# 【ARM】Linux驱动移植

[TOC]

# #1驱动模块移植过程

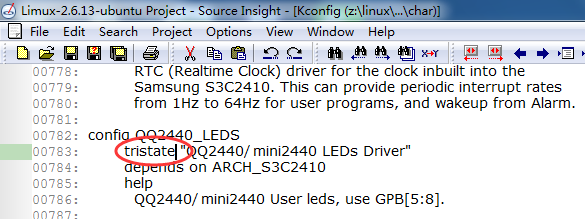
## ##1.1模块的第一种编译方法——改kconfig

### ###第一步：改kconfig

文件在：Drivers/char/kconfig；

tristate三态表示可以编译在模块`<M>`；

bool表示只能被选择编`[\*]`或不编`[ ] `。



### ###第二步：配置内核

```

wuchengbing@ubuntu:~/linux/kernel-2.6.13$ make menuconfig

Device Drivers ---> Character devices --->

[\*] S3C2410 RTC Driver │ │

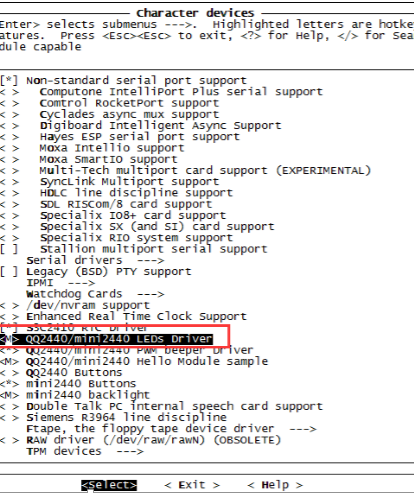
<M> QQ2440/mini2440 LEDs Driver │ │

<M> QQ2440/mini2440 PWM beeper Driver │ │

<M> QQ2440/mini2440 Hello Module sample │ │

<M> QQ2440 Buttons

```



如果` make menuconfig`没有出来该选项，可先`cp config\_n35 .config`->`make`-> ` make menuconfig`，然后成生如下.config文件。

```

wuchengbing@ubuntu:~/linux/kernel-2.6.13$ gedit .config

# CONFIG\_WATCHDOG is not set

# CONFIG\_NVRAM is not set

# CONFIG\_RTC is not set

CONFIG\_S3C2410\_RTC=y

CONFIG\_QQ2440\_LEDS=m

CONFIG\_QQ2440\_PWM\_BEEPER=y

CONFIG\_QQ2440\_HELLO\_MODULE=m

… …

```

### ###第三步：编译内核得到驱动模块

make

```

[wuchengbing@ubuntu:~/linux/kernel-2.6.13$](mailto:wuchengbing@ubuntu:~/linux/kernel-2.6.13$)make

… …

LD vmlinux

SYSMAP System.map

SYSMAP .tmp\_System.map

OBJCOPY arch/arm/boot/Image

Kernel: arch/arm/boot/Image is ready

AS arch/arm/boot/compressed/head.o

GZIP arch/arm/boot/compressed/piggy.gz

AS arch/arm/boot/compressed/piggy.o

CC arch/arm/boot/compressed/misc.o

LD arch/arm/boot/compressed/vmlinux

OBJCOPY arch/arm/boot/zImage

Kernel: arch/arm/boot/zImage is ready

Building modules, stage 2.

MODPOST

CC drivers/char/mini2440\_backlight.mod.o

LD [M] drivers/char/mini2440\_backlight.ko

CC drivers/char/qq2440\_buttons.mod.o

LD [M] drivers/char/qq2440\_buttons.ko

CC drivers/char/qq2440\_hello\_module.mod.o

LD [M] drivers/char/qq2440\_hello\_module.ko

CC drivers/char/qq2440\_leds.mod.o

LD [M] drivers/char/qq2440\_leds.ko

CC drivers/char/qq2440\_pwm.mod.o

LD [M] drivers/char/qq2440\_pwm.ko

wuchengbing@ubuntu:~/linux/kernel-2.6.13$

