ARM开发板烧写uboot

s3c6410 ARM开发板烧写uboot新手入门笔记

s3c6410 ARM开发板烧写uboot新手入门笔记

ARM开发板是依赖 bootloader启动的,是1段小程序,等同x86系统的BIOS,作用是检测硬件并读取内核到内存
bootloader通常需要开发人员手动烧写到ARM板上,而BIOS通常固化在某个硬件里;
通常bootloader是不用自己写的,别人已写好,最多自己改一下,有时候直接就用了;
嵌入式Linux的bootloader最常用的是U-Boot,版本经常更新;
WinCE的bootloader当然是微软自己写的EBoot
向开发板烧写U-Boot之前,开发板的Nand Flash是空的,没有操作系统,更没有文件系统
向没有文件系统的目标板copy文件的过程也就是"烧写"
为了解决这个问题,三星公司在硬件上提供了一种烧写机制,叫dnw,
就是通过USB线把PC机的U-Boot文件上传到目标板上;dnw是基于libusb标准库做的
同时烧写也需要两端都有软件支持,一端是u-boot(u-boot里有dnw),另一端是一个专门的dnw小软件;

烧写的过程:
①usb线连接pc机和目标板;
②此时目标板是空的,需要设置sd卡启动,事先制做的sd卡有个uboot,这样目标板的uboot就起来了
③在PC端通过超级终端等串口软件操作目标板的uboot,输入命令 # dnw 50008000
　这句话意思是启动目标板的usb连接并设置目标板接收USB数据的内存起始地址为0x50008000
④在PC端启动那个dnw软件,有windows版也有linux版的,道理相同,都需要libusb库支持
⑤PC端dnw软件:与目标板的USB线路连通后,再发送u-boot.bin文件到目标板
　这里发送文件是指发到目标板的内存中,起始地址是0x50008000,注意,这时并没有写到目标板的rand flash
⑥在PC端通过超级终端等串口软件操作目标板的uboot,把目标板内存中的u-boot.bin文件写到rand flash
⑦把目标板内存数据写到rand flash也是uboot命令提供的,其实这时也只有uboot能用;
⑧这里目标板的rand flash里已经烧写好u-boot.bin了,关掉目标板,再设置rand启动就可以了;

在windows下有个dnw软件,是超级终端和dnw和合集用起来很方便,

在Linux下分别用到 minicom 和 dnw 这两个软件
安装minicom # rpm -ivh minicom-2.00-12.i386.rpm
进入minicom # minicom
设置minicom ctrl+A O(选择serial port setup)
A - /dev/ttyUSB0
E - 115200 8N1
F - No
G - No
Save setup as dfl (/etc/minirc.dfl)
ctrl + a x 退出minicom

dnw，包括usb驱动和写入工具

安装secbulk驱动
加载模块到Linux内核： # insmod ./secbulk.ko (注意要在root权限下)
dmesg (查看是否加载成功)
secbulk:secbulk loaded

usbcore: registered new interface driver secbulk (看到这样两行就说明成功了)

感觉不安装这个驱动也没有事,usb通常都是免驱动的啊,可能是在开发板上安的;

SD卡启动

SMDK6410 # dnw 50008000

DNW # ./dnw3 ./u-boot.bin

OTG cable Connected!
Now, wating for DNW to transmit data
Down Done!! Down Address: 0x50008000, Download Filesize:0x30000
Checksum is being calculated.
Checksum O.K.

SMDK6410 # nand erase 0 100000
SMDK6410 # nand write.uboot 50008000 0 100000 //write(.uboot是参数不能改,且只在sd-boot中实现)

nand启动

// 写内核
SMDK6410 # dnw 50008000
DNW # ./dnw3 ./zImage
SMDK6410 # nand erase 100000 500000
SMDK6410 # nand write.e 50008000 100000 500000 //write(.e是参数不能改)

// 写文件系统cramfse,也就是Qtopia2.2.0
SMDK6410 # dnw 50008000
DNW # ./dnw3 ./FORLINX_6410_touch.cramfse
SMDK6410 # nand erase 600000 8000000
SMDK6410 # nand write.e 50008000 600000 8000000

// 写yaffs文件系统,也就是Qtopia4.4.3
copy文件MY6410_yaffs2_v3.0.tar.gz至SD卡
完全启动开发板Linux
SMDK6410 # tar zxvf /sdcard/MY6410_yaffs2_v3.0.tar.gz -C /mnt/disk 或
SMDK6410 # tar zxvf /udisk/MY6410_yaffs2_v3.0.tar.gz -C /mnt/disk
重启进入uboot

