目录

[1.1 bootloader启动内核 2](#_Toc526965687)

[1.2 内核启动应用 2](#_Toc526965688)

[1.3编译内核 2](#_Toc526965689)

[1.4 修改内核分区 3](#_Toc526965690)

[1.5 制作根文件系统 4](#_Toc526965691)

[1.5.1 编译安装busybox 4](#_Toc526965692)

[1.5.2 安装glibc库 4](#_Toc526965693)

[1.5.3 制作etc 目录， 4](#_Toc526965694)

[1.5.4创建设备节点 4](#_Toc526965695)

[1.5.5 创建其他目录 4](#_Toc526965696)

## 1.1 bootloader启动内核

把内存拷贝到内存

设置tag参数

启动（需要传输三个参数r0=0,r1=机器Id，r2=tag地址）

## 1.2 内核启动应用

1> 根据r1判断能否支持该单板，支持则调用相应的启动代码

2> 解析tag参数（主要是内存大小起始地址和bootargs）

3> 装载驱动程序

4> 挂接根文件系统

5> 启动应用程序

## 1.3编译内核

1> tar -xjf linux-3.4.20.tar.bz2

2> cd linu x-3.4.2/

3> 配置编译器

vim Makefile，搜索ARCH修改

ARCH ?= arm

CROSS\_COMPILE ?=arm-linux-

4> 查找相近的配置文件

find –name “\*\_defconfig”

cd arch/arm/configs

ls //显示所有的配置文件

查找到s3c2410\_defconfig

5> 使用s3c2410\_defconfig生成.config文件

回到linux内核根目录下执行

make s3c2410\_defconfig //生成.config文件

make menuconfig

选中EABI选项，是ARM二进制标准调用接口

6> make uImage //编译内核

编译出来的内核地址信息

Load Address: 0x30108000

Entry Point: 0x30108000

7> 虚拟机，把内核映像拷贝到网络文件系统中

cp arch/arm/boot/uImage /work/nfs\_root/uImage\_new

8> uboot 命令行,使用nfs 下载内核到开发板

nfs 32000000 192.168.0.111:/work/nfs\_root/uImage\_new

发现串口输出不对

9>修改晶振时钟：

设定uboot的机器ID,为MACH\_TYPE\_MINI2440启动内核串口输出没有问题，使用MACH\_TYPE\_S3C2440机器ID串口输出有问题。是因为MACH\_TYPE\_S3C2440初始化函数

的晶振时钟和我们的开发板不一样。在arch/arm/mach-s3c24xx/Mach-smdk2440:

MACHINE\_START(S3C2440, "SMDK2440")。

smdk2440\_map\_io

s3c24xx\_init\_clocks(16934400);改为s3c24xx\_init\_clocks(12000000);

uimage支持的机器ID在哪里： include/generated/mach-types.h

10> 编译下载

make clean

make s3c2410\_defconfig

make uImage

如果下载时出现bad crc,可能是分配给内核空间不够。

## 1.4 修改内核分区

设置分区位置：

smdk2440\_machine\_init()

smdk\_machine\_init();

s3c\_nand\_set\_platdata(&smdk\_nand\_info);

smdk\_nand\_sets[]

smdk\_default\_nand\_part[] //分区信息

修改分区和uboot2012中保持一致

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,

}

};

## 1.5 制作根文件系统

3.4.2内核并不支持之前的文件系统，因此需要制作新文件系统。采用busybox.1.20.2.

,下载busybox源码，解压后需要先配置busybox。工作目录为：/work/linux3.4.2

### 1.5.1 编译安装busybox

执行make menuconfig命令配置编译工具链

输入/搜索cross,根据提示找到选项位置。空格选中配置，此时前面出现\*号。

修改默认的i386编译前缀为arm-linux-

修改Makefile:

ARCH ?=arm

CROSS\_COMPILE ?=arm-linux-

执行make 命令编译busybox

安装busybox

make install CONFIG\_PREFIX=/home/chao/download/fs\_mini\_mdev\_new/

### 1.5.2 安装glibc库

找到编译器安装目录

echo $PATH

进入编译器安装目录

cd /usr/local/arm/4.3.2/

find -name lib列出所有库目录：

./arm-none-linux-gnueabi/libc/armv4t/lib

./arm-none-linux-gnueabi/libc/armv4t/usr/lib

拷贝库文件

mkdir /home/chao/download/fs\_mini\_mdev\_new/lib

cp /usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/armv4t/lib/\*so\* /home/chao/download/fs\_mini\_mdev\_new/lib -rf -d

mkdir /home/chao/download/fs\_mini\_mdev\_new/usr/lib -p

cp /usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/armv4t/usr/lib/\*so\* /home/chao/download/fs\_mini\_mdev\_new/usr/lib -rf -d

