The system flow and the explanation of the process(functions). (45%)

1. 進入main.c 。
2. 執行int main (void){} 。
3. 建立task。
4. 透過OSTimeSet(0); 將OSTime歸零。
5. 執行OSStart()。
6. 進入os\_core.c。
7. 執行void OSStart (void){} 。
8. 如果OSRunning處於等待步驟命令時開始執行OS\_SchedNew()。
9. 如果OSPrioCur==0 也就是還沒有開始執行，

則列印"Tick\t Form Task\t To Task\n"以及OSTimeGet(), "\*\*\*\*\*\*\*\*", OSPrioHighRdy；

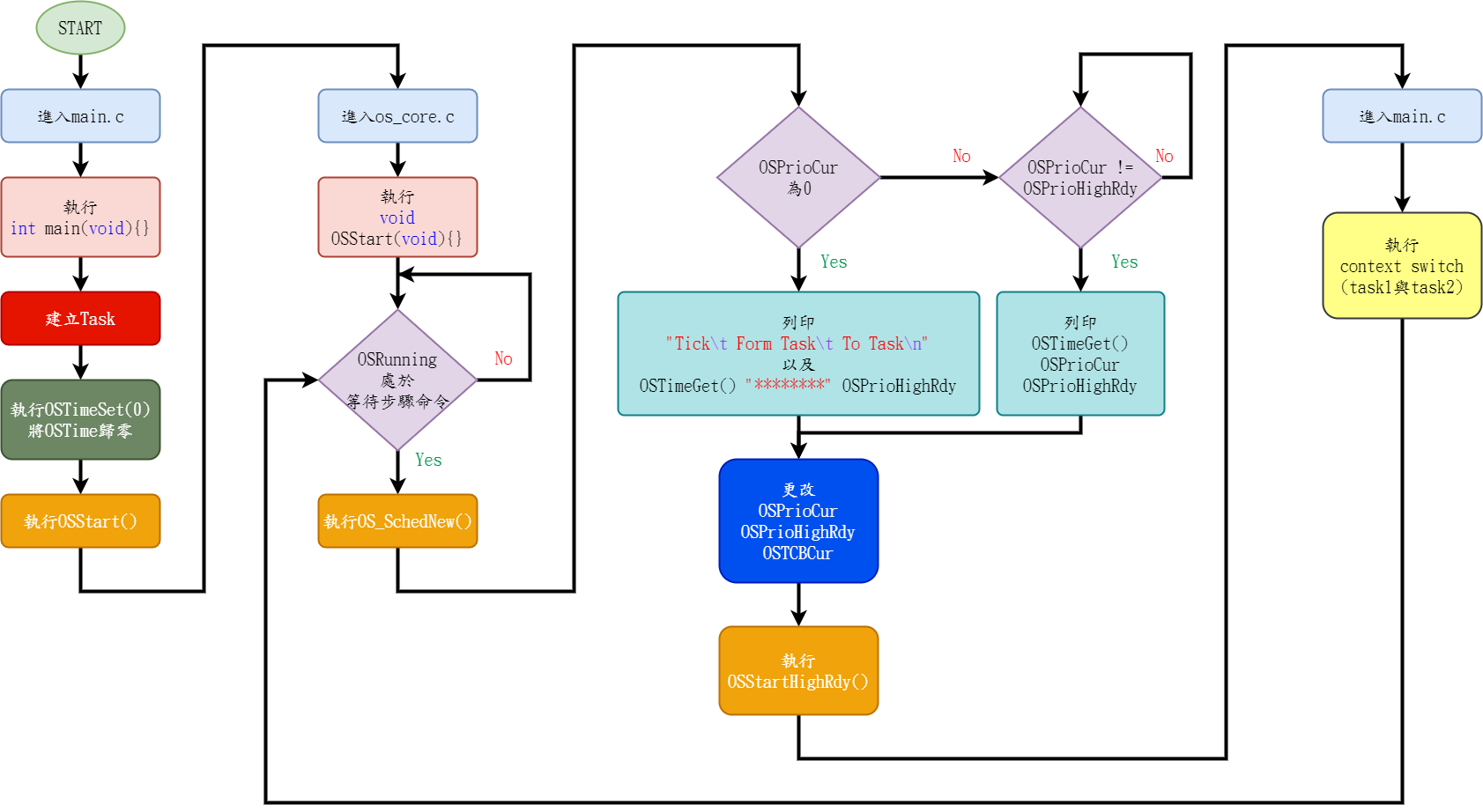
否則如果OSPrioCur != OSPrioHighRdy[現在的task不是最高優先權的]，

