# 阿虚的 S3C2410+Linux 学习笔记 ver1.0

## 序言&致谢

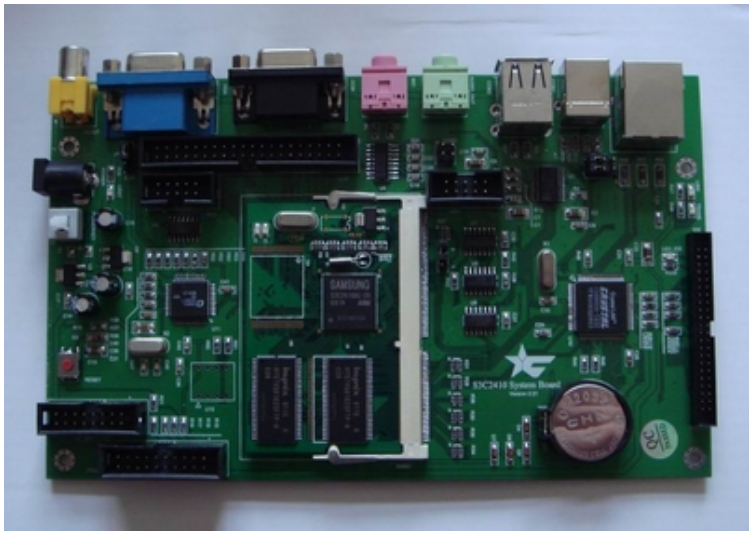
开学在即，为了记录暑期学习 2410+Linux 的过程，我写下这篇文章。文章中可能有 80%的内容不是我原创的，但都是经过了我的验证，确实有用才添加进来，因此首先要感谢致力于嵌入式 Linux，能够无私奉献，写下自己开发例程的各位大侠们！在学习过程中，我遇到了无数的问题，网络就是我的老师，绝大多数问题都是通过网络找到了答案，第二要感谢互联网！写下此文档，希望让更多的人解决开发中遇到的问题，也作为我的随身笔记，随时能够查找。最后，要推荐大家先看一下阿南的《嵌入式 Linux 入门文档》，还有《Linux 系统移植（linux_mig_release）》等文档，值得仔细研究！

## 第一章：开发环境

### 1 硬件环境

主机：台式机
CPU：双核 3G
内存：2G
硬盘：500G
显示器：24 寸 LCD+17 寸 CRT
双网卡：上网+开发

开发板：阳初 2410
CPU：S3C2410
SDRAM：64MB
NAND flash: 64MB
网络芯片：CS8900A
VGA 芯片：CH7004
声卡芯片：UDA1341
JTAG:简易 JTAG 小板



## 2 软件环境

操作系统：Windows XP SP3
虚拟机软件：VMware6.0.4（CPU 分配：单核；内寸分配：768MB；硬盘分配：25GB）
虚拟操作系统：Ubuntu8.04（DVD 完全安装）

交叉编译器：3.4.1
ADS 编程软件：ADS1.2
下载工具：TFTP　　SJF2410.EXE（初次烧录 u-boot 使用）
串口终端：SecureCRT5.1（推荐）　超级终端

内核版本：linux-2.6.14.1
U-boot 版本：u-boot-1.2.0
Busybox 版本：busybox-1.9.2.tar.bz2

# 第二章：虚拟机 VMware6.0+Ubuntu8.04 使用事项

看本文前需具备安装虚拟机和使用 Linux 系统的相关经验。

## 1 Vmware6.0 下 Ubuntu8.04 的 vmware-tools 终极解决办法

解决方法：
1） 下载 open-vm-tools:
http://jaist.dl.sourceforge.net/sourceforge/open-vm-tools/open-vm-tools-2008.05.02-90473.tar.gz
这个是已经编译的版本，只有编译过的版本才能用于接下来的操作；
2） 解压该文件，进入 modules/linux 目录，将 vmxnet 重命名为 vmxnet-only，其他也改为如对应的 xxx-only。分别执行 tar -cf xxx.tar xxx-only；

3） 将 xxx.tar 都复制到 /usr/lib/vmware-tools/modules/source，替换原来的文件；
4） sudo vmware-config-tools.pl 成功编译通过。
来自：http://blog.chinaunix.net/u2/68498/showart_723489.html
成功安装后可以同 indows 系统进行复制粘贴，非常方便。

## 2 解决安装 vmware tools 后 ubuntu 鼠标滚轮无法使用问题

用 root 用户登录编辑 /ect/x11/xorg.conf 文件，把里面的鼠标属性 “p/2” 改成“Imp/2”，重启动即可。

## 3 让 Ubuntu 8.04 firefox 的繁体中文菜单变成简体中文

1） 登录：http://releases.mozilla.org/pub/mozilla.org/firefox/releases/3.0/linux-i686/xpi/ ；
2） 点击 zh-CN.xpi ；
3） 出现“为了保护你的电脑，Firefox 拒绝网站[releases.mozilla.org]安装软件。”点击“允许”，即可安装附加元件“ChineseSimplified(zh-CN)Lannguage Pack3.0b5”，提示“重新启动以完成安装程序”，点击“重新启动 Firefox”，下次打开 Firefox 即是简体中文界面。

## 4 Ubuntu 8.04 的 root 用户以中文桌面登录

修改 root 下隐藏文件.profile 文件最后两句，以下是修改过的：

```
# ~/.profile: executed by Bourne-compatible login shells.
if [ "$BASH" ]; then
if [ -f ~/.bashrc ]; then
. ~/.bashrc
fi
fi
mesg n
# Installed by Debian Installer:
# no localization for root because zh_CN.UTF-8
# cannot be properly displayed at the Linux console
LANG="zh_CN.UTF-8"
LANGUAGE="zh_CN:zh"
```

来自：http://www.linuxdiyf.com

## 5 无法运行 make menuconfig，提示没有 ncurses

解决方法：使用新立得安装工具安装 ncurese 相关软件包。Ubuntu 若缺失其他编译工具，也可以通过新立得安装。

# 6 利用虚拟机挂载 NFS 分区

问：为何要挂载 NFS 分区？

答：可以将 PC 机的一个目录虚拟，通过网络共享给 2410 开发板载 linux 使用，省去了将程序烧入 flash 的烦恼。

1） 安装 nfs-kernel－server（新立得或 apt-get install）；

2） 在根目录建立 nfs 目录，也可以建立在其他地方；

3） Vmware 网卡 0 采用 NAT 模式，共享 PC 机上网网卡。

Vmware 网卡 1 为 bridge 模式，和开发板进行通讯并负责挂载 NFS 分区。

确保 eth1 正在连接，ifconfig eth1 192.168.0.2 。

Windows 下设定连接开发板的网卡地址为 192.168.0.1 。

2410 开发板的 ip 设为 192.168.0.12 。

在 2410 下 ping 192.168.0.1 和 192.168.0.2 ，若都能 ping 通，则继续；

4） 修改 /etc/exports 文件

添加以下内容：

/nfs 192.168.0.12(rw,sync,no_root_squash)

5） 在终端中执行以下命令：

＃exportfs -rav

＃'/etc/init.d/nfs-kernel-server' restart

6）若返回如下：

* Stopping NFS kernel daemon [ OK ]

* Unexporting directories for NFS kernel daemon... [ OK ]

* Exporting directories for NFS kernel daemon...

exportfs: /etc/exports [2]:

Neither 'subtree_check' or 'no_subtree_check' specified for export "192.168.1.241:/nfs".

Assuming default behaviour ('no_subtree_check').

NOTE: this default has changed since nfs-utils version 1.0.x [ OK ]

* Starting NFS kernel daemon [ OK ]

则 NFS 服务启动完毕；

7） 操作开发板中的 linux，输入：mount -t nfs -o nolock 192.168.0.2:/nfs /tmp

意思是把 nfs 目录虚拟到 2410 的 tmp 目录中，如果没有错误提示，恭喜你成功了。

8） 结束挂载命令：umount /tmp

# 7 安装交叉编译器 3.4.1

1） 解压 crosstools_3.4.1 到目录/usr/local/arm/3.4.1 ；

2） gedit /root/.bashrc ；

3） 在最后添加：export PATH=$PATH:/usr/local/arm/3.4.1/bin ；

4） 重启系统；

# 第三章：无操作系统篇

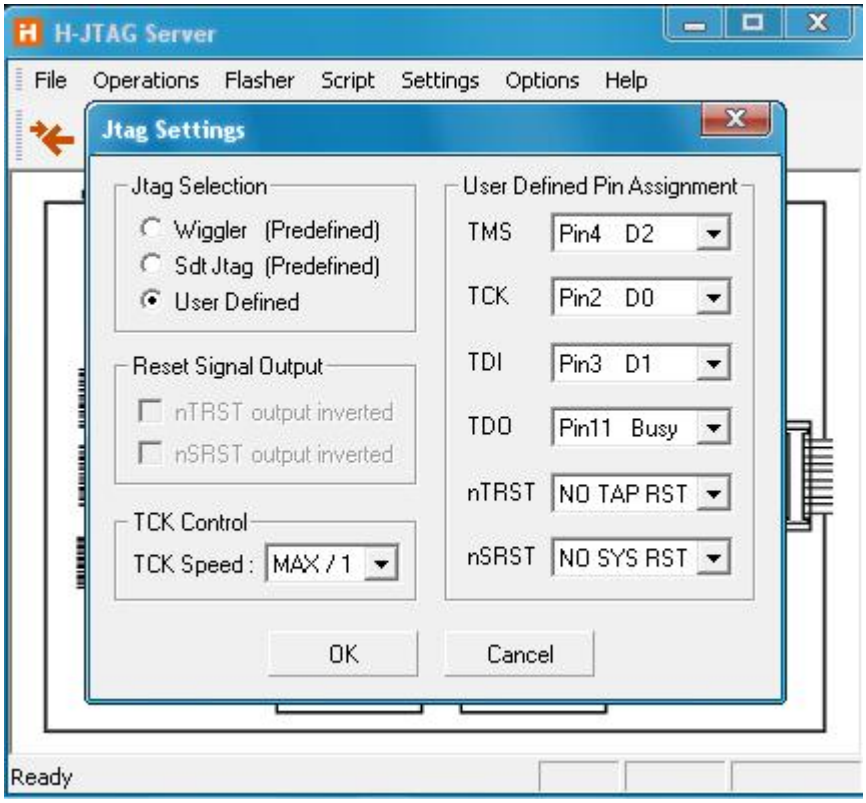
在开发板上跑 linux 前，最好先用 ADS 跑一些基本的程序以便熟悉一下 S3C2410 的寄存器，我跑了一个串口和一个 VGA 的小程序，程序见附件。

## 1 裸跑程序的 3 种运行方法：

（A）利用 ADS 的调试功能；
（B）用 u-boot 加载到 SDRAM 中，用 go 命令加载程序；
（C）用 SJF 软件烧到 flash 中跑（不推荐）。

## 2 使用 H-Jtag 调试 2410

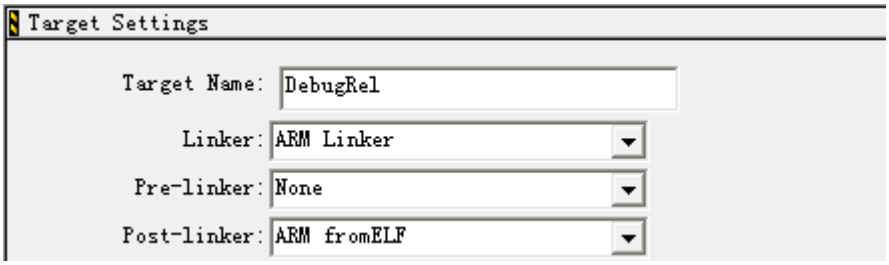
阳初 2410 开发板附带的 JSF 小板，可以使用 H-JTAG 调试。首先去 H-Jtag 官网下载软件，最新的就可以，官网很专业，地址：http://www.hjtag.com/chinese/download.html 。安装后按照下图配置：

