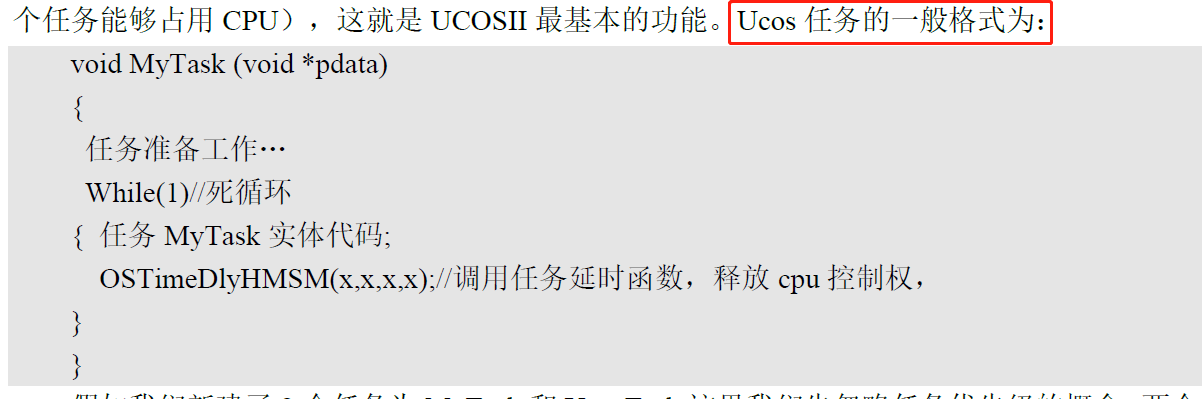
7.1

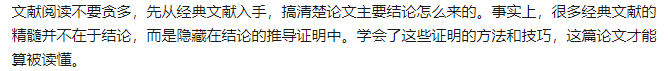
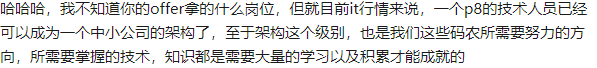
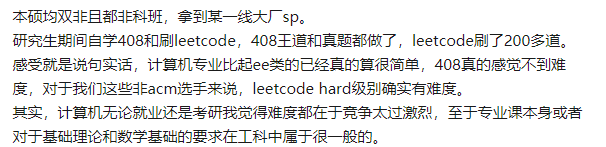
1. 今天调试了一下输出端，原来R62的位置用的是100R电阻，接通后发现输出电压是9V左右——比较低。尝试把100R的电阻换成了10R的电阻，输出的电压是11V。原因猜测：MOS管的导通特性
2. 问题描述：在ucosii环境下建立了一个OUT输出任务。但是在写函数的时候没有采用库函数的形式去写。而是在头文件中用OUT来代替。奇怪的是LED灯端口输出没有问题。但是换成PA0这个端口这样写就出问题了，下载程序后端口并没有反应。原因：高电平设置的时间太短，导致OUT灯闪烁的时间太短，并且闪烁的灯在视野盲区，没看到，错以为程序出错了。
3. 移植ucosii



遇到事情多想一想，不要大惊小怪

1. 明天和海哥讨论一下，我程序需要的细节。把框架敲定，不要遇到以后采取想办法。先查资料。是不是需要文件系统，程序该怎么升级。还有PYTHON 的QT上位机开发该怎么做。通讯协议是什么。为什么要分成两个程序BOOT和APP。多想一下存在的困难，和解决方案。
2. 微信上发消息问

7.2

1. 当你通过更低的复杂度，更好的可读性，去解决一个既定问题；在当程序调通，耗时和耗内存有所降低时，是编程者的高光时刻。
2. 
3. 
4. 
5. 

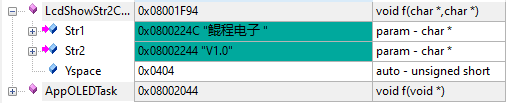
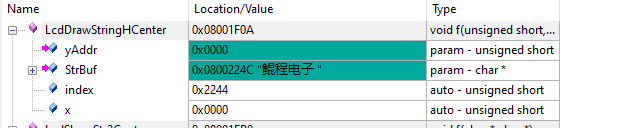
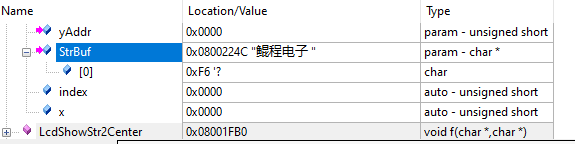
7.6

1. **不同型号的芯片在代码移植的时候要注意其中使用的函数以及参数的变化的变化**，比如在F4芯片里面时钟初始化使用到的函数是APB1而在F1芯片中使用到的函数就不是APB1而是APB2因为在时钟树里面两个不同型号的芯片对应的时钟树是不一样的，所以就不能使用同一个函数来配置。
2. 软件如何仿真，什么时候需要仿真
3. 在OLED模块中的SCL和SDA接口处为什么要接一个上拉电阻。
4. 如何仿真时序
5. 把软件流程写出来