```

qq2440\_leds.ko

```

[wuchengbing@ubuntu:~/linux/kernel-2.6.13$ ls](mailto:wuchengbing@ubuntu:~/linux/kernel-2.6.13$%20ls) driver/char/

…

-rw-rw-r-- 1 wuchengbing wuchengbing 1368 Apr 17 20:38 qq2440\_hello\_module.o

-rw-rw-r-- 1 wuchengbing wuchengbing 3108 Apr 17 20:38 qq2440\_pwm.o

-rw-rw-r-- 1 wuchengbing wuchengbing 1412 Apr 17 20:39 qq2440\_hello\_module.mod.o

-rw-rw-r-- 1 wuchengbing wuchengbing 2247 Apr 17 20:39 qq2440\_hello\_module.ko

-rw-rw-r-- 1 wuchengbing wuchengbing 1404 Apr 17 20:39 qq2440\_leds.mod.o

-rw-rw-r-- 1 wuchengbing wuchengbing 3133 Apr 17 20:39 qq2440\_leds.ko

-rw-rw-r-- 1 wuchengbing wuchengbing 1404 Apr 17 20:39 qq2440\_pwm.mod.o

-rw-rw-r-- 1 wuchengbing wuchengbing 3957 Apr 17 20:39 qq2440\_pwm.ko

```

./mkimage.sh

```

wuchengbing@ubuntu:~/linux/kernel-2.6.13$ cd arch/arm/boot/

wuchengbing@ubuntu:~/linux/kernel-2.6.13/arch/arm/boot$ ./mkimage.sh

Image Name: linux-2.6.13

Created: Mon Apr 17 20:33:08 2017

Image Type: ARM Linux Kernel Image (uncompressed)

Data Size: 1537008 Bytes = 1500.98 kB = 1.47 MB

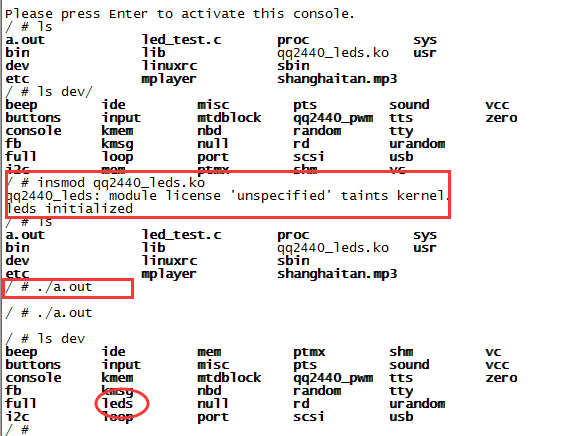
Load Address: 0x30008000

Entry Point: 0x30008040

wuchengbing@ubuntu:~/linux/kernel-2.6.13/arch/arm/boot$

```

### ###第四步：拷贝到文件系统，挂载板子，插入模块



运行示例程序编译出来的./a.out，板子的LED就会亮了。

License问题很需要在驱动代码加入` MODULE\_LICENSE("GPL");`

```

module\_init(qq2440\_leds\_init);

module\_exit(qq2440\_leds\_exit);

MODULE\_LICENSE("GPL");

