

**Ingenic ms800平台使用说明**

文档历史：

|  |  |  |
| --- | --- | --- |
| 版本 | 作者 | 注释 |
| 1.0 |  |  |

**目录**

[1 系统简介 3](#_Toc491868762)

[**1.1 源码获取** 3](#_Toc491868763)

[1.2 目录结构 3](#_Toc491868764)

[2 编译说明 4](#_Toc491868765)

[2.1 全局编译 4](#_Toc491868766)

[2.1.1 编译步骤 4](#_Toc491868767)

[2.1.2 生成文件 5](#_Toc491868768)

[2.1.3 运行测试 6](#_Toc491868769)

[2.2 分段编译 7](#_Toc491868770)

[2.2.1 单独编译某个部分 7](#_Toc491868771)

[2.2.2 编译应用程序 7](#_Toc491868772)

[2.2.3 制作文件系统镜像 8](#_Toc491868773)

[3 烧录说明 9](#_Toc491868774)

[3.1 烧录配置 9](#_Toc491868775)

[4 应用程序 12](#_Toc491868776)

[4.1 运行效果 12](#_Toc491868777)

[4.2 应用变更 12](#_Toc491868778)

[5 ADB调试 12](#_Toc491868779)

[5.1 使用说明 12](#_Toc491868780)

[6文件系统 13](#_Toc491868781)

[6.1 制作过程 13](#_Toc491868782)

[6.2 定制文件系统 13](#_Toc491868783)

1 系统简介

ilock平台是由君正基于X1000芯片自主研发设计的，应用于智能门锁等市场的软件方案，使用前请详细阅读本说明文档, 本文档将对如何获取Ingenic ms800平台代码以及软件系统结构、编译、烧录以及应用开发做详细描述。

**1.1 源码获取**

A) 下面以ubuntu系统说明代码获取过程:

1. 获取repo工具: wget <http://git.ingenic.cn:8082/bj/repo>
2. 完后给repo添加可执行权限：chmod +x repo，再把repo的路径添加到环境变量PATH中
3. 安装git和gitk: sudo apt-get install git gitk
4. 创建ssh public key: ssh-keygen -t rsa ，执行这条命令之后（注意这条命令不能以root用户来执行），一路回车即可完成，成功后需要把~/.ssh/id\_rsa.pub 文件内容提供给我们
5. 建立存放平台代码的目录: mkdir mpos; cd mpos
6. 同步初始化：

* repo init -u [ssh://usrname@58.250.243.8:29418/mirror/mpos/manifest.git](ssh://jiangwen@194.169.1.31:29418/mirror/Manhhatan/halley2/platform/manifest.git)
* repo sync

1.2 目录结构

工程库源码下载结束后，通过ls命令查看工程库目录结构:



各级目录主要包含以下内容

app: 此目录为君正提供APP示例。

用户可在此目录下使用SDK开发自己的APP

bootloader: 包含君正自主研发的x-loader启动引导程序

build: 包含工程编译的环境配置脚本和Makefile

device: 包含平台通用文件系统和特定设备的文件系统patch文件及编译配置文件

docs: 包含平台相关说明文档文件

external: 包含平台拓展模块

kernel: 包含linux3.10内核

sdk: 包含sdk源码，详细介绍请参考文档”Ingenic SDK使用说明.pdf”

tools: 包含编译工具链、编译时PC所需的工具以及烧录工具

2 编译说明

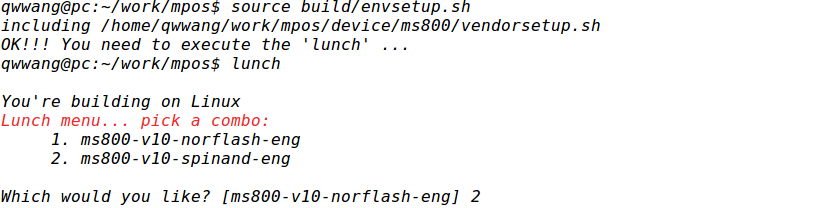
2.1 全局编译

本章主要说明如何自动编译工程源码，以生成烧录镜像、sdk共享库和应用程序

2.1.1 编译步骤

在系统顶级目录执行以下命令:

1. 配置编译环境，在工程根目录执行source build/envsetup.sh，再执行lunch，如下图:

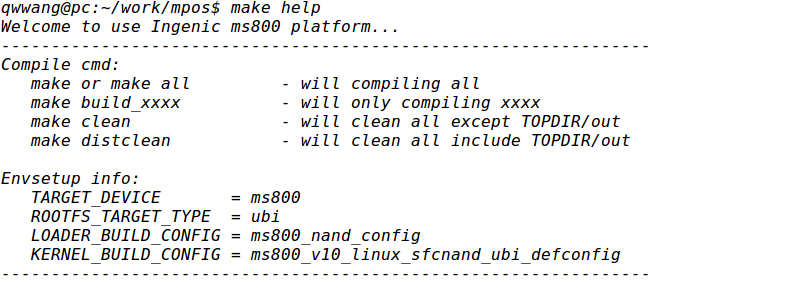


说明:

ms800-v10: 硬件的版本

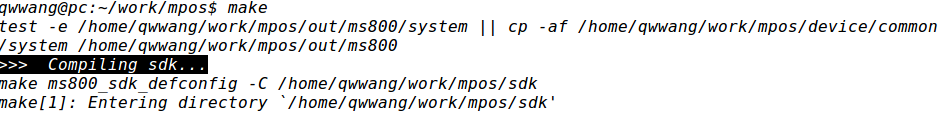
norflash/spinand: 存储介质的类型

1. lunch之后显示选择菜单，下面以nandflash来说明，选择2
2. 选择编译的目标设备和存储介质后，将在工程根目录生成Makefile，可执行make help :

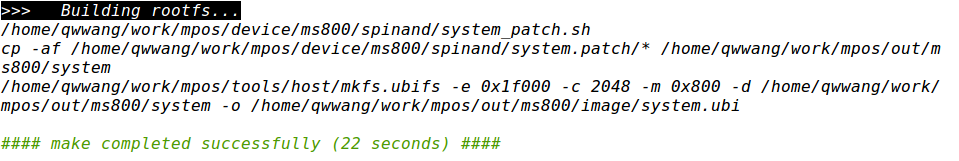


从make help所获取的信息中，我们可以知道编译命令和配置信息，注意make clean和make distclean的区别，更详细请看Makefile文件。

1. 整体编译工程: make



成功编译后的的打印输出，如下图:



说明: 从上图可以看出，全局编译最后阶段是“Building rootfs”，这一阶段包含三个步骤：

1. 调用system\_patch.sh脚步对out/$(TARGET\_DEVICE)/system打patch

2. 把system.patch/目录所有内容拷贝到out/$(TARGET\_DEVICE)/system

3. 调用mkfs.ubifs工具把out/$(TARGET\_DEVICE)/system打包成ubi系统烧录文件

2.1.2 生成文件

编译成功后，生成文件位于out/$(TARGET\_DEVICE)目录下，执行tree –L 2 out/ms800/，如图:



下面具体介绍编译的输出文件:

out/ilock/

├── image

│   ├── system.ubi //文件系统镜像，rootfs分区的烧录文件，对nandflash，这个为system.ubi

│   ├── xImage //编译kernel生成文件，kernel分区的烧录文件

│   └── x-loader-pad-with-sleep-lib.bin //系统的引导程序，bootloader分区的烧录文件

├── sdk

│   ├── app/ //编译app生成的可执行文件的输出目录，跟app/out/一样

│   ├── include //软连接文件，指向sdk供app开发使用的头文件

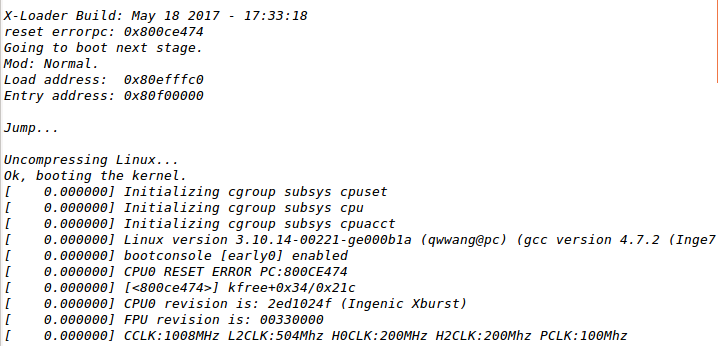
│   ├── lib/ //编译sdk输出的动态库，供app开发使用，会被自动安装到文件系统