7.7

1. 索引是怎么写的
2. 软件仿真，找视频看
3. 在调试OS核的时候，把OLED的任务优先级写成11，之后在下载验证的时候发现，OLED的任务并没有运行，后来发现原因是优先级太高了，不太合适，可能优先级11上面另有其他任务在默认地运行，所以这两个任务冲突了，把优先级换成23以后发现程序和设想的一样正常运行。在OLED 程序里面函数的调用顺序是有讲究的，如果说初始化程序只调用一次，那么他就只执行一次，下次就不在执行了。所以会出现在下载验证的时候亮起，而重启以后就不再亮了。
4. **指针，数组，枚举以及例程去理解**
5. Switch函数
6. 底层函数有什么区别，哪些文件是必须的，晶振和什么相关，滴答定时器。
7. 建立一个能够直接仿真下去的工程
8. 尝试不同的函数（OLED工程下的）

7.8

1. 在对比查看STM32的底层文件的时候，发现与M3核相关的文件是不用修改的，可以直接调用。比如core\_cm3.c core\_cm3.h 启动文件 stm32f10x.h
2. 中断和时钟需要学习
3. Stm32f10x.h这个文件里面修改晶振大小。
4. 
5. 
6. 
7. 什么时候会用到滴答定时器
8. 在调试的时候出现的一般状况解析
9. 什么情况下使用内部晶振，什么时候下使用外部晶振。
10. 系统代码的运行使用到的是什么晶振
11. 系统的内部晶振是8MHz的但是精度不够
12. Extern
13. 在ucosii中，任务建立必须在while循环里，不然他无法跳出来。
14. 学会在UCOSII中操作,必要的流程要清楚
15. 数据左对齐和右对齐怎么理解
16. 缺省值就是默认值=default
17. 影响ADC采集精度的因素是什么？采用多个通道采集？多通道采集然后做误差处理能不能合并数据，这样就可以减少误差？

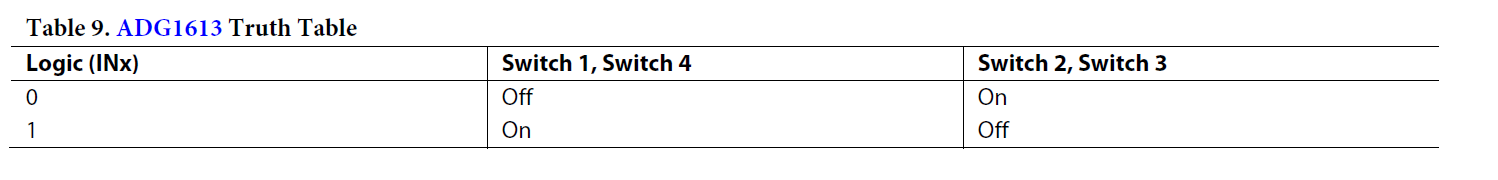
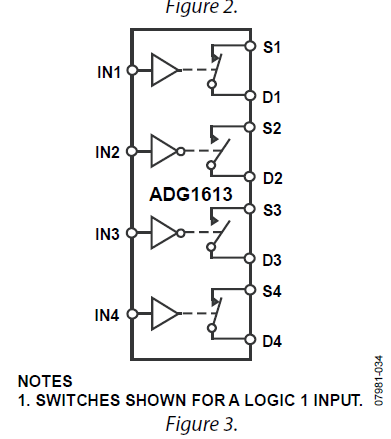
7.9

1. 数据类型如何转化

为什么选择鲲程电子这个公司？

和中北大学有合作，在校招上看到这家公司就感兴趣

经过了解，发现是民爆控制相关技术领域的相关

1. 怎么让他开机的时候就初始化显示屏而且不再闪动呢
2. 确认ADC采集电压的计算方法，以及短路的判断方法，还有的就是总线之间电流或者电压差值的计算方法。什么时候采用什么样的数据类型？
3. 发现LINEB输出6.7V电压，本应该是输出0V的，因为没有对SPST进行设置。
4. 
5. 
6. 外置无线网卡
7. Extern
8. K2的输出 LINEA和LINEB有问题，在置1的情况下

7.10

1. 今天查找K2的错误，从接口一直查到芯片，一直断一直断，最后就剩裸芯片。然后再换芯片观察是否出错，或者把芯片焊在另一块板子上去测试。如果这都是错的。那就是设计出现错误。
2. 今天做了一个很笨的办法，把芯片管脚用一根根线练出来，把芯片粘在纸胶带上，去检查芯片是否有错误。最后发现是设计错误，ADG1613的13管脚上是需要接电源的，，设计中并没有接上 ，所以就会出现控制不稳定，而且出现不该有电压的管脚却又电压出现。
3. 需要AD采集的端口：
4. 需要输入检测的端口
5. USB
6. SPI
7. 一些具体的细节等到做系统的时候再想一想
8. 要测试报告和硬件软件设计报告
9. 文档该怎么写？