```

### ###示例程序

```

#include <stdio.h>

#include <fcntl.h>

#include <sys/ioctl.h>

#include <stdlib.h>

int main(void)

{

int fd;

int i = 0;

fd = open("/dev/leds",O\_RDONLY);

if(fd==-1)

{

perror("open failed");

exit(1);

}

while(1)

{

ioctl(fd,0,0);

ioctl(fd,0,1);

ioctl(fd,0,2);

ioctl(fd,0,3);

sleep(1);

ioctl(fd,1,0);

ioctl(fd,1,1);

ioctl(fd,1,2);

ioctl(fd,1,3);

sleep(1);

}

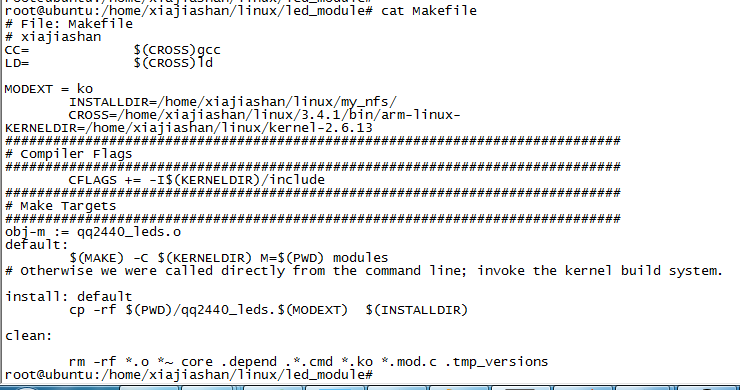
return 0;

}

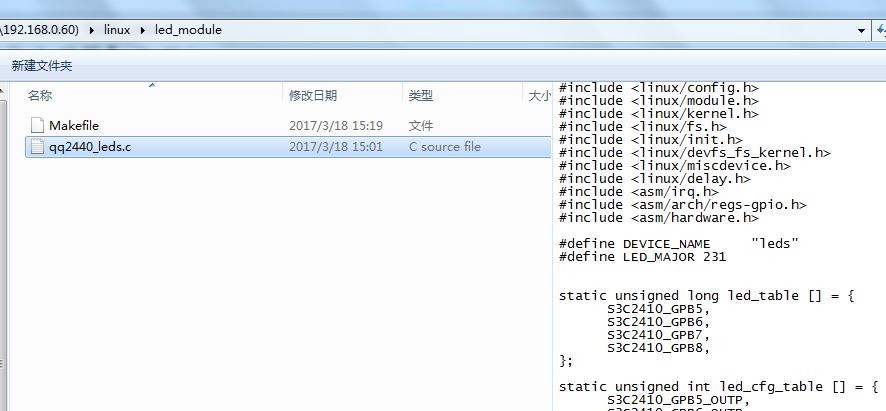
```

