Lichee Linux BSP User Manual

Revision History

Version	*Data*	*Author*	*Change Description*
0.1	2011-5-31	Benn <benn@allwinnertech.com>	初稿
0.2	2011-07-19	Benn <benn@allwinnertech.com>	根据实际对文档作了调整
0.3	2011-7-20	Tom <tangliang@allwinnertech.com>	添加 SD 卡启动
0.4	2011-7-26	Tom <tangliang@allwinnertech.com>	增加打包及 uboot 的说明
0.5	2011-9-21	Panglong <panlong@allwinnertech.com>	面对客户发布做了修改
0.6	2011-9-21	Tom <tangliang@allwinnertech.com>	修订一些细节
0.7	**2011-9-26**	Tom <tangliang@allwinnertech.com>	对打包做了说明和修改

1. 概述

本文档用于介绍全志科技 A10 芯片的 Lichee Linux BSP(Lichee 为开发代号，后简称 Lichee BSP)的结构、内部机制以及简单用法。该文档的目的用于指导用户如何定制和使用该 BSP。Lichee BSP 可以从全志科技的客户 ftp 下载。

2. 目录结构介绍

Lichee BSP 主要由 Buildroot(版本 2011.02), Linux kernel(版本 2.6.36)两大部分组成。其中 Buildroot 负责 ARM 工具链、U-Boot、应用程序软件包、Linux 根文件系统和固件包的生成；Linux Kernel 是 Lichee BSP 的核心部分。

2.1. buildroot

它的主要作用是

(1) 管理包之间的依赖关系

(2) 生成 ARM 交叉工具链

(3) 生成 U-Boot

(4) 制作根文件系统，可以包含 strace, directfb, oprofile 等非常丰富的应用软件和测试软件

(5) 生成最终用于烧写的固件包

它的目录结构如下

```
[git@itlab test]$ tree -L 1 buildroot/
buildroot/
|-- board
|-- boot
|-- CHANGES
|-- Config.in
|-- configs
|-- COPYING
|-- dl
|-- docs
|-- fs
|-- linux
|-- Makefile
|-- package
|-- readme.txt
|-- scripts
|-- target
|-- tools
`-- toolchain
```

其中，boot 目录里存放 Boot 代码，config 目录里存放预定义好的配置文件，比如我们的 sun4i_defconfig，dl 目录里存放已经下载好的软件包，scripts 目录里存放 buildroot 运作的代码，target 目录里存放用于生成根文件系统的一些规则文件，在 tools 目录中存放的是用于打包和量产的 pc 工具。对于我们来说最为重要的是 package 目录，里面存放了将近 3000 个软件包的生成规则，我们可以在里面添加我们自己的软件包或者是中间件。 更多关于 buildroot 的介绍，可以到 buildroot 的官方网站 http://buildroot.uclibc.org/获取。

2.2. linux-2.6.36

目录结构如下，

```
[benn@LServer lichee]$ tree -L 1 linux-2.6.36/
linux-2.6.36/
|-- arch
|-- block
|-- build
|-- COPYING
|-- CREDITS
|-- crypto
|-- Documentation
```