7.13

1. 今天的任务：（1）把所有的GPIO的配置全部都设置好，写成模板。(2)AD中电压和电流与采集到的数据的关系。满量程是多少。（3）研究USB函数的调用方式（4）把硬件原理图和PCB重画，**需不需要修改排布方式**（5）**漏电流的测量方式**，先同7V后通5V通过计算之间的压差来计算电流大小 (6) 延期的时间
2. 任务细则：

配置GPIO——GET\_USB（主从切换信号输入口）**T**

**AD采集**

PC0-ADC\_123\_IN10-AD\_12V(检测电源电量)

PC1- ADC\_123\_IN11-AD\_LINEA **T**

PC2- ADC\_123\_IN12-AD\_LINEB **T**

PC3- ADC\_123\_IN13-AD\_COM\_OUT（检测输出电压是否正常）

PA1- ADC\_123\_IN1-AD\_I1(电流检测量程1)

PA2- ADC\_123\_IN2-AD\_I2（电流检测量程2）

**DAC控制电压**

PA4-DAC\_OUT1-DAC\_REF(6.7V控制电压)

**输出端口**

PA0-ALARM\_EN(输出使能端口)

PA6-5V\_SW（输出5V使能）

PA8-VUSB\_EN

PB12-SPI2\_NSS-USB\_CS

PB13-SPI2\_SCK-USB\_SCK

PB14-SPI2\_MISO-USB\_SDO

PB15-SPI2\_MOSI-USB\_SDI

PC6-USB-INT

PC7-I\_TX(电流读取或者使能输出端口)

PC8-I\_SW1(电流换挡控制端口)

PC9-6.7\_CONVERT(5V和6.7V切换输出控制端口)

PB5-LED1(上电软件正常指示灯)

PB7-LED4(从机)

PA15-LED3(主机)

PB8-OLED\_SDA(I2C)

PB9-OLED\_SCL(I2C)

**输入端口**

PC11-H-OR-SW(USB工作模式切换)

量程怎么计算，怎么看

提前需要测试好的内容：

电源对不对

负载输入电压对不对

AD采集电路分出的电压对不对

AD的数据采集，输出的数据位数是16位的，但是我自己在主函数里面声明的局部变量是8位的。造成一个结果就是显示出来的数据精度误差很大。

1. 上电的时候电路板不能去焊接元器件，必须断电，不然有的元器件会被烧坏。LM828就被我这样烧过一次。
2. 在OLED显示的时候想一想怎么去初始化以前的寄存器，不然它就把以前输入过的东西就显示出来了

7.14

1. 今天把电流部分算是调试出来了，有点瑕疵就是数据漂移有点厉害，尤其是在uA档位的时候，不接入雷管，也有20uA左右的示数。可以使用的数据就是误差处理。上午处理好的硬件问题有：AD\_I1第一档位，第一个放大器的放大倍数不对，应该是十倍的放大倍数，实测是两倍的放大倍数。把放大电阻拆掉以后重新换了两个电阻以后放大倍数正常。I1挡的跟随器在断开输入的情况下输出电压，一开始是5V的电压，后来接上电路以后示数正常。但是如果断开的话还是会有示数从存在，但是不影响电路，没有再深究这个问题。
2. 测电流是需要把表笔串联进电路中的，而不像测电压一样并联在被测试的器件上。
3. 电流电压校准：在看了控制器中的AD采集程序后，有所感悟。所谓的校准就是在大学时候最简单的误差处理一样，通过采集四个数据，然后对这四个数据做分析，建立一个直线方程y=kx+b,得到所谓的校准参数，然后把这个参数写在程序中，程序再根据这个参数做计算，算出y，以前得到的数据就是x。具体的实现方式还是有点不理解，尤其在程序里面有很多右移计算，还有与计算。最常见的是和0X7FFF或者0X8000相与
4. C语言中常见的数据计算和其对应的代码
5. 大量程和小量程是怎么区别的，一个放大10倍，一个放大100倍，哪个是小量程，100倍的是么。那她的量程就得比放大十倍的量程小十倍，因为同样的电压，1V放大十倍就是10V，经过采样电压就是2V。但是放大100倍就是100V，经过采样电压就是20V严重超出了量程范围。这是理论上的计算。

7.15

1. 有返回值的函数该怎么写，返回值为布尔量，返回值为数据
2. 为什么在.c文件里面写完的函数还要在头文件里面再写一遍函数名
3. 两个重要的关键字：extern和static这两个关键字。Extern是用于
4. 全局变量怎么声明，怎么知道他是局部变量还是全局变量。
5. 函数的类型是什么，一般要怎么写？
6. C标准库提供了大量的程序可以调用的内置函数，比如可以连接两个字符串的函数strcat()，复制内存到另一个位置的函数memcpy()
7. 函数**声明**告诉编译器函数的名称、返回类型和参数。函数**定义**提供了函数的实际主体。
8. Break,return,continue
9. 引用调用 传值调用
10. 指针需不需要提前定义，还是说直接引用就好了

