Embedded OS Implementation, Fall 2025

Project #1 (due at November 3rd, 2025 (Monday) 08:00)

[PART I] Task Control Block Linked List

Objective:

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

※ The TCB address is dynamic.

※ This part will be included in the subsequent output and does not require separate code submission.

The example output results are shown below:

```
OSTick    created, Thread ID 11760
Task[ 63] created, TCB Address  92e9e0
------After TCB[63] begin linked------
Previous TCB point to adress         0
Current  TCB point to adress    92e9e0
Next     TCB point to adress         0

The file 'TaskSet.txt' was opened
Task[  1] created, TCB Address  92ea54
------After TCB[ 1] begin linked------
Previous TCB point to adress         0
Current  TCB point to adress    92ea54
Next     TCB point to adress    92e9e0

Task[  2] created, TCB Address  92eac8
------After TCB[ 2] begin linked------
Previous TCB point to adress         0
Current  TCB point to adress    92eac8
Next     TCB point to adress    92ea54

================TCB linked list================
Task    Prev_TCB_addr   TCB_addr   Next_TCB_addr
 2                0       92eac8        92ea54
 1           92eac8       92ea54        92e9e0
63           92ea54       92e9e0             0
```

[PART II] RM Scheduler Implementation

Objective:

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

Problem Definition:

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

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

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

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

Example Task Set 3 = {τ_1 (1, 0, 3, 8), τ_2 (2, 1, 2, 6), τ_3 (3, 0, 4, 15)}

※ If tasks have same period, the task with the smaller TaskID will be executed first.

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

The input file format:

Task ID	Arrive Time	Execution Time	Task Period
##	##	##	##

Example of task set file 1:

```
1 0 1 3
2 0 3 5
```

Evaluation:

The output format:

Tick	Event	CurrentTask ID	NextTask ID	Response Time	Preemption Time	OSTimeDly
##	Preemption	task(ID)(job number)	task(ID)(job number)			
##	Completion	task(ID)(job number)	task(ID)(job number)	##	##	##
##	MissDeadline	task(ID)(job number)	----------			
##	task(ID) is running					

※ If the task is Idle Task, print "*task(priority)*".

Response Time: the duration between the task's arrival time and the time it is completed.

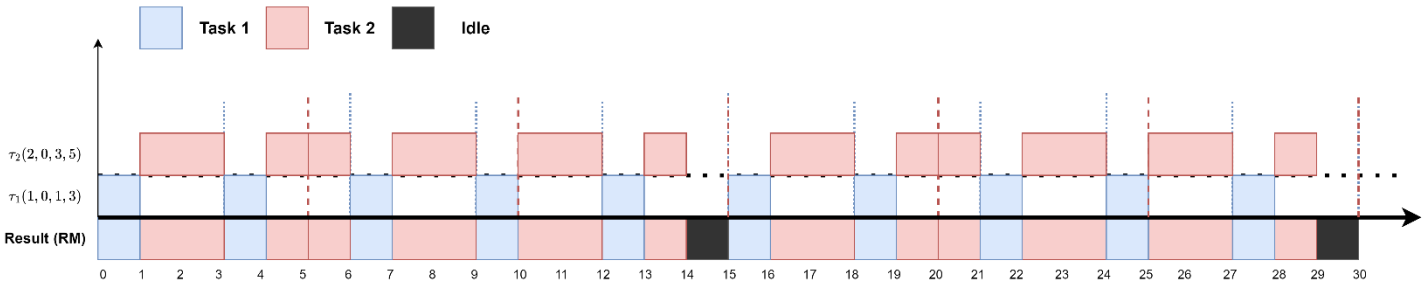
Preemption Time: the time this task is preempted by higher-priority tasks.

OSTimeDly: the remaining delay time for this task

※ You have to print "task(ID) is running" message in task() function.

