《Android 系统移植和驱动开发》

作者：华清远见

第 1 章 Android 系统的编译和移植实例

本章目标

本章主要介绍 Android 系统的编译和移植技术，作为建立在 Linux 内核基础上的 Android 操作系统，其编译和移植无论是过程还是技术都和嵌入式 Linux 非常相似。本章着重介绍一个典型的 Android 系统编译和移植的实例，该实例类似于编程学习中著名的 HelloWorld，虽然简单，但是涵盖了特定平台下 Android 系统编译和移植的整个流程。如果暂时看不懂也没有关系，随着后面章节的继续深入，就会有豁然开朗的感觉。

1.1 移植背景与目标

现有的环境是一套能够正常运行 Linux 2.6.21 的 EZ6410（基于 S3C6410）硬件系统。移植目标是在 EZ6410 系统上运行 Android 2.3 系统。

1.2 移植涉及的主要过程

- 下载 Android Linux 内核。
- 安装交叉工具链。
- 移植 Android Linux 内核支持 EZ6410 平台。
- 安装 Android SDK。
- 获得 Android 根文件系统。
- 设置系统环境，完成 Android 正常启动。

1.3 下载 Android Linux 内核

目前，支持 S3C6410 硬件的 Android 系统可以在网上找到，网址为 http://code.google.com/hosting/，如图 1.1 所示，可以看到有很多支持 S3C6410 的 Android 项目。