比如在下列代码中：

#include <stdio.h>

Void swap(int \*x, int \*y);

Int main()

{

Int a = 100;

Int b = 200;

Printf(“交换前，a的值：%d\n”,a);

Printf(“交换前，b的值：%d\n”,b);

Swap(&a, &b);

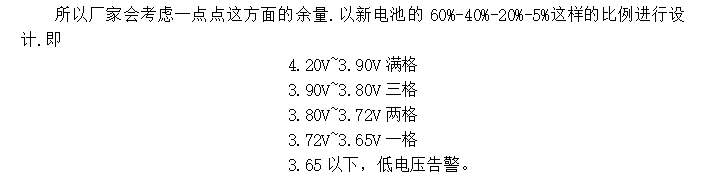
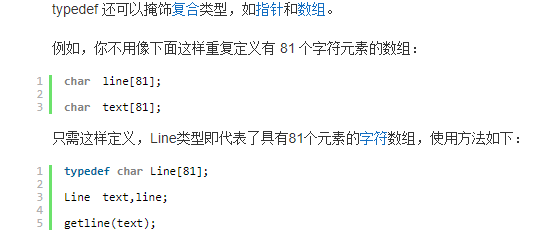
Printf();

Printf();

Return 0;

}



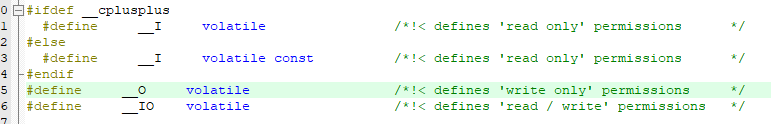
1. 主函数中出现的return 0，等代码设计的作用是用来查看程序是否正常编译或者运行。在程序的表面是看不到最后的返回值的，需要通过电脑中的命令符去调用，查看。相当于进入底层代码中去查看。
2. 
3. 电池电量和电池电压之间的对应关系
4. 
5. 
6. 链表的建立，结构体的建立。使用typedef的场合
7. 
8. 数据类型 int float

Signed unsigned short long 这些是什么含义还有size

1. Sizeof操作符这是什么含义
2. 递归
3. 堆和栈，他们都放在哪里
4. 

7.16

1. STM32的数据类型有哪些
2. 在STM32中有一个对只读/只写/读写均可以的操作命令



Volatile是用来防止编译器优化的关键字，每次读取或者修改值的时候都需要从内存或者寄存器去读取。

1. 为什么不能滥用指针？因为指针指向的地址可能会发生改编，这样会发生灾难性的后果。
2. 为什么在有的宏前会看到下划线？因为宏都是需要大写的，这样会很容易造成命名重复
3. 对底层函数要多做封装，然后好调用，层层封装
4. 
5. 如果输入输出参数、异常处理、日志处理等没有规范，很容易导致大量低级 bug，还很难找到 bug 的原因。
6. 写代码的状态：接到任务，规划，分模块。然后根据功能自己编写函数，在编写完函数的时候要进行白盒测试，编写一段测试代码。
7. 代码很烂的话那就试图去改变一下，**给你单独的时间去做重构很难**，那就**边开发功能边重构，边迭代边优化**，在不影响产品迭代的情况下，尝试着去做些力所能及的改变，在这个过程中你会发现，其实你也在慢慢的成长。
8. 架构，顶层设计。
9. 为了更好的开发，以及后期的维护。把一些在函数中用到的底层的库的宏或者是函数封装起来，放在头文件中方便调用。
10. 把嵌入式开发代码分成三个层次：底层硬件抽象HAL层（提供定时器，ADC，中断，I/O），功能模块层FML，应用程序层APL
11. SFR是定义硬件寄存器地址的特殊关键字
12. **Extern int I;**这句话里面只是声明一个变量i，没有对这个变量进行定义，定义可以在别的文件中定义。

举一个例子：

Extern int a,b;

Int main()

{

Int a,b;

a = 10;

b = 3;

…

}

1. Char是字符型常量，括在单引号中
2. **常量的两种定义方式**：定义在预处理中，**使用 const关键字**
3. 常量要定义成大写的形式
4. .c文件和.h文件之间的关系。最初是只有.c文件的并没有.h文件，但是由于项目的增大以后，同一个变量的相互调用，修改程序就变得复杂无比。在这种情况下.h文件就应运而生。所以发展到现在.c文件就写函数体，而.h文件就写函数的声明和全局变量。
5. 发现了一个认知错误。**&&和||**和！

