# 目录

[目录 1](#_Toc17733082)

[第2章 STM32F103C8-UCOS操作系统 2](#_Toc17733083)

[2.1 操作系统介绍 2](#_Toc17733084)

[2.1.1 操作系统简介 2](#_Toc17733085)

[2.1.2 常见的操作系统 2](#_Toc17733086)

[2.2 UCOS操作系统介绍 2](#_Toc17733087)

[2.2.1 操作系统的调度原则 2](#_Toc17733088)

[2.2.2 操作系统的程序结构 2](#_Toc17733089)

[2.2.3 操作系统的任务调度 4](#_Toc17733090)

[2.2.4 操作系统的任务状态 4](#_Toc17733091)

[2.2.5 操作系统的任务中断 5](#_Toc17733092)

[2.2.6 UCOS操作系统组成结构 5](#_Toc17733093)

[2.3 UCOS操作系统移植 5](#_Toc17733094)

[2.3.1 获取UCOS操作系统源码 5](#_Toc17733095)

[2.3.2 了解内部相关文件 6](#_Toc17733096)

[2.3.3 移植需要文件 6](#_Toc17733097)

[2.3.4 新建一个带UCOS的项目工程 7](#_Toc17733098)

[2.3.5 修改UCOS源码中的内容 8](#_Toc17733099)

[2.3.6 验证UCOS操作系统移植 12](#_Toc17733100)

[2.4 UCOS操作系统应用 15](#_Toc17733101)

# STM32F103C8-UCOS操作系统

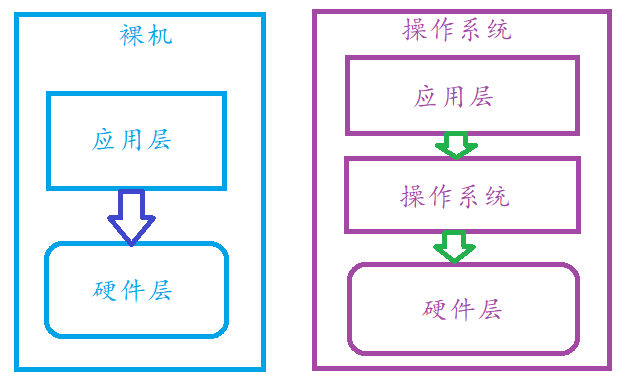
## 操作系统介绍

### 操作系统简介

1. 什么是操作系统。

操作系统是管理和控制计算机硬件与软件资源的计算机程序，是直接运行在“裸机”上的最基本的系统软件，任何其他软件都必须在操作系统的支持下才能运行。介于APP和硬件之间。

1. 操作系统在程序中的结构



### 常见的操作系统

常见的操作系统：

IOS、安卓、Windows、Linux、塞班、UCOS、FreeRTOS、RT-T(国产)、Vxworks、eCos、LiteOS、AWorks等等。

1. 实时操作系统(RTOS)

UCOS（学习免费，商用需要注册）、FreeRTOS、RT-T(国产)、Vxworks、eCos

会给每个任务一个基础的时间长度，根据任务的需求适当调整任务时长。任务与任务直接切换都有相应的规则。

任务调度原则不是以时间为原则。以任务优先级为原则。

1. 分时操作系统

Windows----2000以前、Linux----2.6以前

给每一个任务分配一个固定的时长，如果这个任务在固定的时长内已经空闲，分时操作系统还是会给该任务规定好的时长。

任务调度原则：以时间为原则。

1. 半实时操作系统

Windows----7/8/10、Linux----2.6以后

## UCOS操作系统介绍

### 操作系统的调度原则

UCOS操作系统调度原则是：以任务优先级为调度原则。任务优先级编号越小的，任务优先级越高。

注意点：每个任务只有一个优先级，不同任务优先级编号不能一致。

### 操作系统的程序结构

1. 没有操作系统时

整个裸机项目工程中有且仅有一个死循环，这个死循环存在main函数中。其它地方可以有死循环但是必须要有退出条件。

|  |
| --- |
| int main(void)  {  //各种硬件初始化    while(1)  {  //需要循环实现的功能代码  }  } |

