## 移植驱动出现问题如何做？

[SlickEdit](http://www.baidu.com/link?url=z3bOb6ZgwZN2vRyqVl1eCobeI55D5KIqvGsw87zUI4j0c2vPreSMQhBPV4Q2muX3S61wtWKak_ttkOEFsu5bNK)

[移植驱动出现问题如何做？ 1](#_Toc528154462)

[第一节 移植DM9K网卡驱动 2](#_Toc528154463)

[第二节 移植first\_drv驱动 3](#_Toc528154464)

[第三节 移植second\_drv驱动 4](#_Toc528154465)

[第四节 移植third\_drv驱动 5](#_Toc528154466)

[第五节 移植forth\_drv驱动 6](#_Toc528154467)

[第六节 移植fifth\_drv驱动 6](#_Toc528154468)

[第七节 移植fifth\_drv驱动 7](#_Toc528154469)

[第八节 移植sixth\_drv驱动 7](#_Toc528154470)

[第九节 移植buttons\_all驱动 8](#_Toc528154471)

[第十节 移植LCD驱动 8](#_Toc528154472)

[第十一节 移植buttons\_Input驱动 8](#_Toc528154473)

[第十二节 移植触摸屏驱动 8](#_Toc528154474)

[第十三节 移植usb驱动 10](#_Toc528154475)

[第十四节 移植ramblock驱动 10](#_Toc528154476)

[12](#_Toc528154477)

[12](#_Toc528154478)

[第十五节 移植nand驱动 12](#_Toc528154479)

[第十五节 移植Nor驱动 13](#_Toc528154480)

[第十六节 移植虚拟驱动 13](#_Toc528154481)

[移植驱动总结 14](#_Toc528154482)

## 第一节 移植DM9K网卡驱动

搜索DM9000.c文件，找到初始化函数dm9000\_init(),找到dm9000\_driver设备驱动结构体，找到网卡的名字"dm9000",搜索查看找到mini2440初始化函数中注册了该设备，而我们自己

smdk2440中没有注册该设备。

设置内存控制器

在uboot里面已经设置。

添加dm9k设备

#include <linux/dm9000.h>

#define MACH\_SMDK2440\_DM9K\_BASE (S3C2410\_CS4 + 0x300)

/\* DM9000AEP 10/100 ethernet controller \*/

/\* DM9000AEP 10/100 ethernet controller \*/

static struct resource smdk2440\_dm9k\_resource[] = {

[0] = {

.start = MACH\_SMDK2440\_DM9K\_BASE,

.end = MACH\_SMDK2440\_DM9K\_BASE + 3,

.flags = IORESOURCE\_MEM

},

[1] = {

.start = MACH\_SMDK2440\_DM9K\_BASE + 4,

.end = MACH\_SMDK2440\_DM9K\_BASE + 7,

.flags = IORESOURCE\_MEM

},

[2] = {

.start = IRQ\_EINT7,

.end = IRQ\_EINT7,

.flags = IORESOURCE\_IRQ | IORESOURCE\_IRQ\_HIGHEDGE,

}

};

/\*

\* The DM9000 has no eeprom, and it's MAC address is set by

\* the bootloader before starting the kernel.

\*/

static struct dm9000\_plat\_data smdk2440\_dm9k\_pdata = {

.flags = (DM9000\_PLATF\_16BITONLY | DM9000\_PLATF\_NO\_EEPROM),

};

static struct platform\_device smdk2440\_device\_eth = {

.name = "dm9000",

.id = -1,

.num\_resources = ARRAY\_SIZE(smdk2440\_dm9k\_resource),

.resource = smdk2440\_dm9k\_resource,

.dev = {

.platform\_data = &smdk2440\_dm9k\_pdata,

},

};

如果使用内核里面的驱动，则只需要在menuconfig里面选中DM9000。编译就好了。

移植最新的驱动

配置开发板Ip

ifconfig eth0 192.168.1.112

mount –t nfs –o nolock,vers=2 192.168.1.111:/home/chao/work /mnt

## 第二节 移植first\_drv驱动

出现没有头文件

错误提示：

asm/arch/regs-gpio.h: No such file or directory

asm/hardware.h: No such file or directory

解决办法：

一般去掉该头文件后编译

函数没有声明，一般是缺少头文件，或者函数改名字了。

错误提示：

implicit declaration of function 'class\_create'

implicit declaration of function 'class\_device\_create

implicit declaration of function 'class\_device\_unregister'