&&是二者均不为零则为真

||是二者均为零则为假，有一个不为零则为真

1. &和|和^和~是位运算符
2. 杂项运算符：sizeof和三元

|  |  |  |  |
| --- | --- | --- | --- |
| **&** | 返回变量的地址 | **\*** | 指向一个变量  （指向对应地址下的变量） |
| Sizeof | 返回变量的大小 | ?X:Y | 条件为真则返回X  否则返回Y |

1. 循环有三种：for /while /do while

For(（可不写）;condiction;（可不写)）

{

Statement();

}

While(condiction)

{

Statement();

}

Do

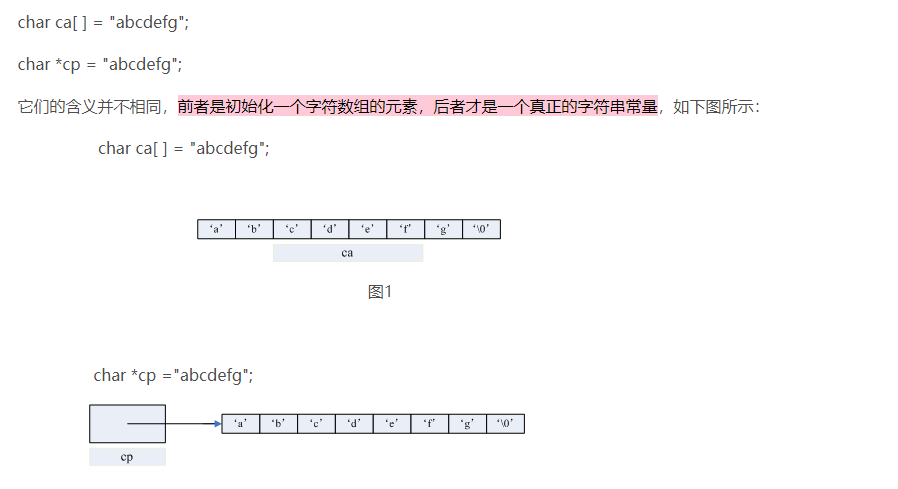
{

Statement();

}while(condiction);

1. 循环中有三种循环控制语句：break continue goto(不建议用)
2. Break和continue这两个有什么区别和联系
3. 变量在定义的时候有三种关键字：默认的为auto，还有static和extern
4. 无限循环; for(;;)，while(1)
5. 在函数的声明中参数的名称不重要，参数的类型才重要，所以在函数声明中可以把参数的名称省略掉
6. 指针和数组，可以用数组去初始化字符串

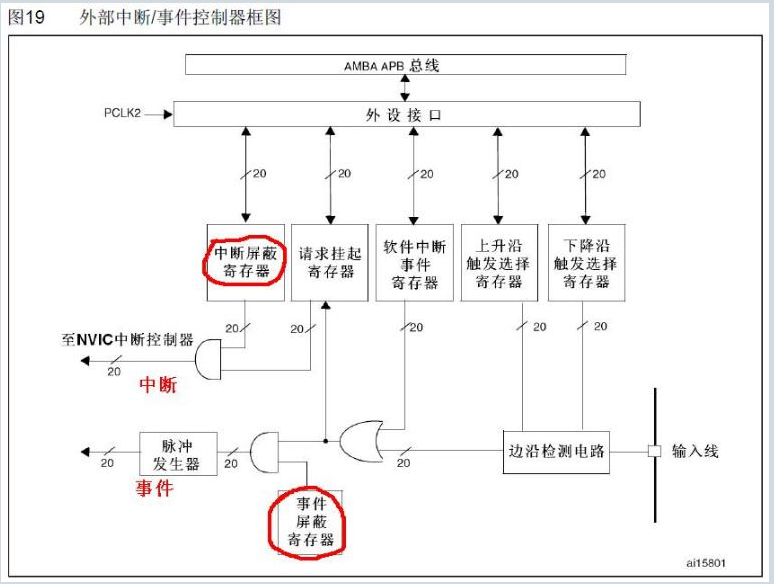
<https://blog.csdn.net/donahue_ldz/article/details/12427329/>

1. 
2. **类名class** <https://www.runoob.com/w3cnote/cpp-static-usage.html>
3. **二维数组在逻辑上是方阵**，由行和列组成。**但是二维数组在物理上是线性的**，按行来依次进行存放，内存是连续的
4. 看不懂为什么二维字符串数组能够当一维数组使用
5. Char型字符串，编译器是怎么处理的
6. <https://www.runoob.com/cprogramming/c-multi-dimensional-arrays.html>
7. 指向数组的指针
8. 数组名是一个指向数组中第一个元素的常量指针
9. 指针是可以加减运算的，比如\*p \*(p+4)就是指向另外一个地址的指针
10. 枚举就是将原来宏定义的别名用一个简洁的方式去定义出来
11. 遇到多个选项可以选的话就可以用枚举
12. 定义指针的时候如果说没有确切的地址可以赋值那么就可以赋值一个空指针NULL，在系统中，空指针意味着地址为零，但是零地址是无法被访问的，以为操作系统在的特殊设定。所以就告诉她不指向任何东西

