

國立台灣科技大學
National Taiwan University of Science and Technology

Embedded OS Implementation, Fall 2020
Project #1 (due November 11, 2020
(Wednesday) 13:00)

老師：陳雅淑

學生：邱曄凱

一、 Task Control Block Linked List

1. screenshot results

```
C:\Users\user\Documents\Git\嵌入式作業系統實作HW\20201028_project1\
OSTick    created, Thread ID 16028
Task[ 63] created, TCB Address 00EF3260
-----After TCB[63] being linked-----
Previous TCB point to address 00000000
Current   TCB point to address 00EF3260
Next      TCB point to address 00000000

Task[  1] created, TCB Address 00EF32B8
-----After TCB[1] being linked-----
Previous TCB point to address 00000000
Current   TCB point to address 00EF32B8
Next      TCB point to address 00EF3260

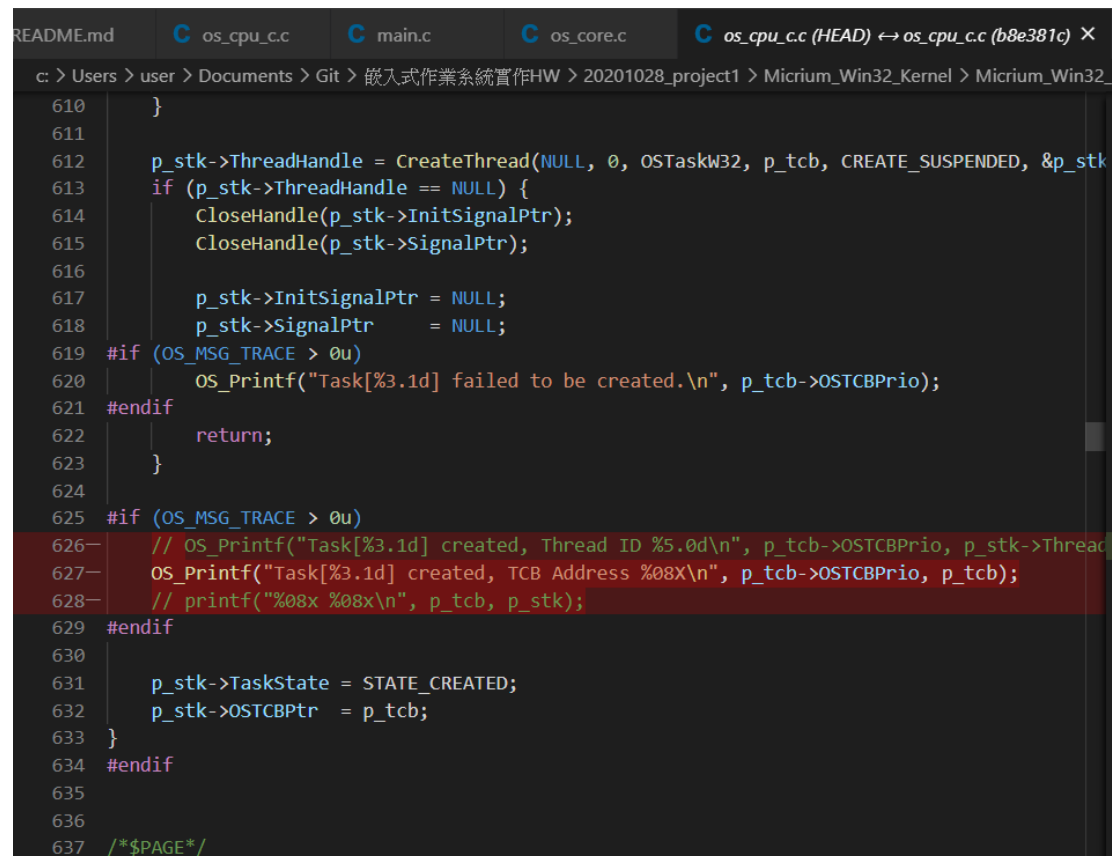
Task[  2] created, TCB Address 00EF3310
-----After TCB[2] being linked-----
Previous TCB point to address 00000000
Current   TCB point to address 00EF3310
Next      TCB point to address 00EF32B8

===== TCB linked list =====
Task    Prev_TCB_addr  TCB_addr    Next_TCB_addr
2       00000000          00EF3310    00EF32B8
1       00EF3310          00EF32B8    00EF3260
63      00EF32B8          00EF3260    00000000
```

2. implementation

利用一些內部變數(`ptcb`, `ostcbprio`)來印出 linked list

`os_cpu.c` 列印 created 的部分



```
610     }
611
612     p_stk->ThreadHandle = CreateThread(NULL, 0, OSTaskW32, p_tcb, CREATE_SUSPENDED, &p_stk
613     if (p_stk->ThreadHandle == NULL) {
614         CloseHandle(p_stk->InitSignalPtr);
615         CloseHandle(p_stk->SignalPtr);
616
617         p_stk->InitSignalPtr = NULL;
618         p_stk->SignalPtr = NULL;
619     #if (OS_MSG_TRACE > 0u)
620         OS_Printf("Task[%3.1d] failed to be created.\n", p_tcb->OSTCBPrio);
621     #endif
622     return;
623 }
624
625 #if (OS_MSG_TRACE > 0u)
626 // OS_Printf("Task[%3.1d] created, Thread ID %5.0d\n", p_tcb->OSTCBPrio, p_stk->Thread
627 OS_Printf("Task[%3.1d] created, TCB Address %08x\n", p_tcb->OSTCBPrio, p_tcb);
628 // printf("%08x %08x\n", p_tcb, p_stk);
629 #endif
630
631 p_stk->TaskState = STATE_CREATED;
632 p_stk->OSTCBPtr = p_tcb;
633 }
634 #endif
635
636
637 /*$PAGE*/
```

os_core.c 列印創建當下 previous current next

```
README.md  C os_cpu.c  C main.c  C os_core.c  C os_core.c (HEAD) ↔ os_core.c (b8e381c) X
c: > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel
2148         OSTCBLIST->OSTCBPrev = ptcb;
2149     }
2150     OSTCBLIST = ptcb;
2151     OSRdyGrp  |= ptcb->OSTCBBitY;      /* Make task ready to run
2152     OSRdyTbl[ptcb->OSTCBY] |= ptcb->OSTCBBitX;
2153     OSTaskCtr++;                      /* Increment the #tasks counter
2154     OS_TRACE_TASK_READY(ptcb);
2155     OS_EXIT_CRITICAL();
2156
2157     /////////////////////////////////// kevin ///////////////////////////////////
2158     printf("-----After TCB[%d] being linked-----\n",ptcb->OSTCBPrio);
2159     printf("Previous TCB point to address %08X\n", ptcb->OSTCBPrev);
2160     printf("Current  TCB point to address %08X\n", ptcb);
2161     printf("Next    TCB point to address %08X\n\n", ptcb->OSTCBNext);
2162     ///////////////////////////////////
2163
2164     return (OS_ERR_NONE);
2165 }
2166 OS_EXIT_CRITICAL();
2167 return (OS_ERR_TASK_NO_MORE_TCB);
2168 }
```

os_core.c 最後在 start 列印所有創建的 task

```
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;
        Kevin_StartContextSwitches();
        OSStartHighRdy(); /* Execute target specific code to start task */
    }
}
```

os_core.c

```
2175 void Kevin_StartContextSwitches(void) {
2176     // kevin linked list
2177     printf("===== TCB linked list =====\n");
2178     printf("Task \t Prev_TCB_addr \t TCB_addr \t Next_TCB_addr\n");
2179     for(int i = kevin_task_num; i > 0; i--)
2180         printf("%d \t %08X \t %08X \t %08X\n",i, OSTCBPrioTbl[i]->OSTCBPrev, OSTCBPrioTbl[i], OSTCBPrioTbl[i]->OSTCBNext);
2181     printf("%d \t %08X \t %08X \t %08X\n",63, OSTCBPrioTbl[63]->OSTCBPrev, OSTCBPrioTbl[63], OSTCBPrioTbl[63]->OSTCBNext);
2182
2183     // kevin rm title
2184     printf("\nTick \t Event \t \t CurrentTask ID \t NextTask ID \t ResponseTime \t # of ContextSwitch \n"); // kevin title
2185 }
```