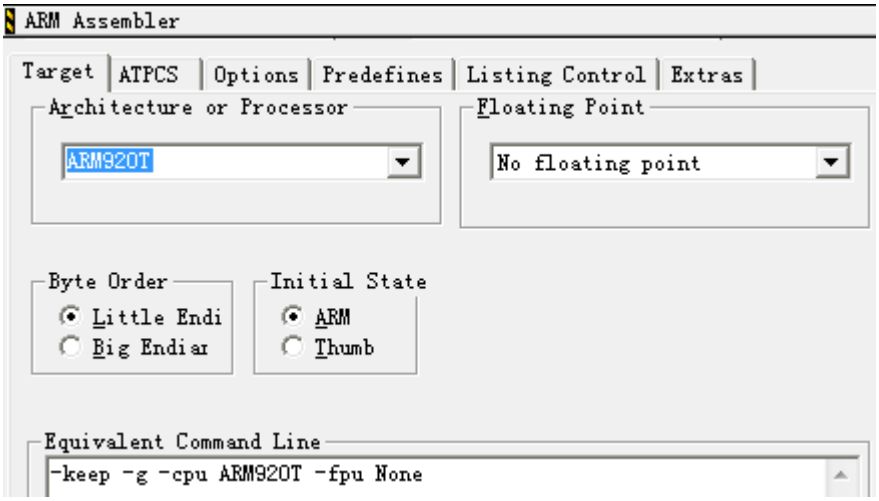


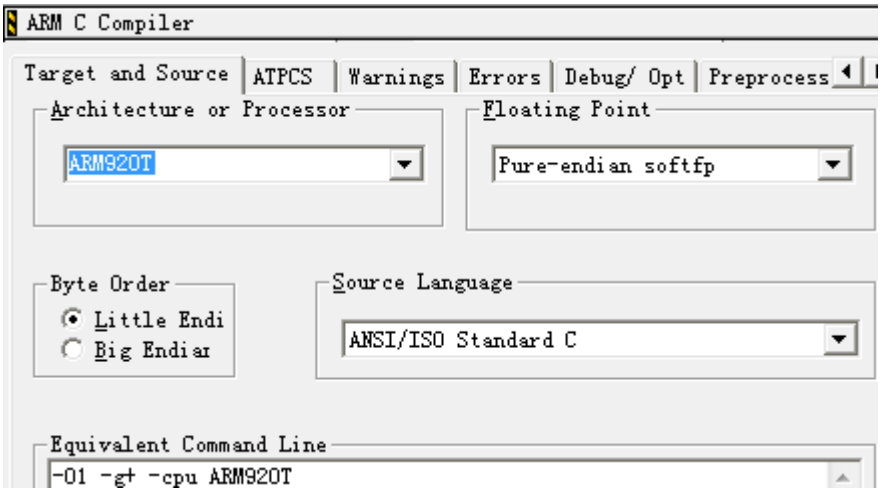

配置好连接下应该可以认出 ARM920T 了，接下来请在官网下载使用手册，按照手册经行 AXD 的配置即可。

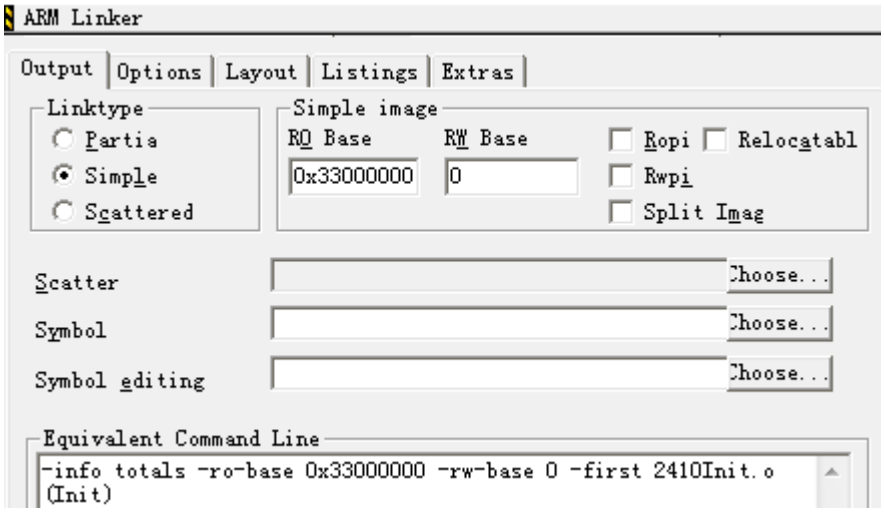
# 3 ADS 的设置方法

我摸索了一段时间，主要是 RO 地址设定为 0x33000000 RW 地址空白，其余见图。编译通过后生成的 bin 文件，在超级终端（SCRT 无法发送 kermit 协议）中使用 loadb 33000000 加载，再 go 33000000 即可运行。其实大部分设定的方法和 ARM7 差不多。






## 4 安装 GIVEIO 驱动的方法

SJF flash 烧写软件需要安装 GIVEIO 驱动才能用。首先将 GIVEIO 目录下的 GIVEIO.SYS 文件复制到 C:\WINDOWS\system32\drivers 目录下面，然后打开控制面板里的添加硬件，选择“是，我已经连接了此硬件”，下一步，选择添加新的硬件设备，下一步，选择“安装我手动从列表选择的硬件”，下一步，选择“端口（COM 和 LPT）”，下一步，选择从磁盘安装，在路径中选择 GIVEIO 目录下的 giveio.inf 文件,然后一直下去就 OK 了。在设备管理器的端口里能看到 GIVEIO 端口，就表示安装正常了。

来自：http://old.techor.com/bbs/context.asp?id=105&listMethod=all

# 第四章 Linux 系统篇

## 1 移植 u-boot-1.2.0 到 S3C2410

自从学习 ARM7 S3C44B0 时，使用的 bootloader 就是 u-boot，当时常用 loadb 和 go 指令呢。刚接触 2410，内置的 bootloader 是 vivi，不习惯，一直想换成 u-boot。不过对于初学者，编译 u-boot 实在有点难度，幸好，阳初论坛有位高手已经写下了完整的移植过程，大家照着一步一步做就可以了（附件：u-boot for yangchu 总结.rar）。提供我正在使用的 u-boot（附件：u-boot.rar），第一次需要使用 SJF 将 u-boot 烧入 nand flash，以后可以用 u-boot 的 tftp 功能，载入内存，实现自身烧写。第一步有点难度，同时需要熟悉 u-boot 的环境变量，文章中有，很重要。

在编译过程中出现的问题：

Ubuntu 下面编译 u-boot1.2 出现许多 tools/img2srec 错误，img2srec 应该是主机上运行的工具，所以判断主机上的编译器有问题，解决方法：使用 apt-get install build-essential 安装编译系统。

# 2 编译 Linux-2.6.14.1 内核到 S3C2410

初次编译内核，我完全参考《Linux 系统移植（linux_mig_release）》（附件：linux_mig_release.rar）的中编译内核的内容，说一些我遇到的问题：

1) 需要在 makefile 中修改：

ARCH ?= arm

CROSS_COMPILE ?= arm-linux-

2) 编译完成后，直接用 go 调用 zImage 出现 bad machine ID 错误：

arch_number 为 0xC1，编辑内核中的 arch/arm/boot/compressed/head.S 文件，修改

1: mov r7, r1 @save architecture ID

改为(arch/arm/tools/mach-types，您所使用的 machine type)

1: mov r7, #0xC1 @save architecture ID

再重新编译内核即可。

3） mkimage 把 zImage 转换为 uImage 的命令：

uImage 是 u-boot 加了头信息的包，可以用 bootm 命令加载，zImage 只能用 go 命令加载。mkimage 是 u-boot 提供的转换工具，需要编译过一遍 u-boot 才有。下面命令使用了完整路径，将 mkimage 放入/usr/bin 中就不需要完整路径了。

'/arm9/u-boot-1.2.0/tools/mkimage' -A arm -O linux -T kernel -C none -a 30008000 -e 30008000 -n linux-2.6.14.1 -d /arm9/u-boot-1.2.0/tools/zImage /arm9/u-boot-1.2.0/tools/uImage

4） uImage 和 zImage 都可以启动，但是都启动到下面信息就停止了：

Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(31,2)

解答：完全正常，缺少文件系统而已。

5） 固化 linux 内核到 flash 中，开机 u-boot 实现自引导：

0x10000 是 kernel 区首地址。

>tftp 0x30008000 uImage

>nand erase 0x10000 0x1d0000

>nand write 0x30008000 0x10000 0x1d0000

>setenv bootcmd nand read 0x31000000 0x30000 0x1d0000\;bootm 0x31000000

>saveenv

# 3 移植 CS8900 网卡驱动到 linux2.6

把 cs8900.c,cs8900.h 拷贝到 drivers/net 目录下，修改 Drivers/net/Kconfig，增加以下内容：

```
config ARM_CS8900
   tristate "CS8900 support"
depends on NET_ETHERNET && ARM && ARCH_SMDK2410
```

修改 Drivers/net/Makeflie，增加以下内容：

```
obj-$(CONFIG_ARM_CS8900)       += cs8900.o
```

建立 smdk2410.h 到 include/asm-arm/arch-s3c2410 目录下，smdk2410.h 的内容为 ：

```
#ifndef _INCLUDE_SMDK2410_H_
#define _INCLUDE_SMDK2410_H_
```

#include <linux/config.h>
#define pSMDK2410_ETH_IO 0x19000000
#define vSMDK2410_ETH_IO 0xE0000000
#define SMDK2410_ETH_IRQ IRQ_EINT9
#endif // _INCLUDE_SMDK2410_H_

修改 arch/arm/machs3c2410/machsmdk2410.c，在 static struct map_desc smdk2410_iodesc[] __initdata 内增加以下部分（本来为空）：

{vSMDK2410_ETH_IO, pSMDK2410_ETH_IO, SZ_1M, MT_DEVICE},

以及增加包含头文件 asm/arch/smdk2410.h //就是建立的那个，在内核配置中选上：

Device Drivers..>
Network device support…>
Ethernet (10 or 100 Mbit)
［*］ CS8900 support.

重新编译内核，启动中可以看到：

Cirrus Logic CS8900A driver for Linux (Modified for SMDK2410)
eth0: CS8900A rev E at 0xe0000300 irq=53, no eeprom , addr: 08: 0:3E:26:0A:5B

利用 ifconfig eth0 192.168.0.53 给 arm 分配地址后，可以成功 ping 通 PC：

```
# ifconfig eth0 192.168.0.53
# ifconfig
eth0        Link encap:Ethernet HWaddr 08:00:3E:26:0A:5B
            inet addr:192.168.0.53 Bcast:192.168.0.255 Mask:255.255.255.0
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
            Interrupt:53 Base address:0x300
# ping 192.168.0.54
PING 192.168.0.54 (192.168.0.54): 56 data bytes
64 bytes from 192.168.0.54: icmp_seq=0 ttl=128 time=5.8 ms
64 bytes from 192.168.0.54: icmp_seq=1 ttl=128 time=0.7 ms
--- 192.168.0.54 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.7/3.2/5.8 ms
#
```

至此 CS8900 工作正常。

来自：http://blog.hjenglish.com/bedford/articles/811128.html

# 4 移植 busybox-1.9.2，定制根目录文件

1） 建一个目录 rootfs 用来装文件系统；
2） 建立如下目录：

bin dev etc home lib mnt proc root sbin tmp usr var usr/bin usr/sbin

3） 应用程序定制：

标准的 Linux 发行版本具有功能种类比较多的应用程序，这些应用程序占用的空间也很大，这对存储容量空间有限的开发板来说就不是理想的选择，在嵌入式开发过程中，经常用 BusyBox 来定制应用程序。BusyBox 具有 shell 的功能，它能提供系统所需要的大部分工具，包括编辑工具、网络工具、模块加载工具、压缩解压缩工具、查找工具、帐号密码管理工具和进程相关工具等。

目前 BusyBox 的最新版本是 BusyBox-1.9.2 版本，下载解压，切换到 BusyBox 的根目录下，修改 Makefile,找到 ARCH 和 CROSS COMPILE 修改如下：

ARCH ?= arm

CROSS_COMPILE ?=/usr/local/arm/3.4.1/bin/arm-linux-

4） 修改编译配置选项：

#make defconfig

#make menuconfig

在默认的选项前提之下，选项设置如下:

BusyBox Settings --->

Build Options ---> (采用静态编译)

[*] Build BusyBox as a static binary (no shared libs)