│   └── testunit/ //sdk各个模块的测试程序，具体使用方法，请参考源码

└── system/ //文件系统镜像就是使用相应的工具打包此目录生成的

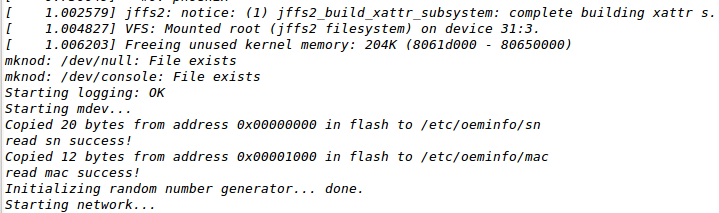
2.1.3 运行测试

通过烧录工具烧录out/ms800/image目录下镜像文件，观察系统启动运行结果，如下:



...

...



从以上的打印信息输开始，标志着kernel已经启动并且mount文件系统完成，可以开始运行用户自定义的程序。

2.2 分段编译

分段编译可以在工程根目录下通过make build\_xxx来进行，也可以到某个部分的目录下进行编译，但这两者是有区别: 通过make build\_xxx编译某个部分，生成的目标文件会被拷贝到out/$(TARGET\_DEVICE)/ 目录下的设定目录，另一种方式则不会拷贝目标文件。

2.2.1 单独编译某个部分

单独编译工程的某个部分首先需要配置编译环境，主要让make能找到交叉编译工具链，参考全局编译前两个步骤即可。

1. bootloader单独编译，以ms800-v10板级nandflash为例:

* 目录: cd bootloader/x-loader
* 配置: make ms800\_nand\_config
* 编译: make
* 目标: out/x-loader-pad-with-sleep-lib.bin
* 说明: x-loader是由君正独立自主开发的引导程序， 串口波特率为3M

1. kernel单独编译，以ms800-v10板级nandflash 为例:
   * 目录: cd kernel
   * 配置: make ms800\_v10\_linux\_sfcnand\_ubi\_defconfig
   * 编译: make xImage –j4
   * 目标: arch/mips/boot/zcompressed/xImage
2. sdk单独编译
   * 目录: cd sdk
   * 编译: make ms800\_sdk\_defconfig; make
   * 目标: out/system/和out/examples/test\_\*
   * 说明: sdk是由君正基于linux开发的一套API程序，API说明请看SDK的说明文档

4）app单独编译

* + 目录: cd app
  + 编译: make
  + 目标: out/ilock\_app

注意: 以上方式编译生成的目标文件都不会拷贝到TOPDIR/out/设定的目录下，在工程根目录使用make xxx 编译某一部分是本文档推荐的分段编译方式。

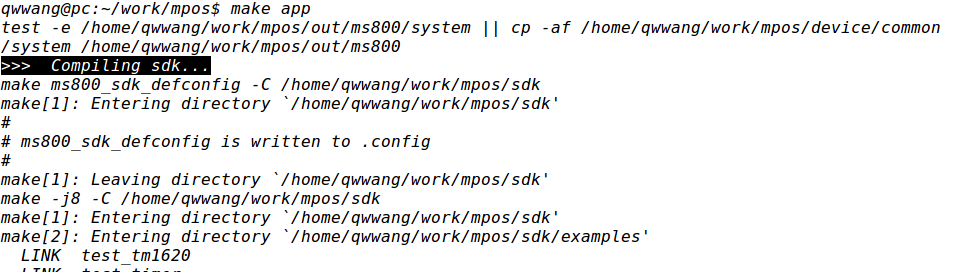
2.2.2 编译应用程序

本节主要说明如何单独编译安装SDK接口库以及应用程序。

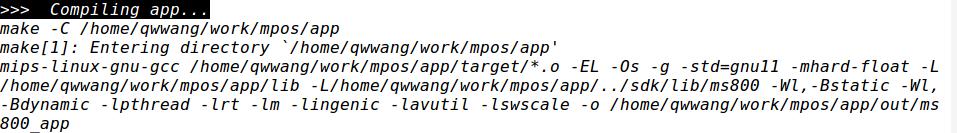
在工程顶级目录执行以下命令:

1. 设置系统环境变量source build/source.sh; lunch （说明: 如果已经设置，此步省略）
2. 清理上次编译结果make clean\_app
3. 开始编译 make app

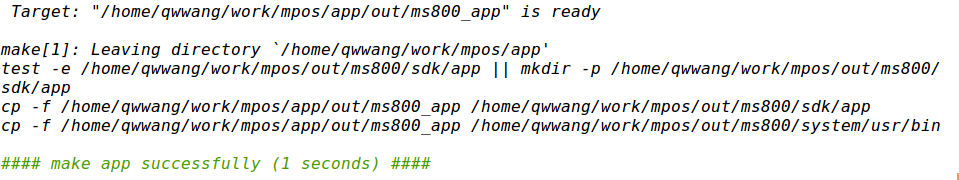
详细请见下图:



…



...



具体包含以下过程:

1. 自动编译sdk接口库源码
2. 自动安装sdk共享库到文件系统usr/lib目录下
3. 自动安装sdk共享库、头文件以及测试用例到out/ms800/sdk相应的目录下
4. 自动安装sdk共享库、头文件到app相应的目录下，以供给应用开发
5. 自动编译app且将生成app拷贝到out/ms800/sdk/app以及文件系统usr/bin目录下

通过以上过程，会得到sdk和app的编译目标文件，并且被安装自动安装到文件系统目录中，再执行make build\_rootfs将可以得到包含sdk共享库和app应用的文件系统镜像

2.2.3 制作文件系统镜像

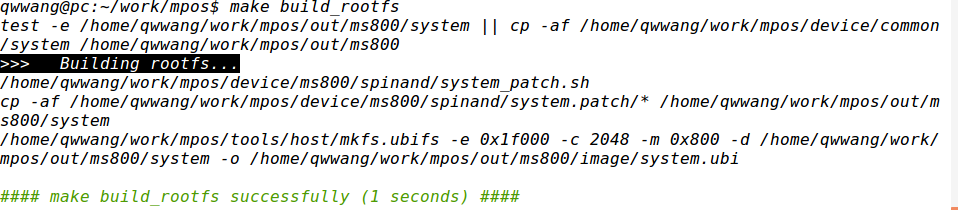
本节主要说明如何单独生成文件系统镜像。

在工程顶级目录执行以下命令:

1. 设置系统环境变量source build/source.sh; lunch （说明: 如果已经设置，此步省略）

2. 开始编译 make build\_rootfs

详细请见下图:

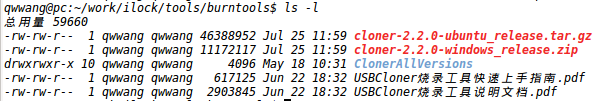
 这条命令的执行过程，通过上图的打印信息已经很清除的表示出来，最终生成out/ilock/image/system.ubi（不同类型的存储介质，制作镜像的工具和命令以及镜像名都不一样）。

3 烧录说明

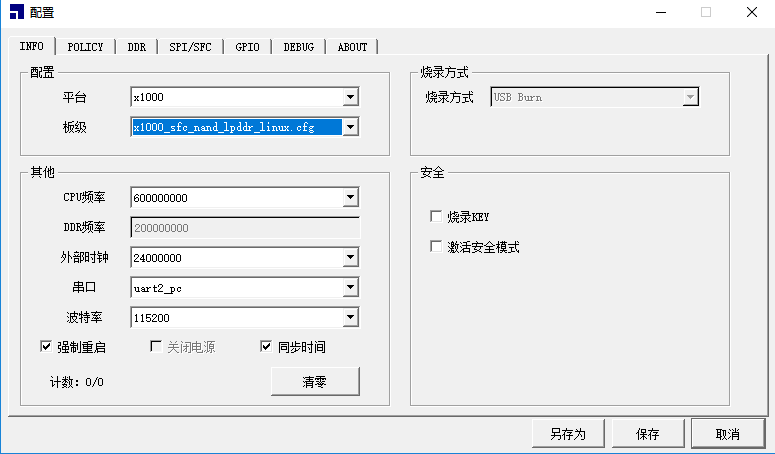
本章将简单说明烧录工具的使用和各个分区及其烧录文件。

3.1 烧录配置

目前最新的烧录工具在tools/burntools/目录下，如下图:

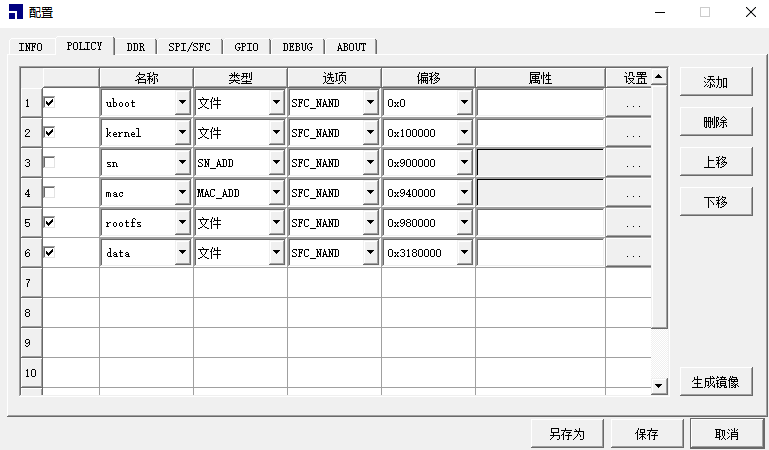


烧录工具详细的使用方法请参考文档《USBCloner烧录工具快速上手指南.pdf》和《USBCloner烧录工具说明文档.pdf》，默认配置文件是x1000\_sfc\_nand\_lpddr\_linux.cfg (注意: 烧录工具版本和配置文件名可能会更新，对norflash的板，配置文件选x1000\_sfc\_nor\_quad\_lpddr\_linux.cfg), 如下图:

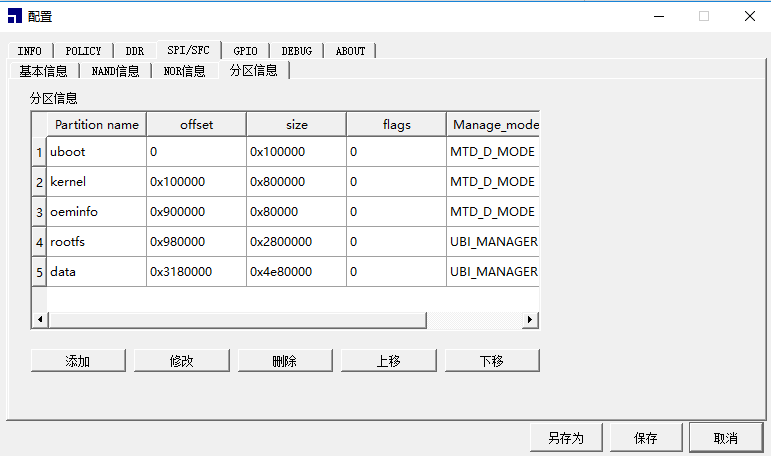


默认是一下5个分区:

1. uboot是bootloader分区，烧录镜像x-loader-pad-with-sleep-lib.bin
2. bootimg是kernel分区，烧录镜像xImage
3. sn和mac同属于oeminfo分区
4. rootfs是文件系统分区, 烧录镜像system.ubi或是system.jffs2, 根据所选文件系统类型来决定
5. data是用户数据分区，用户可以根据需要决定这个分区的用途或者去掉这个分区, 如下图:



分区信息:



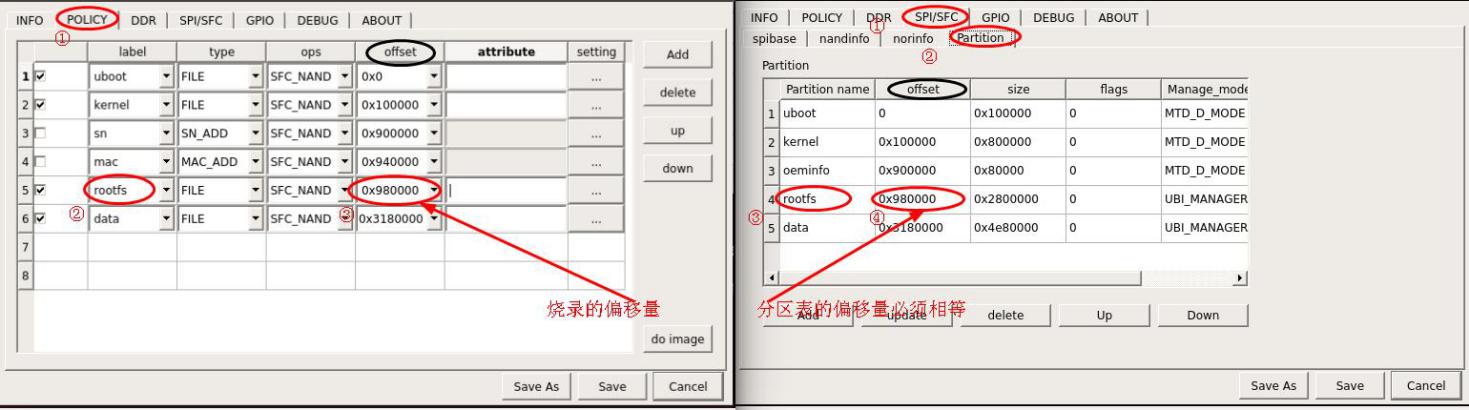
说明:

1. 系统的分区表是通过烧录工具写入到存储介质特定的位置上，kernel启动是会读出分区。

如果需更改分区表的大小、添加或删除分区，注意POLICY和SPI/SFC-→Partition中的

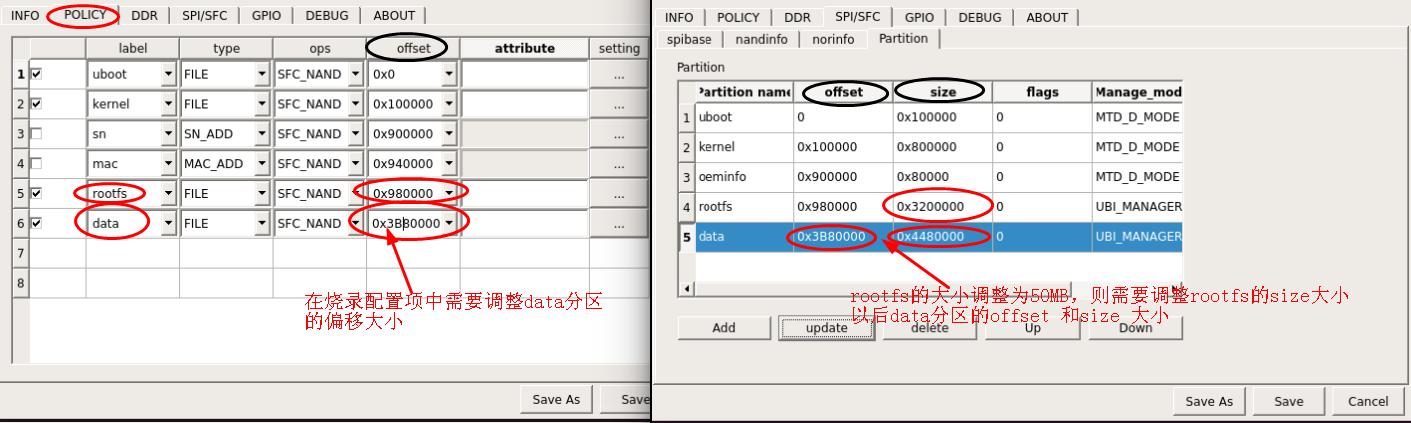
offset必须一致，并且x-loader kernel分区的偏移量、大小以及给kernel的参数也需要做相

应的修改，否则会导致系统启动失败。



例如：如果想将rootfs分区增大(为50MB)，则需要调整rootfs之后所有分区的起始

位置和size大小。如下：



1. 制作UBI文件系统实用工具 mkfs.ubifs.