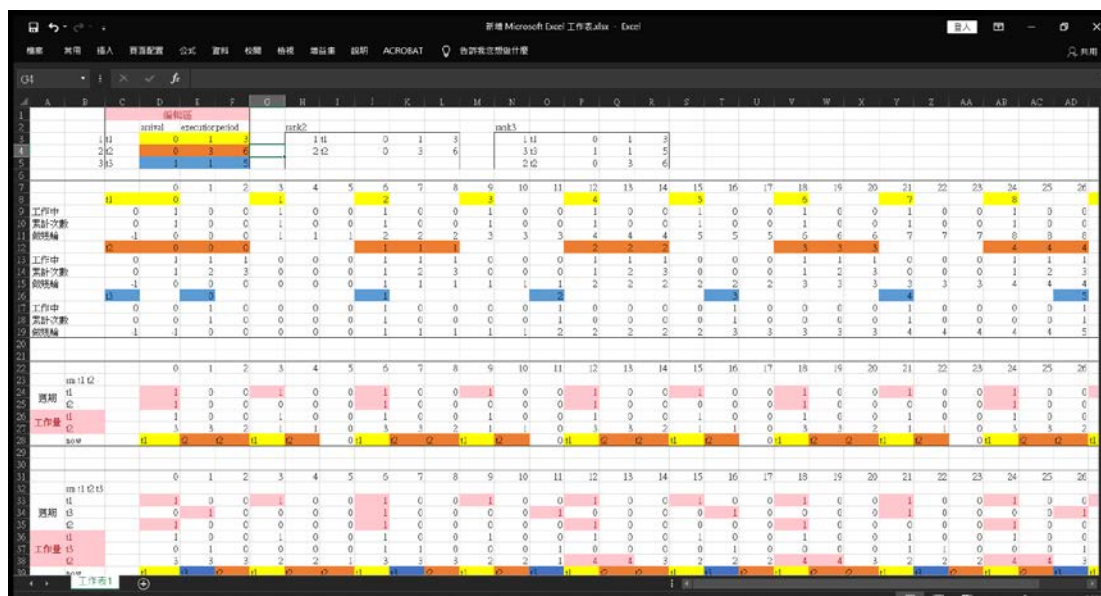
二、RM Scheduler Implementation

1. screenshot results

Task set 1 = $\{\tau_1(0, 1, 3), \tau_2(0, 3, 6)\}$

===== TCB linked list =====			
Task	Prev_TCB_addr	TCB_addr	Next_TCB_addr
2	00DE3368	00DE3310	00DE32E8
1	00DE3310	00DE32E8	00DE3260
63	00DE32E8	00DE3260	00000000

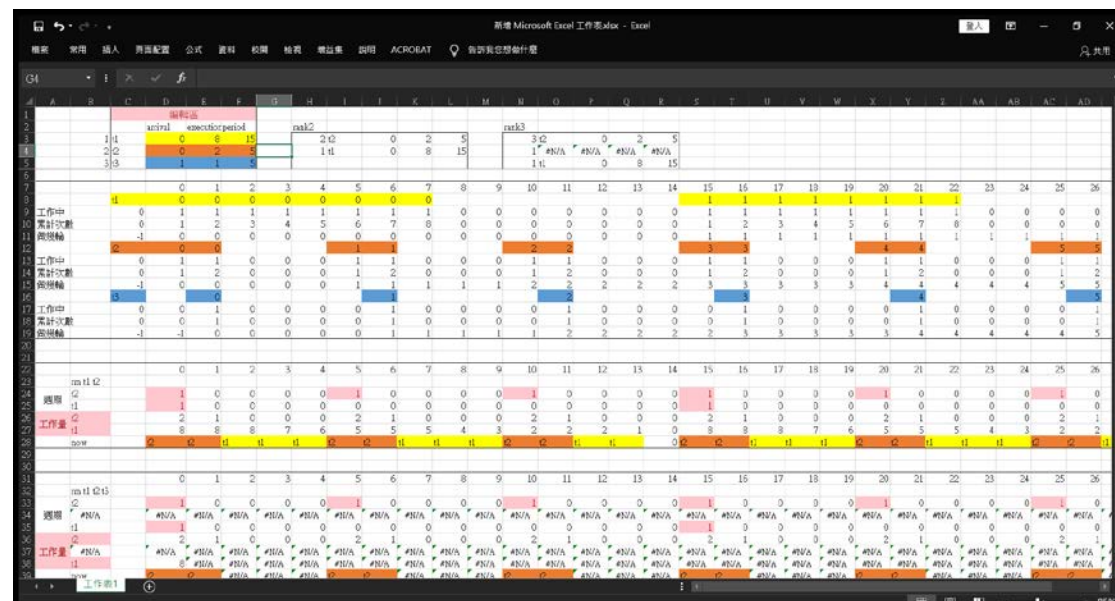
Tick	Event	CurrentTask ID	NextTask ID	ResponseTime	# of ContextSwitch
1	Completion	task(1)(0)	task(2)(0)	1	1
2	Preemption	task(2)(0)	task(1)(1)		
4	Completion	task(1)(1)	task(2)(0)	1	2
5	Completion	task(2)(0)	task(63)	5	4
6	Preemption	task(63)	task(1)(2)		
7	Completion	task(1)(2)	task(2)(1)	1	2
9	Preemption	task(2)(1)	task(1)(3)		
10	Completion	task(1)(3)	task(2)(1)	1	2
11	Completion	task(2)(1)	task(63)	5	4
12	Preemption	task(63)	task(1)(4)		
13	Completion	task(1)(4)	task(2)(2)	1	2
15	Preemption	task(2)(2)	task(1)(5)		
16	Completion	task(1)(5)	task(2)(2)	1	2
17	Completion	task(2)(2)	task(63)	5	4
18	Preemption	task(63)	task(1)(6)		
19	Completion	task(1)(6)	task(2)(3)	1	2
21	Preemption	task(2)(3)	task(1)(7)		
22	Completion	task(1)(7)	task(2)(3)	1	2
23	Completion	task(2)(3)	task(63)	5	4
24	Preemption	task(63)	task(1)(8)		
25	Completion	task(1)(8)	task(2)(4)	1	2
27	Preemption	task(2)(4)	task(1)(9)		
28	Completion	task(1)(9)	task(2)(4)	1	2
29	Completion	task(2)(4)	task(63)	5	4
30	Preemption	task(63)	task(1)(10)		



Task set 2 = { τ_1 (0, 8, 15), τ_2 (0, 2, 5)}

===== TCB linked list =====			
Task	Prev TCB_addr	TCB_addr	Next TCB_addr
2	00103368	00103310	001032B8
1	00103310	001032B8	00103260
63	001032B8	00103260	00000000

Tick	Event	CurrentTask ID	NextTask ID	ResponseTime	# of ContextSwitch
2	Completion	task(2)(0)	task(1)(0)	2	1
5	Preemption	task(1)(0)	task(2)(1)		
7	Completion	task(2)(1)	task(1)(0)	2	2
10	Preemption	task(1)(0)	task(2)(2)		
12	Completion	task(2)(2)	task(1)(0)	2	2
14	Completion	task(1)(0)	task(63)	14	6
15	Preemption	task(63)	task(2)(3)		
17	Completion	task(2)(3)	task(1)(1)	2	2
20	Preemption	task(1)(1)	task(2)(4)		
22	Completion	task(2)(4)	task(1)(1)	2	2
25	Preemption	task(1)(1)	task(2)(5)		
27	Completion	task(2)(5)	task(1)(1)	2	2
29	Completion	task(1)(1)	task(63)	14	6
30	Preemption	task(63)	task(2)(6)		
32	Completion	task(2)(6)	task(1)(2)	2	2
35	Preemption	task(1)(2)	task(2)(7)		



