 //tick\_val = sys\_tick\_soft\_tick();

//debug\_out("t1:", &tick\_val, 4);

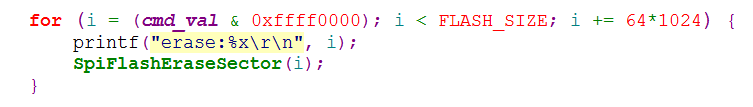
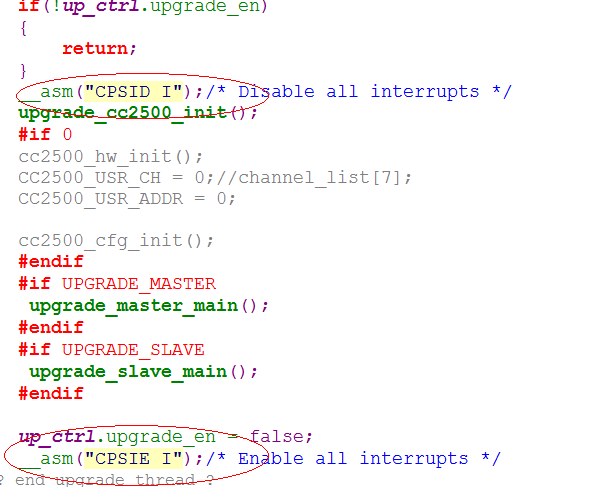
uart\_communation();

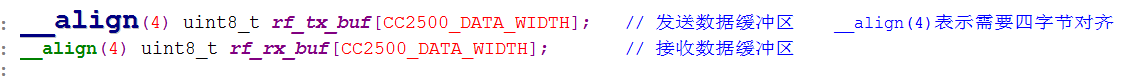
//tick\_val = sys\_tick\_soft\_tick();

//debug\_out("t2:", &tick\_val, 4);

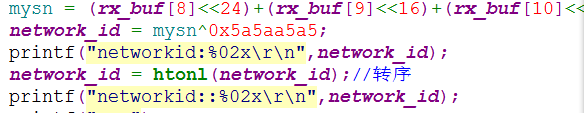
[%Y-%M-%D-%h:%m:%s]

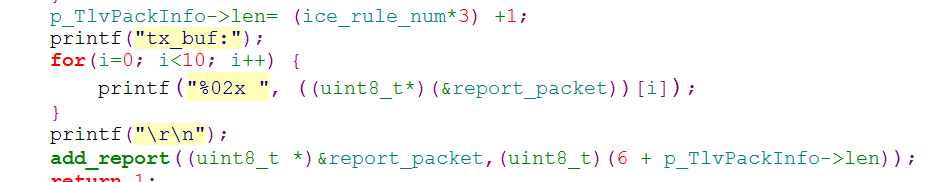
[%h:%m:%s]