Install optin-->

[*] Don't use /usr　//可以不选，选了则没有/usr 文件夹

Busybox Library Tuning --->

(2) MD5: Trade Bytes for Speed

[*] Faster /proc scanning code (+100 bytes)

[*] Support for /etc/networks

[*] Support for /etc/networks

[*] Additional editing keys

[*] vi-style line editing commands

[*] History saving

[*] Tab completion

[*] Username completion

[*] Fancy shell prompts

Linux Module Utilities --->

[ ] Support version 2.2.x to 2.4.x Linux kernels　//不能选

[*] Support version 2.6.x Linux kernels

Linux System Utilities --->

[*] Support for the old /etc/mtab file　　//不确定

Miscellaneous Utilities --->

[*] devfs(obsolete) //不确定

[*] Use devfs names for all device(obsolete) //不确定

Shell --->

Choose your default shell(ash)-->

---ash　　//下面的选项全部选择

5) 编译 busybox

#make install

在 busybox/_install 目录下会生成我们需要的文件。

修改_install/bin/busybox 的属性。为 4755

#chmod 4755 ./_install/bin/busybox
必须要修改属性，否则在 busybox 中很多命令会受限,
将编译好的 busybox 拷贝到 rootfs/bin 下面。拷贝时带上参数-arf 或者-dpR。
除了 busybox 外，所有其他的命令都是他的 link
/sbin 下面也是 busybox 的 link,
/usr/bin 下面也是 busybox 的 link,
/usr/sbin 下面放着所有编译完的可执行文件，具体就不多说了

6) 非常重要之/lib，务必重视

/lib 的库其实就是进行 busybox 编译的库，即交叉编译器的库，
我们这里使用的是 3.4.1(位置 /usr/local/arm/3.4.1/arm-linux/lib)。
# cd /usr/local/arm/3.4.1/arm-linux/lib
# for file in libc libcrypt libdl libm libpthread libresolv libutil
> do
> cp $file-*.so rootfs/lib (复制到你做的文件系统的/lib 目录下)
> cp -d $file.so.[*0-9] rootfs/lib
> done
# cp -d ld*.so* rootfs/lib

7） 系统配置文件的建立

系统配置文件放在/etc 目录下：

(1) profile 文件

#Set search library path
export LD_LIBRARY_PATH=/lib:/usr/lib:$LD_LIBRARY_PATH
#Set user path
PATH=/bin:/sbin:/usr/bin:/usr/sbin:$PATH
alias ll='ls -l'
#Set PS1
USER="`id –un`"
LOGNAME=$USER
PS1='[\u@\h \W]\$ '
PATH=$PATH
export USER LOGNAME PS1 PATH

(2) fstab 文件

proc /proc proc defaults 0 0
none /tmp ramfs defaults 0 0
mdev /dev ramfs defaults 0 0
sysfs /sys sysfs defaults 0 0

(3) inittab 文件

::sysinit:/etc/init.d/rcS
::respawn:-/bin/sh
::ctrlaltdel:/bin/umount -a -r
::shutdown:/bin/umount -a -r
::shutdown:/sbin/swapoff -a

(4) 创建/etc/init.d 文件夹和 rcS，在 rcS 中添加

#! /bin/sh

# 设置主机名，需要在 etc 建立文件 host
./etc/host
hostname ${HOSTNAME}
# mount all filesystem defined in “fstab”
echo "# mount all....."
/bin/mount -a
echo "# Starting mdev....."
/bin/echo /sbin/mdev > /proc/sys/kernel/hotplug
mdev –s

(5) host 文件

HOSTNAME=Hasotech

(6) 创建 mdev.conf 文件 （空文件）

(7) 复制主机/etc/下面的文件 passwd, group, shadow 文件到/etc

# cp /etc/group .
# cp /etc/passwd .
# cp /etc/shadow .
修改 passwd 中用户使用的 shell 名称。FC6 上默认的为 bash,而 vivi 只支持 ash。
root:x:0:0:root:/root:/bin/bash -->
root:x:0:0:root:/root:/bin/ash

(8) 修改各文件和文件夹的权限

# chmod 755 /rootfs/etc/init.d/rcS
# chmod 755 /rootfs/etc/host
来自：http://blog.csdn.net/mimimomo/archive/2008/05/14/2443593.aspx
同时请参考文档：linux_mig_release

# 5 建立 cramfs 文件系统

请参考文档：linux_mig_release ，内容较详细。

问题：make cramfs 缺少 zlib
解决：用新立得安装 zlib 相关。
烧写 cramfs 文件的方法：
>tftp 31000000 rootfs.cramfs
>nand erase 0x400000 0xXXXXXXXX
（400000 为 nand root 分区起始地址，XXXXX 为下载文件大小）
>nand write 31000000 0x400000 0xXXXXXXXX

# 6 为 u-boot 添加 I2C 支持，驱动 CH7004

起因：使用 u-boot 中发现有一个问题，通过 vivi 可以正常引导并驱动 VGA 的 linux2.4 内核文件，使用 u-boot 引导后，VGA 驱动运行不正确，效果如下：


查看 vivi 的启动代码，发现阳初 2410 板是将 vga 的驱动交由 vivi 完成，linux 内核负责驱动 lcd 即可。该板使用的 vga 芯片为 CH7004，网上也有板子使用的芯片为 CH7005，我用过 CH7013，应该都差不多，通过 I2C 协议，赋值芯片里的几个寄存器实现 VGA 输出。

CH7004 需要 I2C 总线向地址 4,7,14,13 分别顺序写入 0x20,0x8c,0x1b,0x03，CH7004 芯片的地址为 0xEC（这儿有点问题，后面会提到）。

既然 vivi 可以使用 i2c，为何 u-boot 不可以，在网上搜了半天也只搜到了一篇外国 blog 写的如何操作 u-boot 里的 i2c 命令，算了，看文档自己来吧。

准备：我是直接对 u-boot-1.2.0 的样板 smdk2410 进行的修改。

操作过程：

修改 include/configs/smdk2410.h 文件

（1）去掉 CFG_CMD_I2C |\ 的注释

（2）在中间某处添加如下内容

```
/*-----------------------------------------------------------------------
* I2C
*-----------------------------------------------------------------------*/
#define CONFIG_DRIVER_S3C24X0_I2C       1
#define CONFIG_HARD_I2C             1       /* I2C with hardware support        */
//#define CONFIG_I2C_CMD_TREE       1
#undef      CONFIG_SOFT_I2C                 /* I2C bit-banged               */
#define CFG_I2C_SPEED               100000      /* I2C speed and slave address      */
#define CFG_I2C_SLAVE               0x0
```

（3）看似很简单，我试了一个晚上，讲解下：

本来以为只要 CFG_CMD_I2C 就行的，结果 make 错误，提示命令未定义 ，之后开始看 u-boot 的 README（写的很好），发现要定义 CONFIG_HARD_I2C 或者 CONFIG_SOFT_I2C，2410 是硬件支持 I2C 的，当然是定义硬件，能简单不少。

CFG_I2C_SPEED 是定义传输速度，不用多说，CFG_I2C_SLAVE 也必须定义，目前我还没搞清楚，是从机地址还是其他？我开始填的是 CH7004 的 0xEC 地址，后来填了 0x0，结果都能用。后来才知道，这个地址是 2410 作为从机的地址，所以随便填都能用。

CONFIG_I2C_CMD_TREE 如果定义这个，可以把所有指令统一归为 i2c xxxxx，但是定义了有一些错误，因为 2410 不能动态修改 i2c 的传输速度，需要在 common/cmd_i2c.c 注释掉相关项才行，还是不定义吧。

最关键的是 CONFIG_DRIVER_S3C24X0_I2C 这个，搞了我一个晚上才发现要定义这

个选项，cmd_i2c.c 里包含了 i2c.h 文件，这个文件在 include 目录里面，是一个通用文件，但我找了好久才找到 i2c.c 文件，在 cpu/arm920t/s3c24x0 目录里面！仔细看一看，需要定义这个 CONFIG_DRIVER_S3C24X0_I2C 才行！我说怎么一直 make 提示未定义的函数呢~~~~~~ToT

（4）编译过后，将 u-boot.bin 下载到 nand flash 的最前面，希望能用吧。

介绍几个主要命令

iprobe　　检测所有在总线上的 i2c 设备号（相当好用的命令）

imw　　i2c 内存赋值，使用方法 imw 从机地址 数据地址 数据

eg. imw 0x76 4 0x20

imd　　观察 i2c 内存

imm　　自动增加地址赋值

首先用 iprobe 观察是否能找到 i2c 设备，我的返回如下：Valid chip addresses: 50 76。

很奇怪，CH7004 的地址竟然是 76！刚开始我还以为 i2c 工作不正常，后来加了块 24c256 在板子上，出来一个新地址 50，i2c 工作是正常的，为什么 CH7004 的地址是 76，而不是原来的 0xEC 呢？求解。

（5）依次执行：

imw 76 4 20

imw 76 7 8c

imw 76 14 1b

imw 76 13 3

CRT 竟然毫无反应，囧，又在这困了半天，还是以为没正常工作。最后想到下载 linux 的 uImage 镜像，执行看看，VGA 显示竟然出来啦！


（6）开机启动自动初始化 VGA

可以利用 env 来搞定

setenv vga imw 76 4 20\; imw 76 7 8c\; imw 76 14 1b\;imw 76 13 3

setenv bootcmd run vga

saveenv

注意斜杠不能漏掉,下面是我的 env，我搞繁琐了，在 bootcmd 中还可以添加 linux 内核启动：

bootargs=noinitrd root=/dev/mtdblock/2 console=ttySAC0,115200

bootdelay=3

baudrate=115200
ethaddr=08:00:3e:26:0a:5b
ipaddr=192.168.0.54
serverip=192.168.0.1
netmask=255.255.255.0
vga1=imw 76 4 20
vga2=imw 76 7 8c
vga3=imw 76 14 1b
vga4=imw 76 13 3
vga=run vga1;run vga2;run vga3;run vga4
bootcmd=run vga
stdin=serial
stdout=serial
stderr=serial

# 7 移植 LCD 驱动程序到 linux-2.6

移植过程：
修改 arch/arm/mach-s3c2410/mach-smdk2410.c 文件
添加如下内容

```
/*my add*/
#include <asm/arch/regs-lcd.h>
#include <asm/arch-s3c2410/fb.h>
static struct s3c2410fb_mach_info s3c2410_lcd_info __initdata = {
.fixed_syncs         = 0,
.regs       = {
            .lcdcon1       = S3C2410_LCDCON1_TFT16BPP |
                        S3C2410_LCDCON1_TFT |
                    // S3C2410_LCDCON1_ENVID |
                        S3C2410_LCDCON1_CLKVAL(1),
            .lcdcon2       = S3C2410_LCDCON2_VBPD(25) | S3C2410_LCDCON2_VFPD(5)
| S3C2410_LCDCON2_VSPW(1),
        .lcdcon3       = S3C2410_LCDCON3_HBPD(67) | S3C2410_LCDCON3_HFPD(40),
         .lcdcon4       = S3C2410_LCDCON4_HSPW(31) | S3C2410_LCDCON4_MVAL(13),
         .lcdcon5          =  S3C2410_LCDCON5_FRM565  |  S3C2410_LCDCON5_HWSWP  |
S3C2410_LCDCON5_PWREN|S3C2410_LCDCON5_INVVLINE|S3C2410_LCDCON5_INVV
FRAME ,
      },
    .lpcsel             = 0,// ((0xCE6) & ~7) | 1<<4,
    .gpccon=            0xaaaaaaaa,
    .gpccon_mask=       0xffffffff,
    .gpcup=             0xffffffff,
    .gpcup_mask=        0xffffffff,
    .gpdcon=            0xaaaaaaaa,
```

```
    .gpdup =            0xFFFFFFFF,
    .gpdup_mask=        0xffffffff,
    .width              = 640,
    .height             = 480,
    .yres               = {
          .min        = 480,
          .max        = 480,
          .defval       = 480,
     },
    .xres               = {
         .min        = 640,
         .max        = 640,
         .defval = 640,
    },
.bpp              = {
        .min        = 16,
        .max        = 16,
        .defval = 16,
    },
};
static void __init smdk2410_lcd_init(void)
{
      s3c24xx_fb_set_platdata(&s3c2410_lcd_info);
}
/******************my add end**********************************/
//分辨率大小可自行修改
在 MACHINE_START 与 MACHINE_END 中添加如下代码：
      .init_machine     = smdk2410_lcd_init,



有的资料如上就可以，我使用的内核版本为 2.6.14.1，编译存在错误
因为 2.6.14 没有 s3c24xx_fb_set_platdata()这个函数,
在 arch/arm/mach-s3c2410/devs.c 中加入该函数定义
/*************************************************************************/

void __init s3c24xx_fb_set_platdata(struct s3c2410fb_mach_info *pd)
{
    struct s3c2410fb_mach_info *npd;
    npd = kmalloc(sizeof(*npd), GFP_KERNEL);
    if (npd) {
        memcpy(npd, pd, sizeof(*npd));
        s3c_device_lcd.dev.platform_data = npd;
    } else {
```

```
        printk(KERN_ERR "no memory for LCD platform data\n");
    }
}
EXPORT_SYMBOL(s3c24xx_fb_set_platdata);
/*****************************************************************************/
```

make menuconfig 中还要加入 S3C2410 frambuffer support 选项。最后编译执行，就能看到小企鹅了。


补充下去掉 10 分钟关闭 LCD 的方法：

注释掉 drivers\char\vt.c 的 blank_screen_t(unsigned long dummy)的函数内容,否则,lcd 会在 10 分钟左右关掉显示。