Task set 3 = { τ_1 (1, 1, 3), τ_2 (0, 4, 6)}

版本一：雖然 task2 做完繼續 task2 沒有經過 context switch，

但還是顯示方便觀測就變成 task(2)(0) task(2)(1)

```
C:\Users\user\Documents\Git\嵌入式作業系統實作HW\20201028_project1\Micrium_Win32_Kernel\Micrium_Win32_Kernel\Microsoft\Windows\...
Current TCB point to address 00983260
Next TCB point to address 00000000

Task[ 1] created, TCB Address 009832B8
-----After TCB[1] being linked-----
Previous TCB point to address 00000000
Current TCB point to address 009832B8
Next TCB point to address 00983260

Task[ 2] created, TCB Address 00983310
-----After TCB[2] being linked-----
Previous TCB point to address 00000000
Current TCB point to address 00983310
Next TCB point to address 009832B8

===== TCB linked list =====
Task   Prev TCB_addr  TCB_addr  Next TCB_addr
2      00000000      00983310  009832B8
1      00983310      009832B8  00983260
63     009832B8      00983260  00000000

Tick   Event          CurrentTask ID  NextTask ID  ResponseTime  # of ContextSwitch
1      Preemption        task(2)(0)     task(1)(0)
2      Completion        task(1)(0)     task(2)(0)   1             2
4      Preemption        task(2)(0)     task(1)(1)
5      Completion        task(1)(1)     task(2)(0)   1             2
6      Completion        task(2)(0)     task(2)(1)   6             4
7      Preemption        task(2)(1)     task(1)(2)
8      Completion        task(1)(2)     task(2)(1)   1             2
10     Preemption        task(2)(1)     task(1)(3)
11     Completion        task(1)(3)     task(2)(1)   1             2
12     Completion        task(2)(1)     task(2)(2)   6             4
13     Preemption        task(2)(2)     task(1)(4)
14     Completion        task(1)(4)     task(2)(2)   1             2
16     Preemption        task(2)(2)     task(1)(5)
17     Completion        task(1)(5)     task(2)(2)   1             2
18     Completion        task(2)(2)     task(2)(3)   6             4
19     Preemption        task(2)(3)     task(1)(6)
20     Completion        task(1)(6)     task(2)(3)   1             2
22     Preemption        task(2)(3)     task(1)(7)
23     Completion        task(1)(7)     task(2)(3)   1             2
24     Completion        task(2)(3)     task(2)(4)   6             4
25     Preemption        task(2)(4)     task(1)(8)
26     Completion        task(1)(8)     task(2)(4)   1             2
28     Preemption        task(2)(4)     task(1)(9)
29     Completion        task(1)(9)     task(2)(4)   1             2
30     Completion        task(2)(4)     task(2)(5)   6             4
```

版本二：就假設題目希望 context switch 時才列印，就會變這樣看不到 task2 completion，但他的 job 其實是有再增加

```

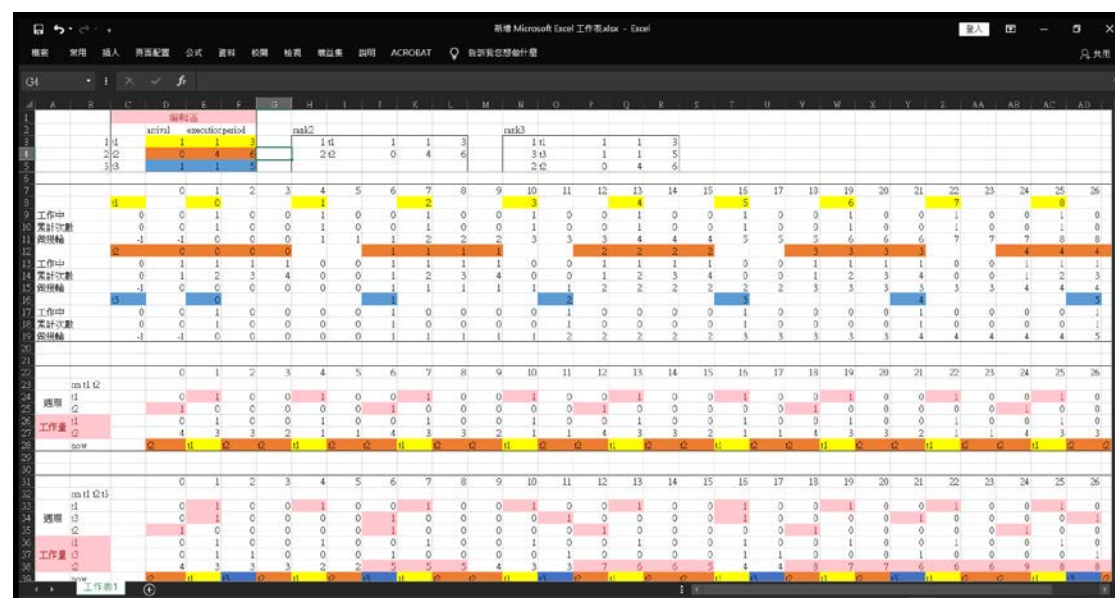
C:\Users\user\Documents\Git\嵌入式作業系統實作HW\20201028_project1\Micrium_Win32_Kernel\Micrium_Win32_Kernel\Microsoft\Windows...
Task[ 1] created, TCB Address 005A42B8
-----After TCB[1] being linked-----
Previous TCB point to address 00000000
Current TCB point to address 005A42B8
Next TCB point to address 005A4260

Task[ 2] created, TCB Address 005A4310
-----After TCB[2] being linked-----
Previous TCB point to address 00000000
Current TCB point to address 005A4310
Next TCB point to address 005A42B8

===== TCB linked list =====
Task  Prev_TCB_addr  TCB_addr  Next_TCB_addr
2      00000000      005A4310  005A42B8
1      005A4310      005A42B8  005A4260
63     005A42B8      005A4260  00000000

Tick  Event          CurrentTask ID  NextTask ID  ResponseTime  # of ContextSwitch
1      Preemption      task(2)(0)     task(1)(0)   1              2
2      Completion      task(1)(0)     task(2)(0)   1              2
4      Preemption      task(2)(0)     task(1)(1)   1              2
5      Completion      task(1)(1)     task(2)(0)   1              2
7      Preemption      task(2)(1)     task(1)(2)   1              2
8      Completion      task(1)(2)     task(2)(1)   1              2
10     Preemption      task(2)(1)     task(1)(3)   1              2
11     Completion      task(1)(3)     task(2)(1)   1              2
13     Preemption      task(2)(2)     task(1)(4)   1              2
14     Completion      task(1)(4)     task(2)(2)   1              2
16     Preemption      task(2)(2)     task(1)(5)   1              2
17     Completion      task(1)(5)     task(2)(2)   1              2
19     Preemption      task(2)(3)     task(1)(6)   1              2
20     Completion      task(1)(6)     task(2)(3)   1              2
22     Preemption      task(2)(3)     task(1)(7)   1              2
23     Completion      task(1)(7)     task(2)(3)   1              2
25     Preemption      task(2)(4)     task(1)(8)   1              2
26     Completion      task(1)(8)     task(2)(4)   1              2
28     Preemption      task(2)(4)     task(1)(9)   1              2
29     Completion      task(1)(9)     task(2)(4)   1              2
31     Preemption      task(2)(5)     task(1)(10)  1              2
32     Completion      task(1)(10)    task(2)(5)   1              2

```



Task set 4 = { τ_1 (0, 4, 6), τ_2 (2, 2, 10), τ_3 (1, 1, 5)}

版本一：在 context 才檢查自己 miss 為了符合標題 currentTaskID

```

C:\Users\user\Documents\Git\嵌入式作業系統實作HW\20201028_project1\Micrium_Win32_Kernel\Micrium_Win32_Kernel...
OSTick created, Thread ID 2860
Task[ 63] created, TCB Address 00A63260
-----After TCB[63] being linked-----
Previous TCB point to address 00000000
Current TCB point to address 00A63260
Next TCB point to address 00000000

Task[ 1] created, TCB Address 00A632B8
-----After TCB[1] being linked-----
Previous TCB point to address 00000000
Current TCB point to address 00A632B8
Next TCB point to address 00A63260

Task[ 2] created, TCB Address 00A63310
-----After TCB[2] being linked-----
Previous TCB point to address 00000000
Current TCB point to address 00A63310
Next TCB point to address 00A632B8

Task[ 3] created, TCB Address 00A63368
-----After TCB[3] being linked-----
Previous TCB point to address 00000000
Current TCB point to address 00A63368
Next TCB point to address 00A63310

===== TCB linked list =====
Task   Prev_TCB_addr  TCB_addr  Next_TCB_addr
3      00000000        00A63368  00A63310
2      00A63368        00A63310  00A632B8
1      00A63310        00A632B8  00A63260
63     00A632B8        00A63260  00000000

Tick   Event          CurrentTask ID  NextTask ID  ResponseTime  # of ContextSwitch
1      Preemption        task(1)(0)      task(3)(0)
2      Completion        task(3)(0)      task(1)(0)      1              2
5      Completion        task(1)(0)      task(2)(0)      5              3
6      Preemption        task(2)(0)      task(3)(1)
7      Completion        task(3)(1)      task(1)(1)      1              2
11     Completion        task(1)(1)      task(3)(2)      5              2
12     Completion        task(3)(2)      task(1)(2)      1              2
16     Completion        task(1)(2)      task(3)(3)      4              2
17     Completion        task(3)(3)      task(2)(0)      1              2
18     MissDeadline      task(2)(0)      -----