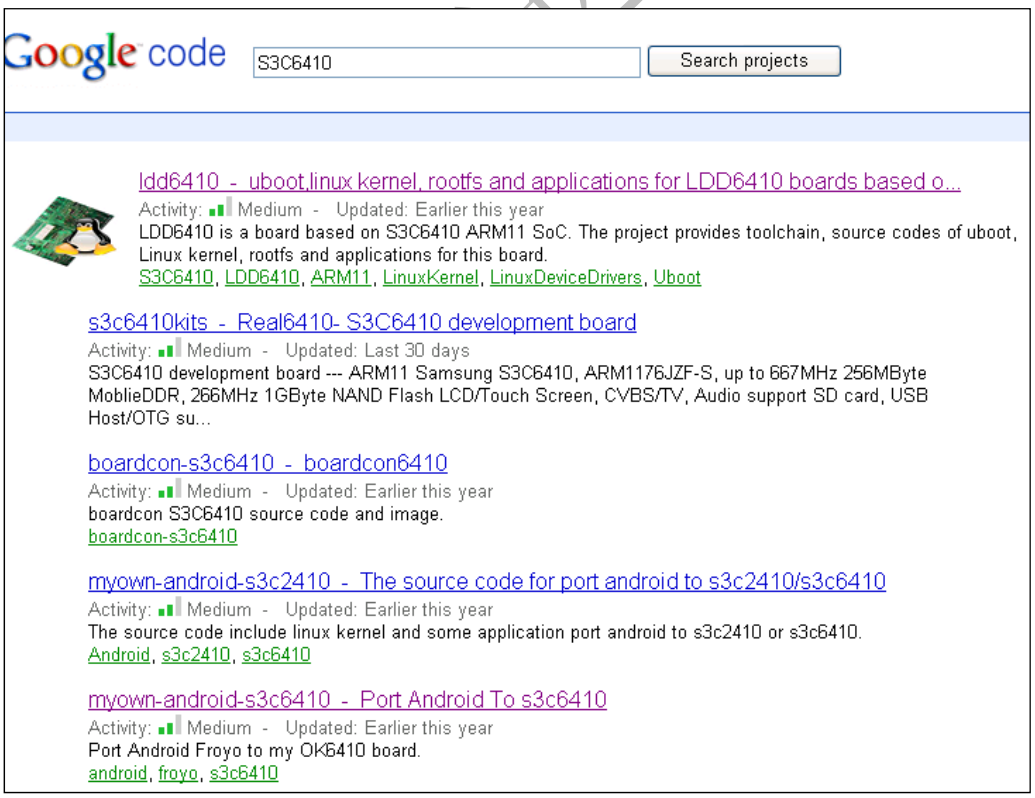

图 1.1 支持 S3C6410 的 Android 项目

单击“ldd6410”链接，打开相应网页。

LDD6410 的硬件结构如图 1.2 所示，我们需要针对其与 EZ6410 硬件结构的差异进行移植。EZ6410 的具体硬件配置请参考开发板手册。

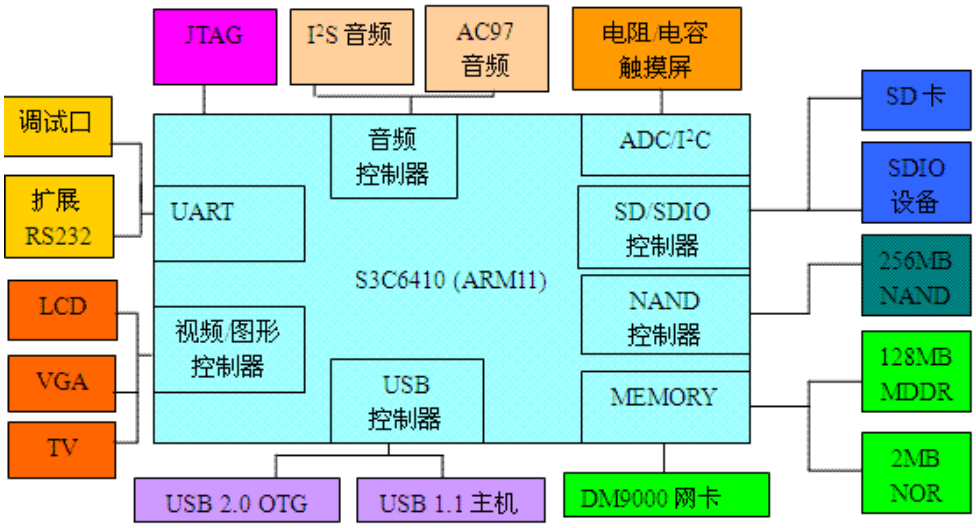

图 1.2　LDD6410 开发板结构图

LDD6410 整合了完整的 Android 驱动（位于 drivers/android 下的 binder、lowmemory killer 等）、内核电源管理（位于 kernel/power 下的 wakelock、userwakelock 等）、ashmem 补丁（mm/ashmem.c）和虚拟电池（drivers/power/fake_battery.c）等。