7.17

1. .obj文件是编译以后的文件
2. 头文件里面是声明，如果说在头文件里面把全局变量的声明和定义放在一起，然后分别在不同的源文件中引用那么有可能出现的问题就是，报错。因为同一个变量他在工程中只存在一个内存，被多次引用以后就会出现多次定义，所以编译器就会报错
3. 结构体，类。什么是类？
4. linux内核源码基本上只能在gcc上面 进行编译
5. 编译，链接，调试，软件工程 GCC（编译器）GDB（调试器）
6. GNU binutils是一组二进制工具程序集，是辅助GCC的主要软件，包括我们熟悉的as（GNU汇编器），ar（建立、修改、提取归档文件，归档文件是包含多个文件内容的一个大文件）、ld（连接器，把目标文件和归档文件结合在一起），我们在写程序的时候，有时候报错：[Error] ld returned 1 exit status，这就是程序在链接的时候出错了。
7. #define和全局变量不是一回事，全局变量的格式是type name。不要混淆了
8. 而且**头文件中只能声明全局变量而不能定义。如果想要被多个文件调用全局变量的话就必须在声明的时候增加extern**
9. 宏定义的高级技巧——自己去总结
10. 
11. 这个代码是什么意思，为什么要展开用小写extern替换EXTERN呢，为什么要用宏来添一脚呢？
12. 空的宏定义的作用以及常见用法：
13. 
14. 变量的命名
15. 一个优秀的程序员通常具备思维缜密、结构清晰的编码基础。变量命名和封装都恰如其分，系统中几乎不用全局变量。函数应保持简短有效
16. 编写程序要用缩进的方式，注释、以及测试
17. 从事嵌入式的人必须掌握一个操作系统，经典的是vxworks，很多概念和技术都和linux，从事C语言开发
18. Linux系统在嵌入式的有被裁减过的：uClinux
19. 移植，写驱动，内核裁剪和优化
20. 学习ARM各种基本应用，先布一个仅有flash，sram，cpu,JTAG和复位信号的小系统板。留好扩展接口
21. 根据硬件地址先写能够启动的小代码。包括：初始化端口，屏蔽中断，把程序拷贝到SRAM中，完成代码的重映射；配置中断句柄。Bootloader有很多东西
22. 不要拿别人的例程改一改。不要依赖心理。要学习别人的代码。多看一些操作系统程序。要对操作系统的机理有所了解。
23. **思考+输出**
24. 中断的思考

7.21

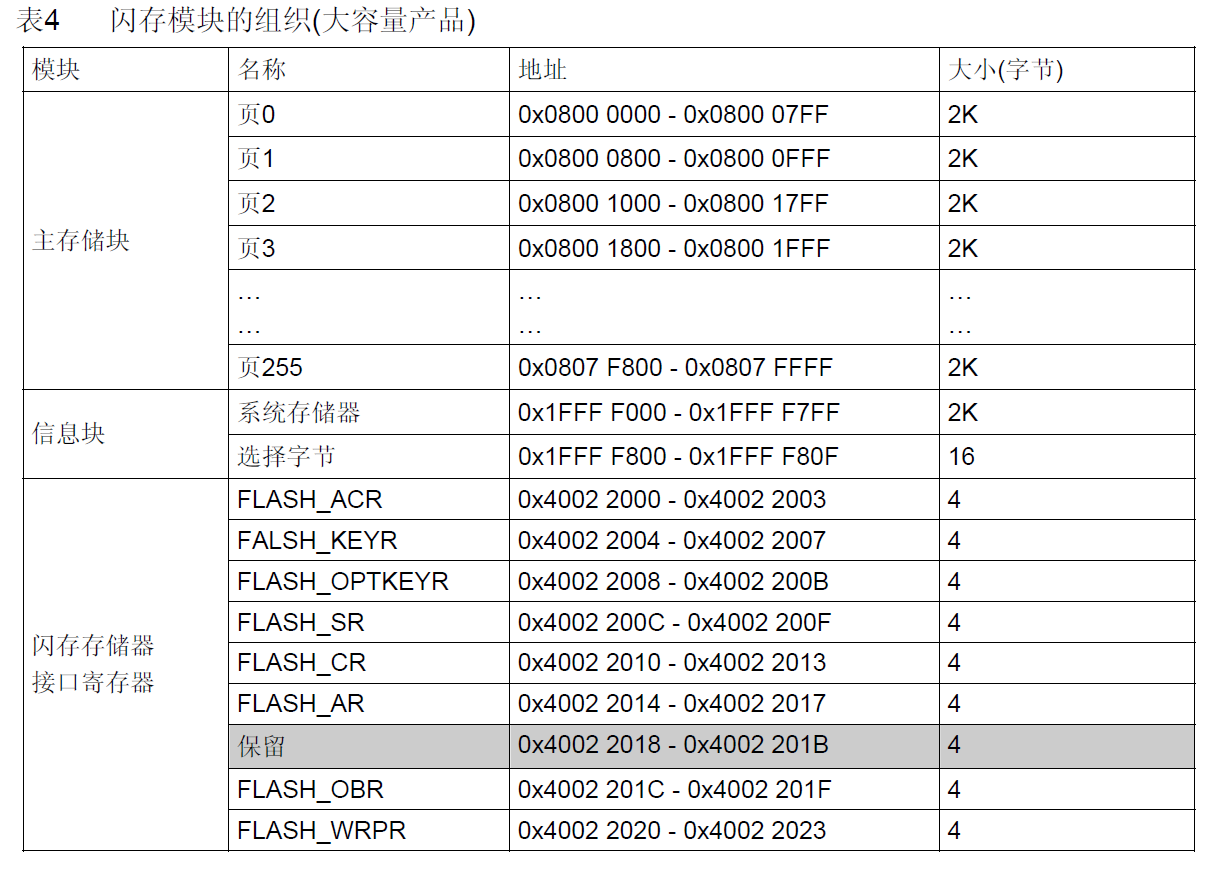
1. Flash的使用寿命，如果要使用文件系统来对FLASH进行擦除的话，有时候需要一个平衡算法，就是可以兼顾flash寿命的文件系统
2. 中断系统，STM32上的中断和UCOSII上面的中断一样不，怎么看。个人觉得ucosii上的中断最终是要回归到STM32上的中断的，因为芯片是32的芯片
3. 中断和事件？
4. 
5. 启动文件能不能自己去写一个出来
6. 栈stack和堆heap
7. 0X0400 = 1024就是相当于1K
8. 0X0200 = 512
9. 堆主要用来动态内存的分配，像malloc()函数申请的内存就在堆上面。但是堆在32中用到的就比较少
10. 向量表，向量表中就是一个32位的数组，每一个数组对应着一个异常
11. 向量表从FLASH 的0 地址开始放置，以4 个字节为一个单位，地址0 存放的是栈