1. 有操作系统时

整个操作系统项目工程中可以有多个死循环，每个任务中有一个死循环，main函数中无死循环。

注意点：操作系统在main中必须完成两点：①设定好操作系统心跳节拍；②创建一个任务(该任务必须创建成功)。

|  |
| --- |
| int main(void)  {  //初始化操作系统  //启动操作系统  }  void app\_one\_task(void \*para)  {  //初始化    while(1)  {  //重复执行的功能代码  }  }  void app\_two\_task(void \*para)  {  //初始化    while(1)  {  //重复执行的功能代码  }  }  void app\_three\_task(void \*para)  {  //初始化    while(1)  {  //重复执行的功能代码  }  }  void app\_four\_task(void \*para)  {  //初始化    while(1)  {  //重复执行的功能代码  }  } |

### 操作系统的任务调度

任务调度：指的是什么时候会发生任务调度。

两种情况下会发生任务调度。

1. 操作系统的心跳节拍到达。心跳节拍称为一个Tick。UCOS中有一个专用的操作系统心跳节拍函数：OSTimeTick();
2. 操作系统中的程序代码调用OS\_Sched();该函数时会产生任务调度。OS\_Sched()：称为任务调度函数。

注意：任务调度函数我们不能够直接调用，只能是操作系统调用。程序员调用该函数会使得UCOS操作系统崩溃。

注意点①：操作系统发生任务调度不一定切换任务；任务切换一定需要产生任务调度。

注意点②：操作系统中高优先级任务必须要有释放CPU控制权函数调用。

### 操作系统的任务中断

中断状态中需要注意以下几点：

1. 从运行状态进入中断状态必须要调用函数OSIntEnter()；该函数的功能是用来告诉操作系统当前程序进入中断。
2. 从中断状态退出后进入运行状态之前必须调用函数OSIntExit()；该函数的功能是用来告诉操作系统当前程序退出中断。
3. 程序从中断中退出，操作系统返回到任务中去，这时不一定返回到原来的任务中。

### 操作系统的任务状态

UCOS操作系统中，任务状态有五种：停止/休眠、准备就绪、运行、挂起/等待、中断。只有当任务进入到运行状态才可以执行任务中的程序代码。

停止：任务被删除。

休眠：当前没有任何一个任务准备就绪，运行状态中没有任务执行。

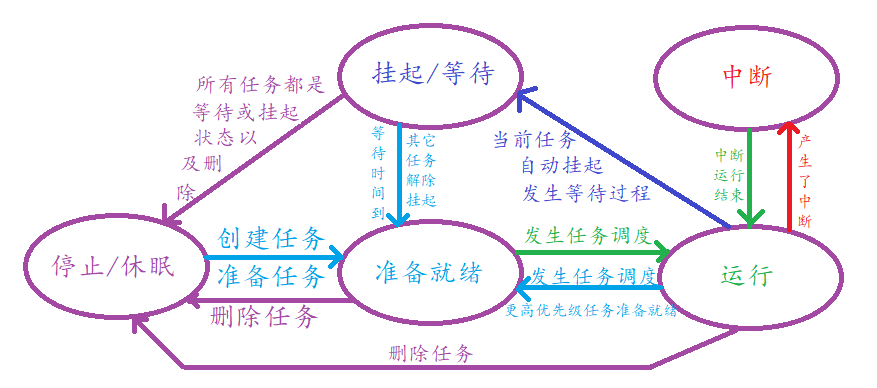
准备就绪：任务运行需要满足的条件都准备完成。

运行：任务正在运行。

挂起：任务需要等待某一个条件发生，其他任务可以将这个任务挂起。

等待：延时相关的。

中断：程序发生了中断事件，CPU要去处理中断服务函数，中断操作系统。



注意点：运行态来源只能是准备就绪态。

两种情况下会发生任务调度。

1. 操作系统的心跳节拍到达。心跳节拍称为一个Tick。UCOS中有一个专用的操作系统心跳节拍函数：OSTimeTick();
2. 操作系统中的程序代码调用OS\_Sched();该函数时会产生任务调度。OS\_Sched()：称为任务调度函数。