# 8 建立 Embedded QT 开发环境

本文主要内容为编译器的安装，安装与建立 Qt 桌面运行环境，Qt/E 的交叉编译，建立本机 Qtopia 虚拟平台，

编译器: arm-linux-gcc-3.4.1.tar.bz2

一. 安装与建立 Qt 桌面运行环境

软件：PC 机操作系统 FC2＋MINICOM ＋ ARM-LINUX 开发环境

tmake-1.13.tar.gz qt-embedded-2.3.10-free.tar.gz

qt-x11-2.3.2 .tar.gz

把本次实验用到的三个文件拷贝到/public/qt 目录下，以下的步骤是假设你在/public/qt 下操作的。

Qt/Embedded 平台的搭建需要以下几步：

第一步，解压安装包并设置环境变量

tar –xzvf tmake-1.13.tar.gz

tar –xzvf qt-x11-2.3.2.tar.gz

tar –xzvf qt-embedded-2.3.10-free.tar.gz

mv qt-2.3.10 qt-2.3.10-host

export TMAKEDIR=$PWD/tmake-1.13

export QT2DIR=$PWD/qt-2.3.2

export QTEDIR=$PWD/qt-2.3.10-host

环境变量的设置是非常重要的，它关系到能否正确的安装及编译这些安装包.
注意: 在以下安装中,make 命令执行前先执行一下 make clean 命令.
第二步，编译 Qt/Embedded。
1． Build Qt2.3.2
cd $QT2DIR
export TMAKEPATH=$TMAKEDIR/lib/linux-g++
export QTDIR=$QT2DIR
export PATH=$QTDIR/bin:$PATH
export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
make clean
./configure -no-xft
make
mkdir $QTEDIR/bin
cp bin/uic $QTEDIR/bin/
2. Build Qvfb
export TMAKEPATH=$TMAKEDIR/lib/linux-g++
export QTDIR=$QT2DIR
export PATH=$QTDIR/bin:$PATH
export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
cd $QTEDIR/tools/qvfb
make clean
tmake -o Makefile qvfb.pro
make
mv qvfb $QTEDIR/bin/
这一步 build qvfb 并建立了从 Qt/Embedded 2.3.10 到 Qt 2.3.2 的静态库的链接。其中 qvfb 工具用来生成 Virtual framebuffer，这是一个非常有用的工具，它可以模拟在开发板上的显示情况，如果在 Virtual framebuffer 中运行没有问题的话，可以直接通过交叉编译在开发板上运行。
3. Build Qt/Embedded
cd $QTEDIR
export TMAKEPATH=$TMAKEDIR/lib/qws/linux-x86-g++
export QTDIR=$QTEDIR
export PATH=$QTDIR/bin:$PATH
export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
make clean
./configure -no-xft -qvfb -depths 4,8,16,32 此时我选择的 5
make
第三步 查看运行结果
如果上面各步都能够成功的编译通过，下面就可以通过运行 Qt/Embedded 自带的 demo 来查看运行结果。
●在 Virtual framebuffer 上运行：
export QTDIR=$QTEDIR
export PATH=$QTEDIR/bin:$PATH
export LD_LIBRARY_PATH=$QTEDIR/lib:$QT2DIR/lib:$LD_LIBRARY_PATH

cd $QTEDIR/examples/launcher
qvfb -width 640 -height 480 &
sleep 10
./launcher -qws
出现
libpng error: Valid palette required for paletted images 这个问题。
二 Qt/E 的交叉编译
要将我们写好的程序发布到开发板上，我们需要对 Qt/Embedded 重新编译，与前面在宿主机上编译类似，步骤如下：
1． Build Qt/Embedded
tar –xzvf qt-embedded-2.3.10-free.tar.gz
mv qt-2.3.10 qt-2.3.10-target
export TMAKEDIR=$PWD/tmake-1.13
export QT2DIR=$PWD/qt-2.3.2
export QTEDIR=$PWD/qt-2.3.10-target
cd $QTEDIR
export TMAKEPATH=$TMAKEDIR/lib/qws/linux-arm-g++
export QTDIR=$QTEDIR
export PATH=$QTDIR/bin:$PATH
export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
make clean
./configure -xplatform linux-arm-g++ -no-xft -no-qvfb -depths 4,8,16,32
make
出现
/public/qt/qt-2.3.10-target/lib/libqte.so: undefined reference to `operator new[](unsigned int)'
/public/qt/qt-2.3.10-target/lib/libqte.so: undefined reference to `operator delete(void*)'
/public/qt/qt-2.3.10-target/lib/libqte.so: undefined reference to `__cxa_pure_virtual'
/public/qt/qt-2.3.10-target/lib/libqte.so: undefined reference to `operator delete[](void*)'
/public/qt/qt-2.3.10-target/lib/libqte.so: undefined reference to `operator new(unsigned int)'
collect2: ld returned 1 exit status
make[4]: *** [t1]
暂时不管它，不影响的。下面的第 2 步： “修改 tmake 配置文件” 即可解决此类问题。这步完成后，我们会在/$QTEDIR/lib/目录下面看到 libqte.so libqte.so.2 libqte.so.2.3 libqte.so.2.3.10 这四个文件，我们可以使用 file 命令来查看这个库文件是否是我们需要的在开发板上跑的库。
file libqte.so.2.3.10
libqte.so.2.3.10: ELF 32-bit LSB shared object, ARM, version 1 (ARM), stripped
有了这个库以后我们就可以把它拷贝到我们的开发板中相应的库目录下面，这里我们选择了开发板上的/usr/lib 目录，将/$QTEDIR/lib/下的 libqte.so*复制到/usr/lib 目录下。
首先要建立宿主机和开发板的通讯，假设本机的 ip 地址为 192.168.0.56 并且/root/share 为共享文件夹。
cp –arf /$QTEDIR/lib/libqte.so* /root/share
启动 minicom
mount －t nfs －o nolock 192.168.0.56:/root/share /mnt/nfs 将文件复制到开发板上

cp –arf /mnt/nfs/libqte.so* /usr/lib

2． 修改 tmake 配置文件

vi $ TMAKEDIR/lib/qws/linux-arm-g++/tmake.conf

将其中“TMAKE_LINK= arm-linux-gcc”

“TMAKE_LINK_SHLIB= arm-linux-gcc”

修改为：“TMAKE_LINK= arm-linux-g++”

“TMAKE_LINK_SHLIB= arm-linux-g++”

3.生成可执行文件

这里我们采用了 Qt/Embedded 自带的一个 demo，它在/$QTEDIR/examples/progressbar 目录下，这个目录包括下面几个文件： main.cpp、Makefile.in 、progressbar.h 、Makefile、progressbar.cpp、progressbar.pro，如果已经有了 progressbar 的执行文件，可以使用 make clean 删除。

progen –t app.t –o progressbar.pro

echo $TMAKEPATH

查看返回的结果的结尾字符是否是“……/qws/linux-arm-g++”，如果不是的话需要在命令行中重新设置 TMAKEPATH

export TMAKEPATH=/tmake 的安装路径（如$TMAKEDIR）/lib/qws/linux-arm-g++

此外还要使 QTDIR 指向 Qt/Embedded 的安装路径，如：

export QTDIR=$QTEDIR 或者直接指定路径

export QTDIR=……/qt-2.3.10-target

完成了上面的环境变量的设置，并用 echo 命令检查无误以后，就可以使用 tmake 工具来生成我们需要的 makefile 文件，在命令行中输入如下命令：

tmake –o makefile progressbar.pro

make

如果没出现错误的话就可以在当前目录下找到 progressbar 这个可执行文件，它就是在我们开发板上的相应目录中运行“./progressbar -qws”就可以运行程序了。

注: 1).如果执行命令

[/mnt/nfs]./progressbar -qws

./progressbar: error while loading shared libraries: libstdc++.so.6: cannot open shared object file: No such file or direy

我的解决办法是从编译器目录下查找,并拷贝到开发板/usr/lib/中

即拷贝/usr/local/arm/3.4.1/arm-linux/lib/libstdc++.so* 到开发板/usr/lib/中

2).[/mnt/nfs]./progressbar –qws

./progressbar: error while loading shared libraries: libgcc_s.so.1: cannot open shared object file: No such fy

方法同上

在板子上：

[root@hjembed qt_bin]# ./progress -qws

Cannot find font definition file /usr/local/qt-embedded/lib/fonts/fontdir - is ?

[root@hjembed qt_bin]# ls

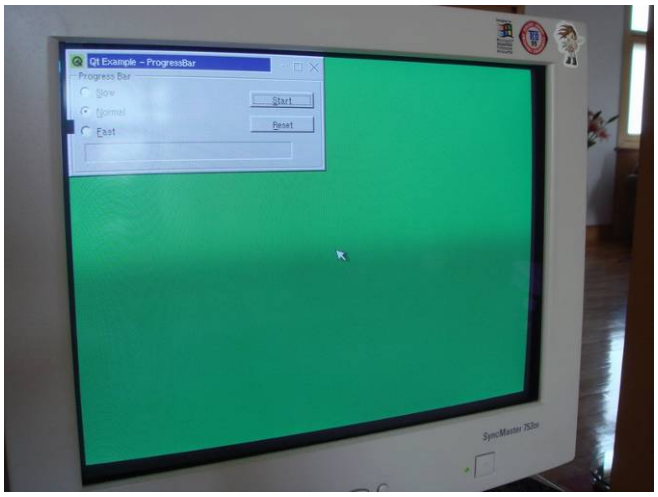
hello progress

[root@hjembed qt_bin]# ./hello -qws

Cannot find font definition file /usr/local/qt-embedded/lib/fonts/fontdir - is ?