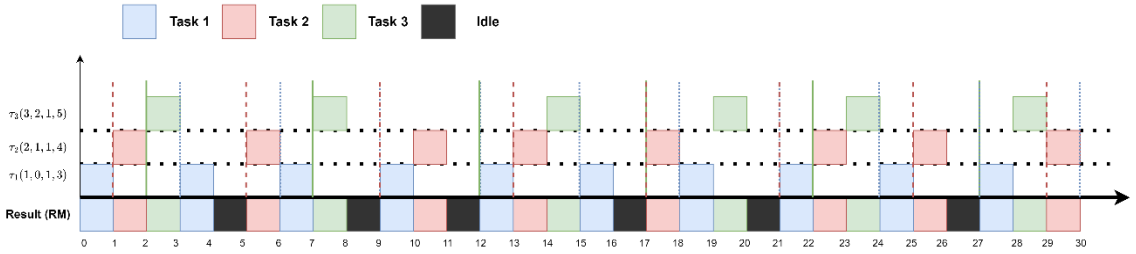
※ Output priority: MissDeadline > Preemption > Completion > task(ID) is running .

The scheduled results of Task Set 1 = {τ_1 (1, 0, 1, 3), τ_2 (2, 0, 3, 5)}:



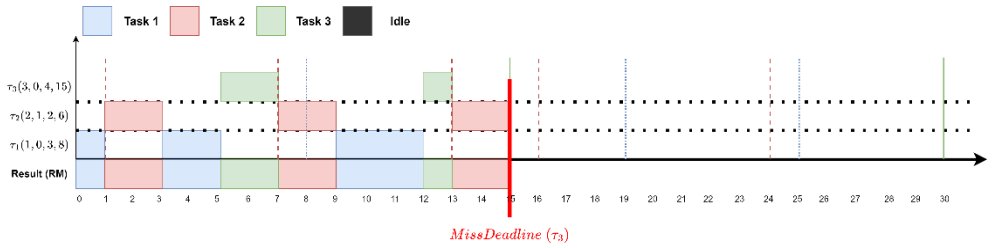

```
 0    task (  1) is running
 1    Completion    task(  1)(  0)    task(  2)(  0)        1             0             2
 1    task (  2) is running
 2    task (  2) is running
 3    Preemption    task(  2)(  0)    task(  1)(  1)
 3    task (  1) is running
 4    Completion    task(  1)(  1)    task(  2)(  0)        1             0             2
 4    task (  2) is running
 5    Completion    task(  2)(  0)    task(  2)(  1)        5             2             0
 5    task (  2) is running
 6    Preemption    task(  2)(  1)    task(  1)(  2)
 6    task (  1) is running
 7    Completion    task(  1)(  2)    task(  2)(  1)        1             0             2
 7    task (  2) is running
 8    task (  2) is running
 9    Completion    task(  2)(  1)    task(  1)(  3)        4             1             1
 9    task (  1) is running
10    Completion    task(  1)(  3)    task(  2)(  2)        1             0             2
10    task (  2) is running
11    task (  2) is running
12    Preemption    task(  2)(  2)    task(  1)(  4)
12    task (  1) is running
13    Completion    task(  1)(  4)    task(  2)(  2)        1             0             2
13    task (  2) is running
14    Completion    task(  2)(  2)    task( 63)             4             1             1
15    Preemption    task( 63)         task(  1)(  5)
15    task (  1) is running
```

The output results of Task Set 2 = {τ_1 (1, 0, 1, 3), τ_2 (2, 1, 1, 4), τ_3 (3, 2, 1, 5) }:




```
 0    task (  1) is running
 1    Completion    task(  1)(  0)    task(  2)(  0)        1              0              2
 1    task (  2) is running
 2    Completion    task(  2)(  0)    task(  3)(  0)        1              0              3
 2    task (  3) is running
 3    Completion    task(  3)(  0)    task(  1)(  1)        1              0              4
 3    task (  1) is running
 4    Completion    task(  1)(  1)    task( 63)             1              0              2
 5    Preemption    task( 63)         task(  2)(  1)
 5    task (  2) is running
 6    Completion    task(  2)(  1)    task(  1)(  2)        1              0              3
 6    task (  1) is running
 7    Completion    task(  1)(  2)    task(  3)(  1)        1              0              2
 7    task (  3) is running
 8    Completion    task(  3)(  1)    task( 63)             1              0              4
 9    Preemption    task( 63)         task(  1)(  3)
 9    task (  1) is running
10    Completion    task(  1)(  3)    task(  2)(  2)        1              0              2
10    task (  2) is running
11    Completion    task(  2)(  2)    task( 63)             2              1              2
12    Preemption    task( 63)         task(  1)(  4)
12    task (  1) is running
13    Completion    task(  1)(  4)    task(  2)(  3)        1              0              2
13    task (  2) is running
14    Completion    task(  2)(  3)    task(  3)(  2)        1              0              3
14    task (  3) is running
15    Completion    task(  3)(  2)    task(  1)(  5)        3              2              2
15    task (  1) is running
```