## ##1.2模块的第二种编译方法——Makefile

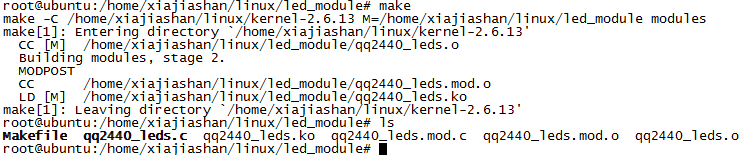
### ###第一步：写Makefile



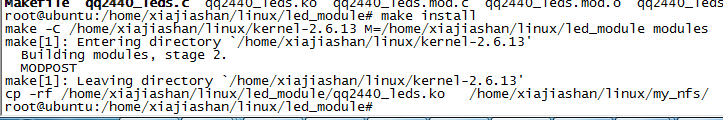
### ###第二步：直接把内核驱动拷贝出来



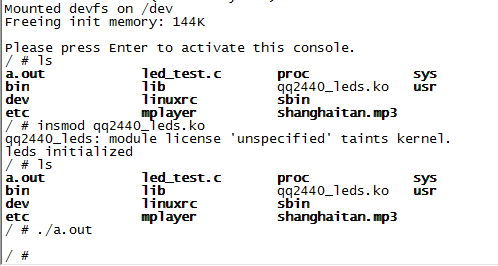
### ###第三步：编译



### ###第四步：安装



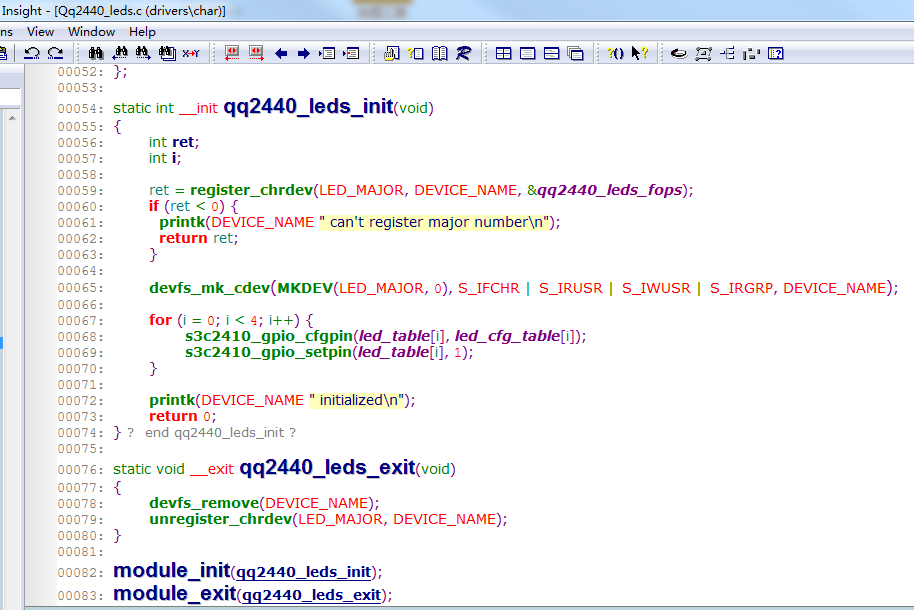
### ###第五步：插入模块看现象



# #2Linux驱动原理

## ##2.0从哪里切入

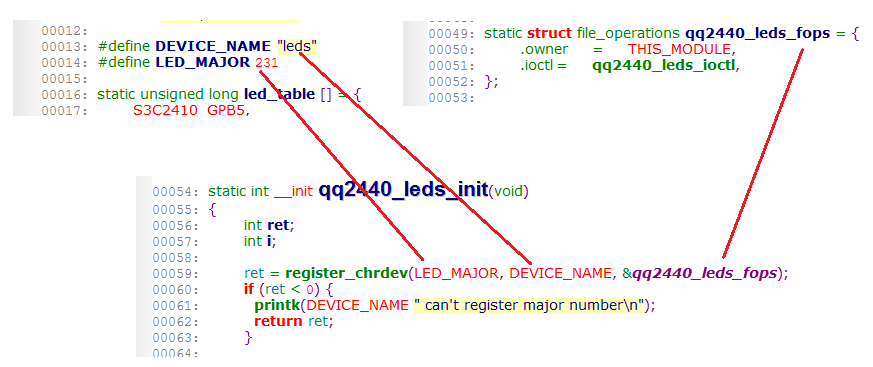
开发板自带的led程序（qq2440\_leds\_init）【**被module\_init调用----知道这里就行了**】



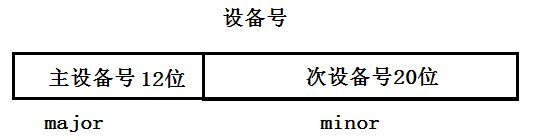
## ##2.1什么是注册



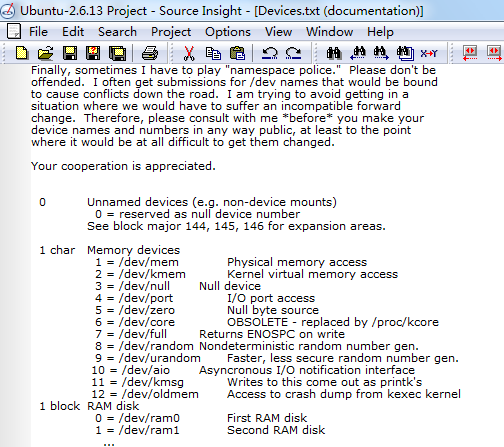
