**東軟機密**

环境构筑手册

**Ver 0.1**

**2010年11月16日**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ページ数 |  | 本文 |  | 付録 |  | 発行日付 |  |
| 作成者 | 吕凯 | | | 承認者 |  | | |

**変更履歴**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版数** | **年/月/日** | **改版内容** | **担当者** | **承認者** |
| 0.01 | 2010/11/16 | 新規作成 | 吕凯 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

※変更種別

A:追加

B:削除

C:変更

**目 次**

[1 编译环境搭建 5](#_Toc277683076)

[1.1 代码路径 5](#_Toc277683077)

[1.2 svn基本使用 5](#_Toc277683078)

[1.3 编译相关 7](#_Toc277683079)

[1.3.1 环境搭建 7](#_Toc277683080)

[1.3.2 代码编译 9](#_Toc277683081)

[1.3.3 运行方法 10](#_Toc277683082)

[1.4 添加一个新项目得方法 10](#_Toc277683083)

[1.5 补充 12](#_Toc277683084)

[2 运行环境搭建 12](#_Toc277683085)

[2.1 pc机端配置 12](#_Toc277683086)

[2.1.1 配置nfs服务 12](#_Toc277683087)

[2.1.2 配置tftp服务 13](#_Toc277683088)

[2.1.3 配置minicom 14](#_Toc277683089)

[2.1.4 准备内核 14](#_Toc277683090)

[2.1.5 准备文件系统 14](#_Toc277683091)

[2.2 开发板设置 14](#_Toc277683092)

[2.2.1 检查环境变量 14](#_Toc277683093)

[2.2.2 设置环境变量 15](#_Toc277683094)

[2.2.3 启动开发板 15](#_Toc277683095)

[2.3 程序运行 15](#_Toc277683096)

[2.4 补充 15](#_Toc277683097)

[3 镜像制作与烧写 16](#_Toc277683098)

[3.1 编译内核 17](#_Toc277683099)

[3.2 制作initrd文件系统镜像 17](#_Toc277683100)

[3.2.1 创建空白镜像 17](#_Toc277683101)

[3.2.2 将文件系统拷入空白镜像 17](#_Toc277683102)

[3.2.3 压缩镜像 17](#_Toc277683103)

[3.3 准备烧写分区 18](#_Toc277683104)

[3.3.1 初始化sdm设备 18](#_Toc277683105)

[3.3.2 初始化mac地址 18](#_Toc277683106)

[3.3.3 创建内核烧写区域 18](#_Toc277683107)

[3.3.4 创建文件系统烧写分区 18](#_Toc277683108)

[3.4 烧写内核和文件系统 20](#_Toc277683109)

[3.4.1 将内核从tftp上拷贝到内核烧写区域 20](#_Toc277683110)

[3.4.2 将文件系统镜像从tftp上面拷贝到文件烧写分区 20](#_Toc277683111)

[3.5 配置uboot环境 20](#_Toc277683112)

[3.5.1 配置内核启动参数 20](#_Toc277683113)

[3.5.2 配置文件系统镜像位置 20](#_Toc277683114)

[3.6 从板子启动加载内核并启动文件系统 21](#_Toc277683115)

[3.7 补充 21](#_Toc277683116)

# 编译环境搭建

本章主要叙述编译环境搭建，以及编译方法。

## 代码路径

http://10.1.29.125/svn/sony/trunk/Source/

应该使用svn对它们检出。

http://10.1.29.125/svn/sony/trunk/Source/Rootfs/diablo/diabloMainCpu

这个是本次base release时候修改得svn目录。

smb://10.1.29.125/SonyFS/src/MediaBlock\_MXFS6\_Release\_20100727-01.zip

svn代码就是基于这个路径得代码修改得,当前已经将最新代码合并到svn里面了。

## svn基本使用

1,检出代码：

$svn checkout http://lv-k@10.1.29.125/svn/sony/trunk/Source/

检出之后，本地就有了一个工作拷贝，可以在此基础上进行修改,修改之后可以把本地修改得部分提交(见后面)到svn服务器上供他人更新(见后面)使用。

检出之时，会提示两次输入用户密码，一次是输入你本地机器的用户名密码，一次是服务器得。

2,提交之前检测：

$svn status

提交之前，可以先检测一下，看自己修改,删除(svn delete)，添加(svn add)了哪些文件，哪些文件没有添加到版本控制，以及做了什么，这个命令输出得含义参考"svn help status",便于在提交之前确认本次提交操作得正确性。