顶地址，0X04 存放的是复位程序的地址，以此类推。从代码上看，向量表中存放的都是中断服务函数的函数名，可我们知道C 语言中的函数名就是一个地址。

1. 在启动文件里面已经帮我们写好所有中断的中断服务函数，跟我们平时写的中断服务

函数不一样的就是这些函数都是空的，真正的中断复服务程序需要我们在外部的C 文件里面重新实现，这里只是提前占了一个位置而已。

1. 闪存存储器接口FLITF
2. 闪存：



1. CRC——冗余校验单元
2. RCC——时钟配置
3. 设置系统时钟函数是需要自己写还是说修改官方的库文件里的函数，或者说直接把函数写在一些基本文件中比如stm32f103x\_it这类型文件中，或者说是delay文件中。——野火零死角玩转单片机中有提供库中的源代码
4. **时钟初始化的实验**，通过LED等闪烁的快慢去判断时钟的频率是否改变。
5. **仿真器和下载器类型有哪些，为什么下载器上面有那么多的接口，他们都是用来做什么的，以及如何使用**
6. 向量表，优先级
7. IRQ——中断请求
8. PendSV与SVC
9. 建立一个齐全的文件库，比如在STM32上某一个型号的芯片技术手册
10. FLASH属于广义上的ROM，和EEPROM的最大区别是FLASH按扇区操作，相对于EEPROM的改进就是擦除时不再以字节为单位，而是以块为单位，一次简化了电路，数据密度更高，降低了成本。
11. USB主机模式和从机模式的调用函数顺序
12. 0xFFFFFFUL——**后缀ul的含义是unsigned long的意思**
13. A->B=C，和A.B=C的含义相同，前者是变量，后者是指针

7.22

1. #define SCS\_BASE (0xE000E000UL) /\*!< System Control Space Base Address \*/

#define SysTick\_BASE (SCS\_BASE + 0x0010UL) /\*!< SysTick Base Address \*/

#define SysTick ((SysTick\_Type \*) SysTick\_BASE) /\*!< SysTick configuration struct \*/

1. 怎么进入汇编调试
2. KEIL-ARM代码在线调试技巧
3. KEIL中各种窗口
4. 汇编和反汇编是怎么回事
5. Keil版本和j-link的版本怎么对应才能兼容
6. 编译窗口有三个
7. 调试命令——KEIL中command窗口上使用
8. Memory窗口可以输入C D I X分别代表存储空间，直接寻址的片内寻址空间，间接寻址的片内寻址空间，扩展外部的RAM空间
9. 寄存器窗口是一些真实存在的寄存器
10. KEIL调试技巧——**打印输出printf**
11. 13:47开始整理USB程序…
12. if(0x01 != (OpData.UsbState&0x07)) 为什么不直接用 UsbState != 0x07?
13. USB的几种状态，什么时候确认发送数据，发送和接收数据有什么步骤？SPI是不是被当做一个函数就好比I2C直接插入到所需要的的函数中就可以使用了？
14. 中断在哪里？先把中断找到
15. 移植+裁剪
16. FATFS文件系统
17. 17:30从机模式程序运行方式大概弄清楚了
18. 今天做的事情：解决昨天结构体指针地址的问题，重新回顾KEIL软件功能问题，寻找J-Link驱动问题，看USB代码（看懂USB从机）