則列印OSTimeGet(), OSPrioCur, OSPrioHighRdy。

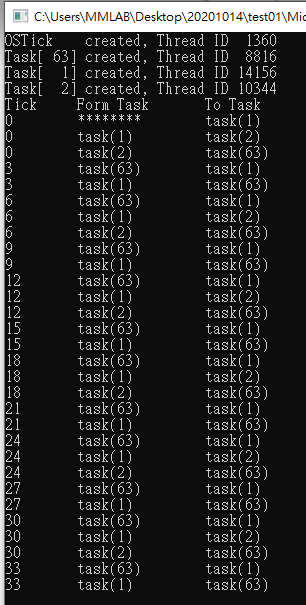
1. 執行完OS\_SchedNew(); 之後更改 OSPrioCur、OSTCBHighRdy、OSTCBCur三個變數。
2. 執行OSStartHighRdy ()。
3. 進入main.c 。
4. 執行context switch

(void task1(void\* p\_arg) {} 🡨🡪void task2(void\* p\_arg) {})。

1. task1與task2的Delay時間分別為3與6個單位，故此程式則會依照OSTime為3與6的倍數時才執行。



The screenshot of the result. (10%)

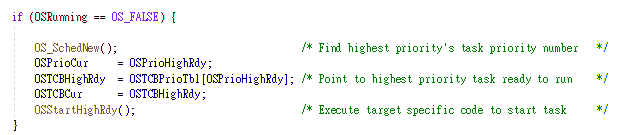


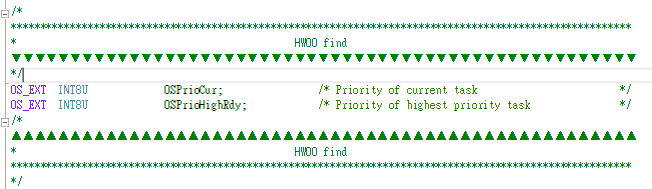
A report that describes your implementation (please attach the screenshot of the code and MARK the modified part). (45%)

1. 先從OSStart();的程式開始查看定義

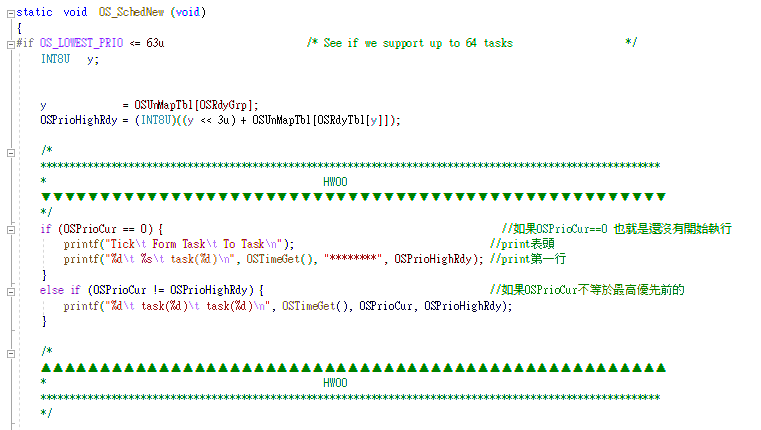


1. 在if (OSRunning == OS\_FALSE) {}中看見了本次作業需要的參數OSPrioCur與OSTCBHighRdy





1. 開始一行一行觀看if (OSRunning == OS\_FALSE) {}中函數的定義
2. 從static void OS\_SchedNew (void){}中可以發現OSTCBHighRdy的更新位置在這裡



1. 再來就是經由不斷嘗試得出最精簡的code以及符合題目要求的排版
2. END