
                    if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0)
                    {
                        fprintf(Output_fp, "%d \t task(%2d)(%2d) \t\t task(%2d)   \t\t %2d\n",
                            OSTimeGet(), OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr, OSTCBHighRdy->OSTCBPrio, OSCtxSwCtr);
                        fclose(Output_fp);
                    }
                }
                else // if the next task is not an idle task.
                {
                    printf("%d \t task(%2d)(%2d) \t\t task(%2d)(%2d) \t\t %2d\n",
                        OSTimeGet(), OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBCtxSwCtr,
                        OSCtxSwCtr);
                    // store into the ouput file
                    if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0)
                    {
                        fprintf(Output_fp, "%d \t task(%2d)(%2d) \t\t task(%2d)(%2d) \t\t %2d\n",
                            OSTimeGet(), OSTCBCur->OSTCBId, OSTCBCur->OSTCBCtxSwCtr, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBCtxSwCtr,
                            OSCtxSwCtr);
                        fclose(Output_fp);
                    }
                }
                OSTCBCur->OSTCBCtxSwCtr++; // Inc. # of context switch when the current task is not a idle task
                OSCtxSwCtr++;
                /*ansel*/
```

I. When scheduler is not locked, it will do OS_SchedNew() to determine which task has the highest priority and it is ready to run.

II. For instance, when task1 is ready to run and it has the highest priority at that time, scheduler should pick task1 as the running task to be executed.

III. Consequently, we can print the current task ID and counter’s value using “OSTCBCur” TCB.

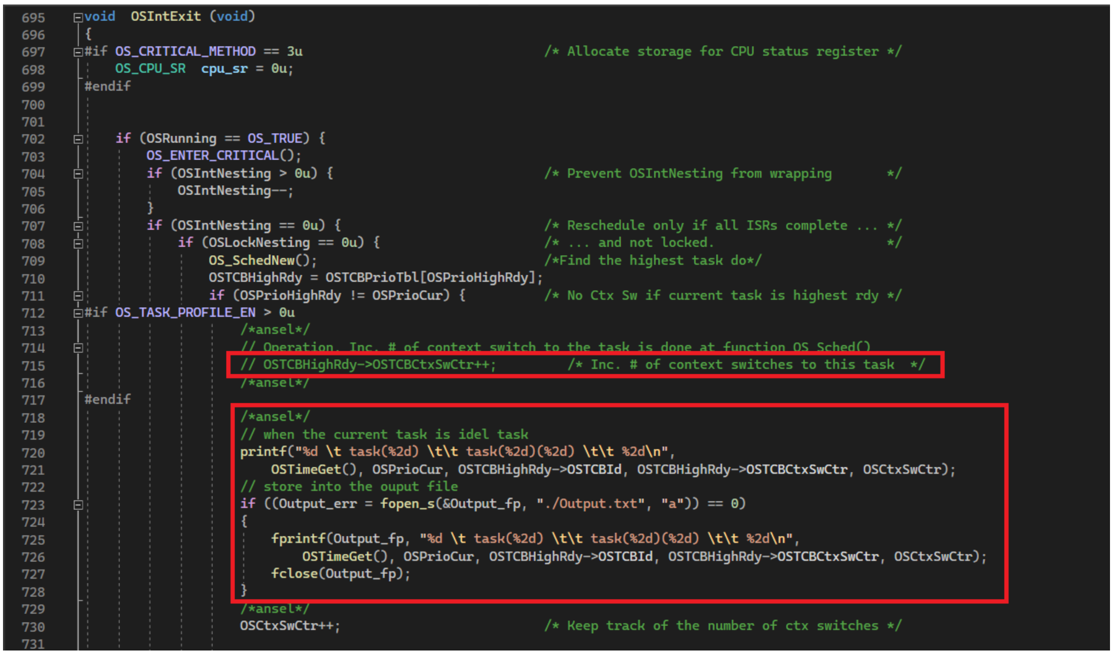
IV. After uC/OS-II executes the OS_SchedNew() function to determine the next highest-priority task, we can access task1's information through the "OSTCBHighRdy" TCB.

V. uC/OS-II should increment context switch counter’s value by 1 whenever a task switch occurs, which can be accomplished with the statement “OSCtxSwCtr++”.

VI. uC/OS-II should increment context switch counter’s value within the TCB by 1 whenever its associated task undergoes a context switch.

VII. If the “OSTCBHighRdy” is the idle task, uC/OS-II needs to print the information according to the requirements. Therefore, I use an “if-else” statement to determine how to the data.

4. OSIntExit ()_[os_core.c]



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


    if (OSRunning == OS_TRUE) {
        OS_ENTER_CRITICAL();
        if (OSIntNesting > 0u) {                           /* Prevent OSIntNesting from wrapping       */
            OSIntNesting--;
        }
        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*/
                    // Operation. Inc. # of context switch to the task is done at function OS_Sched()
                    // OSTCBHighRdy->OSTCBCtxSwCtr++;         /* Inc. # of context switches to this task  */
                    /*ansel*/
#endif
                    /*ansel*/
                    // when the current task is idel task
                    printf("%d \t task(%2d) \t\t task(%2d)(%2d) \t\t %2d\n",
                        OSTimeGet(), OSPrioCur, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBCtxSwCtr, OSCtxSwCtr);
                    // store into the ouput file
                    if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0)
                    {
                        fprintf(Output_fp, "%d \t task(%2d) \t\t task(%2d)(%2d) \t\t %2d\n",
                            OSTimeGet(), OSPrioCur, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->OSTCBCtxSwCtr, OSCtxSwCtr);
                        fclose(Output_fp);
                    }
                    /*ansel*/
                    OSCtxSwCtr++;                          /* Keep track of the number of ctx switches */
```

I. This function is used to notify uC/OS-II that it has completed an ISR.

II. When a task switches from the idle task to task1 or task2, it signifies that the ISR (Interrupt Service Routine) for the idle task has completed its service. This function will be called.

III. Therefore, at this point, “OS_Sched()” will be retriggered to find the highest-priority task that is R-to-R. Consequently, we can observe that the current task will output "idle task," and the next task will output "task1" or "task2" here.

IV. Furthermore, because the switch from the idle task to task1 or task2 occurs at this moment, it can be observed that the idle task and task1 or task2 will output simultaneously, sharing the same tick time.

V. Whenever uC/OS-II switches tasks, “OSCtxSwCtr” value should be incremented by 1.

VI. I will comment out "OSTCBHighRdy->OSTCBCtxSwCtr++" because it should be incremented when OS_Sched() switches to task1 or task2, rather than immediately upon leaving the ISR. Otherwise, it may result in incorrect output.