## ##2.2register\_chrdev提交哪些数据



### ###2.2.1设备号为什么是231



怎么知道设置231



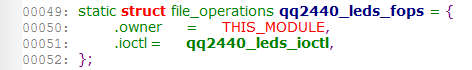
<http://blog.csdn.net/zjjyliuweijie/article/details/7001383>

**设置为231是因为231没有人使用。**

### ###2.2.2设备名随便给



### ###2.2.3文件操作结构体怎么设置



[file\_operations中各项解析](http://blog.csdn.net/sunsea1026/article/details/6586143)

<http://blog.csdn.net/sunsea1026/article/details/6586143>

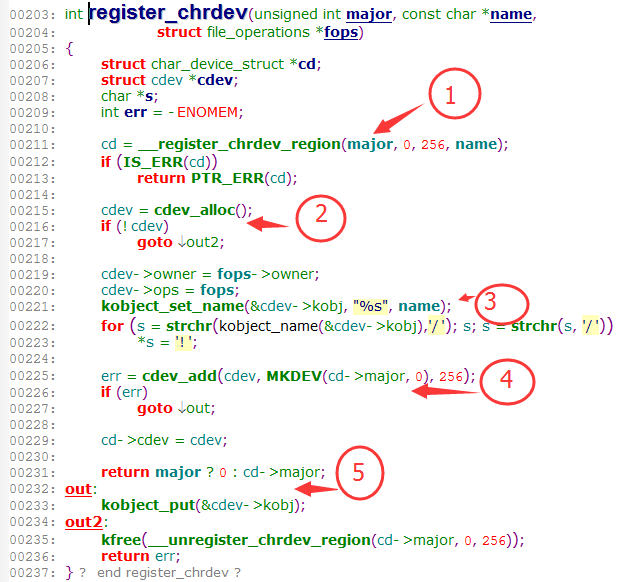
#### ####什么是THIS\_MODULE？

THIS\_MODULE在代码中复制给了owner，owner顾名思义是属主的意思，当把THIS\_MODULE复制给owner，表示该结构体属于当前模块，那么当前模块又是谁呢？当前模块就是：