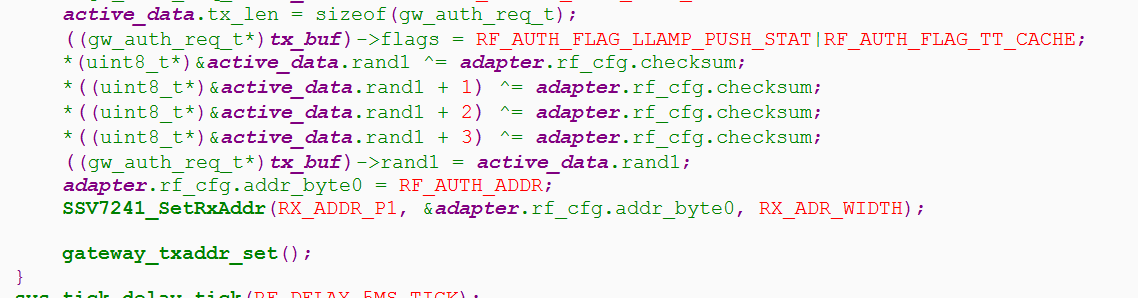
 **i为 uint32\_t i;**

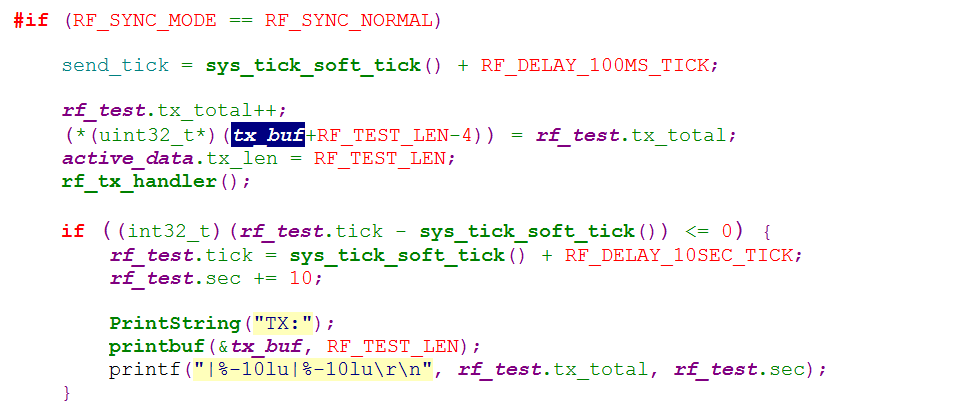




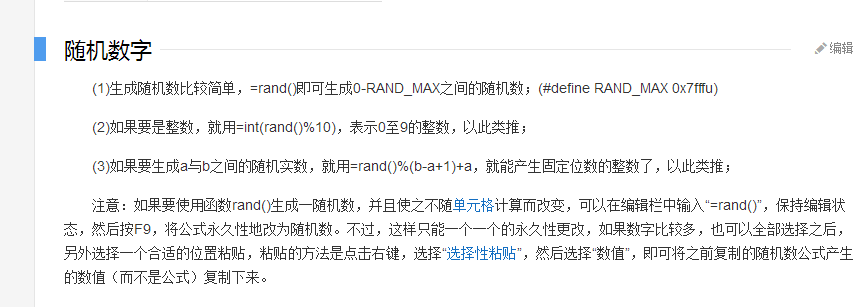
主机直接顺序转换成网络字节顺序：bb 47 21 84 转成84 21 47 bb

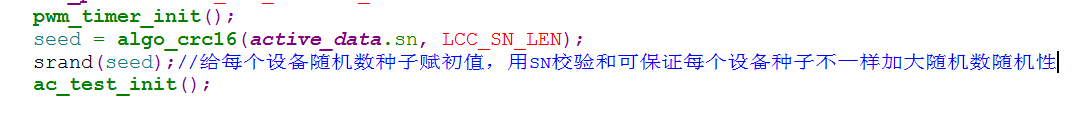






Rf\_test.tx\_total是一个32类型的数据，放在tx\_buf里面





S3网关日志查看操作

Login incorrect

MacBee S3 login: root

Password: galaxywind

BusyBox v1.12.1 (2017-08-11 11:25:51 CST) built-in shell (ash)

Enter 'help' for a list of built-in commands.

# rfgw\_di\_d

-sh: rfgw\_di\_d: not found

# rfgw\_cli -g 255

#

# rfgw\_cli -G 4

**iwulink\_cli -P SET 1 77 0切换信道到77S3网关**

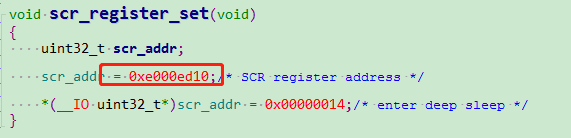
#

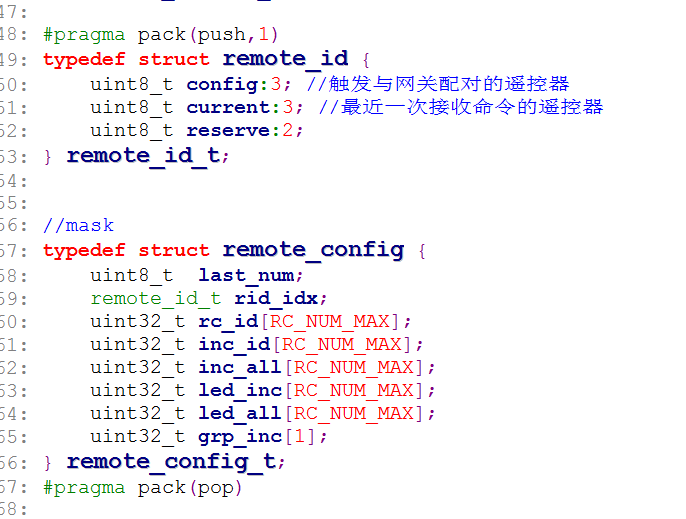
# tail -f var/log/kernel.log

2017-11-08 10:16:32:710 WARN - [MacBee\_TxData:624] - TX: (2)64 6B

**S1网关日志查看**

1. 密码:123456789
2. 开启日志：b4lG
3. 关闭日志：G
4. 系统重启及查看基本配置:s
5. 查看设备列表:d
6. 信道减：m
7. 信道加：a
8. 查看beacon:te0tG(需要特殊镜像看文件夹2018)