The output results of Task Set 3 = {τ_1 (1, 0, 3, 8), τ_2 (2, 1, 2, 6), τ_3 (3, 0, 4, 15)}:




```
 0    task (  1) is running
 1    Preemption    task(  1)(  0)    task(  2)(  0)
 1    task (  2) is running
 2    task (  2) is running
 3    Completion    task(  2)(  0)    task(  1)(  0)        2             0             4
 3    task (  1) is running
 4    task (  1) is running
 5    Completion    task(  1)(  0)    task(  3)(  0)        5             2             3
 5    task (  3) is running
 6    task (  3) is running
 7    Preemption    task(  3)(  0)    task(  2)(  1)
 7    task (  2) is running
 8    task (  2) is running
 9    Completion    task(  2)(  1)    task(  1)(  1)        2             0             4
 9    task (  1) is running
10    task (  1) is running
11    task (  1) is running
12    Completion    task(  1)(  1)    task(  3)(  0)        4             1             4
12    task (  3) is running
13    Preemption    task(  3)(  0)    task(  2)(  2)
13    task (  2) is running
14    task (  2) is running
15    MissDeadline  task(  3)(  0)    ------------------
```

[Part III] FIFO Scheduler Implementation

Objective:

To implement the non-preemptive First In First Out (FIFO) scheduling for periodic tasks, and handle the miss deadline behaviors.

Problem Definition:

Implement the following task set of periodic tasks. 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)}

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

Task Set 2 = {τ_1 (1, 0, 1, 3), τ_2 (2, 1, 2, 7), τ_3 (3, 4, 3, 12)}

※If tasks arrive simultaneously, the task with the smaller TaskID will be executed first.

Evaluation:

The output format:

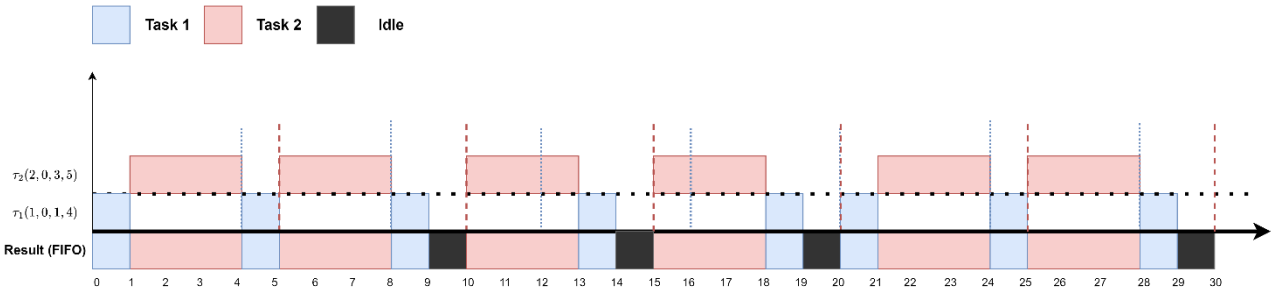
Tick	Event	CurrentTask ID	NextTask ID	Response Time	Preemption Time	OSTimeDly
##	Preemption	task(ID)(job number)	task(ID)(job number)			
##	Completion	task(ID)(job number)	task(ID)(job number)	##	##	##
##	MissDeadline	task(ID)(job number)	----------			
##	task(ID) is running					

※ If the task is Idle Task, print "*task(priority)*".