如图 1.3 所示为 drivers/android 下驱动的配置。

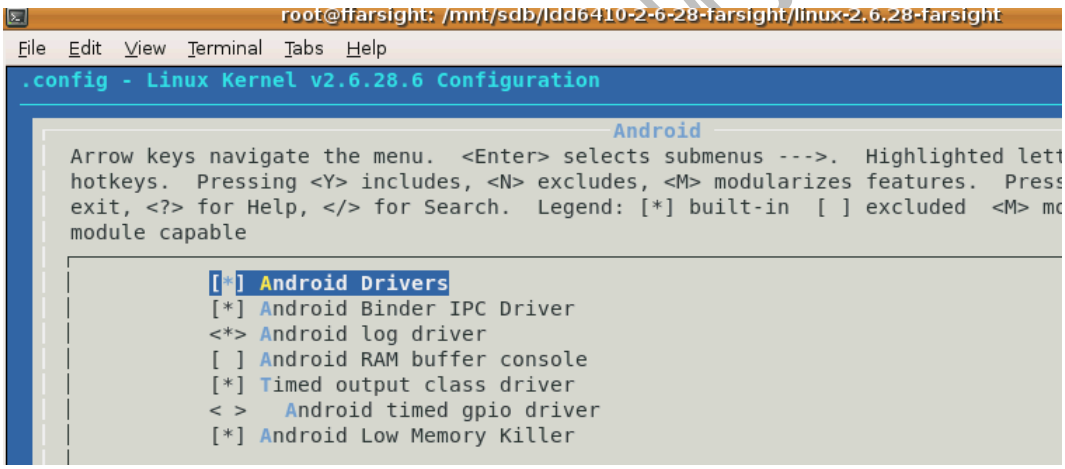

图 1.3　Android 驱动配置

如图 1.4 所示为 kernel/power 下 Android 电源管理的配置。

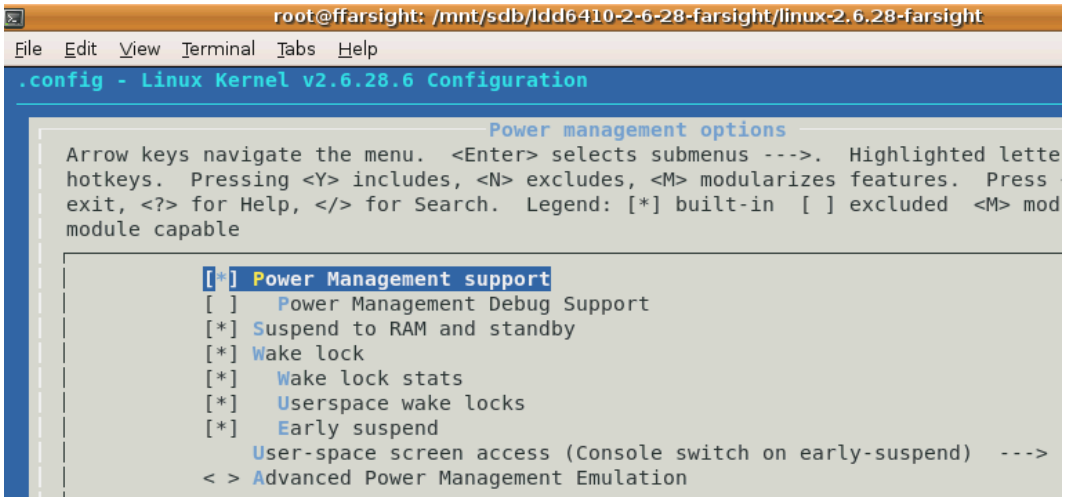

图 1.4　Android 电源配置

1.4 安装交叉工具链

在开始内核移植之前，先完成工具链的搭建。解压 EZ6410 光盘\B\Linux\cross_tools\arm-none-linux-gnueabi.tar.bz2 到 ubantu-8.10 系统的/usr/local/arm 目录下。

在 ldd6410-read-only 目录下修改 vim .cross_compile 内容为：

```
/usr/local/arm/arm-none-linux-gnueabi/bin/arm-none-linux-guneabi-
```

1.5 移植 Android Linux 内核支持 EZ6410 平台

在移植过程中，发现硬件差异主要如下。

- 网卡：LDD6410 平台中的网卡类型是 dm9000，而 EZ6410 平台中的网卡类型是 CS8900a，所以需要移植 CS8900a 驱动到我们的平台上。
- 键盘：键盘接线不一样，需要编写新的键盘驱动。
- 液晶、触摸屏：液晶的品牌、分辨率不一样，需要移植液晶、触摸屏驱动。
- USB 时钟系统：正确使用 USB 功能，需要修改 USB Host 驱动的代码。

1.5.1 CS8900a 驱动移植

将实验代码目录下的 cs89x0.c 复制到内核的 drivers/net/目录下。配置内核支持 cs8900a9A71 驱动，代码如下：

```
#vim Makefile
```

在 obj-$(CONFIG_DM9000) += dm9000.o 下添加 obj-$(CONFIG_CS89x0) += cs89x0.o

```
#vim Kconfig
tristate "CS89x0 support"
depends on NET_ETHERNET && (ISA || EISA || MACH_IXDP2351 \
      || ARCH_IXDP2X01 || ARCH_PNX010X || MACH_MX31ADS)
```