### 1.5.3 制作etc 目录，

把以前的etc文件夹拷贝过来

cp /work/nfsroot/firstfs/etc/ /home/chao/download/fs\_mini\_mdev\_new/etc -rf

### 1.5.4创建设备节点

cd /home/chao/download/fs\_mini\_mdev\_new/

mkdir dev

cd dev/

sudo mknod console c 5 1

sudo mknod null c 1 3

### 1.5.5 创建其他目录

mkdir proc tmp mnt sys root

1.5.6 制作jffs2映像

mkfs.jffs2 -n -s 2048 -e 128KiB -d fs\_mini\_mdev\_new -o fs\_mini\_mdev\_new.jffs2

烧写：

nfs 30000000 192.168.1.123:/work/nfs\_root/fs\_mini\_mdev\_new.jffs2

nand erase.part rootfs

nand write.jffs2 30000000 260000 $filesize

set bootargs console=ttySAC0,115200 root=/dev/mtdblock3 rootfstype=jffs2

nfs 32000000 192.168.1.123:/work/nfs\_root/uImage\_new

bootm 32000000

发现错误：exit\_code=0x04,\意思是非法指令。主要原因在于内核没有支持EABI接口。

EABI接口是arm程序接口标准。配置内核支持EABI接口。注意需要切换到root权限

才能配置编译内核。

Sudo su

make s3c2410\_defconfig

make menuconfig //搜索EABI找到位置，空格支持配置

make uImage

### 1.5.7 制作yaffs2文件系统

ubuntu安装git :

apt-get install git

获得yaffs2最新的源码

git clone git://www.aleph1.co.uk/yaffs2

查看帮助如何使用

gedit README-linux

找到Integrating YAFFS2 into a Linux 2.6.x kernel。介绍了如何将yaffs2移植到Linux内核

cd yaffs-dir

./patch-ker.sh c m linux-tree

如果已经存在执行下面命令，先移除之前的yaffs,再打补丁

rm -rf linux-tree/fs/yaffs2

./patch-ker.sh c m linux-tree

yaffs2打补丁到内核

cd /home/chao/download/yaffs2

./patch-ker.sh c m ../linux3.4.20

配置内核支持yaffs2

make menuconfig🡪找到yaffs2 file system support 选中

编译修改过的linux

制作yaffs2映像

mkyaffs2image fs\_mini\_mdev\_new fs\_mini\_mdev\_new.yaffs2

使用老的mkyaffs2image编译,yaffs2文件夹编译出来的工具和内核不匹配，不能使用。

暂时不知道怎么改

编译mkyaffs2image工具时出现，未定义u8错误。

yportenv.h添加#define CONFIG\_YAFFS\_DEFINES\_TYPES。就会定义u8别名。

cd yaffs2/utils/

make //生成mkyaffs2image

sudo cp mkyaffs2image /usr/local/bin

sudo chmod +x /usr/local/bin/mkyaffs2image

下载实验出现错误

sudo chown jack common/ -R 取消对文件夹内所有文件的锁

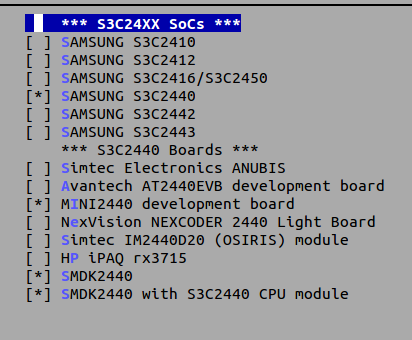
## 1.6 裁剪内核

### 1.6.1 裁剪不需要的cpu类型和单板

查看.config 文件S3C2440 Boards部分，列出了内核支持的单板：

在make menuconfig 里面搜索CONFIG\_ARCH\_S3C2440找到配置项。

### System Type🡪SAMSUNG S3C24XX SoCs Support里面配置，取消不需要的单板。



### 1.6.2 裁剪不需要的文件系统

File systems🡪去掉对ext2,ext3,ext4文件系统的支持。

CD-ROM/DVD Filesystems不支持

Miscellaneous filesystems🡪cramfs不支持，ZLIB不支持，ROM file system support不支持

1.6.3 从.config中查看可以裁剪的部分

CONFIG\_TOUCHSCREEN\_USB\_3M //usb触摸屏

> Device Drivers │

│ -> Input device support │

│ -> Generic input layer (needed for keyboard, mouse, ...) (INPUT │

│ -> Touchscreens (INPUT\_TOUCHSCREEN [=y])