解决办法：

在新内核中ctrl+shift+f 。在drivers/char 目录下搜索。选中include subdirectories。（子文件夹）。看看别的字符设备驱动怎么使用class\_create，包含了什么头文件。

搜索class\_create，查找使用class\_create的源文件，看看这些源文件包含了什么头文件。如果不知道具体哪个，尽量包含多的头文件，在一个个编译删除不需要的。

#include <linux/of.h>

#include <linux/of\_device.h>

#include <linux/of\_platform.h>

编译后还是出现错误，source insight发现class\_device\_create黑色，没有了该函数。

implicit declaration of function 'class\_device\_create

implicit declaration of function 'class\_device\_unregister'

解决办法

搜索

class\_device\_create 改名为device\_create

class\_device\_unregister名为device\_destroy(firstddrv\_class, MKDEV(major, 0));

忽略返回值警告

错误提示：

ignoring return value of 'copy\_from\_user'

解决办法：

检查返回值，如果不知道怎么用可以查内核里面别人怎么用。

if(copy\_from\_user(&val, buf, count)) // copy\_to\_user();

return -EFAULT;

不兼容的指针类型，一般是结构体类型的名字变了

错误提示：

assignment from incompatible pointer type

解决办法：

在新内核中查找该类型变量，找到类型的新名字。

改为static struct device \*firstdrv\_class\_dev;

## 第三节 移植second\_drv驱动

同first\_drv中的1

同first\_drv中的2

同first\_drv中的3

同first\_drv中的4

## 第四节 移植third\_drv驱动

同first\_drv中的1

同first\_drv中的2

同first\_drv中的3

同first\_drv中的4

变量没有定义

错误提示：

error: 'S3C2410\_GPF0' undeclared here (not in a function)

解决办法

拷贝S3C2410\_GPF到sourceinsight 会发现S3C2410\_GPF(0)。内核中不再使用S3C2410\_GPF0，而是使用S3C2410\_GPF(0)形式。

函数没有声明

错误提示：

implicit declaration of function 'S3C2410\_GPF'

解决办法

缺少头文件

搜索S3C2410\_GPF，看看哪个文件用到，用到的文件里面使用了什么头文件。

包含尽可能多的文件编译，删除，确定具体是需要包含哪个头文件。经过测试

需要包含下面的头文件。

#include <mach/regs-gpio.h>

函数没有声明

错误提示

error: implicit declaration of function 's3c2410\_gpio\_getpin'

解决办法

缺少头文件

搜索s3c2410\_gpio\_getpin，看看哪个文件用到，用到的文件里面使用了什么头文件。

发现几乎没有文件用到该函数。但是发现了和该函数一组的函数。s3c2410\_gpio\_setpin。

搜索s3c2410\_gpio\_setpin看看哪个文件用到，用到的文件里面使用了什么头文件。

包含尽可能多的文件编译，删除，确定具体是需要包含哪个头文件。经过测试

需要包含下面的头文件。

#include <linux/gpio.h>

宏没有声明

wake\_up\_interruptible(&button\_waitq);

wake\_up\_interruptible是一个宏，宏里面用到了TASK\_INTERRUPTIBLE。

错误提示

error: 'TASK\_INTERRUPTIBLE' undeclared (first use in this function)

解决办法

缺少相应的头文件

搜索wake\_up\_interruptible，看看使用该宏的函数包含了那些头文件。

需要包含下面的头文件。

#include <linux/sched.h>

error: implicit declaration of function 'request\_irq'

缺少相应的头文件

搜索request\_irq，看看使用该宏的函数包含了那些头文件。和中断相关的头文件

需要包含下面的头文件。

#include <linux/interrupt.h>

error: 'IRQT\_BOTHEDGE' undeclared (first use in this function)

发现内核中并没有IRQT\_BOTHEDGE。搜索request\_irq找到和IRQT\_BOTHEDGE同组的选项。发现IRQF\_SHARED。IRQF\_SHARED组内并没有和边沿触发有关的东西。但是发现宏可能

开头是IRQF\_XXX。输入IRQF\_利用source insight的自动补全功能发现IRQF\_TRIGGER\_FALLING。并没有定义双边沿触发宏，使用如下代替