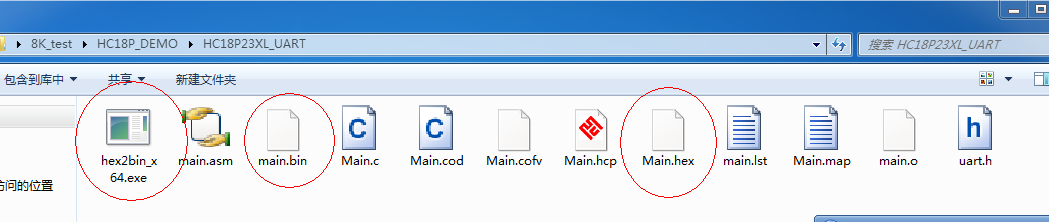
STM32按位使用，注意也有大小端问题，还需要字节对齐

当设备7241加上PA之后，一个是要注意调小7241的发射功率。2.PA芯片在发送模式时，将TXEN高电平使能RXEN任意，接收模式时，TXEN为低，RXEN为高。一定要注意这点。否则会导致收发异常

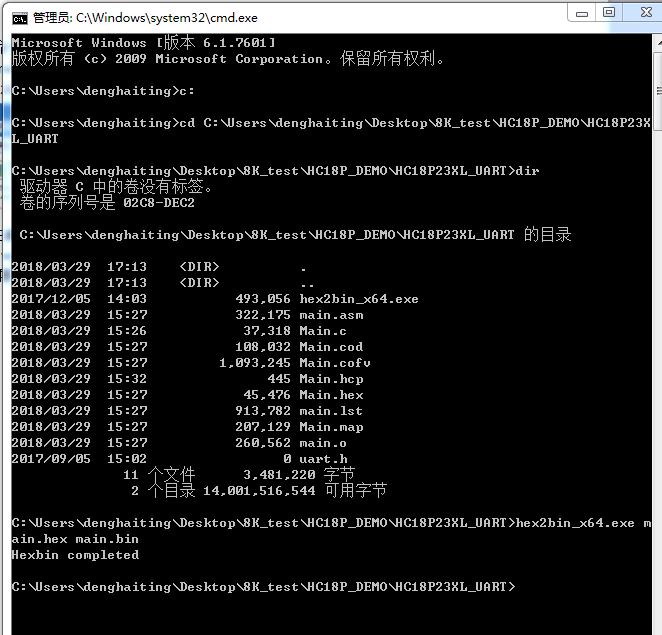
Windows 命令下如何将hex 文件转换成bin文件：

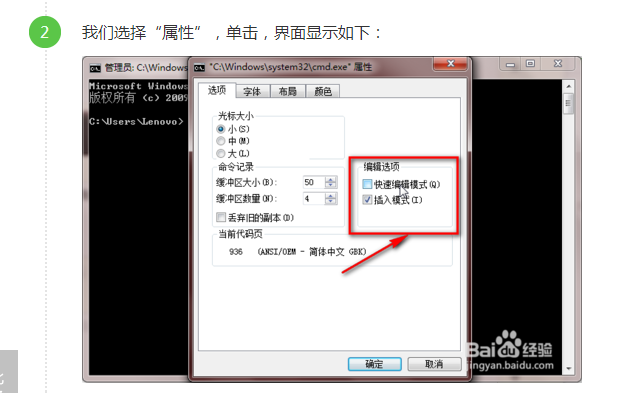
例如要将main.hex转成main.bin。

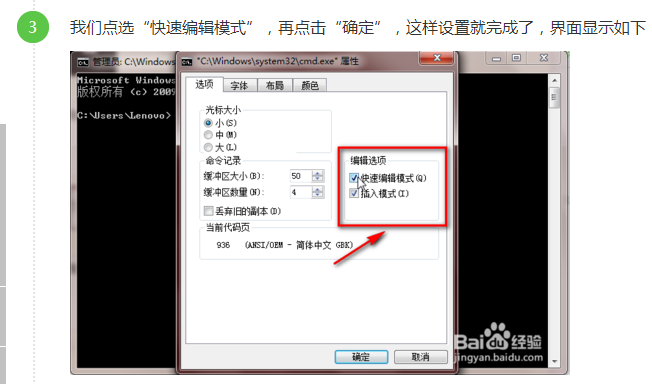
首先将hex2bin工具和main.hex文件放同一个文件夹下

本文是放在桌面的一个文件夹下：

进入windows命令之后输入c:进入到c盘；然后在输入cd 加文件路径。进入到该路径之后再输入dir展示文件。再输入转换命令hex2bin\_x64.exe main.hex main.bin 即可

