Ucosii宏定义的细节

1.预编译阶段把所有#include ”\*\*\*.h“ （“”与<>的区别这里就不说了）用\*\*\*.h的内容来替换了， 所以之后就没有.h了，所有.h的内容都已经包含进了需要它们的.c中

2.生成最后的exe文件是由编译、链接两步完成的， 编译是源代码生成obj二进制目标文件的过程， 注意一个源代码文件（指.c，而非.h，.h已经被包含进.c中了）生成一个obj文件，由于编译是独立的， 所以在两个独立的编译单元里是可以有重名的函数的，例如a.c中可以有一个void fun(); b.c中可同时有一个void fun(); 这点十分重要， 大家可以试一下并且理解清楚

3.编译期间， 我们只要声明了的东西就能使用， 而无需它的定义， 声明可以重复，extern在编译时是告诉该编译单元该变量的定义在别的编译单元里，相当于声明，链接时，定义在整个程序中有且仅有一份，例如如下代码，编译可通过，但链接时失败。

4.#ifndef #define #endif 只在一个.c文件中有效，即在一个.c文件中define了一个宏，那么在另外一个.c中，这个宏是没有被define过的。在下面的例子中，即使使用了宏，也会报错，因为在两个.c中都包含了.h，而.h中有两个定义（而不是声明），造成了linking时的重定义错误

//main.c

#include <a.h>

int main()

{

}

//a.h

#ifndef A\_H\_

#define A\_H\_

void fun(void){};

int x;

#endif

//ac.c

#include <a.h>

void test()

{

}

Build之后发现如下linking错误

5.#ifndef不能防止上述这种情况，那么应该用于哪里呢？有时候可能会在一个.c文件中包含了两次相同的.h文件，或者包含了两个不同的.h，但这两个.h又互相包含，这个时候#ifndef 就会起作用了

6.为了彻底防止错误的出现，.h应该不产生代码，不用于定义，只用于宏定义、声明等

7.在Keil的设置里面，预处理Define下面写的宏会在每个.c文件中都define过

8.uCOS中的define技巧：

在os\_core.c中，有

#ifndef OS\_MASTER\_FILE

#define OS\_GLOBALS

#include <ucos\_ii.h>

#endif

在其他的.c中，有

#ifndef OS\_MASTER\_FILE

#include <ucos\_ii.h>

#endif

然后在ucos\_ii.h中，有

#ifdef OS\_GLOBALS

#define OS\_EXT

#else

#define OS\_EXT extern

#endif

我们可以看到，在os\_core.c中，ucos\_ii.h中带OS\_EXT前缀的都变成了定义，而在其他.c中，带OS\_EXT前缀的都是声明，这样就完成了一个.h即实现了定义，又实现了声明。

在uCOS中，还可以看到如下的宏使用方式，利用宏去选择编译哪段代码，这样比if else要好，if else还要占用空间。之前编程一直靠人工记，测试这块板子时，注释这句；测试那块板子时，注释那句。以后应该多用宏的方式去选择。

#if OS\_TIME\_GET\_SET\_EN > 0

OS\_EXT volatile INT32U OSTime; /\* Current value of system time (in ticks) \*/

#endif