其中的 depends on 加上 ||ARCH_S3C64XX||MACH_6410。

如图 1.5 所示，选择 CS8900 网卡驱动配置项，这样就完成了配置内核支持 CS8900a 驱动的工作。

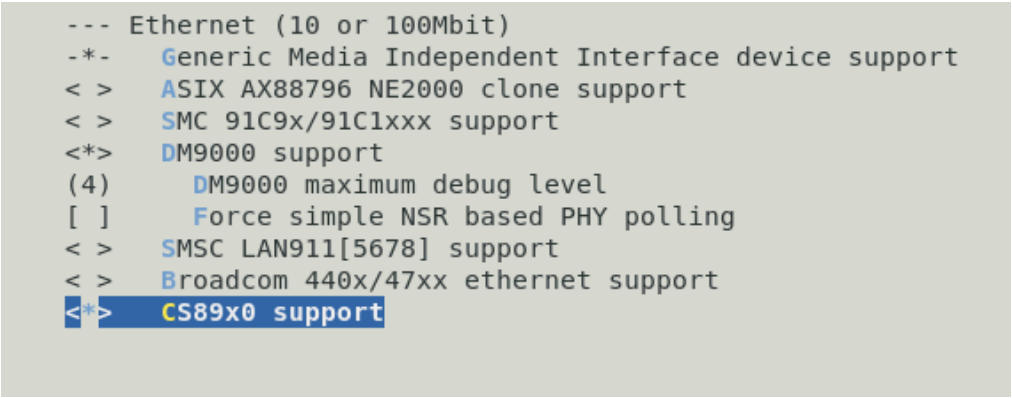

图 1.5 配置 CS8900 网卡驱动

1.5.2 键盘驱动编写

针对 Andorid 要求，编写键盘驱动。EZ6410 实验平台上有 8 个按键，即 K1~K8。

Linux 系统提供了 Input 子系统，按键、触摸屏、键盘、鼠标等输入都可以利用 Input 接口函数实现设备驱动，因此，按键和触摸屏设备驱动都可以作为 Input 设备驱动而实现。

如图 1.6 所示为键盘原理图。

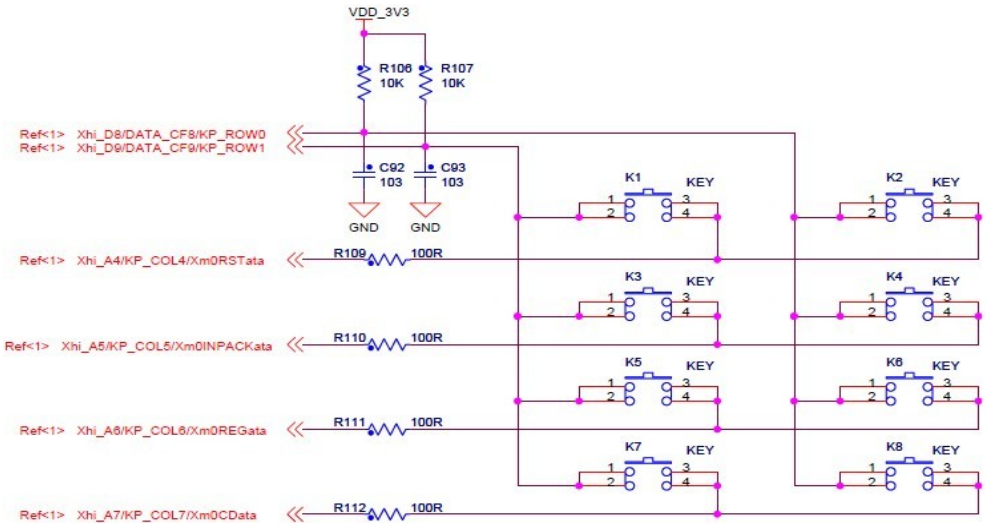

图 1.6　键盘原理图

Android 系统键值要求可以参见 Android 源码，具体的源码文件路径是 system/usr/keylayout/qwerty.kl，我们用 vim 打开这个文件：

```
#vim system/usr/keylayout/qwerty.kl
#define BACK          158
#define SOFT_RIGHT     60
#define MENU          229
#define SEARCH        127
#define HOME          102
#define DPAD_CENTER    232
#define DPAD_DOWN     108
#define DPAD_UP       103
#define DPAD_LEFT     105
#define DPAD_RIGHT    106
……
```

我们实现其中的 8 个按键，对应关系如下：

K1(RIGHT)	K2(CENTER)	K3(MENU)	K4(DOWN)
K5(HOME)	K6(UP)	K7(BACK)	K8(LEFT)

代码参见“实验代码\键盘驱动代码”目录下的 key_drv.c。

1.5.3　液晶驱动

修改 LDD6410 中的液晶驱动支持 EZ6410 平台的 320×240 的液晶屏。

修改的文件为：drivers/video/samsung/s3cfb_wanxin.c。

修改如下：

```
#elif defined (FB_320_240)
/* 320*240 */
#define S3CFB_HFP     30  /* front porch */
#define S3CFB_HSW     41 /* hsync width */
#define S3CFB_HBP     20 /* back porch */

#define S3CFB_VFP     15   /* front porch */
#define S3CFB_VSW     10   /* vsync width */
#define S3CFB_VBP     10  /* back porch */

#define S3CFB_HRES    320    /* horizon pixel  x resolition */
#define S3CFB_VRES    240 /* line cnt      y resolution */

#define S3CFB_HRES_VIRTUAL 320    /* horizon pixel  x resolition */
```

```
#define S3CFB_VRES_VIRTUAL (240*2)    /* line cnt      y resolution */

#define S3CFB_HRES_OSD    320   /* horizon pixel  x resolition */
#define S3CFB_VRES_OSD    240 /* line cnt      y resolution */
```