※ **You have to print "task(ID) is running" message in task() function.**

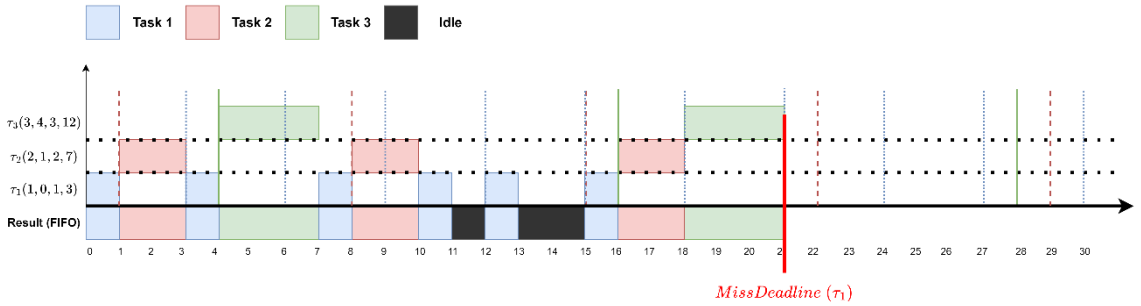
※ **Output priority: MissDeadline > Preemption > Completion > task(ID) is running .**

The output results of Task Set 1 = {τ_1 (1, 0, 1, 4), τ_2 (2, 0, 3, 5)}:




```
 0    task (  1) is running
 1    Completion    task(  1)(  0)    task(  2)(  0)       1            0            3
 1    task (  2) is running
 2    task (  2) is running
 3    task (  2) is running
 4    Completion    task(  2)(  0)    task(  1)(  1)       4            1            1
 4    task (  1) is running
 5    Completion    task(  1)(  1)    task(  2)(  1)       1            0            3
 5    task (  2) is running
 6    task (  2) is running
 7    task (  2) is running
 8    Completion    task(  2)(  1)    task(  1)(  2)       3            0            2
 8    task (  1) is running
 9    Completion    task(  1)(  2)    task( 63)            1            0            3
10    Preemption    task( 63)         task(  2)(  2)
10    task (  2) is running
11    task (  2) is running
12    task (  2) is running
13    Completion    task(  2)(  2)    task(  1)(  3)       3            0            2
13    task (  1) is running
14    Completion    task(  1)(  3)    task( 63)            2            1            2
15    Preemption    task( 63)         task(  2)(  3)
15    task (  2) is running
```

The output results of Task Set 2 = {τ_1 (1, 0, 1, 3), τ_2 (2, 1, 2, 7), τ_3 (3, 4, 3, 12)} :




```
 0    task (  1) is running
 1    Completion    task(  1)(  0)    task(  2)(  0)        1             0             2
 1    task (  2) is running
 2    task (  2) is running
 3    Completion    task(  2)(  0)    task(  1)(  1)        2             0             5
 3    task (  1) is running
 4    Completion    task(  1)(  1)    task(  3)(  0)        1             0             2
 4    task (  3) is running
 5    task (  3) is running
 6    task (  3) is running
 7    Completion    task(  3)(  0)    task(  1)(  2)        3             0             9
 7    task (  1) is running
 8    Completion    task(  1)(  2)    task(  2)(  1)        2             1             1
 8    task (  2) is running
 9    task (  2) is running
10    Completion    task(  2)(  1)    task(  1)(  3)        2             0             5
10    task (  1) is running
11    Completion    task(  1)(  3)    task( 63)             2             1             1
12    Preemption    task( 63)         task(  1)(  4)
12    task (  1) is running
13    Completion    task(  1)(  4)    task( 63)             1             0             2
15    Preemption    task( 63)         task(  1)(  5)
15    task (  1) is running
16    Completion    task(  1)(  5)    task(  2)(  2)        1             0             2
16    task (  2) is running
17    task (  2) is running
18    Completion    task(  2)(  2)    task(  3)(  1)        3             1             4
18    task (  3) is running
19    task (  3) is running
20    task (  3) is running
21    MissDeadline  task(  1)(  6)    ------------------
```