|-- drivers

|-- firmware

|-- fs

|-- include

|-- init

|-- ipc

|-- Kbuild

|-- kernel

|-- lib

|-- MAINTAINERS

|-- Makefile

|-- mm

|-- modules

|-- net

|-- output

|-- README

|-- REPORTING-BUGS

|-- rootfs

|-- samples

|-- scripts

```
|-- security
|-- sound
|-- tools
|-- usr
`-- virt
```

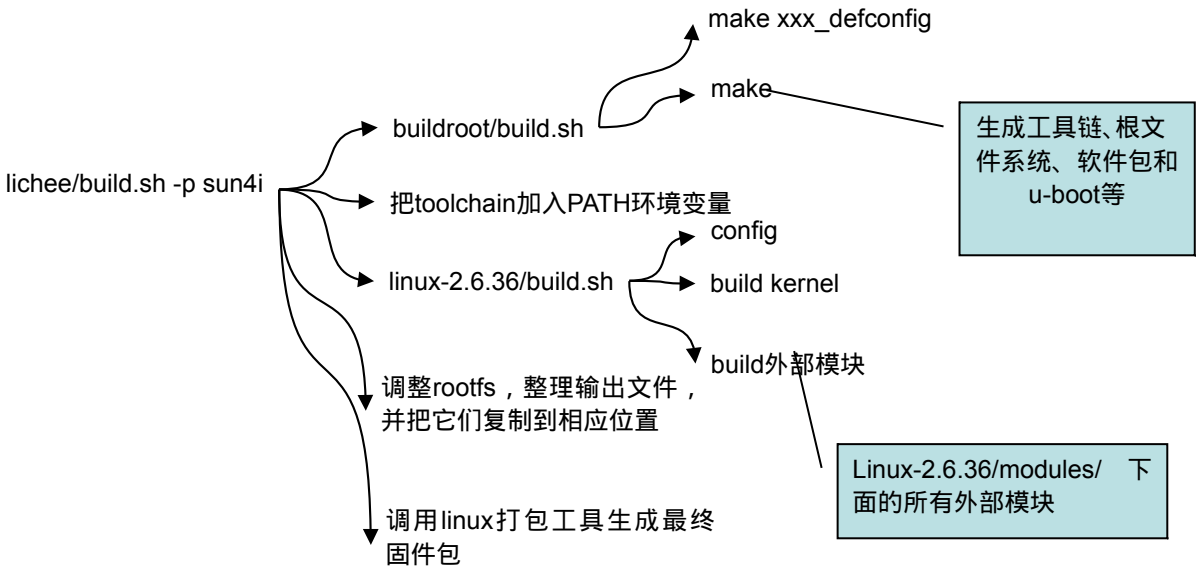
以上目录结构跟标准的 Linux 内核是一致的，除了多一个 modules 目录。modules 目录是我们扩展用来存放没有跟内核的 menuconfig 集成的外部模块的地方。我们目前放了 example, mali 和 wifi 这 3 个外部模块，其中 example 是示例用的，mali 是我们的 3D GPU 驱动。

```
[benn@LServer lichee]$ tree -L 1 linux-2.6.36/modules/
linux-2.6.36/modules/
|-- example
|-- mali
`-- wifi
```

3. 内部工作机制

3.1. 自动化编译流程

以 sun4i 为例子



注意：在执行build.sh 脚本时需要指定参数，具体可以参考./build.sh -h 输出

4. 编译代码

(1) 编译代码

首先将 buildroot/scripts/top_build.sh 拷贝到 lichee 目录下

在 lichee 目录下

$cp buildroot/scripts/top_build.sh build.sh

比如要编译的一个精简版的 Linux BSP 固件，则输入

$cd workdir/lichee

$./build.sh -p sun4i-lite

要编译一个完整版的 Linux BSP 固件，则输入

$cd workdir/lichee

$./build.sh -p sun4i

注意：精简版与完整版的区别在于，精简版包含较少的软件包，编译时间更短。

编译完代码，系统会把生成的用户可能有用的二进制文件复制到 lichee/out 目录下，同时还会把生成的固件包复制到 buildroot/tools/pack 下面的打包工具的工作目录下，并启动 Linux 打包工具生成最终的固件包，比如 buildroot/tools/pack/sun4i_pack_lin/wboot/ePDKv100_nand.img。用户拿到这个固件包后，

直接使用 LiveSuit 工具烧写 Nand Flash 即可。

5. 打包固件

打包是指将我们编译出来的 bootloader，内核，和根文件系统一起写到一个镜像文件中，这个镜像文件也叫固件。然后可以将这个镜像写到 nand flash 或是 sd 卡上，从而启动我们的系统。

5.1. 自动打包

在我们的 bsp 编译系统中，已经为 Linux 下的打包工具添加了自动化打包。在编译代码时，

```
比如要编译的一个精简版的 Linux BSP 固件，则输入
$cd workdir/lichee
$./build.sh -p sun4i-lite

要编译一个完整版的 Linux BSP 固件，则输入
$cd workdir/lichee
$./build.sh -p sun4i
```

已经包含了打包的过程。生成的镜像文件在

workdir/lichee/buildroot/tools/pack/out/ePDKv100_nand.img 里

默认我们打包是 nand 格式的，如果你想打包成 sd 卡格式的，请执行一下操作：

```
cd ~/workdir/buildroot/tools/pack/

./pack sdcard
```

生成的 image 文件在~/workdir/lichee/buildroot/tools/pack/out/ePDKv100_sdcard.img。

5.2. 手动打包

Window 下可以用手动打包方式。打包脚本在

workdir/lichee/buildroot/tools/pack 下，image.bat

window 打包脚本只能用于 nand 打包。

方法：

1.将编译出来的内核 bImage 文件放到 workdir/lichee/buildroot/tools/pack/wboot/bootfs/linux/下(那里可能已经有了一个 bImage，是 build.sh 在编译过程中复制过去的，替换它即可)。整个 bootfs 将打包在 nand 的第一个分区，nanda。