3,提交修改：

$svn commit -m "<你要添加得说明>"

这里，"<你要添加得说明>"将会在后面提到得"svn log"命令输出。

4,查看从前提交的log：

$svn log

这样，会输出每次提交得日志信息。

5,最新版本取出：

$svn update

这一步一般在每次工作（即修改本地工作拷贝）之前，保证自己得修改在最新得版本基础上进行。

6,取出以前版本：

$svn update -r <version>

这里，将会取出以前第"<version>"版得工作拷贝，具体<version>是多少，需要查看"svn log"来确定。

7,添加/删除文件：

$svn add <filename>

$svn delete <filename>

如果我们想要让svn管理新建立得文件，需用"svn add",反之用"svn delete",这两个命令执行完毕之后，需要"svn commit"才能反应到服务器上。

这里，没有add得文件，不会在"svn commit"得时候被提交，可以通过'svn status'查看这个文件是否被add过。

8.查看版本库管理得文件：

$svn ls

9,查看当前路径工作拷贝得全局信息：

$svn info

这条命令可以显示当前路径得工作拷贝在服务器上的检出路径，以及最新修改等信息.

10,撤消修改：

$svn revert

如果提交之前，发现当前修改错了，或者添加/删除文件错了，可以运行这个命令，重新对本地工作拷贝进行修改。

**注意**

以上命令一般都是递归的在当前目录进行，不影响父目录其它文件和目录。以上命令一般可以再添加一个文件/目录参数，对指定得文件/目录进行管理。

使用svn，一般得工作周期是：

1)更新本地工作拷贝

$svn update

2)修改代码

3)添加新增文件(如果需要得话)

$svn add/delete

4)检测修改状态

$svn status

5)提交修改

$svn commit -m ""

更多svn的信息参见:

http://i18n-zh.googlecode.com/svn-history/r734/www/svnbook-1.4/index.html

## 编译相关

### 环境搭建

#### centeros 安装

（省略）

#### yum配置

$sudo vi /etc/yum.conf

添加如下行：

Proxy=http://user:passwd@dl-proxy.neusoft.com:8080/

$sudo yum update

#### 安装编译环境

##### 安装交叉编译工具链

smb://10.1.29.125/share/ALL/SONY-CE-Linux-5.0.4.1-ARM.iso

其中install\_en.txt里面有相关得文档。

##### 编译内核

为了编译内核添加如下路径到环境变量：

/usr/local/arm-sony-linux-gnueabi/devel/bin/

然后下载smb://10.1.29.125/SonyFS/Documents/TechnologyDoc/CELinux/CELinux/diablo/20100326/linux-5\_0\_DIABLO\_20100326-src-kernel.tar.gz

解压之后，

依次运行：

./setup-avp13

make

make kernel\_headers

sudo make kernel\_headers\_install

这样就可以编译应用程序了。

为了编译应用程序，应该添加如下路径到环境变量：

/usr/local/arm-sony-linux-gnueabi/bin

注意：编译内核相关得环境变量添加“/usr/local/arm-sony-linux-gnueabi/devel/bin/“，编译app的环境变量添加“/usr/local/arm-sony-linux-gnueabi/bin”。

为了方便开发，不用每次启动shell时手动添加，可以添加如下到/etc/bashrc中：

#编译内核时：

export PATH=/usr/local/arm-sony-linux-gnueabi/devel/bin/:$PATH

#编译 app时：

export PATH=/usr/local/arm-sony-linux-gnueabi/bin:$PATH

### 代码编译

#### 开发时

每次修改完代码，编译过程如下：

$cd build

$./build.sh

这样会根据时间戳编译最新得内容。在第一次编译前或者增删文件时需要运行./build.sh depend，参照"四、添加一个新项目得方法"

编译得全部log信息存放在build\_Diablo.log

编译如果出错，那么错误信息存放在error.log

上次得编译log以及编译错误分别存放在：build\_Diablo.old.log和error.old.log

清除之前的编译中间文件：

./build.sh clean

#### 发布时

\*编译内核：

$svn checkout http://lv-k@10.1.29.125/svn/sony/trunk/Source/Kernel

$cd Avp13\_Kernel

$ make avp13\_defconfig

$ make

注意，

\*编译内核之前，环境变量设置为"export PATH=/usr/local/arm-sony-linux-gnueabi/devel/bin/:$PATH"