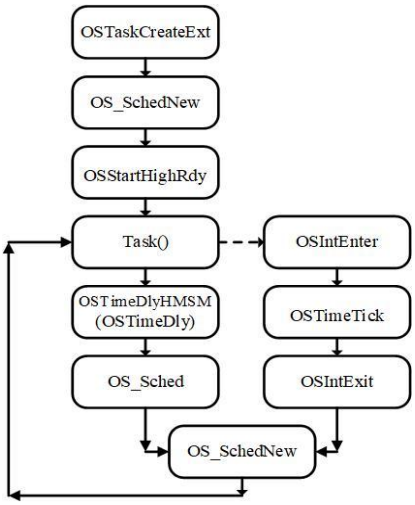
Credit:

[PART I] Task Control Block Linked List [20%]

- The screenshot results. (10%)
- A report that describes your implementation (please attach the screenshot of the code and **MARK** the modified part). (10%)

[PART II] RM Scheduler Implementation [70%]

- The correctness of schedule results of examples (**Output.txt**). Note the testing task set might not be the same as the given example task set. (25%)
- A report that describes your implementation (please attach the screenshot of the code and **MARK** the modified part). (40%)
  - Indicate the modified parts in the given example flowchart and provide reasons for their placement within the process.
- Implement and describe how to handle the deadline missing situation under RM. (5%)



Example Flowchart

[PART III] FIFO Scheduler Implementation [10%]

- The correctness of schedule results of examples (**Output.txt**). Note the testing task set might not be the same as the given example task set. (5%)
- Implement FIFO and compare the schedule results with that of RM (please attach the screenshot of the code and **MARK** the modified part). (5%)

※ 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.

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

※ Please set the system end time as 30 seconds in this project.

```
#define SYSTEM_END_TIME 30
```

※ You must define the algorithm in ucos_ii.h.

```
#define RM 0
#define FIFO 1
#define ALGORITHM RM/FIFO
```

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

※ We will check your answer on Output.txt.

※ When the current task is completed, the completion information shall be printed even if there is one task missing its deadline.

Project submit:

Submit to Moodle.

Submit deadline: November 3rd, 2025 (Moonday) 08:00

File name format: RTOS_Myyyddxxx_PA1.zip

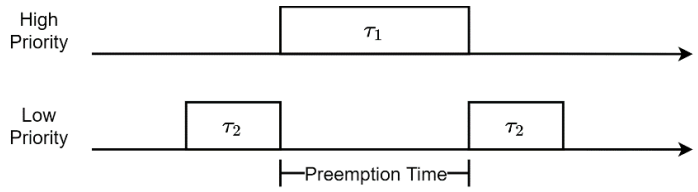
RTOS_Myyyddxxx_PA1.zip includes (The tree structure of files is shown as hints):

- The report (RTOS_Myyyddxxx_PA1.pdf).
- Folder with the executable project (RTOS_Myyyddxxx_PA1).

※ Plagiarizing is strictly prohibited.

Hints:

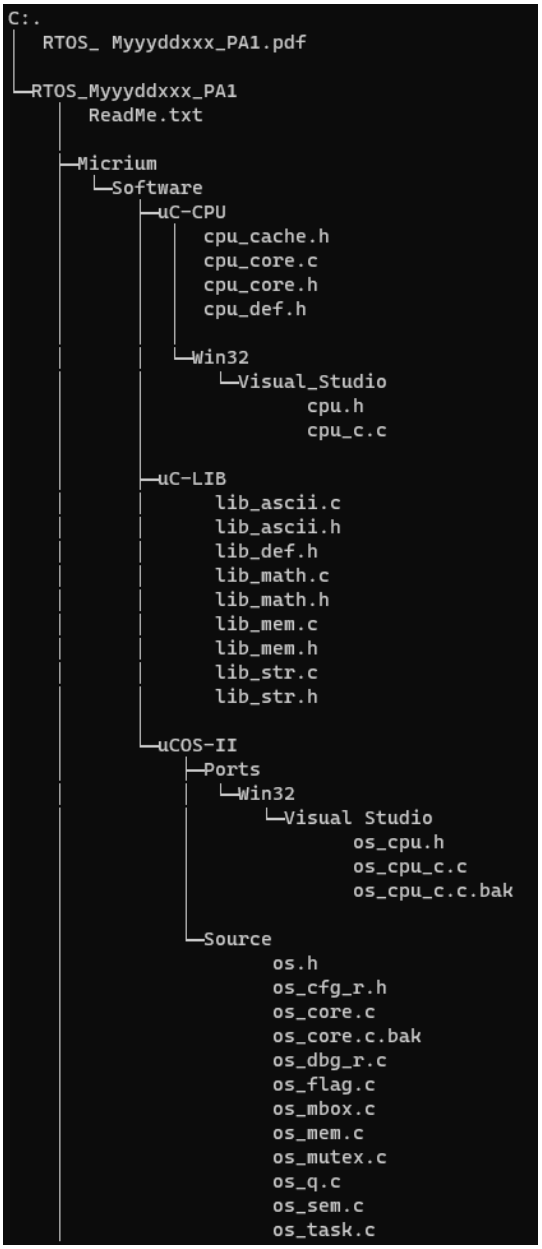
1. Preemption time is introduced in multiple tasking.