2.将你的根文件系统放在 workdir/lichee/buildroot/tools/pack/wboot 下，并改名为 rootfs.fex，它将被打包在 nand 第二个分区，nandb。

3.双击 workdir/lichee/buildroot/tools/pack 下的 image.bat，然后等待完成即可。生成的文件在 workdir/lichee/buildroot/tools/pack/out/下，ePDKv100_nand.img

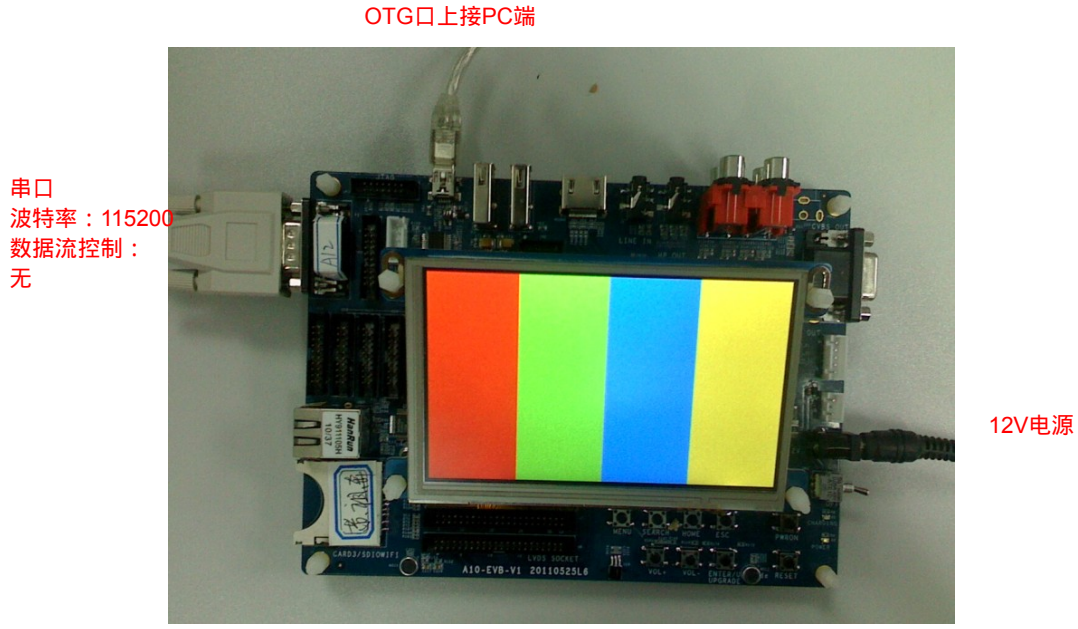
6. 烧写固件

6.1. Nand 启动

这里假设烧写的是 A10-EVB 板，下面是烧写固件包的步骤（从 Nand Flash 启动）

(1) 按照下图连接好开发板

串口、USB 和电源



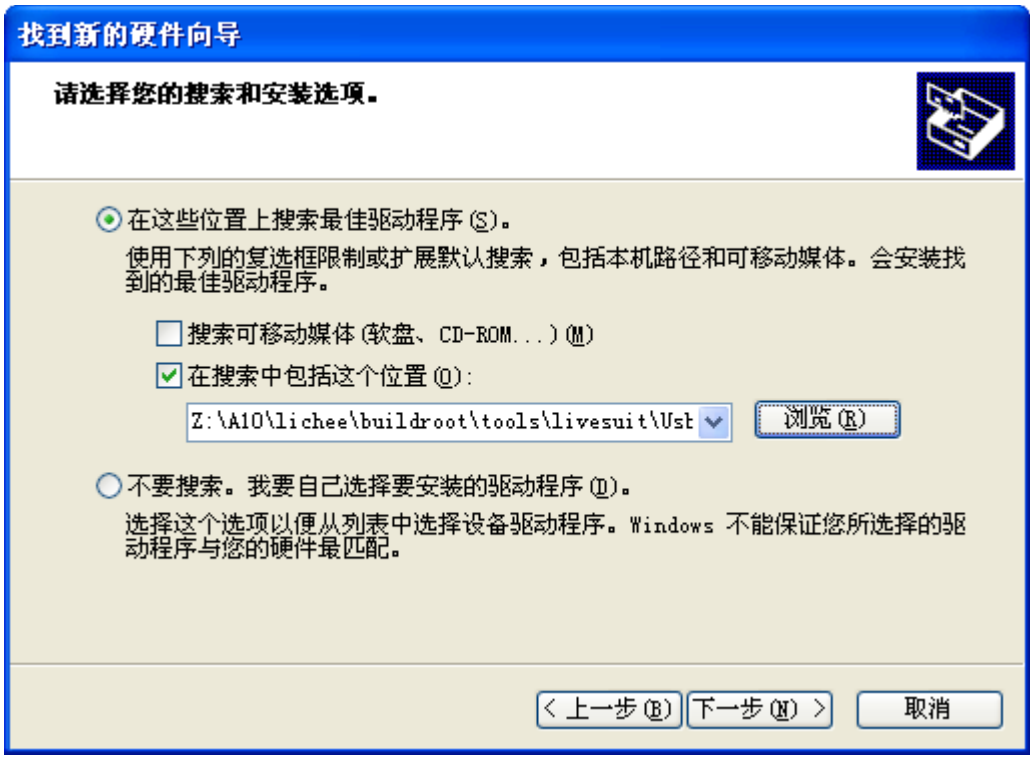