IRQF\_TRIGGER\_RISING | IRQF\_TRIGGER\_FALLING

## 第五节 移植forth\_drv驱动

同first\_drv中的1

同first\_drv中的2

同first\_drv中的3

同first\_drv中的4

同third\_drv中的5，6，7，8，9，10

添加如下头文件

#include <linux/poll.h>

#include <linux/of.h>

#include <linux/of\_device.h>

#include <linux/of\_platform.h>

#include <mach/regs-gpio.h>

#include <linux/gpio.h> #include <linux/sched.h>

#include <linux/interrupt.h>

## 第六节 移植fifth\_drv驱动

同first\_drv中的1

同first\_drv中的2

同first\_drv中的3

同first\_drv中的4

同third\_drv中的5，6，7，8，9，10

#include <linux/poll.h>

#include <linux/of.h>

#include <linux/of\_device.h>

#include <linux/of\_platform.h>

#include <mach/regs-gpio.h>

#include <linux/gpio.h> #include <linux/sched.h>

#include <linux/interrupt.h>

## 第七节 移植fifth\_drv驱动

同first\_drv中的1

同first\_drv中的2

同first\_drv中的3

同first\_drv中的4

同third\_drv中的5，6，7，8，9，10

#include <linux/poll.h>

#include <linux/of.h>

#include <linux/of\_device.h>

#include <linux/of\_platform.h>

#include <mach/regs-gpio.h>

#include <linux/gpio.h> #include <linux/sched.h>

#include <linux/interrupt.h>

error: 'button\_lock' undeclared (first use in this function)

错误行：if (down\_trylock(&button\_lock))

button\_lock定义如下：

static DECLARE\_MUTEX(button\_lock); //定义互斥锁

在source insight 发现没有了DECLARE\_MUTEX。那么就不能再这样定义button\_lock了。

解决办法：

搜索down\_trylock，看看新内核里面别人怎么用的？

#include <linux/semaphore.h>

struct semaphore cmd\_mutex;

sema\_init(&lp->cmd\_mutex, 0);

仿照更改：

#include <linux/semaphore.h>

struct semaphore button\_lock;

sema\_init(&button\_lock, 1);//表示资源可用

## 第八节 移植sixth\_drv驱动

同第七节

## 第九节 移植buttons\_all驱动

同第七节

## 第十节 移植LCD驱动

同第七节

更改内核

去掉内核中对LCD的支持，下载新内核。

make menuconfig🡪 Support for frame buffer devices🡪S3C2410 LCD framebuffer support(设为M生成s3c2410fb.ko;该版本内核不会生成另外的三个模块，这三个模块在内核中编译)

# make uImage

insmod lcd.ko

fb\_test /dev/fb0 //测试

## 第十一节 移植buttons\_Input驱动

同第七节

insmod lcd.ko

insmod buttons.ko //该驱动编译出来的模块

cat /dev/tty1

按下 s2键(l)

按下s3键(s)

按下s4(enter)后终端显示ls

修改inittab,添加tty1::askfirst:-/bin/sh

reboot重新启动系统，然后insmod lcd.ko

lcd上显示：Please press Enter to activate this console.

insmod buttons.ko

按下板子的s2,s3,s4 相当于ls命令。lcd显示根文件系统文件夹。

## 第十二节 移植触摸屏驱动

移植同第七节。

insmod s3c\_ts.ko //触摸屏驱动

cat /dev/event0 点击触摸屏secucrt 会有乱码，证明有反应。

移植tslib：

sudo apt-get install autoconf

sudo apt-get install automake

sudo apt-get install libtool

tar xzf tslib-1.4.tar.gz

cd tslib

./autogen.sh

mkdir tmp

echo "ac\_cv\_func\_malloc\_0\_nonnull=yes" >arm-linux.cache

./configure --host=arm-linux --cache-file=arm-linux.cache --prefix=$(pwd)/tmp

make

make install

安装：

根文件系统在网络上,虚拟机上执行

cd tmp

cp \* -rf /nfsroot

根文件系统在开发板，通过nfs挂载临时文件夹，secucrt上执行

cd tmp

cp \* -rf /

使用：

先安装s3c\_ts.ko, lcd.ko

1．

修改 /etc/ts.conf第1行(去掉#号和第一个空格)：

# module\_raw input

改为：

module\_raw input

2．开发板依次执行

export TSLIB\_TSDEVICE=/dev/event0 (event0还是event1根据触摸屏是哪个设备决定)