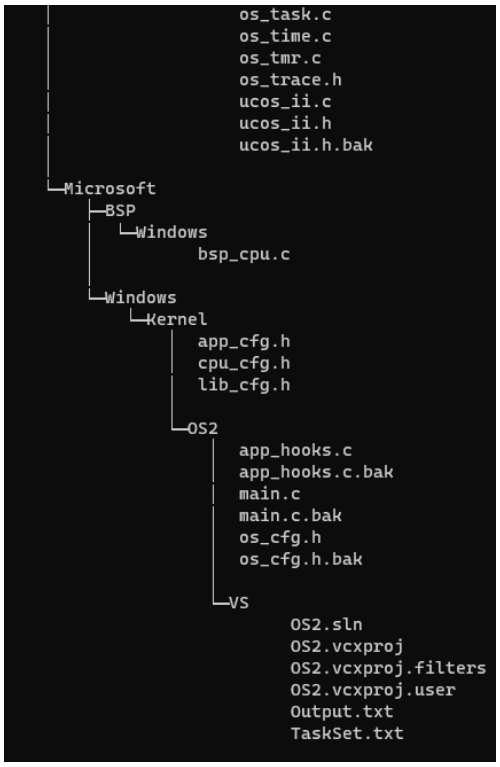


2. Advanced print function

```c
#define LOG_CONSOLE 1
#define LOG_FILE    2
#define LOG_BOTH    3

void LOG_print( INT8U LOG_type, char *outfile,
                _In_z_ _Printf_format_string_ char const* const _Format,
                ...){
    va_list _ArgList;
    __crt_va_start(_ArgList, _Format);

    if((LOG_type & LOG_CONSOLE))
        vprintf(_Format, _ArgList);

    if((LOG_type & LOG_FILE) && ((Output_err = fopen_s(&Output_fp, outfile, "a")) == 0)){
        vfprintf(Output_fp, _Format, _ArgList);
        fclose(Output_fp);
    }
    return;
}
```

3. RTOS_Myyyddxxx_PA1.zip include files as follows:


C:.
RTOS_ Myyyddxxx_PA1.pdf
RTOS_Myyyddxxx_PA1
ReadMe.txt
Micrium
Software
uC-CPU
cpu_cache.h
cpu_core.c
cpu_core.h
cpu_def.h
Win32
Visual_Studio
cpu.h
cpu_c.c
uC-LIB
lib_ascii.c
lib_ascii.h
lib_def.h
lib_math.c
lib_math.h
lib_mem.c
lib_mem.h
lib_str.c
lib_str.h
uCOS-II
Ports
Win32
Visual Studio
os_cpu.h
os_cpu_c.c
os_cpu_c.c.bak
Source
os.h
os_cfg_r.h
os_core.c
os_core.c.bak
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_task.c
os_time.c
os_tmr.c
os_trace.h
ucos_ii.c
ucos_ii.h
ucos_ii.h.bak
Microsoft
BSP
Windows
bsp_cpu.c
Windows
Kernel
app_cfg.h
cpu_cfg.h
lib_cfg.h
OS2
app_hooks.c
app_hooks.c.bak
main.c
main.c.bak
os_cfg.h
os_cfg.h.bak
VS
OS2.sln
OS2.vcxproj
OS2.vcxproj.filters
OS2.vcxproj.user
Output.txt
TaskSet.txt