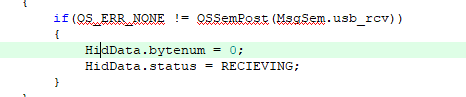
7.23

1. USB三大块：初始化，任务块，底层驱动
2. USB从机模式的中断源是什么——找到了
3. USBicon是做什么用的，有没有必要，在statusbar.c文件中看到有查询USB设备的代码



1. 看到升级程序和读取USB中hex文件的程序——waitdiskinsert.c hexfileimport.c
2. 文件导出——fileexporttomsc.c
3. 记录文件和输出文件都是怎么操作的？
4. 固件升级是怎么升级的？
5. Flash上的地址是怎么操作的，如何才能把文件系统挂载到FLASH上面呢，如何才能对flash进行操作呢
6. AT45DB是什么flash，
7. 源文件移植的时候把所有的函数都移植过去还是只移植需要的函数呢？
8. 基本的驱动+初始化+中断，插上U盘，电脑会不会显示盘符？，或者说需要串口调试助手
9. Slip和package两个源文件有什么联系
10. 文件追踪没有到位！
11. 先配置中断，把硬件层面的问题处理好
12. 先抽象一下，从CH374驱动上抽象出硬件接口出来。
13. USB的状态——被挂起，还是被装载。被读取还是被写入
14. 感觉从硬件层面抽象出来以后，HID，MSC——查一下，发送之后应答，接收之后应答。
15. 查一查USB的资料
16. CH374驱动完成的任务有哪些，它做了哪些工作。
17. USB的两个状态，获得USB的工作状态是否需要供电和传输数据时候的响应。USB Device模式是中断模式，中断是怎么来的呢？
18. 各个操作应该如何衔接
19. 最简单的USB代码有哪些东西？需要打包数据么，对数据进行加工？
20. 调试的时候怎么用逻辑分析仪？调试的其他办法。**输入波形，然后分析函数的输出状态？**

7.24

1. USB抓包工具是什么？
2. **缓冲池**是一个抽象的概念，具体原理是什么，怎样建立起来的
3. 从OS中移植不用OS的代码，比较难，因为有一个叫信号量，但是这个信号量可不可以提取出来，信号量是需要传递的，找到传递的源头。
4. 
5. 这里传送的信号量找不到源头
6. SPI的配置和I2C的配置一样，需要重新设置，407的芯片中有些函数是103没有的
7. 先移植在OS中吧，查一查哪些函数是需要的哪些是不需要的
8. 先写一个中断服务函数吧
9. USB5V的电压该怎么控制，以前没有控制这方面的，都是靠硬件完成的，现在需要自己去完成
10. 写一个关于USB全部，从硬件到软件的框架，要具体到每一个函数，写明白每一步运行的步骤。看一下USB协议
11. Ucosii的信号量
12. 有什么想法可以问
13. 下午要做的事情，先写一个中断，并实现
14. 再移植USB到有os的工程中
15. 研究信号量
16. 找一下底层运行的文件，分析一下什么是必须的
17. 研究中断需要的文件——例程+参考手册+数据手册+M3核编程手册+库文件手册
18. 中断+中断嵌套函数+中断优先级——写一个文档梳理一下
19. 中断是怎么跳到中断服务程序中的
20. 验证一下多个任务下，优先级该怎么设置，并且观察中断的运行顺序
21. Ucosii的中断好像就和单片机的中断一样
22. 试一试多任务调度

7.27

1. 今日任务：移植ucosii + USB代码，学会ucosii的基本操作，信号量，建立任务，删除任务。
2. Hook函数——钩子函数，怎么理解以及怎么实现，和实现什么功能
3. 任务的几种状态：



1. 任何中断服务函数都需要加上OSIntEnter 和 OSIntExit,这两个函数，他们的作用是进入中断函数和退出中断函数，前者记录中断嵌套层数，后者是有可能触发任务切换
2. 任务的控制块叫TCB，事件控制块叫ECB
3. 任务堆栈大小设置成多少合适，他指的是字节还是字
4. 建立信号量，邮箱的实验，从小实验堆积到大实验上
5. 学会使用库里面的函数进行功能搭建
6. **\_align(num) 汇编字节对齐**
7. Ucosii的任务不是循环进行的，而是谁的优先级高谁运行。执行顺序，找一个实际的例子来看，去任务建立函数OSTaskCreate()内部看