1.5.4 触摸屏驱动

将 LDD6410 系统移植到 EZ6410 开发板上后，发现触摸屏在左右方向与系统的桌面应用相反，因此必须修改内核触摸屏的驱动程序。

触摸屏坐标如图 1.7 所示。

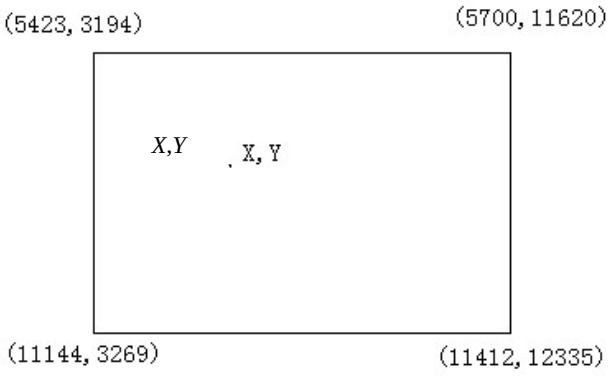

图 1.7　触摸屏坐标图

可以用下面的公式：

```
x:440=(ts->yp-3194):(11620-3194)
Y:272=(ts->xp-5423):(11144-5423)
```

来修改触摸屏驱动 drivers/input/touchscreen/s3c-ts.c，参见“触摸屏驱动”目录下的代码。

```
                ts->yp /= 16; ts->xp /= 16;
                x = (ts->yp-3194/16)*440/(11620/16-3194/16);
                y = (ts->xp-5423/16)*272/(11144/16-5423/16);
                if(x<0) x = 0; if(x>439) x = 439;
        if(y<0) y = 0; if(y>271) y = 271;
//printk("x=%d,y=%d\n",x,y);
```

1.5.5 USB 驱动修改

初始化结束后，插上 USB 就报如下错误：

```
/ # usb 1-1: new full speed USB device using s3c2410-ohci and address 2
usb 1-1: device descriptor read/64, error -62
usb 1-1: device descriptor read/64, error -62
usb 1-1: new full speed USB device using s3c2410-ohci and address 3
usb 1-1: device descriptor read/64, error -62
usb 1-1: new full speed USB device using s3c2410-ohci and address 4
usb 1-1: device not accepting address 4, error -62
usb 1-1: new full speed USB device using s3c2410-ohci and address 5
usb 1-1: device not accepting address 5, error -62
hub 1-0:1.0: unable to enumerate USB device on port 1
```

解决方法：

问题就在 ohci-s3c2410.c 中，时钟设置出了问题，原来是 USB Host 的 48MHz 时钟没有起来。

s3c2410 支持 3 个 PLL，分别是 APLL、MPLL 和 EPLL。APLL 为 ARM 提供时钟，产生 ARMCLK；MPLL 为所有和 AXI/AHB/APB 相连的模块提供时钟，产生 HCLK 和 PCLK；EPLL 为特殊的外设提供时钟，产生 SCLK。