注意：任务调度函数我们不能够直接调用，只能是操作系统调用。程序员调用该函数会使得UCOS操作系统崩溃。

注意点①：操作系统发生任务调度不一定切换任务；任务切换一定需要产生任务调度。

注意点②：操作系统中高优先级任务必须要有释放CPU控制权函数调用。

### UCOS操作系统组成结构

组成结构：任务控制块、任务函数、任务栈、任务优先级。

任务控制块：包含了整个任务需要的所有内存空间，由操作系统自动分配。任务控制块包含：任务函数、任务栈、任务优先级

任务函数：任务需要执行的所有程序代码

任务栈：用来保存任务的运行情况(程序运行到哪一行，程序的临时变量值等等)

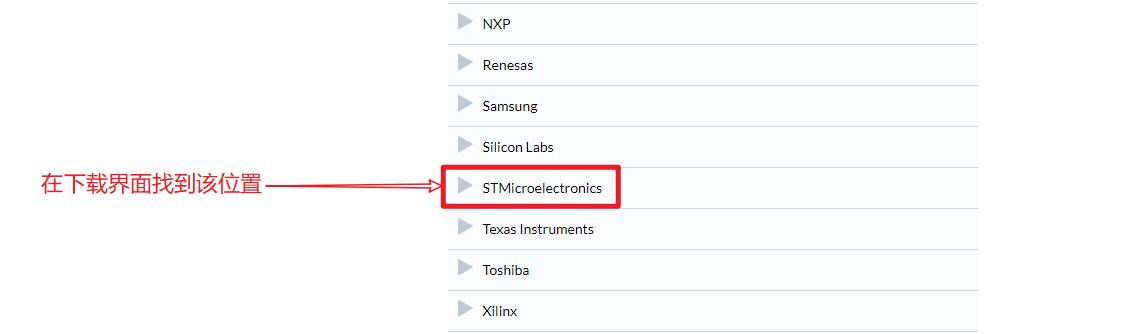
任务优先级：区分每一个任务，任务能够被执行的优先情况

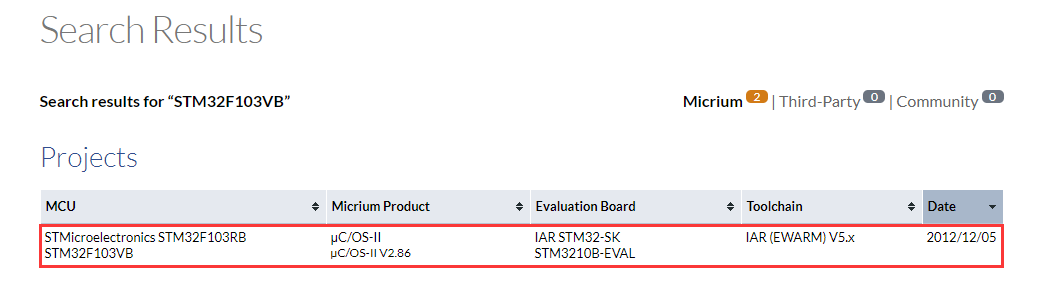
## UCOS操作系统移植

### 获取UCOS操作系统源码

去官网找；官网链接：<https://www.micrium.com/>









### 了解内部相关文件





### 移植需要文件

1. “Micrium\Software\uCOS-II\Source”下所有文件(11个文件)



1. “Micrium\Software\uCOS-II\Ports\ARM-Cortex-M3\Generic\RealView”下所有文件(4个文件)