```

版本二：但其實在週期時檢查，工作量就已經

```
C:\Users\user\Documents\Git\嵌入式作業系統實作HW\20201028_project1\Micrium_Win32_Kernel\Micrium_Win32_Kernel\Microsoft\Windows\...
OSTick created, Thread ID 6080
Task[ 63] created, TCB Address 00BF5CA0
-----After TCB[63] being linked-----
Previous TCB point to address 00000000
Current TCB point to address 00BF5CA0
Next TCB point to address 00000000

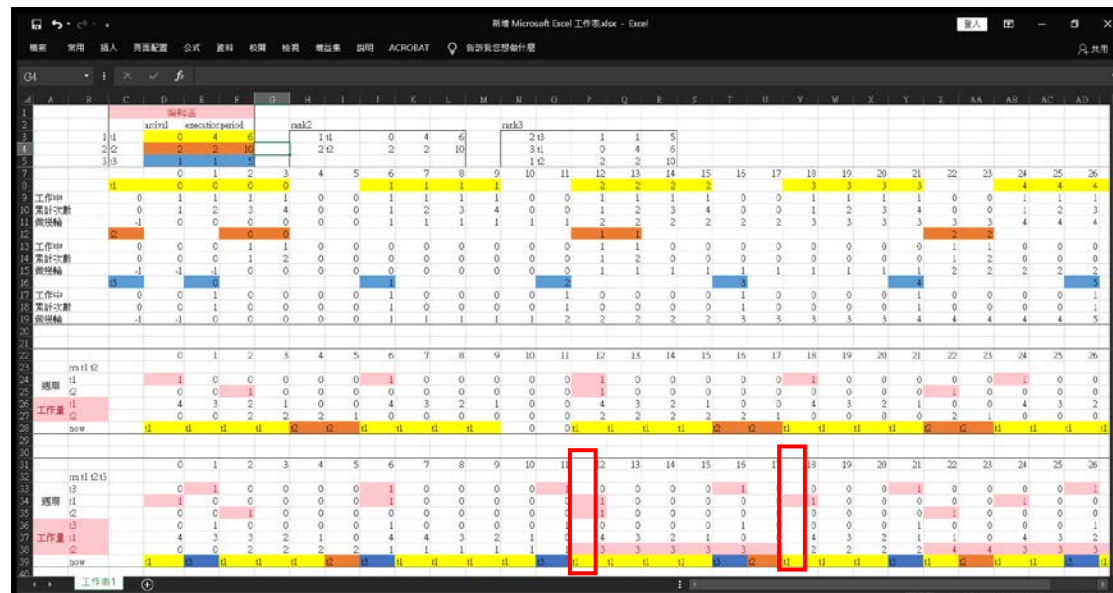
Task[ 1] created, TCB Address 00BF5CF8
-----After TCB[1] being linked-----
Previous TCB point to address 00000000
Current TCB point to address 00BF5CF8
Next TCB point to address 00BF5CA0

Task[ 2] created, TCB Address 00BF5D50
-----After TCB[2] being linked-----
Previous TCB point to address 00000000
Current TCB point to address 00BF5D50
Next TCB point to address 00BF5CF8

Task[ 3] created, TCB Address 00BF5DA8
-----After TCB[3] being linked-----
Previous TCB point to address 00000000
Current TCB point to address 00BF5DA8
Next TCB point to address 00BF5D50

===== TCB linked list =====
Task Prev_TCB_addr TCB_addr Next_TCB_addr
3 00000000 00BF5DA8 00BF5D50
2 00BF5DA8 00BF5D50 00BF5CF8
1 00BF5D50 00BF5CF8 00BF5CA0
63 00BF5CF8 00BF5CA0 00000000

Tick Event CurrentTask ID NextTask ID ResponseTime # of ContextSwitch
1 Preemption task(1)(0) task(3)(0)
2 Completion task(3)(0) task(1)(0) 1 2
5 Completion task(1)(0) task(2)(0) 5 3
6 Preemption task(2)(0) task(3)(1)
7 Completion task(3)(1) task(1)(1) 1 2
11 Completion task(1)(1) task(3)(2) 5 2
12 MissDeadline task(2)(0) -----
```