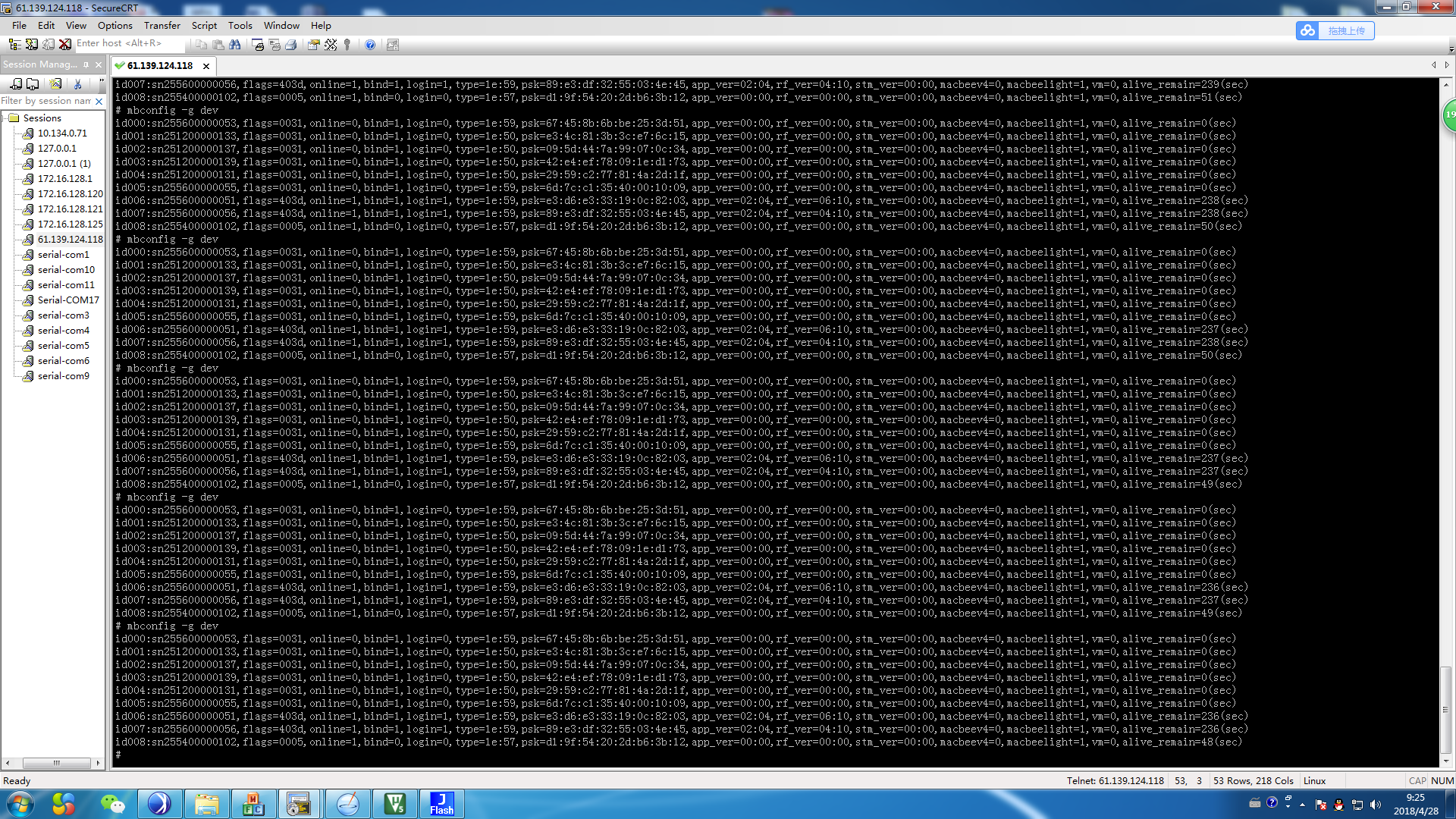




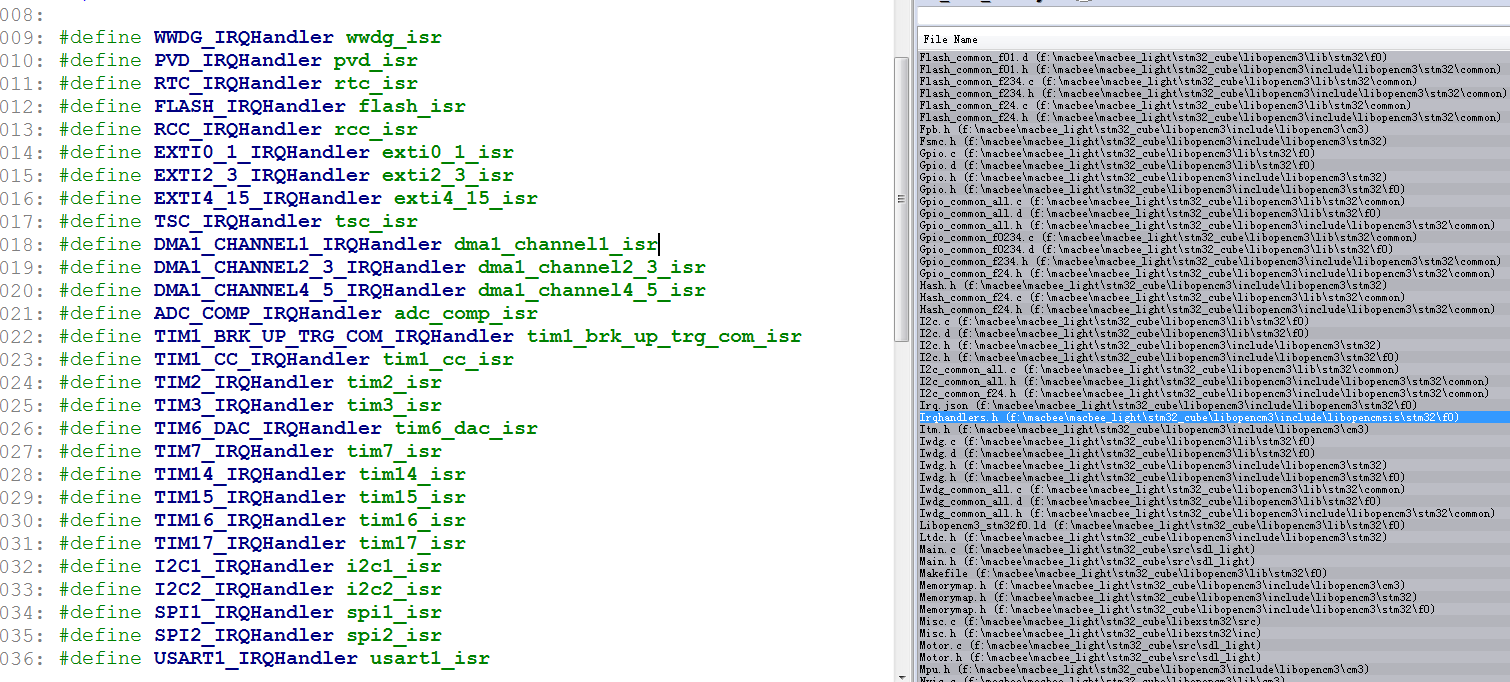




<https://jingyan.baidu.com/article/1876c852bcab82890b13768f.html在windows命令下鼠标右键是复制和粘贴>



想把数据dmx512-address\_t copy到dmx512-send\_buf的第四个开始。前面四个数据空着。以上实例



输出地址： printf("tx\_buf\_addr:%p\r\n",tx\_buf);

printf("time\_addr:%p\r\n",(&led\_config.time));