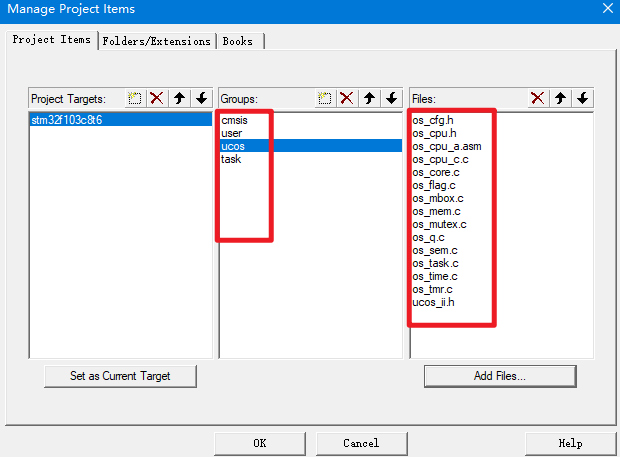
1. “Micrium\Software\EvalBoards\ST\STM3210B-EVAL\RVMDK\OS-Probe”下的os\_cfg.h文件

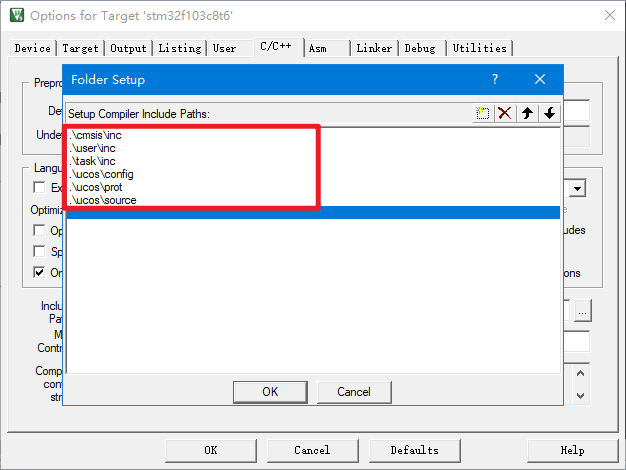


拷贝上述文件到工程中。

### 新建一个带UCOS的项目工程

1. 拷贝寄存器版本的示例工程
2. 在示例工程上新建一个ucos文件夹，在ucos文件夹中建立：config文件夹(存放配置ucos操作系统内核文件)、prot文件夹(存放ucos操作系统的接口配置文件)、source文件夹(存放ucos操作系统的源码)
3. 拷贝上述需要移植的文件到对应的文件夹下
4. 在示例工程上建立一个task文件夹(用来存储后面所写的任务功能代码)；在task文件夹下建立：inc文件夹和src文件夹。
5. 打开工程，添加对应的文件到工程中(新增加ucos相关的文件)，添加头文件路径(新增加ucos相关的头文件路径)





完成上述操作，工程建立与文件添加完成。

### 修改UCOS源码中的内容

#### 需要修改prot文件夹中内容

1. 修改os\_cpu.h文件

|  |
| --- |
| 需要屏蔽下述函数的声明：  void OS\_CPU\_PendSVHandler(void);  /\* See OS\_CPU\_C.C\*/  void OS\_CPU\_SysTickHandler(void);  void OS\_CPU\_SysTickInit(void);  /\* See BSP.C\*/  INT32U OS\_CPU\_SysTickClkFreq(void); |

1. 修改os\_cpu\_c.c文件

|  |
| --- |
| 需要屏蔽以下程序：  void OS\_CPU\_SysTickHandler (void)  {  OS\_CPU\_SR cpu\_sr;  OS\_ENTER\_CRITICAL(); /\* Tell uC/OS-II that we are starting an ISR\*/  OSIntNesting++;  OS\_EXIT\_CRITICAL();  OSTimeTick(); /\* Call uC/OS-II's OSTimeTick()\*/  OSIntExit(); /\* Tell uC/OS-II that we are leaving the ISR\*/  }  void OS\_CPU\_SysTickInit (void)  {  INT32U cnts;  cnts = OS\_CPU\_SysTickClkFreq() / OS\_TICKS\_PER\_SEC;  OS\_CPU\_CM3\_NVIC\_ST\_RELOAD = (cnts - 1);  /\* Enable timer. \*/  OS\_CPU\_CM3\_NVIC\_ST\_CTRL |= OS\_CPU\_CM3\_NVIC\_ST\_CTRL\_CLK\_SRC | OS\_CPU\_CM3\_NVIC\_ST\_CTRL\_ENABLE;    /\* Enable timer interrupt. \*/  OS\_CPU\_CM3\_NVIC\_ST\_CTRL |= OS\_CPU\_CM3\_NVIC\_ST\_CTRL\_INTEN;  } |