(2) 准备好固件包

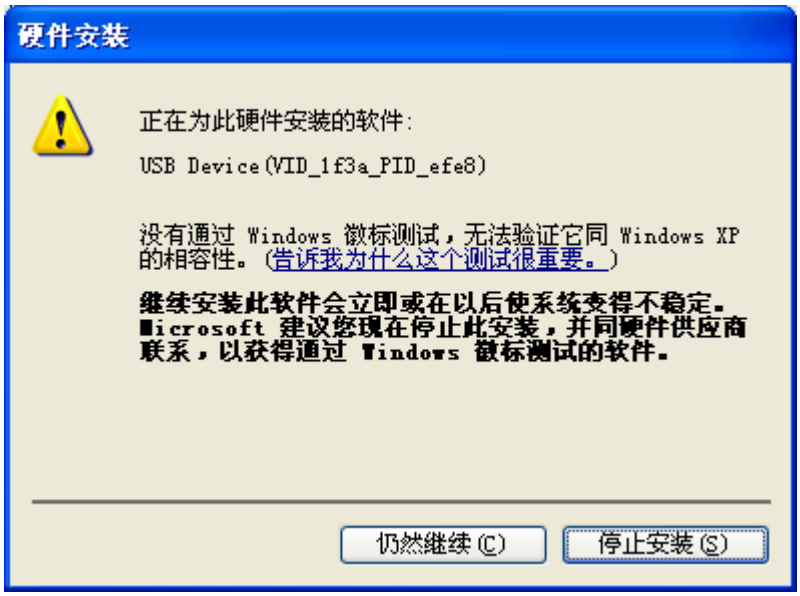
在 5.1 章节的最后，我们得到了 Linux BSP 生成的固件包。

(3) **运行 LiveSuit 工具**

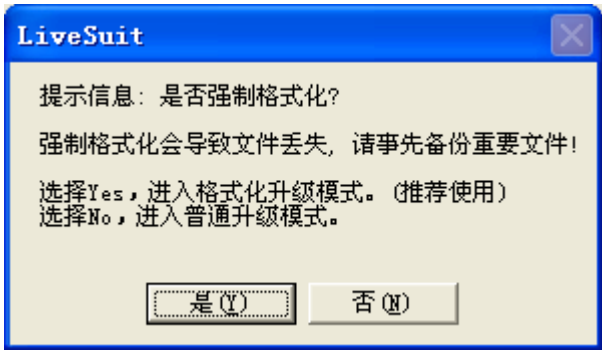
在 buildroot/tools/下面有这个工具，可以从下面把该工具复制到 Windows XP 上。线连接好以后，按 ENTER/UPGRADE 键 + RESET 键(具体是按住 ENTER/UPGRADE 键后再按 RESET 键，先松开 RESET 键，后松开 ENTER/UPGRADE 键)。然后会进入升级模式。

如果第一次使用 LiveSuit，则会提示需要安装 USB 驱动，驱动也放在 buildroot/tools/livesuit/UsbDriver 下面。按照提示安装 USB 驱动，如下图所示