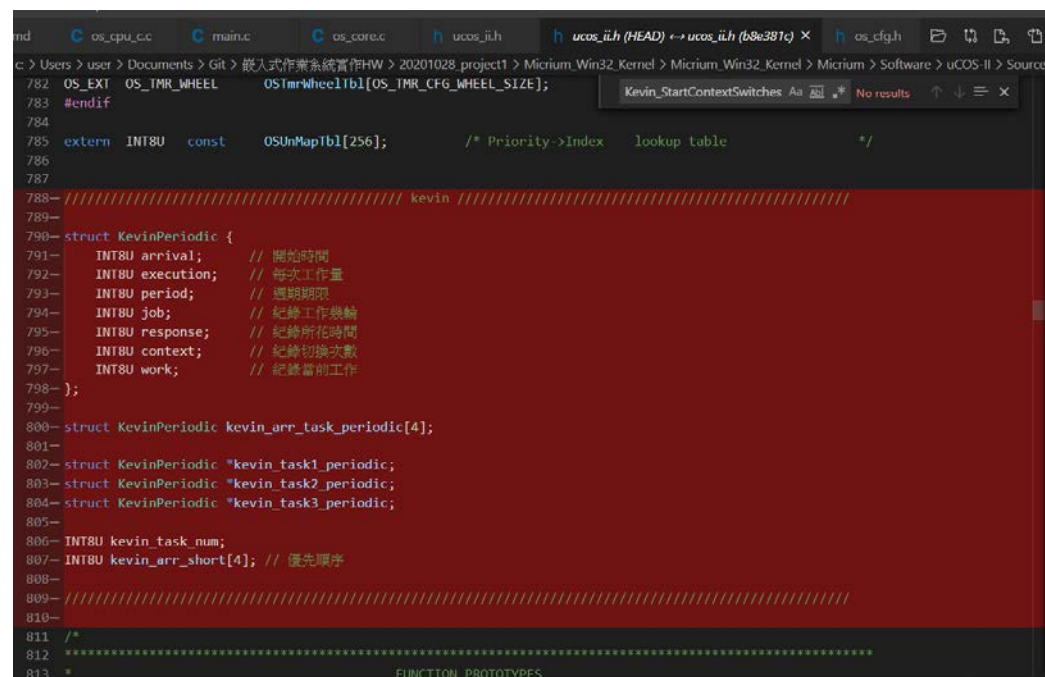
2. implementation

取代了原本 OS_SchedNew 找 OSPrioHighRdy 的方式，

改由自己建立的 RMS 找到當下的 OSPrioHighRdy，

全部都在 kernal 完成，main 只負責卡著任務不斷重複。

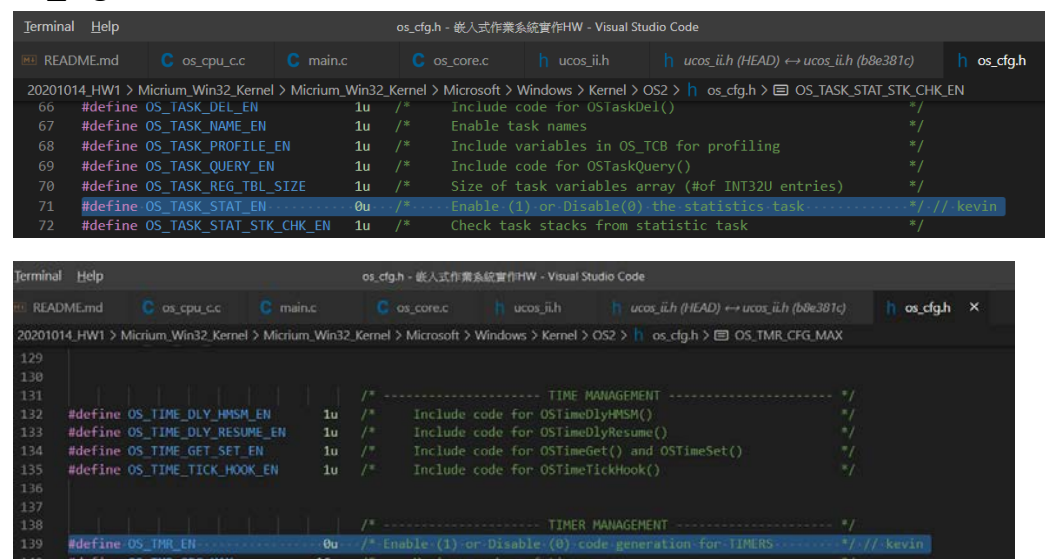
ucos_ii.h



```
782 OS_EXT OS_TMR_WHEEL    OSTmrWheelTbl[OS_TMR_CFG_WHEEL_SIZE];
783 #endif
784
785 extern INT8U const    OSUnMapTbl[256];          /* Priority->Index    lookup table    */
786
787
788- //////////////////////////////////// kevin ////////////////////////////////////
789-
790- struct KevinPeriodic {
791-     INT8U arrival;      // 開始時間
792-     INT8U execution;    // 每次工作量
793-     INT8U period;       // 週期期限
794-     INT8U job;          // 紀錄工作幾輪
795-     INT8U response;     // 紀錄所花時間
796-     INT8U context;      // 紀錄切換次數
797-     INT8U work;         // 紀錄當前工作
798- };
799-
800- struct KevinPeriodic kevin_arr_task_periodic[4];
801-
802- struct KevinPeriodic *kevin_task1_periodic;
803- struct KevinPeriodic *kevin_task2_periodic;
804- struct KevinPeriodic *kevin_task3_periodic;
805-
806- INT8U kevin_task_num;
807- INT8U kevin_arr_short[4]; // 優先順序
808-
809- ////////////////////////////////////
810-
811- /*
812- *****
813- *
814- *
815- *
816- *
817- *
818- *
819- *
820- *
821- *
822- *
823- *
824- *
825- *
826- *
827- *
828- *
829- *
830- *
831- *
832- *
833- *
834- *
835- *
836- *
837- *
838- *
839- *
840- *
841- *
842- *
843- *
844- *
845- *
846- *
847- *
848- *
849- *
850- *
851- *
852- *
853- *
854- *
855- *
856- *
857- *
858- *
859- *
860- *
861- *
862- *
863- *
864- *
865- *
866- *
867- *
868- *
869- *
870- *
871- *
872- *
873- *
874- *
875- *
876- *
877- *
878- *
879- *
880- *
881- *
882- *
883- *
884- *
885- *
886- *
887- *
888- *
889- *
890- *
891- *
892- *
893- *
894- *
895- *
896- *
897- *
898- *
899- *
900- *
901- *
902- *
903- *
904- *
905- *
906- *
907- *
908- *
909- *
910- *
911- *
912- *
913- *
914- *
915- *
916- *
917- *
918- *
919- *
920- *
921- *
922- *
923- *
924- *
925- *
926- *
927- *
928- *
929- *
930- *
931- *
932- *
933- *
934- *
935- *
936- *
937- *
938- *
939- *
940- *
941- *
942- *
943- *
944- *
945- *
946- *
947- *
948- *
949- *
950- *
951- *
952- *
953- *
954- *
955- *
956- *
957- *
958- *
959- *
960- *
961- *
962- *
963- *
964- *
965- *
966- *
967- *
968- *
969- *
970- *
971- *
972- *
973- *
974- *
975- *
976- *
977- *
978- *
979- *
980- *
981- *
982- *
983- *
984- *
985- *
986- *
987- *
988- *
989- *
990- *
991- *
992- *
993- *
994- *
995- *
996- *
997- *
998- *
999- *
1000- */
1001-
1002- #endif
1003-
1004- #endif
1005-
1006- #endif
1007-
1008- #endif
1009-
1010- #endif
1011-
1012- #endif
1013-
1014- #endif
1015-
1016- #endif
1017-
1018- #endif
1019-
1020- #endif
1021-
1022- #endif
1023-
1024- #endif
1025-
1026- #endif
1027-
1028- #endif
1029-
1030- #endif
1031-
1032- #endif
1033-
1034- #endif
1035-
1036- #endif
1037-
1038- #endif
1039-
1040- #endif
1041-
1042- #endif
1043-
1044- #endif
1045-
1046- #endif
1047-
1048- #endif
1049-
1050- #endif
1051-
1052- #endif
1053-
1054- #endif
1055-
1056- #endif
1057-
1058- #endif
1059-
1060- #endif
1061-
1062- #endif
1063-
1064- #endif
1065-
1066- #endif
1067-
1068- #endif
1069-
1070- #endif
1071-
1072- #endif
1073-
1074- #endif
1075-
1076- #endif
1077-
1078- #endif
1079-
1080- #endif
1081-
1082- #endif
1083-
1084- #endif
1085-
1086- #endif
1087-
1088- #endif
1089-
1090- #endif
1091-
1092- #endif
1093-
1094- #endif
1095-
1096- #endif
1097-
1098- #endif
1099-
1100- #endif
1101-
1102- #endif
1103-
1104- #endif
1105-
1106- #endif
1107-
1108- #endif
1109-
1110- #endif
1111-
1112- #endif
1113-
1114- #endif
1115-
1116- #endif
1117-
1118- #endif
1119-
1120- #endif
1121-
1122- #endif
1123-
1124- #endif
1125-
1126- #endif
1127-
1128- #endif
1129-
1130- #endif
1131-
1132- #endif
1133-
1134- #endif
1135-
1136- #endif
1137-
1138- #endif
1139-
1140- #endif
1141-
1142- #endif
1143-
1144- #endif
1145-
1146- #endif
1147-
1148- #endif
1149-
1150- #endif
1151-
1152- #endif
1153-
1154- #endif
1155-
1156- #endif
1157-
1158- #endif
1159-
1160- #endif
1161-
1162- #endif
1163-
1164- #endif
1165-
1166- #endif
1167-
1168- #endif
1169-
1170- #endif
1171-
1172- #endif
1173-
1174- #endif
1175-
1176- #endif
1177-
1178- #endif
1179-
1180- #endif
1181-
1182- #endif
1183-
1184- #endif
1185-
1186- #endif
1187-
1188- #endif
1189-
1190- #endif
1191-
1192- #endif
1193-
1194- #endif
1195-
1196- #endif
1197-
1198- #endif
1199-
1200- #endif
1201-
1202- #endif
1203-
1204- #endif
1205-
1206- #endif
1207-
1208- #endif
1209-
1210- #endif
1211-
1212- #endif
1213-
1214- #endif
1215-
1216- #endif
1217-
1218- #endif
1219-
1220- #endif
1221-
1222- #endif
1223-
1224- #endif
1225-
1226- #endif
1227-
1228- #endif
1229-
1230- #endif
1231-
1232- #endif
1233-
1234- #endif
1235-
1236- #endif
1237-
1238- #endif
1239-
1240- #endif
1241-
1242- #endif
1243-
1244- #endif
1245-
1246- #endif
1247-
1248- #endif
1249-
1250- #endif
1251-
1252- #endif
1253-
1254- #endif
1255-
1256- #endif
1257-
1258- #endif
1259-
1260- #endif
1261-
1262- #endif
1263-
1264- #endif
1265-
1266- #endif
1267-
1268- #endif
1269-
1270- #endif
1271-
1272- #endif
1273-
1274- #endif
1275-
1276- #endif
1277-
1278- #endif
1279-
1280- #endif
1281-
1282- #endif
1283-
1284- #endif
1285-
1286- #endif
1287-
1288- #endif
1289-
1290- #endif
1291-
1292- #endif
1293-
1294- #endif
1295-
1296- #endif
1297-
1298- #endif
1299-
1300- #endif
1301-
1302- #endif
1303-
1304- #endif
1305-
1306- #endif
1307-
1308- #endif
1309-
1310- #endif
1311-
1312- #endif
1313-
1314- #endif
1315-
1316- #endif
1317-
1318- #endif
1319-
1320- #endif
1321-
1322- #endif
1323-
1324- #endif
1325-
1326- #endif
1327-
1328- #endif
1329-
1330- #endif
1331-
1332- #endif
1333-
1334- #endif
1335-
1336- #endif
1337-
1338- #endif
1339-
1340- #endif
1341-
1342- #endif
1343-
1344- #endif
1345-
1346- #endif
1347-
1348- #endif
1349-
1350- #endif
1351-
1352- #endif
1353-
1354- #endif
1355-
1356- #endif
1357-
1358- #endif
1359-