SMDK6410 # setenv bootargs root=/dev/mtdblock3 rootfstype=yaffs2 console=ttySAC0,115200
SMDK6410 # saveenvsd
SMDK6410 # reset

//原来的env
SMDK6410 # printenv
bootargs=root=/dev/mtdblock2 rootfstype=cramfs console=ttySAC0,115200
bootcmd=nand read 0xc0008000 0x100000 0x500000;bootm 0xc0008000
bootdelay=1
baudrate=115200
ethaddr=00:40:5c:26:0a:5b
ipaddr=192.168.1.20
serverip=192.168.1.10
gatewayip=192.168.1.1
netmask=255.255.255.0

stdin=serial
stdout=serial
stderr=serial

烧写**uboot**、内核及文件系统的方法

下面总结的是做这个项目使用过的方法，不一定全面，仅供参考。

1、uboot的烧写

下载Uboot分为两个步骤，第一步将uboot下载到系统的扩展RAM并运行，第二步通过内存中运行的uboot把整个uboot下载到内存再烧写到nandflash

第一步下载uboot到扩展RAM

首先使用短路块选择系统从内部启动，复位或者上电就会在串口软件（115200 8 n 1）看到打印的信息LPC31xx READY FOR PLAIN IMAGE>，此时使用串口软件的发送文件选择发送u-boot-init.bin，再发送u-boot.bin，之后就会在RAM中启动uboot，打印输出启动信息，并进行倒计时，此时发送任意字符给系统，停止计时，关闭串口软件，使用超级终端连接系统。

第二步烧写uboot到nandflash

在超级终端中输入loady命令，再使用传送/发送文件，选择Ymodem协议，发送u-boot.bin，接收完成后，先擦除nand erase，可以全擦掉（不用给参数），也可以使用使用参数指定区域，一般第一次要全擦，而后使用nand_params将flash的信息写入其中，最后使用nand write 0x30001000 0x4000 0x100000

其中，0x30001000 是uboot在内存中的地址；

0x4000 uboot在flash中存放的起始地址，需要根据实际的分区情况而定；

0x100000 是uboot的大小，不小于实际大小；

至此，将uboot写入到nandflash中，可以将短路跳线去掉，使其从nandflash启动。

2、内核的烧写

在uboot启动倒计时，击任意键停止，输入loady，与uboot烧写相似，使用的命令主要有：nand erase 0x200000（地址） 0x200000（大小）

nand write 0x30001000 0x200000 0x200000

3、文件系统的烧写

文件系统的烧写，可以通过与内核相似的方法使用串口烧写，也可以通过挂载nfs使用mtd_debug 工具烧

写，由于文件系统较大，使用第一种方式会慢一些。

串口烧写使用到的命令

loady

nand erase 0x600000 0x3a00000（目前的分区情况）

nand write 0x30001000 0x600000 0x800000（实际大小）

使用nfs烧写用到的命令

mtd_debug erase /dev/mtd2 0 0x3a00000

mtd_debug write /dev/mtd2 0 0x800000 ubi.img

各项的含义可以参考mtd_debug 的帮助，直接mtd_debug 即可获取说明，其中的len可以使用十进制数，但是在uboot中是不可以的，不加0x也会认为是十六进制。

4、启动挂载文件系统的选项

在uboot中可以通过环境变量设置启动的选项，一般只需要配置挂载的文件系统是nfs还是ubi，及内核启动选项。

挂载nanflash中的ubifs：

setenv bootargs console=ttyS0,115200n8 ubi.mtd=2 root=ubi0:rootfs rootfstype=ubifs;

挂载129.1.4.199上/rfs/rootfs，并且本机的ip设为129.1.31.33等：

setenv bootargs noinitrd root=/dev/nfs console=ttyS0,115200n8
nfsroot=129.1.4.199:/rfs/rootfs,proto=tcp,nfsvers=3,nolock
ip=129.1.31.33:129.1.4.199:129.1.88.1:255.255.0.0::eth0:off

启动内核的选项：

setenv bootcmd nand read 0x30001000 0x200000 0x200000\; bootm 0x30001000\;

修改过uboot的环境变量都需要saveenv命令保存修改