S3网关网页升级：  
首先打开telnet,查看Ip，浏览器访问IP+：890即可登录网关页面。

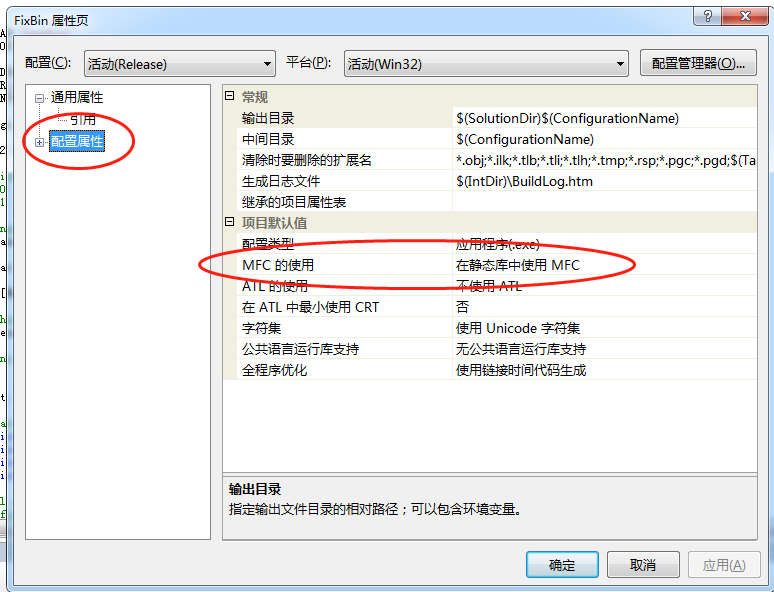
http://61.157.91.158:890/

外部flashW25Q16系列操作：

16Mbit=2097152字节= 8192页=512扇区=32块

1页= 256字节 1扇区 = 16页 1块 = 16扇区

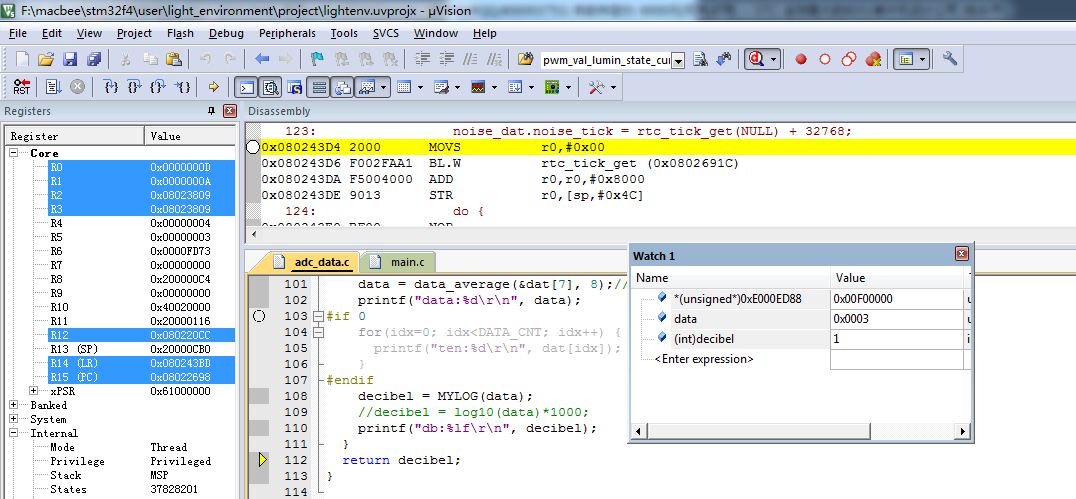
source insight 已经删除了项目文件了，但是还是能代码定义到删除之前的代码，此时执行file->close all.即可完美解决

VC++中在生成fixbin可执行文件时，点击：项目->fixbin属性->

即可按照静态库的方式编译fixbin。

KEIL仿真

如果要打断点，鼠标先指向那行，然后点击红色按钮就打好断点。然后点击debug进入仿真



先点击全速运行，此时会运行到断点处，运行标志（4个括号），

第一个括号表示一步一步执行，

第二个括号表示直接运行完下一步（如果下一步是个函数调用，那么这个函数当作一步执行，不会再将函数中的每一步一步步执行。一步就执行完）

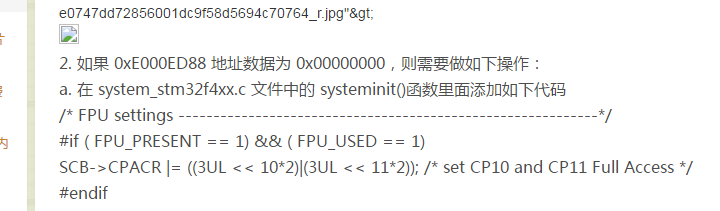
第三个括号表示你如果已经进入一个函数，当中如果有循环或者这个函数较大，它不会每一步每一步执行。点击该按钮，它会一次性执行完这个函数。