#### ####Open,close,read这些都好理解

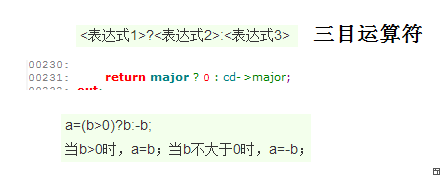
## ##2.3register\_chrdev返回什么？



### ###2.3.1 简洁而高效的goto

用不用goto一直是一个著名的争议话题，Linux内核源码中对goto的应用非常广泛，但一般只局限于错误处理中！这种goto用于错误处理的用法实在是简单而高效，只需保证在错误处理的时候记得注销，释放资源等！（与正常的注册，申请资源顺序相反）

### ###2.3.2 还记得什么是三目运算符吗？



### ###2.3.3 返回0表示什么？

当major为真的时候返回0，是什么意思呢？

Major为真的情况，只有可能>0，因为major的类型是unsigned。所以，当major大于0的时候，函数返回0，表示函数执行成功！

### ###2.3.4 什么情况下返回cd->major？

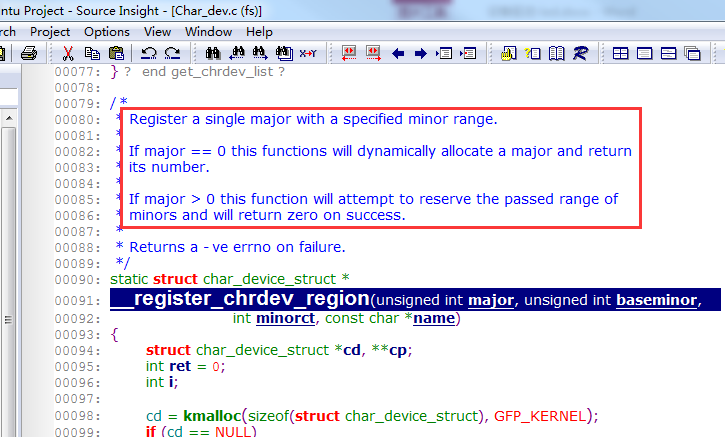
Major为假的情况，只有一种情况，等于零。那么major=0，表示什么呢？

当在调用register\_chrdev函数的时候，

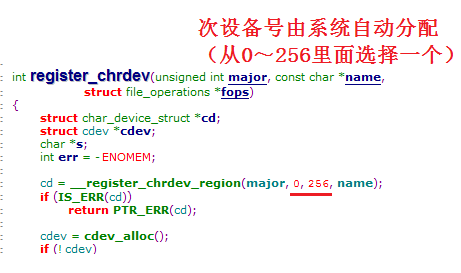
- 如果传进去的major=0，表示自己不定义设备号，由系统自动分配！

- 如果传进去的major>0，表示传入自己定义的设备号，不用系统自动分配！

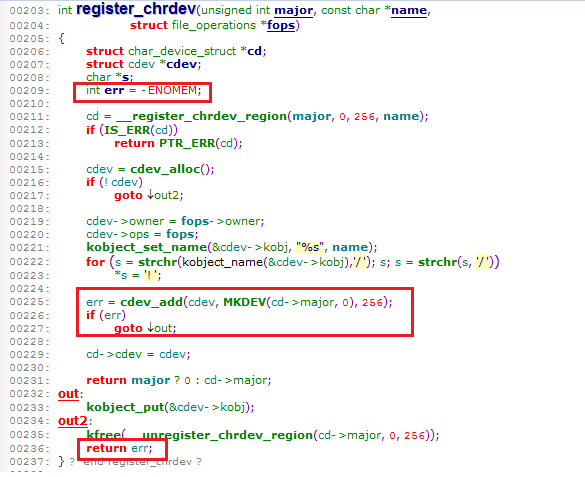
我们是怎么知道的？



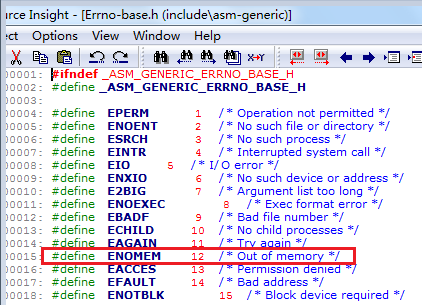