\*生成的内核avp13Build/vmlinux.bin

\*编译app以及生成文件系统镜像：

$svn checkout http://lv-k@10.1.29.125/svn/sony/trunk/Source/Rootfs

$cd For\_Release

$./neu\_release.sh all

注意，

\*运行之前，环境变量设置为"export PATH=/usr/local/arm-sony-linux-gnueabi/bin:$PATH"

\*生成的镜像名称为target/\*.img

\*对于该脚本具体信息，最好参考下For\_Release/readme

### 运行方法

开发时最好用nfs来挂接文件系统，然后进行运行测试，在其它章节中会有详细说明。

大致如下：

编译好的文件在build/diablo.bin下，

需要拷贝diablo.bin到文件系统下，

然后再把osWrapper中得相关库文件拷贝到文件系统下，

然后把新拷贝得osWrapper库路径添加到环境变量LD\_LIBRARY\_PATH中。

运行./diablo.bin

## 添加一个新项目得方法

举例如下：

假设我要添加一个User\_test目录，其中包含test.h test.c,那么：

1)编辑build/incpaths.mk文件

1.1)找到"# Main DIR"注释处，添加定义新目录对应变量的语句如下：

USER\_TEST= $(SRCROOT)/User\_test

1.2)找到"ALL\_PATHS"变量得定义，在其最后添加刚才得新变量如下：

############

$(USER\_TEST) \

2)新建User\_test/usertest.mk文件，内容如下：

USER\_TEST\_SOURCES= \

test.c

USER\_TEST\_OBJS := $(addprefix $(TARGETDIR)/, $(USER\_TEST\_SOURCES:%.c=%.o))

USER\_TEST\_DEPS := $(addprefix $(TARGETDIR)/, $(USER\_TEST\_SOURCES:%.c=%.dep))

vpath %.c $(USER\_TEST)

############

3)编辑build/diablo\_objects.mk文件，在最后添加如下：

include $(USER\_TEST)/usertest.mk

4)编辑Diablo.make,在OBJECTS变量最后追加一条：

$(USER\_TEST\_OBJS) \

5)重新生成依赖：

./build.sh clean

./build.sh depend

6)编译：

./build.sh

可参见上面。

具体参见build.sh脚本。

## 补充

能够进行x86版本编译和调试的diabloMainCpu代码已经更新到SVN，

使用方法如下：

编辑diabloMainCpu/build/build.sh

PLATFORMPARAM="USE\_ARM=yes" //编译ARM版本

PLATFORMPARAM="USE\_ARM=no" //x86 debug版本

这两个版本编译的结果都是diablo.bin文件。

以上,有问题请联系:

Email:lv-k@neusoft.com

# 运行环境搭建

本章主要叙述运行环境搭建，以及运行方法。

这里采用tftp加载内核，挂载nfs文件系统的方法。

其他章节中有直接在开发板flash中启动内核和文件系统的方法。

## pc机端配置

### 配置nfs服务

配置nfs服务得目的是让板通过网络把nfs的目录当做板子环境中得根目录。

(1)编辑/etc/exports

添加类似如下一行：

/mynfs \*(rw,sync,no\_root\_squash)

在配置NFS之前先查看“rpm -q nfs-utils portmap“两个软件包是否安装，默认都是安装的。

这里，'/mynfs'是你本地的nfs目录，可以随意设置。具体含义参见"info exports".

(2)重启服务：

#/etc/init.d/nfs restart

#/etc/init.d/portmap restart

(3)关闭防火墙：

这一步有的人可以不用做。

具体在："系统"->"管理"->"安全级别和防火墙"

至此nfs服务搭建完毕，测试方法如下：

在其它机器上运行：

#mkdir nfstest

#mount -t nfs <你的ip>:<你的nfs目录> nfstest

如果成功则完毕。

### 配置tftp服务

配置tftp服务得目地是让板子通过tftp下载内核并启动。

(1)安装tftp客户/服务端:

sudo yum install tftp.i386

sudo yum install tftp-server.i386

这里，如果不行就先运行"sudo yum makecache"试试。

(2)编辑/etc/xinetd.d/tftp

service tftp

{

socket\_type = dgram

protocol = udp

wait = yes

user = root

server = /usr/sbin/in.tftpd

server\_args = -s /tftpboot

disable = no

per\_source = 11

cps = 100 2

flags = IPv4

}

这里，disable是指关闭还是打开tftp服务，取值yes/no.

server\_args指定tftp服务在本地机器上的导出目录。其它参数具体含义参见“info xinetd.conf”.