如图 1.7 所示为 EPLL_CON 的 M、P 和 S 的取值。

FIN (MHz)	FOUT (MHz)	MDIV	PDIV	SDIV	KDIV
12	36	48	1	4	0
12	48	32	1	3	0
12	60	40	1	3	0

图 1.7　EPLL_CON 的 M、P 和 S 的取值

图 1.8 描述的是用于 IrDA 和 USB Host 的时钟发生器，通常 USB 接口需要 48MHz 的操作时钟。

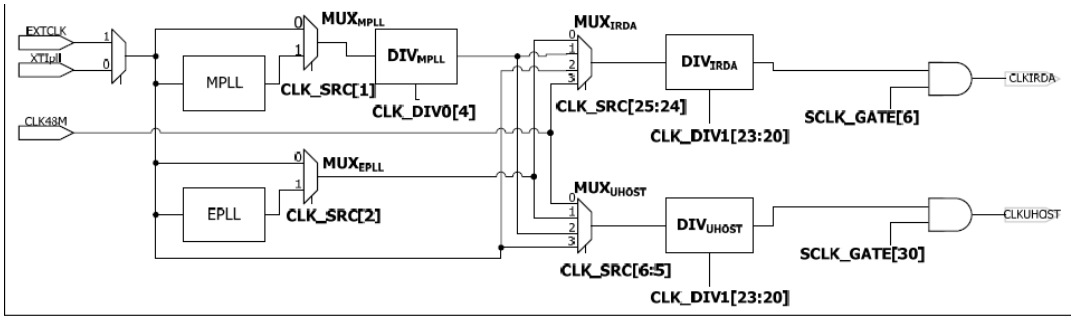

图 1.8　USB 原理图

从图 1.8 中可以看出，HCLK_GATE、PCLK_GATE 和 SCL_GATE 控制时钟操作。如果设置一个位，则通过每个时钟分频器相应的时钟将会被提供，否则，将被屏蔽。

HCLK_GATE 控制 HCLK，用于每个 Ips。每个 IP 的 AHB 接口逻辑被独立地屏蔽，以减少动态电力消耗。PCLK_GATE 控制 PCLK，通过 SCLK_GATE 时钟被控制。

根据图 1.8 中的 EPLL 通道写出以下程序：

```
#define EPLL_CON00   ((1<<31)|(0x20<<16)|(1<<8)|(3<<0))
#define EPLL_CON01        0
#define UPLL_SRC_MASK     ((1<<2)|(3<<5))
#define UPLL_SRC     ((1<<2)|(1<<5))
#define UPLL_DIV1_MASK    (0xf<<20)
#define UPLL_DIV1    (0<<20)
#define UPLL_GATE_MASK    (1<<30)
#define UPLL_GATE         (1<<30)
void set_upll(void)
{
 unsigned int tmp;

 while(__raw_readl(S3C_EPLL_CON0)!=EPLL_CON00)
 __raw_writel(EPLL_CON00,S3C_EPLL_CON0);
 while(__raw_readl(S3C_EPLL_CON1)!=EPLL_CON01)
 __raw_writel(EPLL_CON01,S3C_EPLL_CON1);
 while(((tmp= __raw_readl(S3C_CLK_SRC))&UPLL_SRC_MASK)!=UPLL_SRC)
      __raw_writel((tmp&UPLL_SRC_MASK)|UPLL_SRC,S3C_CLK_SRC);
 while(((tmp=__raw_readl(S3C_CLK_DIV1))&UPLL_DIV1_MASK)!=UPLL_DIV1)
      __raw_writel((tmp&UPLL_DIV1_MASK)|UPLL_DIV1,S3C_CLK_DIV1);
 while(((tmp=__raw_readl(S3C_SCLK_GATE))&UPLL_GATE_MASK)!=UPLL_GATE)
      __raw_writel((tmp&UPLL_GATE_MASK)|UPLL_GATE,S3C_SCLK_GATE);
}
```

在 probe 中加入上面的函数，修改 USB Host 的时钟：

```
set_upll();
```

然后编译内核。

USB 不能识别的错误就解决了。

1.5.6　安装 Android SDK

参照网页 http://slash4.de/tutorials/Android_classes_-_Day_1_-_Installing_the_SDK，完成 SDK 的安装。本例中安装的 SDK 包为 android-sdk-linux_x86-1.6_r1.tgz。

如图 1.9 所示为用命令创建虚拟机平台 EZ6410。

```
# ./android create avd -n EZ6410 -t 2
```

```
root@ffarsight:/mnt/sdc/android-sdk-linux_x86-1.6_r1/tools# ./android create avd -n EZ6410 -t 2
Android 1.6 is a basic Android platform.
Do you wish to create a custom hardware profile [no]y

Device ram size: The amount of physical RAM on the device, in megabytes.
hw.ramSize [96]:128

Touch-screen support: Whether there is a touch screen or not on the device.
hw.touchScreen [yes]:

Track-ball support: Whether there is a trackball on the device.
hw.trackBall [yes]:no

Keyboard support: Whether the device has a QWERTY keyboard.
hw.keyboard [yes]:n

DPad support: Whether the device has DPad keys
hw.dPad [yes]:y

GSM modem support: Whether there is a GSM modem in the device.
hw.gsmModem [yes]:n

Camera support: Whether the device has a camera.
hw.camera [no]:n

Maximum horizontal camera pixels
hw.camera.maxHorizontalPixels [640]:
```