### ###2.3.5 次设备号怎么分配呢？



### ###2.3.6 出错返回什么？



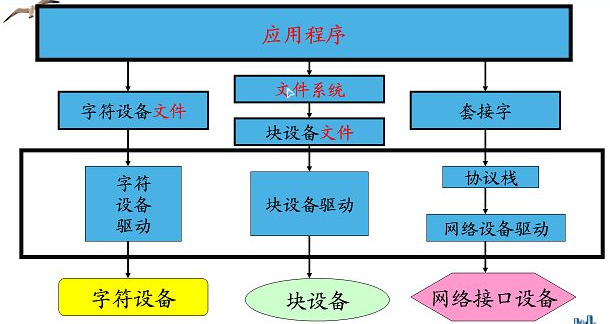
ENOMEM是什么？



现在能猜出cdev\_add返回什么吗？

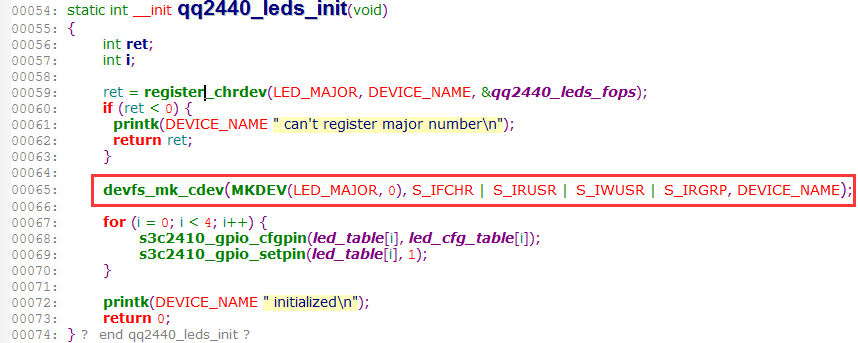


## ##2.4字符设备到底注册到了哪里？

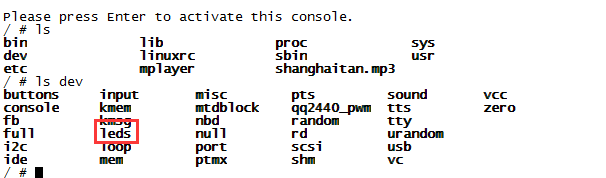


## ##2.5devfs\_mk\_cdev实现了什么？

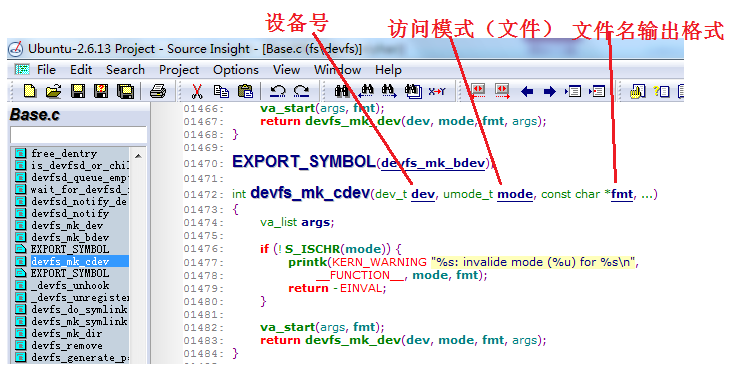
Device FileSystem Make CharDevice



主要实现在文件系统下面建立一个设备文件，设备文件的名称就是DEVICE\_NAME；

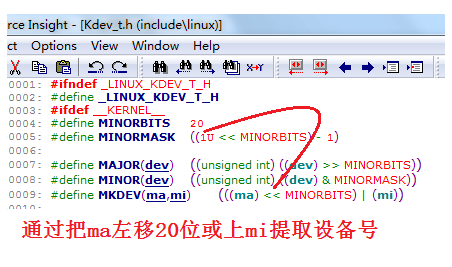


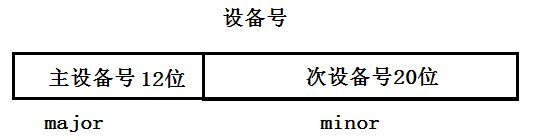
### ###2.5.1 devfs\_mk\_cdev参数怎么用？



### ###2.5.2 MKDEV功能

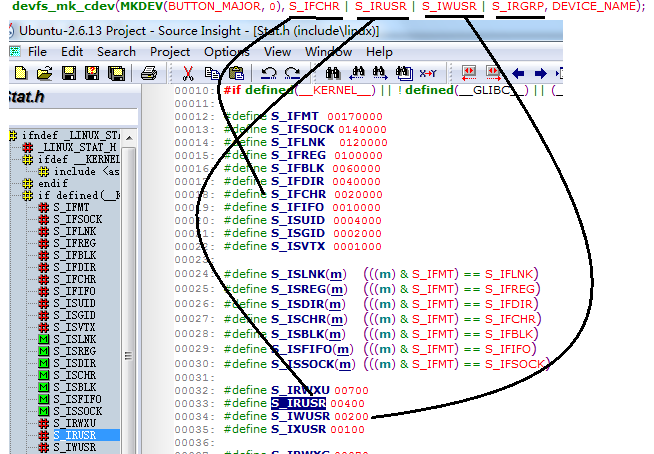
Make Device





### ###2.5.3 访问模式

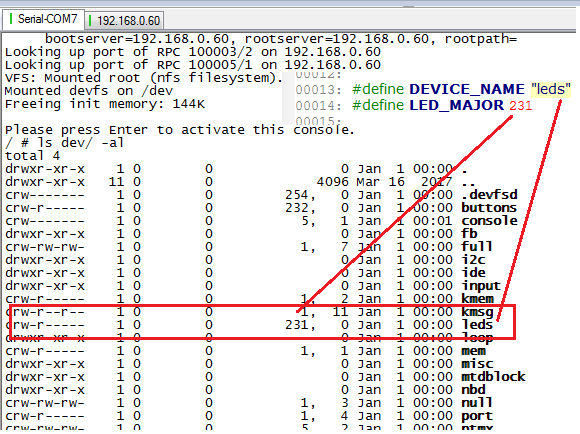
Chmod ,0777,umask



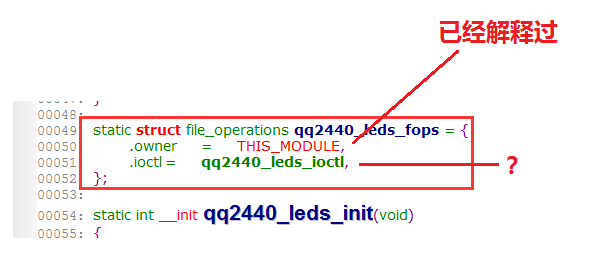
<http://blog.sina.com.cn/s/blog_6dd1df4e0100o50q.html>