export TSLIB\_CALIBFILE=/etc/pointercal

export TSLIB\_CONFFILE=/etc/ts.conf

export TSLIB\_PLUGINDIR=/lib/ts

export TSLIB\_CONSOLEDEVICE=none

export TSLIB\_FBDEVICE=/dev/fb0

ts\_calibrate //校准触摸屏

发生错误：selected device is not a touchscreen I understand。

原因是版本号不一致。

内核版本号

#define EV\_VERSION 0x010001

修改编译器版本号:

vim /usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/usr/include/linux/input.h

EV\_VERSION 改为0x010001

重新编译tslib

cd tslib

make clean

make

make install

重新拷贝到开发板并设置。

最后执行测试：

ts\_test //这些命令已经在bin目录中

触摸屏出现数据

0x00075cf7 0x00000003 0x00000018 0

0x00076617 0x00000003 0x00000018 0

0x00076617 0x00000003 0x00000018 0

## 第十三节 移植usb驱动

error: implicit declaration of function 'usb\_buffer\_alloc

mplicit declaration of function 'usb\_buffer\_free'

usb\_buffer\_alloc复制到source insight 发现其调用的是usb\_alloc\_coherent(),而usb\_buffer\_alloc()已经不再是内核函数了。usb\_buffer\_alloc替换为usb\_alloc\_coherent()。

usb\_buffer\_free 替换为usb\_free\_coherent

insmod usbmouse\_as\_key.ko

鼠标连接开发板后执行ls /dev/event\*

发现有输入设备event1出现，event0是触摸屏。

cat /dev/event1

然后按下鼠标左键(l)，右键(s)，中键(enter),即相当于按下ls🡪enter

此时屏幕上显示根文件目录下的文件。

## 第十四节 移植ramblock驱动

error: expected '=', ',', ';', 'asm' or '\_\_attribute\_\_' before '\*' token

出错行：

static request\_queue\_t \*ramblock\_queue;

出错原因：没有request\_queue\_t这个类型了。

ramblock\_queue = blk\_init\_queue(do\_ramblock\_request, &ramblock\_lock);

搜索blk\_init\_queue函数发现返回值类型为struct request\_queue。

所以用struct request\_queue代替request\_queue\_t就行了。

error: expected ')' before '\*' token

出错行：

do\_ramblock\_request(struct request\_queue \* q)

同样的需要用struct request\_queue代替request\_queue\_t

error: implicit declaration of function 'elv\_next\_request'

出错原因：elv\_next\_request函数已经没有了

搜索blk\_init\_queue函数，在该块设备的请求处理函数中发现

elv\_next\_request被blk\_fetch\_request代替了

end\_request(req, 1);被\_\_blk\_end\_request\_cur()取代。

end\_request(req, 1)替换为\_\_blk\_end\_request\_cur(req, 0);

error: 'struct request' has no member named 'sector

error: 'struct request' has no member named 'current\_nr\_sectors

出错行：

unsigned long offset = req->sector \* 512;

unsigned long len = req->current\_nr\_sectors \* 512;

查看内核中怎么使用

unsigned long start = blk\_rq\_pos(req) << 9;

unsigned long len = blk\_rq\_cur\_bytes(req);

更改：unsigned long offset = blk\_rq\_pos(req) << 9;

unsigned long len = blk\_rq\_cur\_bytes(req);

最终更改

static void do\_ramblock\_request(struct request\_queue \* q)

{

struct request \*req;

req = blk\_fetch\_request(q);

while (req) {

unsigned long offset = blk\_rq\_pos(req) << 9;

unsigned long len = blk\_rq\_cur\_bytes(req);

if (rq\_data\_dir(req) == READ)

{

memcpy(req->buffer, ramblock\_buf+offset, len);

}

else

{

memcpy(ramblock\_buf+offset, req->buffer, len);

}

\_\_blk\_end\_request\_all(req, 0);

req = blk\_fetch\_request(q);