出现此问题， 解决办法：
在上面拷贝 libqte*.so* 时顺带把 fonts 目录也一并考入。并设置环境变量。
[root@hjembed qt_bin]# export QTDIR=/usr/
[root@hjembed qt_bin]# export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
[root@hjembed qt_bin]# export PATH=$QTDIR/qt_bin:$QTDIR/bin:$PATH
[root@hjembed qt_bin]# echo $LD_LIBRARY_PATH
/usr//lib:/lib:/usr/lib

在屏幕上看到了界面， 成功了。




我做到这，的确能够运行，QTOPIA 界面还没有做。
文章来自：记不得原来的地址了，应该是叫这个名字，后面还有搭建 QTOPIA 平 台 的内容，我没有做下去。
编译时遇到的问题：

1） 开发板上执行./progressbar -qws 提示找不到文件
需要拷贝编译器（我用的 3.4.1）中 arm-linux/lib 中的某些库文件到文件系统的 lib；
2） 提示无法在 tmp 目录下面建立 qtembedded-0 目录，tmp 目录不可读
解决方法：mount /tmp 为 ramfs 文件系统；
（在文件系统/etc/fstab 文件中要有 none /tmp ramfs defaults 0 0 这条）
3） 提示:can’t open framebuffer device /dev/fb0
解决方法：执行 ln –sf /dev/fb/0 /dev/fb0；

# 9 移植 uda1341 声音驱动到 linux-2.6

1） linux2.4 内核里面的 s3c2410-uda1341.c 文件肯定是不能用的，需要大量修改，我是从华恒的论坛下载到修改好的文件。
链接：http://www.hhcn.com/cgi-bin/topic.cgi?forum=3&topic=613&start=0&show=0
或看附件：s3c2410-uda1341.rar
2） 仔细看这个 s3c2410-uda1341.c 文件，还是有点问题。首先，我没有这个文件 asm/arch/S3C2410.h ，没办法，先注释掉，硬着头皮编译；
3） 将 s3c2410-uda1341.放到 linux2.6.14/sound/oss/目录下；
4） 修改 oss 目录下的 kconfig：
添加

config S3C2410_SND_UDA1341
tristate "S3C2410 UDA1341 driver (S3C2410)"
depends on SOUND_PRIME!=n && SOUND && ARM && ARCH_SMDK2410

5） 修改 oss 目录下的 makefile
增加
obj-$(CONFIG_S3C2410_SND_UDA1341) += s3c2410-uda1341.o

6） 修改增加内存映射，以使内核知道该虚拟地址可用，而且对应的物理地址是我们的声卡。
修改/arch/arm/mach-s3c2410/smdk-s3c2410.如下：
static struct map_desc smdk2410_iodesc[] __initdata = {
/**//* nothing here yet */
{0xe0000000, 0x19000000, SZ_1M, MT_DEVICE}, // 网卡内存映射
{0xf0d00000, 0x55000000, SZ_1M, MT_DEVICE}, // 声卡内存映射
};

7） make menuconfig
选择 driver->sound->oss->uda1341

8） 编译 make zImage
没有错误的话，启动可以看到
mice: PS/2 mouse device common for all mice
UDA1341 audio driver initialized
NET: Registered protocol family 2

排错：

（1）提示“S3C2410_IISMOD_MASTER”未定义：
网上的解决方法是注释掉相关的两条，因为本来就是主设备；

（2）提示"IISPSR_A" "IISPSR_B" 未定义”：
很无奈的错误，网上基本无解，后来在论坛上看到 s3c2410-uda1341.c，添加了如下内容：
/************************* add by lfc *************************/
#define fIISPSR_A Fld(5, 5) /* Prescaler Control A */
#define IISPSR_A(x) FInsrt((x), fIISPSR_A)
#define fIISPSR_B Fld(5, 0) /* Prescaler Control B */
#define IISPSR_B(x) FInsrt((x), fIISPSR_B)
/************************** end add **************************/

（3）提示"FInsrt"等未定义：
晕，上面白添加了。
继续看不同之处，发现还需要这个文件#include <asm/arch/bitfield.h>
这个文件 2.6 内核没有，我从 2.4 内核里拷贝过来，果然有 Finsrt 的定义。

9）测试
A. 查看在/dev/sound/下是否有 dsp 以及 mixer 两个设备；
B. 如果有，再执行如下的命令：cat 1.wav >/dev/sound/dsp;就是将 PCM 数据放到 dsp 上，如果成功地进行了移植，那么耳机就会听到一些声音，当然还不是音乐。这里只能是用.wav 文件来测试，因为它是 PCM 编码的，不能用 mp3 文件，否则会听不到发声。1.wav 我是通过 tftp 传到 ramfs 分区的。

# 10 移植 madplay mp3 播放器到 linux-2.6

1） 移植 madplay 前的准备。

madplay 的移植需要以下的几个包：madplay-0.15.2b.tar.gz，libmad-0.15.1b.tar.gz，libid3tag-0.15.1b.tar.gz 和 zlib-1.1.4.tar.gz 。其中前面的三个包可以在 http://sourceforge.net/project/showfiles.php?group_id=12349 下载，后一个包可以在 http://www.gzip.org/zlib/zlib-1.1.4.tar.gz 中找到。

为了编译这个播放器，还需要一个交叉编译器，我用的是 arm-linux-gcc 3.4.1 版本，网上很多文章都说用 arm-linux-gcc 2.95.3 这个交叉编译器，但是我在用它配置(./configure)libid3tag 的时候，它最后老是说 configure 文件的某一行有错， check "config.log" for detail。没办法，只好用 3.4.1 版本的编译器。

2) 开始编译

(1) 用交叉编译工具编译 zlib，并且把库生成到交叉编译环境的库目录下，我把它放到了 /usr/local/arm/3.4.1/arm-linux 目录下。./configure --prefix=/usr/local9/arm/3.4.1/arm-linux。

再修改(不能跟上面的步骤调转）makefile 文件：

CC=/usr/local/arm/3.4.1/bin/arm-linux-gcc

AR=/usr/local/arm/3.4.1/bin/arm-linux-ar rcs

RANLIB=/usr/local9/arm/3.4.1/bin/arm-linux-ranlib

make ,make install ,编译好之后就可以在上面 prefix 指定的目录下的 lib 目录下找到 libz.a 这个库。

(2) 编译 libid3tag

./configure CC=/usr/local/arm/3.4.1/bin/arm-linux-gcc --prefix=/usr/local/arm/3.4.1/arm-linux --host=arm-linux --disable-debugging --disable-shared --enable-static CPPFLAGS=-I/usr/local/arm/3.4.1/arm-linux/include LDFLAGS=-L/usr/local/arm/3.4.1/arm-linux/lib

其中，--disable-shared --enable-static 是指定为静态编译。不过我发现这样并不能够进行编译。至于如何进行表态编译，我将在下面中进行介绍。

make, make install

(3) 编译 libmad

. /configure CC=/usr/local/arm/3.4.1/bin/arm-linux-gcc --prefix=/usr/local9/arm/3.4.1/arm-linux --host=arm-linux --disable-debugging --disable-shared --enable-static CPPFLAGS=-I/usr/local9/arm/3.4.1/arm-linux/include LDFLAGS=-L/usr/local9/arm/3.4.1/arm-linux/lib

然后，make ,make install

(4) 编译 madplay

./configure CC=/usr/local/arm/3.4.1/bin/arm-linux-gcc --prefix=/usr/local/arm/3.4.1/arm-linux --host=arm-linux --disable-debugging --disable-shared --enable-static CPPFLAGS=-I/usr/local/arm/3.4.1/arm-linux/include LDFLAGS=-L/usr/local/arm/3.4.1/arm-linux/lib

然后，make ,make install

编译完成后，查看了一下 madplay 这个可执行文件的大小，大概 237K 左右，我猜想应该是用了动态编译的原因，于是把它下载到板子上试了一下，输入：

$> ./madplay

. /madplay: /lib/libc.so.6: version `GLIBC_2.3' not found (required by ./madplay)

我就更加确定上面的--disable-shared --enable-static 参数并没有让它进行静态的编译，后来又用 file madplay 试了一下，输出：

madplay: ELF 32-bit LSB executable, ARM, version 1 (ARM), for GNU/Linux 2.4.3, dynamically linked (uses shared libs), for GNU/Linux 2.4.3, not stripped

果然是动态编译的，那么应该如何进行表静态的编译呢？我在网上找呀，都没有说具体的解决方案，只是说在 make 之后输出的最后的编译命令后加上－ static，开始不太明白，还是以为只是加上--enable-static 命令，我又仔细地体会了这句话的意思，后来就按自己的想法试了一下，如下：

/usr/local/arm/3.4.1/bin/arm-linux-gcc -Wall -O2 -fomit-frame-pointer -o madplay madplay.o getopt.o getopt1.o version.o resample.o filter.o tag.o crc.o rgain.o player.o audio.o audio_aiff.o audio_cdda.o audio_hex.o audio_null.o audio_raw.o audio_snd.o audio_wave.o audio_oss.o -L/usr/local/arm/3.4.1/arm-linux/lib /usr/local/arm/3.4.1/arm-linux/lib/libmad.a /usr/local/arm/3.4.1/arm-linux/lib/libid3tag.a -lz -lm -static

上面这一个命令 arm-linux-gcc 的最后一个参数 static 是我自己加上去的。在终端找上以上命令后，很快就编译完毕了，我又到 madplay 目录下看一下 madplay 文件的大小, 820 多 K!!!，太好了，我知道可能是成功了，就把它下载到板子中，敲入：

$> ./madplay pig.mp3

美妙的歌曲荡漾在我的耳边，真是高兴呀！！cheer!

来自：http://blog.csdn.net/bjgui/archive/2008/03/14/2183989.aspx

补充：文中提到了使用静态编译，我还是使用了动态编译器，用那个 200 多 k 的 madplay 文件。同时把 3.4.1 编译器中几个新生成的 lib 库文件拷贝到开发板的 lib 库中就可以正确运行了。

# 11 为 linux 系统添加 u 盘支持

1） 在 arch/arm/mach-s3c2410/mach-smdk2410.c 中紧接着 LCD 的部分添加

```
//usb
#include <asm/arch/usb-control.h>
#include <asm/arch/regs-clock.h>
#include <linux/device.h>
#include <linux/delay.h>
//-----------------------------------------------usb
struct s3c2410_hcd_info usb_s3c2410_info = {
                .port[0] = {
                        .flags = S3C_HCDFLG_USED
                                }
};
```

```
int smdk2410_usb_init(void) /* USB */
{
        unsigned long upllvalue = (0x78<<12)|(0x02<<4)|(0x03);
        printk("USB Control, (c) 2006 s3c2410\n");
        s3c_device_usb.dev.platform_data = &usb_s3c2410_info;
        while(upllvalue!=__raw_readl(S3C2410_UPLLCON))
        { __raw_writel(upllvalue,S3C2410_UPLLCON);
                mdelay(1);
       }
        return 0;
}
```

2） 在 smdk2410_map_io 函数最后添加

```
smdk2410_usb_init();
```

3） 让内核支持热插拔

```
  General setup --->
  
Support for hot-pluggable devices
```

4） USB 驱动设置，可能有些不选也行，不过没时间去试，至于为什么要选这些选项的话可以看一下这个贴（Linux 下的硬件驱动——USB 设备）：
http://www-128.ibm.com/developerworks/cn/linux/l-usb/index1.html

```
    Device Drivers --->
    Generic Driver Options --->
<*> Hotplug firmware loading support
    Block devices --->
    <*> Low Performance USB Block driver
    SCSI device support --->
    <*> SCSI generic support
    
Probe all LUNs on each SCSI device
    USB support --->
  <*> Support for Host-side USB
    
USB device filesystem
  <*> OHCI HCD support
  <*> USB Mass Storage support
    
USB Monitor
```

5） 加入了 MSDOS fs 和 VFAT fs 的支持。

```
    File systems --->
```

```
|  |  DOS/FAT/NT Filesystems --->
|  ┌──────────────────────────────────────────────────┐  |
|  |  <*> MSDOS fs support                            |  |
|  |  <*> VFAT (Windows-95) fs support                |  |
|  |  (936) Default codepage for FAT                  |  |
|  |  (cp936) Default iocharset for FAT               |  |
|  |  < > NTFS file system support
```

做完这些后，插入 u 盘后，内核应该可以识别到 u 盘，出现：
usb 1-1: new full speed USB device using s3c2410-ohci and address 3
ub(1.3): GetMaxLUN returned 0, using 1 LUNs

但是，还有下面一句出错提示：
/dev/ub/a: unknown partition table

5） 再次查看了贴子上大虾们的讨论，提到：“使能 CONFIG_MSDOS_PARTITION 选项”，再仔细查找，发现配置选项如下：

```
|  |  File systems --->
|  |  Partition Types --->
|  |
PC BIOS (MSDOS partition tables) support
```

加上这个后应该就可以挂载 usb 上的 MSDOS 分区了 。

以下是我的内核插入 u 盘后的提示信息：
usb 1-1: new full speed USB device using s3c2410-ohci and 2
ub(1.2): GetMaxLUN returned 0, using 1 LUNs
/ dev/ub/a: p1
表示 usb 设备已经挂载到/dev/ub/a/part1 设备文件下

4） 加入中文字体库（可惜在我的板上还是没能正常显示中文~_~，知道的朋友麻烦告诉我一声，大家一起探讨）

```
|  |  Native Language Support --->
|  |<*> Simplified Chinese charset (CP936, GB2312)
|  |<*> NLS UTF8
```