图 1.9　创建虚拟机平台 EZ6410

在主机上创建一个 sd card image：

```
# sudo ./mksdcard 128M sdcard.img
```

启动 EZ6410 虚拟机，就会看到如图 1.10 所示的 Android 模拟器界面。

```
# sudo ./emulator -sdcard ./sdcard.img @EZ6410
```

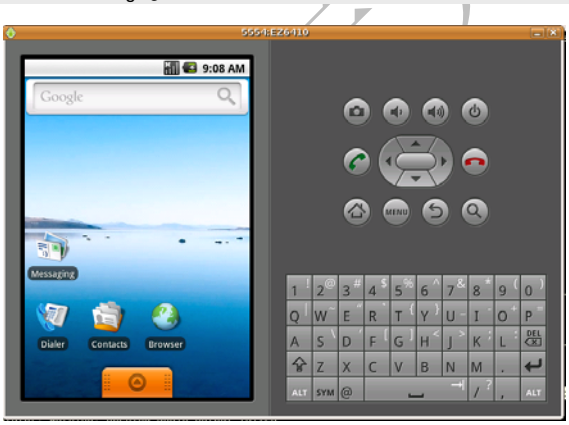

图 1.10　EZ6410 启动 Android 模拟器界面

1.5.7　提取 Android 根文件系统

上一节我们成功地在 Android 模拟器下运行了 Android 程序，但如果要把结果写进开发板，则需要得到 Android 的根文件系统，步骤如下：

连接虚拟机，如图 1.11 所示。

```
#./adb shell
```

```
root@ffarsight:/mnt/sdc/android-sdk-linux_x86-1.6_r1/tools# ./adb shell
* daemon not running. starting it now *
* daemon started successfully *
# mount
rootfs / rootfs ro 0 0
tmpfs /dev tmpfs rw,mode=755 0 0
devpts /dev/pts devpts rw,mode=600 0 0
proc /proc proc rw 0 0
sysfs /sys sysfs rw 0 0
tmpfs /sqlite_stmt_journals tmpfs rw,size=4096k 0 0
/dev/block/mtdblock0 /system yaffs2 ro 0 0
/dev/block/mtdblock1 /data yaffs2 rw,nosuid,nodev 0 0
/dev/block//vold/179:0 /sdcard vfat rw,dirsync,nosuid,nodev,noexec,uid=1000,gid=1015,fmask=0702,dmask=07
02,allow_utime=0020,codepage=cp437,iocharset=iso8859-1,shortname=mixed,utf8 0 0
#
```

图 1.11　连接虚拟机

将 busybox 放入模拟器目标机文件系统中，如图 1.12 所示。

```
root@ffarsight:/mnt/sdc/android-sdk-linux_x86-1.6_r1/tools# ./adb push /mnt/sdc/
busybox-1.16.1/busybox /data
466 KB/s (1875744 bytes in 3.925s)
```

图 1.12　将 busybox 放入模拟器

下一步把/system、/data、/sbin 目录及根目录下的 init、init.rc 等都放入 sdcard 的 image 中，如图 1.13 所示。

```
# ./busybox tar cvf /sdcard/android.tar /data /system /sbin /sqlite_stmt_journals /init.rc /init.goldfis
h.rc /init
```

图 1.13　使用 busybox

结果如图 1.14 所示。

```
# cd /sdcard
# ls
LOST.DIR
android.tar
#
```

图 1.14　结果

在主机上以 loop 方式 mount sdcard 的 image，并将其中的文件放到 EZ6410 的根文件系统下，代码如下：

```
root@ffarsight:/mnt/sdc/android-sdk-linux_x86-1.6_r1/tools# modprobe loop
root@ffarsight:/mnt/sdc/android-sdk-linux_x86-1.6_r1/tools# mount -o loop sdcard.img /mnt/sd
root@ffarsight:/mnt/sdc/android-sdk-linux_x86-1.6_r1/tools# cd /mnt/sd
root@ffarsight:/mnt/sd# ls
android.tar  lost.dir
```

在原有的 Linux 的 NFS 文件系统目录下创建一个新的目录 rootfs_test，并把 android.tar 解压到 rootfs_test 目录下。

```
# tar xvf android.tar -C /source/rootfs_android/rootfs_test/
```