1. 修改os\_cpu\_a.asm文件





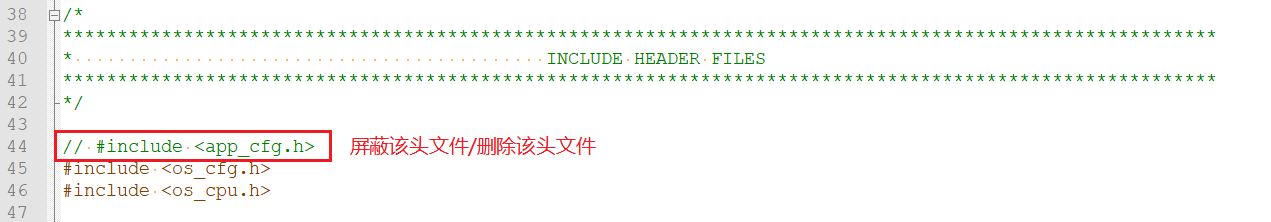
#### 需要修改config文件夹中内容

1. 修改os\_cfg.h文件

|  |
| --- |
| /\* ---------------------- MISCELLANEOUS ----------------------- \*/  #define OS\_APP\_HOOKS\_EN 0 /\* Application-defined hooks are called from the uC/OS-II hooks \*/  #define OS\_ARG\_CHK\_EN 1 /\* Enable (1) or Disable (0) argument checking\*/  #define OS\_CPU\_HOOKS\_EN 1 /\* uC/OS-II hooks are found in the processor port files\*/  #define OS\_DEBUG\_EN 0 /\* Enable(1) debug variables\*/  #define OS\_EVENT\_MULTI\_EN 0 /\* Include code for OSEventPendMulti()\*/  #define OS\_EVENT\_NAME\_SIZE 0 /\* Determine the size of the name of a Sem, Mutex, Mbox or Q \*/  #define OS\_LOWEST\_PRIO 63 /\* Defines the lowest priority that can be assigned ... \*/  /\* ... MUST NEVER be higher than 254!\*/  #define OS\_MAX\_EVENTS 10 /\* Max. number of event control blocks in your application\*/  #define OS\_MAX\_FLAGS 5 /\* Max. number of Event Flag Groups in your application\*/  #define OS\_MAX\_MEM\_PART 0 /\* Max. number of memory partitions\*/  #define OS\_MAX\_QS 5 /\* Max. number of queue control blocks in your application \*/  #define OS\_MAX\_TASKS 15 /\* Max. number of tasks in your application, MUST be >= 2\*/  #define OS\_SCHED\_LOCK\_EN 1 /\* Include code for OSSchedLock() and OSSchedUnlock()\*/  #define OS\_TICK\_STEP\_EN 1 /\* Enable tick stepping feature for uC/OS-View\*/  #define OS\_TICKS\_PER\_SEC 200 /\* Set the number of ticks in one second\*/  /\* --------------------- TASK STACK SIZE ---------------------- \*/  #define OS\_TASK\_TMR\_STK\_SIZE 128 /\* Timer task stack size (# of OS\_STK wide entries)\*/  #define OS\_TASK\_STAT\_STK\_SIZE 128 /\* Statistics task stack size (# of OS\_STK wide entries) \*/  #define OS\_TASK\_IDLE\_STK\_SIZE 128 /\* Idle task stack size (# of OS\_STK wide entries)\*/  /\* --------------------- TASK MANAGEMENT ---------------------- \*/  #define OS\_TASK\_CHANGE\_PRIO\_EN 1 /\* Include code for OSTaskChangePrio() \*/  #define OS\_TASK\_CREATE\_EN 1 /\* Include code for OSTaskCreate() \*/  #define OS\_TASK\_CREATE\_EXT\_EN 1 /\* Include code for OSTaskCreateExt() \*/  #define OS\_TASK\_DEL\_EN 1 /\* Include code for OSTaskDel() \*/  #define OS\_TASK\_NAME\_SIZE 