以下是挂载 usb 设备后的显示：
[root@luofuchong /]# mount -t vfat -o iocharset=cp936 /dev/ub/a/part1 /mnt
[root@luofuchong /]# ls /mnt
cramfs-1.1.tar.gz netkit-base-0.17.tar.gz thttpd-2.25b.tar.gz
lfc settings.dat

卸载 u 盘
umount /mnt

来自文章：s3c2410 全线移植 linux2.6.14.1 u 盘 cs8900a busybox 详细过程
http://www.linuxforum.net/forum/showthreaded.php?Cat=&Board=embedded&Number=648878&page=5&view=collapsed&sb=2&o=all&fpart=&vc=1
内核(2.6.14) + 根文件系统 + Qt4 移植 for S3C2410
http://blog.csdn.net/mimimomo/archive/2008/05/14/2443593.aspx

# 12 移植 MPlayer 到 linux-2.6（声音部分不成功）

整个移植过程，是参考了网上很多文章综合而来的

1） 编译器最好选择 3.3.2 ，我使用 3.4.1 会出现 snow.c 编译错误，找不到解决办法
解压 3.3.2 到/usr/local/arm/下，并 export PATH=/usr/local/arm/3.3.2/bin:$PATH

echo $PATH 看看 3.3.2 是不是在第一个？

2） 播放 mp3 需要禁掉 mp3lib 而使用 madlib ，据说如果使用 mp3lib 会占用 2000%的 CPU，可能嘛？

安装 libmad
（1）交叉编译 libmad
下载 libmad 包（libmad-0.15.1b.tar.gz）；
打开一个终端，进入 libmad 的目录，输入配置命令：
./configure --enable-fpm=arm --host=arm-linux --disable-shared --disable-debugging --prefix=/usr/local/arm/3.3.2/lib
CC=arm-linux-gcc（要保证 arm-linux-gcc 的路径已经有 export 过，否则给出完整路径。）
make
make install
这样就可以看到 /usr/local/arm/3.3.2/lib 目录下多了 include 和 lib 目录，这些就是 libmad 相关的库。
（2）在 configure mplayer 的时候，要加上以下几个选项：
--enable-mad
--with-extraincdir=/usr/local/arm/3.3.2/lib/include（这个指明 mad.h 这个文件所在的路径）
--with-extralibdir=/usr/local/arm/3.3.2/lib/lib（这个指明 libmad 相关链接库所在的路径）
通过以上两个步骤，就可以把 libmad 交叉编译到 mplayer 中。

3） 解压 MPlayer-1.0pre7try2.bz2，改名为 mplayer ，方便而已
在此目录下配置
./configure --cc=arm-linux-gcc --target=arm-armv4-linux --enable-static --disable-win32 --disable-dvdread --enable-fbdev --disable-mencoder --disable-live --disable-mp3lib --enable-mad --enable-libavcodec --with-extraincdir=/usr/local/arm/3.3.2/arm-linux/sys-include/:/usr/local/arm/3.3.2/lib/include --with-extralibdir=/usr/local/arm/3.3.2/arm-linux/lib:/usr/local/arm/3.3.2/lib/lib --prefix=/tmp/mplayer --host-cc=gcc

4）make

排错：

（1）/codec-cfg ./etc/codecs.conf > codecs.conf.h

./codec-cfg: 1: Syntax error: word unexpected (expecting ")")

网上有两种解决方法

A 说在 configure 时添加 --host-cc=gcc 就可以解决

可是会出现其他库找不到的错误

猜想是指定了库--with-extralibdir 导致的，所以还得采用笨笨的 B 方案

B 引用原话

“ 先把 mplayer 编译成 x86 的代码，于是重新配置，简单的 ./configure, make，然后将生成的 codec－cfg 改名为 codec－cfg.x86。

然后再按跨平台方式配置，编译，等编译器出现错误停止编译时，将 codec－cfg.x86 该名为 codec－cfg，再 make，OK,编译就可以继续进行了。”

（2）vobsub.c 错误

这个错误可能在 B 方案时就会出现，

解决方法：修改 vobsub.c 230 行将 getline 函数名称改为 mygetline

5） make 完毕后拷贝目录下的 mplayer 文件到 arm9linux 文件系统运行就好了，我是拷贝在 u 盘里面运行的：

>/mnt # mplayer

>MPlayer 1.0pre7try2-3.3.2 (C) 2000-2005 MPlayer Team

>CPU: ARM

>Usage: mplayer [options] [url|path/]filename

6） 播放 mp3 的问题

A 不带参数运行，如下

/mnt # mplayer 1.mp3

MPlayer 1.0pre7try2-3.3.2 (C) 2000-2005 MPlayer Team

CPU: ARM

Failed to open /dev/rtc: No such file or directory (it should be readable by the user.)

Playing 1.mp3.

Audio file detected.

Clip info:

Title: 高达一年战争秘闻录

Artist: Taja

Album:

Year:

Comment:

Genre: Other

==========
==========
==========
====
Requested audio codec family [mp3] (afm=mp3lib) not available.
Enable it at compilation.
Opening audio decoder: [ffmpeg] FFmpeg/libavcodec audio decoders
AUDIO: 44100 Hz, 2 ch, s16le, 128.0 kbit/9.07% (ratio: 16000->176400)
Selected audio codec: [ffmp3] afm:ffmpeg (FFmpeg MPEG layer-3 audio decoder)
==========
==========
==========
====
Checking audio filter chain for 44100Hz/2ch/s16le -> 44100Hz/2ch/s16le...
AF_pre: 44100Hz/2ch/s16le
[AO OSS] audio_setup: Can't open audio device /dev/dsp: No such file or directory
AO: [null] 44100Hz 2ch s16le (2 bps)
Building audio filter chain for 44100Hz/2ch/s16le -> 44100Hz/2ch/s16le...
Video: no video
Starting playback...
[Mixer] No hardware mixing, inserting volume filter.
A:  0.3 (00.2) 47.4%
A:  0.8 (00.8) 55.9%
A:  1.2 (01.1) 57.2%
No bind found for key l

打不开，因为没有安装 mp3lib 库

B 加参数 mad
/mnt # mplayer -ac mad 1.mp3
MPlayer 1.0pre7try2-3.3.2 (C) 2000-2005 MPlayer Team
CPU: ARM
Failed to open /dev/rtc: No such file or directory (it should be readable by the user.)
Playing 1.mp3.
Audio file detected.
Clip info:
Title:  高达一年战争秘闻录
Artist: Taja
Album:
Year:
Comment:
Genre: Other
==========
==========
==========
====
Forced audio codec: mad
Opening audio decoder: [libmad] libmad mpeg audio decoder
AUDIO: 44100 Hz, 2 ch, s16le, 128.0 kbit/9.07% (ratio: 16000->176400)
Selected audio codec: [mad] afm:libmad (libMAD MPEG layer 1-2-3)

==========
==========
Checking audio filter chain for 44100Hz/2ch/s16le -> 44100Hz/2ch/s16le...
AF_pre: 44100Hz/2ch/s16le
[AO OSS] audio_setup: Can't open audio device /dev/dsp: No such file or directory
AO: [null] 44100Hz 2ch s16le (2 bps)
Building audio filter chain for 44100Hz/2ch/s16le -> 44100Hz/2ch/s16le...
Video: no video
Starting playback...
A:  3.9 (03.8) 46.9%

看似在播放了，但是完全没有声音
仔细看输出，有个错误[AO OSS] audio_setup: Can't open audio device /dev/dsp: No such file or directory

奇怪，我记得 dsp 应该在 sound 下啊
既然它要找/dev/dsp，我做个链接好了
ln -s /dev/sound/dsp /dev/dsp

再运行看看
/mnt # mplayer -ac mad 1.mp3
MPlayer 1.0pre7try2-3.3.2 (C) 2000-2005 MPlayer Team
CPU: ARM
Failed to open /dev/rtc: No such file or directory (it should be readable by the user.)
Playing 1.mp3.
Audio file detected.
Clip info:
Title:  高达一年战争秘闻录
Artist: Taja
Album:
Year:
Comment:
Genre: Other
==========
==========
Forced audio codec: mad
Opening audio decoder: [libmad] libmad mpeg audio decoder
AUDIO: 44100 Hz, 2 ch, s16le, 128.0 kbit/9.07% (rs3c2410-uda1341-superlp: audio_set_dsp_speed:44100 prescaler:66
atio: 16000->176s3c2410-uda1341-superlp: audio_set_dsp_speed:44100 prescaler:66
400)
Selected audio codec: [mad] afm:libmad (libMAD MPEG layer 1-2-3)
==========
==========

Checking audio filter chain for 44100Hz/2ch/s16le -> 44100Hz/2ch/s16le...
AF_pre: 44100Hz/2ch/s16le
AO: [oss] 44100Hz 2ch s16le (2 bps)
Building audio filter chain for 44100Hz/2ch/s16le -> 44100Hz/2ch/s16le...
Video: no video
Starting playback...

MPlayer interrupted by signal 8 in module: play_audio
- MPlayer crashed by bad usage of CPU/FPU/RAM.
Recompile MPlayer with --enable-debug and make a 'gdb' backtrace and
disassembly. Details in DOCS/HTML/en/bugreports_what.html#bugreports_crash.
- MPlayer crashed. This shouldn't happen.
It can be a bug in the MPlayer code _or_ in your drivers _or_ in your
gcc version. If you think it's MPlayer's fault, please read
DOCS/HTML/en/bugreports.html and follow the instructions there. We can't and
won't help unless you provide this information when reporting a possible bug.

还是有错误，但 AO 的错误解决了，我目前仅做到这一步，尚未解决声音问题

6） 播放视频

我尝试了 avi mpg vcd 的 dat 文件，都可以播放，速度有点满，还得禁掉 sound，2410 也就这样吧，放视频还得 DSP。

mplayer 13.mpg -nosound -fs

fs 是中央显示

还有个错误

Failed to open /dev/rtc: No such file or directory (it should be readable by the user.)

RTC 的错误？我找不到这个设备

暂时就这么多吧，声音问题继续研究中。。。。。。。。。。。。。。。。

最后转一段解决声音问题的方法，我按照如上做了，可惜还是没解决：

2．开始调试

编译出来的代码能正常的在 FS2410 上跑吗？心中还是有许多疑问。给板子上电，然后用自己编写的一个网络传输工具将代码传到板子的 ramdisk，接着 再传上一个 10M 左右的视频，好了，先不带参数运行 mplayer，不错，中文的帮助信息弹出来了，说明程序基本编译对了，这时输入命令 ./mplayer matrix.mpg ， 眼睛直盯着屏幕，期待着画面的出现，可惜，在出现了一些视频剪辑的反馈信息后，程序再也不动了，没办法，按 CTRL+C 结束程序，然后就提示出现段错误。 以前听说有 linux 嵌入式的爱好者移植时也出现这样的错误，但是如果不播放声音时，图像可以出现，于是输入命令 ./mplayer －nosound matrix.mpg，这时画面出来了，这个 320x240 大小的从网上下载的视频，播放起来相当流畅，好像比平时看的影碟机解码速度还快，那当然了，毕 竟是 320x240 大小，又没声音解码的。

声音这块不解决，当然是不能说移植成功的，因为 mplayer 还支持那么多格式的音频解码。但是问题究竟出在哪里呢？用排除法吧！找一个未经任何音频压缩 的 WAV 音频文件，其时就是 PCM 文件，上传到 ramdisk，然后用 mplayer 播放，还是出现一样的问题，程序死了。好了，这就说明问题并非出现在 音频解码部分，极有可能出现在音频流的播放部分。现在市面上大多数的嵌入式开发板的音频驱动是 oss 规范的驱动，以前自己做过 oss 的编程，对这块还是比 较熟悉。于是开始查看 mpalyer,c 源文件，看看它是如何实现音频流播放的，

在音频播放部分它使用到了 libao2 库的音频播放/控制模块，通过进一步查看 ao_oss.c 源代码进一步获知 mplayer 是如何与音频的 linux 驱动工作的。这一步弄清楚后，重新编译 mplayer，打开 debug 选项，打开调试字符串输出，并在音频播放处设置多处断点，并加上 printf 语句输 出一些变量内容，最终发现在调用 ao_oss.c 的 play（）函数时出现除零出错，