第四个括号表示运行到光标处。

查看窗口：

Memory窗口：表示内存窗口，你可以在该窗口中输入寄存器的地址，此时你就可以看到此寄存器的内容。

Watch窗口：看变量的值，上诉窗口中的data和decibel中data是我adc采样出来的数据，decibel是我要用data换算出来的分贝数。



\*(unsigned\*)0xE000ED88 |= ((3UL << 10\*2)|(3UL << 11\*2)); //表示STM32F4支持浮点运算

max =(max>noise[idx]?max:noise[idx]); min = (min<noise[idx]?min:noise[idx]);

volatile是一个类型[修饰符](https://baike.baidu.com/item/%E4%BF%AE%E9%A5%B0%E7%AC%A6" \t "https://baike.baidu.com/item/volatile/_blank)（type specifier）.**volatile**的作用是作为指令[关键字](https://baike.baidu.com/item/%E5%85%B3%E9%94%AE%E5%AD%97/7105697" \t "https://baike.baidu.com/item/volatile/_blank)，确保本条指令不会因[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8/8853067" \t "https://baike.baidu.com/item/volatile/_blank)的优化而省略，且要求每次直接读值。

简单地说就是防止编译器对代码进行优化。比如如下程序：

|  |  |
| --- | --- |
| 1  2  3  4 | XBYTE[2]=0x55;  XBYTE[2]=0x56;  XBYTE[2]=0x57;  XBYTE[2]=0x58; |

对外部硬件而言，上述四条语句分别表示不同的操作，会产生四种不同的动作，但是编译器却会对上述四条语句进行优化，认为只有XBYTE[2]=0x58（即忽略前三条语句，只产生一条机器代码）。如果键入**volatile**，则编译器会逐一地进行编译并产生相应的机器代码（产生四条代码）。

2. static

a) static在面向过程编程中的使用场景包括三种：

1) 修饰函数体内的变量（局部）

2) 修饰函数体外的变量（全局）

3) 修饰函数

第一种情况，static延长了局部变量的生命周期，static的局部变量，并不会随着函数的执行结束而被销毁，当它所在 的函数被第再次执行时，该静态局部变量会保留上次执行结束时的值。

对于后面的两种情况，static是对它修饰的对象进行了作用域限定，static修饰的函数以及函数外的变量，都是只能在当前的源文件中被访问，其它的文件不能直接访问。(就算是其他文件能访问，这是它的值也不是源文件计算之后的值，是一个随机值)当多个模块中有重名的对象出现时，我们不妨尝试用static进行修饰。

Const:

int main(void)

{

const uint8\_t n=10;

uint8\_t \*p = (uint8\_t\*)&n;

\*p=20;

printf("NNN:%d\r\n",n);

}

只要编译器适当的对代码进⾏优化，这⾥就可能输出： 10，⽽不是我们改变之后的值。

这是为什么呢？

是编译器在编译期间，可能对代码进⾏优化。

当编译器看到这⾥的num被const修饰，从语义上讲这⾥的num是不期望被改变（不改变）

的，那优化的时候就可以把num的值存放到寄存器（以提⾼访问的效率）中。以后只要使⽤num的地⽅都去寄存器中取，那即使num对应的内存中的值发⽣变化，寄存器也是感知不到

的。所以造成输出10的结果。现对代码如下修改

int main(void)

{

volatile const uint8\_t n=10;

uint8\_t \*p = (uint8\_t\*)&n;

\*p=20;

printf("NNN:%d\r\n",n);

}

结果⼜会是什么呢？

这⾥我们可以看到，当我们对\*p做了修改之后， num的输出变成了20。

这⾥ volatile 这个关键字起到关键的作业。

那我们说说 volatile 关键字的作⽤和使⽤场景。

volatile作⽤：

编译时不优化，执⾏时不缓存，每次需从内存中读出（保证内存的可靠性）。

%c 字符  
%d 有符号十进制整数  
%f 浮点数(包括float和doulbe)  
%e(%E) 浮点数指数输出[e-(E-)记数法]  
%g(%G) 浮点数不显无意义的零"0"  
%i 有符号十进制整数(与%d相同)  
%u 无符号十进制整数  
%o 八进制整数 e.g. 0123  
%x(%X) 十六进制整数0f(0F) e.g. 0x1234  
%p 指针  
%s 字符串

Uint8\_t cnt;

Printf(“addr:%p\r\n”,&cnt);//打印出cnt在内存中的地址

# STM32学习笔记之\_\_attribute\_\_ ((at())绝对定位分析

static volatile const uint32\_t encrypt\_addr1 \_\_attribute\_\_((at(0x08004404)))=0xFFddFF22;

int main(void)