16 /\* Determine the size of a task name \*/  #define OS\_TASK\_PROFILE\_EN 1 /\* Include variables in OS\_TCB for profiling \*/  #define OS\_TASK\_QUERY\_EN 1 /\* Include code for OSTaskQuery() \*/  #define OS\_TASK\_STAT\_EN 1 /\* Enable (1) or Disable(0) the statistics task \*/  #define OS\_TASK\_STAT\_STK\_CHK\_EN 1 /\* Check task stacks from statistic task \*/  #define OS\_TASK\_SUSPEND\_EN 1 /\* Include code for OSTaskSuspend() and OSTaskResume()\*/  #define OS\_TASK\_SW\_HOOK\_EN 1 /\* Include code for OSTaskSwHook() \*/  /\* ----------------------- EVENT FLAGS ------------------------ \*/  #define OS\_FLAG\_EN 1 /\* Enable (1) or Disable (0) code generation for EVENT FLAGS \*/  #define OS\_FLAG\_ACCEPT\_EN 1 /\* Include code for OSFlagAccept() \*/  #define OS\_FLAG\_DEL\_EN 1 /\* Include code for OSFlagDel() \*/  #define OS\_FLAG\_NAME\_SIZE 16 /\* Determine the size of the name of an event flag group \*/  #define OS\_FLAG\_QUERY\_EN 1 /\* Include code for OSFlagQuery() \*/  #define OS\_FLAG\_WAIT\_CLR\_EN 1 /\* Include code for Wait on Clear EVENT FLAGS \*/  #define OS\_FLAGS\_NBITS 16 /\* Size in #bits of OS\_FLAGS data type (8, 16 or 32) \*/  /\* -------------------- MESSAGE MAILBOXES --------------------- \*/  #define OS\_MBOX\_EN 1 /\* Enable (1) or Disable (0) code generation for MAILBOXES \*/  #define OS\_MBOX\_ACCEPT\_EN 1 /\* Include code for OSMboxAccept() \*/  #define OS\_MBOX\_DEL\_EN 1 /\* Include code for OSMboxDel() \*/  #define OS\_MBOX\_PEND\_ABORT\_EN 1 /\* Include code for OSMboxPendAbort() \*/  #define OS\_MBOX\_POST\_EN 1 /\* Include code for OSMboxPost() \*/  #define OS\_MBOX\_POST\_OPT\_EN 1 /\* Include code for OSMboxPostOpt() \*/  #define OS\_MBOX\_QUERY\_EN 1 /\* Include code for OSMboxQuery() \*/  /\* --------------------- MEMORY MANAGEMENT -------------------- \*/  #define OS\_MEM\_EN 0 /\* Enable (1) or Disable (0) code generation for MEMORY MANAGER \*/  #define OS\_MEM\_NAME\_SIZE 16 /\* Determine the size of a memory partition name \*/  #define OS\_MEM\_QUERY\_EN 1 /\* Include code for OSMemQuery() \*/  /\* ---------------- MUTUAL EXCLUSION SEMAPHORES --------------- \*/  #define OS\_MUTEX\_EN 1 /\* Enable (1) or Disable (0) code generation for MUTEX \*/  #define