}

}

测试

insmod ramblock.ko // 提示ramblock: unknown partition table

mkdosfs /dev/ramblock1 /\*应该先分区再格式化？

按道理格式化应该是针对分区的，每一个分区可以格式化为不同格式。\*/

mount -t /dev/ramblock1 /tmp

cd /tmp

vi 1.txt //写入hello world

cat /tmp/1.txt

cd /

umount /tmp

ls /tmp //此时文件应该不存在，证明成功挂接

mount -t /dev/ramblock1 /tmp

cat /tmp/1.txt

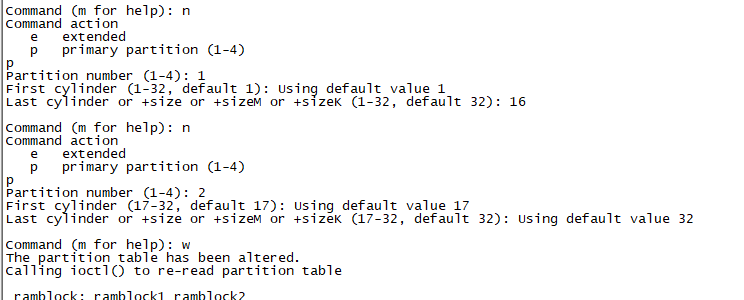
cd /

umount /tmp

fdisk /dev/ramblock //给设备分区

m //先执行disk /dev/ramblock再输入m才能获取fdisk帮助信息

下面是创建两个分区：



ls /dev/ramblock\* //查看创建的分区

/dev/ramblock /dev/ramblock1 /dev/ramblock2

以后可以对/dev/ramblock\*按文件系统方式访问

## 第十五节 移植nand驱动

error: asm/arch/regs-nand.h: No such file or directory

error: asm/arch/nand.h: No such file or directory

解决办法去掉

implicit declaration of function 'add\_mtd\_partitions

error: implicit declaration of function 'del\_mtd\_partitions

搜索add\_mtd\_partitions，该函数存在，但并没有发现内核中nand驱动使用了该函数。

搜索经常同时使用的函数nand\_scan()，发现add\_mtd\_partitions 已经被mtd\_device\_register()取代。del\_mtd\_partitions 被mtd\_device\_unregister(s3c\_mtd);代替。

修改分区信息

static struct mtd\_partition smdk\_default\_nand\_part[] = {

[0] = {

.name = "bootloader",

.size = SZ\_256K,

.offset = 0,

},

[1] = {

.name = "params",

.offset = MTDPART\_OFS\_APPEND,

.size = SZ\_128K,

},

[2] = {

.name = "kernel",

.offset = MTDPART\_OFS\_APPEND,

.size = SZ\_4M,

},

[3] = {

.name = "rootfs",

.offset = SZ\_256K+SZ\_128K+SZ\_4M,

.size = MTDPART\_SIZ\_FULL,

}

};

去掉内核对nand的支持。

insmod s3c\_nand.ko

## 第十五节 移植Nor驱动

## 第十六节 移植虚拟驱动

struct net\_device' has no member named 'hard\_start\_xmit'

搜索register\_netdev()函数。发现函数设置方式改变为：

dev->netdev\_ops = &rtl8139\_netdev\_ops;

static const struct net\_device\_ops rtl8139\_netdev\_ops = {

.ndo\_open = rtl8139\_open,

.ndo\_stop = rtl8139\_close,

.ndo\_start\_xmit = rtl8139\_start\_xmit,

…

};

根据以上变化更改：

vnet\_dev->netdev\_ops=&vnetdev\_ops;

static const struct net\_device\_ops vnetdev\_ops = {

.ndo\_start\_xmit = virt\_net\_send\_packet;

};

error: 'NETIF\_F\_NO\_CSUM' undeclared (first use in this function)

去掉设置//vnet\_dev->features |= NETIF\_F\_NO\_CSUM;

insmod virt\_net.ko

ifconfig vnet0 3.3.3.3

ifconfig

ping 3.3.3.3

ping 3.3.3.4

## 移植驱动总结

卸载模块出现

rmmod: chdir(3.4.20): No such file or directory

解决办法：

创建下面的目录

mkdir /lib/modules/3.4.20

加载模块，卸载模块，启动应用fangf

insmod first\_drv.ko

./firstdrvtest on

rmmod first\_drv