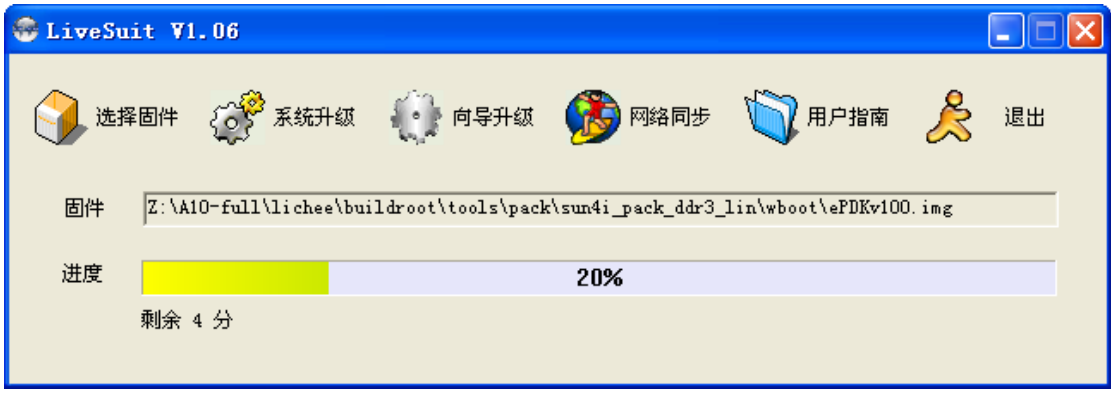




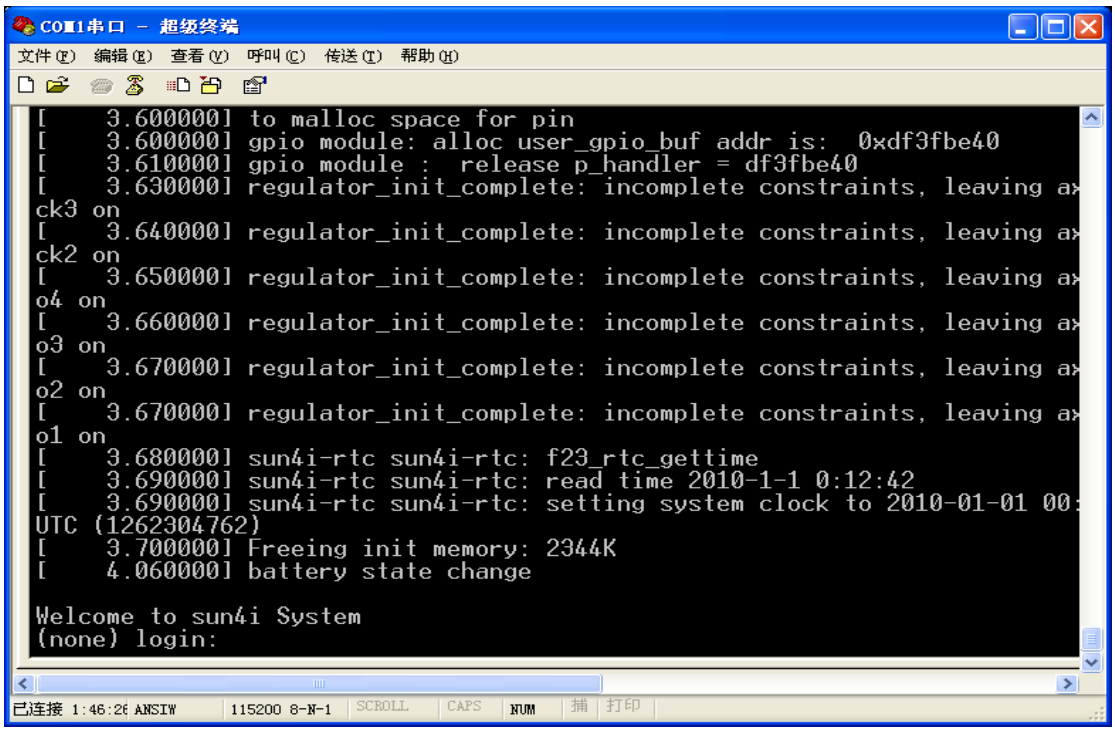
安装完驱动后就进入烧写流程。



在上图的选择中，选择“是”相当与将小机完全格式化后，再烧录固件，而选择“否”，相当于内核升级，一些固件包不涉及的外加的应用还会保留



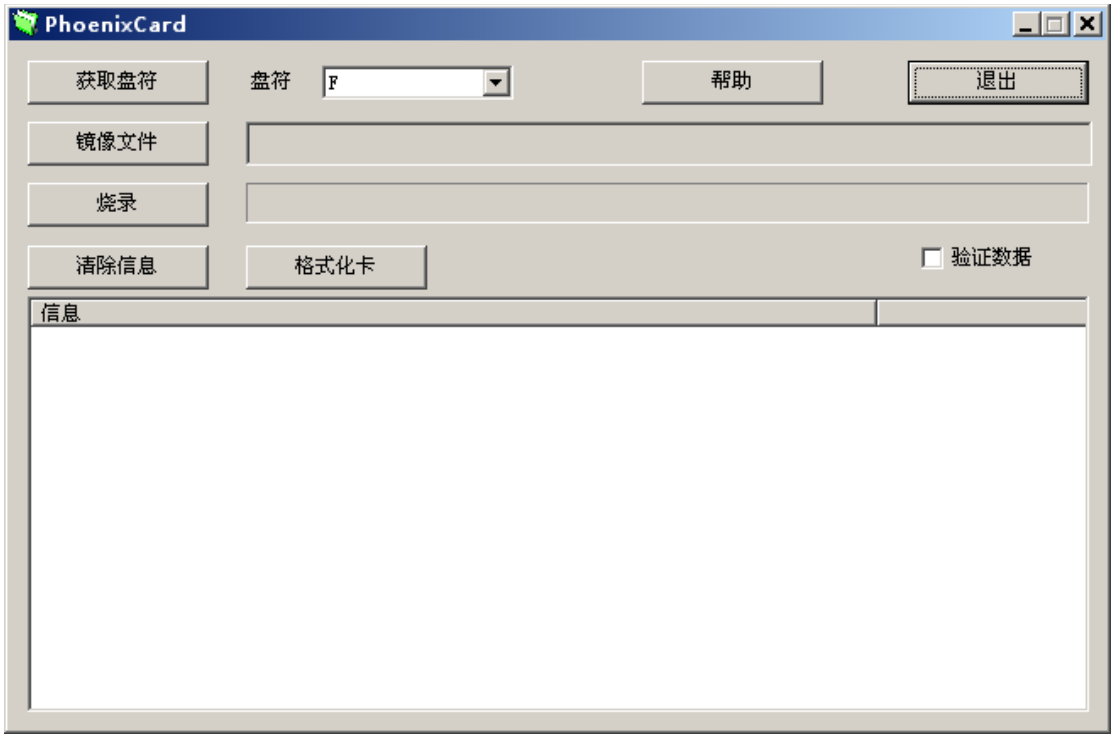