OS\_MUTEX\_ACCEPT\_EN 1 /\* Include code for OSMutexAccept() \*/  #define OS\_MUTEX\_DEL\_EN 1 /\* Include code for OSMutexDel() \*/  #define OS\_MUTEX\_QUERY\_EN 1 /\* Include code for OSMutexQuery() \*/  /\* ---------------------- MESSAGE QUEUES ---------------------- \*/  #define OS\_Q\_EN 1 /\* Enable (1) or Disable (0) code generation for QUEUES\*/  #define OS\_Q\_ACCEPT\_EN 1 /\* Include code for OSQAccept() \*/  #define OS\_Q\_DEL\_EN 1 /\* Include code for OSQDel() \*/  #define OS\_Q\_FLUSH\_EN 1 /\* Include code for OSQFlush() \*/  #define OS\_Q\_PEND\_ABORT\_EN 1 /\* Include code for OSQPendAbort() \*/  #define OS\_Q\_POST\_EN 1 /\* Include code for OSQPost() \*/  #define OS\_Q\_POST\_FRONT\_EN 1 /\* Include code for OSQPostFront() \*/  #define OS\_Q\_POST\_OPT\_EN 1 /\* Include code for OSQPostOpt() \*/  #define OS\_Q\_QUERY\_EN 1 /\* Include code for OSQQuery() \*/  /\* ------------------------ SEMAPHORES ------------------------ \*/  #define OS\_SEM\_EN 1 /\* Enable (1) or Disable (0) code generation for SEMAPHORES \*/  #define OS\_SEM\_ACCEPT\_EN 1 /\* Include code for OSSemAccept() \*/  #define OS\_SEM\_DEL\_EN 1 /\* Include code for OSSemDel() \*/  #define OS\_SEM\_PEND\_ABORT\_EN 1 /\* Include code for OSSemPendAbort() \*/  #define OS\_SEM\_QUERY\_EN 1 /\* Include code for OSSemQuery() \*/  #define OS\_SEM\_SET\_EN 1 /\* Include code for OSSemSet() \*/  /\* --------------------- TIME MANAGEMENT ---------------------- \*/  #define OS\_TIME\_DLY\_HMSM\_EN 1 /\* Include code for OSTimeDlyHMSM() \*/  #define OS\_TIME\_DLY\_RESUME\_EN 1 /\* Include code for OSTimeDlyResume() \*/  #define OS\_TIME\_GET\_SET\_EN 1 /\* Include code for OSTimeGet() and OSTimeSet() \*/  #define OS\_TIME\_TICK\_HOOK\_EN 1 /\* Include code for OSTimeTickHook() \*/  /\* --------------------- TIMER MANAGEMENT --------------------- \*/  #define OS\_TMR\_EN 1 /\* Enable (1) or Disable (0) code generation for TIMERS \*/  #define OS\_TMR\_CFG\_MAX 16 /\* Maximum number of timers \*/  #define OS\_TMR\_CFG\_NAME\_SIZE 16 /\* Determine the size of a timer name \*/  #define OS\_TMR\_CFG\_WHEEL\_SIZE 8 /\* Size of timer wheel (#Spokes) \*/  #define OS\_TMR\_CFG\_TICKS\_PER\_SEC 10 /\* Rate at which timer management task runs (Hz) \*/ |