{

uint32\_t reg;

reg = encrypt\_addr1;

debug\_out("regd:", (uint8\_t\*)&reg, 4);

printf("reggr:%x\r\n",reg);

printf("addr:%p\r\n", &encrypt\_addr1);

}

OUTPUT:

regd:22 ff dd ff

reggr:ffddff22

addr:08004404

这段代码第一句话的意思是将变量encrypt\_addr1放在地址为0x08004404的flash中，并给初值0xFFddFF22。后面是将这地址和地址中存的值打印出来验证。（STM32F4中验证的）

uint32\_t test\_addr1 \_\_attribute\_\_((at(0x08004404)));

//这段代码是在新唐单片机上运行的。16位定时器，定时器主频为22.1184/4，1tick ≈ 0.18us

void timer\_init(void)

{

ET2=0;//关闭定时器中断

T2CON=0;

T2MOD=0;

TL2=TH2=0;

TR2=1;

}

uint32\_t sys\_tick\_soft\_tick(void)

{

uint16\_t tick\_low;

tick\_low = (TH2<<8) + TL2;

sys\_tick = ((sys\_tick&0xffff0000)|tick\_low)+(tick\_low<(sys\_tick&0xffff))\*0x10000;

//上面的代码是定时器每加1，systick也加1，取出当前运行的tick

return sys\_tick;

}

#endif

void sys\_tick\_delay\_tick(uint32\_t tick)

{

uint32\_t delay\_tick = sys\_tick\_soft\_tick()+tick;

while((int32\_t)(delay\_tick-sys\_tick\_soft\_tick())>0);

}

void main(void)

{ uint32\_t tick;

//C(P00) output

P0M1 &= ~ BIT\_SHIFT(0);

P0M2 |= BIT\_SHIFT(0);

uart\_init();

timer\_init();

PrintString("hello world\r\n");

tick = sys\_tick\_soft\_tick() + RF\_DELAY\_1SEC\_TICK;

while(1) {

if ((int32\_t)(tick - sys\_tick\_soft\_tick())<=0) {

PrintString("hello world\r\n");

tick = sys\_tick\_soft\_tick() + RF\_DELAY\_1SEC\_TICK;

}

P00 =1;

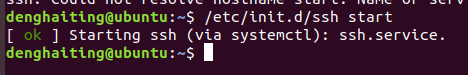
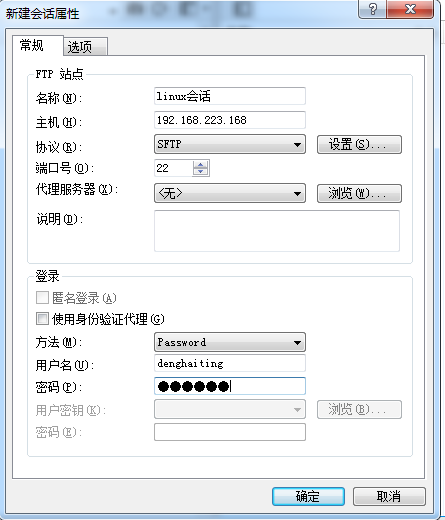
sys\_tick\_delay\_tick(RF\_DELAY\_1MS\_TICK);

P00 =0;

sys\_tick\_delay\_tick(RF\_DELAY\_100US\_TICK);

}

}

如何实现虚拟机ubuntu上的文件与windows互传，首先在windows安装xftp5文件传输app.windows下ipconfig查看自己的ip,linux下ifconfig查看自己的ip,然后linux和windows互ping IP确保双方能够网络互通的情况下在linux环境下首先安装ssh:sudo apt-get install openssh-server 然后开启ssh: /etc/init.d/ssh start .待linux开启成功之后，表示SSH开启成功。

git --version:查看自己有没有装git,以及git的版本

sudo apt install git :安装git软件，安装完成之后用git -- version查看版本

$ git clone https://github.com/libopencm3/libopencm3-dht.git：从git服务器上取libopencm3的代码存放到本地文件夹libopencm3-dht

这个网址是讲如何在linux上部署STM32开发环境：<https://blog.csdn.net/zhengyangliu123/article/details/54925400>

这个网址是讲如何在linux上git代码：<https://blog.csdn.net/Sophiahaha/article/details/83041505>

提交代码：1、git add 目录 2、git commit -m ‘自己的备注’ 3、git push

取代码：git clone <https://github.com/denghaiting/workspace.git> git checkout 目录名

Git账户：denghaiting password:\*\*\*358\*\*\*