测试示例如下：

mkfs.ubifs -e 0x1f000 -c 2048 -m 0x800 -d $SRC\_PATCH -o $TARGET\_UBI\_SYSTEM

-m, --min-io-size=SIZE minimum I/O unit size

-e, --leb-size=SIZE logical erase block size

-c, --max-leb-cnt=COUNT maximum logical erase block count

-o, --output=FILE output to FILE

1. 启动自动挂载data分区文件

A 在文件系统中增加data目录

B 在 /etc/fstab文件中添加如下一行，data分区挂在在data目录下

/dev/ubi1\_0 /data ubifs defaults 0 0

4 应用程序

4.1 运行效果

应用程序源码默认路径位于工程根目录的app目录下，其中已经包含了默认的app测试demo程序，建议用户在此目录下开发自己的应用。

4.2 应用变更

用户在app目录下部署安排自己的应用程序，每次变更内容后，请注意可能需要根据修改情况自行**更新Makefile**重新编译。

新编译出来的app应用程序可以通过adb上传到板子上，再到板子运行该程序，即可看程序的运行效果。另一种方法是在工程跟目录执行make app，再执行make build\_rootfs得到包含新app应用程序的文件系统镜像，重新烧录文件系统分区即可。

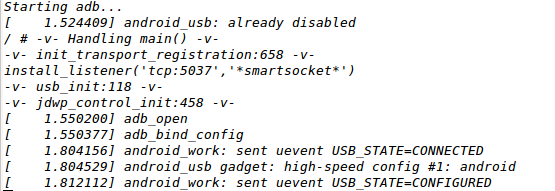
5 ADB调试

5.1 使用说明

adb调试功能用在产品开发调试阶段，主要目的是减少反复烧录的过程，提高程序的开发调试的效率。如何使用adb调试功能？以ubuntu系统为例:

1. 安装adb工具: sudo apt-get install android-tools-adb，（如果已经安装忽略此步，使用虚拟机的需要在windows系统下安装adb）

2. 查看板子串口的系统启动信息，如果有下图的信息，说明支持adb并启动成功:



以调试app应用程序说明adb使用过程，首先adb上传新ilock\_app应用和其依赖库

1. adb push app/out/ilock\_app /usr/bin
2. adb push lib/libingenic.so /usr/lib （如果需要）
3. 完后到串口终端运行刚才的上传的程序

如何把板子上的文件或程序下载到本地PC机？更多adb的使用方法找度娘！

6文件系统

本章将简单介绍整个编译系统制作文件系统镜像的过程以及怎样向文件系统添加用户自定义的文件或程序。

6.1 制作过程

以spinand flash为例，介绍编译系统制作文件系统镜像的过程，如下流程:

1. 在源码根目录执行 make 命令时, 如果out/$(TARGET\_DEVICE)/system目录不存在，首先将 device/common/system 整个目录拷贝到 out/$(TARGET\_DEVICE)目录下;

2. 然后将sdk编译的动态库和app编译的应用程序demo拷贝到out/$(TARGET\_DEVICE)/system的相应目录;

3. 在制作系统镜像之前, 调用device/$(TARGET\_DEVICE)/spinand/system\_patch.sh脚本对out/$(TARGET\_DEVICE)/system目录打补丁;

4.接着把device/$(TARGET\_DEVICE)/spinand/system.patch/目录下的所有内容拷贝到out/$(TARGET\_DEVICE)/system目录下;

5. 最后使用tools/host/mkfs.ubifs工具把out/$(TARGET\_DEVICE)/system目录的内容打包成文件系统镜像;

说明:

norflash的文件系统制作流程也是一样的，不同在于用device/$(TARGET\_DEVICE)/norflash

目录下文件打补丁。

6.2 定制文件系统

了解文件系统镜像制作过程后，在定制文件系统时，用户可以修改system\_patch.sh脚本和修改、添加或删除system.patch目录下的内容。例如让系统启动后自动后台运行ms800\_app应用程序，可以在system.patch/etc/profile脚本中添加一行 /usr/bin/ms800\_app & ，然后执行make build\_rootfs重新制作文件系统镜像，最后烧录新的文件系统镜像即可。