烧写完，按 RESET



用 root 用户登陆，密码为空。

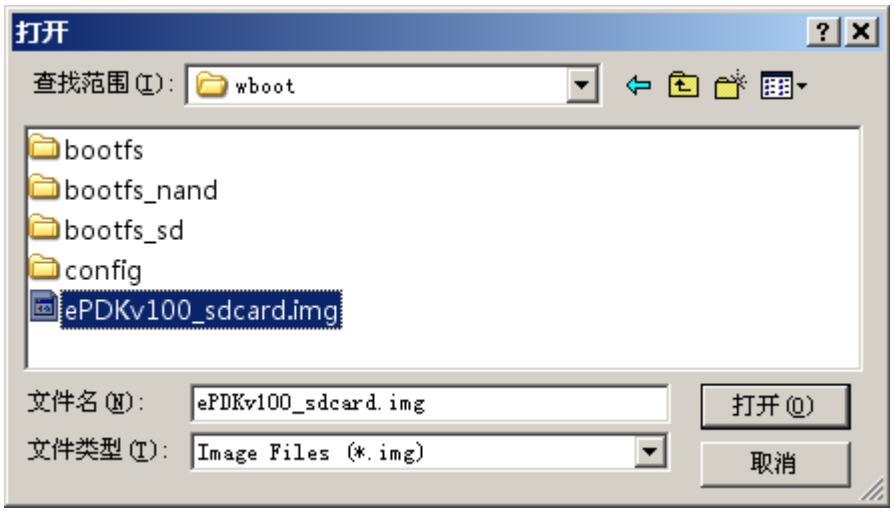
6.2. SD 卡启动

烧写 SD 卡，需要用到 tools 下的 PhoenixCard_V104_20110314 工具。将 SD 卡插在读卡器上，将读卡器插到 PC 上，打开 PhoenixCard 程序。如下图：