#### 修改source文件夹中内容

1. 修改ucos\_ii.h文件



### 验证UCOS操作系统移植

#### 编译程序代码，解决错误

|  |
| --- |
| ucos\source\os\_tmr.c(890): error: #20: identifier "OS\_TASK\_TMR\_PRIO" is undefined  OS\_TASK\_TMR\_PRIO,  上述错误：没有定义OS\_TASK\_TMR\_PRIO；TASK\_TMR优先级。  解决上述错误的方法：  第一种：利用宏定义定义OS\_TASK\_TMR\_PRIO值(os\_cfg.h文件中定义)  第二种：如果项目中可以不使用操作系统的软件定时器，配置#define OS\_TMR\_EN 0即可(关闭软件定时器)  .\Objects\stm32f103c8t6.axf: Error: L6200E: Symbol PendSV\_Handler multiply defined (by os\_cpu\_a.o and stm32f10x\_it.o).  上述错误问题：多次定义PendSV\_Handler；在os\_cpu\_a.c和stm32f10x\_it.c中重复定义。  解决上述错误问题：关闭stm32f10x\_it.c文件中的PendSV\_Handler定义。 |

#### 编写UCOS操作系统的任务

1. 编写ucos操作系统心跳节拍功能，配置系统滴答定时器

|  |
| --- |
| void systick\_init(void)  {  //清空状态控制寄存器  SysTick->CTRL = 0;  //清空计数器的值  SysTick->VAL = 0;  //配置自动重载值  SysTick->LOAD = 1000 / OS\_TICKS\_PER\_SEC \* 9000;//5ms一个心跳节拍  //配置状态控制寄存器  SysTick->CTRL |= 0X1 << 1;//开启中断使能  SysTick->CTRL |= 0X1 << 0;//开启计数器使能  }  void SysTick\_Handler(void)  {  //告诉操作系统进中断了  OSIntEnter();    //调用操作系统心跳节拍函数  OSTimeTick();    //告诉操作系统出中断了  OSIntExit();  } |

1. 掌握任务创建函数

|  |
| --- |
| INT8U OSTaskCreateExt (void (\*task)(void \*p\_arg),  void \*p\_arg,  OS\_STK \*ptos,  INT8U prio,  INT16U id,  OS\_STK \*pbos,  INT32U stk\_size,  void \*pext,  INT16U opt)  函数功能：利用扩展方式创建一个任务  函数形参：  void (\*task)(void \*p\_arg)：函数指针，指向任务函数的指针，填写任务函数的首地址  void \*p\_arg：任务函数的形参传递，如果任务函数不需要形参，填NULL即可  OS\_STK \*ptos：这个指针指向任务栈顶；填写定义的数组的最后一个元素的地址  INT8U prio：任务的优先级；0~OS\_LOWEST\_PRIO；  INT16U id：随便填写；0~65535；一般填写的值与优先级一致即可  OS\_STK \*pbosL：这个指针指向任务栈底；填写的时候填数字的最开始一个元素地址  INT32U stk\_size：这个是填写任务栈大小；填写的值填数组长度  void \*pext：用来管理TCB的扩展内存，可以保存浮点数据寄存器等；如果不需要填NULL；一般填NULL  INT16U opt：在创建任务的时候可以让操作系统帮忙做的事情。  OS\_TASK\_OPT\_STK\_CHK：帮忙检查任务栈  OS\_TASK\_OPT\_STK\_CLR：帮忙创建任务的时候清空任务栈  OS\_TASK\_OPT\_SAVE\_FP：帮忙保存浮点类型数据  函数返回值：  OS\_ERR\_NONE：任务创建成功  OS\_PRIO\_EXIT：创建的任务优先级已经存在；创建失败  OS\_ERR\_PRIO\_INVALID：创建的任务优先级太低了；创建失败  OS\_ERR\_TASK\_CREATE\_ISR：创建的任务在中断中创建；创建失败 |

1. 掌握main函数编写

|  |
| --- |
| int main(void)  {  uint8\_t error = 0;    //初始化系统滴答定时器，配置系统心跳节拍  systick\_init();    //初始化操作系统  OSInit();    //创建一个启动任务  error = OSTaskCreateExt(start\_task,  0,  &start\_task\_stk[START\_TASK\_STK\_SIZE - 1],  START\_TASK\_PRIO,  START\_TASK\_PRIO,  &start\_task\_stk[0],  START\_TASK\_STK\_SIZE,  0,  OS\_TASK\_OPT\_STK\_CLR | OS\_TASK\_OPT\_STK\_CHK  );  if(error != OS\_ERR\_NONE)  {  //创建启动任务失败  }    //启动操作系统  OSStart();    while(1)//防止操作系统启动失败后程序跑飞  {    }  } |

1. 掌握任务函数编写

|  |
| --- |
| /\*  \*启动任务的任务函数  \*/  void start\_task(void \*p\_arg)  {      while(1)  {    }  } |

1. 创建启动任务相关内容定义

|  |
| --- |
| OS\_STK start\_task\_stk[START\_TASK\_STK\_SIZE];//定义任务栈空间  #define START\_TASK\_STK\_SIZE 128//定义任务栈大小  #define START\_TASK\_PRIO 3//定义任务优先级 |

## UCOS操作系统应用