(3)重启服务：

#/etc/init.d/xinetd restart

至此tftp服务搭建完毕，测试方法如下：

在本地机器上运行：

#>>/tftpboot/mytest

在其它机器上运行：

#tftp <你的ip地址>

#get mytest

然后"[Ctrl]D"退出，如果在那个其他机器得当前目录看到mytest说明成功。

### 配置minicom

配置minicom的目的是让pc机可以通过minicom连接开发板。

(1)#minicom -s

(2)选择"Serial port setup"

配置其中的如下配置选项：

A - Serial Device : /dev/ttyS0

E - Bps/Par/Bits : 115200 8N1

F - Hardware Flow Control : No

G - Software Flow Control : No

### 准备内核

即将最新内核拷贝到你机器tftp服务的导出目录。

(1)取出内核源码：

svn取出地址应该是:http://10.1.29.125/svn/sony/trunk/Source/Kernel

(2)交叉编译内核

$svn checkout http://lv-k@10.1.29.125/svn/sony/trunk/Source/Kernel

$cd Avp13\_Kernel

$ make avp13\_defconfig

$ make

(3)将编译好得内核(vmlinux.bin)放到tftp的导出目录。

导出目录按前面配置应该是:/tftpboot

cp avp13Build/vmlinux.bin /tftpboot/vmlinux.bin

### 准备文件系统

从svn或smb服务器上面下载文件系统，并拷贝到你配置得nfs目录中去。

svn地址：http://10.1.29.125/svn/sony/trunk/Source/Rootfs

smb地址：smb://10.1.29.125/SonyFS/Development/Release/\*

## 开发板设置

pc端配置好之后，启动minicom，再启动开发板，会自动进入uboot。

### 检查环境变量

先查看默认变量，看配置是否正确：

print

正确输出应该如下：

myip=10.1.29.180

console=on

servip=10.1.29.44

ethaddr=001112334455a966

nfsdir=10.1.29.44:/home/quietheart/nfs

gateip=10.1.29.1

### 设置环境变量

如果输出不对，做如下配置：

setenv myip 10.1.29.180

setenv console on

setenv servip 10.1.29.44

setenv ethaddr 001112334455a966

nfsdir=10.1.29.44:/home/quietheart/nfs

gateip=10.1.29.1

MAC地址设置可能比较特殊，可能如下形式:"diag ethaddr 00:11:22:33:44:55"

注意，这里servip是tftp服务器地址,myip是板子ip，ethaddr是板的mac,这根据自己情况设置。nfsdir和gateip只为保留用，可以不设置。

### 启动开发板

配置之后:

运行

boot -r -t binary -c "console=ttyS0,115200n8 root=/dev/nfs nfsroot=10.1.29.44:/home/quietheart/nfs ip=10.1.29.180::10.1.29.1:255.255.255.0" net:tftp:vmlinux.bin

这里，"nfsroot=10.1.29.44:/home/quietheart/nfs"对应你机器的nfs服务目录，“ip=10.1.29.180::10.1.29.1:255.255.255.0“对应你板子的ip地址，需要修改。

这样，就可以启动开发了。如果在开发板和pc之间通过你的nfs目录相关联，可以通过这个目录在pc和你开发板间传递数据。

## 程序运行

运行程序：

在开发板根文件系统建立一个目录，用来运行编译好得文件,具体如下.

假设目录是/home/test,

1)将编译好的文件build/diablo.bin拷贝到开发板的/home/test下

2)将编译好的库osWrapper/\*.so拷贝到开发板的/home/test/lib下（没有则建立一个）

3)添加库路径到环境变量LD\_LIBRARY\_PATH:"export LD\_LIBRARY\_PATH=/home/test/lib:$LD\_LIBRARY\_PATH"