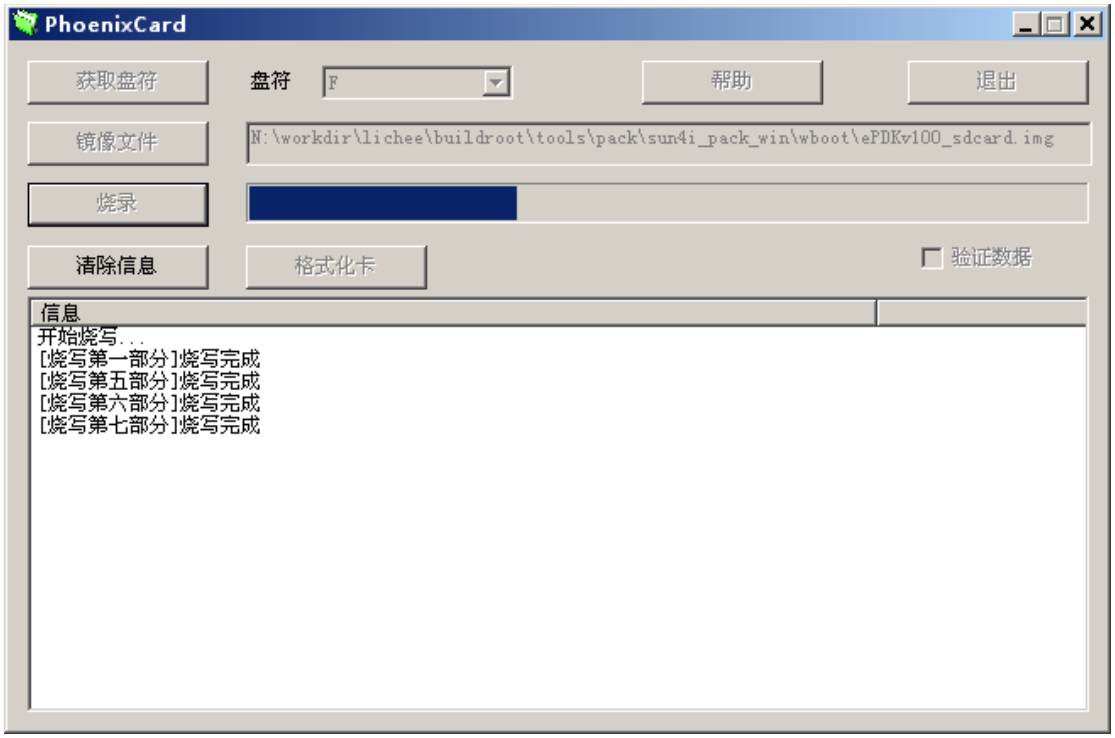


点击获取盘符，找到读卡器所在盘符，应小心核对，以免误写系统到其他盘符。

然后选择镜像文件，注意要选择打包成 SD 卡格式的镜像文件。



然后点击烧录。



等待完成。SD 的烧录就完成了。

将 SD 卡插到开发板的 CARD0 卡槽(在开发板背面)，重启，就可以从 SD 卡启动了。

7. 定制根文件系统

当前 Linux BSP 中使用了 2 级根文件系统。第一级根文件系统和 Linux 内核编译在一起，第二级根文件系统一般烧写到 Nand Flash 的分区中。

7.1. 修改 built-in rootfs

```
$cd lichee/linux-2.6.36/rootfs

$./build.sh e sun4i_rootfs.cpio.gz

此时该目录下多了 skel 目录，里面是 built-in 的根文件系统，直接修改即可。

修改完后

$./build.sh c sun4i_rootfs.cpio.gz
```