这个问题产生的根源最终追溯到音频的驱动部分。现在大部分的嵌入式板子都使用菲利普 uda1341 音频芯片，因而也都使用了相同一个音频驱动，即 MIZI 公司拥有版权的 linux uda1341 音频驱动，这个驱动基本上符合了 oss 的规范，但是当使用到多段 DMA 音频数据传输时，出现了一个问题，即 DMA 缓冲的建立发生在第一次调 用 write（）函数将音频数据传送到设备描述符的时候，然而 oss 驱动的调用者通常要在打开音频设备描述时候，就期望获取 DMA 缓冲的信息，然而因为缓 冲尚未建立，因而返回缓冲大小为 0 这个结果。

解决的办法是在音频驱动源码的 smdk2410_audio_open（）函数体，加上如下一段代码，

```
static int smdk2410_audio_open(struct inode *inode, struct file *file)
{
    int cold = !audio_active;
        DPRINTK("audio_open\n");
        if ((file->f_flags & O_ACCMODE) == O_RDONLY) {
            if (audio_rd_refcount || audio_wr_refcount)
                return -EBUSY;
            audio_rd_refcount++;
    } else if ((file->f_flags & O_ACCMODE) == O_WRONLY) {
            if (audio_wr_refcount)
                return -EBUSY;
            audio_wr_refcount++;
    } else if ((file->f_flags & O_ACCMODE) == O_RDWR) {
```

```
                if (audio_rd_refcount || audio_wr_refcount)
                        return -EBUSY;
                audio_rd_refcount++;
                audio_wr_refcount++;
        } else
                return -EINVAL;
            if (cold) {
                audio_rate = AUDIO_RATE_DEFAULT;
                audio_channels = AUDIO_CHANNELS_DEFAULT;
                audio_fragsize = AUDIO_FRAGSIZE_DEFAULT;
                audio_nbfrags = AUDIO_NBFRAGS_DEFAULT;
                if ((file->f_mode & FMODE_WRITE)){
                                init_s3c2410_iis_bus_tx();
                                audio_clear_buf(&output_stream);
                 //  加上以下这行代码
                   if (!output_stream .buffers && audio_setup_buf(&output_stream
))
                        return -ENOMEM;
                }
                if ((file->f_mode & FMODE_READ)){
                                init_s3c2410_iis_bus_rx();
                                audio_clear_buf(&input_stream);
                }
        }
        MOD_INC_USE_COUNT;
       return 0;
  }
```

改完驱动后，重新编译内核。

==========

解决 Failed to open /dev/rtc: No such file or directory (it should be readable by the user.)问题

在 configure 时添加 --disable-rtc，或者给 2410 编译 RTC 模块

# 13 移植 yaffs 文件系统（暂未成功）

网上下载的 yaffs 源码包是基于 Linux2.4 的，yaffs2 源码包才是基于 Linux2.6 的，为了让内核支持 yaffs2，需要修改 makefile 和 Kconfig。参考以下文章：

1） 下载 YAFFS 文件系统代码，下载网址：
http://www.aleph1.co.uk/cgi-bin/viewcvs.cgi/
点击页面左下角的 Download tarball 即可下载全部相关代码。

当然如果你的 NAND FLASH 只是 512+16B 的，可以只移植 yaffs，因为即使你移植了 yaffs2，它也会自动选择挂载 yaffs1 的。

#cd /public

#tar zxf yaffs2.tar.gz

则/public/yaff2/ 目录之下即是 yaffs2 的源码。

2) 在要移植的内核目录下建立 yaffs2 文件夹，并将需要的文件拷贝过来：

#cd /public/linux-2.6.11.7-2410/

# cd fs

# mkdir yaffs2

# cd yaffs2

# cp /public/yaffs2/*.h .

# cp /public/yaffs2/*.c .

# cp /public/yaffs2/Makefile-kernel Makefile

# cp /public/yaffs2/Kconfig .

3) 修改 /public/linux-2.6.11.7-2410/fs/Makefile 和 Kconfig 文件。

# cd /public/linux-2.6.11.7-2410/fs/

# vi Makefile (将下面一行添加到 Makefile 中)

obj-$(CONFIG_YAFFS_FS) += yaffs2/

# vi Kconfig (将下面一行添加到 Kconfig 中)

source "fs/yaffs2/Kconfig

4) cd /public/linux-2.6.11.7-2410/

make menuconfig

在编译内核时选择：

<*> YAFFS2 file system support

<*> 512 byte / page devices

<*> Lets Yaffs do its own ECC

<*> 2048 byte (or larger) / page devices

<*> Autoselect yaffs2 format

<*> Disable lazy loading

<*> Turn off wide tnodes

<*> Turn off debug chunk erase check

5) 编译内核 make zImage

看网上有人编译过程中出现了问题， 我没有碰到任何问题，很顺利。

By the way : 编译 yaffs 时出现了问题，编译通过不了， 这也是我选择 yaff2 而没有选 yaffs 的一大原因。

来自：http://blog.csdn.net/zht_sir/archive/2007/04/07/1555668.aspx

内核虽然支持 yaffs，但是我并没有使用成功，具体原因请看 14。

# 14 都是 nand ecc 惹得祸

起

使用支持 nand 的 u-boot-1.2.0，发现一个奇怪的问题，烧入阳初 2410 自带的 linux2.4 内核和 yaffs 文件，无法启动。

承

昨天在研究 yaffs 系统，费了好大神才下载到 yaffs 和 yaffs2 的源码（阳初 linux2.4 和迅雷上下到的 yaffs，chinaunix 上下到的 yaffs2，官网根本连不上，网络太差），按照教程使用 yaffs 的源码编译内核，总是出错，仔细看下，发现 2 个 yaffs 源码都是 linux2.4 下的，包含的头文件 linux2.6 里面没有。yaffs2 源码倒是基于 linxu2.6 内核的，可以顺利通过编译。
可惜，通过 u-boot 下载后依然无法启动，linux 认不出 yaffs 呢

转

带着郁闷的心情在网上看了若干 yaffs 的帖子，发现了问题所在，竟然是我的 u-boot！
2410 系统是从 nand flash 上启动的，对 nand flash 编程有一个 ECC 计算，目前一共遇到 3 中 ECC 计算方法
A） u-boot 以前版本中采用的自己独有 ECC 计算，若要在 u-boot 中使用，需打开宏 nand_legacy，简称 legacy ECC 吧
B） linux 内核采用的 nand ECC 算法，简称之为 MTD ECC，在新的 u-boot 中，同样支持 MTD ECC，需打开 nand 宏，和 nand_legacy 对立
C） yaffs 文件系统同样采用特有的 ECC 检查，简称 yaffs ECC 吧

我按照网上资料移植的 u-boot-1.2.0 使用的是 nand_legacy 宏，也就是基于 legacy ECC 检查，通过 u-boot 的 nand write 写入的 yaffs 文件系统，是不能被 linux 或者 yaffs 认出来的，悲哀。
同时，因为 legacy ECC，这也是移植 linux2.6 是要关闭 MTD ECC 的原因。开篇错误，也找到了原因。

没合上~~~~~~ToT

解决方法，重新移植 u-boot,添加烧写 yaffs 的专用指令 nand write.yaffs，使得支持 yaffs ECC， 可能得采用 nand 宏而不是 nand_legacy。
看来工程较大，暂未解决，待研究！

# 15 2410 硬件 RTC 在 linux 下的支持

linux2.6 应该对 s3c2410 的 rtc 提供支持，在 make menuconfig 中的 Device Driver 中有 S3C2410 RTC 的选项。但是实际编译，dev 中却没有 rtc 设备，在网上搜索，发现需要修改 mach-smdk2410.c 文件。

```
static struct platform_device *smdk2410_devices[] __initdata = {
&s3c_device_usb,
&s3c_device_lcd,
&s3c_device_wdt,
```

```
&s3c_device_i2c,
&s3c_device_iis,

/*add this*/
&s3c_device_nand,
&s3c_device_rtc,
};
```

再次编译，启动信息中有如下内容：
S3C2410 RTC, (c) 2004 Simtec Electronics
s3c2410-rtc s3c2410-rtc: rtc disabled, re-enabling
同时可以找到： /dev/misc/rtc
使用 date 命令，显示：
Thu Jan 1 00:00:06 UTC 1970
设定方法：date -s 08211643002008
回显：Thu Aug 21 16:43:00 UTC 2008
设定正确，再执行命令： hwclock -w
作用为写入硬件寄存器，发现 date 每次重启时间丢失，hwclock 不丢失。
可以利用 hwclock --hctosys 同步硬件时间和系统时间。

# 第五章：项目实践——远程监控系统

学了这么多，最后用一个基于 USB 摄像头和 BOA 服务器的远程监控小项目来检验一下！

## 1 移植摄像头到 linux2.6

http://mxhaard.free.fr 上的开源驱动提供个各种摄像头在 linux 下的驱动，我找了两个摄像头，一是市场上使用最多的中星微 zc301p，另一个使用 sonix sn9c120 的芯片。其中 zc301p 工作很正常，sn9c120 好像有点问题。移植过程很简单：

1） 在 http://mxhaard.free.fr 下载驱动
最好不要下页面上提供的 tar 文件，那个是编译成模块的，我想编译进内核，不成功。
在这个地址 http://mxhaard.free.fr/spca50x/embedded/KernelPatch/，
下载里面的 usb-2.6.12LE06.patch；

2） 将 patch 拷贝到 drivers/usb 中，执行 patch -p1 < usb-2.6.12LE06.patch
ubuntu 竟然没有 patch 指令，app-get install patch 吧。

3） 打好补丁后会发现 media 目录下生成了 spca5xx 目录，
make menuconfig
摄像头驱动里面会多出：
USB SPCA5XX Sunplus/Vimicro/Sonix jpeg Cameras
选上，再 make zImage 就搞定了。

还有 V4L 也要选：

<*>Multimedia device

<*>Video for Linux

<*>USB support

4） 启动时看到：

usbcore: registered new driver spca5xx

drivers/usb/media/spca5xx/spca_core.c: spca5xx driver 00.57.06LE registered

5） 插上摄像头提示：

/ # usb 1-1: new full speed USB device using s3c2410-ohci and address 2

drivers/usb/media/spca5xx/spca_core.c: USB SPCA5XX camera found. Type Vimicro Zc301P 0x301b

dev 目录下多出 v4l/video0

6） 测试摄像头：

cat /dev/v4l/video0 > /tmp/1.jpg

将 1.jpg 用 tftp 传到 pc 上看

# 2 linux2.6 同时使用 2 个 usb 口

今天想同时插上 u 盘和摄像头，却发现其中一个 usb 口不工作，网上搜索后得到解答，USB 初始化时 MISCCR 寄存器没有设置正确，贴上最后的 usb 初始化函数：

```
int smdk2410_usb_init(void) /* USB */
{
unsigned long upllvalue = (0x78<<12)|(0x02<<4)|(0x03);
printk("USB Control, (c) 2006 s3c2410\n");
s3c_device_usb.dev.platform_data = &usb_s3c2410_info;
        while(upllvalue!=__raw_readl(S3C2410_UPLLCON))
              {
      __raw_writel(upllvalue,S3C2410_UPLLCON);
                          mdelay(1);
                  }
         //////////////add misccr/////////////////
unsigned long misccr;
misccr = __raw_readl(S3C2410_MISCCR);
misccr |= S3C2410_MISCCR_USBHOST;
misccr &= ~(S3C2410_MISCCR_USBSUSPND0 | S3C2410_MISCCR_USBSUSPND1);
```

```
__raw_writel(misccr,S3C2410_MISCCR);
//////////////////////////////////////////////
        return 0;
}
```

如果出现找不到 S3C2410_MISCCR 的错误，请添加#include <asm/arch/regs-gpio.h>

重新编译后，同时插上 u 盘和摄像头：

```
/ # usb 1-2: new full speed USB device using s3c2410-ohci and address 2
ub(1.2): GetMaxLUN returned 1, using 2 LUNs
/dev/ub/a: p1
/ # usb 1-1: new full speed USB device using s3c2410-ohci and address 3
drivers/usb/media/spca5xx/spca_core.c: USB SPCA5XX camera found. Type Vimicro Zc301P 0x301b
```