1360- #endif
1361-
1362- #endif
1363-
1364- #endif
1365-
1366- #endif
1367-
1368- #endif
1369-
1370- #endif
1371-
1372- #endif
1373-
1374- #endif
1375-
1376- #endif
1377-
1378- #endif
1379-
1380- #endif
1381-
1382- #endif
1383-
1384- #endif
1385-
1386- #endif
1387-
1388- #endif
1389-
1390- #endif
1391-
1392- #endif
1393-
1394- #endif
1395-
1396- #endif
1397-
1398- #endif
1399-
1400- #endif
1401-
1402- #endif
1403-
1404- #endif
1405-
1406- #endif
1407-
1408- #endif
1409-
1410- #endif
1411-
1412- #endif
1413-
1414- #endif
1415-
1416- #endif
1417-
1418- #endif
1419-
1420- #endif
1421-
1422- #endif
1423-
1424- #endif
1425-
1426- #endif
1427-
1428- #endif
1429-
1430- #endif
1431-
1432- #endif
1433-
1434- #endif
1435-
1436- #endif
1437-
1438- #endif
1439-
1440- #endif
1441-
1442- #endif
1443-
1444- #endif
1445-
1446- #endif
1447-
1448- #endif
1449-
1450- #endif
1451-
1452- #endif
1453-
1454- #endif
1455-
1456- #endif
1457-
1458- #endif
1459-
1460- #endif
1461-
1462- #endif
1463-
1464- #endif
1465-
1466- #endif
1467-
1468- #endif
1469-
1470- #endif
1471-
1472- #endif
1473-
1474- #endif
1475-
1476- #endif
1477-
1478- #endif
1479-
1480- #endif
1481-
1482- #endif
1483-
1484- #endif
1485-
1486- #endif
1487-
1488- #endif
1489-
1490- #endif
1491-
1492- #endif
1493-
1494- #endif
1495-
1496- #endif
1497-
1498- #endif
1499-
1500- #endif
1501-
1502- #endif
1503-
1504- #endif
1505-
1506- #endif
1507-
1508- #endif
1509-
1510- #endif
1511-
1512- #endif
1513-
1514- #endif
1515-
1516- #endif
1517-
1518- #endif
1519-
1520- #endif
1521-
1522- #endif
1523-
1524- #endif
1525-
1526- #endif
1527-
1528- #endif
1529-
1530- #endif
1531-
1532- #endif
1533-
1534- #endif
1535-
1536- #endif
1537-
1538- #endif
1539-
1540- #endif
1541-
1542- #endif
1543-
1544- #endif
1545-
1546- #endif
1547-
1548- #endif
1549-
1550- #endif
1551-
1552- #endif
1553-
1554- #endif
1555-
1556- #endif
1557-
1558- #endif
1559-
1560- #endif
1561-
1562- #endif
1563-
1564- #endif
1565-
1566- #endif
1567-
1568- #endif
1569-
1570- #endif
1571-
1572- #endif
1573-
1574- #endif
1575-
1576- #endif
1577-
1578- #endif
1579-
1580- #endif
1581-
1582- #endif
1583-
1584- #endif
1585-
1586- #endif
1587-
1588- #endif
1589-
1590- #endif
1591-
1592- #endif
1593-
1594- #endif
1595-
1596- #endif
1597-
1598- #endif
1599-
1600- #endif
1601-
1602- #endif
1603-
1604- #endif
1605-
1606- #endif
1607-
1608- #endif
1609-
1610- #endif
1611-
1612- #endif
1613-
1614- #endif
1615-
1616- #endif
1617-
1618- #endif
1619-
1620- #endif
1621-
1622- #endif
1623-
1624- #endif
1625-
1626- #endif
1627-
1628- #endif
1629-
1630- #endif
1631-
1632- #endif
1633-
1634- #endif
1635-
1636- #endif
1637-
1638- #endif
1639-
1640- #endif
1641-
1642- #endif
1643-
1644- #endif
1645-
1646- #endif
1647-
1648- #endif
1649-
1650- #endif
1651-
1652- #endif
1653-
1654- #endif
1655-
1656- #endif
1657-
1658- #endif
1659-
1660- #endif
1661-
1662- #endif
1663-
1664- #endif
1665-
1666- #endif
1667-
1668- #endif
1669-
1670- #endif
1671-
1672- #endif
1673-
1674- #endif
1675-
1676- #endif
1677-
1678- #endif
1679-
1680- #endif
1681-
1682- #endif
1683-
1684- #endif
1685-
1686- #endif
1687-
1688- #endif
1689-
1690- #endif
1691-
1692- #endif
1693-
1694- #endif
1695-
1696- #endif
1697-
1698- #endif
1699-
1700- #endif
1701-
1702- #endif
1703-
1704- #endif
1705-
1706- #endif
1707-
1708- #endif
1709-
1710- #endif
1711-
1712- #endif
1713-
1714- #endif
1715-
1716- #endif
1717-
1718- #endif
1719-
1720- #endif
1721-
1722- #endif
1723-
1724- #endif
1725-
1726- #endif
1727-
1728- #endif
1729-
1730- #endif
1731-
1732- #endif
1733-
1734- #endif
1735-
1736- #endif
1737-
1738- #endif
1739-
1740- #endif
1741-
1742- #endif
1743-
1744- #endif
1745-
1746- #endif
1747-
1748- #endif
1749-
1750- #endif
1751-
1752- #endif
1753-
1754- #endif
1755-
1756- #endif
1757-
1758- #endif
1759-
1760- #endif
1761-
1762- #endif
1763-
1764- #endif
1765-
1766- #endif
1767-
1768- #endif
1769-
1770- #endif
1771-
1772- #endif
1773-
1774- #endif
1775-
1776- #endif
1777-
1778- #endif
1779-
1780- #endif
1781-
1782- #endif
1783-
1784- #endif
1785-
1786- #endif
1787-
1788- #endif
1789-
1790- #endif
1791-
1792- #endif
1793-
1794- #endif
1795-
1796- #endif
1797-
1798- #endif
1799-
1800- #endif
1801-
1802- #endif
1803-
1804- #endif
1805-
1806- #endif
1807-
1808- #endif
1809-
1810- #endif
1811-
1812- #endif
1813-
1814- #endif
1815-
1816- #endif
1817-
1818- #endif
1819-
1820- #endif
1821-
1822- #endif
1823-
1824- #endif
1825-
1826- #endif
1827-
1828- #endif
1829-
1830- #endif
1831-
1832- #endif
1833-
1834- #endif
1835-
1836- #endif
1837-
1838- #endif
1839-
1840- #endif
1841-
1842- #endif
1843-
1844- #endif
1845-
1846- #endif
1847-
1848- #endif
1849-
1850- #endif
1851-
1852- #endif
1853-
1854- #endif
1855-
1856- #endif
1857-
1858- #endif
1859-
1860- #endif
1861-
1862- #endif
1863-
1864- #endif
1865-
1866- #endif
1867-
1868- #endif
1869-
1870- #endif
1871-
1872- #endif
1873-
1874- #endif
1875-
1876- #endif
1877-
1878- #endif
1879-
1880- #endif
1881-
1882- #endif
1883-
1884- #endif
1885-
1886- #endif
1887-
1888- #endif
1889-
1890- #endif
1891-
1892- #endif
1893-
1894- #endif
1895-
1896- #endif
1897-
1898- #endif
1899-
1900- #endif
1901-
1902- #endif
1903-
1904- #endif
1905-
1906- #endif
1907-
1908- #endif
1909-
1910- #endif
1911-
1912- #endif
1913-
1914- #endif
1915-
1916- #endif
1917-
1918- #endif
1919-
1920- #endif
1921-
1922- #endif
1923-
1924- #endif
1925-
1926- #endif
1927-
1928- #endif
1929-
1930- #endif
1931-
1932- #endif
1933-
1934- #endif
1935-
1936- #endif
1937-
1938- #endif
1939-
1940- #endif
1941-
1942- #endif
1943-
1944- #endif
1945-
1946- #endif
1947-
1948- #endif
1949-
1950- #endif
1951-
1952- #endif
1953-
1954- #endif
1955-
1956- #endif
1957-
1958- #endif
1959-
1960- #endif
1961-
1962- #endif
1963-
1964- #endif
1965-
1966- #endif
1967-
1968- #endif
1969-
1970- #endif
1971-
1972- #endif
1973-
1974- #endif
1975-
1976- #endif
1977-
1978- #endif
1979-