### ###2.5.4 现象

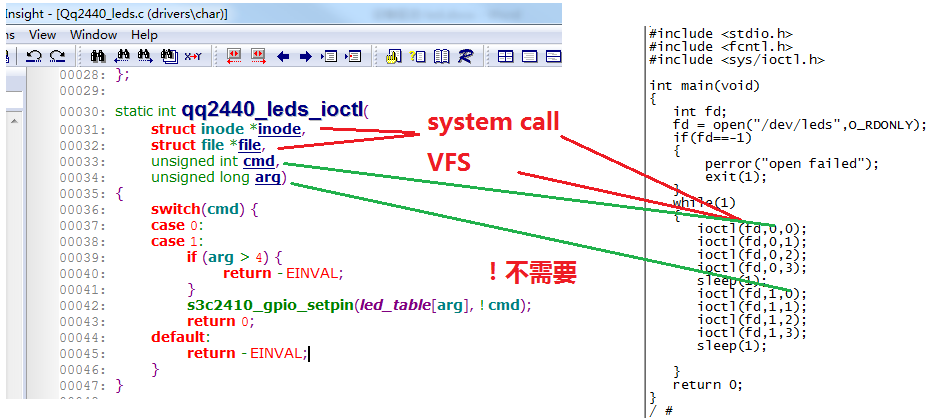
下面是驱动成功注册后，在文件系统中生成的设备文件。



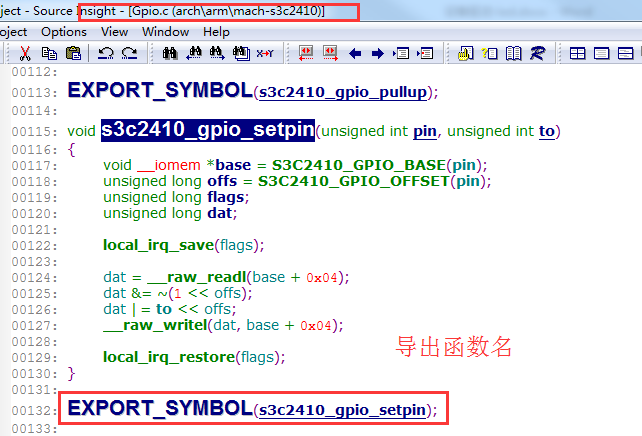
## ##2.6file\_operations结构体如何实现



### ###2.6.1 qq2440\_leds\_ioctl要实现哪些功能？

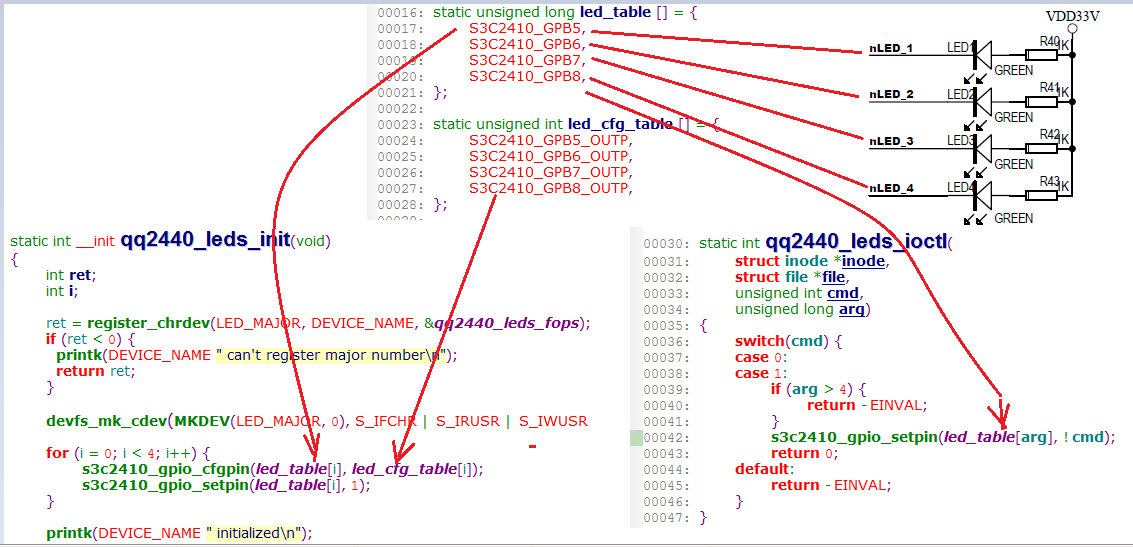


### ###2.6.2 s3c2410\_gpio\_setpin在哪里



## ##2.7引脚控制

这里`s3c2410\_gpio\_cfgpin`和`s3c2410\_gpio\_setpin`就对接上篇文章《ARM接口技术》原理了。



## ##2.8代码调用关系



## ##2.9module\_init函数

参考《linux驱动的入口函数module.docx》

# #附录

工具下载链接：

https://github.com/1040003585/Mini2440/tree/master/Tools

\*\*Wu\_Being 博客声明\*\*：本人博客欢迎转载，请标明博客原文和原链接！谢谢！

《【ARM】Linux驱动移植》

http://blog.csdn.net/u014134180/article/details/72877048

![Wu\_Being 吴兵博客接受赞助费二维码](http://img.blog.csdn.net/20170305211231929?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvdTAxNDEzNDE4MA==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast)

如果你看完这篇博文，觉得对你有帮助，并且愿意付赞助费，那么我会更有动力写下去。