CONFIG\_SSB\_POSSIBLE //USB-based Watchdog Cards

CONFIG\_BCMA\_POSSIBL// Sonics Silicon Backplane

CONFIG\_SND\_USB

1.7 制作内核补丁

cp .config config\_ok

make distclean

mv linux-3.4.2 linux-3.4.2\_100ask/重命名修改好的内核

tar xjf linux-3.4.2.tar.bz2 //解压未修改内核

diff -urN linux-3.4.2 linux-3.4.2\_100ask > linux-3.4.2\_100ask.patch

打补丁：

需要打补丁的内核目录

patch –p1 < ../linux-3.4.2\_100ask.patch

cp config\_ok .config //配置文件

make uImage

1.8 将jz2440作为U盘(没有成功)

修改drivers/usb/gadget/file\_storage.c

static void start\_transfer(struct fsg\_dev \*fsg, struct usb\_ep \*ep,…)

{

int rc;

udelay(800); //增加延时800 us

……

}

/\*修改mod\_data初始值为如下\*/

mod\_data = {

……

.removable = 1,

.can\_stall = 0,

……

};

2.在mach-smdk2440.c中添加gadget设备结构体初始化和 USB device上拉电阻控制，从原理图可以看到GPC5引脚控制上拉电阻。

/\*添加udc头文件支持\*/

#include <plat/udc.h>

static struct platform\_device \*smdk2440\_devices[] \_\_initdata

{

...

&s3c\_device\_usbgadget,/\*Add usb gadget by liwannneg\*/

...

}

...

/\*设置上拉引脚为GPG9\*/

static struct s3c2410\_udc\_mach\_info s3c\_udc\_cfg \_\_initdata = {

.pullup\_pin = S3C2410\_GPG(9),

};

static void \_\_init smdk2440\_machine\_init(void)

{

s3c24xx\_fb\_set\_platdata(&smdk2440\_fb\_info);

s3c\_i2c0\_set\_platdata(NULL);

/\* modify usb gad my liwanneng 2017-4-30 15:40:08 \*/

s3c24xx\_udc\_set\_platdata(&s3c\_udc\_cfg);/\*添加上拉电阻控制\*/

platform\_add\_devices(smdk2440\_devices,ARRAY\_SIZE(smdk2440\_devices));

smdk\_machine\_init();

}

配置 make menuconfig 选项

因为加载file-backed驱动的时候，要添加镜像文件，所以这里我们要选择模块编译

Device Drivers --->

[\*] USB support --->

<\*> USB Gadget Support --->

[\*] S3C2410 udc debug messages(打印内核调试信息)

<M> USB Gadget Drivers （必须选）

<M> Gadget Zero (DEVELOPMENT)

< > Audio Gadget (EXPERIMENTAL)

< > Ethernet Gadget (with CDC Ethernet support)

< > Network Control Model (NCM) support

< > Gadget Filesystem (EXPERIMENTAL)

< > Function Filesystem (EXPERIMENTAL)

<M> File-backed Storage Gadget（必须选）

[ ] File-backed Storage Gadget testing version

< > Mass Storage Gadget

< > Serial Gadget (with CDC ACM and CDC OBEX support)

## 1.9 文件系统和内核ECC一致性

2k页flash,含有64B的oob。2KB需要24字节ecc校验码。40-63放置24B的校验码。

第0字节表示该页是否坏块。每256B数据生成3B的ECC。2K需要3\*8=24B

下面是内核中关于ecc的定义：

static struct nand\_ecclayout nand\_oob\_64 = {

.eccbytes = 24,

.eccpos = {

40, 41, 42, 43, 44, 45, 46, 47,

48, 49, 50, 51, 52, 53, 54, 55,

56, 57, 58, 59, 60, 61, 62, 63},

.oobfree = {

{.offset = 2,

.length = 38} } //从位2开始，总共38B空闲字节可以使用

};

下面是yaffs2文件系统对oob的定义

static struct nand\_oobinfo nand\_oob\_64 = {

.useecc = MTD\_NANDECC\_AUTOPLACE,

.eccbytes = 24,

.eccpos = {

40, 41, 42, 43, 44, 45, 46, 47,

48, 49, 50, 51, 52, 53, 54, 55,

56, 57, 58, 59, 60, 61, 62, 63},

.oobfree = { {2, 38} } }; //在yaffs2里面作为yaffs\_tag