```

os_cfg.h



```
20201014_HW1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Microsoft > Windows > Kernel > OS2 > h os_cfg.h > OS_TASK_STAT_CHK_EN
66 #define OS_TASK_DEL_EN      1u /* Include code for OSTaskDel() */
67 #define OS_TASK_NAME_EN     1u /* Enable task names */
68 #define OS_TASK_PROFILE_EN  1u /* Include variables in OS_TCB for profiling */
69 #define OS_TASK_QUERY_EN    1u /* Include code for OSTaskQuery() */
70 #define OS_TASK_REG_TBL_SIZE 1u /* Size of task variables array (#of INT32U entries) */
71 #define OS_TASK_STAT_EN      0u /* Enable (1) or Disable (0) the statistics task */
72 #define OS_TASK_STAT_CHK_EN  1u /* Check task stacks from statistic task */

20201014_HW1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Microsoft > Windows > Kernel > OS2 > h os_cfg.h > OS_TMR_CFG_MAX
129
130
131 /* ----- TIME MANAGEMENT ----- */
132 #define OS_TIME_DLY_HMSM_EN  1u /* Include code for OSTimeDlyHMSM() */
133 #define OS_TIME_DLY_RESUME_EN 1u /* Include code for OSTimeDlyResume() */
134 #define OS_TIME_GET_SET_EN   1u /* Include code for OSTimeGet() and OSTimeSet() */
135 #define OS_TIME_TICK_HOOK_EN 1u /* Include code for OSTimeTickHook() */
136
137
138 /* ----- TIMER MANAGEMENT ----- */
139 #define OS_TMR_EN             0u /* Enable (1) or Disable (0) code generation for TIMERS */
140 #define OS_TMR_CFG_MAX       16u /* Maximum number of timers */
```

main.c 基本上沒什麼改

```
Terminal Help main.c (HEAD) ↔ main.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code
README.md os_cpu_c.c main.c os_core.c ucos_ii.h main.c (HEAD) ↔ main.c (b8e381c) X
c > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Microsoft
42
43
44 /*
45 ****
46 * LOCAL DEFINES
47 ****
48 */
49-
50- #define TASK_STACKSIZE 2048
51- #define TASK1_PRIORITY 1 // kevin 設多少都沒差後面會改
52- #define TASK2_PRIORITY 2 // kevin 設多少都沒差後面會改
53- #define TASK3_PRIORITY 3 // kevin 設多少都沒差後面會改
54- #define TASK1_ID 1
55- #define TASK2_ID 2
56- #define TASK3_ID 3
57
```

```
Terminal Help main.c (HEAD) ↔ main.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code
README.md os_cpu_c.c main.c os_core.c ucos_ii.h main.c (HEAD) ↔ main.c (b8e381c) X
c > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Microsoft > W
66 static OS_STK Task1_STK[TASK_STACKSIZE];
67 static OS_STK Task2_STK[TASK_STACKSIZE];
68- static OS_STK Task3_STK[TASK_STACKSIZE];
69
70 /*
71 ****
72 * FUNCTION PROTOTYPES
73 ****
74 */
75
76 static void StartupTask (void *p_arg);
77
78- static void task1(void* p_arg);
79- static void task2(void* p_arg);
80- static void task3(void* p_arg); // kevin
81
```

```
Terminal Help main.c (HEAD) ↔ main.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code
README.md os_cpu_c.c main.c os_core.c ucos_ii.h main.c (HEAD) ↔ main.c (b8e381c) X
c > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Microsoft > V
140 0,
141 &Task2_STK[TASK_STACKSIZE - 1],
142 TASK2_PRIORITY,
143 TASK2_ID,
144 &Task2_STK[0],
145 TASK_STACKSIZE,
146 0,
147 (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));
148
149- if(kevin_task_num >= 3) // kevin
150- OSTaskCreateExt(task3,
151- 0,
152- &Task3_STK[TASK_STACKSIZE - 1],
153- TASK3_PRIORITY,
154- TASK3_ID,
155- &Task3_STK[0],
156- TASK_STACKSIZE,
157- 0,
158- (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));
159-
160- OSTimeSet(0); // kevin reset time
161 OSStart(); /* Start multitasking (i.e. give control to u
```

```
Terminal  Help  main.c (HEAD) ↔ main.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code

README.md  os_cpu.c  main.c  os_core.c  ucos_ii.h  main.c (HEAD) ↔ main.c (b8e381c) X

c: > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Microsoft > Win
...
201     while (DEF_TRUE) {                                     /* Task body, always written as an infinite loop */
202         OSTimeDlyHMSM(0u, 0u, 1u, 0u);
203         APP_TRACE_DBG(("Time: %d\n\r", OSTimeGet()));
204     }
205 }
206
207 void task1(void* p_arg) {
208     (void)p_arg;
209     while (1) {
210         //printf("Hello from task1\n");
211         while (1); // kevin 讓他一直卡在裡面 靠OSintexit來切
212     }
213 }
214
215 void task2(void* p_arg) {
216     (void)p_arg;
217     while (1) {
218         //printf("Hello from task2\n");
219         while (1); // kevin 讓他一直卡在裡面 靠OSintexit來切
220     }
221 }
222
223 void task3(void* p_arg) {
224     (void)p_arg;
225     while (1) {
226         //printf("Hello from task3\n");
227         while (1); // kevin 讓他一直卡在裡面 靠OSintexit來切
228     }
229 }
```