在 NFS 服务目录/source/rootfs_android 下添加一个文件 android.sh，如图 1.15 所示。

```
1 #!/bin/bash
2 echo "starting android ..."
3 umount /sys
4 #umount /dev/pts
5 umount /proc
6 umask 000
7 chroot /rootfs_test /system/bin/sh
8 #chroot /android_rootfs /system/bin/sh
9 #chroot /fs_donut_google_generic /system/bin/sh
"android.sh" 9 行 --11%--
```

图 1.15　添加 android.sh

1.5.8　系统环境设置

复制 image 目录下的 zImage 到/tftpboot 目录下。

解压 android rootfs 目录下的 rootfs_android-farsight.tar.gz 到/source/rootfs_android 目录下。

设置好 NFS 环境，添加/source/rootfs_android 目录到 NFS 服务目录中。

设置 uboot 参数如下：

```
SMDK6410 # print
bootdelay=3
baudrate=115200
ethaddr=00:40:5c:26:0a:5b
```

```
bootargs=root=nfs nfsroot=192.168.1.10:/source/rootfs_android ip=192.168.1.20
 console=ttySAC0,115200
filesize=1febe0
fileaddr=C0008000
gatewayip=192.168.1.1
netmask=255.255.255.0
ipaddr=192.168.1.20
serverip=192.168.1.10
bootcmd=tftp 0xc0008000 zImage ; bootm 0xc0008000
stdin=serial
stdout=serial
stderr=serial

Environment size: 366/16380 bytes
SMDK6410 #
```

启动系统后，按下面的步骤启动 Android。

串口终端：

```
[root@192 /]# ls
android.sh   hh          linuxrc     root        tmp
bin         home        mnt         rootfs_test  usr
dev         key_drv.ko   opt         sbin        var
etc         lib         proc        sys
[root@192 /]# ./android.sh
starting android ...
# ./init
init: cannot open '/initlogo.rle'
sh: can't access tty; job control turned off
# init: cannot find '/system/bin/playmp3', disabling 'bootsound'
init: cannot find '/system/bin/dbus-daemon', disabling 'dbus'
init: cannot find '/system/etc/install-recovery.sh', disabling 'flash_recovery'
warning: `rild' uses 32-bit capabilities (legacy support in use)
request_suspend_state: wakeup (3->0) at 169456653237 (2030-09-10 04:05:05.064593851 UTC)
```

此时液晶屏上显示出和虚拟机一样的界面，按键操作。

如果想上网，可以按下面的步骤配置 Android 网络上网。

在 Android 文件系统中配置网络：

```
cd  system/etc/
vim init.goldfish.sh
```

将网络配置：

```
ifconfig eth0 10.0.2.15 netmask 255.255.255.0 up
route add default gw 10.0.2.2 dev eth0
```

修改为：

```
ifconfig eth0 192.168.1.12 netmask 255.255.255.0 up
route add default gw 192.168.1.1 dev eth0
setprop net.eth0.dns1 192.168.1.1
```

1.6 小结

本章介绍了移植 Android 系统到一套 EZ6410（基于 S3C6410）硬件系统上的典型过程，步骤虽然不多，但是涉及很多东西，从下一章节开始我们将循序渐进地介绍相关知识。本章用的 S3C6410 处理器采用的是 ARM1176JZF-S 内核，ARM Core 电压为 1.1V 时，可以运行到 553MHz；在 1.2V 的情况下，可以运行到 667MHz。这样的频率对于不断更新的 Android 系统来讲有点落后了，后面的章节将介绍一个基于 Cortex-A8 内核的开发平台，这个平台用三星公司的基于 Cortex-A8 内核的 S5PC100 处理器设计而成，很适合移植和运行 Android 系统。

1.7 思考题

1. Android 系统的移植主要分几个步骤？
2. 怎么提取 Android 的根文件系统？
3. Android 系统的移植和嵌入式 Linux 的移植有什么异同？
4. 怎么配置 Android 的交叉工具链？

联系方式

集团官网：www.hqyj.com 嵌入式学院：www.embedu.org 移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn 物联网学院：www.topsight.cn 研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路 250 号银海大厦 11 层 B 区，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

广州地址：广州市天河区中山大道 268 号天河广场 3 层，电话：020-28916067