# 3 servfox+spcaview 远程监控

http://mxhaard.free.fr 的开源项目提供了 servfox+spcaview，可以在 PC 机上实时监控摄像头画面。

1） servfox

下载地址：

http://mxhaard.free.fr/spca50x/embedded/Servfox/

servfox-R1_1_3.tar.gz

解压缩后，将 Makefile.arm 改名为 Makefile，编译，然后将生成的 servfox 拷到板子上，我是拷到 u 盘里。

2） spcaview

spcaview 是基于 linux 的，chinaunix 上有个大侠将它移植到 windows 下面：

http://blog.chinaunix.net/u/27015/showart_223753.html

拿来就可以用了，不过不稳定，总是会出现程序错误和花屏。

3） 为了配合那个编译好的 spcaview，将开发板 ip 改为 192.168.0.12

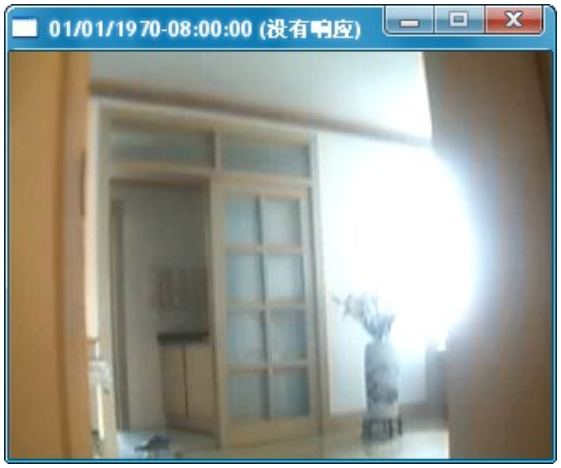
ln －s /dev/v4l/video0 /dev/video0

运行：servfox -d /dev/video0 -s 320x240 -w 7070

打开 pc 软件，成功。

以下是终端输出：

```
servfox version: 1.1.3 date: 11:12:2005 (C) mxhaard@magic.fr
wrong spca5xx device
Reading data from NAND FLASH without ECC is not recommended
Waiting .... for connection. CTrl_c to stop !!!!
Got connection from 192.168.0.1
Got connection from 192.168.0.1
```




4） windows 下的版本貌似还不太稳定，在官网 http://mxhaard.free.fr/下载的 linux 版本则十分稳定，我的两个摄像头都能正常显示。ubuntu 下只要 make 一下就可以使用，此时才发现，sonix 芯片那个收到的图像是反的，奇怪，zc301p 的效果不错。

# 4 搭建 boa 服务器

由于我使用 cramfs 文件系统，只读，设置稍微有点不同。我把网页和错误日志文件都放在了 u 盘的挂载目录 mnt 中，不过这样很方便！安装过程：

1） 在 www.boa.org 下载 boa-0.94.13.tar.gz 并解压；

2） 在 src 目录下运行./configure ；

3） 生成 Makefile 文件，修改：

CC = arm-linux-gcc

CPP = arm-linux-gcc–E

4） make

错误：

A） make：yacc：命令未找到

apt-get install bison

B） make: lex: Command not found

apt-get install flex

C) util.c:100:1: pasting "t" and "->" does not give a valid preprocessing token

修改文件 compat.h

#define TIMEZONE_OFFSET(foo) foo##->tm_gmtoff

修改成：

#define TIMEZONE_OFFSET(foo) (foo)->tm_gmtoff

5） 执行 arm-linux-strip boa

去掉调试信息，小很多，50 多 k，可以编译出 boa 可执行文件，下面是对文件系统的修改：

（1）建立/etc/boa/boa.conf 可以从 boa 源码里拷贝 boa.conf

（2）修改 boa.conf 文件，以下为转载

**********

#监听的端口号，缺省都是 80，一般无需修改

Port 80

# bind 调用的 IP 地址，一般注释掉，表明绑定到 INADDR_ANY，通配于服务器的所有 IP 地址

#Listen 192.68.0.5

#作为哪个用户运行，即它拥有该用户的权限，一般都是 nobody，需要/etc/passwd 中有

#nobody 用户

User nobody

#作为哪个用户组运行，即它拥有该用户组的权限，一般都是 nogroup，需要在/etc/group 文

#件中有 nogroup 组

Group nogroup

#当服务器发生问题时发送报警的 email 地址，目前未用，注释掉

#ServerAdmin root@localhost

#错误日志文件。如果没有以/开始，则表示从服务器的根路径开始。如果不需要错误日志，则用#/dev/null。在下面设置时，注意一定要建立/var/log/boa 目录

ErrorLog /var/log/boa/error_log

#访问日志文件。如果没有以/开始，则表示从服务器的根路径开始。如果不需要错误日志，则用#/dev/null 或直接注释掉。在下面设置时，注意一定要建立/var/log/boa 目录

#AccessLog /var/log/boa/access_log

#是否使用本地时间。如果没有注释掉，则使用本地时间。注释掉则使用 UTC 时间

#UseLocaltime

#是否记录 CGI 运行信息，如果没有注释掉，则记录，注释掉则不记录

#VerboseCGILogs

#服务器名字

ServerName www.hyesco.com

#是否启动虚拟主机功能，即设备可以有多个网络接口，每个接口都可以拥有一个虚拟的 Web 服

#务器。一般注释掉，即不需要启动

#VirtualHost

#非常重要，HTML 文档的主目录。如果没有以/开始，则表示从服务器的根路径开始。

DocumentRoot /var/www

#如果收到一个用户请求的话，在用户主目录后再增加的目录名

UserDir public_html

#HTML 目录索引的文件名，也是没有用户只指明访问目录时返回的文件名

DirectoryIndex index.html

#当 HTML 目录没有索引文件时，用户只指明访问目录时，boa 会调用该程序生成索引文件

然后
#返回给用户，因为该过程比较慢最好不执行，可以注释掉或者给每个 HTML 目录加上
#DirectoryIndex 指明的文件
#DirectoryMaker /usr/lib/boa/boa_indexer
#如果 DirectoryIndex 不存在，并且 DirectoryMaker 被注释，那么就用 Boa 自带的索引
#生成程序来生成目录的索引文件并输出到下面目录，该目录必须是 Boa 能读写
# DirectoryCache /var/spool/boa/dircache
#一个连接所允许的 HTTP 持续作用请求最大数目，注释或设为 0 都将关闭 HTTP 持续作用
KeepAliveMax 1000
#HTTP 持续作用中服务器在两次请求之间等待的时间数，以秒为单位，超时将关闭连接
KeepAliveTimeout 10
#指明 mime.types 文件位置。如果没有以/开始，则表示从服务器的根路径开始。可以注释掉
#避免使用 mime.types 文件，此时需要用 AddType 在本文件里指明
MimeTypes /etc/mime.types
#文件扩展名没有或未知的话，使用的缺省 MIME 类型
DefaultType text/plain
#提供 CGI 程序的 PATH 环境变量值
CGIPath /bin:/usr/bin:/usr/local/bin
#将文件扩展名和 MIME 类型关联起来，和 mime.types 文件作用一样。如果用 mime.types
#文件，则注释掉，如果不使用 mime.types 文件，则必须使用
#AddType application/x-httpd-cgi cgi
#指明文档重定向路径
#Redirect /bar http://elsewhere/feh/bar
#为路径加上别名
Alias /doc /usr/doc
#非常重要，指明 CGI 脚本的虚拟路径对应的实际路径。一般所有的 CGI 脚本都要放在实际
路径
#里，用户访问执行时输入站点+虚拟路径+CGI 脚本名
ScriptAlias /cgi-bin/ /var/www/cgi-bin/

用户可以根据自己需要，对 boa.conf 进行修改，但必须要保证其他的辅助文件和设置必须和 boa.conf 里的配置相符，不然 Boa 就不能正常工作。在上面的例子中，我们还需要创建日志文件所在目录/var/log/boa，创建 HTML 文档的主目录/var/www，将 mime.types 文件拷贝到/etc 目录，创建 CGI 脚本所在目录/var/www/cgi-bin/。mime.types 文件用来指明不同文件扩展名对应的 MIME 类型，一般可以直接从 Linux 主机上拷贝一个，大部分也都是在主机的/etc 目录下。

**********

我做的修改

DocumentRoot /mnt/www　　优盘目录下的 www 目录，需手动建立

ErrorLog /mnt/log/boa/error_log　　优盘目录下的 log 目录，需手动建立

ScriptAlias /cgi-bin/ /mnt/www/cgi-bin/　　优盘目录下的 www/cgi-bin 目录，需手动建立

（3）etc 目录里还要有 passwd group mime.types 等文件

www 目录放 index.html 文件

（4）在板子上运行#boa；

（5）pc 机用 IE 访问 http://192.168.0.12/index.html

ip 是板子的 ip，成功显示：


# 5 搭建网页监控系统

spcaview 软件里带了一个 web 页面访问摄像头的功能，在 http -java-apple 目录里。既然搭建了 boa 服务器，当然要测试下 web 访问。方法简单，复制 http -java-apple 目录到 boa 的 www 目录中。

在开发板上：

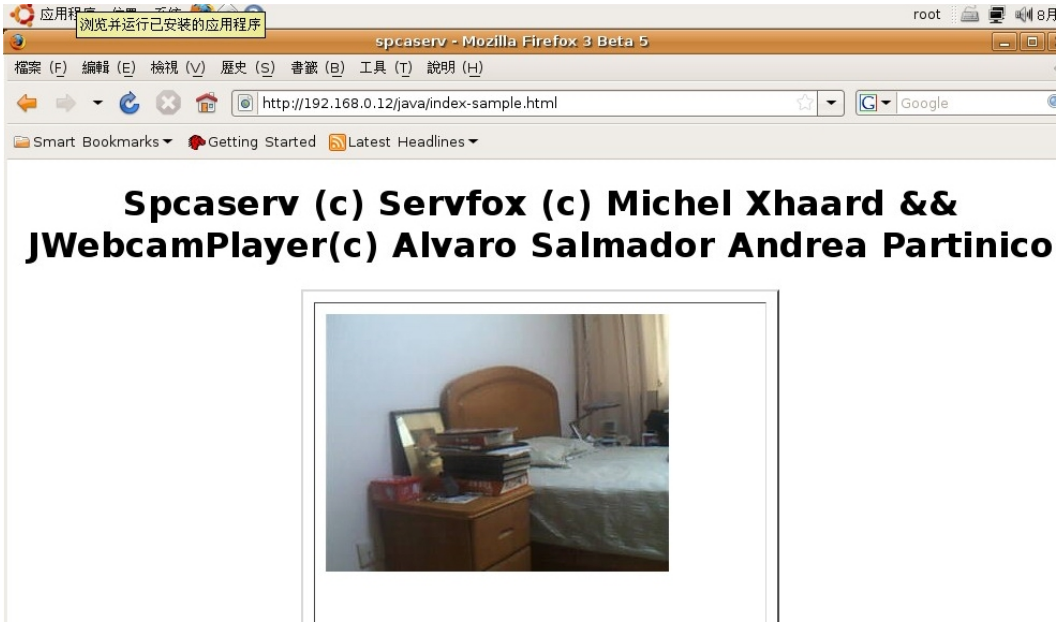
执行 boa

执行 servfox -d /dev/video0 -s 320x240 -w 7070

在浏览器中访问：

http://192.168.0.12/http -java-apple/index-sample.html

ubuntu 中有提示下载 java 插件，装上后就可以看到视频了：




补充： 在 windows xp 中使用 Firefox 浏览器，安装 java 插件，

http://javadl.sun.com/webapps/download/AutoDL?BundleId=23111

也可以正常访问摄像头！


# 后记

终于在奥与会闭幕式之前完成了这篇文章，从 8 月 8 日开始移植 u-boot 到 8 月 24 日完成这篇文章，我也度过了一次自己的奥运会，记得那天把 u-boot 的 I2C 驱动 CH7004 调出来时，就像获得了金牌一样兴奋。Linux 博大精深，这十七天来囫囵吞枣，吃下了许多知识，看来还需要时间好好消化。希望这篇文章对大家能有点帮助，再次声明大多数文章是转载的，我用了一天将自己 blog 的文章整理而成。欢迎热爱嵌入式 linux 的朋友访问我的小屋（http://hi.baidu.com/aokikyon），共同进步！

阿虚

2008 年 8 月 24 日 7 时