这样，即可通过“./diablo.bin”运行程序了。

## 补充

无

以上,有问题请联系

Email:lv-k@neusoft.com

# 镜像制作与烧写

本文详细叙述内核编译、文件系统镜像制作、内核烧写、文件系统镜像烧写、以及从烧写的内核和文件系统启动的NBL设置方法。

步骤比较复杂，具体参见后面的"详细内容"，这里先给出操作过程：

一、,编译内核

$svn checkout http://lv-k@10.1.29.125/svn/sony/trunk/Source/Kernel

$cd Avp13\_Kernel

$ make avp13\_defconfig

$ make

二、制作文件系统镜像

$svn checkout http://lv-k@10.1.29.125/svn/sony/trunk/Source/Rootfs

$cd For\_Release

$./build.sh all

三、设置烧写分区

启动板子后，依次运行：

NBL> sdm init

NBL> diag ethaddr 00:11:12:33:44:55

NBL> sdm create -n -l 0x400000 kernel

NBL> sdm create -n -l 0x800000 initrd\_fat

NBL> fdisk -a sdme::

NBL> mkfs sdme1:uvfat:

NBL> mkdir sdme1:uvfat:/boot

四、设置NBL环境

NBL> set boot\_option "console=ttyS0,115200n8 root=/dev/ram0 ip=10.1.29.xx1::10.1.29.1:255.255.255.0"

NBL> set initrd "sdme1:uvfat:/boot/ramdisk.img"

NBL> set console on

NBL> set myip 10.1.29.xx1

NBL> set gateip 10.1.29.1

NBL> set servip 10.1.29.xx2

五、烧写内核和文件系统,启动系统

将文件系统镜像和内核拷贝到tftp导出目录之后，如下执行：

NBL> cp net:tftp:vmlinux.bin sdmd::

NBL> cp net:tftp:ramdisk.img sdme1:uvfat:/boot/

NBL> boot -r sdmd:raw:vmlinux.bin

至此，内核与文件系统从板子里面正常启动了。

## 编译内核

$svn checkout http://lv-k@10.1.29.125/svn/sony/trunk/Source/Kernel

$cd Avp13\_Kernel

$ make avp13\_defconfig

$ make

注意，

\*编译内核之前，环境变量设置为"export PATH=/usr/local/arm-sony-linux-gnueabi/devel/bin/:$PATH"

\*生成的内核avp13Build/vmlinux.bin

## 制作initrd文件系统镜像

### 创建空白镜像

#dd if=/dev/zero of=ramdisk bs=1M count=16

#mkfs.ext2 -m0 ramdisk

### 将文件系统拷入空白镜像

#mkdir tmpmnt

#mount -o loop ramdisk tmpmnt

#cp -a rootfs/\* tmpmnt

这里，rootfs是文件系统目录,注意，rootfs里面的"dev/ram0"不可少。

### 压缩镜像

#umount tmpmnt

#gzip -v9 ramdisk && mv $\_.gz $\_.img

至此，文件系统镜像创建完毕，可以把它拷贝到你的tftp目录,以便烧写:

#cp ramdisk.img /tftpboot

注：

pc版文件系统镜像制作方法略有不同，如下：

(1)（前面方法没有实践）生成pc版本文件系统镜像如下：

#cd rootfs

#find . | cpio --quiet -c -o | gzip -9 -n > ../ramdisk.img

这里，rootfs是适合x86的pc上的文件系统，生成的镜像放在rootfs同级目录下。

(2)运行镜像时需要:

a)#cp ../ramdisk.img /boot

b)修改/boot/grub/grub.conf

修改添加内容参考grub.conf原有内容。

## 准备烧写分区

### 初始化sdm设备

NBL> sdm init

### 初始化mac地址

NBL> diag ethaddr 00:11:12:33:44:55

这个最好不要与别人冲突。

### 创建内核烧写区域

NBL> sdm create -n -l 0x400000 kernel

这里，区域名为kernel,大小指定4M.

### 创建文件系统烧写分区

1）创建文件系统烧写所在区域

NBL> sdm create -n -l 0x800000 initrd\_fat

这里，指定文件系统镜像大小8M。

2)在区域上创建一个分区：

在区域上创建一个分区：

NBL> fdisk -a sdme::

操作之后，会输出如下信息：

Num. Boot Start Length SystemID

1 8 16376 0x83 (Linux native (ext2))

3）在分区上创建fat文件系统:

NBL> mkfs sdme1:uvfat:

4)创建一个目录：

NBL> mkdir sdme1:uvfat:/boot

至此，烧写分区创建完毕。这时候，查看当前分区情况如下：

NBL> sdm list

sdma: nblconfig

device : 0

start sect : 0x88

image size : 0x20000 bytes(0x20 sectors)

protect : Yes

sdmb: sdmreg

device : 0

start sect : 0xa8

image size : 0x20000 bytes(0x20 sectors)

protect : Yes

sdmc: sdmlog

device : 0

start sect : 0xc8

image size : 0x20000 bytes(0x20 sectors)

protect : Yes

sdmd: kernel

device : 0

start sect : 0xe8

image size : 0x400000 bytes(0x400 sectors)

protect : No

sdme: initrd\_fat

device : 0

start sect : 0x4e8

image size : 0x800000 bytes(0x800 sectors)

protect : No

其中sdma,sdmb,sdmc是执行"sdm init"之后就有了；而sdmd,sdme在"sdm create -n -l 0x800000 initrd\_fat"之后就有了.

另外,需要注意：

(1)“准备烧写分区”这个步骤只需要进行一次，以后除非重新布局，否则不用再做。

(2)可以在NBL下运行一些文件操作的命令,例如：

\*创建的目录内容：

NBL> ls -l sdme1:uvfat:/boot

输出如下：

drwxrwxrwx root 4096 2000/01/01 00:00:00 ./

drwxrwxrwx root 32768 2000/01/01 00:00:00 ../

这里，时间是固定的"2000/01/01 00:00:00".

(3)可以操作块，例如:

删除刚才创建的区域sdme：

NBL> sdm delete fat

(4)创建的分区号不要超过sdmp.总大小不要超过64M.

从文档"avp13\_nbl\_spec.txt"的第三节"3. Memory map of NBL"看，总物理内存512M，但是NBL只用其中64M，其他被别的微处理器使用了。

## 烧写内核和文件系统

### 将内核从tftp上拷贝到内核烧写区域

NBL> cp net:tftp:vmlinux.bin sdmd::

这样，内核烧写完毕.实际这个时候如果配置好uboot(即NBL)环境，大家就可以用:

boot -r -c "console=ttyS0,115200n8 root=/dev/nfs nfsroot=10.1.29.44:/home/quietheart/nfs ip=10.1.29.180::10.1.29.1:255.255.255.0" sdmd:raw:vmlinux.bin

从nfs上启动自己的文件系统了。

### 将文件系统镜像从tftp上面拷贝到文件烧写分区

NBL> cp net:tftp:ramdisk.img sdme1:uvfat:/boot/

这里，在我的机器上面，拷贝之后输出如下信息：

i8254x: Ethernet address: 00:11:22:33:44:55

# [if\_i8254x.c] i8254x\_set\_phy\_to\_mac():822: Auto-negotiation failed: MII\_GBCR=0x00000200, MII\_GBSR=0x%, MII\_ANAR=

0x%, MII\_ANLPAR=0x%

i8254x: Ethernet address: 00:11:22:33:44:55

# [if\_i8254x.c] i8254x\_set\_phy\_to\_mac():822: Auto-negotiation failed: MII\_GBCR=0x00000200, MII\_GBSR=0x%, MII\_ANAR=

0x%, MII\_ANLPAR=0x%

这里，需要注意文件系统镜像不要太大。最好用"rm"命令先删除"sdme1:uvfat:/boot"中无用的文件。

## 配置uboot环境

### 配置内核启动参数

NBL> set boot\_option "console=ttyS0,115200n8 root=/dev/ram0 ip=10.1.29.180::10.1.29.1:255.255.255.0"

### 配置文件系统镜像位置

NBL> set initrd "sdme1:uvfat:/boot/ramdisk.img"

这里，我的uboot全部配置如下：

NBL> print

console=on

myip=10.1.29.180

ethaddr=0011223344559966

gateip=10.1.29.1

servip=10.1.29.44

boot\_option=console=ttyS0,115200n8 root=/dev/ram0 ip=10.1.29.180::10.1.29.1:255.255.255.0

initrd=sdme1:uvfat:/boot/ramdisk.img

## 从板子启动加载内核并启动文件系统

启动系统命令如下：

NBL> boot -r sdmd:raw:vmlinux.bin

至此，内核与文件系统从板子里面正常启动了。

另外，如果烧的内核是压缩版的话，例如vmlinux.bin.gz

那么用

NBL> boot -r -d sdmd:raw:vmlinux.bin.gz

也能启动。

## 补充

无。

以上,有问题请联系

Email:lv-k@neusoft.com