os_core.c functions 大部分都寫在這

```
Terminal Help os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code
c > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Micrium > Software > uCOS-II > Source
2174- /////////////////////////////////////////////////// kevin print function ///////////////////////////////////
2175- void Kevin_StartContextSwitches(void) {
2176-     // kevin linked list
2177-     printf("===== TCB linked list =====\n");
2178-     printf("Task \t Prev_TCB_addr \t TCB_addr \t Next_TCB_addr\n");
2179-     for(int i = kevin_task_num; i > 0; i--)
2180-         printf("%d \t %08X \t %08X \t %08X\n", i, OSTCBPrioTbl[i]->OSTCBPrioTbl[i], OSTCBPrioTbl[i]->OSTCBNext);
2181-     printf("%d \t %08X \t %08X \t %08X\n", 63, OSTCBPrioTbl[63]->OSTCBPrioTbl[63], OSTCBPrioTbl[63]->OSTCBNext);
2182-
2183-     // kevin rm title
2184-     printf("\nTick \t Event \t \t CurrentTask ID \t NextTask ID \t ResponseTime \t # of ContextSwitch \n"); // kevin title
2185- }
```

```
File Edit Selection View Go Run Terminal Help os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code
c > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Micrium > Software > uCOS-II > S
2187- void Kevin_ContextSwitches(void) {
2188-     // context counting
2189-     for(int i = 1; i <= kevin_task_num; i++) // 每個 task
2190-     {
2191-         if(kevin_arr_task_periodic[i].work < kevin_arr_task_periodic[i].execution || OSPrioHighRdy == i) // 當這task開始做
2192-         {
2193-             kevin_arr_task_periodic[i].context++; // counter
2194-             // printf("task:%d work:%d\n", i, kevin_arr_task_periodic[i].work);
2195-         }
2196-     }
2197-
2198-     // print
2199-     Kevin_print();
2200- }
2201- }
```

```
File Edit Selection View Go Run Terminal Help os_core.c - 嵌入式作業系統實作HW - Visual Studio Code
20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Micrium > Software > uCOS-II > Source > os_core.c > Kevin_OS_SchedNew(void)
2199-     Kevin_print();
2200- }
2201-
2202- void Kevin_OS_SchedNew(void) {
2203-     // counter
2204-     if (OSPrioHighRdy <= kevin_task_num && OSTime != 0) // 排除 idle 跟開始
2205-         kevin_arr_task_periodic[OSPrioHighRdy].work--; // 計算execution被做掉了幾次
2206-
2207-     // periodic plus work
2208-     for(int i = 1; i <= kevin_task_num; i++) // 每個 task
2209-     {
2210-         kevin_arr_task_periodic[i].response++; // counter 紀錄從週期開始響應時間
2211-
2212-         // periodic plus work
2213-         int kevin OSTime_arrival = (OSTime - kevin_arr_task_periodic[i].arrival); // arrival
2214-         if(kevin OSTime_arrival >= 0 && kevin OSTime_arrival % kevin_arr_task_periodic[i].period == 0) // 排除0取餘數 找到週期
2215-         {
2216-             if(OSPrioHighRdy == i && kevin_arr_task_periodic[i].work == 0){ // 做完了但接著做 不會進context 所以在這job++ 印出來
2217-                 printf("%d \t Completion \t task(%d)(%) \t \t task(%d)(%) \t \t %d \t \t %d \n",
2218-                     , OSTime, OSPrioHighRdy, kevin_arr_task_periodic[OSPrioHighRdy].job++, OSPrioHighRdy, kevin_arr_task_periodic[OSPrioHighRdy].job, kevin
2219-                     // kevin_arr_task_periodic[OSPrioHighRdy].job++; // 或是不列印就job++
2220-                 }
2221-             if(kevin_arr_task_periodic[i].work != 0){ // 發工作時發現miss
2222-                 printf("%d \t MissDeadline \t task(%d)(%) \t \t -----\n", OSTime, i, kevin_arr_task_periodic[i].job);
2223-                 while (1);
2224-             }
2225-             kevin_arr_task_periodic[i].work += kevin_arr_task_periodic[i].execution; // 發工作
2226-             kevin_arr_task_periodic[i].response = 0;
2227-             kevin_arr_task_periodic[i].context = 0;
2228-         }
2229-         // printf("task:%d OSTime:%d %d work:%d\n", i, (OSTime - kevin_arr_task_periodic[i].arrival), (OSTime - kevin_arr_task_periodic[i].arrival) %
2230-     }
2231-
2232-     // find OSPrioHighRdy
2233-     OSPrioHighRdy = 63; // find OSPrioHighRdy 先假設沒人要做事
2234-     for(int i = 1; i <= kevin_task_num; i++) // 每個 task
2235-     {
2236-         if (kevin_arr_task_periodic[kevin_arr_short[i]].work != 0) // 從 週期短 高優先 開始找有事做的
2237-         {
2238-             OSPrioHighRdy = kevin_arr_short[i];
2239-             break;
2240-         }
2241-     }
2242-
2243-     // printf("OS_SchedNew OSTime:%d OSPrioHighRdy:%d work:%d \n", OSTime, OSPrioHighRdy, kevin_arr_task_periodic[OSPrioHighRdy].work); // kevin
2244-
2245- void Kevin_OSInit(void){
2246-
2247- }
```


os_core.c function 對應位置

```
Terminal Help os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code
README.md os_cpu_c.c main.c os_core.c ucos_ii.h os_core.c (HEAD) ↔ os_core.c (b8e381c) X
c > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Micrium > Software > ucos_ii > uc
88
89 static void OS_SchedNew(void);
90
91- //////////////////////////////////////////////////// kevin function ////////////////////////////////////////
92 void Kevin_StartContextSwitches(void);
93 void Kevin_ContextSwitches(void);
94- void Kevin_OS_SchedNew(void);
95- void Kevin_OSInit(void);
96- void Kevin_print(void);
97- ////////////////////////////////////////
```

```
Terminal Help os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code
README.md os_cpu_c.c main.c os_core.c ucos_ii.h os_core.c (HEAD) ↔ os_core.c (b8e381c) X
c > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Micrium > Software > ucos_ii > uc
590
591 void OSInit (void)
592 {
593- // kevin init
594- Kevin_OSInit();
595-
596- #if OS_TASK_CREATE_EXT_EN > 0u
```

```
Terminal Help os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code
README.md os_cpu_c.c main.c os_core.c ucos_ii.h os_core.c (HEAD) ↔ os_core.c (b8e381c) X
c > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Micrium > Software > ucos_ii > uc
704
705 void OSIntExit (void)
706 {
707
708 #if OS_CRITICAL_METHOD == 3u /* Allocate storage for CPU status register */
709 OS_CPU_SR cpu_sr = 0u;
710 #endif
711
712
713
714 if (OSRunning == OS_TRUE) {
715 OS_ENTER_CRITICAL();
716 if (OSIntNesting > 0u) { /* Prevent OSIntNesting from wrapping */
717 OSIntNesting--;
718 }
719 if (OSIntNesting == 0u) { /* Reschedule only if all ISRs complete ... */
720 if (OSLockNesting == 0u) { /* ... and not locked. */
721 OS_SchedNew();
722 OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy];
723
724 if (OSPrioHighRdy != OSPrioCur) { /* No Ctx Sw if current task is highest rdy */
725 #if OS_TASK_PROFILE_EN > 0u
726 OSTCBHighRdy->OSTCBCtxSwCtr++; /* Inc. # of context switches to this task */
727 #endif
728
729 OSctxSwCtr++; /* Keep track of the number of ctx switches */
730
731 #if OS_TASK_CREATE_EXT_EN > 0u
732 #if defined(OS_TLS_TBL_SIZE) && (OS_TLS_TBL_SIZE > 0u)
733 OS_TLS_TaskSw();
734 #endif
735 #endif
736
737 OS_TRACE_ISR_EXIT_TO_SCHEDULER();
737- Kevin_ContextSwitches(); // Kevin ContextSwitches
738- OSIntNesting++; /* Perform interrupt level context switch */
```



```
Terminal Help os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code
os_cpu.c cc main.c os_core.c uclos_ii.h os_core.c (HEAD) ↔ os_core.c (b8e381c) X
c > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Micrium > Software
1715
1716 */
1717
1718 void OS_Sched (void)
1719 {
1720
1721 #if OS_CRITICAL_METHOD == 3u /* Allocate storage for CPU status register */
1722     OS_CPU_SR cpu_sr = 0u;
1723 #endif
1724
1725
1726
1727     OS_ENTER_CRITICAL();
1728     if (OSIntNesting == 0u) { /* Schedule only if all ISRs done and ... */
1729
1730         if (OSLockNesting == 0u) { /* ... scheduler is not locked */
1731             OS_SchedNew();
1732
1733             OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy];
1734
1735
1736             if (OSPrioHighRdy != OSPrioCur) { /* No Ctx Sw if current task is highest rdy */
1737 #if OS_TASK_PROFILE_EN > 0u
1738                 OSTCBHighRdy->OSTCBCtxSwCtr++; /* Inc. # of context switches to this task */
1739             #endif
1740
1741             #endif
1742             OSCtxSwCtr++; /* Increment context switch counter */
1743 #if OS_TASK_CREATE_EXT_EN > 0u
1744 #if defined(OS_TLS_TBL_SIZE) && (OS_TLS_TBL_SIZE > 0u)
1745             OS_TLS_TaskSw();
1746 #endif
1747 #endif
1748 Kevin_ContextSwitches(); // Kevin ContextSwitches
1749 OS_TASK_SW(); /* Perform a context switch */
```

```
Terminal Help os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code
os_cpu.c cc main.c os_core.c uclos_ii.h os_core.c (HEAD) ↔ os_core.c (b8e381c) X
c > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Micrium > Software
1774 */
1775
1776 static void OS_SchedNew (void)
1777 {
1778 #if OS_LOWEST_PRIO <= 63u /* See if we support up to 64 tasks */
1779     INT8U y;
1780
1781     y = OSUnMapTbl[OSRdyGrp];
1782     // OSPrioHighRdy = (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]); // Kevin
1783
1784     Kevin_OS_SchedNew(); // Kevin find OSPrioHighRdy
1785
1786 #endif
```

```
Terminal Help os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code
os_cpu.c cc main.c os_core.c uclos_ii.h os_core.c (HEAD) ↔ os_core.c (b8e381c) X
c > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Micrium > Software
883 */
884
885 void OSStart (void)
886 {
887     if (OSRunning == OS_FALSE) {
888         OS_SchedNew(); /* Find highest priority's task priority number */
889         OSPrioCur = OSPrioHighRdy;
890         OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy]; /* Point to highest priority task ready to run */
891         OSTCBCur = OSTCBHighRdy;
892         Kevin_StartContextSwitches();
893         OSStartHighRdy(); /* Execute target specific code to start task */
894     }
895 }
```