更新完 rootfs 后记得需要重新编译 Linux Kernel

7.2. 修改 Nand Flash 的 rootfs

(1) 复制一份现有的配置文件

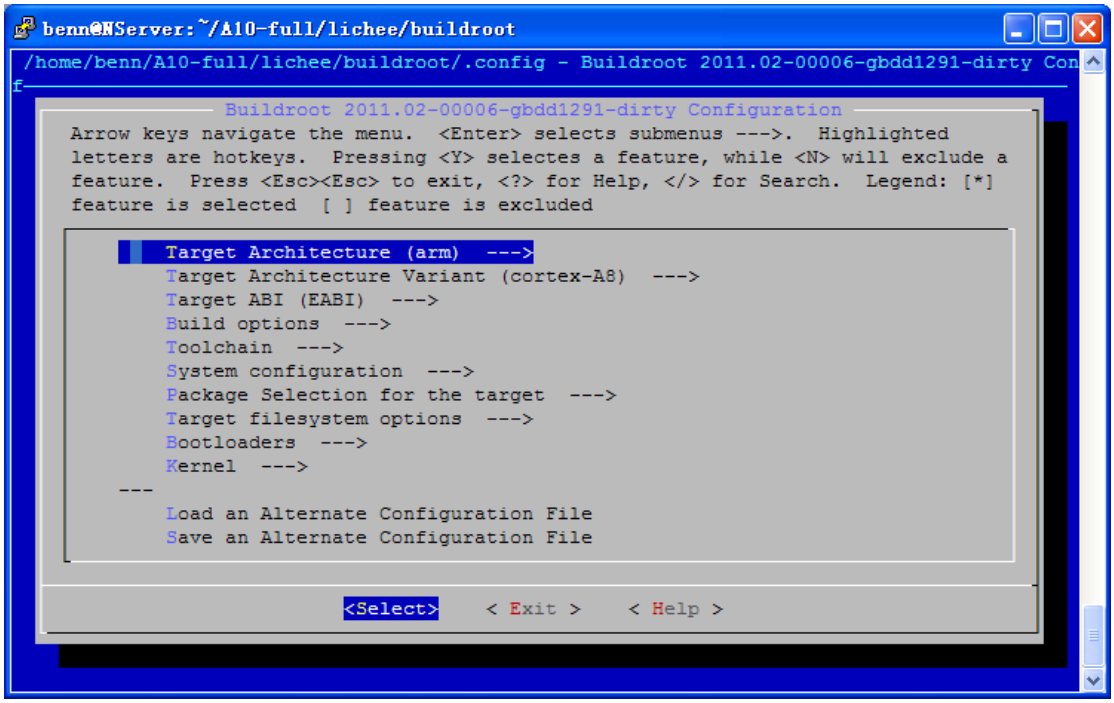
```
$cd lichee/buildroot
```

```
$cp configs/sun4i_defconfig .config
```

(2) 进入 buildroot 界面进行配置

```
$make ARCH = arm menuconfig
```

上述命令后，显示下面的界面



配置完后保存，然后到 lichee 目录下重新运行 build.sh 脚本。

编译过程中，如果有软件包缺失，则 buildroot 会自动从网上下载，而此时如果编译机器无法连接网络，则需要从网上下载相应版本的软件包，把软件包复制到 buildroot/dl 目录下面。

8. 集成软件包

8.1. 源代码包

对于用户态的应用程序、动态库和静态库应该集成到 buildroot 中，在 buildroot/packages 下面 1 个目录对应一个包。关于如何在 buildroot 中集成软件包的说明，请参考 http://buildroot.uclibc.org/docs.html。如果还存在问题请联系本文档作者获得帮助。

对于内核驱动，应该尽量考虑放到 linux-2.6.36/drivers 下面，如果无法直接跟 kernel 的 menuconfig 集成，则应该放在 linux-2.6.36/modules 下面。

8.2. 二进制包

同上，只是忽略掉编译过程。可以参考 buildroot/packages/mali-3d 和 linux-2.6.36/modules/mali

9. 附录

1.1. 关于 U-Boot

我们的 Lichee BSP 已经移植了 u-boot，在 buildroot 目录下运行，make menuconfig，然后在 Bootloaders ---> 选中 u-boot for SUN4I。就可以编译 u-boot 了。如果你选择了 uboot，uboot 会在打包的时候被打包工具打包到镜像中。当 uboot 被打包到镜像中的时候，会由 boot1 启动 uboot，uboot 在 5 秒串口没有输入的情况下接着启动内核。

(1) uboot 可以被打包到 nand 或 sd 卡中启动。

(2) 如果你不想用 uboot，只需要在 buidroot 的 menuconfig 里取消 u-boot for SUN4I。它就不会被打包到镜像中。

(3) 目前 uboot 不支持内核启动参数。所以，如果想给内核设定启动参数，在内核配置里强制使用默认启动参数。

1.2. 附录 1 相关资源

makefile 帮助文档

http://www.gnu.org/software/make/manual/make.html

buildroot 帮助文档

http://buildroot.uclibc.org/downloads/buildroot.html